วิธีการจำแนกแบบหลายคลาสสำหรับการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์จากรายงานจุดบกพร่อง

วิทยานิพนธ์

ของ

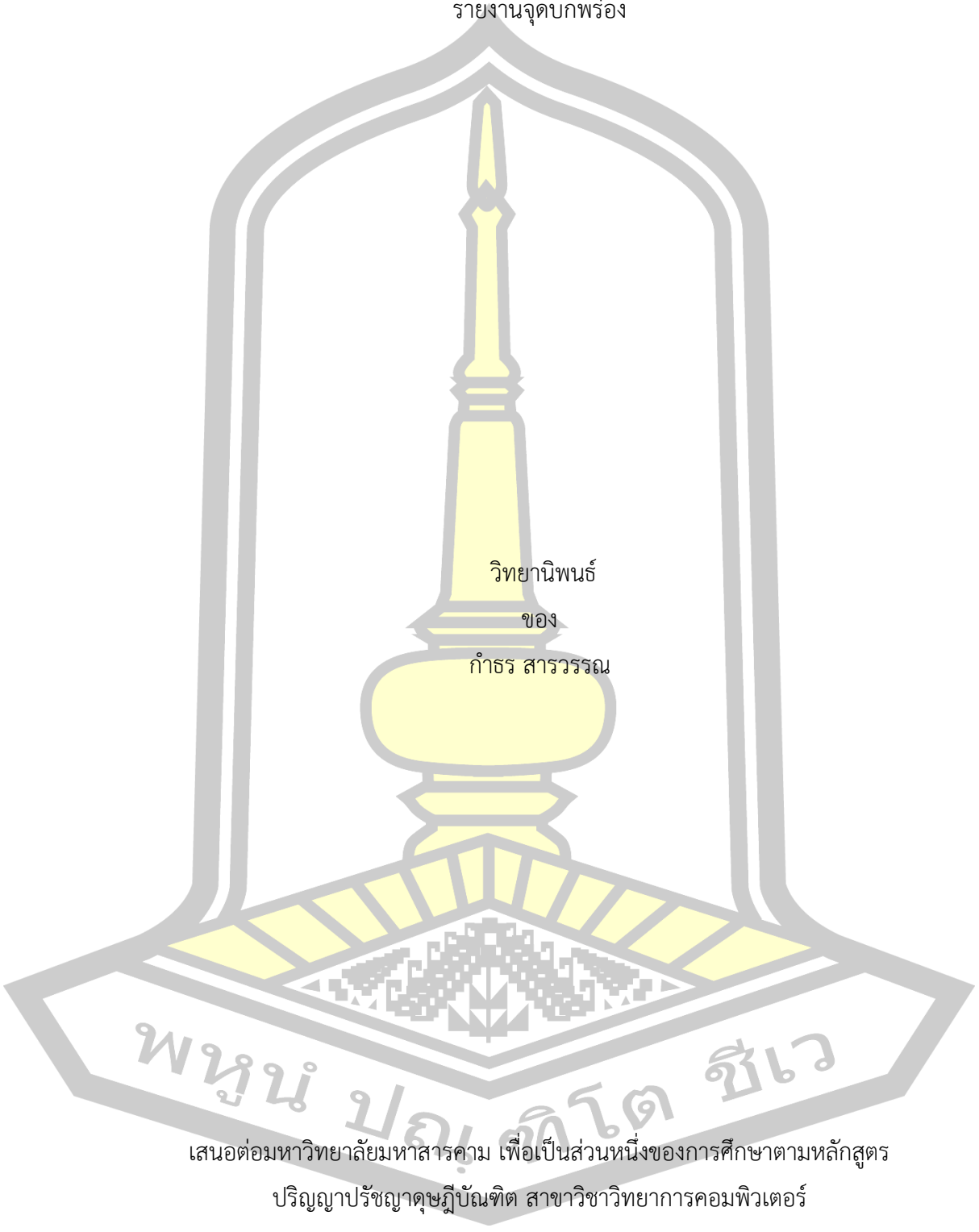
กำธร สารวรรณ

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

พฤษภาคม 2568

วิธีการจำแนกแบบหลายคลาสสำหรับการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์จากรายงานจุดบกพร่อง



วิทยานิพนธ์

ของ

กำธร สารวรรณ

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

พฤษภาคม 2568

Multiclass Classification Approach for Detecting Software Bug Severity Level from Bug Reports

Kamthorn Sarawan

A Thesis Submitted in Partial Fulfillment of Requirements

for Doctor of Philosophy (Computer Science)

May 2025

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของนายกำธร สารวรรณ แล้ว เห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รศ. ดร. ศุภกานต์ พิมลธเรศ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รศ. ดร. จันทิมา พลพินิจ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(รศ. ดร. แกมกาญจน์ สมประเสริฐศรี)

.....กรรมการ
(รศ. ดร. พนิดา ทรงรัมย์)

.....กรรมการผู้ทรงคุณวุฒิภายนอก
(ผศ. ดร. มนัสวี แก่นอำพรพันธ์)

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา ปรัชญาดุษฎีบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....
(รศ. ดร. จันทิมา พลพินิจ)
คณบดีคณะวิทยาการสารสนเทศ

.....
(ผศ. ดร. พลเดช เชาวรัตน์)
คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง วิธีการจำแนกแบบหลายคลาสสำหรับการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์จากรายงานจุดบกพร่อง

ผู้วิจัย กำธร สารวรรณ

อาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร. จันทิมา พลพินิจ
รองศาสตราจารย์ ดร. แกมกาญจน์ สมประเสริฐศรี

ปริญญา ปรัชญาดุษฎีบัณฑิต **สาขาวิชา** วิทยาการคอมพิวเตอร์

มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2568

บทคัดย่อ

ในปัจจุบัน การตรวจจับและวิเคราะห์รายงานจุดบกพร่องของซอฟต์แวร์เป็นกระบวนการสำคัญที่ช่วยเพิ่มประสิทธิภาพในการแก้ไขปัญหาและพัฒนาซอฟต์แวร์ อย่างไรก็ตาม ปริมาณรายงานจุดบกพร่องที่เพิ่มขึ้นในระบบติดตามจุดบกพร่อง ส่งผลให้เกิดความท้าทายในการจำแนกระดับความรุนแรงของจุดบกพร่องอย่างแม่นยำ งานวิจัยนี้จึงนำเสนอวิธีการจำแนกแบบหลายคลาสสำหรับการตรวจจับระดับความรุนแรงของจุดบกพร่องโดยอัตโนมัติ โดยใช้เทคนิคการเรียนรู้ของเครื่องและการเรียนรู้เชิงลึก โมเดลที่ใช้ในการทดลองประกอบด้วย Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Ensemble Stacking และโมเดล Transformer ได้แก่ BERT เพื่อปรับปรุงความแม่นยำของการจำแนกระดับความรุนแรง งานวิจัยนี้ได้นำเสนอวิธีการเสริมข้อมูลโดยใช้เทคนิค T5 Summarization เพื่อแก้ไขปัญหาความไม่สมดุลของข้อมูล และเปรียบเทียบกับวิธีการอื่น เช่น SMOTE, การปรับค่า Class Weight และการแทนที่คำพ้องความหมาย ทั้งนี้ ได้ทำการทดลองโดยใช้ชุดข้อมูลรายงานจุดบกพร่องจาก Mozilla โดยแบ่งข้อมูลเป็นสัดส่วน 80:20 สำหรับการฝึกและทดสอบโมเดล ผลการทดลองแสดงให้เห็นว่าโมเดล BERT2 ซึ่งได้รับการปรับแต่งพารามิเตอร์และใช้เทคนิคการเสริมข้อมูลจาก T5 Summarization ให้ผลลัพธ์ที่ดีที่สุด โดยมีค่า F1-score 64.25% และ Accuracy 65.20% ซึ่งสูงกว่าโมเดลอื่น ๆ ที่ทดสอบ งานวิจัยนี้ชี้ให้เห็นว่าการใช้โมเดล Transformer ควบคู่กับเทคนิคการเสริมข้อมูลสามารถช่วยเพิ่มประสิทธิภาพในการจำแนกระดับความรุนแรงของจุดบกพร่องได้อย่างมีนัยสำคัญ

คำสำคัญ : การจำแนกแบบหลายคลาส, ระดับความรุนแรงของจุดบกพร่อง, การเรียนรู้ของเครื่อง, การเรียนรู้เชิงลึก, T5 Summarization

TITLE Multiclass Classification Approach for Detecting Software Bug Severity Level from Bug Reports

AUTHOR Kamthorn Sarawan

ADVISORS Associate Professor Jantima Polpinij , Ph.D.
Associate Professor Gamgarn Sompasertsri , Ph.D.

DEGREE Doctor of Philosophy **MAJOR** Computer Science

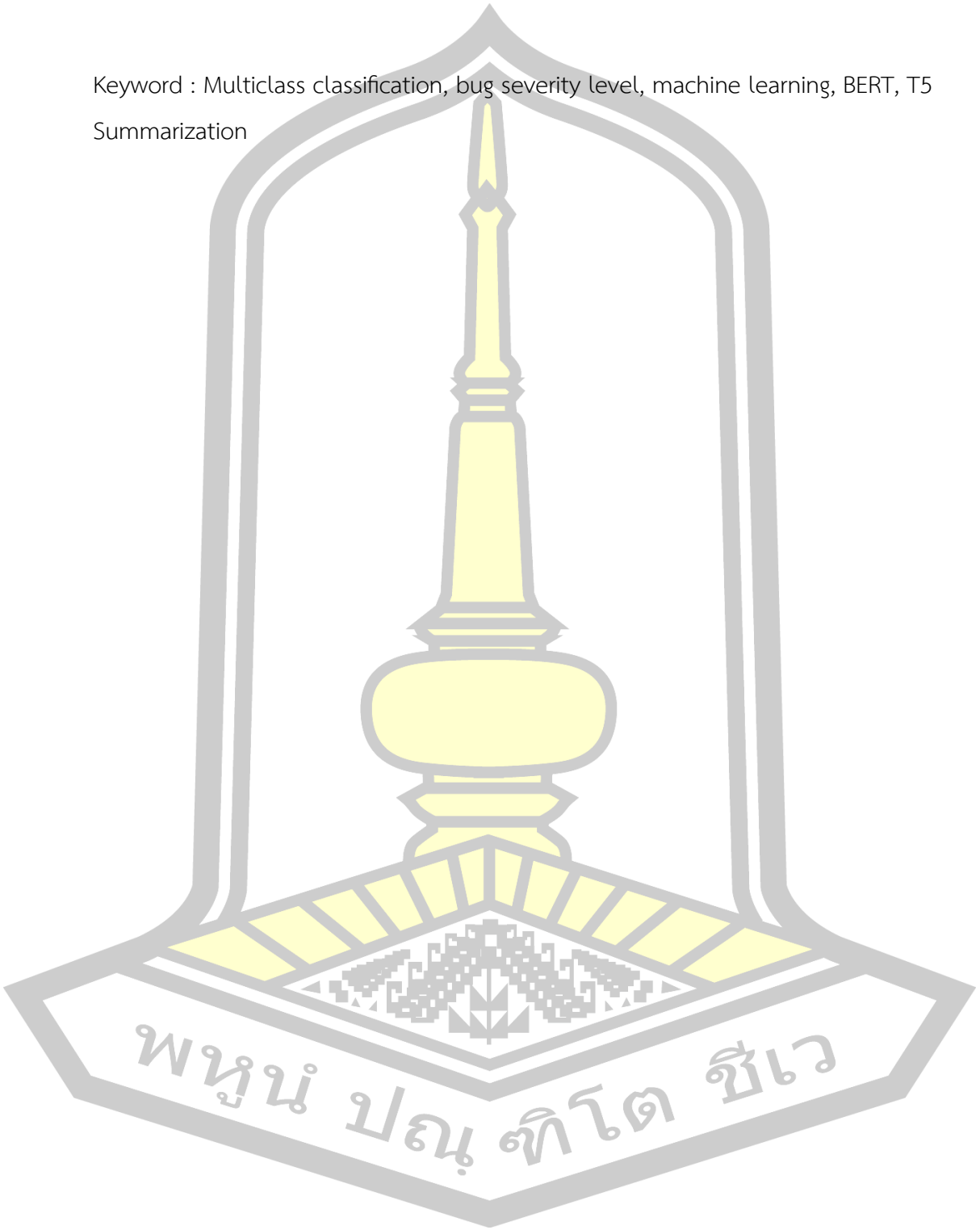
UNIVERSITY Mahasarakham University **YEAR** 2025

ABSTRACT

The detection and analysis of software bug reports play a critical role in enhancing problem-solving efficiency and improving software quality. However, the growing volume of bug reports in bug tracking systems presents substantial challenges in accurately classifying bug severity levels. This study proposes a multiclass classification approach for the automated detection of software bug severity levels, leveraging both machine learning and deep learning techniques. The models employed in the experiments include Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), Ensemble Stacking, and Transformer-based models, specifically BERT. To improve the accuracy of severity classification, this research introduces a data augmentation strategy incorporating the T5 Summarization technique to address data imbalance issues. Furthermore, the study compares the effectiveness of this approach against other established methods, including Synthetic Minority Over-sampling Technique (SMOTE), Class Weight adjustments, and synonym replacement. The experiments were conducted using the Bugzilla bug report dataset, which was partitioned into training and testing sets in an 80:20 ratio. The experimental findings indicate that the BERT2 model, which was fine-tuned and augmented with T5 Summarization-based data augmentation, exhibited the highest performance, achieving an F1-score of 64.25% and an accuracy of 65.20%, surpassing the other tested models. The results of this study suggest that integrating Transformer-based models with data augmentation techniques can substantially

enhance the effectiveness of software bug severity classification.

Keyword : Multiclass classification, bug severity level, machine learning, BERT, T5 Summarization



กิตติกรรมประกาศ

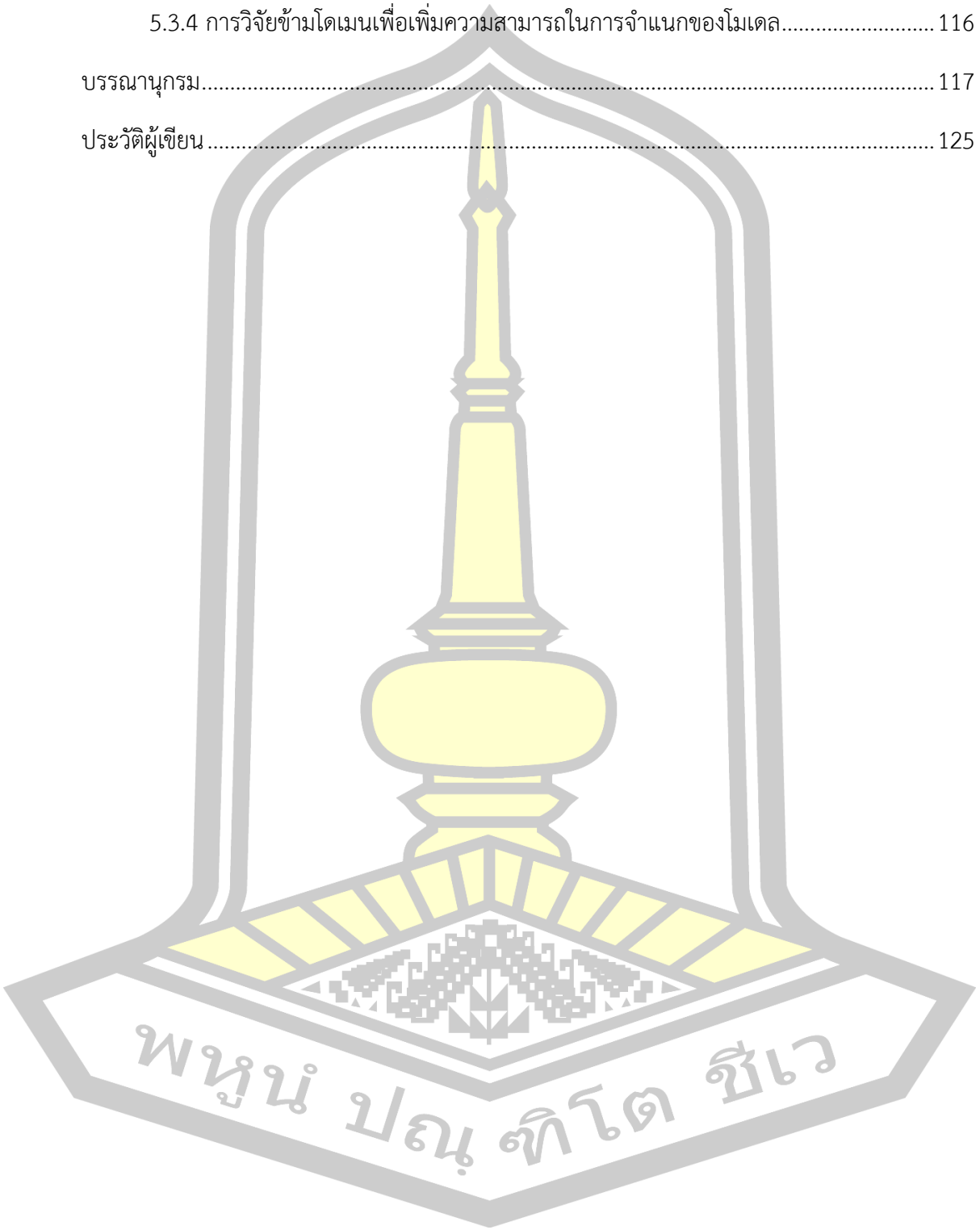
ข้าพเจ้าขอแสดงความขอบคุณอย่างสูงสุดต่ออาจารย์ที่ปรึกษาหลัก รองศาสตราจารย์ ดร. จันทิมา พลพินิจ และอาจารย์ที่ปรึกษาร่วม รองศาสตราจารย์ ดร. แกมกาญจน์ สมประเสริฐศรี สำหรับคำแนะนำที่มีคุณค่า ความอดทน และการสนับสนุนอย่างต่อเนื่องตลอดระยะเวลาการทำวิจัยและเขียนวิทยานิพนธ์ฉบับนี้ ความรู้และประสบการณ์ที่ท่านแบ่งปันเป็นปัจจัยสำคัญที่ทำให้ข้าพเจ้าสามารถดำเนินงานวิจัยจนสำเร็จลุล่วง ข้าพเจ้าขอขอบคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่านที่ได้ให้ข้อเสนอแนะอันเป็นประโยชน์ซึ่งช่วยพัฒนาคุณภาพของงานวิจัยนี้ให้ดียิ่งขึ้น รวมถึงคณาจารย์ทุกท่านในคณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม ที่ได้ให้ความรู้และคำแนะนำในระหว่างการศึกษา ข้าพเจ้าขอขอบคุณเพื่อนนักวิจัยและเพื่อนร่วมรุ่นทุกคนสำหรับการแลกเปลี่ยนความคิดเห็น ความร่วมมือ และการให้กำลังใจซึ่งเป็นแรงผลักดันสำคัญให้ข้าพเจ้าสามารถก้าวข้ามอุปสรรคต่าง ๆ ในระหว่างการทำวิจัย

นอกจากนี้ ข้าพเจ้าขอขอบคุณครอบครัวและคนใกล้ชิดที่ให้การสนับสนุนทั้งทางร่างกายและจิตใจมาตลอด ขอบคุณสำหรับความเข้าใจ ความอดทน และกำลังใจที่ทำให้ข้าพเจ้ามีแรงบันดาลใจในการศึกษาต่อและทำงานวิจัยให้สำเร็จ สุดท้าย ข้าพเจ้าขอขอบคุณทุกหน่วยงานและทุกแหล่งข้อมูลที่สนับสนุนชุดข้อมูลและเครื่องมือที่ใช้ในการศึกษานี้ ซึ่งเป็นองค์ประกอบสำคัญที่ทำให้งานวิจัยนี้เกิดขึ้นได้

กำธร สารวรรณ

สารบัญ

หน้า



สารบัญตาราง

หน้า

สารบัญรูป

หน้า

บทที่ 1

บทนำ

1.1 หลักการและเหตุผล

ซอฟต์แวร์ที่มีการพัฒนาอย่างต่อเนื่อง ขนาดของซอฟต์แวร์ที่ใหญ่ขึ้นและมีความซับซ้อนมากขึ้น ย่อมหลีกลี่ยงไม่ได้ที่จะทำให้การทำงานของซอฟต์แวร์มีจุดบกพร่องได้ ซึ่งจุดบกพร่องของซอฟต์แวร์ (Software Bug หรือ Software Defect) [1, 2] คือ เหตุที่ทำให้ซอฟต์แวร์เกิดปัญหา ไม่สามารถทำงานได้ถูกต้องตามวัตถุประสงค์ที่ถูกออกแบบไว้ หรือการทำงานที่นอกเหนือจากการออกแบบ ส่งผลให้เกิดความผิดพลาดและความเสียหายที่เกิดจากการทำงานของซอฟต์แวร์นั้น รวมทั้งทำให้ซอฟต์แวร์ไม่สามารถทำงานได้เต็มประสิทธิภาพเท่าที่ควร เมื่อมีจุดบกพร่องของซอฟต์แวร์เกิดขึ้นแล้ว สิ่งที่ต้องดำเนินการต่อไปก็คือ การแก้ไขปัญหาของจุดบกพร่อง เพื่อให้ซอฟต์แวร์สามารถทำงานได้ถูกต้อง สำหรับอัตราส่วนต้นทุนโครงการซอฟต์แวร์หนึ่ง ๆ นั้น จะมีสัดส่วนของการบำรุงรักษาซอฟต์แวร์อยู่ที่ 20% - 80% ของต้นทุนทั้งหมด และการบำรุงรักษาเชิงแก้ไขจุดบกพร่องคิดเป็น 20% [3, 4] ของการบำรุงรักษาทั้งหมด หากเป็นซอฟต์แวร์ขนาดเล็ก การรวบรวมรายงานจุดบกพร่องสามารถกระทำได้โดยง่าย แต่หากเป็นซอฟต์แวร์ขนาดใหญ่ที่มีการใช้งานแพร่หลาย เช่น ซอฟต์แวร์โอเพนซอร์ส (Open source) การรวบรวมรายงานจุดบกพร่องอาจจะไม่ใช่เรื่องง่าย ดังนั้นจึงมีหน่วยงานหรือองค์กรที่เกี่ยวข้องกับการพัฒนาซอฟต์แวร์โอเพนซอร์ส รวมทั้งบริษัทเอกชนได้พัฒนา “ระบบติดตามจุดบกพร่อง (Bug Tracking System: BTS)” [2]

เมื่อจำนวนรายงานจุดบกพร่องที่รายงานเข้ามาในระบบ BTS มีจำนวนมากขึ้น ทำให้ยากต่อการคัดกรองและการวิเคราะห์ ดังนั้นการวิเคราะห์รายงานจุดบกพร่องแบบอัตโนมัติจึงได้รับความสนใจในการศึกษา และการศึกษาที่ได้รับความสนใจในการศึกษามากที่สุดและยังมีการศึกษาอย่างต่อเนื่องมาจนถึงปัจจุบันคือการวิเคราะห์รายงานจุดบกพร่องเพื่อแยกเป็นรายงานจุดบกพร่องที่แท้จริง (Real-bug Report) และรายงานที่ไม่ใช่รายงานจุดบกพร่อง (Non-bug Report) เพราะข้อมูลในรายงานจุดบกพร่องที่แท้จริงนั้นจะช่วยให้การแก้ไขจุดบกพร่องของซอฟต์แวร์สามารถดำเนินการได้รวดเร็วมากขึ้น [5-8] อย่างไรก็ตาม การตรวจหารายงานจุดบกพร่องที่แท้จริงเพื่อใช้ข้อมูลจากรายงานเหล่านี้ในการแก้ไขจุดบกพร่องในซอฟต์แวร์เพียงอย่างเดียว อาจจะไม่ได้ช่วยทำให้

การแก้ไขจุดบกพร่องของซอฟต์แวร์สามารถดำเนินการได้รวดเร็วอย่างแท้จริง เพราะจริง ๆ แล้วแต่ละจุดบกพร่องจะมีระดับความรุนแรงของจุดบกพร่อง (Bug Severity Level) [9-15] ที่ส่งผลต่อการทำงานของซอฟต์แวร์แตกต่างกัน การไม่ทราบระดับความรุนแรงของจุดบกพร่องของซอฟต์แวร์โดยทันที หรือการใช้เวลาในการประเมินระดับความรุนแรงที่นานเกินไป อาจจะมีความล่าช้าในการแก้ไขและส่งผลเสียหายต่อผู้ใช้งานซอฟต์แวร์ ดังนั้น การทราบระดับความรุนแรงของจุดบกพร่องจะช่วยให้การกำหนดนักพัฒนาซอฟต์แวร์ในการแก้ปัญหาจุดบกพร่องนั้นได้อย่างเหมาะสมมากขึ้น และช่วยให้ทราบว่าจุดบกพร่องใดในซอฟต์แวร์ควรได้รับการแก้ปัญหาก่อน

ระดับความรุนแรงของจุดบกพร่องจะเป็นข้อมูลประกอบในการจัดลำดับความสำคัญของรายงานจุดบกพร่อง (Bug Priority Level) [16, 17] โดยทั่วไประดับความรุนแรงของจุดบกพร่อง จะมี 5 ระดับ ได้แก่ Blocker, Critical, Major, Minor, และ Trivial เรียงตามลำดับความรุนแรง จากการศึกษาที่ผ่านมา พบว่างานวิจัยที่ผ่านมามักจะทำการศึกษาเรื่องระดับความรุนแรงแบบสองระดับ คือ ระดับรุนแรง (Severity) และระดับไม่รุนแรง (Non-severity) สาเหตุก็เนื่องมาจากข้อมูลรายงานจุดบกพร่องจะมีความไม่สมดุล (Imbalance) [18-20] โดยเฉพาะในคลาสที่เป็น Blocker และ Critical ดังนั้น ในการศึกษาเกี่ยวกับการวิเคราะห์ระดับความรุนแรงของรายงานจุดบกพร่องแบบหลายกลุ่ม (Multiclassification) นั้น จึงมักแก้ปัญหาด้วยการทำ Resampling แบบ Under-sampling กับชุดข้อมูล ปัญหาที่ตามคือ หากข้อมูลในการสร้างโมเดลเพื่อการวิเคราะห์ความรุนแรงของรายงานจุดบกพร่องที่ไม่มากพอ ก็อาจจะทำให้โมเดลเพื่อการวิเคราะห์ความรุนแรงของรายงานจุดบกพร่องที่สร้างขึ้นมามีประสิทธิภาพไม่ดีพอ

จากปัญหาข้างต้น แนวทางที่เริ่มมีการศึกษาคือการนำข้อมูลรายงานจุดบกพร่องจากหลาย ๆ โอเพนซอร์สมารวมกัน เพื่อเพิ่มขนาดของข้อมูลให้มากขึ้น อย่างไรก็ตาม การนำรายงานจุดบกพร่องจากหลายๆ โอเพนซอร์สก็มีปัญหาหลักๆ คือ โครงสร้างของรายงานจุดบกพร่องอาจจะมีความแตกต่างกัน และแต่ละโอเพนซอร์สอาจจะมีโครงสร้างของซอฟต์แวร์และฟังก์ชันในการทำงานที่แตกต่างกัน ซึ่งจะส่งผลให้รูปแบบการรายงานจุดบกพร่องอาจจะแตกต่างกัน รวมทั้งการกำหนดระดับความรุนแรงของจุดบกพร่องในแต่ละโอเพนซอร์สอาจจะมีความแตกต่าง

ซึ่งสิ่งที่กล่าวมานี้ จึงทำให้เป็นประเด็นที่น่าสนใจในการศึกษาการวิเคราะห์ระดับความรุนแรงของรายงานจุดบกพร่องแบบหลายกลุ่ม เพื่อทำให้เกิดประโยชน์ต่อการแก้ไขจุดบกพร่องของซอฟต์แวร์ได้อย่างเหมาะสมมากที่สุด ตลอดจนข้อมูลนี้สามารถช่วยให้เลือกนักพัฒนาซอฟต์แวร์ที่

เหมาะสมต่อการแก้ปัญหาจุดบกพร่องของซอฟต์แวร์ แน่นอนว่าสิ่งเหล่านี้จะนำไปสู่การแก้ไขจุดบกพร่องของซอฟต์แวร์ที่เร็วมากขึ้น

1.2 ปัญหาการวิจัย

ทำอย่างไรจึงจะสามารถตรวจจับระดับความรุนแรงแบบหลายกลุ่มของจุดบกพร่องซอฟต์แวร์แบบอัตโนมัติ เพื่อเป็นแนวทางในการลดเวลาขั้นตอนการตรวจสอบระดับความรุนแรง

1.3 วัตถุประสงค์ของการวิจัย

1. เพื่อวิจัยและพัฒนากระบวนการสำหรับการตรวจจับระดับความรุนแรงแบบหลายกลุ่มของจุดบกพร่องซอฟต์แวร์แบบอัตโนมัติ ด้วยเทคนิคด้านการประมวลผลภาษาธรรมชาติและการจำแนกเอกสารข้อความ

2. เพื่อศึกษาและการสร้างโมเดลแบบทรานฟอเมอร์สำหรับการตรวจจับระดับความรุนแรงของรายงานจุดบกพร่องแบบหลายกลุ่มแบบอัตโนมัติได้อย่างแม่นยำ

1.4 ความสำคัญของการวิจัย

1. เป็นการประยุกต์เทคนิคด้านการประมวลผลภาษาธรรมชาติและเหมืองข้อความ ในการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์แบบอัตโนมัติ เพื่อนำไปสู่การจัดลำดับความสำคัญในการแก้ไขจุดบกพร่อง และช่วยลดระยะเวลาในการแก้ไขจุดบกพร่อง

2. การตรวจจับระดับความรุนแรงหลายระดับของจุดบกพร่องซอฟต์แวร์แบบอัตโนมัติ ด้วยการรวมชุดรายงานจุดบกพร่องของซอฟต์แวร์ที่แตกต่างกันจากหลายแหล่ง น่าจะส่งผลให้กระบวนการที่นำเสนอสามารถนำไปใช้ตรวจจับระดับความรุนแรงกับรายงานจุดบกพร่องของซอฟต์แวร์อื่นได้

3. เป็นการเพิ่มโอกาสของความถูกต้องในการกำหนดระดับความรุนแรง และโอกาสของการลดค่าใช้จ่ายในการแก้ไขจุดบกพร่อง

1.5 ขอบเขตของการวิจัย

1. นำเสนอกระบวนการต้นแบบในการตรวจจับระดับความรุนแรงแบบหลายคลาสของจุดบกพร่องซอฟต์แวร์แบบอัตโนมัติ

2. ชุดข้อมูล

สำหรับงานวิจัยนี้จะใช้ชุดข้อมูลในการศึกษา 3 ชุดข้อมูล ดังนี้

2.1 ชุดข้อมูลรายงานจุดบกพร่องจากการรวบรวมจากการศึกษา [21] ประกอบด้วยจุดบกพร่องซอฟต์แวร์ของ Mozilla จำนวน 5,456 รายงาน

2.2 ชุดข้อมูลรายงานจุดบกพร่องจาก Mozilla, Eclipse, NetBeans และ GCC จำนวน 154,254 รายงาน

2.3 ชุดข้อมูลรายงานจุดบกพร่องดาวน์โหลดจากระบบติดตามจุดบกพร่องของโครงการ Apache จำนวน 2,263 รายงาน

3. กระบวนการที่นำเสนอจะอยู่บนพื้นฐานการประมวลผลภาษาธรรมชาติ (Natural language processing: NLP)

4. การศึกษานี้ได้นำหลักการเสริมข้อมูล (Data Augmentation) มาประยุกต์ใช้ เพื่อทำการแก้ไขปัญหาข้อมูลรายงานไม่สมดุล

5. การประเมินการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์ จะประเมินด้วยค่าความถูกต้อง (Accuracy), ความระลึก (Recall หรือ True positive rate), ค่าความแม่นยำ (Precision), ค่า F1

1.6 นิยามศัพท์เฉพาะ

1. จุดบกพร่อง (Bug) คือ ข้อบกพร่องที่ส่งผลให้ซอฟต์แวร์เกิดปัญหา

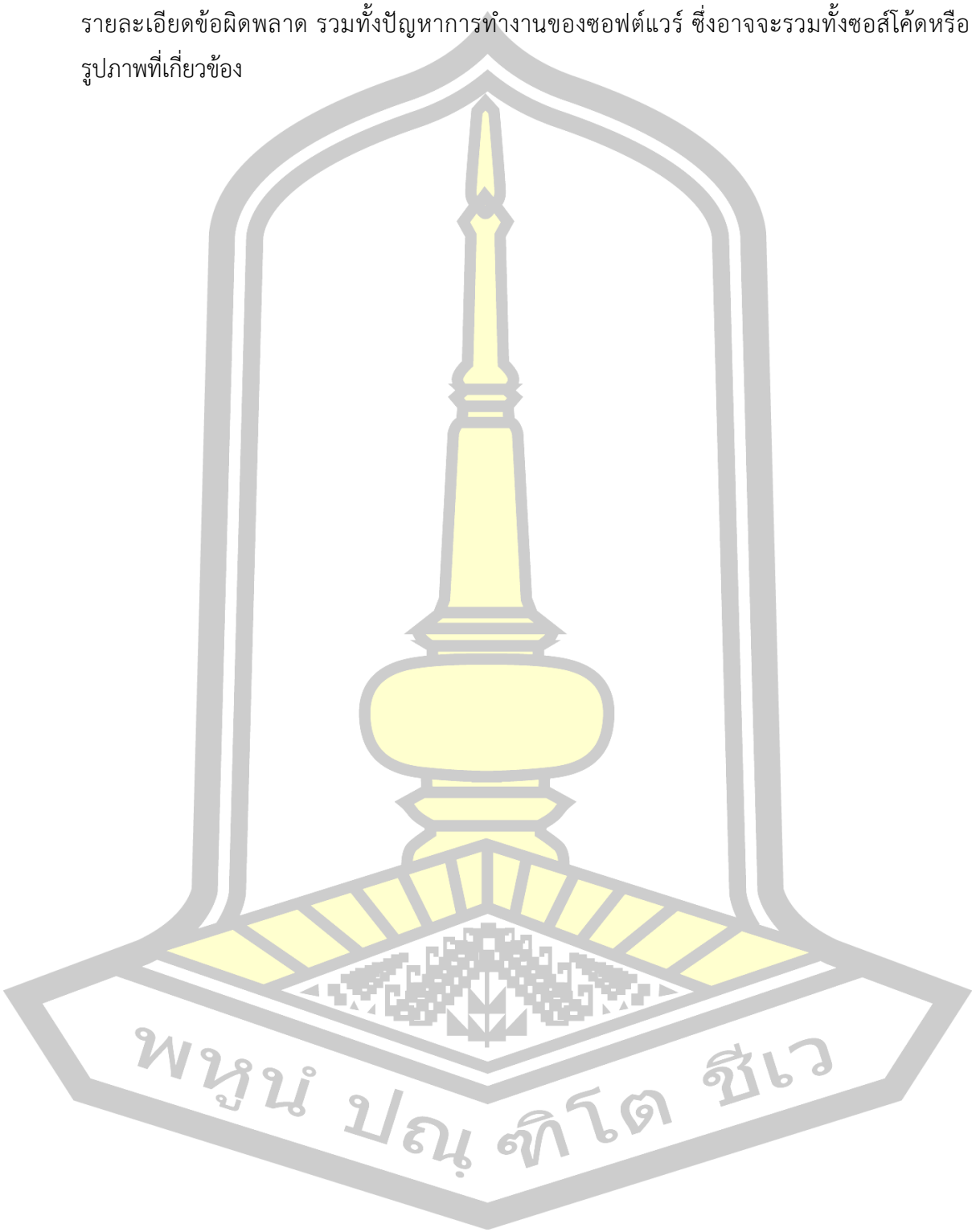
2. รายงานจุดบกพร่อง (Bug report) คือ รายงานที่ประกอบด้วยรายละเอียดต่าง ๆ ของจุดบกพร่องในซอฟต์แวร์ ที่ถูกพบโดยผู้ใช้งานและถูกรายงานให้ทีมพัฒนาหรือชุมชนผู้ใช้งานทราบเพื่อทำการแก้ไข

3. ระบบติดตามจุดบกพร่อง (Bug tracking system: BTS) คือ ระบบติดตามการแก้ไขจุดบกพร่องของซอฟต์แวร์

4. ความรุนแรงของจุดบกพร่อง (Bug Severity) คือ ความรุนแรงของจุดบกพร่องซอฟต์แวร์ที่ส่งผลกระทบต่อการทำงานของซอฟต์แวร์ ยิ่งส่งผลกระทบต่อประสิทธิภาพการทำงานโดยรวมของซอฟต์แวร์มาก ยิ่งมีค่าระดับความรุนแรงที่มาก

5. สรุปจุดบกพร่อง (Bug Summary) คือ ข้อมูลที่เป็นความสั้น ๆ ในรายงานจุดบกพร่องที่อธิบายอย่างกระชับเกี่ยวกับปัญหาของจุดบกพร่อง

6. รายละเอียดจุดบกพร่อง (Bug Description) คือ ข้อมูลที่เขียนอธิบายเพิ่มเติมถึงรายละเอียดข้อผิดพลาด รวมทั้งปัญหาการทำงานของซอฟต์แวร์ ซึ่งอาจจะรวมทั้งซอส์โค้ดหรือรูปภาพที่เกี่ยวข้อง



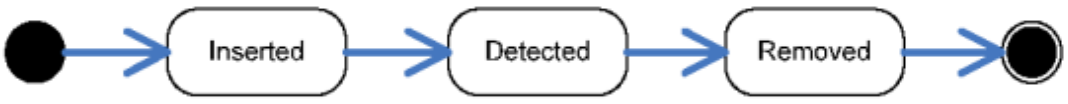
บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ จะเป็นการทบทวนทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการวิจัยรายงานจุดบกพร่องซอฟต์แวร์ โดยมีรายละเอียดดังต่อไปนี้

2.1 รายงานจุดบกพร่อง

จุดบกพร่องของซอฟต์แวร์ (Software Bug หรือ Defect) [1, 2, 22] คือ เหตุที่ทำให้ซอฟต์แวร์นั้น ๆ เกิดปัญหา ไม่สามารถทำงานได้ถูกต้องตามวัตถุประสงค์ที่ถูกออกแบบไว้ หรือการทำงานที่นอกเหนือจากการออกแบบ ส่งผลให้เกิดความผิดพลาดและความเสียหายที่เกิดจากการทำงานของซอฟต์แวร์นั้น รวมทั้งทำให้ซอฟต์แวร์ไม่สามารถทำงานได้เต็มประสิทธิภาพเท่าที่ควร ซึ่งโดยทั่วไปแล้วจุดบกพร่องของซอฟต์แวร์จะแบ่งเป็น 2 รูปแบบ คือ จุดบกพร่องที่บ่งชี้ไปยังฟังก์ชันการทำงานหลักของซอฟต์แวร์นั้น ๆ (Function requirement) เช่น การเรียกข้อมูลไม่ถูกต้อง การทำให้ข้อมูลสูญหาย เป็นต้น และรูปแบบที่สอง คือ จุดบกพร่องที่ไม่ได้บ่งชี้ไปยังการทำงานหลักของซอฟต์แวร์ (Non-functional requirement) แต่เป็นจุดบกพร่องที่บอกถึงปัญหาอื่น ๆ หากได้รับการแก้ไขจะช่วยให้โปรแกรมทำงานได้อย่างมีประสิทธิภาพมากขึ้น ทำงานได้ง่าย หรือ สะดวกมากขึ้น เป็นต้น เมื่อมีจุดบกพร่องของซอฟต์แวร์เกิดขึ้นแล้ว สิ่งที่ต้องดำเนินการต่อไปก็คือ การแก้ไขปัญหาของจุดบกพร่อง เพื่อให้ซอฟต์แวร์สามารถทำงานได้ถูกต้อง ซึ่งกระบวนการค้นหาและแก้ไขจุดบกพร่องนี้ จะเรียกว่าการดีบั๊ก (Debug)



รูปที่ 2.1 Defect life cycle

เอกสาร IEEE 1044-2009 [22] ได้แสดงวงจรชีวิตของจุดบกพร่องไว้ 3 สถานะ คือ การเกิดขึ้นของจุดบกพร่อง (Inserted) ถูกตรวจพบว่าเป็นจุดบกพร่อง (Detected) และจุดบกพร่องถูกทำลายหรือแก้ไข (Removed) ซึ่งการที่จะระบุว่าซอฟต์แวร์มีจุดบกพร่องได้หรือไม่นั้นจะสามารถบอกได้เมื่อมีการตรวจพบ ทั้งนี้ในเอกสารนี้ไม่ได้กล่าวถึงวิธีการและการตรวจสอบจุดบกพร่อง รวมทั้งไม่ได้กล่างถึงกระบวนการพิจารณาเกี่ยวกับการแก้ไขหรือลบจุดบกพร่องด้วย อย่างไรก็ตามเมื่อ

จุดบกพร่องถูกตรวจสอบพบสถานะของวงจรชีวิตจะมีสถานะรายละเอียดเพิ่มเติม ที่เกี่ยวกับขั้นตอนของการดำเนินการกับจุดบกพร่อง ซึ่งจะกล่าวในหัวข้อต่อไป

2.1.1 รายงานจุดบกพร่องของซอฟต์แวร์

รายงานจุดบกพร่องของซอฟต์แวร์ (Bug report) [1, 8, 21, 23] คือ รายงานที่แสดงรายละเอียดของจุดบกพร่องของซอฟต์แวร์ ซึ่งผู้ที่รายงานจุดบกพร่องอาจจะเป็น ผู้พัฒนาซอฟต์แวร์ ผู้ทดสอบ หรือผู้ใช้งาน ที่ทำหน้าที่ในการรายงาน โดยที่รายงานจุดบกพร่องนี้ยังมีชื่อเรียกอื่น ๆ อีก เช่น รายงานข้อบกพร่อง (Defect report) รายงานปัญหา (Issue หรือ Problem หรือ Trouble) รายงานความผิดพลาด (Fault report) รายงานความล้มเหลว (Failure report) เป็นต้น

รูปแบบของรายงานจุดบกพร่องของซอฟต์แวร์นั้นโดยหลัก ๆ แล้วจะมีรายละเอียดของข้อมูลที่จำเป็นต้องรายงานคล้ายกัน ทั้งนี้ขึ้นอยู่กับเครื่องมือที่ใช้ในการรายงาน แต่ตามมาตรฐานของ International Software Testing Qualifications Board (ISTQB) [24] ซึ่งอยู่ในหมวดของรูปแบบการรายงานเหตุการณ์ (Incident Report) จะประกอบไปด้วย 4 ประเด็น ดังตารางที่ 2.1

ตารางที่ 2.1 Test incident report outline

หัวข้อ	**รายละเอียด**
Test incident report identifier	ID ของรายงาน
Summary	ข้อมูลรายงายโดยสรุปเหตุการณ์ ที่ระบุรายละเอียดของการทดสอบ ความคาดหวังของการทำงานโปรแกรม ข้อผิดพลาดที่เกิดขึ้นของเหตุการณ์ที่เกิดจาการใช้งานหรือการทดสอบ
Incident description	คำอธิบายเพิ่มเติมโดยละเอียด ได้แก่ - ข้อมูลนำเข้า - ผลลัพธ์ที่คาดหวัง - ผลลัพธ์ที่แท้จริง - ความผิดปกติ - วันที่และเวลาทดสอบ - ขั้นตอนในการทดสอบ - สภาพแวดล้อม เช่น ระบบปฏิบัติการ หมายเลขเวอร์ชั่นของซอฟต์แวร์

หัวข้อ	รายละเอียด
	- ความคิดเห็นของผู้ทดสอบ - ความคิดเห็นของผู้สังเกตการณ์ และควรรวมข้อมูลอื่น ๆ เกี่ยวกับเหตุการณ์ที่ทำการทดสอบ รวมถึงแนวทางการแก้ไขที่เป็นไปได้
Impact	สิ่งที่จะส่งผลกระทบ

รายงานจุดบกพร่องของซอฟต์แวร์จะถูกรวบรวมเพื่อนำไปสู่การแก้ไขจุดบกพร่องนั้นต่อไป ซึ่งการรวบรวมรายงานจุดบกพร่องของซอฟต์แวร์ขนาดเล็ก หรือซอฟต์แวร์ที่พัฒนาขึ้นมาใช้งานภายในองค์กรสามารถทำได้ง่าย แต่หากเป็นซอฟต์แวร์ขนาดใหญ่หรือซอฟต์แวร์เชิงพาณิชย์ (Commercial software) รวมถึงซอฟต์แวร์โอเพนซอร์ส (Open source) จะทำได้ยากกว่าเพราะมีผู้ใช้งานจำนวนมาก จึงจำเป็นต้องมีเครื่องมือที่ใช้ในการเก็บรวบรวม

2.1.2 คลังเก็บรายงานจุดบกพร่อง

คลังเก็บรายงานจุดบกพร่อง (Bug report repository) [21, 25] คือ แหล่งเก็บรวบรวมรายงานจุดบกพร่องของซอฟต์แวร์ โดยวัตถุประสงค์ของคลังเก็บรายงานจุดบกพร่องมีไว้เพื่อรวบรวมรายงานและโดยส่วนมากจะเป็นระบบที่ผู้ใช้งาน ผู้พัฒนา สามารถเข้ามาแสดงความคิดเห็นเกี่ยวกับรายงานจุดบกพร่อง เป็นลักษณะชุมชนออนไลน์ และการแสดงความคิดเห็นจะเป็นลักษณะของการให้ข้อคิดเห็นเกี่ยวกับการแก้ไขปัญหาจุดบกพร่อง

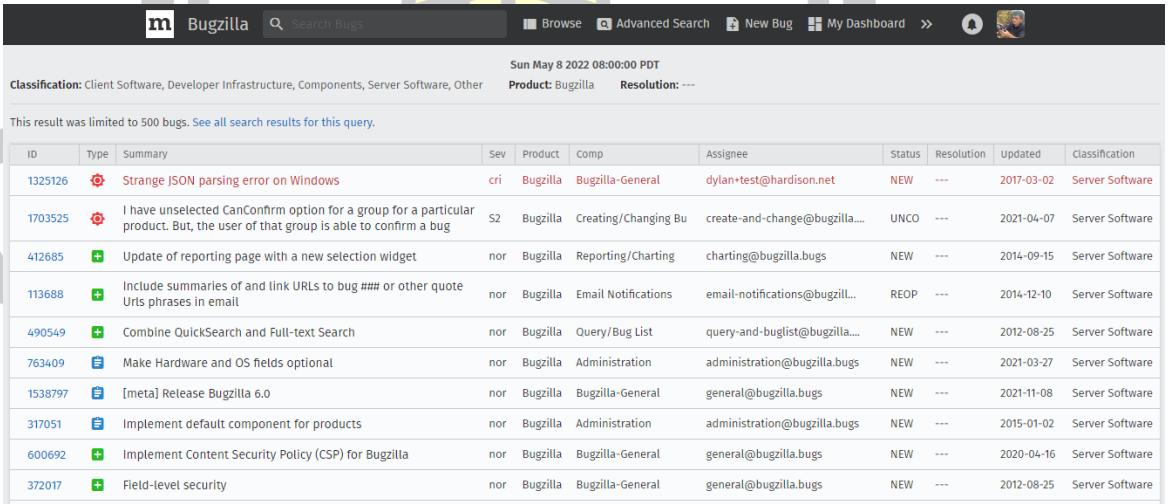
2.1.3 การตรวจสอบรายงานจุดบกพร่อง

การตรวจสอบรายงานจุดบกพร่อง (Bug report triage) [4, 16] คือ กระบวนการกรั่นกรองรายงานจุดบกพร่องที่ถูกรายงานเข้ามาในระบบ โดยการกรั่นกรองจะทำหน้าที่ด้วยบุคคลที่ได้รับมอบหมายจากผู้จัดการโครงการซอฟต์แวร์ (Software project manager) ที่เรียกว่า ผู้ตรวจสอบรายงานจุดบกพร่อง (Bug triager) โดยผู้ตรวจสอบจะทำหน้าที่ในการตรวจสอบกรั่นกรองว่ารายงานเป็นจุดบกพร่องจริงหรือไม่ จุดบกพร่องซ้ำซ้อนกับจุดบกพร่องเดิมในระบบหรือไม่ รวมทั้งการกำหนดระดับความรุนแรงและลำดับความสำคัญ และทำการกำหนดนักพัฒนาซอฟต์แวร์เพื่อแก้ไขจุดบกพร่องต่อไป

2.1.4 ระบบติดตามจุดบกพร่อง

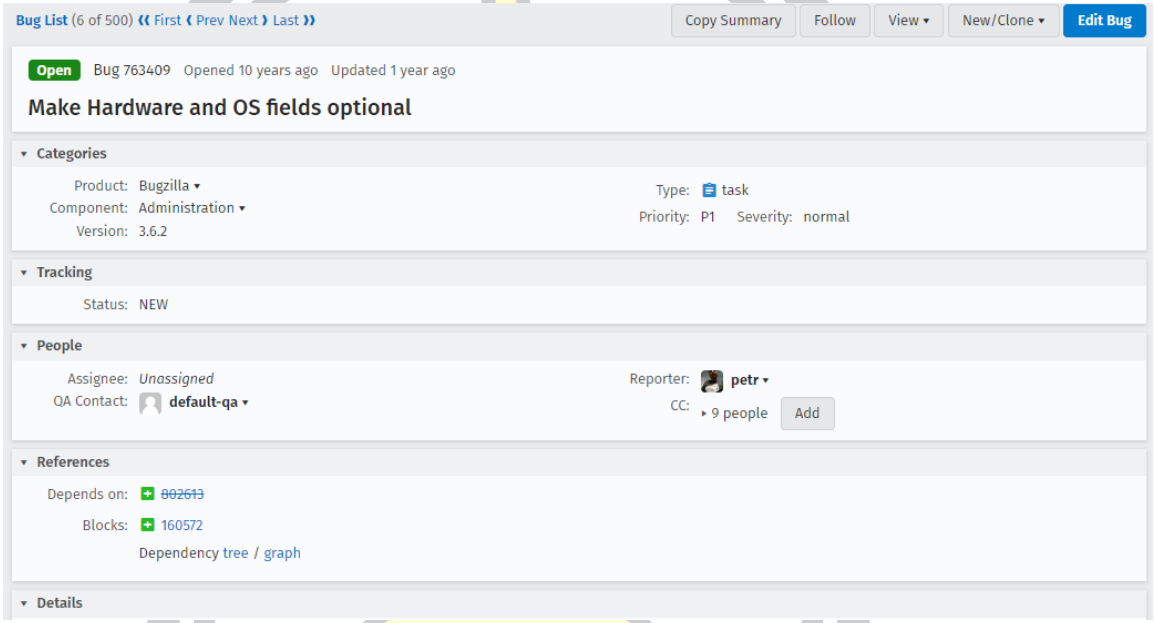
ระบบติดตามจุดบกพร่อง (Bug tracking system หรือ Defect tracking system) [21, 26] คือ ระบบที่ใช้ในการติดตามสถานะของจุดบกพร่องตั้งแต่กระบวนการเริ่มต้นที่ถูกแจ้งเข้ามาในระบบไปจนถึงขั้นตอนการจัดการแก้ไขจุดบกพร่องเสร็จสิ้น รวมทั้งทำหน้าที่เป็นเสมือนคลังเก็บรายงานจุดบกพร่องไปด้วย ซึ่งการใช้งานระบบติดตามจุดบกพร่องโดยส่วนมากจะนิยมใช้กับซอฟต์แวร์ขนาดใหญ่ สามารถแบ่งระบบติดตามจุดบกพร่องตามลักษณะของการนำไปใช้งานได้ 2 กลุ่ม กลุ่มแรก คือ ระบบติดตามจุดบกพร่องของซอฟต์แวร์โอเพนซอร์ส จะเป็นระบบเปิดที่อนุญาตให้ผู้ใช้งานทั่วไปสามารถป้อนรายงานจุดบกพร่องเข้ามาในระบบได้ ตัวอย่างระบบติดตามจุดบกพร่อง เช่น Bugzilla [24] โดยมีซอฟต์แวร์แบบโอเพนซอร์สจำนวนมากที่อยู่บนระบบของ Bugzilla ยกตัวอย่างเช่น Firefox, Thunderbird, SeaMonkey และ Mozilla เป็นต้น และกลุ่มที่สอง คือ ระบบติดตามจุดบกพร่องของซอฟต์แวร์เชิงพานิชย์ จะเป็นระบบปิดที่ไม่อนุญาตให้ผู้ใช้งานทั่วไปใช้งาน แต่จะเป็นการใช้งานภายในบริษัทหรือองค์กรที่เป็นผู้พัฒนาซอฟต์แวร์เท่านั้น เช่น Azure DevOps Server ของบริษัท Microsoft

สำหรับ Bugzilla นั้นเป็นระบบติดตามจุดบกพร่องที่ถูกพัฒนาโดยมูลนิธิมอซิลลา (Mozilla) เมื่อปี พ.ศ. 2541 นอกจากจะเป็นระบบจัดเก็บและติดตามรายงานจุดบกพร่องซอฟต์แวร์ของมูลนิธิมอซิลลาเองแล้ว ยังเปิดให้บริษัท องค์กรอื่น สามารถดาวน์โหลดไปใช้งานของตนเองได้ด้วย ซึ่งข้อมูลจำนวนขององค์กรภายนอกที่นำระบบ Bugzilla ไปใช้งานและถูกเปิดเผยเป็นสาธารณะมีมากถึง 141 องค์กร [27] ยกตัวอย่างเช่น Linux Kernel, Apache, LibreOffice, Open Office, Eclipse และ Red Hat เป็นต้น



รูปที่ 2.2 ระบบรายงานจุดบกพร่องของ Bugzilla

จากรูปที่ 2.2 แสดงรายการของรายงานจุดบกพร่องของระบบ Bugzilla ที่แสดงข้อมูลที่สำคัญ ซึ่งผู้ใช้งานสามารถกำหนดคอลัมน์ให้แสดงข้อมูลได้ และสามารถดูรายละเอียดของรายงานได้โดยการคลิกที่หมายเลข ID ของรายงาน ซึ่งจะแสดงรายละเอียดต่าง ๆ ของรายงานจุดบกพร่องเพิ่มเติม



รูปที่ 2.3 รายละเอียดจุดบกพร่องของระบบ Bugzilla

รายงานจุดบกพร่องของระบบ Bugzilla นั้นมีฟิลด์ข้อมูลที่แสดงเป็นคอลัมน์ตามที่ผู้ใช้งานปรับตามที่กล่าวไว้ข้างต้น ซึ่งสามารถอธิบายฟิลด์ที่สำคัญแสดงดังตารางที่ 2.2

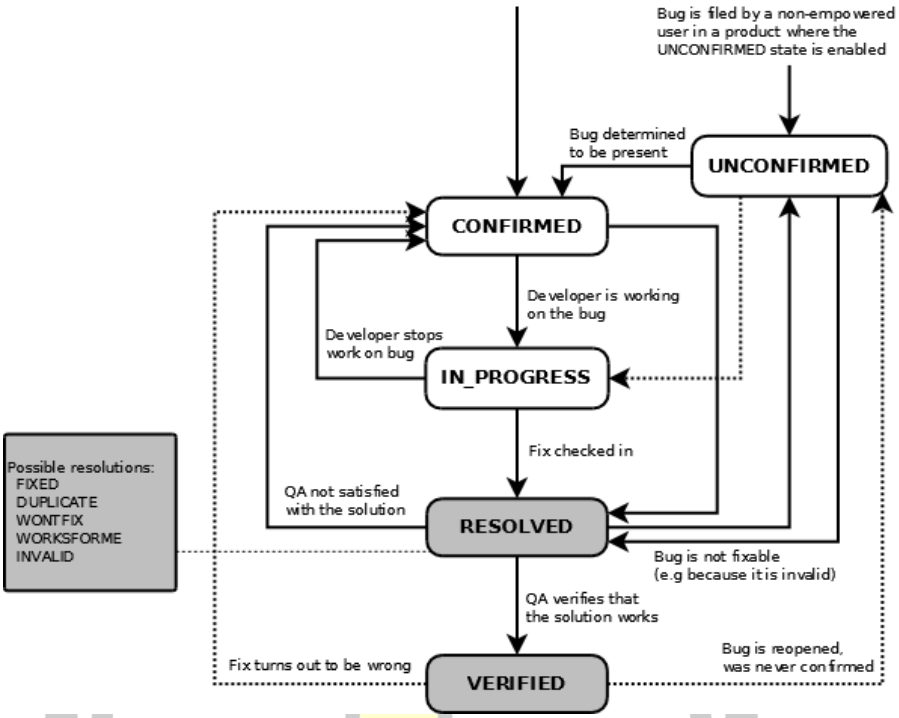
ตารางที่ 2.2 อธิบายฟิลด์รายงานจุดบกพร่องของ Bugzilla

ชื่อฟิลด์	**รายละเอียด**
ID	คือ หมายเลขของรายงาน
Type	คือ ชนิดของรายงาน มี 3 ชนิด ได้แก่ defect enhancement และ task
Summary	คือ ข้อความที่ผู้เขียนรายงานสรุปสั้น ๆ เพื่ออธิบายเกี่ยวกับรายงานนั้น
Status	คือ สถานะของรายงาน แบ่งเป็น 2 ประเภท คือ Open Bugs ประกอบด้วยสถานะ UNCONFIRMED NEW ASSIGNED และ REOPENED และ Closed Bugs ประกอบด้วยสถานะ RESOLVED และ VERIFIED
Assignee	คือ ผู้ที่ถูกมอบหมายให้รับผิดชอบรายงาน

ชื่อฟิลด์	รายละเอียด
Resolutions	คือ สถานะย่อยของรายงานที่ระบุถึงการแก้ไข ได้แก่ FIXED INVALID WONTFIX MOVED DUPLICATE WORKSFORME และ INCOMPLETE
Components	คือ ส่วนประกอบของซอฟต์แวร์ที่ได้รับผลกระทบ
Severity	คือ ระดับความรุนแรงของจุดบกพร่อง โดยจะถูกระบุโดยโปรแกรมเมอร์ หรือผู้พัฒนา เพื่อเป็นข้อมูลประกอบการกำหนดลำดับความสำคัญ ผู้แจ้งรายงานจุดบกพร่องไม่ควรระบุ โดยจะมีสัญลักษณ์ที่แสดงแทนสถานะระดับความรุนแรงดังนี้ - (--) คือ ค่าเริ่มต้นของรายงาน - S1 (Catastrophic) คือ ระดับรุนแรงสูงสุด - S2 (Serious) คือ ระดับรุนแรง - S3 (Normal) คือ ระดับปกติ - S4 (Small/Trivial) คือ ระดับเล็กไม่สำคัญ - N/A (Not Applicable) คือ ไม่ถือว่ามีความรุนแรง ใช้กับสถานะของรายงานชนิด enhancement และ task

2.1.5 วงจรชีวิตของจุดบกพร่อง

เมื่อจุดบกพร่องได้ถูกรายงานขึ้นมาในระบบติดตามจุดบกพร่องแล้ว วงจรชีวิตของจุดบกพร่องจะเริ่มต้นขึ้น ซึ่งสถานะของวงจรชีวิตของจุดบกพร่องนั้นมีอยู่ด้วยกันหลายสถานะ โดยสถานะของจุดบกพร่องจากระบบของ Bugzilla [16] สามารถแสดงได้ดังรูป



รูปที่ 2.4 วงจรชีวิตของจุดบกพร่อง [16]

จากรูปที่ 2.4 สามารถอธิบายสถานะของรายงานจุดบกพร่องได้ดังนี้ เริ่มต้นเมื่อมีการรายงานจุดบกพร่องเข้ามาในระบบสถานะจะเป็น UNCONFIRMED หรือยังไม่มีการยืนยัน จากนั้นผู้ตรวจสอบจะทำการตรวจสอบว่าจุดบกพร่องนี้เป็นจุดบกพร่องที่เคยถูกรายงานเข้ามาแล้วหรือไม่ ถ้าเป็นจุดบกพร่องที่ถูกรายงานเข้ามาแล้วอยู่ในสถานะใด ระหว่าง ASSIGNED (ถูกมอบหมายให้ผู้แก้ไขแล้ว) หรือ RESOLVED (ได้รับการแก้ไขแล้ว) แต่ถ้าหากพบว่าเป็นจุดบกพร่องใหม่สถานะจะถูกยืนยันเป็นรายงานจุดบกพร่องใหม่ หรือ NEW โดยที่ผู้ตรวจสอบจะทำการพิจารณาระดับความสำคัญ (Priority) เพื่อทำการมอบหมายรายงานจุดบกพร่องไปยังผู้แก้ไข ซึ่งโดยส่วนใหญ่การพิจารณาลำดับความสำคัญจะนำระดับความรุนแรง (Severity) มาประกอบด้วย จากนั้นผู้ตรวจสอบจะทำการกำหนดผู้รับผิดชอบตามความเหมาะสม เมื่อมีผู้รับผิดชอบแล้วสถานะจะเปลี่ยนเป็น ASSIGNED และเมื่อผู้รับผิดชอบทำการแก้ไขกับจุดบกพร่องเสร็จแล้วสถานะจะเปลี่ยนเป็น RESOLVED ซึ่งสถานะ RESOLVED จะมีสถานะย่อยอีก แสดงดังตารางที่ 2.3 สุดท้ายเมื่อผู้รับผิดชอบได้ดำเนินการกับจุดบกพร่องแล้วจะถูกตรวจสอบคุณภาพ หากจุดบกพร่องดังกล่าวได้รับการแก้ไขถูกต้องสถานะจะถูกเปลี่ยนเป็น VERIFED และหากไม่มีจุดบกพร่องนี้ถูกรายงานเข้ามาอีกสถานะจะเปลี่ยนเป็น CLOSED แต่หากจุดบกพร่องนั้นยังไม่ถูกต้องสมบูรณ์ สถานะจะกลับไปเป็น REOPEN

ตารางที่ 2.3 อธิบายสถานะของ RESOLVED

ชื่อสถานะ	**รายละเอียด**
FIXED	จุดบกพร่องได้รับการแก้ไขแล้ว
INVALID	ปัญหาที่อธิบายไว้ไม่ใช่จุดบกพร่อง
WONTFIX	จุดบกพร่องที่อธิบายไว้ไม่สามารถแก้ไขได้
MOVED	จุดบกพร่องนี้ถูกย้ายไปเป็นจุดบกพร่องอื่น ควรระบุอ้างอิงปัญหาอื่น
DUPLICATE	จุดบกพร่องซ้ำกับจุดบกพร่องอื่นทุกประการ
WORKSFORME	ไม่สามารถถอดแบบหรือ reproduce จุดบกพร่องได้
INCOMPLETE	จุดบกพร่องถูกรายงานไม่ชัดเจน มีความคลุมเครือ

2.2 ระดับความรุนแรงจุดบกพร่องซอฟต์แวร์

ระดับความรุนแรงจุดบกพร่อง (Bug Severity) [9, 10, 13-15, 17] คือ ระดับของผลกระทบจากจุดบกพร่องซอฟต์แวร์ที่ส่งผลต่อการทำงานของซอฟต์แวร์โดยตรง โดยปกติการกำหนดระดับความรุนแรงของจุดบกพร่องจะถูกกำหนดโดยผู้ที่ได้รับมอบหมายจากผู้จัดการโครงการให้มีหน้าที่ในการตรวจสอบ [16] ซึ่งระดับความรุนแรงของจุดบกพร่องถูกนำไปประกอบในการแก้ไขปัญหาจุดบกพร่องนั้น ๆ สำหรับระดับความรุนแรงของ Bugzilla จะมีอยู่ 5 ระดับ เรียงตามระดับรุนแรงมากไปน้อย ได้แก่ Blocker, Critical, Major, Minor, และ Trivial อธิบายความหมาย และตัวอย่าง Summary ที่บอกถึงแต่ละระดับความรุนแรงของ Bugzilla ได้ดังนี้

ตารางที่ 2.4 อธิบายระดับความรุนแรง

ระดับ	**คำอธิบาย**	**ตัวอย่างประโยค Summary**
Blocker	จุดบกพร่องระดับรุนแรงที่ส่งผลให้ไม่สามารถทดสอบซอฟต์แวร์ต่อไปได้ เช่น ส่งผลให้ซอฟต์แวร์ปิดการทำงาน	[macOS] The operating system’s authentication dialog accepts the empty password only at the second attempt
Critical	จุดบกพร่องระดับที่ส่งผลให้การทำงานหลักของซอฟต์แวร์ผิดพลาด แต่ซอฟต์แวร์ยังสามารถทำงานได้	Firefox deleted all keywords for searches
Major	จุดบกพร่องระดับที่กระทบการทำงานซอฟต์แวร์อย่าง	Can't find where Firefox

ระดับ	คำอธิบาย	ตัวอย่างประโยค Summary
	มาก แต่การทำงานหลักของซอฟต์แวร์ยังทำงานได้	saved my file
Minor	จุดบกพร่องระดับที่ทำให้เกิดความสับสนต่อการใช้งาน โดยส่วนมากจะเป็นความผิดพลาดของ User Interface	Cannot set big picture as background image
Trivial	จุดบกพร่องระดับรุนแรงน้อยที่สุด จะไม่กระทบต่อการทำงานของซอฟต์แวร์ อาจจะเป็นการสะกดคำผิด	Visual issues in the "About Firefox" window on macOS

2.3 การประมวลผลภาษาธรรมชาติ

การประมวลผลภาษาธรรมชาติ (Natural language processing: NLP) [28] คือ วิทยาการส่วนหนึ่งของศาสตร์ปัญญาประดิษฐ์ (Artificial Intelligence: AI) ซึ่งมีหลักการคือช่วยให้คอมพิวเตอร์สามารถเข้าใจ ตีความหมาย และนำไปใช้งาน ภาษาปกติที่มนุษย์ใช้สื่อการกันได้ โดยมีวัตถุประสงค์เพื่อให้ระบบคอมพิวเตอร์สามารถสื่อสารกับมนุษย์ได้ เหตุผลที่มีการพัฒนาการประมวลผลภาษาธรรมชาติขึ้นมา เนื่องจากระบบคอมพิวเตอร์ถูกออกแบบมาให้เข้าใจข้อมูลที่เป็นตัวเลขหรือรหัสที่มีความหมายชัดเจน ซึ่งตรงกันข้ามกับการสื่อสารของมนุษย์ที่มีความซับซ้อนมากกว่ารหัสทางคอมพิวเตอร์ ดังนั้น การประมวลผลภาษาธรรมชาติจึงเกิดมาเพื่อลดช่องว่างของการสื่อสารระหว่างมนุษย์และคอมพิวเตอร์ ในการประมวลผลภาษาธรรมชาติจะมีหลายระดับขึ้นอยู่กับวัตถุประสงค์ของงานที่นำไปใช้งาน สามารถแบ่งระดับการประมวลผลภาษาธรรมชาติได้ 6 ระดับ

1) ระดับสัณฐานวิทยา

ระดับสัณฐานวิทยา (Morphological Level) เป็นระดับขั้นเข้าใจตัวอักษร โดยการประมวลผลจะทำหน้าที่ในการถอดคำเป็นตัวอักษร หาพยัญชนะ สระ รวมถึงตัวสะกด เป็นระดับการแยกคำที่เล็กที่สุดออกมา เช่น คำว่า “unhappiness” สามารถแยกคำเล็กที่สุดออกมาเป็น un, happy และ ness

2) ระดับคำศัพท์

ระดับคำศัพท์ (Lexical Level) เป็นระดับเข้าใจคำศัพท์ โดยทำการประมวลผลเพื่อหาความหมายของคำ (Word or Phrases) เพื่อเตรียมสำหรับการทำความเข้าใจทั้งประโยค ในระดับนี้โดยทั่วไปแล้วจะมีเทคนิคอยู่ 2 วิธี วิธีแรก การตัดส่วนท้ายคำ (Stemming) คือ การลดคำให้อยู่ในรูปแบบต้นกำเนิดหรือรากของคำนั้นโดยการตัดส่วนท้ายออก เช่น กลุ่มคำ Change, Changer,

Changed และ Changing จะประมวลผลออกมาได้ 1 คำ คือ Chang และวิธีที่สอง การแปลงคำ (Lemmatization) คือ การแปลงคำที่มีความหมายเหมือนกันให้เป็นคำเดียวกัน เช่น กลุ่มคำ Change, Changer, Changed และ Changing จะประมวลผลออกมาได้ 1 คำ คือ Change

3) ระดับโครงสร้างประโยค

ระดับโครงสร้างประโยค (Syntactic Level) เป็นระดับเข้าใจประโยค โดยเป็นการประมวลผลเพื่อหาความสัมพันธ์ระหว่างคำในประโยคว่าคำนั้นทำหน้าที่เป็นประธาน (Subject) หรือกรรม (Object) ของประโยค อ้างอิงจากการเข้าใจคำและลำดับโครงสร้างตามมาตรฐานที่ระบุโดยผู้เชี่ยวชาญหรือแบบแผนที่ได้เรียนมา

4) ระดับการตีความ

ระดับการตีความ (Semantic Level) เป็นระดับการเข้าใจบริบทของคำในประโยค เข้าใจถึงความหมายของคำอยู่ในประโยคที่อยู่นอกเหนือโครงสร้างตามมาตรฐานของภาษา โดยจะต้องหาความหมายของคำเมื่อถูกใช้อยู่ในประโยค เนื่องจากคำบางคำสามารถมีความหมายได้หลายความหมายขึ้นอยู่กับบริบทที่ถูกใช้ในประโยคนั้น ๆ ความรู้ที่จำเป็นต่อการประมวลผลระดับการตีความ คือ ความรู้เกี่ยวกับความหมาย (Semantic knowledge)

5) ระดับวาทกรรม

ระดับวาทกรรม (Discourse Level) เป็นระดับการประมวลผลเพื่อให้เข้าใจความหมายเพิ่มเติม โดยดูความเชื่อมโยงของประโยค เข้าใจถึงผลกระทบของประโยคก่อนหน้าต่อความหมายของประโยคที่กำลังสนใจ ความรู้ที่จำเป็นต่อการประมวลผลระดับวาทกรรม คือ ความรู้ทางความหมายเกี่ยวเนื่อง (Discourse knowledge)

6) ระดับปฏิบัติจริง

ระดับปฏิบัติจริง (Pragmatic Level) เป็นการประมวลผลเพื่อแปลความหมายของประโยคนั้นจริง ๆ ว่ามีวัตถุประสงค์ของการสื่อสารถึงอะไร ซึ่งอาจจะไม่ได้มีเนื้อหานั้นอยู่ในประโยค เพื่อให้สามารถตีความได้ใกล้เคียงระดับที่มนุษย์สื่อสารกันมากที่สุด

2.4 การจำแนกเอกสารข้อความ

การจำแนกเอกสารข้อความ (Text Classification) นับว่าเป็นแขนงหนึ่งของการประมวลผลภาษาธรรมชาติ ซึ่งการจำแนกเอกสารข้อความ [29] คือ การแยกกลุ่มเอกสารข้อความตามวัตถุประสงค์แต่ละงาน เช่น การแยกประเภทอีเมล์ขยะหรืออีเมล์ปกติ การแยกระดับความพึง

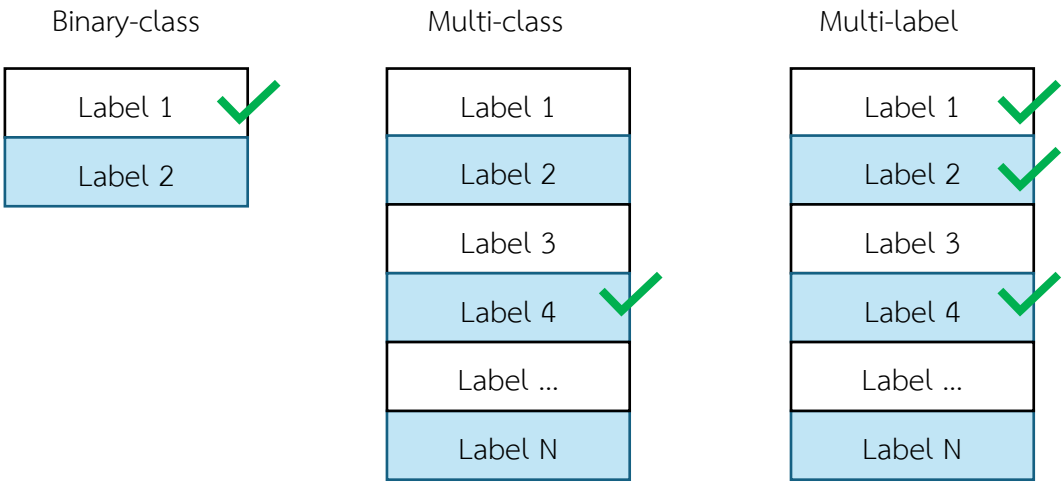
พอใจจากความคิดเห็น การตรวจสอบเว็บไซต์ปลอม เป็นต้น โดยมีพื้นฐานด้วยการประมวลผลจากเอกสารข้อความ ไม่ว่าเอกสารข้อความจะอยู่ในรูปแบบมีโครงสร้าง (Structured data) ข้อความแบบกึ่งโครงสร้าง (Semi structured data) และแบบไม่มีโครงสร้าง (Unstructured data)

โดยทั่วไป การแบ่งประเภทของการจำแนกเอกสารข้อความจะพิจารณา 2 ประเด็น [29] คือ พิจารณาจากจำนวนกลุ่มเป้าหมาย และพิจารณาจากกลุ่มที่เป็นไปได้ของแต่ละเอกสาร

1) การจำแนกเอกสารข้อความแบบสองกลุ่ม (Binary-class Classification) เป็นการจำแนกเอกสารข้อความที่พิจารณาจากจำนวนกลุ่มเป้าหมาย โดยที่กระบวนการนี้จะทำการจำแนกเอกสารข้อความที่ต้องระบุเป้าหมายออกเป็น 2 กลุ่ม คือ “ใช่” และ “ไม่ใช่” หรือ “บวก” และ “ลบ” โดยส่วนมากจะมีการนำไปประยุกต์ใช้ในหลากหลายงาน ยกตัวอย่างเช่น การจำแนกเนื้อหาข่าวว่าเป็น ข่าวจริง (Real News) หรือ ข่าวปลอม (Fake News) จะเห็นได้ว่าเป้าหมายของประเด็นนี้คือการจำแนกข่าว การที่จะจำแนกว่าเป็นข่าวจริงหรือข่าวปลอม ต้องทำการพิจารณาจากบริบทเนื้อหาของข่าว

2) การจำแนกเอกสารข้อความแบบหลายกลุ่ม (Multi-class Classification) เป็นการจำแนกเอกสารข้อความที่พิจารณาจากจำนวนกลุ่มเป้าหมาย โดยที่กระบวนการนี้เป็นกระบวนการจำแนกเอกสารออกเป็นหลายกลุ่ม เช่น การจำแนกประเภทของข่าวเป็น “ข่าวการเมือง” “ข่าวบันเทิง” “ข่าวกีฬา” และ “ข่าวอาชญากรรม” ซึ่งต้องพิจารณาจากบริบทต่าง ๆ ของเนื้อหาข่าว ว่าเป็นข่าวประเภทใด ซึ่งต้องอยู่ในประเภทใดประเภทหนึ่งเท่านั้นจากหลาย ๆ ประเภทที่มี

3) การจำแนกเอกสารข้อความที่หนึ่งเอกสารเป็นได้หลายกลุ่ม (Multi-label Classification) เป็นการจำแนกเอกสารข้อความที่พิจารณาจากกลุ่มที่เป็นไปได้ของแต่ละเอกสาร โดยที่เอกสาร 1 เอกสารสามารถเป็นไปได้หลายกลุ่ม เช่น เอกสารงานวิจัย 1 เอกสาร อาจจะเป็นได้หลายกลุ่ม ยกตัวอย่างเช่น “วิทยาการคอมพิวเตอร์” “คณิตศาสตร์” “สถิติ” และ “การเงิน” ซึ่งต้องพิจารณาจากบริบทต่าง ๆ ของเอกสารงานวิจัยนั้น ๆ ไม่ว่าจะเป็น ชื่อเรื่อง บทคัดย่อ วิธีการดำเนินการ วิธีการประเมินผล เป็นต้น

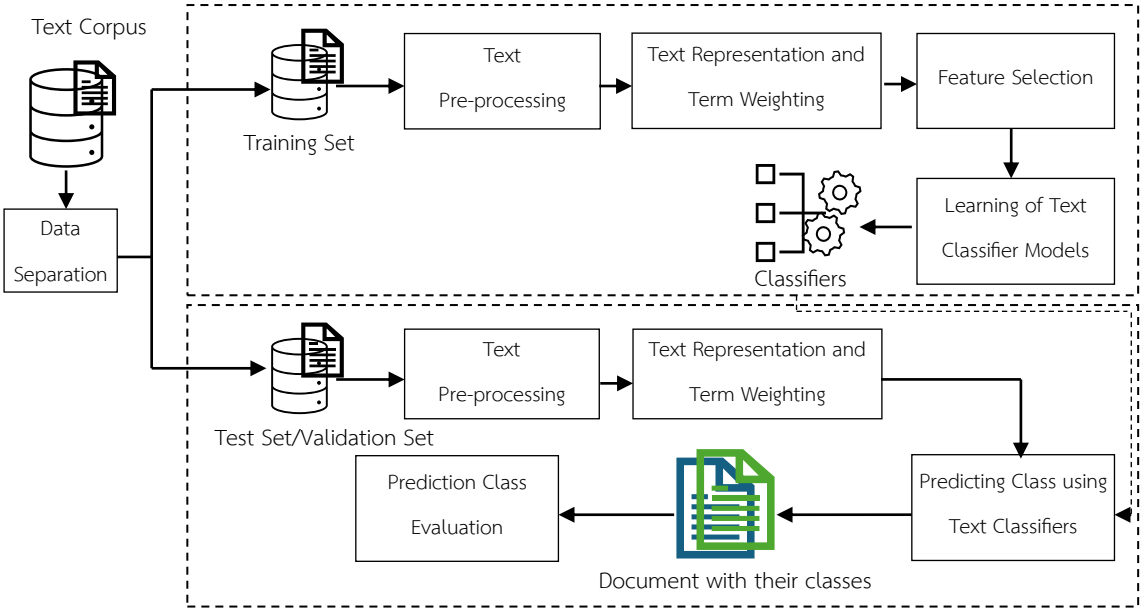


รูปที่ 2.5 เปรียบเทียบการจำแนกเอกสารข้อความแต่ละประเภท

จากรูปที่ 2.5 เปรียบเทียบการจำแนกเอกสารข้อความแต่ละประเภท ซึ่งจะเห็นว่าประเภทแรก คือ แบบสองกลุ่มและเอกสารข้อความต้องอยู่ในกลุ่มใดกลุ่มหนึ่งจากกลุ่มเป้าหมาย 2 กลุ่ม แบบที่สอง คือ แบบหลายกลุ่มและเอกสารข้อความต้องอยู่ในกลุ่มใดกลุ่มหนึ่งจากกลุ่มเป้าหมายหลายกลุ่ม และแบบที่สาม คือ แบบหนึ่งเอกสารสามารถเป็นได้หลายกลุ่ม

2.5 กระบวนการในการจำแนกเอกสารข้อความ

กระบวนการในการจำแนกเอกสารข้อความประกอบด้วย 2 ส่วนหลัก ๆ [29] คือ ส่วนของการสร้างโมเดลเพื่อทำการจำแนกเอกสารข้อความ และส่วนของการประเมินโมเดลที่สร้างขึ้นเพื่อทำการจำแนกเอกสารข้อความ อย่างไรก็ตามใน 2 ส่วนหลัก ๆ นี้จะมีขั้นตอนย่อยที่มีรายละเอียดในแต่ละขั้นตอนอีกด้วย



รูปที่ 2.6 กรอบการทำงานการจำแนกเอกสารข้อความ [29]

กรอบการดำเนินงานสำหรับการจำแนกเอกสารข้อความแสดงดังรูปที่ 2.6 โดยการแบ่งคลังข้อมูลออกเป็น 2 ชุด คือ ชุดสำหรับฝึกโมเดล และชุดข้อมูลทดสอบหรือชุดตรวจสอบ ต่อจากนั้นจะเข้าสู่กระบวนการดำเนินงานสำหรับการจำแนกเอกสารข้อความในแต่ละขั้นตอน

2.5.1 การสร้างโมเดลเพื่อทำการจำแนกเอกสารข้อความ

1) การเตรียมเอกสารข้อความ (Text Pre-processing)

การเตรียมเอกสารข้อความเป็นขั้นตอนที่ต้องเตรียมชุดข้อมูลให้มีความเหมาะสมเพื่อนำไปประมวลผลในขั้นตอนถัดไป ซึ่งในขั้นตอนการเตรียมเอกสารข้อความยังมีขั้นตอนย่อยอีกหลายขั้นตอน ทั้งนี้ขึ้นอยู่กับชุดข้อมูลเอกสาร รวมถึงประเด็นปัญหาการวิจัย อย่างไรก็ตาม ขั้นตอนหลัก ๆ ในส่วนนี้อาจจะประกอบไปด้วย การทำความสะอาดข้อความ (Text Cleaning) เช่น การลบแท็กต่าง ๆ ของเอกสาร เพราะหากเป็นชุดข้อมูลที่อยู่ในรูปแบบกึ่งโครงสร้าง (Semi Structured) จะมีแท็กพิเศษที่ไม่จำเป็นต่อการประมวลผล รวมไปถึงการลบอักขระพิเศษต่าง ๆ ที่ไม่จำเป็นด้วย หลังจากนั้นทำการแยกเอกสารข้อความออกเป็นส่วนย่อย (Tokenization) ซึ่งการแยกเอกสารข้อความออกเป็นส่วนย่อยที่นิยมใช้มาทำการวิเคราะห์ในการจำแนกเอกสารข้อความ ได้แก่ การแยกคำ (Word) กลุ่มคำ (Word Group) วลี (Phrase) เอ็นแกรม (N-Gram) และ ประโยค (Sentence)

ทั้งนี้ในแต่ละชุดข้อมูลอาจจะเหมาะสมสำหรับแต่ละงานวิจัย แต่โดยส่วนมากจะนิยมใช้รูปแบบ “คำ” และ ข้อความที่แยกเป็นส่วนย่อยนี้จะถูกพิจารณาไปเป็น คุณลักษณะหรือฟิเจอร์ (Feature) ของเอกสารข้อความนั้น ๆ ต่อไป

หลังจากขั้นตอนแยกเอกสารข้อความออกเป็นส่วนย่อย ซึ่งในที่นี้อาจจะใช้รูปแบบของ “คำ” แต่ในเอกสารข้อความนั้นยังมีคำที่ไม่มีนัยสำคัญที่จะนำไปใช้ประมวลผล โดยคำเหล่านี้จะเรียกว่า “Stop Words” กล่าวคือ คำที่อยู่ในรูปแบบ วลี หรือคำสันธาน ที่สามารถลบออกได้ ซึ่งในขั้นตอนนี้ จะมีเทคนิคอยู่ 2 รูปแบบ คือ การลดรูปแบบของคำด้วยวิธีการสเต็มมิ่ง (Stemming) หรือการเปลี่ยนรูปแบบของคำด้วยวิธีการเล็มม่าไทเซชั่น (Lemmatization) ตารางที่ 2.5 อธิบายขั้นตอนการเตรียมเอกสารข้อความพร้อมตัวอย่างผลลัพธ์ที่ได้แต่ละขั้นตอน

ตารางที่ 2.5 ตัวอย่างขั้นตอนเตรียมเอกสารข้อความ

ขั้นตอน	**ผลลัพธ์**
Input Text	<report id="354304"> <what>Changing several bugs at once fails if aliases are in use</what> </report>
Text Cleaning	Changing several bugs at once fails if aliases are in use
Tokenization	'Changing', 'several', 'bugs', 'at', 'once', 'fails', 'if', 'aliases', 'are', 'in', 'use'
Stop-word	'changing', 'several', 'bugs', 'fails', 'aliases', 'use'
Stemming	'chang', 'sever', 'bug', 'fail', 'alias', 'use'

2) การแทนเอกสาร (Document Representation)

ขั้นตอนการแทนเอกสารข้อความเป็นขั้นตอนที่มีเป้าหมายเพื่อแทนข้อความรูปแบบต่าง ๆ ให้อยู่ในรูปแบบที่มีมาตรฐาน เช่น การใช้ดัชนีคำ เพื่อให้ง่ายต่อการประมวลผล โดยทั่วไปรูปแบบของการแทนเอกสารข้อความที่ได้รับความนิยม คือ แบบจำลองปริภูมิเวกเตอร์ (Vector Space Model: VSM) ซึ่งแบบจำลองปริภูมิเวกเตอร์จะกำหนดให้เอกสารแต่ละฉบับอยู่ใน

รูปแบบของเวกเตอร์ของฟีเจอร์ (Features) และขนาดของเวกเตอร์จะมีขนาดเท่ากับจำนวนของฟีเจอร์ทั้งหมดของเอกสารชุดนั้น ถ้ากำหนดให้ D เป็นเซตของชุดเอกสารข้อความ นั้นคือ $D = \{d_1, d_2, d_3, \dots, d_i\}$ เมื่อ d_i คือเอกสารข้อความ สมมุติว่าฟีเจอร์ของเอกสาร คือ “Word” กำหนดให้ w_{ik} คือ น้ำหนักของคำ k ที่ปรากฏในเอกสารฉบับที่ i เวกเตอร์ของเอกสาร D_i จึงเขียนแทนด้วย $D_i = \{w_1, w_2, w_3, \dots, w_{it}\}$ สามารถยกตัวอย่างการแทนเอกสารข้อความด้วย VSM สมมุติว่ามีเอกสารข้อความ 3 เอกสาร ได้แก่

D_1: Changing several bugs at once fails if aliases are in use.

D_2: Various issues when changing several bugs at once.

D_3: changing several bugs at once using process.

เมื่ออนุมานว่าเอกสารที่ยกตัวอย่างผ่านขั้นตอนการเตรียมเอกสารข้อความแล้ว สามารถแสดงตัวอย่างในรูปแบบของ VSM

ตารางที่ 2.6 ตัวอย่างการแทนเอกสารด้วย VSM

	chang	sever	variou	bug	fail	alias	issu	use	process
D_1	1	1	0	1	1	1	0	1	0
D_2	1	1	1	1	0	0	1	0	0
D_3	1	1	0	1	0	0	0	1	1

ตารางที่ 2.6 แสดงตัวอย่างการแทนเอกสารข้อความด้วย VSM แต่ละคอลัมน์แสดงคำหรือฟีเจอร์ที่ถูกเลือกของชุดเอกสารข้อความทั้งหมด และข้อมูลแต่ละแถวคือเอกสารแต่ละชุด ตัวเลข 1 หมายถึงคำนั้นปรากฏในเอกสารนั้น และ 0 คือ ไม่ปรากฏในเอกสารนั้น

3) การให้น้ำหนักคำ (Term Weighting)

การให้น้ำหนักคำเป็นขั้นตอนหลังจากการแทนเอกสารข้อความแล้ว เพื่อให้การวิเคราะห์ประเด็นปัญหาที่สนใจและส่งผลให้โมเดลมีประสิทธิภาพ จำเป็นต้องทราบว่า “คำ” ที่ปรากกฏในเอกสารมีนัยสำคัญหรือไม่ การให้น้ำหนักคำจึงเป็นอีกขั้นตอนหนึ่งที่สำคัญในการจำแนกเอกสารข้อความ สำหรับการให้น้ำหนักคำนั้นมีอยู่หลายประเภท ดังนี้

การให้น้ำหนักคำด้วยความถี่ (Term Frequency: tf) [30] แนวคิดของการให้น้ำหนักคำด้วยความถี่ คือคำที่มักปรากฏในเอกสารมักจะมีความสำคัญหรือมีความเกี่ยวข้องกับหัวข้อของเอกสาร โดยจะพิจารณาที่ความถี่ของคำเป็นหลัก และไม่ได้นำลำดับที่ปรากฏของคำมาพิจารณาในการให้ค่าน้ำหนักด้วย การคำนวณการให้น้ำหนักคำด้วยความถี่สามารถคำนวณได้ ดังสมการที่ (2.1)

$$tf_{t,d} = \frac{f_d(t)}{\sum_{i \in d} f_d(t)} \quad (2.1)$$

โดยที่ $f_d(t)$ คือ ค่าความถี่ของคำ t ที่เกิดในเอกสาร d แต่เนื่องจากแต่ละเอกสารอาจจะมีขนาดความยาวที่แตกต่างกัน ดังนั้นการคำนวณหาค่าด้วยน้ำหนักความถี่ จึงทำการหารด้วยจำนวนคำทั้งหมดของเอกสาร

การให้น้ำหนักคำแบบความถี่เอกสารผกผัน (Inverse Document Frequency: idf) [31] วิธีการให้น้ำหนักคำด้วยความถี่ทำการพิจารณาความถี่ของคำแต่ละเอกสาร แต่ในความเป็นจริงแล้วในการแก้ปัญหามีจำนวนเอกสารจำนวนมากที่อยู่ในคลังเอกสารทั้งหมด ซึ่งอาจจะไม่เกิดความเป็นธรรมหากไม่พิจารณาจากชุดเอกสารทั้งหมด ดังนั้น วิธีการนี้จึงทำการคำนวณค่าน้ำหนักของคำหนึ่งคำด้วยการพิจารณาจากคลังเอกสารทั้งหมด การคำนวณการให้น้ำหนักคำแบบความถี่เอกสารผกผัน ดังสมการที่ 2.2

$$idf_t = log \frac{N}{df_t} \quad (2.2)$$

โดยที่ *N* คือ จำนวนเอกสารทั้งหมดในคลังเอกสาร และ df_t จำนวนเอกสารทั้งหมดที่ปรากฏคำ t

การให้น้ำหนักคำแบบความถี่และความถี่เอกสารผกผัน (Term frequency Inverse document frequency: tf-idf) วิธีการนี้จะทำการผสมวิธีการการให้น้ำหนักคำด้วยความถี่และการให้น้ำหนักคำแบบความถี่เอกสารผกผัน ด้วยการนำมาคูณ ดังสมการที่ 2.3

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \quad (2.3)$$

4) การเลือกคุณลักษณะ (Feature Selection)

ก่อนที่จะนำข้อมูลเข้าให้กับโมเดลสำหรับการฝึกนั้น จำเป็นต้องมีการเลือกคุณลักษณะของคำที่มีความจำเป็นและสำคัญมากที่สุดเท่านั้น เพราะการนำคุณลักษณะทั้งหมดเข้าไปอาจจะเกิดการรบกวนต่อการฝึกโมเดลและส่งผลต่อประสิทธิภาพของผลลัพธ์ได้ การเลือกคุณลักษณะที่ดีนั้นจะส่งผลให้ได้โมเดลที่เรียบง่าย เวลาในการฝึกสั้นลง ลดความแปรปรวน ซึ่งจะส่งผลให้ได้โมเดล

ที่มีประสิทธิภาพและได้ผลลัพธ์ที่ดี กระบวนในการเลือกคุณลักษณะสามารภจำแนกได้ 4 ประเภท [29] คือ วิธีฟิลเตอร์ (Filter Method) วิธีแรปเปอร์ (Wrapper Method) วิธีฝังตัว (Embedded Method) และวิธีผสม (Hybrid Method)

การคัดเลือกฟีเจอร์แบบวิธีฟิลเตอร์ (Filter Method) [29] เป็นวิธีการเลือกคุณสมบัติที่ไม่ขึ้นกับอัลกอริทึมการเรียนรู้ของเครื่องใด ๆ แต่คุณลักษณะต่างๆ จะถูกเลือกตามคะแนนในการทดสอบทางสถิติสำหรับความสัมพันธ์กับตัวแปรผลลัพธ์ การคัดเลือกฟีเจอร์แบบนี้มีการใช้งานอย่างแพร่หลายในการจำแนกเอกสาร เพราะเป็นวิธีที่ง่าย รวดเร็ว เพราะให้ประสิทธิภาพในการเลือกชุดของฟีเจอร์ที่จะใช้ในรอบการทำงานเดียว

การคำนวณคะแนนทางสถิติมีด้วยกันหลายฟังก์ชัน เช่น Information Gain (IG), Document frequency (DF), Term Frequency-Inverse Document Frequency (tf-idf) หรือ Chi สามารถอธิบายได้ดังต่อไปนี้

Information Gain (IG) คือ เทคนิคทางทฤษฎีสารสนเทศที่ใช้ในการเลือกคุณลักษณะ ในการสร้างโมเดลการเรียนรู้ของเครื่อง เช่น โมเดลการจำแนกประเภทข้อความ คำนวณโดยการวัดการลดลงของความไม่แน่นอน (uncertainty) หรือความแปรปรวน (entropy) ของกลุ่มเป้าหมายเมื่อใช้คุณลักษณะนั้นในการแบ่งกลุ่มข้อมูล กล่าวคือ IG วัดว่าการใช้คุณลักษณะหนึ่ง ๆ ในการแบ่งกลุ่มข้อมูลนั้นช่วยลดความไม่แน่นอนในการทำนายผลลัพธ์ได้มากเพียงใด สมมุติว่ามีชุดข้อมูลรีวิวสินค้าที่ต้องการจำแนกว่าเป็นรีวิวบวกหรือลบ ก่อนทำการจำแนกความไม่แน่นอนของชุดข้อมูลรีวิวอาจสูง เนื่องจากมีรีวิวทั้งบวกและลบปะปนกัน หากต้องการทราบว่าการใช้คำ "ยอดเยี่ยม" ในรีวิวนั้นมีผลต่อการทำนายผลลัพธ์อย่างไร ดังนั้นการคำนวณ Information Gain ของ "ยอดเยี่ยม" เมื่อพบว่ารีวิวที่มีคำว่า "ยอดเยี่ยม" มักจะเป็นรีวิวบวก ในขณะที่รีวิวโดยไม่มีคำนี้มีการกระจายของรีวิวบวกและลบที่ไม่แน่นอนมากขึ้น จะทำให้ IG สูง บ่งบอกว่าการใช้คำนี้เป็นตัวแบ่งชุดข้อมูลช่วยลดความไม่แน่นอนได้ดี ด้วยเหตุนี้อาจเลือกใช้คำ "ยอดเยี่ยม" เป็นหนึ่งในคุณลักษณะสำคัญในการสร้างโมเดลจำแนกรีวิวบวกหรือลบได้ ซึ่งการคำนวณ IG จาก Entropy ใช้สมการดังนี้

Entropy ของเซตข้อมูล S สามารถคำนวณได้จากสมการ

$$Entropy(S) = \sum_{i=1}^{n} p_i \, log_2(p_i) \tag{2.4}$$

โดยที่

n คือ จำนวนคลาสต่างๆ ในเซต S

p_i คือ สัดส่วนของสมาชิกในคลาส i ต่อทั้งหมดในเซต S

หากแบ่ง S ด้วยแอตทริบิวต์ A ที่มีค่าที่เป็นไปได้ v ค่า และ Entropy หลังจากแบ่งคือ

$$Entropy(S,A) = \sum_v \frac{|S_v|}{|S|} Entrop(S_v) \quad (2.5)$$

โดยที่

S_v คือ เซตข้อมูลที่ได้หลังจากแบ่ง S ด้วยค่า v ของแอตทริบิวต์ A

$|S_v|$ คือ ขนาดของเซต S_v

$|S|$ คือ ขนาดของเซต S เดิมก่อนแบ่ง

Information Gain จากการแบ่งเซต S ด้วยแอตทริบิวต์ A คือ

$$IG(S,A) = Entropy(S) - Entropy(S,A) \quad (2.6)$$

Document Frequency (DF) เป็นเทคนิคทางสถิติที่ใช้ในการประมวลผลข้อความและการวิเคราะห์ข้อมูลข้อความ (text analytics) เพื่อวัดความถี่หรือจำนวนครั้งที่คำปรากฏในชุดข้อมูลข้อความ โดยเฉพาะความถี่เอกสารของคำนั้นคือจำนวนเอกสารหรือข้อความในชุดข้อมูลที่คำนั้นปรากฏอยู่ ในงานจำแนกข้อความการทราบ DF ของแต่ละคำมีความสำคัญในการทำความเข้าใจความสำคัญหรือความหายากของคำนั้นในชุดข้อมูล คำที่มี DF สูงอาจเป็นคำทั่วไปหรือคำที่ไม่มีความเฉพาะเจาะจงมากนักในเรื่องหรือหมวดหมู่ใดหมวดหมู่หนึ่ง ในขณะที่คำที่มี DF ต่ำอาจบ่งชี้ถึงคำที่มีความเฉพาะเจาะจงและสำคัญในหมวดหมู่หรือเรื่องราวนั้น ๆ ถึงแม้เป็นวิธีที่ง่ายและมีประสิทธิภาพ แต่อย่างไรก็ตาม DF ยังมีข้อเสีย คือ คำที่ถูกเลือกเป็นฟีเจอร์อาจจะเป็นคำที่พบได้ในหลาย ๆ เอกสาร ดังนั้นอาจจะส่งผลให้การวิเคราะห์เพื่อการจำแนกเอกสารได้ผลลัพธ์ที่ไม่ดี เนื่องจากการเกิดซ้ำ ๆ ในหลายเอกสารนั้นเอง ซึ่งการทราบ DF ช่วยให้ผู้วิเคราะห์สามารถตัดสินใจได้ว่าควรใช้คำใดเป็นคุณลักษณะ ในการสร้างโมเดลจำแนกข้อความ และเพื่อช่วยปรับปรุงความแม่นยำของโมเดล โดยอาจนำไปสู่การใช้เทคนิคที่เรียกว่า TF-IDF ซึ่งเป็นการน้ำหนักคำตามความสำคัญโดยใช้ DF เป็นส่วนหนึ่งของการคำนวณ

สามารถแสดงตัวอย่างการคำนวณการคัดเลือกฟีเจอร์แบบวิธีฟิลเตอร์ด้วยฟังก์ชัน DF โดยมีชุดเอกสารข้อความจำนวน 4 เอกสาร โดยเนื้อหาประกอบด้วยคำที่เกี่ยวกับ ผัก “Vegetable” และ ผลไม้ “Fruit” เอกสารดังที่แสดงตาราง 2.7

ตารางที่ 2.7 ตัวอย่างเอกสารข้อความสำหรับคัดเลือกฟีเจอร์แบบฟิลเตอร์

Document	Content	Class
D1	Apple Banana Carrot Strawberry Mango Orange	Fruit
D2	Cucumber Tomato Spinach Tomato	Vegetable
D3	Tomato Cucumber Strawberry Carrot	Vegetable
D4	Mango Strawberry	Fruit

ขั้นตอนที่ 1: คำนวณหาจำนวนเอกสารทั้งหมด ซึ่งเท่ากับ 4

ขั้นตอนที่ 2: คำนวณจำนวนของคำจากทุกเอกสาร

ขั้นตอนที่ 3: คำความอัตราส่วนของคำจากทุกเอกสาร

ขั้นตอนที่ 4: เรียงลำดับอัตราส่วนหรือความถี่ของคำจากมากไปน้อย

ขั้นตอนที่ 5: กำหนดจำนวนคำที่ต้องการเลือก สมมุติกำหนดเท่ากับ 5 คำ

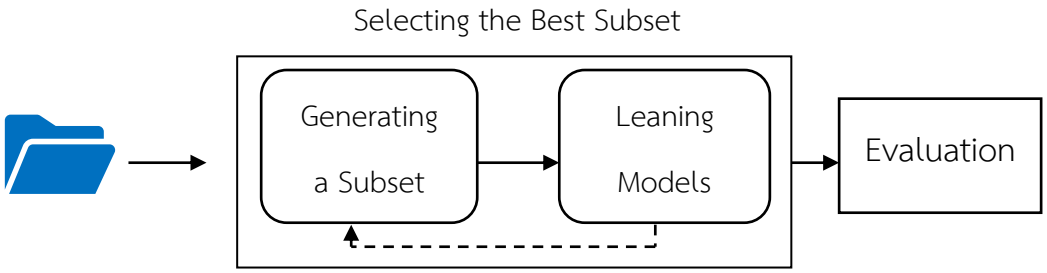
แสดงตัวอย่างการคำนวณการคัดเลือกฟีเจอร์ด้วยฟังก์ชัน df แต่ละขั้นตอน โดยเริ่มจากขั้นตอนที่ 2 ดังตารางที่ 2.8

ตารางที่ 2.8 ตัวอย่างการคำนวณคัดเลือกฟีเจอร์ด้วย DF

ขั้นตอนที่ 2	ขั้นตอนที่ 3	ขั้นตอนที่ 4	ขั้นตอนที่ 5
Apple = 1	Apple = 1/4 = 0.25	Carrot = 0.5	Carrot
Banana = 1	Banana = 1/4 = 0.25	Strawberry = 0.5	Strawberry
Carrot = 2	Carrot = 2/4 = 0.5	Mango = 0.5	Mango
Strawberry = 2	Strawberry = 2/4 = 0.5	Cucumber = 0.5	Cucumber
Mango = 2	Mango = 2/4 = 0.5	Tomato = 0.5	Tomato
Orange = 1	Orange = 1/4 = 0.25	Apple = 0.25	
Cucumber = 2	Cucumber = 2/4 = 0.5	Banana = 0.25	** คำที่ถูกเลือกเป็นฟีเจอร์*
Tomato = 2	Tomato = 2/4 = 0.5	Orange = 0.25	
Spinach = 1	Spinach = 1/4 = 0.25	Spinach = 0.25	

การคัดเลือกฟีเจอร์แบบการควบรวม (Wrapper Method) [29] เป็นการคัดเลือกเซตย่อยของฟีเจอร์ (Subset) จากฟีเจอร์ที่มีทั้งหมด กระบวนการเลือกคุณลักษณะนี้ขึ้นอยู่

กับอัลกอริทึมแมชชีนเลิร์นนิงเฉพาะที่พยายามปรับให้เข้ากับชุดข้อมูลที่กำหนด ซึ่งมีอัลกอริทึมที่สามารถค้นหาฟีเจอร์ได้แบบนี้ได้หลายตัว เช่น การค้นแบบดีที่สุด (Best First Search: BFS) การค้นแบบแตกกิ่งและจำกัดขอบเขต (Branch-and-Bound Search) การค้นหาแบบละโมภ (Greedy) และวิธีเชิงพันธุกรรม (Genetic Algorithm) เป็นต้น ขั้นตอนการดำเนินการในแบบวิธีควบรวมแสดงดังรูปที่ 2.7



รูปที่ 2.7 ขั้นตอนการดำเนินการในแบบวิธีควบรวม

การคัดเลือกฟีเจอร์แบบฝังตัว (Embedded Feature Selection) [29] เป็นการคัดเลือกฟีเจอร์ที่มีกระบวนการคือ รวมเอาขั้นตอนของการคัดเลือกฟีเจอร์เข้าไปเป็นส่วนหนึ่งของการฝึกโมเดล นั้นคือวิธีการนี้จะไม่มีขั้นตอนการคัดเลือกฟีเจอร์ก่อนการฝึกโมเดล แต่จะทำการฝังการดำเนินการคัดเลือกฟีเจอร์ไว้ในกระบวนการฝึกโมเดล ซึ่งอัลกอริทึมที่ทำการฝึกจะกำหนดค่าน้ำหนักหรือความสำคัญให้กับฟีเจอร์แต่ละอย่างตามการมีส่วนร่วม โดยที่คุณลักษณะที่มีค่าความสำคัญต่ำจะถูกลบออกจากโมเดล เหลือเพียงคุณลักษณะที่สำคัญที่สุดเท่านั้น จากนั้นโมเดลจะได้รับการฝึกอบรมใหม่ด้วยชุดคุณลักษณะที่ลดลง และดำเนินการซ้ำ ๆ จนกว่าจะได้ประสิทธิภาพในระดับที่น่าพอใจ ตัวอย่างอัลกอริทึมที่รวมเอาการคัดเลือกฟีเจอร์ไปเป็นส่วนหนึ่งของกระบวนการฝึกอบรม เช่น อัลกอริทึมต้นไม้ตัดสินใจ (Decision Tree: DT) หรือ Gradient Boosting Algorithms [32] เป็นต้น

การคัดเลือกฟีเจอร์แบบผสม (Hybrid Method) [29] เป็นการคัดเลือกฟีเจอร์ที่ทำการผสานทั้งสามวิธีข้างต้น แต่โดยส่วนใหญ่จะเป็นการผสมผสานระหว่างวิธีฟิลเตอร์และวิธีควบรวม โดยจะใช้วิธีฟิลเตอร์เพื่อทำการลดขนาดมิติของฟีเจอร์ และทำการคัดเลือกมาจำนวนหนึ่ง หรืออาจจะเป็นไปได้ว่าฟีเจอร์ที่ได้นั้นมีได้ฟีเจอร์หลายชุด จากนั้นจะใช้วิธีการแร็ปเปอร์ในการเลือกชุดของฟีเจอร์ที่เหมาะสมที่สุดอีกครั้ง

5) การฝึกโมเดลตัวจำแนกเอกสารข้อความ

หลังจากได้ถุงคำที่ประกอบด้วย คำ พร้อมทั้งการให้น้ำหนักและคัดเลือกคุณลักษณะที่อยู่ในรูปแบบที่เป็นมาตรฐานพร้อมการประมวลผลแล้ว ขั้นตอนนี้จะทำการเข้าถุงคำเข้าสู่กระบวนการสร้างโมเดลสำหรับการจำแนกเอกสารข้อความ หรือเรียกว่า ตัวจำแนกเอกสารข้อความ (Text Classifier) ซึ่งโดยทั่วไปแล้วในขั้นตอนนี้มีอยู่ด้วยกันหลายรูปแบบ และมีอัลกอริทึมที่สามารถทำหน้าที่ในการจำแนกได้ สามารถแบ่งรูปแบบวิธีการสร้างตัวจำแนกเอกสารได้ 2 กลุ่ม คือ การเรียนรู้ของเครื่องแบบดั้งเดิม (Traditional Machine Learning) และการเรียนรู้เชิงลึก (Deep Learning) โดยจะอธิบายและยกตัวอย่างอัลกอริทึมแต่ละกลุ่มในหัวข้อที่ 2.6 และ 2.7 ตามลำดับ

2.5.2 การทดสอบและการประเมินโมเดลตัวจำแนกเอกสารข้อความ

การประเมินโมเดลตัวจำแนกเอกสารข้อความ จะทำให้ทราบถึงประสิทธิภาพของโมเดลที่สร้างขึ้น สำหรับเครื่องการประเมินโมเดลด้านการจำแนกเอกสารข้อความ โดยส่วนมากนิยมใช้ตาราง Confusion Matrix มาใช้ในการประเมินประสิทธิภาพ ซึ่งเครื่องมือนี้สามารถนำไปคำนวณตัววัดประสิทธิภาพที่นิยมนำมาใช้หลายงานวิจัย

เครื่องมือ Confusion Matrix จะเป็นรูปแบบตารางแบบจัตุรัสที่จำนวนแถวและคอลัมน์เท่ากับจำนวนคลาสที่ต้องการทำนาย เช่น หากมี 2 คำตอบหรือคลาส คือ จริง และ เท็จ ตาราง Confusion Matrix จะอยู่ในรูปแบบ 2x2

ตารางที่ 2.9 Confusion Matrix แบบ 2x2

	Actually Positive (1)	**Actually Negative (0)**
Predicted Positive (1)	True Positives (TP)	False Positives (FP)
Predicted Negative (0)	False Negatives (FN)	True Negatives (TN)

จากตารางที่ 2.9 อธิบายได้ ดังนี้

1) True Positive (TP) คือ สิ่งที่ทำนายตรงกับความจริง กรณีโมเดลทำนายว่าจริงและข้อมูลเป็นจริง

2) True Negative (TN) คือ สิ่งที่ทำนายตรงกับความจริง กรณีโมเดลทำนายว่าเท็จและข้อมูลเป็นเท็จ

3) False Positive (FP) คือ สิ่งที่ทำนายไม่ตรงกับความจริง กรณีโมเดลทำนายว่าจริง แต่ข้อมูลเป็นเท็จ

4) False Negative (FN) คือ สิ่งที่ทำนายไม่ตรงกับความจริง กรณีโมเดลทำนายว่าเท็จ แต่ข้อมูลเป็นจริง

กรณีการประเมินประสิทธิภาพโมเดลแบบหลายคลาสจะใช้ multiclass confusion matrix โดยรูปแบบตารางจะขึ้นอยู่กับจำนวนคลาสที่ต้องการจำแนก เช่น จำนวน 3 คลาส จะเป็นตาราง 3x3

ตารางที่ 2.10 Confusion Matrix แบบ NxN

		Actually			
		class (1)	**class (2)**	**...**	**class (n)**
Predicted	**class (1)**	TP(1)	FN(2), FP(1)	...	FN(n), FP(1)
	class (2)	FN(1), FP(2)	TP(2)	...	FN(n), FP(2)
	...	...	...	...	...
	class (n)	FN(1), FP(n)	FN(2), FP(n)	...	TP(n)

โดยทีสามารถใช้ Confusion Matrix มาคำนวณเพื่อการประเมินประสิทธิภาพของแบบจำลองในรูปแบบต่าง ๆ ได้หลายค่า ได้แก่

1) ค่าความถูกต้อง (Accuracy) โดยทั่วไปใช้ในการประเมินประสิทธิภาแบบจำลองการจำแนกประเภท โดยจะวัดเปอร์เซ็นต์ความถูกต้องของชุดข้อมูลตัวอย่างจากข้อมูลทั้งหมด สูตรในการคำนวณ ดังสมการที่ 2.7

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} TN_i}{\sum_{i=1}^{n} TP_i + FP_i + FN_i + TN_i} \quad (2.7)$$

2) ค่าความแม่นยำ (Precision) เป็นการวัดที่ใช้ในการประเมินประสิทธิภาพของแบบจำลองการจำแนกประเภท โดยเฉพาะอย่างยิ่งสำหรับประเภทข้อมูลที่เป็นบวก โดยจะวัดสัดส่วนของผลบวกจริง (TP) ในทุกตัวอย่างที่แบบจำลองคาดการณ์ไว้ว่าเป็นบวก สูตรในการคำนวณ ดังสมการที่ 2.8

$$\text{Precision}_i = \frac{TP_i}{TP_i+FP_i} \quad (2.8)$$

ความแม่นยำมีประโยชน์ในสถานการณ์ที่ต้องลดผลบวกลวงให้เหลือน้อยที่สุด เช่น ในการวินิจฉัยทางการแพทย์ ความแม่นยำสูงหมายความว่าแบบจำลองสามารถระบุกรณีเชิงบวกได้อย่างถูกต้องด้วยความมั่นใจสูง และมีโอกาสน้อยที่จะทำการคาดคะเนในเชิงบวกทีผิดพลาด

3) ค่าความระลึก (Recall) คือ ค่าอัตราผลบวกจริงเป็นการวัดที่ใช้ในการประเมินประสิทธิภาพของแบบจำลองการจัดประเภท โดยเฉพาะอย่างยิ่งสำหรับชั้นที่เป็นบวก โดยจะวัดสัดส่วนของค่าบวกจริง (TP) จากค่าบวกจริงทั้งหมด สูตรในการคำนวณ ดังสมการที่ 2.9

$$Recall_i = \frac{TP_i}{TP_i+FN_i} \quad (2.9)$$

ค่าความระลึกมีประโยชน์ในสถานการณ์ที่จำเป็นต้องลดผลลบลวงให้เหลือน้อยที่สุด เช่น ในการวินิจฉัยโรค ค่าความระลึกสูงหมายความว่าโมเดลระบุตัวอย่างเชิงบวกในสัดส่วนที่มากได้อย่างถูกต้อง และมีโอกาสน้อยที่ตัวอย่างเชิงบวกจริงจะผิดพลาด

4) ค่าเฉลี่ยประสิทธิภาพโดยรวม (F1 score) เป็นการวัดที่ใช้ในการประเมินประสิทธิภาพของแบบจำลองการจัดประเภทที่รวมทั้งความแม่นยำและความระลึกเป็นคะแนนเดียว เป็นค่าเฉลี่ยสอดคล้องกันของความแม่นยำและความระลึก และพิจารณาทั้งผลบวกปลอมและผลลบลวง สูตรในการคำนวณ ดังสมการที่ 2.10

$$\text{F1}_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \quad (2.10)$$

เมื่อ

TP_i คือ จำนวนผลบวกที่แท้จริงสำหรับ $class_i$

TN_i คือ จำนวนผลลบที่แท้จริงสำหรับ $class_i$

FP_i คือ จำนวนผลบวกที่เป็นเท็จสำหรับ $class_i$

FN_i คือ จำนวนผลลบที่เป็นเท็จสำหรับ $class_i$

5) ค่าสัมประสิทธิ์สหสัมพันธ์แมตทิวส์ (Matthew's Correlation Coefficient: MCC) ค่าสัมประสิทธิ์สหสัมพันธ์ของ Matthew หรือค่าสัมประสิทธิ์ Phi โดยนิยมเรียกย่อว่า MCC ถูกคิดค้นโดย Brian Matthews ในปี 1975 [33] เป็นเครื่องมือทางสถิติที่ใช้สำหรับการประเมินโมเดล หน้าที่ของมันคือวัดความแตกต่างระหว่างค่าที่ทำนายกับค่าจริง ซึ่งค่า MCC จะมีค่าระหว่าง -1 ถึง +1 โดยที่ค่าสัมประสิทธิ์ +1 หมายถึงการทำนายได้ถูกต้อง 100% ค่าสัมประสิทธิ์ 0 หมายถึง

ไม่ต่างไปจากการทำนายแบบสุ่ม 50% และค่าสัมประสิทธิ์เท่ากับ -1 หมายถึงการทำนายไม่ถูกต้องทั้งหมด หรือการทำนายถูกที่ 0% จากการทำนายทั้งหมด ดังสมการที่ 2.11

$$\mathrm{MCC} = \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{(s^2 - \sum_k^K p_k^2) \times (s^2 - \sum_k^K t_k^2)}} \quad (2.11)$$

เมื่อ

k คือ จำนวน $class$

s คือ จำนวนตัวอย่างทั้งหมด

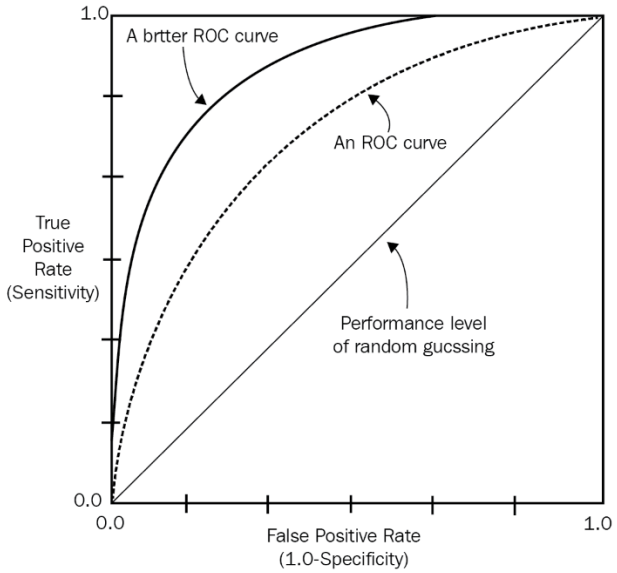
c คือ จำนวนตัวอย่างที่ทำนายได้ถูกต้อง

t_k คือ จำนวนครั้งที่คลาส k เกิดขึ้นจริง

p_k คือ จำนวนครั้งที่ทำนายคลาส k

6) Receiver Operating Characteristic (ROC) Curves [34, 35] หรือ เส้นโค้ง ROC เป็นการแสดงแผนภูมิของประสิทธิภาพของโมเดลการจำแนก โดยทำการสร้างแผนภูมิเส้นของอัตราบวกจริง (True Positive Rate: TPR) เทียบกับอัตราบวกปลอม (False Positive Rate: FPR) สำหรับค่าเกณฑ์ที่แตกต่างกัน TPR คือสัดส่วนของผลบวกที่เกิดขึ้นจริง ซึ่งจัดประเภทเป็นค่าบวกอย่างถูกต้อง ในขณะที่ FPR คือ สัดส่วนของค่าลบที่เกิดขึ้นจริงซึ่งจัดประเภทอย่างไม่ถูกต้องว่าเป็นค่าบวก

เส้นโค้ง ROC ถูกสร้างขึ้นโดยการพล็อต TPR บนแกน y และ FPR บนแกน x สำหรับค่าเกณฑ์ต่างๆ แต่ละจุดบนเส้นโค้งจะสอดคล้องกับค่าเกณฑ์เฉพาะ และเส้นโค้งจะแสดงการแลกเปลี่ยนระหว่าง TPR และ FPR สำหรับค่าเกณฑ์ต่าง ๆ เส้นที่มี TPR สูงกว่าสำหรับ FPR ที่กำหนดจะถือว่าโมเดลมีประสิทธิภาพดีกว่า



รูปที่ 2.8 กราฟเส้นโค้ง ROC

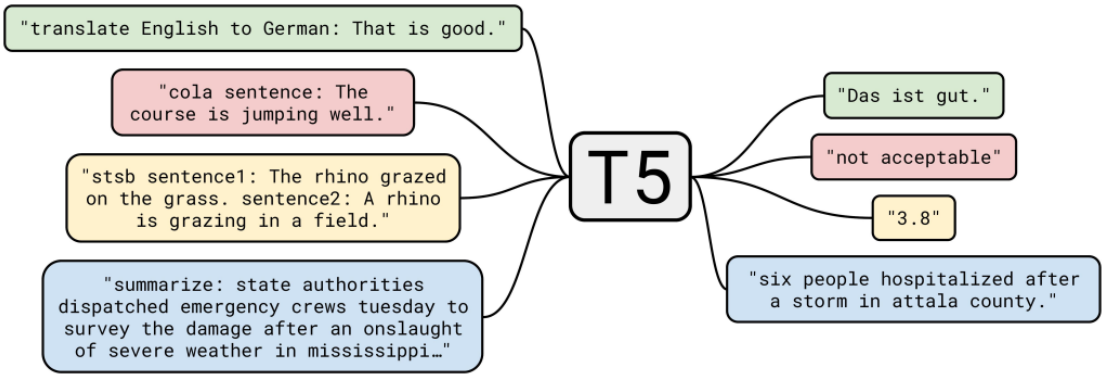
พื้นที่ใต้เส้นโค้ง ROC (Area Under the ROC Curve: AUC) เป็นการวัดประสิทธิภาพของโมเดลการจำแนกที่ใช้กันทั่วไป ค่า AUC อยู่ระหว่าง 0 ถึง 1 โดยที่คะแนน 1 หมายถึงตัวจำแนกประเภทมีความสมบูรณ์แบบ 0.5 หมายถึงตัวจำแนกประเภทแบบสุ่ม และ 0 หมายถึงตัวจำแนกประเภทที่ผิดทั้งหมด ยิ่งคะแนน AUC สูงเท่าใด ประสิทธิภาพของโมเดลก็จะยิ่งดีขึ้นเท่านั้น

2.6 โมเดลภาษา T5

โมเดล T5 (Text-To-Text Transfer Transformer) เป็นโมเดลภาษาขนาดใหญ่ที่พัฒนาโดยทีม Google Research ภายใต้โครงการ "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" [36] ซึ่งออกแบบมาให้สามารถใช้งานได้หลากหลายโดยการแปลงทุกงานให้เป็นปัญหาของ text generation หรือการสร้างข้อความ เช่น การแปลภาษา การสรุปเนื้อหา การตอบคำถาม และการจำแนกประเภทของข้อความ สำหรับ T5 ถูกพัฒนาโดยทีมงาน Google Research Brain Team นำโดย Colin Raffel และทีมงานจากโครงการ "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" ซึ่งมีจุดมุ่งหมายเพื่อสำรวจขีดจำกัดของการเรียนรู้แบบถ่ายโอน (Transfer Learning) สำหรับ NLP

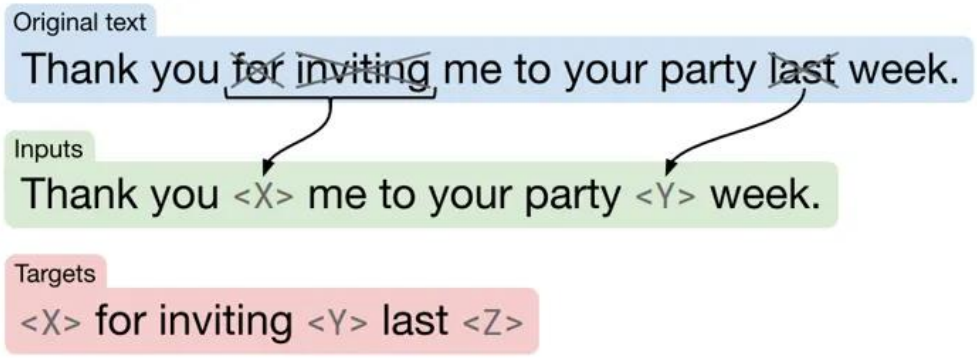
หลักการทำงานของ T5 ใช้โครงสร้าง Transformer Encoder-Decoder และแนวคิด "Text-to-Text" ซึ่งแปลงทุกงาน NLP ให้เป็นปัญหาการสร้างข้อความ โดยอินพุตและเอาต์พุตเป็น

ข้อความ เช่น การแปลภาษา การสรุปข้อความ และการตอบคำถาม โมเดลถูกฝึกโดยใช้ Self-Supervised Learning บนชุดข้อมูลขนาดใหญ่ C4 (Colossal Clean Crawled Corpus) และใช้กลยุทธ์ Pretraining + Fine-tuning โดยในขั้นตอน Pretraining โมเดลเรียนรู้โครงสร้างของภาษาผ่าน Masked Span Prediction ซึ่งลบข้อความบางส่วนแทนการใช้ [MASK] แบบ BERT และในขั้นตอน Fine-tuning โมเดลถูกปรับแต่งให้เหมาะกับงานเฉพาะ เช่น QA Summarization และ Sentiment Analysis



รูปที่ 2.9 แผนผังกรอบการทำงานของ T5 [36]

หลักการของ T5 ใช้แนวทาง Sequence-to-Sequence กับข้อมูลขนาดใหญ่ เพื่อให้สามารถสร้างประโยคใหม่โดยยังคงความหมายเดิม โครงสร้างของโมเดลประกอบด้วยสองส่วนหลัก ได้แก่ Encoder และ Decoder โดย Encoder ทำหน้าที่แปลงข้อความต้นฉบับเป็นลำดับของเวกเตอร์ ขณะที่ Decoder จะนำเวกเตอร์เหล่านั้นมาสร้างเป็นข้อความเป้าหมาย



รูปที่ 2.10 แผนผังของวัตถุประสงค์แบบจำลองพื้นฐานของ T5 [36]

จากรูปที่ 2.10 อธิบายภาพแสดงแผนผังการทำงานแบบพื้นฐานของ T5 ประกอบด้วยข้อความ ดังต่อไปนี้

Original Text หรือ ข้อความต้นฉบับ คือ ประโยคเริ่มต้นคือ "Thank you ~~for inviting~~ me to your party ~~last~~ week." ส่วนที่ถูกลบออกจากข้อความต้นฉบับถูกขีดฆ่า

Inputs (อินพุตที่ใช้ป้อนเข้าโมเดล) คือ ประโยคที่มีการแทนที่บางส่วนด้วย token พิเศษ <X> และ <Y>: "Thank you <X> me to your party <Y> week." ตำแหน่ง <X> และ <Y> คือ ช่องว่างที่โมเดลต้องเติมคำที่เหมาะสมเข้าไป

Targets (ค่าที่คาดหวังให้โมเดลสร้างขึ้น) คือ ส่วนที่โมเดลจะต้องเติมข้อมูลที่ขาดหายไปจากอินพุต โดยใช้เป้าหมาย ดังนี้ <X> for inviting <Y> last <Z> ซึ่งหมายถึงว่า

<X> ควรถูกแทนที่ด้วย "for inviting"

<Y> ควรถูกแทนที่ด้วย "last"

<Z> เป็นช่องว่างที่โมเดลอาจเติมคำให้เหมาะสม

ตารางที่ 2.11 แสดงตัวอย่างการทำงานของ T5 ในการประมวลผลภาษาธรรมชาติ (NLP) โดยใช้แนวคิด Text-to-Text ซึ่งแปลงทุกงานเป็นการสร้างข้อความใหม่ ตัวอย่างแรกคือ การแปลภาษา ซึ่งโมเดลสามารถรับอินพุตที่ระบุภาษาต้นทางและภาษาปลายทาง เช่น การแปลจากอังกฤษเป็นฝรั่งเศส และให้เอาต์พุตที่ถูกต้อง ตัวอย่างที่สองคือ การสรุปข้อความ ซึ่งโมเดลสามารถย่อเนื้อหายาวให้สั้นลงโดยคงความหมายสำคัญไว้ ส่วนตัวอย่างสุดท้ายคือ การตอบคำถาม (QA) ซึ่งโมเดลสามารถเข้าใจบริบทของคำถามและดึงคำตอบที่ถูกต้องออกมา โดยทั้งหมดนี้ใช้หลักการเดียวกันคือเปลี่ยนทุกงาน NLP ให้เป็นปัญหาของการสร้างข้อความ

ตารางที่ 2.11 ตัวอย่างการทำงานโมเดล T5

งานที่ใช้ T5	**อินพุต (Input)**	**เอาต์พุต (Output)**
การแปลภาษา	**translate English to French**: The book is on the table.	Le livre est sur la table.
การสรุปข้อความ	**summarize**: The economy is facing challenges due to inflation.	Inflation is causing economic challenges.
การตอบคำถาม (QA)	**question**: What is the capital of France? context: France's capital is Paris.	Paris

2.7 การเรียนรู้ของเครื่อง

การดำเนินงานจำแนกเอกสารข้อตวามสามารถใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Leaning) ในการสร้างโมเดลสำหรับการฝึกอบรมในการจำแนกเอกสารข้อความ ซึ่งอัลกอริทึมที่นิยมใช้งานมีอยู่หลาอัลกอริทึมด้วยกัน ซึ่งสามารถอธิบาย [29] ได้ดังนี้

2.7.1 Naive Bayes

นาอีฟเบย์ (Naive Bayes) [12, 37] เป็นอัลกอริธึมที่เรียบง่ายและมีประสิทธิภาพ มีความนิยมใช้กันอย่างแพร่หลายสำหรับการจำแนกเอกสารข้อความ สำหรับนาอีฟเบย์เป็นอัลกอริทึมแบบมีผู้สอน (Supervised) ซึ่งหมายความว่าต้องใช้ข้อมูลการฝึกอบรมที่มีป้ายกำกับเพื่อเรียนรู้วิธีจำแนกประเภทข้อมูลอินพุต ในการเรียนรู้แบบมีผู้สอน อัลกอริทึมจะเรียนรู้การคาดคะเนตามคู่อินพุตและเอาต์พุต โดยที่ข้อมูลอินพุตคือชุดของคุณลักษณะ และเอาต์พุตคือป้ายกำกับหรือตัวแปรเป้าหมาย ในกรณีของการจำแนกเอกสารข้อความ โดยทั่วไปข้อมูลที่ป้อนจะเป็นชุดของคำหรือวลีและผลลัพธ์จะเป็นหมวดหมู่หรือคลาสที่กำหนดไว้ล่วงหน้า

หลักการของนาอีฟเบย์ใช้หลักการคำนวณความน่าจะเป็น (Probability) ในการทำนายว่าข้อมูลที่สนใจอยู่กลุ่มไหนหรืออยู่คลาสไหน ดังสมการที่ 2.12

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.12}$$

โดยที่ $P(A|B)$ คือ ความน่าจะเป็นของ A ที่กำหนดด้วย B ขณะที่ $P(B|A)$ ความน่าจะเป็นของ B ที่กำหนดด้วย A โดยที่ $P(A)$ คือ ความน่าจะเป็นของเหตุการณ์ A และ $P(B)$ คือ ความน่าจะเป็นของเหตุการณ์ B

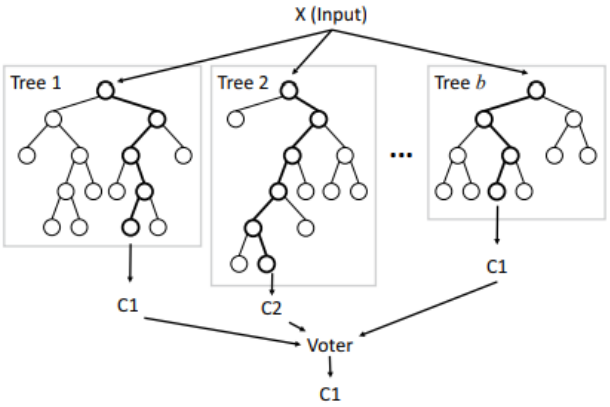
ปัจจุบันมีอัลกอริทึมที่ทำการปรับปรุงจากนาอีฟเบย์ ได้แก่ นาอีฟเบย์แบบ เกาเซียน (Gaussian Naïve Bayes) นาอีฟเบย์แบบเบอร์นูลี่ (Bernoulli Naïve Bayes) และพหุนาอีฟเบย์ (Multinomial Naïve Bayes) แต่สำหรับงานด้านการจำแนกเอกสารข้อความพหุนาอีฟเบย์ได้รับการยอมรับว่าให้ประสิทธิภาพที่ดีที่สุด อย่างไรก็ตามนาอีฟเบย์ยังมีข้อเสีย กรณีข้อมูลที่ถูกนำมาจำแนกกลุ่ม ไม่มีฟีเจอร์ที่สอดคล้องกับข้อมูลชุดสอนเลย อัลกอริทึมนี้อาจจะประสบปัญหาเรื่องความน่าจะเป็นที่มีค่าเป็นศูนย์ ทำให้ข้อมูลนั้น ๆ ไม่สามารถถูกทำนายกลุ่มหรือคลาสของข้อมูลได้

2.7.2 Support Vector Machine

ซัพพอร์ตเวกเตอร์แมชชีน (Support Vector Machine: SVM) [12, 38, 39] เป็นเทคนิคการเรียนรู้ด้วยเครื่องจักรที่ใช้ในการแยกประเภท (classification) และการถดถอย (regression) จุดมุ่งหมายหลักของ SVM คือการหาพื้นที่แบ่ง (hyperplane) ที่ดีที่สุดที่สามารถแยกข้อมูลออกเป็นสองหรือหลายกลุ่มได้อย่างชัดเจน เพื่อให้การทำนายหรือการแบ่งประเภทในอนาคตทำได้ง่ายและแม่นยำขึ้น ความพิเศษของ SVM อยู่ที่การใช้เทคนิคที่เรียกว่า Kernel Trick เป็นเทคนิคสำคัญใน SVM ที่ช่วยให้สามารถแยกข้อมูลที่ไม่สามารถแยกได้เชิงเส้น (linearly separable) ในพื้นที่มิติต่ำไปยังพื้นที่มิติสูงกว่าที่ข้อมูลนั้นสามารถแยกได้เชิงเส้น โดยไม่ต้องทำการคำนวณในพื้นที่มิติสูงโดยตรง ซึ่งปกติแล้วจะเป็นไปไม่ได้หรือมีความซับซ้อนมากเนื่องจากมิติที่สูง ในการทำงานปกติของ SVM หากข้อมูลไม่สามารถแยกออกจากกันเชิงเส้นได้ในพื้นที่มิติปัจจุบันจำเป็นต้องยกพื้นที่ข้อมูลนั้นไปยังพื้นที่มิติที่สูงขึ้นที่ข้อมูลอาจแยกออกจากกันได้ การทำเช่นนี้โดยตรงอาจต้องการการคำนวณที่มากและซับซ้อน แต่ Kernel Trick ช่วยลดความซับซ้อนนั้นลงโดยการใช้ฟังก์ชั่นเคอร์เนล (kernel function) ฟังก์ชั่นเคอร์เนลคำนวณผลคูณภายในของสองจุดข้อมูลในพื้นที่มิติสูงโดยไม่ต้องทำการแปลงข้อมูลไปยังพื้นที่นั้นโดยตรง. มันทำให้การคำนวณในมิติสูงเป็นไปได้โดยการคำนวณในมิติที่ต่ำกว่าแทน ฟังก์ชั่นเคอร์เนลที่ใช้กันอย่างแพร่หลาย ได้แก่ เคอร์เนลเชิงเส้น (linear), เคอร์เนลพหุนาม (polynomial) เคอร์เนล RBF (Radial Basis Function) และเคอร์เนลซิกมอยด์ (sigmoid) อย่างไรก็ตาม SVM ยังมีข้อเสีย [29] ถ้าในกรณีข้อมูลชุดสอนมีขนาดใหญ่จะใช้เวลาในการเรียนรู้โมเดลนาน

2.7.3 Random Forest

ป่าสุ่ม (Random Forest) [40, 41] เป็นอัลกอริทึมประเภทวิธีร่วมตัดสินใจ (Ensemble) ที่รวมแผนผังการตัดสินใจหลาย ๆ แบบเพื่อทำการทำนายขั้นสุดท้าย เป็นอัลกอริทึมอเนกประสงค์ที่สามารถใช้กับจำแนกเอกสารข้อความ ป่าสุ่มมีวิธีการทำงานด้วยการสุ่มเลือกฟีเจอร์ที่แตกต่างกันเพื่อทำหน้าที่เป็นตัวแทยของในแต่ละกลุ่มข้อมูลย่อยด้วย จากนั้นจะนำฟีเจอร์ย่อยมาสร้างโมเดลการจำแนกข้อมูลด้วยอัลกอริทึมต้นไม้ตัดสินใจหลายต้น หากผลลัพธ์การทำนายของต้นไม้ตัดสินใจแต่ละต้นเป็นกลุ่มใดมากที่สุดจะให้เอกสารข้อความที่กำลังพิจารณาอยู่ในกลุ่มนั้น อย่างไรก็ตามอัลกอริทึมยังมีข้อเสีย [29] คือ เป็นอัลกอริทึมที่ต้องการทรัพยากรของคอมพิวเตอร์และเวลาในการประมวลผลมาก และถึงแม้จะเปลี่ยนแปลงข้อมูลในการฝึกเล็กน้อย ผลลัพธ์ของการวิเคราะห์ด้วยอัลกอริทึมป่าสุ่มอาจจะมีการเปลี่ยนแปลง แผนภาพการทำงานของป่าสุ่ม [42] ดังรูป



รูปที่ 2.11 ตัวอย่างป่าสุ่ม [42]

2.7.4 Logistic Regression

การถดถอยโลจีสติก (Logistic Regression) [39, 43, 44] คืออัลกอริทึมการเรียนรู้แบบมีผู้สอนที่ใช้ในงานการจำแนกประเภทและมีความนิยมใช้กันอย่างกว้างขวาง โดยเฉพาะในการจำแนกเอกสารข้อความหรือการจำแนกหมวดหมู่ต่างๆ หลักการของการถดถอยโลจีสติกคือการทำนายความน่าจะเป็นของผลลัพธ์แบบไบนารี ในการทำนาย มันใช้ฟังก์ชั่นลอจิต (logit function) ที่เปลี่ยนค่าของตัวแปรอินพุต ที่อาจจะเป็นคำหรือวลีในข้อความ หรือคุณสมบัติอื่นๆ ของข้อมูล เป็นความน่าจะเป็นของการเป็นสมาชิกในหนึ่งในสองคลาสที่กำหนด ตัวแปรอินพุตเหล่านี้มักจะถูกแปลงเป็นน้ำหนักทางสถิติผ่านกระบวนการฝึกอบรมโมเดลด้วยข้อมูลที่มีการจำแนกประเภทไว้ล่วงหน้า

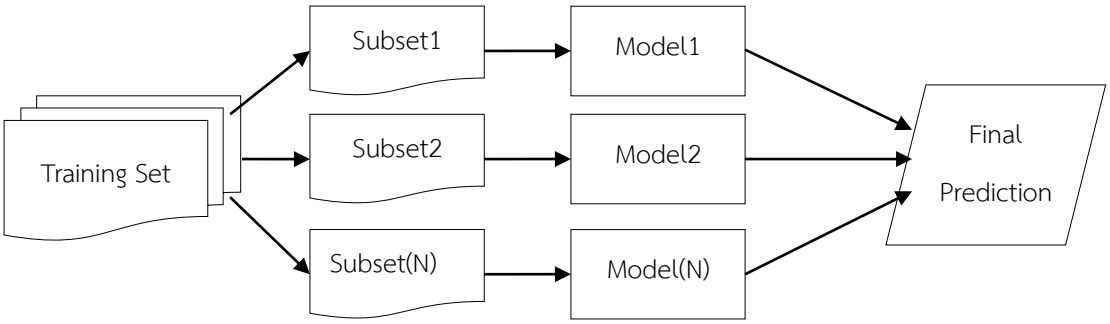
การถดถอยโลจีสติกมีความเหมาะสมในหลายสถานการณ์เนื่องจากเป็นโมเดลที่เข้าใจง่ายและใช้งานได้กับปัญหาที่มีความซับซ้อนน้อย แต่อย่างไรก็ตาม มันอาจมีข้อจำกัดเมื่อใช้กับข้อมูลที่มีความซับซ้อนสูงหรือในกรณีที่ข้อมูลที่มีให้สำหรับการฝึกอบรมนั้นมีน้อยเกินไป ซึ่งอาจทำให้โมเดลมีการทำนายที่ไม่แม่นยำหรือมีความทนทานต่อข้อมูลใหม่ต่ำ นอกจากนี้ หากข้อมูลมีความสัมพันธ์ที่ไม่เป็นเชิงเส้นระหว่างตัวแปรอินพุตและเอาต์พุต การถดถอยโลจีสติกอาจไม่ให้ผลลัพธ์ที่ดีเท่าที่ควร เนื่องจากมันถูกออกแบบมาสำหรับการทำนายที่เป็นเชิงเส้น การใช้เทคนิคการเรียนรู้แบบอื่นๆ หรือการปรับเปลี่ยนข้อมูลอาจจำเป็นเพื่อเพิ่มประสิทธิภาพในกรณีเหล่านี้

2.7.5 Ensemble Learning

การเรียนรู้แบบกลุ่ม (Ensemble learning) [45-47] เป็นเทคนิคในการเรียนรู้ของเครื่องที่ใช้การรวมโมเดลหลาย ๆ ตัวเข้าด้วยกันเพื่อปรับปรุงประสิทธิภาพในการทำนายหรือการจำแนกประเภทข้อมูล (classification) และการทำนายค่าต่อเนื่อง (regression) วิธีการนี้มีพื้นฐานจาก

ความคิดที่ว่าการรวมกันของหลายๆ โมเดลที่ดีหรือแม้แต่โมเดลที่ให้ผลลัพธ์ที่ไม่ดีมากหากรวมโมเดลเหล่านี้ด้วยวิธีที่ถูกต้อง จะสามารถสร้างโมเดลที่มีประสิทธิภาพสูงกว่าการใช้โมเดลเดี่ยวๆ ได้ สำหรับเทคนิคการเรียนรู้แบบกลุ่มสามารถแบ่งได้เป็น 4 ประเภท

Bagging หรือ Bootstrap Aggregating [48] เป็นเทคนิคในวิธีการเรียนรู้แบบรวม (Ensemble Learning) ที่ใช้เพื่อเพิ่มความแม่นยำและลดความแปรปรวนของโมเดลการเรียนรู้ของเครื่อง (Machine Learning) โดยกระบวนการทำงานของ Bagging เริ่มจากการสุ่มตัวอย่างข้อมูลย่อยจากชุดข้อมูลหลักหลายครั้ง (Bootstrap Sampling) โดยที่แต่ละตัวอย่างสามารถถูกเลือกซ้ำได้ จากนั้นทำการสร้างโมเดลเรียนรู้หลายตัวบนชุดข้อมูลย่อยที่แตกต่างกันเหล่านี้ และสุดท้ายทำการรวมผลลัพธ์จากโมเดลทั้งหมดเพื่อให้ได้ผลลัพธ์ที่มีความเสถียรและแม่นยำมากขึ้น ซึ่งโดยทั่วไปสำหรับปัญหาการจำแนกประเภท (Classification) การรวมผลลัพธ์สามารถทำได้โดยใช้วิธีการโหวตเสียงข้างมาก (Majority Voting) หรือสำหรับปัญหาการพยากรณ์ค่าเชิงตัวเลข (Regression) สามารถใช้ค่าเฉลี่ยของผลลัพธ์จากแต่ละโมเดลเป็นคำตอบสุดท้าย เทคนิค Bagging นี้ถูกนำมาใช้ในอัลกอริธึมที่เป็นที่นิยม เช่น Random Forest ซึ่งเป็นโมเดลที่สร้างขึ้นจากการรวมต้นไม้ตัดสินใจหลายต้นเข้าด้วยกัน ทำให้สามารถลดความโน้มเอียงของโมเดลเดี่ยวและเพิ่มความสามารถในการทำนายให้ดียิ่งขึ้น



รูปที่ 2.12 ขั้นตอนของ Bagging

Boosting เป็นเทคนิคการเรียนรู้แบบรวม (Ensemble Learning) ที่มุ่งเน้นการปรับปรุงประสิทธิภาพของโมเดลโดยการลดข้อผิดพลาดอย่างเป็นขั้นเป็นตอน ซึ่งแตกต่างจาก Bagging ที่สร้างโมเดลหลายตัวแบบอิสระต่อกัน Boosting จะสร้างโมเดลในลำดับต่อเนื่อง โดยที่โมเดลถัดไปได้รับการฝึกให้เรียนรู้จากข้อผิดพลาดของโมเดลก่อนหน้า โดยมีหลักการทำงานดังนี้

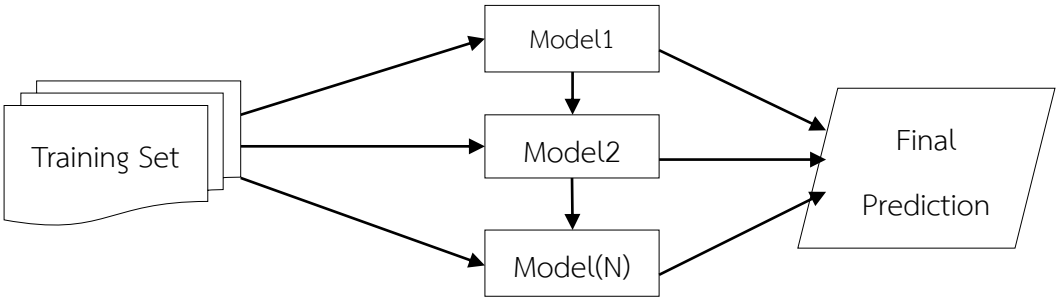
1) การกำหนดน้ำหนักให้กับตัวอย่างข้อมูล ในรอบแรก Boosting จะใช้ชุดข้อมูลทั้งหมดเพื่อฝึกโมเดลพื้นฐาน (Weak Learner) เช่น ต้นไม้ตัดสินใจ (Decision Tree) ที่มีความซับซ้อนต่ำ จากนั้นจะทำการประเมินผลลัพธ์ของโมเดล

2) การระบุข้อผิดพลาด หลังจากการทำนาย โมเดลจะระบุว่าตัวอย่างข้อมูลใดถูกทำนายผิด ซึ่งข้อมูลที่ถูกทำนายผิดจะถูกให้ความสำคัญมากขึ้นโดยการเพิ่มน้ำหนัก เพื่อให้โมเดลถัดไปให้ความสนใจกับตัวอย่างเหล่านี้มากขึ้น

3) การสร้างโมเดลถัดไป โมเดลถัดไปจะถูกฝึกโดยให้ความสำคัญกับตัวอย่างที่โมเดลก่อนหน้าทำนายผิด และกระบวนการนี้จะทำซ้ำหลายรอบ โดยแต่ละโมเดลใหม่จะถูกปรับให้สามารถแก้ไขข้อผิดพลาดของโมเดลก่อนหน้าได้ดียิ่งขึ้น

4) การรวมผลลัพธ์ของโมเดลทั้งหมด หลังจากสร้างโมเดลหลายตัว Boosting จะรวมผลลัพธ์ของโมเดลทั้งหมดเข้าด้วยกัน โดยใช้กลยุทธ์การให้คะแนนหรือน้ำหนักกับแต่ละโมเดล โดยโมเดลที่มีความแม่นยำสูงจะมีน้ำหนักมากกว่าตัวที่มีความแม่นยำน้อยกว่า

Boosting มีหลายอัลกอริธึมที่ได้รับความนิยม เช่น AdaBoost (Adaptive Boosting) Gradient Boosting XGBoost (Extreme Gradient Boosting) เป็นต้น



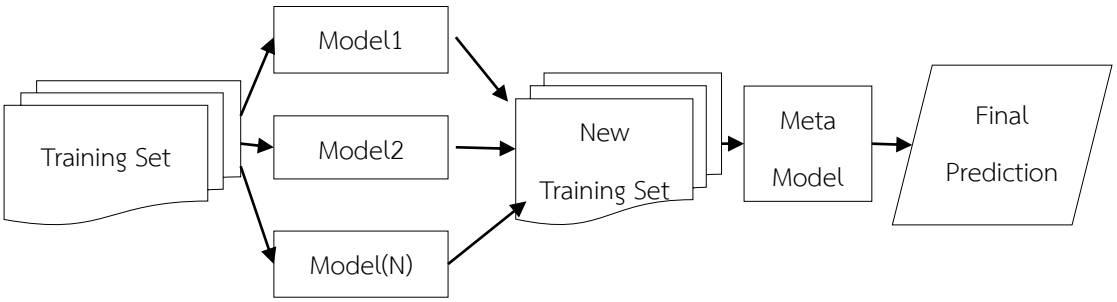
รูปที่ 2.13 ขั้นตอนของ Bootsting

Stacking [46] เป็นเทคนิคการเรียนรู้แบบรวม (Ensemble Learning) ที่มีหลักการทำงานโดยการใช้โมเดลหลายตัว (Base Models หรือ First-Level Models) มาทำนายผลลัพธ์ จากนั้นนำผลลัพธ์ที่ได้จากโมเดลเหล่านั้นไปเป็นข้อมูลอินพุตให้กับโมเดลอีกชั้นหนึ่ง ซึ่งเรียกว่า Meta-Model หรือ Blender เพื่อเรียนรู้และปรับปรุงการรวมผลลัพธ์จากโมเดลระดับแรกให้ได้ผลลัพธ์ที่แม่นยำที่สุด หลักการทำงานของ Stacking เป็นสองชั้นหลัก คือ

1) Base Models หรือ ชั้นที่ 1 เป็นการใช้โมเดลหลายตัว เช่น Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, SVM เป็นต้นโมเดลแต่ละตัวเรียนรู้จากชุดข้อมูลฝึก (Training Data) และให้ผลลัพธ์เป็นค่าทำนายผลลัพธ์ของโมเดลแต่ละตัวอาจเป็นค่าเฉพาะกลุ่มหรือค่าต่อเนื่องขึ้นอยู่กับประเภทของปัญหา

2) Meta-Model หรือ ชั้นที่ 2 นำผลลัพธ์จาก Base Models มาเป็นอินพุตชุดใหม่ แล้วใช้โมเดลอีกตัวหนึ่งเพื่อเรียนรู้วิธีรวมผลลัพธ์เหล่านั้นให้มีความแม่นยำมากที่สุด โมเดลที่ใช้เป็น Meta-

Model อาจเป็นโมเดลที่เรียบง่าย เช่น Logistic Regression, Decision Tree, Neural Networks หรือ Gradient Boosting Meta-Model ทำหน้าที่เรียนรู้ว่าน้ำหนักของผลลัพธ์จากแต่ละ Base Model ควรเป็นอย่างไร เพื่อให้ผลลัพธ์สุดท้ายออกมาดีที่สุด



รูปที่ 2.14 ขั้นตอนของ Stacking

Voting [49] เป็นเทคนิคหนึ่งในแนวทางการเรียนรู้แบบรวม (Ensemble Learning) ที่ใช้หลักการรวมผลลัพธ์จากโมเดลหลายตัวเพื่อตัดสินใจผลลัพธ์สุดท้ายอย่างแม่นยำยิ่งขึ้น วิธีนี้ช่วยลดข้อผิดพลาดของโมเดลเดี่ยวโดยใช้หลักการรวมมุมมองจากโมเดลหลายตัวเข้าด้วยกัน ซึ่งมีแนวทางหลักอยู่สองแบบ ได้แก่ Hard Voting และ Soft Voting โดย Hard Voting เป็นการให้แต่ละโมเดลทำการพยากรณ์ผลลัพธ์และใช้วิธีการโหวตเสียงข้างมาก (Majority Voting) เพื่อเลือกผลลัพธ์ที่ได้รับการทำนายมากที่สุด เช่น หากมีโมเดล 3 ตัว และ 2 ใน 3 โมเดลทำนายว่าเป็น Class A โมเดลสุดท้ายจะให้ผลเป็น Class A ตามเสียงข้างมาก ส่วน Soft Voting จะคำนวณค่าเฉลี่ยจากค่าความน่าจะเป็น (Probability) ของแต่ละโมเดลและเลือกผลลัพธ์ที่มีค่าความน่าจะเป็นสูงสุด เช่น ใช้ในโมเดลที่ให้ค่าความน่าจะเป็นของการทำนาย เช่น Logistic Regression หรือ Random Forest ทั้งนี้การใช้ Voting ช่วยให้ได้ผลลัพธ์ที่มีความแม่นยำและมีเสถียรภาพมากขึ้น โดยลดผลกระทบจากโมเดลที่ให้ผลลัพธ์ผิดพลาดบางตัว และช่วยเพิ่มความสามารถในการทำนายโดยใช้ข้อดีของโมเดลที่มีความแตกต่างกัน

2.8 การเรียนรู้เชิงลึก

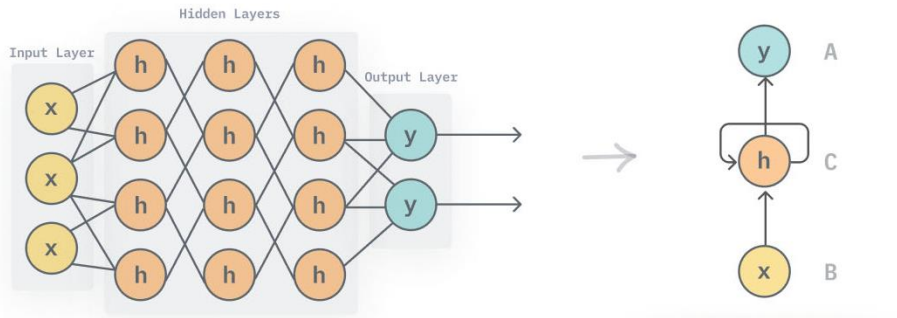
การเรียนรู้เชิงลึก (Deep Learning) [50] เป็นวิธีการหนึ่งในปัญญาประดิษฐ์ที่สอนให้คอมพิวเตอร์ประมวลผลข้อมูลในลักษณะที่ได้รับแรงบันดาลใจจากสมองมนุษย์ โมเดลการเรียนรู้เชิงลึกสามารถจดจำรูปแบบที่ซับซ้อนในข้อมูลประเภท รูปภาพ ข้อความ เสียง และข้อมูลอื่น ๆ เพื่อสร้างข้อมูลเชิงลึกและการคาดคะเนที่แม่นยำ คุณสามารถใช้วิธีการเรียนรู้เชิงลึกเพื่อทำงานโดย

อัตโนมัติซึ่งโดยทั่วไปต้องใช้สติปัญญาของมนุษย์ เช่น การอธิบายรูปภาพหรือการถอดเสียงไฟล์เสียงเป็นข้อความ

การเรียนรู้เชิงลึกในงานจำแนกเอกสารข้อความ มีความสามารถในการเรียนรู้และแสดงความสัมพันธ์ที่ซับซ้อนระหว่างคำและวลีในข้อมูลข้อความ ทำให้สามารถจับลักษณะเฉพาะของข้อมูลได้ดีกว่าโมเดลการเรียนรู้ของเครื่องแบบเดิมในบางกรณี ปัจจุบันมีเทคนิคต่าง ๆ ของการเรียนรู้เชิงลึกที่ได้รับความนิยมและสามารถประยุกต์ใช้กับงานจำแนกเอกสารข้อความ ดังต่อไปนี้

2.8.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [51] เป็นประเภทของโครงข่ายประสาทเทียมที่สามารถจำลองข้อมูลตามลำดับ สามารถเก็บสถานะก่อนหน้าเพื่อนำไปประมวลผลต่อสำหรับข้อมูลลำดับถัดไป ข้อมูลประเภทนี้ เช่น ข้อความ โดยที่วิธี RNN จะมี "หน่วยความจำ" ที่สามารถจับการอ้างอิงระหว่างคำในประโยคหรือเอกสารได้ ซึ่งแนวคิดหลักเบื้องหลังของ RNN คือ ใช้เอาต์พุตจากขั้นตอนเวลาก่อนหน้าเป็นอินพุตไปยังขั้นตอนเวลาปัจจุบัน ซึ่งช่วยให้เครือข่ายสามารถรักษาหน่วยความจำของอินพุตที่ผ่านมาและใช้ข้อมูลนี้เพื่อแจ้งเอาต์พุตปัจจุบัน

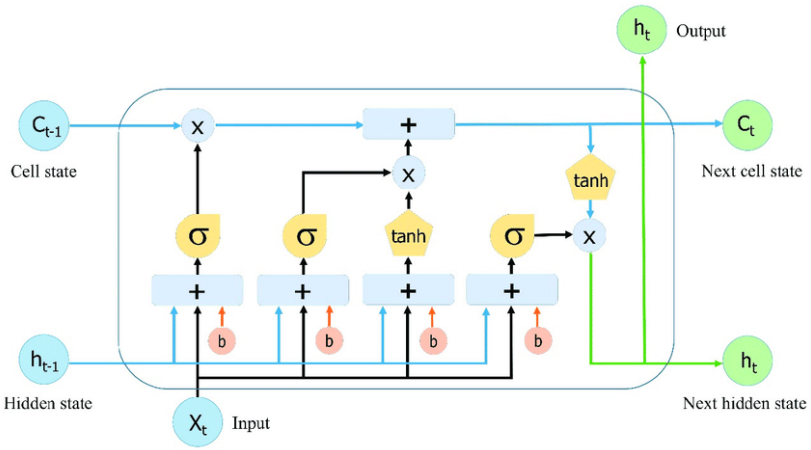


รูปที่ 2.15 ตัวอย่างสถาปัตยกรรมอย่างง่ายของ RNN

2.8.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) [52, 53] เป็นสถาปัตยกรรมการเรียนรู้เชิงลึกประเภทหนึ่ง LSTM ถูกพัฒนาต่อจาก RNN ซึ่งเป็นเทคนิคด้านโครงข่ายประสาทเทียมที่ออกแบบมาเพื่อประมวลผลข้อมูลตามลำดับ เช่น ข้อมูลข้อความ ข้อมูลเสียง เป็นต้น สำหรับ LSTM ประกอบด้วยบล็อกหน่วยความจำหลายชั้น ประกอบด้วยเซลล์สเตท (Cell State) เกตอินพุต (Input Gate) เกตเอาต์พุต (Output Gate) และเกตลืม (Forget Gate) เซลล์สเตทมีหน้าที่จัดเก็บข้อมูลจากขั้นตอนของเวลาก่อนหน้า ในขณะที่แต่ละเกตทำหน้าที่ควบคุมการไหลของข้อมูลเข้าและออกจาก

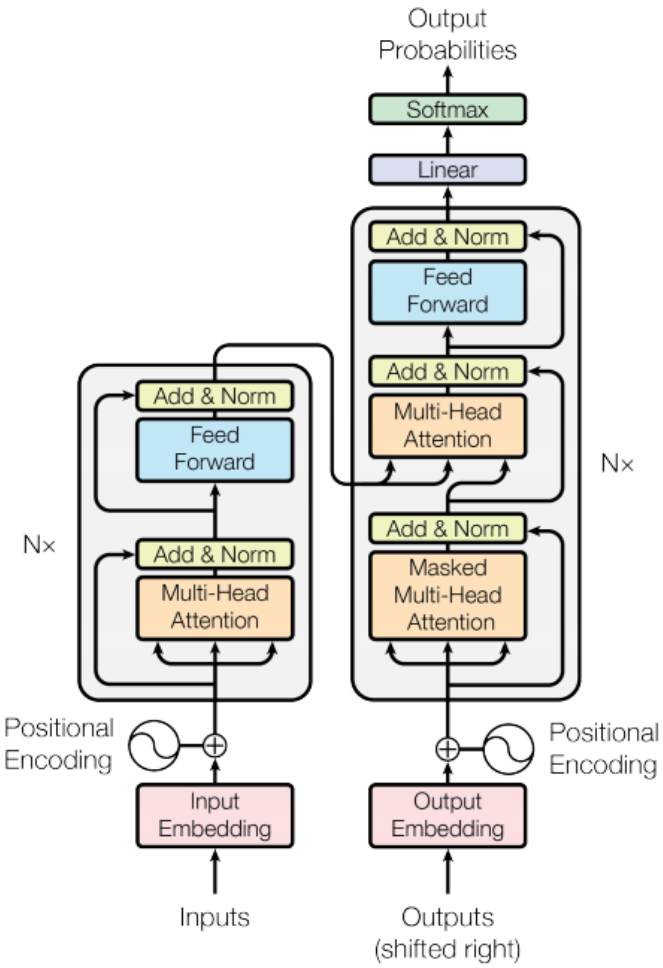
เซลล์หน่วยความจำ โดยสรุป LSTM เป็นสถาปัตยกรรมการเรียนรู้เชิงลึกที่ทรงพลังซึ่งสามารถใช้สร้างแบบจำลองข้อมูลตามลำดับ และมีประโยชน์อย่างยิ่งสำหรับงานที่ต้องใช้หน่วยความจำระยะยาว



รูปที่ 2.16 ตัวอย่างโครงสร้าง LSTM

2.8.3 Transformer-based Language Model

โมเดลภาษาแบบทรานฟอเมอร์ (Transformer-based Language Model) [54] เป็นสถาปัตยกรรมหนึ่งของการเรียนรู้เชิงลึก ที่ใช้กลไกการสนใจตนเอง โดยให้น้ำหนักแตกต่างกันตามความสำคัญของข้อมูลนำเข้าแต่ละส่วน ซึ่งถูกพัฒนาโดยนักวิจัยทีม Google Brain เพื่อแก้ไขข้อบกพร่องของโมเดล Recurrent Neural Network (RNN) และ Convolutional Neural Networks (CNN) ซึ่งอาศัยการฝึกโมเดลแบบคู่ขนานจากข้อมูลขนาดใหญ่เพื่อให้ได้ระบบฝึกไว้ล่วงหน้า (pre-trained) โมเดลที่สำคัญ เช่น BERT (Bidirectional Encoder Representations from Transformers), GPT (Generative Pre-trained Transformer) และ RoBERTa เป็นต้น



รูปที่ 2.17 Transformer Model [54]

จากรูป 2.17 แสดงโมเดล Transformer ซึ่งประกอบด้วย 2 ฝั่ง โดยฝั่งซ้ายจะเป็น Encoder ที่รับ Input sequence และฝั่งขวา คือ Decoder ที่รับ Output sequence

องค์ประกอบหลักของสถาปัตยกรรม Transformer ประกอบด้วย ดังนี้

1) Input Embedding Layer เป็นขั้นตอนแรกของสถาปัตยกรรม โดยลำดับของการป้อนข้อมูลของคำ จะถูกแปลงจากคำเป็นตัวเลข หรือ (Word embeddings) ซึ่งจับความหมายของคำ

2) Multi-Head Self-Attention Mechanism เป็นขั้นตอนที่ทรานฟอเมอร์ใช้กลไกการเอาใจใส่ตนเองเพื่อคำนวณความสัมพันธ์ระหว่างองค์ประกอบทั้งหมดในลำดับอินพุต สิ่งนี้ทำได้โดยใช้กลไกการให้ความสนใจแบบหลายหัวข้อซึ่งจะคำนวณคะแนนความสนใจระหว่างองค์ประกอบทั้งหมดแบบคู่ขนาน แล้วนำผลลัพธ์มาต่อกัน

3) Position-wise Feed-Forward Network คือ เอาต์พุตของกลไกการให้ความสนใจแบบหลายหัวจะถูกป้อนเข้าสู่เครือข่าย feed-forward ที่เชื่อมต่ออย่างสมบูรณ์ ซึ่งจะประมวลผลข้อมูลต่อไป

4) Normalization Layers คือ เอาต์พุตของเครือข่าย feed-forward จะถูกทำให้เป็นมาตรฐานโดยใช้ Layer Normalization ซึ่งช่วยป้องกันการไล่ระดับสีที่หายไปและรับประกันความเสถียรของโมเดลระหว่างการฝึก

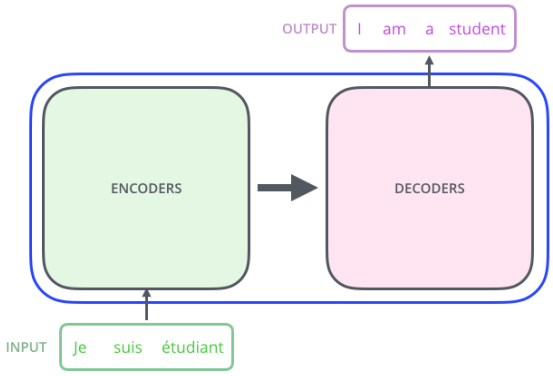
5) Output Layer คือ เอาท์พุตของ Normalization Layers ถูกใช้เพื่อคาดคะเน เช่น จำแนกข้อความที่กำหนดหรือสร้างการแปล

องค์ประกอบเหล่านี้ซ้อนกันหลายชั้นเพื่อสร้างสถาปัตยกรรม ซึ่ง Transformer ได้รับการออกแบบมาให้สามารถขนานกันได้สูงและอนุญาตให้ใช้งาน GPU ได้อย่างมีประสิทธิภาพ ทำให้สามารถฝึกโมเดลกับข้อมูลจำนวนมากได้

สามารถอธิบายขั้นตอนของทรานฟอเมอร์ด้วยโมเดลการแปลภาษา [55] โดยมองตัวโมเดลเป็นกล่องหนึ่งกล่องที่รับข้อความต้นฉบับแล้วทำการแปลภาษาเป็นอีกภาษาในส่วนเอาต์พุต

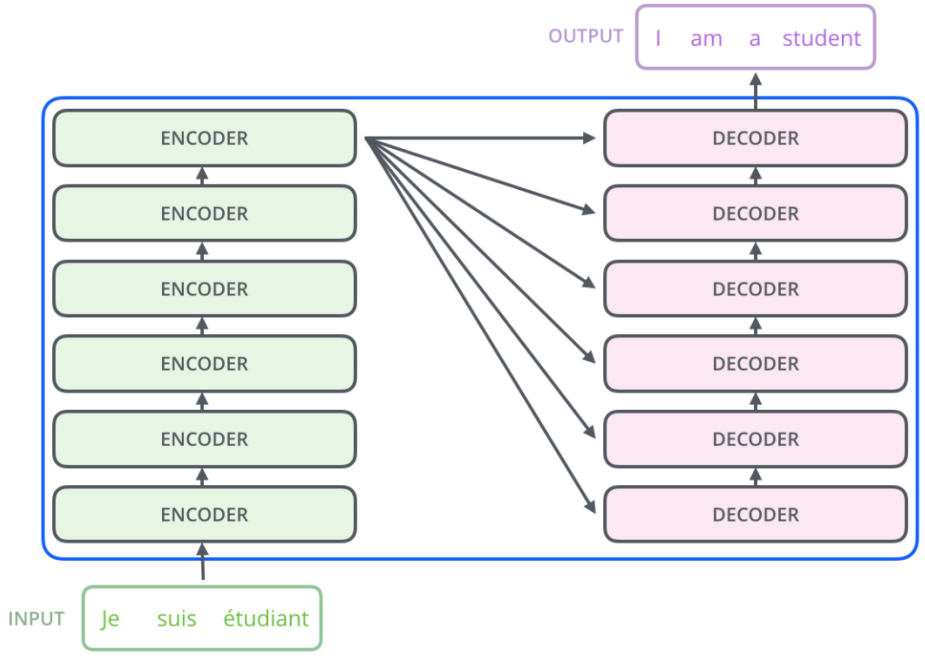
รูปที่ 2.18 ตัวอย่างมุมมองระดับบนของทรานฟอเมอร์ [55]

จากรูปที่ 2.18 ในกล่องของการแปลภาษาประกอบด้วยการทำงานหลัก 2 ส่วน คือ การเข้ารหัส (Encoders) และการถอดรหัส (Decoders)



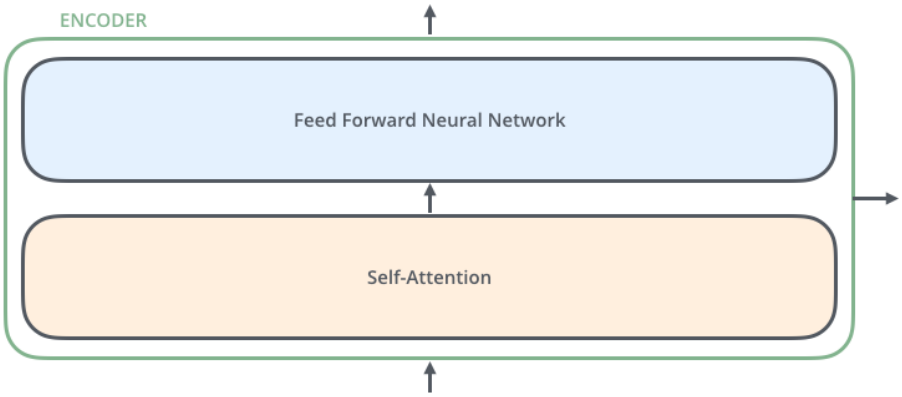
รูปที่ 2.19 ตัวอย่างส่วนทำงาน Encoders และ Decoders [55]

จากรูปที่ 2.19 ส่วนประกอบของการเข้ารหัสและการถอดรหัสนั้นจะประกอบด้วยส่วนย่อย ๆ อีก กล่าวคือ มีตัวเข้ารหัสหลายตัวและตัวถอดรหัสหลายตัว ทั้งนี้ต้องมีจำนวนเท่ากัน



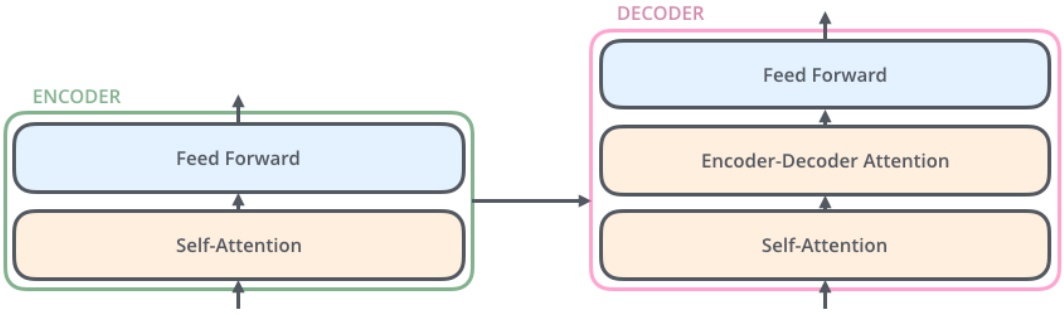
รูปที่ 2.20 จำนวนตัว Encoders และ Decoders ภายใน [55]

จากรูปที่ 2.20 ในตัวเข้ารหัสจะมีโครงสร้างเหมือนกันทั้งหมด แต่ละตัวจะมีกระบวนการทำงานอีกสองส่วน คือ เครือข่ายฟีดฟอร์เวิร์ด (Feed Forward Neural Network) และการเอาใจใส่ตนเอง (Self-Attention)



รูปที่ 2.21 ส่วนประกอบภายใน Encoder [55]

จากรูปที่ 2.21 อินพุตของตัวเข้ารหัสจะไหลผ่านชั้นเอาใจใส่ตนเอง (Self-Attention Layer) ซึ่งเป็นชั้นที่ช่วยให้ตัวเข้ารหัสดูคำอื่นๆ ในประโยคอินพุตในขณะที่เข้ารหัสคำเฉพาะจะพิจารณาการเอาใจใส่ตนเองให้ละเอียดยิ่งขึ้นในภายหลัง ต่อมาเอาต์พุตของชั้นเอาใจใส่ตนเอง จะถูกส่งไปยังโครงข่ายประสาทเทียมแบบฟีดฟอร์เวิร์ด เครือข่ายฟีดฟอร์เวิร์ดเดียวกันนี้ถูกนำไปใช้กับแต่ละตำแหน่งโดยอิสระ ในส่วนของฝั่งตัวถอดรหัสจะมีทั้งชั้นเอาใจใส่ตนเอง และชั้นเครือข่ายฟีดฟอร์เวิร์ด แต่ระหว่างสองชั้นนี้จะมีชั้นที่ทำหน้าที่สนใจของฝั่งตัวถอดรหัส (Encoder-Decoder Attention) เพื่อทำการโฟกัสไปในส่วนที่เกี่ยวข้องกับประโยคอินพุต ดังรูปที่ 2.22



รูปที่ 2.22 การทำงานระหว่าง Encoder-Decoder [55]

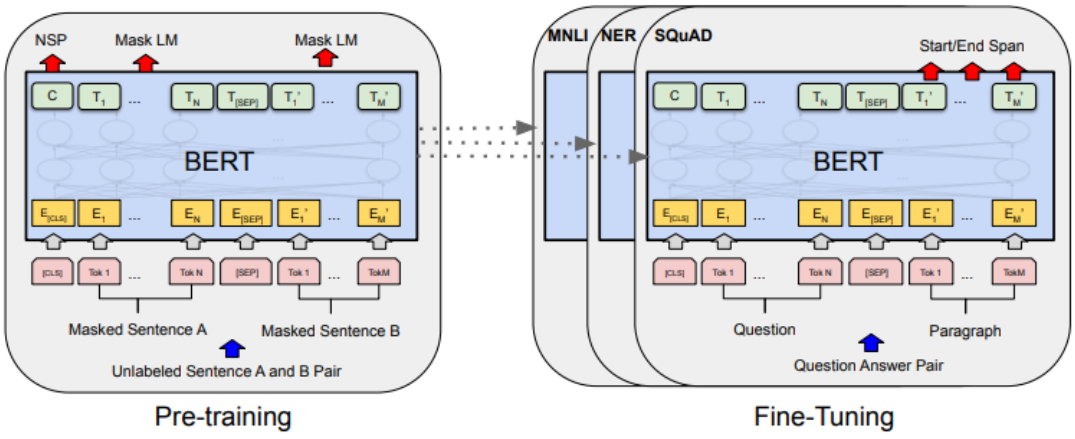
2.8.4 Bidirectional Encoder Representations from Transformers

BERT (Bidirectional Encoder Representations from Transformers) [56] เป็นโมเดลภาษาที่ถูกพัฒนาโดย Google AI และได้รับความสนใจอย่างกว้างขวางในงานวิจัยด้านการประมวลผลภาษาธรรมชาติ ซึ่งโมเดล BERT-Base เป็นหนึ่งในโครงสร้างที่ได้รับความนิยม โดยมีขนาดที่สมดุลระหว่างประสิทธิภาพและความต้องการด้านทรัพยากรคำนวณ ด้วยสถาปัตยกรรมที่

ประกอบด้วย 12 encoder layers, 768 hidden size, และ 110 ล้านพารามิเตอร์ ทำให้สามารถนำไปประยุกต์ใช้กับงาน NLP ได้หลากหลาย เช่น การจัดหมวดหมู่ข้อความ (Text Classification) การตอบคำถาม (Question Answering) และการสรุปผลข้อความ (Text Summarization) โครงสร้างและแนวคิดหลักของ BERT-Base ถูกออกแบบให้สามารถเรียนรู้ข้อมูลในลักษณะ Bidirectional Contextual Representations ซึ่งแตกต่างจากโมเดลภาษารุ่นก่อนหน้า เช่น Word2Vec และ GloVe ที่เป็นการเรียนรู้บริบทของคำแบบ Unidirectional หรือ Context-Free โดย BERT ใช้ Transformer Architecture ซึ่งประกอบไปด้วย Self-Attention Mechanism และ Positional Encoding ทำให้สามารถจับความสัมพันธ์ระหว่างคำในประโยคได้อย่างลึกซึ้งและแม่นยำขึ้น กระบวนการฝึกโมเดล BERT-Base ประกอบด้วย 2 ขั้นตอนหลัก ได้แก่

Pre-training คือ การฝึกโมเดลบนชุดข้อมูลขนาดใหญ่ เช่น Wikipedia และ BookCorpus โดยใช้สองกลยุทธ์หลัก คือ Masked Language Model (MLM) ทำการปิดบัง (Mask) คำบางคำในประโยคและให้โมเดลทำนายคำที่ถูกปิดบัง Next Sentence Prediction (NSP) ให้โมเดลทำนายว่าประโยคที่สองมีความสัมพันธ์ต่อเนื่องกับประโยคแรกหรือไม่

Fine-tuning คือ การนำโมเดลที่ผ่านการ Pre-training แล้วสามารถนำไปปรับแต่งเพิ่มเติม (Fine-tuning) บนชุดข้อมูลเฉพาะทาง เช่น ข้อมูลการวิเคราะห์อารมณ์ (Sentiment Analysis) หรือข้อมูลจากระบบติดตามข้อผิดพลาดของซอฟต์แวร์ (Bug Tracking System)



รูปที่ 2.23 ขั้นตอนการฝึกอบรมเบื้องต้นและปรับแต่ง BERT โดยรวม [56]

รูปที่ 2.23 แสดงกระบวนการฝึกและปรับแต่งโมเดล BERT ซึ่งแบ่งออกเป็นสองส่วนหลัก ได้แก่ Pre-training และ Fine-tuning โดยอธิบายการทำงานของโมเดล BERT ในแต่ละขั้นตอนดังนี้

1. Pre-training ในขั้นตอนนี้โมเดล BERT ถูกฝึกบนชุดข้อมูลขนาดใหญ่ที่ไม่มีการติดป้ายกำกับ (Unlabeled Data) เช่น Wikipedia และ BookCorpus โดยใช้สองกลยุทธ์หลัก คือ

Masked Language Model (MLM) โดยโมเดลจะทำการสุ่มปิดบัง (Mask) คำบางคำในประโยค (T1, T2, ..., TN) ด้วย [MASK] จากนั้นให้โมเดลทำนายคำที่ถูกปิดบังจากบริบทที่เหลืออยู่วิธีนี้ช่วยให้โมเดลเรียนรู้บริบทของคำในทั้งสองทิศทาง (Bidirectional Context)

Next Sentence Prediction (NSP) โมเดลจะได้รับคู่ของประโยค (Sentence A และ Sentence B) และต้องทำนายว่าประโยคที่สองมีความสัมพันธ์ต่อเนื่องกับประโยคแรกหรือไม่ ช่วยให้โมเดลเข้าใจความสัมพันธ์ระหว่างประโยค ซึ่งมีประโยชน์ในการวิเคราะห์ข้อความและภาษาธรรมชาติที่ต้องอาศัยบริบทระหว่างประโยค

องค์ประกอบของโมเดลในขั้นตอนนี้ประกอบด้วย

- [CLS] Token: ใช้เป็นตัวแทนของทั้งประโยคเพื่อใช้สำหรับการจำแนกประเภท (Classification)
- [SEP] Token: ใช้แยกประโยค A และ B
- Encoder Layers: โครงสร้างหลักของ BERT ที่ใช้ Self-Attention Mechanism ในการเรียนรู้บริบทของคำ

2. Fine-tuning หรือ การปรับแต่งเฉพาะทาง หลังจากที่โมเดลได้รับการ Pre-train แล้ว จะสามารถนำไปปรับแต่ง (Fine-tune) บนชุดข้อมูลที่มีการติดป้ายกำกับ (Labeled Data) เพื่อใช้งานเฉพาะทางในงานด้านต่าง ๆ เช่น

MNLI/NLI (Multi-Genre Natural Language Inference) สำหรับงานวิเคราะห์ความสัมพันธ์ระหว่างประโยค เช่น ความเป็นกลางหรือความขัดแย้ง

NER (Named Entity Recognition) สำหรับงานระบุชื่อเฉพาะในข้อความ เช่น ชื่อบุคคล สถานที่ องค์กร

SQuAD (Stanford Question Answering Dataset) สำหรับงานตอบคำถามโดยระบุขอบเขตของคำตอบ (Start/End Span) ในบริบทของข้อความ

กระบวนการ Fine-tuning มีดังนี้

- โมเดลที่ผ่านการ Pre-train จะถูกฝึกใหม่อีกครั้งบนชุดข้อมูลเฉพาะทาง
- ข้อมูลอินพุตที่ใช้ ได้แก่ คำถาม (Question) และย่อหน้าที่มีคำตอบ (Paragraph)
- โมเดลจะเรียนรู้ตำแหน่งของคำตอบในบริบท (Start/End Span) เพื่อให้สามารถทำนายคำตอบจากคำถามได้แม่นยำยิ่งขึ้น

โดยสรุปการทำงานของ BERT จะเริ่มต้นด้วยกระบวนการ Pre-training โดยใช้ Masked Language Model เพื่อให้โมเดลเรียนรู้ความหมายของคำจากบริบททั้งสองทิศทาง และ Next Sentence Prediction เพื่อช่วยให้เข้าใจความสัมพันธ์ระหว่างประโยค จากนั้นเข้าสู่กระบวนการ Fine-tuning ซึ่งเป็นการปรับแต่งโมเดลให้เหมาะสมกับงานเฉพาะทาง เช่น การจำแนกประเภทข้อความ การวิเคราะห์ชื่อเฉพาะ และการตอบคำถาม

2.9 ทบทวนงานวิจัยเกี่ยวกับการตรวจจับระดับความรุนแรงของจุดบกพร่อง

จากการศึกษาที่ผ่านมามีนักวิจัยสนใจศึกษาจุดบกพร่องของซอฟต์แวร์ เพื่อปรับปรุงกระบวนการการแก้ไขจุดบกพร่องให้มีประสิทธิภาพมากที่สุด ซึ่งแต่ละงานจะมีมุมมองของการศึกษาแตกต่างกันไป แต่อย่างไรก็ยังคงมีเป้าหมายร่วม คือ การทำให้กระกวนการปรับปรุงแก้ไขข้อบกพร่องของซอฟต์แวร์เกิดประสิทธิภาพสูงสุด จากการศึกษา [57] ได้ทำสรุปการศึกษาเกี่ยวรายงานจุดบกพร่องไว้ 3 กลุ่ม ดังตาราง ซึ่งงานวิจัยเกี่ยวกับการระดับความรุนแรงของจุดบกพร่องก็เป็นส่วนหนึ่งที่เกี่ยวข้องกับการปรับปรุงรายงานจุดบกพร่อง

ตารางที่ 2.12 สรุปการศึกษาเกี่ยวกับรายงานจุดบกพร่อง

กลุ่มการศึกษา	**ประเด็นการศึกษา**
1. การศึกษาที่เกี่ยวข้องกับการปรับปรุงรายงานจุดบกพร่อง	1.1 ศึกษาเนื้อหารายงานจุดบกพร่อง (Content optimization) 1.2 เทคนิคในการลดการจัดเก็บข้อมูลผิดพลาดของรายงานจุดบกพร่อง (Bug report misclassification หรือ bug or non-bug) 1.3 เทคนิคในการทำนายความรุนแรงของรายงานจุดบกพร่อง (Severity prediction)
2. การศึกษาที่เกี่ยวข้องการตรวจสอบรายงานจุดบกพร่อง	2.1 การจัดลำดับความสำคัญของรายงานจุดบกพร่อง 2.2 การตรวจสอบรายงานจุดบกพร่องที่ซ้ำซ้อน 2.3 การกำหนดรายงานจุดบกพร่องให้กับนักพัฒนาซอฟต์แวร์
3. การศึกษาที่เกี่ยวข้องแนวทางการ	3.1 การระบุตำแหน่งของจุดบกพร่องในซอฟต์แวร์ตาม

กลุ่มการศึกษา	ประเด็นการศึกษา
แก้ไขจุดบกพร่อง	รายงานจุดบกพร่อง 3.2 การกู้คืนลิงก์ระหว่างรายงานจุดบกพร่องและไฟล์การเปลี่ยนแปลง 3.3 การคาดการณ์เวลาในการแก้ไขข้อบกพร่อง

ในการศึกษานี้ผู้วิจัยได้สนใจศึกษาในกลุ่มการศึกษาที่เกี่ยวข้องกับการปรับปรุงรายงานจุดบกพร่อง โดยเฉพาะในประเด็นเทคนิคในการทำนายความรุนแรงของรายงานจุดบกพร่อง (Severity prediction)

2.9.1 การศึกษาเกี่ยวกับการจำแนกจุดบกพร่องรุนแรงกับจุดบกพร่องไม่รุนแรง

ในปี ค.ศ. 2010 Ahmed [11] และคณะ ได้ศึกษาการทำนายระดับความรุนแรงของรายงานจุดบกพร่อง โดยคณะผู้วิจัยได้เล็งเห็นถึงความสำคัญในการทำนายระดับความรุนแรงรายงานจุดบกพร่อง อัตโนมัติจะช่วยให้ลดเวลาในการประเมินระดับความรุนแรงด้วยตนเองจากผู้ตรวจสอบ การศึกษาครั้งนี้ใช้ชุดข้อมูลทดสอบจากกลุ่มซอฟต์แวร์โอเพนซอร์ซ ได้แก่ Mozilla, Eclipse และ GNOME ซึ่งแต่ละซอฟต์แวร์ประกอบด้วย 3 ซอฟต์แวร์ส่วนประกอบ (Component) โดยเป็นการทำนายระดับความรุนแรงแบบ 2 คลาส คือ ไม่รุนแรง หรือ รุนแรง (Non-severe or Severe) ใช้ตัวจำแนกที่พัฒนาด้วยนาอีฟบย์ (Naive Bayes) ซึ่งผลการทำนายมีค่าความแม่นยำอยู่ระหว่าง 0.65 – 0.75 สำหรับ Mozilla และ Eclipse และผลการทำนายมีค่าความแม่นยำอยู่ระหว่าง 0.70 – 0.85 สำหรับ GNOME ทั้งนี้ในการศึกษานี้ยังได้สรุปความถี่คำศัพท์ (Terms) ที่ปรากฏบ่อยในประโยครายงานจุดบกพร่อง โดยแยกประเภทรายงานจุดบกพร่องแบบรุนแรงและไม่รุนแรง และพวกเขายังทำการทดลองด้วยว่าการใช้คำอธิบายรายงานจุดบกพร่องที่ยาวขึ้นจะสามารถส่งผลให้การทำนายแม่นยำขึ้นหรือไม่ ซึ่งผลกลับไม่ได้เป็นเช่นนั้น

ในปี ค.ศ. 2014 Roy และ Rossi [58] ทำการศึกษาการจัดกลุ่มประเภทความรุนแรงของรายงานจุดบกพร่องซอฟต์แวร์ เพื่อการจัดกลุ่มระดับความรุนแรงหรือไม่รุนแรง โดยใช้ชุดข้อมูลทดสอบคือรายงานจุดบกพร่องของ Mozilla และ Eclipse เป็นชุดข้อมูลที่ใช้การแพร่หลายในกลุ่มนักวิจัยที่ทำการศึกษาด้านนี้ ซึ่งรายงานจุดบกพร่องของ Mozilla ประกอบด้วย Thunderbird, Firefox, Bugzilla และ Core ในส่วนของ Eclipse ประกอบด้วย CDT, JDT, PDE และ Platform โดยทำการทำนายระดับความรุนแรง 2 คลาส คือ ไม่รุนแรง หรือ รุนแรง (Non-severe or Severe

และใช้โมเดลในการจำแนก คือ นาอีฟบย์ โดยงานศึกษาครั้งนี้มี 2 คำถามการวิจัย คือ 1) โมเดล bi-gram ให้ประสิทธิภาพมากกว่า unigrams หรือไม่ 2) อะไรคือผลกระทบของการเลือกคุณสมบัติเพิ่มเข้าไปในแบบจำลอง bi-gram และทำการเปรียบเทียบ 3 โมเดลในการจำลองคำ คือ 1) uni-grams 2) uni-grams+bi-grams 3) uni-grams+bi-grams+x2 feature selection. ผลลัพธ์ที่ได้พบว่า ศึกษาด้วยวิธีการ bi-grams ให้ผลลัพธ์ที่ดีในบางกรณี และการคัดเลือกคุณลักษณะ (Feature Selection) ช่วยเพิ่มประสิทธิภาพของผลลัพธ์ได้เป็นอย่างดี โดย แบบจำลอง uni-grams+bi-grams+x2 feature selection. ได้ค่าค่าความแม่นยำอยู่ระหว่าง 0.787 - 0.880

ปี ค.ศ. 2019 Ramay และคณะ [14] ทำการศึกษาเพื่อสร้างโมเดลบนพื้นฐานเทคนิคการเรียนรู้เชิงลึก เพื่อทำนายระดับความรุนแรงของรายงานจุดบกพร่อง โดยการออกแบบการวิจัยครั้งนี้มีขั้นตอนการทำงาน 4 ขั้น คือ ขั้นแรก ใช้เทคนิคการประมวลผลภาษาธรรมชาติสำหรับการประมวลผลข้อความล่วงหน้าของรายงานข้อบกพร่อง ขั้นที่สอง ทำการคำนวณและกำหนดคะแนนอารมณ์ (emotion score) สำหรับรายงานข้อบกพร่องแต่ละรายการ ขั้นที่สาม สร้างเวกเตอร์สำหรับรายงานข้อบกพร่องที่ประมวลผลล่วงหน้าแต่ละรายการ และขั้นที่สี่ ทำการส่งเวกเตอร์ที่สร้างขึ้นและคะแนนอารมณ์ของรายงานข้อบกพร่องแต่ละรายการไปยังเครือข่ายประสาทเทียมระดับลึกตามระดับเพื่อการทำนายความรุนแรงของรายงาน ซึ่งการทำนายระดับความรุนแรง 2 ระดับ คือ รุนแรง หรือไม่รุนแรง โดยใช้ชุดข้อมูล (Dataset) จากการรวบรวมของ Lamkanfi [59] ทำการเปรียบเทียบโมเดล 4 วิธี คือ Deep Neural Network (CNN), Multinomial Naive Bayes, Random Forest และ Long Short Term Memory (LSTM) และทำการเปรียบเทียบกับการศึกษาที่ดีที่สุด (State-of-the-art) พบว่าโดยเฉลี่ยงานวิจัยที่นำเสนอมีค่า f-measure ดีกว่า 7.90%

ในปี 2021 Anh-Hien และ Cheng-Zen [15] ทำการศึกษาการทำนายระดับความรุนแรงจุดบกพร่องด้วยวิธีการเรียนรู้เชิงลึก (Deep learning) และการใช้คุณสมบัติหลายแบบ (Multi-Aspect Features) และใช้ชุดข้อมูลในการทดสอบ คือ รายงานจุดบกพร่องของ Mozilla ประกอบด้วย Core จำนวน 18,168 รายงาน Firefox จำนวน 22,233 รายงาน Thunderbird จำนวน 6,800 รายงาน และ Bugzilla จำนวน 2,139 รายงาน และ Eclipse ประกอบด้วย CDT จำนวน 1,093 รายงาน JDT จำนวน 2,507 รายงาน และ Platform จำนวน 5,884 รายงาน โดยการทำนายระดับความรุนแรง 2 คลาส คือ ไม่รุนแรง หรือ รุนแรง (Non-severe or Severe) ซึ่งทำการจัดกลุ่มระดับไม่รุนแรง หรือ Non-severe ประกอบด้วย Minor และ Trivial ระดับความรุนแรง หรือ Severe ประกอบด้วย Blocker, Critical, และ Major ซึ่งงานวิจัยนำเสนอกรอบการเรียนรู้เชิงลึก (Deep learning framework) โดยเรียกว่า MASP ซึ่งเป็นการใช้เครือข่ายประสาทเทียม (CNN) และใช้คุณลักษณะจากรายงานจุดบกพร่อง ได้แก่ ด้านเนื้อหา (content-aspect) ด้านความรู้สึก

(sentiment-aspect) ด้านคุณภาพ (quality-aspect) และด้านผู้รายงาน สรุปผลการทำนายพบว่าความแม่นยำวิธีที่นำเสนอมีค่าระหว่าง 0.7504 – 0.7586 ซึ่งมากกว่าเทคนิคล่าสุด (state of the art) ที่มีความแม่นยำที่ 0.7427

ปี ค.ศ. 2021 Hamza [10] ได้นำเสนอวิทยานิพนย์ การตรวจจับระดับความรุนแรงด้วยเทคนิคการเรียนรู้ของเครื่อง ซึ่งการศึกษานี้ได้ใช้ชุดข้อมูลรายงานจุดบกพร่องจากระบบปิด โดยรายงานที่นำมาใช้เป็นระบบติดตามรายงานจุดบกพร่องของ JIRA จำนวน 2,300 รายงาน มีระดับความรุนแรง 5 ระดับ คือ Blocker, Critical, Major, Minor, และ Low และทำการจัดกลุ่มระดับความรุนแรงเป็น 2 คลาส คือ รุนแรง และไม่รุนแรง สร้างโมเดลในการทดสอบด้วย RNN และ LSTM ผลการศึกษาพบว่า LSTM ให้ค่าความแม่นยำที่ 0.85 และ RNN ให้ค่าความแม่นยำ 0.58

2.9.2 การศึกษาเกี่ยวกับการจำแนกระดับความรุนแรงของจุดบกพร่องหลายระดับ

ปี ค.ศ. 2012 Tian และคณะ [60] ทำการศึกษาการทำนายระดับความรุนแรงหลายระดับ ได้แก่ blocker, critical, major, minor, และ trivial ชุดข้อมูลจากซอฟต์แวร์แบบเปิดประกอบด้วย OpenOffice, Mozilla and Eclipse โดยเสนอแนวทางใหม่ที่ใช้ประโยชน์จากการดึงข้อมูลโดยเฉพาะอย่างยิ่งฟังก์ชันความคล้ายคลึงกันของเอกสารที่ใช้ BM25 เพื่อคาดการณ์ความรุนแรงของรายงานจุดบกพร่องโดยอัตโนมัติ วิธีการจะทำการวิเคราะห์รายงานจุดบกพร่องที่รายงานในอดีตโดยอัตโนมัติพร้อมกับป้ายกำกับระดับความรุนแรงที่กำหนด และทำนายป้ายกำกับระดับความรุนแรงให้กับรายงานจุดบกพร่องของรายงานใหม่ ซึ่งวิธีที่นำเสนอได้ค่าคะแนน F measure เท่ากับ 74%

ปี ค.ศ. 2017 Sadia และคณะ [61] ได้นำเสนอวิธีการจำแนกระดับความรุนแรงของรายงานจุดบกพร่องหลายระดับ ด้วยวิธี Bug Feature Selection (BFSp) ซึ่งทำการปรับปรุงจากทฤษฎี Pareto Optimality ซึ่งใช้ชุดข้อมูลจาก Eclipse, Mozilla และ GCC พร้อมทั้งทำการประเมินประสิทธิภาพข้ามโครงการ ซึ่งวิธีการที่นำเสนอ BFSp มีขั้นตอนหลัก คือ 1) Feature Extraction เป็นการสกัดคุณลักษณะที่สนใจนำมาศึกษา ซึ่งในรายงานจุดบกพร่องนั้นมีข้อมูลอยู่หลายแอตทริบิวต์ โดยการศึกษานี้สนใจเฉพาะ summary และ description โดยนำข้อความเข้าสู่ขั้นตอนตามหลักการประมวลผลภาษาธรรมชาติ (NLP) ได้แก่ การลบข้อมูลขยะ การตัดคำ การหาความถี่ของคำ (Term Document Matrix: TDM) และขั้นตอนที่ 2) Feature Selection ขั้นตอนนี้จะทำการเลือกคำจาก TDM เนื่องจากบางคำอาจจะไม่ได้มีความจำเป็นในการนำมาใช้ และทำการสร้างโมเดลเพื่อใช้จำแนกด้วย Support Vector Machine (SVM) และ Decision Tree (DT) และทำการประเมินประสิทธิภาพด้วยการทดสอบข้ามชุดข้อมูลโครงการ (cross-project) เช่น ทำการสอนโมเดลด้วยรายงานจุดบกพร่อง Eclipse แล้วทำการทดสอบด้วย GCC และ Mozilla ซึ่งให้ค่า F-measure

อยู่ที่ระหว่าง 5.94% - 40.5% อีกทั้งผู้วิจัยยังได้ทำการทดสอบวิธีที่นำเสนอกับชุดข้อมูล SPAM จาก UCI Machine Learning Repository พบว่าวิธีที่นำเสนอให้ความแม่นยำมากกว่า 10% กับงานวิจัยอื่นที่นำมาเปรียบเทียบ

ปี ค.ศ. 2019 Ashima และคณะ [41] ได้นำเสนอเทคนิควิธีการจำแนกระดับความรุนแรงจากรายงานจุดบกพร่องหลายระดับ ซึ่งได้นำชุดข้อมูลทดสอบมาจากหลายแหล่งข้อมูล ได้แก่ Mozilla, Eclipse, JBoss, OpenFOAM และ Firefox ซึ่งแต่ละชุดข้อมูลมีจำนวนคลาสแตกต่างกันตั้งแต่ 5-8 คลาส เช่น Mozilla งานวิจัยนี้ใช้ระดับความรุนแรง 7 คลาส ประกอบด้วย Blocker, Critical, Enhancement, Major, Normal, Minor และ Trivial และ Eclipse มีระดับความรุนแรง 5 ระดับ ได้แก่ Blocker, Critical, Enhancement, Major และ Normal เทคนิคที่งานวิจัยนี้นำเสนอ คือ การเรียนรู้เชิงลึกที่ชื่อว่า Convolutional Neural Network and Random forest with Boosting (BCR) โดยเป็นการผสมผสานวิธีของ CNN และ Random forest with Boosting โดยมีค่าความแม่นยำเฉลี่ยอยู่ที่ 96.34% และค่า F-measures 96.43% แต่อย่างไรก็ตามในข้อเสนอแนะผู้วิจัยได้เสนอความคิดเห็นว่า งานวิจัยนี้ได้ทำการสอนโมเดลด้วยชุดข้อมูลเฉพาะ และทำการทดสอบเฉพาะข้อมูลนั้น ๆ หากทำการไปทดสอบข้ามชุดข้อมูลหรือข้ามโครงการอาจจะไม่ได้ผลลัพธ์ที่ดี

ปี ค.ศ. 2020 Youshuai Tan และคณะ [62] นำเสนอแนวทางในการทำนายระดับความรุนแรงของจุดบกพร่องของซอฟต์แวร์จากรายงานโดยใช้ข้อมูลจาก Stack Overflow เพื่อเสริมข้อมูลในรายงานจุดบกพร่อง ทำให้สามารถพยากรณ์ระดับความรุนแรงได้แม่นยำขึ้น โมเดลถูกทดสอบกับชุดข้อมูลจาก Mozilla, Eclipse และ GCC โดยรวบรวมรายงานบั๊กจากระบบติดตามบั๊กและเพิ่มข้อมูลจากคำถามคำตอบที่เกี่ยวข้องใน Stack Overflow เพื่อเสริมข้อมูลที่อาจขาดหายไป ซึ่งวิธีการการพยากรณ์ระดับความร้ายแรงของบั๊กรายงานใช้ Logistic Regression เป็นตัวจำแนกหลัก โดยขั้นตอนสำคัญประกอบด้วยการดึงข้อมูลจาก Stack Overflow และเชื่อมโยงกับรายงานจุดบกพร่องผ่านอัลกอริธึม BM25 เพื่อเพิ่มความถูกต้องของข้อมูล จากนั้นเตรียมข้อมูลด้วยวิธีการ tokenization stop word removal stemming และสร้างตัวแทนข้อมูลด้วย POS Tagging และ Word2Vec เพื่อฝึก Logistic Regression และทำนายระดับความร้ายแรงของจุดบกพร่อง และงานวิจัยนี้ประเมินผลโดยใช้ F-measure Precision และ Recall เปรียบเทียบกับวิธีที่มีอยู่ ได้แก่ Naïve Bayesian, K-Nearest Neighbor และ Long Short-Term Memory ผลลัพธ์แสดงให้เห็นว่าโมเดลที่นำข้อมูลจาก Stack Overflow มาเสริมสามารถเพิ่มค่า F-measure ได้ถึง 23.03% สำหรับ Mozilla, 21.86% สำหรับ Eclipse และ 20.59% สำหรับ GCC เมื่อเทียบกับวิธีที่ดีที่สุดใน baseline (Naïve Bayesian)

ปี ค.ศ. 2020 Ashima และคณะ [63] ได้ทำการศึกษาเกี่ยวกับการจำแนกระดับความรุนแรงจากรายงานจุดบกพร่องอีกครั้ง โดยเขาและคณะได้ตั้งคำถามเกี่ยวกับการศึกษาว่า จริง ๆ แล้วรายงานจุดบกพร่องนั้นสามารถบอกถึงระดับความรุนแรงของซอฟต์แวร์ได้หรือไม่ ซึ่งการศึกษามุ่งเป้าการเปรียบเทียบประเด็นของเนื้อหาในรายงานจุดบกพร่อง คือ รายงานที่มีเนื้อหาใจความสรุป กับรายงานที่ไม่มีเนื้อหาใจความสรุป สามารถให้ผลลัพธ์ที่แม่นยำต่อการจำแนกระดับความรุนแรงได้หรือไม่ ซึ่งในการวิจัยนี้ได้ทำการใช้วิธีการเลือกคุณลักษณะมาใช้ด้วยวิธี Ant colony optimization และวิธีการ Naive Bayes เพื่อทำการจำแนกระดับความรุนแรง ซึ่งผลการวิจัยพบว่า ปริมาณของเนื้อหารายงานจุดบกพร่องที่มาก แต่ขาดการสรุป (summarization) ให้ผลลัพธ์ที่น้อยกว่ารายงานที่มีการสรุป

ปี ค.ศ. 2021 Jayalath [18] ทำการศึกษาเพื่อออกแบบโมเดลในการทำนายระดับความรุนแรงหลายระดับ โดยทำการศึกษาภายใต้เงื่อนไขที่จำนวนรายงานของป้ายกำกับระดับความรุนแรงมีความไม่สมดุล (Imbalanced) โดยใช้ชุดข้อมูลของ UNIX Kernels ที่มีจำนวนรายงานจุดบกพร่องมากกว่า 16,000 รายงาน ซึ่งมีระดับความรุนแรง ดังนี้ Normal จำนวน 7,921 รายงาน Blocking จำนวน 551 รายงาน High 1,655 รายงาน และ Low 509 รายงาน และทำการพัฒนาโมเดลเพื่อทำการทำนายด้วย 3 โมเดล คือ Naïve Bayes, Decision Tree และ Logistic Regression ซึ่งพบว่าโมเดลที่พัฒนาด้วย Logistic Regression ให้ค่าความแม่นยำ (Accuracy) สูงกว่าทั้ง 2 โมเดล คือ ทำนายระดับ Normal เท่ากับ 0.62 Blocking เท่ากับ 0.65 Low เท่ากับ 0.62 และ High เท่ากับ 0.64

ปี ค.ศ. 2022 Jungyeon Kim และ Geunseok Yang [64] เสนออัลกอริธึมการทำนายความรุนแรงของจุดบกพร่อง ซึ่งใช้การเลือกคุณสมบัติตามหัวข้อ (topic-based feature selection) และอัลกอริธึม CNN-LSTM เพื่อคาดการณ์ความรุนแรงของจุดบกพร่องในโครงการซอฟต์แวร์ โดยใช้ชุดข้อมูลจากรายงานข้อบกพร่องจากโปรเจ็กต์โอเพ่นซอร์ส Eclipse และ Mozilla เป็นชุดข้อมูลสำหรับการทำนายความรุนแรงของข้อบกพร่อง ชุดข้อมูล Eclipse ประกอบด้วยรายงานข้อบกพร่อง 165,547 รายการ ซึ่งรวบรวมตั้งแต่วันที่ 10 ตุลาคม ค.ศ. 2001 ถึง 14 มิถุนายน ค.ศ. 2005 ชุดข้อมูล Mozilla ประกอบด้วยรายงานข้อบกพร่อง 394,878 รายการ ซึ่งรวบรวมตั้งแต่วันที่ 16 กรกฎาคม ค.ศ. 1999 ถึง 16 กันยายน ค.ศ. 1999 การแบ่งระดับความรุนแรงจุดบกพร่องแบบเดียวกับการศึกษาของ Ashima [49] โดยที่วิธีที่นำเสนอจะเป็นอัลกอริธึมการเลือกคุณสมบัติที่ใช้ในบทความนี้จะกรองคุณสมบัติที่ซ้ำหรือไม่เกี่ยวข้องออก และเลือกชุดย่อยของคุณสมบัติที่เกี่ยวข้องมากที่สุดโดยอัตโนมัติ อัลกอริธึมจะสร้างชุดย่อยของคุณสมบัติที่มีประสิทธิภาพการจำแนกประเภทหรือความสัมพันธ์สูงสุด ซึ่งจะช่วยลดขนาดของข้อมูล ในการศึกษานี้ รายงานข้อบกพร่องถูกจัดประเภท

ตามหัวข้อ และคุณลักษณะต่าง ๆ ถูกดึงออกมาจากความรุนแรงของแต่ละหัวข้อโดยใช้อัลกอริธึมการเลือกคุณลักษณะ ผลการทดลองพบว่าวิธีที่นำเสนอ ได้รับค่า F-measure สูงถึง 90.62% และ 93.22% สำหรับ Eclipse และ Mozilla ตามลำดับ และเหนือกว่าเมื่อเปรียบเทียบกับโมเดลพื้นฐาน

ปี ค.ศ. 2023 Ye Wei และคณะ [65] เสนอแนวทางการทำนายความรุนแรงของจุดบกพร่องที่เรียกว่า KICL ซึ่งรวมแบบจำลองภาษาที่ได้รับการฝึกอบรมล่วงหน้าและกลยุทธ์การฝึกอบรมล่วงหน้าเฉพาะโดเมน (domain-specific) KICL ใช้เลเยอร์ตัวเข้ารหัส Transformer เพื่อประมวลผลลำดับรายงานข้อบกพร่อง และดำเนินการฝังคำและฝังตำแหน่งไว้ โดยใช้ชุดข้อมูลในการทดลองประกอบด้วยรายงานจุดบกพร้องมากกว่า 270,000 รายการจากสี่โครงการที่มีชื่อเสียงใน BugZilla รวมถึง Mozilla, Eclipse, Netbeans และ GNU Compiler Collection (GCC) ในรูปแบบหลายคลาสประกอบด้วย Blocker, Critical, Major, Minor และ Trivial ซึ่งแนวทางที่นำเสนอหรือ KICL มีประสิทธิภาพเหนือกว่าแนวทางพื้นฐานทั้งหมด โดยมีค่า F1 สูงกว่าถึง 30.68%

ปี ค.ศ. 2023 Sen Fang และคณะ [66] นำเสนอ RepresentThemAll ซึ่งเป็นโมเดลที่สามารถเรียนรู้การแทนค่าบั๊กรายงานในรูปแบบสากล (universal representation) และสามารถนำไปใช้กับหลายงานในกระบวนการบำรุงรักษาซอฟต์แวร์ เช่น การตรวจจับจุดบกพร่องที่ซ้ำกัน การสรุปจุดบกพร่อง การทำนายลำดับความสำคัญของจุดบกพร่อง และการทำนายระดับความรุนแรงของจุดบกพร่อง โมเดลถูกฝึกบนชุดข้อมูลรายงานขนาดใหญ่จาก Bugzilla รวม 275,639 รายงาน จากโครงการ Mozilla, Eclipse, Netbeans และ GNU Compiler Collection (GCC) [65] โดยใช้ 80% เป็นข้อมูลฝึก, 10% สำหรับ validation และ 10% สำหรับทดสอบ โมเดล RepresentThemAll ถูกฝึกล่วงหน้าด้วยสองวัตถุประสงค์หลัก ได้แก่ Dynamic Masked Language Model ซึ่งใช้การปิดบังแบบไดนามิกเพื่อให้โมเดลสามารถเรียนรู้บริบทของคำในรายงานจุดบกพร่อง และ Contrastive Learning ("Find Yourself") ที่ใช้ Siamese Network เพื่อเรียนรู้ความแตกต่างของรายงานจุดบกพร่อง โดยสามารถแยกแยะระหว่างบั๊กที่มีบริบทใกล้เคียงและแตกต่างกันได้ การประเมินผลด้วย Precision Recall F1-score แบบ weigthed average และ Accuracy เช่นกับงานวิจัยของ Ye Wei [65]

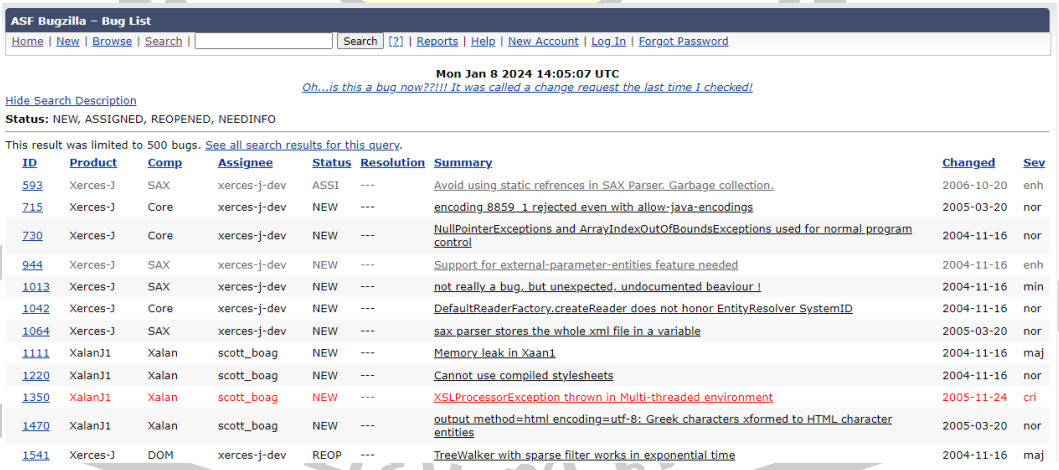
บทที่ 3

วิธีดำเนินการวิจัย

การดำเนินการวิจัยให้บรรลุตามวัตถุประสงค์ ในการจำแนกแบบหลายคลาสสำหรับการตรวจจับระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์ ผู้วิจัยจะนำเสนอวิธีดำเนินการวิจัยในแต่ละขั้นตอน ดังนี้

3.1 ชุดข้อมูล (Dataset)

ในงานวิจัยนี้ใช้ชุดข้อมูลจากคลังเก็บรายงานจุดบกพร่องของ Mozilla Eclipse Netbeans GCC [67] ซึ่งเป็นชุดข้อมูลที่ถูกนำไปศึกษาต่ออย่างแพร่หลายอีกหลายกรณี [65, 66] นอกจากนี้แล้วยังใช้จากการศึกษา [21] เนื่องจากชุดข้อมูลดังกล่าวจะมีคุณลักษณะของข้อมูลที่แสดงรายละเอียดของรายงาน (description) แยกจากส่วนสรุปรายงาน (summary) ทั้งนี้การนำชุดข้อมูลจากการศึกษาก่อนหน้าถูกนำมาใช้มีวัตถุประสงค์เพื่อใช้เปรียบเทียบประสิทธิภาพของงานวิจัยที่นำเสนอนี้ อย่างไรก็ตามผู้วิจัยยังได้ทำการดาวน์โหลดชุดข้อมูลใหม่จากคลังรายงานจุดบกพร่องของ Apache จากเว็บไซต์ระบบรายงานจุดบกพร่อง https://bz.apache.org/bugzilla/ สำหรับการทดสอบ เพื่อทำการเปรียบเทียบชุดข้อมูลและวิธีการต่าง ๆ สามารถแสดงตัวอย่างรายการจุดบกพร่องของซอฟต์แวร์จากระบบติดตามของ Apache ดังรูป



ASF Bugzilla – Bug List

Home | New | Browse | Search | Search [?] | Reports | Help | New Account | Log In | Forgot Password

Mon Jan 8 2024 14:05:07 UTC

Oh...is this a bug now??!!! It was called a change request the last time I checked!

Hide Search Description

Status: NEW, ASSIGNED, REOPENED, NEEDINFO

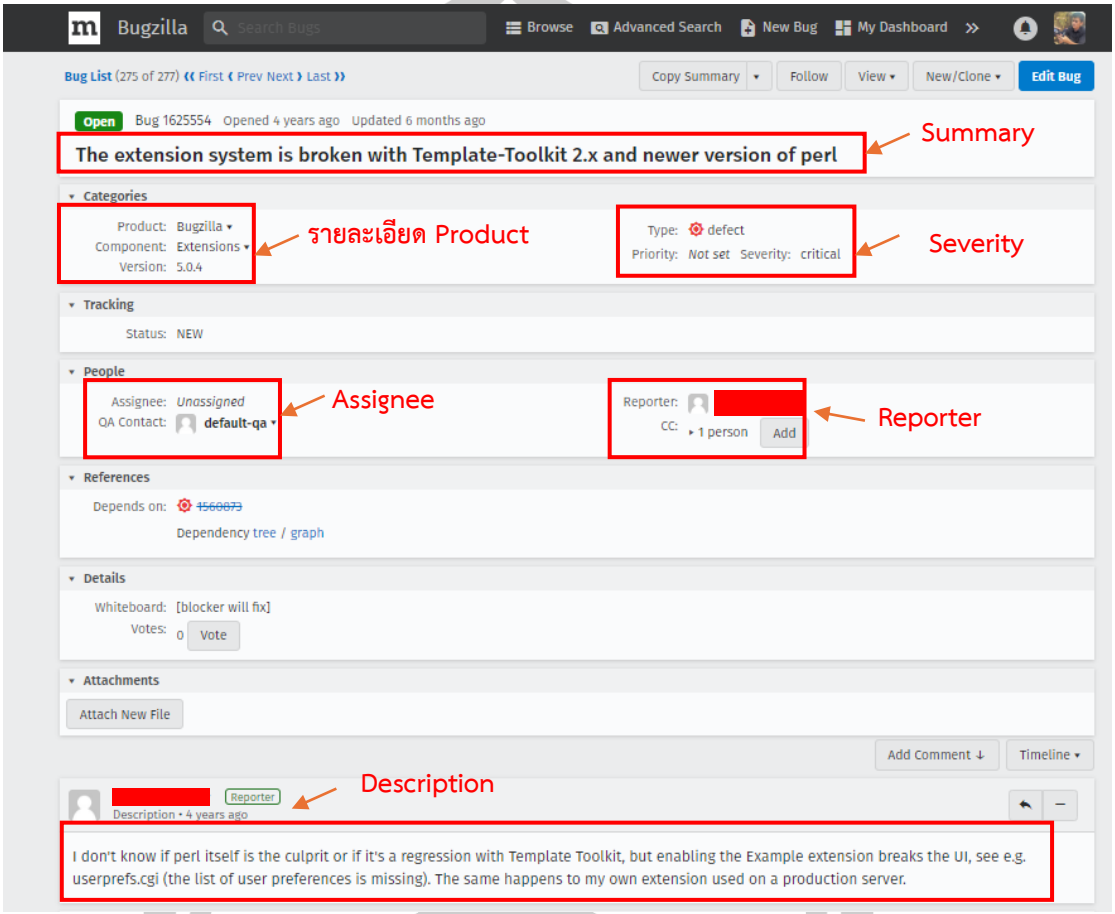
This result was limited to 500 bugs. See all search results for this query.

ID	Product	Comp	Assignee	Status	Resolution	Summary	Changed	Sev
593	Xerces-J	SAX	xerces-j-dev	ASSI	---	Avoid using static refrences in SAX Parser. Garbage collection.	2006-10-20	enh
715	Xerces-J	Core	xerces-j-dev	NEW	---	encoding 8859_1 rejected even with allow-java-encodings	2005-03-20	nor
730	Xerces-J	Core	xerces-j-dev	NEW	---	NullPointerExceptions and ArrayIndexOutOfBoundsExceptions used for normal program control	2004-11-16	nor
944	Xerces-J	SAX	xerces-j-dev	NEW	---	Support for external-parameter-entities feature needed	2004-11-16	enh
1013	Xerces-J	SAX	xerces-j-dev	NEW	---	not really a bug, but unexpected, undocumented beaviour !	2004-11-16	min
1042	Xerces-J	Core	xerces-j-dev	NEW	---	DefaultReaderFactory.createReader does not honor EntityResolver SystemID	2004-11-16	nor
1064	Xerces-J	SAX	xerces-j-dev	NEW	---	sax parser stores the whole xml file in a variable	2005-03-20	nor
1111	XalanJ1	Xalan	scott_boag	NEW	---	Memory leak in Xaan1	2004-11-16	maj
1220	XalanJ1	Xalan	scott_boag	NEW	---	Cannot use compiled stylesheets	2004-11-16	nor
1350	XalanJ1	Xalan	scott_boag	NEW	---	XSLProcessorException thrown in Multi-threaded environment	2005-11-24	cri
1470	XalanJ1	Xalan	scott_boag	NEW	---	output method=html encoding=utf-8; Greek characters xformed to HTML character entities	2005-03-20	nor
1541	Xerces-J	DOM	xerces-j-dev	REOP	---	TreeWalker with sparse filter works in exponential time	2004-11-16	maj

รูปที่ 3.1 คลังเก็บรายงานจุดบกพร่อง Apache

ในแต่ละรายงานจุดบกพร่องจะประกอบไปด้วยข้อมูลต่าง ๆ ที่เกิดจากผู้รายงานและผู้จัดการโครงการ รวมทั้งนักพัฒนา เช่น ผลิตภัณฑ์ (Product) เวอร์ชั่น (Version) ประเภท (Type) การมอบหมาย (Assignee) ความสำคัญ (Priority) ความรุนแรง (Severity) รายละเอียดแบบสรุป

(Summary) ผู้รายงาน (Reporter) รายละเอียดของรายงาน (description) รวมถึงการแสดงความคิดเห็น (comment) สามารถแสดงรายละเอียดของรายงานจุดบกพร่องดังรูป



รูปที่ 3.2 ตัวอย่างรายงานจุดบกพร่อง

การศึกษานี้ได้ใช้ข้อมูลส่วนรายละเอียดแบบสรุป หรือ summary และรายละเอียดของรายงาน หรือ description เพื่อใช้เป็นชุดข้อมูลสำหรับการหาคุณลักษณะในการโมเดล โดยชุดข้อมูลได้ทำการคัดเลือกมาสำหรับการศึกษานี้จะใช้เฉพาะรายงานจุดบกพร่องจริง ซึ่งถูกกำหนดระดับความรุนแรงจาก Triager ของระบบติดตามรายงานจุดบกพร่องของแต่ละระบบ และมีการกำหนดระดับความรุนแรงไว้ 5 ระดับ ได้แก่ blocker critical major minor และ trivial [62, 65, 66] แสดงจำนวนรายงานจุดบกพร่องและระดับความรุนแรงดังตาราง

ตารางที่ 3.1 รายละเอียดจำนวนระดับความรุนแรง

ชุดที่	Project	Severity					Total
		blocker	critical	major	minor	trivial	
1	Mozilla [21]	158	1580	1884	1241	593	**5456**
2	Mozilla, Eclipse, Netbeans, GCC [65, 66]	26164	35084	48739	30699	13568	**154254**
3	Apache	388	529	843	466	37	**2263**

จากตารางที่ 3.1 ประกอบด้วยชุดข้อมูลรายงานจุดบกพร่องจากระบบรายงาน 3 ชุด ซึ่งแต่ละชุดข้อมูลระบุระดับความรุนแรงอย่างชัดเจนแล้วจากระบบ โดยแต่ละชุดจะนำมาเพื่อทดสอบกระบวนการวิจัยในการจำแนกระดับความรุนแรง ซึ่งสามารถอธิบายได้ดังนี้

ชุดที่ 1 เป็นชุดข้อมูลหลักที่ใช้ในงานวิจัยครั้งนี้ ซึ่งรวบรวมรายงานจุดบกพร่องจากระบบติดตามข้อผิดพลาดของซอฟต์แวร์ โดยแต่ละรายงานประกอบด้วยข้อมูลสำคัญ เช่น summary ซึ่งเป็นข้อความสั้นที่อธิบายปัญหาอย่างกระชับ และ description ที่ให้รายละเอียดเพิ่มเติมเกี่ยวกับจุดบกพร่องในรูปแบบของข้อความยาว ข้อมูลในชุดนี้ถูกจัดเก็บในรูปแบบที่มีโครงสร้างอย่างชัดเจนเพื่อให้ง่ายต่อการวิเคราะห์ อย่างไรก็ตาม ข้อมูลนี้มีปัญหาความไม่สมดุลของจำนวนแถวในแต่ละคลาส ซึ่งหมายความว่าจำนวนรายงานในแต่ละระดับความรุนแรงมีการกระจายตัวที่ไม่เท่ากัน โดยเฉพาะคลาส blocker ซึ่งมีจำนวนแถวที่ต่ำกว่าคลาสอื่น ๆ อย่างมีนัยสำคัญ (blocker จำนวน 158 และ major 1884 แถว) การมีข้อมูลที่ไม่สมดุลนี้ส่งผลกระทบต่อประสิทธิภาพของโมเดลการเรียนรู้ของเครื่อง เนื่องจากโมเดลมักจะเรียนรู้จากคลาสที่มีจำนวนตัวอย่างมากกว่าได้ดีกว่า ส่งผลให้การทำนายคลาสที่มีจำนวนน้อยมีความแม่นยำต่ำ ดังนั้น งานวิจัยนี้จึงได้ทำการแก้ไขปัญหาความไม่สมดุลของข้อมูลโดยใช้เทคนิคการเสริมข้อมูลด้วย T5 Model และทดสอบกับการปรับแต่งโมเดล BERT

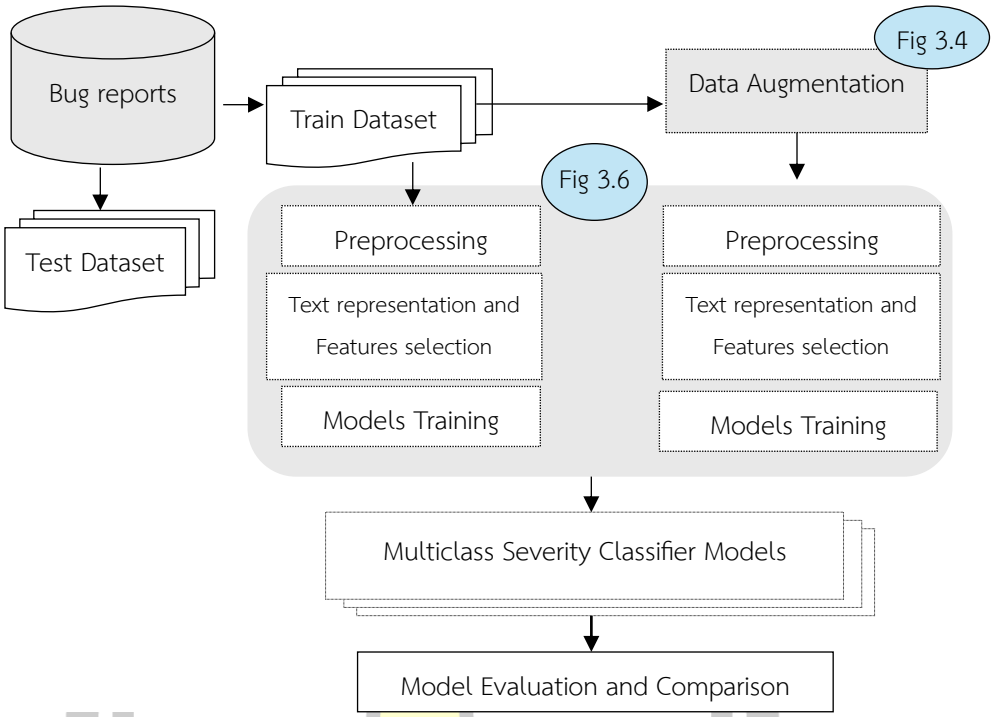
ชุดที่ 2 เป็นชุดข้อมูลที่รวบรวมรายงานข้อผิดพลาดจากหลายโครงการซอฟต์แวร์ ซึ่งเป็นข้อมูลที่มีความหลากหลายมากขึ้นและมีการใช้งานแพร่หลายในงานวิจัยเกี่ยวกับการจำแนกระดับความรุนแรงของจุดบกพร่องในซอฟต์แวร์แบบหลายคลาส นอกจากนี้ ยังมีการศึกษาที่เกี่ยวข้องกับการจำแนกระดับความสำคัญของข้อผิดพลาดเพื่อนำมาใช้เป็นข้อมูลพื้นฐานในการพัฒนาเทคนิคการจำแนกที่แม่นยำยิ่งขึ้น [65-67] ข้อมูลในชุดนี้มีความแตกต่างจากชุดที่ 1 ตรงที่ระดับความรุนแรงของข้อผิดพลาดในชุดข้อมูลนี้มีการกระจายตัวที่ค่อนข้างสมดุลมากกว่า ซึ่งช่วยลดปัญหาการเรียนรู้ที่เกิดจากการมีข้อมูลที่ไม่สมดุล อย่างไรก็ตาม การที่ข้อมูลมีการกระจายตัวที่สมดุลไม่ได้หมายความว่า

โมเดลจะสามารถจำแนกได้อย่างแม่นยำเสมอไป เนื่องจากโครงสร้างของข้อมูลและบริบทของข้อความอาจมีความแตกต่างกัน งานวิจัยนี้จึงได้นำชุดข้อมูลนี้มาใช้เป็นกรณีศึกษาสำหรับการปรับแต่งโมเดล BERT เพื่อตรวจสอบว่าโมเดลสามารถคงประสิทธิภาพของการจำแนกระดับความรุนแรงของจุดบกพร่องได้ดีเพียงใดเมื่อไม่มีปัญหาความไม่สมดุลของข้อมูลเข้ามาเกี่ยวข้อง

ชุดที่ 3 เป็นชุดข้อมูลที่ผู้วิจัยทำการดาวน์โหลดจากระบบรายงานข้อผิดพลาดของโครงการ Apache ซึ่งเป็นซอฟต์แวร์โอเพ่นซอร์สที่แตกต่างจากชุดข้อมูลก่อนหน้า โดยที่ข้อมูลในชุดนี้มีโครงสร้างคล้ายกับชุดที่ 1 โดยประกอบด้วย summary และ description ที่ให้ข้อมูลเกี่ยวกับข้อผิดพลาดของซอฟต์แวร์ อย่างไรก็ตาม ชุดข้อมูลนี้ยังคงประสบปัญหาความไม่สมดุลของข้อมูล โดยเฉพาะในคลาส trivial ซึ่งมีจำนวนน้อยที่สุดเพียง 37 รายการ ขณะที่คลาส major มีมากถึง 843 รายการ ความไม่สมดุลนี้ส่งผลต่อความสามารถของโมเดลในการเรียนรู้รูปแบบของข้อมูลในคลาสที่มีจำนวนน้อย เนื่องจากโมเดลอาจไม่ได้รับตัวอย่างที่เพียงพอในการฝึกเพื่อให้สามารถจำแนกคลาส trivial ได้อย่างถูกต้อง ดังนั้น งานวิจัยนี้จึงได้นำชุดข้อมูล Apache มาใช้เป็นชุดข้อมูลทดสอบสำหรับการประเมินผลของเทคนิคการแก้ปัญหาความไม่สมดุลของข้อมูลที่นำเสนอ โดยเปรียบเทียบผลลัพธ์ของโมเดลที่ผ่านการฝึกด้วยข้อมูลที่เสริมแล้วกับโมเดลที่ใช้ข้อมูลดั้งเดิม เพื่อวิเคราะห์ว่าแนวทางที่นำเสนอสามารถช่วยปรับปรุงความแม่นยำของการจำแนกระดับความรุนแรงของข้อผิดพลาดได้มากน้อยเพียงใด

3.2 กรอบงานภาพรวมการดำเนินงานวิจัย

หัวข้อนี้เป็นการนำเสนอภาพรวมขั้นตอนการดำเนินการวิจัย โดยแสดงดังรูปที่ 3.3



รูปที่ 3.3 ภาพรวมขั้นตอนการดำเนินการวิจัย

จากรูป 3.3 แสดงกรอบการทำงานวิจัยโดยรวม ประกอบด้วย 4 ขั้นตอนหลัก คือ การเตรียมชุดข้อมูลและการแบ่งข้อมูล การเสริมข้อมูล (Augmentation Dataset) การสร้างโมเดลจำแนกระดับความรุนแรง และการประเมินประสิทธิภาพโมเดล

1. การเตรียมชุดข้อมูลและการแบ่งข้อมูล

เริ่มต้นด้วยการนำข้อมูลรายงานข้อผิดพลาด (Bug Reports) มาทำการเตรียมชุดข้อมูลให้พร้อมใช้งานจากนั้น แบ่งชุดข้อมูลออกเป็น 2 ส่วนหลัก Train Dataset ขนาด 80% ใช้สำหรับการฝึกโมเดล และ Test Dataset 20% ใช้สำหรับการประเมินผลการทำงานของโมเดล การแบ่งชุดข้อมูลช่วยป้องกันการเกิดข้อมูลรั่วไหล (Data Leakage) และทำให้การประเมินผลมีความน่าเชื่อถือมากขึ้น

2. การเสริมข้อมูล

งานวิจัยนี้ได้ทำการศึกษาการจำแนกระดับความรุนแรงของจุดบกบกพร่องซอฟต์แวร์ โดยใช้ชุดข้อมูลรายงานจุดบกพร่อง ซึ่งหนึ่งในปัญหาของชุดข้อมูลดังนี้ คือ ความไม่สมดุลของจำนวนรายงานแต่ละคลาส ดังนั้นเพื่อเป็นการวิเคราะห์และประเมินวิธีการสร้างโมเดลที่มีประสิทธิภาพมากยิ่งขึ้น การเสริมข้อมูลจึงถูกนำมาใช้เพื่อวิเคราะห์และประเมินประสิทธิภาพโมเดลในปัญหาดังกล่าว โดยผู้วิจัยได้นำข้อมูลจาก Train Dataset ทำการสร้างชุดข้อมูลเสริม (Augmentation Dataset)

เพื่อเพิ่มความหลากหลายของข้อมูลและช่วยลดปัญหาการเรียนรู้ที่ไม่เพียงพอ มาใช้เพื่อปรับสมดุลของข้อมูล โดยแบ่งเป็น 3 แนวทางหลัก คือ

SMOTE (Synthetic Minority Oversampling Technique) สำหรับโมเดลที่เป็น Machine Learning เช่น SVM Logistic Regression และ Random Forest

Class Weight ใน BERT การใช้ Class Weight เป็นเทคนิคที่ช่วยจัดการกับปัญหาความไม่สมดุลของข้อมูล (Imbalanced Data) ซึ่งพบได้บ่อยในการจำแนกข้อมูลหลายคลาส เช่น การจำแนกระดับความรุนแรงของจุดบกพร่อง ในกรณีที่บางคลาสมีตัวอย่างน้อยกว่า โมเดลอาจให้ความสำคัญกับคลาสที่มีจำนวนตัวอย่างมากกว่า ทำให้การทำนายคลาสที่พบไม่บ่อยมีความแม่นยำน้อย

การสร้างข้อมูลใหม่โดยใช้โมเดล Transformer (T5 Summarization) ซึ่งใช้เทคนิคการสร้างข้อความจากโมเดลภาษาธรรมชาติ (NLP) เพื่อสร้างข้อความสรุปและคำอธิบายใหม่จากรายงานจุดบกพร่องเดิม ซึ่งเป็นวิธีการหลักที่นำเสนอในงานวิจัยนี้ และจะทำการทดลองกับทุกโมเดล ซึ่งทั้ง 3 วิธีการในการแก้ปัญหาข้อมูลไม่สมดุลจะกล่าวในรายละเอียดของหัวข้อต่อไป

3. การสร้างโมเดลจำแนกระดับความรุนแรง

กระบวนการนี้เป็นอีกหนึ่งกระบวนการที่สำคัญของงานวิจัย ซึ่งเป็นขั้นตอนสร้างโมเดลสำหรับการจำแนกระดับความรุนแรง โดยเปรียบเทียบกับชุดข้อมูลเดิมและชุดข้อมูลหลังจากการเสริมข้อมูล ประกอบด้วย 3 ขั้นตอนย่อย ดังนี้

Preprocessing เป็นขั้นตอนทำความสะอาดข้อมูล ได้แก่ การลบข้อมูลที่ไม่จำเป็น เช่น อักขระพิเศษ

Text Representation and Features Selection เป็นขั้นตอนการแปลงข้อความให้อยู่ในรูปแบบตัวเลข โดยใช้วิธี TF-IDF และเลือกฟีเจอร์ที่มีความสำคัญ

Models Training เป็นขั้นตอนการใช้ข้อมูลจาก Train Dataset และ Augmentation Dataset ในการฝึกโมเดลจำแนกระดับความรุนแรง โดยเปรียบเทียบโมเดลแบบการเรียนรู้ของเครื่อง ได้แก่ Logistic Regression Support Vector Machine Random Forest และ Ensemble Model แบบ Stacking รวมถึงโมเดลการเรียนรู้เชิงลึก ได้แก่ LSTM และ BERT

4. การประเมินประสิทธิภาพโมเดล (Model Evaluation and Comparison)

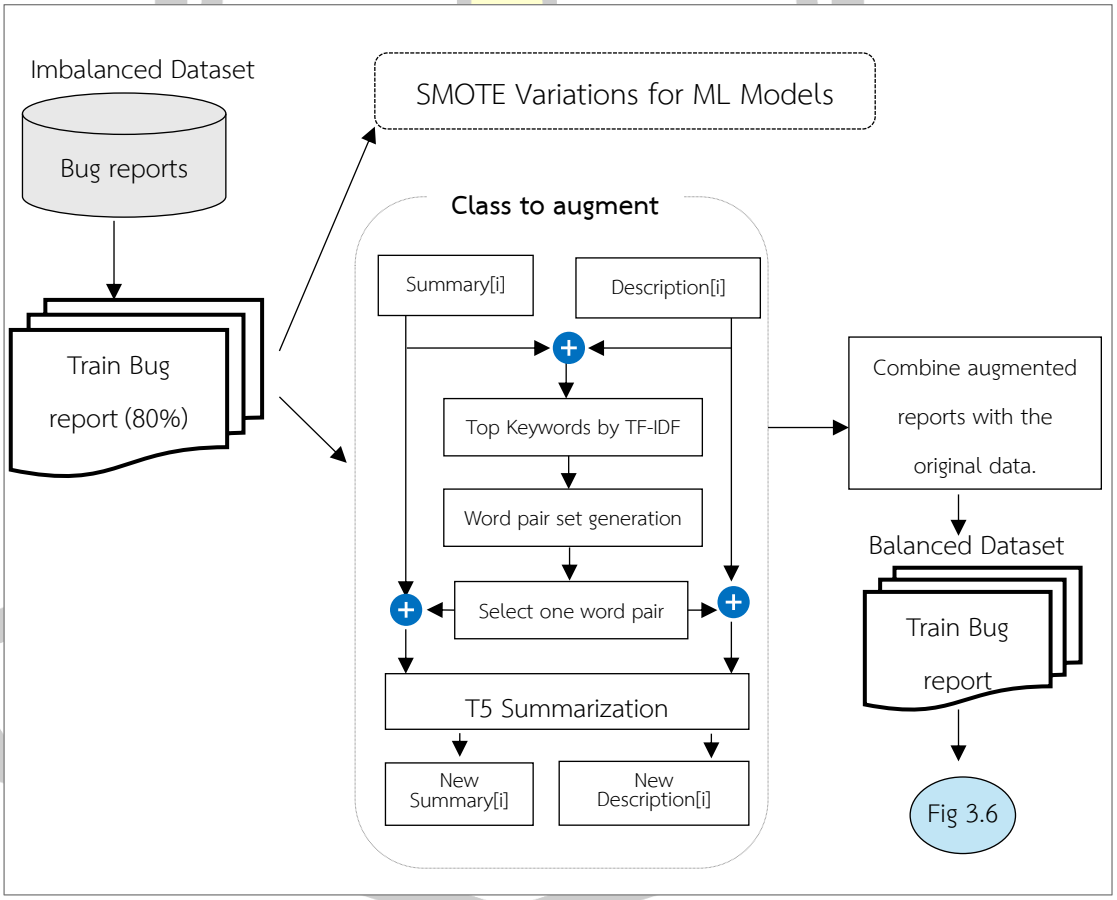
การประเมินประสิทธิภาพของแต่ละโมเดลที่สร้างไว้ใช้ชุดข้อมูลทดสอบในการประเมินผลโมเดลที่ฝึกเสร็จแล้ว เพื่อวัดความสามารถในการทำนายระดับความรุนแรงของโมเดล

เปรียบเทียบโมเดลต่าง ๆ ที่พัฒนา โดยใช้ตัวชี้วัดประสิทธิภาพ (Performance Metrics) เช่น Accuracy, Precision, Recall, F1-Score, MCC และ Area Under the Curve (AUC)

การทดลองทั้งหมด รวมถึงขั้นตอนการเสริมข้อมูล การฝึกโมเดล และการประเมินผล ถูกดำเนินการบนแพลตฟอร์ม Google Colab Pro+ ซึ่งเป็นสภาพแวดล้อมการประมวลผลแบบคลาวด์ที่สนับสนุนการใช้งานทรัพยากรประมวลผลประสิทธิภาพสูง สำหรับกระบวนการเสริมข้อมูลและการฝึกโมเดลการเรียนรู้ของเครื่องแบบดั้งเดิม รวมถึงโมเดล LSTM ได้รับการดำเนินการโดยใช้หน่วยประมวลผลกลาง (CPU) ในขณะที่กระบวนการฝึกโมเดล BERT อาศัยการประมวลผลด้วยหน่วยประมวลผลกราฟิก (GPU A100) เพื่อเพิ่มประสิทธิภาพในการคำนวณและลดระยะเวลาการฝึกโมเดล

3.3 กรอบงานการเสริมข้อมูล

การเสริมข้อมูลเป็นอีกหนึ่งเทคนิคในการเพิ่มประสิทธิภาพโมเดล โดยเฉพาะกับชุดข้อมูลที่มีความไม่สมดุล ซึ่งงานวิจัยนี้ได้นำเทคนิคการเสริมข้อมูลมาประยุกต์ใช้ กรอบงานการเสริมข้อมูลแสดงดังรูป



รูปที่ 3.4 ขั้นตอนการเสริมข้อมูล

รูปที่ 3.4 แสดงกระบวนการในการเสริมข้อมูล (Data Augmentation) สำหรับชุดข้อมูลที่ไม่สมดุล (Imbalanced Dataset) โดยการนำข้อมูลชุดฝึกจำนวน 80% ซึ่งเป็นชุดเดียวกันมาทำการเสริมข้อมูล โดยการวิจัยนี้นำเสนอการเสริมข้อมูล 2 รูปแบบ ได้แก่ วิธีมาตรฐานด้วยวิธี SMOTE (Synthetic Minority Oversampling Technique) ในรูปแบบต่าง ๆ สำหรับโมเดลแบบการเรียนรู้ของเครื่อง และผู้วิจัยทำเสนอวิธีการเสริมข้อมูลด้วยเทคนิคการสร้างข้อความใหม่โดยนำโมเดลทรานฟอร์เมอร์มาใช้ การเสริมข้อมูลมีเป้าหมายเพื่อปรับสมดุลของข้อมูลในกลุ่มที่มีจำนวนน้อย กระบวนการนี้มีรายละเอียดในแต่ละขั้นตอนดังนี้

3.3.1 การใช้ SMOTE

การวิจัยนี้มุ่งเน้นการแก้ปัญหาข้อมูลไม่สมดุลของรายงานจุดบกพร่อง เพื่อเปรียเทียบวิธีการที่นำเสนอและแนวทางพื้นฐานที่มีอยู่ปัจจุบันของการแก้ปัญหาข้อมูลไม่สมดุล วิธีการ SMOTE (Synthetic Minority Oversampling Technique) จึงถูกนำมาเป็นตัวอย่างเปรียบเทียบ โดยหลักการทำงานของ SMOTE จะเป็นการสร้างตัวอย่างใหม่ในกลุ่มข้อมูลที่มีจำนวนน้อย SMOTE ช่วยสร้างข้อมูลใหม่โดยใช้การ Interpolation ระหว่างตัวอย่างที่มีอยู่ใน Feature Space ซึ่งเหมาะสำหรับข้อมูลเชิงตัวเลข สำหรับงานที่เกี่ยวกับข้อความต้องทำการแทนข้อความด้วยตัวเลขก่อนการทำ SMOTE ขั้นตอนการสร้างข้อมูลใหม่โดยการเพิ่มตัวอย่างในกลุ่ม minority class ด้วยวิธีการดังนี้

1. เลือกตัวอย่าง minority class
    - เลือกตัวอย่างข้อมูล (data point) จากกลุ่ม minority class แบบสุ่ม
2. หา k-nearest neighbors
    - ใช้ k-Nearest Neighbors (k-NN) เพื่อหา k ตัวอย่างที่อยู่ใกล้กับตัวอย่างที่เลือกในขั้นตอนที่ 1
3. สร้างข้อมูลใหม่ (Synthetic Data)
    - เลือกตัวอย่าง neighbor ที่ได้มาหนึ่งตัวอย่าง (สุ่มจาก k neighbors)
    - สร้างตัวอย่างใหม่ (synthetic sample) โดยการคำนวณตำแหน่งใหม่ระหว่างตัวอย่างที่เลือก (step 1) กับ neighbor ตัวอย่างที่สุ่ม (step 2) ตามสูตร

$$\text{Synthetic sample} = x_i + \lambda \times (x_{nn} - x_i) \quad (3.1)$$

- x_i ตัวอย่าง minority class ที่เลือก
- x_{nn} หนึ่งใน k-nearest neighbors

- λ ค่าที่สุ่มในช่วง [0, 1]

4. ทำซ้ำตามจำนวนตัวอย่างที่ต้องการ โดยกระบวนการข้างต้นจะถูกทำซ้ำเพื่อสร้างตัวอย่าง synthetic ให้ครบจำนวนที่กำหนด

ตารางที่ 3.2 การเพิ่มข้อมูลชุดฝึกด้วย SMOTE

Class distribution before SMOTE	{'major': 1509, 'critical': 1228, 'minor': 989, 'trivial': 474, 'blocker': 121})
Apply SMOTE	SMOTE(random_state=42) X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)
Class distribution after SMOTE	{'major': 1509, 'minor': 1509, 'critical': 1509, 'trivial': 1509, 'blocker': 1509}

ตารางนี้แสดงให้เห็นถึงผลกระทบของการใช้ SMOTE (Synthetic Minority Over-sampling Technique) ในการปรับสมดุลของข้อมูลสำหรับการจำแนกระดับความรุนแรงของจุดบกพร่อง โดยแบ่งเป็น 3 ส่วนหลัก

1) จำนวนข้อมูลก่อน ใช้ SMOTE โดยก่อนใช้ SMOTE ข้อมูลมีการกระจายตัวที่ไม่สมดุล โดยมีจำนวนตัวอย่างในแต่ละคลาสโดยเรียงตามลำดับความรุนแรงจากรุนแรงสุดไปรุนแรงน้อยสุด ดังนี้ Blocker จำนวน 121 แถว Critical จำนวน 1,228 แถว Major จำนวน 1,509 แถว Minor จำนวน 989 แถว และ Trivial จำนวน 474 แถว จากข้อมูลนี้ คลาส Blocker และ Trivial มีจำนวนตัวอย่างน้อยมากเมื่อเทียบกับคลาส Major และ Critical ซึ่งอาจทำให้โมเดลเรียนรู้ได้ไม่ดีและให้ความสำคัญกับคลาสที่มีจำนวนตัวอย่างมากกว่า

2) การใช้ SMOTE โดยประกาศฟังก์ชัน SMOTE(random_state=42) เพื่อสร้างตัวอย่างใหม่ในคลาสที่มีจำนวนน้อย โดยอาศัยการคำนวณระยะห่างระหว่างจุดข้อมูลที่มีอยู่ แล้วสร้างตัวอย่างสังเคราะห์ที่คล้ายกับจุดข้อมูลเดิม โดยนำข้อมูลชุดฝึกและคลาสชุดฝึก (X_train, y_train) ที่ถูกเตรียมข้อมูลและทรานฟอร์มแล้วเข้าไปยังฟังก์ชัน SMOTE ซึ่งเป็นชุดไลบารี่ของ Imbalanced-learn โปรแกรมภาษาไพธอน จากนั้นจะได้ข้อมูลชุดใหม่ (X_train_resampled, y_train_resampled) ตามรูปแบบวิธีการกำหนดตัวอย่างหรือ Sampling ซึ่งงานวิจัยนี้นำวิธีการ SMOTE แบบ Oversampling โดยจำนวนตัวอย่างของทุกคลาสถูกเพิ่มขึ้นจนเท่ากับคลาสที่มีมากที่สุด

3) จำนวนข้อมูลหลัง SMOTE จะได้ตัวอย่างข้อมูลใหม่ที่มีจำนวนเท่ากับคลาสที่มีจำนวนมากสุด คือ Blocker จำนวน 1,509 แถว Critical จำนวน 1,509 แถว Major จำนวน 1,509 แถว Minor จำนวน 1,509 แถว และ Trivial จำนวน 1,509 แถว เพื่อนำข้อมูลดังกล่าวไปใช้กับโมเดลการฝึกต่อไป

นอกจากการใช้ SMOTE เพื่อปรับให้ข้อมูลจำนวนคลาสนี้เพิ่มขึ้นเท่ากับจำนวนคลาสที่มากที่สุดแล้ว ยังสามารถกำหนดกลยุทธ์ให้ SMOTE ได้โดยการกำหนดคลาสตามความเหมาะสมได้อีกด้วย แต่ต้องไม่น้อยกว่าจำนวนคลาสที่มีอยู่ ยกตัวอย่างเช่น สามารถกำหนดให้คลาส blocker เพิ่มเป็น 500 หรือ trivial เพิ่มเป็น 1000 ได้ เป็นต้น ซึ่งการกำหนดแบบกลยุทธ์นี้ผู้วิจัยจะนำเปรียบเทียบกับการเสริมข้อมูลด้วยจำนวนเท่ากัน เพื่อเป็นการเปรียบเทียบกับวิธีมาตรฐานแบบ SMOTE กับการเสริมข้อมูลที่นำเสนอ

3.3.2 การใช้ Class Weight

จากหัวข้อก่อนหน้าได้อธิบายเกี่ยวกับการนำ SMOTE มาใช้งานเพื่อเปรียบเทียบกับการเสริมข้อมูลโดยตรงของงานวิจัยนี้ แต่อย่างไรก็ตาม SMOTE ไม่สามารถใช้ร่วมกับโมเดล BERT ได้ ผู้วิจัยจึงนำวิธีการ Class Weight มาใช้ เพื่อประยุกต์ร่วมกันโมเดล BERT สำหรับ Class Weights ถูกใช้เพื่อลดอคติที่เกิดจากความไม่สมดุลของคลาส ในปัญหาการจำแนกระดับความรุนแรงจากจุดบกพร่องสามารถนำมาประยุกใช้ร่วมกับ BERT ได้ โดยขั้นตอนการทำงานหลัก มีดังนี้

ใช้ compute_class_weight() จาก sklearn.utils.class_weight เพื่อคำนวณน้ำหนักของแต่ละคลาสโดยอัตโนมัติ โดยพิจารณาจำนวนตัวอย่างที่มีอยู่ในแต่ละคลาส

ค่าของ Class Weights ถูกกำหนดตามสูตร

$$W_i = \frac{N}{n_i} \tag{3.2}$$

โดยที่

W_i= ค่าของ Class Weight สำหรับคลาส i

n = จำนวนตัวอย่างทั้งหมดในชุดข้อมูล

n_i= จำนวนตัวอย่างในคลาส i

จากนั้น ค่า Class Weights ถูกแปลงเป็น Torch Tensor (class_weights_tensor) และย้ายไปที่ GPU เพื่อให้ใช้งานได้ในการฝึกโมเดล

```python
# คำนวณค่า class weights
class_weights = compute_class_weight(
    class_weight="balanced",
    classes=np.unique(train_df[label_column]),
    y=train_df[label_column].values
)

class_weights_tensor = torch.tensor(class_weights, dtype=torch.float).to("cuda")
```

รูปที่ 3.5 การเรียกใช้งาน Class Weights ในโมเดล BERT ภาษาไพธอน

การใช้ Class Weights ช่วยลดอคติของโมเดลที่มักจะทำนายคลาสที่มีจำนวนตัวอย่างมาก ทำให้โมเดลให้ความสำคัญกับคลาสที่มีตัวอย่างน้อยมากขึ้น นอกจากนี้ ยังช่วยเพิ่มค่า F1-score และ Recall สำหรับคลาสที่พบได้น้อย ส่งผลให้การจำแนกมีความแม่นยำและครอบคลุมมากขึ้นในทุกคลาส ซึ่งช่วยให้โมเดลสามารถเรียนรู้จากข้อมูลที่ไม่สมดุลได้ดีขึ้น

3.3.3 การเสริมข้อมูลแบบ EDA

การเสริมข้อมูลข้อความโดยใช้เทคนิค Easy Data Augmentation (EDA) เป็นหนึ่งในวิธีการที่ได้รับความนิยมสำหรับการจัดการปัญหาความไม่สมดุลของข้อมูลประเภทข้อความ เทคนิคดั้งเดิมที่ใช้ใน EDA [68] ประกอบด้วยการแทนที่คำพ้องความหมาย การแทรกคำแบบสุ่ม การสลับลำดับคำแบบสุ่ม และการลบคำแบบสุ่ม เป็นต้น ซึ่งได้รับการนำมาใช้ในงานวิจัยด้านการประมวลผลภาษาธรรมชาติ (NLP) อย่างแพร่หลาย งานวิจัยนี้เลือกใช้เทคนิคการแทนที่คำพ้องความหมายเพื่อเปรียบเทียบประสิทธิภาพกับวิธีการที่นำเสนอ [69] โดยใช้ฐานข้อมูลคำพ้องความหมายจาก WordNet เป็นแหล่งข้อมูลหลักสำหรับการสร้างชุดข้อมูลเสริม ทั้งนี้ กระบวนการเสริมข้อมูลถูกกำหนดให้มีอัตราส่วนที่สอดคล้องกับแนวทางหลักของการวิจัยที่นำเสนอ เพื่อประเมินผลกระทบของเทคนิคดังกล่าวต่อประสิทธิภาพของโมเดลการจำแนกข้อความ

3.3.4 การเสริมข้อมูลด้วยโมเดล T5 Summarization

งานวิจัยนี้นำเสนอแนวทางในการเสริมข้อมูลสำหรับคลาสที่มีจำนวนน้อยในชุดข้อมูลรายงานจุดบกพร่อง ซึ่งเป็นแนวทางสำคัญที่ช่วยปรับปรุงความสมดุลของข้อมูลและเพิ่มประสิทธิภาพของโมเดลในการจำแนกระดับความรุนแรงของข้อผิดพลาด โดยแนวคิดหลักของการเสริมข้อมูลในงานวิจัยนี้คือการสร้างข้อมูลรายงานแถวใหม่ขึ้นมาโดยอาศัยกระบวนการสกัดข้อความที่มีความสำคัญจากรายงานแถวเดิม ซึ่งช่วยให้ข้อมูลที่สร้างขึ้นมานั้นยังคงความสัมพันธ์กับข้อมูลต้นฉบับแต่เพิ่มความหลากหลายของข้อความให้โมเดลสามารถเรียนรู้ได้อย่างครอบคลุมมากขึ้น ในการดำเนินการเสริม

ข้อมูล งานวิจัยนี้ได้นำโมเดลภาษาธรรมชาติ T5 [36, 70] ใช้เป็นเครื่องมือหลักสำหรับสร้างข้อมูลใหม่ โดยโมเดล T5 ถูกออกแบบมาให้สามารถประมวลผลข้อความและสร้างสรุปของเนื้อหาหรือข้อความใหม่ที่คงความหมายเดิมได้อย่างมีประสิทธิภาพ ซึ่งเหมาะสมกับงานวิจัยนี้ที่ต้องการเพิ่มจำนวนข้อมูลโดยไม่สูญเสียสาระสำคัญ กระบวนการเสริมข้อมูลนี้สามารถแบ่งออกเป็นหลายขั้นตอนสำคัญ ได้แก่ การดึงคำสำคัญจากรายงานเดิมโดยใช้เทคนิค TF-IDF เพื่อระบุคำที่มีความสำคัญต่อเนื้อหา จากนั้นนำคำสำคัญเหล่านี้มาสร้างเป็นข้อความใหม่โดยให้โมเดล T5 ทำการสรุปข้อมูลหรือสร้างคำอธิบายใหม่ที่เกี่ยวข้องกับข้อผิดพลาด โดยที่ข้อความที่สร้างขึ้นต้องมีโครงสร้างใกล้เคียงกับข้อมูลต้นฉบับเพื่อรักษาความสมจริง นอกจากนี้ยังมีการปรับแต่งกระบวนการสุ่มเลือกคำสำคัญและกำหนดจำนวนแถวใหม่ที่ต้องการเสริมในแต่ละคลาสให้เหมาะสม เพื่อให้ข้อมูลที่ได้มีความสมดุลมากขึ้นและสามารถช่วยลดปัญหาการเรียนรู้ที่ไม่เพียงพอในคลาสที่มีจำนวนน้อย ทั้งหมดนี้ช่วยให้โมเดลสามารถเรียนรู้รูปแบบของข้อมูลได้ดียิ่งขึ้นและเพิ่มความแม่นยำในการจำแนกระดับความรุนแรงของจุดบกพร่องในซอฟต์แวร์ โดยกระบวนการแต่ละขั้นตอนอธิบายดังต่อไปนี้

1) กระบวนการระบุคลาสและจำนวนแถวที่ต้องการสร้างใหม่อาศัยหลักการ การสุ่มเพิ่มข้อมูล (over sampling) โดยมุ่งเน้นการเพิ่มข้อมูลในคลาสที่มีจำนวนน้อยที่สุด เพื่อให้สมดุลกับคลาสที่มีจำนวนมากกว่า โดยในกระบวนการทดลองนี้จะทำการสุ่มเพิ่มข้อมูลจากคลาสที่มีจำนวนน้อยที่สุดให้มีจำนวนใกล้เคียงกับคลาสที่มีจำนวนน้อยถัดไป ยกตัวอย่างเช่น หากคลาส blocker มีจำนวนแถวเพียง 121 แถว และต้องการเพิ่มให้มีจำนวนใกล้เคียงกับคลาส trivial ที่มี 474 แถว การเพิ่มข้อมูลจะทำโดยการขยายจำนวนแถวของ blocker เป็น 3 เท่า ของจำนวนเดิม (121 × 3 = 363) ซึ่งเมื่อรวมกับข้อมูลเดิม จะทำให้ blocker มีทั้งหมด 484 แถว อย่างไรก็ตาม แม้ว่าการสุ่มเพิ่มข้อมูลจะช่วยลดปัญหาความไม่สมดุลของข้อมูล แต่ในบางกรณีอาจไม่ส่งผลให้ผลลัพธ์การจำแนกดีขึ้นเสมอไป ดังนั้นจึงจำเป็นต้องมีการทดลองปรับจำนวนหลายรูปแบบเพื่อหาค่าที่เหมาะสมที่สุด นอกจากนี้ การเพิ่มข้อมูลโดยไม่มีการควบคุมอาจทำให้เกิดปัญหา Overfitting และ Bias ในข้อมูล ดังนั้นจึงควรเสริมข้อมูลเฉพาะคลาสที่จำเป็นเท่านั้น และสามารถคำนวณจำนวนแถวของคลาสที่ต้องการเสริมโดยใช้สมการที่กำหนดเพื่อช่วยให้กระบวนการเสริมข้อมูลมีโครงสร้างที่เป็นระบบมากขึ้น ซึ่งสามารถเขียนเป็นสมการในการคำนวณจำนวนแถวของคลาสที่ต้องการเสริมเริ่มต้น ได้ดังนี้

$$N_{new} = N_{min} + \left(\left[\frac{N_{next}}{N_{min}}\right] \times N_{min}\right) \quad (3.3)$$

โดยที่

N_{min} = จำนวนตัวอย่างของคลาสน้อยสุด

N_{next} = จำนวนตัวอย่างของคลาสที่ถัดจากคลาสน้อยสุด

2) ขั้นตอนการนำเข้าข้อมูลของคลาสที่ถูกกำหนดให้เสริมข้อมูลเริ่มต้นด้วยการคัดเลือกรายงานที่อยู่ในกลุ่มของคลาสที่มีจำนวนน้อยตามที่กำหนด ซึ่งต้องการเพิ่มจำนวนข้อมูลเพื่อให้สมดุลกับคลาสอื่น ๆ ในชุดข้อมูล รายงานที่ถูกเลือกมาประกอบด้วยสององค์ประกอบหลัก ได้แก่ Summary[i] ซึ่งเป็นข้อความสั้นที่สรุปปัญหาของจุดบกพร่อง และ Description[i] ซึ่งเป็นข้อความที่ให้รายละเอียดเพิ่มเติมเกี่ยวกับข้อผิดพลาดนั้น ขั้นตอนนี้ดำเนินการตามจำนวนรอบที่กำหนดสำหรับการเสริมข้อมูลในแต่ละคลาส โดยอาศัยหลักการการสุ่มเพิ่มเพื่อให้แน่ใจว่ามีการเพิ่มข้อมูลอย่างเป็นระบบ ทั้งนี้ การกำหนดจำนวนรอบของการนำเข้าข้อมูลสำหรับแต่ละคลาสขึ้นอยู่กับอัตราส่วนของจำนวนข้อมูลเดิมกับจำนวนข้อมูลเป้าหมายที่ต้องการเสริม ซึ่งได้จากสมการการเพิ่มข้อมูลที่คำนวณไว้ในขั้นตอนก่อนหน้าเป็นแนวทางในการดำเนินการ กระบวนการนี้ไม่เพียงแต่ช่วยให้คลาสที่มีข้อมูลน้อยสามารถมีจำนวนข้อมูลเพียงพอสำหรับการฝึกโมเดล แต่ยังช่วยให้โมเดลสามารถเรียนรู้รูปแบบของข้อมูลในคลาสที่มีจำนวนน้อยได้อย่างแม่นยำยิ่งขึ้น อย่างไรก็ตาม เพื่อป้องกันปัญหา Overfitting ที่อาจเกิดขึ้นจากการเพิ่มข้อมูลมากเกินไป จำเป็นต้องมีการตรวจสอบความหลากหลายของข้อมูลที่ถูกสร้างขึ้น และปรับจำนวนรอบของการนำเข้าให้เหมาะสมกับแต่ละคลาส เพื่อให้ได้ชุดข้อมูลที่มีความสมดุลและสามารถนำไปใช้กับการฝึกโมเดลได้อย่างมีประสิทธิภาพ

3) ขั้นตอนการดึงคำสำคัญจากข้อมูลเดิมเป็นกระบวนการสำคัญที่ช่วยให้การเสริมข้อมูลมีคุณภาพและสอดคล้องกับเนื้อหาของรายงานจุดบกพร่อง โดยในขั้นตอนนี้จะใช้เทคนิค TF-IDF เพื่อวิเคราะห์ความสำคัญของแต่ละคำใน Summary[i] และ Description[i] ซึ่งเป็นสององค์ประกอบหลักของรายงานจุดบกพร่อง TF-IDF ทำงานโดยการคำนวณความถี่ของคำที่ปรากฏในรายการหรือแถวเดียวกัน (term frequency, TF) และเปรียบเทียบกับความถี่ที่คำดังกล่าวปรากฏในเอกสารทั้งหมด (inverse document frequency, IDF) ซึ่งช่วยให้สามารถระบุคำที่มีความสำคัญต่อบริบทของข้อมูลได้อย่างแม่นยำ กระบวนการนี้เริ่มต้นจากการแยกคำในรายการออกเป็นหน่วยที่สามารถวิเคราะห์ได้ (tokenization) จากนั้นทำการลบคำที่ไม่มีความสำคัญ เช่น คำสันธาน (stopwords) และนำข้อมูลที่ผ่านการทำความสะอาดไปคำนวณคะแนน TF-IDF คำที่มีค่าคะแนนสูงสุด (top k terms) จะถูกเลือกเป็น keywords ที่สำคัญ โดยในการวิจัยนี้กำหนดให้เลือก 6 คำที่มีคะแนนสูงสุด ซึ่งจำนวน 6 คำนี้ได้จากการทดลองเบื้องต้นพบว่าให้ความสมดุลระหว่างความกระชับของข้อมูลและความครอบคลุมของบริบทที่สำคัญเพียงพอสำหรับกระบวนการสร้างข้อความใหม่ผ่านโมเดล T5 ทั้งนี้แนวคิดการเลือก top-k terms จาก TF-IDF เป็นเทคนิคมาตรฐานที่พบในงานวิจัยหลายฉบับ [71, 72] กระบวนการนี้มีความสำคัญอย่างยิ่งในการรักษาความสอดคล้องของข้อมูลที่ถูกเสริม โดยช่วยให้ข้อความที่สร้างขึ้นใหม่สามารถสะท้อนถึงบริบทของรายงานเดิมได้ดีขึ้น นอกจากนี้ การใช้ TF-IDF ยังช่วยลดโอกาสที่คำที่ไม่เกี่ยวข้องหรือคำที่เกิดขึ้นบ่อยแต่ไม่มีความหมายสำคัญ (common words) จะถูกนำไปใช้ใน

กระบวนการสร้างข้อมูล ทำให้ข้อมูลที่ถูกเสริมมีคุณภาพสูงและสามารถช่วยให้โมเดลเรียนรู้บริบทของจุดบกพร่องในซอฟต์แวร์ได้อย่างแม่นยำมากขึ้น

$$K_i = TF - IDF(S_i, D_i)_{top=k} \tag{3.4}$$

ตัวอย่างผลลัพธ์ที่ได้

K = {“error", "failure", "bug", "..."}

4) ในขั้นตอนนี้ คำสำคัญที่ถูกคัดเลือกจากกระบวนการ TF-IDF ในขั้นตอนก่อนหน้าจะถูกนำมาประมวลผลเพิ่มเติมเพื่อนำไปใช้ในกระบวนการสร้างข้อมูลใหม่ โดยหลักการที่ใช้งานคือการสร้างเซตของคู่คำที่เป็นไปได้ (word pair set generation) เพื่อให้สามารถเพิ่มความหลากหลายของข้อความที่ถูกเสริมได้มากขึ้น การสร้างเซตของคู่คำทำโดยการจับคู่คำสำคัญสองคำจากชุดคำสำคัญ (K) ที่ได้รับจาก TF-IDF โดยไม่อนุญาตให้คำเดียวกันถูกจับคู่กับตัวเอง (i ≠ j) ซึ่งสามารถแสดงได้ตามสมการที่กำหนด สมการ 3.5 แสดงการสร้างเซตของคู่คำ

$$P = \{(k_i,\ k_j) \mid k_i,\ k_j \in K\ ,\ i \neq j\} \tag{3.5}$$

ตัวอย่างผลลัพธ์ที่ได้

P = {(error, failure), (error, bug), (failure, bug), (..., ...)}

โดยที่ $k_i,\ k_j$ แทนคำสำคัญที่ถูกเลือกจากแต่ละ Summary[i] และ Description[i] และเซต P จะแสดงถึงคู่คำทั้งหมดที่สามารถสร้างได้จากคำสำคัญในแต่ละรายงาน จากนั้นทำการสุ่มเลือก 1 คู่คำจากเซต P เพื่อนำไปใช้เป็นข้อมูลป้อนเข้าสำหรับกระบวนการสร้างข้อมูลเสริมในขั้นตอนถัดไป การเลือกคู่คำแบบสุ่มช่วยให้ข้อความที่ถูกสร้างขึ้นมีความหลากหลาย และช่วยป้องกันการเกิดรูปแบบที่ซ้ำซ้อนหรือมีโครงสร้างที่แน่นอนเกินไป ซึ่งอาจทำให้โมเดลเกิด overfitting ในการเรียนรู้ได้ นอกจากนี้ กระบวนการนี้ยังช่วยเพิ่มความสมจริงของข้อมูลเสริม เนื่องจากคำสำคัญที่ถูกจับคู่กันนั้นถูกคัดเลือกโดยอิงตามบริบทของรายงานจุดบกพร่องจริง ซึ่งหมายความว่าคำที่ถูกเลือกมีแนวโน้มที่จะมีความเกี่ยวข้องกันและสามารถสะท้อนถึงลักษณะของข้อผิดพลาดในซอฟต์แวร์ได้อย่างถูกต้อง กระบวนการนี้เป็นรากฐานสำคัญสำหรับขั้นตอนการสร้างข้อความใหม่ผ่านโมเดลภาษาธรรมชาติ ซึ่งจะช่วยให้ข้อมูลเสริมมีคุณภาพสูงและสามารถนำไปใช้กับการฝึกโมเดลได้อย่างมีประสิทธิภาพมากขึ้น

สมการ 3.6 แสดงการสุ่ม 1 เซตของคู่คำ

$$p_i = random(P) \tag{3.6}$$

ตัวอย่างผลลัพธ์ที่ได้

p = (failure, bug)

5) ในขั้นตอนนี้ คำสำคัญที่ถูกสุ่มเลือกจากเซตของคู่คำที่สร้างขึ้นในขั้นตอนก่อนหน้าจะถูกนำมารวมเข้ากับข้อมูลรายงานจุดบกพร่องเดิม ซึ่งประกอบด้วย Summary[i] และ Description[i] เพื่อเตรียมข้อมูลสำหรับกระบวนการสร้างข้อความใหม่โดยใช้โมเดลภาษาธรรมชาติ T5 การรวมคำสำคัญเข้ากับข้อมูลต้นฉบับช่วยเพิ่มความสอดคล้องระหว่างข้อมูลเสริมที่ถูกสร้างขึ้นกับบริบทของรายงานเดิม โดยกระบวนการนี้สามารถแสดงในรูปของสมการดังนี้

$$S_i' = S_i \cup p_i \,,\; D_i' = D_i \cup p_i \tag{3.7}$$

โดยที่ S_i และ D_i แทน Summary[i] และ Description[i] ของรายงานจุดบกพร่องลำดับที่ i ส่วน p_i คือคู่คำสำคัญที่ถูกเลือกจากเซต P แล้วนำมาผนวกรวมเข้ากับข้อความต้นฉบับเพื่อขยายบริบทและเพิ่มข้อมูลเชิงลึกให้กับรายงาน การรวมคู่คำเหล่านี้ช่วยให้ข้อมูลใหม่ที่สร้างขึ้นสามารถสะท้อนถึงแนวโน้มของข้อมูลต้นฉบับได้ดีขึ้น รวมถึงช่วยให้โมเดล T5 มีบริบทที่ชัดเจนขึ้นในการสร้างข้อมูลใหม่ นอกจากนี้ กระบวนการนี้ยังช่วยลดปัญหาความซ้ำซ้อนของข้อมูลเสริมที่อาจเกิดขึ้นจากการใช้ข้อมูลเดิมโดยตรง เนื่องจากคู่คำที่เลือกมาแต่ละชุดมีความแตกต่างกันในแต่ละรอบของการประมวลผล ทำให้ข้อมูลที่ได้มีความหลากหลายมากขึ้น อย่างไรก็ตาม การเลือกคู่คำที่มีความสัมพันธ์กันมากเกินไปอาจทำให้ข้อมูลที่ถูกสร้างขึ้นมีลักษณะคล้ายกันเกินไป

6) ในขั้นตอนนี้ ข้อความที่ได้รับการปรับปรุงโดยการรวมคู่คำสำคัญกับ Summary (S_i') และ Description (D_i') จะถูกนำเข้าสู่กระบวนการสร้างข้อความใหม่โดยใช้โมเดลภาษาธรรมชาติ T5 ซึ่งเป็นโมเดลที่ถูกออกแบบมาให้สามารถจัดการงานด้าน NLP ได้หลายประเภท โดยเฉพาะงานที่เกี่ยวข้องกับการสรุปข้อความ การแปลภาษา และการตอบคำถาม ซึ่งงานวิจัยนี้ใช้การทำงานของ T5 เกี่ยวกับการสรุปข้อความ (Summarization) และโมเดล T5 base ถูกเลือกใช้เนื่องจากเป็นเวอร์ชันที่มีขนาดสมดุลระหว่างประสิทธิภาพและทรัพยากรที่ต้องใช้ในการประมวลผล

$$S_i^* = \mathrm{T5}(S_i') \,,\; D_i^* = \mathrm{T5}(D_i') \tag{3.8}$$

S_i^* คือ Summary ที่ถูกเสริมใหม่ D_i^* คือ Description ที่ถูกเสริมใหม่ ของแต่ละแถวตามคลาสที่ถูกกำหนด และทำการวนลอบไปตามจำนวนคลาสที่ต้องการเพิ่ม โดยการสุ่มคู่คำในสมการ 3.5 เป็นการเพิ่มความหลากหลายของข้อความใหม่ที่ถูกเสริมขึ้นและลดการซ้ำเดิมของข้อความ นอกจากนี้ เพื่อให้การสร้างข้อความใหม่มีประสิทธิภาพสูงสุดและลดการซ้ำเดิมของข้อความเดิมจากรายงานจุดบกพร่อง การวิจัยนี้ได้ปรับแต่ง hyperparameters ของโมเดล T5 อย่างเป็นระบบ โดยพิจารณาจากทั้งงานเอกสารที่เกี่ยวข้อง [73, 74] และการทดลองเบื้องต้น เพื่อให้ข้อความที่ได้มีความ

หลากหลายเพียงพอและยังคงรักษาบริบทดั้งเดิมได้อย่างเหมาะสม โดยกระบวนการนี้ปรับแต่ง hyperparameters ของโมเดล T5 ได้แก่ max_length, min_length, และ temperature เพื่อควบคุมความยาวและความหลากหลายของข้อความที่ถูกสร้างขึ้น แสดงค่าการตั้งค่าพารามิเตอร์

ตารางที่ 3.3 การตั้งค่า T5 ในการสร้างข้อมูลใหม่

Parameter	**ค่า**	**คำอธิบาย**
max_length	100 และ 200	กำหนดความยาวสูงสุดของข้อความที่สร้างขึ้น โดยตั้งไว้ที่ 100 สำหรับ summary และ 200 สำหรับ description เพื่อช่วยลดการซ้ำซ้อนและให้ข้อมูลที่ครอบคลุมมากขึ้น
do_sample	True	เปิดใช้งานการสุ่มเพื่อให้โมเดลเลือกคำที่หลากหลายและลดความซ้ำซ้อนของข้อความ
temperature	0.7	ค่าควบคุมความสุ่มในการเลือกคำ ค่าต่ำทำให้โมเดลเลือกคำที่เป็นไปได้มากที่สุด ค่าสูงขึ้นทำให้สุ่มมากขึ้น
top_k	40	จำกัดตัวเลือกของคำที่ใช้ในแต่ละรอบการเลือก ให้อยู่ใน 40 คำที่มีความน่าจะเป็นสูงสุด เพื่อลด noise
top_p	0.9	ใช้ Nucleus Sampling เพื่อเลือกคำที่มีโอกาสสะสมรวม ≥ 90% ป้องกันการเลือกคำที่มีโอกาสน้อยเกินไป

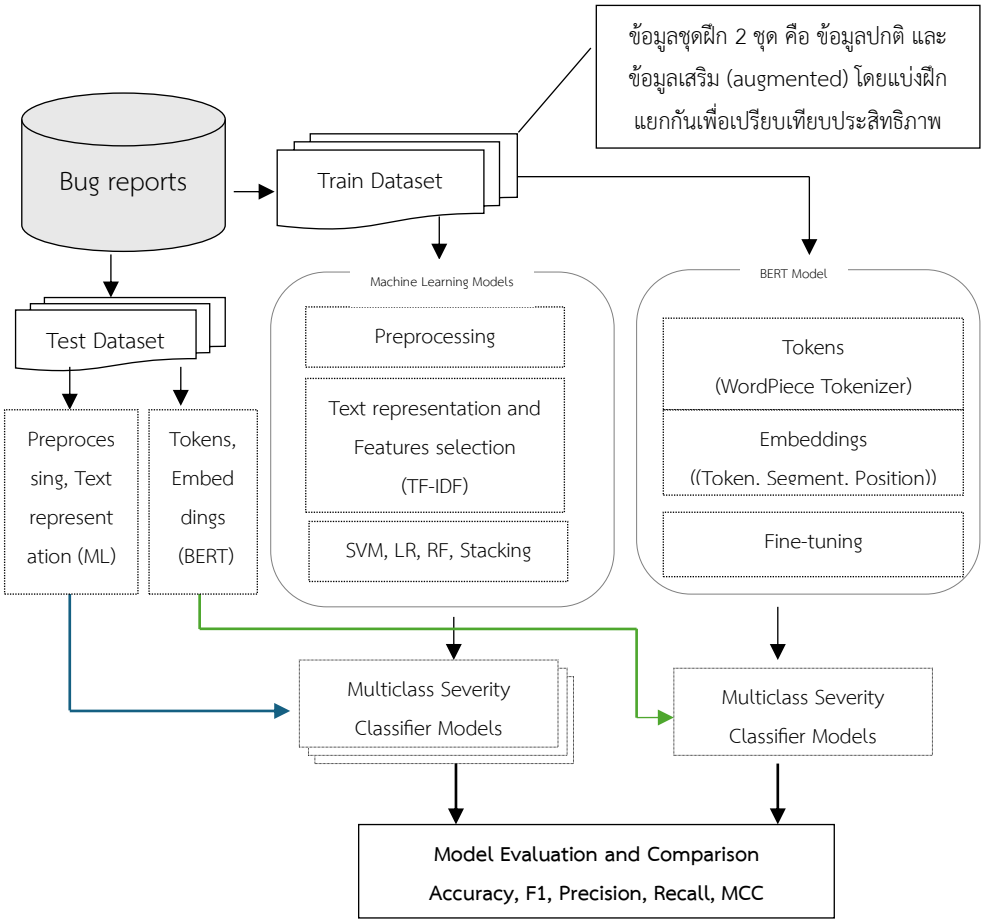
7. หลังจากที่ได้สร้างข้อมูลใหม่จากโมเดล T5 กระบวนการต่อไปคือการนำ Summary และ Description ที่ถูกสร้างขึ้นมาเพิ่มเติมเข้าสู่ชุดข้อมูลฝึกเพื่อให้โมเดลสามารถเรียนรู้ได้อย่างครอบคลุมมากขึ้น โดยข้อมูลใหม่เหล่านี้จะถูกเพิ่มเข้าไปในชุดข้อมูลเดิม พร้อมทั้งกำหนดค่าคลาสของแต่ละแถวให้ตรงกับคลาสต้นฉบับที่ใช้เป็นฐานในการสร้างข้อมูล กระบวนการนี้ช่วยเพิ่มจำนวนตัวอย่างในคลาสที่มีจำนวนน้อย ทำให้โมเดลสามารถเรียนรู้ลักษณะของคลาสนั้น ๆ ได้ดีขึ้นและลดปัญหาความไม่สมดุลของข้อมูล

$$D_{train} = D_{train} \cup (S_i^*, D_i^*) \tag{3.9}$$

3.4 กรอบงานการพัฒนาและประเมินโมเดล

กรอบงานพัฒนาโมเดลเป็นสองส่วนคือ การใช้โมเดลเรียนรู้ของเครื่องแบบดั้งเดิม รวมทั้งการประยุกต์ใช้โมเดลแบบ stacking ensemble และโมเดลแบบทรานฟอร์เมอร์ด้วย bert-base

uncased ซึ่งทำการเปรียบเทียบกับชุดข้อมูลก่อนการเสริม เปรียบเทียบกับการเสริมข้อมูลด้วยวิธีต่าง ๆ ด้วย ภาพรวมของโมเดลแสดงดังรูป 3.6



รูปที่ 3.6 กรอบงานการสร้างโมเดลจำแนกระดับความรุนแรง

จากรูปที่ 3.6 แสดงกรอบการทำงานที่ครอบคลุมสำหรับการพัฒนาโมเดลจำแนกระดับความรุนแรงของจุดบกพร่องในรูปแบบหลายคลาสของงานวิจัยนี้ การทดสอบกับโมเดลที่หลากหลายเพื่อศึกษาและวิเคราะห์ความเหมาะสมในการเลือกใช้โมเดลกับปัญหาในโดเมนนี้ รวมทั้งเพื่อเปรียบเทียบกับโมเดลที่หลากหลายสำหรับการทดลองกับชุดข้อมูลที่ถูกเสริมขึ้นจะสามารถให้ผลลัพธ์ที่ดีกับโมเดลที่แตกต่างกันหรือไม่ ซึ่งเป็นการรวมเอาเทคนิคที่หลากหลายจากทั้งโมเดลการเรียนรู้ของเครื่องแบบดั้งเดิม รวมทั้งการใช้การเรียนรู้แบบเชิงซ้อน (Ensemble Stacking) และโมเดลการเรียนรู้เชิงลึกสมัยใหม่อย่าง BERT โดยกระบวนการในกรอบงานนี้ออกแบบมาอย่างเป็นระบบ เริ่มต้นตั้งแต่การจัดการข้อมูลรายงานจุดบกพร่อง การแบ่งชุดข้อมูลสำหรับการฝึกฝนและการทดสอบโมเดล การ

เตรียมข้อมูลและการเลือกคุณลักษณะที่สำคัญ รวมถึงการปรับแต่งโมเดลให้เหมาะสมกับลักษณะข้อมูลที่ใช้ งานวิจัยนี้ยังได้นำเสนอการเปรียบเทียบผลลัพธ์ระหว่างการใช้โมเดลการเรียนรู้แบบดั้งเดิม เช่น Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF) และ Ensemble Stacking กับโมเดล BERT เพื่อประเมินประสิทธิภาพในแง่มุมต่าง ๆ เช่น ความถูกต้อง (Accuracy) และค่า F1 Score โดยรายละเอียดของแต่ละขั้นตอนในกรอบงานนี้จะถูกอธิบายเพิ่มเติมในลำดับถัดไป

1. ชุดข้อมูล Bug Reports

ข้อมูลรายงานจุดบกพร่องเป็นข้อมูลพื้นฐานที่ใช้ในกระบวนการพัฒนาโมเดลเพื่อจำแนกระดับความรุนแรงของจุดบกพร่อง โดยข้อมูลนี้ประกอบไปด้วยรายละเอียดของจุดบกพร่องที่ถูกรายงาน ได้แก่ ชื่อเรื่อง หรือ summary และคำอธิบายปัญหา หรือ description ซึ่งถูกนำมาผ่านกระบวนการคัดเลือกและจัดการให้เหมาะสมต่อการใช้งาน โดยข้อมูลทั้งหมดถูกแบ่งออกเป็นสองชุด ได้แก่ ข้อมูลชุดฝึก (Train Dataset) และ ข้อมูลชุดทดสอบ (Test Dataset) โดยชุดฝึกถูกใช้สำหรับการฝึกโมเดล ซึ่งเป็นขั้นตอนสำคัญที่ช่วยให้โมเดลสามารถเรียนรู้รูปแบบและความสัมพันธ์ในข้อมูลได้อย่างมีประสิทธิภาพ ส่วนข้อมูลชุดทดสอบถูกใช้สำหรับการทดสอบโมเดลที่ได้รับการฝึกมาแล้ว เพื่อประเมินความสามารถของโมเดลในการจำแนกข้อมูลใหม่ที่ไม่เคยเห็นมาก่อน การแบ่งชุดข้อมูลในลักษณะนี้ช่วยให้กระบวนการทดสอบสามารถสะท้อนถึงประสิทธิภาพที่แท้จริงของโมเดลได้อย่างแม่นยำ นอกจากนี้การเสริมข้อมูลชุดฝึกหรือการปรับปรุงข้อมูลเพิ่มเติม จะช่วยเสริมความหลากหลายของข้อมูลสำหรับการฝึก เพื่อให้โมเดลสามารถรองรับข้อมูลที่มีความแตกต่างและซับซ้อนได้ดีขึ้น ซึ่งงานวิจัยนี้ดำเนินการแบ่งชุดข้อมูลชุดฝึก 80% และชุดทดสอบอีก 20% ทั้งนี้การแบ่งชุดข้อมูลจะดำเนินการก่อนการนำชุดฝึกไปเสริมข้อมูล และใช้ชุดเดียวกับในการประเมินผลเปรียบเทียบ

2. Train Dataset

ชุดข้อมูลสำหรับการฝึกเป็นส่วนสำคัญในกระบวนการสร้างและพัฒนาโมเดลจำแนกระดับความรุนแรงของจุดบกพร่องโดยข้อมูลนี้ถูกออกแบบมาเพื่อใช้ในการฝึกโมเดลให้สามารถเรียนรู้รูปแบบ ความสัมพันธ์ และลักษณะสำคัญที่เชื่อมโยงกับระดับความรุนแรงของจุดบกพร่อง ชุดข้อมูลนี้ถูกแบ่งออกเป็นสองกลุ่มหลัก ได้แก่

ข้อมูลดั้งเดิม (Original Dataset) เป็นข้อมูลที่นำมาจากแหล่งข้อมูลต้นฉบับโดยตรง ซึ่งยังคงมีโครงสร้างและรายละเอียดที่เหมือนกับข้อมูลจริง การใช้ข้อมูลดั้งเดิมช่วยให้โมเดลสามารถเรียนรู้จากข้อมูลที่มีความน่าเชื่อถือและสะท้อนถึงลักษณะของปัญหาในโลกความเป็นจริง

ข้อมูลเสริม (Augmented Dataset) เป็นข้อมูลที่ถูกปรับปรุงหรือขยายให้มีความหลากหลายมากขึ้นผ่านกระบวนการที่เรียกว่าการเสริมข้อมูลจากขั้นตอนก่อนหน้า เพื่อเพิ่มความหลากหลายของข้อมูลที่โมเดลจะได้เรียนรู้ การทำเช่นนี้ช่วยลดปัญหาการกระจายตัวของข้อมูลที่ไม่สมดุล และเพิ่มความสามารถของโมเดลในการรับมือกับข้อมูลในสถานการณ์ต่าง ๆ ที่อาจมีความหลากหลายมากขึ้น

การฝึกโมเดลในแต่ละชุดข้อมูลดำเนินการแยกขั้นตอนกันในแต่ละชุดข้อมูล การนำข้อมูลเสริมมาฝึกโมเดลนอกจากเพิ่มความสามารถของโมเดลแล้ว ยังช่วยเพิ่มประสิทธิภาพในการทำนายฝนคลาสที่ไม่สมดุล

3. โมเดลการเรียนรู้ของเครื่อง (Machine Learning Models)
    - กระบวนการในส่วนนี้ประกอบด้วย
        - การเตรียมข้อมูล (Preprocessing) เป็นขั้นตอนการทำความสะอาดและแปลงข้อมูลให้เหมาะสม
        - การแปลงข้อความและเลือกคุณลักษณะ (Text Representation and Feature Selection) โดยใช้เทคนิค TF-IDF ในการแปลงข้อความให้อยู่ในรูปแบบตัวเลข และใช้ Chi-square (Chi2) ในการทดสอบการเลือกคุณลักษณะ
        - ใช้โมเดลการเรียนรู้ของเครื่องแบบดั้งเดิม ได้แก่ Logistic Regression (LR), Support Vector Machine (SVM), และ Random Forest (RF) ในการประเมินเปรียบเทียบในส่วนของการสร้างโมเดลการจำแนกระดับความรุนแรง
    - ผลลัพธ์คือ Multiclass Severity Classifier Models จากโมเดลเรียนรู้ของเครื่องแบบดั้งเดิม
4. โมเดล BERT
    - กระบวนการใช้โมเดล BERT แบ่งเป็น:
        - การแปลงข้อความเป็นโทเค็นด้วย WordPiece Tokenizer
        - สร้างเวกเตอร์ฝังตัว (Embeddings) โดยรวมข้อมูลโทเค็น (Token), เซ็กเมนต์ (Segment), และตำแหน่ง (Position)

- ทำการ Fine-tuning เพื่อปรับปรุงโมเดลให้เหมาะสมกับข้อมูล

- ผลลัพธ์คือ Multiclass Severity Classifier Models จากโมเดลแบบทรานฟอร์เมอร์ด้วย BERT โมเดล

5. Test Dataset

ข้อมูลนี้ใช้สำหรับการทดสอบโมเดลหลังจากการฝึก โดยผ่านการเตรียมข้อมูลและการแปลงข้อความแบบเดียวกับส่วนการฝึกของแต่ละโมเดล

6. การประเมินและเปรียบเทียบผลลัพธ์ (Model Evaluation and Comparison)

เปรียบเทียบประสิทธิภาพของโมเดลโดยใช้ตัวชี้วัด ได้แก่

- ความถูกต้อง (Accuracy)
- ค่า F1 Score
- ความแม่นยำ (Precision)
- การเรียกคืน (Recall)
- ค่าสัมประสิทธิ์แมทธิวส์ (MCC: Matthews Correlation Coefficient)

3.5 การเตรียมข้อมูลและวิเคราะห์คุณลักษณะสำหรับโมเดลการเรียนรู้ของเครื่อง

การเตรียมข้อมูลและการวิเคราะห์คุณลักษณะเป็นขั้นตอนสำคัญที่ส่งผลต่อประสิทธิภาพของโมเดลการเรียนรู้ของเครื่อง (Machine Learning) ในการจำแนกระดับความรุนแรงของจุดบกพร่อง ขั้นตอนนี้ประกอบด้วยการแปลงข้อมูลที่ไม่เป็นโครงสร้าง (Unstructured Data) จากรายงานจุดบกพร่องให้อยู่ในรูปแบบที่เหมาะสมต่อการเรียนรู้ และการคัดเลือกคุณลักษณะที่มีความสัมพันธ์กับเป้าหมายของการจำแนก เพื่อช่วยเพิ่มความแม่นยำและลดความซับซ้อนของโมเดล โดยกระบวนการในส่วนนี้แบ่งออกเป็นขั้นตอนสำคัญดังนี้

1. การทำความสะอาดข้อมูล (Data Cleaning)

ขั้นตอนการทำความสะอาดข้อมูลจากรายงานจุดบกพร่องประกอบด้วยการลบลักษณะข้อมูลที่ไม่สำคัญ ได้แก่ อักขระพิเศษ ตัวเลข เป็นต้น รวมทั้งการลบข้อมูลซ้ำ รูปที่ 3.8 แสดงตัวอย่างโค้ดการทำงานในส่วนของการทำความสะอาดข้อมูล

```
# Clean documents_train and documents_test
documents_train = documents_train.astype(str).str.lower()
documents_test = documents_test.astype(str).str.lower()

# Removing punctuation
documents_train = documents_train.str.replace('[^\w\s]', '', regex=True)
documents_test = documents_test.str.replace('[^\w\s]', '', regex=True)

# Remove extra whitespaces
documents_train = documents_train.str.strip()
documents_test = documents_test.str.strip()
```

รูปที่ 3.7 โค้ดการทำงานการทำความสะอาดข้อมูล

ในการทำความสะอาดข้อมูลในงานวิจัยนี้ ได้มีการดำเนินการตามขั้นตอนที่กำหนดไว้ โดยผลลัพธ์ที่ได้จากแต่ละขั้นตอนสามารถสังเกตได้จากตารางที่นำเสนอ อย่างไรก็ตาม งานวิจัยนี้ไม่ได้เน้นความเข้มงวดในการทำความสะอาดข้อมูลมากนัก เนื่องจากข้อมูลต้นทางที่นำมาใช้อาจมีลักษณะเฉพาะที่สำคัญต่อการจำแนกระดับความรุนแรงของจุดบกพร่อง ตัวอย่างเช่น ตัวเลขในข้อมูลบางรายการอาจบ่งบอกถึงหมายเลขเวอร์ชันของซอฟต์แวร์ ซึ่งเป็นข้อมูลสำคัญในการพิจารณาว่าจุดบกพร่องนั้นเกี่ยวข้องกับเวอร์ชันใด หรืออาจช่วยให้สามารถตรวจสอบแนวโน้มของปัญหาที่เกิดขึ้นกับแต่ละเวอร์ชันได้ นอกจากนี้ การคงไว้ซึ่งองค์ประกอบของโค้ด เช่น syntax หรือรูปแบบการเขียนโปรแกรม อาจช่วยให้ระบบสามารถวิเคราะห์และระบุข้อผิดพลาดได้อย่างถูกต้องยิ่งขึ้น นอกจากนี้ยังรวมถึง URL หรือที่อยู่เว็บไซต์ที่อาจเป็นข้อมูลอ้างอิงสำคัญในรายงานจุดบกพร่อง ซึ่งสามารถใช้ตรวจสอบแหล่งที่มาของข้อผิดพลาด หรือเชื่อมโยงกับข้อมูลที่เกี่ยวข้องอื่น ๆ ได้ ดังนั้น ในการทำความสะอาดข้อมูลของงานวิจัยนี้จึงพิจารณาเฉพาะข้อมูลที่ไม่จำเป็น เช่น อักขระพิเศษ ข้อความที่ซ้ำซ้อน หรือข้อมูลที่อาจทำให้เกิดความคลาดเคลื่อนในกระบวนการจำแนก แต่ยังคงรักษาข้อมูลสำคัญที่อาจมีผลกระทบต่อคุณลักษณะของชุดข้อมูลไว้ เพื่อให้โมเดลสามารถเรียนรู้ได้อย่างมีประสิทธิภาพและสะท้อนถึงลักษณะของข้อมูลจริงให้ได้มากที่สุด ทั้งนี้ สามารถดูตัวอย่างผลลัพธ์ที่ได้จากแต่ละขั้นตอนของกระบวนการทำความสะอาดข้อมูลในตารางที่แสดงด้านล่าง

ตารางที่ 3.4 การเพิ่มข้อมูลชุดฝึกด้วย SMOTE

ขั้นตอน	**Document bug report**
Original	Bon Echo locks up, consuming 100%, quite often
Lowercase	bon echo locks up, consuming 100%, quite often
No Punctuation	bon echo locks up consuming 100 quite often
No Extra Spaces	bon echo locks up consuming 100 quite often

2. การแปลงข้อความ (Text Transformation)

ข้อมูลที่อยู่ในรายงานจุดบกพร่อง (Bug Reports) มักอยู่ในรูปแบบของข้อความที่ไม่มีโครงสร้างชัดเจน (Unstructured Text) เช่น รายละเอียดข้อผิดพลาดที่เขียนเป็นข้อความธรรมดา ซึ่งอาจมีตัวอักษรหลากหลายรูปแบบ การใช้ตัวพิมพ์ใหญ่-พิมพ์เล็กผสมกัน คำที่มีหลายรูปแบบ หรือแม้แต่การมีอักขระพิเศษและเครื่องหมายต่าง ๆ ที่อาจรบกวนกระบวนการวิเคราะห์ข้อมูล ดังนั้นจำเป็นต้องมีการแปลงข้อความให้อยู่ในรูปแบบที่สามารถนำไปประมวลผลได้อย่างมีประสิทธิภาพ โดยกระบวนการนี้มีองค์ประกอบหลักหลายประการ ได้แก่

1) การแปลงตัวอักษรเป็นตัวพิมพ์เล็กทั้งหมด (Lowercasing)

การแปลงข้อความให้เป็นตัวพิมพ์เล็กทั้งหมดเป็นขั้นตอนพื้นฐานที่สำคัญในกระบวนการประมวลผลภาษาธรรมชาติ เนื่องจากภาษาที่ใช้ในรายงานจุดบกพร่องอาจมีการใช้ตัวพิมพ์ใหญ่และตัวพิมพ์เล็กในลักษณะที่ไม่สม่ำเสมอ การแปลงข้อความให้เป็นตัวพิมพ์เล็กช่วยลดความซ้ำซ้อนของคำ เช่น คำว่า Error, ERROR, และ error ซึ่งมีความหมายเดียวกัน แต่ถ้าไม่มีการแปลง ระบบอาจมองว่าเป็นคำที่แตกต่างกัน การทำให้ข้อความเป็นตัวพิมพ์เล็กจึงช่วยให้การวิเคราะห์ข้อมูลมีความแม่นยำมากขึ้น

2) การแยกคำ (Tokenization)

การแยกคำเป็นกระบวนการแยกข้อความออกเป็นหน่วยที่เล็กลง เช่น คำ (Words), วลี (Phrases), หรือแม้แต่ประโยค (Sentences) เพื่อให้สามารถนำไปวิเคราะห์และใช้เป็นฟีเจอร์สำหรับโมเดลได้ โดยทั่วไปการ Tokenization จะทำโดยการแบ่งคำตามช่องว่างหรืออักขระพิเศษ แต่ในบางกรณี เช่น ภาษาไทย หรือภาษาที่ไม่มีการเว้นวรรคระหว่างคำ อาจต้องใช้เครื่องมือเฉพาะทาง เช่น PyThaiNLP หรือ NLTK

ตัวอย่างการใช้ Tokenization กับภาษาอังกฤษโดยใช้ NLTK:

text = "The server encountered an unexpected error and failed to respond."

ผลลัพธ์ที่ได้: ['The', 'server', 'encountered', 'an', 'unexpected', 'error', 'and', 'failed', 'to', 'respond']

3) การลบคำที่ไม่สำคัญ (Stopword Removal)

Stopwords คือคำที่ไม่มีความหมายเชิงวิเคราะห์ในการจำแนกข้อมูล เช่น "the", "is", "and", "or", "a", "an", ซึ่งมักพบบ่อยในข้อความ การลบคำเหล่านี้ออกสามารถช่วยลดขนาดของข้อมูลและเพิ่มประสิทธิภาพของโมเดล

4) การแปลงคำให้อยู่ในรูปแบบพื้นฐาน (Lemmatization หรือ Stemming)

การแปลงคำให้อยู่ในรูปแบบพื้นฐานเป็นกระบวนการที่ช่วยลดรูปของคำให้เป็นรากศัพท์เดิม เช่น เปลี่ยน "running" เป็น "run", "better" เป็น "good"

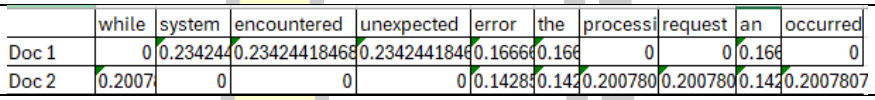
ทั้งหมดเป็นขั้นตอนพื้นฐานในการเตรียมข้อมูลก่อนนำเข้าขั้นตอนต่อไป ซึ่งกระบวนการทั้งหมดถูกนำมาประเมินเบื้องต้องเพื่อทดสอบประสิทธิภาพการทำนายของโมเดล ในงานวิจัยนี้สุดท้ายแล้วบางกระบวนการไม่ได้ถูกใช้ เช่น สำหรับการโมเดลการเรียนรู้ของเครื่องไม่ใช้กระบวนการ stopwords เป็นต้น

3. การแปลงข้อมูลเป็นตัวเลข (Numerical Representation)

เนื่องจากอัลกอริธึมการเรียนรู้ของเครื่องไม่สามารถประมวลผลข้อมูลที่เป็นข้อความโดยตรง ข้อมูลที่เป็นข้อความต้องถูกแปลงให้อยู่ในรูปของตัวเลข (Numerical Representation) ก่อนที่จะสามารถนำไปใช้ในการฝึกโมเดลได้ กระบวนการนี้ช่วยให้โมเดลสามารถวิเคราะห์ความสัมพันธ์ของคำและความหมายในข้อมูลได้อย่างมีประสิทธิภาพ ซึ่งมีหลายเทคนิคที่สามารถใช้ในการแปลงข้อมูลข้อความเป็นตัวเลข โดยเทคนิคที่ใช้ในงานวิจัยนี้คือ TF-IDF (Term Frequency-Inverse Document Frequency) ซึ่งเป็นวิธีการถ่วงน้ำหนักคำเพื่อสะท้อนความสำคัญของคำในเอกสาร

TF-IDF วัดความสำคัญของแต่ละคำภายในเอกสารหนึ่ง ๆ โดยพิจารณาจากความถี่ของคำในเอกสารนั้น และความถี่ของคำเดียวกันในชุดข้อมูลทั้งหมด แนวคิดของ TF-IDF คือการให้คะแนนสูงกับคำที่พบได้บ่อยในเอกสารหนึ่ง ๆ แต่ไม่ค่อยพบในเอกสารอื่น ซึ่งหมายความว่าคำดังกล่าวมีความสำคัญเฉพาะตัวและอาจช่วยในการจำแนกข้อมูลได้ดีขึ้น โดย TF-IDF คำนวณได้จากสององค์ประกอบหลัก คือ TF (Term Frequency) คือคำที่ใช้วัดว่าคำหนึ่ง ๆ ปรากฏอยู่ในเอกสารบ่อยเพียงใด คำนวณได้จากสมการ 2.1 และ IDF (Inverse Document Frequency) ใช้เพื่อลดความสำคัญของคำที่ปรากฏบ่อยเกินไปในทุกเอกสาร เช่น คำว่า "the", "is", "and" ซึ่งมักไม่มีความหมายในการวิเคราะห์ คำนวณได้จากสมการ 2.2 และค่าคะแนนของ TF-IDF คำนวณจากสมการ 2.3 แสดงการคำนวณ TF-IDF ได้ดังตาราง

ตารางที่ 3.5 ตัวอย่างการคำนวณหา TF-IDF

ขั้นตอน	**ตัวอย่าง**
Document	documents = ["The system encountered an unexpected error", "An error occurred while processing the request]
Tokenized	Doc1 = ['the', 'system', 'encountered', 'an', 'unexpected', 'error'] Doc2 = ['an', 'error', 'occurred', ' while', ' processing', 'the', ' request']
TF (“error”)	$TF_{(error,Doc1)} = \frac{1}{6} = 0.1667$ $TF_{(error,Doc2)} = \frac{1}{7} = 0.1429$
IDF (“error”)	$IDF_{(error)} = log\left(\frac{2+1}{2+1}\right) + 1 = 1$
TF-IDF (“error”)	$TF - IDF_{(eror,Doc1)} = 0.1667 \times 1 = 0.1667$ $TF - IDF_{(eror,Doc1)} = 0.1429 \times 1 = 0.1429$
TF-IDF	

	while	system	encountered	unexpected	error	the	processi	request	an	occurred
Doc 1	0	0.234244	0.234244184668	0.234244184668	0.166666	0.166	0	0	0.166	0
Doc 2	0.2007	0	0	0	0.14285	0.142	0.200780	0.200780	0.142	0.2007807

4. การเลือกคุณลักษณะ (Feature Selection)

การเลือกคุณลักษณะ (Feature Selection) เป็นขั้นตอนสำคัญในการเตรียมข้อมูลสำหรับโมเดลการเรียนรู้ของเครื่อง โดยมีเป้าหมายหลักเพื่อลดขนาดของข้อมูล ปรับปรุงประสิทธิภาพของโมเดล และลดความซับซ้อนของการคำนวณโดยไม่กระทบต่อความแม่นยำของการจำแนก เนื่องจากชุดข้อมูลที่ได้จากการแปลงข้อความเป็นตัวเลข เช่น TF-IDF มักจะมีมิติสูง (High-Dimensional Data) ซึ่งอาจส่งผลให้โมเดลเกิดปัญหา Curse of Dimensionality หรือภาวะที่ข้อมูลมีมิติมากเกินไป ส่งผลให้การเรียนรู้ของโมเดลมีประสิทธิภาพลดลง ดังนั้น จำเป็นต้องเลือกเฉพาะคุณลักษณะที่มีความสำคัญต่อการจำแนกระดับความรุนแรงของจุดบกพร่องให้มากที่สุด

ในงานวิจัยนี้ได้ใช้ Information Gain (IG) เป็นเทคนิคหลักในการเลือกคุณลักษณะ Information Gain เป็นการวัดว่าคุณลักษณะ (Feature) หนึ่ง ๆ ช่วยลดความไม่แน่นอน (Entropy) ในการจำแนกข้อมูลได้มากเพียงใด ค่าของ IG ที่สูงบ่งชี้ว่าคุณลักษณะนั้นมีประโยชน์ในการแยกประเภทข้อมูล ซึ่งสามารถคำนวณได้จากความแตกต่างของค่า Entropy ก่อนและหลังการเลือกคุณลักษณะ อย่างไรก็ตามการนำ IG มาใช้เป็นส่วนประกอบเพื่อประเมินคุณลักษณะสำหรับโมเดลการเรียนรู้ของเครื่องเท่านั้น

3.6 การจำแนกระดับความรุนแรงด้วยโมเดลการเรียนรู้ของเครื่อง

ในงานวิจัยนี้ ได้มีการใช้โมเดลการเรียนรู้ของเครื่อง (Machine Learning) แบบดั้งเดิมเพื่อจำแนกระดับความรุนแรงของจุดบกพร่อง (Bug Severity Classification) โดยมุ่งเน้นการประเมินและเปรียบเทียบประสิทธิภาพของโมเดลต่าง ๆ ในการจำแนกข้อมูลหลายคลาส (Multiclass Classification) โมเดลที่เลือกใช้ในงานวิจัยนี้ประกอบด้วย Support Vector Machine (SVM), Logistic Regression (LR), และ Random Forest (RF) ซึ่งแต่ละโมเดลมีลักษณะและกระบวนการทำงานเฉพาะตัวดังนี้

1. Logistic Regression (LR)

Logistic Regression (LR) เป็นโมเดลที่มีโครงสร้างเรียบง่ายและได้รับความนิยมอย่างแพร่หลายในการจำแนกข้อมูลหลายคลาส โดยเฉพาะในกรณีของ Multinomial Logistic Regression ซึ่งสามารถคำนวณค่าความน่าจะเป็นของแต่ละคลาสได้อย่างมีประสิทธิภาพ หลักการทำงานของ Logistic Regression อาศัยฟังก์ชัน Sigmoid เพื่อแปลงผลลัพธ์ให้อยู่ในช่วงค่าความน่าจะเป็นระหว่าง 0 และ 1 ทำให้สามารถใช้กำหนดเกณฑ์การจำแนกข้อมูลได้อย่างชัดเจน ในงานวิจัยนี้ Logistic Regression ถูกนำมาใช้เป็นโมเดลพื้นฐานสำหรับการจำแนกระดับความรุนแรงของจุดบกพร่อง พร้อมทั้งมีการปรับค่าพารามิเตอร์ Regularization เช่น L1 (Lasso) และ L2 (Ridge) เพื่อลดปัญหา Overfitting และช่วยเพิ่มความแม่นยำของโมเดลในการเรียนรู้ข้อมูลที่มีลักษณะเฉพาะ Logistic Regression มีจุดเด่นในการจัดการกับข้อมูลที่มีความสัมพันธ์เชิงเส้น (Linear Relationship) ระหว่างตัวแปรอิสระและผลลัพธ์ ทำให้เหมาะสมกับปัญหาที่ข้อมูลสามารถแยกกันได้ด้วยเส้นตรง

2. Support Vector Machine (SVM)

Support Vector Machine (SVM) เป็นโมเดลที่มีประสิทธิภาพสูงในการจำแนกข้อมูล โดยเฉพาะในปัญหาที่มีข้อมูลซับซ้อนหรือมีความไม่สมดุล หลักการสำคัญของ SVM คือการค้นหาขอบเขตการจำแนกที่เหมาะสมที่สุด (Optimal Hyperplane) ซึ่งใช้ในการแบ่งข้อมูลออกเป็นกลุ่มอย่างมีประสิทธิภาพ โดยโมเดลจะพยายามเพิ่มระยะห่างระหว่างกลุ่มข้อมูลที่แตกต่างกันเพื่อให้การจำแนกมีความแม่นยำสูงสุด ในงานวิจัยนี้ SVM ถูกนำมาใช้สำหรับการจำแนกระดับความรุนแรงของจุดบกพร่อง โดยได้ปรับแต่ง Kernel Function ให้เป็นแบบ RBF Kernel (Radial Basis Function) ซึ่งช่วยเพิ่มความสามารถของโมเดลในการจำแนกข้อมูลที่มีโครงสร้างซับซ้อนและไม่เป็นเชิงเส้น (non-linear classification) จุดเด่นของ SVM คือความสามารถในการทำงานกับชุดข้อมูลที่

มีขนาดเล็กได้อย่างมีประสิทธิภาพ และความสามารถในการลดผลกระทบจากค่าผิดปกติ ซึ่งช่วยให้การจำแนกข้อมูลมีความแม่นยำและเสถียรยิ่งขึ้น

3. Random Forest (RF)

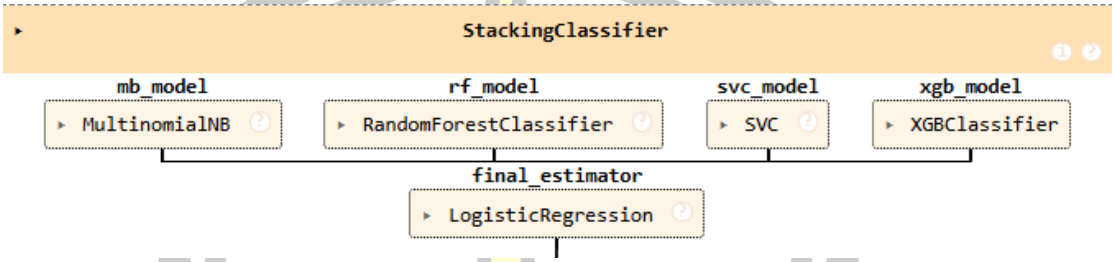
Random Forest (RF) เป็นโมเดลที่อยู่ในกลุ่ม Ensemble Learning ซึ่งทำงานโดยการรวมผลลัพธ์จากหลาย ๆ ต้นไม้การตัดสินใจ (Decision Trees) เพื่อเพิ่มความแม่นยำในการจำแนกข้อมูล โมเดลนี้ใช้เทคนิค Bootstrap Sampling ในการสุ่มตัวอย่างข้อมูลและเลือกคุณลักษณะ (features) เพื่อสร้างต้นไม้หลายต้น ทำให้แต่ละต้นไม้มีรูปแบบการเรียนรู้ที่แตกต่างกัน ส่งผลให้โมเดลสามารถลดอคติและเพิ่มความสามารถในการจำแนกข้อมูลได้อย่างมีประสิทธิภาพ ผลลัพธ์สุดท้ายของ Random Forest จะถูกกำหนดโดยวิธี Majority Voting ซึ่งเป็นการรวมผลการพยากรณ์จากต้นไม้ทั้งหมดและเลือกค่าที่ได้รับการโหวตมากที่สุด ซึ่งช่วยให้โมเดลมีเสถียรภาพและลดความอ่อนไหวต่อข้อมูลที่มีการกระจายตัวสูง จุดเด่นสำคัญของโมเดลนี้คือความสามารถในการจัดการกับข้อมูลที่มีความไม่สมดุล รวมถึงความสามารถในการต้านทานปัญหา Overfitting เนื่องจากการใช้ต้นไม้หลายต้นช่วยลดการพึ่งพาข้อมูลตัวอย่างใดตัวอย่างหนึ่งมากเกินไป ในงานวิจัยนี้ Random Forest ถูกนำมาปรับแต่งโดยการกำหนดจำนวนต้นไม้ (Number of Trees) และความลึกสูงสุด (Max Depth) ให้เหมาะสมกับข้อมูลที่ใช้ เพื่อให้สามารถเพิ่มประสิทธิภาพของการจำแนกและลดข้อผิดพลาดในการทำนายระดับความรุนแรงของจุดบกพร่องได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

4. Ensemble stacking

Ensemble แบบ Stacking เป็นเทคนิคที่ถูกนำมาใช้ในการวิเคราะห์และประเมินเพิ่มเติมในงานวิจัยนี้ เพื่อตรวจสอบว่าการใช้โมเดลเชิงซ้อน (ensemble stacking) สามารถช่วยเพิ่มประสิทธิภาพของการจำแนกระดับความรุนแรงของจุดบกพร่องได้หรือไม่ เทคนิคนี้ช่วยให้โมเดลสามารถรวมความสามารถของโมเดลย่อยหลายตัว (base models) เพื่อให้ได้ผลลัพธ์ที่มีความแม่นยำและเสถียรมากขึ้น โดยในงานวิจัยนี้ใช้โมเดลพื้นฐาน (base models) ได้แก่ Multinomial Naive Bayes (MNB), Random Forest (RF), Support Vector Machine (SVM), และ XGBClassifier ซึ่งแต่ละโมเดลมีจุดเด่นที่แตกต่างกันในการจัดการกับข้อมูลข้อความและการเรียนรู้รูปแบบของข้อมูล นอกจากนี้ การรวมผลลัพธ์จากโมเดลพื้นฐานยังช่วยลดปัญหาการพึ่งพาโมเดลเดียวที่อาจมีข้อจำกัดบางประการ

ในกระบวนการ stacking, โมเดลพื้นฐานจะถูกฝึกแยกกันบนชุดข้อมูลเดียวกันและสร้างผลลัพธ์ออกมาเป็นค่าพยากรณ์ (predictions) จากนั้นค่าพยากรณ์ที่ได้จะถูกนำมาเป็นอินพุตสำหรับโมเดลระดับที่สอง (meta model) ซึ่งในกรณีนี้ใช้ Logistic Regression เพื่อเรียนรู้จากผลลัพธ์ของโมเดลพื้นฐานและสร้างผลลัพธ์สุดท้าย เทคนิคนี้ช่วยให้การจำแนกสามารถอาศัยความ

แข็งแกร่งของโมเดลหลายตัว โดยลดข้อจำกัดของโมเดลเดี่ยวที่อาจมีอคติหรือความผิดพลาดในบางรูปแบบของข้อมูล ทั้งนี้ ensemble stacking มักให้ผลลัพธ์ที่ดีกว่าโมเดลเดี่ยว เนื่องจากอาศัยข้อได้เปรียบของโมเดลแต่ละตัวมาทำงานร่วมกัน ซึ่งกระบวนการนี้สามารถอธิบายได้ผ่านแผนภาพดังรูปที่แสดงกระบวนการทำงานของ ensemble stacking classifier ที่ใช้ในงานวิจัยนี้



รูปที่ 3.8 โครงสร้าง Ensemble stacking

3.7 การจำแนกระดับความรุนแรงด้วยโมเดลทรานฟอร์เมอร์

โมเดลทรานฟอร์เมอร์ (Transformer) เป็นเทคโนโลยีที่ทันสมัยและทรงพลังในด้านการเรียนรู้เชิงลึก (Deep Learning) โดยเฉพาะในงานที่เกี่ยวข้องกับการประมวลผลภาษาธรรมชาติ (Natural Language Processing - NLP) โครงสร้างของโมเดลนี้มีความสามารถในการเรียนรู้และจับความสัมพันธ์ที่ซับซ้อนในข้อมูลข้อความได้อย่างมีประสิทธิภาพ งานวิจัยนี้ได้เลือกใช้โมเดล BERT (Bidirectional Encoder Representations from Transformers) รุ่น bert-base-uncased ซึ่งเป็นหนึ่งในโมเดลทรานฟอร์เมอร์ที่ได้รับความนิยมมากที่สุดในการเปรียบเทียบประสิทธิภาพกับโมเดลการเรียนรู้ของเครื่องแบบดั้งเดิม BERT ได้รับการพัฒนาโดย Google และได้รับการยอมรับในวงการ NLP ว่าเป็นหนึ่งในโมเดลที่มีประสิทธิภาพสูง เนื่องจาก ความสามารถในการเรียนรู้สองทิศทาง (Bidirectional Learning) โดย BERT สามารถเรียนรู้บริบทของคำทั้งจากซ้ายไปขวาและขวาไปซ้าย ทำให้เข้าใจความหมายของคำในบริบทที่กว้างขึ้น และการใช้งานในหลากหลายภาษา โดยเฉพาะรุ่น bert-base-uncased เป็นเวอร์ชันที่ไม่คำนึงถึงตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็ก ทำให้เหมาะสำหรับการประมวลผลภาษาที่ไม่มีการแยกตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่

การใช้โมเดล BERT สำหรับจำแนกระดับความรุนแรงในงานวิจัยนี้มีขั้นตอนดังนี้

1. การเตรียมข้อมูลสำหรับ BERT

ข้อความในรายงานจุดบกพร่อง (Bug Reports) ถูกแปลงเป็นโทเค็น (Token) โดยใช้ WordPiece Tokenizer ซึ่งช่วยจัดการคำที่ไม่อยู่ในคำศัพท์ของโมเดล สำหรับแต่ละข้อความ ระบบจะสร้างเวกเตอร์ที่มีข้อมูลสามส่วน ได้แก่

- Token Embeddings: ตัวแทนของคำในรูปแบบตัวเลข
- Segment Embeddings: ตัวระบุส่วนของข้อความ เช่น ข้อความหลักหรือข้อความเพิ่มเติม
- Position Embeddings: ข้อมูลเกี่ยวกับตำแหน่งของคำในประโยค

2. การ Fine-tuning โมเดล BERT

โมเดล BERT ที่ผ่านการ Pre-training บนชุดข้อมูลขนาดใหญ่ถูกนำมาปรับแต่ง (Fine-tuning) บนชุดข้อมูลรายงานจุดบกพร่อง และใช้ Optimizer AdamW และปรับค่าพารามิเตอร์ เช่น Learning Rate และ Batch Size เพื่อเพิ่มความแม่นยำของโมเดล โดยการฝึกโมเดลถูกดำเนินการในหลาย Epoch เพื่อให้ระบบสามารถเรียนรู้ลักษณะเฉพาะของข้อมูลได้อย่างละเอียด

3. การประเมินผลการจำแนก

โมเดล BERT ที่ได้รับการ Fine-tuning ถูกนำมาทดสอบบนชุดข้อมูลที่ไม่เคยเห็นมาก่อน (Test Dataset) ประสิทธิภาพของโมเดลถูกประเมินด้วยตัวชี้วัด เช่น Accuracy, F1 Score, Precision, Recall และ Matthews Correlation Coefficient (MCC)

สำหรับการวิจัยนี้ ได้ทำการเปรียบเทียบการตั้งค่าโมเดล BERT ในสองรูปแบบ คือ BERT1 และ BERT2 เพื่อศึกษาผลกระทบของการปรับแต่งค่าพารามิเตอร์ที่แตกต่างกันต่อประสิทธิภาพของการจำแนกระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์ โดยแต่ละโมเดลมีลักษณะการตั้งค่าที่แตกต่างกัน ดังนี้:

- BERT1 เป็นโมเดลที่ตั้งค่าพื้นฐานโดยใช้ค่าพารามิเตอร์มาตรฐานของโมเดล BERT โดยเน้นไปที่การใช้โครงสร้างโมเดลเดิม และไม่ได้มีการปรับแต่งค่าที่เกี่ยวข้องกับอัตราการเรียนรู้ (Learning Rate), การถ่วงน้ำหนัก (Weight Decay), หรือกลยุทธ์การกำหนดตารางเวลาเรียนรู้ (Learning Rate Scheduler) เป็นต้น ซึ่งโมเดลนี้ใช้เพื่อเป็นตัวเปรียบเทียบพื้นฐาน
- BERT2 เป็นโมเดลที่ได้รับการปรับแต่งพารามิเตอร์สำคัญหลายตัว เพื่อเพิ่มประสิทธิภาพของการเรียนรู้และการจำแนกข้อมูลได้อย่างแม่นยำมากขึ้น การปรับแต่งนี้รวมถึงการกำหนดค่า Learning Rate เป็น 0.00002 การเพิ่ม Weight Decay ที่ 0.00001 เพื่อช่วยลดการ Overfitting, การใช้ Cosine Learning Rate Scheduler ที่ช่วยให้ค่าการเรียนรู้ลดลงตามรอบการฝึก และการตั้งค่า Gradient Accumulation Steps เป็น 2 เพื่อเพิ่มขนาดแบทช์โดยรวม ซึ่งอาจช่วยปรับปรุงเสถียรภาพของการฝึกโมเดล

การเปรียบเทียบการตั้งค่าทั้งสองโมเดลสามารถดูได้จากตารางด้านล่าง ซึ่งสรุปพารามิเตอร์หลักที่ใช้ในแต่ละโมเดล และแสดงให้เห็นถึงความแตกต่างระหว่าง BERT1 และ BERT2 ในการฝึกอบรมโมเดลเพื่อจำแนกระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์

ตารางที่ 3.6 การตั้งค่าโมเดล BERT1 และ BERT2

Aspect	**BERT1**	**BERT2**
Tokenizer Used	bert-base-uncased	bert-base-uncased
Max Token Length	512	512
Dataset Creation	Dataset.from_dict()	Dataset.from_dict()
Model Architecture	bert-base-uncased	bert-base-uncased
Evaluation Metrics	accuracy, f1, precision, recall	accuracy, f1, precision, recall
Training Arguments		
Batch Size	Train: 16, Eval: 64	Train: 16, Eval: 64
Learning Rate	Not specified (Default)	0.00002
Epochs	5	5
Weight Decay	Not specified (Default)	0.00001
Gradient Accumulation	Not specified (Default)	2
LR Scheduler	Not specified (Default)	cosine

สำหรับการทดลองด้วยโมเดล BERT ในงานวิจัยนี้ ได้ทำการทดลองจำนวน 5 รอบในแต่ละกรณีศึกษาของชุดข้อมูลที่ 1 เพื่อให้ผลลัพธ์ที่ได้มีความน่าเชื่อถือและลดผลกระทบจากความผันผวนของค่าประสิทธิภาพในแต่ละรอบ โดยผลการทดลองในแต่ละรอบจะถูกนำมาคำนวณหาค่าเฉลี่ย เพื่อใช้เป็นตัวแทนผลลัพธ์สุดท้ายของแต่ละกรณี ทั้งนี้แนวทางดังกล่าวช่วยให้สามารถประเมินประสิทธิภาพของโมเดลได้อย่างเป็นธรรมและสม่ำเสมอในทุกการเปรียบเทียบที่เกิดขึ้นในงานวิจัยนี้

3.8 การวิเคราะห์และการประเมินการจำแนกระดับความรุนแรงแบบหลายคลาส

การวิเคราะห์และประเมินผลการจำแนกระดับความรุนแรงแบบหลายคลาส (Multiclass Classification) เป็นขั้นตอนสำคัญในงานวิจัยนี้ เพื่อวัดความสามารถของโมเดลในการแยกข้อมูลออกเป็นกลุ่มต่าง ๆ ตามระดับความรุนแรงของจุดบกพร่อง (Bug Severity) ตัวชี้วัดที่ใช้ในการประเมินประกอบด้วย Accuracy, F1 Score, Precision, Recall และ Matthews Correlation Coefficient (MCC) โดยแต่ละตัวชี้วัดมีบทบาทสำคัญในการแสดงภาพรวมของประสิทธิภาพโมเดลดังนี้ ซึ่งอ้างอิงจากตาราง Confusion Matrix สำหรับแต่ละชุดข้อมูล

ตารางที่ 3.7 Confusion Matrix

Predict / Actual	Blocker (1)	Critical (2)	Major (3)	Minor (4)	Trivial (5)
Blocker (1)	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
Critical (2)	X_{21}	X_{22}	X_{23}	X_{24}	X_{25}
Major (3)	X_{31}	X_{32}	X_{33}	X_{34}	X_{35}
Minor (4)	X_{41}	X_{42}	X_{43}	X_{44}	X_{45}
Trivial (5)	X_{51}	X_{52}	X_{53}	X_{54}	X_{55}

การวัดประสิทธิภาพการจำแนกสามารถอธิบายได้ผ่านเมทริกซ์ความสับสน (Confusion Matrix) ซึ่งเป็นตารางที่จำนวนแถวและคอลัมน์เท่ากับจำนวนคลาสในชุดข้อมูลที่พิจารณา ในงานวิจัยนี้ ชุดข้อมูลที่ใช้มีจำนวนคลาส 5 คลาส ซึ่งหมายถึงระดับความรุนแรง 5 ระดับ ได้แก่ Blocker Critical Major Minor และ Trivial ทำให้ Confusion Matrix มีขนาด 5x5 โดยข้อมูลในคอลัมน์แสดงค่าที่แบบจำลองทำนาย (Predict) และข้อมูลในแถวแสดงค่าจริง (Actual) แสดงดังตารางที่ 3.2

กำหนดให้ค่า TP (True Positive) คือ จำนวนข้อมูลที่ทำนายถูกว่าเป็นคลาสที่ i วิธีการคำนวณแสดงดังสมการที่ 3.2

$$TP_i = X_{ii} \tag{3.10}$$

กำหนดให้ค่า TTP (Total Numbers Of True Positive) คือ จำนวนข้อมูลที่ทำนายถูกว่าเป็นคลาสที่พิจารณาทั้งหมด วิธีการคำนวณแสดงดังสมการที่ 3.11

$$TTP = \sum_{i=1}^{n} X_{ii} \tag{3.11}$$

โดยที่ n คือ จำนวนคลาสทั้งหมด

กำหนดให้ค่า TFN (Total Numbers Of False Negative) คือ จำนวนข้อมูลที่ทำนายว่าเป็นคลาสที่ไม่ได้พิจารณาแต่คำตอบเป็นคลาสที่พิจารณา วิธีการคำนวณแสดงดังสมการที่ 3.12

$$TFN = \sum_{j=1, j \neq i}^{n} X_{ij} \quad 3.12$$

กำหนดให้ค่า TFP (Total Numbers Of False Positive) คือ จำนวนข้อมูลที่ทำนายว่าเป็นคลาสที่พิจารณาแต่คำตอบคือคลาสที่ไม่ได้พิจารณา วิธีการคำนวณแสดงดังสมการที่ 3.13

$$TFP = \sum_{j=1, j \neq i}^{n} X_{ji} \quad 3.13$$

ค่า TTN (Total Numbers Of True Negative) คือ จำนวนข้อมูลที่ทำนายถูกว่าเป็นคลาสที่ไม่ได้พิจารณา วิธีการคำนวณแสดงดังสมการที่ 3.14

$$TTN = \sum_{j=1, j \neq i}^{n} \sum_{k=1, k \neq i}^{n} X_{jk} \quad 3.14$$

การวัดค่าความถูกต้องของการจำแนก (Accuracy) คำนวณจากค่าผลรวมระหว่างจำนวนครั้งที่ทำนายถูก หารด้วยจำนวนการทำนายทั้งหมด วิธีการคำนวณแสดงดังสมการที่ 3.15

$$Accuracy = \frac{TTP}{Total\ Number\ Testing} \quad 3.15$$

การวัดค่าความแม่นยำของการจำแนก (Precision) คือการวิเคราะห์โดยพิจารณาแต่ละคลาสแยกกัน โดยคำนวณจากจำนวนครั้งที่โมเดลทำนายคลาสนั้นได้ถูกต้อง หารด้วยจำนวนครั้งทั้งหมดที่โมเดลทำนายว่าเป็นคลาสนั้น ซึ่งสามารถแสดงได้ในสมการที่ 3.16 เนื่องจากงานวิจัยนี้เกี่ยวข้องกับการทำนายข้อมูลแบบหลายคลาส (Multi-class) และจำนวนคลาสที่ไม่สมดุลกัน ดังนั้นค่าความแม่นยำแบบ Weighted จึงถูกนำมาใช้ประเนิน วิธีการคำนวณสามารถดูได้จากสมการที่ 3.17

$$Precision_i = \frac{TP_i}{TP_i + TFP_i} \quad 3.16$$

$$\text{Weighted Precision} = \frac{\sum_{i=1}^{n} (Precision_i \cdot Support_i)}{\sum_{i=1}^{n} Support_i} \quad 3.17$$

ค่าความระลึกของการจำแนก (Recall) คำนวณจากจำนวนครั้งที่ทำนายถูกในคลาสที่พิจารณา หารด้วยจำนวนครั้งที่ทำนายถูกในคลาสที่พิจารณาบวกจำนวนครั้งที่ทำนายผิดในคลาสที่ไม่ได้พิจารณา ดังสมการที่ 3.18 งานวิจัยชิ้นนี้ทำนายข้อมูลหลายคลาส (Multi class) และจำนวนคลาสที่ไม่สมดุลกัน ดังนั้นค่าความระลึกแบบ Weighted จึงถูกนำมาใช้ประเนิน วิธีการคำนวณแสดงดังสมการที่ 3.11

$$Recall_i = \frac{TP_i}{TP_i + TFN_i} \quad 3.18$$

$$Weighted\ Recall = \frac{\sum_{i=1}^{n}(Recall_i\ .\ Support_i)}{\sum_{i=1}^{n} Support_i} \quad 3.19$$

ค่าเฉลี่ยประสิทธิภาพโดยรวม (F-1) หาจากค่าเฉลี่ยจากค่าความแม่นยำและค่าความระลึก ระบบที่มีประสิทธิภาพดีจะต้องมีค่าความระลึกและค่าความแม่นยำสูงใกล้เคียงกัน ดังสมการที่ 3.20 งานวิจัยชิ้นนี้ทำนายข้อมูลหลายคลาส (Multi class) และจำนวนคลาสที่ไม่สมดุลกัน ดังนั้นค่าประสิทธิภาพโดยรวมแบบ Weighted จึงถูกนำมาใช้ประเนิน วิธีการคำนวณแสดงดังสมการที่ 3.21

$$F1_i = 2 \times \frac{Precision_i \times Recall_i}{Precision_i \times Recall_i} \quad 3.20$$

$$Weighted\ F1 = \frac{\sum_{i=1}^{n}(F1_i\ .\ Support_i)}{\sum_{i=1}^{n} Support_i} \quad 3.21$$

บทที่ 4

ผลการวิจัย

ในบทนี้จะเป็นการแสดงรายละเอียดผลลัพธ์จากแนวทางของงานวิจัยที่นำเสนอ ในการจำแนกระดับความรุนแรงแบบหลายคลาสจากรายงานจุดบกพร่อง ซึ่งมีผลการดำเนินงานในแต่ละขั้นตอน ดังนี้

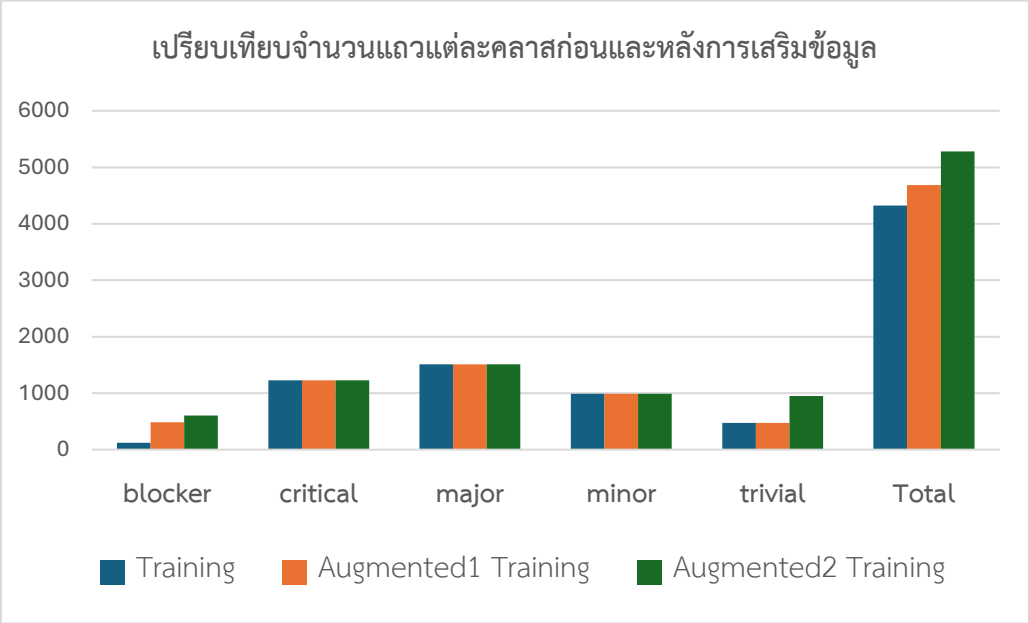
4.1 ผลการเสริมข้อมูลและการวิเคราะห์ข้อมูล

เนื่องจากชุดข้อมูลมีความไม่สมดุล ส่งผลให้โมเดลมีแนวโน้มให้ความสำคัญกับคลาสที่มีจำนวนมากและลดความแม่นยำในการทำนายคลาสที่มีจำนวนน้อย ดังนั้น การเสริมข้อมูลจึงถูกนำมาใช้เพื่อปรับสมดุลของชุดข้อมูล โดยการเพิ่มจำนวนข้อมูลในคลาสที่มีจำนวนน้อยให้มีสัดส่วนที่เหมาะสมมากขึ้นเมื่อเทียบกับคลาสอื่น วิธีการเสริมข้อมูลถูกดำเนินการในสองแนวทาง ได้แก่ (1) การเพิ่มจำนวนข้อมูลในคลาส blocker ขึ้น 3 เท่า และ (2) การเพิ่มข้อมูลในคลาส blocker 4 เท่า และ trivial 1 เท่า เพื่อให้จำนวนของคลาสเหล่านี้ใกล้เคียงกับคลาสที่มีจำนวนมากกว่า ข้อมูลถูกเสริมโดยใช้เทคนิค Text Generation ผ่านโมเดลทรานฟอร์เมอร์ โดยงานวิจัยนี้นำโมเดล T5 มาประยุกต์ใช้สำหรับสร้างข้อความใหม่ที่มีโครงสร้างใกล้เคียงกับข้อมูลต้นฉบับ แต่มีความหลากหลายมากขึ้น กระบวนการนี้ช่วยให้โมเดลสามารถเรียนรู้จากข้อมูลที่สมดุลกว่า ซึ่งช่วยเพิ่มประสิทธิภาพในการจำแนกระดับความรุนแรงของจุดบกพร่อง รายละเอียดการเปรียบเทียบจำนวนข้อมูลก่อนและหลังการเสริมข้อมูลแสดงในตารางที่ 4.1

ตารางที่ 4.1 เปรียบเทียบก่อนและหลังจากการเสริมข้อมูลของชุดข้อมูลที่ 1

Project	**Severity**					**Total**
	blocker	critical	major	minor	trivial	
Mozilla (Original)	158	1564	1866	1224	590	5402
Training Data	121	1228	1509	989	474	4321
Augmented 1	484	1228	1509	989	474	4684
Augmented 2	605	1228	1509	989	948	5279
Test Data	37	336	357	235	116	1081

จากตารางแสดงจำนวนตัวอย่างข้อมูลที่แบ่งตามระดับความรุนแรง (Severity) ของข้อบกพร่องในชุดข้อมูล Bugzilla โดยเปรียบเทียบระหว่างข้อมูลต้นฉบับ (Original) ชุดข้อมูลฝึก (Training Data) และชุดข้อมูลที่ได้รับการเสริม (Augmented 1 และ Augmented 2) รวมถึงชุดข้อมูลทดสอบ (Test Data) ข้อมูลต้นฉบับมีจำนวนรวม 5,402 ตัวอย่าง โดยคลาส major มีจำนวนมากที่สุด (1,866 ตัวอย่าง) และ blocker มีจำนวนน้อยที่สุด (158 ตัวอย่าง) สำหรับชุดข้อมูลฝึกจำนวนตัวอย่างโดยรวม 4,321 ตัวอย่าง โดยยังคงมีความไม่สมดุลระหว่างคลาส ชุดข้อมูล Augmented 1 ดำเนินการเพิ่มตัวอย่าง blocker เป็น 484 ตัวอย่าง ในขณะที่ Augmented 2 เพิ่ม blocker เป็น 605 ตัวอย่าง และเพิ่ม trivial เป็น 948 ตัวอย่าง เพื่อลดความไม่สมดุล ส่วนชุดข้อมูลทดสอบมีขนาดที่สุดที่ 1,081 ตัวอย่าง หรือคิดเป็น 20% จากข้อมูลต้นฉบับ การแบ่งข้อมูลชุดฝึกก่อนการนำไปเสริมเป็นการป้องกันการรั่วไหลของข้อมูล



รูปที่ 4.1 เปรียบเทียบจำนวนแถวแต่ละคลาสก่อนและหลังการเสริมข้อมูลชุดที่ 1

รูปที่ 4.1 กราฟแสดงการเปรียบเทียบจำนวนแถวในแต่ละคลาสของข้อมูลชุดฝึกก่อนและหลังการเสริมข้อมูล โดยพิจารณาจากคลาสของระดับความรุนแรงของข้อบกพร่อง (bug severity level) ได้แก่ blocker, critical, major, minor และ trivial นอกจากนี้ยังมีการแสดงผลรวมของข้อมูลทั้งหมด (Total) เพื่อให้เห็นภาพรวมของปริมาณข้อมูลก่อนและหลังการเสริมข้อมูล จากกราฟพบว่าปริมาณข้อมูลในแต่ละคลาสเพิ่มขึ้นหลังจากกระบวนการเสริมข้อมูล โดยเฉพาะคลาสที่มีจำนวนน้อยในชุดข้อมูลเดิม เช่น blocker และ trivial แสดงให้เห็นว่ากระบวนการเสริมข้อมูลสามารถช่วยเพิ่มความสมดุลของจำนวนข้อมูลในแต่ละคลาสได้ อย่างไรก็ตาม การเพิ่มขึ้นของข้อมูลไม่ได้มีสัดส่วน

เท่ากันทุกคลาส เพราะการงานวิจัยมุ่งเน้นการเสริมข้อมูลในส่วนของคลาสทีมีจำนวนน้อย และในส่วนของ Total พบว่าปริมาณข้อมูลโดยรวมเพิ่มขึ้นอย่างชัดเจน ซึ่งบ่งชี้ว่ากระบวนการเสริมข้อมูลสามารถเพิ่มขนาดของชุดข้อมูลสำหรับการฝึกโมเดลได้ ซึ่งอาจส่งผลให้แบบจำลองเรียนรู้ได้ดีขึ้นและสามารถจำแนกระดับความรุนแรงของข้อบกพร่องได้อย่างมีประสิทธิภาพมากขึ้น โดยเฉพาะในกรณีที่ข้อมูลดั้งเดิมมีความไม่สมดุลระหว่างคลาส โดยสรุป กราฟนี้แสดงให้เห็นถึงผลของการเสริมข้อมูลที่ช่วยเพิ่มจำนวนข้อมูลในแต่ละคลาส ทำให้ชุดข้อมูลมีความสมดุลมากขึ้น ซึ่งเป็นปัจจัยสำคัญในการพัฒนาโมเดลการจำแนกประเภทที่สามารถทำงานได้อย่างแม่นยำยิ่งขึ้น แสดงตัวอย่างข้อความที่ได้จากการเสริมข้อมูลใหม่ดังตาราง

ตารางที่ 4.2 เปรียบเทียบข้อความที่ถูกสร้างจากวิธีการเสริมข้อมูล

	Text
Summary[i] (ต้นฉบับ)	searching from searchbox throws "textbox is undefined"
NewSummary[i][1]	from major xml searchbox blocking search browser
NewSummary[i][2]	from major xml searchbox blocking search browser is a key feature for blocking search results
Description[i] (ต้นฉบับ)	After upgrading to today's nightly build, doing a search using the searchbox throws: Error: textBox is undefined Source file: chrome://browser/content/search/search.xml Line: 431 I'm on Linux, but cbeard noticed this on Mac as well, so setting platform/OS to all, and requesting blocking, as this is major bustage.
NewDescription[i][1]	after upgrading to today's nightly build, doing a search using the searchbox throws: Error: textBox is undefined Source file: chrome://browser/content/search/search.xml Line: 431 I'm on Linux, but cbeard noticed this on Mac as well, so setting platform/OS to all, and requesting blocking, as this is major bustage.

NewDescription[i][2]	the searchbox throws: Error: textbox is undefined Source file: chrome://browser/content/search/search.xml Line: 431 I'm on Linux, but cbeard noticed this on Mac .

นอกจากการใช้โมเดล T5 ในการเสริมข้อมูลแล้ว ผู้วิจัยยังได้ดำเนินการเสริมข้อมูลเพิ่มเติมโดยใช้เทคนิค Easy Data Augmentation (EDA) ผ่านวิธีการแทนที่คำพ้องความหมาย (synonym replacement) โดยอาศัยฐานข้อมูล WordNet ซึ่งเป็นแหล่งรวมคำศัพท์และความหมายที่ใช้กันอย่างแพร่หลายในการประมวลผลภาษาธรรมชาติ วิธีนี้ช่วยเพิ่มความหลากหลายของข้อความในชุดข้อมูลโดยไม่ทำให้ความหมายของข้อความเปลี่ยนแปลงไปอย่างมีนัยสำคัญ ทั้งนี้ การเสริมข้อมูลด้วย synonym replacement ถูกนำมาใช้เพื่อเปรียบเทียบประสิทธิภาพกับการเสริมข้อมูลด้วย T5 ซึ่งเป็นแนวทางหลักของงานวิจัยนี้ การเปรียบเทียบทั้งสองวิธีจะช่วยให้เข้าใจถึงข้อดีและข้อจำกัดของแต่ละแนวทางในการเพิ่มข้อมูลสำหรับการจำแนกระดับความรุนแรงของข้อบกพร่อง โดยรายละเอียดและตัวอย่างของผลลัพธ์จากการแทนที่คำพ้องความหมายแสดงในตัวอย่าง

ตารางที่ 4.3 เปรียบเทียบข้อความที่ถูกสร้างจากวิธีการเสริมข้อมูล

Text	**Original / Augmented**
Summary[i]	searching from searchbox throws "textbox is undefined"
NewSummary[i][1]	searching from searchbox throws "textbox be undefined"
NewSummary[i][2]	searching from searchbox throw " textbox is vague"
Description[i]	After upgrading to today's nightly build, doing a search using the searchbox throws: Error: textBox is undefined Source file: chrome://browser/content/search/search.xml Line: 431 I'm on Linux, but cbeard noticed this on Mac as well, so setting platform/OS to all, and requesting blocking, as this is major bustage.
NewDescription[i][1]	After upgrading to today ' s nightly build, doing a search using the searchbox throws: Error: textBox is undefined Source file cabinet: chrome: / / web browser / content / search / hunt. xml Line: 431 I ' 1000 on Linux, but cbeard noticed this on Mac

Text	Original / Augmented
	equally well, so setting platform / OS to all, and requesting blocking, as this is major bustage.
NewDescription[i][2]	After upgrading to today ' sulphur nightly build, act a search use the searchbox throws: Error: textBox is undefined Rootage file: chrome: / / browser / content / search / search. xml Line: 431 Ane ' m on Linux, but cbeard noticed this on Mac every bit well, so setting platform / OS to all, and requesting blocking, as this is major bustage.

4.2 ผลการจำแนกระดับความรุนแรงจากชุดข้อมูลเดิม

ผลของการจำแนกระดับความรุนแรงของจุดบกพร่องจากการใช้ชุดข้อมูลเดิมก่อนการเสริม โดยใช้โมเดลต่าง ๆ พบว่าโมเดล BERT2 และ BERT1 ให้ผลลัพธ์ที่ดีที่สุดในแง่ของ F1-score และ Accuracy โดย BERT2 มีค่า F1-score 61.77% และ Accuracy 63.16% ขณะที่ BERT1 ให้ค่า F1-score 61.80% และ Accuracy 62.70% ซึ่งสูงกว่าระบบการจำแนกแบบดั้งเดิม เช่น SVM และ Logistic Regression (LR) ที่มีค่า F1-score ประมาณ 56.5% และ Accuracy ประมาณ 58% แสดงให้เห็นว่าโมเดลที่ใช้ Transformer-based architecture สามารถเรียนรู้และจำแนกข้อมูลข้อความของรายงานข้อบกพร่องได้ดีกว่าโมเดลอื่น ๆ นอกจากนี้ Ensemble Stacking ให้ผลลัพธ์ที่ดีกว่าโมเดลพื้นฐานทั่วไป โดยมี F1-score 60.05% และ Accuracy 60.68% ซึ่งบ่งบอกว่าการรวมโมเดลหลายตัวสามารถช่วยเพิ่มประสิทธิภาพการจำแนกได้ ส่วน Random Forest (RF) มีผลลัพธ์ต่ำที่สุดในกลุ่มโมเดลดั้งเดิม โดยมีค่า F1-score 51.87% และ Accuracy 55.69% ขณะที่ LSTM ซึ่งเป็นโมเดลที่ใช้โครงสร้าง RNN-based ให้ผลลัพธ์ต่ำที่สุด โดยมี F1-score 47.87% และ Accuracy 49.40% ซึ่งอาจเกิดจากข้อจำกัดของ LSTM ในการจับความสัมพันธ์ระหว่างข้อความที่ซับซ้อน

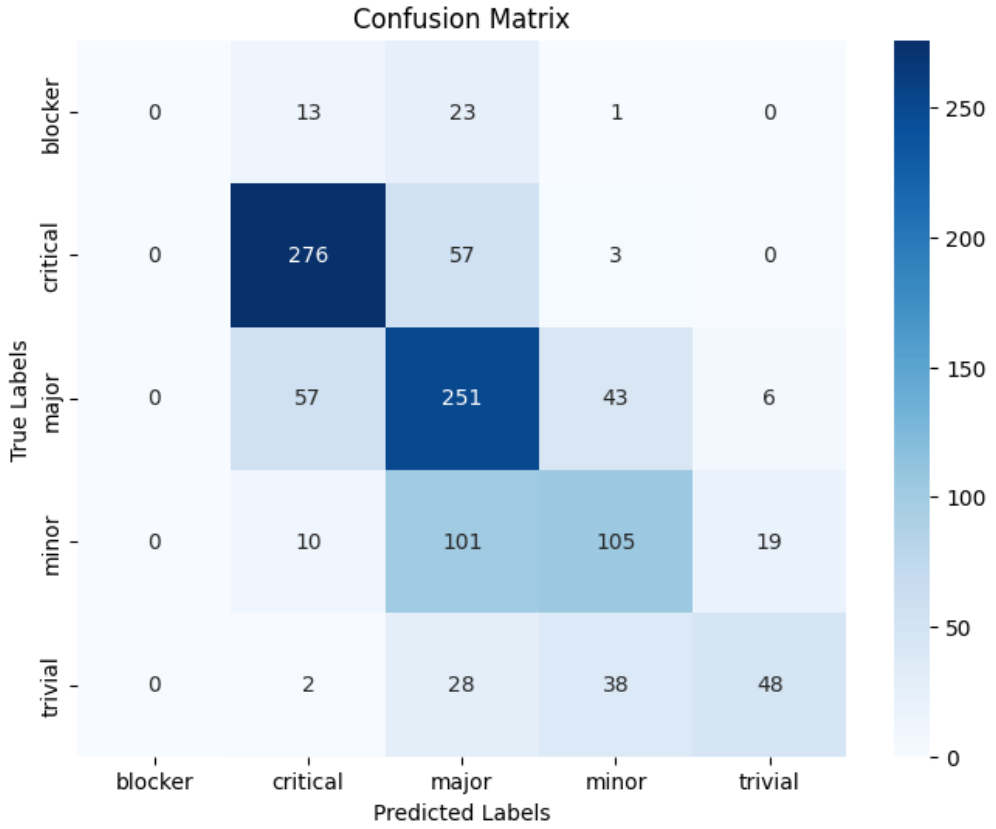
ตารางที่ 4.4 ผลการจำแนกระดับความรุนแรงของจุดบกพร่องจากการใช้ชุดข้อมูลเดิม

Models	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
SVM	62.45	58.46	56.09	58.46	0.42	19.75
LR	61.97	58.93	57.10	58.93	0.43	9.04
RF	58.96	56.80	52.56	56.80	0.40	8.83
SACKING	62.08	60.68	60.05	60.68	0.45	12956.80
LSTM	51.37	47.46	47.46	47.46	0.29	199.81
BERT1	61.78±1.15	62.70±0.35	61.80±0.56	62.70±0.35	0.49±0.005	150.91±3.23
BERT2	62.17±1.80	63.16±0.52	61.77±0.62	63.16±0.52	0.49±0.007	149.62±2.77

โดยสามารถสรุปผลลัพธ์ของการจำแนกระดับความรุนแรงของจุดบกพร่องจากการใช้ชุดข้อมูลเดิม

1. BERT2 และ BERT1 เป็นโมเดลที่มีประสิทธิภาพดีที่สุดในทุกตัวชี้วัด แสดงให้เห็นถึงศักยภาพของโมเดล Transformer ในการจำแนกระดับความรุนแรงของจุดบกพร่อง และ BERT2 ให้ผลลัพธ์ดีกว่า BERT1 ซึ่งมาจากการตั้งค่า parameter ที่เหมาะสมสำหรับการทำงานของโดมเดนชุดข้อมูลรายงานจุดบกพร่อง
2. Ensemble Stacking ให้ผลลัพธ์ที่ดีกว่าโมเดลแบบดั้งเดิม แสดงถึงประโยชน์ของการรวมโมเดลเพื่อเพิ่มความแม่นยำ
3. SVM และ Logistic Regression (LR) ยังเป็นโมเดลพื้นฐานที่ให้ผลลัพธ์ค่อนข้างดี แม้จะด้อยกว่าโมเดลเชิงลึก
4. Random Forest (RF) มีประสิทธิภาพต่ำกว่าทั้ง SVM และ LR เนื่องจากข้อจำกัดในการเรียนรู้ข้อมูลข้อความ
5. LSTM เป็นโมเดลที่ให้ผลลัพธ์ต่ำที่สุด อาจเกิดจากข้อจำกัดด้านโครงสร้างโมเดลที่ไม่สามารถจับความสัมพันธ์ของข้อมูลได้ดีเท่ากับโมเดล Transformer

จากผลลัพธ์ที่ได้ แนะนำให้ใช้ BERT-based models เป็นโมเดลหลักสำหรับการจำแนกระดับความรุนแรงของจุดบกพร่อง เนื่องจากมีความสามารถในการประมวลผลข้อความที่มีโครงสร้างซับซ้อนและให้ผลลัพธ์ที่แม่นยำกว่าโมเดลอื่นๆ



รูปที่ 4.2 Confusion Matrix ของโมเดล BERT2 จากชุดข้อมูลเดิม

การศึกษานี้ได้ทำการวิเคราะห์เฉพาะโมเดลที่ดีที่สุดของชุดข้อมูลนั้น ๆ ซึ่งจาก Confusion Matrix ของโมเดล BERT2 ที่ใช้จำแนกระดับความรุนแรงของจุดบกพร่องจากชุดข้อมูลเดิม สามารถวิเคราะห์ได้ดังนี้

ความถูกต้องของการจำแนกในแต่ละคลาส

- Critical: มีจำนวนตัวอย่างทั้งหมดที่เป็น critical อยู่ที่ 336 ตัวอย่าง (276+57+3) ซึ่งโมเดลจำแนกได้ถูกต้อง 276 ตัวอย่าง คิดเป็น 82.14%
- Major: มีทั้งหมด 357 ตัวอย่าง (57+251+43+6) และโมเดลจำแนกได้ถูกต้อง 251 ตัวอย่าง คิดเป็น 70.31%
- Minor: มีทั้งหมด 235 ตัวอย่าง (10+101+105+19) โมเดลจำแนกถูกต้อง 105 ตัวอย่าง คิดเป็น 44.68%
- Trivial: มีทั้งหมด 116 ตัวอย่าง (2+28+38+48) โมเดลจำแนกถูกต้อง 48 ตัวอย่าง คิดเป็น 41.38%

- Blocker: มีทั้งหมด 37 ตัวอย่าง (13+23+1) แต่โมเดลจำแนกได้ถูกต้อง 0 ตัวอย่าง

แนวโน้มข้อผิดพลาดของโมเดล

- โมเดลสามารถจำแนก Critical และ Major ได้ดีที่สุด โดยมีค่า True Positive สูง
- ข้อมูลประเภท Minor และ Trivial มีการสับสนสูง ซึ่งโมเดลมีแนวโน้มจำแนกคลาส minor ไปเป็น major และ trivial ไปเป็น minor
- Blocker เป็นคลาสที่โมเดลจำแนกผิดพลาดทั้งหมด เนื่องจากไม่มีการพยากรณ์ค่าเป็น blocker เลย (ค่าในแถว Blocker เป็น 0 ในคอลัมน์ Blocker) อาจเป็นเพราะจำนวนตัวอย่างของคลาสนี้น้อยเกินไปในการฝึกโมเดล

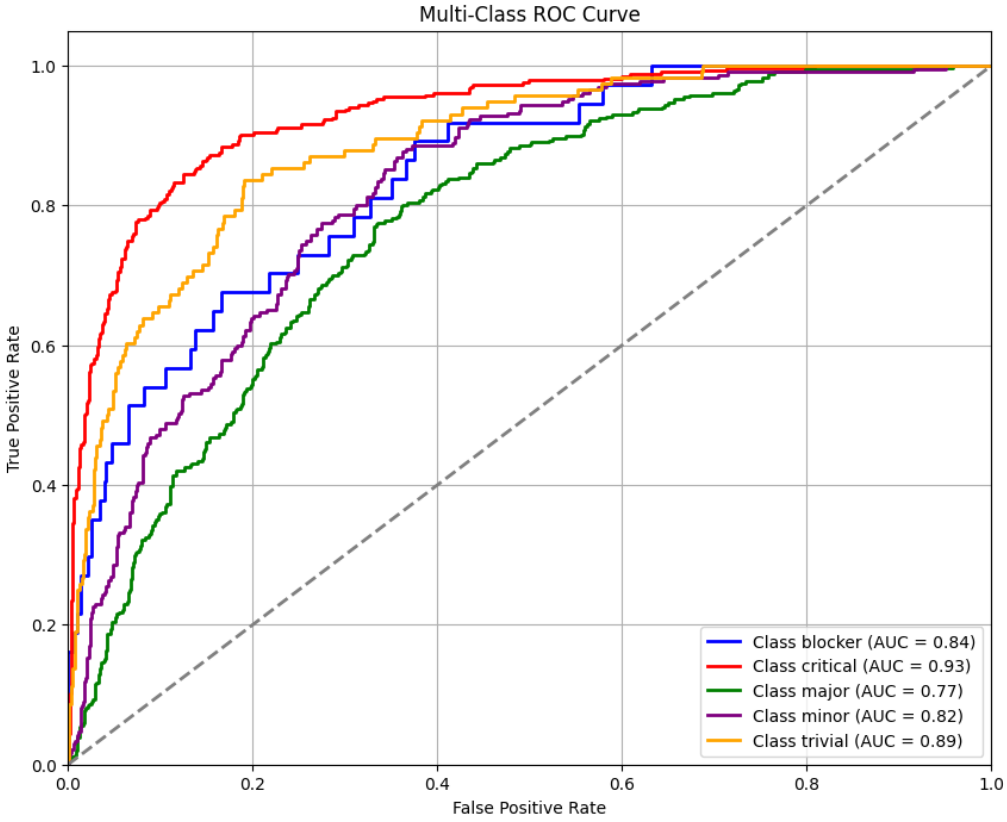
การกระจายของข้อผิดพลาด

- โมเดลมีแนวโน้มที่จะจำแนก minor และ trivial ไปยังคลาสที่มีความรุนแรงใกล้เคียงกัน เช่น จาก minor ⟶ major และ trivial ⟶ minor ซึ่งเป็นไปตามคาด เพราะระดับความรุนแรงของข้อผิดพลาดที่ใกล้เคียงกันมักมีลักษณะข้อความที่คล้ายกัน
- ข้อผิดพลาดใน critical ⟶ major และ major ⟶ critical พบได้เช่นกัน ซึ่งอาจเกิดจากความคล้ายคลึงกันของรายละเอียดในรายงานจุดบกพร่อง
- การจำแนก blocker ผิดพลาดทั้งหมด อาจเป็นเพราะตัวอย่างของ blocker มีจำนวนน้อยเกินไปจนโมเดลไม่สามารถเรียนรู้ได้ดี

จากตาราง Confusion Matrix สามารถสรุปผลการทดลองได้ดังนี้

1. โมเดล BERT2 สามารถจำแนก critical และ major ได้แม่นยำสูง แต่ยังมีความคลาดเคลื่อนใน minor และ trivial
2. Blocker เป็นปัญหาหลัก เนื่องจากโมเดลไม่สามารถจำแนกคลาสนี้ได้เลย ซึ่งอาจต้องเพิ่มตัวอย่างในกลุ่มนี้หรือใช้เทคนิค oversampling
3. ความผิดพลาดส่วนใหญ่เกิดขึ้นในคลาสที่ใกล้เคียงกัน ซึ่งเป็นลักษณะปกติของปัญหาการจำแนกระดับความรุนแรงของจุดบกพร่อง

การปรับปรุงเพิ่มเติมอาจใช้เทคนิคเสริมข้อมูล หรือ Oversampling สำหรับ blocker เพื่อช่วยให้โมเดลสามารถเรียนรู้จากตัวอย่างของคลาสที่มีจำนวนน้อยได้ดีขึ้น



รูปที่ 4.3 ROC Curve ของโมเดล BERT2 จากชุดข้อมูลเดิม

กราฟ Multi-Class ROC Curve ของโมเดล BERT2 จากชุดข้อมูลเดิม แสดงประสิทธิภาพของโมเดลในการจำแนกระดับความรุนแรงของจุดบกพร่อง โดยใช้ค่า AUC เป็นตัวชี้วัด ค่าที่ใกล้ 1.0 แสดงว่าโมเดลสามารถแยกแยะคลาสนั้นได้ดี จากผลลัพธ์พบว่า Critical มีค่า AUC สูงสุดที่ 0.93 ซึ่งหมายความว่าโมเดลสามารถแยกคลาสนี้ออกจากคลาสอื่นได้ดี รองลงมาคือ Trivial (0.89), Blocker (0.84) และ Minor (0.82) ซึ่งยังมีความสามารถในการจำแนกที่ค่อนข้างดี อย่างไรก็ตาม คลาส Major มีค่า AUC ต่ำสุดที่ 0.77 แสดงว่าโมเดลมีปัญหาในการแยกคลาสนี้ออกจากคลาสอื่น

ค่าของ AUC ที่ต่ำในคลาส Major อาจเกิดจาก ความคล้ายคลึงของข้อมูลกับ Critical และ Minor ทำให้โมเดลสับสน และ การกระจายตัวของข้อมูลที่ไม่สมดุล อาจส่งผลให้โมเดลเรียนรู้การจำแนก Major ได้ไม่ดีนัก นอกจากนี้ เส้นโค้ง ROC ของ Major มีความลาดชันต่ำกว่าเมื่อเทียบกับคลาสอื่น แสดงว่าโมเดลมี True Positive Rate ต่ำและ False Positive Rate สูง ซึ่งอาจแก้ไขได้โดยการ ปรับสมดุลข้อมูล หรือ ใช้ Feature Engineering เพิ่มเติมเพื่อลดความคล้ายคลึงระหว่างคลาส Major และคลาสที่ใกล้เคียงกัน

4.3 ผลการจำแนกระดับความรุนแรงด้วยเทคนิคการจัดการข้อมูลไม่สมดุล

การทดลองนี้มีวัตถุประสงค์เพื่อเปรียบเทียบวิธีการที่นำเสนอและเทคนิคการแก้ปัญหาข้อมูลประเภทข้อความไม่สมดุลที่เป็นเทคนิคมาตรฐาน ผลการทดลองประกอบด้วย 2 ส่วนดังนี้

4.3.1 ผลทดลองด้วยการใช้ SMOTE และ Class Weight

การจำแนกระดับความรุนแรงของจุดบกพร่องโดยใช้เทคนิคการแก้ปัญหาความไม่สมดุลของข้อมูลได้รับการดำเนินการโดยการใช้แนวทางมาตรฐานเพื่อปรับสมดุลของคลาสข้อมูล ซึ่งในส่วนของโมเดลการเรียนรู้ของเครื่อง ได้แก่ Support Vector Machine (SVM), Logistic Regression, Random Forest และ Stacking ได้มีการนำเทคนิค SMOTE มาใช้ในสามรูปแบบหลัก ได้แก่ (1) Oversampling แบบสมบูรณ์ ซึ่งเป็นการเพิ่มจำนวนตัวอย่างของทุกคลาสให้เท่ากับคลาสที่มีจำนวนตัวอย่างมากที่สุด (2) Sampling Strategy ตามอัตราส่วนที่กำหนด ซึ่งกำหนดให้คลาส Blocker เพิ่มขึ้น 3 เท่า และ (3) Sampling Strategy แบบขยายขนาดแตกต่างกัน โดยเพิ่มจำนวนคลาส Blocker 4 เท่า และ Trivial 1 เท่า เพื่อให้เกิดความสมดุลระหว่างคลาสมากขึ้นและเป็นอตราส่วนเดียวกันกับวิธีที่นำเสนอด้วยการเสริมโดยโมเดล T5 ในขณะที่สำหรับโมเดล BERT ผู้วิจัยได้นำเทคนิค Class Weight แบบ Balanced ซึ่งเป็นฟังก์ชันที่มีอยู่ในไลบรารี scikit-learn มาใช้ในการลดอคติของโมเดลที่มักให้ความสำคัญกับคลาสที่มีจำนวนตัวอย่างมากกว่า ทั้งนี้ ผลลัพธ์จากการทดลองแสดงในตารางด้านล่างเพื่อเปรียบเทียบประสิทธิภาพของแต่ละเทคนิคในการจัดการกับข้อมูลที่ไม่สมดุล

ตารางที่ 4.5 ผลการจำแนกระดับความรุนแรงด้วยเทคนิคการจัดการข้อมูลไม่สมดุล

Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
SMOTE (Oversampling)						
SVM	60.42	58.09	56.37	58.09	0.42	37.47
LR	58.57	58.28	<u>58.39</u>	58.28	<u>0.44</u>	8.80
RF	57.78	56.61	55.72	56.61	0.39	8.22
SACKING	<u>61.72</u>	<u>59.67</u>	58.29	<u>59.67</u>	<u>0.44</u>	25108.56
Smote (sampling strategy Blocker 3 เท่า)						
SVM	62.56	58.74	56.49	58.74	0.43	58.74
LR	60.65	59.20	58.10	59.20	0.43	10.74

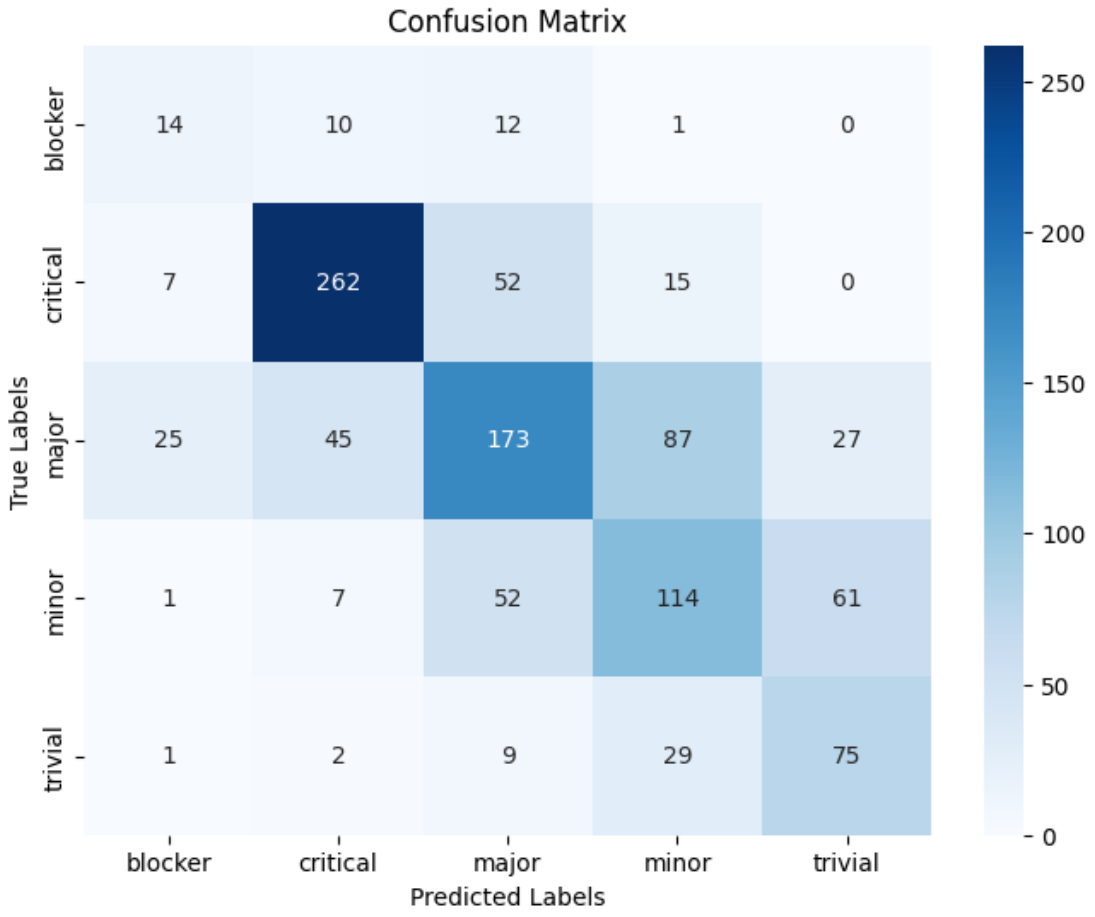
Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
RF	62.51	56.52	52.56	56.52	0.40	9.53
SACKING	63.40	61.98	61.22	61.98	0.47	13435.35
Smote (sampling strategy Blocker 4 เท่า Trivial 1 เท่า)						
SVM	62.78	57.21	59.20	59.20	0.43	51.48
LR	60.57	60.22	59.60	60.22	0.45	10.40
RF	58.89	55.50	51.98	55.50	0.38	12.16
SACKING	63.55	61.33	60.44	61.33	0.46	15930.80
Class weight						
BERT1	61.94±1.00	61.67±0.65	61.54±0.87	61.67±0.65	0.48±0.009	157.11±2.98
BERT2	62.87±0.62	61.82±0.50	61.92±0.33	61.82±0.50	0.49±0.01	149.65±2.86

จากผลลัพธ์ในตารางที่ 4.5 แสดงให้เห็นว่าการใช้เทคนิคการจัดการข้อมูลไม่สมดุลมีผลต่อประสิทธิภาพของโมเดลในด้าน F1-score และ Accuracy โดยรวม ทั้งนี้ เมื่อเปรียบเทียบวิธี SMOTE (Oversampling) กับแนวทาง Sampling Strategy ที่กำหนดอัตราส่วนของการเพิ่มข้อมูล พบว่าการกำหนดอัตราส่วนที่เหมาะสมสามารถเพิ่มประสิทธิภาพของโมเดลได้อย่างมีนัยสำคัญ โดยเฉพาะในกรณีที่ใช้ Blocker 4 เท่า และ Trivial 1 เท่า ซึ่งให้ค่า F1-score สูงกว่า Oversampling แบบสมบูรณ์ในหลายโมเดล

ในการเปรียบเทียบระหว่างโมเดลการเรียนรู้ของเครื่อง พบว่า Stacking Model ให้ผลลัพธ์ที่ดีที่สุดในทุกกรณี โดยมีค่า F1-score และ Accuracy สูงกว่ารุ่นอื่นๆ อย่างต่อเนื่อง ซึ่งบ่งชี้ว่าแนวทางการรวมผลลัพธ์จากหลายโมเดลช่วยเพิ่มความแม่นยำของการจำแนกระดับความรุนแรงของข้อบกพร่อง ในขณะที่ Logistic Regression (LR) และ Support Vector Machine (SVM) มีประสิทธิภาพใกล้เคียงกัน โดยเฉพาะในกรณีของ Sampling Strategy (Blocker 4 เท่า, Trivial 1 เท่า) ซึ่งให้ค่า Accuracy สูงสุดที่ 60.22% สำหรับ LR และ 59.2% สำหรับ SVM

สำหรับโมเดลที่ใช้ Class Weight (BERT1 และ BERT2) ซึ่งฝึกด้วย GPU A100 พบว่า BERT2 ให้ค่า F1-score และ Accuracy สูงสุดที่ 61.92% และ 61.82% ตามลำดับ ซึ่งเหนือกว่าโมเดลการเรียนรู้ของเครื่องทุกตัวที่ใช้เทคนิคการเพิ่มข้อมูลด้วย SMOTE แม้ว่า BERT จะใช้ทรัพยากรประมวลผลสูงกว่า แต่ผลลัพธ์ที่ได้แสดงให้เห็นว่าโมเดลสามารถเรียนรู้บริบทของข้อความได้ดีกว่าการใช้วิธี Oversampling เพียงอย่างเดียว

นอกจากนี้ เมื่อตรวจสอบด้าน เวลาในการประมวลผล พบว่า Stacking Model ใช้เวลาฝึกที่ยาวนานกว่าทุกโมเดล โดยเฉพาะเมื่อใช้ SMOTE (Oversampling) ซึ่งใช้เวลาสูงสุดถึง 25,108.56 วินาที ขณะที่การใช้ Sampling Strategy แบบ Blocker 4 เท่า และ Trivial 1 เท่า สามารถลดระยะเวลาเหลือ 15,930.8 วินาที แสดงให้เห็นว่า การเลือกอัตราส่วนที่เหมาะสมของ SMOTE สามารถลดระยะเวลาการฝึกโมเดลได้โดยไม่ลดทอนประสิทธิภาพมากนัก ส่วนโมเดล BERT แม้จะใช้ GPU A100 ก็ยังคงมีระยะเวลาในการประมวลผลที่สูงกว่าโมเดลการเรียนรู้ของเครื่องทั่วไป แต่ด้วยประสิทธิภาพที่สูงกว่าทำให้เหมาะสมสำหรับงานที่ต้องการความแม่นยำสูง โดยสรุป ผลการทดลองชี้ให้เห็นว่า การใช้ Sampling Strategy ที่เหมาะสม สามารถช่วยให้โมเดลการเรียนรู้ของเครื่องมีประสิทธิภาพที่ดีขึ้น ในขณะที่โมเดล BERT ที่ใช้ Class Weight แบบ Balanced สามารถจำแนกความรุนแรงของจุดบกพร่องได้แม่นยำที่สุด อย่างไรก็ตาม การเลือกใช้โมเดลควรพิจารณา ทรัพยากรในการประมวลผล ควบคู่กับ ความต้องการด้านความแม่นยำ เพื่อให้ได้แนวทางที่เหมาะสมกับข้อจำกัดของระบบ

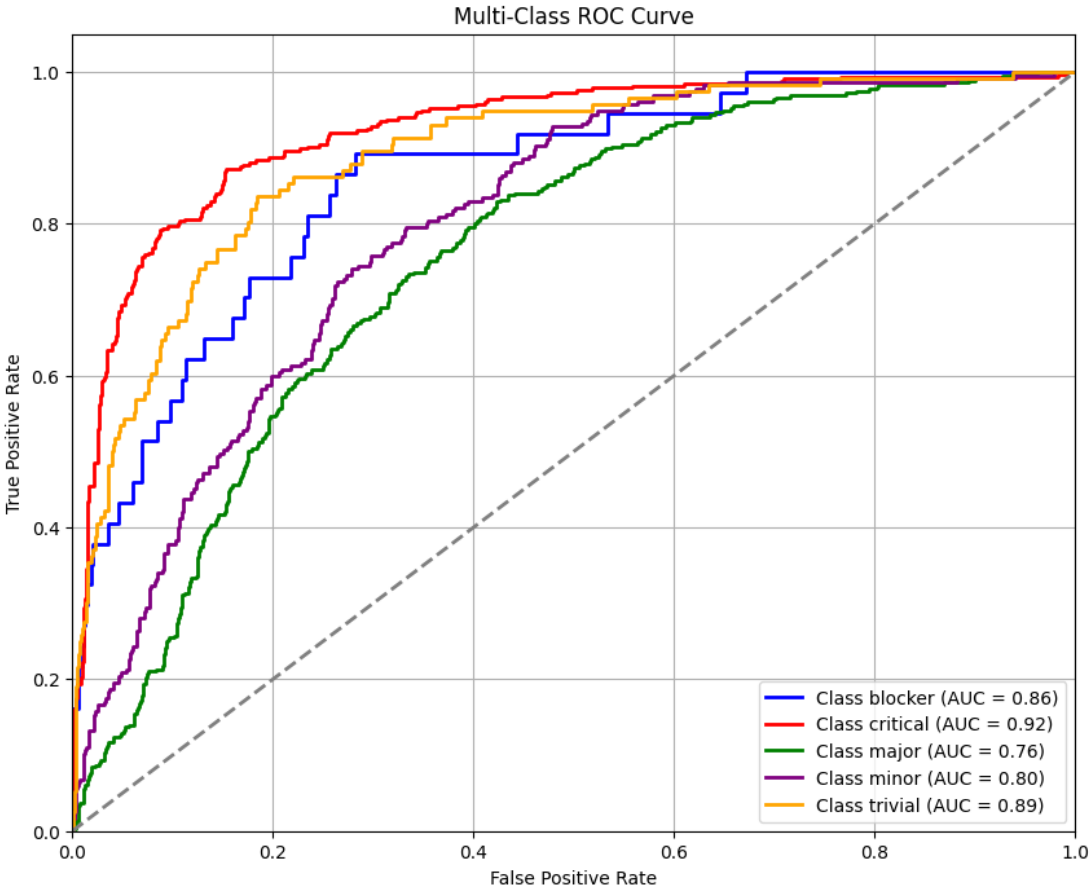


รูปที่ 4.4 Confusion Matrix ของโมเดล BERT2 จากการทำ Class Weight

Confusion Matrix ในรูปที่ 4.4 แสดงผลการจำแนกระดับความรุนแรงของจุดบกพร่องโดยใช้โมเดล BERT2 พร้อมเทคนิค Class Weight เพื่อลดปัญหาข้อมูลไม่สมดุล พบว่าโมเดลสามารถจำแนกคลาส Critical ได้ดีที่สุด โดยมีตัวอย่างที่ถูกต้อง 262 รายการ จากทั้งหมด และมีการพยากรณ์ผิดไปยัง Major (52 รายการ) และ Minor (15 รายการ) ในขณะที่คลาส Blocker ซึ่งเป็นคลาสที่มีจำนวนน้อยที่สุด โมเดลสามารถจำแนกได้ถูกต้อง 14 ตัวอย่าง แต่ยังมีข้อผิดพลาดที่พยากรณ์ไปยัง Critical (10 รายการ) และ Major (12 รายการ)

สำหรับคลาส Major มีการจำแนกที่แม่นยำน้อยกว่าคลาส Critical โดยถูกต้องเพียง 173 รายการ แต่มีข้อผิดพลาดที่พยากรณ์ไปยัง Critical (45 รายการ), Minor (87 รายการ), และ Trivial (27 รายการ) ซึ่งสะท้อนถึงความสับสนของโมเดลที่ยังแยก Major ออกจาก Critical และ Minor ได้ไม่ดีนัก นอกจากนี้ คลาส Minor และ Trivial มีอัตราการพยากรณ์ผิดพลาดสูง โดย Minor มีเพียง 114 รายการที่ถูกต้อง แต่มีการพยากรณ์ผิดไปยัง Major (52 รายการ) และ Trivial (61 รายการ) ส่วน Trivial มี 75 รายการที่ถูกต้อง แต่ยังมีข้อผิดพลาดที่ถูกพยากรณ์เป็น Minor (29 รายการ) และ Major (9 รายการ)

โดยรวม การใช้ Class Weight ช่วยให้โมเดลสามารถจำแนก Critical และ Blocker ได้ดีขึ้น แต่ยังคงมีความสับสนในคลาส Major และ Minor ซึ่งอาจต้องปรับปรุงเพิ่มเติมโดยใช้ การปรับค่าถ่วงน้ำหนักเพิ่มเติม หรือการใช้ Oversampling เพื่อให้โมเดลเรียนรู้ขอบเขตของคลาส Major และ Minor ได้ชัดเจนขึ้น นอกจากนี้ การวิเคราะห์ข้อความที่ทำให้โมเดลพยากรณ์ผิดพลาด อาจช่วยให้เข้าใจสาเหตุของความสับสนและสามารถพัฒนา Feature Engineering ที่ช่วยให้โมเดลเรียนรู้ได้ดีขึ้น



รูปที่ 4.5 ROC Curve ของโมเดล BERT2 จากการทำ Class Weight

กราฟ Multi-Class ROC Curve แสดงประสิทธิภาพของโมเดล BERT2 ในการจำแนกระดับความรุนแรงของจุดบกพร่องโดยใช้เทคนิค Class Weight เพื่อลดปัญหาข้อมูลไม่สมดุล ค่า AUC (Area Under Curve) ที่สูงบ่งบอกถึงความสามารถของโมเดลในการแยกแยะแต่ละคลาสได้ดี จากผลลัพธ์พบว่า Critical มีค่า AUC สูงสุดที่ 0.92 แสดงว่าโมเดลสามารถจำแนกคลาสนี้ได้อย่างแม่นยำ รองลงมาคือ Trivial (0.89), Blocker (0.86), และ Minor (0.80) ซึ่งยังอยู่ในระดับที่ดี อย่างไรก็ตาม Major มีค่า AUC ต่ำสุดที่ 0.76 ซึ่งหมายความว่าโมเดลยังมีปัญหาในการแยกคลาสนี้ออกจากคลาสอื่น

ค่าของ AUC ที่ต่ำในคลาส Major อาจเกิดจาก ความคล้ายคลึงของข้อมูลกับคลาส Critical และ Minor ทำให้โมเดลสับสน นอกจากนี้ จาก Confusion Matrix ในรูปที่ 4.4 พบว่าคลาส Major มีการกระจายไปยัง Minor และ Critical สูง ซึ่งอาจทำให้โมเดลไม่สามารถกำหนดเส้นแบ่งที่ชัดเจนได้ แม้ว่าการใช้ Class Weight จะช่วยให้โมเดลสามารถเรียนรู้จากคลาสที่มีตัวอย่างน้อยได้ดีขึ้น แต่ยัง

จำเป็นต้องปรับปรุงเพิ่มเติม เช่น การใช้เทคนิค Oversampling หรือการปรับค่า Threshold ของการจำแนกผลลัพธ์ เพื่อเพิ่มความแม่นยำในการแยกคลาส Major จากคลาสอื่น

4.3.2 ผลทดลองด้วยการเสริมข้อมูลแบบ EDA

การเสริมข้อมูลประเภทข้อความด้วยเทคนิค Synonym Replacement (SR) ซึ่งเป็นวิธีการแทนที่คำด้วยคำพ้องความหมาย ถูกนำมาใช้เป็นแนวทางพื้นฐานเพื่อเปรียบเทียบกับวิธีการเสริมข้อมูลที่พัฒนาโดยใช้โมเดล T5 ในงานวิจัยนี้ กระบวนการเสริมข้อมูลทั้งสองวิธีถูกดำเนินการภายใต้อัตราส่วนเดียวกันเพื่อให้สามารถเปรียบเทียบประสิทธิภาพของแต่ละแนวทางได้อย่างเป็นระบบ โดยเทคนิค SR อาศัยแหล่งข้อมูลจาก WordNet ในการแทนที่คำพ้องความหมาย ขณะที่โมเดล T5 ใช้โครงสร้างภาษาขั้นสูงในการสร้างข้อความที่มีความหมายใกล้เคียงกัน ทั้งนี้ ผลลัพธ์ของการทดลองจากการใช้วิธีการเสริมข้อมูลด้วยวิธีการแทนที่คำด้วยคำพ้องความหมายของแต่ละโมเดลแสดงดังตาราง

ตารางที่ 4.6 ผลการจำแนกระดับความรุนแรงด้วยการเสริมข้อมูลแบบ EDA

Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
Synonym Replacement Blocker 3 เท่า						
SVM	62.67	58.65	56.32	58.65	0.42	41.92
LR	59.76	59.02	57.25	59.02	0.43	11.44
RF	56.76	52.73	48.44	52.73	0.34	9.71
SACKING	59.77	60.96	59.76	60.96	0.46	14219.86
LSTM	47.62	48.75	47.25	48.75	0.29	787.84
BERT1	61.57±0.84	62.50±0.70	61.62±0.64	62.50±0.70	0.48±0.009	168.28±3.70
BERT2	62.02±0.30	63.64±0.24	62.64±0.34	63.64±0.24	0.50±0.004	163.31±4.64
Synonym Replacement Blocker 4 เท่า Trivial 1 เท่า						
SVM	63.00	60.04%	58.55	60.04	0.44	40.11
LR	61.09	59.67%	58.60	59.67	0.44	9.42
RF	58.01	53.75%	50.13	53.75	0.36	11.67
SACKING	61.67	59.85	58.80	59.85	0.59	13839.53
LSTM	45.13	49.58	46.63	49.58	0.31	885.32

Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
BERT1	63.93±0.48	62.96±0.52	62.19±1.02	62.96±0.52	0.49±0.01	180.89±0.22
BERT2	65.10±0.38	63.92±0.24	63.20±0.23	63.92±0.24	0.50±0.003	174.73±0.82

จากผลการทดลองในตารางที่ 4.6 แสดงให้เห็นว่า การใช้เทคนิค Synonym Replacement [69] ซึ่งเป็นหนึ่งในเทคนิคของ Easy Data Augmentation มีผลต่อประสิทธิภาพของโมเดลการจำแนกระดับความรุนแรงของข้อบกพร่อง โดยเฉพาะเมื่อเปรียบเทียบอัตราส่วนการเสริมข้อมูลในสองรูปแบบ ได้แก่ Blocker 3 เท่า และ Blocker 4 เท่า + Trivial 1 เท่า ผลลัพธ์แสดงให้เห็นว่า การเพิ่มข้อมูลแบบ Blocker 4 เท่า + Trivial 1 เท่า ส่งผลให้ค่า F1-score และ Accuracy ของหลายโมเดลดีขึ้นเมื่อเทียบกับการเพิ่มข้อมูลแบบ Blocker 3 เท่า

ในกรณีของโมเดลการเรียนรู้ของเครื่อง (SVM, LR, RF, Stacking) พบว่า Stacking Model มีค่า F1-score และ Accuracy สูงสุด ในกลุ่มโมเดลที่ไม่ใช้โครงข่ายประสาทเทียม โดยให้ค่า Accuracy 60.96% และ 59.85% สำหรับการเสริมข้อมูลแบบ Blocker 3 เท่า และ Blocker 4 เท่า + Trivial 1 เท่าตามลำดับ ทั้งนี้ ค่า F1-score ที่สูงขึ้นสะท้อนให้เห็นถึงความสามารถของโมเดลในการจัดการกับปัญหาความไม่สมดุลของข้อมูล อย่างไรก็ตาม แม้ว่า Random Forest (RF) จะเป็นหนึ่งในโมเดลที่ใช้กันแพร่หลาย แต่พบว่ามีประสิทธิภาพต่ำกว่าโมเดลอื่น โดยให้ค่า F1-score ต่ำที่สุดในทุกกรณี สะท้อนให้เห็นว่าเทคนิค SR อาจไม่เหมาะสมกับโครงสร้างของโมเดล RF ในบริบทของปัญหานี้

สำหรับโมเดลที่ใช้โครงข่ายประสาทเทียม ได้แก่ LSTM และ BERT พบว่า BERT2 ให้ผลลัพธ์ที่ดีที่สุด โดยให้ค่า Accuracy สูงสุดที่ 63.92% และค่า F1-score 63.2% ซึ่งดีกว่า BERT1 และโมเดลการเรียนรู้ของเครื่องทั้งหมด แสดงให้เห็นว่า โมเดล BERT มีความสามารถในการเข้าใจบริบทของข้อมูลที่ถูกเสริมด้วยคำพ้องความหมายได้ดีกว่า อย่างไรก็ตาม LSTM ซึ่งเป็นโครงข่ายประสาทเทียมเชิงลำดับกลับให้ผลลัพธ์ที่ต่ำกว่ามาก โดยให้ Accuracy อยู่ที่ 48.75% และ 49.58% ตามลำดับ ซึ่งอาจเกิดจากการที่ LSTM ไม่สามารถเรียนรู้บริบทของคำที่ถูกแทนที่ได้อย่างมีประสิทธิภาพเหมือนกับ BERT

เมื่อพิจารณา ระยะเวลาในการฝึกโมเดล พบว่า Stacking Model มีระยะเวลาการฝึกที่สูงที่สุดในกลุ่มของโมเดลการเรียนรู้ของเครื่อง โดยใช้เวลาประมาณ 14,219.86 วินาที และ 13,839.53 วินาที ในแต่ละกรณี ซึ่งสูงกว่ารุ่นอื่นอย่างมีนัยสำคัญ ในขณะที่ BERT1 และ BERT2 ซึ่งใช้ GPU A100 ใช้เวลาการฝึกประมาณ 180.89 วินาที และ 174.73 วินาที ตามลำดับ ซึ่งถือว่าเร็วกว่า Stacking Model อย่างมากเมื่อพิจารณาความสามารถในการจำแนกที่สูงกว่า

โดยสรุป การใช้ Synonym Replacement ภายใต้เทคนิค EDA สามารถช่วยปรับปรุงประสิทธิภาพของโมเดลได้ โดยเฉพาะในกรณีที่ใช้ BERT2 ซึ่งให้ผลลัพธ์ที่ดีที่สุด ทั้งนี้ การเสริมข้อมูลแบบ Blocker 4 เท่า + Trivial 1 เท่า มีแนวโน้มให้ผลลัพธ์ที่ดีกว่า Blocker 3 เท่าในแทบทุกโมเดล อย่างไรก็ตาม แม้ว่า Stacking Model จะให้ผลลัพธ์ที่ค่อนข้างดีในกลุ่มโมเดลการเรียนรู้ของเครื่อง แต่ใช้เวลาในการประมวลผลที่สูงกว่าอย่างมาก ทำให้โมเดล BERT2 เป็นตัวเลือกที่มีความเหมาะสมมากกว่าเมื่อพิจารณาทั้งด้านประสิทธิภาพและเวลาในการฝึกโมเดล

4.4 ผลการจำแนกระดับความรุนแรงจากชุดข้อมูลเสริม

การเสริมข้อมูลของรายงานจุดบกพร่องในงานวิจัยนี้ดำเนินการโดยใช้แนวทางการสร้างข้อความใหม่ ซึ่งอาศัยการ สกัดคำสำคัญด้วยเทคนิค TF-IDF และใช้โมเดล T5 ในการสร้างข้อความเสริมเพิ่มเติม เพื่อเพิ่มความหลากหลายของข้อมูลและลดปัญหาความไม่สมดุลของคลาสข้อมูล ในกระบวนการทดลองนี้ ได้มีการเปรียบเทียบประสิทธิภาพของโมเดลต่าง ๆ โดยแบ่งออกเป็นสองกลุ่ม ได้แก่ โมเดลการเรียนรู้ของเครื่อง ซึ่งประกอบด้วย Support Vector Machine Logistic Regression Random Forest และ Stacking Model และ กลุ่มโครงข่ายประสาทเทียม ซึ่งได้แก่ LSTM และ BERT

กระบวนการเสริมข้อมูลดำเนินการภายใต้สองแนวทาง ได้แก่ Blocker 3 เท่า และ Blocker 4 เท่า และ Trivial 1 เท่า เพื่อศึกษาผลกระทบของอัตราส่วนที่แตกต่างกันต่อความสามารถของโมเดลในการจำแนกระดับความรุนแรงของจุดบกพร่อง ผลการทดลองในแต่ละแนวทางแสดงในตาราง โดยใช้ F1-score และ Accuracy เป็นตัวชี้วัดหลักในการเปรียบเทียบประสิทธิภาพของแต่ละโมเดล รวมถึงการวิเคราะห์ระยะเวลาในการฝึกโมเดล ซึ่งเป็นปัจจัยสำคัญสำหรับการนำไปใช้งานจริง ทั้งนี้ การศึกษานี้มีวัตถุประสงค์เพื่อประเมินความเหมาะสมของเทคนิคการเสริมข้อมูลที่พัฒนาขึ้นเมื่อเปรียบเทียบกับแนวทางการเสริมข้อมูลแบบดั้งเดิม เช่น การใช้ Synonym Replacement และ SMOTE

ตารางที่ 4.7 ผลการจำแนกระดับความรุนแรงจากชุดข้อมูลเสริม

Model	Precision (%)	Recall (%)	F1 (%)	Accuracy (%)	MCC	Time(sec)
Augmented by T5 Blocker 3 เท่า						
SVM	62.45	59.02	57.16	59.02	0.43	33.41
LR	60.49	59.20	58.08	59.20	0.43	12.15
RF	60.10	55.13	51.77	55.13	0.38	9.62
SACKING	62.14	61.61	61.05	61.61	0.47	15290.53
LSTM	50.89	51.06	50.56	51.06	0.33	207.41
BERT1	63.14±1.14	63.12±1.00	62.86±1.00	63.12±1.00	0.49±0.014	163.474±1.62
BERT2	63.31±0.48	63.50±0.45	63.28±0.42	63.50±0.45	0.50±0.006	161.414±4.90
Augmented by T5 Blocker 4 เท่า Trivial 1 เท่า						
SVM	62.31	60.31	59.27	60.31	0.45	23.05
LR	60.62	60.22	59.59	60.22	0.45	9.19
RF	58.11	56.24	54.26	56.24	0.40	10.75
SACKING	61.63	61.24	60.70	61.24	0.46	15432.94
LSTM	54.03	51.71	52.39	51.71	0.35	236.33
BERT1	63.27±0.46	62.96±0.22	62.24±0.67	62.96±0.22	0.49±0.003	179.56±2.51
BERT2	64.71±0.17	65.20±0.13	64.25±0.11	65.20±0.13	0.52±0.002	179.74±1.18

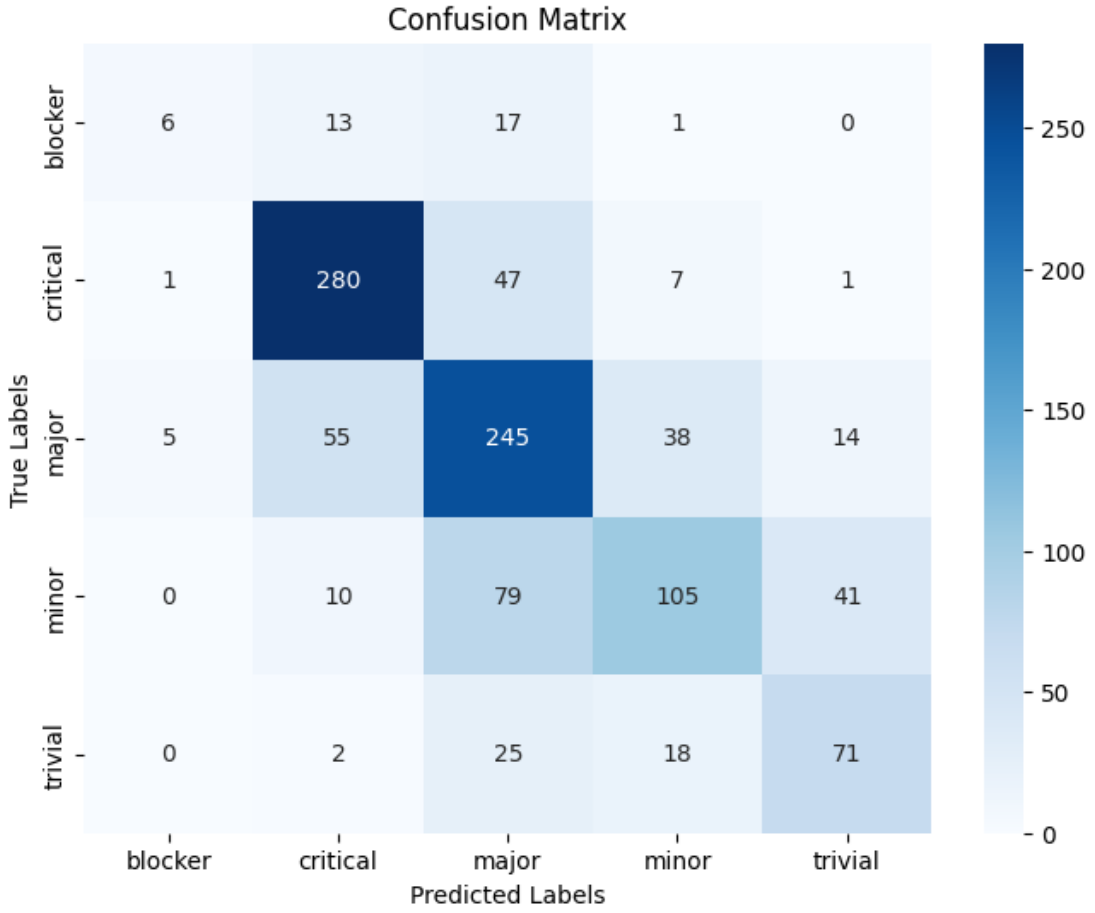
จากผลลัพธ์ที่นำเสนอในตารางที่ 4.7 แสดงให้เห็นว่า การเสริมข้อมูลโดยใช้ โมเดล T5 สามารถช่วยเพิ่มประสิทธิภาพของการจำแนกระดับความรุนแรงของจุดบกพร่องได้ โดยการเปรียบเทียบผลลัพธ์ระหว่างสองแนวทางการเสริมข้อมูล ได้แก่ Blocker 3 เท่า และ Blocker 4 เท่า และ Trivial 1 เท่า พบว่าแนวทางที่สองให้ค่า F1-score และ Accuracy สูงขึ้นในหลายโมเดล ซึ่งแสดงถึงความสามารถของแนวทางนี้ในการช่วยให้โมเดลสามารถเรียนรู้รูปแบบข้อมูลที่สมดุลและมีความหลากหลายมากขึ้น

สำหรับโมเดลการเรียนรู้ของเครื่อง (SVM, LR, RF, Stacking) พบว่า Stacking Model มีประสิทธิภาพสูงที่สุดในกลุ่มนี้ โดยมีค่า F1-score และ Accuracy อยู่ที่ 61.05% และ 61.61% สำหรับ Blocker 3 เท่า และ 60.7% และ 61.24% สำหรับ Blocker 4 เท่า + Trivial 1 เท่า อย่างไรก็ตาม เมื่อเปรียบเทียบกับการเสริมข้อมูลด้วย Synonym Replacement และ SMOTE ในการ

ทดลองก่อนหน้านี้ พบว่าผลลัพธ์ของ Stacking Model ในกรณีของ T5 มีแนวโน้มที่ดีกว่าเล็กน้อย ซึ่งแสดงให้เห็นว่า T5 สามารถสร้างข้อความที่ช่วยให้โมเดลเรียนรู้ความแตกต่างของคลาสได้ดีขึ้น นอกจากนี้ LR ยังคงให้ผลลัพธ์ที่ดีเช่นกัน โดยมี Accuracy ที่ 60.22% ในกรณี Blocker 4 เท่า และ Trivial 1 เท่า และในส่วนของโครงข่ายประสาทเทียม พบว่า BERT2 ให้ผลลัพธ์ที่ดีที่สุดในทุกโมเดล โดยมีค่า F1-score และ Accuracy สูงสุดที่ 64.25% และ 65.2% ตามลำดับ ซึ่งดีกว่าโมเดลอื่นทั้งหมด รวมถึงดีกว่าผลลัพธ์จาก Synonym Replacement และ SMOTE ที่ได้จากตารางก่อนหน้านี้ แสดงให้เห็นว่า BERT มีศักยภาพสูงสุดในการจำแนกระดับความรุนแรงของข้อบกพร่องเมื่อใช้ข้อมูลที่ถูกเสริมด้วย T5 ในขณะที่ LSTM แม้จะเป็นโครงข่ายประสาทเทียมเชิงลำดับ แต่ยังคงมีประสิทธิภาพต่ำกว่า โดยมี Accuracy ต่ำสุดที่ 51.06% และ 51.71% ตามลำดับ

เมื่อพิจารณา ระยะเวลาในการประมวลผล พบว่า Stacking Model ยังคงใช้เวลาในการฝึกสูงสุด โดยอยู่ที่ 15,290.53 วินาที และ 15,432.94 วินาที ตามลำดับ ขณะที่โมเดล BERT1 และ BERT2 ซึ่งใช้ GPU A100 ใช้เวลาการฝึกอยู่ที่ประมาณ 179.56 วินาที และ 179.74 วินาที ตามลำดับ ซึ่งแม้ว่าจะใช้เวลาในการฝึกสูงกว่าโมเดลการเรียนรู้ของเครื่องทั่วไป แต่เมื่อพิจารณาถึงประสิทธิภาพที่สูงกว่ามาก ทำให้ BERT2 เป็นตัวเลือกที่เหมาะสมสำหรับการนำไปใช้งานจริง

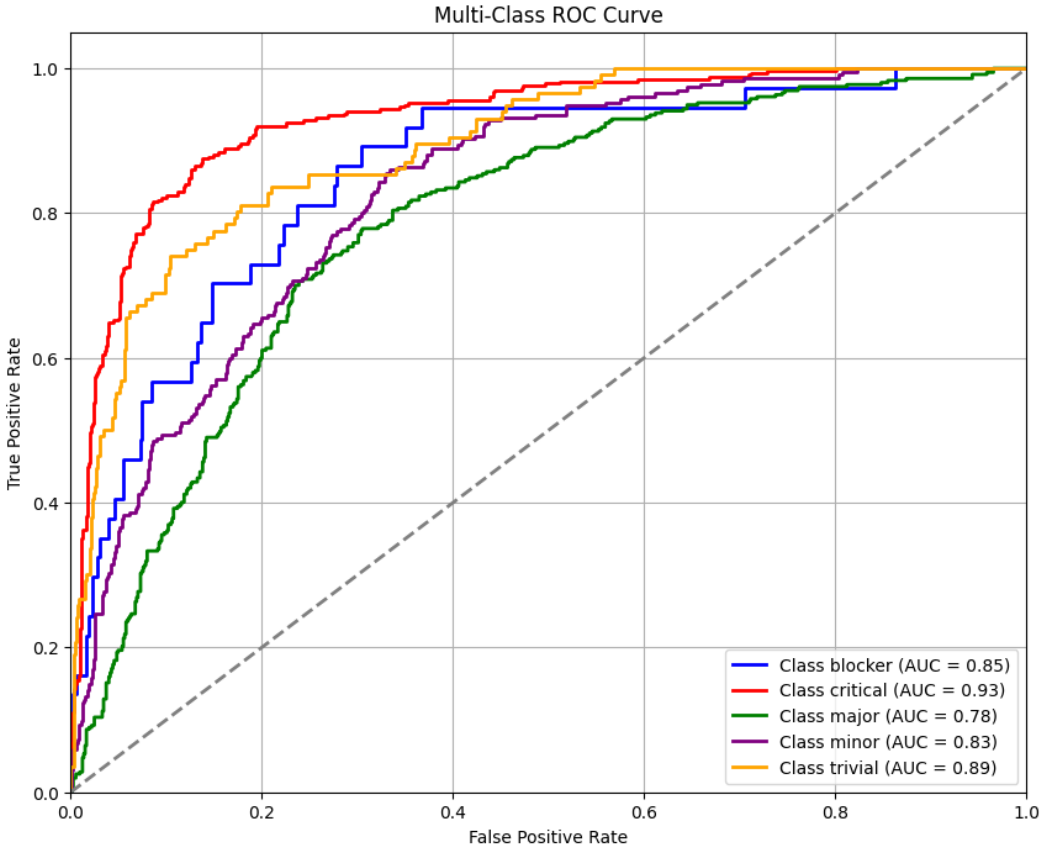
โดยสรุป ผลการทดลองแสดงให้เห็นว่า การเสริมข้อมูลด้วย T5 สามารถเพิ่มประสิทธิภาพของโมเดลได้ดีกว่าการใช้ Synonym Replacement และ SMOTE โดยเฉพาะในกรณีที่ใช้ Blocker 4 เท่า + Trivial 1 เท่า ซึ่งให้ค่า F1-score และ Accuracy สูงกว่าแนวทางอื่นในแทบทุกโมเดล ทั้งนี้ BERT2 เป็นโมเดลที่มีประสิทธิภาพดีที่สุด โดยสามารถจำแนกระดับความรุนแรงของข้อบกพร่องได้อย่างแม่นยำที่สุด ขณะที่ Stacking Model ยังคงเป็นตัวเลือกที่ดีที่สุดในกลุ่มของโมเดลการเรียนรู้ของเครื่อง แต่ต้องแลกมาด้วยเวลาการประมวลผลที่สูงกว่าอย่างมาก



รูปที่ 4.6 Confusion Matrix ของโมเดล BERT2 จากชุดข้อมูลเสริม

จาก Confusion Matrix แสดงผลการจำแนกระดับความรุนแรงของจุดบกพร่องโดยใช้โมเดล BERT2 กับชุดข้อมูลเสริม พบว่าโมเดลสามารถจำแนก Critical ได้ดีที่สุด โดยมี 280 ตัวอย่างที่ถูกต้อง และมีข้อผิดพลาดที่พยากรณ์เป็น Major (47 ตัวอย่าง) และ Minor (7 ตัวอย่าง) ซึ่งสะท้อนว่าโมเดลสามารถแยก Critical ออกจากคลาสอื่นได้ดีขึ้นเมื่อใช้ชุดข้อมูลเสริม

สำหรับ Major แม้จะมีการพยากรณ์ถูกต้อง 245 ตัวอย่าง แต่ยังมีข้อผิดพลาดที่พยากรณ์ไปเป็น Critical (55 ตัวอย่าง) และ Minor (38 ตัวอย่าง) แสดงให้เห็นว่ายังคงมีความสับสนระหว่างคลาสที่มีความใกล้เคียงกัน อย่างไรก็ตาม เมื่อเปรียบเทียบกับผลลัพธ์ก่อนหน้านี้ โมเดลสามารถจำแนกคลาสนี้ได้ดีขึ้นเล็กน้อย นอกจากนี้ Minor และ Trivial มีแนวโน้มที่โมเดลจะพยากรณ์ผิดไปเป็น Major และ Critical มากกว่าคลาสอื่น โดยสรุปการใช้ชุดข้อมูลเสริมช่วยให้โมเดลสามารถจำแนก Critical และ Major ได้ดีขึ้น แต่ยังคงมีข้อผิดพลาดในคลาส Minor และ Trivial



รูปที่ 4.1 ROC Curve ของโมเดล BERT2 จากชุดข้อมูลเสริม

4.5 สรุปผลการจำแนกระดับความรุนแรง

4.5.1 สรุปผลทดลองจากชุดข้อมูลที่ 1

ในส่วนนี้เป็นการสรุปผลการจำแนกระดับความรุนแรงจาก ชุดข้อมูลที่ 1 ซึ่งเป็น ชุดข้อมูลรายงานจุดบกพร่องของ Bugzilla โดยชุดข้อมูลดังกล่าวมีปัญหาความไม่สมดุลของคลาส โดยเฉพาะคลาส Blocker และ Trivial ซึ่งมีปริมาณข้อมูลน้อยกว่าคลาสอื่นอย่างมีนัยสำคัญ ส่งผลต่อประสิทธิภาพของโมเดลในการจำแนกระดับความรุนแรงของข้อบกพร่อง การทดลองที่ดำเนินการในขั้นตอนก่อนหน้าได้ทดสอบวิธีการเสริมข้อมูลหลายแนวทางเพื่อแก้ไขปัญหานี้

จากผลการทดลองโดยภาพรวม พบว่าการใช้กลยุทธ์การเสริมข้อมูลแบบ Blocker 4 เท่า และ Trivial 1 เท่า ให้ผลลัพธ์ที่ดีในทุกวิธีการเสริมข้อมูล ไม่ว่าจะเป็น เทคนิค SMOTE การแทนที่คำพ้องความหมาย และแนวทางที่นำเสนอในงานวิจัยนี้โดยใช้โมเดล T5 โดยเฉพาะอย่างยิ่ง การเสริมข้อมูลด้วยโมเดล T5 แสดงให้เห็นถึงประสิทธิภาพสูงสุดในการช่วยปรับปรุงความสามารถของโมเดลจำแนกข้อความ

ดังนั้น สามารถสรุปผลการทดลองได้โดยใช้ตัวชี้วัดสำคัญ ได้แก่ F1-score, Accuracy, MCC และ ระยะเวลาในการฝึกโมเดล ซึ่งเปรียบเทียบผลลัพธ์ของแต่ละโมเดลภายใต้กลยุทธ์การเสริมข้อมูลที่แตกต่างกัน รายละเอียดของผลลัพธ์ถูกนำเสนอในตารางด้านล่าง

ตารางที่ 4.8 สรุปผลการจำแนกระดับความรุนแรงจากชุดข้อมูลที่ 1

TECHNIQUES	MODEL	F1 (%)	ACCURACY (%)	MCC	TIME (SEC)
Original data	SVM	56.09	58.46	0.42	19.75
	LR	57.10	58.93	0.43	9.04
	RF	52.56	56.80	0.40	8.83
	Sacking	60.05	60.68	0.45	12956.80
	LSTM	47.46	47.46	0.29	199.81
	BERT1	61.80±0.56	62.70±0.35	0.49±0.005	150.91±3.23
	BERT2	61.77±0.62	63.16±0.52	0.49±0.007	149.62±2.77
Synonym replacement (Blocker 4, Trivial 1)	SVM	58.55	60.04	0.44	40.11
	LR	58.60	59.67	0.44	9.42
	RF	50.13	53.75	0.36	11.67
	Sacking	58.80	59.85	0.59	13839.53
	LSTM	46.63	49.58	0.31	885.32
	BERT1	62.19±1.02	62.96±0.52	0.49±0.01	180.89±0.22
	BERT2	63.20±0.23	63.92±0.24	0.50±0.003	174.73±0.82
Augmented By T5 (Blocker 4, Trivial 1)	SVM	59.27	60.31	0.45	23.05
	LR	59.59	60.22	0.45	9.19
	RF	54.26	56.24	0.40	10.75
	Sacking	60.70	61.24	0.46	15432.94
	LSTM	52.39	51.71	0.35	236.33
	BERT1	62.24±0.67	62.96±0.22	0.49±0.003	179.56±2.51
	BERT2	64.25±0.11	65.20±0.13	0.52±0.002	179.74±1.18

จากผลการทดลองในตารางที่ 4.8 พบว่าการเสริมข้อมูลสามารถช่วยเพิ่มประสิทธิภาพของการจำแนกระดับความรุนแรงของข้อบกพร่องได้ โดยเมื่อเปรียบเทียบผลลัพธ์ระหว่างชุดข้อมูลดั้งเดิม (Original Data) และข้อมูลที่ได้รับการเสริมด้วย Synonym Replacement (Blocker 4 เท่า,

Trivial 1 เท่า) และ โมเดล T5 (Blocker 4 เท่า, Trivial 1 เท่า) พบว่าแนวทางการเสริมข้อมูลช่วยเพิ่มค่า F1-score และ Accuracy ในทุกโมเดล โดยเฉพาะการใช้โมเดล T5 ซึ่งให้ผลลัพธ์ที่ดีที่สุด

สำหรับโมเดลการเรียนรู้ของเครื่อง (SVM, LR, RF, Stacking) พบว่า Stacking Model มีค่า F1-score และ Accuracy สูงสุด ในกลุ่มนี้ โดยมีค่า F1-score 60.7% และ Accuracy 61.24% เมื่อใช้ข้อมูลที่เสริมด้วย T5 ซึ่งสูงกว่า Synonym Replacement และข้อมูลดั้งเดิม อย่างไรก็ตาม Random Forest (RF) มีประสิทธิภาพต่ำที่สุดในทุกกรณี โดยให้ค่า F1-score ต่ำสุดที่ 50.13% ในกรณี Synonym Replacement และ 54.26% เมื่อใช้ T5 แสดงให้เห็นว่า RF อาจไม่ได้รับประโยชน์มากจากการเสริมข้อมูลเท่ากับโมเดลอื่น ในส่วนของโครงข่ายประสาทเทียม พบว่า BERT2 ให้ผลลัพธ์ที่ดีที่สุด โดยมีค่า F1-score สูงสุดที่ 64.25% และ Accuracy สูงสุดที่ 65.2% ซึ่งสูงกว่าทุกโมเดลในการทดลอง และดีกว่าการใช้ Synonym Replacement แสดงให้เห็นว่า โมเดล BERT มีความสามารถในการเข้าใจและจัดการกับข้อมูลที่ถูกเสริมโดย T5 ได้ดีที่สุด ขณะที่ LSTM แม้ว่าจะเป็นโครงข่ายประสาทเทียมเชิงลำดับ แต่กลับมีประสิทธิภาพต่ำกว่า โดยให้ค่า Accuracy ต่ำสุดที่ 47.46% บนข้อมูลดั้งเดิม และ 51.71% เมื่อใช้ T5 ซึ่งอาจบ่งชี้ว่า LSTM ยังไม่สามารถเรียนรู้บริบทของข้อความที่ถูกเสริมได้ดีเท่ากับ BERT เมื่อตรวจสอบ เวลาในการประมวลผล พบว่า Stacking Model ใช้เวลาในการฝึกที่สูงที่สุด โดยเฉพาะในกรณีที่ใช้ T5 ซึ่งใช้เวลาถึง 15,432.94 วินาที ในขณะที่ BERT1 และ BERT2 ซึ่งใช้ GPU A100 ใช้เวลาฝึกอยู่ที่ประมาณ 179.56 วินาที และ 179.74 วินาที ตามลำดับ ซึ่งแม้ว่าจะใช้เวลาฝึกสูงกว่าโมเดลการเรียนรู้ของเครื่องทั่วไป แต่เมื่อพิจารณาถึงประสิทธิภาพที่เหนือกว่าอย่างมาก ทำให้ BERT2 เป็นตัวเลือกที่มีความเหมาะสมสูงสุด

โดยสรุป ผลการทดลองแสดงให้เห็นว่า การเสริมข้อมูลด้วย T5 มีประสิทธิภาพเหนือกว่าวิธี Synonym Replacement และ SMOTE โดยเฉพาะอย่างยิ่งในกรณีของ BERT2 ซึ่งให้ผลลัพธ์ที่ดีที่สุด ทั้งในด้าน F1-score และ Accuracy ขณะที่ Stacking Model เป็นทางเลือกที่ดีสำหรับโมเดลการเรียนรู้ของเครื่อง แต่ต้องแลกมาด้วยเวลาในการประมวลผลที่สูงกว่ามาก ทั้งนี้ การเลือกใช้แนวทางการเสริมข้อมูลควรพิจารณาควบคู่กับทรัพยากรที่ใช้ในการประมวลผลและความต้องการด้านความแม่นยำของโมเดล

4.5.2 สรุปผลทดลองจากชุดข้อมูลที่ 2

นอกจากนี้ ผู้วิจัยได้ดำเนินการประเมินประสิทธิภาพของโมเดล BERT1 และ BERT2 บนชุดข้อมูลที่ 2 ซึ่งเป็นชุดข้อมูลที่เกี่ยวข้องกับการศึกษารายงานจุดบกพร่องของซอฟต์แวร์จากแหล่งข้อมูลที่หลากหลาย โดยชุดข้อมูลนี้มีความสมดุลของจำนวนตัวอย่างในแต่ละคลาส ส่งผลให้ ไม่มีความจำเป็นต้องใช้กระบวนการเสริมข้อมูล ในการทดลองครั้งนี้

เพื่อให้สามารถวิเคราะห์ประสิทธิภาพของโมเดลได้อย่างเป็นระบบ การทดลองเปรียบเทียบ BERT1 และ BERT2 กับผลลัพธ์จากงานวิจัยอื่น ๆ ที่ใช้แนวทางที่แตกต่างกันสำหรับการจำแนกระดับความรุนแรงของข้อบกพร่อง ทั้งนี้ การเปรียบเทียบดังกล่าวช่วยให้สามารถประเมิน จุดแข็งและข้อจำกัด ของโมเดลที่พัฒนาในงานวิจัยนี้ได้อย่างชัดเจน ผลลัพธ์ของการเปรียบเทียบประสิทธิภาพของโมเดล BERT1 และ BERT2 กับงานวิจัยอื่น ๆ ถูกนำเสนอในตารางด้านล่าง เพื่อให้เห็นถึงประสิทธิภาพของโมเดลที่ศึกษาในบริบทของชุดข้อมูลที่มีความสมดุล

ตารางที่ 4.9 ผลการทดลองโมเดล BERT1 และ BERT2 กับชุดข้อมูลการศึกษาอื่น

MODELS	PRECISION (%)	RECALL (%)	F1 (%)	ACCURACY (%)
BERT1	62.82	62.27	62.27	62.48
BERT2	64.93	63.63	63.67	63.63
RTA [66]	65.11	64.72	64.73	64.72
CLBSP [64]	61.34	62.77	61.88	62.77
PPWGCN [17]	55.54	55.24	55.22	55.24
DNNSPBP [14]	55.54	54.18	54.12	54.18
BSP-QASO [62]	54.21	53.47	52.42	53.47

จากผลลัพธ์ในตารางที่ 4.9 พบว่าโมเดล BERT1 และ BERT2 มีประสิทธิภาพในการจำแนกระดับความรุนแรงของข้อบกพร่องในชุดข้อมูลที่มีความสมดุล โดย BERT2 ให้ค่า F1-score สูงสุดที่ 63.67% และ Accuracy 62.48% ซึ่งสูงกว่า BERT1 เล็กน้อยที่มีค่า F1-score 62.27% และเช่นเดียวกับ Accuracy ของ BERT2 สูงกว่า BERT 1 ที่ 63.63% และ 62.48% ตามลำดับ ผลลัพธ์ดังกล่าวแสดงให้เห็นว่า BERT2 มีความสามารถในการเรียนรู้บริบทของข้อความได้ดีกว่า BERT1 อย่างไรก็ตาม ทั้งสองโมเดลยังคงมีค่า F1-score ต่ำกว่าบางโมเดลจากงานวิจัยอื่น เช่น RTA ที่ให้ค่า F1-score สูงสุดที่ 64.73% และ Accuracy ที่ 64.72%

เมื่อเปรียบเทียบกับงานวิจัยที่ใช้โมเดลอื่น เช่น CLBSP PPWGCN DNNSPBP และ BSP-QASO พบว่า BERT1 และ BERT2 มีความสามารถเหนือกว่าอย่างชัดเจน โดยค่า F1-score ของ BERT2 สูงกว่า CLBSP ที่ได้ 61.88% และสูงกว่ารุ่นอื่น ๆ ซึ่งมีค่า F1-score ต่ำกว่า 55.22% แสดงให้เห็นว่า BERT-based models มีความสามารถในการเรียนรู้ข้อมูลที่ซับซ้อนกว่าโมเดลที่อิงกับโครงข่ายประสาทเทียมแบบดั้งเดิม อย่างไรก็ตาม แม้ว่าโมเดล RTA จะให้ค่า Accuracy และ F1-score สูงกว่า BERT2 เล็กน้อย แต่อาจต้องพิจารณาปัจจัยอื่น ๆ เช่น ความซับซ้อนของโมเดลและระยะเวลาในการฝึก

นอกจากนี้ ผลการทดลองยังแสดงให้เห็นถึงความแตกต่างของแนวทางการพัฒนาโมเดลในงานวิจัยต่าง ๆ โดยโมเดลที่ใช้ โครงข่ายประสาทเทียมเชิงลึกแบบดั้งเดิม (DNNSPBP และ BSP-QASO) มีค่า Accuracy ต่ำกว่ามาก ซึ่งอาจสะท้อนให้เห็นถึงข้อจำกัดของโมเดลเหล่านี้ในการเรียนรู้ข้อมูลที่มีความซับซ้อน ในขณะที่โมเดลที่ใช้ Graph Convolutional Networks (PPWGCN) มีประสิทธิภาพสูงกว่าโมเดลโครงข่ายประสาทเทียมแบบดั้งเดิม แต่ยังคงต่ำกว่า BERT-based models

โดยสรุป ผลการทดลองแสดงให้เห็นว่า BERT2 มีประสิทธิภาพสูงกว่าหลายโมเดลจากงานวิจัยอื่น ๆ และมีความสามารถในการจำแนกความรุนแรงของข้อบกพร่องได้ดี อย่างไรก็ตาม โมเดล RTA ยังคงให้ผลลัพธ์ที่ดีที่สุดในชุดข้อมูลนี้ ซึ่งบ่งชี้ว่าการพัฒนาแนวทางการประมวลผลเพิ่มเติม หรือการนำเทคนิคการเสริมข้อมูลมาใช้ อาจช่วยให้ BERT-based models สามารถปรับปรุงประสิทธิภาพได้มากขึ้น

4.5.3 สรุปผลทดลองจากชุดข้อมูลที่ 3

ชุดข้อมูลรายงานจุดบกพร่อง ชุดที่ 3 ซึ่งได้มาจากโครงการ Apache ถูกนำมาใช้ในการทดลองเพื่อประเมินประสิทธิภาพของแนวทางการเสริมข้อมูลที่พัฒนาในงานวิจัยนี้ โดยมุ่งเน้นไปที่การใช้ โมเดล T5 สำหรับสร้างข้อความเสริมข้อมูล ทั้งนี้ ชุดข้อมูลดังกล่าวมีปัญหาความไม่สมดุลของคลาสอย่างรุนแรง โดยมีจำนวนแถวของข้อมูลแต่ละคลาส ดังนี้ blocker จำนวน 388 รายการ critical จำนวน 529 รายการ major จำนวน 843 รายการ minor จำนวน 466 รายการ และ trivial จำนวน 37 รายการ เพื่อให้สามารถประเมินผลลัพธ์ของแนวทางที่นำเสนอได้อย่างเป็นระบบ

สำหรับการทดลองนี้ได้ทำการแบ่งชุดข้อมูลสำหรับฝึกออกเป็น 80% และสำหรับทดสอบจำนวน 20% เช่นเดียวกับชุดข้อมูลที่ 1 จากนั้นนำชุดข้อมูลสำหรับฝึกไปทำการเสริมข้อมูลด้วยโมเดล T5 และการทำ SMOTE เพื่อเปรียบเทียบ โดยการเพิ่มจำนวนของคลาส Trivial เพิ่มขึ้นจำนวน 3 เท่า ซึ่งตารางด้านล่างแสดงผลลัพธ์ของการจำแนกโดยใช้โมเดลที่ผ่านการฝึกด้วยชุดข้อมูลที่ได้รับการเสริมข้อมูลผ่านโมเดล T5 เปรียบเทียบกับวิธีอื่น ๆ โดยพิจารณาตัวชี้วัดหลัก ได้แก่ F1-score, Accuracy และ MCC ซึ่งใช้วัดความแม่นยำในการจำแนกระดับความรุนแรงของจุดบกพร่องภายใต้ชุดข้อมูลที่มีความไม่สมดุลสูง

ตารางที่ 4.10 สรุปผลการจำแนกระดับความรุนแรงจากชุดข้อมูลที่ 3

TECHNIQUES	MODELS	F1 (%)	ACCURACY (%)	MCC	TIME (SEC)
Original data	SVM	27.73	39.07	0.14	5.52
	LR	36.82	41.72	0.17	8.37
	RF	33.30	41.28	0.17	3.93
	Sacking	40.90	45.47	0.23	3049.07
SMOTE (TRIVIAL 3)	SVM	27.73	39.07	0.14	6.02
	LR	36.36	41.50	0.16	6.45
	RF	32.24	40.40	0.15	4.01
	Sacking	39.06	43.93	0.21	3162.18
Augmented By T5 (Trivial 3)	SVM	27.97	39.07	0.14	5.62
	LR	36.62	41.72	0.17	7.77
	RF	36.97	44.15	0.23	4.21
	Sacking	39.95	44.81	0.23	3069.93

จากผลลัพธ์ในตารางที่ 4.10 พบว่าชุดข้อมูลรายงานจุดบกพร่องของ Apache (ชุดข้อมูลที่ 3) มีความไม่สมดุลของคลาสสูง ส่งผลให้ประสิทธิภาพของโมเดลจำแนกความรุนแรงลดลงอย่างมีนัยสำคัญเมื่อใช้ชุดข้อมูลดั้งเดิม (Original Data) โดยเฉพาะในกรณีของ SVM ซึ่งให้ค่า F1-score ต่ำสุดที่ 27.73% และ Accuracy เพียง 39.07% อย่างไรก็ตาม Logistic Regression (LR) และ Random Forest (RF) มีประสิทธิภาพที่ดีกว่าเล็กน้อย โดยให้ค่า F1-score อยู่ที่ 36.82% และ 33.3% ตามลำดับ และมีค่า Accuracy ประมาณ 41% ทั้งนี้ Stacking Model ให้ผลลัพธ์ที่ดีที่สุดในชุดข้อมูลดั้งเดิม โดยมี F1-score 40.9% และ Accuracy 45.47% แสดงให้เห็นว่าเทคนิคการรวมโมเดลสามารถช่วยเพิ่มความแม่นยำในข้อมูลที่มีความไม่สมดุลได้บางส่วน เมื่อเปรียบเทียบการใช้ SMOTE (Trivial 3) กับข้อมูลดั้งเดิม พบว่าการเสริมข้อมูลด้วย SMOTE ไม่ได้ช่วยเพิ่มประสิทธิภาพของโมเดลอย่างมีนัยสำคัญ ค่า F1-score และ Accuracy ของ SVM และ LR มีการเปลี่ยนแปลงเพียงเล็กน้อย โดยเฉพาะ SVM ที่มีผลลัพธ์เหมือนเดิม ขณะที่ Stacking Model ที่ใช้ SMOTE ให้ค่า Accuracy ลดลงจาก 45.47% เหลือ 43.93% ซึ่งบ่งชี้ว่า SMOTE อาจไม่สามารถสร้างข้อมูลใหม่ที่มีลักษณะเหมือนคลาสที่มีจำนวนน้อยได้อย่างมีประสิทธิภาพในกรณีนี้

ในทางกลับกัน การเสริมข้อมูลโดยใช้ โมเดล T5 (Trivial 3) แสดงให้เห็นถึงประสิทธิภาพที่สูงขึ้นอย่างชัดเจน โดยเฉพาะในกรณีของ Random Forest (RF) ที่มีค่า F1-score เพิ่มขึ้นจาก

32.24% (SMOTE) เป็น 36.97% และ Accuracy เพิ่มขึ้นเป็น 44.15% เช่นเดียวกับ Stacking Model ที่มีค่า Accuracy สูงขึ้นจาก 43.93% (SMOTE) เป็น 44.81% นอกจากนี้ ค่า Matthews Correlation Coefficient (MCC) ซึ่งใช้วัดความสัมพันธ์ของผลลัพธ์การจำแนก พบว่า โมเดลที่ใช้ T5 มีค่า MCC สูงสุดที่ 0.2316 ใน Stacking Model ซึ่งแสดงให้เห็นว่า T5 สามารถสร้างข้อมูลที่ช่วยปรับปรุงความสามารถของโมเดลในการเรียนรู้บริบทของข้อบกพร่องได้ดีกว่า SMOTE และส่วนในด้านของระยะเวลาในการประมวลผล พบว่า Stacking Model มีเวลาการฝึกที่ยาวนานที่สุดในทุกกรณี โดยใช้เวลาประมาณ 3,069.93 วินาที เมื่อใช้ข้อมูลที่เสริมด้วย T5 ซึ่งสูงกว่าข้อมูลดั้งเดิม (3,049.07 วินาที) และ SMOTE (3,162.18 วินาที) ขณะที่ BERT ไม่ถูกนำมาใช้ในการทดลองนี้ อย่างไรก็ตาม เมื่อพิจารณาประสิทธิภาพที่สูงขึ้นของ T5 และผลกระทบเชิงบวกต่อ Accuracy และ F1-score ทำให้แนวทางนี้เป็นทางเลือกที่มีศักยภาพในการจัดการกับข้อมูลที่มีความไม่สมดุลสูง

โดยสรุป ผลการทดลองจากชุดข้อมูลที่ 3 แสดงให้เห็นว่า การเสริมข้อมูลด้วยโมเดล T5 มีประสิทธิภาพเหนือกว่า SMOTE ในการแก้ปัญหาความไม่สมดุลของคลาส โดยเฉพาะอย่างยิ่งในโมเดล Random Forest และ Stacking Model ที่มีค่า Accuracy และ F1-score สูงขึ้นอย่างชัดเจน ในขณะที่ SMOTE ไม่สามารถเพิ่มประสิทธิภาพของโมเดลได้อย่างมีนัยสำคัญ ผลการทดลองนี้สอดคล้องกับสมมุติฐานของงานวิจัยที่ว่า T5 สามารถสร้างข้อความที่มีคุณภาพสูงและช่วยให้โมเดลเรียนรู้ความแตกต่างของคลาสที่มีจำนวนน้อยได้ดีขึ้น

บทที่ 5

สรุป อภิปรายและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

การศึกษานี้นำเสนอแนวทางการจำแนกระดับความรุนแรงของจุดบกพร่องซอฟต์แวร์แบบหลายคลาสโดยใช้เทคนิคการเรียนรู้ของเครื่องและการเรียนรู้เชิงลึก เพื่อพัฒนาโมเดลที่สามารถช่วยให้การจัดลำดับความสำคัญของการแก้ไขจุดบกพร่องมีประสิทธิภาพมากขึ้น โดยมีการทดสอบกับชุดข้อมูล Bugzilla และชุดข้อมูลจากโครงการ Mozilla, Eclipse, Netbeans และ GCC เพื่อให้ได้ผลลัพธ์ที่มีความน่าเชื่อถือ

ผลการวิจัยพบว่าโมเดล Support Vector Machine (SVM) มีความแม่นยำสูงสุดในบรรดาโมเดลเรียนรู้ของเครื่อง แต่ยังมีข้อจำกัดในการจัดการกับข้อมูลที่ไม่สมดุล โมเดล Random Forest สามารถจัดการกับข้อมูลที่มีความซับซ้อนได้ดี แต่ใช้เวลาประมวลผลสูง เมื่อใช้เทคนิคการเรียนรู้แบบรวม (Ensemble Learning) โดยเฉพาะ Stacking Ensemble ซึ่งผสมผสานโมเดล Naive Bayes, SVM, Random Forest และ XGBoost พบว่ามีประสิทธิภาพสูงกว่าโมเดลเดี่ยว โดยเฉพาะเมื่อใช้ Logistic Regression เป็นโมเดลระดับเมตา (Meta Model) สำหรับการใช้โมเดลทรานฟอร์เมอร์ พบว่า BERT2 ที่ใช้ T5 Summarization เพื่อเสริมข้อมูล ให้ค่า F1-score 64.25% และ Accuracy 65.20% ซึ่งสูงกว่าโมเดลอื่นๆ สำหรับในการจัดการปัญหาข้อมูลไม่สมดุล งานวิจัยนี้ได้นำเทคนิค SMOTE และ Class Weighting มาใช้กับโมเดลเรียนรู้ของเครื่อง และใช้ T5 Summarization ในโมเดล BERT ซึ่งช่วยให้การจำแนกแม่นยำขึ้นกว่าการใช้ SMOTE อย่างเดียว ผลการทดลองสรุปได้ว่าโมเดล BERT2 ที่ใช้ T5 Summarization ให้ผลลัพธ์ที่ดีที่สุด และสามารถช่วยให้การจัดการจุดบกพร่องมีความแม่นยำมากขึ้น นอกจากนี้ การใช้ Ensemble Learning (Stacking) ยังช่วยเพิ่มประสิทธิภาพของโมเดลเรียนรู้ของเครื่อง และการเสริมข้อมูลด้วย T5 Summarization มีประสิทธิภาพเหนือกว่าเทคนิคมาตรฐานเช่น SMOTE ในการจัดการกับข้อมูลที่ไม่สมดุล

ประโยชน์ของการวิจัยนี้คือสามารถช่วยให้ผู้พัฒนาลดระยะเวลาในการตรวจสอบและจัดลำดับความสำคัญของจุดบกพร่องได้อย่างแม่นยำ โมเดลที่พัฒนาขึ้นสามารถนำไปประยุกต์ใช้กับโครงการซอฟต์แวร์อื่นๆ ที่ใช้ระบบติดตามจุดบกพร่องที่คล้ายกัน อีกทั้งสามารถนำไปใช้เป็นแนวทางสำหรับการพัฒนาโมเดลการจำแนกเอกสารข้อความอื่นๆ เช่น การจำแนกรีวิวซอฟต์แวร์ หรือการวิเคราะห์ความคิดเห็นของผู้ใช้ จากผลการทดลองและข้อค้นพบ งานวิจัยนี้แสดงให้เห็นว่า การใช้โมเดล Transformer เช่น BERT และการเสริมข้อมูลด้วย T5 Summarization สามารถช่วยเพิ่ม

ประสิทธิภาพในการจำแนกระดับความรุนแรงของจุดบกพร่องได้ดีกว่าโมเดลทั่วไป ทำให้เป็นแนวทางที่มีศักยภาพในการนำไปใช้งานจริงในระบบติดตามจุดบกพร่องซอฟต์แวร์

5.2 อภิปรายผล

จากผลการวิจัยพบว่าโมเดล BERT ที่ได้รับการเสริมข้อมูลด้วย T5 Summarization สามารถเพิ่มประสิทธิภาพในการจำแนกระดับความรุนแรงของจุดบกพร่องได้ดีกว่าโมเดลที่ไม่ได้ใช้การเสริมข้อมูล เทคนิคนี้ช่วยให้ข้อมูลในคลาสที่มีจำนวนน้อยมีตัวอย่างเพิ่มขึ้น ส่งผลให้โมเดลสามารถเรียนรู้และจำแนกคลาสเหล่านี้ได้ดีขึ้น นอกจากนี้ การใช้ Ensemble Learning (Stacking) ยังแสดงให้เห็นว่าแนวทางนี้สามารถเพิ่มความแม่นยำของการจำแนกได้เมื่อเทียบกับโมเดลเดี่ยว โดยเฉพาะเมื่อใช้ Logistic Regression เป็น Meta Model ซึ่งช่วยลดความคลาดเคลื่อนและเพิ่มความสามารถของโมเดลในการสังเกตความแตกต่างของข้อมูลแต่ละคลาสได้ดียิ่งขึ้น

การใช้เทคนิค SMOTE และ Class Weighting ในโมเดลเรียนรู้ของเครื่องแบบดั้งเดิม เช่น SVM และ Random Forest สามารถช่วยลดปัญหาข้อมูลไม่สมดุลได้ในระดับหนึ่ง อย่างไรก็ตาม เทคนิคเหล่านี้ยังมีข้อจำกัด เช่น การเพิ่มข้อมูลสังเคราะห์อาจไม่ได้สะท้อนลักษณะที่แท้จริงของข้อมูล ทำให้โมเดลเรียนรู้ข้อมูลที่มีความซ้ำซ้อนหรือคล้ายกันเกินไป ในทางตรงกันข้าม การใช้ T5 Summarization สามารถสร้างข้อมูลใหม่ที่มีความเป็นธรรมชาติมากกว่า เนื่องจากใช้โมเดลภาษาที่สามารถสร้างข้อความที่มีความสอดคล้องกับข้อมูลต้นฉบับได้ดีขึ้น นอกจากนี้ การศึกษานี้ยังพบว่าโมเดล BERT มีความสามารถสูงในการเรียนรู้บริบทของข้อความ ซึ่งช่วยให้สามารถจำแนกข้อมูลได้อย่างมีประสิทธิภาพมากกว่าโมเดลเรียนรู้ของเครื่องแบบดั้งเดิม อย่างไรก็ตาม ข้อจำกัดของ BERT คือการใช้ทรัพยากรการประมวลผลที่สูงกว่ามาก และต้องอาศัยฮาร์ดแวร์ที่มีประสิทธิภาพสูง เช่น GPU ในการฝึกและทดสอบโมเดล ซึ่งอาจเป็นอุปสรรคสำหรับการใช้งานในบางองค์กรที่มีข้อจำกัดด้านทรัพยากร

การวิจัยนี้แสดงให้เห็นว่าเทคนิคการเสริมข้อมูลด้วย T5 Summarization และการใช้โมเดล Transformer อย่าง BERT สามารถเพิ่มประสิทธิภาพของการจำแนกระดับความรุนแรงของจุดบกพร่องได้อย่างมีนัยสำคัญ อย่างไรก็ตาม การเลือกใช้โมเดลและเทคนิคการเสริมข้อมูลควรพิจารณาตามบริบทของระบบที่ใช้งานจริง รวมถึงข้อจำกัดด้านทรัพยากรในการฝึกโมเดลเพื่อให้ได้ผลลัพธ์ที่เหมาะสมที่สุด

5.3 ข้อเสนอแนะ

งานวิจัยนี้แสดงให้เห็นถึงศักยภาพของโมเดล Transformer ในการจำแนกระดับความรุนแรงของจุดบกพร่องจากรายงานจุดบกพร่อง โดยเฉพาะการใช้โมเดล BERT ซึ่งมีประสิทธิภาพสูงเมื่อทำการฝึกด้วยข้อมูลที่เหมาะสม อย่างไรก็ตาม ยังมีประเด็นสำคัญที่สามารถพัฒนาเพิ่มเติมได้เพื่อเพิ่มประสิทธิภาพและความสามารถของโมเดลสำหรับการนำไปใช้งานในสภาพแวดล้อมจริง ดังนี้

5.3.1 การพัฒนากระบวนการเสริมข้อมูลที่สมจริง

ปัจจุบัน การเสริมข้อมูล (Data Augmentation) ที่ใช้ในงานวิจัยนี้ประกอบด้วยการใช้ T5 Summarization, การใช้ SMOTE สำหรับการเพิ่มข้อมูลคลาสที่มีจำนวนน้อย, และการใช้ Class Weight เพื่อช่วยลดปัญหาความไม่สมดุลของข้อมูล อย่างไรก็ตาม เทคนิคเหล่านี้ยังมีข้อจำกัด เช่น การสร้างข้อความที่อาจไม่สอดคล้องกับเนื้อหาของรายงานจุดบกพร่องเดิม หรืออาจเกิดปัญหาข้อมูลซ้ำกันมากเกินไป ดังนั้น แนวทางในการพัฒนาต่อยอด ได้แก่

การนำ Paraphrasing มาใช้เพื่อสร้างรายงานจุดบกพร่องที่มีความหมายใกล้เคียงกับต้นฉบับ แต่มีโครงสร้างประโยคที่แตกต่างกัน

การใช้ การขยายข้อมูลเชิงความหมาย (Semantic Expansion) โดยอาศัยเทคนิค Word Embedding เพื่อเพิ่มคำที่มีความหมายใกล้เคียงลงในรายงาน

5.3.2 การศึกษาโมเดล Transformer อื่น

แม้ว่าผลการวิจัยนี้ชี้ให้เห็นถึงความสามารถของ BERT ในการจำแนกระดับความรุนแรงของจุดบกพร่อง แต่ในปัจจุบันยังมีโมเดล Transformer อื่น ๆ ที่ได้รับการพัฒนาเพื่อให้เหมาะกับงานจำแนกข้อความที่ซับซ้อนมากขึ้น ซึ่งควรได้รับการศึกษาเพิ่มเติม ได้แก่

RoBERTa ซึ่งเป็นการปรับปรุงจาก BERT โดยตัดส่วนของ Next Sentence Prediction (NSP) ออกไป และใช้การ Pretraining ที่เข้มข้นขึ้น ทำให้สามารถจับความหมายของข้อมูลข้อความได้ดีขึ้น

DeBERTa (Decoding-enhanced BERT with Disentangled Attention) ซึ่งเป็นโมเดลที่ปรับปรุงจาก BERT และ RoBERTa โดยใช้ Disentangled Self-Attention ทำให้สามารถเข้าใจความสัมพันธ์ของคำได้ดีขึ้น

5.3.3 การใช้เทคนิค Meta Learning และ Zero-shot Learning

การมีชุดข้อมูลจำนวนน้อยเป็นหนึ่งในข้อจำกัดสำคัญของการจำแนกระดับความรุนแรงของจุดบกพร่อง เนื่องจากการติดป้ายกำกับข้อมูลในโดเมนซอฟต์แวร์ต้องอาศัยผู้เชี่ยวชาญ การนำเทคนิคการเรียนรู้ขั้นสูงมาใช้สามารถช่วยลดข้อจำกัดนี้ได้ เช่น Meta Learning หรือ การเรียนรู้แบบเมตาที่ช่วยให้โมเดลสามารถเรียนรู้ได้เร็วขึ้นจากชุดข้อมูลที่มีขนาดเล็ก โดยใช้แนวคิดของ Few-shot Learning และ Zero-shot Learning ซึ่งช่วยให้โมเดลสามารถจำแนกข้อมูลในคลาสที่ไม่เคยพบมาก่อน โดยอาศัยความรู้ที่เรียนรู้จากคลาสที่มีอยู่ ตัวอย่างเช่น การใช้ GPT-4 หรือ T5 ที่สามารถเข้าใจบริบทของข้อมูลใหม่โดยอาศัยการเรียนรู้จากข้อความที่มีความหมายใกล้เคียง

5.3.4 การวิจัยข้ามโดเมนเพื่อเพิ่มความสามารถในการจำแนกของโมเดล

ปัจจุบัน โมเดลที่ถูกพัฒนาและฝึกโมเดลในงานวิจัยนี้อ้างอิงจากชุดข้อมูลของ Bugzilla และ Mozilla ซึ่งเป็นระบบรายงานจุดบกพร่องที่ได้รับความนิยม อย่างไรก็ตาม หากต้องการให้โมเดลสามารถใช้งานได้ในวงกว้างมากขึ้น ควรมีการทดสอบและพัฒนาโมเดลให้สามารถนำไปใช้กับรายงานจุดบกพร่องจากระบบอื่น ๆ เช่น Eclipse หรือ JIRA ได้โดยไม่ต้องมีการปรับแต่งมากนัก ซึ่งสามารถทำได้โดยการฝึกโมเดลด้วยข้อมูลจากหลายระบบติดตามข้อผิดพลาด การใช้เทคนิค Transfer Learning เพื่อปรับแต่งโมเดลให้สามารถรองรับข้อมูลจากหลายแหล่ง และประยุกต์ใช้แนวคิด Domain Adaptation เพื่อให้โมเดลสามารถทำงานได้ดีขึ้นกับข้อมูลที่มาจากระบบที่แตกต่างกัน

ข้อเสนอแนะเหล่านี้สามารถเป็นแนวทางสำคัญสำหรับการพัฒนางานวิจัยต่อไปในด้านการจำแนกระดับความรุนแรงของจุดบกพร่องจากรายงานจุดบกพร่อง โดยมุ่งเน้นการปรับปรุงเทคนิคการเสริมข้อมูล การศึกษาโมเดล Transformer อื่น ๆ ที่อาจให้ผลลัพธ์ที่ดีกว่า การนำเทคนิค Meta Learning และ Zero-shot Learning มาใช้เพื่อลดปัญหาการมีชุดข้อมูลขนาดเล็ก และการพัฒนาแพลตฟอร์มที่สามารถนำไปใช้งานจริงในอุตสาหกรรมซอฟต์แวร์ ซึ่งจะช่วยให้กระบวนการวิเคราะห์จุดบกพร่องมีความแม่นยำและมีประสิทธิภาพมากยิ่งขึ้น

บรรณานุกรม

[1] 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; 308-318.

[2] Herzig K, Just S, Zeller A. It's not a bug, it's a feature: how misclassification impacts bug prediction. 2013 35th international conference on software engineering (ICSE); IEEE; 392-401.

[3] Alaqail H, Ahmed S. Overview of software testing standard ISO/IEC/IEEE 29119. International Journal of Computer Science and Network Security (IJCSNS) 2018; 18[2]: 112-116.

[4] Lee D-G, Seo Y-S. Improving bug report triage performance using artificial intelligence based document generation model. Human-centric Computing and Information Sciences 2020; 10[1]: 1-22.

[5] 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 Advances in Intelligent Systems and Computing; Singapore. Springer Singapore; 423-430.

[6] Polpinij J. A method of non-bug report identification from bug report repository. Artificial Life and Robotics 2021; 26[3]: 318-328.

[7] Qin H, Sun X. Classifying bug reports into bugs and non-bugs using lstm. Proceedings of the tenth Asia-Pacific symposium on internetware; 1-4.

[8] Terdchanakul P, Hata H, Phannachitta P, Matsumoto K. Bug or not? bug report classification using n-gram idf. 2017 IEEE international conference on software maintenance and evolution (ICSME); IEEE; 534-538.

[9] Gomes LAF, da Silva Torres R, Côrtes ML. Bug report severity level prediction in open source software: A survey and research opportunities. Information and software technology 2019; 11558-78.

[10] Hamza A-J. Detecting Bug Severity Level using Machine Learning Techniques. Middle East University; 2021.

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

[12] Lamkanfi A, Demeyer S, Soetens QD, Verdonck T. Compasring mining algorithms for predicting the severity of a reported bug. 2011 15th European Conference on Software Maintenance and Reengineering; IEEE; 249-258.

[13] 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; 92-95.

[14] Ramay WY, Umer Q, Yin XC, Zhu C, Illahi I. Deep Neural Network-Based Severity Prediction of Bug Reports. IEEE Access 2019; 746846-46857.

[15] Dao A-H, Yang C-Z. Severity Prediction for Bug Reports Using Multi-Aspect Features: A Deep Learning Approach. Mathematics 2021; 9[14]: 1644.

[16] Mozilla Wiki. Bugzilla Field Descriptions. 6 July 2022 [30 July 2022]; https://wiki.mozilla.org/BMO/UserGuide/BugFields.

[17] QA Madness Software testing company. Bug Severity vs Priority, or How to Manage Defect Fixing. 20 April 2022]; https://www.qamadness.com/bug-severity-vs-priority.

[18] Ekanayake J. Bug Severity Prediction using Keywords in Imbalanced Learning Environment. Int J Inf Technol Comput Sci(IJITCS) 2021; 1353-60.

[19] Guo S, Chen R, Li H, Zhang T, Liu Y. Identify severity bug report with distribution imbalance by CR-SMOTE and ELM. International Journal of Software Engineering and Knowledge Engineering 2019; 29[02]: 139-175.

[20] Roy NKS, Rossi B. Cost-sensitive strategies for data imbalance in bug severity classification: Experimental results. 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA); IEEE; 426-429.

[21] Luaphol B, Polpinij J, Kaenampornpan M. Mining Bug Report Repositories to Identify Significant Information for Software Bug Fixing. Applied Science and Engineering Progress 2022; 15[3]: 4615-4615.

[22] IEEE Computer Society. IEEE 1044-2009 IEEE Standard Classification for Software Anomalies. 08 May 2022]; https://standards.ieee.org/ieee/1044/4607/.

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

[24] Samaroo A, Thompson G, Hambling B. Software Testing: An ISTQB-BCS Certified Tester Foundation Guide 3rd ed. BCS;

[25] Montgomery L, Lüders C, Maalej W. An Alternative Issue Tracking Dataset of Public Jira Repositories. 2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR); IEEE; 73-77.

[26] Wikipedia. Comparison of issue-tracking systems. 13 July 2022 [20 July 2022]; https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems.

[27] Bugzilla. Installation List. 08 May 2022]; https://www.bugzilla.org/installation-list/.

[28] NITIN INDURKHYA FJD. HANDBOOK OF NATURAL LANGUAGE PROCESSING. CRC Press; 2010.

[29] Polpinij J. Automatic TEXT Classification. Mahasarakham University; 2020.

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

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

[32] Alsahaf A, Petkov N, Shenoy V, Azzopardi G. A framework for feature selection through boosting. Expert Systems with Applications 2022; 187115895.

[33] Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta (BBA)-Protein Structure 1975; 405[2]: 442-451.

[34] Fawcett T. An introduction to ROC analysis. Pattern recognition letters 2006; 27[8]: 861-874.

[35] Ciaburro G, Venkateswaran B. Neural Networks with R: Smart models using CNN, RNN, deep learning, and artificial intelligence principles. Packt Publishing Ltd; 2017.

[36] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research 2020; 21[140]: 1-67.

[37] Zhang H, Li D. Naïve Bayes text classifier. 2007 IEEE international conference on granular computing (GRC 2007); IEEE; 708-708.

[38] Suthaharan S, Suthaharan S. Support vector machine. Machine learning models and algorithms for big data classification: thinking with examples for effective learning 2016; 207-235.

[39] Sarawan K, Polpinij J, Luaphol B. Machine Learning-Based Methods for Identifying Bug Severity Level from Bug Reports. International Conference on Computing and Information Technology; Springer; 199-208.

[40] Parmar A, Katariya R, Patel V. A review on random forest: An ensemble classifier. International conference on intelligent data communication technologies and internet of things (ICICI) 2018; Springer; 758-763.

[41] Kukkar A, Mohana R, Nayyar A, Kim J, Kang B-G, Chilamkurti N. A novel deep-learning-based bug severity classification technique using convolutional neural networks and random forest with boosting. Sensors 2019; 19[13]: 2964.

[42] Nakahara H, Jinguji A, Sato S, Sasao T. A random forest using a multi-valued decision diagram on an FPGA. 2017 IEEE 47th international symposium on multiple-valued logic (ISMVL); IEEE; 266-271.

[43] Dreiseitl S, Ohno-Machado L. Logistic regression and artificial neural network classification models: a methodology review. Journal of biomedical informatics 2002; 35[5-6]: 352-359.

[44] El-Habil AM. An application on multinomial logistic regression model. Pakistan journal of statistics and operation research 2012; 271-291.

[45] Polikar R. Ensemble learning. Ensemble machine learning: Methods and applications 2012; 1-34.

[46] Dey R, Mathur R. Ensemble learning method using stacking with base learner, a comparison. International Conference on Data Analytics and Insights; Springer; 159-169.

[47] Zhang C, Ma Y. Ensemble machine learning. Springer; 2012.

[48] Ngo G, Beard R, Chandra R. Evolutionary bagging for ensemble learning. Neurocomputing 2022; 5101-14.

[49] Raza K. Improving the prediction accuracy of heart disease with ensemble

learning and majority voting rule. *U-healthcare monitoring systems*: Elsevier 2019:179-196.

[50] LeCun Y, Bengio Y, Hinton G. Deep learning. nature 2015; 521[7553]: 436-444.

[51] Baheti P. The Complete Guide to Recurrent Neural Networks. 26-03-2023]; https://www.v7labs.com/blog/recurrent-neural-networks-guide.

[52] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation 1997; 9[8]: 1735-1780.

[53] Le X-H, Ho HV, Lee G, Jung S. Application of long short-term memory (LSTM) neural network for flood forecasting. Water 2019; 11[7]: 1387.

[54] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems 2017; 30

[55] Alammar J. The Illustrated Transformer. 26-03-2023]; http://jalammar.github.io/illustrated-transformer/.

[56] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers); 4171-4186.

[57] Luaphol B. Assembling of Dependent Bug Reports from Bug Report Repository. Mahasarakham University; 2020.

[58] Roy NK-S, Rossi B. Towards an improvement of bug severity classification. 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications; IEEE; 269-276.

[59] Lamkanfi A, Pérez J, Demeyer S. The eclipse and mozilla defect tracking dataset: a genuine dataset for mining bug information. 2013 10th Working Conference on Mining Software Repositories (MSR); IEEE; 203-206.

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

[61] Sharmin S, Aktar F, Ali AA, Khan MAH, Shoyaib M. BFSp: A feature selection method for bug severity classification. 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC); IEEE; 750-754.

[62] Tan Y, Xu S, Wang Z, Zhang T, Xu Z, Luo X. Bug severity prediction using question-and-answer pairs from stack overflow. Journal of Systems and Software 2020; 165110567.

[63] Kukkar A, Mohana R, Kumar Y. Does bug report summarization help in enhancing the accuracy of bug severity classification? Procedia Computer Science 2020; 1671345-1353.

[64] Kim J, Yang G. Bug severity prediction algorithm using topic-based feature selection and CNN-LSTM algorithm. IEEE Access 2022; 1094643-94651.

[65] Wei Y, Zhang C, Ren T. Improving Bug Severity Prediction with Domain-Specific Representation Learning. IEEE Access 2023;

[66] Fang S, Zhang T, Tan Y, Jiang H, Xia X, Sun X. RepresentThemAll: A universal learning representation of bug reports. 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE); IEEE; 602-614.

[67] Fang S, Tan Y-s, Zhang T, Xu Z, Liu H. Effective prediction of bug-fixing priority via weighted graph convolutional networks. IEEE Transactions on Reliability 2021; 70[2]: 563-574.

[68] Wei J, Zou K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:190111196 2019;

[69] Pradana RC, Joddy S, Girsang AS. Easy data augmentation for handling imbalanced data in fake news detection. 2023 International Conference on Technology, Engineering, and Computing Applications (ICTECA); IEEE; 1-5.

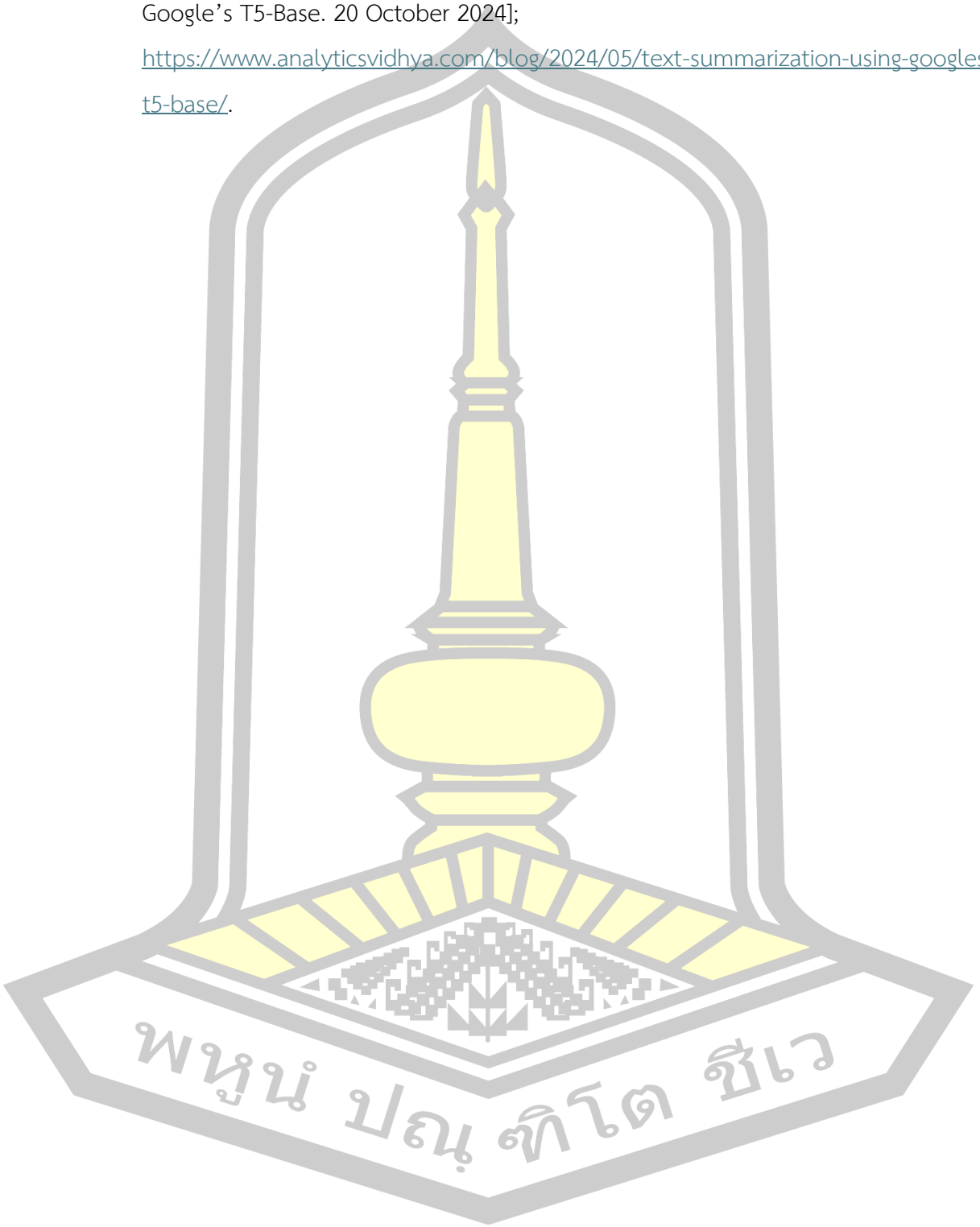
[70] Patil AP, Jere S, Ram R, Srinarasi S. T5w: A paraphrasing approach to oversampling for imbalanced text classification. 2022 IEEE international conference on electronics, computing and communication technologies (CONECCT); IEEE; 1-6.

[71] Wang Y. Research on the TF–IDF algorithm combined with semantics for automatic extraction of keywords from network news texts. Journal of Intelligent Systems 2024; 33[1]: 20230300.

[72] Zhu Z, Liang J, Li D, Yu H, Liu G. Hot topic detection based on a refined TF-IDF algorithm. IEEE access 2019; 726996-27007.

[73] Holtzman A, Buys J, Du L, Forbes M, Choi Y. The curious case of neural text degeneration. arXiv preprint arXiv:190409751 2019;

[74] V M. Analytics Vidhya. How to Build an Effective Text Summarization Model Using Google's T5-Base. 20 October 2024]; https://www.analyticsvidhya.com/blog/2024/05/text-summarization-using-googles-t5-base/.



พหุน ปณฺ ฑิโต ชีเว

ประวัติผู้เขียน

ชื่อ	นายกำธร สารวรรณ
วันเกิด	วันที่ 16 ตุลาคม พ.ศ. 2526
สถานที่เกิด	อำเภอเสลภูมิ จังหวัดร้อยเอ็ด
สถานที่อยู่ปัจจุบัน	145/72 ถนนถีนานนท์ ตำบลตลาด อำเภอเมือง จังหวัดมหาสารคาม รหัสไปรษณีย์ 44000
ตำแหน่งหน้าที่การงาน	อาจารย์
สถานที่ทำงานปัจจุบัน	สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยกาฬสินธุ์ 62/1 ถนนเกษตรสมบูรณ์ ตำบลกาฬสินธุ์ อำเภอเมือง จังหวัดกาฬสินธุ์ ไปรษณีย์ 46000
ประวัติการศึกษา	พ.ศ. 2549 วิทยาศาสตรบัณฑิต (วท.บ.) สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม พ.ศ. 2554 วิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยมหาสารคาม พ.ศ. 2568 ปรัชญาดุษฎีบัณฑิต (ปร.ด.) สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยมหาสารคาม