การจำแนกประเภทบทวิจารณ์ของผู้ใช้โมไบล์แอปพลิเคชันเพื่อการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา

นายกิตติศักดิ์ เพชรรุ่งนภา

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

Classification of Mobile Application User Reviews for Generating Tickets for Issue Tracking System

Mr. Kittisak Phetrungnapha

An Independent Study Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2018

หัวข้อสารนิพนธ์	การจำแนกประเภทบทวิจารณ์ของผู้ใช้โมไบล์แอปพลิเคชัน เพื่อการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา
โดย	นายกิตติศักดิ์ เพชรรุ่งนภา
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาหลัก	รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับสารนิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

คณะกรรมการสอบสารนิพนธ์

..... ประธานกรรมการ
(ดร.ดวงดาว วิชาดากุล)

..... อาจารย์ที่ปรึกษาหลัก
(รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

..... กรรมการ
(ดร.กุลวดี ศรีพานิชกุลชัย)

กิตติศักดิ์ เพชรรุ่งนภา : การจำแนกประเภทบทวิจารณ์ของผู้ใช้โมไบล์แอปพลิเคชันเพื่อการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา. (Classification of Mobile Application User Reviews for Generating Tickets for Issue Tracking System) อ.ที่ปรึกษาหลัก : รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

ปัจจุบันการพัฒนาโมไบล์แอปพลิเคชันได้เข้ามาอยู่ในสายการพัฒนาซอฟต์แวร์หลัก และมีโมไบล์แอปพลิเคชันจำนวนมากที่มีฟังก์ชันการทำงานประเภทเดียวกันให้สามารถใช้งานได้บนแอปสโตร์และเพลย์สโตร์ ทำให้เกิดการแข่งขันทางการตลาดสูงมาก ซึ่งทีมพัฒนาจำเป็นที่จะต้องทราบผลตอบรับ ข้อคิดเห็นของผู้ใช้งานจริง เพื่อนำมาปรับปรุงคุณภาพของโมโบล์แอปพลิเคชันให้ดียิ่งขึ้น แหล่งข้อมูลที่สำคัญคือบทวิจารณ์ของผู้ใช้งานบนแอปสโตร์หรือเพลย์สโตร์ นักพัฒนาสามารถที่จะวิเคราะห์ปัญหาการใช้งานแอปพลิเคชันเพื่อเป็นข้อแนะนำสำหรับการบำรุงรักษาและการพัฒนาแอปพลิเคชันในเวอร์ชันถัดไปได้ แต่เนื่องจากในหนึ่งเวอร์ชันของแอปพลิเคชันมีจำนวนบทวิจารณ์ของผู้ใช้เป็นจำนวนมาก โครงงานมหาบัณฑิตนี้จึงมุ่งเน้นไปที่การจำแนกประเภทของบทวิจารณ์ของผู้ใช้ว่าเป็นรายงานข้อผิดพลาด หรือความต้องการฟังก์ชันการทำงานใหม่ ซึ่งกระบวนการทำงานเป็นไปอย่างอัตโนมัติ โดยผลการจำแนกประเภทของบทวิจารณ์จะถูกนำไปใช้ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหาที่ชื่อว่าจิราต่อไป โดยในการแยกประเภทบทวิจารณ์ของผู้ใช้นี้ได้นำเทคนิคการแยกประเภทของข้อความ การประมวลผลภาษาธรรมชาติ การวิเคราะห์อารมณ์ความรู้สึกจากข้อความ และวิเคราะห์ข้อมูลเมดาเดตาของบทวิจารณ์มาใช้วิเคราะห์ โดยอัลกอริทึมการเรียนรู้ของเครื่องหลายอัลกอริทึมได้ถูกนำมาใช้ในการแยกประเภทของบทวิจารณ์ ได้แก่ นาอีฟเบย์ส ต้นไม้การตัดสินใจ เคเอ็นเอ็น ลิเนียร์เอสวีซี ลอจิสติกรีเกรสชัน วิธีเอนเซมเบิลแบบต่าง ๆ ซึ่งโมเดลที่มีประสิทธิภาพดีที่สุดของแต่ละการจำแนกประเภทบทวิจารณ์จะถูกนำไปใช้ต่อในขั้นตอนการสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจิราผ่านเครื่องมือที่พัฒนาขึ้นในโครงงานมหาบัณฑิต โดยเครื่องมือสามารถตรวจสอบความเหมือนกันเชิงความหมายและกรองบทวิจารณ์ซ้ำซ้อนได้

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2561 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6070908021 : MAJOR SOFTWARE ENGINEERING

KEYWORD: user reviews, issue tracking system, text classification, text similarity, text summarization, machine learning, natural language processing

Kittisak Phetrungnapha : Classification of Mobile Application User Reviews for Generating Tickets for Issue Tracking System. Advisor: Assc.Prof. Dr. TWITTIE SENIVONGSE

Mobile application development has now been in the mainstream and a lot of mobile applications have been released to Apple App Store and Google Play Store. As there are many applications in the same category and the competition is very high in the market, mobile development teams need to get user feedback on their applications so that they can improve quality of the applications. An important source of feedback is user review on App Store and Play Store from which the developers can analyze problems and recommendations for future maintenance and evolution of the applications. Since there might be a large number of user reviews for each release of a mobile application, this master project proposes an automated approach to classifying user reviews as bug reports or feature requests. These classification results are used as a basis for generating tickets for an issue tracking system, i.e. Jira. In user review classification, text classification, natural language processing, sentiment analysis, and review metadata are used with several machine learning algorithms, i.e. Naïve Bayes, Decision Tree, KNN, LinearSVC, Logistic Regression, and Ensemble methods. The best classifiers for both categories of reviews are used further in an implementation of a Jira ticket generating tool. The tool considers semantic similarity of review comments and can filter out duplicate user reviews.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2018 Advisor's Signature .....

กิตติกรรมประกาศ

ขอกราบขอบพระคุณท่าน รศ. ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาโครงงานมหาบัณฑิตของข้าพเจ้าเป็นอย่างยิ่งที่ได้เสียสละเวลาอันมีค่า เพื่อให้คำปรึกษาแนะนำทางด้านการศึกษาหาความรู้ คุณธรรม จริยธรรม และแนวทางสำหรับการทำโครงงานมหาบัณฑิตนี้ ตลอดจนคอยดูแลและประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำโครงงานมหาบัณฑิตทุกคน

ขอขอบพระคุณพี่ที่บริษัท อูซู จำกัด ทุกท่านที่ให้คำปรึกษาแนะนำ รวมทั้งช่วยยืนยันผลการติดฉลากสำหรับบทวิจารณ์สองบทวิจารณ์ใด ๆ ว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่ เพื่อนำผลลัพธ์ที่ได้ไปใช้ในการคำนวณหาค่าขีดแบ่งในการจำแนกบทวิจารณ์ซ้ำซ้อน

ขอขอบคุณเพื่อน ๆ หลักสูตรวิศวกรรมซอฟต์แวร์ สำหรับกำลังใจและคำปรึกษาแนะนำในการจัดทำโครงงานมหาบัณฑิต

สุดท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา และสมาชิกในครอบครัวทุกท่านที่คอยให้กำลังใจและให้การช่วยเหลือสนับสนุนมาโดยตลอด

กิตติศักดิ์ เพชรรุ่งนภา

สารบัญ

หน้า

..... ค

บทคัดย่อภาษาไทย ..... ค

..... ง

บทคัดย่อภาษาอังกฤษ ..... ง

กิตติกรรมประกาศ ..... จ

สารบัญ ..... ฉ

สารบัญตาราง ..... ฌ

สารบัญภาพ ..... ญ

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 วัตถุประสงค์ของโครงงาน 2

1.3 ขอบเขตของโครงงาน 2

1.4 ขั้นตอนและวิธีการดำเนินการ 3

1.5 ประโยชน์ที่คาดว่าจะได้รับ 3

1.6 โครงสร้างของเนื้อหาในโครงงานมหาบัณฑิต 4

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 5

2.1 ทฤษฎีที่เกี่ยวข้อง 5

2.1.1 การเรียนรู้แบบมีผู้สอน [9] 5

2.1.2 นาอีฟเบย์ส [10] 6

2.1.3 เคเอ็นเอ็น [11] 7

2.1.4 ต้นไม้การตัดสินใจ [12] 8

2.1.5 การเรียนรู้แบบรวม [13] 8
2.1.6 จิราซอฟต์แวร์ [14] 9
2.1.7 ระบบติดตามปัญหา [15] 10
2.1.8 ทิคเก็ต [15] 10
2.1.9 โปรดักส์แบล็คล็อก [16] 11
2.1.10 เอไจล์บอร์ด [17] 11
2.1.11 ความเหมือนกันของข้อความ [18] 12
2.1.12 การสกัดข้อความ [19] 12
2.1.13 ยูนิเวอร์ซัล เซนเทนต์ เอนโค้ดเดอร์ [20] 13
2.1.14 โคซายน์ ซิมิลาริตี [21] 13
2.2 งานวิจัยที่เกี่ยวข้อง 14
2.2.1 Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews [22] 14
2.2.2 Mobile App Evolution Analysis based on User Reviews [23] 15
2.2.3 What Do Mobile App Users Complain About? [24] 16
2.2.4 App Store Mining is Not Enough [25] 17
2.2.5 AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace [26] 19
2.2.6 Automatically Create Jira Issue from Firebase Crashlytics [27] 20
2.2.7 Recommending and Localizing Change Requests for Mobile Apps based on User Reviews [28] 21
2.2.8 Ensemble Methods for App Review Classification: An Approach for Software Evolution [29] 22
บทที่ 3 การพัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ 23
3.1 ขั้นตอนการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ 24

3.2 ขั้นตอนการพัฒนาโมเดลเพื่อจำแนกประเภทของบทวิจารณ์ของผู้ใช้ 27
3.3 ขั้นตอนการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ 33
3.4 ขั้นตอนการนำออกโมเดลเพื่อใช้ในการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ 39
บทที่ 4 การตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้ 40
4.1 การเตรียมชุดข้อมูลเพื่อทำการทดลองหาความคล้ายกันของบทวิจารณ์ของผู้ใช้ 41
4.2 การหาค่าขีดแบ่งสำหรับบทวิจารณ์ซ้ำซ้อน 43
บทที่ 5 การสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้ 45
5.1 การรวบรวมบทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน 46
5.2 การวิเคราะห์บทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน 48
5.3 การสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจิรา 50
บทที่ 6 บทสรุปโครงงานและข้อเสนอแนะ 56
6.1 สรุปผลโครงงานมหาบัณฑิต 56
6.2 ข้อจำกัดในโครงงานมหาบัณฑิต 57
6.3 ข้อเสนอแนะ 57
บรรณานุกรม 59
ประวัติผู้เขียน 63

สารบัญตาราง

ตารางที่ 3.1 ฟีเจอร์ที่แตกต่างกันที่ใช้เป็นข้อมูลนำเข้าในการเรียนรู้ให้กับโมเดล 27

ตารางที่ 3.2 รายการฟีเจอร์ที่เกิดจากการนำฟีเจอร์ที่แตกต่างกันแต่ละฟีเจอร์มาใช้งานร่วมกัน 28

ตารางที่ 3.3 จำนวนข้อมูลบทวิจารณ์ของผู้ใช้แยกตามหมวดหมู่ และวัตถุประสงค์ของการนำไปใช้ 29

ตารางที่ 3.4 ค่า *part of speech* ของแต่ละชนิดของคำของประโยคตัวอย่าง 31

ตารางที่ 3.5 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *bug report* หรือ *not bug report* ด้วยใช้ *5-Folds Cross Validation* 34

ตารางที่ 3.6 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *bug report* หรือ *not bug report* โดยใช้อัตราส่วนของข้อมูลระหว่าง *training* และ *test* เป็น *70:30* 35

ตารางที่ 3.7 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *feature request* หรือ *not feature request* ด้วยวิธี *5-Folds Cross Validation* 37

ตารางที่ 3.8 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *feature request* หรือ *not feature request* โดยใช้อัตราส่วนของข้อมูลระหว่าง *training* และ *test* เป็น *67:33* 38

ตารางที่ 4.1 ค่าประสิทธิภาพการวัดความซ้ำซ้อนของบทวิจารณ์ที่ *thresholds* ต่าง ๆ 44

ตารางที่ 5.1 ฟิลด์ที่จำเป็นต้องใช้ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา *Jira* เพื่อเชื่อมต่อกับ *Jira APIs* 50

สารบัญภาพ

ภาพที่ 2.1 กระบวนการเทรนเพื่อให้ได้โมเดลที่ต้องการ [9]..... 5

ภาพที่ 2.2 การนำโมเดลที่ผ่านการเรียนรู้แล้วมาลองใช้งานจริง [9]..... 6

ภาพที่ 2.3 รูปดาวจะมีคลาสคำตอบเป็นวงกลมสีเขียวเมื่อใช้ *k = 3* [11]..... 7

ภาพที่ 2.4 ผลลัพธ์การทำนายของโมเดล เมื่อมีค่า *Bias* และ *Variance* แตกต่างกันออกไป โดยจุดสีแดงคือค่าจริง จุดน้ำเงินคือค่าที่ได้จากการทำนายของโมเดล [13]..... 9

ภาพที่ 2.5 ความสัมพันธ์ระหว่าง *Model Complexity* และค่า *Error* ที่เกิดขึ้นที่เกี่ยวข้องกับค่า *Bias* และ *Variance* [13]..... 9

ภาพที่ 2.6 สตอรีที่บอกว่าผู้ใช้งานสามารถกดปุ่มลืมรหัสผ่านเพื่อทำการรีเซ็ตรหัสผ่านใหม่ได้..... 11

ภาพที่ 2.7 เอไจล์บอร์ดที่มีสี่คอลัมน์ คือ *To Do, Blocked, In Progress,* และ *Done*..... 12

ภาพที่ 2.8 คุณสมบัติของโมเดลแต่ละประเภทที่ถูกนำมาทำ *Universal Sentence Encoder* [20] 13

ภาพที่ 2.9 การเปรียบเทียบประสิทธิภาพของโมเดลโดยใช้เทคนิคของ *Naïve Bayes* ต่าง ๆ กับบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ที่ได้เลือกมา [22]..... 14

ภาพที่ 2.10 ตัวอย่างกราฟการจัดกลุ่มความคิดเห็นของผู้ใช้งานโดยใช้โมเดลของผู้วิจัยของแอปพลิเคชันแอนดรอยด์ *Whatsapp* ที่มีการอัปเดต *major* ทั้งสิ้น *7* ครั้ง และ *minor 39* ครั้ง [23]..... 15

ภาพที่ 2.11 ประเภทบทวิจารณ์เชิงลบของผู้ใช้งานแอปพลิเคชันบนแอปสโตร์ โดยจัดลำดับตามความถี่ของการพบปัญหาจากมากไปน้อยทั้งสิ้น *13* ประเภท [24]..... 17

ภาพที่ 2.12 เปรียบเทียบจำนวน *feature request* และ *bug report* ของทั้ง *70* แอปพลิเคชันที่ผู้วิจัยได้เลือกมา แยกตามแหล่งที่มาของข้อมูล คือ ทวิตเตอร์ สโตร์ ทั้งทวิตเตอร์และสโตร์ และ % แสดง *feature request* กับ *bug report* ที่พบเฉพาะบนทวิตเตอร์ [25]..... 18

ภาพที่ 2.13 การแสดงผลของ *AR-Miner* ของกลุ่มความคิดเห็น *10* กลุ่มแรกที่มีความสำคัญมากที่สุดของแอปพลิเคชัน *SwiftKey* โดย *Keyword “more theme”* ถูกจัดเป็นอันดับหนึ่ง [26]..... 20

ภาพที่ 2.14 ตัวอย่างทิคเก็ตประเภท *bug* ที่ถูกสร้างใน *Jira* จาก *Firebase Crashlytics* ที่ทำการเชื่อมต่อกับ *APIs* ของ *Jira* [27]..... 21

ภาพที่ 2.15 ข้อมูลแต่ละแอปพลิเคชันที่เก็บรวบรวมจากบทวิจาณ์ของผู้ใช้และทำการติดฉลากแล้ว [29] 22

ภาพที่ 2.16 ค่าประสิทธิภาพที่ได้จากชุดข้อมูลทดสอบโดยใช้โมเดลการจำแนกประเภทแบบเดี่ยว และโมเดลแบบรวม [29] 22

ภาพที่ 3.1 ขั้นตอนการดำเนินงานแบบละเอียดของการพัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ 23

ภาพที่ 3.2 หน้าเว็บไซต์ของกลุ่มผู้จัดทำงานวิจัยที่ *2.2.1* ที่เปิดให้ดาวน์โหลดข้อมูลบทวิจารณ์ของผู้ใช้ที่ถูกติดฉลากแล้ว [22] 24

ภาพที่ 3.3 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *bug report* 25

ภาพที่ 3.4 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *not bug report* 26

ภาพที่ 3.5 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *feature request* 26

ภาพที่ 3.6 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *not feature request* 27

ภาพที่ 3.7 ฟังก์ชันการนับจำนวน *part of speech* ที่เขียนด้วยภาษา *Python* เวอร์ชัน *3* 30

ภาพที่ 3.8 ตัวอย่างเวกเตอร์ของคำว่า *"Very"* ที่ได้จากการประมวลผลด้วย *GoogleNews-vectors-negative300* 32

ภาพที่ 3.9 ตัวอย่างการใช้งานฟังก์ชัน *classification_report* จาก *scikit-learn.org* 33

ภาพที่ 3.10 ตัวอย่างการใช้งานฟังก์ชัน *accuracy_score* จาก *scikit-learn.org* 33

ภาพที่ 3.11 ตัวอย่างโค้ดการใช้ไลบรารี *Pickle* ในการนำออกโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ 39

ภาพที่ 4.1 ขั้นตอนการดำเนินงานแบบละเอียดของการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้ 40

ภาพที่ 4.2 ตัวอย่างบทวิจารณ์ใน *base set* 41

ภาพที่ 4.3 ตัวอย่างบทวิจารณ์ใน *test set* 42

ภาพที่ 4.4 ตัวอย่างการติดฉลากระหว่างบทวิจารณ์ใน *base set* และ *test set* 42

ภาพที่ 4.5 ตัวอย่างฟังก์ชันการแปลงข้อความเป็นเวกเตอร์ด้วย *Universal Sentence Encoder* ที่ทำงานบน *Tensorflow* 43

ภาพที่ 4.6 ตัวอย่างฟังก์ชันการหาค่า *Cosine Similarity* ระหว่างสองเวกเตอร์ใด ๆ 43

ภาพที่ 5.1 ขั้นตอนการดำเนินงานแบบละเอียดของการทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้ 45

ภาพที่ 5.2 ฟังก์ชันการดึงข้อมูลบทวิจารณ์ของผู้ใช้ที่อยู่บนแอปสโตร์ของแอปพลิเคชัน *Facebook* 46

ภาพที่ 5.3 ตัวอย่างบทวิจารณ์ของผู้ใช้ที่ได้จากการดึงข้อมูลบนแอปสโตร์ของแอปพลิเคชัน *Facebook* 46

ภาพที่ 5.4 ฟังก์ชันการดึงข้อมูลบทวิจารณ์ของผู้ใช้ที่อยู่บนเพลย์สโตร์ของแอปพลิเคชัน *Facebook* 47

ภาพที่ 5.5 ตัวอย่างบทวิจารณ์ของผู้ใช้ที่ได้จากการดึงข้อมูลบนเพลย์สโตร์ของแอปพลิเคชัน *Facebook* 47

ภาพที่ 5.6 โปรแกรม *Command line* ที่รองรับแพลตฟอร์มและแอปพลิเคชันไอดีที่ต้องการดึงข้อมูลบทวิจารณ์ของผู้ใช้ 47

ภาพที่ 5.7 ตัวอย่างไฟล์ *.pkl* ที่ถูกนำเข้าในโปรแกรม *Command line* 48

ภาพที่ 5.8 ฟังก์ชันการ *decode* ไฟล์ *.pkl* กลับเป็น *prediction code* ภาษา *Python* 48

ภาพที่ 5.9 ฟังก์ชันการนำคำหยุดออก และการทำให้เป็นรากศัพท์ 48

ภาพที่ 5.10 ตัวอย่างฟังก์ชันการทำนายประเภทของบทวิจารณ์ว่าเป็น *bug report* หรือ *not but report* 49

ภาพที่ 5.11 ตัวอย่างฟังก์ชันการทำนายประเภทของบทวิจารณ์ว่าเป็น *feature request* หรือ *not feature request* 49

ภาพที่ 5.12 ข้อมูลที่ผู้ใช้เครื่องมือจำเป็นต้องระบุเองจากตารางที่ *5.1* 51

ภาพที่ 5.13 ฟังก์ชันการคำนวณหาความถี่ของคำที่แตกต่างกันทั้งหมดในบทวิจารณ์ใด ๆ (*count_word_frequencies)* และฟังก์ชันสำหรับคำนวณคะแนนความสำคัญของแต่ละประโยคที่อยู่บทวิจารณ์ (*count_sentence_scores)* 51

ภาพที่ 5.14 ฟังก์ชันการเลือกประโยคที่สำคัญที่สุดจากบทวิจารณ์ของผู้ใช้ออกมา *n* ประโยค โดยในโครงงานมหาบัณฑิตนี้มีค่า *n = 1* 52

ภาพที่ 5.15 ฟังก์ชันการเชื่อมต่อกับ *Jira APIs* ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา *Jira* 53

ภาพที่ 5.16 โปรแกรม *Command line* แสดงผลลัพธ์หลังจากทำการสร้างทิคเก็ตบน *Jira board* แล้ว 54
ภาพที่ 5.17 ตัวอย่างทิคเก็ตบน *Jira board* ที่อธิบายเกี่ยวกับปัญหาการใช้งานแอปพลิเคชัน 54
ภาพที่ 5.18 ตัวอย่างทิคเก็ตที่นำไปใช้งานจริงได้น้อย 55

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันสมาร์ตโฟน (Smartphone) [1] ได้เข้ามาเป็นส่วนหนึ่งในชีวิตประจำวันของคน ผู้คนใช้สมาร์ตโฟนในการทำกิจกรรมต่าง ๆ มากมาย เช่น การติดต่อสื่อสาร การถ่ายภาพ การเชื่อมต่ออินเทอร์เน็ตหาข้อมูลข่าวสาร เป็นต้น โดยเฉพาะอย่างยิ่งการใช้งานโมไบล์แอปพลิเคชันเพื่ออำนวยความสะดวกต่าง ๆ จาก แอปสโตร์ (App Store) [2] หรือ เพลย์สโตร์ (Play Store) [3] มีจำนวนเพิ่มสูงขึ้นอย่างต่อเนื่องทั้งในแง่ของจำนวนแอปพลิเคชัน และยอดดาวน์โหลดแอปพลิเคชัน ก่อให้เกิดเม็ดเงินหมุนเวียนในตลาดของอุตสาหกรรมเทคโนโลยีเป็นจำนวนมหาศาล แต่ละบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระต่างแข่งขันกันอย่างรุนแรง โดยเฉพาะในแง่ของคุณภาพของแอปพลิเคชันที่ใช้งานต้องมีคุณภาพดี ตอบสนองต่อความต้องการของผู้ใช้ เพื่อช่วยให้รักษาฐานผู้ใช้งานให้คงอยู่เอาไว้ให้ได้นานที่สุด

ในการที่บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระจะผลิตหรือปรับปรุงแอปพลิเคชันให้ออกมามีคุณภาพดี ตรงตามความต้องการของผู้ใช้ จำเป็นที่จะต้องทราบถึงความคิดเห็นของผู้ใช้งานแอปพลิเคชัน ว่ามีความพึงพอใจ ชอบหรือไม่ชอบในส่วนใด ความต้องการฟังก์ชันการทำงานที่ยังขาดอยู่ รวมไปถึงจุดบกพร่องต่าง ๆ ที่พบในระหว่างการใช้งานแอปพลิเคชัน ซึ่งหนึ่งในแหล่งข้อมูลที่แอปสโตร์และเพลย์สโตร์อนุญาตให้ผู้ใช้งานสามารถแสดงความคิดเห็นเกี่ยวกับแอปพลิเคชันในรูปแบบของคะแนน และความคิดเห็น ก็คือ บทวิจารณ์ของผู้ใช้ (User Reviews) [4] ซึ่งเป็นสิ่งที่มาจากผู้ใช้งานโดยตรง บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระสามารถที่จะนำข้อมูลในส่วนนี้มาใช้ในการวิเคราะห์และตัดสินใจต่าง ๆ เกี่ยวกับการปรับปรุงแอปพลิเคชันได้

เนื่องจากบทวิจารณ์ของผู้ใช้ของแอปพลิเคชันนั้นเป็นเพียงการให้คะแนนและการแสดงความคิดเห็นในรูปแบบของข้อความเท่านั้น ทำให้บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระต้องทำการอ่านและวิเคราะห์เองว่าสิ่งที่ผู้ใช้งานกำลังพูดถึงนั้นเป็นเรื่องเกี่ยวกับอะไร ประกอบกับจำนวนบทวิจารณ์ของผู้ใช้ก็มีจำนวนมาก บางบทวิจารณ์ของผู้ใช้ก็ไม่มีประโยชน์ต่อการตัดสินใจเนื่องจากมีข้อมูลน้อยเกินไป ทำให้เสียเวลาในส่วนนี้ไปเป็นอย่างมาก แทนที่จะนำเวลาในส่วนนี้ไปใช้ในการพัฒนาฟังก์ชัน หรือแก้ไขจุดบกพร่องต่าง ๆ รวมไปถึงหลังจากที่ได้ข้อมูลที่ผ่านการวิเคราะห์มาแล้ว ยังต้องมาทำการบริหารจัดการโครงการ เช่น การสร้างทิคเก็ต (Tickets) สำหรับระบบติดตามปัญหา (Issue Tracking System) ซึ่งเป็นสิ่งที่ต้องทำในการปรับปรุงแก้ไขแอปพลิเคชันในรอบการทำงานครั้งถัดไปในการพัฒนาแบบเอไจล์ (Agile) [5] เป็นต้น

จากปัญหาดังกล่าวข้างต้น ทางผู้วิจัยจึงมีแนวคิดที่จะนำทฤษฎี ความรู้ต่าง ๆ ในศาสตร์ทางด้านปัญญาประดิษฐ์ [6] มาช่วยแก้ปัญหา เช่น การทำเหมืองข้อมูล [7] การเรียนรู้ของเครื่อง [8] เป็นต้น มาช่วยในการเก็บรวบรวม วิเคราะห์ และจำแนกประเภทบทวิจารณ์ของผู้ใช้ดังกล่าวว่าเป็นประเภทใดระหว่างความต้องการฟังก์ชันงานใหม่ (Feature request) หรือรายงานข้อผิดพลาด (Bug report) เพื่อช่วยลดระยะเวลาที่ต้องใช้คนมาดำเนินการเอง ซึ่งหลังจากได้โมเดลการเรียนรู้แล้ว ทางผู้วิจัยยังมีแนวคิดที่จะสร้างเครื่องมือที่ช่วยในการสร้างทิคเก็ตบนระบบติดตามปัญหาซึ่งในโครงงานมหาบัณฑิตนี้หมายถึง ซอฟต์แวร์จิรา (Jira) โดยจะมี

การแยกประเภทของทิคเก็ตตามประเภทของบทวิจารณ์ของผู้ใช้ ช่วยให้ลดระยะเวลาของงานที่ต้องทำในส่วนนี้ของการพัฒนาซอฟต์แวร์แบบเอไจล์ลง และช่วยสนับสนุนให้บริษัทพัฒนาซอฟต์แวร์และนักพัฒนาอิสระในการแก้ไขจุดบกพร่องต่าง ๆ ที่ผู้ใช้งานพบ รวมไปถึงการพัฒนาฟังก์ชันการทำงานใหม่ ๆ ที่ตอบสนองต่อความต้องการของผู้ใช้งาน ก่อนที่จะทำการอัปเดตเวอร์ชันของแอปพลิเคชันให้ผู้ใช้งานต่อไป

1.2 **วัตถุประสงค์ของโครงงาน**

1.2.1 เพื่อพัฒนาโมเดลที่ใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ของแอปพลิเคชันที่ต้องการได้

1.2.2 เพื่อทดสอบหาค่าขีดแบ่ง (Threshold) ที่มีประสิทธิภาพสูงที่สุดในการจำแนกบทวิจารณ์ของผู้ใช้สองบทวิจารณ์ใด ๆ ว่ามีความซ้ำซ้อนกันหรือไม่จากการคำนวณหาความคล้ายคลึงกันเชิงความหมายของบทวิจารณ์ของผู้ใช้

1.2.3 เพื่อพัฒนาเครื่องมือที่นำบทวิจารณ์ของผู้ใช้ที่จำแนกประเภทเป็น feature request และ bug report ไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของทิคเก็ตที่ถูกสร้างจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้ที่จำแนกประเภทได้

1.3 **ขอบเขตของโครงงาน**

1.3.1 สร้างโมเดลที่สามารถจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์หรือเพลย์สโตร์ ออกเป็นประเภท bug report หรือ feature request

1.3.2 โมเดลที่จำแนกประเภทของบทวิจารณ์ของผู้ใช้รองรับเฉพาะภาษาอังกฤษเท่านั้น

1.3.3 เครื่องมือที่ใช้สร้าง Jira ticket จากบทวิจารณ์ของผู้ใช้ต้องสามารถสร้างทิคเก็ตที่แยกตามประเภทของบทวิจารณ์ของผู้ใช้ที่ได้จากโมเดลอย่างถูกต้องทั้งทิคเก็ตประเภท bug report และทิคเก็ตประเภท feature request

1.3.4 ฟีเจอร์ที่จะใช้ในการเรียนรู้โมเดลอย่างน้อย คือ rating, text (Bags of Word), user review sentiment, และฟีเจอร์ใหม่ ๆ ที่ได้จาก Feature extraction จากบทวิจารณ์ของผู้ใช้ที่ผู้วิจัยจะเพิ่มเติมเข้ามา

1.3.5 บทวิจารณ์ของผู้ใช้ใด ๆ จะถูกจัดให้อยู่ในประเภทใดประเภทหนึ่งได้เพียงประเภทเดียวเท่านั้น

1.3.6 เครื่องมือที่ใช้สร้าง Jira ticket จากบทวิจารณ์ของผู้ใช้สามารถรองรับการตรวจสอบทิคเก็ตซ้ำซ้อนได้

1.3.7 ข้อมูลบางส่วนที่อยู่ภายในทิคเก็ตจะกำหนดค่าเริ่มต้น (Default) ให้ เช่น ความสำคัญของทิคเก็ต (Priority) นั้นจะมีค่าเริ่มต้นเป็นปานกลาง (Medium) เท่านั้น

1.3.8 ประเมินประสิทธิภาพของโมเดล โดยใช้ค่า precision, recall, f1-score, และ accuracy

1.3.9 ในการประเมินความถูกต้องของเครื่องมือสร้าง Jira ticket ที่ใช้บทวิจารณ์ของผู้ใช้แบบอัตโนมัติเป็นข้อมูลนำเข้า ผู้วิจัยจะทำการตรวจสอบแบบ manual ใน dashboard ของ Jira ด้วยตนเอง

1.4 ขั้นตอนและวิธีการดำเนินการ

1.4.1 ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการทำ User Reviews Data Mining บนแอปสโตร์หรือเพลย์สโตร์

1.4.2 ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการทำ Text Similarity

1.4.3 ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการทำ Text Summarization

1.4.4 ศึกษาทฤษฎีที่เกี่ยวข้องกับการทำ Automate Jira Ticket, APIs ของ Jira ที่เปิดให้ใช้งาน

1.4.5 ศึกษาภาษา Python ไลบรารีที่เกี่ยวข้องกับการทำ Data Mining และ Machine Learning

1.4.6 ทำการรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้ที่แบ่งประเภทเรียบร้อยแล้วจากงานวิจัย [22] มาเป็น Training data และ Test data

1.4.7 ทำการเทรนโมเดลโดยใช้ข้อมูลบทวิจารณ์ของผู้ใช้ที่ได้จากขั้นตอนก่อนหน้า

1.4.8 ทดสอบการจำแนกประเภทบทวิจารณ์ของผู้ใช้ของโมเดลที่ได้สร้างขึ้นโดยใช้การแบ่งข้อมูลออกเป็นสองส่วน ส่วนแรกคือ Training data 70 % และส่วนที่สองคือ Test data 30 %

1.4.9 วัดผลประสิทธิภาพในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ โดยใช้ค่า precision, recall, f1-score, และ accuracy

1.4.10 ทำการทดสอบหาค่า Threshold ที่มีประสิทธิภาพสูงที่สุดในการจำแนกบทวิจารณ์ของผู้ใช้สองบทวิจารณ์ใด ๆ ว่ามีความซ้ำซ้อนกันหรือไม่จากการคำนวณหาความคล้ายคลึงกันของบทวิจารณ์ของผู้ใช้

1.4.11 ทำการดึงข้อมูลบทวิจารณ์ของผู้ใช้อันใหม่ที่ยังไม่เคยถูกติดฉลากมาก่อน แล้วทำการแยกประเภท

1.4.12 ทำการคำนวณความคล้ายคลึงกันของบทวิจารณ์ของผู้ใช้ในข้อ 1.4.11 เพื่อกรองบทวิจารณ์ที่ซ้ำซ้อนออก

1.4.13 ทำการสรุปความเนื้อหาบทวิจารณ์ของผู้ใช้ที่มีความยาวมากจากข้อ 1.4.12 เป็นประโยคสั้น ๆ เพื่อใช้เป็นหัวข้อสำหรับทิคเก็ตที่จะสร้างขึ้น รวมไปถึงใช้เป็นรายละเอียดแบบย่อในรายละเอียดของทิคเก็ตด้วย (ถ้ามี)

1.4.14 พัฒนาเครื่องมือที่ใช้ในการสร้าง Jira ticket จากบทวิจารณ์ของผู้ใช้ที่ผ่านกระบวนการข้อ 1.4.11, 1.4.12, 1.4.13 แล้ว

1.4.15 ประเมินความถูกต้อง ครบถ้วน สมบูรณ์ของ Jira ticket ใน Jira dashboard

1.4.16 จัดทำและนำเสนอบทความทางวิชาการ

1.4.17 สรุปผลแนวทางการวิจัย ข้อเสนอแนะ และจัดทำโครงงานฉบับสมบูรณ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้โมเดลการเรียนรู้ที่สามารถจำแนกบทวิจารณ์ของผู้ใช้ของแอปพลิเคชันใด ๆ บนแอปสโตร์หรือเพลย์สโตร์ได้อย่างมี ประสิทธิภาพ โดยแยกเป็นประเภท bug report และ feature request

1.5.2 ได้เครื่องมือสำหรับทำการสร้าง Jira ticket จากบทวิจารณ์ของผู้ใช้ของแอปพลิเคชันใด ๆ บนแอปสโตร์หรือเพลย์สโตร์แยกตามประเภทของบทวิจารณ์ของผู้ใช้

1.5.3 ช่วยลดระยะเวลาในการเก็บรวบรวมความคิดเห็น รายการความต้องการของผู้ใช้งานที่อยู่บนแอปสโตร์หรือเพลย์สโตร์ในรูปแบบของบทวิจารณ์ของผู้ใช้

1.5.4 ช่วยลดระยะเวลาในการสร้าง Jira ticket ลง ส่งผลให้บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระสามารถมุ่งเน้นที่การทำการปรับปรุง แก้ไข เพิ่มเติมแอปพลิเคชันของตนเองให้ดียิ่งขึ้น

1.5.5 ได้ค่า Threshold ที่มีประสิทธิภาพในการจำแนกบทวิจารณ์ของผู้ใช้บนแอปสโตร์หรือเพลย์สโตร์สองบทวิจารณ์ใด ๆ ว่ามีความซ้ำซ้อนกันหรือไม่จากการคำนวณหาความคล้ายคลึงกันของบทวิจารณ์ของผู้ใช้

1.6 **โครงสร้างของเนื้อหาในโครงงานมหาบัณฑิต**

โครงงานมหาบัณฑิตฉบับนี้แบ่งเนื้อหาออกเป็น 6 บทตามรายการดังต่อไปนี้

บทที่ 1 บทนำ

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทที่ 3 การพัฒนาและทดสอบประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้

บทที่ 4 การตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้

บทที่ 5 การสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้

บทที่ 6 บทสรุปโครงงานและข้อจำกัดในการดำเนินการโครงงาน

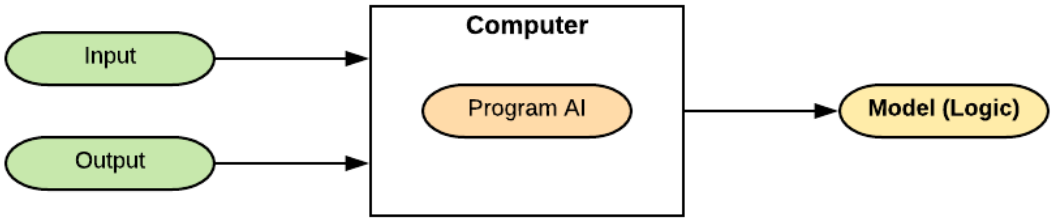
บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การเรียนรู้แบบมีผู้สอน [9]

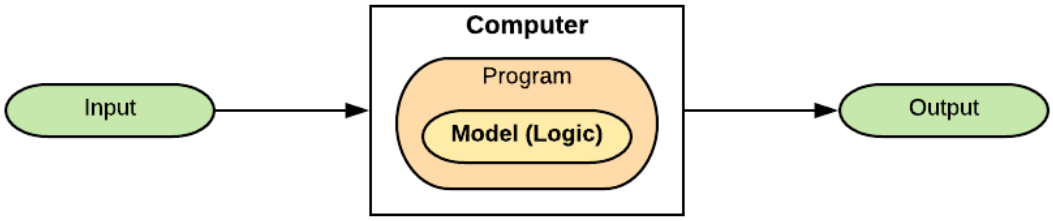
การเรียนรู้แบบมีผู้สอน (Supervised Learning) เป็นศาสตร์หนึ่งทางด้านการเรียนรู้ของเครื่อง (Machine Learning) ที่อยู่ภายใต้ปัญญาประดิษฐ์ (Artificial Intelligence) โดยมีมานานตั้งแต่ช่วงปี 1959 แล้ว แต่ยังไม่ได้รับความนิยมเนื่องจากคอมพิวเตอร์ในสมัยนั้นมีกำลังในการประมวลผลต่ำ การคอมไพล์เพื่อให้ได้ผลลัพธ์ออกมาใช้เวลานาน

หลักการของ Supervised Learning จะแตกต่างจากการเขียนโปรแกรมเพื่อแก้ไขปัญหาทั่วไป ปัญหาทั่วไปจะแก้ได้โดยการเขียน Logic ไว้ภายในโปรแกรม หลังจากนั้นเมื่อมี Input ผ่านเข้ามา logic ก็จะทำการประมวลผลเพื่อให้ได้ Output ออกไป คอมพิวเตอร์เพียงแค่ประมวลผลตามคำสั่งของมนุษย์ที่ได้ระบุไว้ก่อนหน้า แต่สำหรับ Supervised Learning แล้วจะกลับกันตรงที่มนุษย์จะพยายามให้คอมพิวเตอร์เป็นผู้หาคำตอบของปัญหาเอาเองจาก input ที่ให้มา ซึ่งก่อนที่คอมพิวเตอร์สามารถที่จะตอบคำถามได้เองนั้น จะต้องทำการสร้างโมเดลการเรียนรู้จากข้อมูลในอดีตที่มีอยู่แล้วเสียก่อนในปริมาณที่เหมาะสม ตัวอย่างเช่น การสอนให้คอมพิวเตอร์สามารถแยกแยะได้ว่าภาพที่เข้ามาในโปรแกรมเป็นภาพแมวหรือไม่ เราก็จะทำการสอนให้คอมพิวเตอร์รู้จักภาพแมวลักษณะต่าง ๆ เสียก่อน โดยการใส่ภาพแมวเข้าไปในโปรแกรมหลาย ๆ ภาพ พร้อมทั้งระบุว่าภาพเหล่านี้คือภาพแมว จากภาพที่ 2.1 input ก็คือภาพแมวหลาย ๆ ภาพ output ก็คือแมว ผ่านเข้าไปในโปรแกรม หลังจากนั้นจะได้โมเดลที่สามารถแยกแยะได้ว่าภาพใด ๆ เป็นภาพแมวหรือไม่ใช่ภาพแมว



ภาพที่ 2.1 กระบวนการเทรนเพื่อให้ได้โมเดลที่ต้องการ [9]

ขั้นตอนต่อมาก็คือการนำโมเดลที่ได้มาทำการทดสอบประสิทธิภาพ อธิบายโดยอิงจากภาพที่ 2.2 ในบริบทของการทำนายว่าภาพสัตว์ใด ๆ เป็นภาพแมวหรือไม่ ได้คือทำการนำภาพสัตว์หลากหลายประเภทมีทั้งแมวและไม่ใช่แมวใส่เข้าไปในโปรแกรมที่มีโมเดลการจำแนกประเภทอยู่ แล้วดูว่าคอมพิวเตอร์สามารถทำนายว่าเป็นภาพแมวหรือไม่ใช่ภาพแมวได้ถูกต้องหรือไม่



ภาพที่ 2.2 การนำโมเดลที่ผ่านการเรียนรู้แล้วมาลองใช้งานจริง [9]

2.1.2 นาอีฟเบย์ส [10]

นาอีฟเบย์ส (Naïve Bayes) เป็น Supervised Learning ประเภท Classification รูปแบบหนึ่งที่ได้รับความนิยมเป็นอย่างมากในปัจจุบัน เนื่องจากสามารถเทรนโมเดลโดยใช้จำนวนชุดของ Training data ไม่มาก แต่ได้ความแม่นยำในระดับที่น่าพอใจ รวมถึงอิมพลีเมนต์ง่าย หลักการของวิธีการนี้จะใช้การคำนวณความน่าจะเป็นแบบมีเงื่อนไขที่เรียกว่า Conditional Probability ซึ่งแสดงดังสมการ (1)

$$P(A \mid B) = \frac{P\,(A \cap B)}{P\,(B)} \qquad (1)$$

โดย P(A|B) คือ ค่า Conditional Probability หรือค่าความน่าจะเป็นที่เกิดเหตุการณ์ B ขึ้นก่อนและจะมีเหตุการณ์ A ตามมา

P(A ⋂ B) คือ ค่า Joint Probability หรือค่าความน่าจะเป็นที่เหตุการณ์ A และเหตุการณ์ B เกิดขึ้นร่วมกัน

P(B) คือ ค่าความน่าจะเป็นที่เหตุการณ์ B เกิดขึ้น

ในลักษณะเดียวกันเราจะเขียน P(B|A) หรือค่าความน่าจะเป็นที่เหตุการณ์ A เกิดขึ้นก่อนและเหตุการณ์ B เกิดขึ้นตามมาทีหลังได้เป็นสมการ (2)

$$P(B \mid A) = \frac{P\,(B \cap A)}{P\,(A)} \qquad (2)$$

จากทั้งสองสมการจะเห็นว่ามีค่า P(A ⋂ B) ที่เหมือนกันอยู่ ดังนั้นเราสามารถเขียนสมการของ P(A ⋂ B) ได้เป็นดังสมการ (3) และสมการ (4)

$$P\,(A \cap B) = P(A \mid B) \times P(B) = P(B \mid A) \times P(A) \qquad (3)$$

$$P\,(B \mid A) = \frac{P(A \mid B) \times P(B)}{P(A)} \qquad (4)$$

เมื่อนำทฤษฎีของเบย์มาใช้ในงานทางด้าน Data Mining มักจะเปลี่ยนสัญลักษณ์ B เป็น C โดยให้ A คือ แอตทริบิวต์ (Attribute) และ C คือ คลาส (Class) ดังสมการ (5)

$$P(C \mid A) = \frac{P(A \mid C) \times P(C)}{P(A)} \quad (5)$$

โดย Posterior probability หรือ P(C|A) คือ ค่าความน่าจะเป็นที่ข้อมูลที่มีแอตทริบิวต์เป็น A จะมีคลาส C

Likelihood หรือ P(A|C) คือ ค่าความน่าจะเป็นที่ข้อมูล Training data ที่มีคลาส C และมีแอตทริบิวต์ A โดยที่ A = $a_1 \cap a_2 \ldots \cap a_m$ โดยที่ m คือจำนวนแอตทริบิวต์ใน Training data

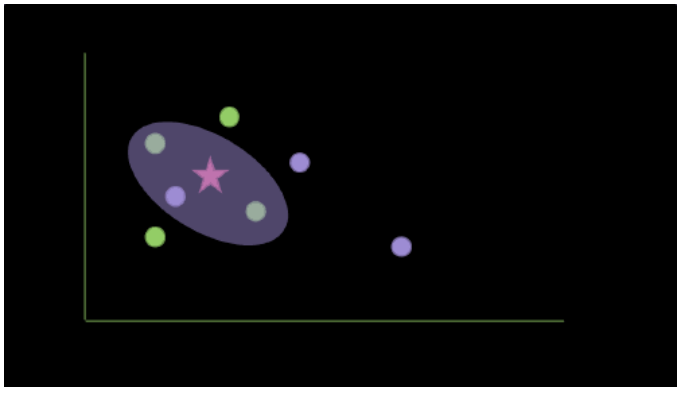
Prior probability หรือ P(C) คือ ค่าความน่าจะเป็นของคลาส C

แต่การที่แอตทริบิวต์ A = $a_1 \cap a_2 \ldots \cap a_m$ ที่เกิดขึ้นใน training data อาจจะมีจำนวนน้อยมากหรือไม่มีรูปแบบของแอตทริบิวต์แบบนี้เกิดขึ้นเลย ดังนั้นจึงได้ใช้หลักการที่ว่าแต่ละแอตทริบิวต์เป็นอิสระต่อกัน ทำให้สามารถเปลี่ยนสมการ P(A|C) ได้เป็นสมการ (6)

$$P(A \mid C) = P(a_1 \mid C) \times P(a_2 \mid C) \times \ldots \times P(a_m \mid C) \quad (6)$$

2.1.3 เคเอ็นเอ็น [11]

เคเอ็นเอ็นเป็นหนึ่งในวิธีการแบ่งกลุ่มของข้อมูล อาศัยหลักการที่ว่าข้อมูลใหม่ที่ยังไม่ถูกติดฉลากจะมีคลาสคำตอบเป็นคลาสเดียวกันกับคลาสที่ปรากฏมากที่สุดของข้อมูลจำนวน k ตัวที่อยู่ใกล้กันมากที่สุดในปริภูมิใด ๆ ซึ่งการเลือกค่า k ให้เหมาะสมมีผลต่อกระทบต่ออัลกอรึทึม ค่า k ที่น้อยเกินไปจะทำให้ความแม่นยำลดลงเนื่องจากสนใจเพียงแค่ข้อมูลเพียงตัวเดียวที่อยู่ใกล้ที่สุด หรือค่า k ที่มากเกินไปก็จะทำให้ครอบคลุมพื้นที่ของข้อมูลที่ไม่เกี่ยวข้องมากตามไปด้วย จากภาพที่ 2.3 ถ้าเลือก k = 1 จะได้ว่ารูปดาวที่พิจารณามีคลาสคำตอบเป็นวงกลมสีม่วง แต่ถ้าเลือก k = 3 จะได้คลาสคำตอบเป็นสีเขียวแทน เนื่องจากอยู่ใกล้วงกลมสีเขียวสองวง แต่อยู่ใกล้วงกลมสีม่วงเพียงหนึ่งวง อัลกอรึทึมจะเลือกคลาสที่ปรากฏมากที่สุดเป็นคลาสคำตอบ



ภาพที่ 2.3 รูปดาวจะมีคลาสคำตอบเป็นวงกลมสีเขียวเมื่อใช้ *k = 3* [11]

สำหรับตัววัดที่ใช้บอกถึงความใกล้ไกลกันคือระยะทางระหว่างข้อมูลสองข้อมูลใด ๆ ในปริภูมิขนาด n มิติเป็นดังสมการ (7)

$$d_E(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (7)$$

โดย $d_E(x, y)$ คือระยะห่างระหว่างข้อมูล x และข้อมูล y ในปริภูมิขนาด n มิติ

n คือจำนวนมิติของข้อมูล x และ y

2.1.4 ต้นไม้การตัดสินใจ [12]

แผนภาพต้นไม้จะทำการเลือกแอตทริบิวต์ที่มีความสัมพันธ์กับคลาสหรือฉลากมากที่สุดมาเป็นโหนดแม่ เพื่อทำการแบ่งข้อมูลออกเป็นกลุ่ม โดยจำนวนกลุ่มเท่ากับจำนวนค่าที่เป็นไปได้ทั้งหมดของแอตทริบิวต์ที่เลือกมาเป็นตัวแบ่ง หลังจากนั้นก็จะทำการหาแอตทริบิวต์ถัดไปเรื่อย ๆ มาแบ่งกลุ่มข้อมูลที่อยู่ในโหนดลูกต่อ จนกระทั่งในแต่ละกลุ่มที่แบ่งออกมานั้นมีคำตอบฉลากเป็นชนิดเดียวกันทั้งหมด หรือแอตทริบิวต์ทุกตัวถูกใช้ในการแบ่งกลุ่มหมดแล้ว ก็จะหยุดแบ่งกลุ่มข้อมูล ซึ่งในการหาความสัมพันธ์ระหว่างแอตทริบิวต์ใด ๆ กับคลาสจะใช้ตัววัด คือ Information Gain (IG) คำนวณได้จากสมการ (8) และ (9)

$$IG\,(parent, child) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \ldots] \quad (8)$$

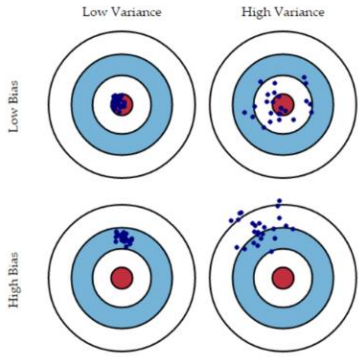
$$entropy(c_1) = -p(c_1)\, log\, p(c_1) \quad (9)$$

เมื่อ P(c) คือ ความน่าจะเป็นที่ข้อมูลจะมีแอตทริบิวต์ c ใด ๆ อยู่ โดยในสมการ (9) c_1 หมายถึงแอตทริบิวต์ตัวแรก

โดยถ้าแอตทริบิวต์ใดมีค่า Information Gain มากที่สุด ก็จะถูกนำมาเป็นโหนดแม่สำหรับการแบ่งกลุ่มข้อมูลในรอบนั้น ข้อดีของโมเดลการจำแนกข้อมูลแบบต้นไม้ตัดสินใจ คือ ไม่จำเป็นต้องใช้ทุกแอตทริบิวต์ในการแบ่งกลุ่มข้อมูล เพราะถ้าเลือกแอตทริบิวต์ที่มีค่า Information Gain สูง ๆ หลายค่า ก็อาจจะเพียงพอต่อการแบ่งกลุ่มข้อมูลทั้งหมดแล้ว

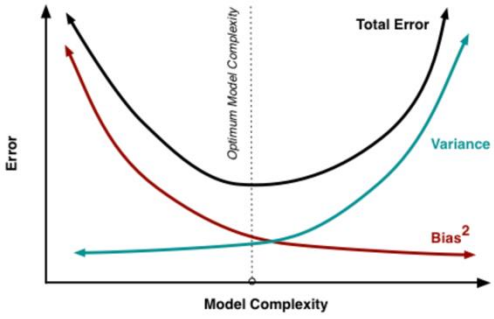
2.1.5 การเรียนรู้แบบรวม [13]

การเรียนรู้แบบรวม (Ensemble Learning) เป็นการปรับปรุงประสิทธิภาพของโมเดลให้ดียิ่งขึ้น ซึ่งถูกใช้ในการแข่งขัน Kaggle ที่เป็นงานเกี่ยวกับ Data Science แนวคิดหลัก ๆ ก็คือ การนำโมเดลที่ผ่านการเทรนด้วยอัลกอริทึมที่แตกต่างกันมาใช้งานร่วมกันเพื่อสร้างเป็นโมเดลใหม่ เนื่องจากโมเดลแต่ละแบบก็จะมีข้อดี ข้อเสีย มี Bias และ Variance แตกต่างกันไป ดังภาพที่ 2.4



ภาพที่ 2.4 ผลลัพธ์การทำนายของโมเดล เมื่อมีค่า *Bias* และ *Variance* แตกต่างกันออกไป โดยจุดสีแดงคือค่าจริง จุดน้ำเงินคือค่าที่ได้จากการทำนายของโมเดล [13]

ดังนั้นการทำ Ensemble Learning ก็คือการหาจุดที่สมดุลกันระหว่าง Bias และ Variance ที่ทำให้เกิดค่า Error น้อยที่สุด ดังภาพที่ 2.5



ภาพที่ 2.5 ความสัมพันธ์ระหว่าง *Model Complexity* และค่า *Error* ที่เกิดขึ้นที่เกี่ยวข้องกับค่า *Bias* และ *Variance* [13]

2.1.6 จิราซอฟต์แวร์ [14]

จิราซอฟต์แวร์คือโปรแกรมติดตามปัญหา และจัดการโครงการพัฒนาซอฟต์แวร์ที่ใช้หลักการเอไจล์ จัดทำขึ้นเพื่อใช้ในการแก้ไขจุดบกพร่องของโปรแกรม ติดตามปัญหา และใช้ในการบริหารโครงการ (Project Management) พัฒนาโดยบริษัท Atlassion ข้อดีของโปรแกรมมีดังนี้

1) สามารถสร้างชิ้นงาน (Task) ได้ โดยสามารถกำหนดระยะเวลา (Estimate) หรือระดับความพยายามที่ใช้ในการพัฒนา (Story Point) และสามารถมอบหมายงานให้แต่ละคนที่อยู่ในทีมได้

2) สามารถแบ่งรอบการทำงาน (Sprint) ได้ ซึ่งในแต่ละรอบจะสามารถวางชิ้นงานได้ใน 3 ส่วนคือ งานที่ต้องทำ (To Do) งานที่กำลังดำเนินอยู่ (In Progress) และงานที่เสร็จแล้ว (Done) โดยระหว่างที่รอบการทำงานยังไม่จบ สามารถที่จะลากชิ้นงานไปมาในระหว่างแต่ละส่วนได้ตลอด หากมีการแก้ไขเกิดขึ้น

3) เมื่อจบรอบการทำงานแล้ว จะมีแผนภูมิในแบบต่าง ๆ (Charts) สรุปผลการทำงานให้ ตามหลักของเอไจล์แล้ว นิยมใช้ Burndown Chart

4) มี Plug-ins และเปิด APIs ให้นักพัฒนาสามารถเชื่อมต่อระบบของตนเองเข้ามายัง Jira ได้ ซึ่งผู้วิจัยจะใช้ APIs ของ Jira ในการทำการสร้างทิคเก็ตจากบทวิจารณ์ของผู้ใช้ที่ได้จากการเรียนรู้ด้วยเครื่องแบบอัตโนมัติ

2.1.7 ระบบติดตามปัญหา [15]

ระบบติดตามปัญหา หมายถึงระบบ ซอฟต์แวร์ หรือโปรแกรมที่เกี่ยวข้องกับการบริหารจัดการโครงการซอฟต์แวร์ต่าง ๆ ตั้งแต่เริ่มต้นการเก็บรวบรวมความต้องการของผู้ใช้งาน การวิเคราะห์ความต้องการของซอฟต์แวร์ การออกแบบระบบซอฟต์แวร์ การอิมพลีเมนต์ซอฟต์แวร์ การทดสอบซอฟต์แวร์ การจำหน่ายซอฟต์แวร์ รวมไปถึงการบำรุงรักษาซอฟต์แวร์หลังจากที่มีการใช้งานจริงแล้ว โดยสิ่งที่จะ Track ก็คือสถานะทำงานของขั้นตอนต่าง ๆ ว่าเป็นอย่างไร มีความคืบหน้าถึงส่วนใดแล้ว โดย Issue ในที่นี้หมายถึง งานใด ๆ ที่ต้องทำซึ่งจะมีระดับความสำคัญประกอบไว้ นอกจากนี้ใน Issue ยังมีรายละเอียดเกี่ยวกับกำหนดวันที่ต้องส่ง รายละเอียดของงานที่ต้องทำหรือปัญหาที่ต้องแก้ไข วิธีการแก้ปัญหาที่ใช้ เป็นต้น Issue สามารถเป็นไปได้หลายประเภท เช่น feature request และ bug report เป็นต้น

2.1.8 ทิคเก็ต [15]

ทิคเก็ต หมายถึง สิ่งที่ระบุ (Artifact) ถึงงานที่ต้องทำในรอบการทำงานใด ๆ ของการพัฒนาซอฟต์แวร์แบบเอไจล์ โดยสามารถแบ่งประเภทของทิคเก็ต (Issue Type) ได้ดังนี้

1) สตอรี (Story) หมายถึง คำอธิบายถึงความต้องการในการใช้งานซอฟต์แวร์ในมุมมองของผู้ใช้งาน เช่น iPhone users need access to a vertical view of the live feed when using the mobile app เป็นต้น

2) งาน (Epic) หมายถึง รายละเอียดใด ๆ ซึ่งสามารถนำไปแตกเป็นงานย่อย ๆ ได้

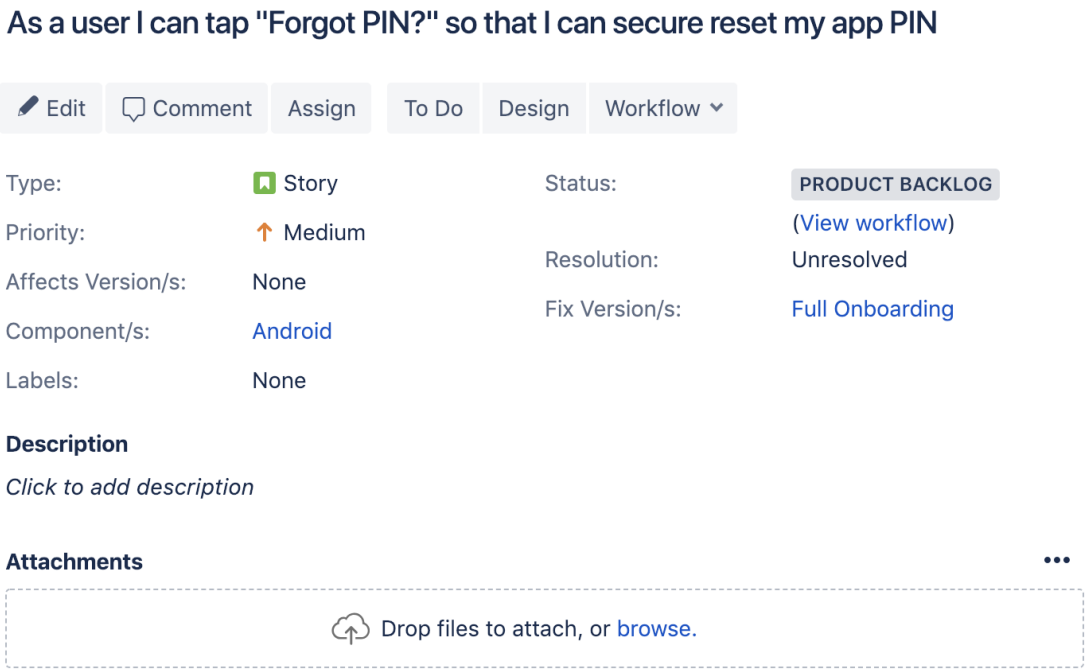
3) งานย่อย (Task) หมายถึง สิ่งที่ต้องทำเพื่อให้บรรลุสตอรีที่อยู่ภายใต้งานใหญ่

4) จุดบกพร่อง (Bug) หมายถึง สิ่งที่แสดงออกมาบ่งบอกว่าซอฟต์แวร์ไม่สามารถทำงานได้อย่างถูกต้องตามความต้องการที่ระบุไว้

5) การปรับปรุง (Improvement) หมายถึง การพัฒนาให้ดีขึ้น

6) ความสามารถ (Feature) หมายถึง ฟังก์ชันการทำงานใด ๆ ที่ซอฟต์แวร์สามารถทำงานเพื่อตอบสนองความต้องการของผู้ใช้ได้

ตัวอย่างสตอรีดังภาพที่ 2.6



ภาพที่ 2.6 สตอรีที่บอกว่าผู้ใช้งานสามารถกดปุ่มลืมรหัสผ่านเพื่อทำการรีเซ็ตรหัสผ่านใหม่ได้

2.1.9 โปรดักส์แบล็คล็อก [16]

โปรดักส์แบล็คล็อก (Product Backlog) หมายถึง งานใด ๆ ที่ถูกสร้างขึ้นที่ต้องทำในรอบการทำงานใด ๆ แต่ยังไม่ถูกนำมาทำในรอบการทำงานปัจจุบัน (Sprint Backlog)

2.1.10 เอไจล์บอร์ด [17]

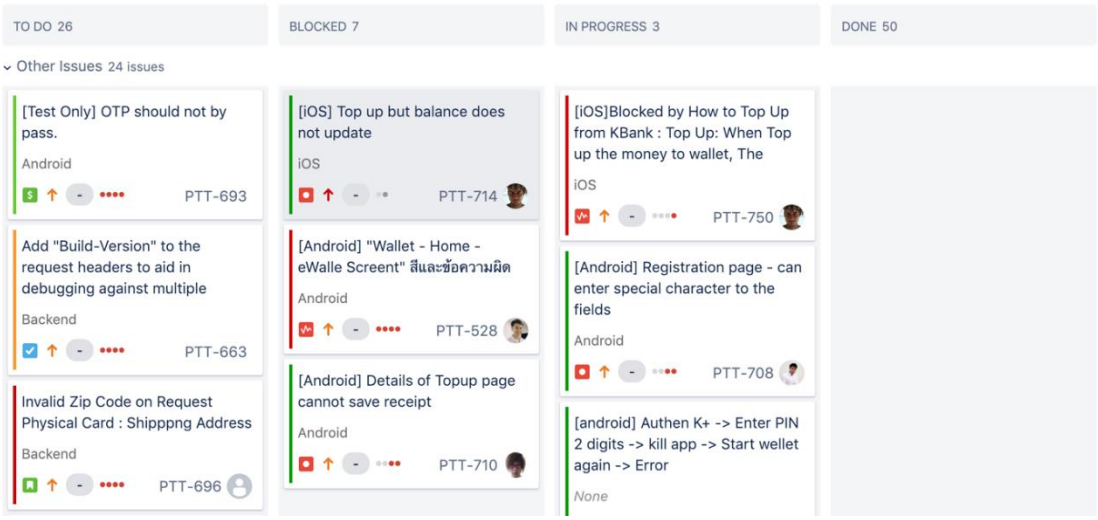
เอไจล์บอร์ด (Agile Board) คือ สิ่งที่ระบุถึงภาพรวมของการทำงานในรอบการทำงานปัจจุบัน โดยทั่วไปแล้วจะอยู่ในรูปแบบตารางที่บรรจุไปด้วยงานย่อยต่าง ๆ ปกติตารางจะมีสามคอลัมน์ แต่อาจจะมีมากกว่านี้ก็ได้ขึ้นอยู่กับการตกลงกันของผู้ที่เกี่ยวข้องในโครงการซอฟต์แวร์ใด ๆ ซึ่งสามคอลัมน์หลัก เป็นดังนี้

1) To Do หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่ยังไม่ได้ทำในรอบการทำงานปัจจุบัน

2) In Progress หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่กำลังทำอยู่ในรอบการทำงานปัจจุบัน

3) Done หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่ทำเสร็จสิ้นแล้วในรอบการทำงานปัจจุบัน

โดยในระหว่างรอบการทำงานปัจจุบันนั้น นักพัฒนาซอฟต์แวร์สามารถที่จะทำการเคลื่อนย้ายทิคเก็ตไปตามคอลัมน์ต่าง ๆ ตามสถานะการทำงานได้ ตัวอย่างเอไจล์บอร์ด ดังภาพที่ 2.7



ภาพที่ 2.7 เอไจล์บอร์ดที่มีสี่คอลัมน์ คือ *To Do, Blocked, In Progress,* และ *Done*

2.1.11 ความเหมือนกันของข้อความ [18]

ความเหมือนกันของข้อความ (Text Similarity) คือ ความคล้ายคลึงกันของข้อความสองข้อความใด ๆ หลักการคือการแปลงตัวอักษรทั้งหมดในข้อความให้เป็นชุดของตัวเลข จากนั้นนำชุดของตัวเลขสองชุดใด ๆ มาทำการคำนวณทางคณิตศาสตร์เพื่อให้ได้ผลลัพธ์ออกมาเป็นตัวเลขที่บ่งบอกถึงความคล้ายคลึงกัน โดยทั่วไปแล้วผลลัพธ์จะมีค่าอยู่ระหว่าง 0 ถึง 1 โดย 0 หมายถึงมีความคล้ายคลึงกันน้อยที่สุด และ 1 หมายถึงมีความคล้ายคลึงกันมากที่สุด ตัวอย่างวิธีการหา Text Similarity คือ Cosine Similarity, Euclidean Distance, Jaccard Similarity เป็นต้น โดย Text Similarity แบ่งออกเป็น Syntactic Text Similarity คือ การหาความคล้ายคลึงกันของข้อความโดยไม่สนใจความหมายของประโยค สนใจแต่เพียงโครงสร้างและลำดับของคำที่เรียงกันในประโยคเท่านั้น ส่วน Sematic Text Similarity คือ การหาความคล้ายคลึงกันของข้อความโดยดูที่ความหมายของประโยคเป็นหลัก

2.1.12 การสกัดข้อความ [19]

การสกัดข้อความ (Text Summarization) คือ การสรุปความของเนื้อหาในย่อหน้าที่มีความยาวมาก ให้ออกมาเป็นประโยคที่เป็นตัวแทนของย่อหน้านั้น ๆ โดยหลัก ๆ แล้วแบ่งออกเป็นสองประเภทคือ Extractive Text Summarization และ Abstractive Text Summarization โดย Extractive Text Summarization เป็นการหาประโยคที่สำคัญมากที่สุดในย่อหน้านั้น ๆ แล้วนำมาเป็นตัวแทนโดยไม่ได้ทำการเปลี่ยนแปลงข้อความในประโยค หลักการในการหาส่วนใหญ่จะใช้วิธีดูจากความถี่ของคำที่แตกต่างกันในย่อหน้าใด ๆ โดยถ้าประโยคใดมีผลรวมของความถี่เฉลี่ยของคำที่แตกต่างกันในย่อหน้ามากที่สุด ก็แปลว่าประโยคนั้น ๆ มีคำที่สำคัญของย่อหน้านั้นอยู่ในประโยคมาก จึงน่าจะเป็นตัวแทนของย่อหน้าได้ ส่วน Abstractive Text Summarization คือ การสร้างประโยคใหม่ที่สั้นลงแต่ยังคงมีความหมายครอบคลุมเนื้อหาทั้งหมดในย่อหน้านั้น ๆ โดยกระบวนการทำต้องอาศัยเรื่องของ Natural Language Processing, Linguistics, Deep learning มาก เพื่อให้ได้ประโยคที่ดีที่สุดออกมาเป็นตัวแทนของย่อหน้า

2.1.13 ยูนิเวอร์ซัล เซนเทนต์ เอนโค้ดเดอร์ [20]

Universal Sentence Encoder (USE) คือ หนึ่งในวิธีการที่มีประสิทธิภาพในการแปลงตัวอักษรที่อยู่ในรูปแบบคำ ประโยค ย่อหน้า หรือเอกสารให้กลายเป็นเวกเตอร์ของจำนวนจริงที่มีขนาด 512 มิติเท่ากันโดยไม่ขึ้นอยู่กับขนาดของเอกสาร ซึ่งถูกพัฒนาโดย Google และอยู่บนพื้นฐานของโมเดลสองประเภท คือ Transformer และ Deep Averaging Network (DAN) โดยทั้งสองโมเดลนี้ถูกเทรนด้วย corpus ที่มีขนาดใหญ่ เมื่อต้องการใช้ USE ให้ทำการส่งข้อมูลนำเข้าเป็นคำ ประโยค ย่อหน้า หรือเอกสาร และจะได้ข้อมูลนำออกมาเป็นเวกเตอร์ที่มีขนาด 512 มิติ ภาพที่ 2.8 แสดงคุณสมบัติของโมเดลทั้งสองประเภทที่ถูกนำมาทำ USE

	Transformer model	**Deep Averaging Network (DAN) model**
Vector Length	512	512
Encoding time with sentence length	Non-Linear	Linear
Memory usage	High	Medium
Accuracy	Very High	High

ภาพที่ 2.8 คุณสมบัติของโมเดลแต่ละประเภทที่ถูกนำมาทำ *Universal Sentence Encoder* [20]

2.1.14 โคซายน์ ซิมิลาริตี [21]

Cosine Similarity คือ เมตริกที่ใช้ในการวัดความใกล้กันของสองข้อมูลใด ๆ โดยไม่ขึ้นอยู่กับขนาดของข้อมูล เช่น การหาความคล้ายกันของสองข้อความ วิธีการคือ นำเวกเตอร์ของสองข้อมูลใด ๆ ที่อยู่ในปริภูมิ n มิติมาทำการ dot product เพื่อหาค่ามุมที่สองเวกเตอร์ใด ๆ กระทำต่อกัน โดยถ้ามุมมีค่าน้อยหมายถึงสองเวกเตอร์มีความใกล้กัน แต่ถ้ามุมมีค่ามากหมายถึงสองเวกเตอร์อยู่ห่างกัน โดยสามารถหาค่า Cosine ของสองเวกเตอร์ใด ๆ ได้ดังสมการ (10)

$$\cos\theta = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \qquad (10)$$

เมื่อ $\cos\theta$ คือ ค่า Cosine ระหว่างสองเวกเตอร์ใด ๆ

a คือ เวกเตอร์ที่หนึ่ง

b คือ เวกเตอร์ที่สอง

n คือ จำนวนมิติของเวกเตอร์

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews [22]

งานวิจัยนี้ได้นำเสนอประเภทของบทวิจารณ์ของผู้ใช้ที่เก็บรวบรวมมาจากแอปสโตร์และเพลย์สโตร์ แล้วใช้เทคนิคทาง Data Mining และ Machine Learning เพื่อจัดประเภทของบทวิจารณ์ของผู้ใช้ออกมาเป็นกลุ่ม ตัวอย่างเทคนิคที่ใช้ เช่น Text Classification, Natural Language Processing, Sentiment Analysis เป็นต้น โดยสามารถแบ่งบทวิจารณ์ของผู้ใช้ออกได้เป็น 4 ประเภท ดังนี้

1) รายงานจุดบกพร่อง (Bug Reports)

2) รายการความต้องการทางฟังก์ชันการทำงาน (Feature Requests)

3) ประสบการณ์การใช้งานแอปพลิเคชัน (User Experiences)

4) ความพึงพอใจในการใช้งาน (Ratings)

โดยเพื่อให้มั่นใจว่าผลลัพธ์การแบ่งกลุ่มที่ได้มีความถูกต้องสูง ทางกลุ่มผู้วิจัยได้เลือกบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ มาอย่างละ 2,200 ความคิดเห็นตามลำดับ โดยมีทั้งเลือกแบบสุ่ม และเลือกแบบเจาะจงมาให้ผู้มีประสบการณ์ด้านเขียนโปรแกรมบอกว่าบทวิจารณ์ของผู้ใช้นี้เป็นบทวิจารณ์ประเภทใด โดยหนึ่งบทวิจารณ์ของผู้ใช้จะถูกระบุโดยคนสองคน แล้วเอาผลลัพธ์มาเปรียบเทียบกันว่าได้ประเภทของบทวิจารณ์ของผู้ใช้ตรงกันหรือไม่ ถ้าไม่ตรงกันเพราะสาเหตุใด โดยทำผ่านเครื่องมือการติดฉลากที่ทางผู้วิจัยได้พัฒนาขึ้นและคู่มือสำหรับการรีวิว ซึ่งหลังจากได้บทวิจารณ์ของผู้ใช้ที่ถูกระบุประเภทเรียบร้อยมาทั้งสิ้น 4,400 ความคิดเห็นแล้ว ผู้วิจัยได้ทำการแบ่งข้อมูล 70 % มาเป็น Training set เพื่อทำการเทรนโมเดลด้วยเทคนิคต่าง ๆ ที่กล่าวไปแล้วข้างต้น และข้อมูล 30 % เป็น Test set เพื่อใช้สำหรับวัดผลโมเดลที่ผ่านการเทรนมาว่ามีประสิทธิภาพมากน้อยเพียงใด ซึ่งผลลัพธ์ที่ได้คือ ค่า precision ประมาณ 70-95 % และค่า recall ประมาณ 80-90% นอกจากนี้ยังมีการนำค่า f1-score มาช่วยในการเปรียบเทียบประสิทธิภาพของโมเดลอีกด้วย ดังภาพที่ 2.9

Classification techniques	Bug Reports			Feature Requests			User Experiences			Ratings		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Basic (string matching)	0.58	0.24	0.33	0.39	0.55	0.46	0.27	0.12	0.17	0.74	0.56	0.64
Document Classification (& NLP)												
Bag of words (BOW)	0.70	0.71	0.70	0.73	0.68	0.70	0.71	0.79	0.75	0.86	0.69	0.77
BOW + lemmatization	0.66	0.75	0.70	0.71	0.68	0.69	0.69	0.78	0.74	0.87	0.67	0.76
BOW - stopwords	**0.71**	**0.72**	**0.72**	**0.76**	0.68	0.72	0.69	0.78	0.74	**0.86**	**0.71**	**0.78**
BOW + lemmatization - stopwords	0.70	0.71	0.70	0.73	0.69	0.71	0.74	0.72	0.73	0.85	0.66	0.74
Metadata												
Rating + length	0.45	0.84	0.59	0.46	0.86	0.60	0.69	**0.82**	0.75	0.63	0.22	0.33
Rating + length + tense	0.46	0.87	0.60	0.48	0.86	0.62	0.69	0.81	0.74	0.70	0.27	0.39
Rating + length + tense + 1x sentiment	0.46	**0.88**	0.61	0.49	0.84	0.62	0.68	0.79	0.73	0.78	0.21	0.34
Rating + length + tens + 2x sentiments	0.46	0.88	0.61	0.49	0.82	0.62	0.68	0.79	0.73	0.80	0.21	0.34
Combined (text and metadata)												
BOW + rating + 1x sentiment	**0.65**	**0.79**	**0.72**	0.70	0.72	0.71	**0.80**	**0.80**	**0.80**	**0.90**	**0.67**	**0.77**
BOW + rating + 1sentiment + tense	0.70	0.71	0.70	0.73	0.68	0.70	0.71	0.79	0.75	0.86	0.69	0.77
BOW - stopwords + lemmatization + rating + 1x sentiment + tense	0.62	**0.85**	**0.72**	**0.71**	**0.79**	**0.75**	**0.81**	0.76	0.78	0.88	0.62	0.73
BOW - stopwords + lemmatization + rating + 2x sentiments + tense	0.68	0.72	0.70	0.73	0.69	0.71	0.74	0.72	0.73	0.85	0.66	0.74

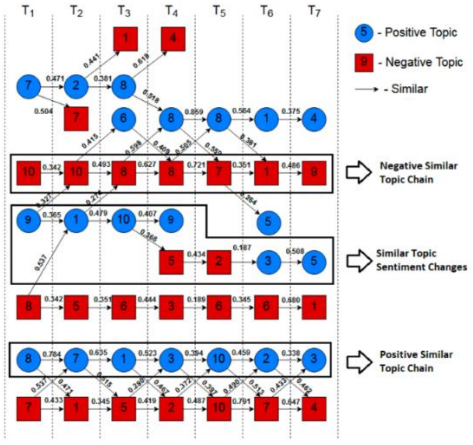
ภาพที่ 2.9 การเปรียบเทียบประสิทธิภาพของโมเดลโดยใช้เทคนิคของ *Naïve Bayes* ต่าง ๆ กับบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ที่ได้เลือกมา [22]

ผู้วิจัยจะนำข้อมูลบทวิจารณ์ของผู้ใช้ที่งานวิจัยนี้ใช้มาเป็น Training set เนื่องจากมีการทำฉลากไว้แล้ว ข้อมูลมีความน่าเชื่อถือเนื่องจากมี methodology ในการติดฉลากที่รัดกุม เพื่อใช้เป็นข้อมูลนำเข้าไปในการเรียนรู้แบบ Supervised Learning ให้กับโมเดล และใช้เป็นประเภทของบทวิจารณ์ของผู้ใช้ที่โครงงานมหาบัณฑิตจะทำการแยกประเภท รวมทั้งเลือกวิธีการจัดประเภทบทวิจารณ์ของผู้ใช้ที่มีประสิทธิภาพสูงที่สุดของงานวิจัยนี้ เพื่อนำมาเปรียบเทียบกับวิธีการจัดประเภทของบทวิจารณ์ของผู้ใช้ในโครงงานมหาบัณฑิต

ข้อแตกต่างระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้มุ่งเน้นไปที่การจัดประเภทของบทวิจารณ์ของผู้ใช้ว่ามีประเภทอะไรบ้าง โดยใช้เทคนิค วิธีการ และอัลกอริทึมต่าง ๆ ทาง Data Mining และ Machine Learning แล้วนำผลลัพธ์ที่ได้มาเปรียบเทียบกันว่าวิธีการใดให้ผลลัพธ์ได้มีประสิทธิภาพมากที่สุดในการจัดประเภทบทวิจารณ์ของผู้ใช้ นอกจากนี้ยังแนะนำถึงประโยชน์ของการนำผลลัพธ์ที่จัดประเภทได้ไปใช้งานในการสร้างเครื่องมือวิเคราะห์การใช้งานแอปพลิเคชันของผู้ใช้งาน เช่น การเลือกจำนวน bug report ทั้งหมดออกมาเพื่อส่งต่อให้นักพัฒนาทำการแก้ไข เป็นต้น ส่วนโครงงานมหาบัณฑิตมุ่งเน้นไปที่การจัดประเภทของบทวิจารณ์ของผู้ใช้ให้อยู่ในกลุ่ม feature request และ bug report ให้ได้อย่างแม่นยำ มีประสิทธิภาพสูง เพื่อนำไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของ Issue ในทิคเก็ตจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้ที่จัดกลุ่มได้

2.2.2 Mobile App Evolution Analysis based on User Reviews [23]

งานวิจัยนี้ได้ศึกษาเกี่ยวกับการนำบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันจากบนสโตร์มาทำการวิเคราะห์ถึงแนวโน้มความคิดเห็นของผู้ใช้งานที่มีต่อแอปพลิเคชัน ผู้ใช้งานชอบหรือไม่ชอบแอปพลิเคชันในเรื่องใดบ้าง เช่น ฟังก์ชันการทำงาน ความยากง่ายในการใช้งาน จุดบกพร่องต่าง ๆ ที่พบในแอปพลิเคชัน รวมไปถึงฟังก์ชันการทำงานที่ต้องการให้มีการเพิ่มหรือปรับปรุงให้ดีขึ้น เป็นต้น โดยวิเคราะห์ในระดับแต่ละเวอร์ชันของแอปพลิเคชัน และนำผลที่ได้มาเปรียบเทียบกันว่า ก่อนอัปเดตแอปพลิเคชันและหลังอัปเดตแอปพลิเคชัน ผู้ใช้งานมีความชอบหรือไม่ชอบในเรื่องใดเปลี่ยนแปลงไปบ้าง และได้เลือกแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ที่ชื่อว่า Whatsapp จากเพลย์สโตร์มาทำการทดลอง เวอร์ชันของแอปพลิเคชันที่ใช้สังเกตมีการอัปเดตแบบ major ทั้งสิ้น 7 ครั้ง และ minor 39 ครั้ง ระยะเวลาอยู่ระหว่างเดือนกันยายนปี 2016 ไปจนถึงเดือนสิงหาคมปี 2017 ดังภาพที่ 2.10



ภาพที่ 2.10 ตัวอย่างกราฟการจัดกลุ่มความคิดเห็นของผู้ใช้งานโดยใช้โมเดลของผู้วิจัยของแอปพลิเคชันแอนดรอยด์ *Whatsapp* ที่มีการอัปเดต *major* ทั้งสิ้น *7* ครั้ง และ *minor 39* ครั้ง [23]

ผู้วิจัยได้ใช้ทฤษฎี Data Mining และ Machine Learning มาทำการกรอง จัดกลุ่ม วิเคราะห์ และหาความสัมพันธ์ของข้อมูลหลายทฤษฎี เช่น Natural Language Processing, Sentiment Analysis, Supervised Learning, Naïve Bayes Classifier, Latent Dirichlet Allocation โดยเฉพาะในการหาความสัมพันธ์ระหว่างบทวิจารณ์ของผู้ใช้สองบทวิจารณ์ใด ๆ ทางผู้วิจัยได้ปรับปรุง Jaccard Similarity โดยนำทฤษฎีของความน่าจะเป็นมาประยุกต์ใช้ เรียกว่า Jaccard Extended Similarity ซึ่งหลักการสำคัญของงานวิจัยนี้คือการจัดกลุ่มข้อมูลเป็นสองกลุ่มหลัก ๆ คือ ความคิดเห็นเชิงบวก และความคิดเห็นเชิงลบ แล้วนำแต่ละกลุ่มมาทำการวิเคราะห์ รวมถึงจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็นประเภทต้องการฟังก์ชันการทำงานใหม่ ต้องการให้แก้ไขจุดบกพร่อง ต้องการให้ประสบการณ์ใช้งานง่ายขึ้น ต้องการให้แก้ไขจุดบกพร่องใด ๆ และอื่น ๆ อีกมากมาย รวมไปถึงมีการนำโมเดลการจัดกลุ่มและวิเคราะห์ข้อมูลที่ได้คิดค้นขึ้นไปทำกรณีศึกษาอีกด้วย

ผู้วิจัยจะนำทฤษฎี Naïve Bayes ที่งานวิจัยนี้ใช้ในการจำแนกประเภทของความคิดเห็นว่าเป็นความคิดเห็นเชิงบวก หรือความคิดเห็นเชิงลบ มาใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้ในโครงงานมหาบัณฑิตเช่นเดียวกัน สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้สนใจศึกษาเกี่ยวกับภาพรวมและแนวโน้มของบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันแอนดรอยด์ที่ชื่อว่า Whatapps ก่อนและหลังอัปเดตเวอร์ชันซอฟต์แวร์ ในขณะที่โครงงานมหาบัณฑิตสนใจเกี่ยวกับการจัดประเภทของบทวิจารณ์ของผู้ใช้แอปพลิเคชันใด ๆ และนำไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของ Issue ในทิคเก็ตจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้ที่โมเดลจำแนกได้

2.2.3 What Do Mobile App Users Complain About? [24]

งานวิจัยนี้ทำการศึกษาเกี่ยวกับประเภทของบทวิจารณ์เชิงลบของผู้ใช้งานแอปพลิเคชัน iOS บนแอปสโตร์ว่ามีอะไรบ้าง โดยทำการจัดลำดับความสำคัญเอาไว้ด้วยเพื่อประโยชน์ในการเลือกพิจารณาบทวิจารณ์เชิงลบกลุ่มที่มีผลต่อภาพรวมของแอปพลิเคชันมากที่สุด เพื่อจะได้รีบทำการแก้ไขโดยเร็ว เพราะบางบริษัทอาจจะมีทรัพยากรหรือบุคลากรทางด้าน Quality Assurance จำกัด จึงต้องบริหารจัดการให้ดี

งานวิจัยนี้เลือกแอปพลิเคชันบนแอปสโตร์ที่ครอบคลุมทุกหมวดหมู่ของแอปพลิเคชันมากที่สุดมา 20 แอปพลิเคชัน แบ่งเป็นกลุ่มที่ rating เฉลี่ยมากกว่าหรือเท่ากับ 3.5 จำนวน 10 แอปพลิเคชัน และกลุ่มที่ rating เฉลี่ยน้อยกว่า 3.5 จำนวน 10 แอปพลิเคชัน หลังจากนั้นทำการเก็บรวบรวมเฉพาะบทวิจารณ์ของผู้ใช้งานแต่ละแอปพลิเคชันที่ให้หนึ่งหรือสองดาว เพราะบทวิจารณ์ที่ได้หนึ่งหรือสองดาว มักจะมีคำอธิบายหรือการพูดถึงที่ค่อนข้างมีประโยชน์ในการนำมาจัดกลุ่ม กลุ่มผู้วิจัยก็จะทำการตรวจสอบบทวิจารณ์โดยอาศัยการอ่าน ทำความเข้าใจ และแบ่งประเภทของบทวิจารณ์ด้วยตนเอง ซึ่งจะทำไล่ทีละแอปพลิเคชันไปเรื่อย ๆ ถ้าพบหมวดหมู่ใหม่ ก็จะมีการบันทึกเพิ่มเข้าไป สุดท้ายได้ผลลัพธ์ทั้งสิ้น 13 ประเภทเรียงตามลำดับความถี่ที่พบ ดังภาพที่ 2.11

TABLE 3

The most frequent and impactful complaint types.*

Complaint type	Most frequent		Most impactful	
	Rank	Median (%)	Rank	1:2 star ratio†
Functional Error	1	26.68	7	2.10
Feature Request	2	15.13	12	1.28
App Crashing	3	10.51	4	2.85
Network Problem	4	7.39	6	2.25
Interface Design	5	3.44	10	1.50
Feature Removal	6	2.73	3	4.23
Hidden Cost	7	1.54	2	5.63
Compatibility	8	1.39	5	2.44
Privacy and Ethics	9	1.19	1	8.56
Unresponsive App	10	0.73	11	1.40
Uninteresting Content	11	0.29	9	1.50
Resource Heavy	12	0.28	8	2.00
Not Specific	—	13.28	—	3.80

** All results are at the 95 percent confidence level.*

† This column indicates the ratio of one- to two-star ratings across all apps.

ภาพที่ 2.11 ประเภทบทวิจารณ์เชิงลบของผู้ใช้งานแอปพลิเคชันบนแอปสโตร์ โดยจัดลำดับตามความถี่ของการพบปัญหาจากมากไปน้อยทั้งสิ้น *13* ประเภท [24]

จากตารางที่ 3.2 ด้านบน จะพบว่า Functional Error, Feature Request, App Crashing เป็นความคิดเห็นเชิงลบสามลำดับแรกที่ผู้ใช้งานพูดถึงมากที่สุด โดยรวมกันแล้วครอบคลุมกว่า 50 % ของบทวิจารณ์เชิงลบทั้งหมด งานวิจัยนี้ไม่ได้ทำการ Train Classification Model แต่อย่างใด เพียงแต่ใช้วิธีการเก็บข้อมูลปกติ จากนั้นนำหลักการทางสถิติมาจัดอันดับเรื่องที่คนพูดถึงในแง่ลบมากที่สุด 13 อันดับแรก ซึ่งกระบวนการตั้งแต่ต้นจนจบกระทำโดยกลุ่มผู้วิจัยแบบ manual เอง

สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้ไม่ได้ใช้ศาสตร์ทางด้าน Data Mining หรือ Machine Learning มาจัดประเภทของบทวิจารณ์ของผู้ใช้ ไม่ได้มีการนำองค์ความรู้ที่วิเคราะห์ได้ไปต่อยอดในเชิงอุตสาหกรรม ซึ่งโครงงานมหาบัณฑิตจะครอบคลุมทั้งในสองส่วนนี้ คือ การจัดประเภทของบทวิจารณ์ของผู้ใช้โดยใช้ศาสตร์ทางด้าน Data Mining และ Machine Learning และนำไปสร้างทิคเก็ตบน Jira เพื่อช่วยลดระยะเวลาในการสร้างทิคเก็ตให้บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระที่ใช้วงจรการพัฒนาแบบเอไจล์ได้

2.2.4 App Store Mining is Not Enough [25]

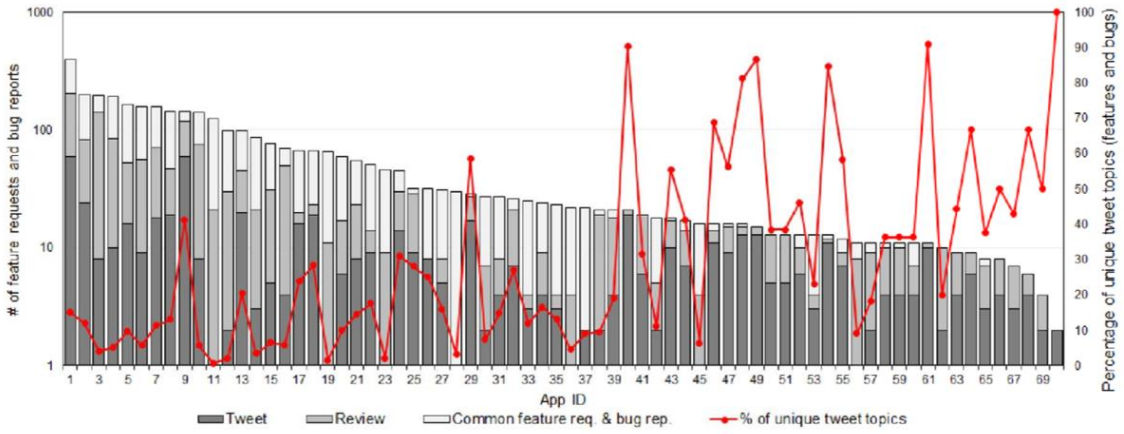
งานวิจัยนี้ได้สำรวจเกี่ยวกับบทวิจารณ์ของผู้ใช้งานแอปพลิเคชัน iOS ที่ไม่ได้อยู่บนแอปสโตร์ แต่อยู่บน Social Media อื่น ๆ และเพลย์สโตร์ ตัวอย่าง Social Media ที่มีจำนวนผู้ใช้งานและจำนวนบทวิจารณ์สูง คือ ทวิตเตอร์ (Twitter) เป็นต้น โดยผู้วิจัยระบุว่า การสำรวจบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันเพื่อดูถึงสิ่งที่ผู้ใช้งานต้องการเพิ่ม หรือสิ่งที่ต้องการให้แก้ไขบนแอปสโตร์อย่างเดียวนั้นไม่เพียงพอ เนื่องจากผู้ใช้งานบางคนไม่ได้มาให้

คะแนนและแสดงความคิดเห็นในแอปสโตร์ แต่ได้ไปโพสต์ผ่านทวิตเตอร์แทน จึงมีความเป็นไปได้สูงว่าข้อมูลการ Tweets จะมีความสำคัญและสัมพันธ์กับข้อมูลที่อยู่บนแอปสโตร์ด้วย

ผู้วิจัยได้ทำการเลือกแอปพลิเคชันบนแอปสโตร์มา 70 แอปพลิเคชัน หลังจากนั้นจึงไปทำการ Mining Tweets และบทวิจารณ์บนเพลย์สโตร์ที่เกี่ยวข้องกับทั้ง 70 แอปพลิเคชัน ภายในระยะเวลาหกสัปดาห์ เพื่อดูว่ามีบทวิจารณ์ของผู้ใช้ใดบ้าง ที่มีความเกี่ยวข้องกับบทวิจารณ์ของผู้ใช้งานบนแอปสโตร์ หรือบทวิจารณ์ของผู้ใช้ใดบ้างที่ปรากฏบนทวิตเตอร์หรือเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์ โดยใช้หลักการ Cosine Similarity วิเคราะห์บทวิจารณ์ของผู้ใช้สองบทวิจารณ์ใด ๆ บนทวิตเตอร์ แอปสโตร์ และเพลย์สโตร์ ว่าถ้ามีค่ามากกว่า 0.6 แปลว่า บทวิจารณ์ของผู้ใช้อันนี้ปรากฏจากทั้งสองแหล่งข้อมูลใด ๆ ที่นำมาเปรียบเทียบกัน หลังจากนั้นจึงทำการตรวจสอบโดยให้คนมาทำการวิเคราะห์แบบ manual อีกครั้งหนึ่ง จากภาพที่ 2.12 พบว่า

1) มีจำนวน feature request 22.4 % และ bug report 12.89 % ที่ปรากฏบนทวิตเตอร์หรือเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์

2) Feature request 43.51 % และ bug report 56.61 % ที่ปรากฏบนทวิตเตอร์และเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์ตามลำดับ จากข้อมูลดังกล่าวจึงทำให้ผู้วิจัยสรุปว่าบทวิจารณ์ของผู้ใช้งานที่อยู่บนแอปสโตร์เพียงอย่างเดียวไม่เพียงพอ จำเป็นต้องใช้ข้อมูลจากแหล่งอื่นเพื่อทำให้การปรับปรุงคุณภาพของแอปพลิเคชันดียิ่งขึ้น



ภาพที่ 2.12 เปรียบเทียบจำนวน *feature request* และ *bug report* ของทั้ง *70* แอปพลิเคชันที่ผู้วิจัยได้เลือกมาแยกตามแหล่งที่มาของข้อมูล คือ ทวิตเตอร์ สโตร์ ทั้งทวิตเตอร์และสโตร์ และ % แสดง *feature request* กับ *bug report* ที่พบเฉพาะบนทวิตเตอร์ [25]

งานวิจัยนี้ได้ใช้ Naïve Bayes ในการแยกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์ เพลย์สโตร์ และทวิตเตอร์ ว่าเป็น feature request หรือ bug report หรืออื่น ๆ ซึ่งโครงงานมหาบัณฑิตก็จะใช้ Naïve Bayes เช่นเดียวกัน เนื่องจากมีประสิทธิภาพสูง ในขณะที่ใช้ Training data ในปริมาณที่ไม่ได้มากจนเกินไป นอกจากนี้งานวิจัยนี้ได้ใช้ Support Vector Machine เนื่องจากมีประสิทธิภาพดีกว่า Decision Tree สามารถกรองบทวิจารณ์ของผู้ใช้บนทวิตเตอร์ที่ไม่เกี่ยวข้องออกไปได้ดีกว่า

สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิตคือ งานวิจัยนี้มุ่งเน้นเรื่องเกี่ยวกับว่ามีบทวิจารณ์ของผู้ใช้ใด ๆ ที่ปรากฏบนแหล่งอื่น ๆ นอกจากแอปสโตร์ เช่น ทวิตเตอร์ เพลย์สโตร์ เป็นต้น ไม่ได้มีการทำเครื่องมือ

ช่วยในการจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์ เพลย์สโตร์ หรือทวิตเตอร์แต่อย่างใด แต่โครงงานมหาบัณฑิตจะมุ่งเน้นที่การจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ และการสร้างทิคเก็ตของ Jira โดยประเภทของ Issue จะขึ้นอยู่กับประเภทของบทวิจารณ์ของผู้ใช้ที่โมเดลจำแนกประเภทได้

2.2.5 AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace [26]

งานวิจัยนี้ได้กล่าวถึงประเด็นที่ตลาดแอปพลิเคชันนั้นเติบโตอย่างรวดเร็วมาก ในแต่ละปีมีผู้ใช้งานสมาร์ตโฟน เพิ่มขึ้นเป็นจำนวนมาก จำนวนแอปพลิเคชันบนสโตร์ที่ถูกดาวน์โหลดจากผู้ใช้งานก็เพิ่มขึ้น ซึ่งหลังจากผู้ใช้งานได้ทดลองใช้งานแอปพลิเคชันแล้ว ก็จะมาให้คะแนนและแสดงความคิดเห็นในช่องทางบนสโตร์ ซึ่งคะแนนและความคิดเห็นเหล่านี้เป็นสิ่งที่มีประโยชน์ต่อนักพัฒนาแอปพลิเคชันเป็นอย่างมาก เพราะช่วยให้ทราบถึงสิ่งที่ดี สิ่งที่ไม่ดีที่ต้องปรับปรุงเพื่อให้ผู้ใช้งานพึงพอใจมากที่สุด

ปัญหาที่เกิดตามมาคือ นักพัฒนาไม่รู้ว่าควรจะปรับปรุงข้อเสียของแอปพลิเคชันด้านใด เพื่อให้ผู้ใช้งานโดยรวมมีความพึงพอใจมากที่สุด เนื่องจากปริมาณข้อมูลมีจำนวนมาก ประกอบกับขาดเครื่องมือในการวิเคราะห์แบบอัตโนมัติ ทางกลุ่มผู้วิจัยจึงได้ทำการพัฒนาเครื่องมือที่ชื่อว่า App Reviews Miner (AR-Miner) ขึ้นมา โดยจุดประสงค์หลักก็เพื่อจัดอันดับกลุ่มของความคิดเห็นที่มีผลกระทบต่อผู้ใช้งานโดยรวมมากที่สุด ซึ่งมีขั้นตอนการทำงานหลัก ๆ ดังนี้

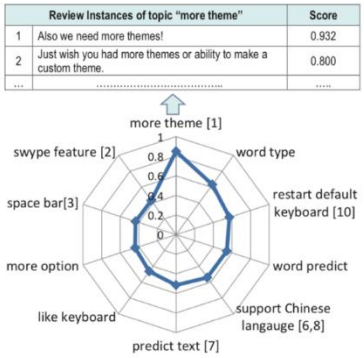
1) ใช้ Expectation Maximization for Naive Bayes (EMNB) ซึ่งเป็น Semi-Supervised Learning ประเภทหนึ่ง ทำการกรองข้อมูลที่ไม่สมบูรณ์ และไม่เกี่ยวข้องออก โดยจำแนกประเภทของบทวิจารณ์ออกเป็นสองกลุ่ม คือ Informative และ Uninformative เพื่อนำโมเดลที่ได้มากรองให้เหลือเฉพาะบทวิจารณ์ที่มีประโยชน์ต่อการพัฒนาซอฟต์แวร์ คือ Informative

2) ใช้ K-Means เพื่อทำการจัดกลุ่มประเภทของบทวิจารณ์ที่มีลักษณะคล้ายกันไว้ด้วยกันบนสมมติฐานที่ว่าหนึ่งบทวิจารณ์ของผู้ใช้ใด ๆ จะถูกจัดกลุ่มให้อยู่ได้เพียงแค่กลุ่มเดียวเท่านั้น

3) ใช้ LDA เพื่อทำการจัดกลุ่มประเภทของบทวิจารณ์ที่มีลักษณะคล้ายกันไว้ด้วยกัน โดยหนึ่งบทวิจารณ์ใด ๆ สามารถถูกจัดให้อยู่ได้มากกว่าหนึ่ง Topic ได้ เนื่องจากบางครั้งบทวิจารณ์ของผู้ใช้ที่มีลักษณะยาว จะมีการอธิบายผสมกันทั้ง feature request และ bug report หรืออื่น ๆ ได้

4) จัดลำดับความสำคัญของแต่ละกลุ่มความคิดเห็นและทำการวัดผล

5) แสดงผลกลุ่มของความคิดเห็นที่มีความสำคัญเรียงตามลำดับจากมากไปน้อยในรูปแบบที่สวยงาม ง่ายต่อนักพัฒนาในการทำความเข้าใจ ดังภาพที่ 2.13



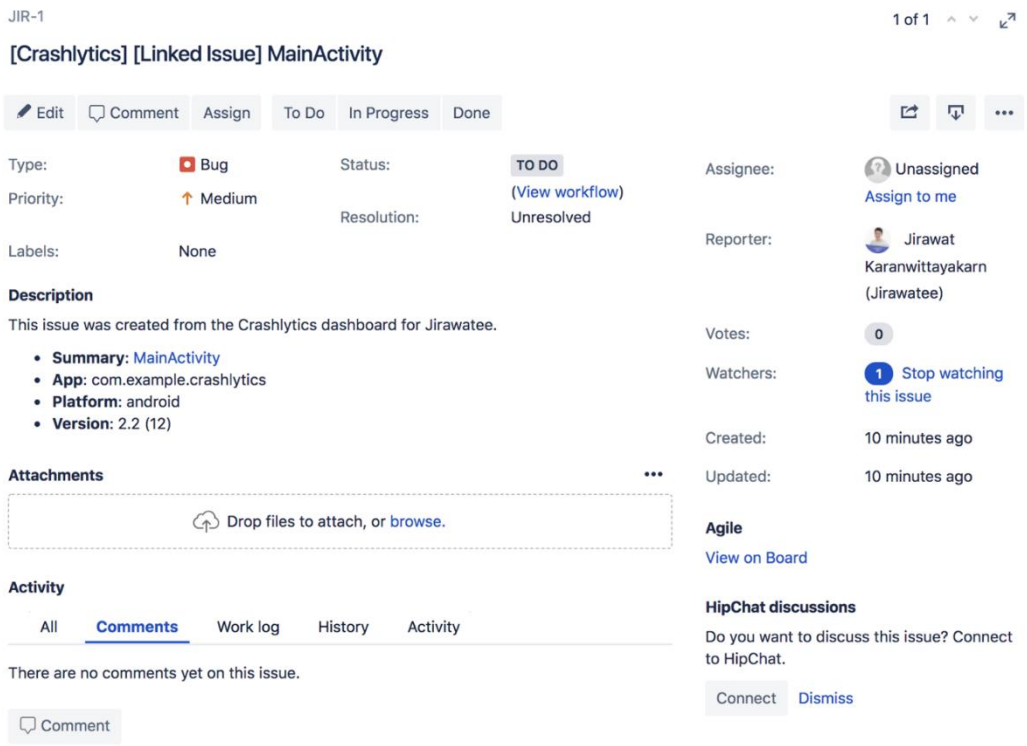
ภาพที่ 2.13 การแสดงผลของ *AR-Miner* ของกลุ่มความคิดเห็น *10* กลุ่มแรกที่มีความสำคัญมากที่สุดของแอปพลิเคชัน *SwiftKey* โดย *Keyword "more theme"* ถูกจัดเป็นอันดับหนึ่ง [26]

กลุ่มผู้วิจัยได้ใช้เครื่องมือ AR-Miner ที่พัฒนาขึ้น มาทดลองเพื่อที่จะทำการวัดประสิทธิภาพของเครื่องมือกับแอปพลิเคชัน Android ที่มีจำนวนบทวิจารณ์ของผู้ใช้มากทั้งสิ้น 4 แอปพลิเคชัน คือ SwiftKey, Facebook, TempleRun 2, TapFish โดยใช้ค่า precision, recall รวมทั้ง f1-score เป็นตัววัดประสิทธิภาพ และนอกจากนี้ยังได้นำไปเทียบกับ forum การแสดงความคิดเห็นออนไลน์ของ SwiftKey ที่เปิดให้ผู้ใช้งานสามารถแสดงความคิดเห็นต่อแอปพลิเคชันได้ โดยผลการทดลองที่ได้คือ เครื่องมือ AR-Miner นั้นช่วยลดงานที่ต้องใช้คนทำลงได้เป็นอย่างมาก รวมทั้งให้ผลลัพธ์ที่มีประสิทธิภาพสูงกว่า นอกจากนี้ยังพบ feature request บางอย่างที่ไม่พบจากการรวบรวมข้อมูลแบบ manual อีกด้วย

โครงงานมหาบัณฑิตจะนำแนวคิดของงานวิจัยนี้ไปใช้เป็นแนวคิดในการทำเครื่องมือเกี่ยวกับบทวิจารณ์ของผู้ใช้ แต่แตกต่างกันตรงที่งานวิจัยนี้สร้างเครื่องมือวิเคราะห์บทวิจารณ์ของผู้ใช้ แล้วนำเสนอออกมาในรูปแบบที่ง่ายต่อการทำความเข้าใจ แต่เครื่องมือของโครงงานมหาบัณฑิตที่จะพัฒนาขึ้นนี้จะเป็นการสร้างทิคเก็ต Jira จากบทวิจารณ์ของผู้ใช้ โดยประเภทของ Issue ขึ้นอยู่กับประเภทของบทวิจารณ์ของผู้ใช้ที่โมเดลจำแนกได้

2.2.6 Automatically Create Jira Issue from Firebase Crashlytics [27]

บทความนี้ไม่ใช่งานวิจัยตีพิมพ์ แต่กล่าวถึงการนำ Jira มาใช้งานร่วมกับ Firebase Crashlytics ซึ่งเป็นความร่วมมือกันทางธุรกิจระหว่าง Atlassian ซึ่งเป็นเจ้าของ Jira และ Google ที่เป็นเจ้าของ Firebase Crashlytics ทั้งนี้ Firebase Crashlytics คือ ไลบรารีที่สามารถนำไปติดตั้งไว้ในโมไบล์แอปพลิเคชันและเว็บไซต์ได้ เมื่อโมไบล์แอปพลิเคชันหรือเว็บไซต์เกิดการปิดตัวลงอย่างกะทันหัน (Crash) ตัว Firebase Crashlytics จะทำการสร้างบันทึกรายงานจุดบกพร่องที่ระบุรายละเอียดเกี่ยวกับจุดที่เกิดปัญหาขึ้น เช่น ชื่อไฟล์ และหมายเลขบรรทัดของโค้ดที่เกิดปัญหา เป็นต้น เก็บไว้ใน Firebase Console ซึ่งนักพัฒนาสามารถเข้ามาดูรายละเอียดในภายหลังได้ พร้อมทั้งมีการแจ้งเตือนทางอีเมลด้วย



ภาพที่ 2.14 ตัวอย่างทิคเก็ตประเภท *bug* ที่ถูกสร้างใน *Jira* จาก *Firebase Crashlytics* ที่ทำการเชื่อมต่อกับ *APIs* ของ *Jira* [27]

ทาง Atlassian และ Google ต้องการให้การทำงานของบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระนั้นราบรื่นมากยิ่งขึ้น โดยเล็งเห็นว่าส่วนใหญ่บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระนิยมใช้ Jira เป็นเครื่องมือในการบริหารจัดการโครงการซอฟต์แวร์แบบเอไจล์อยู่แล้ว ดังนั้นเมื่อเกิดจุดบกพร่องจนทำให้โมไบล์แอปพลิเคชันหรือเว็บไซต์ปิดตัวลงอย่างกะทันหัน Google จะทำการเรียก APIs ของ Jira เพื่อทำการสร้าง Issues ขึ้นในเอไจล์บอร์ดโดยกระบวนการทั้งหมดเป็นไปอย่างอัตโนมัติ เพียงแค่นักพัฒนาติดตั้งไลบรารี Firebase Crashlytics ลงไป และไปทำการตั้งค่าใน Jira กับ Firebase Console เล็กน้อย ตัวอย่าง Issue ที่ถูกสร้างจาก Firebase Crashlytics เข้าไปที่ Jira เป็นดังภาพที่ 2.14 ซึ่งผู้วิจัยจะนำแนวทางการเชื่อมต่อระหว่าง Jira กับ Firebase Crashlytics มาสร้างเป็นเครื่องมือที่สามารถสร้างทิคเก็ตแยกตามประเภทของบทวิจารณ์ของผู้ใช้ โดยทำการเรียกใช้งาน APIs ของ Jira เพื่อส่งค่าพารามิเตอร์เข้าไปเช่นเดียวกับที่ Firebase Crashlytics ทำ

2.2.7 Recommending and Localizing Change Requests for Mobile Apps based on User Reviews [28]

งานวิจัยนี้ได้ทำการจัดกลุ่มบทวิจารณ์ของผู้ใช้ที่มีลักษณะคล้ายกันให้อยู่ในกลุ่มเดียวกัน จากนั้นทำการระบุบทวิจารณ์ของผู้ใช้ที่เกี่ยวกับจุดบกพร่องต่าง ๆ ที่รวบรวมมาได้ เพื่อบอกว่าโค้ดไฟล์ใด ฟังก์ชันใด ที่มีส่วนเกี่ยวข้องกับจุดบกพร่องนั้น ๆ องค์ความรู้ที่ใช้หลัก ๆ จะเป็นเรื่อง Natural Language Processing และ Clustering Algorithms ต่าง ๆ ซึ่งหลังจากได้ผลลัพธ์ออกมาแล้ว งานวิจัยนี้ได้ทำการสร้างเครื่องมือที่ชื่อว่า CHANGEADVISOR โดยมีฟังก์ชันการทำงานดังนี้

1) แยกประเภทของบทวิจารณ์ของผู้ใช้ประเภท maintenance ออกมาได้

2) จัดกลุ่มบทวิจารณ์ของผู้ใช้ที่มีลักษณะคล้ายกันให้อยู่กลุ่มเดียวกันได้

3) บอกได้ว่าไฟล์ และฟังก์ชันใด ที่ทำให้เกิดจุดบกพร่องขึ้น โดยอิงข้อมูลจากบทวิจารณ์ของผู้ใช้ และซอร์ซโค้ดที่ต้องอัปโหลดเข้าไปในเครื่องมือ

ผู้วิจัยจะนำวิธีการทำ Data Cleansing ของงานวิจัยนี้มาใช้ในโครงงานมหาบัณฑิต เนื่องจากมีหลายขั้นตอน และมีความละเอียดสูง ซึ่งจะกล่าวต่อไปในบทถัด ๆ ไป ข้อแตกต่างระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้สร้างเครื่องมือที่ช่วยระบุได้ว่าโค้ดไฟล์ใด ฟังก์ชันอะไรที่ทำให้เกิดจุดบกพร่องขึ้นโดยอิงข้อมูลจากบทวิจารณ์ของผู้ใช้และซอร์ซโค้ดที่อัปโหลดให้กับเครื่องมือ ส่วนเครื่องมือที่จะพัฒนาในโครงงานมหาบัณฑิตจะทำหน้าที่สร้างทิคเก็ต Jira แยกประเภท Issue ตามประเภทของบทวิจารณ์ของผู้ใช้แบบอัตโนมัติ

2.2.8 Ensemble Methods for App Review Classification: An Approach for Software Evolution [29]

งานวิจัยนี้เป็นการรวบรวมบทวิจารณ์ของผู้ใช้เพื่อทำการแยกประเภทเป็น bug report, feature strength, feature shortcoming, user request, praise, complaint, และ usage scenario โดยทำการรวบรวมบทวิจารณ์ของผู้ใช้จาก 7 แอปพลิเคชันดังภาพที่ 2.15 แล้วนำมาติดฉลากให้กับแต่ละบทวิจารณ์เพื่อเป็นชุดข้อมูลเรียนรู้และชุดข้อมูลทดสอบให้กับโมเดลและทำการวัดประสิทธิภาพของโมเดลทั้งโมเดลแบบเดี่ยว เช่น Naïve Bayes, Support Vector Machine, Logistic Regression, Neural Network และโมเดลแบบรวม คือ Ensemble models โดยใช้ค่า precision, recall, และ f1-score ได้ผลลัพธ์เป็นดังภาพที่ 2.16 โดยพบว่าโมเดลแบบรวม มีประสิทธิภาพในการจำแนกบทวิจาณ์ของผู้ใช้สูงกว่าโมเดลแบบเดี่ยว ซึ่งโมเดลที่มีประสิทธิภาพสูงที่สุดได้ค่า precision 0.74, recall 0.59, และ f1-score 0.64

App	Bug report	Feature strength	Feature shortcoming	User request	Praise	Complaint	Usage scenario	Noise
Angrybirds	97	77	205	43	243	71	34	19
Dropbox	190	116	203	93	172	21	130	11
Evernote	226	139	227	82	190	51	172	10
Tripadvisor	102	127	249	80	182	43	157	18
Picsart	87	77	99	24	380	41	27	48
Pininterest	214	77	201	48	256	23	53	5
Whatsapp	74	31	97	34	280	27	19	156
Total	*990*	*644*	*1281*	*404*	*1703*	*277*	*593*	*267*

ภาพที่ 2.15 ข้อมูลแต่ละแอปพลิเคชันที่เก็บรวบรวมจากบทวิจาณ์ของผู้ใช้และทำการติดฉลากแล้ว [29]

	Naive Bayes			SVM			Logistic Regression			Neural Network			Ensemble A			Ensemble B			Ensemble C		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Bug report	0.86	0.66	0.74	0.86	0.68	0.76	0.90	0.60	0.72	0.83	0.75	0.79	0.86	0.72	0.78	0.86	0.72	0.78	0.83	0.80	0.81
Complaint	0.50	0.02	0.03	0.20	0.03	0.06	0.50	0.02	0.03	0.45	0.08	0.14	0.20	0.03	0.06	0.20	0.05	0.07	0.45	0.08	0.14
User request	0.73	0.18	0.26	0.69	0.33	0.45	0.68	0.26	0.38	0.71	0.39	0.50	0.72	0.40	0.51	0.72	0.40	0.51	0.71	0.39	0.50
Feature shortcoming	0.70	0.77	0.73	0.72	0.57	0.64	0.69	0.57	0.62	0.74	0.75	0.75	0.70	0.77	0.73	0.70	0.77	0.73	0.68	0.80	0.73
Feature strength	0.60	0.13	0.22	0.69	0.29	0.41	0.75	0.23	0.35	0.70	0.50	0.59	0.70	0.50	0.59	0.70	0.50	0.59	0.70	0.50	0.59
Noise	0.83	0.42	0.56	0.83	0.42	0.56	1.00	0.58	0.74	0.69	0.75	0.72	0.69	0.75	0.72	0.69	0.75	0.72	0.69	0.75	0.72
Praise	0.67	0.75	0.71	0.74	0.71	0.72	0.76	0.54	0.63	0.76	0.73	0.74	0.71	0.79	0.74	0.71	0.79	0.74	0.71	0.75	0.73
Usage scenario	0.67	0.09	0.16	0.56	0.18	0.27	0.70	0.19	0.29	0.73	0.27	0.39	0.59	0.21	0.31	0.59	0.23	0.34	0.69	0.34	0.46
Average	*0.70*	*0.48*	*0.51*	*0.70*	*0.49*	*0.55*	*0.75*	*0.42*	*0.52*	*0.74*	*0.59*	*0.64*	*0.70*	*0.57*	*0.62*	*0.69*	*0.63*	*0.65*	*0.71*	*0.62*	*0.64*

ภาพที่ 2.16 ค่าประสิทธิภาพที่ได้จากชุดข้อมูลทดสอบโดยใช้โมเดลการจำแนกประเภทแบบเดี่ยว และโมเดลแบบรวม [29]

บทที่ 3
การพัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้

แนวคิดงานวิจัยนี้แบ่งเป็นสามขั้นตอนหลักดังภาพที่ 3.1 โดยขั้นตอนแรกเป็นการพัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ซึ่งรับข้อมูลนำเข้าเป็นบทวิจารณ์ของผู้ใช้ที่ติดฉลากแล้วจากงานวิจัย [22] ได้ผลลัพธ์ออกมาเป็นโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้สองโมเดลที่มีประสิทธิภาพมากที่สุด ขั้นตอนที่สองเป็นการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้โดยมีการรวบรวมข้อมูลสำหรับทำการทดลองเพื่อหาค่าขีดแบ่งที่มีประสิทธิภาพมากที่สุดในการระบุว่าบทวิจารณ์ซ้ำซ้อนหรือไม่ ผลลัพธ์ที่ได้ออกมาคือค่าขีดแบ่งที่เหมาะสมมากที่สุด และขั้นตอนสุดท้ายเป็นการวิเคราะห์บทวิจารณ์ของผู้ใช้ใหม่เพื่อจำแนกประเภทบทวิจารณ์ว่าเป็นประเภท bug report หรือ feature request มีการสร้างเครื่องมือโปรแกรม Command line เพื่อรวบรวมบทวิจารณ์ ทำการจำแนกประเภท ตรวจสอบบทวิจารณ์ซ้ำซ้อน และทำการเชื่อมต่อกับจิราเพื่อสร้างเป็นทิคเก็ตบนจิราบอร์ด โดยเนื้อหาในบทนี้จะอธิบายเกี่ยวกับขั้นตอนการให้ได้มาซึ่งโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ที่สามารถจำแนกประเภทของบทวิจารณ์ได้เป็น bug report หรือ feature request ได้อย่างมีประสิทธิภาพ โดยขั้นตอนการดำเนินงานทั้งหมดแบ่งได้ทั้งสิ้น 4 ขั้นตอน ดังนี้

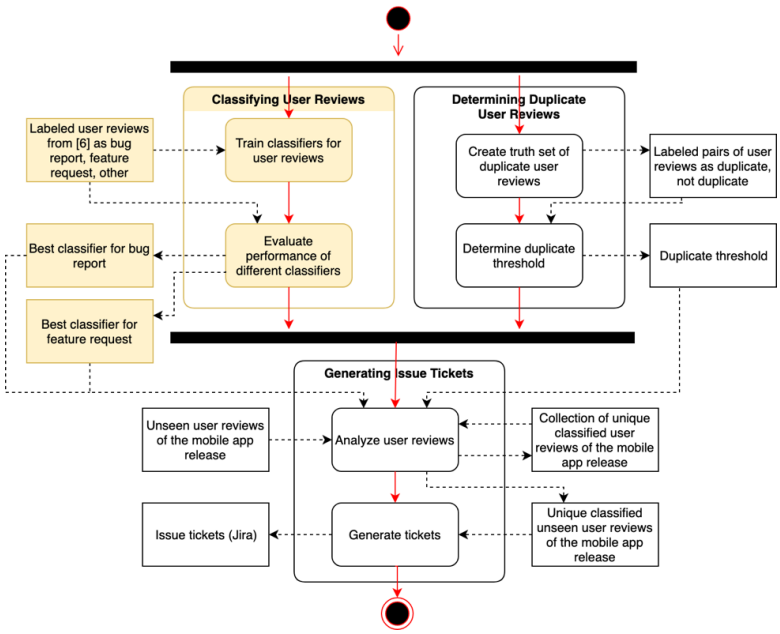
3.1 ขั้นตอนการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ตัวอย่าง

3.2 ขั้นตอนการพัฒนาโมเดลเพื่อจำแนกประเภทของบทวิจารณ์ของผู้ใช้

3.3 ขั้นตอนการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

3.4 ขั้นตอนการนำออกโมเดลเพื่อใช้ในการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

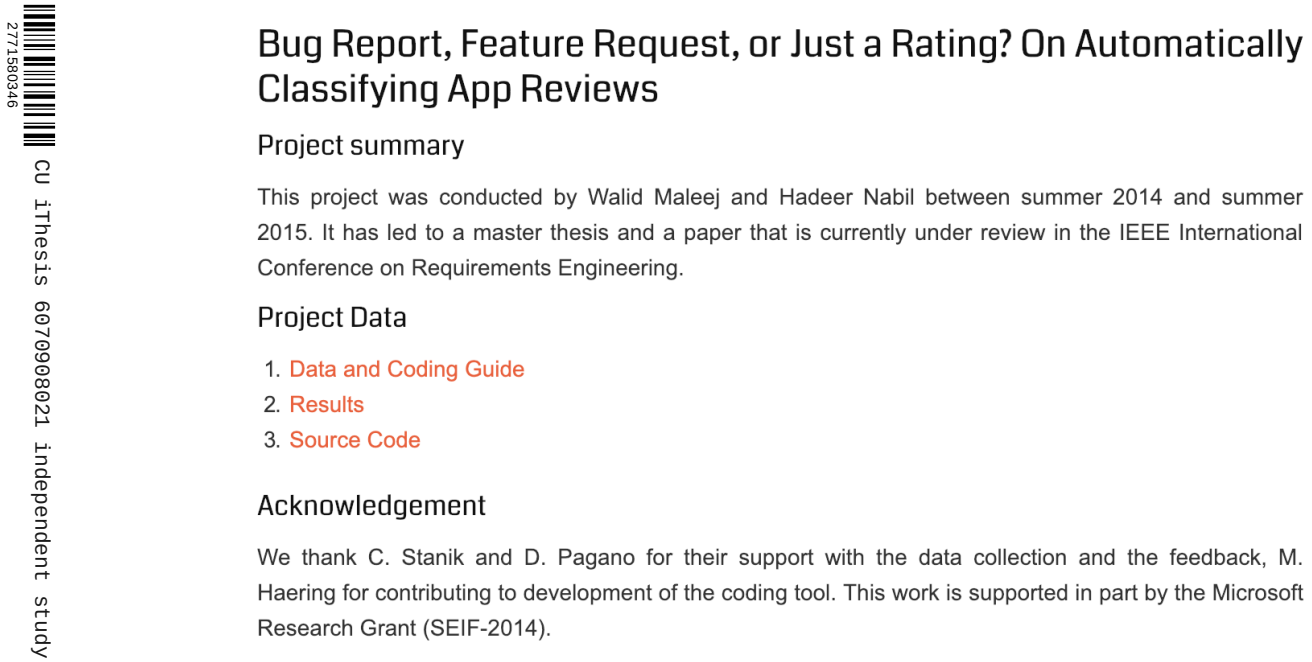
โดยภาพที่ 3.1 ส่วนที่ไฮไลท์แสดงขั้นตอนการดำเนินงานแบบละเอียดของบทนี้



ภาพที่ 3.1 ขั้นตอนการดำเนินงานแบบละเอียดของการพัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้

3.1 ขั้นตอนการเก็บรวบรวมบทวิจารณ์ของผู้ใช้

ขั้นตอนนี้จะเป็นการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ตัวอย่างที่ได้ทำการแยกประเภทไว้แล้วมาเป็น Training data และ Test data เพื่อเป็นการประหยัดเวลาในการติดฉลากให้กับบทวิจารณ์ของผู้ใช้แบบ manual รวมไปถึงเรื่องของความน่าเชื่อถือของข้อมูล ทำให้ผู้วิจัยเลือกใช้ข้อมูลชุดเดียวกับงานวิจัยที่ 2.2.1 [22] ซึ่งได้เปิดให้สามารถดาวน์โหลดข้อมูลบทวิจารณ์ของผู้ใช้ได้ฟรี มี 4,400 บทวิจารณ์อยู่ในรูปโครงสร้างแบบ JSON โดยสามารถโหลดได้จาก https://mast.informatik.uni-hamburg.de/app-review-analysis/ ดังภาพที่ 3.2



Bug Report, Feature Request, or Just a Rating? On Automatically Classifying App Reviews

Project summary

This project was conducted by Walid Maleej and Hadeer Nabil between summer 2014 and summer 2015. It has led to a master thesis and a paper that is currently under review in the IEEE International Conference on Requirements Engineering.

Project Data

1. Data and Coding Guide
2. Results
3. Source Code

Acknowledgement

We thank C. Stanik and D. Pagano for their support with the data collection and the feedback, M. Haering for contributing to development of the coding tool. This work is supported in part by the Microsoft Research Grant (SEIF-2014).

ภาพที่ 3.2 หน้าเว็บไซต์ของกลุ่มผู้จัดทำงานวิจัยที่ *2.2.1* ที่เปิดให้ดาวน์โหลดข้อมูลบทวิจารณ์ของผู้ใช้ที่ถูกติดฉลากแล้ว [22]

หลักการจำแนกประเภทบทวิจารณ์ของผู้ใช้จะอ้างอิงตามงานวิจัยที่ 2.2.1 ซึ่งได้นำเสนอคู่มือการจำแนกประเภทบทวิจารณ์ของผู้ใช้โดยสามารถดาวน์โหลดได้จาก https://mobis.informatik.uni-hamburg.de/wp-content/uploads/2015/03/Data-and-Coding-Guide-.zip ซึ่งมีคำอธิบายของบทวิจารณ์ของผู้ใช้ประเภท bug report และ feature request ดังนี้

1) Bug report ผู้ใช้งานอธิบายถึงปัญหาที่พบในการใช้งานแอปพลิเคชัน พฤติกรรมการทำงานที่ไม่เหมาะสมของฟังก์ชันการทำงานใด ๆ หรือทั้งแอปพลิเคชันโดยรวม เช่น

“Uploading is not working with the iOS6”

“Every time I launch the app, it crashes”

“After the new update, my mobile freezes after I've been using the app for a few minutes”

“I lost all my phone contacts. Great, thank you!”

2) Feature request ผู้ใช้งานอธิบายถึงฟังก์ชันการทำงาน รายละเอียดของแอปพลิเคชันที่ขาดหายไป หรือฟังก์ชันเดิมที่มีอยู่แล้ว แต่ต้องการให้มีการปรับปรุงให้ดียิ่งขึ้น เช่น

“It would be great if we could copy and paste text”

“Great app, only one suggestion, it would be great if it allowed for daily flow values.”

“I wish you could add a link that would allow me to share the information with my Facebook friends”

บทวิจารณ์ของผู้ใช้ที่รวบรวมได้จากงานวิจัยที่ 2.2.1 จะถูกแบ่งออกเป็นสี่ประเภท คือ bug report, not bug report, feature request, และ not feature request ตัวอย่างบทวิจารณ์ของผู้ใช้และข้อมูลอื่น ๆ ที่เกี่ยวข้องถูกเก็บอยู่ในรูปแบบ JSON ดังภาพที่ 3.3, 3.4, 3.5, และ 3.6 แสดงถึง bug report, not bug report, feature request, และ not feature request ตามลำดับ


```
{
    "id": 217,
    "appId": null,
    "reviewId": 3247,
    "title": "PDF Problems",
    "comment": "I liked very much the upgrade to pdfs (divisions and search) However, they aren&#39t
    displaying anymore Fix it and it will be perfect\nBest wishes",
    "rating": 3,
    "reviewer": null,
    "fee": null,
    "date": null,
    "dataSource": "RE2014_app_and_play_store_apps",
    "stopwords_removal": "liked very much upgrade to pdfs (divisions search) however, aren&#39t
    displaying anymore fix will be perfect best wishes",
    "lemmatized_comment": "i like very much the upgrade to pdfs (divisions and search) however, they
    aren&#39t display anymore fix it and it will be perfect best wish",
    "stemmed": "i lik very much the upgrad to pdfs (divisions and search) however, they aren&#39t
    display anym fix it and it wil be perfect best wish",
    "sentiScore": 3,
    "sentiScore_pos": 3,
    "sentiScore_neg": -1,
    "stopwords_removal_nltk": "liked much upgrade pdfs (divisions search) however, aren&#39t displaying
    anymore fix perfect best wishes",
    "stopwords_removal_lemmatization": "like very much upgrade to pdfs (divisions search) however, aren&
    #39t display anymore fix will be perfect best wish",
    "length_words": 31,
    "present_simple": 3,
    "present_con": 1,
    "past": 1,
    "future": 1
},
```


ภาพที่ 3.3 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *bug report*

```
{
    "id": 1481,
    "appId": "350189835",
    "reviewId": 189,
    "title": "Dope AF",
    "comment": "Always a big help, love your video of the venese beach wax king selection",
    "rating": 5,
    "reviewer": null,
    "fee": null,
    "date": null,
    "dataSource": null,
    "stopwords_removal": "always big help, love video of venese beach wax king selection",
    "lemmatized_comment": "always a big help, love your video of the venese beach wax king selection",
    "stemmed": "alway a big help, lov yo video of the ven beach wax king select",
    "sentiScore": 3,
    "sentiScore_pos": 3,
    "sentiScore_neg": -1,
    "stopwords_removal_nltk": "always big help, love video venese beach wax king selection",
    "stopwords_removal_lemmatization": "always big help, love video of venese beach wax king selection",
    "length_words": 15,
    "present_simple": 1,
    "present_con": 1,
    "past": 0,
    "future": 0
},
```


ภาพที่ 3.4 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *not bug report*


```
{
    "id": 199,
    "appId": "358801284",
    "reviewId": 3307,
    "title": "Fantastic",
    "comment": "I would of given it a 5 if I could pin on Pinterest from it.",
    "rating": 4,
    "reviewer": null,
    "fee": null,
    "date": null,
    "dataSource": "AppStore_Random",
    "stopwords_removal": "would of given 5 if could pin pinterest from it",
    "lemmatized_comment": "i would of give it a 5 if i could pin on pinterest from it",
    "stemmed": "i would of giv it a 5 if i could pin on pinterest from it",
    "sentiScore": -1,
    "sentiScore_pos": 1,
    "sentiScore_neg": -1,
    "stopwords_removal_nltk": "would given 5 could pin pinterest it",
    "stopwords_removal_lemmatization": "would of give 5 if could pin pinterest from it",
    "length_words": 16,
    "present_simple": 1,
    "present_con": 0,
    "past": 1,
    "future": 0
},
```


ภาพที่ 3.5 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *feature request*

```
{
    "id": 1508,
    "appId": "436491861",
    "reviewId": 241,
    "title": "Not bad",
    "comment": "I like the chicken bites but they always forget at least 1 sauce every time",
    "rating": null,
    "reviewer": null,
    "fee": null,
    "date": null,
    "dataSource": "AppStore_Random",
    "stopwords_removal": "like chicken bites but always forget at least 1 sauce every time",
    "lemmatized_comment": "i like the chicken bite but they always forget at least 1 sauce every time",
    "stemmed": "i lik the chick bit but they alway forget at least 1 sauc every tim",
    "sentiScore": 2,
    "sentiScore_pos": 2,
    "sentiScore_neg": -1,
    "stopwords_removal_nltk": "like chicken bites always forget least 1 sauce every time",
    "stopwords_removal_lemmatization": "like chicken bite but always forget at least 1 sauce every time",
    "length_words": 15,
    "present_simple": 3,
    "present_con": 0,
    "past": 0,
    "future": 0
},
```


ภาพที่ 3.6 ตัวอย่างบทวิจารณ์ของผู้ใช้ประเภท *not feature request*

3.2 ขั้นตอนการพัฒนาโมเดลเพื่อจำแนกประเภทของบทวิจารณ์ของผู้ใช้

ในขั้นตอนนี้จะเป็นการนำบทวิจารณ์ของผู้ใช้ที่รวบรวมได้จากข้อ 3.1 มาทำการทดลองแยกประเภทของบทวิจารณ์ของผู้ใช้เป็น bug report หรือ not bug report และ feature request หรือ not feature request โดยฟีเจอร์ที่แตกต่างกันที่จะถูกใช้เป็นข้อมูลนำเข้าในการเรียนรู้ให้กับโมเดลเป็นดังตารางที่ 3.1

ตารางที่ 3.1 ฟีเจอร์ที่แตกต่างกันที่ใช้เป็นข้อมูลนำเข้าในการเรียนรู้ให้กับโมเดล

ฟีเจอร์	คำอธิบาย
comment	บทวิจารณ์ต้นฉบับของผู้ใช้
lemmatized_comment	บทวิจารณ์ของผู้ใช้ที่เปลี่ยนคำในประโยคทั้งหมดให้เป็นรากศัพท์
stopwords_removal	บทวิจารณ์ของผู้ใช้ที่นำคำหยุดในภาษาอังกฤษออกจากประโยค
stopwords_removal_lemmatization	บทวิจารณ์ของผู้ใช้ที่นำคำหยุดในภาษาอังกฤษออก และเปลี่ยนคำที่เหลืออยู่ในประโยคทั้งหมดให้กลายเป็นรากศัพท์
rating	คะแนนความพึงพอใจที่ผู้ใช้งานให้กับแอปพลิเคชัน มีค่าระหว่าง 1-5 โดย 1 คือ แย่ที่สุด และ 5 คือ ดีที่สุด
length_words	จำนวนคำศัพท์ทั้งหมดที่อยู่ในบทวิจารณ์ต้นฉบับหนึ่ง ๆ
tenses	จำนวน present simple, present continuous, past simple, และ future ที่อยู่ในบทวิจารณ์ต้นฉบับหนึ่ง ๆ

ตารางที่ 3.1 ฟีเจอร์ที่แตกต่างกันที่ใช้เป็นข้อมูลนำเข้าในการเรียนรู้ให้กับโมเดล (ต่อ)

ฟีเจอร์	คำอธิบาย
sentiScore	คะแนนอารมณ์ความรู้สึกที่คำนวณได้จากบทวิจารณ์ของผู้ใช้ต้นฉบับหนึ่ง ๆ โดยมีค่าระหว่าง -5 คือ เป็นเชิงลบมากที่สุด ถึง 5 คือ เป็นเชิงบวกมากที่สุด
part of speech	จำนวน noun, pronoun, verb, adjective, และ adverb ที่อยู่ในบทวิจารณ์หนึ่ง ๆ
word2vec	เวกเตอร์ของคำศัพท์ที่อยู่ในบทวิจารณ์หนึ่ง ๆ
doc2vec	เวกเตอร์ของเอกสารที่อยู่ในบทวิจารณ์หนึ่ง ๆ

โดยอัลกอริทึมที่แตกต่างกันในการจำแนกประเภทบทวิจารณ์ของผู้ใช้จะถูกแบ่งออกเป็นสองประเภทหลัก คือ โมเดลแบบเดี่ยว (Individual Model) และโมเดลแบบรวม (Ensemble Model) โดยแต่ละประเภทประกอบไปด้วยอัลกอริทึมดังต่อไปนี้

1) Individual Model ประกอบไปด้วย 7 โมเดลดังต่อไปนี้ คือ Multinomial Naïve Bayes, Gaussian Naïve Bayes, Bernoulli Naïve Bayes, Linear Support Vector Machine, Logistic Regression, Decision Tree, และ k-Nearest Neighbors
2) Ensemble Model ประกอบไปด้วย 6 โมเดลดังต่อไปนี้ คือ Random Forest, Extra Trees, Max Voting, AdaBoost, XGBoost, และ Gradient Boosting

ผู้วิจัยได้เลือกฟีเจอร์ที่แตกต่างกันบางฟีเจอร์มาใช้ร่วมกันได้เป็นจำนวนทั้งสิ้น 28 รายการฟีเจอร์ โดยมีรายละเอียดดังตารางที่ 3.2 ต่อไปนี้

ตารางที่ 3.2 รายการฟีเจอร์ที่เกิดจากการนำฟีเจอร์ที่แตกต่างกันแต่ละฟีเจอร์มาใช้งานร่วมกัน

ลำดับที่	รายการฟีเจอร์
1	comment
2	lemmatized_comment
3	stopwords_removal
4	stopwords_removal_lemmatization
5	rating + length_words
6	rating + length_words + tenses
7	rating + length_words + tenses + sentiScore
8	comment + rating + sentiScore
9	comment + rating + sentiScore + tenses
10	stopwords_removal_lemmatization + rating + sentiScore + tenses
11	comment + part of speech

ตารางที่ 3.2 รายการฟีเจอร์ที่เกิดจากการนำฟีเจอร์ที่แตกต่างกันแต่ละฟีเจอร์มาใช้งานร่วมกัน (ต่อ)

ลำดับที่	รายการฟีเจอร์
12	lemmatized_comment + part of speech
13	stopwords_removal + part of speech
14	stopwords_removal_lemmatization + part of speech
15	comment + rating + sentiScore + part of speech
16	comment + rating + sentiScore + tenses + part of speech
17	stopwords_removal_lemmatization + rating + sentiScore + tenses + part of speech
18	rating + length_words + part of speech
19	rating + length_words + tenses + part of speech
20	rating + length_words + tenses + sentiScore + part of speech
21	word2vec + comment
22	word2vec + lemmatized_comment
23	word2vec + stopwords_removal
24	word2vec + stopwords_removal_lemmatization
25	doc2vec + comment
26	doc2vec + lemmatized_comment
27	doc2vec + stopwords_removal
28	doc2vec + stopwords_removal_lemmatization

หลังจากได้รายการฟีเจอร์มาทั้งสิ้น 28 รายการฟีเจอร์ และโมเดลการจำแนกประเภท 12 อัลกอริทึมแล้ว ผู้วิจัยได้ทำการนำเข้าข้อมูลบทวิจารณ์ของผู้ใช้ทั้งสี่หมวดหมู่ คือ bug report, not bug report, feature request, และ not feature request โดยมีรายละเอียดดังตารางที่ 3.3 ต่อไปนี้

ตารางที่ 3.3 จำนวนข้อมูลบทวิจารณ์ของผู้ใช้แยกตามหมวดหมู่ และวัตถุประสงค์ของการนำไปใช้

Data	Bug Report Classification		Feature Request Classification	
Training	Bug Report	256	Feature Request	207
	Not Bug Report	256	Not Feature Request	207
	Total	512	Total	414
Test	Bug Report	114	Feature Request	88
	Not Bug Report	114	Not Feature Request	114
	Total	228	Total	202

ฟีเจอร์ส่วนใหญ่จะได้ค่ามาพร้อมกับข้อมูลนำเข้าแล้ว ยกเว้นสามฟีเจอร์ คือ part of speech, word2vec, และ doc2vec ซึ่งทางผู้วิจัยจะแสดงวิธีการหาค่าแต่ละค่าโดยยกตัวอย่างบทวิจารณ์ของผู้ใช้มาหนึ่งบทวิจารณ์ คือ “Very useful. But it is difficult sometimes trying to type an essay without landscape mode. It is ridiculous how a word processing app doesnt let u type landscape.”

วิธีหาค่า Part of Speech

เนื่องจากบทวิจารณ์ของผู้ใช้อยู่ในรูปแบบประโยคภาษาอังกฤษประกอบไปด้วยชนิดของคำต่าง ๆ เช่น noun, pronoun, verb, adjective, adverb เป็นต้น ผู้วิจัยจึงมีสมมติฐานว่าจำนวนของคำชนิดต่าง ๆ ที่อยู่ในบทวิจารณ์ของผู้ใช้อาจจะเกี่ยวข้องกับประเภทของบทวิจารณ์ของผู้ใช้ที่จะจำแนก จึงได้นำ part of speech มาเป็นหนึ่งในฟีเจอร์ที่จะให้โมเดลเรียนรู้ หนึ่งในวิธีที่ทำได้ก็คือ การนับจำนวน noun, pronoun, verb, adjective, และ adverb ของคำต่าง ๆ ที่ปรากฏอยู่ในบทวิจารณ์หนึ่ง ๆ ซึ่งฟังก์ชันการนับจำนวนคำเหล่านี้ที่เขียนด้วยภาษา Python เวอร์ชัน 3 นั้นปรากฏดังภาพที่ 3.7

```python
# function to check and get the part of speech tag count of a words in a given sentence
def check_pos_tag(x, flag):
    pos_family = {
                    'noun': ['NN', 'NNS', 'NNP', 'NNPS'],
                    'pron': ['PRP', 'PRP$', 'WP', 'WP$'],
                    'verb': ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'],
                    'adj':  ['JJ', 'JJR', 'JJS'],
                    'adv': ['RB', 'RBR', 'RBS', 'WRB']
                }

    cnt = 0
    try:
        wiki = textblob.TextBlob(x)
        for tup in wiki.tags:
            ppo = list(tup)[1]
            if ppo in pos_family[flag]:
                cnt += 1
    except:
        pass
    return cnt
```

ภาพที่ 3.7 ฟังก์ชันการนับจำนวน *part of speech* ที่เขียนด้วยภาษา *Python* เวอร์ชัน *3*

อธิบายอย่างง่ายคือ ใช้ Dictionary ที่เก็บ tag ต่าง ๆ ที่เกี่ยวข้องกับ noun, pronoun, verb, adjective, และ adverb เช่น ถ้าเป็น noun จะมี tag ที่เป็นไปได้คือ NN, NNS, NNP, NNPS หรือถ้าเป็น verb จะมี tag ที่เป็นไปได้คือ VB, VBD, VBG, VBN, VBP, VBZ เป็นต้น จากนั้นทำการสกัด tag ออกมาจากข้อความว่ามี tag อะไรบ้าง ถ้า tag ที่ได้ออกมานั้นมีอยู่ภายใน Dictionary ก็จะทำการบวกตัวเคาน์เตอร์ cnt ไปทีละหนึ่ง ทำเช่นนี้ไปจนจบประโยคทีละประเภทของคำ ก็จะได้ออกมาว่าจำนวน noun, pronoun, verb, adjective, และ adverb ที่ปรากฏในประโยคนั้นมีจำนวนเท่าไหร่บ้าง และสามารถนำตัวเลขเหล่านั้นไปเป็นหนึ่งในฟีเจอร์เพื่อให้โมเดลทำการเรียนรู้ได้ ตารางที่ 3.4 แสดงค่า part of speech ของประโยคตัวอย่าง

ตารางที่ 3.4 ค่า *part of speech* ของแต่ละชนิดของคำของประโยคตัวอย่าง

Sentence: Very useful. But it is difficult sometimes trying to type an essay without landscape mode. It is ridiculous how a word processing app doesnt let u type landscape.	
Type	**Total**
Noun	9
Pronoun	2
Verb	5
Adjective	4
Adverb	3

วิธีหาค่า word2vec

เนื่องจากคำ ๆ เดียวกันในภาษาอังกฤษสามารถมีได้มากกว่าหนึ่งความหมายขึ้นอยู่กับตำแหน่งของคำที่ถูกวางอยู่ในประโยค ทำให้การใช้ Bag of Words หรือ TF-IDF ที่สนใจเพียงแค่การปรากฏของคำโดยไม่คำนึงถึงตำแหน่งที่ปรากฏอยู่ในประโยคอาจทำให้ประสิทธิภาพในการจำแนกประเภทของบทวิจารณ์ลดลง ผู้วิจัยจึงได้นำ word2vec ซึ่งเป็นหนึ่งในวิธีการทำ word embedding และมีการพิจารณาถึงคำรอบด้านของคำที่สนใจหาความหมายด้วย โดยจะต้องมี corpus ขนาดใหญ่มากที่เก็บรวบรวมคำศัพท์ต่าง ๆ ที่ใช้กันในชีวิตประจำวันไว้ ในโครงงานมหาบัณฑิตนี้ใช้ corpus ของ Google ที่ชื่อว่า GoogleNews-vectors-negative300 ที่เปิดให้ดาวน์โหลดได้ฟรีที่ https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM จากนั้นใช้ไลบรารี Gensim ทำการ decode ไฟล์ดังกล่าวออกมาเป็นเวกเตอร์ขนาดใหญ่ แล้วทำการส่งคำแต่ละคำในประโยคที่ต้องการเข้าไปประมวลผลกับเวกเตอร์นั้น จะได้ผลลัพธ์คืนกลับออกมาเป็นเวกเตอร์ของคำนั้น ๆ ภาพที่ 3.8 แสดงตัวอย่างเวกเตอร์ของคำว่า “Very” จากประโยคตัวอย่าง “Very useful. But it is difficult sometimes trying to type an essay without landscape mode. It is ridiculous how a word processing app doesnt let u type landscape.” จะเห็นว่าเวกเตอร์ของคำที่ได้จะเป็นอาเรย์หนึ่งมิติที่เก็บค่าจำนวนจริงไว้หลายค่า ซึ่งถ้าต้องการเวกเตอร์ของทั้งประโยคก็จะกลายเป็นอาเรย์สองมิติ โดยแถวหมายถึงคำแต่ละคำที่อยู่ในประโยค หลักหมายถึงค่าต่าง ๆ ของคำใดคำหนึ่ง และเมทริกซ์จัตุรัสคือประโยคใดประโยคหนึ่ง

```
[-0.06923115  0.05459658  0.04707791  0.05625349 -0.01346863  0.03091693
 -0.06291606 -0.07083508 -0.02088246  0.04161089 -0.0393168  -0.0422241
 -0.0789726   0.02657797 -0.09373336  0.04820842  0.05639772  0.1240375
  0.00569079  0.05822063 -0.13822374  0.02340301  0.01874197  0.04318891
  0.01512907  0.00066189 -0.11086778  0.01711468  0.01032464 -0.05153672
 -0.03336386  0.0018027  -0.0593381  -0.06494875 -0.03257814  0.03479282
 -0.06730127  0.02027018 -0.07200469 -0.01203033 -0.03049812  0.0152317
  0.01343047  0.05434244  0.03032401 -0.00955622 -0.03677965 -0.05669002
 -0.00403236  0.05126869 -0.11419708  0.16615307  0.02051493  0.03080825
  0.0294996   0.12223063 -0.10379854 -0.12261698  0.01706665 -0.07893523
 -0.07897192 -0.01801085 -0.10613263 -0.07166662 -0.03787835 -0.08998851
 -0.07565644  0.05964809 -0.03545242  0.05228333  0.03405948 -0.0475216
  0.02286626  0.03111826  0.02490376  0.03725582  0.11626203 -0.00307333
  0.03425157 -0.05167219 -0.04232765  0.03996501 -0.03257963 -0.09101194
  0.0333107   0.04679427 -0.05590959  0.10991649  0.06580894  0.04804354
  0.02758543 -0.057156   -0.03973571 -0.02979081  0.03931472  0.05288552
 -0.01238688  0.09600443  0.11569116 -0.03563013 -0.01366004  0.04201121
 -0.05402599 -0.01080261 -0.00087962  0.05391148 -0.05529828  0.01983927
  0.02317429  0.01195612 -0.02524971 -0.12565361  0.02654742 -0.05879319
 -0.0255023   0.10664965  0.01442145 -0.01816013  0.00643078 -0.03549129
  0.09272593  0.04515116 -0.02551438 -0.047616    0.136681   -0.08364907
 -0.0763023   0.02688084  0.05825032  0.06142589  0.01967762  0.02745123
 -0.05653101  0.02109269 -0.04384974  0.09947056  0.01874313  0.04655955
  0.12130003  0.05638605  0.10129064 -0.05764392 -0.04670751 -0.01693143
 -0.03202759 -0.05261893 -0.00939573  0.10855307 -0.06607649 -0.005692
  0.010299   -0.13497059 -0.07673916 -0.0194075  -0.04968471 -0.09218837
  0.06027617  0.09054399 -0.00562163 -0.0208575   0.01306758  0.04781865
 -0.03493357 -0.02468323  0.02993818 -0.02120253  0.07464425 -0.00571638
 -0.09588838  0.01607293 -0.03953784 -0.10415982  0.0045896  -0.04080921
 -0.0152321   0.1495711   0.0969942  -0.17675646 -0.02073094  0.02025472
  0.00862345 -0.01065595 -0.01067358 -0.03456781 -0.01574122 -0.00988568
 -0.02273293  0.02579389 -0.00184724  0.00072742  0.00061091 -0.06204804
 -0.13288054 -0.023084    0.07622007  0.02777682 -0.03875524  0.02830531
  0.0190299  -0.02788347 -0.04297249  0.01132634 -0.11994946 -0.02524768
  0.05120984 -0.06719942  0.00856833  0.04299518 -0.05095441  0.11565189
  0.1008895   0.02875134 -0.06335233  0.0109929   0.00954604  0.07320318
  0.08693491 -0.00977756 -0.06898671  0.0233388  -0.06872299  0.0313792
  0.04988222 -0.00120362 -0.03855223 -0.0402755   0.01528711  0.01411317
  0.04585714 -0.03584175 -0.05536484 -0.03673518 -0.04535295  0.01668883
  0.00568568  0.01393303 -0.01512432  0.07490748  0.06252936 -0.01223504
  0.0063559   0.00837259  0.11595158  0.02613683 -0.01198711 -0.01141754
 -0.04288967  0.09134505  0.01493061 -0.02451384  0.01746074 -0.00290724
  0.02430766  0.05141103 -0.01254325 -0.08538302 -0.06105365  0.02613997
  0.06118597 -0.09266408  0.03108155  0.06405422  0.07295054 -0.05797282
 -0.04617298 -0.01108697 -0.02203737 -0.00736919  0.0572028   0.03975913
 -0.00717838  0.00260435 -0.02037444 -0.04411002  0.04874372 -0.08596696
 -0.09459512 -0.06317939 -0.05945202  0.04880878 -0.10007739  0.02990375
 -0.02048662 -0.03622325  0.04664787  0.0442193  -0.00187282 -0.05856924
 -0.03779399 -0.05530582  0.04290785 -0.0417159  -0.06670085  0.02938533
  0.00848846 -0.04419276 -0.01424101 -0.01650182 -0.04682277  0.14471814]
```

ภาพที่ 3.8 ตัวอย่างเวกเตอร์ของคำว่า *"Very"* ที่ได้จากการประมวลผลด้วย *GoogleNews-vectors-negative300*

วิธีหาค่า doc2vec

เนื่องจากบทวิจารณ์ของผู้ใช้บางบทวิจารณ์โดยเฉพาะบทวิจารณ์ที่อยู่บนแอปสโตร์มีจำนวนหลายประโยค บางบทวิจารณ์มีมากกว่าหนึ่งย่อหน้า การใช้ doc2vec ซึ่งจะมีการติดแทคให้กับแต่ละประโยคที่อยู่ในย่อหน้าเดียวกันเพื่อช่วยในการหาความหมายโดยรวมของทั้งย่อหน้า ผู้วิจัยจึงมีสมมติฐานที่ว่าการนำ doc2vec มาเป็นหนึ่งในฟีเจอร์การจำแนกประเภทอาจจะมีประสิทธิภาพในการจำแนกประเภทดีกว่า word2vec ในกรณีที่บทวิจารณ์ของผู้ใช้มีความยาวมาก สำหรับ doc2vec นั้นคล้ายกับ word2vec เพียงแต่เปลี่ยนจากการดูความสัมพันธ์ของเวกเตอร์ของคำใด ๆ กับคำรอบข้าง เป็นการดูความสัมพันธ์ของประโยคใด ๆ กับประโยครอบข้างในย่อหน้าหนึ่ง ๆ รูปแบบของเวกเตอร์ doc2vec มีลักษณะคล้ายกับภาพที่ 3.8 เพียงแต่ตัวเลขแต่ละค่าจะแตกต่างกันออกไป

เมื่อทำการเรียนรู้การจำแนกประเภทของบทวิจารณ์ของผู้ใช้จากข้อมูลข้างต้นที่มีทั้งหมด คือ รายการฟีเจอร์จำนวน 28 รายการฟีเจอร์ และอัลกอริทึมการจำแนกประเภทจำนวน 12 อัลกอริทึม ทำให้ได้โมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น bug report หรือ not bug report 336 โมเดล และโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น feature request หรือ not feature request 336 โมเดล ซึ่งจะแสดงรายละเอียดอีกครั้งในขั้นตอนถัดไป

3.3 ขั้นตอนการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

ในการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้นั้น ทำได้โดยการคำนวณค่าprecision, recall, f1-score, และ accuracy ของแต่ละโมเดล ซึ่งค่า precision, recall, f1-score สามารถใช้ไลบรารีการเรียนรู้ของเครื่องภาษา Python ที่ชื่อ Scikit-learn โดยใช้ฟังก์ชัน classification_report รับพารามิเตอร์จำเป็นสองพารามิเตอร์ คือ ค่าฉลากจริง และค่าฉลากที่โมเดลทำนายได้ และจะส่งกลับผลลัพธ์ออกมาเป็น Dictionary ซึ่งสามารถนำไปใช้งานต่อได้ถ้าต้องการ ตัวอย่างการใช้ฟังก์ชัน classification_report จากเว็บไซต์ scikit-learn.org เป็นดังภาพที่ 3.9

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

ภาพที่ 3.9 ตัวอย่างการใช้งานฟังก์ชัน *classification_report* จาก *scikit-learn.org*

ในทำนองเดียวกันค่า accuracy ก็สามารถหาได้โดยใช้ฟังก์ชัน accuracy_score จากไลบรารี Scikit-learn โดยส่งพารามิเตอร์จำเป็นสองตัวเช่นเดียวกับ classification_report ตัวอย่างการใช้ฟังก์ชัน accuracy_score จากเว็บไซต์ scikit-learn.org เป็นดังภาพที่ 3.10

```
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
```

ภาพที่ 3.10 ตัวอย่างการใช้งานฟังก์ชัน *accuracy_score* จาก *scikit-learn.org*

เมื่อทราบวิธีการวัดผลประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้แล้ว ก็นำโมเดลที่ได้จากการเทรนด้วยบทวิจารณ์ของผู้ใช้ประเภท bug report 256 บทวิจารณ์ not bug report 256 บทวิจารณ์จาก

ทั้งสองสโตร์ มาทดสอบด้วยชุดข้อมูลทดสอบประเภท bug report 114 บทวิจารณ์ not bug report 114 บทวิจารณ์ เพื่อหาค่า precision, recall, f1-score ของคลาส bug report รวมไปถึงหา accuracy ของโมเดล โดยอัตราส่วนระหว่างชุดข้อมูลเรียนรู้ และชุดข้อมูลทดสอบเป็น 70:30 และทำ 5-Folds Cross Validation เพื่อให้โมเดลถูกเทรนและทดสอบด้วยชุดข้อมูลที่มีความหลากหลาย แต่เนื่องจากจำนวนโมเดลการจำแนกประเภท bug report หรือ not bug report นั้นมีจำนวนมากถึง 336 โมเดล ผู้วิจัยจึงได้ทำการเลือกเพียงแค่โมเดลสิบอันดับแรกที่มีค่าความแม่นยำมากที่สุดจากการใช้ 5-Folds Cross Validation มาแสดงดังตารางที่ 3.5 และโมเดลอีกสิบดับแรกที่มีความแม่นยำมากที่สุดจากการแบ่งชุดข้อมูลเรียนรู้และชุดทดสอบเป็น 70:30 ดังตารางที่ 3.6 โดยจะสนใจผลลัพธ์ที่ได้จากการทำ Cross Validation ในตารางที่ 3.5 เป็นหลัก เนื่องจากเป็นข้อปฏิบัติที่ดีที่สุดในการวัดประสิทธิภาพของโมเดลการจำแนกประเภทเพราะโมเดลผ่านการเรียนรู้และทดสอบจากข้อมูลหลายชุด

ตารางที่ 3.5 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *bug report* หรือ *not bug report* ด้วยใช้ *5-Folds Cross Validation*

Classification Techniques	**Bug Report**			**Accuracy**
	Precision	***Recall***	***F1***	
BOW – stopwords + doc2vec + RandomForestClassifier	**0.8**	0.82	**0.81**	**0.8063**
BOW + lemmatization + LogisticRegression	0.77	0.83	0.8	0.7878
BOW + lemmatization + PoS + LogisticRegression	0.77	0.83	0.8	0.7878
BOW – stopwords + lemmatization + word2vec + VotingClassifier	0.74	**0.9**	**0.81**	0.7838
BOW – stopwords + lemmatization + rating + sentiScore + tense + PoS + ExtraTreesClassifier	0.76	0.84	0.79	0.7837
BOW + lemmatization + PoS + VotingClassifier	0.78	0.78	0.78	0.781
BOW + lemmatization + VotingClassifier	0.78	0.78	0.78	0.7797
BOW – stopwords + lemmatization + rating + sentiScore + tense + ExtraTreesClassifier	0.75	0.83	0.79	0.7797
BOW – stopwords + lemmatization + VotingClassifier	**0.8**	0.74	0.77	0.777
BOW – stopwords + lemmatization + rating + sentiScore + tense + LogisticRegression	0.77	0.79	0.78	0.777
...				
BOW + rating + sentiScore + tenses + GaussianNB	0.6296	0.8947	0.7391	0.6842

ตารางที่ 3.6 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *bug report* หรือ *not bug report* โดยใช้อัตราส่วนของข้อมูลระหว่าง *training* และ *test* เป็น *70:30*

Classification Techniques	Bug Report			Accuracy
	Precision	*Recall*	*F1*	
BOW + doc2vec + ExtraTreesClassifier	0.7983	0.8333	**0.8154**	**0.8063**
BOW – stopwords + doc2vec + ExtraTreesClassifier	0.7982	0.7982	0.7982	0.7928
BOW + doc2vec + VotingClassifier	**0.83**	0.7281	0.7757	0.7838
BOW – stopwords + lemmatization + word2vec + VotingClassifier	0.7391	0.8947	0.8095	0.7838
BOW + lemmatization + word2vec + LogisticRegression	0.76	0.8333	0.795	0.7793
BOW – stopwords + word2vec + LogisticRegression	0.7623	0.8158	0.7881	0.7748
BOW + lemmatization + pos + VotingClassifier	0.6623	0.8947	0.7612	0.7193
BOW + lemmatization + LinearSVC	0.654	**0.9123**	0.7619	0.7149
BOW – stopwords + ExtraTreesClassifier	0.6601	0.886	0.7566	0.7149
BOW – stopwords + lemmatization + ExtraTreesClassifier	0.656	0.9035	0.7601	0.7149
...				
BOW + rating + sentiScore + tenses + GaussianNB	0.6296	0.8947	0.7391	0.6842

จากตารางที่ 3.5 แสดงให้เห็นว่า ถุงคำ (Bag of Words) ทั้งที่สามารถหาได้จากการนับจำนวนความถี่ธรรมดาของคำที่แตกต่างกันที่ปรากฏอยู่ในบทวิจารณ์ของผู้ใช้ และการใช้ TF-IDF แปลงเวกเตอร์ของถุงคำที่นับได้เป็นเวกเตอร์ของค่า TF-IDF ซึ่งเป็นค่าจำนวนจริงที่มีค่าระหว่างศูนย์ถึงหนึ่ง เป็นฟีเจอร์ที่มีความสำคัญและมีผลต่อการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น bug report หรือ not bug report มากที่สุด เนื่องจากฟีเจอร์นี้เป็นหนึ่งในส่วนประกอบของทั้งสิบอันดับแรกของโมเดลที่มีประสิทธิภาพสูงที่สุด ในขณะที่ข้อมูล meta-data ของแอปพลิเคชัน คือ คะแนนความพึงพอใจ ความยาวของบทวิจารณ์ คะแนนอารมณ์ความรู้สึกจากบทวิจารณ์เพียงลำพังอย่างเดียวนั้นไม่สามารถใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้ได้อย่างมีประสิทธิภาพพอ ข้อสังเกตที่น่าสนใจก็คืออันดับที่หนึ่งและสี่ใช้เทคนิค word-embedding คือ doc2vec และ word2vec มาช่วยในการจำแนกประเภทบทวิจารณ์ได้อย่างมีประสิทธิภาพ กล่าวได้ว่าเทคนิค word-embedding ในที่นี้คือ doc2vec และ word2vec นั้นมีประสิทธิภาพในการจำแนกประเภทบทวิจารณ์สูงกว่าเทคนิคแบบดั้งเดิมอย่างการนำคำหยุดออก (stopwords removal) หรือการทำให้เป็นรากศัพท์ (lemmatization) เพียงอย่างเดียวโดยไม่ใช้ word-embedding เข้ามาช่วย ในขณะเดียวกัน part of speech นั้นไม่ใช่ฟีเจอร์ที่ส่งผลต่อการจำแนกประเภทบทวิจารณ์ซึ่งสังเกตได้จากอันดับที่สองและสามมีค่าวัดประสิทธิภาพเท่ากันไม่ว่าจะมีฟีเจอร์ pos รวมอยู่ด้วยหรือไม่ก็ตาม ข้อสังเกตอีกข้อหนึ่งก็คือการใช้อัลกอริทึมแบบรวมอย่างพวก ensemble methods เช่น RandomForestClassifier หรือ VotingClassifier มาช่วยในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น bug report หรือ not bug report นั้นมีประสิทธิภาพสูงกว่าการใช้เพียงอัลกอริทึมแบบเดี่ยวเพียงอัลกอริทึมใดอัลกอริทึมหนึ่งเท่านั้น

สำหรับวิธีการที่มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น bug report หรือ not bug report นั้น คือการใช้ Random Forest ensemble model ร่วมกับ doc2vec ที่ได้จากการแปลงข้อความในบทวิจารณ์ของผู้ใช้ที่เอาคำหยุดออกไปแล้ว โดยมีค่า precision, recall, f1-score, และ accuracy ประมาณ 80-82% นอกจากนี้อีกวิธีการหนึ่งที่น่าสนใจคือการนำข้อความที่อยู่ในบทวิจารณ์ของผู้ใช้มานำคำหยุดออกและทำให้เป็นรากศัพท์ หลังจากนั้นใช้ word2vec แปลงข้อความให้เป็นเวกเตอร์ของประโยค แล้วใช้อัลกอริทึมแบบรวมอย่าง VotingClassifier มาแยกประเภทบทวิจารณ์ ผลการทดลองพบว่าได้ค่า recall สูงถึง 90 % และผู้วิจัยยังได้เปรียบเทียบกับวิธีการของงานวิจัยที่ 2.2.1 ซึ่งวิธีการที่มีประสิทธิภาพสูงที่สุดของงานวิจัยดังกล่าวและได้ทดลองในโครงงานนี้ ได้แสดงไว้ที่แถวสุดท้ายของตารางที่ 3.5 จะเห็นว่าทั้งสิบอันดับแรกของวิธีการในโครงงานนี้มีประสิทธิภาพสูงกว่าวิธีการที่ดีที่สุดของงานวิจัยที่ 2.2.1 ทั้งสิ้น

สำหรับบทวิจารณ์ของผู้ใช้ประเภท feature request และ not feature request นั้น ได้นำโมเดลที่ได้จากการเทรนด้วยบทวิจารณ์ของผู้ใช้ประเภท feature request 207 บทวิจารณ์ not feature request 207 บทวิจารณ์จากทั้งสองสโตร์ มาทดสอบด้วยชุดข้อมูลทดสอบประเภท feature request 88 บทวิจารณ์ not feature request 114 บทวิจารณ์ เพื่อหาค่า precision, recall, f1-score ของคลาส feature request รวมไปถึงหา accuracy ของโมเดล โดยอัตราส่วนระหว่างชุดข้อมูลเรียนรู้ และชุดข้อมูลทดสอบเป็น 67:33 และทำ 5-Folds Cross Validation เพื่อให้โมเดลถูกเทรนและทดสอบด้วยชุดข้อมูลที่มีความหลากหลาย แต่เนื่องจากจำนวนโมเดลการจำแนกประเภท feature request หรือ not feature request นั้นมีจำนวนมากถึง 336 โมเดล ผู้วิจัยจึงได้ทำการเลือกเพียงแค่โมเดลสิบอันดับแรกที่มีค่าความแม่นยำมากที่สุดมาแสดงดังตารางที่ 3.7 และโมเดลอีกสิบดับแรกที่มีความแม่นยำมากที่สุดจากการแบ่งชุดข้อมูลเรียนรู้และชุดทดสอบเป็น 67:33 ดังตารางที่ 3.8 โดยจะสนใจผลลัพธ์ที่ได้จากการทำ Cross Validation ในตารางที่ 3.7 เป็นหลัก เนื่องจากเป็นข้อปฏิบัติที่ดีที่สุดในการวัดประสิทธิภาพของโมเดลการจำแนกประเภทเพราะใช้ข้อมูลหลายชุดในการเรียนรู้และทดสอบ

ตารางที่ 3.7 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *feature request* หรือ *not feature request* ด้วยวิธี *5-Folds Cross Validation*

Classification Techniques	**Feature Request**			**Accuracy**
	Precision	***Recall***	***F1***	
BOW + rating + sentiScore + XGBClassifier	**0.81**	0.83	**0.82**	**0.8279**
BOW + rating + sentiScore + PoS + XGBClassifier	**0.81**	0.83	**0.82**	**0.8279**
BOW – stopwords + lemmatization + rating + sentiScore + tense + PoS + ExtraTreesClassifier	0.8	0.85	**0.82**	0.8247
BOW – stopwords + lemmatization + rating + sentiScore + tense + PoS + XGBClassifier	0.8	0.85	**0.82**	0.8247
BOW + rating + sentiScore + tense + XGBClassifier	0.8	0.84	**0.82**	0.8231
BOW + rating + sentiScore + tense + PoS + XGBClassifier	0.8	0.84	**0.82**	0.8231
BOW – stopwords + lemmatization + rating + sentiScore + tense + ExtraTreesClassifier	0.79	0.85	**0.82**	0.8166
Rating + words length + tense + GradientBoostingClassifier	0.77	**0.9**	**0.82**	0.8166
BOW – stopwords + lemmatization + doc2vec + XGBClassifier	0.75	0.88	0.81	0.8108
BOW + lemmatization + word2vec + LinearSVC	0.69	0.88	0.77	0.7676
...				
BOW + rating + sentiScore + tenses + GaussianNB	0.6576	0.8295	0.7337	0.7376

ตารางที่ 3.8 โมเดลสิบอันดับแรกที่มีประสิทธิภาพสูงที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ *feature request* หรือ *not feature request* โดยใช้อัตราส่วนของข้อมูลระหว่าง *training* และ *test* เป็น *67:33*

Classification Techniques	Feature Request			Accuracy
	Precision	*Recall*	*F1*	
BOW + lemmatization + pos + ExtraTreesClassifier	**0.7238**	0.8636	0.7876	**0.797**
BOW – stopwords + doc2vec + GradientBoostingClassifier	**0.7238**	0.8941	**0.8**	0.7946
BOW – stopwords + doc2vec + ExtraTreesClassifier	0.7212	0.8824	0.7936	0.7892
BOW + ExtraTreesClassifier	0.7143	0.8523	0.7772	0.7871
BOW + rating + sentiScore + ExtraTreesClassifier	0.7027	0.8864	0.7839	0.7871
BOW + lemmatization + doc2vec + AdaBoostClassifier	0.7419	0.8118	0.7753	0.7838
BOW – stopwords + ExtraTreesClassifier	0.7157	0.8295	0.7684	0.7822
BOW – stopwords + word2vec + ExtraTreesClassifier	0.6937	0.9059	0.7857	0.773
BOW – stopwords + lemmatization + word2vec + ExtraTreesClassifier	0.6838	**0.9412**	0.7921	0.773
BOW + lemmatization + word2vec + LinearSVC	0.6944	0.8824	0.7772	0.7676
...				
BOW + rating + sentiScore + tenses + GaussianNB	0.6576	0.8295	0.7337	0.7376

จากตารางที่ 3.7 แสดงให้เห็นว่า ถุงคำ ทั้งที่สามารถหาได้จากการนับจำนวนความถี่ธรรมดาของคำที่แตกต่างกันที่ปรากฏอยู่ในบทวิจารณ์ของผู้ใช้ และการใช้ TF-IDF แปลงเวกเตอร์ของถุงคำที่นับได้เป็นเวกเตอร์ของค่า TF-IDF ซึ่งเป็นค่าจำนวนจริงที่มีค่าระหว่างศูนย์ถึงหนึ่ง เป็นฟีเจอร์ที่มีความสำคัญและมีผลต่อการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น feature request หรือ not feature request มากที่สุดเช่นเดียวกับการจำแนกประเภทบทวิจารณ์ว่าเป็น bug report หรือ not bug report เนื่องจากฟีเจอร์นี้เป็นหนึ่งในส่วนประกอบของทั้งสิบอันดับแรกของโมเดลที่มีประสิทธิภาพสูงที่สุด ข้อสังเกตที่น่าสนใจก็คือมีถึงแปดอันดับที่ไม่ได้ใช้เทคนิค word-embedding คือ doc2vec และ word2vec มาช่วยในการจำแนกประเภทบทวิจารณ์ กล่าวได้ว่าเทคนิค word-embedding ในที่นี้คือ doc2vec และ word2vec นั้นมีประสิทธิภาพในการจำแนกประเภทบทวิจารณ์น้อยกว่าเทคนิคแบบดั้งเดิมอย่างการนำคำหยุดออก หรือการทำให้เป็นรากศัพท์ ในขณะเดียวกัน part of speech นั้นไม่ใช่ฟีเจอร์ที่ส่งผลต่อการจำแนกประเภทบทวิจารณ์ซึ่งสังเกตได้จากอันดับที่หนึ่งและสองมีค่าวัดประสิทธิภาพเท่ากันไม่ว่าจะมีฟีเจอร์ pos รวมอยู่ด้วยหรือไม่ก็ตาม ข้อสังเกตอีกข้อหนึ่งก็คือการใช้อัลกอริทึมแบบรวมอย่างพวก ensemble methods เช่น ExtraTreesClassifier หรือ XGBClassifier มาช่วยในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น feature request หรือ not feature request นั้นมีประสิทธิภาพสูงกว่าการใช้เพียงอัลกอริทึมแบบเดี่ยวเพียงอัลกอริทึมใดอัลกอริทึมหนึ่งเท่านั้น

สำหรับวิธีการที่มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น feature request หรือ not feature request นั้น คือการใช้ XGBoost ensemble model ร่วมกับ TF-IDF ที่ได้จากการแปลงข้อความในบทวิจารณ์ของผู้ใช้ และใช้ข้อมูลเมตาเดตาของบทวิจารณ์ คือ คะแนนความพึงพอใจ และคะแนน

อารมณ์ความรู้สึก โดยมีค่า precision, recall, f1-score, และ accuracy ประมาณ 81-83% นอกจากนี้อีกวิธีการหนึ่งที่น่าสนใจคือ การใช้เพียงข้อมูลเมดาเดตาของบทวิจารณ์ คือ คะแนนความพึงพอใจ จำนวนคำศัพท์ที่อยู่ในบทวิจารณ์ และจำนวน tense ต่าง ๆ ที่อยู่ในบทวิจารณ์ หลังจากนั้นใช้อัลกอริทึมแบบรวมอย่าง GradientBoostingClassifier มาแยกประเภทบทวิจารณ์ ผลการทดลองพบว่าได้ค่า recall สูงถึง 90 % นอกจากนี้ผู้วิจัยยังได้เปรียบเทียบกับวิธีการของงานวิจัยที่ 2.2.1 ซึ่งวิธีการที่มีประสิทธิภาพสูงที่สุดของงานวิจัยดังกล่าวและได้ทดลองในโครงงานนี้ ได้แสดงไว้ที่แถวสุดท้ายของตารางที่ 3.6 จะเห็นว่าทั้งสิบอันดับแรกของวิธีการในโครงงานนี้ที่ใช้การผสมกันของฟีเจอร์ที่แตกต่างกันออกไปมีประสิทธิภาพสูงกว่าวิธีการที่ดีที่สุดของงานวิจัย 2.2.1 ทั้งสิ้น

3.4 ขั้นตอนการนำออกโมเดลเพื่อใช้ในการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

หลังจากที่ทำการวัดประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ทั้งประเภท bug report หรือ not bug report และประเภท feature request หรือ not feature request จนได้โมเดลที่มีประสิทธิภาพสูงที่สุดออกมาทั้งสองโมเดลแล้ว ในขั้นตอนนี้จะเป็นการนำออกโมเดลเหล่านั้นเพื่อเตรียมไว้ใช้ในบทที่ 5 ต่อไป ซึ่งการนำออกโมเดลนั้นสามารถทำได้โดยใช้ไลบรารีของภาษา Python ที่ชื่อว่า Pickle ทำการนำออกโมเดลออกมาเป็นไฟล์นามสกุล .pkl ซึ่งสามารถนำไฟล์เหล่านี้ไปนำเข้าให้กับเครื่องมือที่ใช้สร้างทิคเก็ตสำหรับระบบติดตามปัญหาได้ โค้ดสำหรับการนำออกโมเดลแสดงดังภาพที่ 3.11 โดยฟังก์ชันรับพารามิเตอร์สองตัว คือ โมเดลและชื่อไฟล์ที่จะถูกสร้างขึ้นเมื่อทำการนำออก

```
import pickle

def export_model(model, file_name):
    #serializing our model to a file called {file_name}.pkl
    pickle.dump(model, open('models/' + file_name + '.pkl', 'wb'))
```

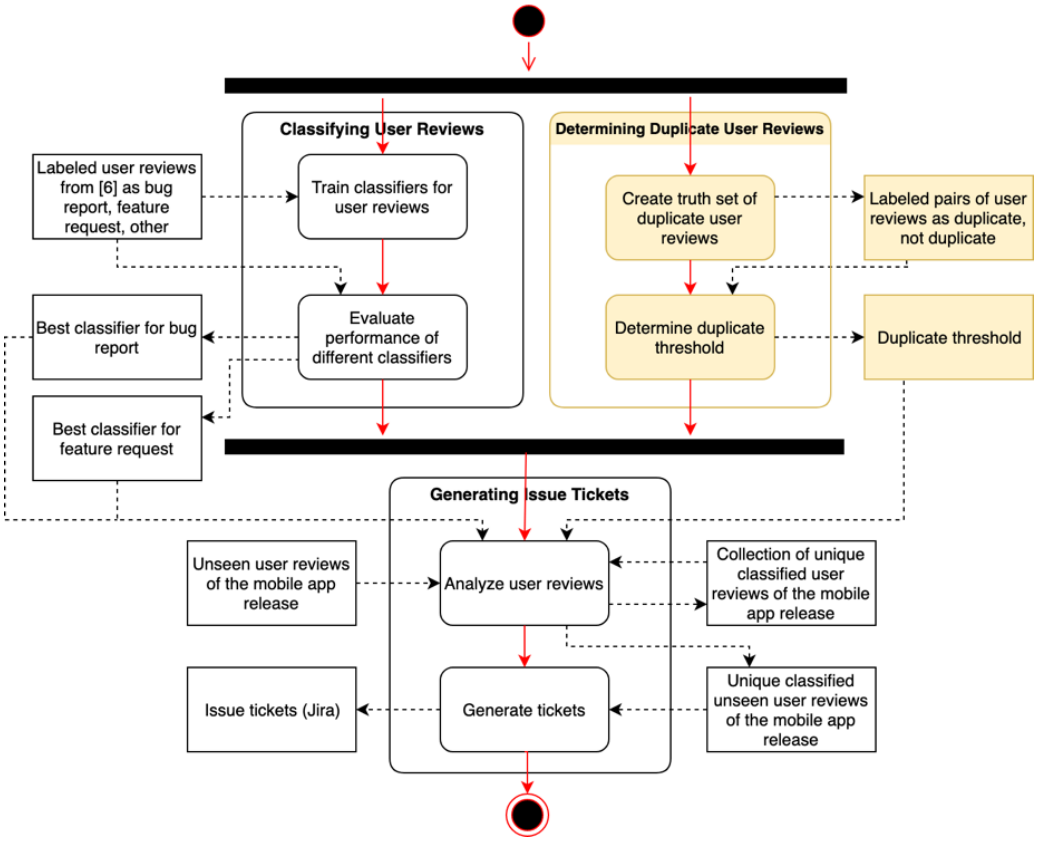
ภาพที่ 3.11 ตัวอย่างโค้ดการใช้ไลบรารี *Pickle* ในการนำออกโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้

บทที่ 4
การตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้

บทวิจารณ์ของผู้ใช้มีความหลากหลายมาก บางบทวิจารณ์มีความยาวน้อย บางบทวิจารณ์มีความยาวมาก แต่บ่อยครั้งที่บทวิจารณ์เหล่านี้พูดถึงเรื่องเดียวกัน พยายามอธิบายถึงสิ่ง ๆ เดียวกันไม่ว่าจะเป็นปัญหาที่พบในการใช้งานแอปพลิเคชัน หรือเป็นการร้องขอฟังก์ชันการทำงานที่ยังขาดหายไป เราสามารถนิยามบทวิจารณ์เหล่านี้ว่า “บทวิจารณ์ซ้ำซ้อน” ซึ่งก่อนที่จะทำการสร้างทิคเก็ตสำหรับระบบติดตามปัญหานั้น จะต้องมีการตรวจสอบบทวิจารณ์ซ้ำซ้อนก่อนเสมอ เพื่อป้องกันจำนวนทิคเก็ตที่จะถูกสร้างขึ้นซ้ำกันเป็นจำนวนมากบนระบบติดตามปัญหา วิธีการหนึ่งที่สามารถใช้ในการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ก็คือ การวัดความคล้ายคลึงกันเชิงความหมายของข้อความ (Semantic text similarity) แต่ก่อนที่เราจะสามารถบอกได้ว่าบทวิจารณ์สองบทวิจารณ์ใด ๆ มีความเหมือนหรือคล้ายกันมากพอที่จะสรุปว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่นั้น จำเป็นจะต้องหาค่าขีดแบ่ง (Threshold) ของความคล้ายที่มีความเหมาะสมก่อน ในบทนี้จะแบ่งเนื้อหาออกเป็นสองส่วนดังต่อไปนี้

1) การเตรียมชุดข้อมูลเพื่อทำการทดลองหาความคล้ายกันของบทวิจารณ์ของผู้ใช้
2) การหาค่า threshold ที่บ่งบอกว่าเป็นบทวิจารณ์ที่ซ้ำซ้อนหรือไม่

โดยภาพที่ 4.1 แสดงขั้นตอนการดำเนินงานอย่างละเอียดในบทนี้



ภาพที่ 4.1 ขั้นตอนการดำเนินงานแบบละเอียดของการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้

4.1 การเตรียมชุดข้อมูลเพื่อทำการทดลองหาความคล้ายกันของบทวิจารณ์ของผู้ใช้

ข้อมูลที่จะนำมาใช้ทดลองหาความคล้ายกันของบทวิจารณ์ของผู้ใช้ ผู้วิจัยได้ทำการรวบรวมบทวิจารณ์ของผู้ใช้ที่มีความแตกต่างกันอย่างเห็นได้ชัดจำนวน 94 บทวิจารณ์ จากแอปพลิเคชัน Facebook เวอร์ชัน 210.0 บนแอปสโตร์ เรียกข้อมูลชุดนี้ว่า base set และได้ทำการสุ่มบทวิจารณ์จำนวน 20 บทวิจารณ์จากแอปพลิเคชันเดียวกัน เวอร์ชันเดียวกัน เรียกว่า test set หลังจากนั้นผู้วิจัยได้เทียบบทวิจารณ์ใน test set ทีละหนึ่งบทวิจารณ์กับทุก ๆ บทวิจารณ์ใน base set เพื่อทำการติดฉลากให้กับบทวิจารณ์สองคู่ใด ๆ ว่ามีความซ้ำซ้อนกันหรือไม่ โดยระบุว่า Duplicate ถ้าผู้วิจัยอ่านแล้วพบว่าบทวิจารณ์คู่นั้นกล่าวถึงเรื่องเดียวกัน และระบุว่า Not duplicate ถ้าพบว่ากล่าวถึงคนละเรื่องกัน โดยตัวอย่างข้อมูล base set และ test set เป็นดังภาพที่ 4.2 และ 4.3 ตามลำดับ ซึ่งหลังจากที่ทำการติดฉลากบทวิจารณ์ครบทั้ง 1,880 คู่แล้ว ผู้วิจัยได้นำบทวิจารณ์ base set และ test set นี้ไปให้วิศวกรโมไบล์แอปพลิเคชันซอฟต์แวร์ที่มีประสบการณ์ในการพัฒนาโมไบล์แอปพลิเคชันห้าปีทำการตรวจสอบผลการติดฉลากเพื่อยืนยันความถูกต้อง รวมไปถึงเพิ่มความน่าเชื่อถือของการติดฉลากของข้อมูลอีกด้วย ตัวอย่างการติดฉลากของบทวิจารณ์คู่ใด ๆ เป็นดังภาพที่ 4.4 โดย B0 – B93 คือรหัสของบทวิจารณ์ที่อยู่ใน base set และ T0 – T19 คือรหัสของบทวิจารณ์ที่อยู่ใน test set เลข 1 ในแต่ละช่องหมายถึงบทวิจารณ์ i ใน base set ซ้ำกับบทวิจารณ์ j ใน test set และเลข 0 หมายถึงไม่ซ้ำกัน


```
{
    "id": "3885565204",
    "userName": "The royal high",
    "userUrl": "https://itunes.apple.com/us/reviews/id271402015",
    "version": "210.0",
    "score": 1,
    "title": "Cannot stream",
    "text": "What in the name of anything is wrong with Facebook on apple these days can’t stream a cough.
    Even with the fastest WiFi. Useless. Let’s go back to the old days",
    "url": "https://itunes.apple.com/us/review?id=284882215&type=Purple%20Software",
    "similarity_id": "B1"
},
{
    "id": "3885524621",
    "userName": "jj_le",
    "userUrl": "https://itunes.apple.com/us/reviews/id991899160",
    "version": "210.0",
    "score": 1,
    "title": "Moving phots to other albums",
    "text": "Why can’t I move mobile uploads to other photo albums on Facebook anymore.",
    "url": "https://itunes.apple.com/us/review?id=284882215&type=Purple%20Software",
    "similarity_id": "B2"
},
```


ภาพที่ 4.2 ตัวอย่างบทวิจารณ์ใน *base set*

```
{
  "id": "3878331158",
  "userName": "eshaa□",
  "userUrl": "https://itunes.apple.com/us/reviews/id986693297",
  "version": "210.0",
  "score": 1,
  "title": "Can't go live",
  "text": "Fb won't let me go live 🙄🤦🏾♀",
  "url": "https://itunes.apple.com/us/review?id=284882215&type=Purple%20Software",
  "similarity_id": "T0"
},
{
  "id": "3878330205",
  "userName": "meliss pink",
  "userUrl": "https://itunes.apple.com/us/reviews/id957661148",
  "version": "210.0",
  "score": 1,
  "title": "Facebook not working",
  "text": "My Facebook is not working",
  "url": "https://itunes.apple.com/us/review?id=284882215&type=Purple%20Software",
  "similarity_id": "T1"
},
```


ภาพที่ 4.3 ตัวอย่างบทวิจารณ์ใน *test set*

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
B0	0	0	0	0	0	0	0	0	0	0	0	0
B1	1	0	0	0	0	0	0	0	0	0	0	0
B2	0	0	0	0	0	0	0	0	0	0	0	0
B3	0	0	0	0	0	0	0	0	0	0	0	0
B4	0	0	0	0	0	0	0	0	0	0	0	0
B5	0	0	0	0	0	0	0	0	0	0	0	0
B6	0	1	0	0	0	0	0	0	0	0	0	0
B7	0	0	0	0	0	0	0	0	0	0	0	0
B8	0	0	0	0	0	0	0	0	0	0	0	0
B9	0	0	0	0	0	0	0	0	0	0	0	0
B10	0	0	0	0	0	0	0	0	0	0	0	0
B11	0	1	0	0	0	1	1	0	0	0	0	0
B12	0	0	0	0	0	0	0	0	0	0	0	0
B13	0	0	0	0	0	0	0	0	0	0	0	0
B14	0	0	0	0	0	0	0	0	0	0	0	0
B15	0	0	0	0	0	0	0	0	0	0	0	0
B16	0	0	0	0	0	0	0	0	0	0	0	0
B17	0	0	0	0	0	0	0	0	0	0	0	0
B18	0	0	0	0	0	0	0	0	0	0	0	0
B19	0	0	0	0	0	0	0	0	0	0	0	0
B20	0	0	0	0	0	0	0	0	0	0	0	0
B21	0	0	0	0	0	0	0	0	0	0	0	0
B22	0	0	0	0	0	0	0	0	0	0	0	0
B23	0	0	0	0	0	0	0	0	0	0	0	0
B24	0	0	0	0	0	0	0	0	0	0	0	0
B25	0	0	0	0	0	0	0	0	0	0	0	0
B26	0	0	0	0	0	0	0	0	0	0	0	0
B27	0	0	0	0	0	0	0	0	0	0	0	0
B28	0	0	0	0	0	0	0	0	0	0	0	0
B29	0	0	0	0	0	0	0	0	0	0	0	0
B30	0	0	0	0	0	0	0	0	0	0	0	0
B31	0	0	0	0	0	0	0	0	0	0	0	0

ภาพที่ 4.4 ตัวอย่างการติดฉลากระหว่างบทวิจารณ์ใน *base set* และ *test set*

ในการหาความคล้ายคลึงกันเชิงความหมายของข้อความสองข้อความใด ๆ นั้นทำได้โดยการใช้ Universal Sentence Encoder [20] ซึ่งเป็นหนึ่งในวิธีการที่มีประสิทธิภาพ ทำการแปลงบทวิจารณ์ทั้งหมดใน base set และ

test set เป็นเวกเตอร์หลายมิติที่มีขนาด 512 มิติเท่ากัน โดยไม่สนใจเรื่องความยาวของบทวิจารณ์ หลังจากนั้นทำการหาระยะห่างของเวกเตอร์ด้วยการคำนวณ Cosine Similarity ระหว่างเวกเตอร์ใน base set และ test set ทีละคู่ โดยค่าที่ได้กลับมาจะมีค่าอยู่ระหว่าง 0 ถึง 1 โดย 0 หมายถึงแตกต่างกันมากที่สุด และ 1 หมายถึงคล้ายคลึงกันมากที่สุด ทำไปจนครบทั้ง 1,880 คู่ ตัวอย่างฟังก์ชันของการแปลงข้อความเป็นเวกเตอร์ด้วย Universal Sentence Encoder ที่ทำงานบนโมดูล Tensorflow เป็นดังภาพที่ 4.5 และตัวอย่างฟังก์ชันการหาค่า Cosine Similarity ระหว่างสองเวกเตอร์ใด ๆ เป็นดังภาพที่ 4.6

```
def get_features(texts):
    if type(texts) is str:
        texts = [texts]
    with tf.Session() as sess:
        sess.run([tf.global_variables_initializer(), tf.tables_initializer()])
        return sess.run(embed(texts))
```

ภาพที่ 4.5 ตัวอย่างฟังก์ชันการแปลงข้อความเป็นเวกเตอร์ด้วย *Universal Sentence Encoder* ที่ทำงานบน *Tensorflow*

```
def cosine_similarity(v1, v2):
    mag1 = np.linalg.norm(v1)
    mag2 = np.linalg.norm(v2)
    if (not mag1) or (not mag2):
        return 0
    return np.dot(v1, v2) / (mag1 * mag2)
```

ภาพที่ 4.6 ตัวอย่างฟังก์ชันการหาค่า *Cosine Similarity* ระหว่างสองเวกเตอร์ใด ๆ

4.2 การหาค่าขีดแบ่งสำหรับบทวิจารณ์ซ้ำซ้อน

หลังจากที่ได้ค่า Cosine Similarity ที่อยู่ระหว่าง 0 ถึง 1 ของบทวิจารณ์ทั้ง 1,880 คู่แล้ว ผู้วิจัยได้ทำการกำหนดค่า threshold หลายค่าระหว่าง 0 ถึง 1 เพื่อใช้เป็นตัวบอกว่าบทวิจารณ์คู่ใด ๆ ซ้ำซ้อนกันหรือไม่ โดยมีเงื่อนไขคือถ้าบทวิจารณ์คู่ใดมีค่า Cosine Similarity มากกว่าหรือเท่ากับค่า threshold ที่กำหนด ให้ถือว่าบทวิจารณ์คู่นั้นซ้ำซ้อนกัน แต่ถ้า Cosine Similarity มีค่าน้อยกว่า threshold ที่กำหนด ให้ถือว่าบทวิจารณ์คู่นั้นแตกต่างกัน หลังจากกำหนดเงื่อนไขและ thresholds ต่าง ๆ เรียบร้อยแล้ว ผู้วิจัยได้ทำการเปรียบเทียบผลการตรวจสอบความซ้ำซ้อนกันของบทวิจารณ์ที่ได้จากการตัดสินใจของมนุษย์ในข้อ 4.1 กับผลการทดลองในข้อนี้ที่ระดับ thresholds ต่าง ๆ และทำการวัดประสิทธิภาพของค่า threshold โดยการใช้ Confusion Matrix หาค่าตัววัดต่อไปนี้มาทำการวิเคราะห์

True Positive (TP) คือ ทั้งมนุษย์และการทดลองระบุว่าบทวิจารณ์คู่นั้นซ้ำซ้อน

True Negative (TN) คือ ทั้งมนุษย์และการทดลองระบุว่าบทวิจารณ์คู่นั้นไม่ซ้ำซ้อน

False Positive (FP) คือ มนุษย์ระบุว่าบทวิจารณ์คู่นั้นไม่ซ้ำซ้อน แต่การทดลองระบุว่าซ้ำซ้อน

False Negative (FN) คือ มนุษย์ระบุว่าบทวิจารณ์คู่นั้นซ้ำซ้อน แต่การทดลองระบุว่าไม่ซ้ำซ้อน

Accuracy คือ ค่าความแม่นยำของการทำนายว่าบทวิจารณ์คู่นั้นซ้ำซ้อนกันหรือไม่

เนื่องจากในการใช้งานเครื่องมือที่ช่วยในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหานั้น สามารถที่จะมีทิคเก็ตที่ซ้ำซ้อนกันถูกสร้างได้บ้าง แต่จะต้องป้องกันกรณีที่ทิคเก็ตไม่ซ้ำซ้อนแต่ไม่ถูกสร้างให้ได้มากที่สุด (คือกรณี False Positive) เนื่องจากในอุตสาหกรรมการทำงานด้านการพัฒนาซอฟต์แวร์ การไม่ได้แก้ไขปัญหาที่ผู้ใช้งานระบุมาถือเป็นความผิดพลาดร้ายแรง และอาจส่งผลเสียต่อธุรกิจจนถึงขั้นผู้ใช้งานเลิกใช้งานแอปพลิเคชันนั้นเลยก็ได้ถ้าข้อผิดพลาดนั้นร้ายแรง ดังนั้น threshold ที่มีความเหมาะสมที่จะเป็นตัวบ่งบอกว่าบทวิจารณ์ของผู้ใช้คู่ใด ๆ นั้นซ้ำซ้อนกันหรือไม่ จะต้องมีค่า false positive ที่ต่ำที่สุด และมีค่า accuracy ที่สูงที่สุด เพื่อให้มั่นใจได้ว่าเครื่องมือที่ช่วยสร้างทิคเก็ตสำหรับระบบติดตามปัญหานั้นจะเกิดการตกหล่นในการสร้างทิคเก็ตที่ไม่ซ้ำซ้อนได้น้อยที่สุด ผลลัพธ์แสดงดังตารางที่ 4.1 ต่อไปนี้

ตารางที่ 4.1 ค่าประสิทธิภาพการวัดความซ้ำซ้อนของบทวิจารณ์ที่ *thresholds* ต่าง ๆ

Threshold	TN	TP	FN	FP	Accuracy
0	1	25	0	1854	0.0138
0.1	32	25	0	1823	0.0303
0.2	172	25	0	1683	0.1048
0.3	543	23	2	1312	0.3011
0.4	1028	18	7	827	0.5564
0.5	1448	14	11	407	0.7776
0.6	1734	7	18	121	0.9261
0.7	1838	2	23	17	0.9787
0.75	1852	0	25	3	0.9851
0.76	1854	0	25	1	0.9862
0.77	1854	0	25	1	0.9862
0.78	**1855**	**0**	**25**	**0**	**0.9867**
0.79	1855	0	25	0	0.9867
0.8	1855	0	25	0	0.9867
0.9	1855	0	25	0	0.9867
1	1855	0	25	0	0.9867

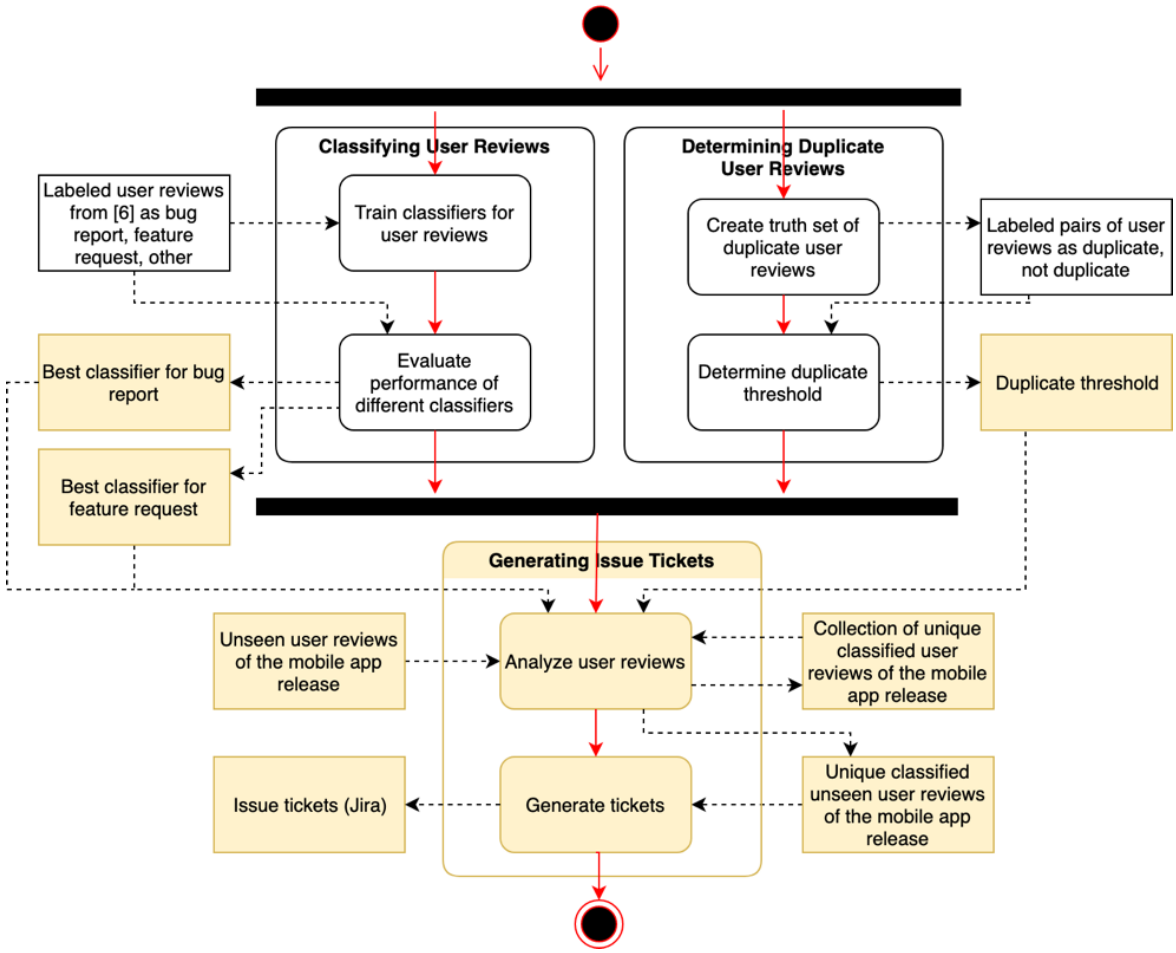
จะเห็นได้ว่าเมื่อค่า threshold เพิ่มขึ้น ค่า false positive จะมีค่าลดลง และค่า accuracy จะมีค่าเพิ่มขึ้นจนกระทั่งไม่เปลี่ยนแปลงที่ threshold 0.78 ดังนั้นจะได้ว่าค่า threshold ที่เหมาะสมที่สุดในการระบุว่าบทวิจารณ์ของผู้ใช้คู่ใด ๆ มีความซ้ำซ้อนกันหรือไม่ในโครงงานนี้คือ 0.78 ซึ่งจะถูกนำไปใช้อีกครั้งในบทที่ 5 ต่อไป

บทที่ 5
การสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้

ในบทนี้จะอธิบายเกี่ยวกับการนำผลลัพธ์ต่าง ๆ ที่ได้จากขั้นตอนก่อนหน้า มาใช้ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา และอธิบายเกี่ยวกับเครื่องมือที่ช่วยในการสร้างทิคเก็ตแบบอัตโนมัติที่ได้พัฒนาขึ้นมาสำหรับโครงงานมหาบัณฑิตนี้ ระบบติดตามปัญหาที่ได้เลือกมาเป็นกรณีศึกษาคือ Jira เนื่องจากได้รับความนิยมอย่างมากในอุตสาหกรรมการพัฒนาซอฟต์แวร์ และยังมีช่องทางให้ระบบภายนอกสามารถติดต่อมายัง Jira เพื่อทำการบริหารจัดการโครงการได้ เช่น การสร้างทิคเก็ต การอัปเดตสถานะของทิคเก็ต เป็นต้น โดย Jira มี REST APIs ที่สามารถติดต่อได้ผ่านโปรโตคอล HTTPS สำหรับเนื้อหาในบทนี้จะแบ่งออกเป็นสามส่วนดังต่อไปนี้

1) การรวบรวมบทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน
2) การวิเคราะห์บทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน
3) การสร้างทิคเก็ตสำหรับระบบติดตามปัญหา Jira

โดยภาพที่ 5.1 แสดงขั้นตอนการดำเนินงานอย่างละเอียดในบทนี้



ภาพที่ 5.1 ขั้นตอนการดำเนินงานแบบละเอียดของการทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้

5.1 การรวบรวมบทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน

สำหรับการรวบรวมบทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ เนื่องจากมีบทวิจารณ์ที่อยู่บนแอปสโตร์และเพลย์สโตร์เป็นจำนวนมาก การรวบรวมโดยใช้การวิเคราะห์ด้วยตาทีละบทวิจารณ์นั้นไม่สามารถทำได้อย่างมีประสิทธิภาพ เนื่องจากต้องใช้ระยะเวลาและจำนวนคนมาก ทางผู้วิจัยจึงใช้โอเพนซอร์ซไลบรารีภาษา Node.js สองไลบรารี คือ app-store-scraper (https://github.com/facundoolano/app-store-scraper#reviews) สำหรับดึงบทวิจารณ์ของผู้ใช้ที่อยู่บนแอปสโตร์ และ google-play-scraper (https://github.com/facundoolano/google-play-scraper#reviews) สำหรับดึงบทวิจารณ์ของผู้ใช้ที่อยู่บนเพลย์สโตร์ โดยระบุพารามิเตอร์ทั้งหมดสี่ค่า คือ ไอดีของแอปพลิเคชัน ประเทศที่ต้องการดึงข้อมูลบทวิจารณ์ของผู้ใช้ (ค่าเริ่มต้นเป็น US) ลำดับหน้าของสโตร์ที่ต้องการดึงข้อมูล (หน้า 1 ถึงหน้า 10) และประเภทของบทวิจารณ์ของผู้ใช้ที่จะทำการดึง แบ่งเป็นสองประเภท คือ บทวิจารณ์ใหม่ล่าสุด (Recent) และบทวิจารณ์ที่มีประโยชน์ (Helpful) ตัวอย่างฟังก์ชันสำหรับการดึงข้อมูลบทวิจารณ์ของผู้ใช้ที่อยู่บนแอปสโตร์และเพลย์สโตร์ กับผลลัพธ์ที่ไลบรารีส่งกลับมาให้เป็นดังภาพที่ 5.2, 5.3, 5.4, และ 5.5 ตามลำดับ

```
const store = require('app-store-scraper');

store.reviews({
  appId: 'com.facebook.Facebook',
  sort: store.sort.RECENT,
  page: 1
})
.then(console.log)
.catch(console.log);
```

ภาพที่ 5.2 ฟังก์ชันการดึงข้อมูลบทวิจารณ์ของผู้ใช้ที่อยู่บนแอปสโตร์ของแอปพลิเคชัน *Facebook*

```
{
  id: '4253959763',
  userName: 'djdnfjfkkf',
  userUrl: 'https://itunes.apple.com/us/reviews/id771825748',
  version: '223.0',
  score: 3,
  title: 'Notification HELL',
  text: 'So lately I've been getting notifications every time one of my ' +
    'friends posts or reposts anything on Facebook and it is super ' +
    'annoying. I've tried turning off the notifications however ' +
    'they keep coming. Please fix this or tell me how I can fix it.',
  url: 'https://itunes.apple.com/us/review?id=284882215&type=Purple%20Software'
}
```

ภาพที่ 5.3 ตัวอย่างบทวิจารณ์ของผู้ใช้ที่ได้จากการดึงข้อมูลบนแอปสโตร์ของแอปพลิเคชัน *Facebook*

```
const store = require('google-play-scraper');

store.reviews({
  appId: 'com.facebook.katana',
  page: 1,
  sort: gplay.sort.RECENT
}).then(console.log, console.log);
```

ภาพที่ 5.4 ฟังก์ชันการดึงข้อมูลบทวิจารณ์ของผู้ใช้ที่อยู่บนเพลย์สโตร์ของแอปพลิเคชัน *Facebook*

```
  {
    id: 'gp:AOqpTOG482iAwxwD8CBQNLUIwW4CvUH6J5DQZP6p8noz-tCXEvzu_avE206HJosqJ6pepaFa9e-hKQsFSlZK-A',
    userName: 'J Rivera',
    userImage: 'https://lh3.googleusercontent.com/a-/AAuE7mDhnRQPUXDPNIFCYK9wkOSy0dYIZZtFzx8OjBJshw=w96-h96-p',
    date: 'June 3, 2019',
    url: 'https://play.google.com/store/apps/details?id=com.facebook.katana&reviewId=Z3A6QU9xcFRPRzQ4MmlBd3h3RDhDQlFOTFVJd1c0Q3ZVSDZKNURRWlA2cDhub3otdENYRXZ6dV9hdkUyMDZISm9zcUo2cGVwYUZhOWUtaEtRc0ZTbFpLLUE',
    score: 1,
    title: '',
    text: 'now I can even see FB market cuz it keeps crashing and ' +
      'I can see any post . please fix asap . everything else ' +
      'is fine beside the market place and everyone is selling',
    replyDate: undefined,
    replyText: undefined
  }
```

ภาพที่ 5.5 ตัวอย่างบทวิจารณ์ของผู้ใช้ที่ได้จากการดึงข้อมูลบนเพลย์สโตร์ของแอปพลิเคชัน *Facebook*

โดยเครื่องมือช่วยในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหาที่ผู้วิจัยได้พัฒนาขึ้นในโครงงานมหาบัณฑิตนี้ เป็นโปรแกรม Command line ที่รองรับพารามิเตอร์แพลตฟอร์มของแอปพลิเคชันที่ต้องการทำการสร้างทิคเก็ต และแอปพลิเคชันไอดีของแอปพลิเคชันที่ต้องการสร้างทิคเก็ต ดังภาพที่ 5.6 เป็นการเลือกแพลตฟอร์ม iOS และแอปพลิเคชัน Facebook เพื่อทำการดึงข้อมูลบทวิจารณ์ของผู้ใช้ล่าสุดมาเก็บไว้ใน memory ของโปรแกรม

```
Kittisaks-MacBook-Pro:jira-issues-generator kittisakp$ node main.js
Please select platform that you want. If you want to exit, type 0 then press enter.
1. iOS
2. Android
1
Please input your Bundle ID.
com.facebook.Facebook
```

ภาพที่ 5.6 โปรแกรม *Command line* ที่รองรับแพลตฟอร์มและแอปพลิเคชันไอดีที่ต้องการดึงข้อมูลบทวิจารณ์ของผู้ใช้

5.2 การวิเคราะห์บทวิจารณ์ของผู้ใช้ที่เป็นบทวิจารณ์ใหม่ของแอปพลิเคชัน

หลังจากที่ได้บทวิจารณ์ของผู้ใช้จากข้อ 5.1 ในขั้นตอนนี้จะเป็นวิเคราะห์เกี่ยวกับบทวิจารณ์ที่ได้มา ไม่ว่าจะเป็นการจำแนกประเภทบทวิจารณ์ของผู้ใช้ว่าเป็น bug report หรือ feature request หรือการกรองบทวิจารณ์ซ้ำซ้อนออกก่อนที่จะทำการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา โดยภายในโปรแกรม Command line ได้มีการนำเข้าสองโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ที่มีประสิทธิภาพสูงที่สุดในการจำแนกบทวิจารณ์ว่าเป็น bug report หรือ feature request ที่ได้จากบทที่ 3 ที่อยู่ภายในรูปแบบไฟล์ .pkl และทำการ decode ไฟล์เหล่านั้นให้กลับเป็น prediction code ภาษา Python ด้วยไลบรารี Pickle ตัวอย่างไฟล์ .pkl ที่ถูกนำเข้าในโปรแกรม Command line และฟังก์ชันการ decode กลับเป็นโค้ด Python แสดงดังภาพที่ 5.7 และ 5.8 ตามลำดับ

ภาพที่ 5.7 ตัวอย่างไฟล์ *.pkl* ที่ถูกนำเข้าในโปรแกรม *Command line*

```
import pickle

classifier_model = pickle.load(open("./ClassifierModel/lemmatized_logistic_regression_bug.pkl", "rb"))
```

ภาพที่ 5.8 ฟังก์ชันการ *decode* ไฟล์ *.pkl* กลับเป็น *prediction code* ภาษา *Python*

หลังจากนั้นบทวิจารณ์ของผู้ใช้ที่ถูกเก็บอยู่ใน memory ของโปรแกรม Command line ที่ได้จากขั้นตอน 5.1 จะถูกนำมาเข้ากระบวนการ pre-processing ทำ text cleansing เพื่อลด noise และโอกาสที่โมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้จะทำนายผลลัพธ์ผิดพลาดลง โดย text cleansing ที่ทำได้แก่ การทำให้เป็นตัวอักษรภาษาอังกฤษตัวพิมพ์เล็กทั้งหมด การนำคำหยุดออก และการทำให้เป็นรากศัพท์ โดยฟังก์ชันการทำงานเป็นดังภาพที่ 5.9

```
def remove_stopwords(tokens):
    return [item for item in tokens if item not in stopwords]

def lemmatize(tokens):
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(item) for item in tokens]
```

ภาพที่ 5.9 ฟังก์ชันการนำคำหยุดออก และการทำให้เป็นรากศัพท์

เมื่อทำ text cleansing ให้กับบทวิจารณ์ของผู้ใช้เรียบร้อยแล้ว ก็จะทำ feature extraction เช่นเดียวกับที่ทำในขั้นตอนการพัฒนาโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ เพื่อนำไปเข้าโมเดลการจำแนกประเภททั้งสองโมเดล คือ โมเดล bug report หรือ not bug report และ โมเดล feature request หรือ not feature request เพื่อทำนายว่าบทวิจารณ์ต่าง ๆ นั้นเป็นประเภท bug report หรือ feature request เพื่อที่จะได้แยกประเภทของทิคเก็ตที่จะสร้างบนระบบติดตามปัญหาได้อย่างถูกต้อง ในกรณีที่โมเดลทั้งสองทำนายออกมาว่าเป็น bug report และ feature request ก็จะทำการดูค่า probability ของการทำนายของแต่ละคลาส ถ้าคลาสใดมีค่าสูงกว่า ก็จะระบุว่าบทวิจารณ์เป็นประเภทคลาสนั้น ๆ ตัวอย่างฟังก์ชันการทำนายประเภทของบทวิจารณ์ของผู้ใช้เป็นดังภาพที่ 5.10 และ 5.11

```
def classify(user_reviews_data_frame):
    classifier_model = pickle.load(open("./ClassifierModel/
    w2v_stopwords_lemmatization_voting", "rb"))
    user_reviews_data_frame['stopwords_removal_lemmatization'] =
    user_reviews_data_frame['text'].apply(lambda text: '
    '.join(lemmatize(remove_stopwords(tokenize(text)))))
    test_tokenized = user_reviews_data_frame.apply(lambda r:
    w2v_tokenize_text(r['stopwords_removal_lemmatization']), axis=1).values
    X_test_word_average = word_averaging_list(test_tokenized)
    y_pred = classifier_model.predict(X_test_word_average)
    return y_pred
```

ภาพที่ 5.10 ตัวอย่างฟังก์ชันการทำนายประเภทของบทวิจารณ์ว่าเป็น *bug report* หรือ *not but report*

```
def classify(user_reviews_data_frame):
    classifier_model = pickle.load(open("./ClassifierModel/
    lemmatization_pos_extra_tree", "rb"))
    user_reviews_data_frame['lemmatized_comment'] =
    user_reviews_data_frame['text'].apply(lambda text: '
    '.join(lemmatize(tokenize(text))))

    user_reviews_data_frame['noun'] = compute_pos_text(user_reviews_data_frame,
    'noun', 'lemmatized_comment')
    user_reviews_data_frame['pron'] = compute_pos_text(user_reviews_data_frame,
    'pron', 'lemmatized_comment')
    user_reviews_data_frame['verb'] = compute_pos_text(user_reviews_data_frame,
    'verb', 'lemmatized_comment')
    user_reviews_data_frame['adj'] = compute_pos_text(user_reviews_data_frame,
    'adj', 'lemmatized_comment')
    user_reviews_data_frame['adv'] = compute_pos_text(user_reviews_data_frame,
    'adv', 'lemmatized_comment')

    y_pred = classifier_model.predict(user_reviews_data_frame)
    return y_pred
```

ภาพที่ 5.11 ตัวอย่างฟังก์ชันการทำนายประเภทของบทวิจารณ์ว่าเป็น *feature request* หรือ *not feature request*

หลังจากแยกประเภทของบทวิจารณ์แล้ว โปรแกรม Command line ก็จะทำการตรวจสอบบทวิจารณ์ซ้ำซ้อนและกรองออก ทำได้โดยการสร้าง collection ขึ้นมาไว้สำหรับเก็บ Universal Sentence Encoder vector ของบทวิจารณ์ที่ซ้ำซ้อน เมื่อมีบทวิจารณ์ใหม่เข้ามา ก็ทำการแปลงเป็น vector ขนาดมิติเดียวกัน แล้วทำการหาค่า Cosine Similarity จากบทที่ 4 ที่ผ่านมา ถ้าค่าที่ได้ออกมามีค่ามากกว่าหรือเท่ากับ 0.78 ให้สรุปว่าเป็นบทวิจารณ์ซ้ำซ้อน โปรแกรม Command line ก็จะไม่สนใจบทวิจารณ์นั้น ๆ แต่ถ้าออกมาแล้วมีค่าต่ำกว่า 0.78

โปรแกรม Command line ก็จะเตรียมสร้างทิคเก็ตให้สำหรับบทวิจารณ์นั้น ๆ พร้อมกับอัปเดต collection ด้วยการเพิ่มข้อมูลเวกเตอร์ของบทวิจารณ์อันใหม่เข้าไปด้วย

5.3 การสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจิรา

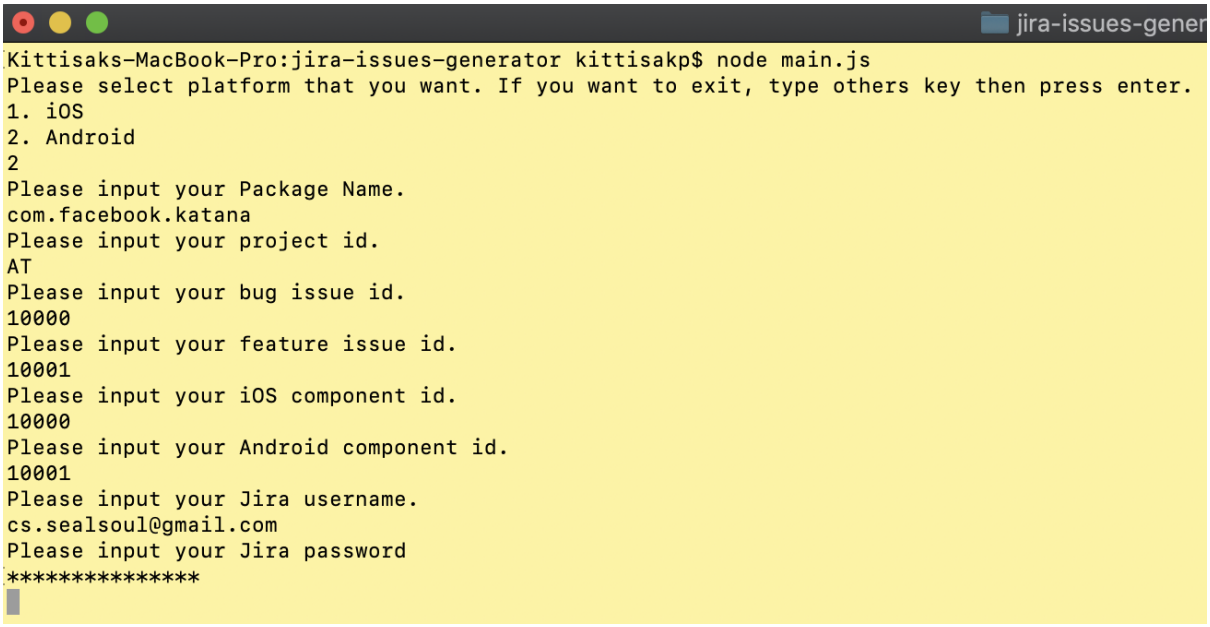
อย่างที่กล่าวไปตอนต้นของบทที่ 5 ว่า Jira เปิด REST APIs ให้ระบบภายนอกสามารถเชื่อมต่อผ่านโปรโตคอล HTTPS เข้ามาเพื่อทำการบริหารจัดการโครงการพัฒนาซอฟต์แวร์ได้ ซึ่งรวมไปถึงการสร้างทิคเก็ตใหม่ ๆ ขึ้นบน Jira board ด้วย โดยฟิลด์ที่จำเป็นจะต้องส่งไปเป็น request body ที่ Jira APIs เพื่อทำการสร้างทิคเก็ตจากบทวิจารณ์ของผู้ใช้ในโครงงานมหาบัณฑิตแสดงดังตารางที่ 5.1

ตารางที่ 5.1 ฟิลด์ที่จำเป็นต้องใช้ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา *Jira* เพื่อเชื่อมต่อกับ *Jira APIs*

ฟิลด์	คำอธิบาย
project_key	รหัสของโครงการซึ่งจะแตกต่างกันไปตามแต่ละ Jira board
issue_type_id	รหัสของประเภททิคเก็ตซึ่งจะแตกต่างกันไปตามแต่ละ Jira board โดยในโครงงานมหาบัณฑิตนี้จะใช้ bug_issue_type_id และ feature_issue_type_id ทั้งสิ้นสองรหัส
priority_id	รหัสของระดับความสำคัญของทิคเก็ต ซึ่งถ้าไม่ระบุค่าที่เป็น default จะเป็นปานกลาง (Medium)
label	ฉลากของทิคเก็ตซึ่งเอาไว้ช่วยในการค้นหาทิคเก็ต สามารถใส่อะไรก็ได้ แต่ในโครงงานมหาบัณฑิตนี้จะใช้ iOS หรือ Android ขึ้นอยู่กับ component_id
component_id	รหัสกลุ่มของทิคเก็ตซึ่งจะแตกต่างกันไปตามแต่ละ Jira board ในโครงงานมหาบัณฑิตนี้จะมีการสร้างรหัสกลุ่มเพื่อแสดงถึงกลุ่มของ iOS และ Android เพื่อช่วยในการค้นหาทิคเก็ต
summary	หัวข้อของทิคเก็ตที่ได้จากบทวิจารณ์ของผู้ใช้ หรือหาจาก text summarization
description	คำอธิบายรายละเอียดของทิคเก็ตประกอบด้วย ข้อความในบทวิจารณ์ของผู้ใช้ เลขเวอร์ชันของแอปพลิเคชัน วันที่ทำการสร้างทิคเก็ต และลิงค์สำหรับเปิดดูคอมเมนท์ต้นฉบับบนสโตร์

เนื่องจาก project_key, issue_type_id, priority_id, label, และ component_id จะมีความแตกต่างกันตามแต่ละ Jira board ผู้ที่นำเครื่องมือไปใช้จึงจำเป็นต้องทำกรอกข้อมูลเหล่านี้ผ่านโปรแกรม Command line เอง ดังภาพที่ 5.12 สำหรับ summary นั้น ใน iOS ใช้ title ที่ได้จากการดึงบทวิจารณ์ของผู้ใช้ในขั้นตอนที่ 5.1 มาเป็นหัวข้อของทิคเก็ต แต่สำหรับ Android นั้นจำเป็นจะต้องหาหัวข้อให้กับทิคเก็ต เนื่องจากเพลย์สโตร์ไม่ได้คืนค่า title กลับมาให้ในขั้นตอนที่ 5.1 ดังนั้นผู้วิจัยจึงเลือกใช้หลักการสรุปความที่เรียกว่า Extractive Text Summarization มาช่วยในการสรุปความจากบทวิจารณ์ของผู้ใช้ที่มาจากเพลย์สโตร์เพื่อนำไปใช้เป็นหัวข้อของทิคเก็ตที่จะสร้างขึ้น โดยเลือกจากประโยคที่มีความสำคัญมากที่สุดในบทวิจารณ์ของผู้ใช้เพียงประโยคเดียวมาเป็นตัวแทน หลักการแบบง่าย คือ การนับจำนวนความถี่ของคำที่แตกต่างกันที่ปรากฏอยู่ในบทวิจารณ์นั้น ๆ หารด้วยจำนวนความถี่ของคำที่ปรากฏบ่อยครั้งมากที่สุด เพื่อให้ได้ค่าน้ำหนักของคำ จากนั้นแยกบทวิจารณ์ออกเป็นประโยคย่อย ๆ แล้วทำการหาผลรวมของค่าน้ำหนักของคำที่แตกต่างกันในทุก ๆ ประโยค ประโยคที่ได้ค่าผลรวมมากที่สุด

จะถูกนำมาเป็นหัวข้อของบทวิจารณ์ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหาต่อไป ฟังก์ชันที่เกี่ยวข้องกับ extractive text summarization แสดงดังภาพที่ 5.13 และ 5.14



ภาพที่ 5.12 ข้อมูลที่ผู้ใช้เครื่องมือจำเป็นต้องระบุเองจากตารางที่ *5.1*

```
def count_word_frequencies(formatted_text):
    word_frequencies = {}
    for word in nltk.word_tokenize(formatted_text):
        if word not in stopwords:
            if word not in word_frequencies.keys():
                word_frequencies[word] = 1
            else:
                word_frequencies[word] += 1

    return word_frequencies

def count_sentence_scores(sentence_list, word_frequencies):
    sentence_scores = {}
    for sent in sentence_list:
        for word in nltk.word_tokenize(sent.lower()):
            if word in word_frequencies.keys():
                if len(sent.split(' ')) < 30:
                    if sent not in sentence_scores.keys():
                        sentence_scores[sent] = word_frequencies[word]
                    else:
                        sentence_scores[sent] += word_frequencies[word]

    return sentence_scores
```

ภาพที่ 5.13 ฟังก์ชันการคำนวณหาความถี่ของคำที่แตกต่างกันทั้งหมดในบทวิจารณ์ใด ๆ *(count_word_frequencies)* และฟังก์ชันสำหรับคำนวณคะแนนความสำคัญของแต่ละประโยคที่อยู่บทวิจารณ์ *(count_sentence_scores)*

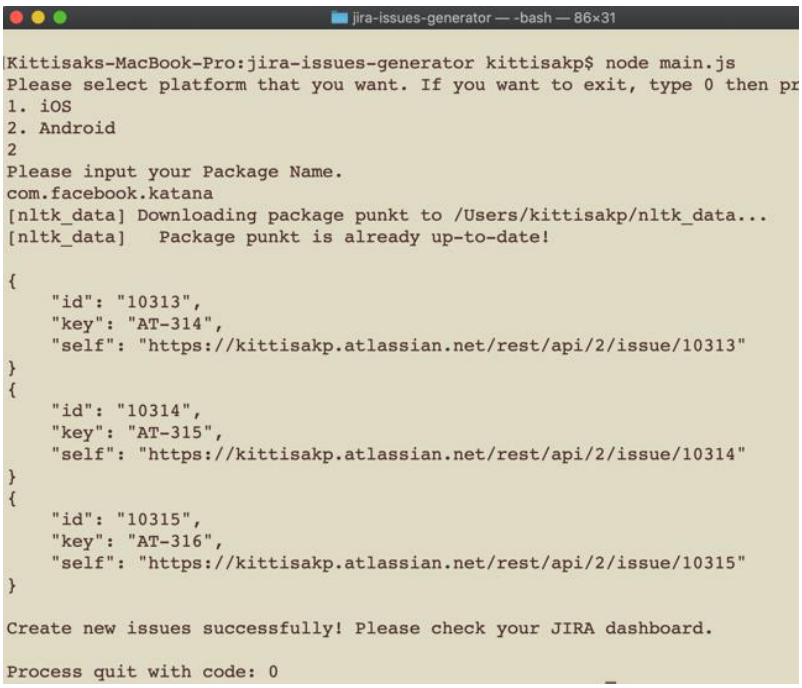
```python
def summarize(data_frame_column, expected_number_of_word):
    summaries = []

    for raw_text in data_frame_column:
        # Removing special characters and digits
        formatted_text = re.sub('[^a-zA-Z]', ' ', raw_text)
        formatted_text = re.sub(r'\s+', ' ', formatted_text)

        sentence_list = nltk.sent_tokenize(raw_text)
        word_frequencies = count_word_frequencies(formatted_text)
        maximum_frequncy = max(word_frequencies.values())

        for word in word_frequencies.keys():
            word_frequencies[word] = (word_frequencies[word] / maximum_frequncy)

        sentence_scores = count_sentence_scores(sentence_list, word_frequencies)

        summary_sentences = heapq.nlargest(expected_number_of_word, sentence_scores, key=sentence_scores.get)
        summary = ' '.join(summary_sentences)
        summaries.append(summary)

    return summaries
```

ภาพที่ 5.14 ฟังก์ชันการเลือกประโยคที่สำคัญที่สุดจากบทวิจารณ์ของผู้ใช้ออกมา *n* ประโยค โดยในโครงงานมหาบัณฑิตนี้มีค่า *n = 1*

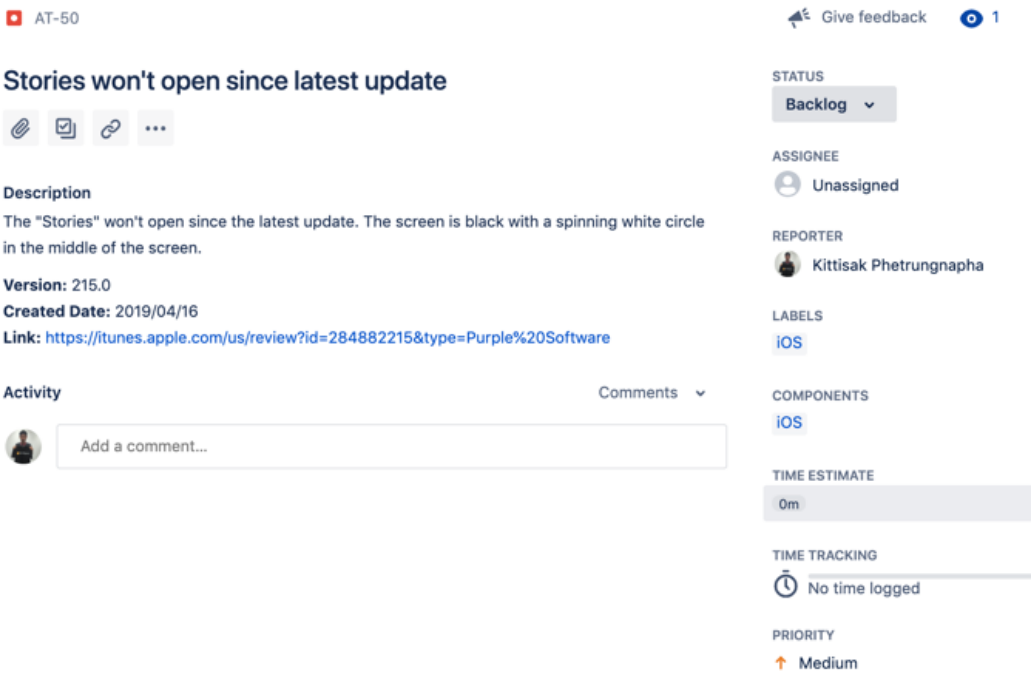
ซึ่งหลังจากได้ค่าฟิลด์ต่าง ๆ ที่จำเป็นต้องใช้ในการเชื่อมต่อกับ Jira APIs ครบแล้ว โปรแกรม Command line ก็จะทำการสร้าง request body โดยบรรจุฟิลด์ต่าง ๆ ให้อยู่ในรูปแบบโครงสร้าง JSON ดังภาพที่ 5.15 และทำการ post request body นี้ไปทำการสร้างทิคเก็ตบน Jira board โดยมีข้อมูลตามที่ระบุไว้ในฟิลด์แต่ละฟิลด์ เมื่อทำการสร้างทิคเก็ตเสร็จสิ้นแล้วโปรแกรม Command line จะแสดงผลลัพธ์กลับมาเป็น JSON response ดังภาพที่ 5.16 และเมื่อเข้าไปที่ Jira board ทิคเก็ตที่ถูกสร้างสามารถแยกประเภทได้จากมุมซ้ายบนของทิคเก็ตในภาพที่ 5.17 โดย bug report จะเป็นไอคอนสีแดง ส่วน feature request จะเป็นไอคอนสีเขียว หลังจากนี้นักพัฒนาก็สามารถที่จะเข้าไปที่ Jira board แล้ว assign ตนเองให้กับทิคเก็ตใด ๆ ที่ต้องการจะทำ เปลี่ยนสถานะจากสถานะเริ่มต้นที่ Backlog เป็น In Progress หรือติดตามระยะเวลาที่ใช้ทำงานได้

```python
def generate_ticket_on_jira(row):
    # Prepare payload data
    project_key = constant.project_key

    if row.title != '':
        summary = row.title
    elif row.summarized_title != '':
        summary = row.summarized_title
    else:
        summary = row.text

    description = row.text + '\n\n' + '*Version:* ' + row.version + '\n' + '*Created Date:* ' + row.created_date + '\n' + '*Link:* ' + row.url
    priority = constant.priority_id
    label = 'iOS' if row.platform == 'iOS' else 'Android'
    component_id = constant.ios_components_id if row.platform == 'iOS' else constant.android_companents_id

    if row.bug_predict == 'Bug':
        issue_type_id = constant.bug_issue_type_id
    elif row.feature_predict == 'Feature':
        issue_type_id = constant.feature_issue_type_id
    else:
        return

    payload = json.dumps({
      "fields": {
        "project": {
          "key": project_key
        },
        "summary": summary,
        "description": description,
        "issuetype": {
          "id": issue_type_id
        },
        "priority": {
          "id": priority
        },
        "labels": [
            label
        ],
        "components": [
          {
            "id": component_id
          }
        ]
      }
    })

    # Connect Jira's apis
    response = requests.request(
       "POST",
       constant.url_issue,
       data=payload,
       headers=constant.headers
    )

    # Show result
    print(json.dumps(json.loads(response.text), sort_keys=True, indent=4, separators=(",", ": ")))
```

ภาพที่ 5.15 ฟังก์ชันการเชื่อมต่อกับ *Jira APIs* ในการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา *Jira*

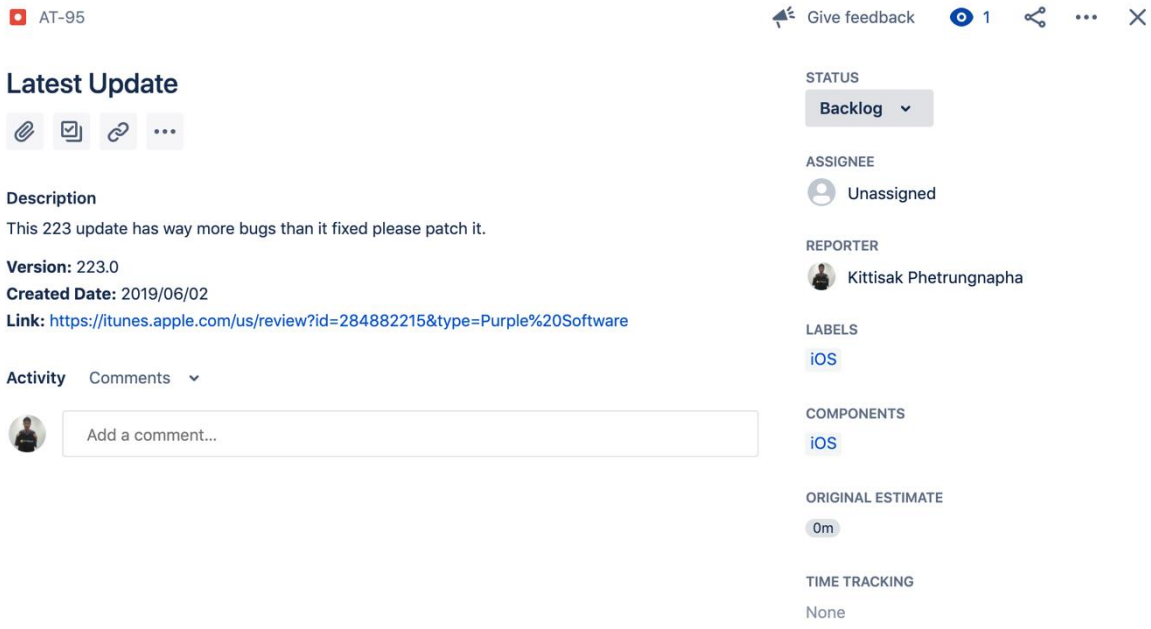


ภาพที่ 5.16 โปรแกรม *Command line* แสดงผลลัพธ์หลังจากทำการสร้างทิคเก็ตบน *Jira board* แล้ว



ภาพที่ 5.17 ตัวอย่างทิคเก็ตบน *Jira board* ที่อธิบายเกี่ยวกับปัญหาการใช้งานแอปพลิเคชัน

สำหรับทิคเก็ตที่ถูกสร้างขึ้นสามารถนำไปใช้งานจริงได้มากน้อยเพียงใดนั้นขึ้นอยู่กับบทวิจารณ์ของผู้ใช้ที่เป็นข้อมูลนำเข้าเป็นหลัก บทวิจารณ์ที่มีการระบุหัวข้อของบทวิจารณ์ และมีการเขียนคำอธิบายที่ละเอียดสื่อถึงปัญหาที่พบในการใช้งานหรือความต้องการฟังก์ชันใหม่ได้มากพอ นักพัฒนาก็สามารถนำทิคเก็ตนั้นไปทำงานต่อได้ทันที เนื่องจากสามารถอ่านเข้าใจได้ว่าปัญหาที่จำเป็นต้องแก้ไขหรือสิ่งที่ต้องเพิ่มเติมให้กับแอปพลิเคชันคืออะไร แต่บางบทวิจารณ์ที่ไม่มีหัวข้อ มีคำอธิบายที่สั้นจนเกินไป มีการใช้คำศัพท์ที่เฉพาะกลุ่ม หรือผิดหลักไวยากรณ์มาก ก็อาจทำให้นักพัฒนานำทิคเก็ตไปใช้งานจริงได้น้อย ส่วนการนำคอมเมนท์ของผู้ใช้มาใส่เป็นคำอธิบายของทิคเก็ตโดยตรงก็ยังมีปัญหาอยู่บ้างในกรณีที่บทวิจารณ์นั้นมีความยาวมาก เพราะจะทำให้รายละเอียดของทิคเก็ตนั้นยาวตามไปด้วย อาจจะทำให้นักพัฒนาไม่ให้ความสนใจในการอ่านทำความเข้าใจมากพอ หรือการสกัดประโยคอย่างง่ายเพื่อเลือกประโยคที่สำคัญที่สุดจากคอมเมนท์มาเป็นหัวข้อให้กับทิคเก็ตแทนนั้น ในกรณีที่แต่ละประโยคในย่อหน้าของบทวิจารณ์มีการใช้คำที่แตกต่างกัน เท่า ๆ กัน ก็อาจจะทำให้สูญเสียประโยคสำคัญบางประโยคที่จะนำไปเป็นหัวข้อของทิคเก็ตได้ แต่อย่างไรก็ตามนักพัฒนาก็ยังสามารถที่จะอ่านบทวิจารณ์ต้นฉบับได้จากลิงค์ต้นฉบับของแอปสโตร์หรือเพลย์สโตร์ที่อยู่ในคำอธิบายทิคเก็ต ตัวอย่างทิคเก็ตที่สามารถนำไปใช้งานได้จริงแสดงดังภาพที่ 5.17 และตัวอย่างทิคเก็ตที่นำไปใช้งานจริงได้น้อยแสดงดังภาพที่ 5.18 ที่อธิบายเพียงแค่มีข้อผิดพลาดเพิ่มมากขึ้น แต่ไม่ได้บอกว่าเป็นข้อผิดพลาดอะไร



ภาพที่ 5.18 ตัวอย่างทิคเก็ตที่นำไปใช้งานจริงได้น้อย

บทที่ 6
บทสรุปโครงงานและข้อเสนอแนะ

6.1 สรุปผลโครงงานมหาบัณฑิต

โครงงานมหาบัณฑิตนี้ประกอบด้วยขั้นตอนหลักทั้งสิ้นสามขั้นตอน โดยในแต่ละขั้นตอนจะได้ผลลัพธ์ออกมาเพื่อนำไปใช้ในขั้นตอนถัดไปได้ ขั้นตอนแรกเป็นการพัฒนาและทดสอบประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ มีการเก็บรวบรวมข้อมูลที่ถูกติดฉลากจากงานวิจัย [22] ประเภท bug report, not bug report, feature request, และ not feature request เพื่อนำมาเป็นชุดข้อมูลเรียนรู้และทดสอบให้กับโมเดลการจำแนกประเภทที่พัฒนาขึ้น มีการใช้ฟีเจอร์ที่สกัดได้จากข้อความในบทวิจารณ์ผู้ใช้ เช่น Bag of Word, TF-IDF, word2vec, doc2vec ฟีเจอร์ที่ได้จากเมตาเดตาของบทวิจารณ์ของผู้ใช้ เช่น คะแนนความพึงพอใจ คะแนนอารมณ์ความรู้สึกที่ได้จากการวิเคราะห์บทวิจารณ์ ความยาวของข้อความในบทวิจารณ์ จำนวน part of speech ในบทวิจารณ์ มีการผสมกันของทั้งฟีเจอร์ที่สกัดได้จากข้อความและฟีเจอร์จากเมตาเดตา โดยอัลกอริทึมการเรียนรู้ของเครื่องแบบเดี่ยวและแบบรวมได้ถูกนำมาใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้ ผลลัพธ์ได้ออกมาเป็นโมเดลการจำแนกประเภท bug report หรือ not bug report และโมเดลการจำแนกประเภท feature request หรือ not feature request ซึ่งโมเดลการจำแนกบทวิจารณ์ประเภท bug report ที่ดีที่สุด คือ การใช้ Random Forest ensemble model ร่วมกับ doc2vec ที่ได้จากการแปลงข้อความในบทวิจารณ์ของผู้ใช้ที่เอาคำหยุดออกไปแล้ว ส่วนโมเดลที่ดีที่สุดในการจำแนกบทวิจารณ์ประเภท feature request คือ การใช้ XGBoost ensemble model ร่วมกับ TF-IDF ที่ได้จากการแปลงข้อความในบทวิจารณ์ของผู้ใช้ และใช้ข้อมูลเมตาเดตาของบทวิจารณ์ คือ คะแนนความพึงพอใจ และคะแนนอารมณ์ความรู้สึก ซึ่งทั้งสองโมเดลจะถูกนำไปใช้ในขั้นตอนการพัฒนาเครื่องมือสร้างทิคเก็ตสำหรับระบบติดตามปัญหา

ขั้นตอนที่สองเป็นการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้ เนื่องจากบทวิจารณ์ของผู้ใช้จำนวนมากมีโอกาสที่จะซ้ำซ้อนกัน จึงจำเป็นต้องกรองบทวิจารณ์เหล่านั้นออก โดยทำการหาค่าความเหมือนกันเชิงความหมายของข้อความโดยเทคนิค universal sentence encoder มีการทดลองเพื่อหาค่าขีดแบ่งที่มีจำนวน false positive น้อยที่สุด และ accuracy มากที่สุด เพื่อลดโอกาสที่เครื่องมือสร้างทิคเก็ตสำหรับระบบติดตามปัญหาไม่ทำการสร้างทิคเก็ตที่จริง ๆ แล้วไม่ซ้ำซ้อนกัน โดยข้อมูลในการทดลองได้รวบรวมมาจากแอปพลิเคชัน Facebook ผลลัพธ์ที่ได้ออกมาคือค่าขีดแบ่งที่เหมาะสมที่สุดในการระบุว่าบทวิจารณ์คู่ใด ๆ มีความซ้ำซ้อนกันหรือไม่คือ 0.78 ซึ่งจะถูกนำไปใช้ในขั้นตอนถัดไป

ขั้นตอนสุดท้ายเป็นการสร้างทิคเก็ตสำหรับระบบติดตามปัญหาจากบทวิจารณ์ของผู้ใช้ โดยมีการรวบรวมบทวิจารณ์ของผู้ใช้ใหม่ที่ยังไม่เคยถูกวิเคราะห์มาก่อนจากแอปสโตร์และเพลย์สโตร์ ซึ่งเครื่องมือถูกพัฒนาออกมาในรูปแบบโปรแกรม Command line ที่สามารถระบุพารามิเตอร์แพลตฟอร์มของบทวิจารณ์ ไอดีของแอปพลิเคชันที่ต้องการดึงข้อมูบทวิจารณ์ และค่าที่เฉพาะเจาะจงกับจิราบอร์ด เช่น โปรเจคไอดี หลังจากได้ข้อมูลบทวิจารณ์ใหม่มาแล้ว เครื่องมือก็จะนำโมเดลการจำแนกประเภทที่ได้จากขั้นตอนแรกมาจำแนกประเภทบทวิจารณ์ว่าเป็น bug report หรือ feature request เพื่อแยกประเภทของทิคเก็ตจิราที่จะสร้างขึ้นออกเป็นหมวดหมู่แล้วนำข้อมูลจากบทวิจารณ์ที่จำเป็นสำหรับการสร้างทิคเก็ต เช่น หัวข้อ คำอธิบาย เวอร์ชันที่พบปัญหา เป็นต้น มาเชื่อมต่อกับ APIs

ของจิราได้ออกมาเป็นชุดของทิคเก็ตบนบอร์ดของจิราที่ผ่านการกรองบทวิจารณ์ซ้ำซ้อนแล้ว นอกจากนี้ยังมีการสกัดประโยคอย่างง่ายจากคอมเมนท์เพื่อนำมาสร้างเป็นหัวข้อของทิคเก็ตในบางบทวิจารณ์ที่ไม่มีข้อมูลหัวข้อของบทวิจารณ์ด้วย กระบวนการทั้งหมดเหล่านี้ช่วยลดภาระของนักพัฒนาโมไบล์แอปพลิเคชันในการวิเคราะห์บทวิจารณ์ด้วยสายตา รวมไปถึงลดระยะเวลาที่ต้องใช้ในการสร้างทิคเก็ตบนจิราอีกด้วย

6.2 ข้อจำกัดในโครงงานมหาบัณฑิต

6.2.1 การทดลองทั้งหมดเกี่ยวกับบทวิจารณ์ของผู้ใช้ในโครงงานมหาบัณฑิตนี้รองรับเพียงแค่ภาษาอังกฤษเท่านั้น

6.2.2 จำนวนบทวิจารณ์ของผู้ใช้ที่ถูกติดฉลากเรียบร้อยแล้ว ยังมีจำนวนน้อยอยู่ โดย bug report มีเพียง 740 บทวิจารณ์ และ feature request มีเพียง 616 บทวิจารณ์

6.2.3 จำนวนผู้เชี่ยวชาญที่ช่วยในการยืนยันการติดฉลากว่าบทวิจารณ์คู่ใด ๆ นั้นมีความซ้ำซ้อนกันหรือไม่มีเพียงคนเดียวเท่านั้น

6.2.4 จำนวนข้อมูลที่ใช้ทดสอบหาค่า threshold ที่เหมาะสมที่สุดในการบ่งบอกว่าบทวิจารณ์คู่ใด ๆ มีความซ้ำซ้อนกันหรือไม่ ยังมีจำนวนน้อยอยู่เพียง 1,880 คู่บทวิจารณ์

6.2.5 ระดับความสำคัญของทิคเก็ตสำหรับระบบติดตามปัญหาที่ถูกสร้างขึ้นยังมีค่าพื้นฐานคือ ปานกลาง

6.2.6 ทิคเก็ตที่ถูกสร้างขึ้นยังไม่สามารถระบุรายละเอียดปลีกย่อยลงไปถึงขั้น task หรือ sub-task ได้ นักพัฒนายังต้องเป็นผู้ระบุเองอยู่บ้าง

6.2.7 การนำ comment จากบทวิจารณ์ของผู้ใช้ที่มีความยาวมากมาใส่เป็นคำอธิบายของทิคเก็ตโดยตรงนั้น บางครั้งไม่เหมาะสมเนื่องจากทำให้คำอธิบายทิคเก็ตยาวเกินไป

6.2.8 การสกัดประโยคอย่างง่าย แล้วเลือกประโยคที่มีความสำคัญมากที่สุดมาไม่กี่ประโยค ถ้าบทวิจารณ์ของผู้ใช้มีความยาวมาก บางครั้งไม่สามารถครอบคลุมใจความทั้งหมดของย่อหน้าได้

6.2.9 การพิจารณาความคล้ายกันระหว่างสองบทวิจารณ์ใน base set และ test set โดยผู้วิจัยและวิศวกรโมไบล์แอปพลิเคชัน เพื่อระบุว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่ ทำในระดับบทวิจารณ์

6.2.10 ข้อมูล base set และ test set ที่นำมาทดลองเพื่อหาค่าขีดแบ่งในการระบุว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่ มีแหล่งที่มาจากเพียงแอปพลิเคชันเดียว คือ Facebook บนแอปสโตร์

6.3 ข้อเสนอแนะ

6.3.1 ทำการตรวจสอบไวยากรณ์ของบทวิจารณ์ของผู้ใช้ก่อนเทรนโมเดลการจำแนกประเภท

6.3.2 เปลี่ยนคำสแลงให้เป็นคำปกติก่อนเทรนโมเดลการจำแนกประเภท

6.3.3 รวบรวมบทวิจารณ์ของผู้ใช้ที่ติดฉลากแล้วจากแหล่งอื่น ๆ เพื่อเพิ่มจำนวนชุดข้อมูลสำหรับใช้เทรนโมเดลการจำแนกประเภทให้มากขึ้น

6.3.4 การสกัดฟีเจอร์ใหม่ ๆ ที่มีความหลากหลายมากขึ้นจากบทวิจารณ์ของผู้ใช้

6.3.5 ทำการทดลองจำแนกประเภทบทวิจารณ์ของผู้ใช้ในภาษาอื่น ๆ นอกเหนือจากภาษาอังกฤษ

6.3.6 ทำการวิเคราะห์ระดับความสำคัญของทิคเก็ตที่จะถูกสร้างขึ้นจากบทวิจารณ์ของผู้ใช้

6.3.7 ทำการสกัดประโยคจากบทวิจารณ์ของผู้ใช้เพื่อนำมาเป็นหัวข้อให้กับทิคเก็ตโดยใช้เทคนิค Abstractive Text Summarization

6.3.8 นำเครื่องมือที่ได้พัฒนาขึ้นในโครงงานมหาบัณฑิตนี้ไปทดลองใช้งานจริงในอุตสาหกรรมเพื่อสำรวจความพึงพอใจของผู้ใช้งาน

6.3.9 การพิจารณาความคล้ายกันระหว่างสองบทวิจารณ์ใน base set และ test set โดยผู้เชี่ยวชาญเพื่อระบุว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่ ควรพิจารณาในระดับประโยค เนื่องจากในบทวิจารณ์หนึ่ง ๆ อาจมีใจความสำคัญหลายเรื่อง

6.3.10 ใช้ข้อมูล base set และ test set ในการทดลองหาค่าขีดแบ่งว่าเป็นบทวิจารณ์ซ้ำซ้อนหรือไม่จากหลายแอปพลิเคชัน และหลายสโตร์

บรรณานุกรม

1. Wikipedia. *Smartphone*. [cited 2018 Dec 23]; Available from: https://en.wikipedia.org/wiki/Smartphone.
2. Apple. *App Store*. [cited 2018 Dec 23]; Available from: https://www.apple.com/th/ios/app-store/.
3. Google. *Play Store*. [cited 2018 Dec 23]; Available from: https://play.google.com/store?hl=en.
4. Wikipedia. *User Reviews*. [cited 2018 Dec 23]; Available from: https://en.wikipedia.org/wiki/User_review.
5. Atlassian. *Agile*. [cited 2019 Jan 15]; Available from: https://www.atlassian.com/agile/.
6. Berkeley. *Artificial Intelligence*. [cited 2019 Jan 15]; Available from: http://aima.cs.berkeley.edu/.
7. ACM. *Data Mining*. [cited 2019 Jan 15]; Available from: https://dl.acm.org/citation.cfm?id=2778285/.
8. University, C.M. *Machine Learning*. [cited 2019 Jan 15]; Available from: http://www.cs.cmu.edu/~tom/mlbook.html/.
9. Saravanan, R. and P. Sujatha, *A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification*. 2018. 945-949.
10. Pacharawongsakda, E. *Naïve Bayes introduction*. [cited 2018 Dec 20]; Available from: http://dataminingtrend.com/2014/naive-bayes/.
11. Babelcoder. *Introduction to k-Nearest Neighbors*. [cited 2019 Jun 5]; Available from: https://www.babelcoder.com/blog/posts/k-nearest-neighbors#%E0%B8%97%E0%B8%9A%E0%B8%97%E0%B8%A7%E0%B8%99%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%AB%E0%B8%A1%E0%B8%B2%E0%B8%A2%E0%B8%82%E0%B8%AD%E0%B8%87-Classification.
12. Dataminingtrend. *How to create Decision Tree*. [cited 2019 Jun 5]; Available from: http://dataminingtrend.com/2014/decision-tree-model/.

13. Rojarath, A., W. Songpan, and C. Pong-inwong. *Improved ensemble learning for classification techniques based on majority voting*. in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2016.
14. Atlassian. *JIRA Software*. [cited 2019 Jan 15]; Available from: https://www.atlassian.com/software/jira/.
15. Wikipedia. *Issue Tracking System*. [cited 2019 Jan 7]; Available from: https://en.wikipedia.org/wiki/Issue_tracking_system/.
16. Scrum.org. *What is a Product Backlog?* [cited 2019 Jan 7]; Available from: https://www.scrum.org/resources/what-is-a-product-backlog/.
17. Atlassian. *What is a board?* [cited 2019 Jan 9]; Available from: https://confluence.atlassian.com/jirasoftwarecloud/what-is-a-board-764477964.html.
18. Jiao, Y. *A Method of Calculating Comment Text Similarity Based on Tree Structure*. in *2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics*. 2015.
19. Rahimi, S.R., A.T. Mozhdehi, and M. Abdolahi. *An overview on extractive text summarization*. in *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*. 2017.
20. SHRIMALI, V. *Universal Sentence Encoder*. [cited 2019 Jun 15]; Available from: https://www.learnopencv.com/universal-sentence-encoder/.
21. Plus, M.L. *Cosine Similarity – Understanding the math and how it works*. [cited 2019 Jun 15]; Available from: https://www.machinelearningplus.com/nlp/cosine-similarity/.
22. Maalej, W. and H. Nabil. *Bug report, feature request, or simply praise? On automatically classifying app reviews*. in *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. 2015.
23. Li, X., Z. Zhang, and K. Stefanidis. *Mobile App Evolution Analysis Based on User Reviews*. in *SoMeT*. 2018.
24. Khalid, H., et al., *What Do Mobile App Users Complain About?* IEEE Software, 2015. **32**(3): p. 70-77.

25. Nayebi, M., et al. *App Store Mining Is Not Enough*. in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017.

26. Chen, N., et al., *AR-miner: mining informative reviews for developers from mobile app marketplace*, in *Proceedings of the 36th International Conference on Software Engineering*. 2014, ACM: Hyderabad, India. p. 767-778.

27. Karanwittayakarn, J. *Automatically create Jira Issue from Firebase Crashlytics*. [cited 2018 Dec 31]; Available from: https://medium.com/firebasethailand/%E0%B8%A1%E0%B8%B2%E0%B8%AA%E0%B8%A3%E0%B9%89%E0%B8%B2%E0%B8%87-jira-issues-%E0%B8%88%E0%B8%B2%E0%B8%81-crash-%E0%B8%97%E0%B8%B5%E0%B9%88%E0%B9%80%E0%B8%81%E0%B8%B4%E0%B8%94%E0%B8%82%E0%B8%B6%E0%B9%89%E0%B8%99%E0%B9%83%.

28. Palomba, F., et al. *Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews*. in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 2017.

29. Guzman, E., M. El-Haliby, and B. Bruegge. *Ensemble Methods for App Review Classification: An Approach for Software Evolution (N)*. in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2015.

ประวัติผู้เขียน

ชื่อ-สกุล	กิตติศักดิ์ เพชรรุ่งนภา
วัน เดือน ปี เกิด	7 พฤษภาคม 2533
สถานที่เกิด	สุโขทัย
ที่อยู่ปัจจุบัน	141/557 อาคารรีทีมสาทร ถนนสาทรใต้ 21 แขวงยานนาวา เขตสาทร กรุงเทพมหานคร 10120