การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อจากคลังรายงานจุดบกพร่องของซอฟต์แวร์

วิทยานิพนธ์

ของ

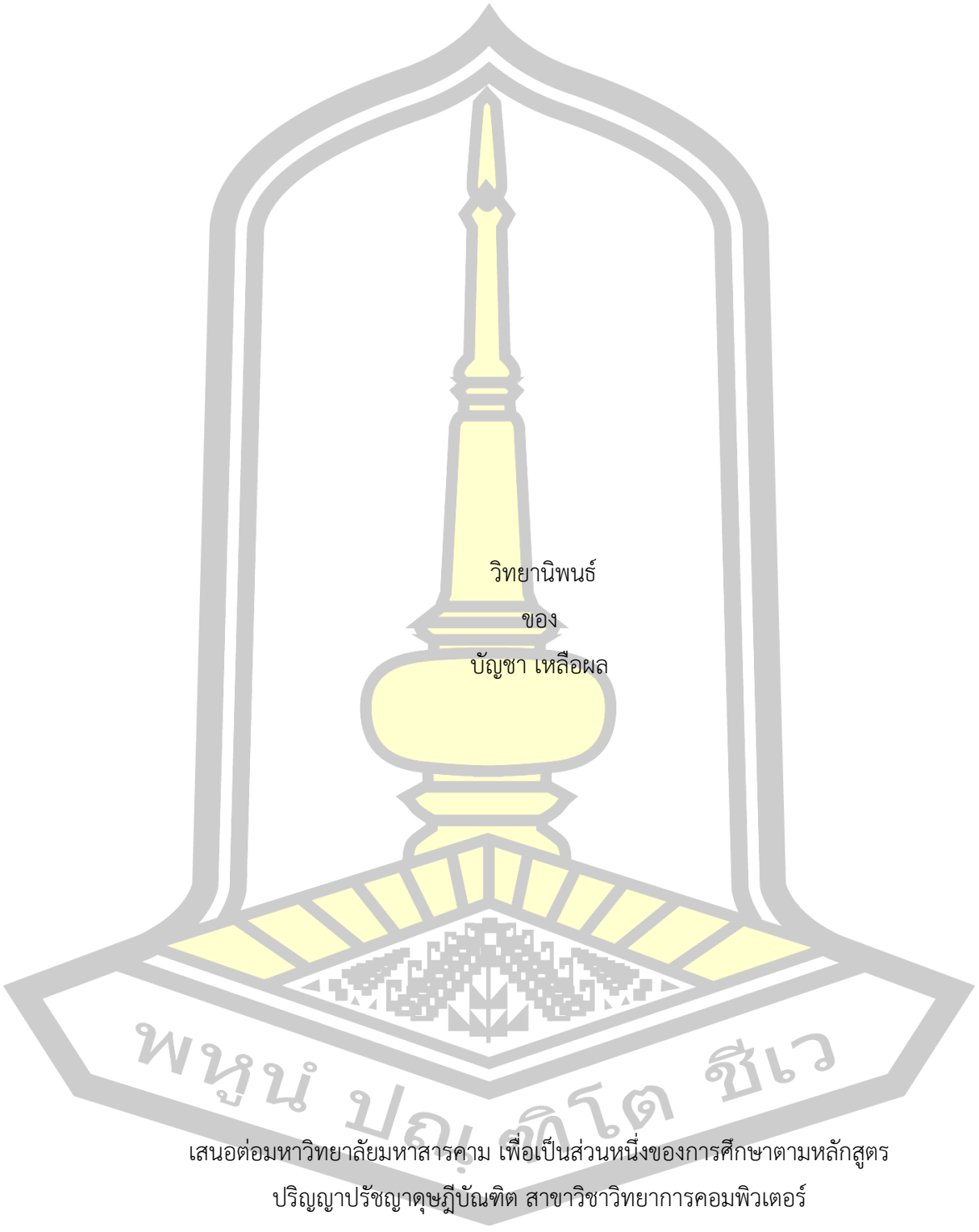
บัญชา เหลือผล

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

เมษายน 2563

การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อจากคลังรายงานจุดบกพร่องของซอฟต์แวร์



วิทยานิพนธ์

ของ

บัญชา เหลือผล

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

เมษายน 2563

Assembling of Dependent Bug Reports from Bug Report Repository

Bancha Luaphol

A Thesis Submitted in Partial Fulfillment of Requirements

for Doctor of Philosophy (Computer Science)

April 2020

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของนายบัญชา เหลือผล แล้ว เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รศ. ดร. ธัชพงศ์ กตัญญูกุล)

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

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(ผศ. ดร. มนัสวี แก่นอำพรพันธ์)

.....กรรมการ
(ผศ. ดร. รพีพร ช่ำชอง)

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

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

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา ปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....	.....
(ผศ. ศศิธร แก้วมั่น)	(รศ. ดร. กริสน์ ชัยมูล)
คณบดีคณะวิทยาการสารสนเทศ	คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อจากคลังรายงานจุดบกพร่องของซอฟต์แวร์
ผู้วิจัย บัญชา เหลือผล
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. จันทิมา พลพินิจ
ผู้ช่วยศาสตราจารย์ ดร. มนัสวี แก่นอำพรพันธ์
ปริญญา ปรัชญาดุษฎีบัณฑิต **สาขาวิชา** วิทยาการคอมพิวเตอร์
มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2563

บทคัดย่อ

ในการพัฒนาซอต์ฟแวร์นั้น จุดบกพร่อง (หรือที่เรียกแบบสั้นว่า บั๊ก) เป็นสิ่งที่หลีกเลี่ยงไม่ได้ จุดบกพร่องจะให้ผลลัพธ์ของซอฟต์แวร์ที่ไม่ถูกต้องหรือไม่คาดคิด ซึ่งจุดบกพร่องจะต้องได้รับการตรวจจับและแก้ไขโดยเร็วที่สุด โดยทั่วไปแล้ว มันไม่ยากนักในการตรวจหาจุดบกพร่องในซอฟต์แวร์ขนาดเล็ก แต่ในทางตรงข้าม การตรวจหาจุดบกพร่องในซอฟต์แวร์โอเพนซอร์ซขนาดใหญ่ในขั้นตอนของการทดสอบซอฟต์แวร์นั้นเป็นไปได้ยาก เนื่องจากจุดบกพร่องหลายจุดอาจจะซ่อนอยู่ในโปรแกรมที่ไม่สามารถตรวจจับได้ทั้งหมดจากการทดสอบ ดังนั้นระบบติดตามจุดบกร่อง (BTS) หลายๆ ระบบ เช่น จิรา เทรช และบั๊กซิลลา ที่ถูกพัฒนาและนำเสนอเพื่อใช้ในการรวบรวมข้อมูลที่สำคัญเกี่ยวกับบั๊กที่เรียกว่า “รายงานจุดบกพร่อง” จากผู้ใช้งานซอฟต์แวร์โอเพนซอร์ซทั่วโลก ซึ่งรายงานจุดบกพร่องเหล่านั้นจะบรรจุข้อมูลที่สำคัญที่สามารถใช้สำหรับการบำรุงรักษาและปรับปรุงคุณภาพของซอฟต์แวร์ อย่างไรก็ตาม หลายๆ งานใน BTS นั้น ยังคงดำเนินการโดยผู้ตรวจสอบรายงานจุดบกพร่อง ผู้ซึ่งเป็นผู้เชี่ยวชาญด้านซอฟต์แวร์ แต่ด้วยจำนวนรายงานจุดบกพร่องที่เพิ่มขึ้นเรื่อย ๆ งานต่างๆ ที่วิเคราะห์ด้วยผู้ตรวจสอบรายงานจุดบกพร่องนั้น กลายเป็นงานที่ต้องใช้เวลาเป็นอย่างมาก ดังนั้น หลายๆ งานวิจัยที่เกี่ยวข้องกับรายงานจุดบกพร่องจึงได้รับการศึกษาและนำเสนออย่างต่อเนื่อง งานวิจัยเหล่านั้นคือ การจำแนกรายงานจุดบกพร่องว่าเป็นรายงานจุดบกพร่องและที่ไม่ใช่รายงานจุดบกพร่อง การวิเคราะห์รายงานจุดบกพร่องที่ซ้ำซ้อน การทำนายระดับความรุนแรงของรายงานจุดบกพร่อง การค้นหาตำแหน่งของจุดบกพร่องในโปรแกรม และการพยากรณ์เวลาในการแก้ปัญหาจุดบกพร่อง เป็นต้น แต่น่าเสียดายที่ปัญหาหนึ่งเกี่ยวกับรายงานจุดบกพร่องที่เคยมีการกล่าวถึงในการศึกษาก่อนหน้า แต่กลับยังไม่เคยมีการศึกษาอย่างจริงจัง ปัญหาดังกล่าวนั้นเรียกว่า “รายงานจุดบกพร่องที่เป็นส่วนต่อ” นั่นคือ หากจุดบกพร่องหนึ่งยังคงอยู่แม้ว่าจะได้รับการแก้ไขจากนักพัฒนาซอฟต์แวร์แล้ว อาจเป็นเพราะว่ามีจุดบกพร่องอื่นที่เกี่ยวข้องกันยังไม่ได้รับการแก้ไขให้สมบูรณ์ จึงส่งผลกระทบกับจุดบกพร่องนั้นๆ แนวทางหนึ่งที่เป็นไปได้ในการแก้ไขปัญหาจุดบกพร่องที่

เป็นส่วนต่อ คือการรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อเข้าด้วยกัน ซึ่งการทำเช่นนี้น่าจะเป็นการให้โอกาสในการแก้ไขจุดบกพร่องของซอฟต์แวร์ได้อย่างสมบูรณ์ เพราะด้วยการรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อเข้าด้วยกัน ทีมพัฒนาซอฟต์แวร์สามารถมองเห็นภาพรวมของจุดบกพร่องทั้งหมดในซอฟต์แวร์ได้ ในงานวิจัยนี้ คุณลักษณะของรายงานจุดบกพร่อง 7 อย่าง ได้แก่ คำเดี่ยว กลุ่มคำแบบสองคำ คำจากกลุ่มคำผสม คำเดี่ยวร่วมกับกลุ่มคำแบบสองคำ คำเดี่ยวร่วมกับคำจากกลุ่มคำผสม กลุ่มคำแบบสองคำร่วมกับคำจากกลุ่มคำผสม และคำเดี่ยวร่วมกับกลุ่มคำแบบสองคำและคำจากกลุ่มคำผสม ได้ถูกศึกษาเชิงเปรียบเทียบ ภายหลังจากทำการทดสอบผ่านการจำแนกรายงานข้อผิดพลาด การวิเคราะห์ความรุนแรงของรายงานข้อบกพร่องและการจัดกลุ่มรายงานข้อบกพร่อง ซึ่งคำเดี่ยวร่วมกับคำจากกลุ่มคำผสม ได้รับการพิจารณาว่าเป็นคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมสำหรับการศึกษา นอกจากนั้น เทคนิคในการให้น้ำหนักคำ 4 เทคนิคได้แก่ tf, tf-idf, BM25, และ Multi-Aspect tf (MATF) ได้ถูกประยุกต์ใช้ โดยกระบวนการพื้นฐานของ 6 เทคนิคได้ถูกนำเสนอเพื่อการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อกัน เทคนิคเหล่านั้นได้แก่ การจัดกลุ่มแบบเคมีนที่มีข้อจำกัด การจัดกลุ่มแบบเคมีนทรงกลมที่มีข้อจำกัด การวิเคราะห์ความคล้ายคลึงแบบโคไซน์, BM25, MATF, และการใช้ฟังก์ชันการค้นคืน (REP) สำหรับ BM25 และ MATF ไม่เพียงถูกใช้เป็นเทคนิคในการให้น้ำหนักแล้ว ยังใช้เป็นฟังก์ชันในการวิเคราะห์ความคล้ายคลึงด้วย ในการศึกษานี้จะใช้รายงานจุดบกพร่องจากไฟร์ฟอกซ์และคอร์ซึ่งเป็นผลิตภัณฑ์จากมอซิลลา โดยเป็นรายงานจุดบกพร่องที่รวบรวมจาก BTS ของมอซิลลาที่มีการรายงานจุดบกพร่องขึ้นมาในระหว่างเดือนตุลาคม ปี 2000 ถึงเดือนกันยายน ปี 2017 หลังจากทดสอบด้วยค่าความระลึก (อัตราผลบวกจริง) ค่าความแม่นยำ ค่าเอฟ ค่าความถูกต้อง กราฟเส้นโค้ง และค่าพื้นที่ใต้เส้นโค้ง พบว่าฟังก์ชันในการวิเคราะห์ความคล้ายคลึงแบบ BM25 ให้ผลลัพธ์ที่น่าพอใจมากที่สุด อย่างไรก็ตาม ถ้าใช้ชุดข้อมูลขาดใหญ่ในการปรับค่าพารามิเตอร์ที่ใช้ในฟังก์ชันการค้นคืน ผลลัพธ์ของ REP มีแนวโน้มที่ให้ผลลัพธ์ที่ดีกว่าผลลัพธ์จาก BM25 แต่เวลาที่ใช้ในการประมวลผลของ REP จะสูงมาก นอกจากนี้ยังพบว่าผลลัพธ์ของแต่ละเทคนิคที่ใช้ในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อกันนั้นให้ผลลัพธ์ที่ดีกว่า เมื่อมีการเปรียบเทียบกับงานวิจัยอื่นๆ อย่างไรก็ตาม งานวิจัยที่นำมาเปรียบเทียบนั้นให้ความสนใจในปัญหาที่แตกต่าง

คำสำคัญ : รายงานจุดบกพร่อง, จุดบกพร่องส่วนต่อ, ฟังก์ชันการค้นคืน, การประมวลผลภาษาธรรมชาติ, การวิเคราะห์ความคล้ายคลึง, ระบบติดตามรายงานจุดบกพร่อง

TITLE Assembling of Dependent Bug Reports from Bug Report Repository

AUTHOR Bancha Luaphol

ADVISORS Assistant Professor Jantima Polpinij , Ph.D.

Assistant Professor Manasawee Kaenampornpan , Ph.D.

DEGREE Doctor of Philosophy **MAJOR** Computer Science

UNIVERSITY Mahasarakham University **YEAR** 2020

ABSTRACT

When developing software, defects (also known as bugs) are inevitable. Bugs return incorrect or unexpected software results; they must be detected and fixed as quickly as possible. In general, it is easier to locate and track bugs in smaller software. By contrast, bugs are rarely encountered during testing on larger open-source software since issues have many places to 'hide' in the coding. Therefore, various bug tracking systems (BTS) such as Jira, Trace, and Bugzilla have been developed and proposed to gather important bug-related information, called as "bug reports", from users worldwide. These bug reports contain significant information that can be used for software quality maintenance and improvement. However, many tasks in BTS are still performed manually by bug triagers who are software experts. With the ever-increasing numbers of bug reports, manual analysis has become a time-consuming task. Therefore, many researches related to bug reports have been studied and proposed studied continuously. Those researches are bug report misclassification, bug reports duplication analysis, bug report severity prediction, bug localization, and bug fixing time prediction. Unfortunately, one problem mentioned in many previous studies has never been fully investigated. This issue is called "bug dependency". That is, if the bug persists after being fixed by a developer, the problem may be caused by another bug that impacts completion of the bug fixing process. A potential solution to resolve the bug dependency problem is to assemble the dependent bug reports. This may provide an opportunity to further completely fix the software bugs. By grouping relevant bug reports together, the development

team can better visualize the overall picture of the bugs in a piece of software. In this study, seven features of bug report as unigram, bigram, compound word, unigram+bigram, unigram+compound word, bigram+compound word, and all features together can be comparatively studied. After experimenting through bug report misclassification, bug report severity analysis and bug report grouping, the unigram+compound word was selected as the proper feature of bug report for this study. In addition, four term-weighting schemes as tf, tf-idf, BM25, and multi-aspect tf (MATF) were also applied. A method based on six techniques was proposed to assemble the dependent bug reports. Those techniques included constraint-based k-means clustering, constraint-based spherical k-means clustering, cosine similarity, BM25, MATF, and the retrieval function (REP). For BM25 and MATF, they are not only applied as the term weighting scheme but also as the similarity function. Bug reports from Firefox and Core as products from Mozilla open-source were used for this study. Bugs identified and reported between October 2000 and September 2017 were gathered through Mozilla's BTS. After testing by recall (true positive rate), precision, F1, accuracy, ROC, and AUC, the BM25 similarity function returned the most satisfactory results. However, if a larger dataset was used for tuning the REP parameters, results of REF tended to be better than those returned by BM25. Unfortunately, processing the REP may require high computational time. Furthermore, it can be found that the results of each technique used for assembling the dependent bug reports may be better when compared to previous studies. However, these studies were concentrated on different issues.

Keyword : Bug report, Bug dependency, Retrieval function, Natural language processing, Similarity analysis, Bug tracking system

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจสำเร็จสมบูรณ์ขึ้นมาได้ หากปราศจากความเมตตา กรุณาจาก ท่านอาจารย์ ผู้ช่วยศาสตราจารย์ ดร. จันทิมา พลพินิจ และ ผู้ช่วยศาสตราจารย์ ดร. มนัสวี แก่นอำพรพันธ์ ที่ได้กรุณารับเป็นอาจารย์ที่ปรึกษาและอาจารย์ที่ปรึกษาร่วมของผู้เขียน ซึ่งได้ให้ข้อมูลและคำแนะนำต่าง ๆ ที่เป็นประโยชน์ต่อผู้เขียน โดยเฉพาะการวางเค้าโครง แนวทางการเขียนเนื้อหาและบทวิเคราะห์ ตลอดจนการกำหนดกรอบเวลาในการเสนอความคืบหน้าของงาน ซึ่งถือเป็นแรงกระตุ้นให้แก่ผู้เขียนได้อย่างดียิ่ง ทั้งท่านอาจารย์ยังได้สละเวลาอันมีค่าตรวจสอบความถูกต้องของงานอีกด้วย ผู้เขียนรู้สึกซาบซึ้งใจและสำนึกในพระคุณของท่านอาจารย์เป็นอย่างยิ่ง จึงขอกราบขอบพระคุณท่านอาจารย์ไว้ ณ ที่นี้

ผู้เขียนขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. ธัชพงศ์ กตัญญูกุล ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร. รพีพร ช่ำชอง ผู้ช่วยศาสตราจารย์ ดร. ฉัตรเกล้า เจริญผล และผู้ช่วยศาสตราจารย์ ดร. พัฒนพงษ์ ชมภูวิเศษ กรรมการสอบวิทยานิพนธ์ ที่ท่านได้กรุณาชี้แนะแนวทางและคำแนะนำ ตลอดจนข้อสังเกตต่าง ๆ ทำให้ผู้เขียนได้พัฒนาแนวความคิดและไตร่ตรองปัญหาต่าง ๆ ได้อย่างรอบคอบมากยิ่งขึ้นจนทำให้วิทยานิพนธ์ฉบับนี้ สำเร็จลงได้

ผู้เขียนขอกราบขอบพระคุณคณาจารย์ต่าง ๆ อันผู้เขียนมิได้เอ่ยนาม ที่ได้อบรมสั่ง สอนให้ความรู้ทางด้านวิชาการแก่ผู้เขียน จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ผู้เขียนขอขอบคุณพี่ ๆ เพื่อน ๆ น้อง ๆ สาขาวิทยการคอมพิวเตอร์ ทุกท่านที่เป็นกำลังใจตลอดมาและคอยช่วยเหลือตลอดระยะเวลาที่ศึกษาและจัดทำวิทยานิพนธ์

สุดท้ายผู้เขียนขอกราบขอบพระคุณ คุณพ่อพันเมือง และคุณแม่แดง เหลือผล ขอขอบใจภรรยาและบุตรชายทั้งสอง ที่มีความรัก ความเข้าใจและเป็นกำลังใจสำคัญให้ ซึ่งทำให้วิทยานิพนธ์ฉบับนี้สำร็จลุล่วงลงได้ โดยหากวิทยานิพนธ์ฉบับนี้มีประโยชน์และคุณค่าทางการศึกษาอยู่บ้าง ผู้เขียนขอยกความดีทั้งหมดแด่ท่านอาจารย์ ผู้ช่วยศาสตราจารย์ ดร. จันทิมา พลพินิจ ผู้ช่วยศาสตราจารย์ ดร. มนัสวี แก่นอำพรพันธ์ และกรรมการสอบวิทยานิพนธ์ทุกท่าน รวมทั้งกราบเป็น กตเวทิตาแก่บิดา มารดา คณาจารย์และผู้มีพระคุณที่ได้อบรมเลี้ยงดู ให้ความรู้ ความเมตตา แก่ผู้เขียน แต่หากวิทยานิพนธ์ฉบับนี้มีความบกพร่องประการใด ผู้เขียนขอน้อมรับความผิดพลาด ไว้แต่เพียงผู้เดียว

บัญชา เหลือผล

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1

บทนำ

1.1 หลักการและเหตุผล

จุดบกพร่อง (Bug หรือ Software defects) [1-3] หรือที่เรียกสั้นๆ ว่า “บั๊ก” เป็นปัญหาที่สามารถเกิดขึ้นในซอฟต์แวร์ อันเนื่องมาจากคำสั่งที่ใช้ในซอฟต์แวร์นั้นๆ ทำงานไม่ถูกต้องตามวัตถุประสงค์ของผู้พัฒนา หรือมีความผิดพลาด จนทำให้การใช้งานซอฟต์แวร์นั้นๆ ไม่ราบรื่นเท่าที่ควร ดังนั้นจำนวนจุดบกพร่องที่พบในซอฟต์แวร์จึงเป็นสิ่งสำคัญอย่างหนึ่งในการประเมินคุณภาพของซอฟต์แวร์ [4]

หากซอฟต์แวร์มีขนาดเล็ก การตรวจหาจุดบกพร่องในซอฟต์แวร์มักถูกดำเนินการโดยนักพัฒนา (Developer) หรือนักทดสอบซอฟต์แวร์ (Software tester) อย่างไรก็ตาม หากเป็นซอฟต์แวร์ขนาดใหญ่การตรวจหาจุดบกพร่องในซอฟต์แวร์จะกลายเป็นเรื่องยากและต้องใช้ความพยายามเป็นอย่างมาก [3, 5] ทำให้ขณะนี้ในการผลิตซอฟต์แวร์ขนาดใหญ่มักจะยังพบข้อผิดพลาดข้อผิดพลาดหลายๆ อย่างจะถูกค้นพบโดยผู้ใช้งาน (Users) โดยตรง [6] จากนั้นผู้ใช้งานจะรายงานจุดบกพร่องที่พบเหล่านั้นไปยังผู้ผลิต โดยคาดหวังว่าความผิดพลาดเหล่านั้นจะได้รับการแก้ปัญหาโดยเร็ว [1, 7]

จากข้างต้น ทำให้หลายองค์กรที่เป็นผู้ผลิตซอฟต์แวร์รายใหญ่เกิดแนวคิดในการสร้างช่องทางให้ผู้ใช้งานซอฟต์แวร์ของตนได้รายงานจุดบกพร่องที่คนเหล่านั้นพบภายหลังการใช้งานจริง เช่น Bugzilla ซึ่ง Bugzilla เป็นซอฟต์แวร์ในการติดตามจุดบกพร่อง (Bug tracking system: BTS) พัฒนาโดยมูลนิธิมอซิลลา (Mozilla) ที่อยู่ในระบบเว็บ [5, 8] ที่เริ่มเปิดใช้ในปี พ.ศ. 2541 นับตั้งแต่การใช้งานมาจนถึงปัจจุบันมีการสำรวจพบว่ามีหน่วยงานมากกว่า 137 หน่วยงาน ที่มีการนำเอา BTS ของมอซิลลา ไปใช้ในการรวบรวมรายงานจุดบกพร่องของซอฟต์แวร์ที่ตนเองพัฒนาขึ้นมาจากผู้ใช้จริง และอาจมีอีกอย่างน้อย 10 เท่าที่นำระบบนี้ไปใช้ในรูปแบบระบบปิด (Closed system) [8] และมีรายงานจุดบกพร่องใน Bugzilla เป็นจำนวนมากกว่า 3,000,000 รายงาน (ข้อมูลเมื่อวันที่ 27 กันยายน 2560 จากซอฟต์แวร์เสรี ต่อไปนี้ Mozilla, Eclipse, OpenOffice, LibreOffice, Apache, KDE, GNOME และ Kernel) จากเหตุนี้ Bugzilla จึงได้กลายเป็นแหล่งข้อมูลที่สำคัญเกี่ยวกับรายงานจุดบกพร่อง เพื่อใช้ประโยชน์สำหรับประกอบการแก้ไขจุดบกพร่องในซอฟต์แวร์

อย่างไรก็ตามเมื่อข้อมูลรายงานจุดบกพร่องที่มีการรวบรวมผ่าน BTS มีจำนวนมากขึ้นเนื่องจากมีการรายงานจุดบกพร่องของซอฟต์แวร์มาจากผู้คนจากทั่วโลก ทำให้เกิดปัญหามากมายในการบริหารจัดการรายงานจุดบกพร่อง [1, 3, 9] เช่น ปัญหาการรายงานจุดบกพร่องที่ซ้ำซ้อน [3, 10-13] ปัญหาการแก้ไขจุดบกพร่องล่าช้าหรือไม่ได้รับการแก้ไข [1, 3, 10, 11, 14] ปัญหาการกำหนดระดับความสำคัญและระดับความรุนแรงของจุดบกพร่อง [14-18] เป็นต้น

จากปัญหาข้างต้นทำให้รายงานจุดบกพร่องกลายเป็นแหล่งข้อมูลที่กำลังได้รับการศึกษาในเชิงวิจัยมากมาย จุดประสงค์เพื่อให้เกิดการใช้รายงานจุดบกพร่องได้อย่างมีประสิทธิภาพ เหมาะสม

และทำให้ซอฟต์แวร์ที่พบปัญหาได้รับการแก้ไขอย่างรวดเร็ว [19-22] จากการศึกษาพบว่า โดยทั่วไปแล้วปัญหาเกี่ยวกับการศึกษารายงานจุดบกพร่องที่ได้รับความนิยม ได้แก่ การพยากรณ์เวลาในการปรับปรุงซอฟต์แวร์ (Fix-time prediction) [1, 7] การมอบหมายรายงานจุดบกพร่อง (Bug report assignment) [23, 24] การทำนายระดับความสำคัญหรือความรุนแรงของจุดบกพร่อง (Severity หรือ Priority prediction) [14-18] การเพิ่มประสิทธิภาพการรายงานจุดบกพร่อง (Bug report optimization) [20, 21, 25] และการตรวจจับจุดบกพร่องซ้ำซ้อน (Duplicated bug detection) [3, 11] เป็นต้น

อย่างไรก็ตาม แม้ปัญหาข้างต้นที่ได้กล่าวมาแล้ว จะเป็นปัญหาเกี่ยวกับรายงานจุดบกพร่องที่ได้รับความนิยมในการศึกษาอย่างต่อเนื่องจนถึงปัจจุบัน อย่างไรก็ตาม ในปี ค.ศ. 2011 นักวิจัยด้านรายงานจุดบกพร่องที่มีชื่อเสียงคือ Bhattacharya และ Neamtiu [1] ได้นำเสนอประเด็นที่น่าสนใจเกี่ยวกับจุดบกพร่องเป็นส่วนต่อหรือที่มีความสัมพันธ์กัน นั่นคือ จุดบกพร่องหนึ่งๆ สามารถที่จะส่งผลต่อจุดบกพร่องอื่นๆ ได้ ซึ่งแนวคิดนี้ไปสอดคล้องกับ Sandusky และคณะ [26] และเรียกปัญหานี้ว่า “จุดบกพร่องส่วนต่อ (Bug dependency)” โดยสามารถอธิบายได้คือ สมมติจุดบกพร่อง B_1 มีส่วนต่อคือจุดบกพร่อง B_2 และ B_3 ดังนั้นหากจะแก้ปัญหาในจุดบกพร่อง B_1 ได้สำเร็จ จุดบกพร่อง B_2 และ B_3 ควรได้รับการแก้ไขก่อน ไม่เช่นนั้น จุดบกพร่อง B_1 จะไม่สามารถแก้ไขให้เสร็จสมบูรณ์ได้

ซึ่งปัจจุบันกระบวนการในการตรวจสอบส่วนต่อของจุดบกพร่องยังคงใช้นักพัฒนาซอฟต์แวร์เป็นผู้ตรวจสอบ ซึ่งทำให้เกิดปัญหาเรื่องค่าใช้จ่าย (Cost) ที่อาจจะเพิ่มมากขึ้นทั้งในเรื่องทรัพยากรบุคคลและเวลา [1, 3, 10, 11, 14] ในขั้นตอนการการปรับปรุงระบบ (Software maintenance) ที่โดยปกติขั้นตอนนี้ก็เป็นขั้นตอนที่มีค่าใช้จ่ายที่สูงที่สุดในวัฏจักรการพัฒนาซอฟต์แวร์ (Software development life cycle: SDLC) อยู่แล้ว [11] และประเด็นปัญหาเรื่อง “จุดบกพร่องส่วนต่อ” แม้จะถูกกล่าวถึงใน [1, 26-28] แต่ปัญหาดังกล่าวก็ยังไม่ได้รับการแก้ไขอย่างจริงจัง

แนวทางหนึ่งที่น่าจะเป็นไปได้ในการแก้ปัญหา “จุดบกพร่องส่วนต่อ” คือการจัดกลุ่มจุดบกพร่องส่วนต่อที่อยู่ภายใต้โดเมนปัญหาเดียวกัน ให้เข้ามาอยู่เป็นกลุ่มเดียวกัน เพราะน่าจะทำให้นักพัฒนาซอฟต์แวร์สามารถมองเห็นภาพรวมของจุดบกพร่องในโดเมนปัญหานั้นๆ ทำให้ช่วยเพิ่มโอกาสในการที่จะแก้ไขจุดบกพร่องในซอฟต์แวร์ได้มีประสิทธิภาพและสมบูรณ์มากยิ่งขึ้น รวมถึงอาจจะช่วยลดเวลาในการแก้ไขจุดบกพร่องได้ดียิ่งขึ้น

1.2 ปัญหาวิจัย

ทำอย่างไรจึงจะจัดกลุ่มรายงานจุดบกพร่องที่เกี่ยวข้องกันหรือเป็นส่วนต่อกันให้เข้ามาอยู่ในกลุ่มเดียวกันแบบอัตโนมัติ เพื่อเป็นแนวทางในการแก้ปัญหา “รายงานจุดบกพร่องที่เป็นส่วนต่อ (Bug report dependency)”

1.3 วัตถุประสงค์ของการวิจัย

เพื่อวิจัยและพัฒนากระบวนการในการจัดกลุ่มรายงานจุดบกพร่องที่เกี่ยวข้องกันหรือเป็นส่วนต่อกันให้เข้ามาอยู่ในกลุ่มเดียวกันแบบอัตโนมัติ เพื่อเป็นแนวทางในการแก้ปัญหา “รายงานจุดบกพร่องที่เป็นส่วนต่อ” ด้วยเทคนิคด้านการประมวลผลภาษาธรรมชาติและเหมืองข้อความ

1.4 ความสำคัญของการวิจัย

1. เป็นการประยุกต์เทคนิคด้านการประมวลผลภาษาธรรมชาติและเหมืองข้อความเพื่อวิเคราะห์ถึงจุดบกพร่องที่สัมพันธ์กันจากคลังรายงานจุดบกพร่องแบบอัตโนมัติ เพื่อที่นักพัฒนาซอฟต์แวร์สามารถมองเห็นรายงานจุดบกพร่องที่เป็นปัญหาในโดเมนเดียวกัน อันจะนำไปสู่การลดระยะเวลาของการแก้ไขจุดบกพร่อง

2. การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่อยู่ภายใต้โดเมนปัญหาเดียวกัน ให้เข้ามาอยู่เป็นกลุ่มเดียวกัน น่าจะทำให้นักพัฒนาซอฟต์แวร์สามารถมองเห็นภาพรวมของจุดบกพร่องในโดเมนปัญหานั้นๆ ทำให้ช่วยเพิ่มโอกาสในการที่จะแก้ไขจุดบกพร่องในซอฟต์แวร์ได้มีประสิทธิภาพและสมบูรณ์มากยิ่งขึ้น

3. เพิ่มโอกาสในการลดค่าใช้จ่ายในขั้นตอนของการปรับปรุงระบบ เพราะขั้นตอนดังกล่าวเป็นขั้นตอนนักวิจัยหลายท่านยืนยันว่าเป็นขั้นตอนที่มีค่าใช้จ่ายสูงที่สุดในวัฏจักรการพัฒนาซอฟต์แวร์

1.5 ขอบเขตการวิจัย

1. นำเสนอกระบวนการต้นแบบในการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่สอดคล้องกันเข้าเป็นกลุ่มเดียวกันแบบอัตโนมัติ

2. ชุดข้อมูล

สำหรับในงานวิจัยนี้จะใช้ชุดข้อมูลในการศึกษา 3 ชุดข้อมูลดังนี้

2.1. ชุดข้อมูลรายงานจุดบกพร่องของ Firefox ที่มีการดาวน์โหลดจากระบบติดตามจุดบกพร่อง Mozilla (https://bugzilla.mozilla.org/) เมื่อวันที่ 1 ตุลาคม 2560 มีจำนวนทั้งสิ้น 176,971 รายงาน โดยเป็นรายงานจุดบกพร่องที่เป็น Mata-bug จำนวน 15,396 รายงาน และที่เหลือเป็นรายงานจุดบกพร่องที่เป็น depend on (แสดงสถานะความเป็นรายงานจุดบกพร่องส่วนต่อ) ซึ่งในงานวิจัยนี้จะเลือกใช้รายงานจุดบกพร่องที่เป็น Mata-bug ที่มีรายงานจุดบกพร่องส่วนต่ออย่างน้อย 10 รายงาน

2.2. ชุดข้อมูลรายงานจุดบกพร่องของ Core ที่มีการดาวน์โหลดจากระบบติดตามจุดบกพร่อง Mozilla เมื่อวันที่ 1 ตุลาคม 2560 โดยมีจำนวนรายงานจุดบกพร่องทั้งสิ้น 1,300 รายงาน

2.3. ชุดข้อมูลรายงานจุดบกพร่องมาตรฐานของ Herzig โดยมีจำนวนรายงานจุดบกพร่องทั้งสิ้น 7,401 รายงาน

3. กระบวนการที่นำเสนอจะอยู่บนพื้นฐานการประยุกต์เทคนิคด้านเหมืองข้อความ (Text mining) และการประมวลผลภาษาธรรมชาติ (Natural language processing: NLP)

4. คุณลักษณะที่ใช้ในงานวิจัยนี้จะทำการศึกษาใน 4 รูปแบบคือ

4.1 คำเดี่ยว (Unigram)

4.2 คำเดี่ยวร่วมกับกลุ่มคำแบบสองคำ (Unigram and bigram)

4.3 คำเดี่ยวร่วมกับคำจากกลุ่มคำผสม (Unigram and compound words)

4.4 คำเดี่ยวร่วมกับกลุ่มคำแบบสองคำและคำจากกลุ่มคำผสม (Combination)

5. การให้น้ำหนักของคุณลักษณะที่ใช้ในงานวิจัยจะทำการศึกษาใน 4 เทคนิค คือ

5.1 Term frequency: *tf*

5.2 Term frequency – inverse document frequency: *tf-idf*

5.3 Best Match 25: *BM25*

5.4 Multi Aspect TF: *MATF*

6. การประเมินการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อจะประเมินด้วยค่าความถูกต้อง (Accuracy), ความระลึก (Recall หรือ True positive rate), ค่าความแม่นยำ (Precision), ค่า F1, ค่า True negative rate (TNR) ค่า ROC และค่า AUC

1.6 นิยามศัพท์เฉพาะ

1. จุดบกพร่อง (Bug) คือ จุดบกพร่องที่พบในซอฟต์แวร์

2. รายงานจุดบกพร่อง (Bug report) คือ รายงานจุดบกพร่องที่พบในซอฟต์แวร์ ที่ผู้ใช้งานพบและมีการรายงานผลให้กับทีมพัฒนาได้ทราบผ่านเครือข่ายอินเทอร์เน็ต

3. ระบบติดตามจุดบกพร่อง (Bug tracking system: BTS) คือ ระบบที่อำนวยความสะดวกในการรายงานความผิดพลาดของซอฟต์แวร์จากผู้ใช้ อีกทั้งยังใช้ในการติดตามและจัดการรายงานจุดบกพร่องและกำหนดนักพัฒนาซอฟต์แวร์ที่จะทำการแก้ไขจุดบกพร่อง

4. ผู้ตรวจสอบรายงานจุดบกพร่อง (Bug triager) คือบุคลากรที่ทำหน้าที่ในการคัดแยกรายงานจุดบกพร่องเพื่อดำเนินการกำหนดนักพัฒนาซอฟต์แวร์ที่จะแก้ไขรายงานจุดบกพร่องเหล่านั้น

5. ความสัมพันธ์ของจุดบกพร่อง (Bug dependency) คือ จุดบกพร่องที่เป็นส่วนต่อหรือมีความขึ้นต่อกัน เช่น จุดบกพร่อง B_1 ไม่สามารถแก้ไขได้เนื่องจากมีจุดบกพร่อง B_2 ที่ต้องได้รับการแก้ไขก่อน ฉะนั้นจะกล่าวได้ว่า B_1 ขึ้นอยู่กับ (Depend on) B_2 หรือ B_2 หยุดการทำงาน (Block) ของ B_1

6. Meta-bug คือ รายงานจุดบกพร่องที่ถูกกำหนดให้เป็นตัวแทนของกลุ่มปัญหาในโดเมนเดียวกันซึ่งถูกกำหนดโดยผู้ตรวจสอบรายงานจุดบกพร่อง

7. การสกัดจุดบกพร่องเชิงสัมพันธ์ (Bug dependency extraction) คือ การวิเคราะห์และจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อหรือเกี่ยวข้องกันเข้าด้วยกันแบบอัตโนมัติ โดยรายงานจุดบกพร่องเหล่านั้นจะขึ้นตรงต่อรายงานจุดบกพร่องที่เป็น Meta-bug นั่นคือในงานวิจัยนี้จะใช้รายงานจุดบกพร่องที่เป็น Meta-bug เป็นคำขอ (Query) ในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อเข้าด้วยกัน

8. Summary คือ ข้อมูลในรายงานจุดบกพร่องที่ใช้บอกถึงความผิดพลาดของจุดบกพร่องโดยสังเขป

9. Description คือ ข้อมูลในรายงานจุดบกพร่องที่ใช้อธิบายเพิ่มเติมเกี่ยวกับความผิดพลาดของจุดบกพร่อง

10. Compound words คือ คำที่เกิดจากการนำคำมาประสมกันตั้งแต่สองคำ เช่น “UrlbarInput” ลักษณะนี้จะเรียกว่า “Camel case” หรือหากเป็น browser_social_activation” ลักษณะนี้จะเรียกว่า “Snake case” เป็นต้น

11. Combination คือคุณลักษณะของรายงานจุดบกพร่องที่เป็นการรวมกันของคำเดี่ยว (Unigram) ร่วมกับกลุ่มคำแบบสองคำ (Bigram) และคำจากกลุ่มคำผสม (Compound word)

12. กฎเชิงไวยากรณ์ คือ รูปแบบกลุ่มคำนามแบบสองคำ

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ จะเป็นการทบทวนวรรณกรรมและงานวิจัยที่เกี่ยวข้องกับการวิจัยรายงานจุดบกพร่อง โดยมีรายละเอียดดังต่อไปนี้

2.1 รายงานจุดบกพร่อง

การศึกษาวิจัยเกี่ยวกับรายงานจุดบกพร่องนั้น จะต้องเข้าใจความหมายของคำศัพท์ที่เกี่ยวข้องดังต่อไปนี้

2.1.1 จุดบกพร่องและรายงานจุดบกพร่อง (Bug and bug reports)

จุดบกพร่อง หรือ บั๊ก (Bug) [1-3] เป็นปัญหาที่สามารถเกิดขึ้นกับซอฟต์แวร์ อันเนื่องมาจากคำสั่งที่ใช้ในซอฟต์แวร์นั้น ๆ ทำงานไม่ถูกต้อง ผิดพลาด หรือไม่ราบรื่นเท่าที่ควร โดยทั่วไปจุดบกพร่องสามารถพบได้ทั้งในส่วนของฟังก์ชันการทำงานหลัก (Function requirement) และฟังก์ชันการทำงานที่ไม่ใช่การทำงานหลัก (Non-functional requirement) ของซอฟต์แวร์ ซึ่งฟังก์ชันการทำงานหลัก คือ ฟังก์ชันในโปรแกรมที่จำเป็นต่อการทำงานในซอฟต์แวร์ ส่วนฟังก์ชันการทำงานที่ไม่ใช่การทำงานหลัก คือ ฟังก์ชันในโปรแกรมเสริมการทำงานในซอฟต์แวร์ เช่น การทำงานเร็ว ง่ายต่อการใช้งาน เป็นต้น และกระบวนการค้นหาและแก้ไขจุดบกพร่อง เรียกว่า ดีบั๊ก (Debug)

รายงานจุดบกพร่อง (Bug reports) คือ เอกสารที่อธิบายจุดบกพร่องของซอฟต์แวร์ซึ่งผู้พัฒนาซอฟต์แวร์ (Developer) ผู้ทดสอบ (Tester) หรือผู้ใช้ (User) เป็นผู้รายงานจุดบกพร่องเหล่านั้นมา [6] ซึ่งรายงานจุดบกพร่องนั้นมีชื่อเรียกอื่นๆ อีกมากมาย เช่น รายงานจุดบกพร่อง (Defect report) [15] รายงานความผิดพลาด (Fault reports) รายงานความล้มเหลว (Failure reports) [29] รายงานข้อผิดพลาด (Error reports) รายงานปัญหา (Problem หรือ Trouble reports) เป็นต้น

ในปัจจุบันซอฟต์แวร์ขนาดใหญ่โดยเฉพาะซอฟต์แวร์ที่ถูกใช้งานในเชิงพาณิชย์ เช่น ไมโครซอฟต์ (Microsoft) หรือ กูเกิ้ล (Google) ได้หาช่องทางอื่นในการตรวจหาจุดบกพร่อง เพราะผู้ใช้งานซอฟต์แวร์มีอยู่ทั่วโลก ผู้ใช้งานเหล่านี้เองได้กลายเป็นแหล่งการตรวจหาจุดบกพร่องในซอฟต์แวร์ผ่านเว็บไซต์ที่ได้จัดเตรียมไว้ เช่น Bugzilla [3]

โดยปกติรายงานจุดบกพร่องจะประกอบด้วยหมายเลขเอกสาร (Identified number) ชื่อเรื่อง (Title หรือ Summary) สถานะของการแก้ไข (Status of edition) เช่น รายงานจุดบกพร่องใหม่ (NEW) รายงานจุดบกพร่องยังไม่ระบุสถานะ (UNCONFIRMED) รายงานจุดบกพร่องที่ได้รับการแก้ไขแล้ว (RESOLVED) เป็นต้น รวมทั้งคำอธิบายถึงลักษณะของรายงานจุดบกพร่อง (Description of the report) ยกตัวอย่างเช่น ขั้นตอนในการถอดแบบข้อผิดพลาด (Steps to reproduce) ร่องรอยจุดบกพร่อง (Stack trace) และลักษณะของการเกิดจุดบกพร่อง (Expected behavior) เป็นต้น ความคิดเห็นเพิ่มเติม (Comment) ซึ่งเป็นการอภิปรายเกี่ยวกับแนวทางแก้ไขที่เป็นไปได้ สิ่งที่แนบมา (Attachments) เช่น กรณีทดสอบ (Test case) และรายการรายงานที่ต้องแก้ไขก่อนที่จะมีการแก้ไขรายงานนี้ [30]

2.1.2 คลังเก็บรายงานจุดบกพร่อง (Bug report repository)

คลังเก็บรายงานจุดบกพร่อง คือ แหล่งเก็บข้อมูลรายงานจุดบกพร่อง โดยส่วนใหญ่แล้วรายงานจุดบกพร่องที่ใช้เก็บในคลังเก็บรายงานจุดบกพร่องเหล่านี้มักจะเป็นรายงานจุดบกพร่องของซอฟต์แวร์ประเภทโอเพนซอร์ส (Open-source software) จุดประสงค์ของการสร้างคลังเก็บรายงานจุดบกพร่องก็เพื่อเปิดโอกาสให้ผู้พัฒนาและผู้ใช้งานซอฟต์แวร์เหล่านี้ ได้มีโอกาสเข้ามาพูดคุยกันเกี่ยวกับคุณภาพของซอฟต์แวร์ผ่านการเขียนโพสต์หรือรายงานจุดบกพร่อง รวมทั้งการแนะนำเบื้องต้นในการแก้ปัญหาที่ผู้ใช้กำลังประสบอยู่ [31, 32] โดยทั่วไปแล้ว คลังเก็บรายงานจุดบกพร่องจะเป็นระบบเปิด ที่สามารถเข้ามาใช้บริการได้ทุกคน เนื่องจากรายงานจุดบกพร่องสามารถเขียนและโพสต์โดยผู้ใช้งานคนใดก็ได้ [33] ตัวอย่างคลังเก็บรายงานจุดบกพร่อง ได้แก่ Project and Issue Tracking System: PITS หรือ Google code bug tracker เป็นต้น

2.1.3 การตรวจสอบรายงานจุดบกพร่อง (Bug report triage)

การตรวจสอบรายงานจุดบกพร่อง [32, 34] คือ การคัดแยกรายงานจุดบกพร่องที่ถูกรายงานเข้ามา โดยผู้ที่ทำหน้าที่นี้เรียกว่า “ผู้ตรวจสอบรายงานจุดบกพร่อง (Bug triager)” ซึ่งการตรวจสอบรายงานจุดบกพร่อง มีขั้นตอนดังนี้

ขั้นแรกผู้ตรวจสอบรายงานจุดบกพร่อง จะพิจารณาว่า “จุดบกพร่อง” นั้นเป็นความบกพร่องจริงหรือไม่ รวมทั้งการตรวจสอบด้วยว่าจุดบกพร่องที่รายงานเข้ามาเป็นจุดบกพร่องที่ซ้ำกันหรือไม่ (Duplicated bug report)

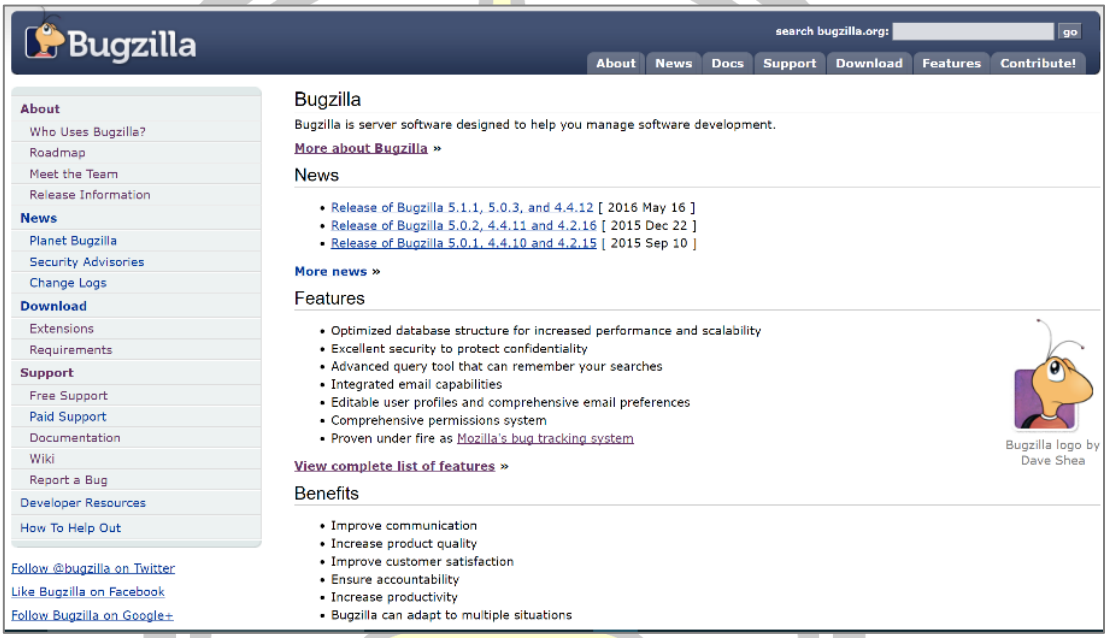
จากนั้นผู้ตรวจสอบรายงานจุดบกพร่อง จะจัดลำดับความสำคัญของรายงานจุดบกพร่อง (Prioritizing bug report) และทำการตัดสินใจกำหนดนักพัฒนาซอฟต์แวร์ที่จะแก้ไขจุดบกพร่องนั้นๆ ต่อไป

2.1.4 ระบบติดตามจุดบกพร่อง (Bug tacking system: BTS)

ระบบติดตามจุดบกพร่อง คือ ระบบที่ใช้ในการติดตามและจัดการรายงานจุดบกพร่องและกำหนดนักพัฒนาซอฟต์แวร์ที่จะทำการแก้ไขจุดบกพร่อง [34] ระบบติดตามจุดบกพร่องได้รับการออกแบบมาเพื่อติดตามจุดบกพร่องของซอฟต์แวร์ที่ถูกรายงานเข้ามาสู่ระบบ จากนั้นระบบติดตามจุดบกพร่อง จะทำการจัดเก็บรายงานจุดบกพร่องในคลังเก็บรายงานจุดบกพร่องที่ตระเตรียมไว้ โดยภายหลังจากผู้ตรวจสอบรายงานจุดบกพร่อง ได้วิเคราะห์ว่ารายงานจุดบกพร่องนั้นเป็นรายงานที่ถูกต้อง และมีการมอบหมายการแก้ไขไปยังผู้รับผิดชอบ ระบบติดตามจุดบกพร่องก็จะจัดส่งการมอบหมายงานไปยังผู้ที่ได้รับมอบแบบอัตโนมัติ ซึ่งก็คือนักพัฒนาซอฟต์แวร์ของโครงการโอเพนซอร์สนั่นเอง เช่น Mozilla Eclipse และ Linux kernel ใช้ระบบติดตามจุดบกพร่อง เช่น Bugzilla Jira Mantis และ Trac เป็นต้น

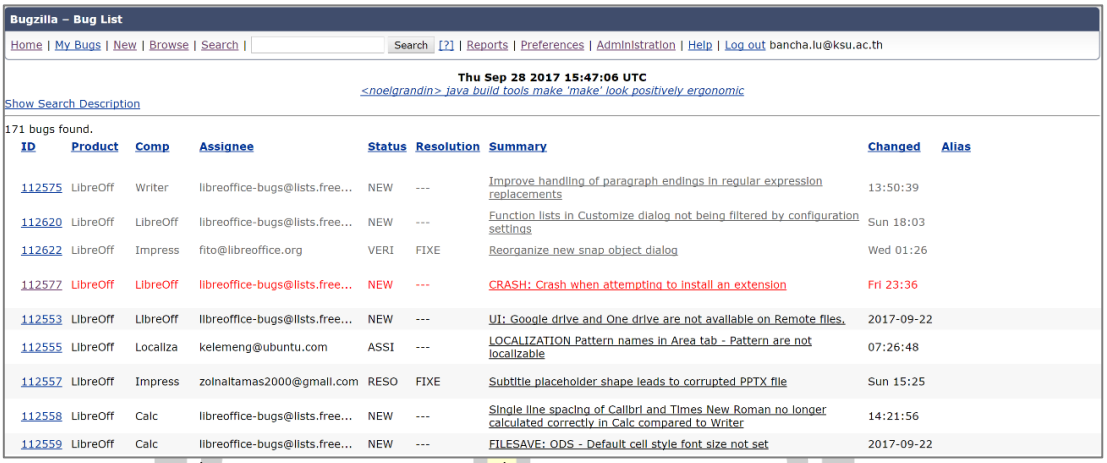
Bugzilla เป็นซอฟต์แวร์ในการติดตามจุดบกพร่อง ที่ถูกพัฒนาโดยมูลนิธิมอซิลลา (Mozilla) ทำงานผ่านระบบเว็บ ซึ่งใช้ภาษาเพิร์ล (Perl) ในการพัฒนา [5, 8] เป็นซอฟต์แวร์โอเพนซอร์ส เริ่มเปิดใช้ในปี พ.ศ. 2541 โดยใช้จัดเก็บข้อมูลการรายงานจุดบกพร่องโครงการซอฟต์แวร์ของ Mozilla ทั้งหมด อาทิเช่น Firefox, Thunderbird, SeaMonkey เป็นต้น จากนั้นจึงมีบริษัท องค์กร

หรือโครงการเข้าไปใช้ประโยชน์กับ Bugzilla (คือ www.bugzilla.org) จำนวนมาก ตั้งแต่เริ่มการใช้งาน Bugzilla จนถึงปัจจุบัน มีรายการซอฟต์แวร์ที่เปิดให้ผู้คนเข้าไปรายงานจุดบกพร่องแบบสาธารณะ 137 หน่วยงาน และมีอีกอย่างน้อย 10 เท่า หรือมากกว่า 1,370 หน่วยงานที่ใช้งาน Bugzilla ในภาคเอกชนแบบไม่เปิดเผย [8] ในรูปที่ 2.1 แสดงหน้าเว็บไซต์ของ Bugzilla ซึ่งปัจจุบันเป็นเวอร์ชัน 5.1.2



รูปที่ 2.1 เว็บไซต์ของ Bugzilla

ความนิยมในตัว Bugzilla สามารถพิจารณาได้จากการมีซอฟต์แวร์ประเภทโอเพนซอร์สที่เป็นที่รู้จักกันดีมานำไปใช้เป็นจำนวนมาก ทั้งนี้เพื่อจัดเก็บข้อมูลการรายงานจุดบกพร่องของซอฟต์แวร์ตนเอง ตัวอย่างเช่น Apache project, Eclipse, OpenOffice, LibreOffice, Linux Kernel, GNOME และ KDE เป็นต้น อีกทั้งมีบริษัทที่ผลิตระบบปฏิบัติการหลายแห่ง ที่ได้นำเอา Bugzilla ไปใช้ในการเก็บข้อมูลรายงานจุดบกพร่อง เช่น RedHat, Gento, OpenSolaris และ SUSE Linux/Novell เป็นต้น



Bugzilla – Bug List

Home | My Bugs | New | Browse | Search | Search [?] | Reports | Preferences | Administration | Help | Log out bancha.lu@ksu.ac.th

Thu Sep 28 2017 15:47:06 UTC

<noelgrandin> java build tools make 'make' look positively ergonomic

Show Search Description

171 bugs found.

ID	Product	Comp	Assignee	Status	Resolution	Summary	Changed	Alias
112575	LibreOff	Writer	libreoffice-bugs@lists.free...	NEW	---	Improve handling of paragraph endings in regular expression replacements	13:50:39	
112620	LibreOff	LibreOff	libreoffice-bugs@lists.free...	NEW	---	Function lists in Customize dialog not being filtered by configuration settings	Sun 18:03	
112622	LibreOff	Impress	fito@libreoffice.org	VERI	FIXE	Reorganize new snap object dialog	Wed 01:26	
112577	LibreOff	LibreOff	libreoffice-bugs@lists.free...	NEW	---	CRASH: Crash when attempting to install an extension	Fri 23:36	
112553	LibreOff	LIbreOff	libreoffice-bugs@lists.free...	NEW	---	UI: Google drive and One drive are not available on Remote files.	2017-09-22	
112555	LibreOff	Localiza	kelemeng@ubuntu.com	ASSI	---	LOCALIZATION Pattern names in Area tab - Pattern are not localizable	07:26:48	
112557	LibreOff	Impress	zolnaltamas2000@gmail.com	RESO	FIXE	Subtitle placeholder shape leads to corrupted PPTX file	Sun 15:25	
112558	LibreOff	Calc	libreoffice-bugs@lists.free...	NEW	---	Single line spacing of Calibri and Times New Roman no longer calculated correctly in Calc compared to Writer	14:21:56	
112559	LibreOff	Calc	libreoffice-bugs@lists.free...	NEW	---	FILESAVE: ODS - Default cell style font size not set	2017-09-22	

รูปที่ 2.2 รายการจุดบกพร่องที่ถูกรายงานของซอฟต์แวร์ LibreOffice

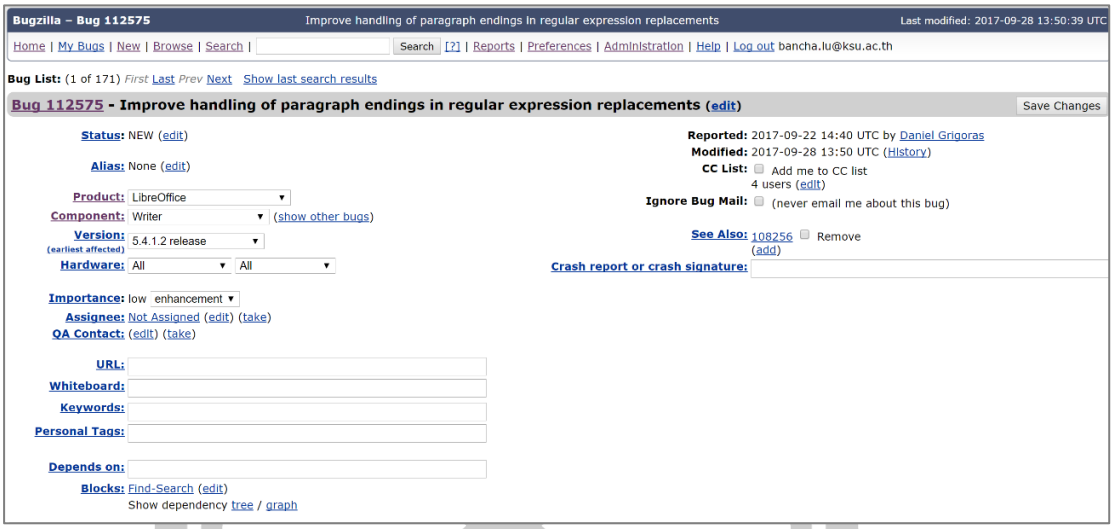
ที่มา : https://bugs.documentfoundation.org/

จากรูปที่ 2.2 ที่แสดงรายการจุดบกพร่องที่ถูกรายงานของซอฟต์แวร์ LibreOffice ซึ่งผู้ใช้งานสามารถดูรายละเอียดของรายงานจุดบกพร่องเบื้องต้นได้ดังที่ปรากฏ และสามารถปรับการแสดงข้อมูลแอตริบิวท์ได้ ซึ่งมีแอตริบิวท์ พอสังเขปที่แสดงจากรูปที่ 2.2 โดยแสดงตามตารางที่ 2.1 ดังนี้

ตารางที่ 2.1 คำอธิบายแอตริบิวท์ของรายงานจุดบกพร่องพอสังเขป

ชื่อแอตริบิวท์	คำอธิบาย
ID	หมายเลขรายงานจุดบกพร่อง
Product	ชื่อผลิตภัณฑ์ที่จุดบกพร่องนั้นปรากฏ
Component	จุดบกพร่องที่ถูกรายงานนั้น มีผลกระทบต่อส่วนใดของซอฟต์แวร์
Assignee	จุดบกพร่องนั้นมีการมอบหมายให้กับนักพัฒนาซอฟต์แวร์คนใดในทีมพัฒนาซอฟต์แวร์รับผิดชอบ
Status	สถานะของจุดบกพร่อง เช่น NEW, ASSIGNED, RESOLVED เป็นต้น
Resolution	รายละเอียดสถานะย่อยของการแก้ไข เช่น FIXED, INVALID, WONTFIX, WORKFORME เป็นต้น
Summary	ประโยคที่ผู้รายงานเขียนสรุปสั้นๆ เกี่ยวกับจุดบกพร่อง

อย่างไรก็ตาม หากต้องการดูรายละเอียดของรายงานจุดบกพร่องเพิ่มเติม ผู้ใช้ระบบติดตามจุดบกพร่อง สามารถคลิกที่หมายเลข ID ของรายงานจุดบกพร่อง จากนั้นก็จะปรากฏรายละเอียดเพิ่มเติมดังแสดงรูปที่ 2.3 - รูปที่ 2.6

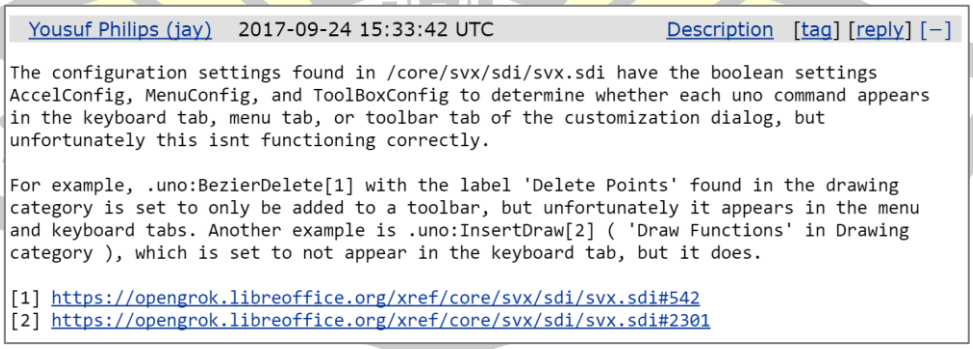


รูปที่ 2.3 รายละเอียดเพิ่มเติมจุดบกพร่องส่วนข้อมูลทั่วไป

ที่มา : https://bugs.documentfoundation.org/

รูปที่ 2.4 รายละเอียดเพิ่มเติมจุดบกพร่องส่วนข้อมูลไฟล์แนบ และการเพิ่มข้อคิดเห็น

ที่มา : https://bugs.documentfoundation.org/



รูปที่ 2.5 รายละเอียดเพิ่มเติมจุดบกพร่องส่วนข้อมูลรายละเอียด

Maxim Monastirsky 2017-09-24 18:03:57 UTC Comment 1 [tag] [reply] [−]

Indeed, it doesn't work that way for many years. Currently the protocol between the core application and the customization dialog is the css::frame::XDispatchInformationProvider UNO interface, which allows filtering only by a command group [1]. The current implementation on the application side is that if one of those *Config booleans is true, the command will be returned via XDispatchInformationProvider [2][3], and therefore on the customization dialog side it will be visible in all tabs.

So what we probably need here is to introduce a new optional interface which (when supported by the currently active application) will allow asking the application for a specific kind of commands, based on the current customization tab.

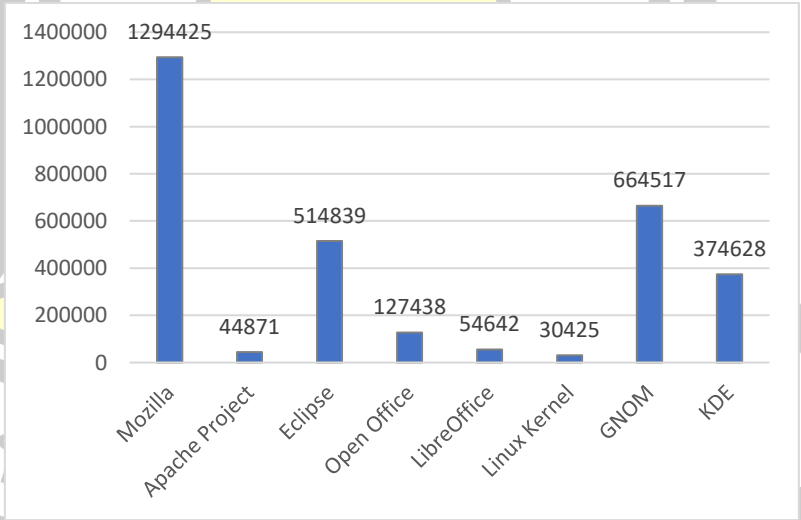
[1]
https://api.libreoffice.org/docs/idl/ref/interfacecom_1_1sun_1_1star_1_1frame_1_1XDispatchInformationProvider.html

[2]
https://opengrok.libreoffice.org/xref/core/sfx2/source/appl/appdispatchprovider.cxx#207
[3] https://opengrok.libreoffice.org/xref/core/sfx2/source/view/sfxbasecontroller.cxx#1143

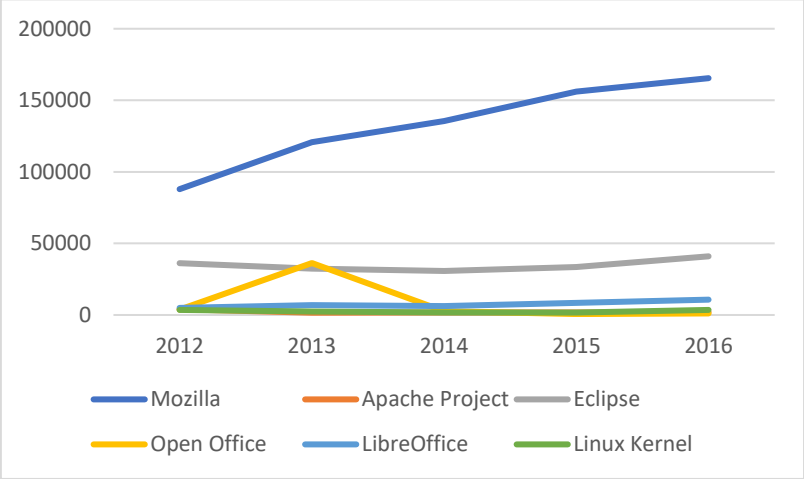
รูปที่ 2.6 รายละเอียดเพิ่มเติมจุดบกพร่องส่วนข้อมูลข้อคิดเห็น
ที่มา : https://bugs.documentfoundation.org/

จากสถิติของการใช้งาน Bugzilla ในการจัดเก็บและรวบรวมรายงานจุดบกพร่องนั้น พบว่ามีผู้มาใช้บริการทั้งสิ้นมากกว่า 1,370 โครงการ โดยในโอเพนซอร์สที่เป็นที่รู้จักกันดีมีอย่างน้อย 8 โครงการได้แก่ Mozilla, Apache project, Eclipse, Open Office, LibreOffice, Linux Kernel, GNOME และ KDE

หากพิจารณาจากรูปที่ 2.7 จะพบว่ามีการรายงานจุดบกพร่องภายใต้ระบบ Bugzilla มากกว่า 3,000,000 รายงาน ถ้านับเฉพาะซอฟต์แวร์โอเพนซอร์ส 8 โครงการที่ได้กล่าวไว้ข้างต้น ในข้อมูลระหว่างปี 2012 ถึง 2016 จะพบว่ามีแนวโน้มของการรายงานจุดบกพร่องเข้ามามากขึ้นดังรูปที่ 2.8



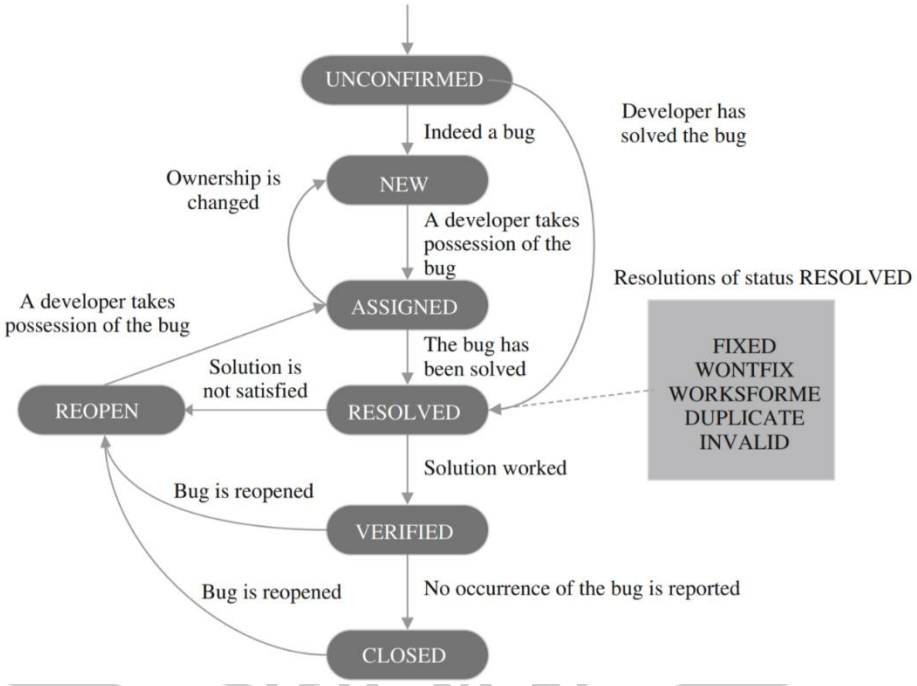
รูปที่ 2.7 จำนวนรายงานจุดบกพร่องของซอฟต์แวร์โอเพนซอร์ส



รูปที่ 2.8 จำนวนรายงานจุดบกพร่องของซอฟต์แวร์โอเพนซอร์ส ระหว่างปี 2012 ถึง 2016

2.2 วงจรชีวิตของรายงานจุดบกพร่อง (Bug report life cycle)

สถานะของรายงานจุดบกพร่องที่มีหลากหลาย จะมีการกำหนดตามช่วงของวงจรชีวิตการใช้งานรายงานจุดบกพร่อง โดยวงจรชีวิตของรายงานจุดบกพร่องแสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 วงจรชีวิตของรายงานจุดบกพร่อง
ที่มา : [35]

จากรูปที่ 2.9 และในหลายงานวิจัย [7, 9, 32, 35] ได้อธิบายวงจรชีวิตของรายงานจุดบกพร่องดังนี้ เริ่มจากเมื่อมีผู้รายงานจุดบกพร่องของซอฟต์แวร์เข้ามาครั้งแรก รายงานจุดบกพร่องนั้นจะถูกกำหนดสถานะเป็น “UNCONFIRMED” จากนั้นผู้ตรวจสอบรายงานจุดบกพร่อง จะทำการ

ตรวจสอบว่ารายงานจุดบกพร่องดังกล่าวมีความซ้ำซ้อนกับรายงานจุดบกพร่องก่อนหน้าหรือไม่ หากไม่ซ้ำซ้อนและเป็นรายงานจุดบกพร่องใหม่จริง รายงานจุดบกพร่องจะถูกระบุสถานะให้เป็น “NEW”

หลังจากนั้นผู้ตรวจสอบรายงานจุดบกพร่องจะกำหนดนักพัฒนาซอฟต์แวร์ที่จะทำการแก้ไขหรือปรับปรุงซอฟต์แวร์ตามรายงานจุดบกพร่อง หากมีการกำหนดนักพัฒนาซอฟต์แวร์ที่เหมาะสมในการแก้ไขหรือปรับปรุงซอฟต์แวร์ได้แล้ว สถานะของรายงานจุดบกพร่องจะกำหนดเป็น “ASSIGNED” และเมื่อรายงานจุดบกพร่องใดได้รับการแก้ไขแล้ว สถานะจะเปลี่ยนไปเป็น “RESOLVED” โดยในสถานะนี้ยังอาจถูกแบ่งออกไปเป็นสถานะย่อยโดยละเอียดดังต่อไปนี้

ตารางที่ 2.2 สถานะโดยละเอียด (Resolution)

Resolution	คำอธิบาย
FIXED	การที่นักพัฒนาซอฟต์แวร์เข้าแก้ไขจุดบกพร่องด้วยการแก้ไขโค้ดโปรแกรมรายงานจุดบกพร่อง
WONTFIX	การที่นักพัฒนาซอฟต์แวร์ไม่สามารถแก้ไขจุดบกพร่องนั้นได้
WORKFORME	การที่นักพัฒนาซอฟต์แวร์ ไม่สามารถถอดแบบหรือ reproduce ได้
DUPLICATE	การที่นักพัฒนาซอฟต์แวร์พบว่าเป็นจุดบกพร่องที่ซ้ำกัน
INVALID	รายงานจุดบกพร่องนั้นไม่ใช่จุดบกพร่องจริง

จากนั้นนักทดสอบซอฟต์แวร์ (Software tester) จะเข้าตรวจสอบการแก้ไขจุดบกพร่อง หากไม่เป็นที่น่าพอใจ สถานะจะเปลี่ยนเป็น “REOPEN” เพื่อย้อนกลับไปสู่การกำหนดนักพัฒนาซอฟต์แวร์ที่จะทำการแก้ไขหรือปรับปรุงซอฟต์แวร์อีกครั้ง แต่ถ้าแก้ไขเรียบร้อยดี ก็จะมีการกำหนดเป็นสถานะ “VERIFIED” และหากไม่มีการรายงานจุดบกพร่องในส่วนนี้กลับเข้ามาอีก ก็จะปิดการแก้ไขในส่วนนั้น และเปลี่ยนสถานะเป็น “CLOSED”

2.3 การทบทวนวรรณกรรมเกี่ยวกับรายงานจุดบกพร่อง

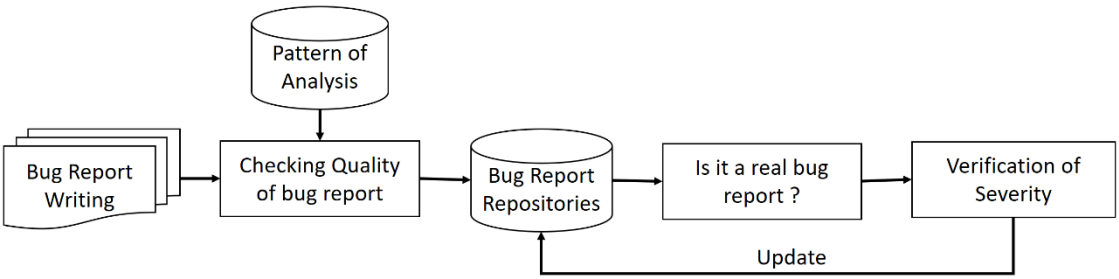
จากการศึกษางานวิจัยที่ผ่านมา พบว่ารายงานจุดบกพร่องเป็นแหล่งข้อมูลเพื่อใช้ในการปรับปรุงคุณภาพการทำงานของซอฟต์แวร์ รวมถึงการพัฒนาซอฟต์แวร์ให้สอดคล้องกับความต้องการในการใช้งานของผู้ใช้งานอย่างแท้จริง งานวิจัยเหล่านั้นสามารถแบ่งเป็น 3 กลุ่ม ได้แก่ การเพิ่มประสิทธิภาพรายงานจุดบกพร่อง (Bug report optimization), การตรวจสอบรายงานจุดบกพร่อง (Bug report triage) และ การแก้ไขจุดบกพร่อง (Bug fixing) โดยในแต่ละกลุ่มจะมีรายละเอียดดังต่อไปนี้

2.3.1 การศึกษาที่เกี่ยวข้องกับการปรับปรุงรายงานจุดบกพร่อง

รายงานจุดบกพร่องมีความสำคัญในการพัฒนาซอฟต์แวร์ และเนื่องจากคุณภาพของรายงานจุดบกพร่องมีผลต่อเวลาในการแก้ไขข้อบกพร่อง ดังนั้นการเพิ่มประสิทธิภาพรายงาน

จุดบกพร่องจึงเป็นสิ่งสำคัญ อย่างไรก็ตามในทางปฏิบัติรายงานจุดบกพร่องจำนวนมากนั้นเป็นรายงานที่ไม่ได้มีคุณภาพ [20, 21, 36]

จากการศึกษาพบว่ากระบวนการโดยทั่วไปของการปรับปรุงคุณภาพการรายงานจุดบกพร่องจะแสดงได้ดังรูปที่ 2.10



รูปที่ 2.10 กระบวนการทั่วไปในการปรับปรุงคุณภาพรายงานจุดบกพร่อง

จากรูปที่ 2.10 จะเห็นว่าเมื่อมีการเขียนรายงานจุดบกพร่องจะมีการตรวจสอบคุณภาพของรายงานจุดบกพร่อง (Checking quality of bug report) โดยอาศัยรูปแบบการรายงานที่ดีที่ได้รับการวิเคราะห์ไว้จากนักพัฒนาซอฟต์แวร์ (Pattern of analysis) แล้วบันทึกเข้าคลังเก็บรายงานจุดบกพร่อง (Bug report repositories) จากนั้นจะทำการตรวจดูว่าใช่รายงานจุดบกพร่องหรือไม่ (Is it a real bug report) แล้วจึงเข้าสู่ขั้นตอนของการกำหนดระดับความรุนแรงของจุดบกพร่อง (Verification of severity) แล้วขั้นสุดท้ายก็จะบันทึกรายงานจุดบกพร่องนั้นเข้าสู่คลังเก็บรายงานจุดบกพร่อง

เพื่อลดรายงานจุดบกพร่องที่ไม่ถูกต้องและปรับปรุงคุณภาพของรายงานจุดบกพร่องให้เหมาะสม นักวิจัยหลายๆ ท่านได้ศึกษาแนวทางด้านการเพิ่มประสิทธิภาพรายงานจุดบกพร่อง (Bug report optimization) โดยการปรับปรุงคุณภาพรายงานจุดบกพร่องจะมีการศึกษาใน 3 กรณี ได้แก่ ศึกษาเนื้อหารายงานจุดบกพร่อง (Content optimization) เทคนิคในการลดการจัดเก็บข้อมูลผิดพลาดของรายงานจุดบกพร่อง (Bug report misclassification หรือ bug or non-bug) และเทคนิคในการทำนายความรุนแรงของรายงานจุดบกพร่อง (Severity prediction)

1) ศึกษาเนื้อหาของรายงานจุดบกพร่อง

ปี ค.ศ. 2004 Sandusky และคณะ [26] ได้ศึกษาเครือข่ายรายงานจุดบกพร่อง: ความหลากหลาย กลยุทธ์ และผลกระทบในชุมชนการพัฒนาซอฟต์แวร์โอเพนซอร์ส งานวิจัยของเขาได้แสดงให้เห็นว่าโครงสร้างที่เด่นชัดของระบบการติดตามจุดบกพร่องคือ เครือข่ายรายงานจุดบกพร่อง (Bug report network: BRN) ซึ่งมี 3 รูปแบบดังต่อไปนี้ (1) ความซ้ำซ้อน (Duplications) คือ การรายงานจุดบกพร่องที่ซ้ำซ้อน (2) การขึ้นต่อกัน (Dependencies) คือ การที่จุดบกพร่องมีการขึ้นต่อกันกับจุดบกพร่องหนึ่ง ซึ่งเป็นสองลักษณะ คือ "block" และ "depend on" และมีการกำหนดให้รายงานจุดบกพร่องเป็น "Meta-bug" เพื่อประโยชน์ในการรวมกลุ่มรายงานจุดบกพร่องที่มีความขึ้นต่อกัน (3) ความสัมพันธ์ไม่เป็นทางการ (Informal relationships) มักจะพบในข้อความคิดเห็นในรายงานจุดบกพร่อง ซึ่งพยายามอธิบายว่าจุดบกพร่องดังกล่าวคล้ายกับ

จุดบกพร่องใดในลักษณะของการอ้างถึง จากกลุ่มตัวอย่างจำนวน 385 ตัวอย่างที่ได้รับการสุ่มตัวอย่างจากรายงานจุดบกพร่อง 182,000 ฉบับ ที่เกิดขึ้นในช่วงห้าปีของซอฟต์แวร์โอเพนซอร์ส พบว่า 65% ของรายงานจุดบกพร่องเหล่านี้จะมีความสัมพันธ์ของรายงานจุดบกพร่องอย่างน้อยหนึ่งรูปแบบ

Bettenburg และคณะ [20] เป็นกลุ่มแรกที่ได้ศึกษาคุณภาพของรายงานจุดบกพร่องในปี ค.ศ. 2007 พวกเขาได้ทำการสำรวจนักพัฒนาซอฟต์แวร์บางราย ของ Eclipse จำนวน 336 คน แต่ได้รับการตอบกลับ 48 คน งานวิจัยของพวกเขาได้ทำเครื่องมือในการวัดคุณภาพรายงานจุดบกพร่อง ผลการวิจัยของพวกเขาแสดงให้เห็นว่าขั้นตอนในการทำซ้ำจุดบกพร่อง (Reproduce) และติดตามร่องรอยของจุดบกพร่อง (Stack trace) เป็นสิ่งที่สำคัญที่สุดสำหรับนักพัฒนาซอฟต์แวร์และความผิดพลาดในขั้นตอนในการทำซ้ำจุดบกพร่อง และข้อมูลที่ไม่สมบูรณ์เป็นอันตรายมากที่สุด

Hooimeijer และ Weimer [9] ในปี ค.ศ. 2007 ยังนำเสนอโมเดลเพื่อวัดคุณภาพรายงานจุดบกพร่อง แต่ได้แบ่งรายงานข้อผิดพลาดออกเป็น "cheap" และ "expensive" ข้อมูลได้ใช้รายงานจุดบกพร่องจำนวน 27,000 รายงาน จากโครงการ Mozila Firefox ด้วยการคาดการณ์ว่ารายงานจุดบกพร่องสามารถแก้ไขได้ภายในเวลาที่กำหนดหรือไม่ การวิจัยของพวกเขาดำเนินการโดยอาศัยสมมติฐานว่า "โดยทั่วไปรายงานที่มีคุณภาพสูงกว่าจะได้รับการจัดการที่รวดเร็วกว่าคุณภาพที่ต่ำกว่า" อย่างไรก็ตามสมมติฐานนี้ไม่สมเหตุสมผลเนื่องจากรายงานจุดบกพร่องจำนวนมากได้รับการแก้ไขอย่างรวดเร็วไม่ใช่เพราะคุณภาพ แต่เนื่องจากเป็นปัญหาเร่งด่วน

จากนั้นในปี ค.ศ. 2008 Bettenburg และคณะ [21] ขยายการสำรวจไปยังสามโครงการโอเพนซอร์ส (Apache, Eclipse และ Mozilla) งานวิจัยนี้ได้รวบรวมข้อมูลเพิ่มเติมจากนักพัฒนาซอฟต์แวร์และผู้รายงานจุดบกพร่องจำนวน 2,226 คน ได้รับการตอบ 466 คน จากนักพัฒนาซอฟต์แวร์ 156 คน และจากผู้รายงาน 310 คน และได้ผลลัพธ์ที่น่าเชื่อถือมากขึ้น สำหรับนักพัฒนาซอฟต์แวร์นั้นข้อมูลที่ใช้กันอย่างแพร่หลายในรายงานจุดบกพร่อง ได้แก่ ขั้นตอนในการทำซ้ำจุดบกพร่อง พฤติกรรมที่คาดหมาย และการติดตามร่องรอยจุดบกพร่อง พวกเขาเสนอว่าข้อมูลที่สำคัญที่สุด ประกอบด้วยขั้นตอนในการทำซ้ำจุดบกพร่อง การติดตามร่องรอยจุดบกพร่อง และกรณีทดสอบ (Test case) พวกเขายังได้สร้างเครื่องมือใหม่ใช้ชื่อว่า CUEZILLA โดยการสร้างแบบจำลองการเรียนรู้แบบมีผู้สอน (Supervise learning) ใช้สำหรับการให้คะแนนคุณภาพของรายงานจุดบกพร่องจากนักพัฒนาซอฟต์แวร์ เครื่องมือนี้ไม่เพียงแต่วัดคุณภาพของรายงานจุดบกพร่องใหม่เท่านั้น แต่ยังแนะนำให้ผู้รายงานทราบว่าจะเพิ่มรายงานจุดบกพร่องได้อย่างไร

Xie และคณะ [25] ดำเนินการศึกษาทดลองเกี่ยวกับรายงานจุดบกพร่องของ Gnome และ Mozilla ในปี ค.ศ. 2014 พวกเขาพบว่ารายงานจุดบกพร่องเหล่านั้นมีปัญหาขาดข้อมูลที่เพียงพอในการทำความเข้าใจปัญหา ถึง 15% และ 6% ของ Gnome และ Mozilla ตามลำดับ เพื่อทำความเข้าใจว่าโครงการใช้ประโยชน์จากอาสาสมัครที่ไม่ใช่นักพัฒนาซอฟต์แวร์ ที่เรียกว่า ผู้ตรวจสอบรายงานจุดบกพร่อง เพื่อปรับปรุงคุณภาพของรายงานจุดบกพร่อง งานวิจัยของพวกเขาพบว่าผลกระทบหลักของผู้ตรวจสอบรายงานจุดบกพร่องเกี่ยวข้องกับการกรองรายงาน การเติมข้อมูลที่ขาดหายไป และการกำหนดผลิตภัณฑ์ที่เกี่ยวข้อง งานวิจัยนี้ชี้ว่า ผู้ตรวจสอบรายงานจุดบกพร่องสามารถคัดกรองรายงานที่ไม่ถูกต้องได้ แต่ยังมีปัญหาในการระบุผลิตภัณฑ์ที่เกี่ยวข้องได้อย่างแม่นยำ

Davies และ Roper [36] ได้ทำการศึกษาข้อมูลที่ถูกบันทึกในรายงานจุดบกพร่องจากซอฟต์แวร์โอเพนซอร์ส 4 โครงการ คือ Eclipse, Firefox, Apache HTTP และ Facebook API จำนวน 1,600 รายงาน ในปี ค.ศ. 2014 งานวิจัยนี้ได้พบว่า รายงานจุดบกพร่องเป็นวิธีหลักที่ผู้ใช้ระบบสามารถสื่อสารปัญหากับนักพัฒนาซอฟต์แวร์ได้ แต่เนื้อหาที่สำคัญจะต้องให้ข้อมูลไว้อย่างมีคุณภาพด้วย อาทิเช่น ขั้นตอนในการถอดแบบข้อผิดพลาด พฤติกรรมที่คาดหวัง (Expected behavior) พฤติกรรมที่สังเกต (Observed behavior) ซึ่งสามส่วนนี้มักจะถูกเขียนอธิบายไว้ในคำอธิบายรายงานจุดบกพร่อง (Description) ร่องรอยความผิดพลาด ภาพหน้าจอ เป็นต้น

2) เทคนิคในการลดการจัดเก็บข้อมูลผิดพลาดของรายงานจุดบกพร่อง

ในปี ค.ศ. 2008 Antoniol และคณะ [19] ได้ยกปัญหาความผิดพลาดจากการแยกแยะ (Misclassification) รายงานจุดบกพร่องว่าเป็น “จุดบกพร่อง” หรือ “ไม่ใช่จุดบกพร่อง” (Bug or non-bug) นอกจากนี้ยังมีการสร้างแบบจำลองการจำแนก (Classifier) 3 ตัวหลักๆ คือตัวจำแนกที่พัฒนาด้วยเทคนิคต้นไม้ตัดสินใจ (Decision trees: DT) นาอีฟเบย์ (Naïve bayes: NB) และการวิเคราะห์การถดถอยโลจิสติค (Logistic regression: LR) เพื่อใช้แยกแยะรายงานจุดบกพร่องจากปัญหาดังกล่าว ใน Mozilla, Eclipse และ JBoss ซึ่งได้ข้อสรุปว่า สามารถแสดงผลลัพธ์ที่มีความแม่นยำตั้งแต่ 77% ถึง 82%

ในปี ค.ศ. 2013 Herzig และคณะ [37] ใช้วิธีการตรวจสอบด้วยตนเองพบว่า มีรายงานจุดบกพร่องมากกว่า 7,000 รายงานจาก BTS สองระบบนั่นคือ Bugzilla และ Jira ที่มีการจำแนกผิดพลาด (Misclassified) นอกจากนั้น Herzig และคณะ [37] พบว่า "หนึ่งในสามของรายงานที่กำหนดว่าเป็นจุดบกพร่อง แท้จริงแล้วไม่ใช่รายงานจุดบกพร่อง" นอกจากนั้น ยังพบว่า 40% ของรายงานจุดบกพร่องมีการระบุไว้ไม่ถูกต้อง และมีรายงานถึง 39% ที่ถูกทำเครื่องหมายว่าเป็นจุดบกพร่องแต่กลับไม่มีจุดบกพร่องเลย

ต่อมาในปีเดียวกัน Pingclasai และคณะ [38] ได้เสนอวิธีการเพื่อใช้ในการจำแนกประเภทของรายงานจุดบกพร่องโดยอัตโนมัติ นอกจากนี้ยังได้เปรียบเทียบรูปแบบการจำแนกจาก 3 อัลกอริทึม คือ ต้นไม้ตัดสินใจ นาอีฟเบย์ และการวิเคราะห์การถดถอยโลจิสติค เพื่อหารูปแบบการจำแนกประเภทของรายงานจุดบกพร่องโดยอัตโนมัติที่มีประสิทธิภาพมากที่สุด โดยใช้ข้อมูลรายงานจุดบกพร่องจาก 3 ซอฟต์แวร์โอเพนเซอร์ส คือ HTTPClient, Jackrabbit และ Lucene ซึ่งมีจำนวนข้อมูลรายงานจุดบกพร่องเป็น 745, 2,402 และ 2,443 ตามลำดับ นอกจากนี้ยังได้เปรียบเทียบประสิทธิภาพการจำแนกระหว่าง topic-based model และ word-based model ซึ่งได้ผลของงานวิจัยคือ ค่า F1 อยู่ระหว่าง 0.65 ถึง 0.82 และพบว่านาอีฟเบย์เป็นแบบจำลองการจำแนกที่มีประสิทธิภาพมากที่สุด

ในปี ค.ศ. 2014 Limsetho และคณะ [39] ได้เสนอการพัฒนากรอบอัตโนมัติที่จะสามารถจัดกลุ่มรายงานข้อบกพร่องตามลักษณะและโครงสร้างโดยใช้การจัดกลุ่มด้วยการเรียนรู้แบบไม่มีผู้สอน (Unsupervised learning) โดยในการจัดกลุ่มรายงานจุดบกพร่องจะอาศัยการวัดความคล้ายของข้อความ นอกจากนี้ยังเสนอวิธีการติด label กำกับเป็นชื่อตัวแทนของกลุ่ม ซึ่งในงานวิจัยนี้ใช้รายงานจุดบกพร่องที่รวบรวมจากโครงการ Lucene, Jackrabbit และ HTTPClient

จำนวน 5,441 รายงาน ซึ่งทั้ง 3 โครงการใช้ Jira เป็น BTS การทดลองได้เปรียบเทียบการจัดกลุ่มสองวิธี คือ X-Means และ Expectation Maximization (EM) ซึ่งทั้งสองวิธีนี้เป็นวิธีที่สามารถประมาณจำนวนกลุ่มได้เอง และเทียบกับการจำแนกรายงานจุดบกพร่องด้วยการเรียนรู้แบบมีผู้สอน (Supervised learning) นั่นคือ J48 และการวิเคราะห์การถดถอยเชิงโลจิสติค แม้ผลลัพธ์จะแสดงให้เห็นว่าค่าเฉลี่ยประสิทธิภาพโดยรวมของ X-means มีค่าน้อยกว่าวิธีการเรียนรู้แบบมีผู้สอน แต่ก็ไม่แตกต่างมากนัก

3) เทคนิคในการทำนายความรุนแรงของรายงานจุดบกพร่อง

ผู้รายงานจุดบกพร่อง ในที่นี้อาจจะเป็นผู้ใช้ทั่วไปหรือนักทดสอบซอฟต์แวร์ก็ตาม มักไม่เข้าใจถึงความรุนแรงของจุดบกพร่องเนื่องจากไม่มีประสบการณ์หรือด้วยเหตุผลอื่น ๆ ดังนั้นเพื่อช่วยผู้รายงานจุดบกพร่องในการกำหนดระดับความรุนแรงให้กับรายงานจุดบกพร่องให้เหมาะสม

ในปี ค.ศ. 2008 Menzies และ Marcus [15] ได้นำเสนอการทำนายความรุนแรงของจุดบกพร่องด้วยเทคนิคการทำเหมืองข้อความ (Text mining) และการเรียนรู้ด้วยเครื่อง (Machine learning) โดยเทคนิคการทำเหมืองข้อความที่ใช้ เช่น การตัดคำ (Tokenization) การตัดคำหยุด (Stop words removal) และการแปลงคำให้อยู่ในรูปแบบของรากศัพท์ (Stemming) การให้น้ำหนักของคำด้วย *tf-idf* เพื่อจัดการกับคำอธิบายที่เป็นข้อความของรายงานจุดบกพร่อง ซึ่งงานวิจัยนี้ใช้ Java เวอร์ชันของ Cohen's RIPPER rule learner เพื่อสร้างกฎ โดยงานวิจัยนี้เสนอการทำนายระดับความรุนแรงของจุดบกพร่องแบบอัตโนมัติที่มีชื่อว่า “SEVERIS” และมีกรณีศึกษาจากข้อมูลในโครงการของ NASA และระบบติดตามจุดบกพร่องที่มีชื่อว่า “Project and Issue Tracking System: PITS” ซึ่งจำนวนรายงานจุดบกพร่องที่ใช้ในการศึกษามีทั้งสิ้น 3,877 รายงาน และผลการวิจัยพบว่าให้ผลลัพธ์ที่น่าพอใจ โดยการพิจารณาจากคำศัพท์ที่ top-100 และ top-3 ได้ค่าประสิทธิภาพที่ไม่แตกต่างกัน

ในทำนองเดียวกันปี ค.ศ. 2010 Lamkanfi และคณะ [14] ใช้เทคนิคการทำเหมืองข้อความและการเรียนรู้ด้วยเครื่อง เพื่อทำนายความรุนแรงของรายงานจุดบกพร่องโดยใช้ชุดสอน (Training set) ที่มีขนาดใหญ่จากโครงการโอเพนซอร์ส 3 โครงการ ได้แก่ Mozilla Eclipse และ Gnome งานวิจัยนี้ได้ใช้ตัวจำแนกนาอีฟเบย์ในการวิเคราะห์ระดับความรุนแรงของรายงานจุดบกพร่อง โดยเป้าหมายหลักของงานวิจัยนี้จะเน้นการคัดกรองจุดบกพร่องที่ไม่รุนแรงออกจากจุดบกพร่องที่รุนแรง และจากงานวิจัยนี้สรุปได้ว่าหากข้อมูลที่ใช้ในการสอนที่มีขนาดมากพอ อย่างน้อยประมาณ 500 รายงานในแต่ละระดับความรุนแรง จะสามารถสร้างตัวจำแนกที่จำแนกรายงานจุดบกพร่องได้อย่างน่าพอใจ ซึ่งงานวิจัยนี้ให้ค่าความแม่นยำและค่าความระลึกอยู่ระหว่าง 0.65-0.75 สำหรับรายงานจุดบกพร่องจาก Mozilla และ Eclipse ส่วนในกรณีของรายงานจุดบกพร่องที่มาจาก GNOME นั้น ค่าความแม่นยำและค่าความระลึกจะอยู่ระหว่าง 0.70-0.85

ในปี ค.ศ. 2012 Tian และคณะ [18] ได้นำเสนอแนวทางการค้นคืนข้อมูลเพื่อทำนายระดับความรุนแรงของรายงานจุดบกพร่อง โดยใช้ฟังก์ชันวัดความคล้ายคลึง BM25F เป็นเครื่องมือหลักสำหรับการวิเคราะห์และประเมินความคล้ายคลึงกันระหว่างรายงานจุดบกพร่อง ว่ามีความแตกต่างกันหรือไม่ จากนั้นจึงจะใช้ขั้นตอนวิธีเพื่อนบ้านใกล้ที่สุด (K-nearest neighbour) ใน

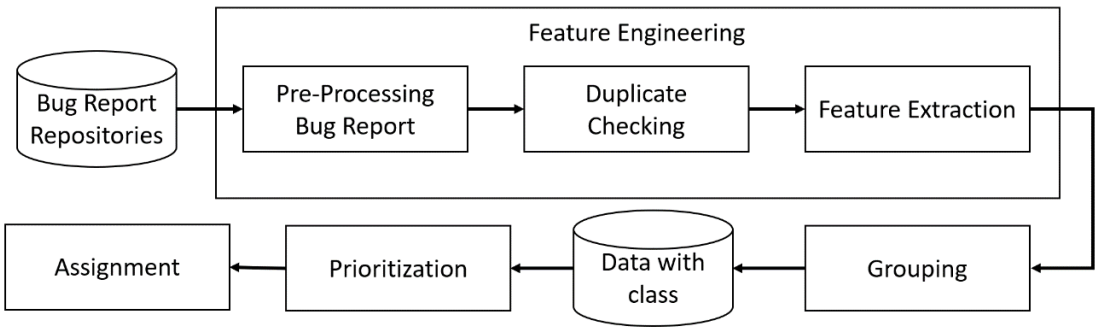
การกำหนดระดับความรุนแรงให้กับรายงานจุดบกพร่อง สำหรับข้อมูลที่ใช้ในการศึกษาจะเป็นข้อมูลจาก OpenOffice Mozilla และ Eclipse จำนวนรวมกันทั้งสิ้นมากกว่า 65,000 รายงาน งานวิจัยนี้มีความคล้ายคลึงกับงานของ Menzies และ Marcus [15] แต่ผลลัพธ์ของงานวิจัยนี้ได้รับการประเมินว่าดีกว่างานวิจัยของ Menzies และ Marcus [15]

2.3.2 การศึกษาที่เกี่ยวข้องการตรวจสอบรายงานจุดบกพร่อง

การคัดแยกรายงานจุดบกพร่อง (Bug report triage) เป็นกระบวนการในการตรวจสอบจุดบกพร่อง เพื่อจัดลำดับความสำคัญของจุดบกพร่อง และกำหนดนักพัฒนาซอฟต์แวร์ที่เหมาะสม [25] ในการแก้ไขจุดบกพร่องนั้นๆ โดยปกติในรายงานจุดบกพร่องส่วนใหญ่ได้รับการกำหนดค่าคุณลักษณะ (Bug report features) บางประการจากผู้รายงาน เช่น องค์ประกอบ (Component) คำอธิบาย (Description) ชื่อเรื่อง (Title or summary) ระดับความรุนแรง (Severity level) หรือไฟล์แนบ เป็นต้น

อย่างไรก็ตามสำหรับโครงการโอเพนซอร์สขนาดใหญ่ เมื่อมีรายงานจุดบกพร่องเข้ามาอาจจะมีรายงานจุดบกพร่องจำนวนมากที่ยังไม่กำหนดให้นักพัฒนาซอฟต์แวร์ที่จะทำการแก้ไข สาเหตุเพราะว่าการคัดแยกรายงานจุดบกพร่องเป็นงานที่ต้องใช้เวลาและความละเอียดในการอ่านเพื่อวิเคราะห์และจัดลำดับความสำคัญ ตรวจสอบความซ้ำซ้อนของรายงานจุดบกพร่อง รวมถึงการกำหนดนักพัฒนาซอฟต์แวร์ที่เหมาะสมที่จะปรับแก้จุดบกพร่องดังที่ปรากฏในงานวิจัย [3, 10-13, 16, 17, 23, 24]

จากการศึกษาพบว่ากระบวนการโดยทั่วไปของการตรวจสอบการรายงานจุดบกพร่องนั้น จะมีขั้นตอน คือ ขั้นตอนของวิศวกรรมคุณลักษณะของจุดบกพร่อง (Feature engineering) ซึ่งครอบคลุมการทำงาน คือ การเตรียมข้อมูลรายงานจุดบกพร่อง (Pre-processing bug report) การตรวจสอบว่าเป็นรายจุดบกพร่องซ้ำ (Duplicate checking) และการสกัดคุณลักษณะของรายงานจุดบกพร่อง (Feature extraction) เมื่อเสร็จสิ้นขั้นตอนของวิศวกรรมคุณลักษณะของจุดบกพร่องแล้วจึงดำเนินการจัดกลุ่มหรือจัดประเภทให้กับรายงานจุดบกพร่อง (Grouping) จากนั้นจะเข้าสู่การกำหนดลำดับความสำคัญของรายงานจุดบกพร่อง (Prioritization) และขั้นตอนสุดท้ายของการคัดแยกรายงานจุดบกพร่อง คือ การกำหนดหรือมอบหมายรายงานจุดบกพร่องให้กับนักพัฒนาซอฟต์แวร์ที่เหมาะสม (Assignment) เพื่อทำการแก้ไข โดยสามารถแสดงได้เป็นแผนภาพดังรูปที่ 2.1



รูปที่ 2.11 กระบวนการทั่วไปในการตรวจสอบรายงานจุดบกพร่อง

เพื่อช่วยให้การทำงานของ Bug traiger ในการคัดแยกจุดบกพร่องของรายงานจุดบกพร่อง รวมถึงทำให้การแก้ไขจุดบกพร่องมีประสิทธิภาพมากขึ้น นักวิจัยหลายท่านจึงมุ่งเน้นงานวิจัยไปที่การตรวจสอบรายงานจุดบกพร่องแบบอัตโนมัติ เช่น การจัดลำดับความสำคัญของรายงานจุดบกพร่อง (Prioritizing bug reports) การตรวจสอบรายงานจุดบกพร่องที่ซ้ำซ้อน (Checking duplication of bug reports) และการกำหนดรายงานจุดบกพร่องให้กับนักพัฒนาซอฟต์แวร์ที่เหมาะสม (Assigning bug reports to developers)

1) การจัดลำดับความสำคัญของรายงานจุดบกพร่อง

ในปี ค.ศ. 2010 Yu และคณะ [16] ได้ประยุกต์เทคนิคโครงข่ายประสาทเทียม (Neural Network) เพื่อทำนายลำดับความสำคัญของรายงานจุดบกพร่อง โดยมีการใช้กระบวนการสอนเชิงวิวัฒน์เพื่อแก้ปัญหาเกี่ยวกับคุณลักษณะใหม่ๆ ของซอฟต์แวร์ ที่มีการรายงานเพิ่มเติมเข้ามา นอกจากนี้งานวิจัยนี้ยังได้นำชุดข้อมูลจากระบบซอฟต์แวร์ที่คล้ายคลึงกันมาใช้ในการเรียนรู้ (Training) เพื่อการวิเคราะห์ที่ครอบคลุมและลดปัญหาข้อผิดพลาดที่อาจจะเกิดขึ้นจากการที่ไม่รู้จักคุณลักษณะของซอฟต์แวร์

ปี ค.ศ. 2012 Kanwal และ Maqbool [17] ได้นำเสนอการจัดลำดับความสำคัญจุดบกพร่อง โดยใช้แบบจำลองการจำแนกด้วยซัพพอร์ตเว็กเตอร์ (Support vector machine: SVM) และนาอีฟเบย์ ซึ่งใช้รายงานจุดบกพร่องจาก Eclipse ที่มีการรายงานเข้ามาในระหว่างปี 2001 ถึง 2006 จำนวนทั้งสิ้น 12,082 รายงาน (โดยแยกเป็นระดับความสำคัญ P1-P5 เป็น 705, 1,073, 9,441, 536 และ 327 ตามลำดับ) นอกจากนั้น ในงานวิจัยนี้ยังได้เปรียบเทียบผลลัพธ์ของแบบจำลองทั้งสองในด้วยค่าความแม่นยำ (Precision) และค่าความระลึก (Recall) รวมไปถึงการพิจารณาค่า False negative และ False positive พบว่าหากใช้แบบจำลองซัพพอร์ตเว็กเตอร์จะให้ผลการทำนายกลุ่มได้ดีเมื่อใช้ข้อมูลรายงานจุดบกพร่องในส่วนของ Summary และ Description ร่วมกัน ในขณะที่แบบจำลองนาอีฟเบย์นั้น หากใช้เพียง Summary และ Description ร่วมกันจะให้ผลลัพธ์ที่ไม่ดีนัก แต่เมื่อเพิ่มข้อมูลในส่วนอื่นๆ ของรายงานจุดบกพร่อง เช่น ข้อมูล Product และ Component กลับทำให้แบบจำลองนาอีฟเบย์ให้ผลลัพธ์ที่ดีขึ้นแล้วน่าพอใจ

2) การตรวจสอบรายงานจุดบกพร่องที่ซ้ำซ้อน

การศึกษาในเรื่องนี้ได้เริ่มต้นในปี ค.ศ. 2007 โดย Runeson และคณะ [10] โดยมีการใช้เทคนิคการประมวลผลภาษาธรรมชาติในการตรวจสอบรายงานข้อผิดพลาดที่ซ้ำซ้อน (Duplicate bug report) ซึ่งมีการศึกษาทดลองโดยใช้ข้อมูลรายงานจุดบกพร่องของ Sony Ericsson Mobile Communication และได้พบว่า 2 ใน 3 ของรายงานที่ซ้ำซ้อนสามารถตรวจจับได้ด้วยการใช้เทคนิคการประมวลผลภาษาธรรมชาติ ซึ่งขั้นตอนในการวิเคราะห์จะประกอบไปด้วย การตัดคำ การตัดส่วนขยายของคำ การตัดคำหยุด โมเดลเชิงพื้นที่ (Vector space model) และการคำนวณความคล้ายคลึง (Similarity) ซึ่งงานวิจัยนี้ใช้การวัดความคล้ายคลึงด้วยโคไซน์ ในการเปรียบเทียบความคล้ายคลึงของเอกสาร นอกจากนี้ในงานยังทำการสร้างพจนานุกรมคำศัพท์ที่ใช้ในรายงานจุดบกพร่อง

เพื่อแก้ไขคำผิด คำพ้อง หรือคำที่สามารถใช้แทนกันได้ ซึ่งเป็นคำในพจนานุกรมจะเป็นคำที่ใช้บ่อยในซอฟต์แวร์ประมาณ 1,000 คำ

ต่อมาในปี ค.ศ. 2008 Jalbert และ Weimer [11] ได้นำเสนอการจัดประเภทรายงานจุดบกพร่องออกเป็น 2 กลุ่มคือ รายงานจุดบกพร่อง และรายงานจุดบกพร่องที่ซ้ำซ้อน โดยใช้ข้อมูลรายงานจุดบกพร่องจาก Mozilla จำนวน 29,000 รายงานในการศึกษา งานวิจัยนี้ใช้ความหมายของข้อความ (Textual semantics) และการจัดกลุ่มกราฟ (Graph clustering) เพื่อทำนายสถานะที่ซ้ำกัน หรือ “DUPLICATED” ซึ่งงานวิจัยนี้ไม่เพียงแต่จัดประเภทรายงานจุดบกพร่องและลำดับรายการของรายงานจุดบกพร่องที่มีความคล้ายกันกับรายงานจุดบกพร่องใหม่ๆ แต่ยังให้ข้อเสนอแนะว่ารายงานจุดบกพร่องใหม่นี้เป็นรายงานจุดบกพร่องที่ซ้ำกันหรือไม่

นอกจากนี้ ในปี ค.ศ. 2012 Tian และคณะ [3] ได้ทำการขยายงานวิจัยของ Jalbert และ Weimer [11] โดยเสนอให้พิจารณาความคล้ายคลึงกันระหว่างรายงานจุดบกพร่องใหม่และรายงานจุดบกพร่องที่มีอยู่หลายรายการ แทนการพิจารณาจากรายงานจุดบกพร่องที่คล้ายกันมากที่สุด เพื่อใช้ในการตัดสินใจว่าจุดบกพร่องที่รายงานใหม่นั้นเป็นจุดบกพร่องที่ซ้ำกันหรือไม่ โดย Tian และคณะ [3] ได้ใช้เทคนิค BM25F แทนที่การพิจารณาความถี่ของคำที่นิยมใช้กันในการศึกษาด้านรายงานจุดบกพร่อง นอกจากนั้นงานวิจัยนี้ยังได้เพิ่มคุณลักษณะประเภท เช่น “ผลิตภัณฑ์ (Product)” ในการพิจารณาเพิ่มเติมจากงานวิจัยก่อนหน้าที่ใช้ “Summary” และ “Description” เนื่องจาก Tian และคณะ [3] ให้ความเห็นว่ารายงานจุดบกพร่องที่กล่าวถึงผลิตภัณฑ์ต่างกันอาจจะมีแนวโน้มที่จะเป็นจุดบกพร่องที่ไม่ซ้ำกัน และงานวิจัยของ Tian และคณะ [3] ได้ผลลัพธ์ของวิธีการที่เสนอดีกว่าของ Jalbert และ Weimer [11] ในทุกประเด็น นอกจากนั้นยังสามารถปรับปรุงค่าความระลึกได้ประมาณ 160% ซึ่งข้อมูลที่ใช้ในการศึกษานี้คือรายงานจุดบกพร่องจาก Mozilla

ต่อมาในปี ค.ศ. 2014 Gopalan และคณะ [12] ได้เสนอวิธีการใหม่เพื่อใช้ในการตรวจสอบรายงานจุดบกพร่องที่ซ้ำซ้อนด้วยเทคนิคการจัดกลุ่ม (Clustering) ที่มีการตรวจสอบความคล้ายคลึงแบบโคไซน์ และมีการระบุเกณฑ์ (Threshold) เพื่อพิจารณาความคล้ายคลึงกันระหว่างรายงานจุดบกพร่องสองรายงานว่ามีความสำคัญมากพอสำหรับการรวมกลุ่มกันหรือไม่ โดยชุดข้อมูลที่ใช้ในการศึกษา ได้แก่ชุดข้อมูลของโครงการ Mozilla OpenOffice และ Eclipse โดยงานวิจัยนี้ใช้ข้อความที่ระบุใน Summary และ Description ของรายงานจุดบกพร่องในการศึกษา นอกจากนี้ Gopalan และคณะ [12] ยังใช้คุณลักษณะอื่นๆ ร่วมในการพิจารณาด้วย อันได้แก่ Product, Component, Version และ Priority และผลลัพธ์ที่ได้มีการนำไปเปรียบเทียบกับงานของ Jalbert และ Weimer [11] รวมทั้ง Tian และคณะ [3] ซึ่งในงานของ Gopalan และคณะ [12] มีค่า True positive rate เพิ่มขึ้นจากทั้งสองงาน แต่กลับมีค่า True negative rate ลดลง

สำหรับงานของ Lee และ คณะ [13] ที่นำเสอนในปี ค.ศ. 2015 ได้ทำการตรวจจับความซ้ำซ้อนของรายงานจุดบกพร่องด้วยการใช้ข้อมูลที่ไม่ถาวร (Temporal information) ซึ่งก็คือ ข้อมูลของเวอร์ชันและเวลาในการส่งซอฟต์แวร์ เข้ามาใช้ในการพิจารณาเชิงเปรียบเทียบกับการตรวจจับความซ้ำซ้อนของรายงานจุดบกพร่องที่ใช้คุณลักษณะ Product Component และ Priority ซึ่งเทคนิคที่ใช้ในการวัดค่าความคล้ายคลึงนั้น จะใช้เทคนิค $BM25F_{ext}$ และมีการจัดลำดับ (Ranking) รายงานที่คล้ายกันด้วยเทคนิค REP ซึ่งชุดข้อมูลรายงานจุดบกพร่องที่ใช้นั้นมาจาก

Eclipse และผลลัพธ์ในงานวิจัยนี้จะให้ค่าค่าความระลึกที่ 0.6409 เมื่อใช้คุณลักษณะ Product Component และ Priority ในการตรวจสอบความคล้ายคลึง ซึ่งมีประสิทธิภาพเพิ่มขึ้น 1.91% เมื่อเทียบกับการใช้คุณลักษณะที่เป็นข้อมูลที่ไม่ถาวร

3) การกำหนดรายงานจุดบกพร่องให้กับนักพัฒนาซอฟต์แวร์

ปี ค.ศ. 2009 Jeong และคณะ [23] แนะนำ Tossing Graph Model เพื่อลดข้อผิดพลาดในขั้นตอน Tossing ซึ่ง Tossing ในที่นี้ คือ การที่นักพัฒนาซอฟต์แวร์คนหนึ่งๆ ได้รับมอบให้ทำการแก้ไขจุดบกพร่องหนึ่ง แต่อาจจะทำไม่ได้หรือไม่สะดวกที่จะทำ จึงทำการส่งต่อจุดบกพร่องนั้นไปยังนักพัฒนาซอฟต์แวร์รายอื่นเพื่อทำการแก้ไขแทน โดยโมเดลที่เสนอนี้มีประสิทธิภาพ เนื่องจากได้รับการประเมินว่าสามารถลดเหตุการณ์การ Tossing ได้ 72% และสามารถปรับปรุงความถูกต้องและเหมาะสมในการกำหนดนักพัฒนาซอฟต์แวร์ในการแก้ไขรายงานจุดบกพร่องแบบอัตโนมัติได้ 23%

นอกจากนี้ ในปี ค.ศ. 2010 Bhattacharya และ Neamtiu [24] ได้ขยายการศึกษาต่อจากงานวิจัยของ Jeong และคณะ [23] โดยเพิ่มคุณลักษณะเกี่ยวกับขอบ (Edge) และโหนด (Node) ใน Tossing Graph Model โดยใช้ข้อมูลรายงานจุดบกพร่องจาก Mozilla และ Eclipse จำนวน 856,259 รายงาน ในการศึกษาผลการทดลองได้แสดงให้เห็นว่าสามารถลดเส้นทางของการ Tossing ได้ 86.31% และให้ความถูกต้องในการคาดการณ์ว่ารายงานจุดบกพร่องนั้นได้มอบหมายไปยังนักพัฒนาซอฟต์แวร์ได้อย่างเหมาะสมถึง 83.62%

2.3.3 การศึกษาที่เกี่ยวข้องแนวทางการแก้ไขจุดบกพร่อง

นอกจากการวิจัยเกี่ยวกับการเพิ่มประสิทธิภาพรายงานจุดบกพร่องและการตรวจสอบรายงานจุดบกพร่องแล้ว ยังมีการวิจัยเกี่ยวกับการแก้ไขจุดบกพร่อง (Bug fixing) ซึ่งมีงานวิจัยในด้านนี้ ได้แก่ การระบุตำแหน่งของจุดบกพร่องในซอฟต์แวร์ตามรายงานจุดบกพร่อง (Bug localization) การกู้คืนลิงก์ระหว่างรายงานจุดบกพร่องและไฟล์การเปลี่ยนแปลง (Recovering links between bug reports and change files) และการคาดการณ์เวลาในการแก้ไขข้อบกพร่อง (Bug-fixing time prediction)

1) การระบุตำแหน่งของจุดบกพร่องในซอฟต์แวร์ตามรายงานจุดบกพร่อง

Zhou และคณะ [40] ได้เสนอ BugLocator ในปี ค.ศ. 2012 ซึ่งเป็นเครื่องมือในการค้นหาไฟล์โปรแกรมที่เกี่ยวข้อง เพื่อเป็นประโยชน์ต่อการแก้ไขจุดบกพร่อง โดย BugLocator จะสืบค้นและจัดอันดับไฟล์โปรแกรมทั้งหมดที่สอดคล้องกับรายงานจุดบกพร่อง ด้วยตามค่าความคล้ายคลึงภายใต้เทคนิคที่ชื่อว่า revised Vector Space Model (rVSM) งานวิจัยนี้ใช้รายงานจุดบกพร่องจำนวนกว่า 3,000 จุดบกพร่อง เพื่อทำการทดลองการค้นหาไฟล์โปรแกรมที่เกี่ยวข้องในโครงการโอเพนซอร์ส 4 โครงการ ผลลัพธ์แสดงให้เห็นว่า BugLocator สามารถค้นหาไฟล์โปรแกรมเพื่อจะแก้ไขจุดบกพร่องได้อย่างมีประสิทธิภาพ

ต่อมาในปี ค.ศ. 2016 Almhana และคณะ [41] ได้เสนอการระบุตำแหน่งและจัดอันดับคลาสที่เกี่ยวข้องกับรายงานจุดบกพร่องแบบอัตโนมัติ โดยงานของพวกเขาความถูกต้องใน

การจัดลำดับของคลาสในโปรแกรมที่ถูกแนะนำขึ้นมานั้น จะขึ้นกับประวัติการแก้ไขจุดบกพร่อง, ความคล้ายคลึงของคำ ระหว่างคำอธิบายในรายงานจุดบกพร่องกับคำอธิบายในซอร์สโค้ด (Source code) รวมทั้งพิจารณาจากความคล้ายของคำอธิบายในรายงานจุดบกพร่องกับเอกสารคำอธิบายประกอบของ API ที่ถูกเรียกใช้ในคลาสของโปรแกรม งานวิจัยนี้ได้ศึกษากับโครงการโอเพนซอร์ส 6 ตัว คือ UI Eclipse, Tomcat, AspectJ, Birt, SWT และ JDT ซึ่งทั้งหมดใช้รายงานจุดบกพร่องมากกว่า 22,000 รายงานจาก BTS ของ Bugzilla ผลลัพธ์ในงานวิจัยของ Almhana และคณะ [41] เมื่อพิจารณาจากค่าความแม่นยำ@k ค่าความระลึก@k และค่าความถูกต้อง@k เมื่อ *k* คือจำนวนไฟล์โปรแกรมถูกค้นคืนที่จำนวน 5, 10 ,15 และ 20 ไฟล์ตามลำดับ ได้แสดงให้เห็นว่าทำงานได้ดีกว่า BugScout BugLocator และ Learning-to-rank ที่นำเสนอออกมาก่อนหน้า และเมื่อทดสอบการค้นคืนไฟล์ที่จำนวน 10 คลาส สามารถให้ค่าความถูกต้องของตำแหน่งที่ต้องแก้ไขจุดบกพร่องได้สูงถึง 87%

Ye และคณะ [42] ได้เสนอวิธีอัตโนมัติที่จัดอันดับไฟล์ซอร์สโค้ดโดยคำนึงถึงความเกี่ยวข้องของรายงานจุดบกพร่องอาจช่วยให้กระบวนการค้นหาจุดผิดพลาดเร็วขึ้น งานวิจัยนี้ใช้ข้อมูลรายงานจุดบกพร่องจาก 6 โครงการเช่นเดียวกับงานของ Almhana และคณะ [41] มีการใช้คุณลักษณะทั้งหมด 19 คุณลักษณะ โดยใช้ประโยชน์จากไฟล์ซอร์สโค้ดโปรแกรม คำอธิบาย API ของส่วนประกอบของไลบรารี ประวัติการแก้ไขจุดบกพร่อง ประวัติการเปลี่ยนแปลงซอร์สโค้ดโปรแกรม และกราฟการพึ่งพาไฟล์ ซึ่งใน 19 คุณลักษณะที่ใช้นั้นนำคำอธิบายทั้งในส่วนของ Summary และ Description ที่ปรากฏในรายงานจุดบกพร่องมาวัดความคล้ายคลึงเทียบกับซอร์สโค้ดโปรแกรม ในส่วนของคำอธิบาย API ชื่อคลาส ชื่อตัวแปร และชื่อเมธอด เป็นต้น รวมทั้งยังมีการหาความถี่ในการแก้ไข และคำนวณหาค่าความซับซ้อนของซอร์สโค้ด โดยผลลัพธ์ที่ได้นั้น แสดงให้เห็นว่าทำงานได้ดีกว่าวิธีการของ BugScout BugLocator และ BLUiR เพราะสามารถค้นหาไฟล์โปรแกรมที่เกี่ยวข้องกับรายงานจุดบกพร่องได้มากกว่า 70% ใน Eclipse Platform และ Tomcat นอกจากนี้งานวิจัยยังแสดงให้เห็นถึงคุณลักษณะที่สำคัญที่สุด ที่มีผลต่อการแก้ไขจุดบกพร่องให้ความถูกต้องในแต่ละโครงการด้วย

2) การกู้คืนลิงก์ระหว่างรายงานจุดบกพร่องและไฟล์การเปลี่ยนแปลง

ในรายงานวิจัยของ Fischer และคณะ [41] เมื่อปี ค.ศ. 2003 ได้เสนอให้มีการเก็บรายงานจุดบกพร่องในฐานข้อมูลแบบที่แสดงเวอร์ชันของการแก้ไข โดยเฉพาะรายงานจุดบกพร่องที่มีการเก็บบันทึกการเปลี่ยนแปลงเอาไว้ด้วยกัน เพื่อให้ง่ายต่อการดูในเชิงเปรียบเทียบระหว่างรายงานจุดบกพร่องในเวอร์ชันเก่าและใหม่ว่ามีความแตกต่างกันอย่างไร เพราะบันทึกการเปลี่ยนแปลงมักจะถูกเก็บไว้ในคลังข้อมูลเวอร์ชัน ที่มีข้อมูลไม่เพียงพอสำหรับการวิเคราะห์วิวัฒนาการซอฟต์แวร์ระหว่างการควบคุมเวอร์ชันและระบบติดตามจุดบกพร่อง

3) การคาดการณ์เวลาในการแก้ไขข้อบกพร่อง

ในปี ค.ศ. 2007 Weiss และคณะ [7] นำเสนอ Lucene framework ในการวิเคราะห์ความคล้ายคลึงของรายงานจุดบกพร่องเพื่อการคาดการณ์เวลาในการแก้ไขข้อผิดพลาดของ

ซอฟต์แวร์ ซึ่งนักวิจัยจะนำรายงานจุดบกพร่องที่รายงานเข้ามาใหม่ ไปเปรียบเทียบกับรายงานจุดบกพร่องเดิมที่มีอยู่ ซึ่งจุดบกพร่องในรายงานเดิมเหล่านั้นได้รับการแก้ไขปัญหาไปแล้ว และมีการบันทึกระยะเวลาในการแก้ไขปัญหาไว้ในแต่ละรายงานจุดบกพร่องเดิมเหล่านั้นด้วย ดังนั้นหากรายงานจุดบกพร่องใหม่ถูกวิเคราะห์ว่ามีความคล้ายคลึงกับรายงานจุดบกพร่องเก่าตัวใด ก็จะไปตรวจสอบว่าจุดบกพร่องในรายงานเก่านั้นใช้เวลาในการแก้ไขปัญหาเท่าไหร่ และเวลาที่ใช้ในการแก้ปัญหานั้นก็น่าจะเป็นเวลาที่ใช้ในการแก้ไขจุดบกพร่องที่ปรากฏในรายงานที่เข้ามาใหม่นั่นเอง ซึ่งงานวิจัยนี้ใช้รายงานจุดบกพร่องจากซอฟต์แวร์ที่ชื่อ JBoss ที่ใช้ระบบติดตามจุดบกพร่อง Jira จำนวน 567 รายงานในการศึกษา

ต่อมาในปี ค.ศ. 2011 Bhattacharya และ Neamtiu [1] ได้เสนอสมมติฐานเรื่อง “นักพัฒนาซอฟต์แวร์ที่มีชื่อเสียงในการแก้ไขจุดบกพร่อง จะส่งผลให้รายงานจุดบกพร่องที่นักพัฒนาซอฟต์แวร์เหล่านี้เป็นผู้รายงานขึ้นมา ได้รับการแก้ไขจุดบกพร่องก่อนหรือไม่” โดยงานวิจัยนี้ ได้ดำเนินการทดลองกับรายงานจุดบกพร่องมากกว่า 512,474 รายงาน จากซอฟต์แวร์ขนาดใหญ่ 5 โครงการ ได้แก่ Chrome Eclipse FireFox Seamonkey และ Thunderbird โดยใช้การทดสอบด้วยถดถอยพหุคูณแบบหลายตัวแปรและตัวแปรเดียว ผลการวิจัยนี้พบว่า ไม่มีความสัมพันธ์เชิงเส้นระหว่างชื่อเสียงของผู้รายงานและเวลาในการแก้ไขจุดบกพร่อง แต่งานวิจัยนี้ได้อธิบายถึงการแก้ไขจุดบกพร่องที่ไม่สำเร็จโดยสมบูรณ์ มีเหตุดังนี้ การระบุความรุนแรงของจุดบกพร่องผิด, จำนวนนักพัฒนาซอฟต์แวร์ที่ Tossing จนถึงนักพัฒนาซอฟต์แวร์ที่แก้ไขสำเร็จ, ไฟล์แนบที่เพิ่มจำนวนเข้ามาเนื่องจากต้องตรวจสอบไฟล์แนบเหล่านั้นก่อน และการขึ้นต่อกันของจุดบกพร่อง (Bug dependency) ซึ่งจากรายงานผลการวิจัยของ Bhattacharya และ Neamtiu [1] นี้ สรุปว่า การแก้ไขจุดบกพร่องที่จะสำเร็จหรือไม่ มีสาเหตุหลักมาจาก 4 สาเหตุ คือ (1) การระบุความรุนแรงของจุดบกพร่องผิด [14, 15, 18] (2) จำนวนนักพัฒนาซอฟต์แวร์ที่ Tossing จนถึงนักพัฒนาซอฟต์แวร์ที่ทำการแก้ไขจุดบกพร่องสำเร็จ [23, 24] (3) ไฟล์แนบที่อธิบายลักษณะการเกิดความผิดพลาด เพราะควรตรวจสอบไฟล์แนบเหล่านั้นก่อน และ (4) การขึ้นต่อกันของจุดบกพร่อง นั่นหมายความว่า “นักพัฒนาซอฟต์แวร์ที่มีชื่อเสียงในการแก้ไขจุดบกพร่อง ไม่ส่งผลให้รายงานจุดบกพร่องที่นักพัฒนาซอฟต์แวร์เหล่านี้เป็นผู้รายงานขึ้นมา ได้รับการแก้ไขความบกพร่องก่อน” นั่นเอง

ค.ศ. 2012 Ohira และคณะ [43] ได้ศึกษาผลกระทบเกี่ยวกับรูปแบบการจัดการเพื่อการแก้ไขจุดบกพร่อง ซึ่งนักวิจัยกลุ่มนี้ได้ใช้ข้อมูลรายงานจุดบกพร่องจาก Platform จำนวน 21,308 รายงาน และรายงานจุดบกพร่องจาก JDT จำนวน 8,110 รายงาน โดยเป็นรายงานของ Eclipse จาก BTS ของ Bugzilla ซึ่ง Ohira และคณะ [43] ได้ค้นพบรูปแบบของการจัดการจุดบกพร่องโดยพิจารณาจากการปฏิสัมพันธ์ของบุคคล 3 กลุ่มต่อไปนี้ คือ ผู้รายงาน (Reporter: R), ผู้ตรวจสอบรายงานจุดบกพร่อง (Triager: T), และผู้แก้ไขจุดบกพร่อง (Fixer or Developer: F) ซึ่งอ้างอิงจากขั้นตอนของการรายงานจุดบกพร่องจนถึงการแก้ไขจุดบกพร่องนั่นเอง และรูปแบบที่ค้นพบมีทั้งสิ้น 4 รูปแบบคือ

(1) R=T=F หมายถึง Reporter, Triager และ Fixer เป็นคนเดียวกัน
(2) R=T≠F หมายถึง Reporter เป็นคนเดียวกับ Triager แต่เป็นคนละคนกับ Fixer

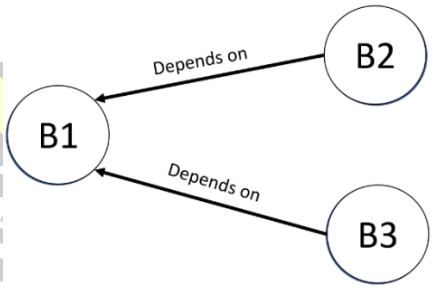
(3) R≠T=F หมายถึง Triager เป็นคนเดียวกับ Fixer แต่ไม่ใช่คนเดียวกับ Reporter

(4) R≠T≠F หมายถึง Triager Fixer และ Reporter ไม่ใช่คนเดียวกันทั้งหมด

และจากรูปแบบการจัดการเพื่อการแก้ไขจุดบกพร่องทั้ง 4 แบบ นักวิจัยกลุ่มนี้ได้เปรียบเทียบระยะเวลาในการมอบหมายจุดบกพร่องเพื่อแก้ไข (Time assingment) และระยะเวลาในการแก้ไขจุดบกพร่อง (Time fix) พบว่ารูปแบบที่ 1 มีความรวดเร็วที่สุดทั้งในการมอบหมายจุดบกพร่องและระยะเวลาในการแก้ไขจุดบกพร่อง และรูปแบบที่ 3 และ 4 มักจะใช้เวลาทั้งสองยาวนานกว่า นั่นเป็นเพราะการที่ Reporter และ Triager ไม่ใช่คนเดียวกันข้อมูลที่ได้จากรายงานจุดบกพร่องอาจไม่มีข้อมูลที่สมบูรณ์มากพอที่ Triager จะจำเป็นต้องใช้ ซึ่งรายงานดังกล่าวสอดคล้องกับรายงานวิจัยของ Bettenburg และคณะ [20, 21] รวมถึงรายงานวิจัยของ Davies และ Roper [36] อีกด้วย

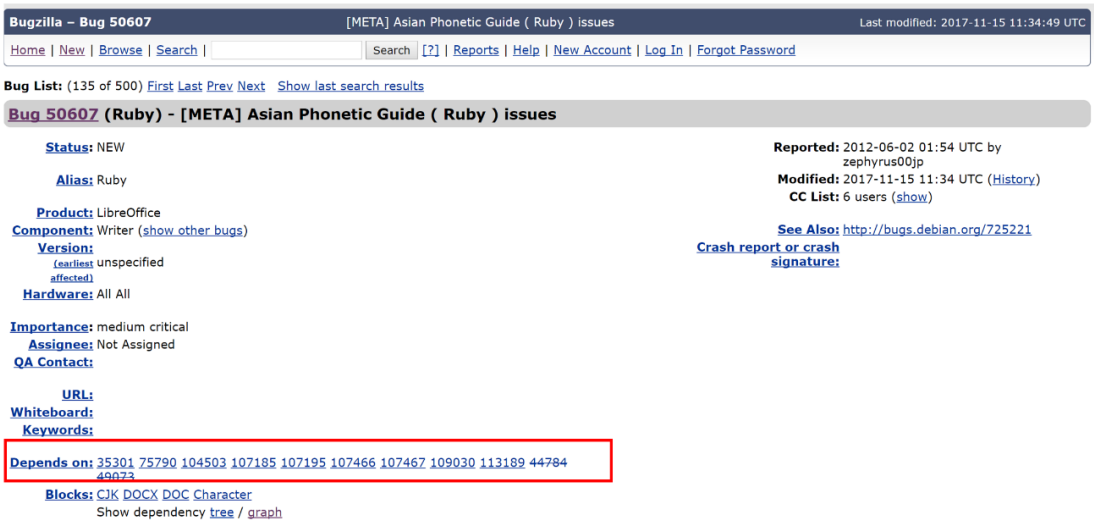
2.4 ส่วนต่อของจุดบกพร่อง (Bug dependency)

จากการศึกษาพบว่าปัญหาวิจัยในประเด็นเรื่องการขึ้นต่อกันของจุดบกพร่อง ที่ได้แสดงความเห็นไว้โดย Bhattacharya และ Neamtiu [1] และ Sandusky และคณะ [26] โดยทั้งสองงานวิจัยได้ลงความเห็นว่า “ส่วนต่อของจุดบกพร่อง” เป็นปัญหาหนึ่งในงานวิจัยด้านซอฟต์แวร์ที่สำคัญ เนื่องจาก ถ้าจุดบกพร่อง B_1 เป็นส่วนต่อของจุดบกพร่อง B_2 และ B_3 (ซึ่งแสดงได้ดังรูปที่ 2.12) ดังนั้นหากจุดบกพร่อง B_2 และ B_3 ยังไม่ได้รับการแก้ไข การแก้ปัญหาจุดบกพร่อง B_1 ก็จะไม่สามารถแก้ไขได้อย่างสมบูรณ์ ดังนั้นในระบบติดตามรายงานจุดบกพร่องจึงควรต้องมีการวิเคราะห์ถึงส่วนต่อของจุดบกพร่องด้วย เพื่อให้การมอบหมายจุดบกพร่องไปยังนักพัฒนาซอฟต์แวร์เพื่อแก้ไขจุดบกพร่องมีความเหมาะสมและมีแนวทางในการแก้ไขจุดบกพร่องได้อย่างสมบูรณ์มากที่สุด

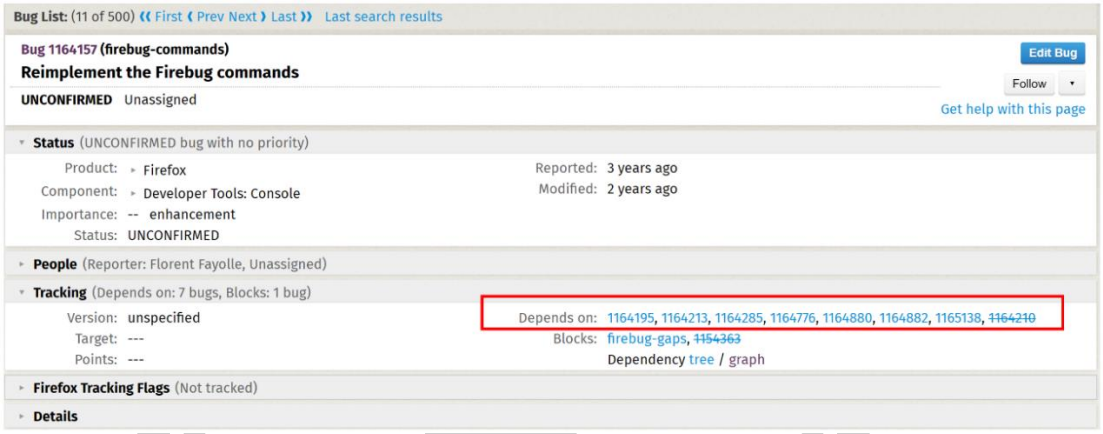


รูปที่ 2.12 ภาพจำลองแสดงการเป็นส่วนต่อของจุดบกพร่อง

จากการศึกษารายงานจุดบกพร่องจาก Mozilla Firefox และ LibreOffice พบว่าทั้งสองซอฟต์แวร์ได้ตระหนักถึงปัญหาส่วนต่อของจุดบกพร่อง เพราะจะเห็นได้จากที่นักพัฒนาซอฟต์แวร์ของทั้งสองโครงการได้วิเคราะห์รายงานจุดบกพร่องและแสดงส่วนต่อของจุดบกพร่องไว้ในแอตริบิวท์ ที่ชื่อ “Depends on” ดังแสดงในรูปที่ 2.13 และ รูปที่ 2.14

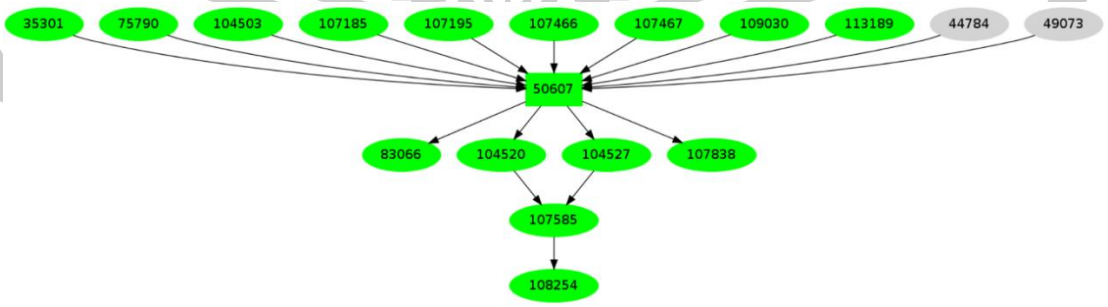


รูปที่ 2.13 รายงานจุดบกพร่องจาก LibreOffice ที่แสดงส่วนต่อของจุดบกพร่อง
ที่มา : https://bugs.documentfoundation.org/

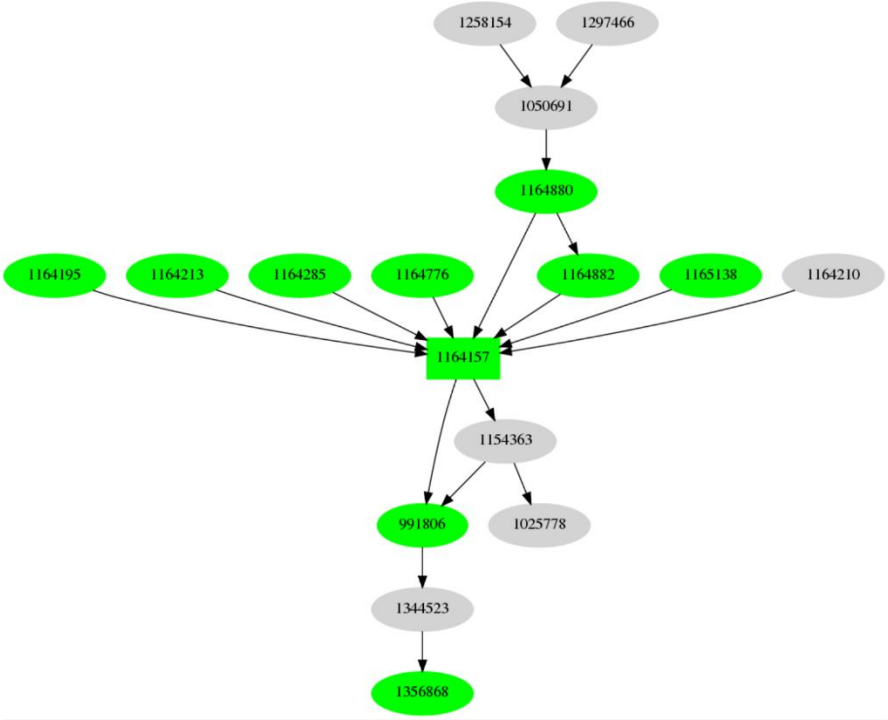


รูปที่ 2.14 รายงานจุดบกพร่องจาก Mozilla Firefox ที่แสดงข้อมูลส่วนต่อของจุดบกพร่อง
ที่มา : https://bugzilla.mozilla.org/

และจากตัวอย่างจุดบกพร่องทั้งสองนี้ สามารถแสดงเป็นแผนภาพเพื่อแสดงส่วนต่อของจุดบกพร่องทั้งสองตัวอย่างได้ดังนี้



รูปที่ 2.15 ตัวอย่างส่วนต่อของจุดบกพร่องที่แสดงในรายงานจุดบกพร่องจาก LibreOffice
ที่มา : https://bugs.documentfoundation.org/



รูปที่ 2.16 ตัวอย่างส่วนต่อของจุดบกพร่องที่แสดงในรายงานจุดบกพร่องจาก Mozilla Firefox
ที่มา : https://bugzilla.mozilla.org/

ซึ่งจากรูปที่ 2.15 และ รูปที่ 2.16 แสดงให้เห็นถึงส่วนต่อของจุดบกพร่อง และจากตัวอย่างในรูปที่ 2.15 ที่เป็นตัวอย่างส่วนต่อของจุดบกพร่องที่แสดงในรายงานจุดบกพร่องจาก LibreOffice จะเห็นว่ารายงานจุดบกพร่องหมายเลข 50607 จะแก้ไขสำเร็จได้จะต้องทำการแก้ไขจุดบกพร่องที่แสดงในรายงานจุดบกพร่องหมายเลข 35301, 44784, 49073, 75790, 104503, 107185, 107195, 107466, 107467, 109030 และ 113189 เสียก่อน แต่จะเห็นว่าเพิ่งแก้ไขจุดบกพร่องในรายงานหมายเลข 44784 และ 49073 เรียบร้อย ดังนั้นหากจะแก้ไขจุดบกพร่องในรายงานหมายเลข 50607 ในขณะนี้ จึงไม่สามารถแก้ไขจุดบกพร่องได้อย่างสมบูรณ์

จากข้างต้น การวิเคราะห์เพื่อแสดงให้เห็นถึงส่วนต่อของรายงานจุดบกพร่องเป็นสิ่งที่จำเป็นอย่างมาก หากต้องการแก้ไขจุดบกพร่องในซอฟต์แวร์ให้สมบูรณ์ ซึ่งจะส่งผลกระทบต่อค่าใช้จ่ายและเวลาที่อาจจะเพิ่มมากขึ้นในขั้นตอนของการบำรุงรักษาซอฟต์แวร์ (Software maintenance) ที่ปกติก็เป็นขั้นตอนที่มีค่าใช้จ่ายมากที่สุดในวัฏจักรของการพัฒนาซอฟต์แวร์ (Software development life cycle: SDLC) อยู่แล้ว

ปัจจุบัน การวิเคราะห์หรือพิจารณาส่วนต่อของจุดบกพร่องยังเป็นการดำเนินการโดยนักพัฒนาซอฟต์แวร์ ซึ่งทำให้งานนี้กลายเป็นภาระงานที่ใช้เวลามากขึ้น สิ่งที่ตามมาคือ ระยะเวลาของการแก้ไขจุดบกพร่องก็จะเพิ่มขึ้นตามไปด้วย [1, 3, 10, 11, 14] ดังนั้นหากสามารถสกัดความสัมพันธ์ระหว่างจุดบกพร่องเพื่อหาแสดงส่วนต่อของจุดบกพร่องแบบอัตโนมัติได้ ก็น่าจะช่วยเพิ่มประสิทธิภาพและลดเวลาในการแก้ไขจุดบกพร่องได้ดียิ่งขึ้น

2.5 การประมวลผลภาษาธรรมชาติ (Natural language processing: NLP)

การประมวลผลภาษาธรรมชาติ [44] บางครั้งเรียก Computational Linguistics เป็นการผสมผสานระหว่างปัญญาประดิษฐ์และภาษาศาสตร์ ที่ศึกษาปัญหาในการประมวลผลและใช้งานภาษาธรรมชาติ รวมทั้งการทำความเข้าใจภาษาธรรมชาติ ทั้งนี้เพื่อให้คอมพิวเตอร์สามารถเข้าใจภาษามนุษย์ได้ ในการประมวลผลภาษาธรรมชาติจะมีหลายระดับ ขึ้นกับวัตถุประสงค์ในการทำงาน โดยทั่วไป จะสามารถแบ่งระดับของการประมวลผลภาษาธรรมชาติได้ 5 ระดับ [45] ได้แก่

1) ระดับการผสมคำ (Morphological level)

ระดับการผสมคำ คือการนำเอาส่วนที่ออกเสียงในคำแต่ละคำที่เรียกว่า “พยางค์” มาปะติดปะต่อเข้าด้วยกันให้เกิดเป็นคำที่มีความหมาย ความรู้ที่จำเป็นต่อการวิเคราะห์คำ ได้แก่ ความรู้เกี่ยวกับคำ (Lexical knowledge) เป็นความรู้ที่เกี่ยวกับคำ การสะกด การจัดกลุ่มตัวอักษรเป็นคำ (Morphological data) การรวมคำ ผันคำ และการแบ่งแยกคำในประโยค

2) ระดับไวยากรณ์หรือโครงสร้างประโยค (Syntactic level)

ระดับไวยากรณ์ เป็นการศึกษาเกี่ยวกับหน้าที่ของคำ และคำในประโยครวมทั้งความสัมพันธ์ระหว่างคำดังกล่าว ซึ่งตำแหน่งของคำในประโยคสามารถบอกได้ว่าคำๆ นั้นทำหน้าที่เป็นประธาน (Subject) หรือกรรม (Object) ของประโยค ความรู้ที่จำเป็นต่อการประมวลผลระดับไวยากรณ์ คือ ความรู้เกี่ยวกับไวยากรณ์ (Syntactic knowledge) เป็นการนำคำมารวมกันเป็นวลี การวิเคราะห์ไวยากรณ์ การตรวจสอบไวยากรณ์ การสร้างประโยค

3) ระดับการตีความ (Semantic level)

ระดับการตีความเป็นการสำรวจความหมายของคำ แต่ละคำคล้ายกับการเปิดหาความหมายของคำในพจนานุกรม และจะต้องหาความหมายของคำเมื่อถูกใช้อยู่ในประโยคด้วย เนื่องจากคำบางคำสามารถมีความหมายได้หลายความหมายขึ้นอยู่กับบริบทที่ถูกใช้ในประโยคนั้นๆ ความรู้ที่จำเป็นต่อการประมวลผลระดับการตีความ คือ ความรู้เกี่ยวกับความหมาย (Semantic knowledge) เป็นความรู้เกี่ยวกับความหมายของภาษา ซึ่งได้แก่ มโนทัศน์ ข่ายความหมาย (Semantic nets)

4) ระดับวาทกรรม (Discoures level)

ระดับวาทกรรม เป็นการสำรวจเกี่ยวกับโครงสร้างของข้อความแต่ละประเภท และโครงสร้างของเอกสารเพื่อค้นหาความหมายเพิ่มเติม ความรู้ที่จำเป็นต่อการประมวลผลระดับวาทกรรม คือ ความรู้ทางความหมายเกี่ยวเนื่อง (Discourse knowledge) เป็นความรู้เกี่ยวกับรูปแบบของผู้ใช้ การตอบโต้และสนทนา การถ่ายทอดความหมายและการตัดสินปัญหาโดยใช้ความเข้าใจ นอกจากนี้ยังต้องอาศัยความรู้เกี่ยวกับการถ่ายทอดความรู้ (Inferential knowledge) เป็นความรู้ที่ใช้ในหลักการของปัญญาประดิษฐ์เกี่ยวกับเทคนิคการสร้างกฎเกณฑ์การถ่ายทอดและขบวนการวินิจฉัย กฎเกณฑ์การโต้ตอบและการสนทนา รวมถึงการแทนด้วยหลักการทางคณิตศาสตร์ตรรกและการวินิจฉัยหาคำตอบ

5) ระดับในทางปฏิบัติ (Pragmatic Level)

ระดับในทางปฏิบัติ เป็นการแปลความหมายของประโยค เพื่อดูความตั้งใจในการสื่อสารของผู้สื่อสารว่าจุดประสงค์จะกล่าวถึงอะไร เพราะประโยคที่พูดออกมาบางครั้งอาจจะไม่ได้มีความหมายตรงตามข้อความนั้นๆ จึงจะต้องตีความตามสถานการณ์ที่เกิดขึ้น โดยที่ทั้งผู้ส่งข่าวสารและผู้รับข่าวสารจะต้องอยู่ในสถานการณ์เดียวกัน

เทคนิคที่ใช้สำหรับการวิเคราะห์ภาษาธรรมชาติ มีหลายเทคนิคโดยเทคนิคบางส่วนที่เกี่ยวข้องกับงานวิจัยนี้แสดงได้ดังต่อไปนี้

2.5.1 การตัดคำ (Word segmentation หรือ Tokenization)

การตัดคำเป็นกระบวนการนำเอกสารข้อความมาทำการแบ่งเป็นประโยค (Sentence) หรือ คำ (Word) เนื่องจากคำจัดว่าเป็นหน่วยที่เล็กที่สุดในภาษา ที่สื่อความหมายได้ ซึ่งโดยทั่วไปในการตัดคำในภาษาอังกฤษนิยมใช้ ช่องว่าง (White space) มหัพภาคหรือจุด (Point : .) จุลภาคหรือคอมมา (Comma: ,) เครื่องหมายอัฒภาค หรือเซมิโคลอน (Semicolon: ;) เครื่องหมายปรัศนี (Question mask: ?) หรือสัญลักษณ์ต่างๆ [46] กระบวนการตัดคำนั้นจะเริ่มจากสำรวจข้อความทั้งหมดเพื่อหาขอบเขตคำและประโยค ซึ่งภาษาอังกฤษ จะใช้ช่องว่างที่คั่นระหว่างคำในการตัดคำและจะใช้จุด “.” เพื่อบอกการจบประโยค [47] ดังแสดงตัวอย่างได้ในตารางที่ 2.3

ตารางที่ 2.3 ตัวอย่างการตัดคำภาษาอังกฤษ

ข้อมูลนำเข้า	ผลลัพธ์
The sky is beautiful.	The / sky / is / beautiful
Have you got any friends?	Have / you / got / any / friends

เทคนิคที่ประยุกต์ใช้ในการตัดคำสามารถแบ่งออกเป็น 3 วิธีหลัก คือ การใช้กฎไวยากรณ์ทางภาษา (Rule-based) การอ้างอิงคำจากพจนานุกรม (Dictionary-based) และการสร้างโมเดลการเรียนรู้จากฐานข้อความขนาดใหญ่ (Machine learning or corpus based)

1) การใช้กฎไวยากรณ์

การตัดคำโดยการใช้กฎไวยากรณ์ นั้นเป็นการตัดคำจากการตรวจสอบกฎเกณฑ์ทางอักขระวิธีซึ่งกำหนดลักษณะของการประสมอักษร ลักษณะการเว้นวรรค และการขึ้นย่อหน้า เพื่อบ่งบอกขอบเขตของคำ วิธีนี้มีข้อดี คือ จะมีความถูกต้องของการตัดคำระดับพยางค์สูง ใช้ทรัพยากรน้อย และประมวลผลได้รวดเร็ว แต่มีข้อจำกัดคือ ความถูกต้องของการตัดคำข่อนข้างต่ำ

2) การอ้างอิงคำจากพจนานุกรม

การตัดคำโดยการอ้างอิงจากพจนานุกรมเป็นการตัดคำโดยใช้สายอักขระ มาเทียบกับคำที่มีอยู่ภายในพจนานุกรม วิธีการนี้จำเป็นจะต้องทำพจนานุกรมขึ้นเพื่อจัดเก็บคำไว้ก่อน ข้อดีคือ ทำให้ได้ความถูกต้องจากการตัดคำได้สูงกว่าวิธีการใช้กฎไวยากรณ์ ข้อจำกัดของวิธีการนี้ คือ ใช้เวลามากกว่าการใช้กฎไวยากรณ์

3) การสร้างโมเดลการเรียนรู้จากฐานข้อความขนาดใหญ่

การตัดคำโดยการสร้างโมเดลการเรียนรู้จากฐานข้อความขนาดใหญ่ เป็นการตัดคำที่นำวิธีทางสถิติมาใช้ในการประมวลผลภาษา โดยใช้คลังข้อมูลทางภาษาเป็นฐานความรู้เก็บค่าความถี่ที่ใช้ในการตัดคำ ซึ่งการตัดคำด้วยคลังข้อมูลแบบนี้ มี 2 วิธี การตัดคำโดยอาศัยความน่าจะเป็น (Probabilistic word segmentation) และ วิธีการตัดคำโดยอาศัยคุณลักษณะของคำ (Feature-based word segmentation)

2.5.2 การกำจัดคำหยุด (Stop-words removal)

การกำจัดคำหยุด เป็นกระบวนการนำคำที่ไม่มีนัยสำคัญออกไป ซึ่งไม่ทำให้ความหมายของเอกสารเปลี่ยนแปลงไป คำที่ไม่มีนัยสำคัญนี้ หมายถึง คำที่ใช้กันโดยทั่วไปไม่มีความสำคัญต่อเอกสารนั้นๆ เมื่อนำออกจากเอกสารแล้วไม่ทำให้ใจความสำคัญของเอกสารเปลี่ยนแปลง ส่วนใหญ่เป็นคำบุพบท (Prepositions) คำสันธาน (Conjunctions) หรือคำสรรพนาม (Pronouns) [10, 48] ดังแสดงใน ตารางที่ 2.4

ตารางที่ 2.4 ตัวอย่างคำหยุดทั่วไป

a	can	has	mightn	same	too
about	couldn	hasn	mightn't	shan	under
above	couldn’t	hasn't	more	shant	until
after	did	have	most	she	up
again	didn	haven	needn	she's	very
against	didn't	haven't	needn't	should	was
ain	do	i	no	should've	weren't
all	does	if	nor	shouldn	what
am	doesn	in	not	shouldn't	when
an	doesn't	into	now	their	where
and	doing	is	o	theirs	which
be	don	isn	of	them	while
because	don't	isn't	off	themselves	who
been	each	it	on	then	whom
before	few	it's	once	there	y
being	for	its	only	these	you
below	from	just	or	they	you'd
between	further	ll	other	this	you'll
both	had	m	our	those	you're
but	hadn	ma	ours	through	you've
by	hadn't	me	re	to	yours

อย่างไรก็ตาม คำหยุดที่ใช้ในการศึกษาโดนเมนหนึ่ง อาจจะไม่ใช่คำหยุดในการศึกษาของอีกโดเมนหนึ่ง ดังนั้นในการที่จะระบุว่า คำใดควรในเป็นคำหยุด จึงควรกำหนดในเหมาะสมกับโดเมนที่กำลังศึกษา

2.5.3 การหารูปแบบพื้นฐานร่วมของคำศัพท์ (Stemming)

การหารูปแบบพื้นฐานร่วมของคำศัพท์ คือรูปแบบเดิมของคำที่ยังไม่ได้เติมคำอุปสรรค (Prefixes) หรือคำปัจจัย (Suffixes) การหารูปแบบพื้นฐานร่วมของคำศัพท์ จึงเป็นการหารูปแบบเดิมของคำ เพื่อปรับให้เป็นคำเดียวกัน [49] ขั้นตอนนี้เป็นกระบวนการที่ควรดำเนินการก่อนนำคำเหล่านั้นมาเพื่อใช้ในการประมวลผล โดยการหารูปแบบพื้นฐานร่วมของคำศัพท์ของคำในภาษาอังกฤษมีขั้นตอนที่เป็นรูปแบบซึ่งสามารถเขียนเป็นขั้นตอนวิธีได้ ดังตัวอย่าง Snowball Stemmer [50] หรืออาจเรียกได้ว่า Porter2 โดยประกอบไปด้วย 5 ขั้นตอน ดังแสดงตัวอย่างในตารางที่ 2.5

ตารางที่ 2.5 ขั้นตอนการประมวลผลของ Snowball Stemmer

Suffix	Replacement	Examples	Result
Step 1a			
sses	ss	caresses	caress
ies	i	poines	poni
ss	ss	carress	carress
s	NULL	cats	cat
Step 1b			
eed	ee	agrred	agree
ed	NULL	plastered	plaster
ing	NULL	motoring	motor
		sing	sing
Step 1b1			
at	ate	conflat(ed)	conflate
bl	ble	troubl(ing)	trouble
iz	ize	siz(ed)	size
NULL	single letter	hopp(ing)	hop
(d an not L or S or Z)		tann(ed)	tan
		fall(ing)	fall
NULL	e	fail(ing)	fail
		fil(ing)	file
Step 1c			
y	i	happy	happi
		sky	sky
Step 2			
ational	ate	relational	relate
tional	tion	conditional	condition

Suffix	Replacement	Examples	Result
enci	ence	valenci	valence
anci	ance	hesitanci	hesitance
izer	ize	digitizer	digitize
abli	able	conformabli	conformable
alli	al	radicalli	radical
enti	ent	diffentli	different
eli	e	vileli	vile
ousli	ous	analogusli	analogous
ization	ize	vietnamization	vietnamize
ation	ate	predication	predicate
ator	ate	operator	operate
alism	al	feudalism	feudal
iveness	ive	decisiveness	decisive
fulness	ful	hopefulness	hopeful
ousness	ous	callousness	callous
aliti	al	formality	formal
iviti	ive	sensitivity	sensitive
biliti	ble	sensibility	sensible
Step 3			
icate	ic	Triplicate	Triplic
ative	NULL	formative	form
alize	al	formalize	formal
iciti	ic	electriciti	electric
ical	ic	electrical	electric
ful	NULL	hopeful	hope
ness	NULL	goodness	good
Step 4			
al	NULL	revival	reviv
ance	NULL	allowance	allow
ence	NULL	inference	infer
er	NULL	airliner	airlin
ic	NULL	gyroscopic	gyroscop
able	NULL	adjustable	adjust
ible	NULL	defensible	defens
ant	NULL	irritant	irrit
ement	NULL	replacement	replac
ment	NULL	adjustment	adjust
ent	NULL	dependent	depend
ion	NULL	adoption	adopt
ou	NULL	homologou	homolog

Suffix	Replacement	Examples	Result
ism	NULL	communism	commun
ate	NULL	activate	activ
iti	NULL	angulariti	angular
ous	NULL	homologous	homolog
ive	NULL	effective	effect
ize	NULL	bowdlerize	bowdler
Step 5a			
e	NULL	probate	probat
Step 5b			
NULL	single letter	control	control
		roll	roll

2.5.4 POS Tagging

POS Tagging หรือ Part of speech tagging คือกระบวนการในการกำหนดขอบเขตและระบุหน้าที่ของคำในประโยค การกำหนดของเขตของคำเพื่อกำกับหน้าที่ของคำโดยอัตโนมัติซึ่งเรียกว่า “Tagging” และตัวกำกับคำจะเรียกว่า “tag” [51]

ไวยากรณ์ในภาษาอังกฤษ สามารถแบ่งหน้าที่ของคำได้ 8 ประเภท คือ คำนาม (Noun) คำกิริยา (Verb) คำสรรพนาม (Pronoun) คำคุณศัพท์ (Adjective) คำสันธาน (Conjunction) คำบุพบท (Preposition) คำวิเศษณ์ (Adverb) และคำอุทาน (Interjection)

ในการระบุหน้าที่ของคำนั้นควรกำหนดมาตรฐานของ tag ที่ใช้ระบุหน้าที่ของคำเป็นอันดับแรก ซึ่งในปัจจุบันมีชุด tag มาตรฐานที่ใช้ระบุหน้าที่ของคำ ซึ่งสำหรับภาษาอังกฤษที่นิยมใช้ได้แก่ Penn treebank, Brown corpus tag-set และ British national corpus ซึ่ง Penn treebank กำหนดไว้ที่ 45 tag ในขณะที่ Brown corpus tag-set กำหนดไว้ที่ 192 tag และ British national corpus กำหนดไว้ที่ 61 tag โดยในที่นี้จะแสดงตัวอย่างของ tag มาตรฐานของ Penn treebank ได้ดังตารางที่ 2.6

ตารางที่ 2.6 Penn treebank tagset และตัวอย่าง

Tag	Description	Example	Tag	Description	Example
CC	coordin. Conjunction	*and, but, or*	SYSM	symbol	*+, %, &*
CD	Carinal number	*one, two, three*	TO	“to”	*to*
DT	Determiner	*a, the*	UH	Injection	*ah, oops*
EX	Existential ‘three’	*there*	VB	verb, base from	*eat*
FW	Foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*widest*	VBZ	verb, 3sg pres	*eats*
LS	list item market	*1, 2, One*	WDT	wh-determiner	*which, that*

Tag	Description	Example	Tag	Description	Example
MD	Modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing, or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	Predeterminer	*all, both*	“	left quote	‘ or “
POS	possessive ending	*’s*	”	right quote	’ or ”
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one’s*	)	right parenthesis	],), }, >
RB	Adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... _ -
RP	particle	*up, off*			

โดยทั่วไปมีเทคนิคการทำ POS Tagging อยู่ 3 วิธี [51] ได้แก่ การจำแนกประเภทของคำด้วยกฎ (Rule-based POS tagging) การจำแนกประเภทของคำด้วยโมเดลมาร์คอฟ (Markov models) และการเรียนรู้บนพื้นฐานการเปลี่ยนแปลง (Transformation-based learning)

ในงานนี้ได้ใช้ประโยชน์จากการทำ POS Tagging ด้วยเทคนิคที่อยู่บนพื้นฐานของความน่าจะเป็น ซึ่งเทคนิคนี้จะสกัดองค์ความรู้ทางภาษาศาสตร์จากคลังข้อมูลขนาดใหญ่และทำ POS Tagging โดยอัตโนมัติ ดำเนินด้วยหลักความน่าจะเป็นของความเป็นไปได้ที่จะเกิด tag ดังนั้น เทคนิค POS Tagging อยู่บนพื้นฐานของหลักความน่าจะเป็นและต้องมีชุดข้อมูลสำหรับเรียนรู้ความน่าจะเป็นที่จะเกิด tag และโมเดลมาร์คอฟมักถูกนำมาใช้ในการดำเนินการในขั้นตอนนี้

วัตถุประสงค์ของโมเดลมาร์คอฟคือการหาลำดับที่เหมาะสมที่สุดของ Tag $T = t_1, t_2, t_3, \ldots, t_n$ ของลำดับคำ $W = w_1, w_2, w_3, \ldots, w_n$ ซึ่งหมายถึงการหาลำดับของ tag ที่มีความน่าจะเป็นที่สุดสำหรับกลุ่มคำเหล่านั้น โดยมีแนวคิดว่าโอกาสที่จะเกิด tag จะขึ้นอยู่กับความน่าจะเป็นของ tag ที่เกิดขึ้นก่อนหน้ารูปแบบนี้เรียกว่า ไบแกรมโมเดล (Bigram model) แต่ละขั้นของไบแกรมโมเดลจะสอดคล้องกับ POS tag ความน่าจะเป็นระหว่าง POS state สามารถแสดงได้ด้วยสมการ $P(t_i, t_j)$ และความน่าจะเป็นของคำที่ถูกดึงออกมาจาก tag สามารถแสดงเป็น $P(w_i, t_j)$

2.5.5 ตัวแจงประโยค (Parser)

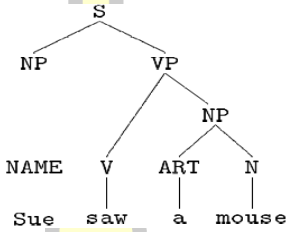
ตัวแจงประโยค หรือที่เรียกว่า พาสเซอร์ ทำหน้าที่รับประโยคอินพุตแล้วใช้ไวยากรณ์และคลังศัพท์เพื่อวิเคราะห์โครงสร้างของประโยค ซึ่งหมายถึงการวิเคราะห์หาความสัมพันธ์ของส่วนประกอบในประโยคโปรแกรมที่ใช้สำหรับแจงประโยคเรียกว่า พาสเซอร์ เอาต์พุตที่ได้จะเป็นข้อมูลที่อยู่ในรูปแบบต้นไม้ที่แจกแจงออกมานั่นคือ ต้นไม้ไวยากรณ์

กระบวนการที่ใช้ในการแจงประโยค เรียกว่า พาสซิ่ง (Parsing) เพื่อใช้ในการหาโครงสร้างข้อมูลของประโยคในภาษาหนึ่งๆ โดยใช้ไวยากรณ์อัลกอริทึมของการพาสซิ่งจึงเป็นกระบวนการที่ค้นหาความเป็นไปได้ต่างๆ ในการนำไวยากรณ์ไปใช้ในการวิเคราะห์ประโยค ซึ่งการ

พาสซิ่งมี 2 รูปแบบคือ การพาสซิ่งจากบนลงล่าง (Top-down parsing) และ การพาสซิ่งจากล่างขึ้นบน (Bottom-up parsing) [52]

1) การพาสซิ่งจากบนลงล่าง

การพาสซิ่งจากบนลงล่าง จะเริ่มประโยค แล้วพิจารณาถึงโครงสร้างของส่วนประกอบในประโยค ซึ่งได้แก่ประเภทของคำต่างๆ ทำให้เกิดการสร้างประโยคขึ้นใหม่ที่ประกอบด้วยลำดับของชนิดคำตามหลักไวยากรณ์ โดยพิจารณาจากซ้ายไปขวา เช่นการแจงประโยค “Sue saw a mouse” ซึ่งสามารถแสดงโครงสร้างไวยากรณ์ดังรูปที่ 2.17 และแสดงการพาสซิ่งจากบนลงล่างดังรูปที่ 2.18



รูปที่ 2.17 Parse Tree ของประโยค

S	
NP VP	(rewrite S)
NAME VP	(rewrite NP)
Sue VP	(rewrite NAME)
Sue V NP	(rewrite VP)
Sue saw NP	(rewrite V)
Sue saw ART N	(rewrite NP)
Sue saw the N	(rewrite ART)
Sue saw the mouse	(rewrite N)

รูปที่ 2.18 ตัวอย่างการพาสซิ่งจากบนลงล่าง

2) การพาสซิ่งจากล่างขึ้นบน

ใช้ตรวจสอบประโยคโดยพิจารณาจากลำดับของชนิดคำที่เรียงกันอยู่ในประโยค เพื่อพิจารณาว่าการเรียงลำดับนั้นถูกต้องหรือไม่ โดยพิจารณาจากขวาไปซ้าย จากประโยคตัวอย่าง “Sue saw the mouse” สามารถแสดงการพาสซิ่งจากล่างขึ้นบนได้ดัง รูปที่ 2.19

Sue saw the mouse	
NAME saw the mouse	(rewrite Sue)
NAME V the mouse	(rewrite saw)
NAME V ART mouse	(rewrite the)
NAME V ART N	(rewrite mouse)
NP V ART N	(rewrite NAME)
NP V NP	(rewrite ART N)
NP VP	(rewrite V NP)
S	(rewrite NP VP)

รูปที่ 2.19 ตัวอย่างการพาสซิ่งจากล่างขึ้นบน

2.5.6 ตัวแบบมาร์คอฟ (Markov model)

ตัวแบบมาร์คอฟ คือ ตัวแบบทางคณิตศาสตร์ที่ใช้ในการวิเคราะห์พฤติกรรมของตัวแปร เพื่อพยากรณ์พฤติกรรมในอนาคตของตัวแปรนั้น ซึ่งวิธีการใช้ตัวแบบมาร์คอฟ ได้รับการพัฒนาโดยนักคณิตศาสตร์ชาวรัสเซียชื่อ อังเดร เอ มาร์คอฟ

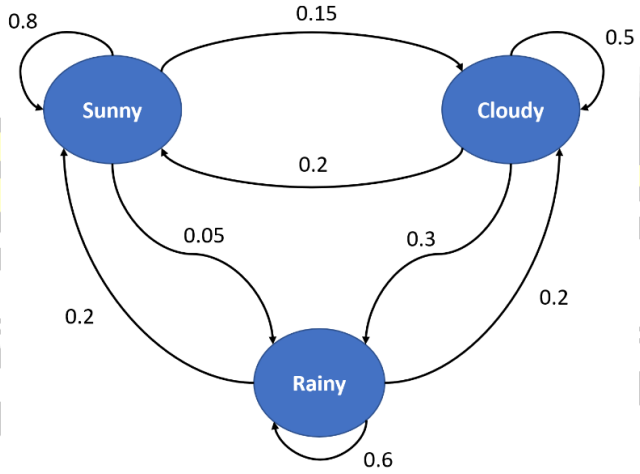
คุณสมบัติสำคัญของปัญหาที่นำตัวแบบมาร์คอฟมาแก้ปัญหา

1) ปัญหานั้นต้องมีผลลัพธ์ที่จะเกิดขึ้นจำนวนที่แน่นอนจำนวนหนึ่ง
2) ค่าความน่าจะเป็นของผลลัพธ์ถัดไป ต้องขึ้นอยู่กับผลลัพธ์ก่อนหน้า
3) ค่าความน่าจะเป็นของการเกิดผลลัพธ์ต่างๆ ต้องมีค่าคงที่เสมอไม่เปลี่ยนแปลงตามเวลาที่เปลี่ยนไป

ห่วงโซ่มาร์คอฟ (Markov chain)

ห่วงโซ่มาร์คอฟ [53, 54] คือ ลำดับของการเกิดเหตุการณ์ ซึ่งค่าความน่าจะเป็นของการเกิดเหตุการณ์ แต่ละเหตุการณ์จะขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้นก่อนหน้านั้น โดยที่เหตุการณ์ (Event) คือ สิ่งที่อาจเกิดขึ้น หรือ การเปลี่ยนแปลงที่เกิดขึ้น และสถานะ (State) คือ สภาพที่เป็นอยู่ในเวลาใดเวลาหนึ่ง ซึ่งสถานะนั้นๆ อาจจะเปลี่ยนแปลงหรือไม่เปลี่ยนแปลงก็ได้ แต่ในระยะยาวสถานะนั้นๆ จะคงที่

ในการนำเสนอความน่าจะเป็นแบบทรานสิชัน สามารถทำได้ 2 วิธี โดยวิธีแรกเรียกว่าการใช้ไดอะแกรมแสดงสถานะ (State diagram) อีกวิธีหนึ่งเรียกว่า ทรานสิชันเมทริกซ์ (Transition matrix) ซึ่งในที่นี้สมมติตัวอย่างเพื่อทำนายสภาพอากาศของวันพรุ่งนี้ โดยใช้ข้อมูลของวันที่ผ่านมา ซึ่งโมเดลมี 3 สถานะ คือ $S = s_1, s_2, s_3$ ซึ่งนั่นคือ s_1 = *Sunny*, s_2 = *Rainy*, s_3 = *Cloudy* โดยจะสามารถสร้างเป็นไดอะแกรมสถานะ และ ทรานสิชันเมทริกซ์ ได้ดัง รูปที่ 2.20 และ ตารางที่ 2.7 ตามลำดับ



รูปที่ 2.20 ตัวอย่างไดอะแกรมสถานะของสภาพอากาศ

ตารางที่ 2.7 ตัวอย่างตารางทรานสิชันเมทริกซ์ของสภาพอากาศ

จากสถานะ	เปลี่ยนไปเป็นสถานะ			รวม
	Sunny	Rainy	Cloudy	
Sunny	0.8	0.05	0.15	1
Rainy	0.2	0.6	0.2	1
Cloudy	0.2	0.3	0.5	1

จากรูปที่ 2.20 และ ตารางที่ 2.7 สามารถอธิบายได้ว่า *P*(*Sunny*|*Rainy*) = 0.2 นั่นคือ ค่าความน่าจะเป็นที่วันก่อนหน้าที่เป็น *Rainy* และวันต่อมาเป็น *Sunny* มีค่าความน่าจะเป็นเท่ากับ 0.2 หรือ *P*(*Rainy*|*Cloudy*) =0.3 คือ ความน่าจะเป็นของวันก่อนหน้าเป็น *Cloudy* และวันต่อมาจะเป็น *Rainy* มีค่าเท่ากับ 0,3 หรือ *P*(*Sunny*|*Sunny*) = 0.8 คือ ความน่าจะเป็นของวันก่อนหน้าเป็น *Sunny* แล้ววันต่อมาจะเป็น *Sunny* มีค่าเท่ากับ 0.8 เป็นต้น

อย่างไรก็ตาม ข้อมูลนี้ใช้พยากรณ์ได้แบบวันต่อวัน ถ้าหากต้องการพยากรณ์ข้อมูลถัดไปอีกสองวัน (วันมะรืน) เมื่อทราบข้อมูลวันนี้ จะไม่สามารถทำไม่ได้โดยตรง นั่นคือ ต้องเก็บข้อมูลเพิ่มและคำนวณหาความน่าจะเป็นใหม่อีก ซึ่งจำนวนข้อมูลที่ต้องเก็บจะเท่ากับ จำนวนสถานะยกกำลังด้วยจำนวนวัน ซึ่งในตัวอย่างนี้มีสามสถานะ และสนใจสามวัน จะต้องเก็บข้อมูลอย่างต่ำ 3^3 ข้อมูลเพื่อมาคำนวณหาความน่าจะเป็น ดังนั้นเพื่อที่จะทำให้ง่ายขึ้น สามารถใช้การประมาณมาทดแทน โดยกำหนดว่าถ้าต้องการพยากรณ์ข้อมูลวันนี้ ให้ใช้ข้อมูลของวันก่อนหน้านี้เท่านั้น นั่นคือ

$$P(w_n|w_1, w_2, w_3, \ldots, w_n) \approx P(w_n|w_{n-1}) \tag{2.1}$$

การประมาณนี้มีชื่อเรียกว่า การประมาณมาร์คอฟ (Markov assumption) ซึ่งจะทำให้ลดการเก็บข้อมูลจาก สถานะยกกำลัง *n* เหลือแค่สถานะยกกำลัง 2 เพราะสนใจข้อมูลเพียงแค่สองเวลา

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_n|w_{n-1}) \tag{2.2}$$

ด้วยเหตุนี้เราสามารถหาค่าความน่าจะเป็นของสภาพอากาศวันมะรืนได้ ยกตัวอย่างเช่น ถ้าวันนี้เป็น *Sunny* แล้วโอกาสที่พรุ่งนี้จะเป็น *Sunny* และวันมะรืนเป็น *Rainy* จะเป็นเท่าใด สามารถคำนวณได้โดย ให้ w_1 = *Sunny*, w_2 = *Sunny* และ w_3 = *Rainy*

$$\begin{aligned} P(w_3|w_1, w_2) &= P(w_3|w_1, w_2) * (Pw_2|w_1) \\ &= P(w_3|w_2) * (Pw_2|w_1) \\ &= P(Rainy|Sunny) * P(Sunny|Sunny) \\ &= 0.05 * 0.8 \ = \ 0.04 \end{aligned}$$

นั่นคือ ถ้าวันนี้เป็น *Sunny* โอกาสที่พรุ่งนี้จะเป็น *Sunny* และ วันมะรืนเป็น *Rainy* มีค่าความน่าจะเป็นเท่ากับ 0.04

ค่าความน่าจะเป็นแบบทรานเซียนท์ (Transient probability)

ค่าความน่าจะเป็นแบบทรานเซียนท์ คือ ค่าความน่าจะเป็นของการอยู่ในสถานะใดสถานะหนึ่งของห่วงโซ่มาร์คอฟก่อนที่จะเข้าสู่สถานะคงตัว ซึ่งสามารถคำนวณได้จากการนำทรานสิชันเมทริกซ์ คูณกับค่าความน่าจะเป็น ณ ปัจจุบัน ดังตัวอย่างต่อไปนี้

ตารางที่ 2.8 ตัวอย่างตารางทรานสิชันเมทริกซ์เพื่อหาค่าความน่าจะเป็นแบบทรานเซียนท์

จากสถานะ	เปลี่ยนไปเป็นสถานะ			รวม
	S1	S2	S3	
S1	0.4	0.3	0.3	1
S2	0.1	0.8	0.1	1
S3	0.1	0.3	0.6	1

โดยมีค่าความน่าจะเป็นในรอบปัจจุบัน คือ $P(S1) = 0.4$, $P(S2) = 0.3$, $P(S3) = 0.3$ ซึ่งสามารถคำนวณได้โดยวิธีการหาผลคูณของเมทริกซ์ได้ดังนี้ ทำเมทริกซ์สลับเปลี่ยน (Transpose metrix) นั่นคือเปลี่ยนแนวแถวเป็นคอลัมน์ และเปลี่ยนคอลัมน์เป็นแถว เมื่อทำเมตริกซ์สลับเปลี่ยนแล้วให้นำค่าความน่าจะเป็นรอบปัจจุบันคูณเมทริกซ์สลับเปลี่ยน ดังตัวอย่างต่อไปนี้

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.3 & 0.6 \end{bmatrix}$$ เมทริกซ์สลับเปลี่ยนจะได้ $$A' = \begin{bmatrix} 0.4 & 0.1 & 0.1 \\ 0.3 & 0.8 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix}$$

นำเมทริกซ์ที่ได้ไปหาผลคูณของเมทริกซ์กับเมทริกซ์ความน่าจะเป็นในรอบเวลาปัจจุบัน โดยผลลัพธ์ที่ได้จะเป็นความน่าจะเป็นของครั้งต่อไปดังนี้

$$\begin{bmatrix} 0.4 & 0.1 & 0.1 \\ 0.3 & 0.8 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.4 \\ 0.3 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.22 \\ 0.45 \\ 0.33 \end{bmatrix}$$

จากตัวอย่างจะได้ผลลัพธ์เป็นความน่าจะเป็นในครั้งต่อไปของ $S1 = 0.22$, $S2 = 0.45$ และ $S3 = 0.33$ และหากต้องการหาความน่าจะเป็นในครั้งต่อไปอีก ก็สามารถทำได้โดยนำผลลัพธ์ในรอบที่ผ่านมาคูณกับเมทริกซ์สลับเปลี่ยนของทรานสิชันเมทริกซ์อีกครั้ง ดังตัวอย่าง

$$\begin{bmatrix} 0.4 & 0.1 & 0.1 \\ 0.3 & 0.8 & 0.3 \\ 0.3 & 0.1 & 0.6 \end{bmatrix} \times \begin{bmatrix} 0.22 \\ 0.45 \\ 0.33 \end{bmatrix} = \begin{bmatrix} 0.166 \\ 0.525 \\ 0.309 \end{bmatrix}$$

จากตัวอย่างจะได้ผลลัพธ์เป็นความน่าจะเป็นในครั้งต่อไปของ $S1 = 0.166$, $S2 = 0.525$ และ $S3 = 0.309$ และหากต้องการหาในรอบต่อไปอีกให้ทำเช่นเดิมนี้ต่อไปเรื่อยๆ จนเข้าสู่สภานะคงตัว นั่นคือผลลัพธ์ความน่าจะเป็นในรอบต่อไปนั้นไม่มีการเปลี่ยนแปลงค่าแล้ว

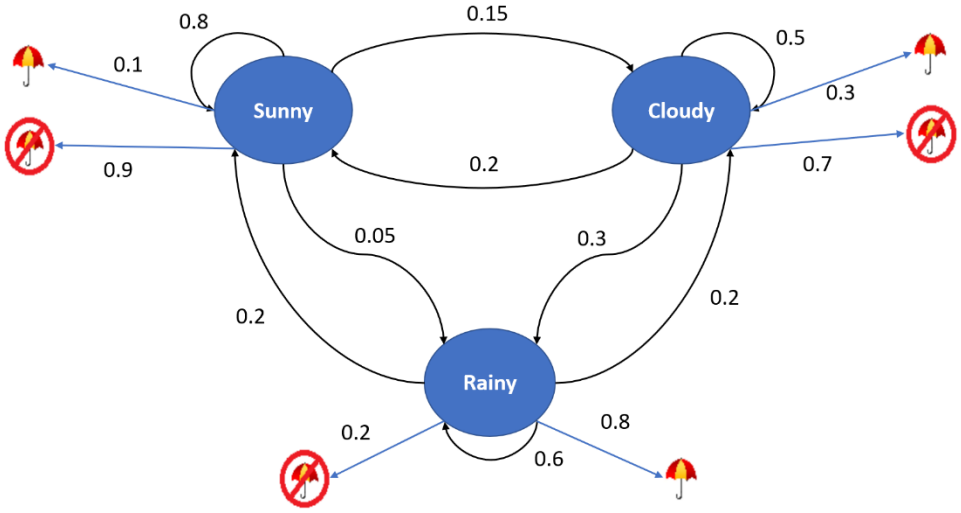
ความน่าจะเป็นแบบสเตดีเสตท (Steady-state probability)

ความน่าจะเป็นแบบสเตดีเสตท ความน่าจะเป็นของการอยู่ในสถานะใดสถานะหนึ่งของห่วงโซ่มาร์คอฟซึ่งมีค่าคงตัว (ค่าความน่าจะเป็นในระยะยาว)

2.5.7 ตัวแบบมาร์คอฟซ่อนเร้น (Hidden markov model)

ตัวแบบมาร์คอฟซ่อนเร้น [53, 54] คือ แบบจำลองมาร์คอฟที่ไม่ทราบว่าลำดับของสถานะคืออะไร แต่รู้เพียงฟังก์ชันความน่าจะเป็นบางส่วนของแบบจำลองนั้น กล่าวคือ ไม่ทราบเหตุการณ์ก่อนหน้าอย่างชัดเจน หรือการพยากรณ์ต้องใช้ปัจจัยอื่นมาเทียบเคียงด้วย

จากรูปตัวอย่างไดอะแกรมสถานะสภาพอากาศใน รูปที่ 2.20 ซึ่งเป็นแผนภาพที่เป็นมาร์คอฟโมเดล หากเป็นตัวแบบมาร์คอฟซ่อนเร้นจะแสดงได้ดังรูปที่ 2.21



รูปที่ 2.21 ตัวอย่างไดอะแกรมสถานะของสภาพอากาศแบบมาร์คอฟซ่อนเร้น

จากรูปที่ 2.21 เป็นแผนภาพไดอะแกรมสถานะของสภาพอากาศที่เป็นแบบมาร์คอฟซ่อนเร้น เนื่องด้วยมีผลลัพธ์ของสถานะสังเกต (Observations) ดังตัวอย่างเช่น *P*(*Umbrella*|*Sunny*) = 0.1 ซึ่งหมายความว่าในวันที่เป็น *Sunny* มีความน่าจะเป็นที่ *Umbrella* (ในที่นี้หมายความว่าคนจะพกร่ม) มีค่าเท่ากับ 0.1 หรือ *P*(*Umbrella*|*Rainy*) = 0.8 นั่นหมายความว่าในวันที่เป็น *Rainy* มีความน่าจะเป็นที่คนจะพกร่มเป็น 0.8 เป็นต้น โดยในที่นี้จะเรียกความน่าจะเป็นในสถานะสังเกตนี้ว่า Emission probabilities

จากตัวอย่างในหัวข้อตัวแบบมาร์คอฟ หากต้องการพยากรณ์สภาพอากาศที่ยังไม่เกิดขึ้นจำเป็นที่จะต้องมีข้อมูลสภาพอากาศในอดีตก่อน แต่หากไม่ทราบข้อมูลในอดีต หรือข้อมูลดังกล่าวถูกปกปิดไว้ (Hidden) จะพยากรณ์ได้อย่างไร ซึ่งตัวแบบมาร์คอฟซ่อนเร้นนั้นสามารถใช้เหตุการณ์ที่เกี่ยวเนื่องกันเพื่อเทียบเคียงเพื่อใช้ในการพยากรณ์ได้โดยที่ไม่รู้เหตุการณ์ในอดีตว่าเป็นอย่างไรได้ ดังตัวอย่างต่อไปนี้ สมมติว่าไม่สามารถทราบสภาพอากาศจริงภายนอกได้ซึ่งอาจจะเป็นเพราะถูกขังในห้อง แต่พอจะทราบข้อมูลความสัมพันธ์ของสภาพอากาศกับการใช้ร่มของบุคคลที่มาเยี่ยมในทุกๆวัน เช่น ถ้าสภาพอากาศเป็น *Sunny Rainy* และ *Cloudy* โอกาสที่บุคคลที่มาเยี่ยมจะพกร่มเป็น 0.1, 0.8 และ 0.3 ตามลำดับ และมีโอกาสที่บุคคลที่มาเยี่ยมจะพกร่มโดยไม่สนใจสภาพอากาศ คือ 0.5 หากวันที่ถูกขังมีสภาพอากาศเป็น *Sunny* วันต่อมามีบุคคลมาเยี่ยมและพกร่มมาด้วย โอกาสที่ฝนจะตกวันนี้เป็นเท่าใด

ดังนั้น สิ่งที่เราต้องการหาคือ $P(w_2=Rainy|\ w_1=Sunny,\ u_2=True)$

$$P(w_2|w_1,u_2) = \frac{P(w_2,w_1|u_2)}{P(w_1|u_2)}$$

แต่เนื่องจากเหตุการณ์ w_1 กับ w_2 ไม่ขึ้นต่อกัน (คนละวัน) จึงได้สมการเป็น

$$u_2 \; and \; w_1 \; independent = \frac{P(w_2,w_1|u_2)}{P(w_1)}$$

แต่สมการนี้ไม่สามารถหาค่าได้เนื่องจากข้อมูลที่ทราบคือ $P(u|w)$ ดังนั้นต้องใช้ทฤษฎีบทของเบย์ (Bayes'Theorem) ที่เปลี่ยนลำดับของความน่าจะเป็นแบบมีเงื่อนไข คือ $P(A|B) = P(B|A)P(A)/P(B)$ ซึ่งจะได้เป็น

$$Bayes'Rule = \frac{P(u_2|w_1,w_2)P(w_2,w_1)}{P(w_1)P(u_2)}$$

จากการประมาณมาร์คอฟ สิ่งที่สนใจคือข้อมูลสภาพอากาศวันล่าสุดเท่านั้น $P(u_2|w_1,w_2)$ จึงมีค่าเป็น $P(u_2|w_2)$ และจะได้เป็น

$$(Markov \; assumption) = \frac{P(u_2|w_2)P(w_2,w_1)}{P(w_1)P(u_2)}$$

จัดรูปความน่าจะเป็นใหม่เนื่องจากมีค่าที่ไม่รู้คือ $P(w_2,w_1)$ แต่รู้ค่า $P(w_2,w_1)/P(w_1)$ มีค่าเท่ากับ $P(w_2|w_1)$ จึงได้เป็น

$$(P(A,B) = P(A|B)P(B)) = \frac{P(u_2|w_2)P(w_2|w_1)P(w_1)}{P(w_1)P(u_2)}$$

เอา $P(w_1)$ ออกเนื่องจากไม่ทราบค่า

$$(Cancle\text{: } P(Sunny)) = \frac{P(u_2|w_2)P(w_2|w_1)}{P(u_2)}$$

ซึ่งเมื่อกลับมาเขียนแทนค่าในรูปแบบ และแทนค่าในสมการความน่าจะเป็นจะได้

$$= \frac{P(u_2 = True|Rainy)P(Rainy|Sunny)}{P(u_2 = True)}$$

$$= \frac{(0.8)(0.05)}{(0.5)}$$

$$= 0.08$$

เพราะฉะนั้นความน่าจะเป็นที่สภาพอากาศเป็น *Rainy* เมื่อวันที่บุคคลมาเยี่ยมพกร่มและเมื่อวานสภาพอากาศเป็น *Sunny* คือ 0.08

ส่วนประกอบของตัวแบบมาร์คอฟซ่อนเร้น

ส่วนประกอบต่างๆ ที่ตัวแบบมาร์คอฟซ่อนเร้นจะต้องมีคือ

S คือ เซตของสถานะ : $\{s_1, s_2, \ldots, s_n\}$

K คือ ผลลพธ์ที่เป็นไปได้ทั้งหมด : $\{k_1, k_2, \ldots, k_m\}$

Π คือ initial state probabilities : $\{\Pi_i\}, i \epsilon S$

A คือ state transition probabilities : $\{a_{ij}\}, i, j \epsilon S$

B คือ symbol emission probabilities : $\{b_{ijk}\}, i, j \epsilon S, k \epsilon K$

ลำดับสถานะ (state sequence) แทนโดย $X = (X_1, \ldots, X_{T+1}), X{:}\, S \longrightarrow \{1, \ldots, N\}$ และลำดับของการสังเกต (output sequence) แทนโดย $O = (o_1, \ldots, o_T), o_1 \epsilon K$

ปัญหาขั้นพื้นฐานในตัวแบบมาร์คอฟซ่อนเร้น

ปัญหาที่ 1 : ถ้าแบบจำลอง $\lambda = (A, B, \Pi)$ จะคำนวณหาค่าความน่าจะเป็นของผลลัพธ์ที่กำหนดได้อย่างไร

ปัญหาที่ 2 : ถ้าให้ลำดับของการสังเกต *O* และแบบจำลอง λ จะหาลำดับสถานะที่ดีที่สุดในการอธิบายลำดับของการสังเกตได้อย่างไร

ปัญหาที่ 3 : ถ้าให้ลำดับของการสังเกต *O* และแบบจำลองต่างๆ ที่เกิดจากการปรับตัวแปรต่างๆ $\lambda = (A, B, \Pi)$ จะหาแบบจำลองที่ดีที่สุดในการอธิบายลำดับของการสังเกตได้อย่างไร

2.5.8 เอ็นแกรม (N-gram)

เอ็นแกรม [44, 51, 55] คือ แบบจำลองที่ใช้คำนวณค่าความน่าจะเป็นของชุดอักขระ (Character sequence) ที่เกิดขึ้นร่วมกันเป็นคำ หรือค่าความน่าจะเป็นของคำที่เขียนเรียงกัน (Word sequence) ที่เกิดขึ้นร่วมกันเป็นประโยค โดยค่าความน่าจะเป็นของชุดอักขระหรือคำประมาณได้จากคลังข้อมูลที่สร้างไว้ ซึ่งเอ็นแกรมได้ใช้หลักการของสถิติในหลายๆ ด้านมาประยุกต์ใช้

แกรม (gram) คือ หน่วยที่ใช้ในการสร้างแบบจำลอง อาจจะเป็นเสียง คำ หรือ อักขระก็ได้และแกรมมีได้หลายขนาดแล้วแต่จะกำหนด ตั้งแต่ 1 จนถึง *N* ในแบบจำลองเอ็นแกรมนี้ใช้ความยาวของชุดอักขระและคำที่เขียนเรียงกันแตกต่างกัน ได้แก่ 2-gram , 3-gram , 4-gram ฯลฯ

วิธีการทั่วไปของเอ็นแกรม มักนิยมใช้การประมาณมาร์คอฟเพื่อประมาณค่าโครงสร้างทางสถิติของข้อความที่ว่า การปรากฏตัวของอักษรตัวหนึ่งขึ้นกับตัวอักษรก่อนหน้าเพียง *n-1* ตัว จึงเรียกแบบจำลองลักษณะนี้ว่าแบบจำลองเอ็นแกรม (N-gram model) โดยสามารถประมาณได้ดังนี้

ไบแกรม $P(c_1 c_2 c_3 \ldots c_n) = P(c_1)P(c_2|c_1)P(c_3|c_2)\ldots P(c_n|c_{n-1})$

ไตรแกรม $P(c_1 c_2 c_3 \ldots c_n) = P(c_1)P(c_2|c_1)P(c_3|c_1 c_2)\ldots P(c_n|c_{n-2} c_{n-1})$

ควอดิแกรม $P(c_1 c_2 c_3 \ldots c_n) = P(c_1)P(c_2|c_1)P(c_3|c_1 c_2)P(c_4|c_1 c_2 c_3)\ldots$
$P(c_n|c_{n-3} c_{n-2} c_{n-1})$

โดยที่ *P* แทน ค่าความน่าจะเป็น, *c* แทน อักขระหรือตัวอักษร และ $(c_1 c_2 c_3 \ldots c_n)$ แทนสายอักขระที่ประกอบด้วยอักขระตั้งแต่ 3 ขึ้นไปจนถึง *n* ตัว

ส่วนความน่าจะเป็นของคำที่รวมกันเป็นประโยค $w_1 w_2 w_3 \ldots w_n$ หากประมาณค่าด้วยเอ็นแกรมต่างๆ ผลที่ได้ดังนี้ โดย *P* แทน ค่าความน่าเป็น *w* แทน คำ *n* แทน จำนวนนับต่อไป และ $(w_1 w_2 w_3 \ldots w_n)$แทนสายคำที่ประกอบด้วยคำมากกว่า 3 คำขึ้นไป

ความน่าจะเป็นของประโยคโดยใช้วิธี ไบแกรม คือ

$$P(w_1 w_2 w_3 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2)\dots P(w_n|w_{n-1})$$

ความน่าจะเป็นของประโยคโดยใช้วิธี ไตรแกรม คือ

$$P(w_1 w_2 w_3 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\dots P(w_n|w_{n-2} w_{n-1})$$

ความน่าจะเป็นของประโยคโดยใช้วิธี ควอดิแกรม คือ

$$P(w_1 w_2 w_3 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)P(w_4|w_1 w_2 w_3) \dots P(w_n|w_{n-3} w_{n-2} w_{n-1})$$

ยกตัวอย่างความน่าจะเป็นของประโยค “The boy drinks water” ซึ่งสามารถใช้สมการความน่าจะเป็นพื้นฐานดังสมการ (2.3)

$$P(w_1 w_2 K w_T) = \prod_{i=1}^{T} P(w_1|w_1 K w_{i-1}) \qquad (2.3)$$

โดยที่ *P* คือ ค่าความน่าจะเป็น ซึ่งประมาณได้จากคลังข้อมูล *T* คือ จำนวนของคำ *i* คือ ลำดับของคำโดยเริ่มต้นที่ลำดับที่ 1 และ $P(w_i|w_1 \dots w_{i-1})$ คือค่าความน่าจะเป็นของคำ w_i หลังจากเกิดคำ $w_1 w_2 \dots w_{i-1}$ ก่อนหน้านี้แล้ว

ความน่าจะเป็นของประโยคนี้ $P(w_1 w_2 \dots w_T)$ สามารถประมาณได้ โดยถือว่าการปรากฏของคำ w_i ขึ้นอยู่กับจำนวนคำข้างหน้า *n-1* ตัวเท่านั้น หรือขึ้นอยู่กับขนาดของเอ็นแกรม ดังนั้นถ้าหากประมาณค่าความน่าจะเป็นของประโยคนี้ โดยใช้ไบแกรม จะปรับเปลี่ยนสมการได้ดังนี้

$$P(w_1 w_2 K w_T) = P(w_i| < s >)P(w_2|w_1)KP(w_i|w_{i-1}) \qquad (2.4)$$

$P(w_i| < s >)$ คือ ความน่าจะเป็นของคำที่แรกเมื่อเกิดเป็นคำแรกของประโยค ซึ่งคำก่อนหน้าจึงเป็นช่องว่าง

$P(w_2|w_1)$ คือ ความน่าจะเป็นของคำ w_2 หลังจากเกิดคำ w_1

$P(w_i|w_{i-1})$ คือ ความน่าจะเป็นของคำ w_i หลังจากเกิดคำ w_{i-1}

ดังนั้นจากสูตรประมาณค่าความน่าจะเป็นด้วย ไบแกรม หากต้องการหาความน่าจะเป็นของประโยค “The boy drinks water” โดยใช้เอ็นแกรมในระดับของคำ ผลที่ได้คือ

$$P(The| < s >)P(boy|The)P(drinks|boy)P(water|drinks)$$

ส่วนค่าความน่าจะเป็นจะหาได้จากคลังข้อมูล เช่น

$$P(boy|The) = \frac{count(The - boy)}{count(The)}$$

จากสมการนี้ หมายความว่า ค่าความน่าจะเป็นของ boy เมื่อเกิดร่วมกับคำว่า The คำนวณได้จากนำจำนวนนับของ The ที่เกิดร่วมกับคำว่า boy หารด้วยจำนวนนับของการเกิด The เดี่ยวๆ

2.6 โมเดลเชิงพื้นที่แบบเวกเตอร์ (Vector space model: VSM)

โมเดลเชิงพื้นที่แบบเวกเตอร์ เป็นรูปแบบของการแทนข้อความ (Text representation) เป็นหนึ่งในการทำงานขั้นพื้นฐานของการทำเหมืองข้อความ (Text mining) และ การค้นคืนข้อมูล (Information retrieval) มีวัตถุประสงค์เพื่อให้ตัวเลขแทนเอกสารข้อความที่ไม่มีโครงสร้าง ทั้งนี้เพื่อให้สามารถดำเนินการคำนวณทางคณิตศาสตร์ได้ ในเอกสารชุดหนึ่งที่ระบุ $D = \{d_i, i = 1,2,\dots n\}$ ซึ่งแต่ละ d_i ย่อมาจากเอกสาร ปัญหาของการแทนข้อความคือการแทนแต่ละ d_i ของ D เป็นจุด พื้นที่เชิงตัวเลข S ซึ่งมีระยะทาง (Distance) / ความคล้ายคลึงกัน (Similarity) อยู่ระหว่างคู่ของจุดในพื้นที่ S แต่ละจุด [56]

สำหรับโมเดลเชิงพื้นที่แบบเวกเตอร์เป็นเทคนิคที่คิดค้นโดย Salton [57] ซึ่งเป็นวิธีที่ใช้หาความคล้ายคลึงกันระหว่างเอกสารและข้อคำถาม เอกสารต่างๆ จะถูกอธิบายโดยใช้คุณลักษณะ เช่น คำหลักที่ปรากฏในเอกสารนั้นๆ $d_i = (a_{i1}, a_{i2}, a_{i3}\dots a_{ij}\dots a_{im})$

โดยที่ d_1 คือหมายเลขเอกสาร a_{ij} คือคุณลักษณะที่ใช้อธิบายเอกสาร ซึ่งค่าของมันแสดงถึงความสำคัญของคุณลักษณะนั้นๆ ต่อเอกสาร d_1 และ n คือจำนวนคุณลักษณะทั้งหมด หรือเรียกว่ามิติพื้นที่แบบเวกเตอร์ (Dimensionality of the vector space) ซึ่งสามารถจำลองความสัมพันธ์ระหว่าง คำหลักและเอกสาร โดยใช้แบบจำลองพื้นที่เวกเตอร์ โดยสามารถแสดงในรูปแบบของตารางเรียกว่า เมทริกซ์คำ-เอกสาร โดยที่แถวคือ คำ (Term) และคอลัมน์คือหมายเลขเอกสาร (Document ID) แสดงได้ดังรูปที่ 2.22

$$D = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & a_{ij} & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

รูปที่ 2.22 ตัวอย่างเมทริกซ์คำ-เอกสาร (Term-document matrix)

2.7 การให้น้ำหนักคำ (Term weighting)

2.7.1 การแทนค่าด้วยค่าการเกิดขึ้นหรือไม่เกิดขึ้นของคำ (Boolean weighting)

การแทนค่าด้วยค่าการเกิดขึ้นหรือไม่เกิดขึ้นของคำ เป็นวิธีการแทนข้อความด้วยคำที่รวบรวมไว้ในถุงคำ(Bag of word) หากมีคำในถุงคำเกิดขึ้นในข้อความที่นำมาวิเคราะห์ จะแทนค่าด้วย 1 แต่หากในข้อความไม่มีคำที่กำหนดไว้เกิดขึ้น จะแทนค่าด้วย 0 ดังสมการ (2.5)

$$B_{t,d} \begin{cases} 1, for\ term\ present\ in\ document \\ 0, othewise \end{cases} \quad (2.5)$$

โดยที่ $B_{t,d}$ คือ ค่าการเกิดคุณลักษณะ t เอกสาร d

2.7.2 การให้น้ำหนักคำตามความถี่ (Term frequency: *tf*)

การให้น้ำหนักคำตามความถี่ แบบนี้จะขึ้นอยู่กับความถี่ของการปรากฏของคำนั้นๆ นั่นคือการให้น้ำหนักคำเท่ากับความถี่ของการปรากฏของคำนั้นๆ ซึ่งเรียกว่า “ความถี่ของคำ” ซึ่งใช้สัญลักษณ์ $tf_{t,d}$ เทคนิคนี้ให้ความสำคัญของความถี่คำเป็นหลัก แต่ตำแหน่งหรือลำดับของคำที่ปรากฏไม่ได้นำมาพิจารณาในการให้น้ำหนักคำ ซึ่งคำนวณได้จากสมการ (2.6)

$$tf_{t,d} = freq(t,d) \tag{2.6}$$

โดยที่ $freq(t,d)$ คือ ค่าความถี่ของการเกิดคุณลักษณะ t ในเอกสาร d

แต่เนื่องจากแต่ละเอกสารอาจมีความยาวที่แตกต่างกัน ดังนั้นการคำนวณน้ำหนักคำ จึงมักจะหารด้วยความยาวของเอกสาร ซึ่งในที่นี้คือจำนวนคำทั้งหมดในเอกสารนั้น เพื่อเป็นการนอร์มัลไลซ์ ดังสมการ (2.7)

$$tf_{t,d} = \frac{freq(t,d)}{\sum_{i \in d} freq(t,d)} \tag{2.7}$$

และการให้น้ำหนักคำด้วยการนอร์มัลไลซ์ด้วยค่า *log* ดังสมการ (2.8)

$$tf_{t,d} = log(1 + freq(t,d)) \tag{2.8}$$

2.7.3 การให้น้ำหนักคำแบบความถี่เอกสารผกผัน (Inverse document frequency: *idf*)

เนื่องจากวิธีการให้น้ำหนักคำตามความถี่ จะพิจารณาเพียงความถี่คำเป็นหลัก วิธีการนี้จึงปรับปรุงด้วยการนำความถี่ของเอกสาร (Document frequency: *df*) มาพิจารณาด้วย [58] ความถี่ของเอกสารหมายถึง จำนวนเอกสารทั้งหมดในคลังเอกสาร (Collection) ที่คำ t ปรากฏ วิธีการนี้จะทำให้เกิดความยุติธรรมสำหรับคำที่มีความถี่สูงในเอกสารใดเอกสารหนึ่งเท่านั้น แต่ปรากฏไม่มากในเอกสารอื่นๆ ซึ่งคำนวณได้จากสมการ (2.9)

$$idf_t = log \frac{N}{df_t} \tag{2.9}$$

โดยที่ N คือ จำนวนเอกสารทั้งหมดในคอเลคชั่น และ df_t คือ จำนวนเอกสารทั้งหมดที่คุณลักษณะ t ปรากฏ

2.7.4 การให้น้ำหนักคำแบบความถี่และความถี่เอกสารผกผัน (Term frequency – Inverse document frequency: *tf-idf*)

การให้น้ำหนักคำแบบความถี่และความถี่เอกสารผกผัน เป็นวิธีการให้ค่าน้ำหนักที่รวมวิธีการของ *tf* และ *idf* เข้าด้วยกันในการคำนวณ [59] และเป็นวิธีการที่ได้รับความนิยมเป็นอย่างมาก

[60] ซึ่งหมายถึงคุณลักษณะที่ใช้เป็นตัวแทนของเอกสาร ควรจะปรากฏอยู่เป็นจำนวนมากในเนื้อหาของเอกสารเฉพาะฉบับนั้นและปรากฏน้อยในชุดของเอกสารที่เหลือทั้งหมด โดยแสดงได้ดังสมการ (2.10)

$$tf - idf_{t,d} = tf_{t.d} \times idf_t \tag{2.10}$$

2.7.5 การให้น้ำหนักคำโดยใช้แทคนิค BM25

BM25 (BM ย่อมาจาก Best Matching) หรือบางครั้งเรียกว่า Okapi BM25 เป็นฟังก์ชันการจัดอันดับเอกสารที่พัฒนาขึ้นในระบบค้นคืนข้อมูล Okapi ของมหาวิทยาลัย London's City ในปี 1980 ถึง 1990 [61-63] นอกจากนี้ยังเป็นเทคนิคสำหรับการกำหนดน้ำหนักคำ ซึ่งจะคำนวณความสัมพันธ์ภายในระหว่างคำค้นหาและเอกสาร แทนที่จะเกี่ยวข้องกับความสัมพันธ์ภายนอกระหว่างคำค้นและเอกสาร ฟังก์ชันการให้น้ำหนักของ *BM25* สำหรับข้อความค้นหา q และเอกสาร d สามารถกำหนดได้ดังสมการ (2.11)

$$sim(d,q) = \sum_{i=1}^{q} idf(q_i) \times \left(\frac{tf(q_i,d) \times (k_1+1)}{tf(q_i,d) + k_1\left(1 - b + b\left(\frac{|d|}{dl_{avg}}\right)\right)} \right) \tag{2.11}$$

โดยที่ $tf(q_i, d)$ เป็นความถี่คำที่กำหนดให้เป็นจำนวนครั้งที่ข้อความค้นหา *i-th* (q_i) ปรากฏในเอกสาร d ส่วน $|d|$ หมายถึงความยาวของเอกสาร d ในขณะที่ dl_{avg} คือความยาวเฉลี่ยของเอกสารสำหรับเอกสารทั้งหมดในคลังเอกสาร พารามิเตอร์ k_1 และ b เป็นพารามิเตอร์อิสระสำหรับการควบคุมการถ่วงน้ำหนักระหว่างความถี่ของคำและความยาวของเอกสารที่มีการทำนอร์มัลไลซ์ โดยทั่วไปแล้วค่าของพารามิเตอร์ทั้งสองตัวจะได้รับการกำหนดให้เป็น $1.2 < k_1 < 2.0$ และ $0.5 < b < 0.8$ ซึ่งจากการศึกษางานวิจัยอื่นๆ ในการทดลองจะพบว่าการตั้งค่าที่พบมากที่สุดของ k_1 และ b จะมีค่าเป็น *2.0* และ *0.8* ตามลำดับ [35, 64]

สำหรับ $idf(q_i)$ หมายถึงความถี่ของเอกสารผกผันของคำค้น q_i สามารถคำนวณได้ด้วยสมการ (2.12)

$$idf(q_i) = log\left(\frac{N - df(q_i) + 0.5}{df(q_i) + 0.5}\right) \tag{2.12}$$

โดยที่ N คือจำนวนเอกสารทั้งหมดในคลังเอกสาร และ $df(q_i)$ คือจำนวนเอกสารที่มีคำค้นหา q_i

2.7.6 การให้น้ำหนักคำโดยใช้เทคนิค Multi Aspect TF (*MATF*)

การให้น้ำหนักคำโดยใช้เทคนิค *MATF* นำเสนอโดย Jiaul H. Paik [65] ซึ่งได้เสนอวิธีการที่รวมสองปัจจัยที่มีผลในการค้นคืนเอกสารในการใช้คำค้นที่สั้นและยาวได้อย่างสมดุล โดย

ปัจจัยแรกพิจารณาความสำคัญของคำจากความถี่ที่สัมพันธ์กับค่าเฉลี่ยของความถี่ของคำในเอกสาร ซึ่งเรียกว่า Relative intra-document: *RITF* โดยสูตรดังสมการ (2.13)

$$RITF(t,d) = \frac{TF(t,d)}{Avg.TF(d)} \tag{2.13}$$

โดยที่ *t* คือ คำ และ *d* คือเอกสาร *TF(t, d)* หมายถึงความถี่ของคำ *t* ในเอกสาร *d* ในขณะที่ *Avg.TF(d)* หมายถึงค่าเฉลี่ยของความถี่ของคำในเอกสาร *d* ซึ่งจากสมการ (2.13) นี้ อาจต้องการเอกสารที่ยาวมาก เนื่องจากส่วนที่ใกล้เคียงกับ 1 เป็นเอกสารที่ยาวมากๆ จึงมีการลดทอนค่า *TF* จึงได้สมการต่อไปนี้

$$RITF(t,d) = \frac{log_2(1 + TF(t,d))}{log_2(1 + Avg.TF(d))} \tag{2.14}$$

ซึ่งสมการ (2.14) ได้ถูกใช้โดย Singhal และคณะ [66] ในการทำนอร์มัลไลซ์เพื่อปรับค่า *TF* ให้สอดคล้องกับจำนวนคำที่ไม่ซ้ำกันในเอกสาร

ปัจจัยที่สอง Length regularized TF: *LRTF* จะนอร์มัลไลซ์ความถี่ของคำโดยพิจารณาจำนวนของคำที่มีอยู่ในเอกสาร ซึ่งมีแนวคิดเดียวกันกับ Robertson [63] ที่ว่าความยาวที่เหมาะสมของเอกสารควรเป็นค่าเฉลี่ยของคลังเอกสารและความถี่ของคำและค่าเฉลี่ยความยาวเอกสารยังคงเดิม ซึ่งจะได้สมการ (2.15)

$$LRTF(t,d) = TF(t,d) \times \frac{ADL(C)}{len(d)} \tag{2.15}$$

โดยที่ *ADL(C)* คือค่าเฉลี่ยของความยาวเอกสารในคลังเอกสาร และ *len(d)* คือความยาวเอกสาร *d* แต่ดูเหมือนว่าการเพิ่มขึ้นของความถี่ในระยะยาวจะไม่มีผลต่อความสัมพันธ์เชิงเส้นกับความยาวของเอกสารดังนั้นสมการนี้ จึงให้ผลด้านลบกับเอกสารยาว เพื่อที่จะไม่ให้เกิดความลำเอียงนี้ จึงมีการปรับสมการ (2.15) เพื่อให้ได้ค่าที่ขึ้นอยู่กับความยาวเอกสารมากขึ้น จึงได้สมการ (2.16) ซึ่งสมการนี้ยังคงมีผลกระทบด้านลบกับเอกสารที่ยาว แต่มีผลกระทบนั้นลดลง

$$LRTF(t,d) = TF(t,d) \times log_2\left(1 + \frac{ADL(C)}{len(d)}\right) \tag{2.16}$$

จากตัวอย่างต่อไปนี้ จะแสดงความเข้าใจในเรื่องของความเหมาะสมของการใช้ *RITF* และ *LRTF*

ตัวอย่างที่ 1 ให้เอกสาร d_1 และ d_2 มีความยาวของเอกสารเท่ากันและมีค่าต่างๆ ดังนี้

- $len(d_1) = 20$, #distinct term of $d_1 = 5$, $TF(t, d_1) = 4$
- $len(d_2) = 20$, #distinct term of $d_2 = 15$, $TF(t, d_2) = 15$

จากตัวอย่างที่ 1 ในกรณีที่ใช้ *LRTF* จะถือว่าความสำคัญเท่ากัน ซึ่งไม่เหมาะสมเพราะเอกสาร d_1 ดูเหมือนจะถูกแบ่งอย่างเท่าเทียมกันใน 5 คำ จึงไม่ควรพิจารณาว่าสำคัญ ในขณะที่เอกสาร d_2 ที่คำ t ปรากฏจะมีความสำคัญมากกว่า ดังนั้นในกรณีนี้ *RITF* น่าจะเหมาะสมกว่า

ตัวอย่างที่ 2 ให้เอกสาร d_1 และ d_2 มีค่าต่างๆ ดังนี้

- $len(d_1) = 20$, #distinct term of $d_1 = 15$, $TF(t, d_1) = 4$
- $len(d_2) = 200$, #distinct term of $d_2 = 150$, $TF(t, d_2) = 4$

ในตัวอย่างที่ 2 นี้ *RITF* พิจารณาคำ t ว่ามีความสำคัญเท่ากันทั้งเอกสาร d_1 และ d_2 ซึ่งไม่ถูกต้องเนื่องจาก d_2 มีข้อกำหนดที่แตกต่างกันมากดังนั้นในกรณีนี้ *LRTF* ดูเหมือนจะเป็นทางเลือกที่มีศักยภาพมากกว่า *RITF*

จากนั้นได้แปลงสมการทั้งสอง สมการ (2.14) และ สมการ (2.16) โดยใช้ฟังก์ชัน $f(x) = x/(1+x)$ เพื่อกำหนดค่าขอบบนให้มีค่าเป็น 1 ซึ่งจะได้สมการ (2.17) และ สมการ (2.18)

$$BRITF(t,d) = \frac{RITF(t,d)}{1 + RITF(t,d)} \tag{2.17}$$

$$BLRTF(t,d) = \frac{LRTF(t,d)}{1 + LRTF(t,d)} \tag{2.18}$$

ต่อมาคือการรวมปัจจัยทั้งสองเข้าด้วยกัน (Combination two TF factor) โดยแสดงได้ดังสมการ (2.19)

$$TFF(t,d) = w \times BRITF(t,d) + (1-w) \times BLRTF(t,d)) \tag{2.19}$$

แต่สมการ *BRITF*(*t, d*) มีแนวโน้มที่จะชอบเอกสารยาว เนื่องจากเอกสารที่ยาว *RITF*(*t, d*) มีค่าใกล้เคียงกับ 1 และมักจะมีความถี่ของ *TF* มากกว่า ในทางตรงกันข้าม *BLRTF*(*t ,d*) มีแนวโน้มที่จะชอบเอกสารสั้นๆ เนื่องจาก *LRTF*(*t, d*) ที่มีค่าเข้าใกล้ 0 จะมีค่า *len*(*d*) เป็นอินฟินิตี้ จึงต้องมีการให้ค่า *w* ที่เหมาะสมเพื่อรวมทั้งสองปัจจัยเข้าด้วยกันโดยไม่มีความลำเอียงไปยังเอกสารที่สั้นหรือเอกสารที่ยาว ซึ่งค่า *w* หาได้ตามสมการ (2.20) โดยที่ *Q* คือจำนวนคำในคำขอ (Query) นั้น

$$w = \frac{2}{1 + log_2(1 + |Q|)} \tag{2.20}$$

ต่อมาเป็นการพิจารณาการให้น้ำหนักกับคำที่หาได้ยากในคลังเอกสารนั่นก็คือ ความถี่เอกสารผกผัน (*IDF*) โดยสมการของ *IDF* ที่ใช้คือ

$$IDF = log\left(\frac{CS(C) + 1}{DF(t,C)}\right) \tag{2.21}$$

โดยที่ *CS*(*C*) คือ จำนวนเอกสารทั้งหมดในคลังเอกสาร *C* และ *DF*(*t, C*) คือ จำนวนเอกสารที่มีคำ *t* ปรากฏในคลังเอกสาร *C*

ค่า *IDF* ข้างต้นพิจารณาเฉพาะการปรากฏตัวหรือไม่ปรากฏของคำในเอกสารเท่านั้นแต่ไม่คำนึงถึงการปรากฏของคำเฉพาะในเอกสาร ดังนั้นจากสมมติฐานว่าถ้ามีคำสองคำที่มีความถี่เท่ากัน การแบ่งแยกคำควรเพิ่มขึ้นด้วยการเพิ่มค่าเฉลี่ยตามความถี่ของคำ (The average elite set term frequency: *AEF*) ดังสมการ (2.22)

$$AEF = \frac{CTF(t, C)}{DF(t, C)} \tag{2.22}$$

โดยที่ *CTF*(*t, C*) หมายถึง จำนวนครั้งที่ปรากฏคำ *t* ในคลังเอกสาร *C*

จากนั้นใช้ $f(x) = x/(1+x)$ เพื่อแปลงค่า *AEF* สำหรับการใช้งานในสมการ (2.23) Term discrimination factor: *TDF*

$$TDF(t, d) = IDF(t, C) \times \frac{AEF(t, C)}{1 + AEF(t, C)} \tag{2.23}$$

จากการรวมปัจจัยข้างต้นจะได้สมการสุดท้าย ซึ่งปรับให้ค่าคะแนนความคล้ายคลึงมีผลลัพธ์มีค่าอยู่ระหว่าง 0-1 ดังสมการ (2.24)

$$Sim_{norm}(Q, d) = \frac{\sum_{i=1}^{|Q|} TFF(q_i, d) \times TDF(q_i, C)}{\sum_{i=1}^{|Q|} TDF(q_i, C)} \tag{2.24}$$

2.7.7 การให้น้ำหนักคำแบบความถี่และแรงดึงดูดผกผัน (Term frequency – inverse gravity moment: *tf-igm*)

การให้น้ำหนักคำแบบความถี่และแรงดึงดูดผกผันเป็นรูปแบบการให้น้ำหนักคำแบบมีผู้สอน (Supervise term weighting: STW) ที่นำเสนอโดย Chen และคณะ [67] เพื่อใช้ในงานการจำแนกข้อความ (Text classification) ที่เหมาะสำหรับใช้ได้ทั้ง ไบนารีคลาส และมัลติคลาส โดยมีสมการดังนี้

$$tf - igm_{t,d} = f_{t,d} \times (1 + \lambda \times igm(t_k)) \tag{2.25}$$

โดยที่ $f_{t,d}$ คือ จำนวนของคำ *t* ที่ปรากฏในเอกสาร *d* สำหรับ λ คือค่าสัมประสิทธิที่ปรับค่าได้ระหว่าง 5.0-9.0 ซึ่ง λ จะพยายามรักษาสมดุลระหว่างการให้น้ำหนักแบบ global และการให้น้ำหนักแบบ local

ในขณะที่ *igm* ถูกใช้เพื่อวัดความถี่ของการกระจายระหว่างคลาสของคำ ซึ่งสามารถกำหนดได้ดังต่อไปนี้

$$igm(t_k) = \frac{f_{k1}}{\sum_{r=1}^{m} f_{kr} \cdot r} \tag{2.26}$$

โดยที่ $igm(t_k)$ แทนแรงดึงดูดผกผันของการกระจายระหว่างคลาสของคำ t_k และ f_{kr} (r = 1, 2,...,m) เป็นความถี่ของการเกิดคำ t_k ในแต่ละคลาส ซึ่งเรียงลำดับจากมากไปน้อยโดยมี r เป็นลำดับ ส่วน f_{kr} คือจำนวนเอกสารที่คำ t_k ปรากฏในแต่ละคลาสที่ r

2.8 การจัดกลุ่มข้อมูลแบบคลัสเตอริง

2.8.1 การจัดกลุ่มข้อมูลแบบเคมีน (*K*-means clustering)

การจัดกลุ่มข้อมูลแบบเคมีน (k-means) เป็นอัลกอริธึมสำหรับการจัดกลุ่มแบบคลัสเตอริง (Clustering) ที่ได้รับความนิยมในการจัดกลุ่มข้อมูลที่เป็นข้อความอย่างมาก ซึ่งการจัดกลุ่มแบบคลัสเตอริงจัดเป็นการเรียนรู้แบบไม่มีผู้สอน ซึ่งหมายถึง การเรียนรู้จากข้อมูลที่ไม่มีการกำหนดค่าเป้าหมาย (Target) หรือฉลาก (Label) ของคลาส (Class) ไว้ ซึ่งแตกต่างจากการเรียนรู้แบบมีผู้สอน ที่ข้อมูลตัวอย่างต้องมีการกำหนดค่าเป้าหมายหรือฉลากไว้ก่อน [68] ซึ่งวัตถุประสงค์ของการจัดกลุ่มนั้นเพื่อการจำแนกกลุ่มข้อมูลที่มีคุณลักษณะคล้ายกันอยู่ในกลุ่มเดียวกัน [69] โดยข้อมูลในแต่ละกลุ่มจะถูกเรียกว่า “คลัสเตอร์ (Cluster)”

การจัดกลุ่มข้อมูลแบบเคมีน [70] คือวิธีการจัดกลุ่มข้อมูลด้วยวิธีการแบ่งข้อมูลอัตโนมัติตามค่า k ที่กำหนด โดยกระบวนการทำงานเลือกค่า k เริ่มต้นสำหรับเป็นค่ากลาง (Centroid) ในการจัดกลุ่ม และมีการปรับค่าตามกระบวนการดังนี้

1) เลือกข้อมูล *di* สำหรับวัดระยะห่างกับค่า Centroid เริ่มต้นทุกค่าของแต่ละคลัสเตอร์

2) กำหนดชุดข้อมูล *di* ให้กับคลัสเตอร์ที่มีระยะห่างของ Centroid และ *di* ที่ใกล้ที่สุด

3) ปรับค่า Centroid ใหม่ของแต่ละคลัสเตอร์ตามข้อมูลในที่อยู่ในแต่ละคลัสเตอร์ และหยุดเมื่อ Centroid ไม่เปลี่ยนแปลง [68, 71]

ดังนั้นการใช้การจัดกลุ่มด้วยขั้นตอนวิธีเคมีน จึงสามารถนำมาใช้งานเมื่อทราบจำนวนกลุ่มที่ต้องการจัดกลุ่มที่แน่นอน ทั้งนี้การวัดระยะห่างระหว่างข้อมูลสามารถใช้การคำนวณระยะทางได้หลากหลาย เช่น ระยะทางยูคลิด (Euclidean distance) ดังสมการ (2.27) หรือระยะทางแบบแมนฮัตตัน (Manhattan distance measure) ดังสมการ (2.28) เป็นต้น

$$\delta(x,y) = \left(\sum_{i=1}^{n}(x_i - y_i)\right)^{1/2} \tag{2.27}$$

$$\delta(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|\right) \tag{2.28}$$

และสามารถแสดงขั้นตอนวิธีเคมีน ได้ดังรูปที่ 2.23

Algorithm	:	The k-means clustering
Input	:	Data point *D*, Number of cluster *k*
Output	:	Data points with cluster memberships
Step 1	:	Initialize *k* centroids randomly
Step 2	:	Associate each data point in *D* with the nearest centroid. This will divide the data points into *k* clusters.
Step 3	:	Recalculate the position of centroids.
Repeat step 2 and 3 until there are no more changes in the memberships of the data points		

รูปที่ 2.23 รหัสเทียมขั้นตอนวิธีของเคมีน

2.8.2 การจัดกลุ่มข้อมูลด้วยเคมีนทรงกลม (Spherical *k*-means clustering)

การจัดกลุ่มข้อมูลด้วยเคมีนทรงกลม [72] เป็นการประยุกต์จากอัลกอริทึมเคมีน ซึ่งโดยปกติเคมีน จะวัดระยะทางระหว่างเวกเตอร์เอกสารกับเวกเตอร์ตัวแทนของคลัสเตอร์ (Centroid) โดยใช้ระยะทางยูคลิด แต่สำหรับเคมีนทรงกลมจะใช้ค่าสัมประสิทธิ์โคไซน์ (Cosine coefficient) แทนดังสมการ (2.29)

$$sim(x,y) = \frac{\sum_{i=1}^{n} x_i \times y_i}{\left(\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} y_i^2\right)^{1/2}} \tag{2.29}$$

ค่าสัมประสิทธิ์โคไซน์ เป็นวิธีการหาความคล้ายคลึงกันจากค่าความต่างของมุมของวัตถุ 2 อันที่เกิดขึ้นบนพื้นที่เวกเตอร์ การวัดความคล้ายคลึงกันแบบโคไซน์ เสนอโดย Singhal เมื่อปี ค.ศ. 2001 จะมีค่าอยู่ระหว่าง 0-1 เท่านั้น โดยตัวหารคือระยะห่างยูคลิด ซึ่งวิธีการนี้จะมีประสิทธิภาพในกรณีที่เอกสาร 2 เอกสารมีความยาวไม่เท่ากันหรือทำให้มีความยุติธรรมต่อเอกสารที่สั้นกว่า โดยแนวคิดในการจัดกลุ่มเอกสารด้วยเคมีนทรงกลม คือ หาเวกเตอร์ ที่เป็นตัวแทนของกลุ่มเวกเตอร์เอกสารแต่ละคลัสเตอร์ แล้วให้เวกเตอร์เอกสารทั้งหมดเปรียบเทียบกับเวกเตอร์ที่เป็นตัวแทนของแต่ละคลัสเตอร์เหล่านี้เพื่อหาคลัสเตอร์ที่อยู่ใกล้ที่สุดจนกว่าจะพบเงื่อนไขการหยุด

กำหนดให้ X แทนเวกเตอร์เอกสาร π_j แทนส่วนของเวกเตอร์เอกสารโดยที่

$\cup_{j=1}^{k} \pi_i = \{x_1, x_2, \ldots, x_n\}$ และ $\pi_j \cap \pi_\ell \neq \emptyset$ ถ้า $j \neq \ell$

m_j คือค่าเฉลี่ยเวกเตอร์ หรือ centroid ของเวกเตอร์เอกสารที่อยู่ในกลุ่ม π_j ซึ่งคือ $m_j = \frac{1}{n_j}\sum_{x\epsilon\pi_j} x$ โดยที่ n_j คือจำนวนเวกเตอร์เอกสารที่อยู่ใน π_j และสามารถคำนวณหาค่า Concept vector ได้ดังสมการต่อไปนี้

$$c_j = \frac{m_j}{\|m_j\|} \tag{2.30}$$

Concept vector c_j มีคุณสมบัติที่สำคัญคือ สำหรับเวกเตอร์ z ใน R^d มีความไม่เท่ากันของ Cauchy-Schwarz ที่ $\sum_{x\epsilon\pi_j} x^T z \leq \sum_{x\epsilon\pi_j} x^T c_j$

ดังนั้น Concept vector อาจเป็นเวกเตอร์ที่ใกล้เคียงที่สุดในความคล้ายคลึงกันแบบโคไซน์กับเวกเตอร์เอกสารทั้งหมดในคลัสเตอร์ π_j

การวัดคุณภาพของคลัสเตอร์$\pi_j, 1 \le J \le k$ ใช้ objective function ด้วยสมการต่อไปนี้

$$Q\left(\{\pi_j\}_{j=1}^{k}\right) = \sum_{j=1}^{k}\sum_{X\epsilon\pi_j} x^T c_j \qquad (2.31)$$

โดยที่ $x^T c_j$ แทน inner product ระหว่างเวกเตอร์เอกสารและ Concept vector ดังนั้นจึงสามารถเขียนเป็นขั้นตอนวิธีเคมีนทรงกลมได้ดังต่อไปนี้

ตารางที่ 2.9 ขั้นตอนวิธีเคมีนทรงกลม

1	:	เลือกจำนวนกลุ่ม k แล้วแบ่งเวกเตอร์ เอกสารด้วยการสุ่มออกเป็น $\pi_j, 1 \le J \le k$ ให้ $c_j^{(0)}$ แทน concept vector ที่สอดคล้องกับส่วนของเวกเตอร์ ที่กำหนด และกำหนดค่าของ iteration t = 0
2	:	สำหรับแต่ละเวกเตอร์ เอกสาร x_i หา concept vector ที่อยู่ใกล้กับ x_i มากที่สุด แล้วคำนวณ $\pi_j^{(t+1)}$ ใหม่ ซึ่งกำหนดโดย $c_j^{(t)}$ โดยที่ $\pi_j^{t+1} = \left\{x\epsilon\{x_i\}_{i=1}^{n} : x^T c_j^{(t)} > x^T c_\iota^{(t)}, 1 \le \iota \le n, \iota \ne j\right\}, 1 \le j \le k$ กล่าวคือ $\pi_j^{(t+1)}$ คือชุดเวกเตอร์ เอกสารทั้งหมดที่อยู่ใกล้กับ concept vector $c_j^{(t)}$ หากเวกเตอร์เอกสารตัวใดใกล้กับ concept vector มากกว่าหนึ่งตัวแล้วจะได้ทำการสุ่มให้เป็นสามาชิกในกลุ่มใดกลุ่มหนึ่ง
3	:	คำนวณหา concept vector ใหม่ ซึ่งสอดคล้องกับกลุ่มคลัสเตอร์ $\pi_j^{(t+1)}$ ที่ได้จากขั้นตอนที่ 2 ดังนี้ $c_j^{(t+1)} = \frac{m_j^{(t+1)}}{\left\|m_j^{(t+1)}\right\|}, 1 \le j \le k$ โดยที่ $m_j^{(t+1)}$ คือ centroid หรือ mean ของเวกเตอร์ เอกสารในคลัสเตอร์ $\pi_j^{(t+1)}$
4	:	ถ้า objective function เปลี่ยนแปลงน้อยกว่าค่า threshold แล้ว ให้จบการทำงาน หากไม่เป็นเช่นนั้น ให้เพิ่มค่า t ด้วย 1 แล้วกลับไปทำขั้นตอนที่ 2 ตัวอย่างของเกณฑ์ในการกำหนดค่า threshold ที่ทำให้จบการทำงาน $\left\|Q\left(\{\pi_j^{(t)}\}_{j=1}^{k}\right) - Q\left(\{\pi_j^{(t+1)}\}_{j=1}^{k}\right)\right\| \le \varepsilon$

2.9 ฟังก์ชันการค้นคืน (Retrieval function: REP)

ฟังก์ชันการค้นคืนข้อมูลได้ถูกเสนอโดย Sun และคณะ [73] ในงานวิจัยเพื่อแก้ปัญหาการตรวจหารายงานจุดบกพร่องซ้ำซ้อน ซึ่งงานนี้ได้สังเกตว่ารายงานจุดบกพร่องที่ซ้ำซ้อนกันมีลักษณะคล้ายกันไม่เพียงเฉพาะข้อความที่แสดงในข้อมูล Summary หรือ Description เท่านั้น แต่ยังปรากฏในข้อมูลประเภทอื่นๆ เช่น ในข้อมูลของ ผลิตภัณฑ์ (Product) ส่วนประกอบ (Component) ลำดับความสำคัญ (Priority) เป็นต้น พวกเขาจึงคิดวิธีฟังก์ชันการค้นคืนที่เรียกว่า REP ซึ่งเป็นชุดค่าผสมเชิงเส้นของคุณลักษณะหลายๆ คุณลักษณะ ในที่นี้ w_i เป็นน้ำหนักความสำคัญสำหรับคุณลักษณะต่างๆ ซึ่งสามารถแสดงได้ดังสมการ (2.32)

$$REP(d,q) = \sum_{i=1}^{n} w_i \times feature_i \qquad (2.32)$$

โดยที่ q คือเอกสารที่เป็น query และ d คือเอกสารที่นำมาเทียบความคล้ายกับเอกสาร q ส่วน n คือจำนวนของคุณลักษณะที่ใช้ในการหาความคล้ายคลึง ซึ่งแต่ละน้ำหนักแสดงถึงระดับ

ความสำคัญของคุณลักษณะที่สอดคล้องกัน และคุณลักษณะเหล่านั้นแบ่งได้เป็น 2 ประเภท คือ คุณลักษณะแบบข้อความ (Textual features) และ คุณลักษณะแบบหมวดหมู่ (Category features)

คุณลักษณะแบบข้อความ คือ การวัดความคล้ายกันของเอกสารรายงานจุดบกพร่อง โดยใช้ข้อมูลใน Summary และ Description ส่วนคุณลักษณะแบบหมวดหมู่ คือ ความเหมือนกันของข้อมูล ผลิตภัณฑ์ (Product) ส่วนประกอบ (Component) หรือความห่างกันในข้อมูล ลำดับความสำคัญ (Priority) และ เวอร์ชัน (Version)

จากฟังก์ชันการค้นคืนที่ Sun และคณะ เสนอนั้นได้มีหลายงานวิจัยได้นำไปประยุกต์ใช้ อาทิ Tian และคณะ [3] หรือ Ye และคณะ [42] เป็นต้น นอกจากนี้สมการ (2.32) นี้ถูกเรียกว่า ฟังชันความคล้าย หรือ *sim*() และสามารถนำไปปรับค่าพารามิเตอร์อิสระภายในได้ซึ่งจะกล่าวในลำดับถัดไป

2.10 การเพิ่มประสิทธิภาพฟังก์ชันความคล้ายโดยการเคลื่อนลงตามความชัน (Optimizing similarity functions by gradient descent)

ขั้นตอนวิธีการเคลื่อนลงตามความชัน (Gardient descent) [68, 74] เป็นวิธีหนึ่งที่นิยมใช้ในการแก้ปัญหาการหาค่าน้อยที่สุด รองรับการแก้ปัญหาที่มีตัวแปรหลายมิติ [74] ซึ่งวิธีการนี้เป็นการใช้ค่าความชันของฟังก์ชัน $f(x)$ ในการเคลื่อนที่จากจุดเริ่มต้นที่ได้กำหนดขึ้น x_0 ไปสู่จุดต่ำสุด (ค่าน้อยที่สุด) โดยจุดแรกที่พบจากจุดเริ่มต้นนั้น สามารถแสดงได้ดังสมการ

$$x_{k+1} = x_k - \eta \nabla f(x_k) \tag{2.33}$$

โดยที่ η เรียกว่าค่าอัตราการเรียนรู้ (Learning rate) หรือ ขนาดก้าว (Size step) ที่มีค่าน้อย และ $\eta \geq 0$ โดยค่าของฟังก์ชันของจุดต่อไปจะลดน้อยลงกว่าเดิม $f(x_{k+1}) < f(x_k)$

เนื่องจากในงานวิจัยของ [3, 73, 75] ได้ใช้สมการเชิงเส้นในการรวมค่าของแต่คุณลักษณะในการหาความคล้ายของเอกสารดังที่แสดงในสมการ (2.32) ซึ่งต้องใช้พารามิเตอร์ในการให้ความสำคัญกับแต่ละคุณลักษณะจึงได้มีการประยุกต์ใช้การเคลื่อนลงตามความชัน เพื่อปรับค่าพารามิเตอร์ในฟังก์ชันความคล้าย

โดยที่ฟังก์ชันความคล้าย หรือ *sim*(*d*, *q*) คือการคำนวณความคล้ายคลึงกันระหว่างเอกสาร *d* และ เอกสาร *q* ซึ่ง *q* คือเอกสารที่เป็นแบบสอบถาม (Query) และมีเวกเตอร์ *n* ที่เป็นพารามิเตอร์อิสระ (x_1, x_2,...,x_n) เพื่อให้ฟังก์ชัน *sim*() สามารถทำงานได้มีประสิทธิภาพที่ดีที่สุดจำเป็นต้องมีข้อมูลชุดสอน (Training set) เพื่อปรับแต่งพารามิเตอร์ *n* ใน ฟังก์ชัน *sim*()

ข้อมูลชุดสอนในแต่ละกรณีจะประกอบไปด้วยเอกสารสามเอกสารด้วยกันคือ (*q*, *rel*, *irr*) โดยที่ *q* คือเอกสารที่เป็นแบบสอบถาม *rel* คือเอกสารที่มีความเกี่ยวข้องกับ *q* และ *irr* คือเอกสารที่ไม่มีความเกี่ยวข้องกับ *q* ซึ่งมีแนวคิดมาจาก *sim*(*d* ,*q*) ที่มีความเกี่ยวข้องกันย่อมมีค่าสูง ในขณะที่เอกสารที่ไม่เกี่ยวข้องกันจะต้องมีค่าที่ต่ำกว่า หรือ *sim*(*rel*, *q*) > *sim*(*irr*, *q*) และ Ranknet cost

function: RNC [76] ในการคำนวณค่า Cost ในการสอนแต่ละกรณีของ (*q*, *rel*, *irr*) เป็นดังสมการ (2.34)

$$RNC(I) = log(1 + e^{Y}) \tag{2.34}$$

โดยที่ *Y* คือ *sim*(*irr*, *q*) – *sim*(*rel*, *q*) การเพิ่มประสิทธิภาพของ *sim*() คือการลดค่า cost ของ RNC ในแต่ละรอบการสอนและการลดค่าทำได้โดยการปรับค่าพารามิเตอร์อิสระใน *sim*() [73, 75, 77] ดังสมการ (2.35)

$$x = x - \eta \times \frac{\partial RNC}{\partial x}(I) \tag{2.35}$$

โดยที่ *x* คือพารามิเตอร์อิสระใน *sim*() η คือค่าอัตราการเรียนรู้ที่มีค่าขนาดเล็ก เช่น 0.001 ในขณะที่ $\frac{\partial RNC}{\partial x}$ คืออนุพันธ์บางส่วนของ RNC ที่เกี่ยวข้องกับพารามิเตอร์ *x* ซึ่งตามหลักการการปรับค่าของพารามิเตอร์อิสระ *x* จะช่วยให้ RNC เข้าสู่ค่าน้อยที่สุดได้

2.11 เทคนิคในการประเมินผลการวิจัย

จากการศึกษางานวิจัยที่ผ่านมาดังกล่าวข้างต้น ได้พบว่าการประเมินผลการวิจัยได้ใช้การประเมินประสิทธิภาพหลากหลาย [78, 79] ซึ่งมีตัววัดที่นิยมใช้ดังนี้

- การวัดค่าความแม่นยำ (Precision) ค่าที่บอกว่าโปรแกรมทำนายว่าจริง ถูกต้องเท่าไร
- การวัดค่าความระลึก (Recall) ค่าที่บอกว่าโปรแกรมทำนายได้ว่าจริง เป็นอัตราส่วนเท่าไรของจริงทั้งหมด
- ค่าเฉลี่ยประสิทธิภาพโดยรวม (F-measure) เป็นการวัดค่าความถูกต้องการทดสอบที่พิจารณาทั้งความแม่นยำ และ ค่าความระลึก พร้อมกัน
- การวัดค่าความถูกต้อง (Accuracy) ค่าที่บอกว่าโปรแกรมสามารถทำนายได้ถูกต้องขนาดไหน

ซึ่งสามารถอธิบายค่าวัดที่กล่าวข้างต้นได้ จากตาราง Confusion matrix โดยที่เป็นตารางแบบจัตุรัส มีจำนวนแถวเท่ากับจำนวนคอลัมน์ และเท่ากับจำนวนคลาส เช่น ในที่นี้กำหนดให้คลาสมีคำตอบ 2 ค่า คือ True (+) และ False (-) ดังนั้นตาราง Confusion matrix นี้ จะแสดงตารางขนาด 2x2 ดังแสดงในตารางที่ 2.10

ตารางที่ 2.10 ตาราง Confusion matrix

		Predicted	
		+	-
Actual	+	TP	FN
	-	FP	TN

จากตารางที่ 2.10 ค่าที่แสดงในช่องต่างๆ ของตารางประกอบด้วย

- True Positive (TP) คือ สิ่งที่โปรแกรมทำนายว่าจริง และมนุษย์บอกว่าจริง
- True Negative (TN) คือ สิ่งที่โปรแกรมทำนายว่าไม่จริง และมนุษย์บอกว่าไม่จริง
- False Positive (FP) คือ สิ่งที่โปรแกรมทำนายว่าจริง แต่มนุษย์บอกว่าไม่จริง
- False Negative (FN) คือ สิ่งที่โปรแกรมทำนายว่าไม่จริง แต่มนุษย์บอกว่ามันจริง

จากตารางที่ 2.10 จะสามารถเขียนเป็นสูตรในการคำนวณค่าวัดต่างๆ ได้ดังนี้

1) Accuracy คือ ค่าที่บอกว่าโปรแกรมสามารถทำนายได้ถูกต้องขนาดไหน

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.36}$$

2) Recall (True positive rate) คือ ค่าที่บอกว่าโปรแกรมทำนายได้ว่าจริง เป็นอัตราส่วนเท่าไรของจริงทั้งหมด

$$Recall = \frac{TP}{TP + FN} \tag{2.37}$$

3) True negative rate (TNR) คือ ค่าที่บอกว่าโปรแกรมทำนายได้ว่าไม่จริง เป็นอัตราส่วนเท่าไรของจริงทั้งหมด

$$TNR = \frac{TN}{TN + FP} \tag{2.38}$$

4) False positive rate (FPR) คือ ค่าที่บอกว่าโปรแกรมทำนายว่าจริง เป็นอัตราส่วนเท่าไรของไม่จริงทั้งหมด

$$FPR = \frac{FP}{TN + FP} \tag{2.39}$$

5) False negative rate (FNR) คือ ค่าที่บอกว่าโปรแกรมทำนายว่าไม่จริง เป็นอันตราส่วนเท่าไรของจริงทั้งหมด

$$FNR = \frac{FN}{TP + FN} \tag{2.40}$$

6) Precision ค่าที่บอกว่าโปรแกรมทำนายว่าจริง ถูกต้องเท่าไร

$$Precision = \frac{TP}{TP + FP} \tag{2.41}$$

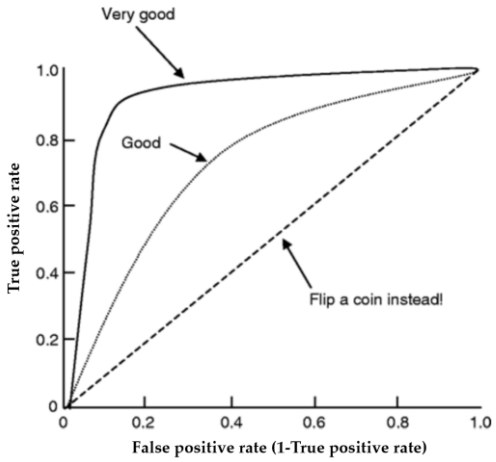
7) F-measure เป็นการวัดค่าความถูกต้องการทดสอบที่พิจารณาทั้งความแม่นยำ และค่าความระลึก พร้อมกัน

$$F-measure\ = 2 \times \frac{Precision \times Recall}{(Precision + Recall)} \quad (2.42)$$

แต่งานทางด้านรายงานจุดบกพร่อง เช่น การตรวจสอบรายงานจุดบกพร่องที่ซ้ำซ้อน [3, 11, 12] ได้ใช้ค่า F-measure ในรูปแบบสมการต่อไปนี้

$$F-measure\ = 2 \times \frac{TPR\ \times TNR}{(TPR + TNR)} \quad (2.43)$$

8) Receiver Operating Characteristic (ROC) Curves หรือเส้นโค้ง ROC เป็นวิธีการที่สามารถนำมาใช้เพื่อเลือกจุดตัดที่เหมาะสมได้ คือการสร้างเส้นโค้ง ROC โดยที่ ROC คือการพล็อตกราฟความสัมพันธ์ระหว่างค่า TPR (Sensitivity) กับ FPR (1-TNR or 1–Specificity) ซึ่งแกน y แทน TPR และ แกน x แทน FPR โดยการแปรค่าจุดตัด (cut – off point) ที่ใช้ต่างๆ กัน ดังแสดงตามรูปที่ 2.24



รูปที่ 2.24 ตัวอย่างเส้นโค้ง ROC

โดยที่ในการวิเคราะห์ผลควรมีค่า TPR และ TNR สูง ซึ่งหาก TNR ที่มีค่าสูงก็จะทำให้มีค่า FPR ต่ำ ส่งผลให้เส้นโค้ง ROC เข้าชิดมุมซ้ายบนมากที่สุด นอกจากนี้การสร้างเส้นโค้ง ROC ยังสามารถช่วยเปรียบเทียบประสิทธิภาพของการวิเคราะห์ ด้วยการเปรียบเทียบพื้นที่ใต้เส้นโค้ง (Area under the ROC curve) ของแต่ละตัวแบบการวิเคราะห์ ซึ่งถ้าพื้นที่ใต้เส้นโค้ง ROC มีค่ามาก แสดงว่าตัวแบบการวิเคราะห์นั้นมีความถูกต้องมาก ดังต่อไปนี้

AUC = 0.50 หมายถึง ตัวแบบการวิเคราะห์ข้อมูลให้ผลลัพธ์คล้ายมาจากการเดา ซึ่งไม่ดี

AUC > 0.70 หมายถึงตัวแบบการวิเคราะห์ข้อมูลให้ผลลัพธ์อยู่ในเกณฑ์มาตรฐานของตัวแบบการวิเคราะห์ทั่วไป

AUC > 0.80 หมายถึงตัวแบบการวิเคราะห์ข้อมูลให้ผลลัพธ์อยู่ในเกณฑ์ที่ดี

AUC > 0.90 หมายถึงตัวแบบการวิเคราะห์ข้อมูลให้ผลลัพธ์อยู่ในเกณฑ์ที่ดีมาก

บทที่ 3
วิธีดำเนินการวิจัย

การดำเนินการวิจัยให้บรรลุวัตถุประสงค์ ในการที่จะจัดกลุ่มของรายงานจุดบกพร่องที่เป็นส่วนต่อนั้น ผู้วิจัยจะได้นำเสนอในแต่ละขั้นตอนโดยละเอียดดังหัวข้อต่อไปนี้ (1) ชุดข้อมูล (2) กระบวนการเตรียมขั้นต้น (3) การคัดเลือกคุณลักษณะของรายงานจุดบกพร่อง (4) กรอบการดำเนินงานสำหรับ Dependent Bug Report Analysis (5) การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ (6) การจัดกลุ่มรายงานจุดบกพร่องที่เกี่ยวข้องกันด้วยเทคนิคการวัดความคล้าย และ (7) การจัดกลุ่มรายงานจุดบกพร่องที่เกี่ยวข้องกันด้วยเทคนิคฟังก์ชันการค้นคืนข้อมูล

3.1 ชุดข้อมูล (Dataset)

ในงานวิจัยนี้ใช้ชุดข้อมูลหลัก 3 ชุด ได้แก่ (1) ชุดข้อมูลจากมอซิลลา Firefox (2) ชุดข้อมูลของมอซิลา Core และ (3) ชุดข้อมูลมาตรฐานของ Herzig

โดยได้ทำการดาวน์โหลดรายงานจุดบกพร่องจาก http://bugzilla.mozilla.org ในผลิตภัณฑ์ มอซิลลา Firefox และ มอซิลลา Core ที่มีการรายงานจุดบกพร่องเข้าไปในระบบ BTS ของ Mozilla ในระหว่างเดือนตุลาคม พ.ศ. 2543 ถึงกันยายน พ.ศ. 2560 โดยจัดอยู่ในโครงสร้างของ XML

ซึ่งสาเหตุที่งานวิจัยนี้ได้เลือกใช้รายงานจุดบกพร่องจากมอซิลลา ประการแรกเนื่องจากมีหลายงานวิจัยที่ได้เลือกใช้ [1, 3, 9, 11, 12, 23, 37] ประการที่สอง คือระบบรายงานจุดบกพร่องของมอซิลลานั้นมีผู้ดูแลและทีมงานในแต่ละผลิตภัณฑ์อย่างชัดเจน ดังจะเห็นได้จากรูปที่ 3.1 ซึ่งจะเห็นได้ว่า ซอฟต์แวร์มอซิลลา Firefox นั้นได้มีการกำหนดผู้ดูแลรับผิดชอบในการติดตามแก้ไขปัญหาจุดบกพร่องอย่างชัดเจนและสามารถติดต่อได้ผ่านอีเมล์ของแต่ละบุคคล จึงเป็นที่เชื่อมั่นในข้อมูลรายงานจุดบกพร่องที่ผ่านการประมวลผลจากบุคลากรเหล่านี้จะมีความน่าเชื่อถือมากยิ่งขึ้น ยกตัวอย่างเช่น การตัดสินว่ารายงานจุดบกพร่องเป็นจุดบกพร่องหรือไม่ การตรวจสอบความซ้ำซ้อนของรายงานจุดบกพร่อง การกำหนดความสำคัญให้กับรายงานจุดบกพร่อง เป็นต้น

ประการสุดท้าย เพราะมอซิลลา Firefox เป็นซอฟต์แวร์โอเพนซอร์ส ในกลุ่มซอฟต์แวร์บราวเซอร์ที่มีสัดส่วนการใช้งานเป็นอันดับสองของโลกรองจาก Chrome ของกูเกิล (Chrome 59.32% Firefox 12.87% ข้อมูลจากเว็บไซต์ netmarketshare.com ณ วันที่ 26 กุมภาพันธ์ 2561)

ด้วยเหตุนี้ผู้วิจัยจึงได้ทำการรวบรวมไว้รายงานจุดบกพร่อง เมื่อวันที่ 1 ตุลาคม 2560 โดยมีข้อกำหนดในการรวบรวมดังนี้ รายงานจุดบกพร่องจะต้องมีสถานะเป็น NEW, ASSIGNED, REOPENED, RESOLVED, VERIFIED และ CLOSED เท่านั้น เนื่องจากสถานะเหล่านี้ได้ยืนยันความเป็นรายงานจุดบกพร่องหรือไม่ และ ความเป็นรายงานซ้ำซ้อนหรือไม่ จากผู้รับผิดชอบในซอฟต์แวร์มอซิลลา Firefox เป็นที่เรียบร้อยแล้ว (ในที่นี่หมายรวมถึงผู้ตรวจสอบรายงานจุดบกพร่อง และผู้พัฒนาซอฟต์แวร์) ดังรูปที่ 3.2

Firefox

Name:	Firefox (#)
Description:	Standalone Web Browser
Owner:	Dave Townsend
Peer(s):	Ehsan Akhgari, Paolo Amadini, Mark Banner, Mike de Boer, Marco Bonardo, Brian Bondy, Kit Cambridge, Shane Caraveo, Tantek Çelik, Luke Chang, Ricky Chien, Mike Conley, Neil Deakin, Justin Dolske, Georg Fritzsche, Nathan Froyd, Felipe Gomes, Dão Gottwald, Luca Greco, Tim Guan-tin Chien, Mark Hammond, Axel Hecht, Rob Helmer, Johann Hofmann, Mike Hommey, Matt Howell, Kate Hudson, Tomislav Jovanovic, Mike Kaply, Gijs Kruitbosch, Edward Lee, KM Lee Rex, Fred Lin, Ray Lin, Fischer Liu, Kris Maglione, François Marier, Jim Mathies, Bill McCloskey, Mark Mentovai, Ted Mielczarek, Nicholas Nethercote, Brian Nicholson, Matthew Noorenberghe, Gian-Carlo Pascutto, Olli Pettay, Florian Quèze, David Rajchenbach-Teller, Neil Rashbrook, Asaf Romano, Marina Samuel, J Ryan Stinnett, Robert Strong, Nihanth Subramanya, Andrew Sutherland, Gabriele Svelto, Andrew Swan, Gregory Szorc, Tim Taubert, Jan Varga, Jonathan Watt, Jared Wein, Drew Willcoxon
Owner(s) Emeritus:	*Mike Shaver, Mike Connor, Gavin Sharp, Dave Camp*
Peer(s) Emeritus:	*Dietrich Ayala, Ian Gilman, Blair McBride, Paul O'Shannessy, Benjamin Smedberg*
Source Dir(s):	browser/
Bugzilla Component(s):	Firefox
URL(s):	Code Review Guidelines
Discussion Group:	firefox-dev

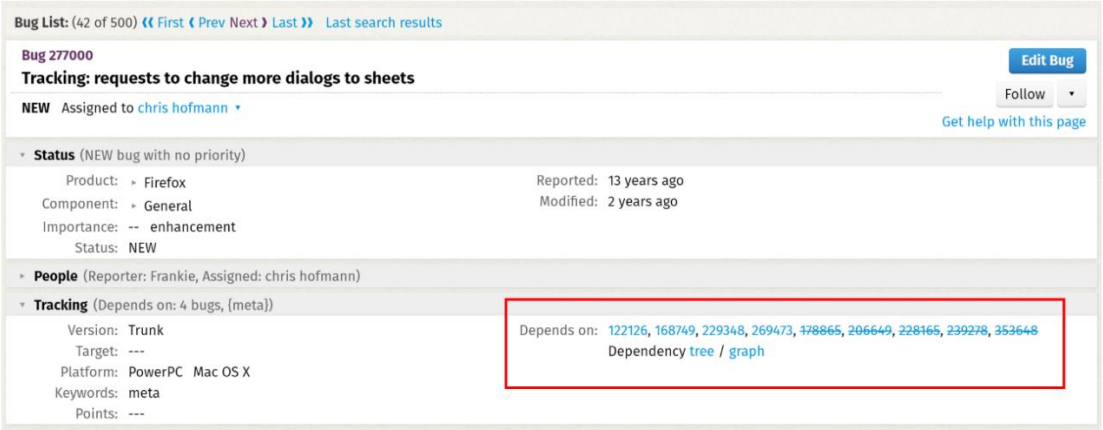
รูปที่ 3.1 รายชื่อผู้ดูแลรับผิดชอบผลิตภัณฑ์ Firefox ของมอซิลลา

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

ID	Product	Comp	Assignee ▲	Status ▲	Resolution	Summary	Changed
944228	Firefox	Theme	abhinav.koppula@gmail.com	NEW	---	Add a test to verify that the space above tabs is consistent across PB, LWT and sizemode (where appropriate)	2017-08-19
688130	Firefox	Toolbars and Customi	acelists@atlas.sk	NEW	---	Popup blocker yellow info bar user interface: menu popping out of "Options" button	2012-01-31
120309	Firefox	File Handling	adamlock@eircom.net	NEW	---	Save as web page complete saving the 'block images from this server' images	2016-06-22
120316	Firefox	File Handling	adamlock@eircom.net	NEW	---	When saving files, use Last-Mod: HTTP header to set date info in file system	2016-06-22
125729	Firefox	File Handling	adamlock@eircom.net	NEW	---	When saving page, add source URL address as comment ("saved from ...") [Save As, Save Page As]	2017-06-04
133010	Firefox	File Handling	adamlock@eircom.net	NEW	---	Save As Web page complete, malformed local url paths	2016-06-22
136720	Firefox	File Handling	adamlock@eircom.net	NEW	---	[persist] FixupNode does not deal with XLinks	2016-06-22
159241	Firefox	File Handling	adamlock@eircom.net	NEW	---	should save shortcut icon with web page	2016-06-22
751712	Firefox	General	adhurle@gmail.com	NEW	---	Add support for HTML5 sizes attribute in <link rel=icon>	2018-01-11
961936	Firefox	Build Config	ajvincent@gmail.com	NEW	---	Migrate XULRunner stub and install_app.py into the Gecko SDK	2017-08-18
383348	Firefox	Build Config	allamsetty.anup@gmail.com	NEW	---	configure should fail if you don't have gconf, gnome-vfs	2015-06-12
1350386	Firefox	Session Restore	amarchesini@mozilla.com	NEW	---	SessionStore Utils.serializeInputStream() seems buggy	2017-05-17
1428725	Firefox	Developer Tools: Con	amarchesini@mozilla.com	NEW	---	Crash in `anonymous namespace''::Wrap	2018-02-18
766460	Firefox	Session Restore	andres@appcoast.com	NEW	---	Try to keep app tabs and normal tabs together when restoring a window	2013-04-15
768648	Firefox	Session Restore	andres@appcoast.com	NEW	---	Move text and scroll data from sessionstore into a new JSM	2013-04-15
1197364	Firefox	New Tab Page	aosmond@mozilla.com	NEW	---	Do image color conversion on the GPU	2017-09-14
1197376	Firefox	New Tab Page	aosmond@mozilla.com	NEW	---	Perform Lanczos image downscaling, then color conversion on the downscaled smaller image	2017-09-14

รูปที่ 3.2 รายงานจุดบกพร่องที่เป็นไปตามข้อกำหนดในสถานะ NEW

ขั้นต่อมาทำการตรวจสอบรายงานจุดบกพร่องที่เป็นส่วนต่อกันของจุดบกพร่อง (Bug dependency) เพื่อใช้เป็นข้อมูลในการทดลอง โดยมีรายงานจุดบกพร่องมีส่วนต่อของจุดบกพร่องอื่นๆ 15,396 ดังรูปที่ 3.3 ที่แสดงให้เห็นว่ารายงานจุดบกพร่องหมายเลข 277000 มีจุดบกพร่องที่เป็นส่วนต่อของรายงานจุดบกพร่องนี้อยู่ 9 รายงาน คือ จุดบกพร่องหมายเลข 122126, 168749, 229348, 269473, 178865, 206649, 228165, 239278 และ 353648



รูปที่ 3.3 ลักษณะของรายงานจุดบกพร่องที่แสดงการว่ามีจุดบกพร่องอื่นที่ขึ้นต่อกัน

นอกจากนี้ยังได้ทำการดาวน์โหลดรายงานจุดบกพร่องซึ่งเป็นชุดข้อมูลมาตรฐานในการศึกษาทางด้าน Bug report misclassification คือ ข้อมูลของ Herzig [37] ซึ่งจัดอยู่ในโครงสร้างของ XMLและสามารถสรุปชุดข้อมูลโดยรวมได้ดังตารางที่ 3.1

ตารางที่ 3.1 ชุดข้อมูลรายงานจุดบกพร่อง

BTS	Software/Product	จำนวนรายงาน	สถานะของรายงานจุดบกพร่อง
Bugzilla	Mozilla Firefox Mozilla Core	176,971 1,300	New, Assigned, Reopened, Resolved, Verified, and Closed
Jira (Herzig's Dataset)	HttpClient Lucence Jackrabbit	745 2,441 2,401	Resolved, Verified, Closed and Fixed

หลังจากรวบรวมรายงานจุดบกพร่องดังกล่าวได้แล้ว ขั้นตอนต่อไปคือการแบ่งชุดข้อมูลดังกล่าวเพื่อใช้การทดลองตามกระบวนการที่นำเสนอในวิธีดำเนินงานวิจัย ซึ่งแบ่งออกเป็น 3 ชุดการทดลองดังนี้

3.1.1 ชุดข้อมูลสำหรับการเรียนรู้กฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ

ชุดข้อมูลสำหรับการเรียนรู้กฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ เป็นชุดข้อมูลที่ใช้ในขั้นการเตรียมเบื้องต้น ซึ่งได้นำชุดข้อมูลจากมอซิลลา Core จำนวน 1,300 รายงาน และได้แบ่งข้อมูลเป็นชุดสอน 1,000 รายงาน เพื่อใช้ในการเรียนรู้เพื่อสร้างกฎเชิงไวยากรณ์ ในขณะที่ชุดทดสอบ 300 รายงาน เพื่อใช้ในการทดสอบกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ

3.1.2 ชุดข้อมูลสำหรับการคัดเลือกคุณลักษณะของรายงานจุดบกพร่อง

ชุดข้อมูลสำหรับการคัดเลือกคุณลักษณะของรายงานจุดบกพร่อง ได้แบ่งการศึกษาออกเป็นสองงานวิจัยทางด้านรายงานจุดบพร่อง คือ การตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug และการทำนายระดับความรุนแรงของจุดบกพร่อง

โดยที่ข้อมูลที่ใช้ในการศึกษาการระบุรายงานจุดบกพร่องแบบ bug และ non-bug ได้นำรายงานจุดบกพร่องมาศึกษา 2 กลุ่มด้วยกันคือ กลุ่มแรกคือ ข้อมูลของ Herzig และกลุ่มที่สองคือ มอซิลลา Firefox เพื่อสร้างแบบจำลองในการจำแนกรายงานว่าเป็นจุดบกพร่องของซอฟต์หรือไม่ และสามารถสรุปชุดข้อมูลได้ดังตารางที่ 3.2

ตารางที่ 3.2 ชุดข้อมูลสำหรับการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug

BTS	Software/Product	จำนวนรายงาน	Class bug	Class non-bug
JIRA	HttpClient	745	305	440
Herzig's Dataset	Lucence	2,441	697	1,744
	Jackrabbiit	2,401	937	1,464
Bugzill	Mozilla Firefox	4,000	2,000	2,000

สำหรับข้อมูลที่ใช้ในอีกหนึ่งงานวิจัยทางด้านรายงานจุดบกพร่อง คือ การทำนายระดับความรุนแรงของจุดบกพร่อง ได้ใช้รายงานจุดบกพร่องจากซอฟต์แวร์มอซิลลา Firefox เพื่อใช้ในการสร้างแบบจำลองในการทำนายความรุนแรงของรายงานจุดบกพร่อง ซึ่งภายในระบบติดตามรายงานจุดบกพร่อง Bugzilla ได้มีการกำหนดระดับความรุนแรงออกเป็น 6 ระดับ ได้แก่ Blocker, Critical, Major, Normal, Minor และ Trivial แต่งานวิจัยทางด้านนี้นิยมแบ่งความรุนแรงของจุดบกพร่องออกเป็นไบนารีคลาสเพื่อใช้ในการวิจัย คือ รุนแรง (Severe) และ ไม่รุนแรง (Non-severe) [14, 80-84] ซึ่ง Severe ประกอบไปด้วยความรุนแรงใน 3 ระดับแรก ในขณะที่ Non-severe จะใช้ 3 ระดับถัดมา และในตารางที่ 3.3 คือ จำนวนของข้อมูลที่ใช้ในการศึกษา

ตารางที่ 3.3 ชุดข้อมูลสำหรับการทดลองการทำนายระดับความรุนแรงของจุดบกพร่อง

BTS	Software/Product	จำนวนรายงาน	Severe	Non-Severe
Bugzilla	Mozilla Firefox	4,000	2,000	2,000

3.1.3 ชุดข้อมูลสำหรับวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง

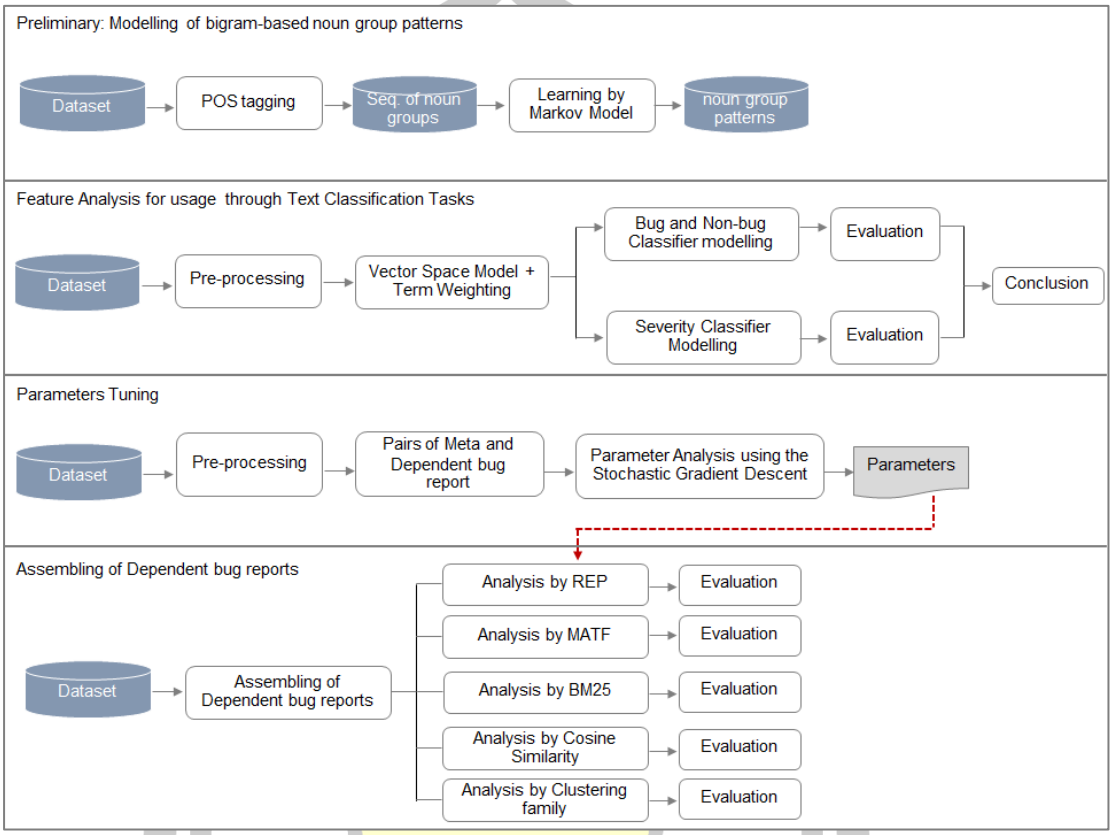
สำหรับชุดข้อมูลที่ใช้ในการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องนั้น ได้ใช้รายงานจุดบกพร่องจากซอฟต์แวร์มอซิลลา Firefox ที่แสดงดังตารางที่ 3.4

ตารางที่ 3.4 ชุดข้อมูลสำหรับวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง

BTS	**Software/Product**	**จำนวนรายงาน**	**Meta-bug**	**Dependent bug report**
Bugzilla	Mozilla Firefox	4,781	200	4,581

3.2 ภาพรวมในการดำเนินงานวิจัย

ในส่วนนี้เป็นการแสดงภาพรวมในการดำเนินงานวิจัย โดยสามารถแสดงได้ดังรูปที่ 3.4



รูปที่ 3.4 ภาพรวมในการดำเนินงานวิจัย

จากรูปที่ 3.4 จะเห็นภาพรวมในการดำเนินงานวิจัย ซึ่งมี 4 ขั้นตอนหลักๆ ซึ่งอธิบายได้ดังนี้

1. ส่วนของการเตรียมขั้นต้น (Preliminary): การสร้างรูปแบบกลุ่มคำนามแบบสองคำ (Modelling of Bigram-based Noun group)

เป็นส่วนของการสร้างรูปแบบกลุ่มคำนามแบบสองคำ โดยมีเครื่องมือหลักในการสร้างรูปแบบกลุ่มคำนามแบบสองคำ คือ POS tagging และห่วงโซ่มาร์คอฟ ซึ่งรูปแบบกลุ่มคำนามแบบสองคำจะถูกเรียกในงานวิจัยนี้ว่า *“กฎเชิงไวยากรณ์”*

2. ส่วนของการวิเคราะห์คุณลักษณะเพื่อการใช้งานผ่านงานด้านการจำแนกข้อมูล (Feature analysis for usage through text classification tasks)

เนื่องจากคุณลักษณะที่ถูกใช้ในงานวิจัยทางด้านรายงานจุดบกพร่องนั้นมีหลายรูปแบบและไม่เคยมีงานวิจัยใดบ่งบอกว่าคุณลักษณะใด เป็นคุณลักษณะที่เหมาะสมกับงานวิจัยทางด้านรายงานจุดบกพร่อง ดังนั้นในงานวิจัยนี้จึงได้ศึกษาว่าคุณลักษณะใดน่าจะเหมาะสมในรายงานเรื่องนี้ โดยคุณลักษณะที่ศึกษาเชิงเปรียบเทียบ เป็นคุณลักษณะที่เคยมีการใช้ในงานวิจัยทางด้านรายงานจุดบกพร่องก่อนหน้า [10, 11, 18, 19, 41, 42, 73] ซึ่งการศึกษาเพื่อการวิเคราะห์คุณลักษณะสำหรับการเลือก คุณลักษณะผ่านงานด้านการจำแนกข้อมูล 2 งานคือ การตรวจจับรายงาน

จุดบกพร่องแบบ bug และ non-bug (Misclassification or Identification of bug and bon-bug reports) และการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง (Severity prediction)

3. ส่วนของการปรับค่าพารามิเตอร์ (Parameter tuning)

เป็นการปรับค่าพารามิเตอร์เพื่อการใช้งานในฟังก์ชันการค้นคืน (Retrieval function: REP) ซึ่งเป็นเทคนิคอย่างหนึ่งที่จะใช้ในขั้นตอนถัดไป

4. ส่วนของการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ (Assembling of dependent bug reports)

เป็นการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อของแต่ละ Meta-bug เข้ามารวมกัน เพราะแสดงให้เห็นว่ารายงานจุดบกพร่องใดบ้างที่เป็นส่วนต่อของ Meta-bug นั้นๆ โดยมีเทคนิคที่ศึกษาคือ

(1) อัลกอริทึมการเรียนรู้ของเครื่องในตระกูล Clustering ซึ่งในที่นี้ทำการศึกษา 2 อัลกอริทึมคือ การจัดกลุ่มแบบเคมีน (*K*-means clustering) และการจัดกลุ่มแบบเคมีนทรงกลม (Spherical *k*-means clustering)

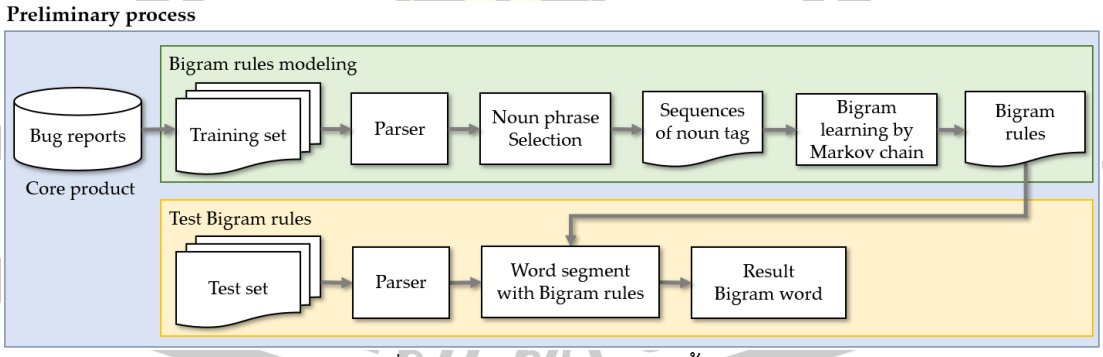
(2) เทคนิคการวิเคราะห์ความคล้ายคลึงของข้อความ (Text similarity) ซึ่งในที่นี้ทำการศึกษา 3 เทคนิคคือ การวิเคราะห์ความคล้ายคลึงแบบโคไซน์ (Cosine similarity) การวิเคราะห์ความคล้ายคลึงแบบ BM25 และการวิเคราะห์ความคล้ายคลึงแบบ MATF

(3) การวิเคราะห์ด้วยฟังก์ชันการค้นคืน (Retrieval function: REP)

โดยรายละเอียดของแต่ละขั้นตอนในการทำงาน จะแสดงรายละเอียดในลำดับต่อไป

3.3 การเตรียมขั้นต้น (Preliminary process)

กระบวนการเตรียมขั้นต้นที่จำเป็นต้องดำเนินการก่อนในการวิจัยนี้ คือ การสร้างกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ โดยอาศัยการเรียนรู้ด้วยห่วงโซ่มาร์คอฟ ทั้งนี้เพื่อจะนำไปใช้ในขั้นตอนการตัดคำเพื่อให้ได้กลุ่มคำนามแบบสองคำ (Bigram) ซึ่งสามารถแสดงขั้นตอนได้ดังรูปที่ 3.5



รูปที่ 3.5 กระบวนการเตรียมขั้นต้น

จากรูปที่ 3.5 รายงานจุดบกพร่องจาก Mozilla ในส่วนของ Core product จะถูกแบ่งเป็นข้อมูลเป็นสองชุดคือ ชุดสอน (Training set) 1,000 รายงาน เพื่อดำเนินการสร้างกฎเชิงไวยากรณ์ ส่วนอีกชุดคือ ชุดทดสอบ (Test set) จำนวน 300 รายงาน เพื่อใช้ทดสอบกฎเชิงไวยากรณ์ของกลุ่มคำแบบสองคำ ที่ได้จากการเรียนรู้ จากรูปจะมีการดำเนินการระบุหน้าที่ของคำในประโยค ซึ่งใน

ที่นี่ได้ใช้ Parser เพื่อแสดงหน้าที่ของคำในประโยคโดยอาศัยการแจงประโยคด้วย Stanford Parser ซึ่งสามารถแสดงตัวอย่างขั้นตอนการระบุหน้าที่ของคำในประโยคได้ตารางที่ 3.5

จากตารางที่ 3.5 จะเห็นว่าข้อมูล Summary จากรายงานจุดบกพร่องหมายเลข 1480518 มีข้อความว่า *"Add Top Sites Search Shortcut UI to AS Top Sites"* สามารถสกัดกลุ่มคำนาม (Noun phrase) ในประโยคได้สองกลุ่มคือ [Top/JJ, Sites/NNP, Search/NNP, Shortcut/NNP, UI/NNP] และ [AS/NNP, Top/NNP, Sites/NNP] หลังจากนั้นจะนำกลุ่มคำนามที่สกัดได้ทั้งหมดจากรายงานจุดบกพร่อง 500 รายงาน ไปประมวลผลเพื่อหากฎเชิงไวยากรณ์ของกลุ่มคำแบบสองคำ โดยอาศัยการเรียนรู้ด้วยห่วงโซ่มาร์คอฟ ซึ่งสามารถแสดงได้ตามตัวอย่างในตารางที่ 3.6

ตารางที่ 3.5 การแจงประโยคด้วย Stanford Parser

BugID 1480518	Add Top Sites Search Shortcut UI to AS Top Sites
Tagging	Add/VB, Top/JJ, Sites/NNP, Search/NNP, Shortcut/NNP, UI/NNP, to/TO, AS/NNP, Top/NNP, Sites/NNP
Parser	(ROOT (S (VP (VB Add) (NP (JJ Top) (NNP Sites) (NNP Search) (NNP Shortcut) (NNP UI)) (PP (TO to) (NP (NNP AS) (NNP Top) (NNP Sites))))))
Noun phrase Selected	[Top/JJ, Sites/NNP, Search/NNP, Shortcut/NNP, UI/NNP], [AS/NNP, Top/NNP, Sites/NNP]

ตารางที่ 3.6 ตัวอย่างประโยคที่ระบุหน้าที่ของคำเพื่อสร้างกฎเชิงไวยากรณ์ของกลุ่มคำแบบสองคำ

ลำดับ	**แสดงประโยค/หน้าที่ของคำในประโยค**
1	top sites search shortcut ui
	JJ NNP NNP NNP NNP
2	as top sites
	NNP NNP NNP
3	privileged content process
	JJ NN NN
4	the page action menu
	DT NN NN NN
5	a better time
	DT JJR NN

จากนั้นดำเนินการหาค่าความน่าจะเป็นการเปลี่ยนแปลง (Transition probabilities) ของ tag ที่แสดงหน้าที่ของคำด้วยสมการ (3.1) เพื่อสร้างเป็นไดอะแกรมสถานะของห่วงโซ่มาร์คอฟ ซึ่งจะได้ค่าดังแสดงในตารางที่ 3.7 และสามารถแสดงกฎเชิงไวยกรณ์ของกลุ่มคำแบบสองคำ ได้ดังตารางที่ 3.8

$$P(w_i|w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})} \quad (3.1)$$

ตารางที่ 3.7 แสดงความน่าจะเป็นแบบ Transition ของ tag หน้าที่ของคำ

P(NNP | JJ)	= 1/5	P(NN | DT)	= 1/5
P(NNP | NNP)	= 5/5	P(JJR | DT)	= 1/5
P(NN | JJ)	= 1/5	P(NN | JJR)	= 1/5
P(NN | NN)	= 3/5		

ตารางที่ 3.8 ตัวอย่างกฎเชิงไวยากรณ์สำหรับสกัดกลุ่มคำนามแบบสองคำ

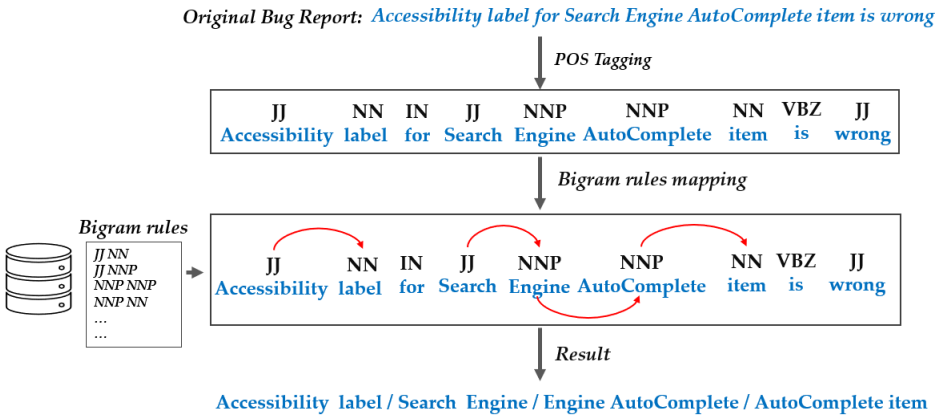
Rule No.	Rule	Definition
1	JJ + NN	Adjective + Noun
2	JJR + NN	Adjective (comparative) + Noun
3	JJR + NNS	Adjective (comparative) + Noun (plural)
4	JJ + NNS	Adjective + Noun (plural)
5	NN + NN	Noun + Noun

หลังจากที่การสร้างกฎด้านไวยากรณ์สำหรับคำนามด้วยจำนวน 1,000 เอกสาร ซึ่งมีจำนวนประโยคทั้งสิ้น 2,503 ประโยค และสกัดกลุ่มคำนามเพื่อใช้ในการเรียนรู้ด้วยห่วงโซ่มาร์คอฟได้ 4,845 กลุ่มคำนาม และผลการเรียนรู้ได้จำนวนกฎเชิงไวยากรณ์สำหรับคำนามแบบสองคำทั้งสิ้น 57 กฎ ดังตารางที่ 3.9

ตารางที่ 3.9 กฎเชิงไวยากรณ์สำหรับคำนามแบบสองคำ

NN NN	JJ NN	NNP NNP	JJ NNS	NN NNS	NNP NN
VB NN	VB JJ	RB VBN	CD NN	RB VBN	NN RB
VB RB	VB VBN	VB NNS	NN NNP	JJR NN	JJ JJ
NN VBG	NN VBN	VBN JJ	NNS RB	VBG NN	CD NNS
VBG JJ	RB VBG	VB NNP	NN VB	NNS NN	NNS VBD
VBN NN	VBN RB	RB NN	NNS VBG	JJ NNP	NN JJR
JJR NNS	NNS VBN	NN JJ	NNS VBZ	RBR JJ	VBN NNS
NNS JJ	VBP NNP	VBG NNP	VBD NN	VBG NNS	VBD JJ
VB VBG	VBZ NNS	VBD NNS	VBZ NNP	NNS NNS	JJ RB
NNS NNP	JJ VB				

เมื่อได้ผลการสกัดกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ แล้ว ขั้นตอนต่อไปคือการนำกฎที่ได้ไปทดสอบในการตัดคำแบบกลุ่มคำนามแบบสองคำ กับรายงานจุดบกพร่องจำนวน 100 รายงาน เพื่อหากฎที่ถูกใช้จริงแล้วจึงนำกฎที่ได้นั้นไปใช้งานในการตัดคำเป็นกลุ่มคำแบบสองคำต่อไป ซึ่งผลลัพธ์ของกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ ที่ใช้ในงานวิจัยนี้ได้แสดงไว้ในหัวข้อที่ 4.1 สำหรับการนำกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำไปใช้งานสามารถแสดงดังรูปที่ 3.6



รูปที่ 3.6 ตัวอย่างการนำกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำไปใช้งาน

จากรูปที่ 3.6 เป็นการแสดงการตัดคำด้วยการใช้กฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ ซึ่งอธิบายขั้นตอนพอสังเขปได้ดังต่อไปนี้ (1) นำประโยคที่ต้องการตัดคำด้วยกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำไประบุหน้าที่ของคำในประโยค (2) ใช้กฎเชิงไวยากรณ์กลุ่มคำแบบสองคำเทียบกับหน้าที่ของคำในประโยคแบบไบแกรม (Bigram rules mapping) (3) ผลลัพธ์ที่ได้คือคำที่เป็นกลุ่มคำแบบสองคำ

3.4 การวิเคราะห์คุณลักษณะของรายงานจุดบกพร่องเพื่อการใช้งาน (Feature Analysis of Bug reports for Usage)

ในขั้นตอนนี้จะแสดงการศึกษาว่าคุณลักษณะย่อยที่สุดของรายงานจุดบกพร่องที่มีความหมายนั่นคือ “คำ” หรือ “กลุ่มคำ” ที่สามารถใช้เป็นคุณลักษณะที่เหมาะสมกับงานวิจัยนี้คืออะไร โดยจะเป็นการศึกษาผ่านการวิจัยด้านรายงานจุดบกพร่องใน 2 ประเด็นหลัก คือ การระบุรายงานจุดบกพร่องว่าเป็น bug หรือ non-bug และการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง โดยงานวิจัยนี้ได้ทำการศึกษาคุณลักษณะของรายงานจุดบกพร่อง 7 ประเภท ดังต่อไปนี้

(1) คำเดี่ยว (Unigram)
(2) กลุ่มคำแบบสองคำ (Bigram)
(3) คำจากกลุ่มคำผสม (Compound words)
(4) คำเดี่ยวร่วมกับกลุ่มคำแบบสองคำ (Unigram and bigram)
(5) คำเดี่ยวร่วมกับคำจากกลุ่มคำผสม (Unigram and compound words)
(6) กลุ่มคำแบบสองคำร่วมกับคำจากกลุ่มคำผสม (Bigram and compound words)

(7) คำเดียวร่วมกับกลุ่มคำแบบสองคำและคำจากกลุ่มคำผสม
(Combination: unigram and bigram and compound word)

คุณลักษณะรายงานจุดบกพร่องทั้ง 7 ประเภทอยู่บนพื้นฐานของคำใน 3 รูปแบบด้วยกันคือ (1) Unigram (2) Compound words และ (3) Bigram ซึ่ง Bigram จะเป็นการตัดคำด้วยกฎเชิงไวยากรณ์เพื่อให้ได้กลุ่มคำแบบสองคำ

ซึ่งต่อไปนี้จะเป็นการอธิบาย การตัดคำแบบ Unigram และ การตัดคำจากกลุ่มคำผสมหรือ Compound words ส่วนการตัดคำเป็น Bigram ด้วยกฎเชิงไวยากรณ์นั้น ได้นำเสนอไว้แล้วในหัวข้อที่ 3.2 และรูปที่ 3.6

1) การตัดคำแบบคำเดี่ยว (Unigram)

การตัดคำแบบ Unigram คือการนำประโยคข้อความที่ปรากฏในส่วนของ Summary และ Description ของเอกสารรายงานจุดบกพร่องผ่านการประมวลผลเพื่อแบ่งแยกออกเป็นคำเดี่ยวด้วยการอาศัยตัดเป็นคำเดี่ยวจากการเว้นรรค (White space) ในประโยค โดยผลลัพท์ที่ได้จะเป็นการตัดคำแบบคำเดี่ยวดังแสดงได้ในตารางที่ 3.10

ตารางที่ 3.10 การตัดคำแบบคำเดี่ยว

Summary's bug report	"Accessibility label for Search Engine AutoComplete item is wrong"
Tokenization	Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong

ดังนั้นจากรายงานจุดบกพร่องที่มีข้อมูล Summary คือ "*Accessibility label for Search Engine AutoComplete item is wrong*" จะได้ผลลัพท์ของการตัดคำแบบ Unigram เป็น Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong

2) การตัดคำจากกลุ่มคำผสม (Compound words)

เอกสารรายงานจุดบกพร่องทั้งในส่วนของ Summary และ Description บ่อยครั้งที่มักจะปรากฏคำที่มีการประสมกันของคำตั้งแต่สองคำ ยกตัวอย่างเช่น "UrlbarInput" ," URL-bar", "browser_social_activation" เป็นต้น ซึ่งคำลักษณะนี้เรียกว่ากลุ่มคำผสม (Compound words) [42] หรือบางงานวิจัยเรียกว่า "Camel Case" [41] หรือเรียกว่า "Concatenation of words" [40] โดยในงานวิจัยเหล่านั้นได้นำกลุ่มคำผสมเหล่านี้มาแยกเป็น Unigram ดังแสดงตัวอย่างในตารางที่ 3.11

ตารางที่ 3.11 ตัวอย่างการตัดคำจากกลุ่มคำผสมหรือ Compound word

ตัวอย่างคำผสม	**ผลลัพธ์จากการตัดคำจากกลุ่มคำผสม**
UrlbarInput	UrlbarInput / Url / bar / Input
URL-bar	URL-bar / URL / bar
browser_social_activation	browser_social_activation / browser / social / activation

อย่างไรก็ตาม คำที่เป็นคำผสมต้นฉบับ ก็ยังคงถูกนำมาใช้เป็นคุณลักษณะของรายงานจุดบกพร่อง เพียงแต่ในบางรายงานนั้นพบว่ามีจำนวนคำน้อย ดังนั้นการแยกคำที่อยู่ใน Compound words ออกมา ก็คือการขยายคำ (keyword expansion) ซึ่งการกระทำดังกล่าวเป็นการเพิ่มโอกาสในการค้นหานั่นเอง

ยกตัวอย่างเช่น ในรายงานจุดบกพร่องตัวหนึ่งมีข้อความคือ *"Accessibility label for Search Engine AutoComplete item is wrong"* จะเห็นว่าในข้อความดังกล่าวมี Compound words อยู่ 1 คำ คือคำว่า "AutoComplete" ซึ่งเมื่อดำเนินการตัดคำออกจาก Compound words คำนี้ ก็จะได้คำทั้งหมดคือ "AutoComplete/ Auto/ Complete" และจากประโยคตัวอย่างนี้สามารถแสดงตัวอย่างการตัดคำเพื่อให้ได้คุณลักษณะของรายงานจุดบกพร่องใน 7 รูปแบบดังตารางที่ 3.12

ตารางที่ 3.12 ตัวอย่างคุณลักษณะของรายงานจุดบกพร่องที่แตกต่างกันทั้ง 7 รูปแบบ

ประเภทคุณลักษณะ	**ผลลัพธ์การทำคุณลักษณะ**
Unigram	Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong
Bigram	Accessibility label/ Search Engine/ Engine AutoComplete/ AutoComplete item
Compound word	AutoComplete/ Auto/ Complete
Unigram and bigram	Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong/ Accessibility label/ Search Engine/ Engine AutoComplete/ AutoComplete item
Unigram and compound word	Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong/ Auto/ Complete
Bigram and compound word	Accessibility label/ Search Engine/ Engine AutoComplete/ AutoComplete item/ AutoComplete/ Auto/ Complete
Combination	Accessibility/ label/ for/ Search/ Engine/ AutoComplete/ item/ is/ wrong/ Accessibility label/ Search Engine/ Engine AutoComplete/ AutoComplete item/ Auto/ Complete

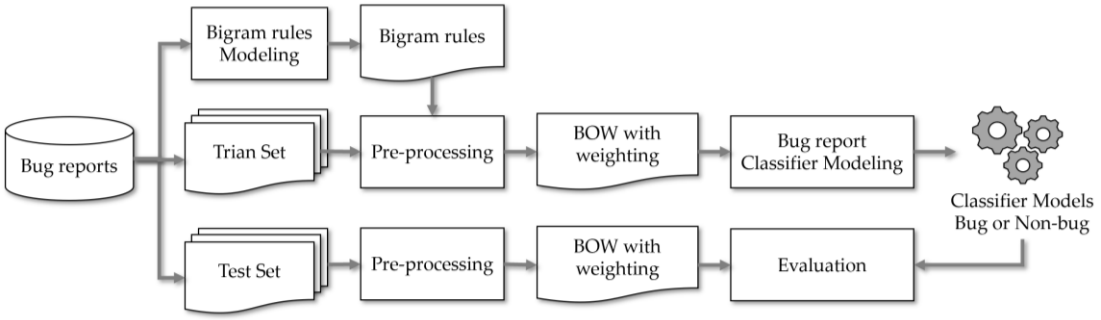
3.4.1 การพิจารณาเพื่อเลือกคุณลักษณะของรายงานจุดบกพร่องผ่านการศึกษาด้านการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug (Identifying of bug and non-bug report)

การตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug เป็นการศึกษาเพื่อวิเคราะห์หรือพิจารณาว่ารายงานจุดบกพร่องที่มีการรายงานเข้ามานั้นเป็นรายงานจุดบกพร่องจริงๆ (Real bug report) หรือไม่ [19, 37-39, 85] เพราะมีรายงานจุดบกพร่องที่ถูกเข้ามาจำนวนมากเป็นการบอกเล่าการใช้งาน การแนะนำ เป็นต้น

โดยผู้วิจัยได้ศึกษาและนำเสนอผลวิจัยในปี 2019 [86] เพื่อคัดเลือกคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสม ซึ่งข้อมูลที่นำมาใช้ในการศึกษานี้ส่วนหนึ่งเป็นชุดข้อมูลมาตรฐานที่เรียกว่า Herzig's Dataset [37] โดยมีจำนวนรายงานจุดบกพร่องทั้งสิ้น 7,401 รายงาน และอีกหนึ่งชุดเป็นข้อมูลรายงานจุดบกพร่องจากมอซิลลา Firefox ดาวน์โหลดเมื่อวันที่ 1 ตุลาคม 2560 จำนวน

4,300 รายงาน ซึ่งจะเลือกเฉพาะสถานะที่เป็น “VERIFED” และ “CLOSE” เพราะสถานะดังกล่าวจะบ่งบอกว่ารายงานเหล่านั้นเป็นรายงานจุดบกพร่องจริงๆ

โดยข้อมูลรายงานจุดบกพร่องทั้งหมดนั้น จะมีการแบ่งออกเป็น 2 ส่วนทั้งสองชุดข้อมูล คือ ส่วนสำหรับการเรียนรู้ (Training set) จำนวน 4,000 รายงาน และส่วนสำหรับการทดสอบ (Test set) จำนวน 600 รายงาน จากนั้นทำการศึกษาคุณลักษณะที่เป็นไปได้ 7 รูปแบบ ดังที่ได้กล่าวไว้ในหัวข้อ 3.4 ซึ่งกระบวนการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug สามารถแสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 กระบวนการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug

ภายหลังจากการตัดคำ และแสดงรายงานจุดบกพร่องในรูปแบบของถุงคำที่มีการให้น้ำหนักคำแบบ *tf* เนื่องจากในงานวิจัยก่อนหน้า [19, 38] ได้ให้ความเห็นว่าเทคนิคการให้น้ำหนักแบบ *tf* เป็นการให้น้ำหนักแบบภายใน (Local weight) จึงเน้นความสำคัญของคำเฉพาะในพื้นที่เอกสารนั้นๆ ทำให้สามารถแสดงค่าน้ำหนักของ “คำ” ในแต่ละเอกสารได้แตกต่างกัน ซึ่งค่าที่แตกต่างกันนี้เป็นการบอกว่า “คำ” คำนั้นในเอกสารมีความสำคัญต่างกันอย่างไร

สำหรับอัลกอริทึมที่ใช้ในการสร้างตัวจำแนกรายงานจุดบกพร่อง (Bug report classifiers) จะมีอยู่ 3 อัลกอริทึมได้แก่ มัลติโนเมียลนาอีฟเบย์ (Multinomial naïve bayes: MNB) การถดถอยโลจีสติก (Logistic regression: LR) และซัพพอร์ตเวกเตอร์แมชชีน (Support vector machine: SVM)

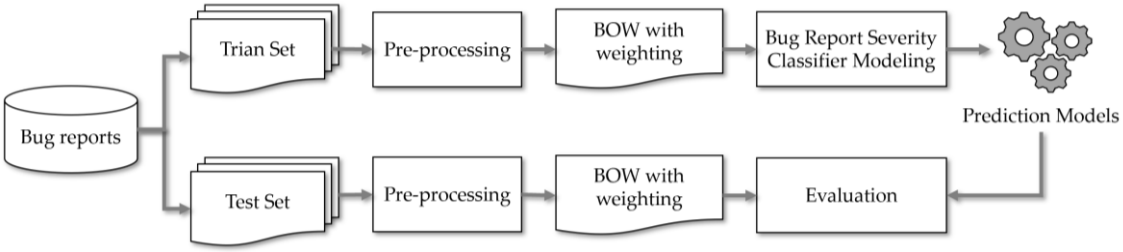
เมื่อประเมินผลการทดลองของแต่ละกระบวนการในการจำแนกรายงานจุดบกพร่องออกเป็น bug และ non-bug ด้วยค่าความระลึก ค่าความแม่นยำ และค่า F1 พบว่าคุณลักษณะที่เหมาะสมและใช้งานได้ดีกับทุกอัลกอริทึมการเรียนรู้แบบมีผู้สอนคือ Unigram + compound words และ Combination แต่หากใช้คุณลักษณะแบบ Combination จะใช้เวลาในการเตรียมข้อมูลนานที่สุด (ผลการทดลองแสดงในหัวข้อที่ 4.2.1)

3.4.2 การศึกษาเพื่อเลือกคุณลักษณะของรายงานจุดบกพร่องผ่านการศึกษาด้านการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง

สำหรับการศึกษาเพื่อคัดเลือกคุณลักษณะของรายงานจุดบกพร่องผ่านการทำนายระดับความรุนแรงของรายงานจุดบกพร่องนั้น จะใช้รายงานจุดบกพร่องที่ได้ดาวน์โหลดจากมอซิลลา Firefox เมื่อวันที่ 1 ตุลาคม 2560 โดยจะเลือกเอาเฉพาะรายงานจุดบกพร่องที่มีสถานะเป็น

“VERIFIED” และ “CLOSE” เพราะเป็นการยืนยันว่าเป็นรายงานจุดบกพร่องที่ไม่ใช่รายงานจุดบกพร่องที่ซ้ำซ้อน และไม่ใช่รายงานจุดบกพร่องแบบ non-bug [87]

ซึ่งรายงานจุดบกพร่องที่ใช้ในงานวิจัยส่วนนี้เป็นรายงานที่ถูกรายงานเข้ามาในระบบ BTS ของ Mozilla ในระหว่างปี 2543 – 2560 โดยจะใช้รายงานจุดบกพร่องจำนวนทั้งสิ้น 4,000 รายงาน จากนั้นแบ่งเป็นชุดสอน (Training set) จำนวน 3,000 รายงาน และชุดทดสอบ (Test set) จำนวน 1,000 รายงาน โดยกระบวนการที่นำเสนอสามารถแสดงได้ดังรูปที่ 3.8



รูปที่ 3.8 กระบวนการศึกษาการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง

ในการเตรียมข้อมูลของงานวิจัยนี้ก็ได้ทำการศึกษาเชิงเปรียบเทียบคุณลักษณะที่เป็นไปได้ของรายงานจุดบกพร่องและได้รับการประยุกต์ใช้งานวิจัยก่อนหน้าคือ [16, 19, 40, 41] และใช้วิธีการให้น้ำหนักใน 3 วิธี ได้แก่ *tf*, *tf-idf*, และ *tf-igm*

สำหรับการสร้างตัวจำแนกรายงานจุดบกพร่องสำหรับการทำนายหรือการจำแนกระดับความรุนแรงของรายงานจุดบกพร่องนั้น จะใช้อัลกอริทึมการเรียนรู้แบบมีผู้สอน 4 อัลกอริทึมได้แก่ มัลติโนเมียลนาอีฟเบย์ เพื่อนบ้านที่ใกล้ที่สุด (K-nearest neighbors: KNN) ซัพพอร์ตเวกเตอร์แมชชีน และป่าสุ่ม (Random forest: RF)

เมื่อประเมินผลการทดลองของแต่ละกระบวนการในการทำนายระดับความรุนแรงของรายงานจุดบกพร่องด้วยค่าความระลึก ค่าความแม่นยำ และค่า F1 พบว่าคุณลักษณะที่เหมาะสมและใช้งานได้ดีกับทุกอัลกอริทึมคือ Unigram+ compound words (ผลการทดลองแสดงในหัวข้อที่ 4.2.2) ซึ่งเป็นผลลัพธ์ที่สอดคล้องการการศึกษาในหัวข้อ 3.4.1

นอกจากนี้ หากพิจารณาถึงผลการทำนายระดับความรุนแรงของรายงานจุดบกพร่องนั้น พบว่าตัวจำแนกที่สร้างจากอัลกอริทึมซัพพอร์ตเวกเตอร์แมชชีน โดยมีการให้น้ำหนักคำในถุงคำแบบ *tf-igm* จะให้ผลลัพธ์ที่น่าพอใจที่สุด เพราะ *tf-igm* จะเป็นการพิจารณาค่า global weight เฉพาะคลาสหรือเฉพาะกลุ่ม เพราะฉะนั้น “คำ” คำหนึ่งจะมีความสำคัญที่แตกต่างกันในแต่ละคลาส ทำให้ลดความคลุมเครือของการจัดกลุ่ม

3.5 ตัวอย่างแสดงการคำนวณการให้น้ำหนักแบบ *tf*, *tf-idf*, *BM25*, *MATF* และ *tf-igm*

สำหรับขั้นตอนการเตรียมข้อมูลจะประกอบไปด้วย การตัดคำ การตัดคำหยุด การหารูปแบบพื้นฐานร่วมของคำศัพท์ ภายหลังเมื่อผ่านขั้นตอนการเตรียมข้อมูลแล้ว รายจุดบกพร่องจะถูกแสดงในรูปแบบโมเดลเชิงพื้นที่ หรือที่เรียกว่า “ถุงคำ (Bag of words: BOW)” (ซึ่งจะนำเสนอในหัวข้อถัดไป) โดยคำแต่ละคำในถุงคำจะมีการให้น้ำหนัก ซึ่งในงานวิจัยนี้เป็นการศึกษาเชิงเปรียบเทียบ

ใน 4 เทคนิคการให้น้ำหนักคือ *tf, tf-idf, BM25* และ *MATF* โดยสมมติมีรายงานจุดบกพร่อง 4 รายงานที่ผ่านการประมวลผลข้างต้นมาแล้วดังนี้

B_1: implement / translat / infobar
B_2: implement / linux / style / translat / infobar
B_3: implement / window / style / translat / infobar
B_4: set / right / maxresult / urlbarinput

3.5.1 ตัวอย่างการให้น้ำหนักแบบ *tf*

tf เป็นการให้น้ำหนักแบบที่นิยมนำมาใช้ในงานวิจัยด้านรายงานจุดบกพร่องในโดยเฉพาะงานวิจัยสำหรับตรวจสอบการทำซ้ำของรายงานจุดบกพร่อง [10, 11, 86] โดยสมการ *tf* เป็นดังนี้

$$tf_{t,d} = log(1 + freq(t,d))$$

จากข้อมูลรายงานจุดบกพร่อง B_1 จะเห็นว่าพบคำ "*implement*" จำนวน 1 ครั้ง ดังนั้นจะคำนวณค่า *tf* ของคำว่า "*implement*" ได้ดังนี้ ตัวอย่าง หากจะหาค่า *tf* ของคำ "*implement*" ใน B_1 ดังนั้นสามารถคำนวณตามสมการข้างต้นได้ดังนี้

$$tf_{implement,\,B_1} = log(1+1) = 0.301$$

3.5.2 ตัวอย่างการให้น้ำหนักแบบ *tf-idf*

โดยทั่วไป *tf-idf* จะใช้สมการต่อไปนี้

$$tf - idf_{t,d} = log(1 + freq(t,d)) \times log\left(\frac{N}{df_t}\right)$$

สมมติพิจารณาคำว่า "*implement*" จะเริ่มจากการหาค่า $idf_{implement}$ จะเห็นว่าจำนวนรายงานจุดบกพร่องมีทั้งหมด (*N*) มีทั้งสิ้น 4 รายงาน และมีรายงานจุดบกพร่องที่พบคำว่า "*implement*" ทั้งหมด ($df_{implement}$) จำนวน 3 รายงาน ดังนั้นค่า $idf_{implement}$ คือ

$$idf_{implement} = log(4/3)$$

และเมื่อนำมาหาค่า $tf\text{-}idf_{implement}$, B_1 จะมีคำนวณค่าได้ดังนี้

$$tf - idf_{implement,B_1} = log(1+1) \times log(4/3)$$

$$tf - idf_{implement,B_1} = 0.038$$

3.5.3 ตัวอย่างการให้น้ำหนักแบบ *BM25*

การใช้เทคนิค *BM25* ในงานวิจัยที่เกี่ยวข้องกับรายงานจุดบกพร่องนั้นได้มีการนำมาประยุกต์ใช้ในงานตรวจสอบการทำซ้ำของรายงานจุดบกพร่อง [73, 88] ด้วยการวัดความคล้ายกันของรายงานจุดบกพร่องที่เปรียบเทียบกัน ซึ่งในงานวิจัยนี้ได้ประยุกต์สมการ (3.2) เพื่อให้น้ำหนักคำแบบ *BM25* ได้ดังสมการต่อไปนี้

$$BM25(t,d) = log\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \times \left(\frac{tf(t,d) \times (k_1 + 1)}{tf(t,d) + k_1 \times \left(1 - b + b \times \left(\frac{|d|}{dl_{avg}}\right)\right)}\right) \quad (3.2)$$

จากสมการ (3.2) สามารถแทนค่าต่างๆ t คือคำว่า "*implement*" ดังนั้น N คือ 4 หมายถึง จำนวนรายงานจุดบกพร่องทั้งหมด df_t คือจำนวนเอกสารที่มีคำ t ปรากฏซึ่งเท่ากับ 3 ในขณะที่ $tf(t,d)$ ในที่นี้คือความถี่ของคำ t ที่ปรากฏในเอกสาร B_1 มีค่าเท่ากับ 1 ค่าพารามิเตอร์ k_1 และ b ที่ใช้ควบคุมการให้ความสำคัญกับเอกสารที่มีความสั้นยาวแตกต่างกันมีค่าเป็น 2.0 และ 0.5 ตามลำดับ ส่วน $|d|$ คือค่าความยาวของรายงานจุดบกพร่อง B_1 มีค่าเท่ากับ 3 และสุดท้ายค่า dl_{avg} คือ ค่าความยาวเฉลี่ยของรายงานจุดบกพร่องทั้งหมดมีค่าเท่ากับ 4.25 เมื่อแทนค่าในสมการ (3.2) ได้ผลลัพธ์ดังนี้

$$BM25_{implement,B_1} = log\left(\frac{4 - 3 + 0.5}{3 + 0.5}\right) \times \left(\frac{1 \times (2 + 1)}{1 + 2 \times \left(1 - 0.5 + 0.5 \times \frac{3}{4.25}\right)}\right)$$

$$BM25_{implement,B_1} = -0.314$$

3.5.4 ตัวอย่างการให้น้ำหนักแบบ *MATF*

ในการหาค่าน้ำหนักด้วย *MATF* สามารถประยุกต์จากสมการ 2.24 ซึ่งได้สมการในการให้น้ำหนักคำได้เป็น

$$MATF_{(t,d)} = TFF(t,d) \times TDF(t,C) \quad (3.3)$$

โดยที่ *TFF*(*t, d*) คือการรวมปัจจัย 2 ปัจจัยเข้าด้วยกัน คือ Relative intra-document (RITF) ปัจจัยที่สองคือ Length regularized TF (LRTF) ส่วน *TDF*(*implement, B_1*) คือ การพิจารณาการให้น้ำหนักคำที่หาได้ยากในคลังรายงานจุดบกพร่อง ซึ่งจะต้องคำนวณหาค่าความถี่เอกสารผกผัน (IDF) และค่าเฉลี่ยตามความถี่ของคำ (The average elite set term frequency: AEF) ซึ่งจะแสดงตัวอย่างการหาค่าน้ำหนักของคำว่า *implement* จากเอกสารรายงานจุดบกพร่อง B_1 ดังนั้นจากสมการ (3.3) สามารถแสดงการประยุกต์ใช้เพื่อคำนวณหาค่าน้ำหนักของคำว่า *implement* จากเอกสาร B_1 ได้ดังนี้

$$MATF_{implement,B_1} = TFF(implement, B_1) \times TDF(implement, C)$$

$$TFF(implement, B_1) = w \times BRITF(implement, B_1) + (1 - w) \times BLRTF(implement, B_1)$$

จากสมการข้างต้น จะเห็นว่าในการหาค่า *TFF*(*implement, B_1*) จะต้องหาค่า w, *BRITF*(*implement, B_1*) และ *BLRTF*(*implement, B_1*) ก่อน ดังนั้นเริ่มต้นหาค่าของ w โดยที่ $|Q|$ ซึ่งในที่นี้หมายถึงความยาวของเอกสาร B_1 และมีค่าเท่ากับ แทนค่าในสมการต่อไปนี้

$$w = \frac{2}{1 + log_2(1 + |Q|)} = \frac{2}{1 + log_2(1 + 3)}$$

$$w = 0.667$$

ต่อจากนั้นคำนวณหาค่า *BRITF*(*implement, B_1*) ซึ่งมีค่า *tf* คือความถี่ของคำ *implement* ที่ปรากฏในรายงานจุดบกพร่อง B_1 ซึ่งมีค่าเท่ากับ 1 และ *Avg.tf* คือค่าเฉลี่ยความถี่ของคำในรายงานจุดบกพร่อง B_1 มีค่าเท่ากับ 1 ดังนั้นจะคำนวณหาค่า *BRITF*(*implement, B_1*) ได้ดังนี้

$$BRITF(implement, B_1) = \frac{log_2(1 + tf(implement, B_1))/log_2(1 + Avg.tf(B_1))}{1 + (log_2(1 + tf(implement, B_1))/log_2(1 + Avg.tf(B_1)))}$$

แทนค่า *tf* และ *Avg.tf* ลงในสมการจะได้

$$BRITF(implement, B_1) = \frac{log_2(1 + 1)/log_2(1 + 1)}{1 + (log_2(1 + 1)/log_2(1 + 1))}$$

$$BRITF(implement, B_1) = 0.5$$

เมื่อได้ค่า *BRITF*(*implement, B_1*) จากนั้นดำเนินการคำนวณหาค่า (*implement, B_1*) โดยมีค่าที่ต้องพิจารณาคือ *tf* คือความถี่ของคำ *implement* ที่ปรากฏในรายงานจุดบกพร่อง B_1 ซึ่งมีค่าเท่ากับ 1 ขณะที่ *ADL(C)* คือ ค่าเฉลี่ยของความยาวเอกสารในคลังเอกสารในที่นี้คือ 4.25 และ *len(B_1)* คือ ความยาวของรายงานจุดบกพร่อง B_1 ในที่นี้คือ 3 นำไปแทนค่าลงในสมการต่อไปนี้เพื่อคำนวณหาค่า *BLRTF*(*implement, B_1*)

$$BLRTF(implement, B_1) = \frac{tf(implement, B_1) \times log_2(1 + ADL(C)/len(B_1))}{1 + (tf(implement, B_1) \times log_2(1 + ADL(C)/len(B_1)))}$$

เมื่อแทนค่า *tf, len*(B_1) และ *ADL*(*C*) ลงในสมการจะได้

$$BLRTF(implement, B_1) = \frac{1 \times log_2(1 + 4.25/3)}{1 + (1 \times log_2(1 + 4.25/3))}$$

$$BLRTF(implement, B_1) = 0.56$$

จากนั้นนำค่า *w*, *BRITF*(*implement, B_1*) และ *BLRTF*(*implement, B_1*) มาแทนค่าในสมการเพื่อหาค่า *TFF*(*implement, B_1*) จะได้ค่า *TFF*(*implement, B_1*) ดังต่อไปนี้

$$TFF(implement, B_1) = 0.667 \times 0.5 + (1 - 0.667) \times 0.56$$

$$TFF(implement, B_1) = 0.5204$$

จากนั้น ทำการคำนวณหาค่า *TDF*(*implement, B_1*) ได้จากสมการต่อไปนี้

$$TDF(implement, B_1) = IDF(implement, C) \times \frac{AEF(implement, C)}{1 + AEF(implement, C)}$$

จากนั้นหาค่า *IDF*(*implement*, *C*) ในการคำนวณค่า *IDF*(*implement*, *C*) จะต้องทราบค่า 2 ค่า คือ ค่า *CS*(*C*) และค่า *DF*(*implement*, *C*) โดยค่า *CS*(*C*) ซึ่งเป็นจำนวนรายงานจุดบกพร่องทั้งหมดในคลังเอกสารและมีค่าเท่ากับ 4 ขณะที่ค่า *DF*(*implement*, *C*) คือจำนวน

รายงานจุดบกพร่องที่มีคำ *implement* ปรากฏในคลังเอกสารและมีค่าเท่ากับ 3 เมื่อนำค่า *CS*(*C*) และค่า *DF*(*implement*, *C*) แทนในสมการเพื่อหาค่า *IDF*(*implement*, *C*)

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

$$IDF(t, C) = log\left(\frac{4 + 1}{3}\right) = 0.222$$

ต่อมาคำนวณหาค่า *AEF*(*implement*, *C*) โดยที่ *CTF*(*implement*, *C*) คือ จำนวนความถี่ที่คำ *implement* ปรากฏในคลังเอกสาร *C* ซึ่งมีค่าเท่ากับ 3 แล้วนำไปแทนค่าในสมการ

$$AEF(implement, C) = \frac{CTF(implement, C)}{DF(implement, C)}$$

$$AEF(t, C) = \frac{3}{3} = 1$$

จากนั้นนำกลับไปแทนค่าในสมการเพื่อหาค่า *TDF(implement, B_1)* ซึ่งจะได้เป็น

$$TDF(implement, B_1) = IDF(implement, C) \times \frac{AEF(implement, C)}{1 + AEF(implement, C)} = 0.222 \times \frac{1}{1 + 1}$$

$$TDF(implement, B_1) = 0.111$$

สุดท้ายนำค่า *TFF(implement, B_1)* และ *TDF (implement, B_1)* กลับไปคำนวณในสมการ (3.3) เพื่อคำนวณหาค่าน้ำหนักแบบ *MATF(implement, B_1)* และได้ผลลัพธ์ดังนี้

$$MATF_{implement,B_1} = TFF(implement, B_1) \times TDF(implement, C) = 0.5204 \times 0.111$$

$$MATF_{implement,B_1} = 0.058$$

ตารางที่ 3.13 เปรียบเทียบค่าการให้น้ำหนักคำทั้ง 4 เทคนิค

เทคนิค	**ค่าน้ำหนักของคำ “*implement*” ในเอกสาร B_1**
tf	0.301
tf-idf	0.038
BM25	-0.314
MATF	0.058

เนื่องด้วยในขั้นตอนการศึกษาเพื่อเลือกคุณลักษณะของรายงานจุดบกพร่องผ่านการศึกษาด้านการทำนายระดับความรุนแรงของจุดบกพร่อง ได้มีการศึกษาเทคนิคการให้น้ำหนักคำที่เหมาะกับการจำแนกข้อความได้เป็นอย่างดี ทั้งในแบบไบนารีคลาสและมัลติคลาส นั่นคือ การให้น้ำหนักคำแบบความถี่และแรงดึงดูดผกผัน จึงขออธิบายวิธีการคำนวนพอสังเขปดังนี้

3.5.5 ตัวอย่างการให้น้ำหนักคำแบบความถี่และแรงดึงดูดผกผัน (Term frequency – inverse gravity moment: *tf-igm*)

สำหรับการให้น้ำหนักคำแบบความถี่และแรงดึงดูดผกผันนั้น เป็นการให้น้ำหนักแบบมีผู้สอน โดยมีสมการดังนี้

$$tf - igm_{t,d} = f_{t,d} \times (1 + \lambda \times igm(t_k)) \quad (3.4)$$

โดยที่ $f_{t,d}$ คือ จำนวนของคำ t ที่ปรากฏในเอกสาร d สำหรับ λ คือค่าสัมประสิทธิที่ปรับค่าได้ระหว่าง 5.0-9.0

ในการคำนวณค่าน้ำหนักตามสมการ (3.4) ต้องมีชุดข้อมูลเพื่อทำการสอนเพื่อหาค่าแรงดึงดูดผกผัน (inverse gravity moment: *igm*) ของแต่ละคำเสียก่อนตามสมการ (3.5)

$$igm(t_k) = \frac{f_{k1}}{\sum_{r=1}^{m} f_{kr} \cdot r} \quad (3.5)$$

โดยที่ $igm(t_k)$ แทนแรงดึงดูดผกผันของการกระจายระหว่างคลาสของคำ t_k และ f_{kr} (r = 1, 2,...,m) เป็นความถี่ของการเกิดคำ t_k ในแต่ละคลาส ซึ่งเรียงลำดับจากมากไปน้อยโดยมี r เป็นลำดับ ส่วน f_{kr} คือจำนวนเอกสารที่คำ t_k ปรากฏในแต่ละคลาสที่ r

ดังนั้นหากมีคลาสที่ใช้ในการจำแนกเอกสารเป็น 5 คลาส และสมมติให้ในแต่ละคลาสมีเอกสารจำนวน 10 เอกสาร ซึ่งในตัวอย่างนี้กำลังพิจารณาคำว่า “implement” ในเอกสาร B_1 ที่มีคุณลักษณะของคำคือ [“implement”, “translat”, “infobar”] ลำดับแรกจะต้องหาว่าคำ $t_{implement}$ มีจำนวนเอกสารที่ปรากฏคำนี้ในแต่ละคลาสเป็นเท่าใด สมมติมีค่าเป็น {4, 8, 0, 0, 0}

ขั้นต่อมานำความถี่เอกสารที่พบคำ $t_{implement}$ ในแต่ละคลาสมาเรียงลำดับจากน้อยไปมากจะได้เป็น {8, 4, 0, 0, 0} จากนั้นคำนวณค่า $igm(t_{implement})$ ตามสมการ (3.5) จะได้

$$igm(t_{implement}) = \frac{8}{(8 \times 1) + (4 \times 2) + (0 \times 3) + (0 \times 4) + (0 \times 5)}$$

$$igm(t_{implement}) = \frac{8}{16} = 0.5$$

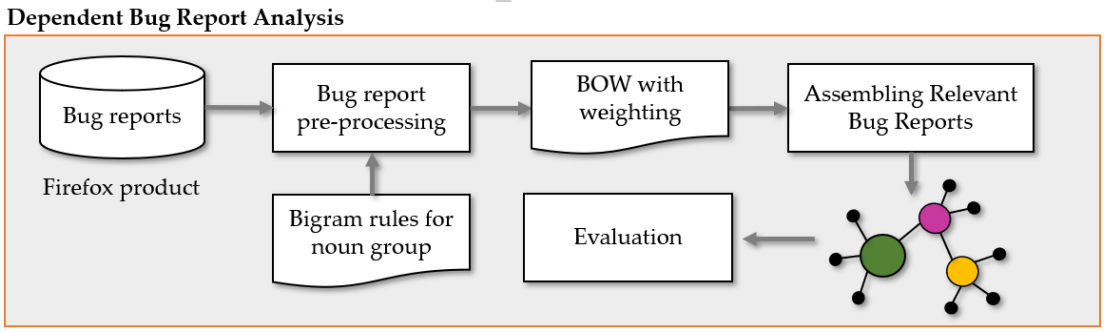
จากนั้นนำค่า $igm(t_{implement})$ กลับไปแทนค่าในสมการ (3.4) โดยกำหนดให้ค่าสัมประสิทธิ λ มีค่าเท่ากับ 7 จะได้ค่าน้ำหนักของคำ “*implement*” ดังนี้

$$tf - igm_{implement,B_1} = 1 \times (1 + 7 \times 0.5)$$

$$tf - igm_{implement,B_1} = 4.5$$

3.6 กรอบการดำเนินงานสำหรับการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ (Generic Framework for Dependent Bug Report Assemblage)

การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ คือ ขั้นตอนของการรวบรวมรายงานจุดบกพร่องที่เกี่ยวข้องกันเข้ามาอยู่ในกลุ่มเดียวกัน เพื่อให้ง่ายต่อการพิจารณาความสัมพันธ์ของจุดบกพร่อง โดยมีขั้นตอนการดำเนินการดังรูปที่ 3.9

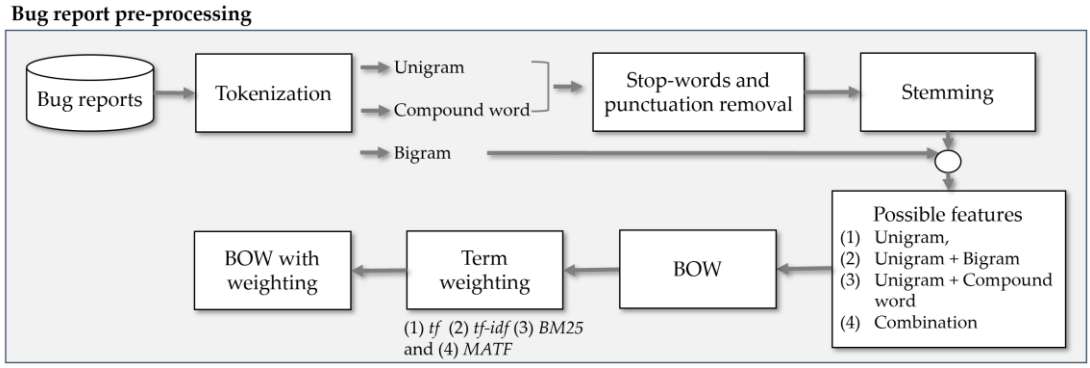


รูปที่ 3.9 กรอบการดำเนินงานสำหรับการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ (Generic Framework for Dependent Bug Report Assemblage)

จากรูปที่ 3.9 ที่แสดงกรอบการดำเนินงานของการวิเคราะห์รายงานจุดบกพร่องที่เป็นส่วนต่อ สามารถแบ่งเป็นขั้นตอนหลัก 3 ขั้นตอนคือ (1) การเตรียมรายงานจุดบกพร่อง (2) การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ และ (3) การประเมินผล ซึ่งอธิบายโดยละเอียดได้ดังต่อไปนี้

3.6.1 การเตรียมรายงานจุดบกพร่อง

การเตรียมรายงานจุดบกพร่อง คือขั้นตอนของการจัดเตรียมรายงานจุดบกพร่องให้อยู่ในรูปแบบที่เหมาะสมก่อนการเข้าสู่กระบวนการการจัดกลุ่ม โดยมีขั้นตอนย่อยๆ ที่สามารถแสดงได้ดังรูปที่ 3.10



รูปที่ 3.10 ขั้นตอนการเตรียมรายงานจุดบกพร่อง

จากรูปที่ 3.10 ที่แสดงขั้นตอนการเตรียมรายงานจุดบกพร่องนั้นประกอบไปด้วยขั้นตอนหลักอาทิเช่น การตัดคำ การกำจัดคำหยุด การหารูปแบบพื้นฐานร่วมของคำศัพท์ และการให้น้ำหนักคำ ซึ่งสามารถอธิบายโดยละเอียดในแต่ละขั้นตอนได้ดังต่อไปนี้

1) การทำความสะอาดรายงานจุดบกพร่อง (Bug report clearing)

ขั้นตอนนี้มีวัตถุประสงค์เพื่อลดข้อมูลที่ไม่จำเป็นออกจากรายงานจุดบกพร่อง เนื่องจากชุดข้อมูลที่ได้จากระบบติดตามรายงานจุดบกพร่องของมอซิลลาอยู่ในรูปแบบ XML ดังรูปที่ 3.11 ในขั้นตอนนี้จะเลือกข้อมูล 6 ส่วนจากรายงานจุดบกพร่องซึ่งประกอบไปด้วย

(1) Summary (ซึ่งบางงานวิจัยใช้คำว่า Header [10] หรือ Title [89]) ที่เป็นข้อความสรุปถึงจุดบกพร่องอย่างย่อ โดยปรากฏในแท็ก <short_desc/>

(2) Description เป็นข้อความอธิบายจุดบกพร่องโดยละเอียด และอยู่เป็นข้อความภายในแท็ก <long_desc/>

(3) Product เป็นข้อมูลที่มีคุณสมบัติเป็นหมวดหมู่ (Category) ให้ข้อมูลผลิตภัณฑ์ของรายงานจุดบกพร่องเหล่านั้น ซึ่งอยู่ภายในแท็ก <product/>

(4) Component เป็นข้อมูลที่มีคุณสมบัติเป็นหมวดหมู่ ที่จะแสดงองค์ประกอบของซอฟต์แวร์ที่จุดบกพร่องมีผลกระทบในรายงานจุดบกพร่อง อยู่ภายในแท็ก <component/>

(5) Priority เป็นข้อมูลที่มีคุณสมบัติเป็นหมวดหมู่ ให้ข้อมูลระดับความสำคัญของรายงานจุดบกพร่องเหล่านั้น ซึ่งอยู่ภายในแท็ก <priority/>

(6) Severity เป็นข้อมูลที่มีคุณสมบัติเป็นหมวดหมู่ ให้ข้อมูลระดับรุนแรงของรายงานจุดบกพร่องเหล่านั้น ซึ่งอยู่ภายในแท็ก <bug_severity/>

ทั้งนี้เนื่องจากหลายงานวิจัยที่วิเคราะห์รายงานจุดบกพร่องได้ใช้ข้อมูลเหล่านี้ในการดำเนินการวิจัย เช่น การทำนายความรุนแรงของจุดบกพร่อง [14, 15, 18] การตรวจสอบรายงานซ้ำซ้อน [3, 10, 11] เป็นต้น

```
▼<bugzilla version="20180223.1" urlbase="https://bugzilla.mozilla.org/" maintainer="bugzilla-
 admin@mozilla.org" exporter="bancha.lu@ksu.ac.th">
  ▼<bug>
     <bug_id>277000</bug_id>
     <creation_ts>2005-01-04 08:18:06 -0800</creation_ts>
    ▼<short_desc>
       Tracking: requests to change more dialogs to sheets
     </short_desc>
     <delta_ts>2016-06-23 11:53:06 -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>General</component>
     <version>Trunk</version>
     <rep_platform>PowerPC</rep_platform>
     <op_sys>Mac OS X</op_sys>
     <bug_status>NEW</bug_status>
     <resolution/>
```

รูปที่ 3.11 ลักษณะของรายงานจุดบกพร่องที่แสดงในรูปแบบ XML

โดยข้อมูล Summary และ Description จะปรากฏในเอกสาร XML ในตำแหน่งของแท็ก <short_desc> และ <long_desc> ตามลำดับ ดังรูปที่ 3.12 และ รูปที่ 3.13

```
▼<bug>
   <bug_id>277000</bug_id>
   <creation_ts>2005-01-04 08:18:06 -0800</creation_ts>
  ▼<short_desc>
     Tracking: requests to change more dialogs to sheets
   </short_desc>
   <delta_ts>2016-06-23 11:53:06 -0700</delta_ts>
```

รูปที่ 3.12 ข้อมูล Summary ที่ปรากฏในแท็ก <short_desc>

```
<long_desc isprivate="0">
  <commentid>2376311</commentid>
  <who name="Frankie">francis.uy@gmail.com</who>
  <bug_when>2005-01-04 08:18:06 -0800</bug_when>
  <thetext>
    There are a lot of independent bugs floating around with the same basic goal: more sheets. Some of
    them are probably dupes. Some of them would require bug 123908.
  </thetext>
</long_desc>
```

รูปที่ 3.13 ข้อมูล Description ที่ปรากฏในแท็ก <long_desc>

นอกจากนี้ในขั้นตอนนี้ยังเพิ่มการตรวจสอบข้อมูลในส่วน Summary และ Description เพื่อดำเนินการลบรายงานจุดบกพร่องที่มีข้อมูลไม่สมบูรณ์ เช่น รายงานที่มีข้อมูล Description ที่ไม่สามารถหาความสำคัญของเอกสารได้ เนื่องจากมีการแสดงหมายเลขหน่วยความจำที่ใช้ในการประมวลเป็นจำนวนมากดังรูปที่ 3.14

```
12:11:14     INFO -  /home/cltbld/talos-slave/test/build/application/firefox/firefox[0x40287b]
12:11:14     INFO -  /lib64/libc.so.6(__libc_start_main+0xfd)[0x34d2e1eb1d]
12:11:14     INFO -  /home/cltbld/talos-slave/test/build/application/firefox/firefox[0x401d29]
12:11:14     INFO -  ======= Memory map: ========
12:11:14     INFO -  00400000-00415000 r-xp 00000000 fd:00 1320850                    /home/cltbld/talos-
slave/test/build/application/firefox/firefox
12:11:14     INFO -  00614000-00615000 rw-p 00014000 fd:00 1320850                    /home/cltbld/talos-
slave/test/build/application/firefox/firefox
12:11:14     INFO -  01bf2000-1a87d000 rw-p 00000000 00:00 0                          [heap]
12:11:14     INFO -  41c62000-41c64000 rwxp 00000000 00:10 2711                       /dev/zero
12:11:14     INFO -  34d2800000-34d281e000 r-xp 00000000 fd:00 675                    /lib64/ld-2.11.so
12:11:14     INFO -  34d2a1d000-34d2a1e000 r--p 0001d000 fd:00 675                    /lib64/ld-2.11.so
12:11:14     INFO -  34d2a1e000-34d2a1f000 rw-p 0001e000 fd:00 675                    /lib64/ld-2.11.so
12:11:14     INFO -  34d2a1f000-34d2a20000 rw-p 00000000 00:00 0
12:11:14     INFO -  34d2c00000-34d2c01000 r-xp 00000000 fd:00 193158
/usr/lib64/nvidia/tls/libnvidia-tls.so.190.42
12:11:14     INFO -  34d2c01000-34d2d01000 ---p 00001000 fd:00 193158
/usr/lib64/nvidia/tls/libnvidia-tls.so.190.42
12:11:14     INFO -  34d2d01000-34d2d02000 rw-p 00001000 fd:00 193158
/usr/lib64/nvidia/tls/libnvidia-tls.so.190.42
12:11:14     INFO -  34d2e00000-34d2f6f000 r-xp 00000000 fd:00 677                    /lib64/libc-2.11.so
12:11:14     INFO -  34d2f6f000-34d316e000 ---p 0016f000 fd:00 677                    /lib64/libc-2.11.so
12:11:14     INFO -  34d316e000-34d3172000 r--p 0016e000 fd:00 677                    /lib64/libc-2.11.so
12:11:14     INFO -  34d3172000-34d3173000 rw-p 00172000 fd:00 677                    /lib64/libc-2.11.so
```

รูปที่ 3.14 รายงานจุดบกพร่องที่มีข้อความที่ไม่สามารถหาความสำคัญเอกสารได้

2) การตัดคำ (Tokenization)

ขั้นตอนนี้มีวัตถุประสงค์เพื่อแยกข้อความออกเป็นหน่วยทางภาษาที่มีความหมาย สำหรับขั้นตอนการตัดคำในงานวิจัยนี้ได้ศึกษาใน 3 รูปแบบคือ (1) การตัดคำแบบคำเดี่ยว (Unigram), (2) การตัดคำเป็นกลุ่มคำแบบสองคำ (Bigram) ด้วยกฎเชิงไวยากรณ์กลุ่มคำแบบสองคำ และ (3) การตัดคำจากกลุ่มคำผสม (Compound words) ดังที่กล่าวไว้แล้วในหัวข้อที่ 3.3

3) การกำจัดคำหยุด (Stop-words removal)

การกำจัดคำหยุดและเครื่องหมายวรรคตอน เป็นขั้นตอนการนำคำที่ไม่มีนัยสำคัญและเครื่องหมายวรรคตอนต่างๆ ออกไป เนื่องจากคำเหล่านี้ไม่ส่งผลต่อการประมวลผลข้อความ ส่วนใหญ่เป็นคำบุพบท คำสันธาน หรือคำสรรพนาม โดยใช้คำหยุดจากคลังคำหยุดภาษาอังกฤษของ Natural Language Tool Kit: NLTK ซึ่งสามารถแสดงการกำจัดคำหยุดได้ดังตัวอย่างในตารางที่ 3.14

ตารางที่ 3.14 การกำจัดคำหยุด

BugID 1410344	Add automated test for "Hover and `Delete from History` Top Sites tile"
Unigram tokenization	Add / automated / test / for / Hover / and / Delete / from / History / Top / Sites / tile
Stop-words removal	Add / automated / test / Hover / Delete / History / Top / Sites / tile

ซึ่งจากตารางที่ 3.14 จะพบว่า “for”, “and” และ “from” เป็นคำหยุดที่ต้องถูกลบออกไป โดยในการกำจัดคำหยุดในงานวิจัยนี้จะดำเนินการหลังจากขั้นตอนในการตัดคำแบบคำเดี่ยว และการตัดคำจากกลุ่มคำผสมเท่านั้น จะไม่ดำเนินการกับการตัดคำแบบกลุ่มคำแบบสองคำ

4) การหารูปแบบพื้นฐานร่วมของคำศัพท์ (Stemming)

การหารูปแบบพื้นฐานร่วมของคำศัพท์ คือ ขั้นตอนการหารูปแบบพื้นฐานร่วมกันที่พบในคำศัพท์ภาษาอังกฤษที่มีรากศัพท์เดียวกัน แต่อาจจะมีการเปลี่ยนรูปของคำไปตามกาล (Tenses) หรือเกิดการเปลี่ยนแปลงคำอันเนื่องมาจากการเติมส่วนของ Prefix และ Suffix เข้าไป เช่น กลุ่มคำว่า Computer, Computers, Computing, Computes, Compted, Conputational, และคำว่า Computation โดยกลุ่มคำเหล่านี้ล้วนมาจากรากศัพท์เดียวกันคือ “Computar” ในภาษาลาติน ซึ่งภายหลังจากการหารูปแบบพื้นฐานร่วมของคำศัพท์ จะได้รูปแบบคือ “Comput”

สำหรับในงานวิจัยนี้ ในการหารูปแบบพื้นฐานร่วมของคำศัพท์ของคำในภาษาอังกฤษได้ใช้ขั้นตอนวิธีในการหารูปแบบพื้นฐานร่วมของคำศัพท์ของ Snowball Stemmer [50] ซึ่งสามารถแสดงตัวอย่างได้ในตารางที่ 3.15

ตารางที่ 3.15 การหารูปแบบพื้นฐานร่วมของคำศัพท์

BugID 1410344	Add automated test for "Hover and `Delete from History` Top Sites tile"
Unigram tokenization	Add / automated / test / for / Hover / and / Delete / from / History / Top / Sites / tile
Stop-words removal	Add / automated / test / Hover / Delete / History / Top / Sites / tile
Stemming	add / autom / test / hover / delet / histori / top / site / tile

ซึ่งจากตารางที่ 3.15 จะพบว่าคำว่า “automated”, “Delete”, “History” และ “Sites” ถูกเปลี่ยนเป็น “autom”, “delet”, “histori” และ “site” ตามลำดับ โดยในงานวิจัยนี้ขั้นตอนการหารูปแบบพื้นฐานร่วมของคำศัพท์จะดำเนินการหลังจากขั้นตอนการกำจัดคำหยุดในการตัดคำแบบคำเดี่ยว และการตัดคำจากกลุ่มคำผสม เท่านั้น แต่จะไม่ดำเนินการกับการตัดคำแบบกลุ่มคำแบบสองคำ

5) การนำเสนอรายงานจุดบกพร่องเพื่อการประมวลผล (Bug report representation)

ภายหลังการตัดคำ การกำจัดคำหยุด และการหารูปแบบพื้นฐานร่วมของคำศัพท์แล้ว รายงานจุดบกพร่องจะถูกจัดให้อยู่ในโครงสร้างมาตรฐานสำหรับการประมวลผลด้านค้นคืนเอกสาร และการจัดกลุ่มหรือการจำแนกเอกสารนั้นคือ โมเดลเชิงพื้นที่แบบเวกเตอร์ เพื่อแสดงความสัมพันธ์ระหว่างรายงานจุดบกพร่องและคุณลักษณะของรายงานจุดบกพร่องนั้นๆ ซึ่งในที่นี้ก็คือ “คำ” หรือ “กลุ่มคำ”

เนื่องจากคุณลักษณะโดยพื้นฐานในการประมวลผลด้านค้นคืนเอกสารและการจัดกลุ่มหรือการจำแนกเอกสารจะเป็น “คำ” ดังนั้นโครงสร้างโมเดลเชิงพื้นที่แบบเวกเตอร์จึงมักจะเรียกว่า “ถุงคำ” สำหรับในงานวิจัยนี้ได้ทำการศึกษาคุณลักษณะของรายงานจุดบกพร่องใน 4 รูปแบบดังต่อไปนี้ (1) Unigram (2) Unigram + bigram (3) Unigram + compound words และ (4) Combination

จากการทำคุณลักษณะทั้ง 4 แบบดังกล่าวสามารถนำเสนอในรูปแบบของโมเดลเชิงพื้นที่แบบเวกเตอร์ได้ดังตัวอย่างในตารางที่ 3.16 โดยสมมติว่ามีรายงานจุดบกพร่องจำนวน 4 รายงานที่จัดทำคุณลักษณะแบบคำเดี่ยว ดังต่อไปนี้

B_1: implement / translat / infobar

B2: implement / linux / style / translat / infobar

B3: implement / window / style / translat / infobar

B4: set / right / maxresult / urlbarinput

ตารางที่ 3.16 ตัวอย่างโมเดลเชิงพื้นที่แบบเวกเตอร์แสดงคำและน้ำหนักคำในแต่ละรายงานจุดบกพร่อง

Document/ words	implement	translat	infobar	linux	style	window	set	right	maxresult	urlinput
B_1	1	1	1	0	0	0	0	0	0	0
B_2	1	1	1	1	1	0	0	0	0	0
B_3	1	1	1	0	1	1	0	0	0	0
B_4	0	0	0	0	0	0	1	1	1	1

จากตารางที่ 3.16 จะเห็นได้ว่านอกจากจะสามารถแสดงความสัมพันธ์ระหว่างคำกับรายงานจุดบกพร่องแล้ว ยังแสดงให้เห็นว่ารายงานจุดบกพร่องใดที่มีคำนั้นจะมีค่าเท่ากับ 1 ขณะที่หากไม่มีคำนั้นๆ จะแสดงด้วยค่า 0 ซึ่งค่าเหล่านี้คือความถี่ของคำ หรือ term frequency ของคำที่ปรากฏในรายงานจุดบกพร่องเหล่านั้น

6) การให้น้ำหนัก (Term weighting)

ในขั้นตอนของการให้น้ำหนักของคำที่ปรากฏในรายงานจุดบกพร่อง ซึ่งงานวิจัยนี้ได้ทำการศึกษากับเทคนิคการให้น้ำหนัก 4 รูปแบบนั่นคือ *tf*, *tf-idf*, *BM25* และ *MATF* โดย 3 เทคนิคแรกเป็นเทคนิคที่ได้รับความนิยมในการศึกษาเกี่ยวกับรายงานจุดบกพร่อง [3, 12, 13, 15, 17-19,

39] ขณะที่ *MATF* เป็นเทคนิคที่ค่อนข้างใหม่ที่ถูกนำเสนอในปี 2013 [65] ที่ให้ความสำคัญกับความแตกต่างของขนาดของรายงานจุดบกพร่องแต่ละรายงานที่นำเข้าสู่การจัดกลุ่มแบบอัตโนมัติ ที่มีการปรับความสมดุลไม่ให้เกิดความโน้มเอียง (Bias) ของการจัดกลุ่มรายงานจุดบกพร่องไปทางรายงานที่มีขนาดใหญ่มากจนเกินไป

จากการดำเนินงานในขั้นตอนการเตรียมรายงานจุดบกพร่องได้เลือกใช้คุณลักษณะ (Features) ของรายงานจุดบกพร่องเป็น 4 แบบด้วยกันคือ (1) Unigram (2) Unigram + bigram (3) Unigram + compound words และ (4) Combination และนอกจากนี้ยังดำเนินการให้น้ำหนักของคำหรือกลุ่มคำเป็น 4 เทคนิคด้วยได้แก่ *tf*, *tf-idf*, *BM25* และ *MATF* ซึ่งสามารถแสดงให้อยู่ในรูปแบบของโมเดลเชิงพื้นที่แบบเวกเตอร์ได้ดังรูปที่ 3.15

0.0833333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0833333	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.125	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.1111111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0666667	0	0	0	0	0	0	0	0	0	0	0	0	0.0666667	0	0	0	0	0	0
0.0909091	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0714286	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.0555556	0.0555556	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.125	0	0	0	0	0
0.1818182	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.0714286	0.0714286	0	0	0	0	0	0	0	0	0	0	0.0714286	0.0714286	
0	0	0	0	0	0.0555556	0.0555556	0	0	0	0	0	0	0	0	0	0	0.0555556	0.0555556	
0	0	0	0	0	0.1538462	0.0769231	0	0	0	0	0	0.0769231	0	0	0	0	0.0769231	0.0769231	
0	0	0	0	0	0.1428571	0.1428571	0	0	0	0	0	0	0	0	0	0	0.1428571	0.1428571	
0	0	0	0	0	0.1428571	0.1428571	0	0	0	0	0	0	0	0	0	0	0.1428571	0.1428571	

รูปที่ 3.15 โมเดลเชิงพื้นที่แบบเวกเตอร์ของรายงานจุดบกพร่อง

จากรูปที่ 3.15 เป็นตัวอย่างของโมเดลเชิงพื้นที่แบบเวกเตอร์ ซึ่งทดลองกับจำนวนรายงานจุดบกพร่อง 50 รายงาน และดำเนินการให้น้ำหนักคำด้วยเทคนิค *tf*

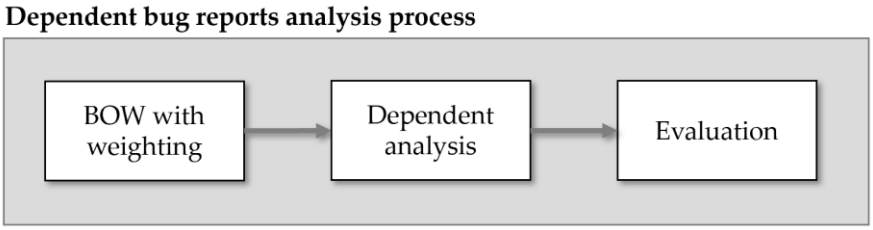
เมื่อได้โมเดลเชิงพื้นที่แบบเวกเตอร์ที่สกัดคุณลักษณะและดำเนินการให้ค่าน้ำหนักทั้ง 4 เทคนิค คือ ได้แก่ *tf*, *tf-idf*, *BM25* และ *MATF* แล้วจึงนำโมเดลเชิงพื้นที่แบบเวกเตอร์เหล่านั้นมาทำขั้นตอนการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องในขั้นตอนต่อไป

3.6.2 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ (Assembling of dependent bug reports)

การวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง ได้อาศัยสมมติฐานจากศึกษาและการสำรวจรายงานจุดบกพร่องที่เป็นส่วนต่อหรือมีความขึ้นต่อกัน และพบว่ารายงานจุดบกพร่องเหล่านั้นมีส่วนร่วมกัน ดังเช่น ใช้คำที่สอดคล้องกันในการอธิบายความผิดปกติของจุดบกพร่องในคุณลักษณะข้อความทั้งในส่วน Summary และ Description นอกจากนี้ในคุณลักษณะหมวดหมู่อื่นๆ ยังพบความสอดคล้องเพิ่มเติมเช่น ผลิตภัณฑ์ (Product) องค์ประกอบ (Component) ระดับความสำคัญ (Priority) และระดับความรุนแรง (Severity) มักเหมือนกัน หรือใกล้เคียงกัน ทั้งนี้หากเราสามารถวิเคราะห์ความขึ้นต่อกันของจุดบกพร่องแบบอัตโนมัติโดยอาศัยคุณลักษณะข้อความและคุณลักษณะหมวดหมู่จากรายงานจุดบกพร่องได้ จะสามารถรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อ

กันไว้ในกลุ่มเดียวกันและท้ายที่สุดจะช่วยให้นักพัฒนาซอฟต์แวร์สามารถมองเห็นภาพรวมของปัญหาเหล่านั้นได้ อีกทั้งอาจช่วยเชื่อมโยงกลุ่มของปัญหาเหล่านั้นได้อีกด้วยว่าจะมีผลกระทบกับกลุ่มปัญหาใดอีกบ้าง และเป็นแนวทางในการกำหนดนักพัฒนาซอฟต์แวร์ เวลา และงบประมาณ ในการแก้ไขปัญหาเหล่านั้นต่อไป

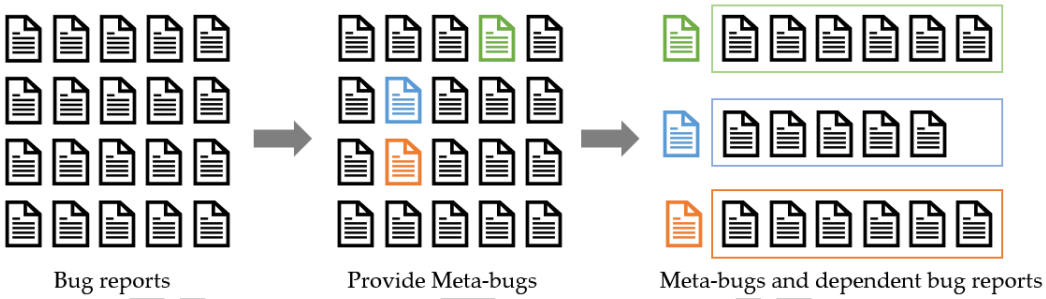
จากเหตุที่กล่าวข้างต้นขั้นตอนนี้จึงมีความสำคัญอย่างมากในงานวิจัยฉบับนี้ จึงแสดงภาพรวมการทำงานในการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง จะประกอบไปด้วยขั้นตอนดังต่อไปนี้



รูปที่ 3.16 ขั้นตอนการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง

จากรูปที่ 3.16 ที่แสดงขั้นตอนการการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง จะเป็นการนำโมเดลเชิงพื้นที่แบบเวกเตอร์ของรายงานจุดบกพร่องที่ผ่านการให้น้ำหนักคำในเทคนิค *tf*, *tf-idf*, *BM25* และ *MATF* มาดำเนินการรวบรวมเข้ากลุ่ม จากนั้นจะนำผลลัพธ์ที่ได้มาประเมินผลในค่าอัตราผลบวกจริง (True positive rate :TPR or Recall) อัตราผลลบจริง (True negative rate: TNR) อัตราผลบวกเท็จ (False positive rate: FPR or 1-TNR) ความแม่นยำ (Precision: P) ค่าประสิทธิภาพ (F-measure: F1) ค่าความถูกต้อง (Accuracy: Acc) การหาค่า Threshold ที่ดีที่สุดโดยอาศัย เส้นโค้ง ROC และการหาพื้นที่ใต้เส้นโค้ง ROC (AUC)

จากการศึกษารายงานจุดบกพร่อง เพื่อให้ทราบถึงวิธีการที่ผู้ตรวจสอบรายงานจุดบกพร่องและนักพัฒนาซอฟต์แวร์ใช้ในการรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อเข้าไว้ด้วยกันนั้น พบว่าผู้ตรวจสอบรายงานจุดบกพร่องและนักพัฒนาซอฟต์แวร์ จะมีการกำหนดรายงานจุดบกพร่องพิเศษ เพื่อใช้สำหรับการรวบรวมรายงานจุดบกพร่องอื่นๆ ที่มีการรายงานปัญหาในกลุ่มปัญหาเดียวกันเข้ารวมกลุ่มกัน ซึ่งเรียกรายงานจุดบกพร่องพิเศษที่เปรียบเสมือนกับเป็นตัวแทนของโดเมนปัญหานี้ว่า Meta-bug [1, 26] โดย Meta-bug ถูกกำหนดขึ้นมาได้จากสองกรณีด้วยกัน โดยกรณีแรก Meta-bug นั้น เป็นรายงานฉบับแรกที่กล่าวถึงโดเมนปัญหานั้น จึงถูกกำหนดสถานะพิเศษนี้ให้ หรือกรณีที่สองอาจเป็นรายงานที่ถูกกำหนดขึ้นเป็นกรณีพิเศษจากความเห็นของผู้ตรวจสอบรายงานจุดบกพร่องหรือนักพัฒนาซอฟต์แวร์ที่พิจารณาว่าควรมีรายงานจุดบกพร่องที่เป็นศูนย์รวมของโดเมนปัญหานั้น และ Meta-bug ในกรณีที่สองนี้ มักจะใช้คำที่เหมาะสมในการอธิบายเพื่อครอบคลุมถึงกลุ่มโดเมนปัญหาเหล่านั้น [1, 26] ซึ่งสามารถแสดงแนวคิดการใช้ Meta-bug เป็นตัวแทนของโดเมนปัญหาและเพื่อประโยชน์ในการรวบรวมรายงานที่เป็นส่วนต่อที่กล่าวถึงปัญหาในกลุ่มเดียวกัน ดังรูปที่ 3.17



รูปที่ 3.17 แนวคิดการใช้ Meta-bug เป็นตัวแทนโดเมนปัญหา

ด้วยเหตุนี้กระบวนการวิจัย จึงใช้ประโยชน์จากความรู้ที่ได้รับจากการศึกษาการใช้งาน Meta-bug และประยุกต์เป็นวิธีการวิเคราะห์รายงานจุดบกพร่องที่เป็นส่วนต่อของรายงานจุดบกพร่อง เพื่อรวบรวมรายงานที่เป็นส่วนต่อกันเข้าด้วยกันเป็น 3 วิธีการ ดังต่อไปนี้

1) วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยเทคนิคการจัดกลุ่ม Clustering (Dependent bug reports analysis using clustering techniques)

2) วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยเทคนิคค่าเทรชโฮลด์และการวัดความคล้าย (Dependent bug reports analysis using thresholds-based similarity measure)

3) วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยฟังก์ชันการค้นคืน (Dependent bug reports analysis using retrieval function: REP)

ซึ่งทั้ง 3 วิธีการเป็นการประยุกต์วิธีการที่แตกต่างกันทั้งนี้ เพื่อหาวิธีการที่มีความเหมาะสมเป็นที่น่าพอใจที่สุดในการวิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่อง โดยจะได้อธิบายรายละเอียดในแต่ละวิธีการหัวข้อลำดับถัดไป

3.7 วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยเทคนิคการจัดกลุ่ม Clustering (Dependent bug reports analysis using clustering techniques)

ในการศึกษาเริ่มต้น ในงานวิจัยนี้ได้ประยุกต์การจัดกลุ่มด้วย Clustering เนื่องจากแนวคิดการใช้รายงานจุดบกพร่องที่เป็น Meta-bug เป็นตัวแทนของโดเมนปัญหาในแต่ละกลุ่ม ซึ่งเทียบได้กับแนวคิดการกำหนด Centroid ให้กับแต่ละกลุ่มในการจัดกลุ่มแบบ Clustering ดังนั้นเมื่อ Meta-bug ที่เป็นตัวแทนของแต่ละโดเมนปัญหาก็คือ Centroid และถ้ารายงานจุดบกพร่องอื่นๆ ที่รายงานเข้ามา จะสามารถนำมาวิเคราะห์ตามแนวคิดของการจัดกลุ่มแบบ Clustering เพื่อกำหนดให้รายงานเหล่านั้นอยู่ในกลุ่มโดเมนปัญหาเดียวกันได้ เนื่องด้วยรายงานจุดบกพร่องที่กล่าวถึงโดเมนปัญหาเดียวกันมักใช้คำที่อธิบายถึงปัญหาเหล่านั้นที่สอดคล้องกัน และการทดลองตามขั้นตอนนี้จะใช้ข้อมูลที่เป็นข้อความจากส่วน Summary เพียงอย่างเดียวเนื่องจาก ข้อมูลในส่วนนี้ได้รับการยอบรับว่าเป็นข้อมูลที่เป็นประโยชน์มากกว่าในการวิเคราะห์ [11, 85, 90] อีกทั้งยังพบข้อความที่ไม่ถูกต้อง (Noise) น้อยกว่าในส่วน Description และในการทดลองนี้ ได้ใช้รายงานจุดบกพร่องในการทดลองจำนวน 2,750 รายงาน โดยที่มีจำนวน Meta-bug 100 รายงาน และออกแบบการทดสอบเป็นการจัดกลุ่ม

โดยจะเพิ่มขนาดของกลุ่มตัวอย่างเข้าไป และเพิ่มจำนวนกลุ่ม (*k*) เข้าไป เพื่อดูผลลัพธ์และความสามารถของการจัดกลุ่มของแต่ละวิธีการ

3.7.1 การจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัด (Constraint-based *k*-means clustering)

ในงานวิจัยนี้แรกเริ่มได้ประยุกต์ใช้การจัดกลุ่มด้วยเคมีนแบบดั้งเดิม โดยมีขั้นตอนของอัลกอริทึมเคมีนซึ่งแสดงได้ดังรูปที่ 3.18

Algorithm	:	The constraint-based *k*-means clustering with centroid changing
Input	:	Data point *D*, Number of cluster *k (where D is bug report dataset)*
Output	:	Cluster with relevant bug reports
1	:	Read the number of Meta-bugs
2	:	Setting the value of k with the number of Meta-bugs
3	:	Initialize *k* centroid randomly
4	:	For each cluster c_i
5	:	Repeat :
6	:	Cluster the data points based on the distance of their intensities from the centroid intensities : $c_i = \arg\min \| x_i - \mu_i \|$ where x is a data point (or bug report), while μ is a centroid
7	:	Compute the new centroid for each of the cluster : $\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$ Where i iterates over the all intensities, j iterates over all the centroids, and μ_i is the centroid intensity.
8	:	Until: cluster label of the bug report does not change anymore.
9	:	End of for

รูปที่ 3.18 รหัสเทียมอัลกอริทึมการจัดกลุ่มด้วยเคมีน

โดยในการทดลองตามขั้นตอนวิธีเคมีน ได้มีการกำหนด *k* เท่ากับจำนวน Meta-bug ที่นำเข้าประมวลผล ซึ่งผลการทดลองเบื้องต้นการจัดกลุ่มด้วยเคมีนให้ผลไม่เป็นที่น่าพอใจเนื่องจากการสุ่มค่า Centroid ของแต่ละคลัสเตอร์ อาจจะได้ค่าไม่เหมาะสม เพราะค่า Centroid ที่สุ่มขึ้นมาในทุกรอบอาจจะไม่ได้ค่าเดียวกัน และบางรอบค่า Centroid ที่ได้อาจจะมาจากคลัสเตอร์เดียวกัน ดังนั้นหากได้ตำแหน่งของ Centroid ที่ไม่ดีก็จะส่งผลให้ได้ผลลัพธ์ที่ไม่ดีตามไปด้วย ซึ่งเป็นปัญหาที่คล้ายกับงาน Bradley และคณะ [77]

จากข้อสังเกตที่พบจากการจัดกลุ่มด้วยขั้นตอนวิธีเคมีน จึงได้ถูกปรับเพื่อให้เกิดความเหมาะสมในการวิเคราะห์เพื่อการจัดกลุ่มรายงานจุดบกพร่องแบบอัตโนมัติ ซึ่งการปรับปรุงการจัดกลุ่มด้วยขั้นตอนวิธีเคมีนนี้ คือการปรับเปลี่ยนขั้นตอนวิธีเคมีนด้วยการเพิ่มข้อจำกัด (Constraints) [71, 91] เข้าไปในขั้นตอนวิธีเคมีนเพื่อเพิ่มประสิทธิภาพในการจัดกลุ่ม ซึ่งโดยทั่วไปข้อจำกัดที่ใช้มักจะเป็นข้อจำกัดเฉพาะภายใต้ชุดข้อมูลที่ศึกษา [71, 91] และใน [71, 91] ยังพบว่าข้อจำกัดที่เพิ่มเข้าไปในขั้นตอนวิธีเคมีนแบบดั้งเดิม หากมีความเหมาะสมก็มีแนวโน้มที่จะสามารถทำให้การจัดกลุ่มในแต่ละคลัสเตอร์ดียิ่งขึ้น

ดังนั้นภายหลังจากที่พบข้อสังเกตุดังกล่าวข้างต้น จึงได้ปรับอัลกอริทึมขั้นตอนวิธีเคมีนด้วยการเพิ่มข้อจำกัด เข้าไป 2 ข้อจำกัดคือ คือ (1) จำนวนคลัสเตอร์หรือค่า *k* จะมีค่าเท่ากับจำนวน Meta-bug ที่สุ่มเข้ามา และ (2) กำหนดให้รายงานจุดบกพร่องที่เป็น Meta-bug คือค่า Centroid เริ่มต้น ซึ่งขั้นตอนวิธีเคมีนแบบดั้งเดิมที่ได้รับการเพิ่มข้อจำกัดเข้าไป เราเรียกว่า การจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัด (Constraint-based *k*-means clustering)

จากการเพิ่มข้อจำกัดให้กับเคมีนเป็นขั้นตอนวิธีการจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัดที่งานวิจัยนี้ได้นำเสนอสามารถแสดงเป็นรหัสเทียมได้ดังรูปที่ 3.19

Algorithm	:	The constraint-based *k*-means clustering
Input	:	Data point *D*, Number of cluster *k (where D is bug report dataset)*
Output	:	Cluster with relevant bug reports
1	:	Read the number of Meta-bugs
2	:	Setting the value of k with the number of Meta-bugs (** 1st constraint)
3	:	Providing Meta-bug as the centroid of each cluster (** 2nd constraint)
4	:	For each cluster c_i
5	:	Repeat :
6	:	Cluster the data points based on the distance of their intensities from the centroid intensities : $c_i = \arg\min \| x_i - \mu_i \|$ where x is a data point (or bug report), while μ is a centroid
7	:	Compute the new centroid for each of the cluster : $\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$ Where i iterates over the all intensities, j iterates over all the centroids, and μ_i is the centroid intensity.
8	:	Until: cluster label of the bug report does not change anymore.
9	:	End of for

รูปที่ 3.19 รหัสเทียมอัลกอริทึมการจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัด

3.7.2 การจัดกลุ่มด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัด (Constraint-based spherical *k*-means clustering)

จากการศึกษาการจัดกลุ่มด้วยเคมีนพบว่า การจัดกลุ่มด้วยเคมีนจะให้ผลดีกับลักษณะกลุ่มข้อมูลที่มีการรวมกลุ่มเป็นทรงกลม จึงได้ทำการทดลองกับเคมีนทรงกลม (Spherical *k*-means) ที่สามารถจัดกลุ่มเอกสารที่มีจำนวนมากได้เป็นอย่างดี และมีประสิทธิภาพในกรณีที่เอกสารที่เทียบกันสองเอกสารมีความยาวไม่เท่ากัน หรือเป็นการทำให้มีความยุติธรรมต่อเอกสารที่สั้นกว่านั่นเอง ซึ่งสามารถแสดงเป็นรหัสเทียมได้ดังรูปที่ 3.20

Algorithm : The spherical *k*-means clustering

Input : Data point *D*, Number of cluster *k (where D is bug report dataset)*

Output : Cluster with relevant bug reports

1 : Read the number of Meta-bugs

2 : Setting the value of *k* with the number of Meta-bugs

3 : Partitioning data point *D* into *k* disjoint cluster by random and set the index of iteration $t = 0$

$$\bigcup_{j=1}^{k} \pi_j = \{x_1, x_2, ..., x_n\} \text{ and } \pi_j \cap \pi_\ell = \phi \text{ if } j \neq \ell$$

4 : Compute the concept vector for each of the cluster :

$$c_j^{(t)} = \frac{m_j^{(t)}}{\left\| m_j^{(t)} \right\|} \text{ where } m_j^{(t)} = \frac{1}{n_j} \sum_{X \in \pi_j} x,$$

where x is a data point (or bug report), n is the number of data point in π_j

5 : Cluster the data points based on the cosine similarity :

$$sim(x, c_j) = \frac{\sum_{i=1}^{n} x_i \times c_i}{\left(\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} c_i^2 \right)^{1/2}}$$

where x is a data point (or bug report), while c is a concept vector

6 : Compute the new concept vector for each of the cluster

7 : Repeat step 5, 6 and increment t by 1 until the objective function is less than the threshold,

$$\left| Q\left(\{\pi_j^{(t)}\}_{j=1}^{k} \right) - Q\left(\{\pi_j^{(t+1)}\}_{j=1}^{k} \right) \right| \leq \varepsilon \quad where \quad Q\left(\{\pi_j\}_{j=1}^{k} \right) = \sum_{j=1}^{k} \sum_{X \in \pi_j} x^T c_j$$

รูปที่ 3.20 รหัสเทียมอัลกอริทึมเคมีนทรงกลม

เนื่องด้วยแนวทางการเพิ่มข้อจำกัดให้กับเคมีน ดังนั้นการทดลองกับเคมีนทรงกลม จึงได้กำหนดข้อจำกัดให้กับเคมีนทรงกลมด้วย โดยได้เพิ่มข้อจำกัดให้กับเคมีนทรงกลม ดังนี้

(1) ให้จำนวนกลุ่มเท่ากับจำนวนของ Meta-bug

(2) กำหนดให้ Meta-bug เป็น Centroid ของแต่ละคลัสเตอร์

ซึ่งจะเห็นได้ว่าข้อจำกัดข้างต้นเป็นข้อจำกัดเดียวกันกับวิธีการเคมีนแบบเพิ่มข้อจำกัด ดังนั้นสามารถแสดงรหัสเทียมของเคมีนทรงกลมแบบเพิ่มข้อจำกัดได้ดังรูปที่ 3.21

Algorithm : The spherical *k*-means clustering

Input : Data point *D*, Number of cluster *k (where D is bug report dataset)*

Output : Cluster with relevant bug reports

1 : Read the number of Meta-bugs

2 : Setting the value of k with the number of Meta-bugs (** 1st constraint)

3 : Providing Meta-bug as the centroid of each cluster (** 2nd constraint)

4 : Compute the concept vector for each of the cluster :

$$c_j^{(t)} = \frac{m_j^{(t)}}{\left\| m_j^{(t)} \right\|} \text{ where } m_j^{(t)} = \frac{1}{n_j} \sum_{X \in \pi_j} x,$$

where x is a data point (or bug report), n is the number of data point in π_j

5 : Cluster the data points based on the cosine similarity :

$$sim(x, c_j) = \frac{\sum_{i=1}^{n} x_i \times c_i}{\left(\sum_{i=1}^{n} x_i^2 \times \sum_{i=1}^{n} c_i^2 \right)^{1/2}}$$

where x is a data point (or bug report), while c is a concept vector

6 : Compute the new concept vector for each of the cluster

7 : Repeat step 5, 6 and increment t by 1 until the objective function is less than the threshold,

$$\left| Q\left(\{\pi_j^{(t)}\}_{j=1}^{k} \right) - Q\left(\{\pi_j^{(t+1)}\}_{j=1}^{k} \right) \right| \le \varepsilon \quad where \quad Q\left(\{\pi_j\}_{j=1}^{k} \right) = \sum_{j=1}^{k} \sum_{X \in \pi_j} x^T c_j$$

รูปที่ 3.21 รหัสเทียมอัลกอริทึมการจัดกลุ่มด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัด

3.8 วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยค่าเทรสโฮลด์และการวัดความคล้าย (Dependent bug reports analysis using thresholds-based similarity measure)

ขั้นตอนการวิเคราะห์การขึ้นต่อกันของรายงานจุดบกพร่องด้วยค่าเทรสโฮลด์และการวัดความคล้ายนี้ ยังคงใช้แนวคิดการใช้ Meta-bug เป็น Centroid คล้ายกับการจัดกลุ่มด้วย Clustering โดยประยุกต์ใช้การวัดความคล้าย (Similarity measure) ร่วมกับค่าเทรสโฮลด์ (Threshold) [27, 28, 92] ซึ่งในหลายงานวิจัยที่เกี่ยวข้องกับจุดบกพร่องได้ประยุกต์ใช้วิธีการนี้เช่นเดียวกันเพื่อวัดค่าความคล้าย สำหรับงานวิจัยนี้เพื่อจัดกลุ่มรายงานจุดบกพร่องที่มีความเป็นส่วนต่อ จะเป็นการใช้ Meta-bug เป็นเอกสารแบบสอบถาม (Query) เพื่อเทียบกับรายงานจุดบกพร่องอื่นๆ ว่ามีความคล้ายมากเท่าใด และใช้ค่าเทรสโฮลด์เป็นตัวตัดสินในการจัดเข้ากลุ่มของแต่ละ Meta-bug ซึ่งสามารถนำเสนอได้ตามขั้นตอนวิธีดังรูปที่ 3.22 โดยในการทดลองนี้ได้เตรียมชุดข้อมูลรายงานจุดบกพร่อง 4,871 รายงาน ซึ่งมีจำนวน Meta-bug 200 รายงาน และยังคงใช้ข้อมูลเฉพาะในส่วนของ Summary เพื่อสกัดเป็นคุณลักษณะของรายงานจุดบกพร่องเท่านั้น

Algorithm	:	Dependent bug reports assemblage with a threshold
Input	:	M is a set of Meta-bugs
		B is a set of bug reports
		T is a set of threshold, {0.1, 0.2, 0.3,…, 1.0}
Output	:	Cluster of each Meta-bug and its relevant bug reports
Parameter	:	*R* : a set of *M* U *B*
		m_i : the current Meta-bug that is analyzed
		r_i : the current bug report that is analyzed
		Sim : Similarity measure
		C_{mi} : Cluster of m_i
1	:	Let *R* be *M U B*;
2	:	**while** not end of *M* **do**
3	:	$r_i \longleftarrow R$ //read the next bug reports;
4	:	**if** r_i != m_i **then**
5	:	similarity score ⟵ $Sim(m_i,r_i)$;
6	:	**if** similarity score >= *T* **then**
7	:	Add r_i into C_{mi};
8	:	**end**
9	:	**end**
10	:	**end**

รูปที่ 3.22 ขั้นตอนวิธีการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทรสโฮลด์และการวัดความคล้าย

จากรูปที่ 3.22 ที่แสดงขั้นตอนวิธีการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทรสโฮลด์และการวัดความคล้ายนั้น จะมีการกำหนดค่าเทรสโฮลด์อยู่ระหว่าง 0-1 โดยการเพิ่มค่าเทรสโฮลด์ในแต่ละรอบการทดลองทีละ 0.1 โดยการวัดความคล้ายที่ใช้ในการทดลองนี้มีด้วยกันสามวิธี คือ การวัดความคล้ายด้วยโคไซน์ การวัดความคล้ายด้วย BM25 และ การวัดความคล้ายด้วย MATF และผลลัพธ์ที่ได้จะเป็นกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อของแต่ละ Meta-bug

3.8.1 การวัดความคล้ายด้วยโคไซน์ (Cosine similarity)

การวัดความคล้ายด้วยโคไซน์ จะเป็นการเปรียบเทียบความคล้ายของสองเอกสาร ซึ่งในที่นี้คือรายงานจุดบกพร่องที่เป็น Meta-bug และรายงานจุดบกพร่องอื่นๆ ตามสมการ (3.6)

$$sim_{cos(\theta)}(V_1V_2) = \frac{v_1 v_2}{\|v_1\|\|v_2\|} \quad (3.6)$$

โดยที่ V_1 คือเวกเตอร์ของรายงานจุดบกพร่องที่นำมาหาความคล้ายกับ V_2 ที่เป็นเวกเตอร์ของรายงานจุดบกพร่องที่เป็น Meta-bug ซึ่งค่าความคล้ายจะมีค่าอยู่ระหว่าง 0-1 จากนั้นจะนำไปเทียบกับค่าเทรสโฮลด์ที่กำหนด เพื่อใช้ในการตัดสินเข้ากลุ่มของ Meta-bug

3.8.2 การวัดความคล้ายด้วย BM25 (BM25 similarity)

การวัดความคล้ายด้วย BM25 จะเป็นการเปรียบเทียบความคล้ายของสองเอกสาร ซึ่งในที่นี้คือรายงานจุดบกพร่องที่เป็น Meta-bug และรายงานจุดบกพร่องอื่นๆ ดังสมการ (3.7)

$$sim(d,q) = \sum_{i=1}^{|q|} log\left(\frac{N - df(q_i) + 0.5}{df(q_i) + 0.5}\right) \times \left(\frac{tf(q_i,d) \times (k_1 + 1)}{tf(q_i,d) + k_1\left(1 - b + b \times \left(\frac{|d|}{dl_{avg}}\right)\right)}\right) \quad (3.7)$$

โดยที่ d คือรายงานจุดบกพร่องๆ ในขณะที่ q คือรายงานจุดบกพร่องที่เป็น Meta-bug จากนั้นจะนำคำที่เป็นคุณลักษณะของ Meta-bug ทีละคำ มาคำนวณหาความคล้ายกับรายงานจุดบกพร่องอื่นๆ ตามสมการและรวมผลลัพธ์ความคล้ายในแต่ละคำ โดยภายในสมการกำหนดให้พารามิเตอร์อิสระ k_1 และ b มีค่าเป็น 2 และ 0.5 ตามลำดับ เนื่องจากข้อมูลจาก Summary ที่เป็นข้อความไม่ยาวมากจึงกำหนดให้ b เท่ากับ 0.5 [73] เมื่อคำนวณตามสมการเสร็จค่าความคล้ายที่ได้มีโอกาสที่จะมีค่ามากกว่า 1 จึงจำเป็นที่จะต้องใช้ฟังชันก์ $f(x) = x/(1+x)$ เพื่อปรับค่าความคล้ายที่ได้นี้ให้มีค่าไม่เกิน 1 [63, 65, 93] หลังจากนั้นจึงนำไปเทียบกับค่าเทรสโฮลด์ที่กำหนด เพื่อใช้ในการตัดสินเข้ากลุ่มของ Meta-bug

3.8.3 การวัดความคล้าย MATF (MATF similarity)

การวัดความคล้ายด้วย MATF จะเป็นการเปรียบเทียบความคล้ายของสองเอกสาร ซึ่งในที่นี้คือรายงานจุดบกพร่องที่เป็น Meta-bug และรายงานจุดบกพร่องอื่นๆ เช่นเดียวกันแต่ไม่มีการกำหนดค่าพารามิเตอร์อิสระเหมือนกับ BM25 เพราะภายในการคำนวณค่าความคล้ายด้วย MATF จะมีพารามิเตอร์ที่ให้ความสำคัญกับความสั้นและยาวของเอกสารที่เป็นแบบสอบถามได้โดยอัตโนมัติแล้ว [65] ดังสมการ(3.8)

$$sim_{norm}(Q,d) = \frac{\sum_{i=1}^{|Q|} TFF(q_i,d) \times TDF(q_i,C)}{\sum_{i=1}^{|Q|} TDF(q_i,C)} \quad (3.8)$$

โดยที่ Q คือรายงานจุดบกพร่องที่เป็น Meta-bug ในขณะที่ d คือรายงานจุดบกพร่องอื่นๆ ซึ่งจะนำคำที่ปรากฏใน Meta-bug ไปหาความคล้ายกับแต่ละรายงานจุดบกพร่องและรวมเป็นผลลัพธ์ของค่าความคล้ายกับรายงานจุดบกพร่องนั้น โดยจะมีค่าความคล้ายอยู่ระหว่าง 0-1

3.9 วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยฟังก์ชันการค้นคืน (Dependent bug reports analysis using retrieval function: REP)

เนื่องจากการทดลองทั้งสองวิธีข้างต้นคือ วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยเทคนิคการจัดกลุ่ม Clustering และ วิเคราะห์การเป็นส่วนต่อของรายงานจุดบกพร่องด้วยค่าเทรสโฮลด์และการวัดความคล้าย นั้นได้ใช้ข้อมูลเพียงส่วน Summary จากรายงานจุดบกพร่องอย่างเดียวเท่านั้น แต่จากการศึกษาและการสังเกตรายงานจุดบกพร่องที่เป็นส่วนต่อ ยังมีความสอดคล้องกันในส่วน ของ Description Product Component Priority และ Severity จึงได้นำข้อมูลเหล่านี้มาร่วมเป็นคุณลักษณะเพื่อประเมินความคล้ายของรายงานจุดบกพร่อง Meta-bug กับรายงานจุดบกพร่องอื่นๆ

นอกจากนี้ในหลายงานวิจัยได้ใช้ลักษณะของการใช้คุณลักษณะอื่นๆ เช่นกัน และในงานวิจัยเหล่านั้นจะใช้สมการเชิงเส้นเพื่อรวมค่าความคล้ายของแต่ละคุณลักษณะต่างๆ เช่นงานของ Ye และคณะ [42] Sun และคณะ [73] และ Sabor และคณะ [75] เป็นต้น และในงานวิจัยนี้ได้ประยุกต์ใช้ฟังก์ชัน REP [73] ตามสมการ (3.9) เพื่อจะนำข้อมูลจากรายงานจุดบกพร่องในส่วน Description Product Component Priority และ Severity เข้ามาเพื่อประเมินความคล้ายด้วย

$$REP(d,q) = \sum_{i=1}^{n} w_i \times feature_i \tag{3.9}$$

จากสมการ (3.9) จะเป็นการใช้ฟังก์ชัน REP ที่ภายในจะมีการเปรียบเทียบในคุณลักษณะหลายคุณลักษณะ (*feature*) และแต่ละคุณลักษณะจะต้องมีค่าความสำคัญ (Weight) ของแต่ละคุณลักษณะนั่นคือ *w* โดยที่ *i* คือหมายเลขคุณลักษณะ และ *n* คือจำนวนของคุณลักษณะที่ใช้ในการเปรียบเทียบระหว่างรายงานจุดบกพร่อง *d* และ *q* ซึ่งสามารถเรียก ฟังก์ชัน REP ในที่นี้ได้ว่าฟังก์ชัน *sim*() เพื่อหาความคล้ายของรายงานจุดบกพร่องเป็น *sim*(*d*, *q*)

เนื่องจากสมการ (3.9) มีพารามิเตอร์ *w* ของแต่ละคุณลักษณะและจำเป็นต้องหาค่าเหล่านี้ที่เหมาะสมด้วย ดังนั้นจากการศึกษางานวิจัย [73, 75, 77] ได้ใช้วิธีการเคลื่อนลงตามความชัน (Gradient descent) และใช้ RankNet Cost function: RNC เป็นสมการคำนวณหาค่า cost เพื่อปรับค่าพารามิเตอร์อิสระ ซึ่งสามารถสรุปกระบวนการปรับค่าพารามิเตอร์ *w* (ในที่นี้เรียกว่าพารามิเตอร์อิสระ) เพื่อให้ได้ค่าที่เหมาะสมดังนี้

ขั้นตอนที่ 1: กำหนดฟังก์ชัน *sim*() เป็นไปตามสมการ (3.10)

$$sim(d,q) = REP(d,q) = \sum_{i=1}^{n} w_i \times feature_i \tag{3.10}$$

ขั้นตอนที่ 2: สร้างข้อมูลชุดสอนโดยใช้ข้อมูลบางส่วน เพื่อใช้ในการปรับค่าพารามิเตอร์ภายในฟังก์ชัน *sim*() (Stochastic gradient descent) ซึ่งสามารถแสดงขั้นตอนนี้ได้ตามรูปที่ 3.23

ขั้นตอนที่ 3: กำหนดรอบ (epoch) ในการปรับค่าพารามิเตอร์ และกำหนดค่าเริ่มต้นให้กับแต่พารามิเตอร์อิสระ

ขั้นตอนที่ 4: ใช้สมการ RNC เพื่อใช้เป็น cost function ในการปรับค่าพารามิเตอร์อิสระ ดังสมการ (3.11)

$$RNC = log(1 + e^{Y}) \ where \ Y = sim(irr, q) - sim(rel, q) \quad (3.11)$$

ขั้นตอนที่ 5: ในแต่ละการประมวลผลของแต่ละ Instance *(I)* จะถูกนำมาคำนวณหาค่าเพื่อปรับค่าพารามิเตอร์อิสระ ด้วยการทำอนุพันธ์บางส่วน (Partial derivative) กับค่าพารามิเตอร์แต่ละตัวโดยประยุกต์จากสมการพื้นฐาน RankNet Cost Derivatives ดังสมการ (3.12) และสามารถแสดงขั้นตอนนี้ได้ตามรูปที่ 3.24

$$\frac{\partial RNC(Y)}{\partial Y} \equiv RNC'(Y) = \frac{e^{Y}}{1 + e^{Y}} \quad (3.12)$$

Algorithm	:	Constructing a Training Set from a Training Dataset
Input	:	$TS = \emptyset$: resultant training set
		M is set of Meta-bug from a Training Dataset
		B is set of dependent bug report of each Meta-bug from a Training Dataset
		$N = 30$: parameter controlling the size of TS
Output	:	Training Set for tuning parameter in REP
1	:	**for each** M in Training Dataset **do**
2	:	$R = B_{Mi}$: B_{Mi} is set of dependent bug report of current Meta-bug
3	:	$q = M_i$: M_i is current Meta-bug that considering
4	:	**for each** *rel* in R **do**
5	:	**for** $i = 1$ to N **do**
6	:	**randomly** choose a bug report *irr* , where $irr \notin R$
7	:	$TS = TS \cup \{(q, rel, irr)\}$
8	:	**end**
9	:	**end**
10	:	**end**
11	:	**return** *TS*

รูปที่ 3.23 ขั้นตอนวิธีการสร้างชุดสอนสำหรับปรับค่าพารามิเตอร์ในฟังก์ชันการค้นคืน REP

Algorithm : Parameter Tuning
Input : TS = : a training set
$N = 24$: the times to iterate through TS (or epoch)
$\eta = 0.001$: the tuning rate
Output : Adjusted parameter x
1 : **for** $i = 1$ to N **do**
2 : **for** each instance $I \in TS$ in random order **do**
3 : **for** each free parameter x in $sim()$ **do**
4 : $$x = x - \eta \times \frac{\partial RNC}{\partial x}(I)$$
5 : **end**
6 : **end**
7 : **end**

รูปที่ 3.24 ขั้นตอนวิธีการปรับค่าพารามิเตอร์ในฟังก์ชัน REP โดยย่อ

ขั้นตอนที่ 6: ในรอบของ epoch ถัดไปจะใช้ค่าพารามิเตอร์ ที่ถูกปรับค่าจากรอบที่แล้วเป็นค่าเริ่มต้น และจำนวนของข้อมูลชุดสอนจะถูกปรับเหลือ 85% เพื่อใช้ในรอบถัดไป

ขั้นตอนที่ 7: หากค่าพารามิเตอร์อิสระในแต่ละ epoch มีค่าเท่ากันติดต่อกัน 3 epoch จะหยุดการปรับค่าพารามิเตอร์แต่ถ้าไม่ จะทำจนครบตามจำนวนของ epoch

ผลลัพธ์ที่ได้ตามขั้นตอนดังกล่าวจะได้ค่าพารามิเตอร์อิสระที่ปรับค่าเรียบร้อย แล้วจึงนำค่าเหล่านี้ไปใช้งานตามฟังก์ชัน REP

โดยข้อมูลที่ใช้ในการทดลองตามวิธีการนี้ ได้กำหนดชุดข้อมูลทดสอบเป็น 4,781 รายงานที่มี Meta-bug 200 รายงาน และข้อมูลเพื่อสร้างชุดสอนในการปรับค่าพารามิเตอร์จำนวน 10% 478 รายงาน และมี Meta-bug 25 รายงาน เพื่อสร้างชุดข้อมูลที่ใช้ในการสอนจะชุดข้อมูลเหล่านี้จะอยู่ในรูปแบบสามเอกสารคือ (*q*, *rel*, *irr*) ซึ่ง *q* คือ รายงานจุดบกพร่องที่เป็น Meta-bug ในขณะ *rel* คือรายงานจุดบกพร่องที่เป็นส่วนต่อของ *q* และ *irr* คือ รายงานจุดบกพร่องที่ไม่เป็นส่วนต่อของ *q*

ในส่วนของข้อมูลที่นำมาใช้ในการทดลองนี้แบ่งได้เป็นสองกลุ่มคือ กลุ่มแรกคือข้อมูลที่เป็นข้อความ ได้แก่ Summary และ Description ในขณะที่กลุ่มที่สองเป็นข้อมูลหมวดหมู่ ประกอบไปด้วย Product Component Priority และ Severity โดยในส่วนข้อมูลที่เป็นข้อความจะใช้เทคนิคในการหาความคล้าย ซึ่งได้แยกย่อยออกเป็นอีกสามเทคนิคด้วยกันคือ การหาความคล้ายด้วยโคไซน์ BM25F และ MATF ในกรณีของการใช้ BM25F นั้นเนื่องจากเป็นส่วนขยายของ BM25 อยู่แล้วและเป็นเทคนิคที่เหมาะกับการใช้กับเอกสารที่มีฟิลด์ข้อมูลที่เป็นข้อความมากกว่า 1 ฟิลด์ ซึ่งจะได้นำเสนอในลำดับถัดไป ในขณะที่ส่วนข้อมูลที่เป็นข้อมูลหมวดหมู่จะใช้สมการดังต่อไปนี้เป็นส่วนหนึ่งของ ฟังก์ชัน REP ในงานวิจัยฉบับนี้

$$feature_{product}(d,q) = \begin{cases} 1, & if\ d.product = q.product \\ 0, & otherwise \end{cases} \quad (3.13)$$

$$feature_{component}(d,q) = \begin{cases} 1, & if\ d.component = q.component \\ 0, & otherwise \end{cases} \quad (3.14)$$

$$feature_{priority}(d,q) = \frac{1}{1+|d.priority - q.priority|} \quad (3.15)$$

$$feature_{severity}(d,q) = \frac{1}{1+|d.severity - q.severity|} \quad (3.16)$$

3.9.1 ฟังก์ชันการค้นคืนด้วยเทคนิคโคไซน์ (REP with cosine similarity)

ฟังก์ชันการค้นคืนด้วยเทคนิคโคไซน์ เป็นวิธีการกำหนดให้ฟังก์ชันการค้นคืน REP ใช้วิธีการเปรียบเทียบความคล้ายระหว่างรายงานจุดบกพร่องในส่วนคุณลักษณะที่เป็นข้อความด้วยโคไซน์ ซึ่งสามารถสรุปคุณลักษณะของฟังก์ชันการค้นคืนด้วยโคไซน์ดังต่อไปนี้

ตารางที่ 3.17 คุณลักษณะในฟังก์ชันการค้นคืนด้วยโคไซน์

คุณลักษณะ	สมการ
$feature_1$	$cosine_{summary}(d, q)$ ที่ใช้ Unigram+compound words เป็นคุณลักษณะ
$feature_2$	$cosine_{description}(d, q)$ ที่ใช้ Unigram+compound words เป็นคุณลักษณะ
$feature_3$	$feature_{product}(d, q)$ ตามสมการ (3.13)
$feature_4$	$feature_{component}(d, q)$ ตามสมการ (3.14)
$feature_5$	$feature_{priority}(d, q)$ ตามสมการ (3.15)
$feature_6$	$feature_{severity}(d, q)$ ตามสมการ (3.16)

3.9.2 ฟังก์ชันการค้นคืนด้วยเทคนิค BM25F (REP with BM25F)

ฟังก์ชันการค้นคืนด้วยเทคนิค BM25F นั่นเป็นการประยุกต์ใช้ BM25F ที่เหมาะการค้นคืนและลำดับเอกสารที่มีฟิลด์ข้อความมากกว่าหนึ่งฟิลด์ ซึ่งสมการ BM25F แสดงได้ดังนี้

$$BM25F(d,q) = \sum_{t \in d \cap q} IDF(t) \times \frac{TF_D(d,t)}{k_1 + TF_D(d,t)} \quad (3.17)$$

โดยที่ $IDF(t)$ คือ

$$IDF(t) = log\frac{N}{df_t} \quad (3.18)$$

โดย N คือจำนวนเอกสารทั้งหมด ขณะที่ df_t คือจำนวนเอกสารที่มีคำ t ปรากฏ

และ $TF_D(d, t)$ เป็นการรวมกันของความสำคัญของคำ t ใน เอกสาร d ที่ปรากฏในแต่ละฟิลด์ ซึ่งมีแนวคิดที่ว่าความถี่คำ t ที่ปรากฏในเอกสาร d ในแต่ละฟิลด์ย่อมมีความสำคัญแตกต่างกัน ดังนั้น $TF_D(d, t)$ จึงมีสมการเป็นดังต่อไปนี้

$$TF_D(d,t) = \sum_{f=1}^{K} \frac{w_f \times occurrence(d[f],t)}{1 - b_f + \frac{b_f \times length_f}{avg\_length_f}} \quad (3.19)$$

จากสมการ (3.19) K คือจำนวนจำนวนฟิลด์ที่เป็นข้อความในเอกสาร d ส่วน f คือฟิลด์ในเอกสาร d ขณะที่ $occurrence(d[f], t)$ คือ ความถี่ของคำ t ในฟิลด์ f ของเอกสาร d

w_f คือค่าความสำคัญของฟิลด์ f ในขณะที่ $length_f$ คือความยาวของฟิลด์ f ส่วน avg_length_f คือ ความยาวเฉลี่ยของฟิลด์ f จากเอกสารทั้งหมด ในขณะที่ b_f ค่าอยู่ระหว่าง 0-1 เพื่อควบคุมความ bias ของเอกสารที่ยาวและสั้นต่างกัน

ดังนั้นจากสมการ BM25F ที่มีการให้ความสำคัญของค่าความถี่ของคำในแต่ละฟิลด์ของเอกสารดังนั้นรายงานจุดบกพร่องที่ใช้ข้อมูลข้อความจากส่วน Summary และ Description จะถูกประมวลผลตามสมการของ BM25F ที่ให้ความสำคัญของคำที่ปรากฏในแต่ละฟิลด์ที่แตกต่างกันอยู่แล้ว ด้วยเหตุนี้ คุณลักษณะที่ใช้ในฟังก์ชันการค้นคืนด้วย BM25F จึงเป็นไปตาม ตารางที่ 3.18

ตารางที่ 3.18 คุณลักษณะในฟังก์ชันการค้นคืนด้วยเทคนิค BM25F

คุณลักษณะ	**สมการ**
$feature_1$	$BM25F(d, q)$ ที่ใช้ Unigram + compound words เป็นคุณลักษณะ
$feature_2$	$feature_{product}(d, q)$ ตามสมการ (3.13)
$feature_3$	$feature_{component}(d, q)$ ตามสมการ (3.14)
$feature_4$	$feature_{priority}(d, q)$ ตามสมการ (3.15)
$feature_5$	$feature_{severity}(d, q)$ ตามสมการ (3.16)

3.9.3 ฟังก์ชันการค้นคืนด้วยเทคนิค MATF (REP with MATF)

ฟังก์ชันการค้นคืนด้วยเทคนิค MATF เป็นวิธีการกำหนดให้ฟังก์ชันการค้นคืน REP ใช้วิธีการเปรียบเทียบความคล้ายระหว่างรายงานจุดบกพร่องในส่วนคุณลักษณะที่เป็นข้อความด้วย MATFซึ่งสามารถสรุปคุณลักษณะของฟังก์ชันการค้นคืนด้วย MATF ดังต่อไปนี้

ตารางที่ 3.19 คุณลักษณะในฟังก์ชันการค้นคืนด้วยเทคนิค MATF

คุณลักษณะ	สมการ
$feature_1$	$MATF_{summary}(d, q)$ ที่ใช้ Unigram + compound words เป็นคุณลักษณะ
$feature_2$	$MATF_{description}(d, q)$ ที่ใช้ Unigram + compound เป็นคุณลักษณะ
$feature_3$	$feature_{product}(d, q)$ ตามสมการ (3.13)
$feature_4$	$feature_{component}(d, q)$ ตามสมการ (3.14)
$feature_5$	$feature_{priority}(d, q)$ ตามสมการ (3.15)
$feature_6$	$feature_{severity}(d, q)$ ตามสมการ (3.16)

บทที่ 4
ผลการวิจัย

ในบทนี้จะเป็นการแสดงรายละเอียดของผลลัพธ์จากกระบวนการรวบรวมข้อมูล กระบวนการเบื้องต้น(การสร้างกฎเชิงไวยกรณ์) การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ ซึ่งมีผลการดำเนินงานตามแต่ละขั้นตอนดังกล่าวดังนี้

4.1 ผลการการทดสอบกฎเชิงไวยกรณ์สำหรับกลุ่มคำนาม

ในการสร้างกฎเชิงไวยากรณ์สำหรับกลุ่มคำนามแบบสองคำ นั้น ได้ใช้รายงานจุดบกพร่องจากซอฟต์แวร์มอซิลลา Core จำนวน 1,300 รายงาน ตามที่เสนอไว้ในตารางที่ 3.1 โดยที่แบ่งชุดข้อมูลดังกล่าวออกเป็น ชุดข้อมูลในการสร้างกฎจำนวน 1,000 รายงาน (3,241 ประโยค) และชุดข้อมูลที่ใช้ในการทดสอบกฎจำนวน 300 รายงาน (1,174 ประโยค)

ภายหลังจากการสร้างกฎเชิงไวยากรณ์สำหรับกลุ่มคำนาม เสร็จเรียบร้อย ได้กฎทั้งสิ้น 57 กฎ (ที่เสนอไว้ในบทที่ 3 ตารางที่ 3.9) และเมื่อทำการทดสอบการสกัดกลุ่มคำนามด้วยกฎเชิงไวยากรณ์ด้วยค่าความระลึก จากเอกสารจำนวน 300 เอกสาร ที่มีจำนวนกลุ่มคำนามทั้งสิ้น 234 กลุ่มคำ ซึ่งสามารถแสดงผลการทดสอบตามสมการหาค่าความระลึกได้ดังต่อไปนี้

$$Recall = \frac{TP}{TP + FN} = \frac{195}{195 + 39} = 0.83$$

จากสมการหาค่าความระลึกคำนวณได้ค่าความระลึกเท่ากับ 0.83 ซึ่งพบว่ามีข้อผิดพลาดเกิดขึ้น อันเนื่องมาจากการเขียนผิดหลักไวยากรณ์ หรือเขียนคำศัพท์ผิด เช่น ในประโยค “Show/Hide scrollbars depending on activity for B2G.” จะเห็นว่าคำว่า “scrollbars” ในประโยคนั้น ในความเป็นจริงควรจะเป็นคำนาม แต่ Stanford Parser กลับระบุว่าคำนี้เป็นอยู่ในประเภทคำกริยา นั่นคือ VBZ ดังนั้นกฎเชิงไวยากรณ์สำหรับกลุ่มคำนามที่มีอยู่ จึงไม่สามารถสกัดกลุ่มคำนามจากประโยคนี้ได้

อย่างไรก็ตาม การสกัดกลุ่มคำนามนั้น เป็นเพียงการพยายามที่จะขยายจำนวนคุณลักษณะเพิ่มจากการใช้คำเดี่ยว (Unigram) เพียงอย่างเดียว เพื่อลดปัญหาข้อความสั้นเกินไป (Short text) นั่นเอง [94-96] ที่อาจจะทำให้การวิเคราะห์ผลทางด้านข้อความขาดประสิทธิภาพ เพราะคุณลักษณะหรือคำในข้อความที่จะนำมาวิเคราะห์มีน้อยเกินไป [94] ดังนั้น ข้อผิดพลาดดังกล่าวจึงไม่น่าจะส่งผลกระทบกับการวิเคราะห์การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อมากนัก

4.2 ผลการทดลองการวิเคราะห์เพื่อการทดสอบการใช้คุณลักษณะผ่านเทคนิคการจำแนกข้อมูล

จากการศึกษาที่ผ่านมา หลายงานวิจัยอาทิเช่น [3, 10, 11, 18, 19, 40-42, 73, 97] พบว่าจะมีใช้คุณลักษณะของรายงานจุดบกพร่องที่แตกต่างใน 3 รูปแบบ นั่นคือ คือ (1) Unigram (2) Bigram และ (3) Unigram + compound words และไม่มีงานวิจัยใดยืนยันว่าคุณลักษณะของรายงานจุดบกพร่องรูปแบบใดข้างต้น ที่เหมาะสมในการวิเคราะห์รายงานจุดบกพร่องมากที่สุด ซึ่งในงานวิจัยนี้ก็จะใช้คุณลักษณะของรายงานจุดบกพร่องตามที่เคยมีการศึกษามาเป็นบรรทัดฐาน (Baseline) แต่จะเพิ่มการศึกษาคุณลักษณะของรายงานจุดบกพร่องที่เป็นแบบรวม หรือ

Combination ด้วย เพื่อแสดงให้เห็นถึงคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมกับการศึกษาในครั้งนี้

อย่างไรก็ตาม เพื่อเลือกรูปแบบคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมที่สุด ในการศึกษานี้ได้ทำการศึกษาเบื้องต้นผ่านงานวิจัยด้านการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug และ การทำนายระดับความรุนแรงของรายงานจุดบกพร่องจุดบกพร่อง ซึ่งล้วนเป็นงานวิจัยที่ประยุกต์เทคนิคด้านการจำแนกข้อมูล (Classification technique) เข้ามาใช้

ซึ่งผลของการศึกษาเรื่องคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมผ่านการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug และการศึกษาการทำนายระดับความรุนแรงของจุดบกพร่อง สามารถแสดงได้ดังนี้

4.2.1 ผลการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug

จากการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug ได้ใช้ชุดข้อมูลดังที่แสดงในตารางที่ 3.2 เพื่อสร้างแบบจำลองในการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug โดยใช้อัลกอริทึมการเรียนรู้แบบมีผู้สอน 3 อัลกอริทึม ได้แก่ นาอีฟย์เบย์ (Naïve Bayes: NB) การถดถอยโลจิสติก (Logistic regression: LR) และ ซัพพอร์ตเวกเตอร์แมชชีน (Support vector machine: SVM) ซึ่งในการทดลองได้ศึกษาเชิงเปรียบเทียบกับคุณลักษณะทั้ง 7 แบบ (ดังที่กล่าวไว้ในหัวข้อ 3.4) ซึ่งในการศึกษานี้ใช้เทคนิคการให้น้ำหนักคำแบบ tf และผลการทดลองสามารถแสดงได้ดังตารางที่ 4.1 และ ตารางที่ 4.2

ตารางที่ 4.1 ผลการศึกษาในชุดข้อมูลของ Herzig

รูปแบบคุณลักษณะ	NB			LR			SVM		
	R	P	F1	R	P	F1	R	P	F1
1	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.65**	**0.61**	**0.63**
2	0.58	0.54	0.56	0.56	0.53	0.55	0.59	0.51	0.54
3	0.76	0.75	0.76	**0.77**	**0.77**	**0.77**	0.62	0.60	0.61
4	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	0.61	0.52	0.56
5	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	**0.77**	0.62	0.58	0.60
6	0.76	0.76	0.76	**0.77**	**0.77**	**0.77**	0.59	0.52	0.56
7	0.76	0.76	0.76	**0.77**	**0.77**	**0.77**	0.60	0.53	0.56

ตารางที่ 4.2 ผลการศึกษาในชุดข้อมูล Mozilla Firefox

Feature type	NB			LR			SVM		
	R	P	F1	R	P	F1	R	P	F1
1	0.56	0.66	0.61	0.65	0.67	0.66	0.62	0.66	0.64
2	0.53	0.64	0.58	0.63	0.65	0.64	0.60	0.64	0.62
3	0.53	0.63	0.58	0.63	0.65	0.64	0.61	0.63	0.62
4	0.62	0.67	0.64	0.71	0.71	0.71	0.67	0.67	0.67
5	**0.65**	**0.69**	**0.67**	**0.72**	**0.73**	**0.72**	**0.69**	**0.69**	**0.69**
6	0.60	0.64	0.62	0.69	0.69	0.69	0.67	0.67	0.67
7	**0.65**	**0.69**	**0.67**	0.71	0.72	0.71	**0.69**	**0.69**	**0.69**

จากผลการทดลองดังตารางที่ 4.1 จะเห็นว่าคุณลักษณะของรายงานจุดบกพร่องแบบที่ (4) คือ Unigram + bigram คุณลักษณะแบบ (5) นั่นคือ Unigram + compound words คุณลักษณะแบบที่ (6) คือ Bigram + compound words และคุณลักษณะแบบที่ (7) คือ Combination จะให้ผลลัพธ์ที่น่าพอใจ และเมื่อนำผลลัพธ์ที่ได้ไปเปรียบเทียบกับงานวิจัยก่อนหน้า นั่นคือ [38, 98, 99] ที่มีการนำเสนอผลลัพธ์เฉพาะค่า F1 โดยอยู่ที่ระหว่าง 0.674 -.0763 จะพบว่าผลลัพธ์ในตารางที่ 4.6 จะดีกว่า โดยเฉพาะอย่างยิ่งตัวจำแนกข้อมูลที่สร้างจากอัลกอริทึมการถดถอยโลจิสติกจะให้ผลลัพธ์ที่ดีที่สุด

และในขณะที่ตารางที่ 4.2 ที่เป็นการทดลองกับชุดข้อมูลของมอซิลลา Firefox แสดงให้เห็นว่าผลลัพธ์ในการจำแนกจากทั้ง 3 แบบจำลองให้ผลลัพธ์ในการจำแนกรายงานจุดบกพร่องที่น่าพอใจ ซึ่งตัวจำแนกที่สร้างจากอัลกอริทึมการถดถอยโลจิสติกยังให้ผลลัพธ์ที่ดีที่สุดเช่นเดิม โดยเฉพาะเมื่อใช้คุณลักษณะของรายงานจุดบกพร่องในแบบที่ (5) คือ Unigram + compound words และผลการทดลองดังกล่าว ให้ค่าใกล้เคียงกับงานวิจัยก่อนหน้า [19, 38, 99] แม้ว่าจะใช้จำนวนของชุดข้อมูลที่มีขนาดเล็กกว่าในการสร้างตัวจำแนกข้อมูล

จากผลการวิจัยข้างต้น จะเห็นได้ว่าการใช้คุณลักษณะของรายงานจุดบกพร่องแบบ Unigram + bigram, Unigram + compound words และ Combination น่าจะเป็นคุณลักษณะที่เหมาะสมของรายงานจุดบกพร่อง เพราะให้ผลการทดลองที่น่าพอใจ โดยเฉพาะอย่างยิ่งในงานวิจัยด้านการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug

อย่างไรก็ตามการใช้คุณลักษณะของรายงานจุดบกพร่องที่เป็น Unigram + bigram และ Combination จะใช้เวลาในการประมวลผลเพื่อตัดคำมากกว่าคุณลักษณะของรายงานจุดบกพร่องที่เป็น Uunigram + compound words ดังนั้นคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมทั้งในเรื่องของผลลัพธ์และเวลาน่าจะเป็น Unigram + compound words

4.2.2 ผลการศึกษาการทำนายระดับความรุนแรงของจุดบกพร่อง

จากการศึกษาการทำนายระดับความรุนแรงของจุดบกพร่อง เพื่อสร้างโมเดลสำหรับทำนายความรุนแรงให้กับรายงานจุดบกพร่องโดยใช้ชุดข้อมูลตามตารางที่ 3.3 โดยใช้อัลกอริทึมการ

เรียนรู้แบบมีผู้สอน เช่นเดียวกันกับที่ใช้ในการศึกษาการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug แต่ได้เพิ่มอัลกอริทึมป่าสุ่มเข้ามาใช้ในการสร้างโมเดลด้วย

ซึ่งในการทำนายระดับความรุนแรงของจุดบกพร่องที่มีการศึกษาเชิงเปรียบเทียบคุณลักษณะของรายงานจุดบกพร่อง 2 รูปแบบคือ Unigram และ Unigram + compound words เนื่องจากรูปแบบการศึกษานี้สอดคล้องกับการศึกษาของ Lamkanfi และคณะ [14] นอกจากการศึกษาเชิงเปรียบเทียบเกี่ยวกับคุณลักษณะของรายงานจุดบกพร่องแล้ว งานวิจัยนี้ยังได้ศึกษาเชิงเปรียบเทียบการให้น้ำหนักคำใน 3 เทคนิคด้วย เทคนิคเหล่านั้นได้แก่ *tf tf-idf* และ *tf-igm* ซึ่งผลลัพธ์จากการทดสอบการทำนายระดับความรุนแรงของจุดบกพร่องสามารถแสดงดังตารางที่ 4.3 และตารางที่ 4.4

ตารางที่ 4.3 ผลการศึกษาการทำนายระดับความรุนแรงของรายงานจุดบกพร่องเมื่อใช้ Unigram เป็นคุณลักษณะ

Weighting Scheme	NB			KNN			SVM			RF		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
tf	0.79	0.79	0.79	0.72	0.72	0.72	0.79	0.80	0.79	0.75	0.77	0.75
tf-idf	0.77	0.78	0.77	0.63	0.64	0.62	0.79	0.79	0.79	0.75	0.77	0.75
tf-igm	0.77	0.78	0.77	0.71	0.72	0.71	**0.84**	**0.84**	**0.84**	0.75	0.77	0.75

ตารางที่ 4.4 ผลการศึกษาการทำนายระดับความรุนแรงของรายงานจุดบกพร่องเมื่อใช้ Unigram + compound words เป็นคุณลักษณะ

Weighting Scheme	NB			KNN			SVM			RF		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
tf	0.81	0.82	0.81	0.71	0.71	0.71	0.78	0.78	0.78	0.77	0.79	0.77
tf-idf	0.78	0.78	0.78	0.64	0.64	0.63	0.80	0.80	0.80	0.77	0.79	0.77
tf-igm	0.78	0.79	0.78	0.70	0.71	0.70	**0.84**	**0.85**	**0.84**	0.77	0.79	0.77

จากผลการศึกษาในตารางที่ 4.3 และตารางที่ 4.4 พบว่าการใช้คุณลักษณะรายงานจุดบกพร่องที่เป็น Unigram + compound words จะให้ผลลัพธ์ที่น่าพอใจในทุกๆ ตัวแบบที่ใช้ในการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง โดยเฉพาะตัวแบบที่ใช้ร่วมกับการให้น้ำหนักคำแบบ *tf-igm* เพราะภายหลังจากการทดลอง ปรากฏว่าให้ผลที่ดีกว่าผลการทดลองงานวิจัยของ Lamkanfi และคณะ [14] ที่เป็นบรรทัดฐาน โดยมีการใช้ข้อมูลรายงานจุดบกพร่องจากมอซิลลา Firefox เช่นเดียวกัน

จากผลการทดลองที่แสดงในหัวข้อที่ 4.2.1 และ 4.2.2 จะเห็นได้ว่าการใช้คุณลักษณะของรายงานจุดบกพร่องที่เป็น Unigram + compound words ให้ผลลัพธ์ที่น่าพอใจในทั้งสองการศึกษา และเมื่อพิจารณาจากงานวิจัยของ [19, 40-42] ร่วมด้วย ก็มีความคิดเห็นที่สอดคล้องกันว่า

“Unigram + compound words” เป็นคุณลักษณะที่น่าจะเหมาะสมสำหรับการศึกษาที่เกี่ยวข้องกับรายงานจุดบกพร่อง

4.3 ผลการจัดกลุ่มการเป็นส่วนต่อของรายงานจุดบกพร่องด้วยเทคนิคการจัดกลุ่ม Clustering

ในการทดลองการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ เบื้องต้นได้ดำเนินการสุ่มข้อมูลรายงานจุดบกพร่องที่เป็น Meta-bug มาจำนวน 100 รายงาน โดย Meta-bug เหล่านี้จะมีรายงานจุดบกพร่องที่เป็นส่วนต่อ (Dependent bug reports) จำนวนทั้งสิ้น 2,650 รายงาน

เริ่มแรกรายงานจุดบกพร่องเหล่านั้นจะผ่านกระบวนการเตรียมข้อมูล โดยคุณลักษณะของรายงานจุดบกพร่องที่เลือกใช้มี 4 รูปแบบตามที่ได้ข้อสรุปจากหัวข้อที่ 4.2.1 และ 4.2.2 นั่นคือ Unigram, Unigram + bigram, Unigram + compound words, และ Combination จากนั้นรายงานจุดบกพร่องเหล่านี้จะถูกแสดงในรูปแบบถุงคำ ที่มีการให้น้ำหนัก 4 แบบได้แก่ *tf*, *tf-idf*, *BM25* และ *MATF*

โดยในการทดลองการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่อยู่ภายใต้แต่ละ Meta-bug ด้วยเทคนิคการจัดกลุ่ม Clustering จะทำการทดสอบการจัดกลุ่ม 4 ครั้ง เริ่มจากการใช้ Meta-bug เป็น Centroid ครั้งละ 25, 50, 75 และ 100 กลุ่มตามลำดับ

อย่างไรก็ตาม ภายหลังผ่านกระบวนการตัดคำและตัดคำหยุดแล้ว จะได้คุณลักษณะที่จะเป็นตัวแทนของรายงานจุดบกพร่องดังตารางที่ 4.5

ตารางที่ 4.5 จำนวนคุณลักษณะของรายงานจุดบกพร่องทั้ง 4 แบบ

จำนวนกลุ่ม	จำนวนรายงานจุดบกพร่อง	คุณลักษณะของรายงานจุดบกพร่อง			
		Unigram	Unigram + bigram	Unigram + compound words	Combination
25	1,006	1,388	3,300	1,536	3,448
50	1,483	1,758	4,532	1,937	4,711
75	2,136	2,425	6,543	2,745	6,854
100	2,650	2,813	7,750	3,222	8,159

4.3.1 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบเพิ่มข้อจำกัด

การจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัดมีผลการทดลองที่สามารถแสดงได้ดัง ตารางที่ 4.7 ซึ่งแสดงให้เห็นว่าคุณลักษณะของรายงานจุดบกพร่องแบบ Unigram + compound words เป็นคุณลักษณะที่น่าจะเหมาะสมต่อการศึกษานี้ เพราะเมื่อพิจารณาจากการให้น้ำหนักในแบบต่างๆ ได้แก่ *tf*, *tf-idf*, *BM25* หรือ *MATF* ปรากฏว่า Unigram + compound words ให้ผลลัพธ์ที่ดีที่สุดสำหรับการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบเพิ่มข้อจำกัด เพราะไม่ว่าจะเป็นการจัดกลุ่มที่มี Meta-bug ที่จำนวน 25 รายงาน, 50 รายงาน, 75 รายงาน และ 100 รายงานตามลำดับ อาจจะเป็นเพราะว่า Compound words เป็นคำที่สามารถใช้บ่งบอกลักษณะเฉพาะในซอฟต์แวร์ก็เป็นได้

ยกตัวอย่างเช่น Compound words อาจจะเป็นชื่อฟังก์ชัน หรือเครื่องมือ ที่ใช้ในซอฟต์แวร์นั้นๆ ดังนั้นเมื่อใช้คุณลักษณะนี้เข้ามาเป็นคุณลักษณะร่วมของ Unigram จึงทำช่วยในฟังก์ชันที่ใช้ในการจัดกลุ่มทำงานได้ดีขึ้น โดยเฉพาะอย่างยิ่งเมื่อมีการให้น้ำหนักแบบ *tf* เพราะ *tf* เป็นการให้น้ำหนักแบบภายใน จึงเน้นให้ความสำคัญกับคำที่พบในพื้นที่เอกสารนั้นๆ แน่นอนว่าค่าน้ำหนักของคำเดียวกันในแต่ละเอกสารจึงมีความแตกต่าง ซึ่งค่าที่แตกต่างกันนี้เป็นการบ่งบอกว่า “คำ” คำนั้นในเอกสารมีความสำคัญต่างกันอย่างไร ซึ่งข้อสรุปนี้มีความสอดคล้องกับงานวิจัยของ [19, 40-42]

อย่างไรก็ตามการให้น้ำหนักคำแบบ MATF ก็ให้ผลลัพธ์ที่ไม่แตกต่างจากการให้น้ำหนักแบบ *tf* มากนัก แต่จะใช้เวลาในการประมวลผลนานกว่าการให้น้ำหนักแบบ *tf* มาก

อย่างไรก็ตาม จะเห็นว่าประสิทธิภาพของการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบเพิ่มข้อจำกัดยังไม่น่าพอใจ เพราะค่าความระลึก ค่าความแม่นยำ ค่า F1 และค่าความถูกต้องยังค่อนข้างต่ำ ดังนั้นงานวิจัยนี้จึงได้ทำการศึกษาการจัดกลุ่มแบบคลัสเตอริง ในรูปแบบอื่น นั่นคือการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัด

4.3.2 การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัด

สำหรับการทดลองในส่วนนี้ ก็ปรากฏว่า Unigram + compound words น่าจะเป็นคุณลักษณะที่เหมาะสมต่อการศึกษานี้ เพราะเมื่อพิจารณาจากการที่คำที่ปรากฏในรายงานจุดบกพร่องได้ถูกให้น้ำหนักด้วย *tf*, *tf-idf*, *BM25* หรือ *MATF* จากนั้นนำไปทำการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบทรงกลมที่เพิ่มข้อจำกัด ปรากฏว่า Unigram + compound words ยังให้ผลลัพธ์ที่ดีที่สุด ไม่ว่าจะเป็นการจัดกลุ่มที่มี Meta-bug ที่จำนวน 25 รายงาน, 50 รายงาน, 75 รายงาน และ 100 รายงานตามลำดับ

อย่างไรก็ตาม จะเห็นว่าประสิทธิภาพของการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบทรงกลมที่เพิ่มข้อจำกัดก็ยังไม่น่าพอใจ เพราะค่าความระลึก ค่าความแม่นยำ ค่า F1 และค่าความถูกต้องยังค่อนข้างต่ำ ดังนั้นงานวิจัยนี้จึงได้ทำการศึกษาการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยวิธีการอื่น ดังที่จะแสดงในหัวข้อ 4.4, และ 4.5

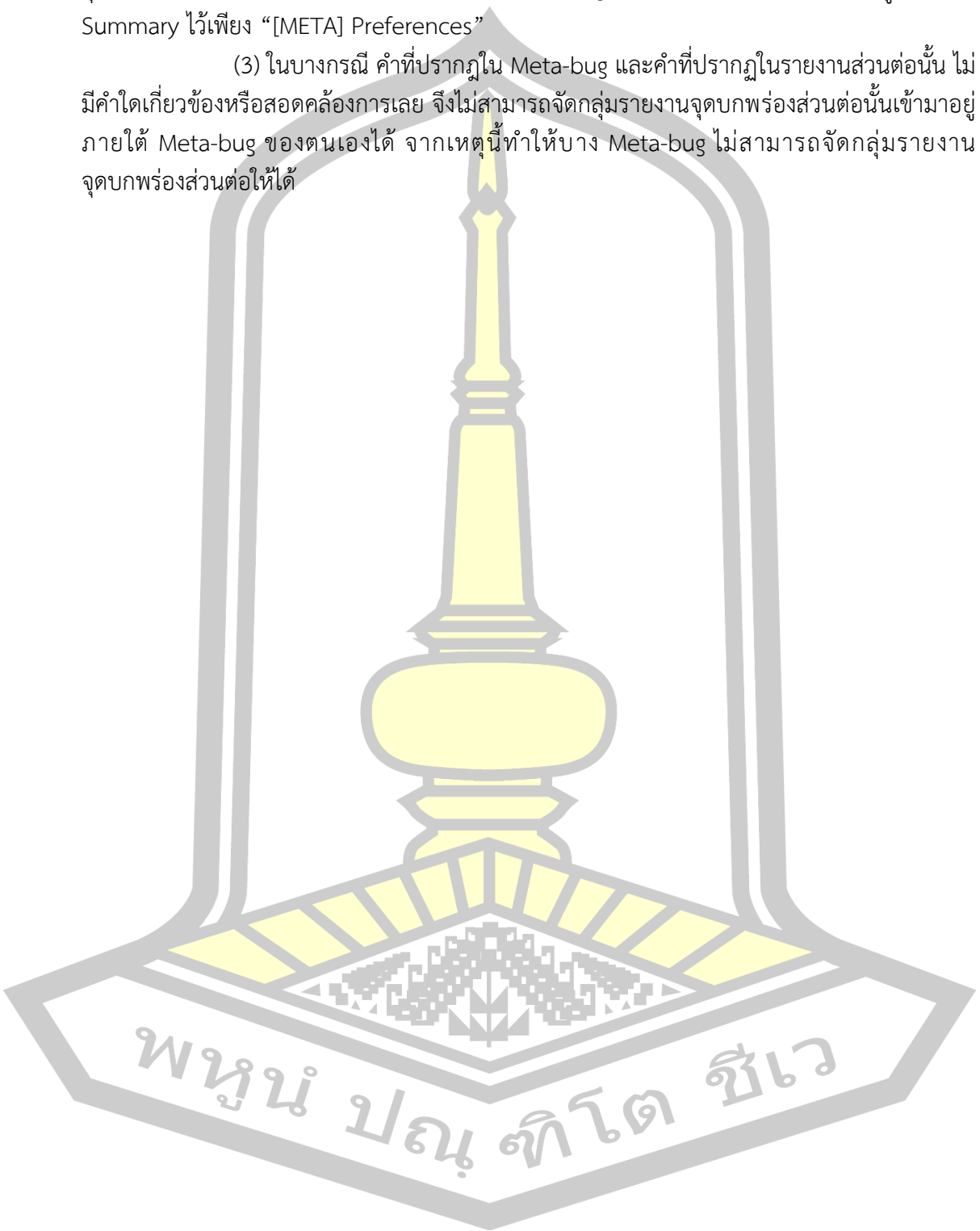
นอกจากนี้ หากพิจารณาภาพรวมของปัญหาที่ทำให้การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบเพิ่มข้อจำกัดและการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัดมีประสิทธิภาพที่ไม่ดีนัก อาจจะเนื่องมากจาก

(1) จำนวนรายงานที่เป็นส่วนต่อกันในแต่ละ Meta-bug มีจำนวนแตกต่างกันมาก เช่น บาง Meta-bug มีมากถึง 259 รายงาน ในขณะที่บาง Meta-bug มีเพียง 5 รายงาน ด้วยลักษณะของอัลกอริทึม Clustering รายงานจุดบกพร่องเหล่านั้นจะมีแนวโน้มในการที่จะถูกพิจารณาเข้าไปอยู่ใน Cluster ที่เป็นกลุ่มใหญ่กว่าเพราะว่าค่า Centroid จะถูกคำนวณใหม่ในแต่ละรอบ ทำให้เป็นไปได้ว่าเมื่อค่า Centroid เปลี่ยน ก็อาจจะกลายเป็นค่าที่สอดคล้องกับรายงานจุดบกพร่องที่ควรจะอยู่ในกลุ่มอื่น ทำให้เกิดการจัดกลุ่มที่ผิดพลาด

(2) ในรายงานจุดบกพร่องที่เป็น Meta-bug หลายๆ ฉบับ มีคำในส่วนของ Summary น้อยมาก จึงทำให้ไม่ครอบคลุมต่อการใช้เป็น “query” เพื่อใช้ในการค้นหารายงาน

จุดบกพร่องที่เป็นส่วนต่อได้ ยกตัวอย่างเช่น Meta-bug หมายเลข 1432589 ที่ให้ข้อมูลในส่วน Summary ไว้เพียง “[META] Preferences”

(3) ในบางกรณี คำที่ปรากฏใน Meta-bug และคำที่ปรากฏในรายงานส่วนต่อนั้น ไม่มีคำใดเกี่ยวข้องหรือสอดคล้องการเลย จึงไม่สามารถจัดกลุ่มรายงานจุดบกพร่องส่วนต่อนั้นเข้ามาอยู่ภายใต้ Meta-bug ของตนเองได้ จากเหตุนี้ทำให้บาง Meta-bug ไม่สามารถจัดกลุ่มรายงานจุดบกพร่องส่วนต่อให้ได้



ตารางที่ 4.6 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนแบบเพิ่มข้อจำกัด

คุณลักษณะ	*K*	*tf*				*tf-idf*				*BM25*				*MATF*			
		R	P	F1	Acc	R	P	F1	Acc	R	P	F1	Acc	R	P	F1	Acc
Unigram	25	0.37	0.54	0.44	0.15	0.28	0.61	0.38	0.20	0.11	0.47	0.18	0.04	0.37	0.71	0.49	0.23
Unigram+bigram		0.20	0.46	0.28	0.15	0.25	0.66	0.36	0.25	0.25	0.48	0.33	0.07	0.36	0.68	0.47	0.18
Unigram+compound words		0.39	0.54	0.46	0.20	0.46	0.85	0.59	0.31	0.27	0.49	0.35	0.07	0.37	0.71	0.49	0.26
Combination		0.20	0.45	0.28	0.14	0.19	0.57	0.29	0.22	0.25	0.48	0.33	0.07	0.25	0.67	0.36	0.16
Unigram	50	0.16	0.58	0.25	0.18	0.19	0.55	0.28	0.16	0.07	0.47	0.12	0.06	0.18	0.55	0.27	0.22
Unigram+bigram		0.16	0.47	0.24	0.15	0.28	0.64	0.39	0.15	0.05	0.47	0.09	0.05	0.18	0.60	0.27	0.18
Unigram+compound words		0.36	0.71	0.47	0.18	0.28	0.84	0.42	0.22	0.08	0.53	0.14	0.09	0.21	0.63	0.31	0.24
Combination		0.21	0.76	0.32	0.16	0.19	0.56	0.28	0.21	0.06	0.48	0.11	0.07	0.18	0.60	0.28	0.20
Unigram	75	0.31	0.73	0.44	0.18	0.19	0.54	0.28	0.18	0.07	0.48	0.12	0.08	0.24	0.46	0.32	0.23
Unigram+bigram		0.21	0.77	0.32	0.16	0.28	0.68	0.40	0.16	0.05	0.47	0.09	0.05	0.22	0.59	0.32	0.19
Unigram+compound words		0.34	0.71	0.46	0.21	0.28	0.69	0.40	0.23	0.06	0.48	0.11	0.08	0.25	0.59	0.36	0.24
Combination		0.21	0.77	0.32	0.18	0.19	0.57	0.28	0.22	0.06	0.67	0.10	0.05	0.23	0.59	0.33	0.21
Unigram	100	0.31	0.76	0.44	0.19	0.19	0.49	0.27	0.18	0.06	0.31	0.10	0.11	0.24	0.55	0.33	0.24
Unigram+bigram		0.07	0.57	0.13	0.18	0.28	0.59	0.38	0.15	0.05	0.33	0.09	0.08	0.24	0.56	0.33	0.19
Unigram+compound words		0.35	0.72	0.47	0.21	0.28	0.66	0.39	0.26	0.06	0.34	0.10	0.13	0.24	0.54	0.34	0.27
Combination		0.21	0.77	0.32	0.21	0.19	0.51	0.28	0.25	0.06	0.47	0.10	0.09	0.25	0.55	0.34	0.19

ตารางที่ 4.7 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเคมีนทรงกลมแบบเพิ่มข้อจำกัด

คุณลักษณะ	K	*tf*				*tf-idf*				*BM25*				*MATF*			
		R	P	F1	Acc	R	P	F1	Acc	R	P	F1	Acc	R	P	F1	Acc
Unigram	25	0.43	0.25	0.32	0.26	0.31	0.30	0.30	0.25	0.34	0.30	0.32	0.25	0.52	0.35	0.42	0.27
Unigram+bigram		0.44	0.28	0.34	0.26	0.48	0.33	0.39	0.29	0.47	0.30	0.37	0.29	0.52	0.32	0.40	0.29
Unigram+compound words		0.46	0.28	0.35	0.30	0.49	0.36	0.42	0.33	0.47	0.43	0.45	0.32	0.51	0.38	0.44	0.30
Combination		0.44	0.31	0.36	0.27	0.33	0.30	0.31	0.29	0.32	0.32	0.32	0.27	0.41	0.35	0.38	0.28
Unigram	50	0.17	0.13	0.15	0.20	0.18	0.25	0.21	0.24	0.18	0.24	0.20	0.22	0.29	0.23	0.26	0.23
Unigram+bigram		0.12	0.16	0.14	0.24	0.18	0.27	0.21	0.27	0.18	0.28	0.22	0.24	0.30	0.22	0.26	0.24
Unigram+compound words		0.30	0.26	0.28	0.28	0.31	0.35	0.33	0.29	0.21	0.34	0.26	0.28	0.32	0.28	0.29	0.28
Combination		0.28	0.24	0.26	0.27	0.32	0.33	0.33	0.27	0.17	0.26	0.20	0.27	0.31	0.24	0.27	0.27
Unigram	75	0.10	0.10	0.10	0.20	0.17	0.27	0.21	0.24	0.17	0.22	0.19	0.24	0.25	0.29	0.27	0.25
Unigram+bigram		0.09	0.10	0.09	0.23	0.19	0.30	0.23	0.27	0.17	0.23	0.19	0.26	0.25	0.27	0.26	0.26
Unigram+compound words		0.29	0.28	0.29	0.27	0.31	0.35	0.33	0.29	0.32	0.34	0.33	0.30	0.26	0.37	0.30	0.28
Combination		0.23	0.13	0.17	0.23	0.31	0.34	0.32	0.27	0.30	0.33	0.31	0.26	0.25	0.25	0.25	0.28
Unigram	100	0.11	0.17	0.13	0.19	0.16	0.21	0.18	0.26	0.17	0.23	0.19	0.25	0.22	0.26	0.24	0.24
Unigram+bigram		0.12	0.21	0.15	0.25	0.17	0.32	0.22	0.29	0.17	0.30	0.22	0.24	0.22	0.37	0.28	0.30
Unigram+compound words		0.28	0.38	0.32	0.25	0.30	0.37	0.33	0.30	0.31	0.36	0.33	0.29	0.30	0.34	0.32	0.33
Combination		0.27	0.30	0.29	0.22	0.28	0.33	0.30	0.27	0.17	0.33	0.23	0.27	0.23	0.39	0.29	0.31

4.4 ผลการทดลองการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเทคนิคการวัดความคล้าย

ในการทดลองการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเทคนิคการวัดความคล้าย ซึ่งในที่นี้ประยุกต์ใช้การวัดความคล้ายคลึงแบบโคไซน์ การวัดความคล้ายคลึงด้วย BM25 และการวัดความคล้ายคลึงด้วย MATF

เบื้องต้นได้ดำเนินการสุ่มข้อมูลรายงานจุดบกพร่องที่เป็น Meta-bug จำนวน 200 รายงาน โดยข้อมูล Meta-bug ชุดนี้ จะมีรายงานจุดบกพร่องที่เป็นส่วนต่อ (Dependent bug reports) ทั้งหมดจำนวน 4,581 รายงาน จากนั้นจะนำชุดข้อมูลทดสอบมาผ่านการเตรียมข้อมูล ซึ่งคุณลักษณะของรายงานจุดบกพร่องที่ใช้ยังคงเป็น Unigram, Unigram+bigram, Unigram+compound words และ Combination

โดยในการหาความคล้ายคลึงแบบโคไซน์ จะศึกษาเชิงเปรียบเทียบการให้น้ำหนักคำ 4 วิธี คือ *tf*, *tf-idf, BM25* และ *MATF* ในขณะที่เทคนิค BM25 และ MATF จะใช้การให้น้ำหนักแบบ *tf* เท่านั้น โดยเป็นไปตามของเทคนิคทั้งสอง ซึ่งได้ผลลัพธ์การทดลองดังนี้

สำหรับการประเมินการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยเทคนิคการวัดความคล้าย จะใช้วิธีการตามงานวิจัย [3, 11, 12] ซึ่งเป็นงานวิจัยเกี่ยวกับรายงานจุดบกพร่องเพื่อการตรวจสอบความซ้ำซ้อนของรายงานจุดบกพร่องที่ถูกรายงานเข้ามาในระบบ BTS โดยเทคนิคหลักที่นิยมใช้ในการประเมินก็คือค่าความระลึก หรือค่า True positive rate (TPR) เพราะค่าดังกล่าวทำให้มองเห็นได้ว่าประสิทธิภาพของการค้นคืนรายงานที่ซ้ำซ้อนนั้นเป็นอย่างไร นอกจากค่า TPR แล้ว นักวิจัยกลุ่มดังกล่าวยังมีค่า TNR และ ค่า F1 ที่นักวิจัยใช้ในการประเมิน ดังนั้นในงานวิจัยส่วนนี้ ก็จะใช้ค่า TPR, TNR และค่า F1 ในการประเมินเช่นเดียวกัน

4.4.1 การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยการวัดความคล้ายทคนิคโคไซน์

การจัดกลุ่มด้วยโคไซน์มีผลการทดลองที่แสดงในดังตารางที่ 4.8 - ตารางที่ 4.11 ดังนี้

ตารางที่ 4.8 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์ที่มีการให้น้ำหนักคำด้วย *tf*

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.65	0.92	0.71	0.92	0.60	0.95	0.68	0.95	0.69	0.92	0.74	0.92	0.62	0.95	0.70	0.95
0.2	0.48	0.98	0.58	0.98	0.36	0.99	0.47	0.99	0.49	0.99	0.59	0.98	0.38	0.99	0.49	0.99
0.3	0.31	0.99	0.41	0.99	0.20	1.00	0.29	0.99	0.32	1.00	0.42	0.99	0.22	1.00	0.30	0.99
0.4	0.19	1.00	0.26	0.99	0.12	1.00	0.18	0.99	0.20	1.00	0.28	0.99	0.13	1.00	0.19	0.99
0.5	0.11	1.00	0.16	0.99	0.06	1.00	0.10	0.99	0.12	1.00	0.18	0.99	0.07	1.00	0.11	0.99
0.6	0.05	1.00	0.09	0.99	0.03	1.00	0.05	0.99	0.06	1.00	0.09	0.99	0.03	1.00	0.05	0.99
0.7	0.03	1.00	0.04	0.99	0.02	1.00	0.03	0.99	0.03	1.00	0.05	0.99	0.02	1.00	0.03	0.99
0.8	0.01	1.00	0.01	0.99	0.00	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.00	1.00	0.01	0.99
0.9	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99

ตารางที่ 4.9 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์ที่มีการให้น้ำหนักคำด้วย *tf-idf*

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.54	0.97	0.63	0.97	0.37	0.99	0.47	0.99	0.56	0.97	0.65	0.97	0.39	0.99	0.49	0.99
0.2	0.35	0.99	0.45	0.99	0.18	1.00	0.25	0.99	0.36	0.99	0.46	0.99	0.19	1.00	0.27	0.99
0.3	0.21	1.00	0.29	0.99	0.10	1.00	0.15	0.99	0.21	1.00	0.29	0.99	0.11	1.00	0.16	0.99
0.4	0.12	1.00	0.18	0.99	0.06	1.00	0.09	0.99	0.13	1.00	0.19	0.99	0.06	1.00	0.09	0.99
0.5	0.07	1.00	0.11	0.99	0.03	1.00	0.05	0.99	0.08	1.00	0.12	0.99	0.03	1.00	0.05	0.99
0.6	0.04	1.00	0.06	0.99	0.02	1.00	0.03	0.99	0.04	1.00	0.07	0.99	0.02	1.00	0.03	0.99
0.7	0.02	1.00	0.03	0.99	0.01	1.00	0.02	0.99	0.02	1.00	0.04	0.99	0.01	1.00	0.02	0.99
0.8	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99
0.9	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99

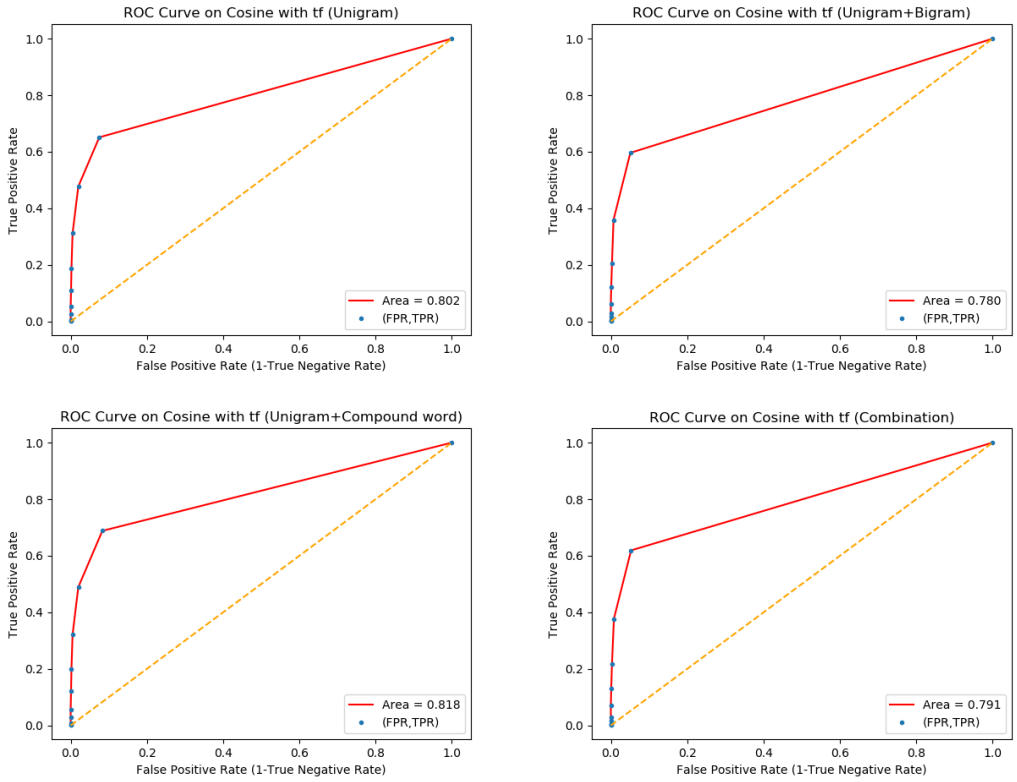
ตารางที่ 4.10 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์ที่มีการให้น้ำหนักคำด้วย *BM25*

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.54	0.97	0.64	0.97	0.38	0.99	0.48	0.99	0.56	0.97	0.65	0.97	0.39	0.99	0.50	0.99
0.2	0.35	0.99	0.46	0.99	0.18	1.00	0.26	0.99	0.36	0.99	0.46	0.99	0.19	1.00	0.27	0.99
0.3	0.21	1.00	0.29	0.99	0.10	1.00	0.15	0.99	0.21	1.00	0.30	0.99	0.11	1.00	0.17	0.99
0.4	0.12	1.00	0.18	0.99	0.06	1.00	0.09	0.99	0.13	1.00	0.19	0.99	0.06	1.00	0.09	0.99
0.5	0.07	1.00	0.11	0.99	0.03	1.00	0.05	0.99	0.08	1.00	0.12	0.99	0.03	1.00	0.05	0.99
0.6	0.04	1.00	0.06	0.99	0.02	1.00	0.03	0.99	0.04	1.00	0.06	0.99	0.02	1.00	0.03	0.99
0.7	0.02	1.00	0.03	0.99	0.01	1.00	0.02	0.99	0.02	1.00	0.04	0.99	0.01	1.00	0.02	0.99
0.8	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99
0.9	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99

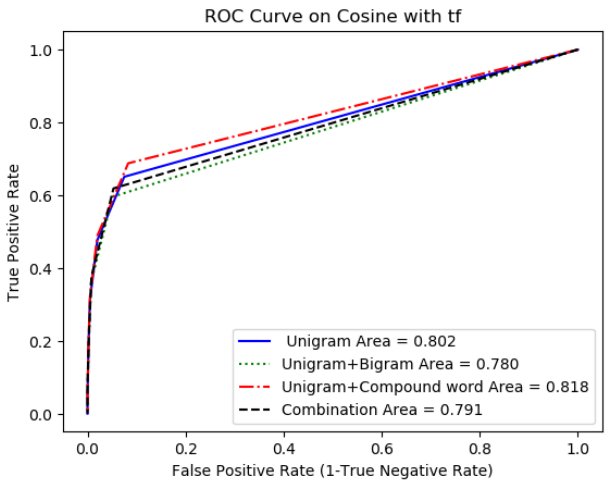
ตารางที่ 4.11 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์ที่มีการให้น้ำหนักคำด้วย *MATF*

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.54	0.97	0.64	0.97	0.38	0.99	0.48	0.99	0.56	0.97	0.65	0.97	0.39	0.99	0.49	0.99
0.2	0.36	0.99	0.46	0.99	0.18	1.00	0.26	0.99	0.36	0.99	0.47	0.99	0.19	1.00	0.27	0.99
0.3	0.21	1.00	0.30	0.99	0.10	1.00	0.15	0.99	0.22	1.00	0.30	0.99	0.11	1.00	0.16	0.99
0.4	0.12	1.00	0.18	0.99	0.06	1.00	0.09	0.99	0.13	1.00	0.19	0.99	0.06	1.00	0.09	0.99
0.5	0.07	1.00	0.11	0.99	0.03	1.00	0.05	0.99	0.08	1.00	0.11	0.99	0.03	1.00	0.05	0.99
0.6	0.04	1.00	0.06	0.99	0.02	1.00	0.03	0.99	0.04	1.00	0.07	0.99	0.02	1.00	0.03	0.99
0.7	0.02	1.00	0.03	0.99	0.01	1.00	0.02	0.99	0.02	1.00	0.04	0.99	0.01	1.00	0.02	0.99
0.8	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99	0.01	1.00	0.01	0.99
0.9	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99

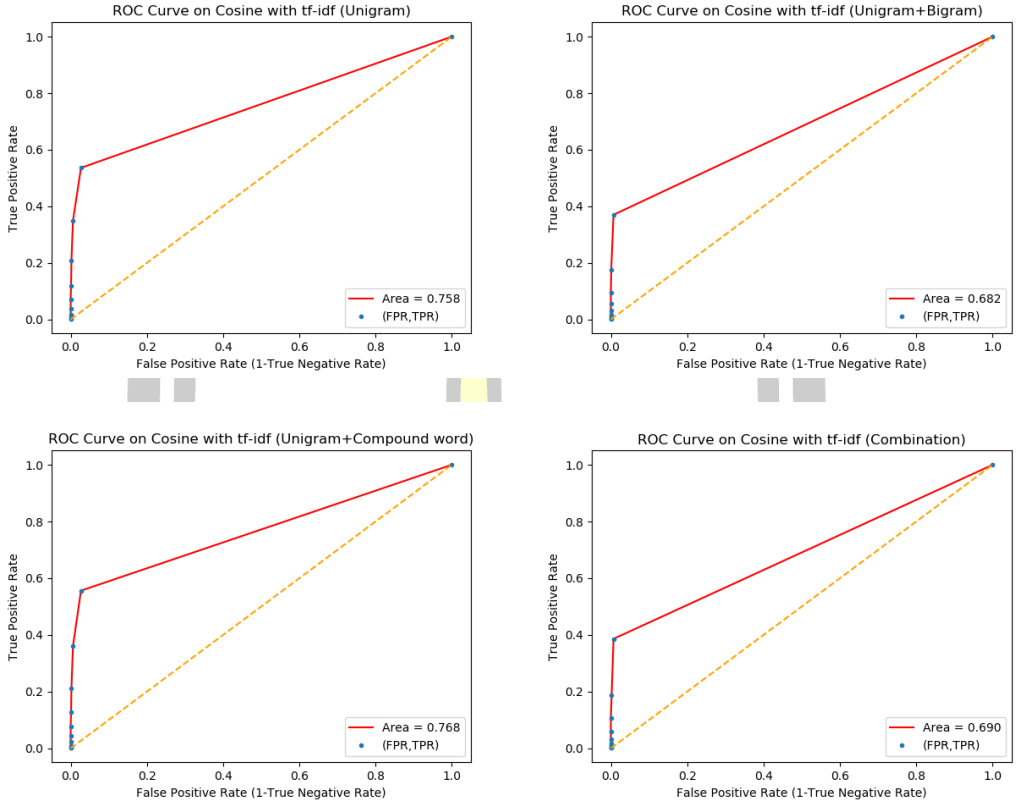
เมื่อได้ผลลัพธ์ดังที่แสดงในตารางที่ 4.8 จนถึงตารางที่ 4.11 แล้ว ค่า TPR ที่ได้ในแต่ละตารางไปพล็อตเป็นกราฟเส้นโค้ง หรือ ROC โดยในการพล็อตกราฟนอกจากจะใช้ค่า TPR แล้ว ยังต้องใช้ค่า FPR (ซึ่งหาค่านี้มาจาก 1-TNR) ในการสร้างเส้นโค้ง ROC ด้วย (ดังแสดงดังรูปที่ 4.1 จนถึงรูปที่ 4.8)



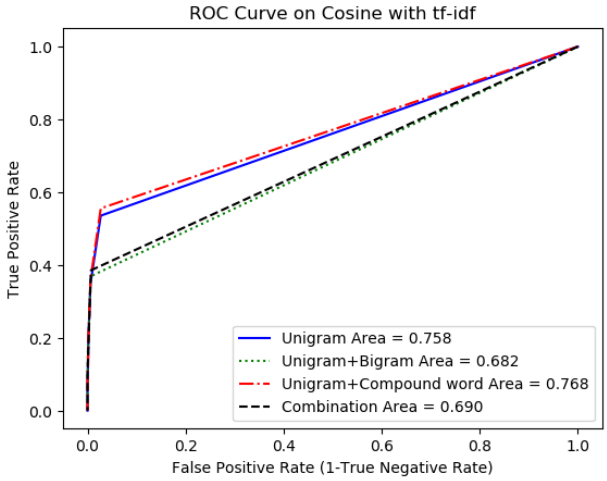
รูปที่ 4.1 การหาเทรสโฮสด์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิคโคไซน์และให้น้ำหนักคำแบบ *tf*



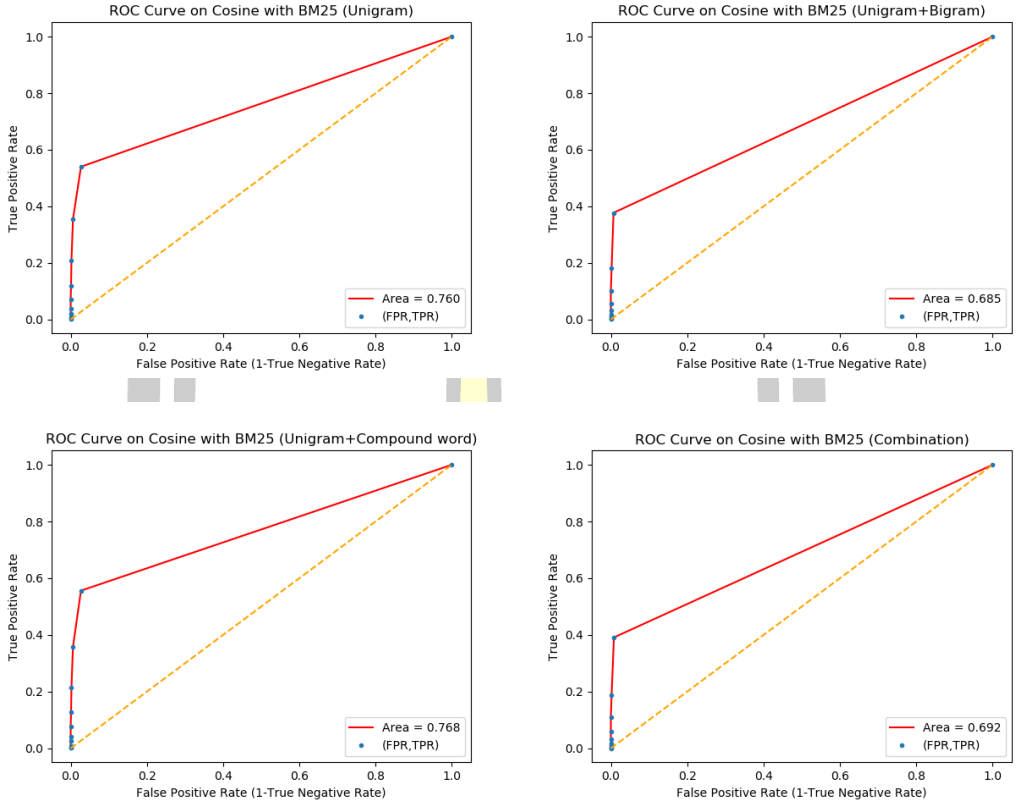
รูปที่ 4.2 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิคโคไซน์และให้น้ำหนักคำแบบ *tf*



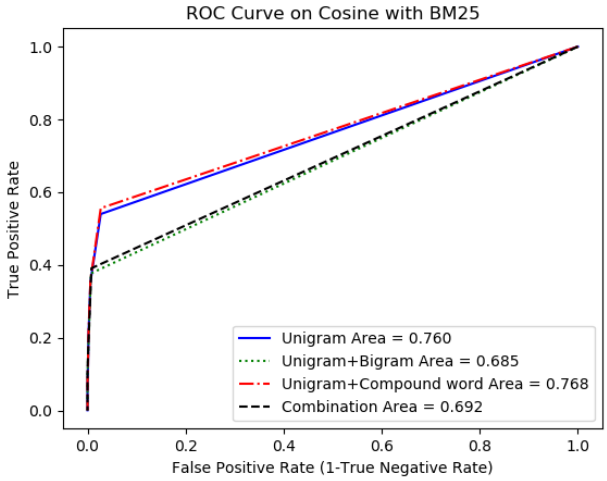
รูปที่ 4.3 การหาเทรสโฮสด์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิคโคไซน์และให้น้ำหนักคำแบบ *tf-idf*



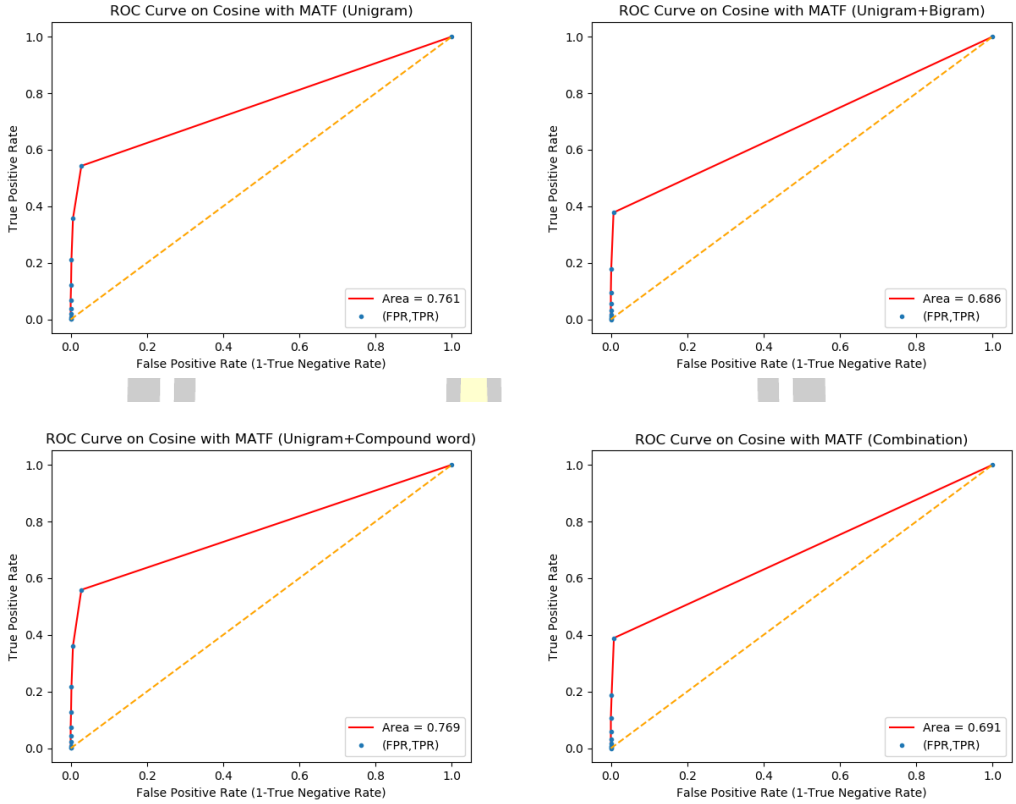
รูปที่ 4.4 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิคโคไซน์และให้น้ำหนักคำแบบ *tf-idf*



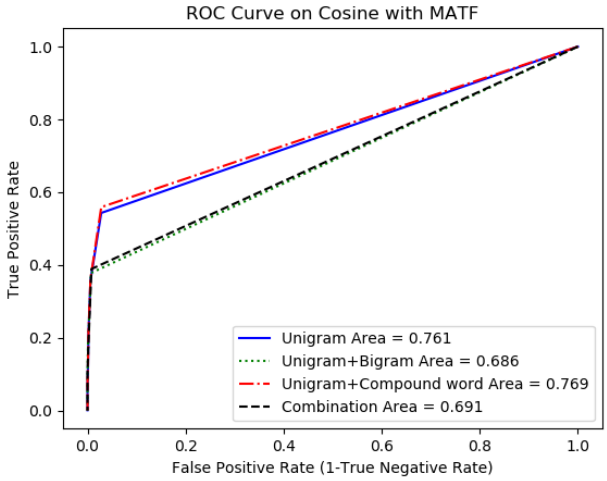
รูปที่ 4.5 การหาเทรสโฮสต์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิคโคไซน์และให้น้ำหนักคำแบบ *BM25*



รูปที่ 4.6 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิคโคไซน์และให้น้ำหนักคำแบบ *BM25*



รูปที่ 4.7 การหาเทรสโฮสต์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิคโคไซน์และให้น้ำหนักคำแบบ *MATF*



รูปที่ 4.8 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิคโคไซน์และให้น้ำหนักคำแบบ *MATF*

โดยเส้นโค้ง ROC จะใช้เพื่อพิจารณาหาตำแหน่งของค่า Threshold ที่ทำให้ได้ผลลัพธ์ที่ดีที่สุดในการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์ โดยจากการพล็อตกราฟดังกล่าวทำให้มองเห็นว่า ค่า Threshold ที่มีค่าเท่ากับ 0.1 เป็นค่า Threshold ที่ให้ผลลัพธ์ที่ดีที่สุด ไม่ว่าจะใช้ร่วมกับการให้น้ำหนักคำแบบใดก็ตาม นอกจากนี้เมื่อได้เส้นโค้ง ROC ได้ทำการคำนวณค่าพื้นที่ใต้เส้นโค้งที่เรียกว่า AUC ซึ่งหากการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์รูปแบบใดมีพื้นที่ใต้โค้งที่มากที่สุด ก็แสดงว่าให้ประสิทธิภาพในการจัดกลุ่มได้ดี

ในกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.1 และรูปที่ 4.2) จะเห็นว่า เมื่อมีการใช้ Unigram+compound words ยังคงเป็นคุณลักษณะร่วมกับการให้น้ำหนักแบบ *tf* จะให้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุด โดยมีค่าเท่ากับ 0.818 ในขณะที่การใช้คุณลักษณะที่เป็น Unigram, Unigram+bigram และ Combination ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.802, 0.780 และ 0.791 ตามลำดับ

ในกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.3 และรูปที่ 4.4) จะเห็นว่า เมื่อมีการใช้ Unigram+compound words ยังคงเป็นคุณลักษณะร่วมกับการให้น้ำหนักแบบ *tf-idf* จะให้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุด โดยมีค่าเท่ากับ 0.768 ในขณะที่การใช้คุณลักษณะที่เป็น Unigram, Unigram+bigram และ Combination ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.758, 0.682 และ 0.690 ตามลำดับ

ในกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.5 และรูปที่ 4.6) จะเห็นว่า เมื่อมีการใช้ Unigram+compound words ยังคงเป็นคุณลักษณะร่วมกับการให้น้ำหนักแบบ BM25 จะให้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุด โดยมีค่าเท่ากับ 0.768 ในขณะที่การใช้คุณลักษณะที่เป็น Unigram, Unigram+bigram และ Combination ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.760, 0.685 และ 0.692 ตามลำดับ

และสุดท้าย ในกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.7และรูปที่ 4.8) จะเห็นว่า เมื่อมีการใช้ Unigram+compound words ยังคงเป็นคุณลักษณะ ร่วมกับการให้น้ำหนักแบบ MATF จะให้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุด โดยมีค่าเท่ากับ 0.769 ในขณะที่การใช้คุณลักษณะที่เป็น Unigram, Unigram+bigram และ Combination ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.761, 0.686 และ 0.691 ตามลำดับ

ดังนั้น หากพิจารณาจากพื้นที่ใต้โค้ง AUC จะเห็นว่าการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วยการวัดความคล้ายแบบโคไซน์นั้น รูปแบบที่ดีที่สุดในการจัดกลุ่มด้วยวิธีการนี้คือการใช้ Unigram+compound words เป็นคุณลักษณะรายงานจุดบกพร่อง ร่วมกับการให้น้ำหนักแบบ *tf*

อย่างไรก็ตาม ผลลัพธ์ที่แสดงใน ตารางที่ 4.8 จนถึงตารางที่ 4.11 มีผลลัพธ์ที่สูงกว่าการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยอัลกอริทึมในตระกูลคลัสเตอริง ซึ่งได้แก่ เคมีนคลัสเตอริง และเคมีนแบบทรงกลม

นอกจากนี้ หากพิจารณาจากค่า TPR เมื่อนำไปเปรียบเทียบกับผลของงานวิจัยที่เกี่ยวข้องกับรายงานจุดบกพร่อง อาทิเช่น [3, 11, 12] ปรากฏว่าค่า TPR ในงานวิจัยนี้มีทิศทางที่

ดีกว่า โดยเฉพาะเมื่อมีการใช้คุณลักษณะแบบ Unigram + compound words และข้อแตกต่างอีกประการหนึ่งที่น่าจะเป็นปัจจัยสำคัญในการทำให้ผลลัพธ์ในงานวิจัยนี้มีแนวโน้มที่ค่า TPR จะสูงกว่างานวิจัยก่อนหน้า ก็คือเพิ่มค่า Threshold เข้ามาช่วยในการประเมินความคล้าย

4.4.2 การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยการวัดความคล้ายเทคนิค BM25

สำหรับการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยการวัดความคล้ายเทคนิค BM25 ซึ่งในการทดลองนี้ได้กำหนดค่า k_1 = 2.0 และ b = 0.8 โดยผลการทดลองการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่ 200 คลัสเตอร์สามารถแสดงในดังตารางที่ 4.12

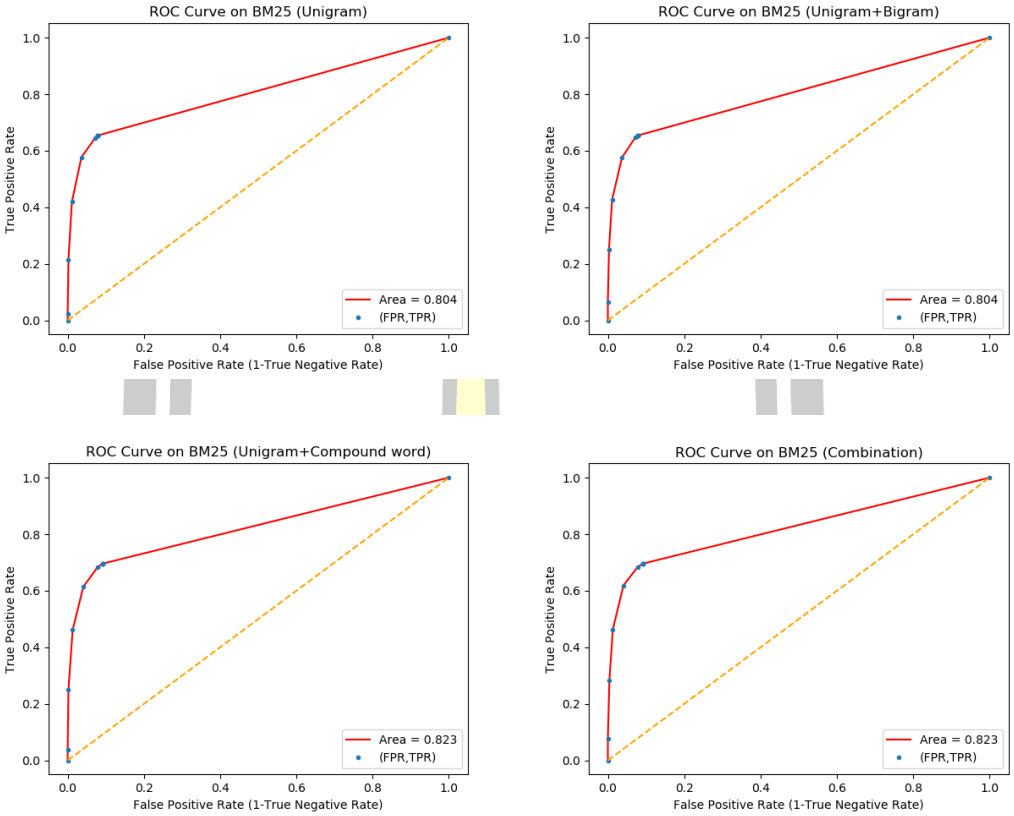
จากผลลัพธ์ดังที่แสดงในตารางที่ 4.12 ได้ทำการคำนวณหาค่า FPR จากค่า 1 - TNR จากนั้นนำค่า TPR และ FPR ที่ได้ไปพล็อตเป็นกราฟเส้นโค้ง ROC (ดังแสดงในรูปที่ 4.9) เพื่อใช้ประกอบการพิจารณาค่า Threshold ที่ดีที่สุดของของการใช้ BM25 ซึ่งจากรูปที่ 4.9 สรุปว่าได้ว่าค่า Threshold ที่ดีที่สุดของของการใช้ BM25 นั้นคือค่า 0.5

ภายหลังจากที่ได้เส้นโค้ง ROC จะทำการคำนวณหาค่าพื้นที่ใต้เส้นโค้งหรือค่า AUC ซึ่งกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.9 และรูปที่ 4.10) จะเห็นว่า เมื่อมีการใช้คุณลักษณะของรายงานจุดบกพร่องแบบ Unigram + compound words หรือ Combination ได้ให้พื้นที่ใต้โค้ง AUC มีค่าเท่ากันคือ 0.823 ซึ่งเป็นค่ามากที่สุด เมื่อเทียบกับพื้นที่ใต้เส้นโค้ง AUC ที่มีการใช้คุณลักษณะของรายงานจุดบกพร่องแบบ แบบ Unigram หรือ Unigram + bigram ซึ่งทั้งสองคุณลักษณะจะให้ค่าพื้นที่ใต้โค้งเท่ากัน คือมีค่าเท่ากับ 0.804

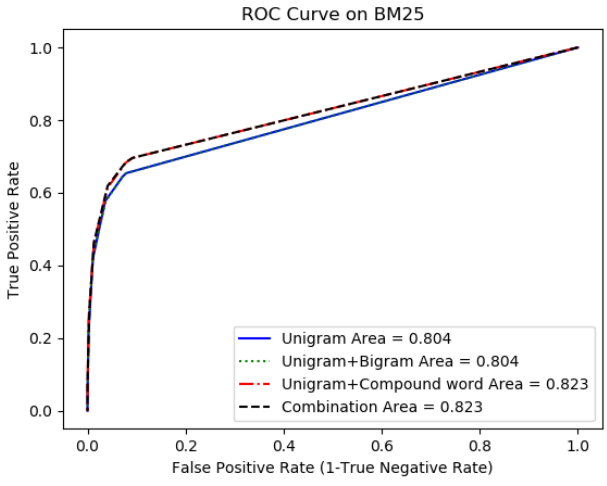
อย่างไรก็ตาม หากพิจารณาจากค่า TPR เมื่อนำไปเปรียบเทียบกับผลของงานวิจัยที่เกี่ยวข้องกับรายงานจุดบกพร่อง อาทิเช่น [3, 11, 12] ซึ่งจะมีค่า TPR อยู่ระหว่าง 0.08 – 0.27 ในขณะที่ค่า TPR ในงานวิจัยนี้ ไม่ว่าเป็นการใช้คุณลักษณะแบบใดก็ตาม ล้วนมีค่าที่ดีกว่างานวิจัยดังกล่าว เพราะให้ค่า TPR ค่าอยู่ระหว่าง 0.65 – 0.68 สาเหตุอาจจะเนื่องมากจากการใช้เทคนิคการวัดความคล้ายที่ต่างกัน โดยเฉพาะมีการกำหนดค่า Threshold เข้ามาช่วยในการประเมินความคล้าย

ตารางที่ 4.12 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วย BM25

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.2	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.3	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.4	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.5	0.65	0.93	0.71	0.93	0.65	0.93	0.71	0.93	0.68	0.92	0.74	0.92	0.68	0.92	0.74	0.92
0.6	0.58	0.96	0.67	0.96	0.58	0.96	0.67	0.96	0.61	0.96	0.70	0.96	0.62	0.96	0.70	0.96
0.7	0.42	0.99	0.52	0.99	0.43	0.99	0.53	0.99	0.46	0.99	0.56	0.98	0.46	0.99	0.57	0.98
0.8	0.21	1.00	0.29	0.99	0.25	1.00	0.34	0.99	0.25	1.00	0.33	0.99	0.28	1.00	0.37	0.99
0.9	0.02	1.00	0.04	0.99	0.07	1.00	0.10	0.99	0.04	1.00	0.06	0.99	0.08	1.00	0.12	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99



รูปที่ 4.9 การหาเทรสโฮสด์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิค BM25



รูปที่ 4.10 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิค BM25

4.4.3 การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยการวัดความคล้ายเทคนิค MATF

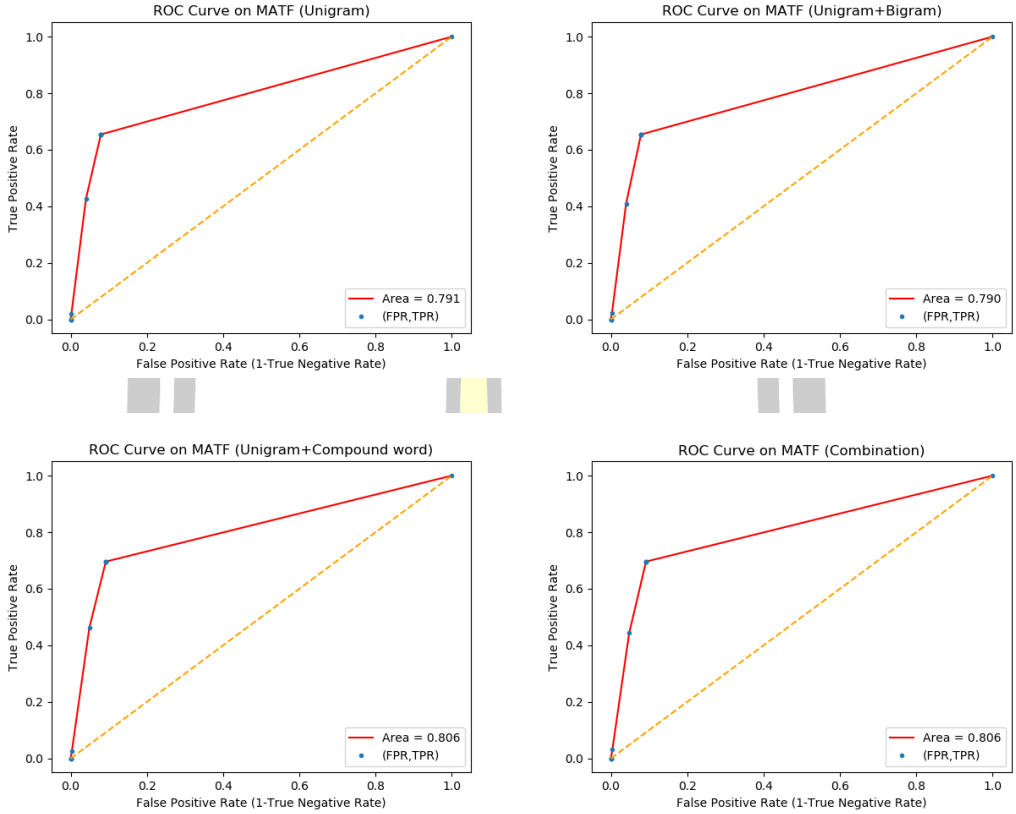
สำหรับการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยการวัดความคล้ายเทคนิค MATF มีผลการทดลองที่แสดงในดัง ตารางที่ 4.13 และจากผลลัพธ์ในตารางที่ 4.13 ได้ทำการคำนวณหาค่า FPR จากค่า 1 - TNR จากนั้นนำค่า TPR และ FPR ที่ได้ไปพล็อตเป็นกราฟเส้นโค้ง ROC (ดังแสดงในรูปที่ 4.11) เพื่อใช้ประกอบการพิจารณาค่า Threshold ที่ดีที่สุดของของการใช้ BM25 ซึ่งจากรูปที่ 4.11 สรุปว่าได้ว่าค่า Threshold ที่ดีที่สุดของของการใช้ MATF นั้นคือค่า 0.4

จากนั้นเมื่อได้เส้นโค้ง ROC ได้ทำการคำนวณค่าพื้นที่ใต้เส้นโค้ง ซึ่งกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.11 และรูปที่ 4.12) จะเห็นว่า เมื่อมีการใช้ Unigram + compound words และ Combination ได้ให้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุดเท่ากัน โดยมีค่าเท่ากับ 0.806 ในขณะที่การใช้คุณลักษณะที่เป็น Unigram และ Unigram+bigram ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.791 และ 0.790 เท่ากัน

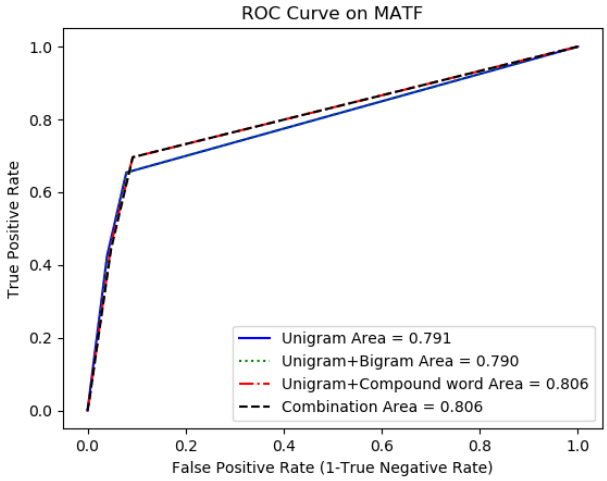
แน่นอนว่าเมื่อพิจารณาจากค่า TPR และเปรียบเทียบกับผลของงานวิจัยใน [3, 11, 12] ซึ่งจะมีค่า TPR อยู่ระหว่าง 0.08 – 0.27 ก็พบว่าผลการวิจัยที่ได้นำเสนอในส่วนนี้ก็มีค่า TPR สูงกว่างานวิจัยก่อนหน้า โดยมีค่า TPR ระหว่าง 0.65-0.70 และสาเหตุน่าจะเป็นเพราะการใช้เทคนิคการวัดความคล้ายที่ต่างกัน โดยเฉพาะมีการกำหนดค่า Threshold เข้ามาช่วยในการประเมินความคล้ายนั่นเอง

ตารางที่ 4.13 ผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ด้วย MATF

Threshold	Unigram				Unigam+bigram				Unigram+compound words				Combination			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.2	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.3	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.4	0.65	0.92	0.71	0.92	0.65	0.92	0.71	0.92	0.70	0.91	0.74	0.91	0.70	0.91	0.74	0.91
0.5	0.43	0.96	0.54	0.96	0.41	0.96	0.53	0.96	0.46	0.95	0.57	0.95	0.44	0.95	0.55	0.95
0.6	0.02	1.00	0.04	0.99	0.02	1.00	0.04	0.99	0.02	1.00	0.04	0.99	0.03	1.00	0.06	0.99
0.7	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
0.8	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
0.9	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99



รูปที่ 4.11 การหาเทรสโฮสด์ที่ดีที่สุดด้วยเส้นโค้ง ROC เทคนิค MATF



รูปที่ 4.12 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของเทคนิค MATF

4.5 ผลการทดลองการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อด้วยฟังก์ชันการค้นคืน REP

จากการทดลองการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืน REP นั้น ในขั้นตอนแรกจะต้องดำเนินการสร้างชุดข้อมูลสอน สำหรับการปรับค่าพารามิเตอร์ในฟังก์ชันการค้นคืน REP ก่อน ซึ่งได้นำเสนอวิธีการไว้ในรูปที่ 3.24 ในหัวข้อที่ 3.8 ซึ่งผลการสร้างได้ชุดข้อมูลสอนที่อยู่ในรูปแบบ (*q*, *rel*, *irr*) จำนวน 8,070 ชุด และต่อไปนี้คือ ค่าพารามิเตอร์เริ่มต้นและ ผลของการปรับค่าพารามิเตอร์ในฟังก์ชันการค้นคืน REP ในแต่ละรูปแบบที่แสดงได้ดังตารางที่ 4.14 จนถึงตารางที่ 4.16

ตารางที่ 4.14 พารามิเตอร์ที่ได้รับการปรับค่าในฟังก์ชันการค้นคืนด้วยเทคนิค Cosine similarity

พารามิเตอร์	รายละเอียด	ค่าเริ่มต้น	ค่าที่ปรับ
w_1	ค่าน้ำหนักของ *feature*$_1$ (cosine(*d*, *q*)) ในข้อมูล Summary ที่สกัดคุณลักษณะของคำแบบ Unigram + compound words	0.9	2.809
w_2	ค่าน้ำหนักของ *feature*$_2$ (cosine(*d*, *q*)) ในข้อมูล Description ที่สกัดคุณลักษณะของคำแบบ Unigram + compound words	0.3	1.156
w_3	ค่าน้ำหนักของ *feature*$_3$ (Product)	0	0.775
w_4	ค่าน้ำหนักของ *feature*$_4$ (Component)	0	1.498
w_5	ค่าน้ำหนักของ *feature*$_5$ (Priority)	0	0.009
w_6	ค่าน้ำหนักของ *feature*$_6$ (Severity)	0	0.027

ตารางที่ 4.15 พารามิเตอร์ที่ได้รับการปรับค่าในฟังก์ชันการค้นคืนด้วยเทคนิค MATF

พารามิเตอร์	รายละเอียด	ค่าเริ่มต้น	ค่าที่ปรับ
w_1	ค่าน้ำหนักของ *feature*$_1$ (MATF(*d*, *q*)) ในข้อมูล Summary ที่สกัดคุณลักษณะของคำแบบ Unigram + compound words	0.9	2.398
w_2	ค่าน้ำหนักของ *feature*$_3$ (MATF(*d*, *q*)) ในข้อมูล Description ที่สกัดคุณลักษณะของคำแบบ Unigram + compound words	0.3	0.548
w_3	ค่าน้ำหนักของ *feature*$_3$ (Product)	0	0.703
w_4	ค่าน้ำหนักของ *feature*$_4$ (Component)	0	1.381
w_5	ค่าน้ำหนักของ *feature*$_5$ (Priority)	0	0.056
w_6	ค่าน้ำหนักของ *feature*$_6$ (Severity)	0	0.055

ตารางที่ 4.16 พารามิเตอร์ที่ได้รับการปรับค่าฟังก์ชันการค้นคืนด้วยเทคนิค BM25F

พารามิเตอร์	รายละเอียด	ค่าเริ่มต้น	ค่าที่ปรับ
w_1	ค่าน้ำหนักของ *feature*$_1$ (BM25F(d, q)) ในข้อมูล Summary และ Description คุณลักษณะของคำแบบ Unigram + compound words	0.9	0.740
w_2	ค่าน้ำหนักของ *feature*$_2$ (Product)	0	0.788
w_3	ค่าน้ำหนักของ *feature*$_3$ (Component)	0	1.400
w_4	ค่าน้ำหนักของ *feature*$_4$ (Priority)	0	-0.097
w_5	ค่าน้ำหนักของ *feature*$_5$ (Severity)	0	0.034

ในการทดลองนี้ได้อ้างอิงการตั้งค่าเริ่มต้นของพารามิเตอร์ตามงานวิจัยของ Sun และคณะ [73] ซึ่งได้ให้ข้อคิดเห็นว่าการให้ความสำคัญของคุณลักษณะข้อความในส่วนของ Summary ควรมีค่าเป็นสามเท่าของ Description

หลังจากได้พารามิเตอร์ที่ทำการปรับค่าเรียบร้อยแล้ว จึงนำไปใช้ตามฟังก์ชัน REP ที่ร่วมกับเทคนิค Cosine similarity, BM25F และ MATF ซึ่งได้ผลการทดลองที่แสดงดังตารางที่ 4.17

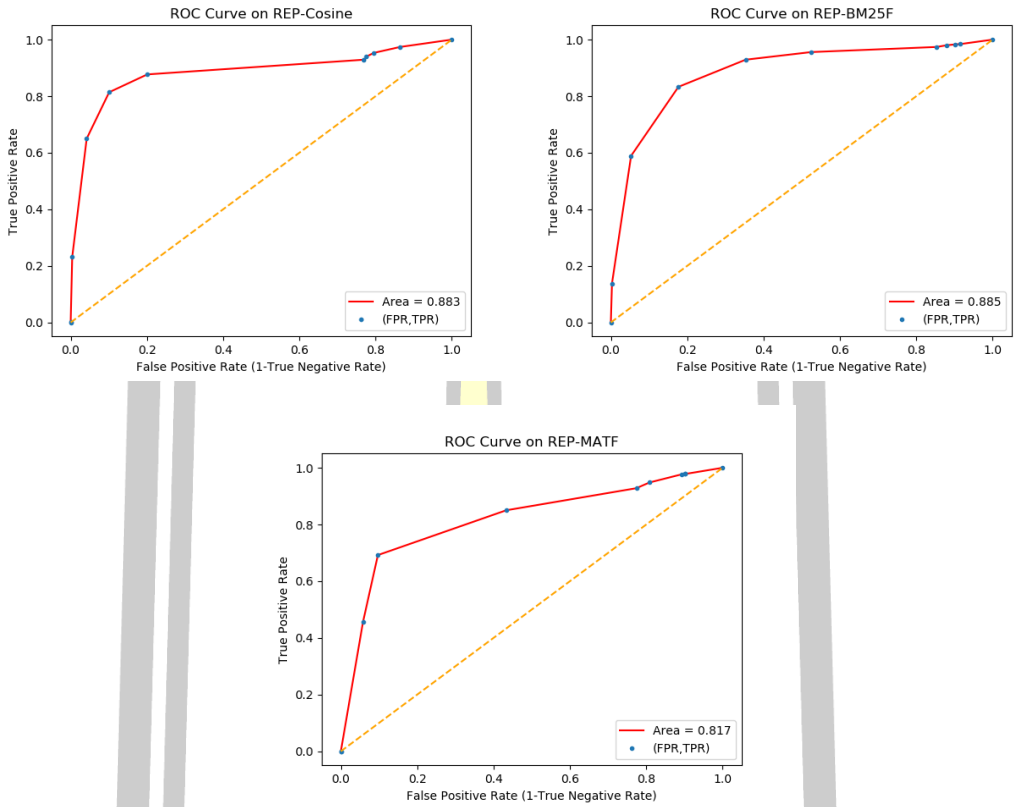
จากผลลัพธ์ในตารางที่ 4.17 ได้ทำการคำนวณหาค่า FPR จากค่า 1 - TNR จากนั้นนำค่า TPR และ FPR ที่ได้ไปพล็อตเป็นกราฟเส้นโค้ง ROC (ดังแสดงในรูปที่ 4.13) เพื่อใช้ประกอบการพิจารณาค่า Threshold ที่ดีที่สุดของของการใช้ REP ที่ร่วมกับ Cosine similarity (REP-Cosine), BM25F (REP-BM25F) และ MATF (REP-MATF) ซึ่งจากรูปที่ 4.13 สรุปว่าได้ว่าค่า Threshold ที่ดีที่สุดของ REP-Cosine, REP-BM25F และ REP-MATF มีค่าเท่ากับ 0.6 0.7 และ 0.7 ตามลำดับ

จากนั้นเมื่อได้เส้นโค้ง ROC ได้ทำการคำนวณค่าพื้นที่ใต้เส้นโค้ง ซึ่งกรณีที่เป็นการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อใน 200 คลัสเตอร์ (แสดงในรูปที่ 4.13 และรูปที่ 4.14) จะเห็นว่าเป็น REP-BM25F จะได้พื้นที่ใต้โค้ง AUC มีค่ามากที่สุด โดยมีค่าเท่ากับ 0.885 ในขณะที่ REP-Cosine และ REP-MATF ให้ค่าพื้นที่ใต้โค้งอยู่ที่ 0.883 และ 0.817 ตามลำดับ

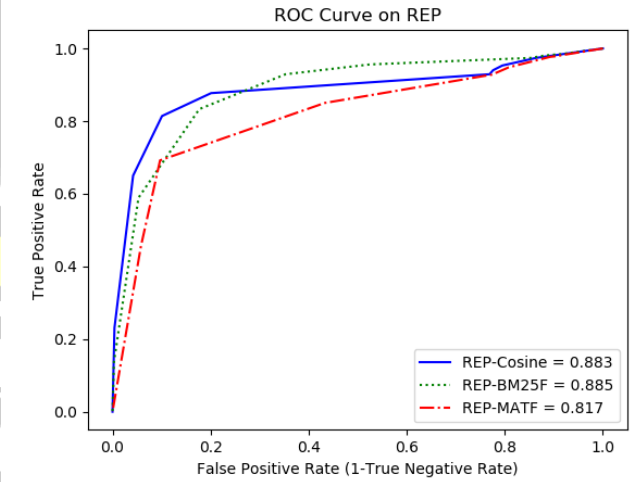
หากพิจารณาจากค่า TPR เมื่อนำไปเปรียบเทียบกับผลของงานวิจัยเช่น [3, 11, 12] ซึ่งจะมีค่า TPR อยู่ระหว่าง 0.08 – 0.27 พบว่าค่า TPR ในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืนมีค่าที่สูงกว่าค่า TPR ในงานวิจัยก่อนหน้า โดยให้ค่า TPR ค่าอยู่ระหว่าง 0.69 – 0.83 สาเหตุอาจจะเนื่องมาฟังก์ชันการค้นคืนจะมีการใช้คุณลักษณะอื่นๆ ของซอฟต์แวร์เข้ามาร่วมในการพิจารณาความคล้าย นั่นคือไม่ได้ใช้คุณลักษณะที่เป็น “คำ” จาก Summary หรือ Description เพียงอย่างเดียว รวมทั้งยังมีการใช้ค่า Threshold เข้ามาช่วยในการประเมินความคล้าย

ตารางที่ 4.17 ผลลัพธ์การจัดกลุ่มด้วยฟังก์ชันการค้นคืน

Threshold	REP-Cosine				REP-BM25F				REP-MATF			
	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc	TPR	TNR	F1	Acc
0.1	0.97	0.14	0.23	0.14	0.98	0.09	0.15	0.09	0.98	0.10	0.17	0.10
0.2	0.95	0.20	0.34	0.21	0.98	0.10	0.17	0.10	0.98	0.10	0.17	0.10
0.3	0.94	0.22	0.36	0.23	0.98	0.12	0.21	0.13	0.98	0.11	0.19	0.11
0.4	0.93	0.23	0.37	0.23	0.97	0.15	0.25	0.15	0.95	0.19	0.32	0.20
0.5	0.88	0.80	0.82	0.80	0.96	0.47	0.59	0.48	0.93	0.23	0.36	0.23
0.6	0.81	0.90	0.83	0.90	0.93	0.65	0.72	0.65	0.85	0.57	0.63	0.57
0.7	0.65	0.96	0.73	0.96	0.83	0.82	0.78	0.82	0.69	0.90	0.74	0.90
0.8	0.23	1.00	0.32	0.99	0.59	0.95	0.65	0.94	0.46	0.94	0.54	0.94
0.9	0.00	1.00	0.00	0.99	0.14	1.00	0.18	0.99	0.00	1.00	0.00	0.99
1.0	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99	0.00	1.00	0.00	0.99



รูปที่ 4.13 การหาเทรสโฮสต์ที่ดีที่สุดด้วยเส้นโค้ง ROC ของฟังก์ชัน REP



รูปที่ 4.14 เปรียบเทียบเส้นโค้ง ROC และพื้นที่ใต้เส้นโค้งของฟังก์ชัน REP

ดังนั้นสามารถสรุปได้ว่าฟังก์ชันการค้นคืน REP ร่วมกับ Cosine similarity หรือร่วมกับ BM25F ให้ผลเป็นที่น่าพอใจสูงสุด แต่มีข้อสังเกตเรื่องของเวลาในการประมวลผลของ ฟังก์ชันการค้นคืน REP ร่วมกับ BM25F ใช้เวลาในการปรับค่าพารามิเตอร์ที่มาก นอกจากนี้เวลาที่ใช้ในการ

ประมวลผลเพื่อทำการจัดกลุ่มก็ยังใช้เวลาที่มากอีกด้วย ซึ่งจะแสดงผลเวลาในการประมวลผลของทุกเทคนิคที่ใช้ในการศึกษาในหัวข้อที่ 4.6 ในลำดับถัดไป

4.6 สรุปผลการจัดกลุ่มในภาพรวม

จากการทดลองเพื่อรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อนั้น ในการทดลองได้ใช้รายงานจุดบกพร่องจำนวน 4,781 รายงาน และมีจำนวนรายงานจุดบกพร่องที่เป็น Meta-bug จำนวน 200 รายงาน ซึ่งสามารถสรุปผลการทดลองในทุกวิธีการที่นำเสนอโดยนำผลการทดลองที่ให้ผลเป็นที่น่าพอใจสูงสุดในแต่ละวิธีการมาเปรียบเทียบกันดังตารางที่ 4.18 ยกเว้นการจัดกลุ่มด้วยเคมีนแบบเพิ่มข้อจำกัด และเคมีนทรงกลมแบบเพิ่มข้อจำกัด ที่ได้ผลลัพธ์ที่มีแนวโน้มที่ต่ำมากจึงสิ้นสุดการทดลองเมื่อทดสอบกับการจัดกลุ่มที่มี Meta-bug 100 รายงาน

ตารางที่ 4.18 เปรียบเทียบผลลัพธ์การจัดกลุ่มที่เป็นส่วนต่อจากทุกวิธีการที่นำเสนอ

Method		Threshold	AUC	TPR	TNR	F1	เวลา (นาที)
Cosine with *tf*	Unigram	0.1	0.802	0.65	0.92	0.71	15
	Unigram + bigram	0.1	0.780	0.60	0.95	0.68	23
	Unigram + compound words	0.1	0.818	0.69	0.92	0.74	16
	Combination	0.1	0.791	0.62	0.95	0.70	26
Cosine with *tf-idf*	Unigram	0.1	0.758	0.54	0.97	0.63	16
	Unigram + bigram	0.1	0.682	0.37	0.99	0.47	25
	Unigram + compound words	0.1	0.768	0.56	0.97	0.65	16
	Combination	0.1	0.690	0.39	0.99	0.49	28
Cosine with *BM25*	Unigram	0.1	0.760	0.54	0.97	0.64	16
	Unigram + bigram	0.1	0.685	0.38	0.99	0.48	25
	Unigram + compound words	0.1	0.768	0.56	0.97	0.65	16
	Combination	0.1	0.692	0.39	0.99	0.50	31
Cosine with *MATF*	Unigram	0.1	0.761	0.54	0.97	0.64	17
	Unigram + bigram	0.1	0.686	0.38	0.99	0.65	26
	Unigram + compound words	0.1	0.769	0.56	0.97	0.65	17
	Combination	0.1	0.691	0.39	0.99	0.49	30
BM25	Unigram	0.5	0.804	0.65	0.93	0.71	7
	Unigram + bigram	0.5	0.804	0.65	0.93	0.71	9
	Unigram + compound words	0.5	0.823	0.68	0.92	0.74	9
	Combination	0.5	0.823	0.68	0.92	0.74	10
MATF	Unigram	0.4	0.791	0.65	0.92	0.71	10
	Unigram + bigram	0.4	0.790	0.65	0.92	0.71	11

Method		Threshold	AUC	TPR	TNR	F1	เวลา (นาที)
	Unigram + compound words	0.4	0.806	0.70	0.91	0.74	11
	Combination	0.4	0.806	0.70	0.91	0.74	12
REP-Cosine		0.6	0.883	0.81	0.90	0.83	78
REP-BM25F		0.7	0.885	0.83	0.82	0.78	5,892
REP-MATF		0.7	0.817	0.69	0.90	0.74	530

จากตารางที่ 4.18 แสดงให้เห็นว่าแนวทางที่นำเสนอเพื่อการรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อนั้น สามารถรวบรวมรายงานจุดบกพร่องซึ่งมีค่า TPR และ TNR เฉลี่ยอยู่ที่ 0.59 และ 0.94 ตามลำดับ ซึ่งหมายถึง สามารถรวบรวมรายงานจุดบกพร่องที่เป็นส่วนต่อของ Meta-bug ได้โดยประมาณ 59% จากรายงานที่เป็นส่วนต่อทั้งหมด และสามารถระบุรายงานที่ไม่เป็นส่วนต่อของแต่ละ Meta-bug ได้ 94% จากรายงานที่ไม่ใช่ส่วนต่อของ Meta-bug ทั้งหมด และหากพิจารณาวิธีการที่เป็นที่น่าพอใจทั้งผลการประเมินและเวลาที่ใช้ในการประมวลผลแล้ว วิธีการหาความคล้ายด้วย BM25 ดูจะเป็นวิธีการที่เหมาะสมทั้งผลการประเมินและระยะเวลาในการประมวลผลด้วย

จากตารางที่ 4.18 จะเห็นว่าบางเทคนิคที่ใช้นั้น จะใช้เวลาเป็นค่อนข้างมากในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ อย่างไรก็ตาม จากงานวิจัยของ Runeson และคณะ [10] ได้กล่าวไว้ว่า โดยทั่วไปแล้วการวิเคราะห์รายงานจุดบกพร่องโดย Bug triager นั้นจะใช้เวลาประมาณ 30 นาทีต่อรายงานจุดบกพร่อง 1 รายงาน เพื่อวิเคราะห์ในหลายๆ ประเด็น เช่น การวิเคราะห์รายงานจุดบกพร่องว่าเป็น bug หรือ non-bug การวิเคราะห์ว่ารายงานจุดบกพร่องนั้นซ้ำซ้อนหรือไม่ การวิเคราะห์ระดับความรุนแรง การวิเคราะห์ระดับความสำคัญของจุดบกพร่อง รวมไปถึงการวิเคราะห์ส่วนต่อของรายงานจุดบกพร่อง

สมมติว่ารายงานจุดบกพร่อง 1 รายงาน Bug triager จะใช้เวลาวิเคราะห์ส่วนต่อของรายงานจุดบกพร่องประมาณ 5 นาที หากเป็นการวิเคราะห์รายงานจุดบกพร่องจำนวน 4,781 รายงาน (เท่ากับจำนวนรายงานจุดบกพร่องที่ใช้ศึกษาในงานวิจัยนี้) จะพบว่า Bug triager จะใช้เวลาถึง 23,905 นาที หรือ 16 วัน 14 ชั่วโมง 25 นาที ดังนั้น หากเปรียบเทียบเวลาในการวิเคราะห์การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อที่นำเสนอในงานวิจัยฉบับนี้ (ตารางที่ 4.20) จะเห็นได้ว่าสามารถลดเวลาในการประมวลผลได้ค่อนข้างมาก และแน่นอนว่าค่าใช้จ่ายในขั้นตอนของการปรับปรุงซอฟต์แวร์ก็น่าจะมีโอกาสลดลง ตามไปด้วย

ตารางที่ 4.19 การเปรียบเทียบเวลาในการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อถ้าหากเป็นการดำเนินการของ Bug triager และ การประมวลผลแบบอัตโนมัติในวิธีการต่างๆ ที่นำเสนอ

Method		Threshold	จำนวนรายงานที่ใช้ (รายงาน)	TPR	จำนวนรายงานที่ประมวลผลจัดกลุ่มได้ถูกต้อง	เวลาประมาณการที่ Bug triager ใช้ในการวิเคราะห์การจัดกลุ่ม (นาที)	เวลาที่ใช้ในการประมวลผลเพื่อวิเคราะห์การจัดกลุ่ม (นาที)
Cosine with *tf*	Unigram	0.1	4,781	0.65	3,108	15,538	15
	Unigram + bigram	0.1	4,781	0.60	2,869	14,343	23
	Unigram + compound words	0.1	4,781	0.69	3,299	16,494	16
	Combination	0.1	4,781	0.62	2,964	14,821	26
Cosine with *tf-idf*	Unigram	0.1	4,781	0.54	2,582	12,909	16
	Unigram + bigram	0.1	4,781	0.37	1,769	8,845	25
	Unigram + compound words	0.1	4,781	0.56	2,677	13,387	16
	Combination	0.1	4,781	0.39	1,865	9,323	28
Cosine with *BM25*	Unigram	0.1	4,781	0.54	2,582	12,909	16
	Unigram + bigram	0.1	4,781	0.38	1,817	9,084	25
	Unigram + compound words	0.1	4,781	0.56	2,677	13,387	16
	Combination	0.1	4,781	0.39	1,865	9,323	31
Cosine with *MATF*	Unigram	0.1	4,781	0.54	2,582	12,909	17
	Unigram + bigram	0.1	4,781	0.38	1,817	9,084	26
	Unigram + compound words	0.1	4,781	0.56	2,677	13,387	17
	Combination	0.1	4,781	0.39	1,865	9,323	30
BM25	Unigram	0.5	4,781	0.65	3,108	15,538	7
	Unigram + bigram	0.5	4,781	0.65	3,108	15,538	9
	Unigram + compound words	0.5	4,781	0.68	3,251	16,255	9
	Combination	0.5	4,781	0.68	3,251	16,255	10

Method		Threshold	จำนวนรายงานที่ใช้ (รายงาน)	TPR	จำนวนรายงานที่ประมวลผลจัดกลุ่มได้ถูกต้อง	เวลาประมาณการที่ Bug triager ใช้ในการวิเคราะห์การจัดกลุ่ม (นาที)	เวลาที่ใช้ในการประมวลผลเพื่อวิเคราะห์การจัดกลุ่ม (นาที)
MATF	Unigram	0.4	4,781	0.65	3,108	15,538	10
	Unigram + igramb	0.4	4,781	0.65	3,108	15,538	11
	Unigram + compound words	0.4	4,781	0.70	3,347	16,734	11
	Combination	0.4	4,781	0.70	3,347	16,734	12
REP-Cosine		0.6	4,781	0.81	3,873	19,363	78
REP-BM25F		0.7	4,781	0.83	3,968	19,841	5,892
REP-MATF		0.7	4,781	0.69	3,299	16,494	530

จากการศึกษาการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อ สามารถแสดงตัวอย่างผลลัพธ์ของการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อจากการงานวิจัยนี้ได้ดัง ตารางที่ 4.20 และ ตารางที่ 4.21

ตารางที่ 4.20 แสดงผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อตัวอย่างที่ 1

Meta-bug 1501751 "[META] Pinned Tabs CFR 67"

ผลการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดย Bug triager

1501815 1501816 1503693 1508744 1509078 1517303 1517306 1528952 1528953
1528955 1528959 1528966 1529078 1529334 1529340 1530359 1530431 1530433
1531734 1532693 1532707 1532777 1533053 1533138 1533359 1533452 1533588
1534254 1534444 1534719 1534742 1534752 1534934 1534990 1534996 1535268
1535344 1535464

ผลการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อโดยอัตโนมัติด้วยเทคนิคความคล้าย BM25

1509078 1529340 1534752 1501816 1528959 1532707 1534990
530433 1533138 1534444 1534719 1535344 1528966 1535268
1529078 387413 1508744 1528955

ตารางที่ 4.21 แสดงผลลัพธ์การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อตัวอย่างที่ 2

Meta-bug 1354500 "Remove the proprietary persistent indexedDB permission"

ผลการจัดกลุ่มรายงานที่เป็นส่วนต่อโดย Bug triager

1334411 1360567 1409054 1428320 1442560 1445318 1445326 1448491 1451486
1451794

ผลการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อโดยอัตโนมัติด้วยเทคนิคความคล้าย BM25

1451794 1334411 1448491 1445326 1193862 1451486 806722 1445318
1459229

จากตารางที่ 4.20 และ ตารางที่ 4.21 เป็นตัวอย่างผลการทดลองการเปรียบเทียบผลลัพธ์ของการจัดกลุ่มรายงานส่วนต่อ ระหว่าง Bug triager และ การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดยอัตโนมัติด้วยเทคนิคความคล้าย BM25 เมื่อใช้ Threshold เท่ากับ 0.5

ในตัวอย่างผลการจัดกลุ่มของ Meta-bug 1501751 มีผลการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดย Bug triager จำนวน 38 รายงาน ในขณะที่การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดยอัตโนมัติด้วยเทคนิคความคล้าย BM25 นั้นมีจำนวน 18 รายงาน แต่มี 1 รายงาน ที่ต่างไปจากการวิเคราะห์โดย Bug triager

ขณะเดียวกันในตัวอย่างการจัดกลุ่มของ Meta-bug 1354500 ได้ผลการจัดกลุ่มรายงานจุดบกพร่องจาก Bug triager จำนวน 10 รายงาน ในขณะที่ที่การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดยอัตโนมัติด้วยเทคนิคความคล้าย BM25 ได้จำนวนรายงานจุดบกพร่องที่เป็นส่วนต่อของ Meta-bug นี้จำนวน 9 รายงาน แต่มี 3 รายงานที่ต่างไปจากการวิเคราะห์โดย Bug tirager

เมื่อพิจารณารายงานที่ต่างไปจากการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดย Bug triager นั้น เป็นเพราะ รายงานเหล่านั้นใช้คำที่มีสอดคล้องกับ Meta-bug และเมื่อคำนวณค่าความคล้ายด้วย

BM25 ได้ค่ามากกว่า Threshold ที่กำหนด จึงถูกจัดเข้ากลุ่มรายงานจุดบกพร่องส่วนต่อของ Meta-bug เหล่านั้น ยกตัวอย่างเช่น รายงานจุดบกพร่องหมายเลข 1459229 มีข้อความ Summary คือ *"Calls to navigator.storage.persist should not spawn permission requests when called from extension pages"* ซึ่งรายงานจุดบกพร่องมี "คำ" ที่สอดคล้องกับ Meta-bug หมายเลข 1354500 ที่มีคำ *"permission"* เช่นเดียวกัน อย่างไรก็ตาม หากทำการปรับค่า Threshold เพิ่มขึ้นเป็น 0.6 รายงานจุดบกพร่องหมายเลข 1459229 ก็จะไม่ถูกจัดเข้ากลุ่มรายงานจุดบกพร่องส่วนต่อแต่อย่างใด นั้นหมายถึงสามารถเพิ่มค่า Threshold เพื่อกรองรายงานจุดบกพร่องที่ไม่ควรเกี่ยวข้องออกไปได้ แต่ก็อาจสูญเสียรายงานจุดบกพร่องส่วนต่อแท้จริงออกไปด้วยก็เป็นได้

ส่วนในกรณีการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อ ได้ผลลัพธ์ที่ไม่ดีในบางกลุ่มของ Meta-bug นั้นเนื่องมากจาก คำที่ใช้ในส่วน Summary ของ Meta-bug นั้นน้อยเกินไป หรือ ใช้คำที่ไม่สอดคล้องกันเลยกับรายงานจุดบกพร่องที่เป็นส่วนต่อ ดังตัวอย่างนี้

Meta-bug หมายเลข 1437659 "[META] Reduce Backlog" ได้รับการวิเคราะห์โดย Bug triager ว่ารายงานจุดบกพร่องที่ควรอยู่กลุ่มเดียวกับ Meta-bug 1437659 คือ

(1) รายงานจุดบกพร่อง หมายเลข 1383599 ที่มีข้อมูล Summary คือ "browser.newtabpage.enabled = false is not working"

(2) รายงานจุดบกพร่อง หมายเลข 1385306 ที่มีข้อมูล Summary คือ "Make Activity Stream unprivileged: ensure about:newtab document runs with null principal"

จากตัวอย่างข้างต้นจะเห็นว่าข้อมูลในส่วน Summary ของ Meta-bug และ รายงานจุดบกพร่องส่วนต่อ ใช้คำที่ไม่สอดคล้องกันเลย จึงเป็นผลทำให้การจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดยอัตโนมัติ ได้ผลลัพธ์ที่ไม่สอดคล้องกับการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อโดย Bug triager

บทที่ 5
สรุป อภิปรายและข้อเสนอแนะ

งานวิจัยนี้เป็นงานวิจัยเพื่อค้นหารายงานจุดบกพร่องที่มีเป็นส่วนต่อกันโดยอัตโนมัติ ทั้งนี้เพื่อให้กระบวนการในการแก้ไขปัญหาจุดบกพร่องของซอฟต์แวร์ได้รับการแก้ไขเป็นไปอย่างมีประสิทธิภาพมากยิ่งขึ้น เพราะเมื่อสามารถรวบรวมรายงานจุดบกพร่องที่กล่าวถึงปัญหาในโดเมนปัญหาเดียวกันแล้ว นักพัฒนาซอฟท์แวร์สามารถวางแนวทางการแก้ไขจุดบกพร่องเหล่านั้นได้จากมุมมองในภาพรวมของปัญหาได้ทั้งหมด ซึ่งมีลำดับขั้นตอนการสรุปผล อภิปรายผลและข้อเสนอแนะดังนี้

5.1 สรุปเกี่ยวกับที่มาของงานวิจัย

จุดบกพร่อง หรือ “บั๊ก” เป็นปัญหาที่สามารถเกิดขึ้นในซอฟต์แวร์ และส่งผลกับซอฟต์แวร์จนทำให้ซอฟต์แวร์นั้นๆ ทำงานไม่ถูกต้องตามวัตถุประสงค์ของผู้พัฒนา หรือให้ผลลัพธ์ที่มีความผิดพลาด หากเป็นซอฟต์แวร์ขนาดเล็ก การค้นหาจุดบกพร่องอาจจะทำได้สำเร็จในขั้นตอนการทดสอบซอฟต์แวร์ แต่สำหรับซอฟต์แวร์ขนาดใหญ่ โดยเฉพาะซอฟต์แวร์ที่เป็นโอเพนซอร์ส ซึ่งการตรวจหาจุดบกพร่องอาจจะไม่สามารถตรวจหาได้ครบในขั้นตอนของการทดสอบ ดังนั้น การรวบรวมข้อมูลเกี่ยวกับจุดบกพร่องจากผู้ใช้งานซอฟต์แวร์ทั่วโลกจึงเป็นสิ่งจำเป็น ดังนั้นระบบติดตามรายงานจุดบกพร่อง (Bug tracking system: BTS) จึงได้ถูกพัฒนาขึ้น เพื่อใช้เป็นเครื่องมือในการรับรายงานจุดบกพร่องของซอฟต์แวร์จากผู้ใช้ทั่วโลก

โดยทั่วไปแล้ว รายงานจุดบกพร่องที่ถูกรายงานเข้ามา จะมี Bug traigers ซึ่งเป็นผู้เชี่ยวชาญทางด้านซอฟต์แวร์ทำหน้าที่เป็นผู้วิเคราะห์รายงานจุดบกพร่อง งานของ Bug traigers เช่น การคัดแยกรายงานที่ไม่ใช่รายงานจุดบกพร่องออกไป การตรวจสอบความซ้ำซ้อนของรายงานจุดบกพร่อง [3, 10-13] การวิเคราะห์ระดับความรุนแรงของจุดบกพร่องที่รายงานเข้ามา [14-18] ตลอดจนการวิเคราะห์เพื่อส่งมอบจุดบกพร่องไปยังนักพัฒนาซอฟต์แวร์ที่เหมาะสม อย่างไรก็ตาม เมื่อมีการรายงานจุดบกพร่องของซอฟต์แวร์เข้าไปยัง BTS อย่างต่อเนื่องและเป็นจำนวนมาก ทำให้ Bug traigers ต้องใช้เวลาเพิ่มขึ้น และเกิดการตกค้างของรายงานจุดบกพร่องจำนวนมากที่ยังไม่ได้รับการวิเคราะห์จาก Bug traigers [1, 3, 10, 11, 14]

จากปัญหาข้างต้นทำให้รายงานจุดบกพร่องกลายเป็นแหล่งข้อมูลที่กำลังได้รับการศึกษาในเชิงวิจัยมากมาย จุดประสงค์เพื่อให้เกิดการใช้รายงานจุดบกพร่องได้อย่างมีประสิทธิภาพ เหมาะสมและทำให้ซอฟต์แวร์ที่พบปัญหาได้รับการแก้ไขอย่างรวดเร็ว [19-22] จากการศึกษาพบว่า โดยทั่วไปแล้วปัญหาเกี่ยวกับการศึกษารายงานจุดบกพร่องที่ได้รับความนิยม ได้แก่ การพยากรณ์เวลาในการปรับปรุงซอฟต์แวร์ (Fix-time prediction) [1, 7] การมอบหมายรายงานจุดบกพร่อง (Bug report assignment) [23, 24] การทำนายระดับความสำคัญหรือความรุนแรงของจุดบกพร่อง (Severity หรือPriority prediction) [14-18] การเพิ่มประสิทธิภาพการรายงานจุดบกพร่อง (Bug report optimization) [20, 21, 25] และการตรวจจับจุดบกพร่องซ้ำซ้อน (Duplicated bug detection) [3, 11] เป็นต้น

อย่างไรก็ตาม แม้ปัญหาข้างต้นที่ได้กล่าวมาแล้วจะเป็นปัญหาที่ได้รับความนิยมในการศึกษา แต่ Sandusky และคณะ [26] และ Bhattacharya และ Neamtiu [1] ได้นำเสนอประเด็นที่น่าสนใจไว้ในปี ค.ศ. 2004 และ ค.ศ. 2011 ตามลำดับ เกี่ยวกับจุดบกพร่องส่วนต่อ นั่นคือ จุดบกพร่องหนึ่งๆ สามารถที่จะส่งผลต่อจุดบกพร่องอื่นๆ ได้ [1, 26] เช่น สมมติจุดบกพร่อง B_1 มีจุดบกพร่อง B_2 และ B_3 เป็นส่วนต่อ ดังนั้นแม้ว่าจุดบกพร่อง B_1 จะได้รับการแก้ไข แต่หากจุดบกพร่อง B_2 และ B_3 ที่เป็นส่วนต่อ ยังไม่ได้รับการแก้ไขให้สมบูรณ์ ก็ไม่สามารถแก้ไขจุดบกพร่อง B_1 ได้สมบูรณ์เช่นกัน ซึ่งปัญหาดังกล่าวเรียกว่า “จุดบกพร่องส่วนต่อ (Bug dependency)” และปัญหานี้ค่อนข้างส่งผลต่อขั้นตอนการบำรุงรักษาซอฟต์แวร์ (Software maintenance) ซึ่งเป็นขั้นตอนที่สำคัญมากขั้นตอนหนึ่งในวัฏจักรการพัฒนาซอฟต์แวร์ (Software development lifecycle: SDLC)

โดยทั่วไปแล้วขั้นตอนการบำรุงรักษาซอฟต์แวร์จะเป็นขั้นตอนที่มีค่าใช้จ่าย (Cost) มากที่สุดใน SDLC และเมื่อเพิ่มปัญหาเรื่อง “จุดบกพร่องส่วนต่อ (Bug dependency)” เข้าไป ก็จะยิ่งทำให้เกิดค่าใช้จ่ายในส่วนนี้เพิ่มมากขึ้น ทั้งในเรื่องทรัพยากรบุคคลและเวลา [1, 3, 10, 11, 14]

อย่างไรก็ตาม หากสามารถรวบรวมรายงานจุดบกพร่องส่วนต่อหรือรายงานที่เกี่ยวข้องกันเข้าด้วยกัน ก็น่าจะทำให้นักพัฒนาระบบได้มองเห็นว่า ในโดเมนปัญหาจุดบกพร่องหนึ่งๆ นั้น มีจุดบกพร่องใดบ้างที่เป็นส่วนต่อกันอยู่ ทำให้นักพัฒนาซอฟต์แวร์เพิ่มโอกาสที่จะแก้ไขจุดบกพร่องที่เกี่ยวข้องกันทั้งหมดได้อย่างสมบูรณ์มากยิ่งขึ้น ดังนั้นงานวิจัยนี้จึงนำเสนอกระบวนการในการรวมรวมรายงานจุดบกพร่องที่เป็นส่วนต่อเข้าด้วยกันแบบอัตโนมัติ

5.2 สรุปเกี่ยวกับกระบวนการวิจัยและผลลัพธ์ที่ได้จากงานวิจัย

5.2.1 สรุปเกี่ยวกับชุดข้อมูล

ชุดข้อมูลที่ใช้ในการศึกษานี้มี 3 ชุดข้อมูล ได้แก่

1. ชุดข้อมูลรายงานจุดบกพร่องของมอซิลลา Firefox ที่มีการดาวน์โหลดจากระบบติดตามจุดบกพร่องของ Mozilla (Bugzilla@Mozilla) จำนวนทั้งสิ้น 176,971 รายงาน เมื่อวันที่ 1 ตุลาคม 2560 โดยรายงานจุดบกพร่องเหล่านั้นมีสถานะเป็น NEW, ASSIGNED, REOPENED, RESOLVED, VERIFIED และ CLOSED เพราะสถานะเหล่านี้เป็นการยืนยันว่าเป็นรายงานจุดบกพร่องจริงๆ และไม่ได้เป็นรายงานจุดบกพร่องที่มีความซ้ำซ้อน โดยจัดเก็บรายงานจุดบกพร่องเหล่านี้ในโครงสร้าง XML สำหรับชุดข้อมูลรายงานจุดบกพร่องของมอซิลลา Firefox นี้แบ่งเป็นรายงานจุดบกพร่องที่เป็น Meta-bug จำนวน 15,396 รายงาน ที่ส่วนที่เหลือเป็นรายงานที่มีสถานะเป็น depend on ซึ่งสถานะเป็นการแสดงให้ทราบว่าเป็น “เป็นรายงานจุดบกพร่องส่วนต่อ” ที่กำหนดโดย Bug triagers โดยชุดข้อมูลรายงานจุดบกพร่องของมอซิลลา Firefox บางส่วนนี้จะเป็นชุดข้อมูลหลักที่ใช้ในการศึกษาเรื่องการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่สอดคล้องกันเข้าด้วยกัน และบางส่วนจะใช้ในการศึกษาเรื่องคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสม อย่างไรก็ตาม ในงานวิจัยนี้เลือกใช้รายงานจุดบกพร่อง Meta-bug ที่มีรายงานส่วนต่ออย่างน้อย 10 รายงาน

2 ชุดข้อมูลรายงานจุดบกพร่องของมอซิลลา Core ที่มีการดาวน์โหลดจากระบบติดตามจุดบกพร่องของ Mozilla เมื่อวันที่ 1 ตุลาคม 2560 โดยมีจำนวนรายงานจุดบกพร่องทั้งสิ้น

1,300 รายงาน และจัดเก็บรายงานจุดบกพร่องเหล่านี้ในโครงสร้าง XML สำหรับข้อมูลชุดนี้จะนำมาใช้ในการสร้างกฎเชิงไวยากรณ์

3. ชุดข้อมูลรายงานจุดบกพร่องมาตรฐานของ Herzig [37] โดยมีจำนวนรายงานจุดบกพร่องทั้งสิ้น 7,401 รายงาน และจัดเก็บรายงานจุดบกพร่องเหล่านี้ในโครงสร้าง XML สำหรับข้อมูลชุดนี้จะนำมาใช้ในจะใช้ในการศึกษาเรื่องคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสม

5.2.2 สรุปเกี่ยวกับคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมที่ใช้ในการศึกษา

สำหรับคุณลักษณะของรายงานจุดบกพร่องที่ใชในการวิจัยนี้ จะใช้คุณลักษณะของรายงานจุดบกพร่องพื้นฐานที่เคยใช้ในงานวิจัยด้านรายงานจุดบกพร่องสำคัญที่ผ่านมา [11, 14, 19, 40-42, 73, 81] โดยคุณลักษณะของรายงานจุดบกพร่องเหล่านั้นได้แก่ Unigram, Bigram และ Compound words จากนั้นงานวิจัยนี้ได้นำคุณลักษณะของรายงานจุดบกพร่องเหล่านี้มาศึกษาในรูปแบบที่เป็นไปได้ โดยได้ทำการศึกษาคุณลักษณะของรายงานจุดบกพร่องใน 7 รูปแบบ อันได้แก่

(1) คำเดี่ยว (Unigram)
(2) กลุ่มคำแบบสองคำ (Bigram)
(3) คำจากกลุ่มคำผสม (Compound words)
(4) คำเดี่ยวร่วมกับกลุ่มคำแบบสองคำ (Unigram and bigram)
(5) คำเดี่ยวร่วมกับคำจากกลุ่มคำผสม (Unigram and compound words)
(6) กลุ่มคำแบบสองคำร่วมกับคำจากกลุ่มคำผสม (Bigram and compound words)
(7) คำเดียวร่วมกับกลุ่มคำแบบสองคำและคำจากกลุ่มคำผสม (Combination: unigram and bigram and compound words)

ซึ่งการศึกษาเกี่ยวกับคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมจะทำการศึกษาผ่านงานวิจัยด้านการจำแนกเอกสารที่เกี่ยวข้องกับรายงานจุดบกพร่องใน 2 ลักษณะคือ การศึกษาด้านการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug และการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง

และผลสรุปจากการศึกษาเกี่ยวกับคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมที่ทำการศึกษาผ่านงานวิจัยทั้ง 2 เรื่อง พบว่าคุณลักษณะของรายงานจุดบกพร่องที่เป็น “Unigram + compound words” เป็นคุณลักษณะที่ให้ผลลัพธ์ที่ดีในการจำแนกรายงานจุดบกพร่อง

5.2.3 สรุปเกี่ยวกับกรอบการวิจัยในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ

สำหรับกรอบการวิจัยในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อนั้น จะประกอบด้วย 4 ขั้นตอนได้แก่ การเตรียมเบื้องต้น การเตรียมข้อมูล การปรับพารามิเตอร์สำหรับฟังก์ชันการค้นคืน และการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ

ขั้นตอนที่ 1: การเตรียมเบื้องต้น

ในขั้นตอนนี้ เป็นขั้นตอนในการนำชุดข้อมูลมอซิลลา Core มาสร้างกฎเชิงไวยากรณ์ซึ่งเป็นกฎที่จะใช้สำหรับการตัดคำนามแบบ 2 คำในรายงานจุดบกพร่องที่เรียกว่า “Bigram” ซึ่งขั้นตอนหลักๆ ในการสร้างกฎไวยากรณ์คือ การทำ POS tagging ผ่าน Stanford Parser เพื่อสกัดเอาส่วนที่เป็นกลุ่มคำนามออกจากประโยค จากนั้นนำกลุ่มคำนามเหล่านั้นมาเรียนรู้เพื่อสกัดรูปแบบ

(Pattern) ร่วมด้วยห่วงโซ่มาร์คอฟ ซึ่งท้ายที่สุดได้รูปแบบที่ใช้เป็นกฎเชิงไวยากรณ์จำนวนทั้งสิ้น 57 รูปแบบ

ขั้นตอนที่ 2: การเตรียมข้อมูลรายงานจุดบกพร่อง

ในขั้นตอนนี้จะเป็นขั้นตอนในการเตรียมข้อมูลรายงานจุดบกพร่องของ Firefox ให้อยู่ในรูปแบบที่พร้อมต่อการนำไปประมวลผลในขั้นตอนต่อไป โดยในขั้นตอนนี้จะประกอบไปด้วยการตัดคำ (ซึ่งกฎเชิงไวยากรณ์ที่ได้จากขั้นตอนที่ 1 จะนำมาตัดคำแบบ Bigram) การตัดคำหยุด และ Stemming โดยคุณลักษณะที่ใช้ในการศึกษามี 4 คุณลักษณะได้แก่ Unigram, Unigram + bigram, Unigram + compound words และ Combination

สำหรับการให้น้ำหนักคำในงานวิจัยนี้ประกอบไปด้วยการให้น้ำหนักแบบ *tf*, *tf-idf*, *BM25* และ *MATF* สำหรับผลลัพธ์ที่ได้ในขั้นตอนนี้จะอยู่ในรูปแบบถุงคำจำนวน 16 แบบ นั่นคือ

(1) ถุงคำที่มีคุณลักษณะแบบ Unigram ที่มีการให้น้ำหนัก 4 แบบคือ *tf*, *tf-idf*, *BM25* และ *MATF*

(2) ถุงคำที่มีคุณลักษณะแบบ Unigram + bigram ที่มีการให้น้ำหนัก 4 แบบ คือ *tf*, *tf-idf*, *BM25* และ *MATF*

(3) ถุงคำที่มีคุณลักษณะแบบ Unigram + compound words ที่มีการให้น้ำหนัก 4 แบบคือ *tf*, *tf-idf*, *BM25* และ *MATF*

(4) ถุงคำที่มีคุณลักษณะแบบ combination ที่มีการให้น้ำหนัก 4 แบบคือ *tf*, *tf-idf*, *BM25* และ *MATF*

ขั้นตอนที่ 3: การปรับพารามิเตอร์สำหรับฟังก์ชันการค้นคืน

ในขั้นตอนนี้จะเป็นขั้นตอนในการเตรียมข้อมูลรายงานจุดบกพร่องของมอซิลลา Firefox บางส่วนมาเรียนรู้เพื่อให้ได้ค่าพารามิเตอร์ที่เหมาะสมสำหรับฟังก์ชันการค้นคืน โดยการเรียนรู้เพื่อให้ได้ค่าพารามิเตอร์ที่เหมาะสมนั้น ซึ่งจะเรียนรู้ 3 พารามิเตอร์สำหรับฟังก์ชันการค้นคืน นั่นคือ พารามิเตอร์ของการวิเคราะห์ความคล้ายคลึงโคไซน์ พารามิเตอร์ของ BM25F และพารามิเตอร์ของ MATF

ขั้นตอนที่ 4: การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ

สำหรับการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อนั้น จะสามารถแบ่งการศึกษาตามเทคนิคที่ใช้ในการศึกษาได้ 3 รูปแบบดังนี้

1. การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริง ซึ่งสามารถแบ่งการศึกษาออกเป็น 2 รูปแบบคือ

1.1 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริงแบบเคมีนแบบเพิ่มข้อจำกัด

1.2 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริงเคมีนทรงกลมแบบเพิ่มข้อจำกัด

2. การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึง ซึ่งสามารถแบ่งการศึกษาออกเป็น 3 รูปแบบคือ

2.1 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึงเทคนิคโคไซน์

2.2 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึงทคนิค BM25

2.3 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึงทคนิค MATF

3. การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืน ซึ่งสามารถแบ่งการศึกษาออกเป็น 3 รูปแบบคือ

3.1 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยด้วยฟังก์ชันการค้นคืนแบบโคไซน์

3.2 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยด้วยฟังก์ชันการค้นคืนแบบ BM25F

3.3 การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยด้วยฟังก์ชันการค้นคืนแบบ MATF

5.3 สรุปผลการศึกษา

ภายหลังการทดสอบการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยกระบวนการต่างๆ ที่นำเสนอข้างต้น พบว่า คุณลักษณะของรายงานจุดบกพร่องที่เป็น “Unigram + compound words” น่าเป็นคุณลักษณะที่ให้ผลลัพธ์ที่ดีในการจำแนกรายงานจุดบกพร่อง ซึ่งสอดคล้องกับผลลัพธ์ที่ได้การศึกษาด้านการตรวจจับรายงานจุดบกพร่องแบบ bug และ non-bug และการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง [19, 40-42]

การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริง พบว่าการให้น้ำหนักแบบ *tf-idf* ให้ประสิทธิภาพที่ดีที่สุด และการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริงเคมีนทรงกลมแบบเพิ่มข้อจำกัด ให้ผลลัพธ์ที่น่าพอใจกว่าการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยเทคนิคคลัสเตอริงเคมีนแบบเพิ่มข้อจำกัด เมื่อพิจารณาจากค่า F1 และค่าความถูกต้อง

การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึง พบว่าหากใช้เทคนิคแบบ BM25 จะให้ผลลัพธ์ที่ดีกว่า โคไซน์ และ MATF เมื่อพิจารณาจากค่าพื้นที่ใต้โค้ง AUC

การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืน พบว่าหากใช้เทคนิคแบบ BM25F ที่ใช้ค่า Threshold ที่ 0.7 จะให้ผลลัพธ์ที่ดีกว่าเทคนิคแบบ โคไซน์ และ เทคนิคแบบ MATF เมื่อพิจารณาจากค่าพื้นที่ใต้โค้ง AUC

สำหรับการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อทั้ง 3 รูปแบบ พบว่าการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยการวิเคราะห์ความคล้ายคลึง จะให้ผลลัพธ์ที่ดีที่สุดเมื่อพิจารณาจากค่าพื้นที่ใต้โค้ง AUC อย่างไรก็ตาม หากเพิ่มข้อมูลในการเรียนรู้ค่าพารามิเตอร์ของฟังก์ชันการค้นคืนให้มากขึ้น การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืนจะมีแนวโน้มที่สูงขึ้น อย่างไรก็ตามการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อด้วยฟังก์ชันการค้นคืนจะใช้เวลาในการประมวลผลสูงมาก

ผลการทดลองที่ได้ในงานวิจัยนี้ เมื่อนำไปเทียบกับผลการทดลองในงานวิจัยอื่นๆ ที่ใช้เทคนิคคล้ายๆ กัน โดยเฉพาะเทคนิคการวิเคราะห์ความคล้ายคลึงที่พบในงาน [11, 27, 28, 92] พบว่าผลการทดลองที่ได้ในงานวิจัยนี้มีค่าที่สูงกว่า ในหลายงานวิจัยที่ใช้การประเมินรูปแบบเดียวกัน [3, 11, 12] อาจจะเพราะ 5 ประเด็นหลักๆ คือ

1. ในงานวิจัยนี้ได้ศึกษาจากงานวิจัยก่อนหน้าและเห็นว่า โดยส่วนใหญ่การวิเคราะห์รายงานจุดบกพร่องในทุกๆ ด้าน จะใช้ประโยชน์จากรายงานจุดบกพร่องในส่วนของ Summary เพราะจะเน้นความเร็วในการประมวลผล เนื่องจากเป็นการสรุปปัญหาแบบข้อความสั้น (Short text) แต่โดยทั่วไปแล้วข้อความสั้นจะขาดการอธิบายที่ชัดเจน [94-96] และคำที่มีในรายงานจุดบกพร่องนั้นอาจจะน้อยเกินไปจนยากต่อการวิเคราะห์ นั่นอาจจะเป็นสาเหตุที่ทำให้ผลการศึกษาในงานวิจัยที่เกี่ยวข้องกับรายงานจุดบกพร่องก่อนหน้านี้ ยังมีประสิทธิภาพที่ไม่สูงนัก ซึ่งในการศึกษานี้ได้เล็งเห็นปัญหาดังกล่าว จึงได้ศึกษาครอบคลุมไปถึงการเลือกใช้คุณลักษณะของรายงานจุดบกพร่อง เทคนิคการให้น้ำหนัก และเทคนิคการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อที่เหมาะสม

2. นอกจากการพิจารณาปัญหาข้อความสั้น ในงานวิจัยนี้ยังพบว่าในส่วนของ Summary ในรายงานจุดบกพร่องที่เป็น Meta-bug นั้น บางรายงานก็อาจจะมีส่วนของ Summary ที่ยาวมาก และบางรายงานก็อาจจะมีส่วนของ Summary ที่สั้นมาก ดังนั้นเมื่อใช้ Meta-bug เหล่านั้นเป็น Query ในการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อ รายงานจุดบกพร่องส่วนต่อเหล่านี้มีโอกาสถูกจัดกลุ่มเข้าไปยัง Meta-bug ที่มี Summary ที่ยาว ซึ่งปัญหานี้คือปัญหาความไม่สมดุลของข้อมูล (Imbalanced data) ซึ่งก็ได้รับการพิจารณาในงานวิจัยนี้เช่นกัน ซึ่งประเด็นนี้ได้ทำการปรับแก้ผ่านเทคนิคการให้น้ำหนักคำ นั่นคือการใช้เทคนิคการให้น้ำหนักคำแบบ BM25 และ MATF โดยสองเทคนิคนี้จะให้ความสำคัญกับเอกสารที่มีข้อความสั้นและยาวแตกต่างกัน ซึ่งงานวิจัยนี้เป็นงานวิจัยแรกทางด้านรายงานจุดบกพร่องที่มีการใช้ 2 เทคนิคนี้ในการให้น้ำหนักคำ [100, 101] เพื่อแก้ปัญหาความไม่สมดุลของข้อมูลในรายงานจุดบกพร่อง

3. จากข้อที่ (1) ข้างต้น ที่มีการกล่าวถึงการศึกษา เพื่อการเลือกใช้คุณลักษณะของรายงานจุดบกพร่องที่เหมาะสม ซึ่งในการศึกษาพบว่างานวิจัยก่อนหน้าที่เกี่ยวกับรายงานจุดบกพร่องมักจะเลือกใช้ Unigram เป็นคุณลักษณะของรายงานจุดบกพร่อง อย่างไรก็ตาม คำในลักษณะของ Unigram ทั่วๆ ไป ดังนั้นทำให้การวิเคราะห์โดยเฉพาะเรื่องการจำแนกข้อมูลหรือการจัดกลุ่ม อาจจะเป็นไปได้ยาก แม้ในงานวิจัยก่อนหน้าบางงานวิจัยจะมีการใช้ Compund words เข้ามาร่วมด้วย แต่ Compound words เหล่านั้น ก็อาจจะเป็นคำที่แสดงถึงลักษณะของซอฟต์แวร์แบบทั่วไป ไม่ได้เฉพาะเจาะจง ดังนั้นการวิเคราะห์ข้อมูลรายงานจุดบกพร่องจึงยังคงต้องการการปรับปรุง

ดังนั้น ในงานวิจัยนี้ จึงได้ศึกษาการใช้ Bigram แบบกลุ่มคำนามแบบ 2 คำ ซึ่งจากการศึกษาที่ผ่านมาพบว่ายังไม่เคยมีการนำเสนอการใช้คุณลักษณะของรายงานจุดบกพร่องแบบนี้ โดยในงานวิจัยได้นำเอา Bigram ไปรวมกับ Unigram เป็น Unigram + bigram นอกจากนี้ยังมีการใช้คุณลักษณะของรายงานจุดบกพร่องที่เรียกว่า Combination (Unigram + bigram + compound words) ผลลัพธ์ที่ได้จากการศึกษาดังกล่าว พบว่าให้ผลที่น่าพอใจ โดยเฉพาะการใช้คุณลักษณะของรายงานจุดบกพร่องแบบ Combination จะให้ประสิทธิภาพในการจัดกลุ่มที่ดีกว่าการใช้ Unigram เป็นคุณลักษณะของรายงานจุดบกพร่องเพียงอย่างเดียวอย่างเห็นได้ชัด แต่จะใช้เวลาเพิ่มขึ้นเล็กน้อยเมื่อเปรียบเทียบกับการใช้ Unigram เพียงอย่างเดียว

4. จากข้อที่ (3) ข้างต้น แม้ว่าในการศึกษานี้จะพบคุณลักษณะของรายงานจุดบกพร่องที่เหมาะสมต่อการศึกษา แต่ก็ยังพบว่าการใช้ “คำ” เพียงอย่างเดียว อาจจะไม่เพียงพอต่อการวิเคราะห์เพื่อการจัดกลุ่มรายงานจุดบกพร่องส่วนต่อ เพราะว่า “คำ” ที่ใช้ในรายงานจุดบกพร่อง ไม่ว่าจะเป็นปัญหาในโดเมนหรือส่วนใดของซอฟต์แวร์ ก็อาจจะใช้ “คำ” ที่คล้ายๆ กัน ทำให้ยากต่อการวิเคราะห์การจัดกลุ่ม ดังนั้น การเพิ่มคุณลักษณะของรายงานจุดบกพร่องอย่างอื่นเข้าไปด้วย ได้แก่ ข้อมูลที่บอกว่ารายงานจุดบกพร่องนี้ถูกรายงานในซอฟต์แวร์ตัวใด (Product) ข้อมูลที่บอกรายงานจุดบกพร่องนี้ถูกรายงานในส่วนใดของซอฟต์แวร์ (Component) ข้อมูลที่บอกรายงานจุดบกพร่องนี้มีระดับความสำคัญมากน้อยเพียงใดในซอฟต์แวร์นั้นๆ (Prioirty) ข้อมูลที่บอกรายงานจุดบกพร่องนี้มีจุดบกพร่องที่มีระดับความรุนแรงมากน้อยเพียงใดในซอฟต์แวร์นั้นๆ (Serverity) และข้อมูลในส่วนของคำอธิบาย (Description)

ซึ่งการเพิ่มคุณลักษณะของรายงานจุดบกพร่องเหล่านี้เข้าไปร่วมกับการใช้คุณลักษณะของรายงานจุดบกพร่องที่เป็น “คำ” จาก Summary ทำให้สามารถเพิ่มประสิทธิภาพในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อให้ดีขึ้น

5. นอกจากการศึกษาเรื่องคุณลักษณะของรายงานจุดบกพร่องและการให้น้ำหนักที่มีผลอย่างมากต่อการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อแล้ว ในงานวิจัยยังได้ศึกษาเทคนิคที่หลากหลายในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ โดยเฉพาะในเรื่องเคมีนแบบเพิ่มข้อจำกัด, เคมีนทรงกลมแบบเพิ่มข้อจำกัด, การวัดความคล้ายคลึงด้วยเทคนิคโคไซน์ การวัดความคล้ายคลึงด้วยเทคนิค BM25 การวัดความคล้ายคลึงด้วยเทคนิค MATF และ การใช้ฟังก์ชันการค้นคืน REP แบบปรับการใช้คุณลักษณะภายใน REP ที่แตกต่างจาก REP ที่เป็นต้นฉบับ

อย่างไรก็ตาม จะเห็นได้ว่าบางเทคนิคที่ใช้ในงานวิจัยนี้ จะใช้เวลาเป็นค่อนข้างมากในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ อย่างไรก็ตาม จากงานวิจัย Runeson และคณะ [10] ได้กล่าวไว้ว่า โดยทั่วไปแล้วการวิเคราะห์รายงานจุดบกพร่องโดย Bug triager นั้นจะใช้เวลาประมาณ 30 นาทีต่อรายงานจุดบกพร่อง 1 รายงาน เพื่อวิเคราะห์ในหลายๆ ประเด็น เช่น การวิเคราะห์รายงานจุดบกพร่องว่าเป็น bug หรือ non-bug การวิเคราะห์ว่ารายงานจุดบกพร่องนั้นซ้ำซ้อนหรือไม่ การวิเคราะห์ระดับความรุนแรง การวิเคราะห์ระดับความสำคัญของจุดบกพร่อง รวมไปถึงการวิเคราะห์ส่วนต่อของรายงานจุดบกพร่อง

สมมติว่ารายงานจุดบกพร่อง 1 รายงาน Bug triager จะใช้เวลาวิเคราะห์ส่วนต่อของรายงานจุดบกพร่องประมาณ 5 นาที หากเป็นการวิเคราะห์รายงานจุดบกพร่องจำนวน 4,781 รายงาน (เท่ากับจำนวนรายงานจุดบกพร่องที่ใช่ศึกษาในงานวิจัยนี้) จะพบว่า Bug triager จะใช้เวลาถึง 23,905 นาที หรือ 16 วัน 14 ชั่วโมง 25 นาที ดังนั้นหากเปรียบเทียบเวลาในการวิเคราะห์การจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อที่นำเสนอในงานวิจัยฉบับนี้ จะเห็นได้ว่าสามารถลดเวลาในการประมวลผลได้ค่อนข้างมาก และแน่นอนว่าค่าใช้จ่ายในขั้นตอนของการปรับปรุงซอฟต์แวร์ก็น่าจะมีโอกาสลดลงตามไปด้วย

5.4 ปัญหาและอุปสรรค

1. เนื่องรายงานจุดบกพร่องที่เป็นส่วนต่อ หรือรายงานจุดบกพร่องที่มีสถานะเป็น Depend on หลายฉบับไม่มี “คำ” ใดๆ ที่สอดคล้องกับ Meta-bug หรือรายงานจุดบกพร่องอื่นๆ ที่อยู่ในกลุ่มเดียวกัน แต่ Bug triagers ได้ระบุไว้ว่ารายงานเหล่านั้นเป็นส่วนต่อกันสอดคล้องกัน ทำให้ไม่สามารถจัดกลุ่มรายงานจุดบกพร่องได้ถูกต้อง

2. รายงานจุดบกพร่องบางรายงานจะประกอบด้วยจำนวน “คำ” ที่น้อยมาก ทำให้ไม่สามารถสร้างตัวแทนเอกสารที่เหมาะสมได้ ซึ่งก็คือปัญหาของข้อความสั้น (Short text)

3. รายงานจุดบกพร่องที่เป็น Meta-bug บางรายงานมีข้อความจำนวนมาก แต่บางรายงานจุดบกพร่องที่เป็น Meta-bug กลับมีข้อความสั้น ดังนั้นในการจัดกลุ่มรายงานจุดบกพร่องที่เป็นส่วนต่อ หรือ depend on เข้ากับ Meta-bug หลายๆ รายงาน มีแนวโน้มจะถูกจัดเข้าไปอยู่กับรายงานจุดบกพร่องที่เป็น Meta-bug ที่มีข้อความจำนวนมาก ซึ่งเป็นปัญหาอย่างหนึ่งด้านความสมดุลข้อมูล (Imbalanced data)

บรรณานุกรม

[1] Bhattacharya P, Neamtiu I. Bug-fix time prediction models: can we do better? Proceedings of the 8th Working Conference on Mining Software Repositories; ACM 2011; 207-210.

[2] Ferreira I, Cirilo E, Vieira V, Mourao F. Bug Report Summarization: An Evaluation of Ranking Techniques. Software Components, Architectures and Reuse (SBCARS), 2016 X Brazilian Symposium on; IEEE 2016; 101-110.

[3] Tian Y, Sun C, Lo D. Improved duplicate bug report identification. Software Maintenance and Reengineering (CSMR), 2012 16th European Conference on; IEEE 2012; 385-390.

[4] Rajlich V. Software Engineering: The Current Practices. New York: CRC Press; 2011.

[5] Reis CR, de Mattos Fortes RP. An overview of the software engineering process and tools in the Mozilla project. Proceedings of the Open Source Software Development Workshop; 2002; 155-175.

[6] Nguyen AT, Nguyen TT, Al-Kofahi J, Nguyen HV, Nguyen TN. A topic-based approach for narrowing the search space of buggy files from a bug report. Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on; IEEE 2011; 263-272.

[7] Weiss C, Premraj R, Zimmermann T, Zeller A. How long will it take to fix this bug? Proceedings of the Fourth International Workshop on Mining Software Repositories; IEEE Computer Society 2007; 1.

[8] Who Uses Bugzilla? [online]. 8 June 2017 [cited 24 September 2017]; https://www.bugzilla.org/installation-list/.

[9] Hooimeijer P, Weimer W. Modeling bug report quality. ACM, 2007.

[10] Runeson P, Alexandersson M, Nyholm O. Detection of duplicate defect reports using natural language processing. Proceedings of the 29th international conference on Software Engineering; IEEE Computer Society 2007; 499-510.

[11] Jalbert N, Weimer W. Automated duplicate detection for bug tracking systems. Dependable Systems and Networks With FTCS and DCC, 2008 DSN 2008 IEEE International Conference on; IEEE 2008; 52-61.

[12] Gopalan RP, Krishna A. Duplicate bug report detection using clustering. Software Engineering Conference (ASWEC), 2014 23rd Australian; IEEE 2014; 104-109.

[13] Lee C-Y, Hu D-D, Feng Z-Y, Yang C-Z. Mining Temporal Information to Improve Duplication Detection on Bug Reports. Advanced Applied Informatics (IIAI-AAI),

2015 IIAI 4th International Congress on; IEEE 2015; 551-555.

[14] Lamkanfi A, Demeyer S, Giger E, Goethals B. Predicting the severity of a reported bug. Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on; IEEE 2010; 1-10.

[15] Menzies T, Marcus A. Automated severity assessment of software defect reports. Software Maintenance, 2008 ICSM 2008 IEEE International Conference on; IEEE 2008; 346-355.

[16] Yu L, Tsai W-T, Zhao W, Wu F. Predicting defect priority based on neural networks. Advanced Data Mining and Applications 2010; 356-367.

[17] Kanwal J, Maqbool O. Bug prioritization to facilitate bug report triage. Journal of Computer Science and Technology 2012; 27[2]: 397-412.

[18] Tian Y, Lo D, Sun C. Information retrieval based nearest neighbor classification for fine-grained bug severity prediction. Reverse Engineering (WCRE), 2012 19th Working Conference on; IEEE 2012; 215-224.

[19] Antoniol G, Ayari K, Di Penta M, Khomh F, Guéhéneuc Y-G. Is it a bug or an enhancement?: a text-based approach to classify change requests. Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds; ACM 2008; 23.

[20] Bettenburg N, Just S, Schröter A, Weiß C, Premraj R, Zimmermann T. Quality of bug reports in Eclipse. Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange; ACM 2007; 21-25.

[21] Bettenburg N, Just S, Schröter A, Weiss C, Premraj R, Zimmermann T. What makes a good bug report? Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering; ACM 2008; 308-318.

[22] Xia X, Lo D, Wen M, Shihab E, Zhou B. An empirical study of bug report field reassignment. Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on; IEEE 2014; 174-183.

[23] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs. Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering; ACM 2009; 111-120.

[24] Bhattacharya P, Neamtiu I. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging. Software Maintenance (ICSM), 2010 IEEE International Conference on; IEEE 2010; 1-10.

[25] Xie J, Zhou M, Mockus A. Impact of triage: a study of mozilla and gnome.

Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on; IEEE 2013; 247-250.

[26] Sandusky RJ, Gasser L, Ripoche G. Bug report networks: Varieties, strategies, and impacts in af/oss development community. Proc of 1st Int'l Workshop on Mining Software Repositories; IET 2004; 80-84.

[27] Rocha H, De Oliveira G, Marques-Neto H, Valente MT. NextBug: a Bugzilla extension for recommending similar bugs. Journal of Software Engineering Research and Development 2015; 3[1]: 3.

[28] Rocha H, Oliveira G, Maques-Neto H, Valente M. Nextbug: A tool for recommending similar bugs in open-source systems. V Brazilian Conference on Software: Theory and Practice–Tools Track (CBSoft Tools) SBC, Maceio, AL, Brazil; 2014; 53-60.

[29] Podgurski A, Leon D, Francis P, Masri W, Minch M, Sun J, et al. Automated support for classifying software failure reports. Software Engineering, 2003 Proceedings 25th International Conference on; IEEE 2003; 465-475.

[30] Tamrawi A, Nguyen TT, Al-Kofahi J, Nguyen TN. Fuzzy set-based automatic bug triaging: NIER track. Software Engineering (ICSE), 2011 33rd International Conference on; IEEE 2011; 884-887.

[31] Anvik J. Automating bug report assignment. Proceedings of the 28th international conference on Software engineering; ACM 2006; 937-940.

[32] Uddin J, Ghazali R, Deris MM, Naseem R, Shah H. A survey on bug prioritization. Artificial Intelligence Review 2017; 47[2]: 145-180.

[33] Raymond E. The cathedral and the bazaar. Philosophy & Technology 1999; 12[3]: 23.

[34] Murphy G, Cubranic D. Automatic bug triage using text categorization. Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering; 2004;

[35] Zhang J, Wang X, Hao D, Xie B, Zhang L, Mei H. A survey on bug-report analysis. Science China Information Sciences 2015; 58[2]: 1-24.

[36] Davies S, Roper M. What's in a bug report? Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement; ACM 2014; 26.

[37] 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; IEEE Press 2013; 392-401.

[38] Pingclasai N, Hata H, Matsumoto K-i. Classifying bug reports to bugs and other

requests using topic modeling. Software Engineering Conference (APSEC), 2013 20th Asia-Pacific; IEEE 2013; 13-18.

[39] Limsettho N, Hata H, Monden A, Matsumoto K. Automatic unsupervised bug report categorization. Empirical Software Engineering in Practice (IWESEP), 2014 6th International Workshop on; IEEE 2014; 7-12.

[40] Zhou J, Zhang H, Lo D. Where should the bugs be fixed?-more accurate information retrieval-based bug localization based on bug reports. Proceedings of the 34th International Conference on Software Engineering; IEEE Press 2012; 14-24.

[41] Almhana R, Mkaouer W, Kessentini M, Ouni A. Recommending relevant classes for bug reports using multi-objective search. Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering; ACM 2016; 286-295.

[42] Ye X, Bunescu R, Liu C. Mapping bug reports to relevant files: A ranking model, a fine-grained benchmark, and feature evaluation. IEEE Transactions on Software Engineering 2016; 42[4]: 379-402.

[43] Ohira M, Hassan AE, Osawa N, Matsumoto K-i. The impact of bug management patterns on bug fixing: A case study of Eclipse projects. Software Maintenance (ICSM), 2012 28th IEEE International Conference on; IEEE 2012; 264-273.

[44] Indurkhya N, Damerau FJ. Handbook of natural language processing. CRC Press; 2010.

[45] Dale R, Moisl H, Somers H. Handbook of natural language processing. CRC Press; 2000.

[46] Selvam B, Abirami S. A survey on opinion mining framework. International Journal of Advanced Research in computer and communication Engineering 2013; 2[9]: 3544-3549.

[47] Mumu T. Social Network Opinion and Posts Mining for Community Preference Discovery. 2013;

[48] Frankes WB, Baeza-Yates R. Information retrieval: Data structure & algorithms. PrenticeHall, Englewood cliffs, NJ 1992;

[49] Porter MF. An algorithm for suffix stripping. Program 1980; 14[3]: 130-137.

[50] Porter MF. Snowball: A language for stemming algorithms. 2001.

[51] Dale R, Moisl H, Somers H. Handbook of natural language processing. MIT Press, 2001.

[52] Jurafsky D, Martin JH. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition. Prentice

Hall, Pearson Education International, 2009.

[53] Daniel Jurafsky JHM. Speech and Language Processing. [online].25 March 2018]; https://web.stanford.edu/~jurafsky/slp3/9.pdf.

[54] เฉลิมวุฒิ ไวชนะ. ทฤษฎีพื้นฐานของ Hidden Markov Model. [ออนไลน์].สืบค้นเมือ 25 มีนาคม 2561]; http://pioneer.netserv.chula.ac.th/~wlunchak/ping.pdf.

[55] อัครพล เอกวงศ์อนันต์. การระบุคำไทยและคำทับศัพท์ด้วยแบบจำลองเอ็นแกรม. วิทยานิพนธ์ปริญญาอักษรศาสตรมหาบัณฑิต. กรุงเทพฯ: จุฬาลงกรณ์มหาวิทยาลัย; 2548.

[56] Yan J. Text Representation. In: Liu L, ÖZsu MT, eds. *Encyclopedia of Database Systems*. Boston, MA: Springer US 2009:3069-3072.

[57] Salton G, Wong A, Yang C-S. A vector space model for automatic indexing. Communications of the ACM 1975; 18[11]: 613-620.

[58] Sparck Jones K. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation 1972; 28[1]: 11-21.

[59] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information processing & management 1988; 24[5]: 513-523.

[60] Quan X, Wenyin L, Qiu B. Term weighting schemes for question categorization. IEEE transactions on pattern analysis and machine intelligence 2011; 33[5]: 1009-1021.

[61] Yang C-Z, Du H-H, Wu S-S, Chen X. Duplication detection for software bug reports based on bm25 term weighting. Technologies and Applications of Artificial Intelligence (TAAI), 2012 Conference on; IEEE 2012; 33-38.

[62] Robertson SE, Walker S, Jones S, Hancock-Beaulieu MM, Gatford M. Okapi at TREC-3. Nist Special Publication Sp 1995; 109109.

[63] Robertson SE, Walker S. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval; Springer-Verlag New York, Inc. 1994; 232-241.

[64] Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval: The Concepts and Technology behind Search (ACM Press Books). Addison-Wesley Professional Harlow, 2011.

[65] Paik JH. A novel TF-IDF weighting scheme for effective ranking. Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval; ACM 2013; 343-352.

[66] Singhal A, Salton G, Mitra M, Buckley C. Document length normalization. Information Processing & Management 1996; 32[5]: 619-633.

[67] Chen K, Zhang Z, Long J, Zhang H. Turning from TF-IDF to TF-IGM for term

weighting in text classification. Expert Systems with Applications 2016; 66245-260.
[68] ปริญญา สงวนสัตย์. Machine Learning การเรียนรู้ของเครื่อง. กรุงเทพฯ; 2560.
[69] Tsai C-F, Tsai C-T, Hung C-S, Hwang P-S. Data mining techniques for identifying students at risk of failing a computer proficiency test required for graduation. Australasian Journal of Educational Technology 2011; 27[3]:
[70] MacQueen J. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability; Oakland, CA, USA 1967; 281-297.
[71] Wagstaff K, Cardie C, Rogers S, Schrödl S. Constrained k-means clustering with background knowledge. ICML; 2001; 577-584.
[72] Dhillon IS, Modha DS. Concept decompositions for large sparse text data using clustering. Machine learning 2001; 42[1-2]: 143-175.
[73] Sun C, Lo D, Khoo S-C, Jiang J. Towards more accurate retrieval of duplicate bug reports. Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering; IEEE Computer Society 2011; 253-262.
[74] ธัชพงศ์ กตัญญูกุล. การเรียนรู้ของเครื่องเบื้องต้น,Introduction to Machine Learning. ขอนแก่น: คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น; 2560.
[75] Sabor KK, Hamou-Lhadj A, Larsson A. Durfex: a feature extraction technique for efficient detection of duplicate bug reports. 2017 IEEE international conference on software quality, reliability and security (QRS); IEEE 2017; 240-250.
[76] Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, et al. Learning to rank using gradient descent. Proceedings of the 22nd international conference on Machine learning; 2005; 89-96.
[77] Taylor M, Zaragoza H, Craswell N, Robertson S, Burges C. Optimisation methods for ranking functions with multiple parameters. Proceedings of the 15th ACM international conference on Information and knowledge management; 2006; 585-593.
[78] เอกสิทธิ์ พัชรวงศ์ศักดา. การวิเคราะห์ข้อมูลด้วยเทคนิคดาต้า ไมน์นิง เบื้องต้น. กรุงเทพฯ: เอเชีย ดิจิตอลการพิมพ์; 2557.
[79] Sensitivity and specificity. [online]. 5 October 2017 [cited 22 November 2017]; https://en.wikipedia.org/wiki/Sensitivity_and_specificity.
[80] Jin K, Dashbalbar A, Yang G, Lee B, Lee J-W. Improving predictions about bug severity by utilizing bugs classified as normal. Contemp Eng Sci 2016; 9[19]: 933-942.
[81] Lamkanfi A, Demeyer S, Soetens QD, Verdonck T. Comparing mining algorithms for predicting the severity of a reported bug. 2011 15th European Conference on

Software Maintenance and Reengineering; IEEE 2011; 249-258.

[82] Otoom AF, Al-Shdaifat D, Hammad M, Abdallah EE. Severity prediction of software bugs. 2016 7th International Conference on Information and Communication Systems (ICICS); IEEE 2016; 92-95.

[83] Yang C-Z, Chen K-Y, Kao W-C, Yang C-C. Improving severity prediction on software bug reports using quality indicators. 2014 IEEE 5th International Conference on Software Engineering and Service Science; IEEE 2014; 216-219.

[84] Yang C-Z, Hou C-C, Kao W-C, Chen X. An empirical study on improving severity prediction of defect reports using feature selection. 2012 19th Asia-Pacific Software Engineering Conference; IEEE 2012; 240-249.

[85] Pandey N, Hudait A, Sanyal DK, Sen A. Automated classification of issue reports from a software issue tracker. *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*: Springer 2018:423-430.

[86] Luaphol B, Srikudkao B, Kachai T, Srikanjanapert N, Polpinij J, Bheganan P. Feature Comparison for Automatic Bug Report Classification. International Conference on Computing and Information Technology; Springer 2019; 69-78.

[87] Luaphol B, Polpinij J, Kaneampornpun M. Automatic Bug Report Severity Prediction by Binary Text Classification Techniques. The Twenty-Fifth International Symposium on Artificial Life and Robotics 2020 (AROB 25th 2020) Japan, January 22-24, 2020; B-Con Plaza, Beppu, Japan. International Society of Artificial Life and Robotics 2020; 206-211.

[88] Libreoffcie bug report. 31 August 2017]; https://bugs.documentfoundation.org/.

[89] da Cunha CEA, Cavalcanti YC, Neto PAdMS, de Almeida ES, de Lemos Meira SR. A Visual Bug Report Analysis and Search Tool. SEKE; 2010; 742-747.

[90] Zhou Y, Tong Y, Gu R, Gall H. Combining text mining and data mining for bug report classification. Journal of Software: Evolution and Process 2016; 28[3]: 150-176.

[91] Bradley P, Bennett K, Demiriz A. Constrained k-means clustering. Microsoft Research, Redmond 2000; 1-8.

[92] Lee J, Kim D, Jung W. Cost-Aware Clustering of Bug Reports by Using a Genetic Algorithm. J Inf Sci Eng 2019; 35[1]: 175-200.

[93] Amati G, Van Rijsbergen CJ. Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Transactions on Information Systems (TOIS) 2002; 20[4]: 357-389.

[94] Mitra M, Singhal A, Buckley C. Improving automatic query expansion. Proceedings of the 21st annual international ACM SIGIR conference on Research and

development in information retrieval; 1998; 206-214.

[95] Voorhees EM. Query expansion using lexical-semantic relations. SIGIR'94; Springer 1994; 61-69.

[96] Xu J, Croft WB. Quary expansion using local and global document analysis. Acm sigir forum; ACM New York, NY, USA 2017; 168-175.

[97] Sun C, Lo D, Wang X, Jiang J, Khoo S-C. A discriminative model approach for accurate duplicate bug report retrieval. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1; ACM 2010; 45-54.

[98] Qin H, Sun X. Classifying Bug Reports into Bugs and Non-bugs Using LSTM. Proceedings of the Tenth Asia-Pacific Symposium on Internetware; ACM 2018; 20.

[99] Terdchanakul P, Hata H, Phannachitta P, Matsumoto K. Bug or Not? Bug Report Classification Using N-Gram IDF. Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on; IEEE 2017; 534-538.

[100] Luaphol B, Srikudkao B, Polpinij J, Kaenampornpan M. Assembling Relevant Bug Report using the Constraint-based k-means Clustering. 2018 International Conference on Information Technology (InCIT); IEEE 2018; 1-6.

[101] Srikudkao B, Luaphol B, Srikanjanapert N, Bheganan P, Polpinij J. Automatic Bug Report Analysis: Actual-bug and Non-bug. The Twenty-Fourth International Symposium on Artificial Life and Robotics 2019; January 23-25, 2019; B-Con Plaza, Beppu, Japan. International Society of Artificial Life and Robotics 2019; 235-240.

พหุน ปณฺ ฑิโต ชีเว

ประวัติผู้เขียน

ชื่อ	นายบัญชา เหลือผล
วันเกิด	วันที่ 16 ธันวาคม พ.ศ. 2521
สถานที่เกิด	อำเภอเมืองกาฬสินธุ์ จังหวัดกาฬสินธุ์
สถานที่อยู่ปัจจุบัน	บ้านเลขที่ 280/2 ถนนอนรรฆนาค อำเภอเมืองกาฬสินธุ์ จังหวัดกาฬสินธุ์ รหัสไปรษณีย์ 46000
ตำแหน่งหน้าที่การงาน	อาจารย์
สถานที่ทำงานปัจจุบัน	สาขาวิชาเทคโนโลยีดิจิทัล คณะบริหารศาสตร์ มหาวิทยาลัยกาฬสินธุ์ เลขที่ 62/1 ถนนเกษตรสมบูรณ์ อำเภอเมืองกาฬสินธุ์ จังหวัดกาฬสินธุ์ รหัสไปรษณีย์ 46000
ประวัติการศึกษา	พ.ศ. 2544 วิทยาศาสตร์บัณฑิต (วท.บ.) สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม พ.ศ. 2549 วิทยาศาสตร์มหาบัณฑิต (วท.ม.) สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยขอนแก่น พ.ศ. 2563 ปรัชญาดุษฎีบัณฑิต (ปร.ด.) สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม