การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากการจำแนกบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน

น.ส.ธนัชชา พันธ์ธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

Generation of Functional and Non-Functional Software Requirements Based on Classification of Mobile Application User Reviews

Miss Tanutcha Panthum

A Thesis 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 2021

หัวข้อวิทยานิพนธ์	การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากการจำแนกบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
โดย	น.ส.ธนัชชา พันธ์ธรรม
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

..... กรรมการ
(ดร.ดวงดาว วิชาดากุล)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.เบญจพร ลิ้มธรรมาภรณ์)

ธนัชชา พันธ์ธรรม : การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากการจำแนกบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน. (Generation of Functional and Non-Functional Software Requirements Based on Classification of Mobile Application User Reviews) อ.ที่ปรึกษาหลัก : รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

บทวิจารณ์ของผู้ใช้งานเป็นแหล่งข้อมูลที่สำคัญสำหรับนักพัฒนาโมไบล์แอปพลิเคชันเพื่อใช้ในการปรับปรุงและวิวัฒนาการแอปพลิเคชันหลังจากที่ได้ปล่อยให้ใช้งานไปแล้ว เนื่องจากข้อมูลบทวิจารณ์ของผู้ใช้งานมีจำนวนมากจึงเป็นเรื่องยุ่งยากสำหรับทีมนักพัฒนาโมไบล์แอปพลิเคชันที่จะระบุว่าบทวิจารณ์ของผู้ใช้งานใดประกอบไปด้วยข้อมูลที่เป็นประโยชน์ต่อการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชันเพิ่มเติม วิทยานิพนธ์นี้นำเสนอความพยายามที่จะอำนวยความสะดวกให้แก่ทีมนักพัฒนาในขั้นต้นด้วยการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันโดยอัตโนมัติจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันบนแอปสโตร์และเพลย์สโตร์

แนวทางที่นำเสนอประกอบด้วยสามขั้นตอน เริ่มจากการใช้อัลกอริทึมการจำแนกข้อความเพื่อจำแนกบทวิจารณ์ของผู้ใช้งานออกเป็นบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันหรือที่ไม่ใช่เชิงฟังก์ชัน ขั้นตอนที่สองบทวิจารณ์ของผู้ใช้งานที่ไม่ซ้ำกันจะถูกระบุโดยใช้เทคนิคการจัดกลุ่มและการวิเคราะห์ความคล้ายคลึงกันของข้อความ ในขั้นตอนสุดท้ายข้อมูลที่มีความสำคัญจะถูกสกัดจากบทวิจารณ์ของผู้ใช้งานเพื่อใช้สร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันโดยใช้แบบรูปข้อมูลบทวิจารณ์ของผู้ใช้งานและแม่แบบความต้องการ ในส่วนของการประเมินผล ความต้องการที่ถูกสร้างขึ้นจากแนวทางที่นำเสนอได้รับคะแนนต่ำถึงสูงแตกต่างกันไปในแง่ของความสามารถในการอ่านได้ง่าย ความไม่กำกวม ความสมบูรณ์ และความสมเหตุสมผล ซึ่งแนวทางที่วิทยานิพนธ์นำเสนอนี้สามารถช่วยทีมนักพัฒนาระบุถึงความต้องการการเปลี่ยนแปลงทั้งในเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากเสียงสะท้อนโดยตรงของผู้ใช้งานซึ่งควรได้รับการพิจารณาเพื่อใช้ในการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชันต่อไป

สาขาวิชา	วิศวกรรมซอฟต์แวร์	ลายมือชื่อนิสิต .....
ปีการศึกษา	2564	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6070450921 : MAJOR SOFTWARE ENGINEERING

KEYWORD: Software Requirement Generation, Functional Requirements, Non-functional Requirements, User Reviews, Text Classification, Text Similarity, Requirement Boilerplates, Machine Learning, Natural Language Processing

Tanutcha Panthum : Generation of Functional and Non-Functional Software Requirements Based on Classification of Mobile Application User Reviews. Advisor: Assoc. Prof. Dr. TWITTIE SENIVONGSE

User reviews are important resources for mobile application developers for maintaining and evolving mobile applications that have been released. Since there can be a lot of user reviews, it is cumbersome for the mobile development team to identify which ones contain useful information for further maintenance and evolution. This thesis proposes an initial attempt to facilitate a development team by automating the generation of functional and non-functional requirements from mobile application user reviews on the App Store and Play Store.

The proposed approach consists of three steps. Firstly, text classification algorithms are used to classify user reviews into functional or non-functional user reviews. Secondly, distinct user reviews are identified by clustering techniques and text similarity analysis. Finally, relevant information is extracted from the user reviews to generate requirements by using user review patterns and requirement boilerplates. In an evaluation, the generated requirements obtained varying scores from low to high in terms of readability, unambiguity, completeness, and validity. The approach can help the development team identify both functional and non-functional change requirements from direct feedback of the users which should be considered and further refined in the maintenance and evolution of the mobile application.

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

Academic Year: 2021 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีจากความช่วยเหลือของ รศ.ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ของข้าพเจ้า ข้าพเจ้าได้รับคำชี้แนะแนวทางการแก้ไขปัญหาในเรื่องต่าง ๆ รวมถึงได้รับกำลังใจที่ดี ความทุ่มเทและความใส่ใจจากอาจารย์ ข้าพเจ้าขอขอบพระคุณอาจารย์เป็นอย่างสูง

ขอขอบพระคุณ รศ.ดร. วิวัฒน์ วัฒนาวุฒิ ประธานกรรมการสอบวิทยานิพนธ์ ดร. ดวงดาว วิชาดากุล และ รศ.ดร. เบญจพร ลิ้มธรรมาภรณ์ กรรมการสอบวิทยานิพนธ์ ที่ได้เสียสละเวลาตรวจสอบและให้คำแนะนำแนวทางการแก้ไขวิทยานิพนธ์ฉบับนี้ให้สมบูรณ์มากขึ้น

ขอขอบคุณเพื่อน ๆ รุ่นพี่และรุ่นน้องทั้งในและนอกภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้คำปรึกษาเพื่อปรับปรุงวิทยานิพนธ์นี้และความช่วยเหลือต่าง ๆ ทั้งช่วยติดป้ายข้อมูล รวมถึงประเมินผลการทดลองอีกด้วย ขอบคุณสำหรับกำลังใจดี ๆ ที่มอบให้กันเสมอมา

ธนัชชา พันธ์ธรรม

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันเป็นแหล่งข้อมูลที่สำคัญสำหรับนักพัฒนาโมไบล์แอปพลิเคชัน ซึ่งนักพัฒนาสามารถนำข้อมูลเสียงสะท้อนจากการใช้งานของผู้ใช้งานมาประกอบการวิเคราะห์ พัฒนาและปรับปรุงแอปพลิเคชันให้ตรงตามความต้องการของผู้ใช้งานแอปพลิเคชันได้ โดยทั่วไปลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันจะอธิบายถึงปัญหาที่พบจากการใช้งาน การร้องขอการปรับปรุง หรือนำเสนอไอเดียการทำงานของแอปพลิเคชันที่ผู้ใช้งานต้องการ รวมถึงแสดงความรู้สึกที่มีต่อการใช้งาน เช่น รู้สึกชื่นชม รู้สึกไม่พึงพอใจ เป็นต้น บทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันมีลักษณะเป็นข้อความภาษาธรรมชาติ มีโครงสร้างการเขียนที่ไม่ตายตัว และบางครั้งประโยคบทวิจารณ์ของผู้ใช้งานมีโครงสร้างที่ไม่สมบูรณ์และขาดความหมาย เนื่องจากบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันมีจำนวนมาก จึงเป็นเรื่องยุ่งยากสำหรับทีมนักพัฒนาที่จะระบุว่าบทวิจารณ์ของผู้ใช้งานใดประกอบด้วยข้อมูลที่เป็นประโยชน์ต่อการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชันเพิ่มเติม

มีหลายงานวิจัยก่อนหน้านี้ศึกษาถึงแนวทางการรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานที่เป็นประโยชน์ต่อการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชันหลังจากที่ได้ปล่อยให้ใช้งานไปแล้ว เช่น ศึกษาแนวทางจำแนกประเภทบทวิจารณ์ของผู้ใช้งานออกเป็นประเภทต่าง ๆ [1], [2], [3] เช่น ประเภทรายงานการทำงานผิดพลาด (Bug Report) การร้องขอฟีเจอร์ (Feature Request) ความต้องการเชิงฟังก์ชัน (Functional Requirement) และความต้องการที่ไม่ใช่เชิงฟังก์ชัน (Non-functional Requirement) ศึกษาแนวทางการจัดกลุ่มบทวิจารณ์ของผู้ใช้งานที่เกี่ยวข้องกัน เป็นต้น [4], [5] ซึ่งเทคนิคส่วนใหญ่ที่ใช้ได้แก่ การประมวลผลภาษาธรรมชาติ (Natural Language Processing) การวิเคราะห์ข้อความ (Text Analysis) การวิเคราะห์ความรู้สึก (Sentiment Analysis) การจัดกลุ่ม (Clustering) และการสร้างแบบจำลองหัวข้อ (Topic Modeling)

วิทยานิพนธ์ได้รับแรงบันดาลใจมาจากงานวิจัยก่อนหน้านี้และมีจุดมุ่งหมายเพื่อเพิ่มประโยชน์ให้กับการวิเคราะห์ข้อมูลบทวิจารณ์ของผู้ใช้งานสำหรับใช้ในกระบวนการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชันด้วยการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์โดยอัตโนมัติจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันบนแอปสโตร์และเพลย์สโตร์ โดยอาศัยแนวทางการเรียนรู้ของเครื่อง (Machine Learning) [6] และ Natural Language Processing [7] ประกอบด้วยสามขั้นตอนหลัก เริ่มจากจำแนกประเภทบทวิจารณ์ของผู้ใช้งานออกเป็นบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันด้วยวิธีการจำแนกข้อความ

(Text Classification) ขั้นตอนที่สองคัดเลือกบทวิจารณ์ของผู้ใช้งานที่ไม่ซ้ำกันด้วยแนวทาง Clustering และการวิเคราะห์ความคล้ายคลึงกันของข้อความ (Text Similarity) ในขั้นตอนสุดท้าย ข้อมูลที่มีความสำคัญจะถูกสกัดจากบทวิจารณ์ของผู้ใช้งานเพื่อใช้สร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันโดยใช้แบบรูปข้อมูลบทวิจารณ์ของผู้ใช้งาน (User Review Patterns) [1] และแม่แบบความต้องการ (Requirement Boilerplates) [8] นอกจากนี้วิทยานิพนธ์ได้ประเมินประสิทธิภาพของแต่ละแนวทางในแต่ละขั้นตอนที่นำเสนอด้วย รวมถึงประเมินคุณภาพของความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากแนวทางที่ได้นำเสนอ โดยประเมินทั้งหมดสี่เกณฑ์คุณภาพ คือ ความสามารถในการอ่านได้ง่าย (Readability) ความไม่กำกวม (Unambiguity) ความสมบูรณ์ (Completeness) และความสมเหตุสมผล (Validity)

1.2 วัตถุประสงค์

1) พัฒนาวิธีการจำแนกประเภทความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
2) พัฒนาวิธีการตรวจสอบความซ้ำซ้อนกันของข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
3) พัฒนาวิธีการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน

1.3 ขอบเขตการวิจัย

1) สร้างโมเดลที่สามารถจำแนกประเภทความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันทั้งหมด 3 ประเภท คือ Functional, Availability, Operability มีความสามารถดังนี้
    a) หนึ่งประโยคบทวิจารณ์ของผู้ใช้งาน สามารถจำแนกประเภทได้เพียงประเภทเดียวเท่านั้น
2) คุณสมบัติของข้อมูลบทวิจารณ์ของผู้ใช้งาน
    a) ข้อมูลบทวิจารณ์ของผู้ใช้งานจะถูกประมวลผลในระดับประโยคเท่านั้น
    b) รองรับข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันในรูปแบบภาษาอังกฤษเท่านั้น
    c) ข้อมูลบทวิจารณ์ของผู้ใช้งานเกี่ยวข้องกับรายงานการทำงานผิดพลาดและการร้องขอการปรับปรุงเท่านั้น
    d) ประโยคบทวิจารณ์ของผู้ใช้งานต้องมีความยาวเกิน 3 คำ
3) สร้างวิธีตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน

4) สร้างความต้องการซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งาน ซึ่งมีความสามารถดังนี้
    a) สร้างความต้องการซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งานเพียงประโยคเดียวเท่านั้น ไม่นำประโยคข้างเคียง บริบทอื่น ๆ มาประมวลผลด้วย
    b) สร้างประโยคความต้องการซอฟต์แวร์โดยใช้แม่แบบความต้องการ
5) เครื่องมือที่ใช้พัฒนา
    a) การดึงข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันใช้ไลบรารี app-store-scraper และไลบรารี google-play-scraper
    b) การประมวลผลภาษาธรรมชาติต่าง ๆ ใช้ไลบรารี NLTK ไลบรารี PYENCHANT และไลบรารี Stanford
    c) การสร้างโมเดลการเรียนรู้ของเครื่องใช้ไลบรารี Scikit-learn
6) พัฒนาด้วยภาษาไพธอน

1.4 ขั้นตอนการดำเนินงาน

1) ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับลักษณะข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
2) ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการจำแนกประเภทความต้องการซอฟต์แวร์ทั้งจากข้อมูลบทวิจารณ์ของผู้ใช้งานและข้อมูลลักษณะอื่น ๆ
3) ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการตรวจสอบความซ้ำซ้อนกันของข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
4) ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการสร้างประโยคความต้องการจากบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันหรือข้อมูลภาษาธรรมชาติในลักษณะอื่น ๆ
5) ศึกษาภาษาไพธอนและไลบรารีที่เกี่ยวข้องกับการพัฒนาโมเดลการเรียนรู้ของเครื่องการประมวลผลภาษาธรรมชาติ และการรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
6) รวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์ เพลย์สโตร์ และจากงานวิจัยอื่น ๆ
7) พัฒนาและทดสอบประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน
8) พัฒนาและทดสอบประสิทธิภาพของการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งาน
9) พัฒนาการสร้างความต้องการซอฟต์แวร์และประเมินคุณภาพความต้องการซอฟต์แวร์
10) สรุปผลการวิจัยและข้อเสนอแนะ
11) จัดทำบทความทางวิชาการ

12) จัดทำเล่มวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้แบบจำลองสำหรับการจำแนกประเภทความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
2) ได้วิธีการตรวจสอบความซ้ำซ้อนของข้อมูลบทวิจารณ์ของผู้ใช้งาน
3) ได้วิธีการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน
4) ลดระยะเวลาและทรัพยากรที่ใช้สำหรับการวิเคราะห์ความต้องการของผู้ใช้งานจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน

1.6 บทความวิจัยที่ได้รับการตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์ได้รับการตีพิมพ์เป็นบทความวิชาการเรื่อง “Generating Functional Requirements Based on Classification of Mobile Application User Reviews” ในงานประชุมวิชาการ 2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA) ระหว่างวันที่ 20-22 มิถุนายน พ.ศ. 2564 ณ เมืองคานาซาว่า ประเทศญี่ปุ่น

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 วิศวกรรมความต้องการซอฟต์แวร์

วิศวกรรมความต้องการ [9] คือศาสตร์ที่ศึกษาเกี่ยวกับการสื่อสารระหว่างผู้ซื้อและผู้ผลิตหรือนักพัฒนา เพื่อสร้างและบำรุงรักษาความต้องการให้ตรงกับความต้องการของระบบ ของซอฟต์แวร์หรือบริการที่ให้ความสนใจ วิศวกรรมความต้องการประกอบไปด้วยการค้นหาความต้องการ (Discovering Requirement) การสกัดความต้องการ (Eliciting Requirement) การพัฒนาความต้องการ (Developing Requirement) การวิเคราะห์ความต้องการ (Analyzing Requirement) การตรวจสอบความต้องการ (Validating Requirement) การทวนสอบความต้องการ (Verification Requirement) การสื่อสาร (Communicating) การทำเอกสาร (Documenting) และการจัดการความต้องการ (Managing Requirement) กระบวนการวิศวกรรมความต้องการเป็นขั้นตอนแรกสุดของวัฏจักรชีวิตการพัฒนาซอฟต์แวร์ (Software Development Life Cycle Model)

ความต้องการ (Requirements) [10] คือ (1) เงื่อนไขหรือความสามารถที่ผู้ใช้งานต้องการสำหรับแก้ปัญหาหรือเพื่อบรรลุวัตถุประสงค์ที่ต้องการ (2) เงื่อนไขหรือความสามารถที่ตรงตามหรือเป็นของระบบหรือส่วนของระบบตามสัญญา มาตรฐาน ข้อกำหนด หรือเอกสารข้อกำหนดอย่างเป็นทางการอื่น ๆ (3) เอกสารที่แสดงถึงเงื่อนไขหรือความสามารถใน (1) หรือ (2) โดยความต้องการควรจะกล่าวถึง ระบบควรจะเป็นอย่างไร ภาพรวมของระบบ การทำงาน ข้อบังคับการทำงาน การพัฒนาของระบบ เป็นต้น ตัวอย่างของประเภทความต้องการ [11] เช่น

1) Functional Requirement อธิบายถึงฟังก์ชัน ความสามารถ ที่ระบบหรือซอฟต์แวร์สามารถทำได้
2) Non-functional Requirement อธิบายถึงข้อบังคับหรือความต้องการด้านคุณภาพ (Quality Requirements) ของซอฟต์แวร์ในด้านต่าง ๆ เช่น ความต้องการด้านประสิทธิภาพ (Performance Requirements) ความต้องการด้านส่วนต่อประสาน (Interface Requirements) ข้อบังคับด้านการออกแบบ (Design Constraint) ความต้องการด้านความมั่นคง (Security Requirements)

การเขียนความต้องการควรสอดคล้องกับแนวทางโครงสร้างของความต้องการที่มีรูปแบบที่ดี (Well-Formed Requirement) [9] กล่าวคือความต้องการต้องสามารถทวนสอบได้ เป็นความต้องการของระบบเพื่อแก้ไขปัญหาหรือบรรลุวัตถุประสงค์ของผู้มีส่วนได้ส่วนเสีย เงื่อนไขต้องสามารถวัดได้ ทราบถึงขอบเขตข้อบังคับและกล่าวถึงความสามารถของระบบ

แนวทางการเขียนความต้องการที่มีรูปแบบที่ดีในภาษาธรรมชาติประกอบด้วย ข้อความควรจะกล่าวถึงประธาน (Subject) สิ่งที่กระทำ (Action) พร้อมกับองค์ประกอบอื่น ๆ (Complement) ที่จำเป็นและเพียงพอต่อความต้องการ ตัวอย่างโครงสร้างความต้องการ (Requirements Syntax) ดังรูปที่ 2-1 รูปแบบแรกประกอบด้วย [Condition] [Subject] [Action] [Object] [Constraint of Action] คือเมื่อใดที่เกิดเงื่อนไขตามที่ระบุ ระบบจะกระทำตามวัตถุประสงค์ด้วยข้อบังคับของการกระทำที่กำหนดไว้

ลักษณะของคำศัพท์ที่ใช้ในความต้องการมีแนวทางดังนี้

1) ความต้องการที่จำเป็นควรใช้คำกริยา ‘shall’
2) ความต้องการที่ไม่จำเป็น ควรหลีกเลี่ยง เช่น ความต้องการที่จะทำในอนาคตควรใช้คำกริยา ‘will’, ‘should’, ‘may’
3) Non-functional Requirements ควรใช้คำกริยา ‘are’, ‘is’, ‘was’ ควรหลีกเลี่ยงการใช้ ‘must’
4) ควรหลีกเลี่ยงความต้องการเชิงลบ (Negative Requirements) เช่น ‘shall not’
5) ประโยคความต้องการควรกล่าวถึงประธานคือผู้กระทำ (Active Voice) มากกว่าประธานคือผู้ถูกกระทำ (Passive Voice)

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: When signal x is received **[Condition]**, the system **[Subject]** shall set **[Action]** the signal x received bit **[Object]** within 2 seconds **[Constraint of Action].**

Or

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: At sea state 1 **[Condition]**, the Radar System **[Subject]** shall detect **[Action]** targets **[Object]** at ranges out to 100 nautical miles **[Constraint of Action].**

Or

[Subject] [Action] [Constraint of Action]

EXAMPLE: The Invoice System **[Subject]** shall display pending customer invoices **[Action]** in ascending order of invoice due date **[Constraint of Action].**

รูปที่ 2-1 โครงสร้างการเขียนความต้องการเชิงฟังก์ชันตามแนวทางการเขียนความต้องการที่มีรูปแบบที่ดี [9]

2.1.2 มาตรฐานคุณภาพของซอฟต์แวร์ [12]

มาตรฐานคุณภาพของซอฟต์แวร์อธิบายถึงลักษณะของคุณภาพของซอฟต์แวร์ ในมาตรฐานนี้แบ่งออกเป็นสองมุมมอง คือ ด้านคุณภาพของผลิตภัณฑ์ (Product Quality) และด้านคุณภาพในการใช้งาน (Quality In Use Model) สามารถนำมาตรฐานไปใช้ระบุความต้องการด้านคุณภาพของซอฟต์แวร์ สร้างตัววัดคุณภาพของซอฟต์แวร์ได้

มาตรฐานด้านคุณภาพของผลิตภัณฑ์ (Product Quality Model) เน้นไปที่ลักษณะของคุณภาพระบบคอมพิวเตอร์ที่ประกอบไปด้วยผลิตภัณฑ์ซอฟต์แวร์ (Software Product) แบ่งลักษณะคุณภาพของผลิตภัณฑ์ได้ 8 หมวดหมู่ดังต่อไปนี้

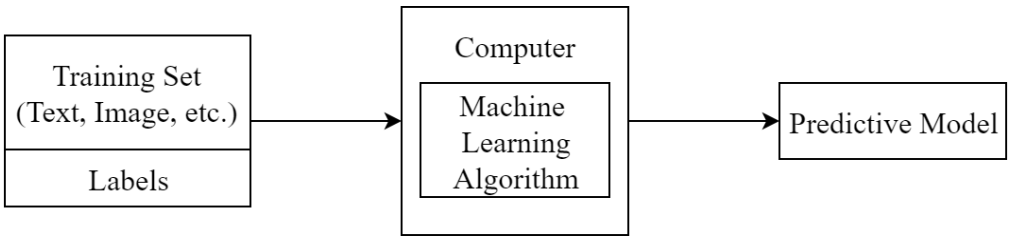
1) Functional Suitability อธิบายถึงระดับของฟังก์ชันการทำงานของผลิตภัณฑ์หรือระบบตรงตามความต้องการภายใต้เงื่อนไขที่ระบุไว้
2) Performance Efficiency อธิบายถึงความสัมพันธ์ของประสิทธิภาพกับจำนวนทรัพยากรที่ใช้ภายใต้เงื่อนไขที่ระบุไว้ของระบบหรือซอฟต์แวร์
3) Compatibility อธิบายถึงระดับที่ผลิตภัณฑ์หรือระบบสามารถแลกเปลี่ยนข้อมูลกับผลิตภัณฑ์หรือระบบในสภาพแวดล้อมเดียวกันหรือสภาพแวดล้อมอื่น ๆ ได้
4) Usability อธิบายถึงระดับผลิตภัณฑ์หรือระบบที่ผู้ใช้งานสามารถใช้งานและบรรลุวัตถุประสงค์การใช้งานด้วยความพึงพอใจในประสิทธิภาพและประสิทธิผล
5) Reliability อธิบายถึงระดับความสามารถในการทำงานของฟังก์ชันของระบบและซอฟต์แวร์ตามระยะเวลาที่กำหนด
6) Security อธิบายถึงระดับการเข้าถึงข้อมูลตามสิทธิการเข้าถึงข้อมูล รวมถึงการปกป้องข้อมูลจากบุคคล ผลิตภัณฑ์หรือระบบอื่น ๆ
7) Maintainability อธิบายถึงระดับของประสิทธิภาพและประสิทธิผลในการปรับปรุงผลิตภัณฑ์หรือระบบในมุมของนักพัฒนา
8) Portability อธิบายถึงระดับประสิทธิภาพและประสิทธิผลของระบบหรือผลิตภัณฑ์เมื่อมีการเปลี่ยนแปลงสภาพแวดล้อมการทำงาน เช่น ย้ายการประมวลผลระบบเดิมไปยังอุปกรณ์ฮาร์ดแวร์ใหม่ สภาพแวดล้อมการทำงานใหม่

2.1.3 การเรียนรู้แบบมีผู้สอน (Supervised Learning) [13]

การเรียนรู้แบบมีผู้สอนเป็นหนึ่งในเทคนิคการเรียนรู้ของเครื่อง [6] ซึ่งการเรียนรู้ของเครื่องคือการพยายามให้โปรแกรมคอมพิวเตอร์มีความสามารถในการเรียนรู้จากข้อมูลด้วยตัวเองเพื่อใช้สำหรับแก้ปัญหาต่าง ๆ โดยปราศจากการเขียนโปรแกรมที่กำหนดตรรกะอย่างชัดเจนเพื่อแก้ปัญหา ซึ่งคือข้อแตกต่างระหว่างการเรียนรู้ของเครื่องกับการเขียนโปรแกรมแก้ไขปัญหาทั่ว ๆ ไป การเรียนรู้ของเครื่องจึงมีแรงบันดาลใจมาจากกระบวนการเรียนรู้ซ้ำ ๆ ของคน

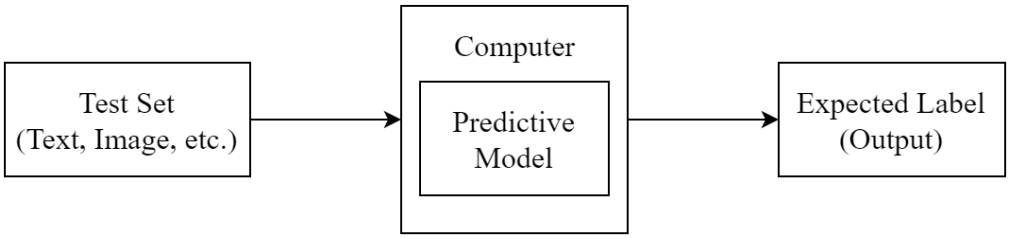
หลักการของการเรียนรู้แบบมีผู้สอน ดังรูปที่ 2-2 คือการสอนให้คอมพิวเตอร์มีความสามารถในการเรียนรู้จากข้อมูลเพื่อแก้ปัญหา โดยจะสร้างโมเดลการเรียนรู้จากข้อมูลชุดฝึกสอน (Training Set) ที่มีการระบุคำตอบ (Label) ของผลลัพธ์ไว้แล้ว ภายในข้อมูลชุดฝึกสอนจะมีคุณลักษณะของข้อมูลต่าง ๆ สำหรับให้คอมพิวเตอร์เรียนรู้ลักษณะของข้อมูลเพื่อใช้แก้ปัญหา

ผลลัพธ์ที่ได้หลังจากการฝึกสอนคือโมเดลการเรียนรู้ของเครื่องที่ใช้แก้ปัญหานั้น ๆ (Predictive Model) ตัวอย่างเช่น การสอนให้คอมพิวเตอร์สามารถจำแนกภาพกล้วยว่าเป็นภาพกล้วยหรือไม่ โดยให้คอมพิวเตอร์รู้จักภาพกล้วยลักษณะต่าง ๆ จากข้อมูลชุดฝึกสอนที่ได้ระบุคำตอบของภาพแล้วว่าเป็นกล้วยหรือไม่ใช่กล้วย และเลือกใช้อัลกอริทึมการเรียนรู้ของเครื่องที่เหมาะสมกับลักษณะของปัญหา ผลลัพธ์ที่ได้คือโมเดลการเรียนรู้ของเครื่องที่สามารถจำแนกภาพกล้วยหรือไม่ใช่ภาพกล้วยได้



รูปที่ 2-2 กระบวนการสร้างโมเดลการเรียนรู้ของเครื่องจากข้อมูลชุดฝึกสอน

เพื่อทดสอบประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องที่ได้ ดังรูปที่ 2-3 จะนำข้อมูลชุดทดสอบ (Test Set) ที่ประกอบด้วยข้อมูลต่าง ๆ เช่น ข้อมูลภาพกล้วย ไม่ใช่ภาพกล้วย และไม่มีการระบุคำตอบของข้อมูลชุดทดสอบที่นำมาทดสอบกับโมเดลการเรียนรู้ของเครื่อง เพื่อดูความถูกต้องความแม่นยำของโมเดลการเรียนรู้ของเครื่อง ผลลัพธ์ที่ได้จากโมเดลคือผลการทำนายภาพว่าเป็นภาพกล้วยหรือไม่ใช่ภาพกล้วยตามที่คาดหวัง (Expected Label)



รูปที่ 2-3 กระบวนการทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่องด้วยชุดข้อมูลทดสอบ

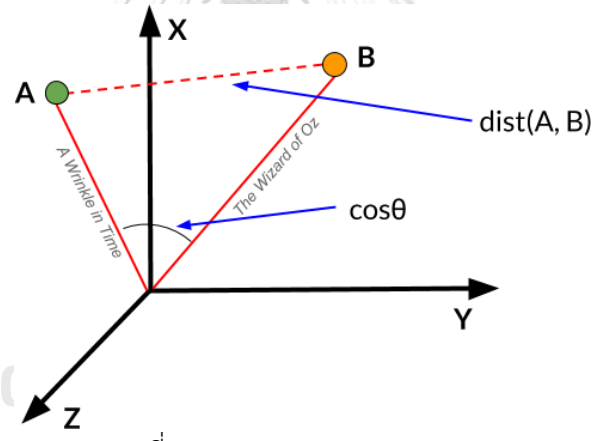
2.1.4 การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning) [14]

การเรียนรู้แบบไม่มีผู้สอนเป็นอีกหนึ่งในเทคนิคการเรียนรู้ของเครื่อง จะไม่ใช้ข้อมูลชุดฝึกสอนที่มีการระบุคำตอบของผลลัพธ์มาใช้ฝึกสอนคอมพิวเตอร์สำหรับแก้ปัญหาต่าง ๆ เหมือนเทคนิคการเรียนรู้แบบมีผู้สอน แต่ให้คอมพิวเตอร์เรียนรู้หาความสัมพันธ์ของข้อมูลเอง งานที่ใช้การ

เรียนรู้แบบไม่มีผู้สอน เช่น Clustering โดยแบ่งกลุ่มจากความสัมพันธ์ของข้อมูลที่คล้ายคลึงกัน ในการแบ่งกลุ่มของข้อมูลประกอบไปด้วยสองส่วน คือ ตัววัดความคล้ายคลึงกันของข้อมูลและอัลกอริทึมของการจัดกลุ่มข้อมูล

ตัวอย่างตัววัดความคล้ายคลึงกันของข้อมูล เช่น

1) Jaccard Similarity Coefficient หรืออีกชื่อที่รู้จักกันอย่างแพร่หลายคือ Jaccard Index [15] เป็นตัววัดความคล้ายคลึงกันที่ได้รับความนิยมอย่างมาก โดยคำนวณความคล้ายคลึงกันจากการหารจำนวนข้อมูลที่ซ้ำกันระหว่างสองเอกสารด้วยจำนวนข้อมูลทั้งหมดของทั้งสองเอกสารที่ไม่ซ้ำกัน
2) Cosine Similarity [15] คำนวณความคล้ายคลึงกันจากระยะห่างของเวกเตอร์ ด้วยการวัดมุมโคไซน์ (Cosine of Angle) ระหว่างทั้งสองเวกเตอร์ ถ้าเวกเตอร์ทำมุมโคไซน์ขนานกันมากแสดงว่าข้อมูลระหว่างเอกสารนั้นคล้ายคลึงกันมาก ดังรูปที่ 2-4 หาความคล้ายคลึงกันระหว่างเวกเตอร์ A และ B ด้วยการวัดมุมโคไซน์ ($\cos\theta$) ระหว่างเวกเตอร์ A และ B



รูปที่ 2-4 Cosine Similarity [14]

ซึ่งค่าที่ได้จากการคำนวณความคล้ายคลึงกันของตัววัด Jaccard Similarity Coefficient และ Cosine Similarity มีค่าอยู่ระหว่าง 0 กับ 1 โดยค่า 0 คือคู่ของข้อมูลที่ไม่คล้ายคลึงกัน และค่า 1 คือคู่ของข้อมูลที่คล้ายคลึงกัน ส่วนค่าที่อยู่ระหว่าง 0 กับ 1 คือระดับความคล้ายคลึงกันของข้อมูล

3) Cosine Distance [14] คำนวณระยะห่างระหว่างข้อมูล ดังรูปที่ 2-4 คือการคำนวณหาค่า dist(A, B) โดยหาได้จาก 1- ค่า Cosine Similarity ค่า Cosine Distance จะสวนทางกับค่า Cosine Similarity เมื่อข้อมูลมีความคล้ายคลึงกันมาก

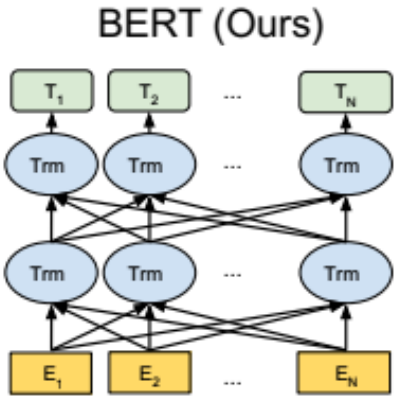
ระยะห่างระหว่างข้อมูลจะน้อย และหากข้อมูลไม่คล้ายคลึงกันจะมีระยะห่างระหว่างข้อมูลมาก นิยมใช้กับการทำ Clustering

ตัวอย่างอัลกอริทึมการแบ่งกลุ่ม เช่น

1) Partitional Clustering [14] เป็นอัลกอริทึมการจัดกลุ่มที่มีค่าเฉลี่ยของข้อมูลทั้งกลุ่ม (Centroid) เป็นตัววัดความคล้ายคลึงกันกับข้อมูลทั้งกลุ่ม อัลกอริทึมในกลุ่มนี้ เช่น K-means, Bisecting K-means, K-medoids เป็นต้น
2) Hierarchical Clustering [14] เป็นอัลกอริทึมการจัดกลุ่มจากการรวมข้อมูลที่คล้ายคลึงกันมากที่สุดไปเรื่อย ๆ จนครบจำนวนของข้อมูล อัลกอริทึมในกลุ่มนี้แบ่งออกได้เป็นสองแบบหลักๆ คือ Agglomerative และ Divisive
3) Density-Based Clustering [15] เป็นอัลกอริทึมการจัดกลุ่มข้อมูลเชิงพื้นที่โดยจะจัดกลุ่มข้อมูลจากความหนาแน่นของข้อมูลในบริเวณพื้นที่นั้น ๆ เหมาะกับข้อมูลที่มีรูปร่างไม่แน่นอน (Arbitrary Shape) เช่น ข้อมูลลักษณะรูปตัว S ลักษณะรูปวงรี ตัวอย่างของอัลกอริทึมในกลุ่มนี้ เช่น DBSCAN, OPTICS, DENCLUE

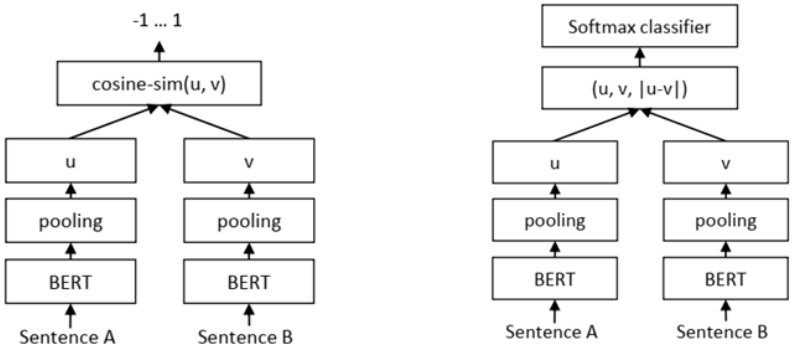
2.1.5 Sentence-BERT (SBERT) [16]

SBERT คือแนวทางการฝังประโยค (Sentence Embeddings) พัฒนามาจากแนวทาง BERT ซึ่งเป็นแนวทางการฝังประโยคที่คำนึงถึงบริบทของคำ มีประสิทธิภาพในการเก็บรวบรวมความหมายของคำ [17] โดยโครงสร้างสถาปัตยกรรมของ BERT เป็นแบบ Multi-Layer Bidirectional Transformer ดังรูปที่ 2-5 BERT เหมาะสำหรับงาน Question and Answering, Sentence Classification, Sentence-Pair Regression และประโยคที่ถูกฝังรหัสสามารถคำนวณค่าความคล้ายคลึงกันระหว่างข้อมูลด้วยตัววัด Cosine Similarity ได้ ตัวอย่างเช่น [18] ประโยค “play the record” มีค่า Cosine Similarity ที่ 0.754 และ 0.582 เมื่อวัดกับประโยค “play the game” และ “record the play” ตามลำดับ ซึ่งมีประสิทธิภาพดีกว่าเมื่อเทียบกับแนวทางการทำเวกเตอร์คุณลักษณะของข้อความด้วย TF-IDF หรือ Word2Vec เนื่องจากสถาปัตยกรรมโครงสร้างของ BERT ทำให้ใช้ทรัพยากรและเวลาในการคำนวณจำนวนมาก ไม่เหมาะกับงานที่มีข้อมูลขนาดใหญ่



รูปที่ 2-5 โครงสร้างสถาปัตยกรรมของ BERT [17]

SBERT ได้ปรับปรุงข้อจำกัดของ BERT โดยใช้สถาปัตยกรรมโครงสร้างเครือข่าย Siamese และ Triplet แทน ดังรูปที่ 2-6 ช่วยลดทรัพยากรและเวลาในการคำนวณและยังคงความแม่นยำของ BERT ไว้ นอกจากนี้ยังสามารถปรับใช้กับงานด้านใหม่ ๆ ได้ เช่น Clustering การสืบค้นข้อมูล (Information Retrieval) การเปรียบเทียบความคล้ายคลึงกันทางความหมายในข้อมูลขนาดใหญ่ เป็นต้น ตัวอย่างโมเดล Pretrained SBERT [19] ที่สามารถนำมาใช้ได้ในปัจจุบัน เช่น โมเดล paraphrase-MiniLM-L3-v2 โมเดล all-mpnet-base-v2 ซึ่งขึ้นอยู่กับความเหมาะสมในการเลือกใช้งาน



รูปที่ 2-6 โครงสร้างสถาปัตยกรรมของ SBERT เพื่อคำนวณ Similarity Scores (ฝั่งซ้าย), โครงสร้างสถาปัตยกรรมของ SBERT สำหรับงานจำแนกประเภทของข้อมูล (ฝั่งขวา) [16]

สามารถใช้งานโมเดล Pretrained SBERT ผ่าน Sentence Transformers เฟรมเวอร์ค [20] สำหรับภาษาไพธอน (Python) รองรับการฝังประโยคมากกว่า 100 ภาษา วิธีการติดตั้งและเรียกใช้งานโมเดล Pretrained SBERT ดังรูปที่ 2-7

Installation

You can install it using pip:

```
pip install -U sentence-transformers
```

We recommand **Python 3.6** or higher, and at least **PyTorch 1.6.0**. See installation for further installation options, especially if you want to use a GPU.

Usage

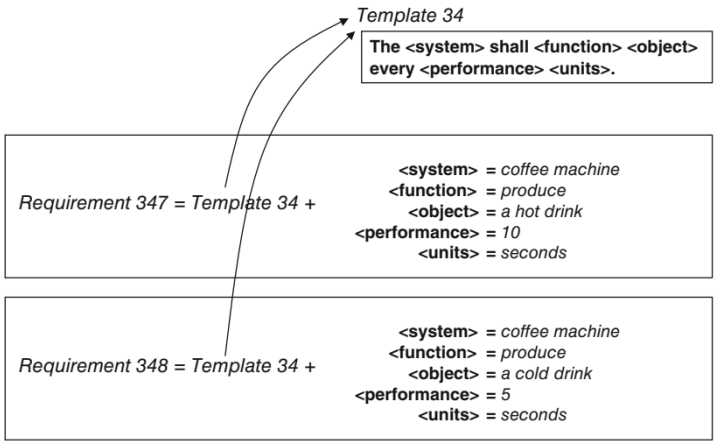
The usage is as simple as:

```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

#Our sentences we like to encode
sentences = ['This framework generates embeddings for each input sentence',
    'Sentences are passed as a list of string.',
    'The quick brown fox jumps over the lazy dog.']

#Sentences are encoded by calling model.encode()
embeddings = model.encode(sentences)

#Print the embeddings
for sentence, embedding in zip(sentences, embeddings):
    print("Sentence:", sentence)
    print("Embedding:", embedding)
    print("")
```

รูปที่ 2-7 วิธีการติดตั้งและใช้งานโมเดล Pretrained SBERT ผ่าน Sentence Transformers เฟรมเวอร์คสำหรับภาษา Python [20]

2.1.6 Requirement Boilerplates [8]

แม่แบบความต้องการ (Requirement Boilerplates) คือแนวทางการเขียนความต้องการที่มีการกำหนดโครงสร้างของประโยคในรูปของข้อความภาษาธรรมชาติไว้แล้ว เป็นวิธีการสร้างมาตรฐานทางภาษาสำหรับประโยคความต้องการ ตัวอย่าง Requirement Boilerplate ดังรูปที่ 2-8 คือแม่แบบความต้องการทั่วไป ซึ่งวิธีการสร้างประโยคความต้องการจากแม่แบบความต้องการนั้น ในแม่แบบจะมีช่องว่างที่กำหนดคุณลักษณะของข้อมูล (Placeholders) ที่จะนำมาแทนที่ในแม่แบบความต้องการ ตัวอย่างเช่น จากรูปจะแทนที่คุณลักษณะของข้อมูล <system> ด้วย “coffee machine” ซึ่งเป็นชื่อของระบบ เป็นต้น สุดท้ายจะได้ประโยคความต้องการจากแม่แบบความต้องการทั่วไปคือ “The coffee machine shall produce a hot drink every 10 seconds.”



รูปที่ 2-8 แม่แบบความต้องการทั่วไป [8]

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องแบ่งได้ทั้งหมด 3 หมวดหมู่ได้แก่ การจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นประเภทต่าง ๆ โดยใช้วิธีการเรียนรู้ของเครื่อง การจัดกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งานที่เกี่ยวข้องกัน การสร้างเอกสารความต้องการของซอฟต์แวร์

หมวดหมู่การจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นประเภทต่าง ๆ โดยใช้วิธีการเรียนรู้ของเครื่อง

2.2.1 How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution [1]

งานวิจัยนี้มีเป้าหมายเพื่อช่วยนักพัฒนาจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์และเพลย์สโตร์ที่เกี่ยวข้องกับการปรับปรุงและวิวัฒนาการซอฟต์แวร์ ประเภทของข้อมูลที่จำแนกได้แก่ รายงานปัญหาที่พบ การร้องขอฟีเจอร์ การแจ้งข้อมูล การค้นหาข้อมูล โดยใช้วิธีการเรียนรู้ของเครื่อง

งานวิจัยนี้ได้ตั้งสมมติฐานว่า รูปแบบการเขียนบทวิจารณ์ของผู้ใช้งานหมวดหมู่เดียวกันมีแนวโน้มที่จะมีรูปแบบการเขียนที่คล้าย ๆ หรือซ้ำกัน (Recurrent Linguistic Patterns) ซึ่งสามารถนำรูปแบบการเขียนมาจำแนกประเภทของข้อมูลได้ โดยอาศัยลักษณะของคำและความสัมพันธ์ของคำตามโครงสร้างของประโยค ตัวอย่างเช่น บทวิจารณ์ของผู้ใช้งานที่มีเจตนาสื่อถึงการร้องขอฟีเจอร์ “You should add a new button” โครงสร้างของประโยคประกอบด้วย “add” ที่เป็นกริยาหลัก “you” ที่เป็นประธาน “button” ที่เป็นกรรม “new” ที่เป็นคุณสมบัติของกรรม “should” เป็นกริยาช่วย ซึ่งสามารถนำโครงสร้างและความสัมพันธ์ที่ปรากฏไปใช้วิเคราะห์ประโยคที่เป็นประเภทเดียวกับการร้องขอฟีเจอร์ได้ ในงานวิจัยนี้ได้วิเคราะห์โครงสร้างของประโยคบทวิจารณ์ของผู้ใช้งาน

จากข้อมูลบทวิจารณ์ของผู้ใช้งานทั้งหมด 500 บทวิจารณ์ของผู้ใช้งาน พบ Recurrent Linguistic Patterns ทั้งหมด 246 รูปแบบโครงสร้างของประโยค เรียกรูปแบบโครงสร้างของประโยคนี้ว่า NLP Heuristic ตัวอย่างของ NLP Heuristic ประเภทการร้องขอฟีเจอร์ เช่น “[someone] should add [something]”

วิทยานิพนธ์นำข้อมูล NLP Heuristic มาดัดแปลงเป็น User Review Patterns เพื่อใช้สกัดข้อมูลที่สำคัญจากประโยคบทวิจารณ์ของผู้ใช้งาน

2.2.2 Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews [2]

งานวิจัยนี้นำเสนอแนวทางการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์และเพลย์สโตร์ ประเภทของข้อมูลที่จำแนกเกี่ยวข้องกับวิศวกรรมความต้องการซอฟต์แวร์ ซึ่งนักพัฒนาสามารถนำมาใช้พัฒนาและปรับปรุงแอปพลิเคชันได้ แบ่งเป็น 4 ประเภทได้แก่ รายงานการทำงานผิดพลาด การร้องขอฟีเจอร์ ประสบการณ์การใช้งานแอปพลิเคชัน (User Experiences) และความพึงพอใจในการใช้งาน (Ratings) โดยใช้เทคนิคการเรียนรู้ของเครื่อง

งานวิจัยนี้ได้รวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์และเพลย์สโตร์มาติดฉลากข้อมูลโดยคนสองคนผ่านเครื่องมือติดฉลากและคู่มือ เพื่อลดข้อผิดพลาดที่อาจเกิดขึ้นจากการติดฉลาก ผลลัพธ์ที่ได้พบว่าแต่ละประเภทของข้อมูลบทวิจารณ์ของผู้ใช้งานมีเทคนิคและประสิทธิภาพที่ได้แตกต่างกัน ค่า Precision อยู่ที่ประมาณ 70-95 % ขณะที่ Recall อยู่ที่ประมาณ 80-90 % โดยใช้อัลกอริทึมการเรียนรู้ของเครื่อง Naïve Bayes

วิทยานิพนธ์นำข้อมูลบทวิจารณ์ของผู้ใช้งานประเภทรายงานการทำงานผิดพลาดและการร้องขอฟีเจอร์ของงานวิจัยนี้มาร่วมใช้ในการทดลอง โดยใช้เป็นข้อมูลชุดฝึกสอนและชุดทดสอบโมเดลการเรียนรู้ของเครื่อง

2.2.3 Automatic Classification of Non-Functional Requirements from Augmented App User Reviews [3]

งานวิจัยนี้นำเสนอแนวทางการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์และเพลย์สโตร์โดยวิธีการเรียนรู้ของเครื่อง งานวิจัยนี้จำแนกประเภทบทวิจารณ์ของผู้ใช้งานออกเป็นประเภท 1) ความต้องการที่ไม่ใช่เชิงฟังก์ชันได้แก่ Usability, Reliability, Portability, Performance 2) ความต้องการเชิงฟังก์ชัน และ 3) ข้อมูลอื่น ๆ (Others)

งานวิจัยนี้รวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานมาจากแอปพลิเคชัน iBooks จากแอปสโตร์และแอปพลิเคชัน WhatsApp จากเพลย์สโตร์ สุ่มเลือกข้อมูลบทวิจารณ์ของผู้ใช้งานมาทั้งหมด

2,000 ประโยคจากแต่ละแอปพลิเคชัน รวม 4,000 ประโยค เพื่อนำมาติดฉลากประเภทของบทวิจารณ์ของผู้ใช้งานว่าเป็นประเภทใดตามมาตรฐานคุณภาพของซอฟต์แวร์ [12] และมีการตกลงกันระหว่างผู้ติดฉลาก เมื่อผลลัพธ์ของการติดฉลากประเภทบทวิจารณ์ของผู้ใช้งานไม่ตรงกัน

ผลลัพธ์การจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานของงานวิจัยนี้ มีประสิทธิภาพดีที่สุด คือ ค่า Precision อยู่ที่ 71.4% ขณะที่ค่าเฉลี่ย Recall อยู่ที่ 72.3% และค่าเฉลี่ย F-Measure อยู่ที่ 71.8%

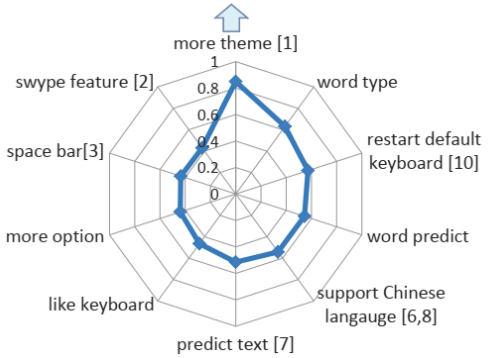
วิทยานิพนธ์นำข้อมูลบทวิจารณ์ของผู้ใช้งานของงานวิจัยนี้มาร่วมใช้ในการทดลอง โดยใช้เป็นข้อมูลชุดฝึกสอนและชุดทดสอบโมเดลการเรียนรู้ของเครื่อง

หมวดหมู่การจัดกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งานที่เกี่ยวข้องกัน

2.2.4 AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace [4]

งานวิจัยนี้นำเสนอเครื่องมือซอฟต์แวร์ที่ชื่อว่า AR-Miner เพื่อช่วยทีมนักพัฒนาโมไบล์แอปพลิเคชันวิเคราะห์ข้อมูลที่เป็นประโยชน์ต่อการพัฒนาและปรับปรุงแอปพลิเคชันเพิ่มเติมจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชัน มีขั้นตอนการทำงานสี่ขั้นตอนหลักคือ เริ่มจากการจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นบทวิจารณ์ของผู้ใช้งานที่ให้ข้อมูล (Informative) และไม่ให้ข้อมูล (Uninformative) ขั้นตอนถัดมาคือการจัดกลุ่มบทวิจารณ์ของผู้ใช้งานที่ให้ข้อมูลด้วยอัลกอริทึม Topic Modeling ขั้นตอนที่สามคือการจัดลำดับความสำคัญของกลุ่มบทวิจารณ์ของผู้ใช้งานที่ให้ข้อมูล และขั้นตอนสุดท้ายคือการแสดงผลกลุ่มของบทวิจารณ์ของผู้ใช้งานที่มีความสำคัญ ดังรูปที่ 2-9 ผลลัพธ์ที่ได้จากเครื่องมือ AR-Miner แสดงการจัดอันดับความสำคัญของกลุ่มบทวิจารณ์ของผู้ใช้งานจากมากไปน้อย สูงสุด 10 อันดับแรก โดยกลุ่มหัวข้อ “more theme” มีความสำคัญสูงที่สุด

	Review Instances of topic “more theme”	Score
1	Also we need more themes!	0.932
2	Just wish you had more themes or ability to make a custom theme.	0.800
...	.....	.....



รูปที่ 2-9 การแสดงผลของกลุ่มบทวิจารณ์ของผู้ใช้งานที่มีความสำคัญสูงสุด 10 อันดับแรกของเครื่องมือ AR-Miner [4]

แนวทางของงานวิจัยนี้คล้ายคลึงกันกับแนวทางการวิจัยของวิทยานิพนธ์คือ เริ่มจากการจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งาน และหลังจากนั้นจัดกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งาน ข้อแตกต่างจากงานวิจัยนี้คือ วิทยานิพนธ์จำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นประเภทความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน และระบุบทวิจารณ์ของผู้ใช้งานที่แตกต่างกันด้วยแนวทาง Clustering และ Text Similarity ก่อนจะนำข้อมูลบทวิจารณ์ของผู้ใช้งานไปสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์

2.2.5 Release Planning of Mobile Apps Based on User Reviews [5]

งานวิจัยนี้นำเสนอเครื่องมือที่ใช้สำหรับวางแผนกิจกรรมเพื่อพัฒนาและปรับปรุงแอปพลิเคชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานสำหรับนักพัฒนา มีทั้งหมดสามขั้นตอนคือ เริ่มจากจำแนกข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นประเภท Bug Report และการแนะนำฟีเจอร์ (Feature Suggestion) ขั้นตอนถัดมาคือการจัดกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งานที่เกี่ยวข้องกันด้วยอัลกอริทึมการจัดกลุ่มข้อมูล DBSCAN และขั้นตอนสุดท้ายคือการจัดลำดับความสำคัญของกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งานเพื่อใช้วางแผนพัฒนาแอปพลิเคชัน ผลการทดลองของงานวิจัยนี้พบว่ามีความแม่นยำสูงในจำแนกประเภทข้อมูลและการจัดกลุ่มข้อมูลที่เกี่ยวข้องกัน

แนวทางการวิจัยของงานวิจัยนี้คล้ายคลึงกันกับแนวทางการวิจัยของวิทยานิพนธ์คือ เริ่มจากการจำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานและหลังจากนั้นจัดกลุ่มข้อมูลบทวิจารณ์ของผู้ใช้งาน ข้อแตกต่างจากงานวิจัยนี้คือ วิทยานิพนธ์จำแนกประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็นประเภทความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน หลังจากนั้นระบุบทวิจารณ์ของผู้ใช้งานที่

แตกต่างกันด้วยแนวทาง Clustering และ Text Similarity ก่อนจะนำข้อมูลบทวิจารณ์ของผู้ใช้งานไปสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์

หมวดหมู่การสร้างเอกสารความต้องการของซอฟต์แวร์

2.2.6 A Semi-Automated Approach for Generating Natural Language Requirements Documents Based on Business Process Models [21]

งานวิจัยนี้นำเสนอแนวทางกึ่งอัตโนมัติในการสร้างเอกสารความต้องการในรูปแบบภาษาธรรมชาติจากโมเดลกระบวนการทางธุรกิจ (Business Process Models) แบ่งได้สองระยะคือ

1) ระยะการเตรียมการข้อมูล (Preparation Phase) เป็นระยะการเตรียมข้อมูลที่เกี่ยวข้อง ระยะนี้ดำเนินงานด้วยคน วิเคราะห์กิจกรรมที่ระบบสามารถทำงานได้ วิเคราะห์ข้อมูลที่จำเป็นต้องใช้ในการสร้างความต้องการของระบบ เช่น ใครมีบทบาทอะไร ทำอะไรได้ มีการติดต่อกับระบบอื่นหรือไม่ ข้อบังคับต่าง ๆ เป็นต้น
2) ระยะการสร้างเอกสารความต้องการ (Generation Phase) เป็นระยะการสร้างเอกสารความต้องการ โดยสร้างความต้องการในรูปแบบภาษาธรรมชาติจากการใช้ Requirement Boilerplates และเติมข้อมูลลงในแม่แบบด้วยข้อมูลที่สกัดได้จากโมเดลกระบวนการทางธุรกิจในระยะการเตรียมการข้อมูล แนวทางนี้เรียกว่า Template Filling เป็นวิธีที่รวดเร็ว มีความสามารถในการอ่านสูงและอาศัยความรู้เกี่ยวกับการสร้างประโยคภาษาธรรมชาติไม่มากนัก

ผลลัพธ์ของงานวิจัยนี้พบว่า เอกสารความต้องการมีคุณภาพทั้งด้านของความสามารถในการอ่าน ความครบถ้วน และความง่ายในการปรับปรุงเอกสารเมื่อมีการเปลี่ยนแปลงกระบวนการทางธุรกิจ

วิทยานิพนธ์นำแนวทางการสร้างประโยคความต้องการในรูปแบบภาษาธรรมชาติด้วย Requirement Boilerplates และแนวทาง Template Filling ของงานวิจัยนี้มาประยุกต์ใช้สร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งาน ข้อแตกต่างจากงานวิจัยนี้คือ วิทยานิพนธ์สร้างความต้องการของซอฟต์แวร์จากข้อมูลบทวิจารณ์ของผู้ใช้งาน ซึ่งลักษณะของข้อมูลและการได้มาของข้อมูลที่ใช้ในแม่แบบของประโยคแตกต่างกัน

2.2.7 Automatic Generation of a Software Requirements Specification (SRS) Document [22]

งานวิจัยนี้นำเสนอเครื่องมือซอฟต์แวร์ของกระบวนการวิศวกรรมความต้องการ เนื่องจากปัญหาการขาดเครื่องมือซอฟต์แวร์ในการสร้างความต้องการในรูปแบบภาษาธรรมชาติที่มีรูปแบบที่ดี

การสร้างข้อกำหนดความต้องการเป็นงานที่ยากและซับซ้อนต้องใช้การวิเคราะห์จากนักวิศวกรรมความต้องการ งานวิจัยนี้นำเสนอเครื่องมือชื่อว่า NALASS ประกอบด้วย 4 ส่วน

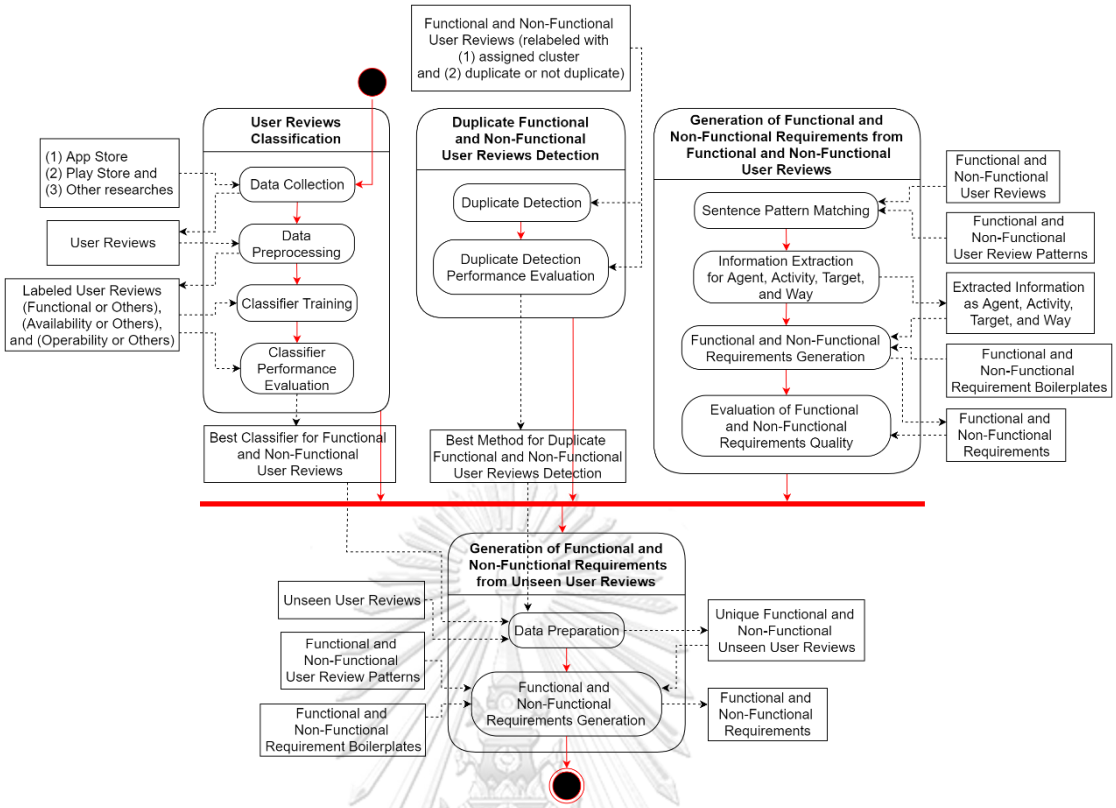
1) The Formalization Component เป็นส่วนที่ใช้สร้างโครงสร้างรูปแบบของประโยคที่จะใช้เป็นข้อกำหนดความต้องการ
2) The Questions Component เป็นส่วนการสร้างคำถามอัตโนมัติที่ได้จากระบบเพื่อนำไปใช้เป็นคำถามสอบถามข้อมูลที่ต้องการกับลูกค้า
3) The Diagrams Component เป็นส่วนการสร้างไดอะแกรมอัตโนมัติจากระบบ โดยไดอะแกรมสร้างจากข้อมูลที่ผู้ใช้งานตอบคำถามที่ระบบถาม หรือใส่ข้อมูลที่ต้องการสำหรับสร้างไดอะแกรมด้วยตัวเอง
4) The Documentation Component เป็นส่วนการสร้างเอกสารข้อกำหนดความต้องการตามแม่แบบที่กำหนด ข้อมูลที่ใช้สร้างเอกสารข้อกำหนดความต้องการประกอบด้วย แม่แบบ กฎในการแทนที่ข้อมูลในแม่แบบ ข้อมูลบทบาท หน้าที่ ข้อบังคับต่าง ๆ ที่เกี่ยวข้อง เป็นต้น นำข้อมูลต่าง ๆ มาประมวลผลสร้างเป็นเอกสารข้อกำหนดความต้องการตามแม่แบบที่กำหนด

แนวทางการสร้างเอกสารข้อกำหนดความต้องการของงานวิจัยนี้คล้ายคลึงกันกับแนวทางของวิทยานิพนธ์คือสร้างประโยคความต้องการในรูปแบบภาษาธรรมชาติจากการใช้ Requirement Boilerplates และวิธีการเติมข้อมูลลงในแม่แบบ ข้อแตกต่างจากงานวิจัยนี้คือวิทยานิพนธ์สร้างความต้องการจากข้อมูลบทวิจารณ์ของผู้ใช้งานและข้อมูลที่เติมลงในแม่แบบได้จากการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน

บทที่ 3
ภาพรวมวิธีการวิจัย

วิทยานิพนธ์มีแนวคิดในการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากการจำแนกบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันโดยเริ่มจาก 1) นำบทวิจารณ์ของผู้ใช้งานมาจำแนกประเภทออกเป็นบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน 2) เนื่องจากบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันนั้นมีโอกาสที่จะพูดถึงเรื่องเดียวกันได้ เพื่อลดความซ้ำซ้อนกันของข้อมูลบทวิจารณ์ของผู้ใช้งาน วิทยานิพนธ์จึงตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกัน โดยจะละทิ้งบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกันและเก็บบทวิจารณ์ของผู้ใช้งานที่แตกต่างกันไว้ เพื่อนำมาใช้สร้างประโยคความต้องการในขั้นตอนถัดไป 3) วิทยานิพนธ์สร้างความต้องการซอฟต์แวร์จากบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันที่คงเหลือจากการตรวจสอบความซ้ำซ้อนกันแล้ว ด้วยการสกัดข้อมูลที่สำคัญจากข้อมูลบทวิจารณ์ของผู้ใช้งาน โดยใช้ User Review Patterns และสร้างความต้องการซอฟต์แวร์ด้วยการแทนที่ข้อมูลที่สกัดได้ใน Requirement Boilerplates ที่กำหนดไว้ โดยภาพรวมวิธีการวิจัยของวิทยานิพนธ์ดังรูปที่ 3-1 มีทั้งหมด 4 เฟสคือ

1) **เฟสการจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน** เฟสนี้ข้อมูลบทวิจารณ์ของผู้ใช้งานจะถูกนำมาใช้ฝึกสอนโมเดลการเรียนรู้ของเครื่อง เพื่อใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้งานที่เป็นเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน
2) **เฟสการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน** เนื่องจากบทวิจารณ์ของผู้ใช้งานมีโอกาสกล่าวถึงเรื่องเดียวกันได้ เฟสนี้บทวิจารณ์ของผู้ใช้งานจะถูกพิจารณาถึงความซ้ำซ้อนกันเพื่อเก็บเฉพาะข้อมูลบทวิจารณ์ของผู้ใช้งานที่แตกต่างกันเท่านั้น
3) **เฟสการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ถูกจำแนกประเภทแล้ว** เฟสนี้จะสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ใช้ฝึกสอนโมเดลการเรียนรู้ของเครื่องในเฟส 1)
4) **เฟสการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ไม่เคยเห็นมาก่อน** เฟสนี้คือการทดลองนำแนวทางวิจัยของวิทยานิพนธ์มาประยุกต์ใช้งาน โดยจะนำแนวทางที่ดีที่สุดของทุกเฟสมารวมกัน ซึ่งทีมนักพัฒนาสามารถรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานที่ไม่เคยเห็นมาก่อน (Unseen) มาใช้สร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันได้



รูปที่ 3-1 ภาพรวมวิธีการวิจัยของวิทยานิพนธ์

รายละเอียดของเฟสที่ 1) ถึงเฟสที่ 4) อธิบายไว้ในบทที่ 4 ถึงบทที่ 7 ตามลำดับ

บทที่ 4
การจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน

เฟสนี้คือการพัฒนาและทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่องแบบมีผู้สอนเพื่อใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งาน ผลลัพธ์ที่ได้จากเฟสนี้คือโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันที่มีประสิทธิภาพมากที่สุด วิทยานิพนธ์กำหนดนิยามบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันดังนี้

1) บทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชัน (Functional User Reviews) คือบทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ฟังก์ชัน ความสามารถที่ระบบ ซอฟต์แวร์ หรือส่วนของซอฟต์แวร์สามารถทำได้ โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหาการร้องขอเนื้อหาที่ขาดหายไปหรือเพิ่มฟังก์ชันใหม่ รวมถึงการปรับปรุงหรือเพิ่มเติมฟังก์ชันที่มีอยู่ของแอปพลิเคชัน

ในส่วนของบทวิจารณ์ของผู้ใช้งานที่ไม่ใช่เชิงฟังก์ชัน (Non-functional User Reviews) แบ่งเป็น 2 ประเภทคือ

2) บทวิจารณ์ของผู้ใช้งานด้านความพร้อมใช้งาน (Availability User Reviews) คือบทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ระดับของความสามารถที่จะใช้งานและเข้าถึงระบบซอฟต์แวร์ หรือส่วนของซอฟต์แวร์เมื่อผู้ใช้งานต้องการใช้งาน โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหาหรือการร้องขอการปรับปรุงประสิทธิภาพการทำงาน
3) บทวิจารณ์ของผู้ใช้งานด้านความสามารถในการใช้งาน (Operability User Reviews) คือบทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ระดับคุณสมบัติของระบบซอฟต์แวร์ หรือส่วนของซอฟต์แวร์ที่ง่ายต่อการใช้งานและควบคุม เพื่อบรรลุวัตถุประสงค์การใช้งาน โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหาหรือการร้องขอการปรับปรุงประสิทธิภาพการทำงาน

เฟสนี้แบ่งออกเป็น 4 ขั้นตอนได้แก่ 4.1) ขั้นตอนการเก็บรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งาน 4.2) ขั้นตอนการเตรียมข้อมูลบทวิจารณ์ของผู้ใช้งาน 4.3) ขั้นตอนการพัฒนาโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน 4.4) ขั้นตอนการประเมินประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน

4.1 ขั้นตอนการเก็บรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งาน

วิทยานิพนธ์เก็บรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากทั้งหมด 3 แหล่งข้อมูล สำหรับใช้เป็นชุดข้อมูลฝึกสอนและทดสอบประสิทธิภาพของโมเดล วิทยานิพนธ์ตัดข้อมูลบทวิจารณ์ของผู้ใช้งานที่รวบรวมจากแต่ละแหล่งข้อมูลให้อยู่ในระดับประโยค รวมทั้งหมด 4,000 ประโยค บทวิจารณ์ของผู้ใช้งาน ข้อมูลบทวิจารณ์ของผู้ใช้งานจากแหล่งข้อมูลแรกจำนวน 1,295 ประโยค จากแหล่งข้อมูลที่สองจำนวน 953 ประโยคและจากแหล่งข้อมูลที่สามจำนวน 1,752 ประโยค

หมายเหตุ ในการทดลองของวิทยานิพนธ์ บทวิจารณ์ของผู้ใช้งานจะหมายถึงบทวิจารณ์ของผู้ใช้งานในระดับประโยค

รายละเอียดลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานและวิธีการรวบรวมจากแต่ละแหล่งข้อมูลมีดังต่อไปนี้

ลักษณะของข้อมูลที่ได้จากแหล่งข้อมูลแรก

ผู้วิจัยรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากงานวิจัย [2] ลักษณะของชุดข้อมูลบทวิจารณ์ของผู้ใช้งานถูกจำแนกออกเป็น 4 ประเภท ได้แก่ Bug Report, Feature Request, User Experience, Rating โดยวิทยานิพนธ์เลือกใช้ข้อมูลบทวิจารณ์ของผู้ใช้งานจากแหล่งข้อมูลนี้แค่เพียงสองประเภทคือ Bug Report และ Feature Request ดังตารางที่ 4-1 เนื่องจากเป็นข้อมูลที่สำคัญสำหรับการพัฒนาและปรับปรุงแอปพลิเคชัน

ตารางที่ 4-1 ตัวอย่างลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานจากงานวิจัย [2]

ประเภทข้อมูล	ตัวอย่างข้อมูลบทวิจารณ์ของผู้ใช้งาน
Bug Report	“Uploading is not working with the iOS6”
	“Every time I launch the app, it crashes”
Feature Request	“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.”

ลักษณะของข้อมูลที่ได้จากแหล่งข้อมูลที่สอง

ผู้วิจัยรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากงานวิจัย [3] ลักษณะของชุดข้อมูลบทวิจารณ์ของผู้ใช้งานถูกจำแนกออกเป็น 5 ประเภท คือ Functional, Usability, Reliability, Portability, Performance ตามนิยาม ISO/IEC 25010 Software Quality Model [12] วิทยานิพนธ์เลือกใช้

ข้อมูลบทวิจารณ์ของผู้ใช้งานจากแหล่งข้อมูลนี้แค่เพียงสองประเภท คือ Usability และ Reliability ดังตารางที่ 4-2

ตารางที่ 4-2 ตัวอย่างลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานจากงานวิจัย [3]

ประเภทข้อมูล	ตัวอย่างข้อมูลบทวิจารณ์ของผู้ใช้งาน
Usability	“I have to flip through chapters to start the book.”
Reliability	“The app just crashes after idling for five minutes.”

ลักษณะของข้อมูลที่ได้จากแหล่งข้อมูลที่สาม

ผู้วิจัยรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานจากแอปสโตร์และเพลย์สโตร์ด้วยวิธีการสกัดข้อมูลจากหน้าเว็บไซต์ (Screen Scraping) โดยใช้ไลบรารี app-store-scraper [23] สำหรับดึงข้อมูลบทวิจารณ์ของผู้ใช้งานบนแอปสโตร์และไลบรารี google-play-scraper [24] สำหรับดึงข้อมูลบทวิจารณ์ของผู้ใช้งานบนเพลย์สโตร์ รวบรวมจาก 8 แอปพลิเคชัน คือ Agoda, Airbnb, Instagram, Kindle, Messenger, TikTok, Wattpad, WhatsApp โดยทั้ง 8 แอปพลิเคชันมีหมวดหมู่คล้ายคลึงกันกับข้อมูลบทวิจารณ์ของผู้ใช้งานจากแหล่งข้อมูลแรกและแหล่งข้อมูลที่สอง ลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้จากแหล่งข้อมูลที่สามดังรูปที่ 4-1


```
{   id: 'gp:AOqpTOF39mpW-6gurlkCCTV_8qnKne7O5wcFsLc6iGVot5hHpplqPCqIiVL2fjximXNujuM
    userName: 'Millie Hawthorne',
    userImage: 'https://lh5.googleusercontent.com/-Q_FTAEBH2Qg/AAAAAAAAAAI/AAAAAAAA
    date: '2013-11-10T18:31:42.174Z',
    url: 'https://play.google.com/store/apps/details?id=com.dxco.pandavszombies&rev
    score: 5,
    scoreText: '5',
    title: 'CAN NEVER WAIT TILL NEW UPDATE',
    text: 'Love it but needs to pay more attention to pocket edition',
    replyDate: null,
    replyText: null,
    version: null,
    thumbsUp: 29
    criterias: [] } ]
```


รูปที่ 4-1 ตัวอย่างข้อมูลบทวิจารณ์ของผู้ใช้งานที่ได้จากการดึงข้อมูลบนเพลย์สโตร์ [24]

4.2 ขั้นตอนการเตรียมข้อมูลบทวิจารณ์ของผู้ใช้งาน

ในขั้นตอนการเตรียมข้อมูลบทวิจารณ์ของผู้ใช้งานประกอบด้วย 3 ขั้นตอนคือ 4.2.1) การติดป้ายประเภทของข้อมูล (Data Labeling) 4.2.2) การทำความสะอาดข้อมูล (Data Cleaning) และ 4.2.3) การสกัดคุณลักษณะ (Feature Extraction)

4.2.1 Data Labeling

วิทยานิพนธ์นำข้อมูลบทวิจารณ์ของผู้ใช้งานที่ได้รวบรวมในขั้นตอนที่ 4.1 มาติดป้ายประเภทของข้อมูล เพื่อใช้เป็นชุดข้อมูลฝึกสอนและทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่อง
โดย 1 ประโยคบทวิจารณ์ของผู้ใช้งานสามารถติดป้ายได้เพียง 1 ประเภทป้ายเท่านั้น วิทยานิพนธ์ติดป้ายทั้งหมด 3 ชุดข้อมูลบทวิจารณ์ของผู้ใช้งานคือ

1) ชุดข้อมูลบทวิจารณ์ของผู้ใช้งานติดป้ายประเภท Functional และ Others สำหรับใช้เป็นชุดฝึกสอนและทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่องเพื่อใช้จำแนก Functional User Reviews
2) ชุดข้อมูลบทวิจารณ์ของผู้ใช้งานติดป้ายประเภท Availability และ Others สำหรับใช้เป็นชุดฝึกสอนและทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่องเพื่อใช้จำแนก Availability User Reviews
3) ชุดข้อมูลบทวิจารณ์ของผู้ใช้งานติดป้ายประเภท Operability และ Others สำหรับใช้เป็นชุดฝึกสอนและทดสอบประสิทธิภาพโมเดลการเรียนรู้ของเครื่องเพื่อใช้จำแนก Operability User Reviews

นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานแต่ละประเภทป้ายของทั้ง 3 ชุดข้อมูล ดังตารางที่ 4-3 ถึงตารางที่ 4-5

ตารางที่ 4-3 นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานของป้ายประเภท Functional และ Others ของชุดข้อมูลที่ 1)

ประเภทป้าย	นิยาม	ตัวอย่างบทวิจารณ์ของผู้ใช้งาน
Functional	บทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ฟังก์ชัน ความสามารถที่ระบบ ซอฟต์แวร์ หรือส่วนของซอฟต์แวร์สามารถทำได้ โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหา การร้องขอเนื้อหาที่ขาดหายไปหรือเพิ่มฟังก์ชันใหม่ รวมถึงการปรับปรุงหรือเพิ่มเติมฟังก์ชันที่มีอยู่ของแอปพลิเคชัน	“It would be great if we could copy and paste text.”

ตารางที่ 4-3 นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานของป้ายประเภท Functional และ Others ของชุดข้อมูลที่ 1) (ต่อ)

ประเภทป้าย	นิยาม	ตัวอย่างบทวิจารณ์ของผู้ใช้งาน
Others	บทวิจารณ์ของผู้ใช้งานที่ไม่เกี่ยวข้องกับประเภทป้ายบทวิจารณ์ของผู้ใช้งานข้างต้น เช่น การแสดงออกทางอารมณ์ การบ่นทั่วไป	“so i spent $399 on nothing.”

ตารางที่ 4-4 นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานของป้ายประเภท Availability และ Others ของชุดข้อมูลที่ 2)

ประเภทป้าย	นิยาม	ตัวอย่างบทวิจารณ์ของผู้ใช้งาน
Availability	บทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ระดับของความสามารถที่จะใช้งานและเข้าถึงระบบซอฟต์แวร์ หรือส่วนของซอฟต์แวร์เมื่อผู้ใช้งานต้องการใช้งาน โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหาหรือการร้องขอการปรับปรุงประสิทธิภาพการทำงาน	“The app just crashes after idling for five minutes.”
Others	บทวิจารณ์ของผู้ใช้งานที่ไม่เกี่ยวข้องกับประเภทป้ายบทวิจารณ์ของผู้ใช้งานข้างต้น เช่น การแสดงออกทางอารมณ์ การบ่นทั่วไป	“so i spent $399 on nothing.”

ตารางที่ 4-5 นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานของป้ายประเภท Operability และ Others ของชุดข้อมูลที่ 3)

ประเภทป้าย	นิยาม	ตัวอย่างบทวิจารณ์ของผู้ใช้งาน
Operability	บทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ระดับคุณสมบัติของระบบ ซอฟต์แวร์ หรือส่วนของซอฟต์แวร์ที่ง่ายต่อการใช้งานและควบคุม เพื่อบรรลุวัตถุประสงค์การใช้งาน โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหาหรือการร้องขอการปรับปรุงประสิทธิภาพการทำงาน	“I have to flip through chapters to start the book.”

ตารางที่ 4-5 นิยามและตัวอย่างบทวิจารณ์ของผู้ใช้งานของป้ายประเภท Operability และ Others ของชุดข้อมูลที่ 3) (ต่อ)

ประเภทป้าย	นิยาม	ตัวอย่างบทวิจารณ์ของผู้ใช้งาน
Others	บทวิจารณ์ของผู้ใช้งานที่ไม่เกี่ยวข้องกับประเภทป้ายบทวิจารณ์ของผู้ใช้งานข้างต้น เช่น การแสดงออกทางอารมณ์ การบ่นทั่วไป	“so i spent $399 on nothing.”

กระบวนการติดป้ายข้อมูลของวิทยานิพนธ์มีรายละเอียดดังนี้

a) ติดป้ายประเภทข้อมูลบทวิจารณ์ของผู้ใช้งานด้วยวิศวกรซอฟต์แวร์ ทั้งหมด 3 คน
b) ผู้ติดป้ายทุกคนจะได้รับคู่มือแนวทางการติดป้าย ดังรูปที่ 4-2 ซึ่งอธิบายถึงแนวทางการติดป้าย นิยามของป้ายแต่ละประเภท และตัวอย่างข้อมูลบทวิจารณ์ของผู้ใช้งานแต่ละประเภทป้าย นอกจากนี้ก่อนการติดป้าย ผู้ติดป้ายจะได้รับการอธิบายและทดลองติดป้ายร่วมกันก่อน เพื่อสร้างความเข้าใจให้ตรงกันระหว่างผู้ติดป้ายด้วย
c) กรณีที่พบว่าติดป้ายบทวิจารณ์ของผู้ใช้งานไม่ตรงกันระหว่างผู้ติดป้าย จะมีการพูดคุยกันระหว่างผู้ติดป้าย เพื่อหามติเอกฉันท์ของป้ายบทวิจารณ์ของผู้ใช้งานนั้น หากไม่สามารถหาข้อสรุปได้จากสาเหตุข้อมูลบทวิจารณ์ของผู้ใช้งานมีความกำกวม บทวิจารณ์ของผู้ใช้งานนั้นจะถูกลบออก

คู่มือการติดป้าย (Coding Guide)

ข้อมูลที่นำมาใช้ในการติดป้ายข้อมูลคือ ข้อมูลบทวิจารณ์ของผู้ใช้งาน (User reviews) ของแอปพลิเคชันต่าง ๆ บนแอปสโตร์และเพลย์สโตร์ จำนวนทั้งหมด 4,000 ข้อความ สำหรับการติดป้ายข้อมูลประกอบด้วยประเภทป้ายทั้งหมด 4 ประเภทได้แก่ Functional, Operability, Availability, และ Others มีนิยามและตัวอย่างของแต่ละประเภทดังต่อไปนี้

หมายเหตุ โปรดใช้คู่มือการติดป้ายควบคู่ไปกับการติดป้ายข้อมูลแต่ละประเภท ควรอ่านข้อความให้ครบก่อนติดป้ายข้อมูล ถ้าข้อความไม่ใช่ภาษาอังกฤษและความยาวไม่เกิน 3 คำข้ามได้เลย **ในแต่ละข้อความเลือกประเภทป้ายได้เพียง 1 ประเภทเท่านั้น**

Functional: บทวิจารณ์ของผู้ใช้งานที่อธิบายถึง ฟังก์ชัน ความสามารถที่ระบบ ซอฟต์แวร์ หรือส่วนของซอฟต์แวร์สามารถทำได้ โดยการอธิบายอยู่ในรูปแบบของการรายงานปัญหา การร้องขอเนื้อหาที่ขาดหายไปหรือเพิ่มฟังก์ชันใหม่ รวมถึงการปรับปรุงหรือเพิ่มเติมฟังก์ชันที่มีอยู่ของแอปพลิเคชัน

ตัวอย่างของข้อมูลประเภทนี้

“It would be great if we could copy and paste text.”,
“It would be nice if you could pin posts to your page from the app.”,
“can you plz make a screen recorder so we can make cool videos for youtube.”,
“It would be helpful if entrance and communion antiphons were included.”,
“It needs to be able to have music during the presentation and more animations.”

รูปที่ 4-2 ตัวอย่างคู่มือการติดป้าย

หมายเหตุ บทวิจารณ์ของผู้ใช้งานที่ไม่ใช่ภาษาอังกฤษ หรือมีความยาวของบทวิจารณ์ของผู้ใช้งานไม่มากกว่า 3 คำจะถูกลบออกเนื่องจากมีความยาวสั้นเกินกว่าจะสื่อความได้

4.2.2 Data Cleaning

เนื่องจากข้อมูลบทวิจารณ์ของผู้ใช้งานส่วนมากเขียนด้วยภาษาที่ไม่เป็นทางการและบางครั้งประกอบด้วยข้อมูลที่ไม่สามารถนำมาประกอบการวิเคราะห์ได้ เช่น คำแสลง อักษรย่อ อีโมจิ เครื่องหมายวรรคตอน เป็นต้น วิทยานิพนธ์จึงได้ปรับข้อมูลให้อยู่ในแนวทางเดียวกันและทำความสะอาดข้อมูล เพื่อให้ได้ข้อมูลที่เหมาะสมต่อการวิเคราะห์และเพื่อเพิ่มประสิทธิภาพการเรียนรู้ให้กับโมเดลการเรียนรู้ของเครื่อง มีรายละเอียดดังนี้

1) ลบขึ้นบรรทัดใหม่ (Newline) ลบพื้นที่ว่าง (Space) ลบตัวเลขหัวข้อรายการ (List Numbering) ตัวอย่างเช่น “ this is a string ” เป็น “this is a string”
2) ลบสัญรูปอารมณ์ (Emoticon) และอีโมจิ (Emoji) ด้วยไลบรารี Demoji [25] ตัวอย่างเช่น “win great start by 👨‍✈️ ” เป็น “win great start by”
3) คืนรูปคำแสลงและอักษรย่อด้วยข้อมูลจาก www.noslang.com/dictionary/ โดยใช้วิธีการดึงข้อมูลจากหน้าเว็บไซต์ด้วยไลบรารี BeautifulSoup
4) คืนรูปสัญลักษณ์ ได้แก่ & ด้วย “and” และ “/” ด้วย “or”
5) คืนรูปคำย่อทั่วไป อ้างอิงคำย่อทั่วไปและรูปคำเดิมด้วยข้อมูลจาก [26] ตัวอย่างเช่น “can’t” เป็น “cannot”
6) แก้ไขข้อผิดพลาดของการสะกดคำและโครงสร้างไวยากรณ์ด้วยไลบรารี Gingerit [27] ตัวอย่างเช่น “The smelt of fliwers bring back memories.” เป็น “The smell of flowers brings back memories.”
7) ลบเครื่องหมายวรรคตอน (Punctuation) ตัวอย่างเช่น “!”, “?”, “@”
8) ลบคำที่ไม่มีนัยสำคัญต่อบริบท (Stopwords) คือคำที่พบได้บ่อยในบริบทนั้นและสามารถพบได้บ่อยเช่นกันในบริบทอื่น ๆ ทำให้คำนั้นไม่มีความสำคัญใด ๆ ไม่สามารถบอกได้ถึงใจความสำคัญของบริบทนั้น เช่น a, an, the วิทยานิพนธ์ใช้รายการ Stopwords ของไลบรารี Gensim [28] และยกเว้นการลบ Stopwords บางคำที่อาจจะมีประโยชน์ต่อการเรียนรู้ของโมเดลการเรียนรู้ของเครื่องได้แก่ “cant”, “don”, “bill”, “hasnt”, “didn”, “couldnt”, “please”, “doing”, “cannot”, “must”, “can”, “get”, “has”, “does”, “would”, “down”, “un”, “without”, “made”, “should”, “doesn”, “not”, “do”, “might”, “done”, “had”

9) ลบบทวิจารณ์ของผู้ใช้งานที่ไม่มีข้อมูล (Null) และซ้ำซ้อนกัน (Duplicate) เนื่องจากหลังจากผ่านการทำความสะอาดข้อมูลต่าง ๆ แล้ว บางบทวิจารณ์ของผู้ใช้งานไม่มีข้อมูลเหลืออยู่เลย และเพื่อไม่เป็นการเพิ่มจำนวนข้อมูลบทวิจารณ์ของผู้ใช้งานเดิมซ้ำ ซึ่งมีผลต่อประสิทธิภาพการเรียนรู้ของโมเดลการเรียนรู้ของเครื่อง

จำนวนข้อมูลบทวิจารณ์ของผู้ใช้งานสุทธิหลังจากผ่านขั้นตอนการทำ Data Labeling และ Data Cleaning ข้อมูลบทวิจารณ์ของผู้ใช้งานคงเหลือ 3,989 ประโยค จากสาเหตุคือมีความยาวไม่มากกว่า 3 คำจำนวน 1 บทวิจารณ์ของผู้ใช้งาน, ไม่ใช่ภาษาอังกฤษจำนวน 1 บทวิจารณ์ของผู้ใช้งาน, ความหมายกำกวมจำนวน 7 บทวิจารณ์ของผู้ใช้งาน, และสาเหตุจาก Null จำนวน 2 บทวิจารณ์ของผู้ใช้งาน

จำนวนบทวิจารณ์ของผู้ใช้งานติดป้ายประเภท Functional และ Others, ประเภท Availability และ Others, ประเภท Operability และ Others ดังตารางที่ 4-6 ถึงตารางที่ 4-8

ตารางที่ 4-6 จำนวนบทวิจารณ์ของผู้ใช้งานที่ติดป้ายประเภท Functional และ Others ของชุดข้อมูลที่ 1)

แหล่งที่มา	จำนวนทั้งหมด	Functional	Others
จากงานวิจัย [2]	1,288	178	1,110
จากงานวิจัย [3]	951	77	874
จาก Screen Scraping	1,750	640	1,110
จำนวนทั้งหมด	3,989	895	3,094

ตารางที่ 4-7 จำนวนบทวิจารณ์ของผู้ใช้งานที่ติดป้ายประเภท Availability และ Others ของชุดข้อมูลที่ 2)

แหล่งที่มา	จำนวนทั้งหมด	Availability	Others
จากงานวิจัย [2]	1,288	374	914
จากงานวิจัย [3]	951	355	596
จาก Screen Scraping	1,750	229	1,521
จำนวนทั้งหมด	3,989	958	3,031

ตารางที่ 4-8 จำนวนบทวิจารณ์ของผู้ใช้งานที่ติดป้ายประเภท Operability และ Others ของชุดข้อมูลที่ 3)

แหล่งที่มา	จำนวนทั้งหมด	Operability	Others
จากงานวิจัย [2]	1,288	70	1,218
จากงานวิจัย [3]	951	80	871
จาก Screen Scraping	1,750	607	1,143
จำนวนทั้งหมด	3,989	757	3,232

4.2.3 Feature Extraction

เนื่องจากบทวิจารณ์ของผู้ใช้งานมีลักษณะเป็นข้อความ วิทยานิพนธ์แปลงข้อความให้อยู่ในรูปเวกเตอร์คุณลักษณะ (Representation) ของข้อความ เพื่อให้อัลกอริทึมการเรียนรู้ของเครื่องสามารถเข้าใจและประมวลผลได้ คุณลักษณะของข้อความ (Feature) ที่ใช้ทดลองจำแนกประเภทบทวิจารณ์ของผู้ใช้งานของวิทยานิพนธ์ แบ่งได้ 2 ประเภท คือ

1) คุณลักษณะที่เกี่ยวกับข้อความ (Textual Feature) ได้แก่
    a) Bag of Words (BoW) คือความถี่ของคำที่ปรากฏในเอกสาร
    b) Term Frequency-Inverse Document Frequency (TF-IDF) คือการคำนวณความถี่และความสำคัญของคำในเอกสาร
    c) Doc2vec คือเวกเตอร์เอกสารของข้อมูล
    d) Stemming of Words คือการคืนรูปรากศัพท์ของคำแต่คืนรูปจากการตัดส่วนท้าย (Suffix) ของคำ ตัวอย่างเช่น “flying” คืนรูปเป็น “fly”
    e) Lemmatization of Words คือการคืนรูปรากศัพท์ดั้งเดิมของคำ ตัวอย่างเช่น “is” คืนรูปเป็น “be”
2) คุณลักษณะที่เกี่ยวกับรายละเอียดของบทวิจารณ์ของผู้ใช้งาน (User Review Metadata) ได้แก่
    a) Number of Words (n_tokens) คือจำนวนของคำในประโยค
    b) Frequency of Part of Speech Tagging (POS) คือจำนวนหน้าที่ของคำที่พบในประโยค วิทยานิพนธ์ใช้รายการแท็กซึ่งประกอบด้วย คำนาม (Noun) คำกริยา (Verb) คำคุณศัพท์ (Adjective) คำกริยาวิเศษณ์ (Adverb) จำนวนนับ (Cardinal Digit) คำกริยาช่วย (Modal) และคำแสดงเจ้าของ (Possessive)

c) Sentiment คือการวิเคราะห์ความรู้สึกของประโยค วิทยานิพนธ์คำนวณความรู้สึกด้วยแนวทาง Rule Based โดยใช้ไลบรารี Vadar [29] แต่ละประโยคประกอบด้วยคะแนนทั้งหมด 3 ค่า คือ บวก (Positive) กลาง (Negative) และลบ (Negative)

4.3 ขั้นตอนการพัฒนาโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน

ขั้นตอนนี้เป็นการทดลองพัฒนาโมเดลการเรียนรู้ของเครื่อง เพื่อใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้งานออกเป็นบทวิจารณ์เชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน ทั้งหมด 3 ประเภท ได้แก่ Functional User Reviews, Availability User Reviews, Operability User Reviews โดยนำข้อมูลบทวิจารณ์ของผู้ใช้งานที่ได้ติดป้าย รวมถึงแปลงเป็นเวกเตอร์คุณลักษณะของข้อความเรียบร้อยแล้วจากขั้นตอนที่ 4.2) มาใช้สร้างโมเดลการเรียนรู้ของเครื่องแบบมีผู้ฝึกสอน

วิทยานิพนธ์แบ่งข้อมูลบทวิจารณ์ของผู้ใช้งานออกเป็น Training Set และ Test Set ด้วยวิธีการสุ่มตัวอย่างแบบแบ่งชั้น (Stratified Random Sampling) ในอัตราส่วน 80 ต่อ 20 ตามลำดับ รายละเอียดจำนวนบทวิจารณ์ของผู้ใช้งานแต่ละประเภทหลังจากแบ่งเป็น Training Set และ Test Set ดังตารางที่ 4-9 ถึงตารางที่ 4-11

วิทยานิพนธ์ฝึกสอนโมเดลการเรียนรู้ของเครื่องแบบ Binary Classifier และได้นำเทคนิคต่าง ๆ มาใช้ฝึกสอนโมเดลการเรียนรู้ของเครื่อง เพื่อช่วยลดเวลาและทรัพยากรสำหรับฝึกสอนโมเดลการเรียนรู้ รวมถึงเพื่อเพิ่มประสิทธิภาพให้กับโมเดลการเรียนรู้ มีทั้งหมด 4 เทคนิค โดยมีรายละเอียดดังนี้

1) SVMSMOTE [30] คือการสร้างข้อมูลสังเคราะห์จากข้อมูลที่มีอยู่ เป็นวิธีการจัดการกับข้อมูลที่มีลักษณะสัดส่วนประเภทของข้อมูลไม่เท่ากัน (Data Imbalancing) จำนวนบทวิจารณ์ของผู้ใช้งานแต่ละประเภทหลังจากการทำ SVMSMOTE ของวิทยานิพนธ์ ดังตารางที่ 4-9 ถึงตารางที่ 4-11
2) Chi-Square [31] คืออัลกอริทึมที่ใช้เลือก Feature ที่สำคัญให้กับโมเดลการเรียนรู้ของเครื่อง ซึ่งสามารถช่วยลดมิติของข้อมูลที่สูงได้ (High Dimension) ตัวอย่างเช่น ข้อมูลข้อความ ข้อมูลรูปภาพ เป็นต้น
3) MinMaxScaler [32] คือการทำ Data Normalization ประเภทหนึ่ง เพื่อปรับข้อมูลให้อยู่ในหน่วยเดียวกันและสามารถเปรียบเทียบกันได้ ซึ่งค่าที่ได้จะอยู่ระหว่างช่วง 0 และ 1

4) GridSearchCV [32] คือการทำ Hyperparameter Tuning เพื่อค้นหาค่าพารามิเตอร์ที่เหมาะสมกับโมเดลการเรียนรู้ของเครื่อง

ตารางที่ 4-9 จำนวนบทวิจารณ์ของผู้ใช้งานประเภท Functional และ Other หลังจากแบ่งข้อมูลออกเป็น Training Set และ Test Set และหลังจากทำ SVMSMOTE

Data (Split 80:20)			**Data (Training SVMSMOTE)**		
Training Set	Functional	716	Training Set	Functional	2,475
	Others	2,475		Others	2,475
	Total	3,191		Total	4,950
Test Set	Functional	179			
	Others	619			
	Total	798			

ตารางที่ 4-10 จำนวนบทวิจารณ์ของผู้ใช้งานประเภท Availability และ Other หลังจากแบ่งข้อมูลออกเป็น Training Set และ Test Set และหลังจากทำ SVMSMOTE

Data (Split 80:20)			**Data (Training SVMSMOTE)**		
Training Set	Availability	766	Training Set	Availability	2,425
	Others	2,425		Others	2,425
	Total	3,191		Total	4,850
Test Set	Availability	192			
	Others	606			
	Total	798			

ตารางที่ 4-11 จำนวนบทวิจารณ์ของผู้ใช้งานประเภท Operability และ Other หลังจากแบ่งข้อมูลออกเป็น Training Set และ Test Set และหลังจากทำ SVMSMOTE

Data (Split 80:20)			**Data (Training SVMSMOTE)**		
Training Set	Operability	606	Training Set	Operability	2,585
	Others	2,585		Others	2,585
	Total	3,191		Total	5,170

ตารางที่ 4-11 จำนวนบทวิจารณ์ของผู้ใช้งานประเภท Operability และ Other หลังจากแบ่งข้อมูลออกเป็น Training Set และ Test Set และหลังจากทำ SVMSMOTE (ต่อ)

Data (Split 80:20)		
Test Set	Operability	151
	Others	647
	Total	798

อัลกอริทึมการเรียนรู้ของเครื่องที่นำมาทดลองในวิทยานิพนธ์ โดยเลือกจากอัลกอริทึมที่ได้รับความนิยมจากงานวิจัยส่วนใหญ่ [33] ให้ประสิทธิภาพผลการทำนายการจำแนกประเภทที่ดี ง่ายต่อการตั้งค่าใช้งาน และเหมาะกับลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานที่มีลักษณะเป็นข้อความ ซึ่งมีจำนวนของ Feature ค่อนข้างมาก มีรายละเอียดดังต่อไปนี้ [7]

1) Naïve Bayes คืออัลกอริทึมทางสถิติ จัดประเภทของข้อมูลด้วยการคำนวณความน่าจะเป็นของประเภทข้อมูล เป็นอัลกอริทึมการจำแนกประเภทชนิด Generative Classifiers คือจะคืนค่าประเภทของข้อมูลที่มีความน่าจะเป็นใกล้เคียงกับข้อมูลมากที่สุด เป็นอัลกอริทึมที่ให้ประสิทธิภาพผลการทำนายการจำแนกประเภทที่ดี ง่ายต่อการตั้งค่าใช้งาน
2) Logistic Regression คืออัลกอริทึมทางสถิติ จัดประเภทของข้อมูลด้วยการคำนวณความน่าจะเป็นของประเภทข้อมูล เป็นอัลกอริทึมการจำแนกประเภทชนิด Discriminative Classifiers โดยจะเรียนรู้ Feature ที่สำคัญของข้อมูล เพื่อแยกความแตกต่างของประเภทของข้อมูล มีความแม่นยำมากกว่า Generative Classifiers เหมาะกับข้อมูลขนาดใหญ่
3) Decision Trees คืออัลกอริทึมเงื่อนไขการตัดสินใจ มีลักษณะเป็นโครงสร้างต้นไม้ ประกอบด้วย โหนด (Node) ที่เป็นเงื่อนไขการตัดสินใจ และผลลัพธ์ (Leaves) ของต้นไม้ ซึ่งก็คือประเภทของข้อมูลที่จะจำแนก โดยเงื่อนไขการตัดสินใจของต้นไม้ได้จาก Feature ของ Training Set
4) Random Forest เป็นอัลกอริทึมกลุ่มเดียวกับ Decision Trees มีลักษณะเป็นโครงสร้างต้นไม้หลายต้น เพื่อนำต้นไม้หลาย ๆ ต้นที่มีความแตกต่างกันเล็กน้อยมาคำนวณค่าเฉลี่ยประสิทธิภาพการจำแนกประเภท เพื่อช่วยลดการเกิด Overfitting คือโมเดลการเรียนรู้ของเครื่องที่ได้จากการฝึกสอนทำงานได้ดีเพียงกับข้อมูลชุดฝึกสอน ซึ่งเป็นจุดอ่อนของอัลกอริทึม Decision Trees แต่ยังคงประสิทธิภาพความ

แม่นยำในการทำนายผลการจำแนกประเภทอยู่ Random Forest เป็นอัลกอริทึมที่ได้รับความนิยมมากในปัจจุบัน

4.4 ขั้นตอนการประเมินประสิทธิภาพโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน

สำหรับการประเมินประสิทธิภาพของแต่ละโมเดลการเรียนรู้ของเครื่อง วิทยานิพนธ์เปรียบเทียบประสิทธิภาพของแต่ละโมเดลการเรียนรู้ของเครื่องจากการคำนวณค่า Precision, Recall, F1-Score, Accuracy ด้วยไลบรารี Scikit-learn [34] วิทยานิพนธ์ใช้วิธี Stratified 10-Fold Cross Validation ขณะที่ฝึกสอนโมเดลการเรียนรู้ของเครื่องด้วยการผสมผสานทั้ง Feature เทคนิค และอัลกอริทึมการเรียนรู้ต่าง ๆ เข้าด้วยกัน หลังจากนั้นประเมินประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องด้วยชุดข้อมูล Test Set

ผลลัพธ์จากการประเมินประสิทธิภาพของแต่ละโมเดลการเรียนรู้ของเครื่องด้วยชุดข้อมูล Test Set วิทยานิพนธ์เลือกโมเดลการเรียนรู้ของเครื่องที่มีค่า F1-Score สูงสุด 3 อันดับแรกจากแต่ละอัลกอริทึมการเรียนรู้ของเครื่องมาแสดงผล ทั้งหมด 12 โมเดลการเรียนรู้ โดยแบ่งตามการจำแนกประเภทของบทวิจารณ์ของผู้ใช้งานคือประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ดังตารางที่ 4-12 ถึงตารางที่ 4-14 ในส่วนของเครื่องมือที่วิทยานิพนธ์ใช้ในการทดลองคือ Google Colab Pro รันบน TPU ด้วย High Ram

จากตารางที่ 4-12 ประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews พบว่า Textual Feature ที่มีความสำคัญต่อการเรียนรู้ของโมเดลคือ BoW และ TF-IDF ในขณะที่ Doc2vec ไม่มีส่วนช่วยให้โมเดลมีประสิทธิภาพการเรียนรู้ที่สูง การทำ Stemming และ Lemmatization มีประโยชน์ต่อประสิทธิภาพการเรียนรู้ของโมเดล นอกจากนี้ User Review Metadata Feature ทั้งหมดคือ n_tokens, POS, Sentiment มีนัยสำคัญต่อประสิทธิภาพการเรียนรู้ของโมเดล ท่ามกลางเทคนิคที่นำมาใช้ทดลองเพื่อเพิ่มประสิทธิภาพการเรียนรู้ให้แก่โมเดล เทคนิค SVMSMOTE, MinMaxScaler, GridSearchCV มีส่วนช่วยให้โมเดลมีประสิทธิภาพการเรียนรู้ที่สูง นอกจากนี้เทคนิค MinMaxScaler มีส่วนช่วยลดเวลาสำหรับใช้ฝึกสอนโมเดล ในขณะที่เทคนิค Chi-Square ไม่มีผลต่อประสิทธิภาพการเรียนรู้ของโมเดล โดยช่วงค่า F1-Score ของโมเดลการเรียนรู้ของเครื่องสูงสุด 3 อันดับแรกจากแต่ละอัลกอริทึมการเรียนรู้ทั้งหมดประมาณ 69.4% - 75.4% ซึ่งอัลกอริทึมการเรียนรู้ที่มีประสิทธิภาพสูงที่สุดคือ Logistic Regression รองลงมาคือ Random Forest สำหรับโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews คืออัลกอริทึมการเรียนรู้ Logistic Regression ที่ถูกฝึกสอนด้วย BoW, Sentiment, POS ร่วมกับ

เทคนิค SVMSMOTE และ MinMaxScaler โดยมีค่า F1-Score อยู่ที่ 75.4% ค่า Precision อยู่ที่ 72.3% ค่า Recall อยู่ที่ 78.8% และค่า Accuracy อยู่ที่ 88.5% ซึ่งใช้เวลาฝึกสอนโดยประมาณอยู่ที่ 39.914 วินาที

ตารางที่ 4-12 ประสิทธิภาพของโมเดลจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews สูงสุด 3 อันดับแรกของแต่ละอัลกอริทึมการเรียนรู้

Configuration	Functional User Reviews				
	Precision	Recall	F1-Score	Accuracy	Fit Time (Sec)
Naïve Bayes + BoW + POS + GridSearchCV	0.684	0.726	0.705	0.863	0.868
Naïve Bayes + BoW + n_tokens + POS + GridSearchCV	0.684	0.726	0.705	0.863	0.755
Naïve Bayes + BoW + n_tokens + Sentiment + POS + GridSearchCV	0.683	0.721	0.701	0.862	0.702
Logistic Regression + BoW + Sentiment + POS + SVMSMOTE + MinMaxScaler	**0.723**	**0.788**	**0.754**	**0.885**	**39.914**
Logistic Regression + TF-IDF + Lemmatization + Sentiment + POS + SVMSMOTE	0.719	0.788	0.752	0.883	32.767
Logistic Regression + BoW + Lemmatization + Sentiment + POS + SVMSMOTE + MinMaxScaler	0.716	0.788	0.750	0.882	32.616
Decision Trees + BoW + Stemming + n_tokens + POS	0.662	0.743	0.700	0.857	2.208
Decision Trees + BoW + Stemming + n_tokens + POS + MinMaxScaler	0.662	0.743	0.700	0.857	1.521
Decision Trees + TF-IDF + n_tokens + SVMSMOTE	0.685	0.704	0.694	0.861	7.997
Random Forest + TF-IDF + Lemmatization + n_tokens + SVMSMOTE + GridSearchCV	0.778	0.726	0.751	0.892	93.570
Random Forest + TF-IDF + Lemmatization + n_tokens + SVMSMOTE + MinMaxScaler + GridSearchCV	0.778	0.726	0.751	0.892	92.541
Random Forest + TF-IDF + Stemming + Sentiment + POS	0.772	0.721	0.746	0.890	3.570

จากตารางที่ 4-13 ประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Availability User Reviews พบว่า Textual Feature ที่มีความสำคัญต่อการเรียนรู้ของโมเดลคือ BoW และ TF-IDF ในขณะที่ Doc2vec ไม่มีส่วนช่วยให้โมเดลมีประสิทธิภาพการเรียนรู้ที่สูง การทำ Stemming และ Lemmatization สนับสนุนประสิทธิภาพการเรียนรู้ให้กับโมเดล นอกจากนี้ User Review Metadata Feature ได้แก่ n_tokens, POS, Sentiment มีนัยสำคัญต่อประสิทธิภาพการเรียนรู้ของโมเดล เทคนิค SVMSMOTE, MinMaxScaler, GridSearchCV ช่วยเพิ่มประสิทธิภาพการเรียนรู้ให้แก่โมเดล ในขณะที่เทคนิค Chi-Square ไม่มีผลต่อประสิทธิภาพการเรียนรู้ของโมเดล โดยช่วงค่า F1-Score ของโมเดลการเรียนรู้ของเครื่องสูงสุด 3 อันดับแรกจากแต่ละอัลกอริทึมการเรียนรู้ทั้งหมดประมาณ 64.3% - 73.5% ซึ่งอัลกอริทึมการเรียนรู้ที่มีประสิทธิภาพสูงที่สุดคือ Random Forest รองลงมาคือ Naïve Bayes สำหรับโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Availability User Reviews คืออัลกอริทึมการเรียนรู้ Random Forest ที่ถูกฝึกสอนด้วย TF-IDF, Stemming, n_tokens, Sentiment ร่วมกับเทคนิค SVMSMOTE และ GridSearchCV ซึ่งมีการปรับค่าพารามิเตอร์ของโมเดลการเรียนรู้ได้แก่ ค่า max_depth เท่ากับ 100 ค่า n_estimates เท่ากับ 1,000 และใช้ criterion คือ gini ได้รับค่า F1-Score อยู่ที่ 73.5% ค่า Precision อยู่ที่ 75.7% ค่า Recall อยู่ที่ 71.4% และค่า Accuracy อยู่ที่ 87.6% ซึ่งใช้เวลาฝึกสอนโดยประมาณอยู่ที่ 45.432 วินาที นอกจากนี้พบว่าการใช้เทคนิค MinMaxScaler ช่วยลดเวลาที่ใช้ฝึกสอนโมเดลและยังคงประสิทธิภาพการเรียนรู้ที่สูงของโมเดล ซึ่งใช้เวลาฝึกสอนอยู่ที่ 43.848 วินาทีน้อยกว่าประมาณเกือบ 2 วินาที

ตารางที่ 4-13 ประสิทธิภาพของโมเดลจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Availability User Reviews สูงสุด 3 อันดับแรกของแต่ละอัลกอริทึมการเรียนรู้

Configuration	Availability User Reviews				
	Precision	Recall	F1-Score	Accuracy	Fit Time (Sec)
Naïve Bayes + BoW	0.743	0.708	0.725	0.871	0.624
Naïve Bayes + BoW + Lemmatization	0.739	0.708	0.723	0.870	0.495
Naïve Bayes + BoW + n_tokens + Sentiment + POS + GridSearchCV	0.690	0.755	0.721	0.860	0.676
Logistic Regression + BoW + Stemming + n_tokens + Sentiment + POS	0.673	0.771	0.718	0.855	17.791
Logistic Regression + BoW + Stemming + n_tokens + Sentiment + SVMSMOTE + MinMaxScaler	0.670	0.771	0.717	0.853	32.840
Logistic Regression + TF-IDF + Stemming + n_tokens	0.667	0.771	0.715	0.852	16.655
Decision Trees + TF-IDF + Lemmatization + Sentiment + SVMSMOTE	0.624	0.667	0.645	0.823	4.755
Decision Trees + TF-IDF + Lemmatization + Sentiment + SVMSMOTE + MinMaxScaler	0.624	0.667	0.645	0.823	3.974
Decision Trees + TF-IDF + Lemmatization + SVMSMOTE + GridSearchCV	0.617	0.672	0.643	0.821	11.713
Random Forest + TF-IDF + Stemming + n_tokens + Sentiment + SVMSMOTE + GridSearchCV	**0.757**	**0.714**	**0.735**	**0.876**	**45.432**
Random Forest + TF-IDF + Stemming + n_tokens + Sentiment + SVMSMOTE + MinMaxScaler + GridSearchCV	**0.757**	**0.714**	**0.735**	**0.876**	**43.848**
Random Forest + TF-IDF + Lemmatization + n_tokens + Sentiment + SVMSMOTE + MinMaxScaler + GridSearchCV	0.760	0.677	0.716	0.871	29.219

จากตารางที่ 4-14 ประสิทธิภาพของโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Operability User Reviews พบว่า Textual Feature ทั้งหมดมีความสำคัญต่อการเรียนรู้ของโมเดลคือ BoW, TF-IDF, Doc2vec การทำ Stemming และ Lemmatization สนับสนุนประสิทธิภาพการเรียนรู้ให้กับโมเดล นอกจากนี้ User Review Metadata Feature ได้แก่ n_tokens, POS, Sentiment มีนัยสำคัญต่อประสิทธิภาพการเรียนรู้ของโมเดล เทคนิค SVMSMOTE, MinMaxScaler, GridSearchCV ช่วยเพิ่มประสิทธิภาพการเรียนรู้ให้แก่โมเดล ในขณะที่เทคนิค Chi-Square มีผลต่อประสิทธิภาพการเรียนรู้ในโมเดลอัลกอริทึมตระกูลต้นไม้คือ Decision Trees และ Random Forest โดยช่วงค่า F1-Score ของโมเดลการเรียนรู้ของเครื่องสูงสุด 3 อันดับแรกจากแต่ละอัลกอริทึมการเรียนรู้ทั้งหมดประมาณ 52.1% - 62.1% สำหรับโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Operability User Reviews คืออัลกอริทึมการเรียนรู้ Logistic Regression ที่ถูกฝึกสอนด้วย BoW, n_tokens, POS, Sentiment ร่วมกับเทคนิค SVMSMOTE และ GridSearchCV ซึ่งมีการปรับค่าพารามิเตอร์ของโมเดลการเรียนรู้ได้แก่ ค่า c เท่ากับ 10 และค่า penalty คือ L2 ได้รับค่า F1-Score อยู่ที่ 62.1% ค่า Precision อยู่ที่ 60.8% ค่า Recall อยู่ที่ 63.6% และค่า Accuracy อยู่ที่ 85.3% ซึ่งใช้เวลาฝึกสอนโดยประมาณอยู่ที่ 44.785 วินาที

วิทยานิพนธ์เลือกโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงที่สุดสำหรับใช้จำแนกบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews เพื่อนำไปใช้ในบทที่ 7 การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ไม่เคยเห็นมาก่อน

ตารางที่ 4-14 ประสิทธิภาพของโมเดลจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Operability User Reviews สูงสุด 3 อันดับแรกของแต่ละอัลกอริทึมการเรียนรู้

Configuration	Operability User Reviews				
	Precision	Recall	F1-Score	Accuracy	Fit Time (Sec)
Naïve Bayes + BoW + Stemming + n_tokens + GridSearchCV	0.617	0.576	0.596	0.852	1.511
Naïve Bayes + BoW + Stemming + POS + GridSearchCV	0.614	0.570	0.591	0.851	0.895
Naïve Bayes + BoW + Stemming + n_tokens + POS + GridSearchCV	0.614	0.570	0.591	0.851	0.947
Logistic Regression + BoW + n_tokens + Sentiment + POS + SVMSMOTE + GridSearchCV	**0.608**	**0.636**	**0.621**	**0.853**	**44.785**
Logistic Regression + TF-IDF + Sentiment + POS + SVMSMOTE	0.570	0.675	0.618	0.842	41.468
Logistic Regression + BoW + Lemmatization + n_tokens + Sentiment + POS + GridSearchCV	0.553	0.695	0.616	0.836	24.005
Decision Trees + TF-IDF + SVMSMOTE + GridSeachCV	0.558	0.576	0.567	0.833	10.780
Decision Trees + TF-IDF + SVMSMOTE + MinMaxScaler + GridSeachCV	0.558	0.576	0.567	0.833	12.703
Decision Trees + BoW + Stemming + MinMaxScaler + Chi-Square + GridSeachCV	0.513	0.530	0.521	0.816	0.933
Random Forest + Doc2vec + Lemmatization + Sentiment + POS + SVMSMOTE + MinMaxScaler + Chi-Square	0.592	0.556	0.573	0.843	2.549
Random Forest + Doc2vec + Lemmatization + Sentiment + POS + SVMSMOTE + GridSeachCV	0.603	0.543	0.571	0.846	4.464
Random Forest + Doc2vec + Lemmatization + Sentiment + POS + SVMSMOTE + MinMaxScaler + GridSeachCV	0.603	0.543	0.571	0.846	4.360

บทที่ 5
การตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน

เนื่องจากบทวิจารณ์ของผู้ใช้งานทั้งที่ได้รับจากผู้ใช้งานคนเดียวกันหรือคนละคนกัน มีโอกาสกล่าวถึงเรื่องเดียวกันได้ ตัวอย่างเช่น “you should add a like button for the comment section” และ “We need a like button for comments” ทั้งสองประโยคเป็นบทวิจารณ์ของผู้ใช้งานจากแอปพลิเคชัน Wattpad ซึ่งใจความสำคัญของบทวิจารณ์ของผู้ใช้งานกล่าวถึงเรื่องเดียวกัน วิทยานิพนธ์จึงพิจารณาว่าทั้งสองบทวิจารณ์ของผู้ใช้งานนั้น “ซ้ำซ้อนกัน” (Duplicate) และควรนำเพียง 1 บทวิจารณ์ของผู้ใช้งานมาสร้างประโยคความต้องการ ขั้นตอนของวิทยานิพนธ์ที่ใช้ตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกัน ประกอบด้วย 2 ขั้นตอน คือ 5.1) ขั้นตอนการตรวจสอบความซ้ำซ้อน และ 5.2) ขั้นตอนการประเมินประสิทธิภาพการตรวจสอบความซ้ำซ้อน

5.1 ขั้นตอนการตรวจสอบความซ้ำซ้อน

วิทยานิพนธ์นำบทวิจารณ์ของผู้ใช้งานโดยสุ่มเลือกบางส่วนจากข้อมูลที่ได้ติดป้ายแล้วมาใช้ในการทดลองการตรวจสอบความซ้ำซ้อน เรียกชุดข้อมูลนี้ว่า ชุดข้อมูลทดสอบความซ้ำซ้อน (Duplicate Test Set) ประกอบด้วยข้อมูล 1) Functional User Reviews จำนวน 127 บทวิจารณ์ของผู้ใช้งาน 2) Availability User Reviews จำนวน 41 บทวิจารณ์ของผู้ใช้งาน และ 3) Operability User Reviews จำนวน 118 บทวิจารณ์ของผู้ใช้งาน Duplicate Test Set มาจากแหล่งข้อมูลที่ 3 คือการทำ Screen Scraping จากทั้ง 8 แอปพลิเคชัน ได้แก่ Agoda, Airbnb, Instagram, Kindle, Messenger, TikTok, Wattpad, WhatsApp

ในการทดลองการตรวจสอบความซ้ำซ้อน วิทยานิพนธ์ทดลองตรวจสอบความซ้ำซ้อนกับข้อมูลบทวิจารณ์ของผู้ใช้งานภายในแอปพลิเคชันเดียวกันเท่านั้น วิทยานิพนธ์ทำความสะอาด Duplicate Test Set ตามขั้นตอนใน 4.2.2) Data Cleaning และรายการ Stopwords จะถูกลบทั้งหมด ไม่มีการยกเว้นคำใด เนื่องจากไม่มีนัยสำคัญสำหรับการตรวจสอบความซ้ำซ้อน แนวทางที่วิทยานิพนธ์นำมาใช้ทดลองตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานมีทั้งหมด 2 แนวทางคือ 5.1.1) Clustering และ 5.1.2) Pairwise Text Similarity มีรายละเอียดดังนี้

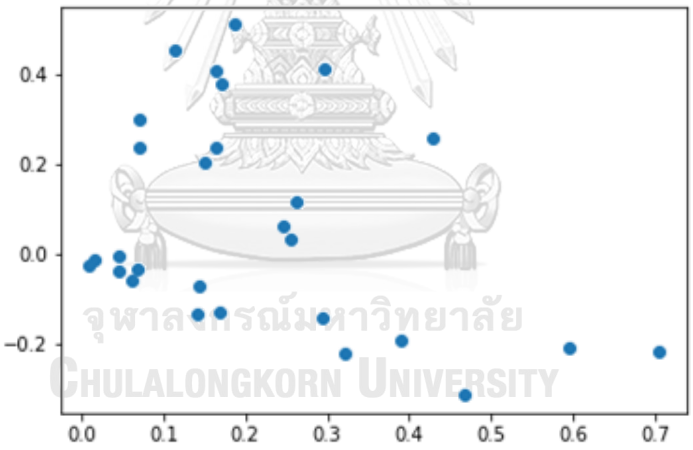
5.1.1 Clustering [14]

Clustering คืออัลกอริทึมการจัดกลุ่มข้อมูล โดยจัดข้อมูลออกเป็นกลุ่มข้อมูล (Cluster) ที่แตกต่างกัน ซึ่งข้อมูลภายในกลุ่มเดียวกันจะมีความคล้ายคลึงกันมากกว่าข้อมูลในกลุ่มอื่น ๆ จากแนวคิดนี้ บทวิจารณ์ของผู้ใช้งานที่อยู่ใน Cluster เดียวกันจะถือว่าเป็นข้อมูลที่ “Duplicate” กัน และตัวแทนของกลุ่มเท่านั้นจะถูกนำมาสร้างความต้องการซอฟต์แวร์ ในส่วนของการเตรียมข้อมูล

ก่อนการทดลอง Duplicate Test Set จะถูกทำ Representation ด้วย Lemmatization และ TF-IDF ก่อนนำไปทดลองกับอัลกอริทึมการจัดกลุ่มข้อมูล

อัลกอริทึม Clustering โดยทั่วไป จะร้องขอให้กำหนดตัววัดค่าความคล้ายคลึงกันของข้อมูล และจำนวนกลุ่มของข้อมูล (Number of Cluster) ที่ต้องการจัดกลุ่มก่อนการประมวลผล โดยอัลกอริทึมที่วิทยานิพนธ์นำมาทดลองได้แก่ 1) K-means จากไลบรารี NLTK [35] 2) K-medoids, 3) DBSCAN จากไลบรารี Scikit-learn [34] ทั้ง 3 อัลกอริทึมใช้ Cosine Distance เป็นตัววัดค่าความคล้ายคลึงกันระหว่างข้อมูล และ 4) Agglomerative จากไลบรารี Scipy [36] ใช้ตัววัดค่าความความคล้ายคลึงกันระหว่างข้อมูลด้วย Complete-Link และ Single-Link ในส่วนของเทคนิคที่นำมาใช้พิจารณาลักษณะของข้อมูลและจำนวนกลุ่มของข้อมูล ได้แก่

1) Principal Component Analysis (PCA) [32] คือวิธีการลดจำนวน Dimension ของข้อมูล วิทยานิพนธ์นำมาใช้วิเคราะห์ดูลักษณะการกระจายของข้อมูลบทวิจารณ์ของผู้ใช้งานประกอบการวิเคราะห์จำนวนกลุ่มของข้อมูล ตัวอย่างการกระจายของข้อมูล (Data Distribution) หลังจากทำ PCA ดังรูปที่ 5-1

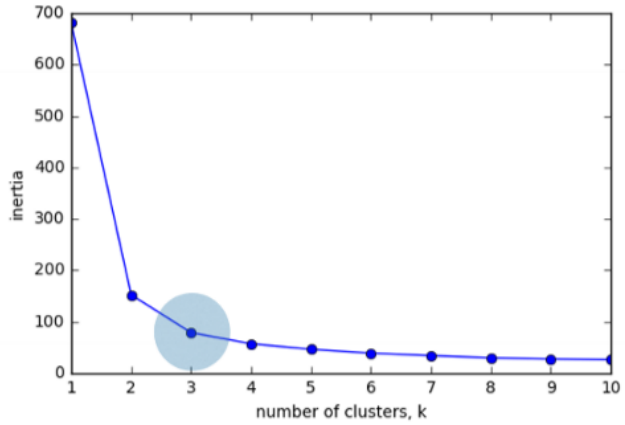


รูปที่ 5-1 Data Distribution ของข้อมูล Functional User Reviews จากแอปพลิเคชัน Wattpad

2) Hopkins Statistic [37] คือการคำนวณทางสถิติ นำมาใช้ประเมินความเป็นกลุ่มก้อนของข้อมูล เพื่อพิจารณาลักษณะการกระจายของข้อมูลว่าสามารถทำ Clustering ได้หรือไม่

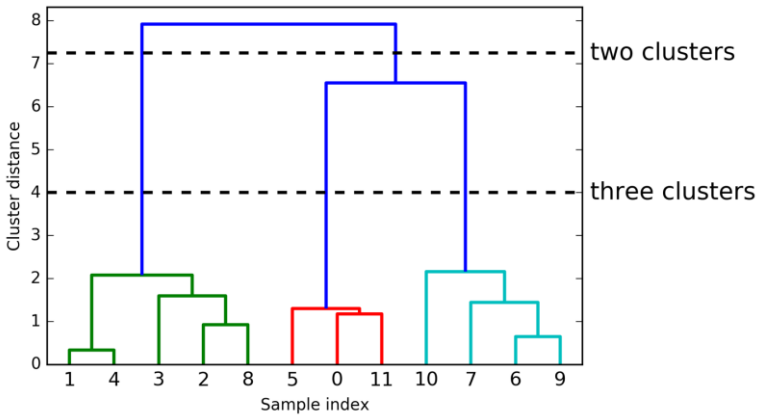
3) Elbow [38] คือการแสดงความแปรปรวนของข้อมูล ใช้พิจารณาจำนวนกลุ่มของข้อมูลสำหรับอัลกอริทึม K-means, K-medoids, DBSCAN ตัวอย่างการพิจารณาจำนวนกลุ่มดังรูปที่ 5-2 Elbow ระหว่างค่า Inertia และ Number of Cluster ซึ่งค่า Inertia แสดง

ถึงการกระจายภายในกลุ่มของข้อมูล มีค่าน้อยยิ่งดี หมายความว่าข้อมูลเกาะกลุ่มกัน เราพิจารณาจำนวนกลุ่มเมื่อค่า Inertia เริ่มคงที่ จากตัวอย่าง จำนวน Cluster ที่ได้คือ 3 กลุ่ม



รูปที่ 5-2 Elbow ระหว่างค่า Inertia และ Number of Cluster [38]

4) Dendrogram [32] คือแผนภาพแสดงลำดับการรวมกันของข้อมูลเป็นลำดับชั้น ดังรูปที่ 5-3 ลำดับชั้นล่างสุดคือระดับที่ข้อมูลใกล้กันมากที่สุด ใช้พิจารณาจำนวนกลุ่มของข้อมูลสำหรับอัลกอริทึม Agglomerative



รูปที่ 5-3 ตัวอย่าง Dendrogram [32]

5.1.2 Pairwise Text Similarity [39]

Pairwise Text Similarity คือแนวทางการวัดความคล้ายคลึงกันระหว่างคู่เอกสารหรือข้อความ วิทยานิพนธ์นำแนวทางมาประยุกต์ใช้พิจารณาความคล้ายคลึงกันระหว่างคู่ของบทวิจารณ์ของผู้ใช้งาน หากตรวจสอบแล้วพบว่าบทวิจารณ์ของผู้ใช้งานมีความคล้ายคลึงกันมากพอกับ

บทวิจารณ์ของผู้ใช้งานที่มีอยู่แล้วจะพิจารณาว่าซ้ำซ้อน และบทวิจารณ์ของผู้ใช้งานจะถูกละทิ้งหลังจากตรวจสอบความคล้ายคลึงกันระหว่างคู่ของบทวิจารณ์ของผู้ใช้งานทั้งหมดแล้ว บทวิจารณ์ของผู้ใช้งานที่แตกต่างกันจะถูกนำมาสร้างความต้องการซอฟต์แวร์ วิทยานิพนธ์พิจารณาเฉพาะบทวิจารณ์ของผู้ใช้งานภายในแอปพลิเคชันเดียวกันเท่านั้น มีรายละเอียดการดำเนินงานดังนี้

เริ่มจากวิทยานิพนธ์ทำ Representation ให้กับประโยคบทวิจารณ์ของผู้ใช้งานด้วย Lemmatization และโมเดล Pretrained SBERT โดยโมเดลที่ใช้ [19] คือ bert-large-nil-mean-token model (Bert-large-nil) ที่ 300 Dimension ซึ่งแนวทางของ SBERT สามารถเก็บรวบรวมบริบททางความหมายของคำในประโยคได้ดีกว่าการทำ Representation ด้วย TF-IDF เหมาะสำหรับการหาความคล้ายคลึงทางความหมาย (Semantic Text Similarity) เนื่องจากทางผู้พัฒนาโมเดล Pretrained SBERT ได้ปรับปรุงประสิทธิภาพของโมเดล Pretrained SBERTเพิ่มเติม ทำให้โมเดลมีประสิทธิภาพดีขึ้น วิทยานิพนธ์จึงได้นำโมเดลที่ได้รับการพัฒนาใหม่มาทดลองเพิ่มเติมภายหลังคือโมเดล paraphrase-MiniLM-L6-v2 (Paraphrase-mini)

วิทยานิพนธ์ใช้ Cosine Similarity วัดความคล้ายคลึงกันระหว่างบทวิจารณ์ของผู้ใช้งาน เพื่อพิจารณาระดับความคล้ายคลึงกัน (Similarity Threshold) ระหว่างคู่ของบทวิจารณ์ของผู้ใช้งานระดับเท่าไรถึงจะถือว่าคล้ายคลึงกันมากพอที่จะสามารถละทิ้งบทวิจารณ์ของผู้ใช้งานได้ วิทยานิพนธ์ได้ทดลองหา Similarity Threshold ระหว่างคู่ของบทวิจารณ์ของผู้ใช้งาน หากว่าคู่ของบทวิจารณ์ของผู้ใช้งานมีค่าความคล้ายคลึงกัน (Similarity) ไม่น้อยกว่า Similarity Threshold ที่กำหนดไว้ แสดงว่าบทวิจารณ์ของผู้ใช้งานทั้งสองคล้ายคลึงกันมากพอ และจะถือว่าบทวิจารณ์ของผู้ใช้งานคู่นั้นซ้ำซ้อนกัน วิทยานิพนธ์ได้นำช่วง Similarity Threshold จากงานวิจัย [40] มาประยุกต์ใช้เพื่อพิจารณา Similarity Threshold ระหว่างคู่ของบทวิจารณ์ของผู้ใช้งาน วิทยานิพนธ์ได้ทดลองด้วยค่า Similarity Threshold ที่หลากหลาย เพื่อให้ได้การตรวจสอบความซ้ำซ้อนที่มีประสิทธิภาพมากที่สุด ได้แก่ 0.60, 0.65, 0.70, 0.73, 0.75, 0.80

5.2 ขั้นตอนการประเมินประสิทธิภาพการตรวจสอบความซ้ำซ้อน

เพื่อประเมินประสิทธิภาพการทดลองแนวทางการตรวจสอบความซ้ำซ้อนแต่ละแนวทางของวิทยานิพนธ์ สำหรับแต่ละชุดข้อมูล Duplicate Test Set ของแต่ละประเภทบทวิจารณ์ (Functional, Availability และ Operability) วิทยานิพนธ์ได้สร้างชุดข้อมูลเฉลยจาก Duplicate Test Set เรียกว่า Truth Set มีทั้งหมด 2 ชุด มีรายละเอียดดังนี้

1) Truth Set ชุดแรกสำหรับประเมินประสิทธิภาพของอัลกอริทึม Clustering แต่ละประเภท วิทยานิพนธ์ทำการจัดกลุ่มให้กับบทวิจารณ์ของผู้ใช้งานใน Duplicate Test

Set ว่าบทวิจารณ์ใดอยู่ในคลัสเตอร์ใดร่วมกันบ้าง จากการพิจารณาการกล่าวถึงเนื้อหาเรื่องเดียวกัน

2) Truth Set ชุดสองสำหรับประเมินประสิทธิภาพแนวทาง Pairwise Text Similarity วิทยานิพนธ์ทำการพิจารณาแต่ละบทวิจารณ์ในชุดข้อมูล และติดป้ายบทวิจารณ์ของผู้ใช้งานเป็น “Duplicate” หากว่าบทวิจารณ์ของผู้ใช้งานนั้นซ้ำซ้อนกับบทวิจารณ์ใด ๆ ในชุดข้อมูล และ “Not Duplicate” หากว่าบทวิจารณ์ของผู้ใช้งานนั้นไม่ซ้ำซ้อนกับข้อมูลบทวิจารณ์ใด ๆ เลยในชุดข้อมูล

ผลลัพธ์ที่ได้จากแต่ละแนวทางการตรวจสอบความซ้ำซ้อนทั้งการทำ Clustering และ Pairwise Text Similarity จะถูกนำมาเทียบกับ Truth Set เพื่อคำนวณค่า Accuracy สำหรับวิเคราะห์และประเมินผลประสิทธิภาพความแม่นยำของแต่ละแนวทางการตรวจสอบความซ้ำซ้อนกัน และเพื่อเปรียบเทียบความแม่นยำระหว่างแต่ละ Similarity Threshold ของการทำ Pairwise Text Similarity ด้วย วิทยานิพนธ์คำนวณค่า Accuracy ด้วยไลบรารี metrics.accuracy_score ของ Scikit-learn [34] ผลลัพธ์ค่าเฉลี่ย Accuracy ของทั้ง 8 แอปพลิเคชันในแต่ละประเภทบทวิจารณ์ของผู้ใช้งานจากแต่ละอัลกอริทึม Clustering แสดงดังตารางที่ 5-1 และแนวทาง Pairwise Text Similarity ตามแต่ละ Similarity Threshold แสดงดังตารางที่ 5-2 และตารางที่ 5-3

ตารางที่ 5-1 ค่าเฉลี่ย Accuracy ของการตรวจสอบความซ้ำซ้อนในแต่ละประเภทบทวิจารณ์ของผู้ใช้งานจากแต่ละอัลกอริทึม Clustering

Clustering Algorithm	**Average Accuracy of Eight Applications**		
	Functional	**Availability**	**Operability**
K-means	0.399	0.746	0.556
K-medoids	**0.507**	0.709	0.571
DBSCAN	0.467	0.746	**0.611**
Agglomerative	0.426	**0.788**	0.583

จากตารางที่ 5-1 ค่าเฉลี่ย Accuracy จากการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานทั้ง 8 แอปพลิเคชันด้วย Clustering อัลกอริทึมต่าง ๆ สำหรับ Functional User Reviews อัลกอริทึมที่ดีที่สุดคือ K-medoids ด้วยค่าเฉลี่ย Accuracy ที่ 0.507 สำหรับ Availability User Reviews อัลกอริทึมที่ดีที่สุดคือ Agglomerative ด้วยค่าเฉลี่ย Accuracy ที่ 0.788 และ Operability User Reviews อัลกอริทึมที่ดีที่สุดคือ DBSCAN ด้วยค่าเฉลี่ย Accuracy ที่ 0.611 จากการทดลอง

Availability User Reviews ได้รับค่าเฉลี่ย Accuracy สูง ในขณะที่ Functional User Reviews และ Operability User Reviews อยู่ในระดับพอใช้ได้

วิทยานิพนธ์ได้วิเคราะห์ลักษณะของข้อมูลบทวิจารณ์ของผู้ใช้งานหลังการจัดกลุ่มด้วยอัลกอริทึม Clustering พบว่าบทวิจารณ์ของผู้ใช้งานภายในกลุ่มเดียวกันมีความคล้ายคลึงกันทางโครงสร้าง (Syntactic Similarity) มากกว่าคล้ายคลึงกันทางความหมาย (Semantic Similarity) ตัวอย่างเช่น การจัดกลุ่มบทวิจารณ์ของผู้ใช้งานจากแอปพลิเคชัน Messenger ด้วยอัลกอริทึม K-means, DBSCAN, Agglomerative ภายในกลุ่มเดียวกันประกอบด้วยบทวิจารณ์ของผู้ใช้งานดังนี้ 1) “please bring the reminder feature back”, 2) “please bring back the expanding or blown up images”, 3) “ could you please bring back the filter with the big mouth and big eyes and no nose” ซึ่งแต่ละประโยคบทวิจารณ์ของผู้ใช้งานประกอบด้วยคำเดียวกันแต่คนละความหมาย ปัจจัยส่วนหนึ่งที่ส่งผลต่อค่าเฉลี่ย Accuracy ของการทดลองมาจากการทำ Representation ของประโยคบทวิจารณ์ของผู้ใช้งานด้วย TF-IDF ซึ่งไม่สามารถรวบรวมความหมายของประโยคบทวิจารณ์ของผู้ใช้งานได้ นอกจากนี้ประสิทธิภาพของอัลกอริทึม Clustering สามารถขึ้นอยู่กับการพิจารณาเลือก Number of Cluster ด้วย

ตารางที่ 5-2 ค่าเฉลี่ย Accuracy ของการตรวจสอบความซ้ำซ้อนในแต่ละประเภทบทวิจารณ์ของผู้ใช้งานที่ถูกทำ Representation ด้วยโมเดล Bert-large-nil ตาม Similarity Threshold ระดับต่าง ๆ

Similarity Threshold	**Average Accuracy of Eight Applications (Bert-large-nil)**		
	Functional	**Availability**	**Operability**
0.60	0.60	0.65	**0.67**
0.65	0.60	0.71	0.65
0.70	0.69	**0.77**	0.65
0.73	**0.72**	0.77	0.64
0.75	0.71	0.77	0.58
0.80	0.70	0.76	0.47

ตารางที่ 5-3 ค่าเฉลี่ย Accuracy ของการตรวจสอบความซ้ำซ้อนในแต่ละประเภทบทวิจารณ์ของผู้ใช้งานที่ถูกทำ Representation ด้วยโมเดล Paraphrase-mini ตาม Similarity Threshold ระดับต่าง ๆ

Similarity Threshold	Average Accuracy of Eight Applications (Paraphrase-mini)		
	Functional	Availability	Operability
0.60	**0.69**	**0.85**	**0.63**
0.65	0.69	0.85	0.60
0.70	0.67	0.85	0.51
0.73	0.67	0.85	0.47
0.75	0.67	0.85	0.46
0.80	0.66	0.85	0.46

จากตารางที่ 5-2 และตารางที่ 5-3 ค่าเฉลี่ย Accuracy จากการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานทั้ง 8 แอปพลิเคชันด้วยแนวทาง Pairwise Text Similarity ข้อมูล Functional User Reviews ที่ถูกทำ Representation ด้วยโมเดล Bert-large-nil มีค่าเฉลี่ย Accuracy เพิ่มขึ้นตามระดับ Similarity Threshold และมีค่าเฉลี่ย Accuracy สูงสุดที่ Similarity Threshold เท่ากับ 0.73 ได้ค่าเฉลี่ย Accuracy สูงที่สุดคือ 0.72 จากนั้นค่าเฉลี่ย Accuracy ค่อย ๆ ลดลง ในขณะที่โมเดล Paraphrase-mini มีค่าเฉลี่ย Accuracy สูงสุดคือ 0.69 เมื่อ Similarity Threshold เท่ากับ 0.60 และยังคงที่เมื่อ Similarity Threshold เท่ากับ 0.65 หลังจากนั้นค่าเฉลี่ย Accuracy ค่อย ๆ ลดลง ซึ่งทั้งสองโมเดลได้รับค่าเฉลี่ย Accuracy ใกล้เคียงกัน

สำหรับข้อมูล Availability User Reviews โมเดล Bert-large-nil มีค่าเฉลี่ย Accuracy เพิ่มขึ้นตามระดับ Similarity Threshold และมีค่าเฉลี่ย Accuracy สูงสุดอยู่ที่ 0.77 เมื่อ Similarity Threshold เท่ากับ 0.70 และมีค่าเฉลี่ย Accuracy คงที่ถึง Similarity Threshold ที่ 0.75 และหลังจากนั้นค่าเฉลี่ย Accuracy ค่อย ๆ ลดลงตามระดับของ Similarity ในขณะที่โมเดล Paraphrase-mini ได้ค่าเฉลี่ย Accuracy สูงกว่าโมเดล Bert-large-nil โดยมีค่าเฉลี่ย Accuracy อยู่ที่ 0.85 เท่ากันในทุกระดับของ Similarity Threshold ที่นำมาทดสอบ

สำหรับข้อมูล Operability User Reviews ทั้งโมเดล Bert-large-nil และ Paraphrase-mini ที่ Similarity Threshold เท่ากับ 0.60 ได้ค่าเฉลี่ย Accuracy สูงที่สุดคือ 0.67 และ 0.63 ตามลำดับ ซึ่งทั้งสองโมเดลได้รับค่าเฉลี่ย Accuracy ใกล้เคียงกันและมีค่าเฉลี่ย Accuracy ค่อย ๆ ลดลงตามลำดับของ Similarity Threshold เช่นเดียวกัน

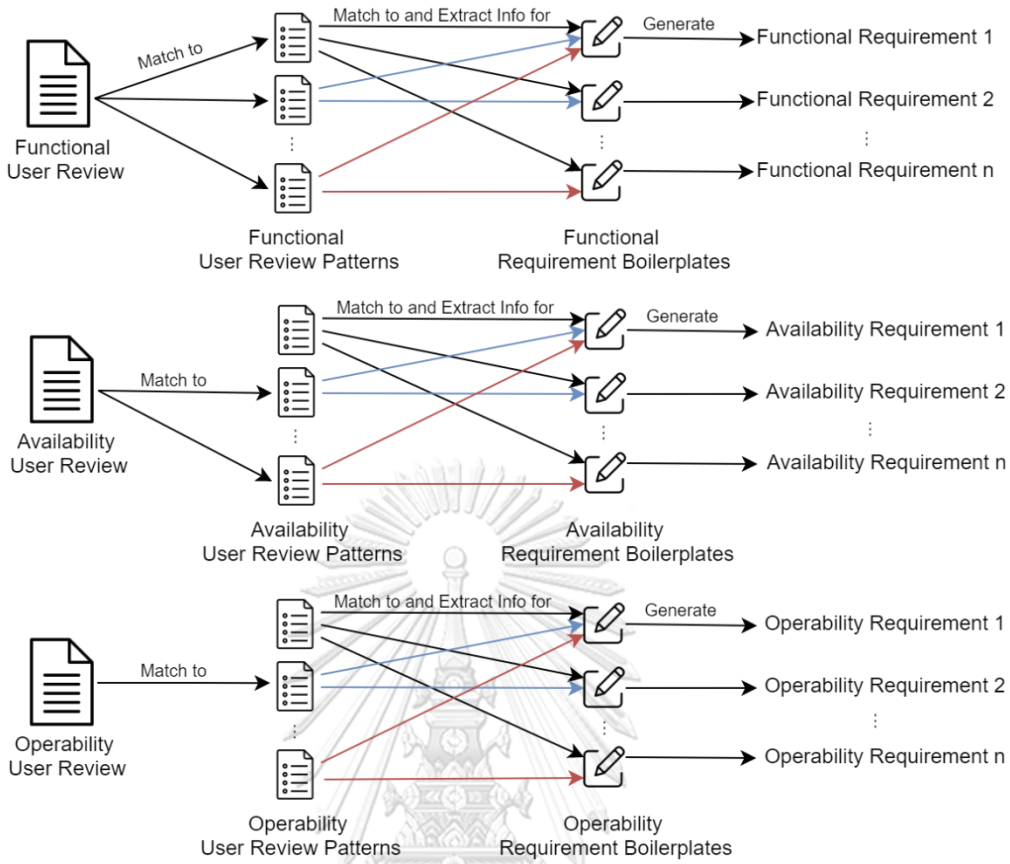
วิทยานิพนธ์ได้วิเคราะห์ลักษณะบทวิจารณ์ของผู้ใช้งานหลังจากตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกันด้วยแนวทาง Pairwise Text Similarity ปัจจัยที่ส่งผลต่อประสิทธิภาพการตรวจสอบความซ้ำซ้อนกันได้แก่ 1) ลักษณะประโยคบทวิจารณ์ของผู้ใช้งานต้องอาศัยการตีความใจความสำคัญของประโยค ตัวอย่างเช่น Functional User Reviews จากแอปพลิเคชัน IG “I wish there was an advanced setting that prevented your work from being downloaded” และ “Please add a way to protect your work or prevent people from downloading it” ทั้งสองประโยคมีใจความสำคัญเหมือนกันคือต้องการแนวทางที่จะป้องกันการดาวน์โหลดผลงาน แต่เนื่องจากมีรูปแบบการเขียนบทวิจารณ์ของผู้ใช้งานค่อนข้างแตกต่างกันมาก ซึ่งได้รับผลการตรวจสอบความซ้ำซ้อนกันว่าทั้งสองประโยคนั้น “not duplicate” ระหว่างกัน การทำ Representation ประโยคด้วยโมเดล Pretrained SBERT อาจไม่เพียงพอ นอกจากนี้ 2) การทำ Data Cleaning มีผลทำให้ข้อมูลใจความสำคัญบางส่วนขาดหายไปหรือผิดเพี้ยนไปจากเดิม ส่งผลต่อความหมายของประโยคบทวิจารณ์ของผู้ใช้งานได้ ตัวอย่างเช่น Operability User Reviews จากแอปพลิเคชัน Wattpad “But you get **added** after every two chapters” โดยอักษรตัวหนาในประโยคได้รับการเปลี่ยนคำจาก “ads” ซึ่งหมายถึงโฆษณา เป็นคำว่า “added” จากการแก้ไขข้อผิดพลาดของการสะกดคำและโครงสร้างไวยากรณ์ด้วยไลบรารี Gingerit

โมเดล Paraphrase-mini มีประสิทธิภาพในการทำ Representation ให้กับประโยคบทวิจารณ์ของผู้ใช้งานได้ดีกว่าโมเดล Bert-large-nil แต่เนื่องจากว่าลักษณะข้อมูลบทวิจารณ์ของผู้ใช้งานต้องอาศัยการตีความใจความสำคัญร่วมด้วย การทำเพียงแค่ Representation จึงไม่เพียงพอ เนื่องจากลักษณะของข้อมูล Functional User Reviews และ Operability User Reviews ส่วนใหญ่ต้องอาศัยการตีความใจความสำคัญ ในขณะที่ข้อมูล Availability User Reviews ค่อนข้างแตกต่างกันอย่างชัดเจน ดังนั้นจึงทำให้ค่าเฉลี่ย Accuracy ของข้อมูล Functional User Reviews และ Operability User Reviews จากทั้งสองโมเดลใกล้เคียงกันแม้ว่าโมเดล Paraphrase-mini จะมีประสิทธิภาพในการทำ Representation ที่ดีกว่าก็ตาม ในขณะที่ข้อมูล Availability User Reviews ที่ทำ Representation ด้วยโมเดล Paraphrase-mini ได้รับค่าเฉลี่ย Accuracy สูง

จากการทดลองตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานด้วย 2 แนวทาง แนวทาง Pairwise Text Similarity มีประสิทธิภาพดีกว่าแนวทาง Clustering เมื่อเทียบกันด้วยค่าเฉลี่ย Accuracy ในทุกประเภทของข้อมูลบทวิจารณ์ของผู้ใช้งาน ดังนั้นวิทยานิพนธ์เลือกแนวทาง Pairwise Text Similarity เพื่อนำไปใช้ในบทที่ 7 สำหรับ Functional User Reviews และ Operability User Reviews จะถูกทำ Representation ด้วยโมเดล Bert-large-nil และมี Similarity Threshold ที่ 0.73 และ 0.60 ตามลำดับ สำหรับ Availability User Reviews จะถูกทำ Representation ด้วยโมเดล Paraphrase-mini และมี Similarity Threshold ที่ 0.60

บทที่ 6
การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ถูกจำแนกประเภทแล้ว

หลังจากที่ผู้วิจัยได้ทำการทดลองถึงแนวทางการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกันแล้วในบทที่ 5 ลำดับถัดมาในบทที่ 6 นี้จะอธิบายถึงรายละเอียดการทดลองสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ถูกจำแนกประเภทแล้ว โดยผู้วิจัยมีแนวคิดของกระบวนการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งาน ดังรูปที่ 6-1 หลังจากบทวิจารณ์ของผู้ใช้งานได้ถูกจำแนกออกเป็นแต่ละประเภทแล้ว ได้แก่ Functional User Reviews, Availability User Reviews, Operability User Reviews ผู้วิจัยจับคู่บทวิจารณ์ของผู้ใช้งานแต่ละบทวิจารณ์กับ User Review Patterns ของแต่ละประเภทคือ Functional User Review จะถูกจับคู่กับ Functional User Review Patterns, Availability User Review จะถูกจับคู่กับ Availability User Review Patterns, Operability User Review จะถูกจับคู่กับ Operability User Review Patterns โดยที่แต่ละประโยคบทวิจารณ์ของผู้ใช้งานจะสามารถจับคู่ได้กับอย่างน้อยหนึ่ง User Review Pattern หากว่าบทวิจารณ์ของผู้ใช้งานใดได้รับการจับคู่กับ User Review Patterns ที่กำหนดไว้ ผู้วิจัยจะนำบทวิจารณ์ของผู้ใช้งานที่ได้รับการจับคู่นั้นมาสกัดข้อมูลที่สำคัญต่อการสร้างความต้องการซอฟต์แวร์ และสร้างประโยคความต้องการซอฟต์แวร์โดยใช้ Requirement Boilerplate ที่ถูกจับคู่ไว้แล้วกับ User Review Patterns นั้น ๆ ซึ่งแต่ละ User Review Pattern จะถูกจับคู่ไว้กับอย่างน้อยหนึ่ง Requirement Boilerplate ผลลัพธ์สุดท้ายคือหนึ่ง Requirement Boilerplate จะสามารถสร้างหนึ่งประโยคของความต้องการเชิงฟังก์ชันหรือที่ไม่ใช่เชิงฟังก์ชันได้ ในการทดลองนี้ไม่ได้นำข้อมูลบทวิจารณ์ของผู้ใช้งานที่ไม่สามารถจับคู่กับ User Review Patterns ใด ๆ ได้มาสร้างความต้องการซอฟต์แวร์



รูปที่ 6-1 กระบวนการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากบทวิจารณ์ของผู้ใช้งานที่ถูกจำแนกประเภทแล้ว

สำหรับข้อมูลที่ใช้ในการทดลอง เพื่อให้มีรูปแบบการเขียนประโยคของบทวิจารณ์ของผู้ใช้งานที่หลากหลายและมากพอเพื่อทดลอง วิทยานิพนธ์ได้เก็บรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้งานเพิ่มด้วยวิธี Screen Scraping ซึ่งมีจำนวนบทวิจารณ์ของผู้ใช้ที่นำมาทดลองประกอบด้วย บทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews จำนวน 3,475, 1,589, 2,969 บทวิจารณ์ของผู้ใช้งานตามลำดับ วิทยานิพนธ์ทำความสะอาดข้อมูลบทวิจารณ์ของผู้ใช้งานที่ได้รวบรวมมาตามขั้นตอนใน 4.2.2) Data Cleaning ยกเว้นการลบ Punctuation และรายการ Stopwords เนื่องจากมีนัยสำคัญกับรูปประโยคของบทวิจารณ์ของผู้ใช้งาน ซึ่งนำมาใช้สำหรับการสร้างความต้องการ

บทที่ 6 นี้ประกอบด้วย 4 ขั้นตอนหลักคือ 1) การจับคู่แบบรูปประโยค (Sentence Pattern Matching) 2) การสกัดข้อมูลสำหรับผู้กระทำ กิจกรรม เป้าหมาย และแนวทาง (Information Extraction for Agent, Activity, Target, and Way) 3) การสร้างความต้องการ (Requirement Generation) และ 4) การประเมินคุณภาพของความต้องการ (Evaluation of Requirement Quality) รายละเอียดของแต่ละขั้นตอนอธิบายไว้ในหัวข้อที่ 6.1 ถึง 6.4 ตามลำดับ

6.1 Sentence Pattern Matching

เนื่องจากบทวิจารณ์ของผู้ใช้งานมีรูปแบบการเขียนที่หลากหลาย วิทยานิพนธ์จึงระบุแบบรูปการเขียนบทวิจารณ์ของผู้ใช้งานก่อน จากงานวิจัย [1] ได้วิเคราะห์โครงสร้างของประโยคบทวิจารณ์ของผู้ใช้งานทั้งหมด 500 บทวิจารณ์ของผู้ใช้งาน พบ Recurrent Linguistic Patterns ทั้งหมด 246 แบบรูปของประโยคบทวิจารณ์ของผู้ใช้งาน วิทยานิพนธ์นำ Recurrent Linguistic Patterns มาปรับปรุง แก้ไขเพื่อให้ได้ User Review Pattern ที่สอดคล้องกับลักษณะรูปแบบการเขียนบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews โดยนำเสนอในรูปแบบของไวยากรณ์ Extended Backus-Naur (EBNF Syntax) ดังตารางที่ 6-1 ถึงตารางที่ 6-3 ได้แก่ Functional User Review Patterns จำนวน 84 User Review Patterns, Availability User Review Patterns จำนวน 37 User Review Patterns, Operability User Review Patterns จำนวน 86 User Review Patterns

ทั้งนี้ระหว่าง Functional User Review Patterns, Availability User Review Patterns และ Operability User Review Patterns จะมีบางแบบรูปเหมือนกัน จากการวิเคราะห์ข้อมูลบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ทั้งหมด 2,610 ประโยคบทวิจารณ์ของผู้ใช้งาน พบว่าประโยคบทวิจารณ์ของผู้ใช้งานแต่ละประเภทมีโอกาสมีรูปแบบการเขียนที่เหมือนกัน แต่มีบริบททางความหมายของประโยคแตกต่างกันได้ ตัวอย่างเช่น

Operability User Reviews: “you **cannot edit an individual picture** after you post it very inconvenient.” และ

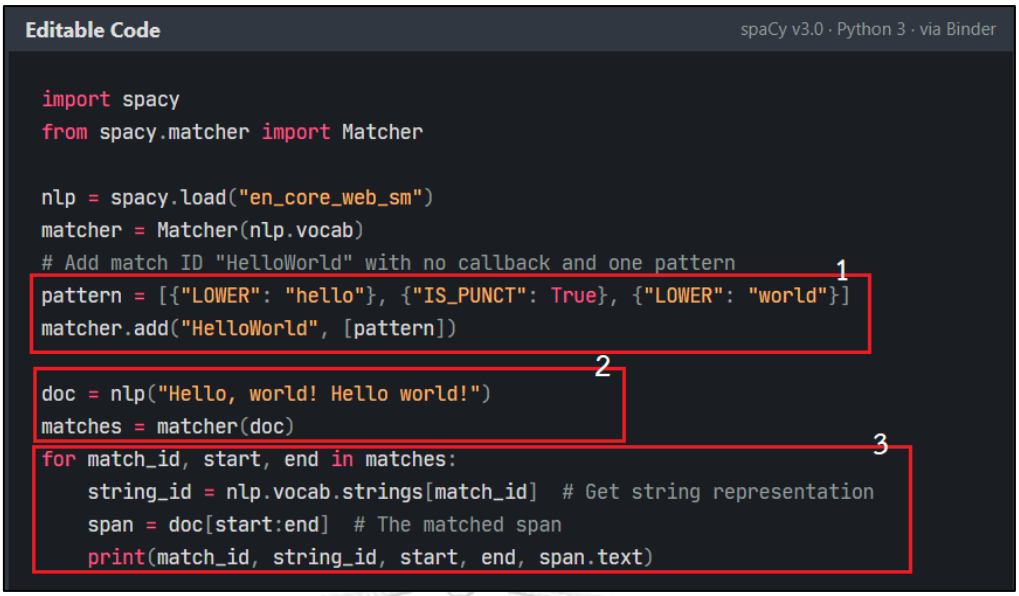
Availability User Reviews: “**cannot change my payment method** because application keeps crashing.”

ทั้งสองประโยคบทวิจารณ์ของผู้ใช้งานมีรูปแบบการเขียนที่เหมือนกันดังตัวอักษรหนาในประโยคตัวอย่างคือรูปแบบการเขียน (“could” | “would” | “can”) “not” VERB *something*; แต่มีบริบททางความหมายของประโยคแตกต่างกัน โดยประโยค Operability User Reviews แสดงถึงความไม่สะดวกในการใช้งานและประโยค Availability User Reviews แสดงถึงความไม่สามารถใช้งานได้เนื่องจากแอปพลิเคชันมีปัญหา

ภายใน User Review Pattern ประกอบด้วย 3 Non-terminals ได้แก่ someone, something, somehow เนื่องจากงานวิจัย [1] ไม่ได้ระบุถึงโครงสร้างไวยากรณ์ทางภาษาที่ชัดเจนของ Non-terminals เหล่านี้ วิทยานิพนธ์จึงได้กำหนดโครงสร้างไวยากรณ์ทางภาษาระดับพื้นฐานเพิ่มเติมขึ้นมา โครงสร้างไวยากรณ์ทางภาษาของ Non-terminals ในรูปของ EBNF Syntax ดังตารางที่ 6-4 ซึ่งมีรายละเอียดของแต่ละ Non-terminals ดังนี้

1) someone หมายถึงวิสามานยนาม (Proper Nouns) หรือบุรุษสรรพนาม (Personal Pronouns)
2) something หมายถึงนามวลี (Noun Phrase) ซึ่ง Noun สามารถประกอบด้วยตัวดัดแปลงทั้งพรีโมดิฟายเออร์ (Premodifiers) และโพสต์โมดิฟายเออร์ (Postmodifiers) โดยที่ Premodifier สามารถเป็นคำบ่งชี้ (Determiner), จำนวน (Number), หรือ คำคุณศัพท์ (Adjective) ส่วน Postmodifier สามารถเป็นบุพบทวลี (Prepositional Phrase) หรือกริยาไม่แท้ประเภท To-infinitive
3) somehow หมายถึงประโยคที่ประกอบด้วยประธานและกริยา หรือ ประธาน กริยา กรรมตรง

แต่ละบทวิจารณ์ของผู้ใช้งานมีโอกาสได้รับการจับคู่กับ User Review Pattern ได้มากกว่า 1 User Review Pattern วิทยานิพนธ์จับคู่ระหว่างบทวิจารณ์ของผู้ใช้งานกับ User Review Patterns ด้วยวิธี Token-Based Matching โดยใช้ไลบรารี spaCy [41] ตัวอย่างรหัสต้นฉบับ (Source Code) การใช้งานไลบรารี spaCy ด้วยวิธี Token-Based Matching จากเว็บไซต์ spaCy.io ดังรูปที่ 6-2 ในกรอบหมายเลขที่ 1 มีตัวแปรสำคัญคือ pattern ไว้กำหนดแบบรูปของข้อมูลที่ต้องการจับคู่ จากตัวอย่างแบบรูปจะประกอบด้วยคำว่า “hello” ตามด้วย punctuation และ “world” เราสามารถกำหนดชื่อของ pattern ด้วยคำสั่ง matcher.add ซึ่งประกอบด้วยค่าพารามิเตอร์สองค่าคือ ชื่อของ pattern และแบบรูปของข้อมูลที่เรากำหนดไว้ ถัดมาในกรอบหมายเลขที่ 2 คำสั่ง nlp เป็นคำสั่งรับข้อมูลอินพุตซึ่งคือข้อมูลที่เราต้องการจับคู่กับแบบรูปของข้อมูลที่กำหนดไว้ และคำสั่ง matcher คือคำสั่งให้โปรแกรมเริ่มประมวลผลการจับคู่ข้อมูลอินพุตกับแบบรูปที่กำหนดไว้ และสุดท้ายในกรอบหมายเลขที่ 3 หากว่าข้อมูลอินพุตได้รับการจับคู่กับแบบรูปที่กำหนดไว้ ข้อมูลที่ตรงกับแบบรูปจะได้รับการสกัดข้อมูล ดังนั้นผลลัพธ์ที่ได้จาก source code ของโปรแกรมในรูปที่ 6-2 คือข้อมูล “Hello, world”



รูปที่ 6-2 ตัวอย่าง Source Code การทำ Token-Based Matching โดยใช้ไลบรารี spaCy [41]

วิทยานิพนธ์จับคู่ระหว่างบทวิจารณ์ของผู้ใช้งานกับ User Review Patterns โดยจับคู่ Part of Speech และคำสำคัญ (Key words) ตามที่ระบุใน User Review Patterns นอกจากนี้ ตำแหน่งที่เป็น Verb to be ภายใน User Review Pattern และบทวิจารณ์ของผู้ใช้งานจะถูกจับคู่ผ่านการทำ Lemmatization โดยจะคืนรากศัพท์ให้กับ Verb to be เพื่อเพิ่มความยืดหยุ่นให้กับการจับคู่ของข้อมูล ในตำแหน่งของ MODAL ภายใน User Review Pattern วิทยานิพนธ์ตีความหมายว่าเป็นคำขยายของประโยคจึงแทนที่ Part of Speech ด้วย Adjective, Adverb, หรือ Adverb ตามด้วย Adjective

ยกตัวอย่างการจับคู่ระหว่างบทวิจารณ์ของผู้ใช้งานกับ User Review Patterns ดังประโยคตัวอย่างด้านล่าง Functional User Reviews จะถูกจับคู่กับ Functional User Review Patterns รหัส RPFN01

Functional User Reviews: “you should add a like button in the comment section”

Functional User Review Patterns: *rpfn-01* ⟵ *someone* (“could” | “should”) (“add” | “provide” | “offer” | “integrate”) *something*;

ในตัวอย่างนี้ “you” จะถูกจับคู่กับ *someone* และ “a like button in the comment section” จะถูกจับคู่กับ *something*

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
1.	RPFN01	*rpfn-01* ⟵ *someone* (“could” | “should”) (“add” | “provide” | “offer” | “integrate”)[Activity] *something*[Target];	RBFN01, RBFN05, RBFN06
2.	RPFN02	*rpfn-02* ⟵ *someone* “would love to” (“see” | “use”[Activity]) *something*[Target];	RBFN04, RBFN05
3.	RPFN03	*rpfn-03* ⟵ *someone* “thinks” *something*[Agent] “would be” MODAL “to have[Activity]” *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05
4.	RPFN04	*rpfn-04* ⟵ *someone* “thinks” *something*[Target] “should” VERB[Activity];	RBFN01, RBFN06
5.	RPFN05	*rpfn-05* ⟵ “Let us implement” *something*[Target];	RBFN01, RBFN04, RBFN05
6.	RPFN06	*rpfn-06* ⟵ *something*[Target] “needs to be” VERB[Activity];	RBFN01, RBFN06
7.	RPFN07	*rpfn-07* ⟵ “All that is needed is” *something*[Target];	RBFN01, RBFN04, RBFN05
8.	RPFN08	*rpfn-08* ⟵ *something*[Target] “is what” *someone* “needs”;	RBFN01, RBFN04, RBFN05
9.	RPFN09	*rpfn-09* ⟵ “Could we add[Activity]” *something*[Target] “?”;	RBFN01, RBFN05, RBFN06
10.	RPFN10	*rpfn-10* ⟵ *someone* “thinks” *something*[Target] “is” MODAL “to” VERB[Activity];	RBFN01, RBFN06
11.	RPFN11	*rpfn-11* ⟵ “Why do not change[Activity]” *something*[Target] “?”;	RBFN01, RBFN06

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
12.	RPFN12	*rpfn-12* ⟵ “it would be” MODAL “to” VERB[Activity];	RBFN01, RBFN04, RBFN06
13.	RPFN13	*rpfn-13* ⟵ *someone* “needs to create[Activity]” *something*[Target];	RBFN01, RBFN04, RBFN05, RBFN06
14.	RPFN14	*rpfn-14* ⟵ *something*[Target] “is required”;	RBFN01, RBFN04, RBFN05
15.	RPFN15	*rpfn-15* ⟵ *something*[Agent] “could have” *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05
16.	RPFN16	*rpfn-16* ⟵ “It would be” (“possible” | “beneficial”) “to have” *something*[Target];	RBFN01, RBFN04, RBFN05
17.	RPFN17	*rpfn-17* ⟵ “Please” (“remove” | “check”)[Activity] *something*[Target];	RBFN01, RBFN06
18.	RPFN18	*rpfn-18* ⟵ *something*[Agent] “requires” *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05
19.	RPFN19	*rpfn-19* ⟵ *something*[Target] “should be treated” (“in” | “as”) *something*[Agent];	RBFN01, RBFN02, RBFN04, RBFN05
20.	RPFN20	*rpfn-20* ⟵ *someone* “thinks” *something*[Target] “needs” VERB[Activity];	RBFN01, RBFN06

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
21.	RPFN21	*rpfn-21* ⟵ *something*[Agent] (“should” | “could”) (“define” | “support”)[Activity] *something*[Target];	RBFN01, RBFN05, RBFN06
22.	RPFN22	*rpfn-22* ⟵ *something*[Target] (“should” | “could”) “be” VERB[Activity];	RBFN01, RBFN06
23.	RPFN23	*rpfn-23* ⟵ “adding”[Activity] *something*[Target];	RBFN01, RBFN05, RBFN06
24.	RPFN24	*rpfn-24* ⟵ “All” *someone* “needs is” *something*[Target];	RBFN01, RBFN04, RBFN05
25.	RPFN25	*rpfn-25* ⟵ *someone* “requests” *something*[Target];	RBFN01, RBFN04, RBFN05
26.	RPFN26	*rpfn-26* ⟵ *something*[Agent] (“should” | “could”) “include” *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05
27.	RPFN27	*rpfn-27* ⟵ *something*[Target] “ought to be” *somehow*[Way];	RBFN03
28.	RPFN28	*rpfn-28* ⟵ *someone* (“wants” | “likes”) “to have” *something*[Target];	RBFN01, RBFN04, RBFN05
29.	RPFN29	*rpfn-29* ⟵ *something*[Agent] “needs” *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05
30.	RPFN30	*rpfn-30* ⟵ *something*[Target] “is missing”;	RBFN01, RBFN04, RBFN05
31.	RPFN31	*rpfn-31* ⟵ *something*[Target] “is not defined[Activity]”;	RBFN01, RBFN05, RBFN06
32.	RPFN32	*rpfn-32* ⟵ *something*[Target] “would be a good” (“idea” | “thing”);	RBFN01, RBFN04, RBFN05

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
33.	RPFN33	*rpfn-33* ⟵ *something*[Agent] (“can” | “could”) “use”[Activity] *something*[Target];	RBFN01, RBFN02, RBFN04, RBFN05, RBFN06
34.	RPFN34	*rpfn-34* ⟵ *someone* “is” (“thinking” | “planning”) “of adding[Activity]” *something*[Target];	RBFN01, RBFN05, RBFN06
35.	RPFN35	*rpfn-35* ⟵ “A new” *something*[Target] “can be made”;	RBFN01, RBFN04, RBFN05
36.	RPFN36	*rpfn-36* ⟵ “could allow” *something*[Target] “to” VERB[Activity];	RBFN01, RBFN06
37.	RPFN37	*rpfn-37* ⟵ “I propose” *something*[Target];	RBFN01, RBFN04, RBFN05
38.	RPFN38	*rpfn-38* ⟵ *something*[Target] (“could” | “might”) “be not a bad” (“idea” | “thing”);	RBFN01, RBFN04, RBFN05
39.	RPFN39	*rpfn-39* ⟵ *something*[Target] “would allow to” VERB[Activity];	RBFN01, RBFN06
40.	RPFN40	*rpfn-40* ⟵ *someone* (“should” | “could”) “modify”[Activity] *something*[Target];	RBFN01, RBFN04, RBFN06
41.	RPFN41	*rpfn-41* ⟵ *something*[Agent] (“should” | “could”) “hold”[Activity] *something*[Target];	RBFN01, RBFN06
42.	RPFN42	*rpfn-42* ⟵ “New changes include” *something*[Target];	RBFN01, RBFN04, RBFN05
43.	RPFN43	*rpfn-43* ⟵ (“Plan” | “idea” | “aim”) “is” *something*[Target];	RBFN01, RBFN04, RBFN05

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
44.	RPFN44	*rpfn-44* ⟵ *someone* “recommends” *something*[Target];	RBFN01, RBFN04, RBFN05
45.	RPFN45	*rpfn-45* ⟵ *something*[Agent] (“provides” | “supports”)[Activity] *something*[Target];	RBFN01, RBFN05, RBFN06
46.	RPFN46	*rpfn-46* ⟵ *someone* “should create[Activity]” *something*[Target];	RBFN01, RBFN04, RBFN05, RBFN06
47.	RPFN47	*rpfn-47* ⟵ *something*[Target] “is removed”;	RBFN01, RBFN04, RBFN05
48.	RPFN48	*rpfn-48* ⟵ *someone* “is looking” (“to” | “for”) VERB[Activity];	RBFN01, RBFN04, RBFN06
49.	RPFN49	*rpfn-49* ⟵ *someone* “expects” *something*[Target];	RBFN01, RBFN04, RBFN05
50.	RPFN50	*rpfn-50* ⟵ *something*[Target] “should be fixed”;	RBFN07
51.	RPFN51	*rpfn-51* ⟵ “Can” *something*[Agent] “fix” *something*[Target] “?”;	RBFN08
52.	RPFN52	*rpfn-52* ⟵ “An issue to fix is” *something*[Target];	RBFN07
53.	RPFN53	*rpfn-53* ⟵ “It is a” (“mistake” | “problem”) (“at” | “of” | “with”) *something*[Target];	RBFN07
54.	RPFN54	*rpfn-54* ⟵ *something*[Target] “is unable to” VERB[Activity];	RBFN09
55.	RPFN55	*rpfn-55* ⟵ *something*[Agent] “not do” *something*[Target];	RBFN01, RBFN04, RBFN05

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
56.	RPFN56	*rpfn-56* ⟵ *something*[Target] “is not able to” VERB[Activity];	RBFN09
57.	RPFN57	*rpfn-57* ⟵ “problem related to” *something*[Target];	RBFN07
58.	RPFN58	*rpfn-58* ⟵ *something*[Target] “cannot be found”;	RBFN01, RBFN04, RBFN05
59.	RPFN59	*rpfn-59* ⟵ “Please” (“adjust” | “fix”)[Activity] *something*[Target];	RBFN01, RBFN06
60.	RPFN60	*rpfn-60* ⟵ *someone* “suggests” *something*[Target];	RBFN01, RBFN04, RBFN05
61.	RPFN61	*rpfn-61* ⟵ *something*[Agent] (“should” | “could” | “can” | “must”) “solve” *something*[Target];	RBFN08
62.	RPFN62	*rpfn-62* ⟵ *someone* (“could” | “can”) (“run” | “prepare”)[Activity] *something*[Target];	RBFN01, RBFN04, RBFN06
63.	RPFN63	*rpfn-63* ⟵ *someone* “thinks” *something*[Target] “can help”;	RBFN01, RBFN04, RBFN05
64.	RPFN64	*rpfn-64* ⟵ *something*[Agent] “resolves” *something*[Target];	RBFN08
65.	RPFN65	*rpfn-65* ⟵ *something*[Target] “could be done”;	RBFN01, RBFN04, RBFN05
66.	RPFN66	*rpfn-66* ⟵ “You need to do” *something*[Target];	RBFN01, RBFN04, RBFN05

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
67.	RPFN67	*rpfn-67* ⟵ “goal is to provide[Activity]” *something*[Target];	RBFN01, RBFN04, RBFN05, RBFN06
68.	RPFN68	*rpfn-68* ⟵ “solution consists” (“of” | “in”) *something*[Target];	RBFN01, RBFN04, RBFN05
69.	RPFN69	*rpfn-69* ⟵ “solution is” *something*[Target];	RBFN01, RBFN04, RBFN05
70.	RPFN70	*rpfn-70* ⟵ “a way is” *something*[Target];	RBFN01, RBFN04, RBFN05
71.	RPFN71	*rpfn-71* ⟵ *someone* (“should” | “could”) “recommend[Activity]” *something*[Target];	RBFN01, RBFN04, RBFN06
72.	RPFN72	*rpfn-72* ⟵ *someone* (“should” | “could”) “work on” something[Target];	RBFN01, RBFN04, RBFN05
73.	RPFN73	*rpfn-73* ⟵ “I am interested to contribute to” *something*[Target];	RBFN01, RBFN04, RBFN05
74.	RPFN74	*rpfn-74* ⟵ *someone* “wants to know” *something*[Target];	RBFN05, RBFN10
75.	RPFN75	*rpfn-75* ⟵ (“Is” | “Are”) “there” *something*[Target] “?”;	RBFN01, RBFN04, RBFN05
76.	RPFN76	*rpfn-76* ⟵ “Where is” *something*[Target] [“?”];	RBFN01, RBFN04, RBFN05
77.	RPFN77	*rpfn-77* ⟵ “would like to know” *something*[Target];	RBFN05, RBFN10
78.	RPFN78	*rpfn-78* ⟵ “Does” *something*[Agent] “provide[Activity]” *something*[Target] “?”;	RBFN01, RBFN05, RBFN06

ตารางที่ 6-1 User Review Patterns ที่สอดคล้องกับลักษณะของ Functional User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Functional User Review Pattern	Requirement Boilerplate Code
79.	RPFN79	*rpfn-79* ⟵ “Can” *someone* (“get” | “do” | “use”)[Activity] *something*[Target] “?”;	RBFN01, RBFN04, RBFN05, RBFN06
80.	RPFN80	*rpfn-80* ⟵ “Do you have” *something*[Target] “?”;	RBFN01, RBFN04, RBFN05
81.	RPFN81	*rpfn-81* ⟵ “Can I find[Activity]” *something*[Target] “?”;	RBFN01, RBFN04, RBFN05, RBFN06
82.	RPFN82	*rpfn-82* ⟵ “Please share[Activity]” *something*[Target];	RBFN01, RBFN06
83.	RPFN83	*rpfn-83* ⟵ (“could” | “would” | “can”) “not” VERB[Activity] *something*[Target];	RBFN01, RBFN04, RBFN06
84.	RPFN84	*rpfn-84* ⟵ “Is” *something*[Target] “possible?”;	RBFN01, RBFN04, RBFN05

ตารางที่ 6-2 User Review Patterns ที่สอดคล้องกับลักษณะของ Availability User Reviews ในรูปของ EBNF Syntax

No.	Review Pattern Code	Availability User Review Pattern	Requirement Boilerplate Code
1.	RPA01	*rpa-01* ⟵ *someone* “expects” *something*[Target] “to work”;	RBA02, RBA04
2.	RPA02	*rpa-02* ⟵ *something*[Target] “should be fixed”;	RBA02
3.	RPA03	*rpa-03* ⟵ “Can” *something*[Agent] “fix” *something*[Target] “?”;	RBA06
4.	RPA04	*rpa-04* ⟵ *someone* “has tried” to VERB[Activity] “but” SUBJECT VERB;	RBA01
5.	RPA05	*rpa-05* ⟵ *something*[Target] “goes wobbly”;	RBA02
6.	RPA06	*rpa-06* ⟵ *something*[Target] “seems to be an issue”;	RBA02
7.	RPA07	*rpa-07* ⟵ “An issue to fix is” *something*[Target];	RBA02
8.	RPA08	*rpa-08* ⟵ “It is a” (“mistake” | “problem”) (“at” | “of” | “with”) *something*[Target];	RBA02
9.	RPA09	*rpa-09* ⟵ *something*[Target] “is missing”;	RBA04
10.	RPA10	*rpa-10* ⟵ *something*[Target] “crashes”;	RBA02, RBA04
11.	RPA11	*rpa-11* ⟵ *something*[Target] (“demonstrates” | “represents”) “the problem”;	RBA02
12.	RPA12	*rpa-12* ⟵ *something*[Target] “is unable to” VERB[Activity];	RBA03

ตารางที่ 6-2 User Review Patterns ที่สอดคล้องกับลักษณะของ Availability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Availability User Review Pattern	Requirement Boilerplate Code
13.	RPA13	*rpa-13* ⟵ *something*[Target] “fails”;	RBA02, RBA04
14.	RPA14	*rpa-14* ⟵ *something*[Target] “does not work”;	RBA02, RBA04
15.	RPA15	*rpa-15* ⟵ *something*[Agent] “not do” *something*[Target];	RBA06
16.	RPA16	*rpa-16* ⟵ *something* “works fine but not” *something*[Target];	RBA02, RBA04
17.	RPA17	*rpa-17* ⟵ “Error:” *something*[Target];	RBA02
18.	RPA18	*rpa-18* ⟵ *something*[Target] “is not able to” VERB[Activity];	RBA03
19.	RPA19	*rpa-19* ⟵ *something*[Target] “cannot be found”;	RBA02, RBA04
20.	RPA20	*rpa-20* ⟵ “Please” (“adjust” | “fix”) *something*[Target];	RBA02
21.	RPA21	*rpa-21* ⟵ *something*[Target] “is knock out”;	RBA02, RBA04
22.	RPA22	*rpa-22* ⟵ “You just do not” VERB[Activity];	RBA01
23.	RPA23	*rpa-23* ⟵ *something*[Target] “is removed”;	RBA04
24.	RPA24	*rpa-24* ⟵ (“could” | “would” | “can”) “not” VERB[Activity] *something*[Target];	RBA05
25.	RPA25	*rpa-25* ⟵ *something*[Target] “out of ordinary”;	RBA02
26.	RPA26	*rpa-26* ⟵ “Let us implement” *something*[Target];	RBA02, RBA04

ตารางที่ 6-2 User Review Patterns ที่สอดคล้องกับลักษณะของ Availability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Availability User Review Pattern	Requirement Boilerplate Code
27.	RPA27	*rpa-27* ⟵ “problem related to” *something*[Target];	RBA02
28.	RPA28	*rpa-28* ⟵ *something*[Agent] (“can” | “could”) “use”[Activity] *something*[Target];	RBA06, RBA07
29.	RPA29	*rpa-29* ⟵ *something*[Agent] (“should” | “could” | “can” | “must”) “solve” *something*[Target];	RBA06
30.	RPA30	*rpa-30* ⟵ *someone* (“could” | “can”) (“run” | “prepare”)[Activity] *something*[Target];	RBA03
31.	RPA31	*rpa-31* ⟵ *something*[Agent] “resolves” *something*[Target];	RBA06
32.	RPA32	*rpa-32* ⟵ *something*[Target] “will work”;	RBA02, RBA04
33.	RPA33	*rpa-33* ⟵ “goal is to provide” *something*[Target];	RBA02, RBA04
34.	RPA34	*rpa-34* ⟵ *someone* (“should” | “could”) “modify”[Activity] *something*[Target];	RBA03
35.	RPA35	*rpa-35* ⟵ “if” SUBJECT VERB[Activity] *something*[Target] “would work”;	RBA02, RBA03, RBA05
36.	RPA36	*rpa-36* ⟵ *something*[Target] “needs to be” VERB[Activity];	RBA05
37.	RPA37	*rpa-37* ⟵ “Try to create[Activity]” *something*[Target];	RBA02, RBA03, RBA05

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
1.	RPOP01	*rpop-01* ⟵ “To make” *something*[Target] “workable” (*something* “needs” *something*)[Way];	RBOP01
2.	RPOP02	*rpop-02* ⟵ *someone* “expects” *something*[Target] “to work”;	RBOP02, RBOP03, RBOP04
3.	RPOP03	*rpop-03* ⟵ *someone* (“could” | “should”) (“add” | “provide” | “offer” | “integrate”)[Activity] *something*[Target];	RBOP02, RBOP04, RBOP05
4.	RPOP04	*rpop-04* ⟵ *someone* “would love to” (“see” | “use”[Activity]) *something*[Target];	RBOP03, RBOP04
5.	RPOP05	*rpop-05* ⟵ *someone* “thinks” *something*[Agent] “would be” MODAL “to have[Activity]” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP06
6.	RPOP06	*rpop-06* ⟵ *someone* “thinks” *something*[Target] “should” VERB[Activity];	RBOP02, RBOP05
7.	RPOP07	*rpop-07* ⟵ “Let us implement” *something*[Target];	RBOP02, RBOP03, RBOP04
8.	RPOP08	*rpop-08* ⟵ *something*[Target] “needs to be” VERB[Activity];	RBOP2, RBOP05
9.	RPOP09	*rpop-09* ⟵ “All that is needed is” *something*[Target];	RBOP02, RBOP03, RBOP04
10.	RPOP10	*rpop-10* ⟵ *something*[Target] “is what” *someone* “needs”;	RBOP02, RBOP03, RBOP04

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
11.	RPOP11	*rpop-11* ⟵ “Could we add[Activity]” *something*[Target] “?”;	RBOP02, RBOP04, RBOP05
12.	RPOP12	*rpop-12* ⟵ *someone* “thinks” *something*[Target] “is” MODAL “to” VERB[Activity];	RBOP02, RBOP05
13.	RPOP13	*rpop-13* ⟵ “Why do not change[Activity]” *something*[Target] “?”;	RBOP02, RBOP05
14.	RPOP14	*rpop-14* ⟵ *something*[Target] “should be fixed”;	RBOP09
15.	RPOP15	*rpop-15* ⟵ “it would be” MODAL “to” VERB[Activity];	RBOP02, RBOP03, RBOP05
16.	RPOP16	*rpop-16* ⟵ *someone* “needs to create[Activity]” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP05
17.	RPOP17	*rpop-17* ⟵ *something*[Target] “is required”;	RBOP02, RBOP03, RBOP04
18.	RPOP18	*rpop-18* ⟵ *something*[Agent] “could have” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP06
19.	RPOP19	*rpop-19* ⟵ “It would be” (“possible” | “beneficial”) “to have” *something*[Target];	RBOP02, RBOP03, RBOP04
20.	RPOP20	*rpop-20* ⟵ “Please” (“remove” | “check”)[Activity] *something*[Target];	RBOP02, RBOP05
21.	RPOP21	*rpop-21* ⟵ “Can” *something*[Agent] “fix” *something*[Target] “?”;	RBOP10

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
22.	RPOP22	*rpop-22* ⟵ *something*[Agent] “requires” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP06
23.	RPOP23	*rpop-23* ⟵ *something*[Target] “should be treated” (“in” | “as”) *something*[Agent];	RBOP02, RBOP03, RBOP04, RBOP06
24.	RPOP24	*rpop-24* ⟵ *someone* “thinks” *something*[Target] “needs” VERB[Activity];	RBOP02, RBOP05
25.	RPOP25	*rpop-25* ⟵ *something*[Agent] (“should” | “could”) (“define” | “support”)[Activity] *something*[Target];	RBOP02, RBOP04, RBOP05
26.	RPOP26	*rpop-26* ⟵ *something*[Target] (“should” | “could”) “be” VERB[Activity];	RBOP02, RBOP05
27.	RPOP27	*rpop-27* ⟵ “adding”[Activity] *something*[Target];	RBOP02, RBOP04, RBOP05
28.	RPOP28	*rpop-28* ⟵ “All” *someone* “needs is” *something*[Target];	RBOP02, RBOP03, RBOP04
29.	RPOP29	*rpop-29* ⟵ *someone* “requests” *something*[Target];	RBOP02, RBOP03, RBOP04
30.	RPOP30	*rpop-30* ⟵ *something*[Agent] (“should” | “could”) “include” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP06
31.	RPOP31	*rpop-31* ⟵ *something*[Target] “ought to be” *somehow*[Way];	RBOP07
32.	RPOP32	*rpop-32* ⟵ *someone* (“wants” | “likes”) “to have” *something*[Target];	RBOP02, RBOP03, RBOP04

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
33.	RPOP33	*rpop-33* ⟵ *something*[Agent] “needs” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP06
34.	RPOP34	*rpop-34* ⟵ “An issue to fix is” *something*[Target];	RBOP09
35.	RPOP35	*rpop-35* ⟵ “It is a” (“mistake” | “problem”) (“at” | “of | “with”) *something*[Target];	RBOP09
36.	RPOP36	*rpop-36* ⟵ *something*[Target] “is missing”;	RBOP02, RBOP03, RBOP04
37.	RPOP37	*rpop-37* ⟵ *something*[Target] “is unable to” VERB[Activity];	RBOP08
38.	RPOP38	*rpop-38* ⟵ *something*[Target] “does not work”;	RBOP02, RBOP03, RBOP04
39.	RPOP39	*rpop-39* ⟵ *something*[Agent] “not do” *something*[Target];	RBOP02, RBOP03, RBOP04
40.	RPOP40	*rpop-40* ⟵ *something*[Target] “is not able to” VERB[Activity];	RBOP08
41.	RPOP41	*rpop-41* ⟵ “problem related to” *something*[Target];	RBOP09
42.	RPOP42	*rpop-42* ⟵ *something*[Target] “is not defined[Activity]”;	RBOP02, RBOP04, RBOP05
43.	RPOP43	*rpop-43* ⟵ *something*[Target] “cannot be found”;	RBOP02, RBOP03, RBOP04
44.	RPOP44	*rpop-44* ⟵ “Please” (“adjust” | “fix”)[Activity] *something*[Target];	RBOP02, RBOP05

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
45.	RPOP45	*rpop-45* ⟵ *someone* “suggests” *something*[Target];	RBOP02, RBOP03, RBOP04
46.	RPOP46	*rpop-46* ⟵ *something*[Target] “would be a good” (“idea” | “thing”);	RBOP02, RBOP03, RBOP04
47.	RPOP47	*rpop-47* ⟵ *something*[Agent] (“can” | “could”) “use”[Activity] *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP05, RBOP06
48.	RPOP48	*rpop-48* ⟵ *someone* “is” (“thinking” | “planning”) “of adding[Activity]” *something*[Target];	RBOP02, RBOP04, RBOP05
49.	RPOP49	*rpop-49* ⟵ *something*[Agent] (“should” | “could” | “can” | “must”) “solve” *something*[Target];	RBOP10
50.	RPOP50	*rpop-50* ⟵ “A new” *something*[Target] “can be made”;	RBOP02, RBOP03, RBOP04
51.	RPOP51	*rpop-51* ⟵ *someone* (“could” | “can”) (“run” | “prepare”)[Activity] *something*[Target];	RBOP02, RBOP03, RBOP05
52.	RPOP52	*rpop-52* ⟵ *someone* “thinks” *something*[Target] “can help”;	RBOP02, RBOP03, RBOP04
53.	RPOP53	*rpop-53* ⟵ *something*[Agent] “resolves” *something*[Target];	RBOP10
54.	RPOP54	*rpop-54* ⟵ *something*[Target] “could be done”;	RBOP02, RBOP03, RBOP04
55.	RPOP55	*rpop-55* ⟵ “could allow” *something*[Target] “to” VERB[Activity];	RBOP02, RBOP05

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
56.	RPOP56	*rpop-56* ⟵ “I propose” *something*[Target];	RBOP02, RBOP03, RBOP04
57.	RPOP57	*rpop-57* ⟵ “You need to do” *something*[Target];	RBOP02, RBOP03, RBOP04
58.	RPOP58	*rpop-58* ⟵ “goal is to provide[Activity]” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP05
59.	RPOP59	*rpop-59* ⟵ “solution consists” (“of” | “in”) *something*[Target];	RBOP02, RBOP03, RBOP04
60.	RPOP60	*rpop-60* ⟵ *something*[Target] (“could” | “might”) “be not a bad” (“idea” | “thing”);	RBOP02, RBOP03, RBOP04
61.	RPOP61	*rpop-61* ⟵ *something*[Target] “would allow to” VERB[Activity];	RBOP02, RBOP05
62.	RPOP62	*rpop-62* ⟵ *someone* (“should” | “could”) “modify”[Activity] *something*[Target];	RBOP02, RBOP03, RBOP05
63.	RPOP63	*rpop-63* ⟵ “solution is” *something*[Target];	RBOP02, RBOP03, RBOP04
64.	RPOP64	*rpop-64* ⟵ “a way is” *something*[Target];	RBOP02, RBOP03, RBOP04
65.	RPOP65	*rpop-65* ⟵ *someone* (“should” | “could”) “recommend[Activity]” *something*[Target];	RBOP02, RBOP03, RBOP05
66.	RPOP66	*rpop-66* ⟵ *something*[Agent] (“should” | “could”) “hold”[Activity] *something*[Target];	RBOP02, RBOP05

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
67.	RPOP67	*rpop-67* ⟵ *someone* (“should” | “could”) “work on” something[Target]	RBOP02, RBOP03, RBOP04
68.	RPOP68	*rpop-68* ⟵ “New changes include” *something*[Target];	RBOP02, RBOP03, RBOP04
69.	RPOP69	*rpop-69* ⟵ “I am interested to contribute to” *something*[Target];	RBOP02, RBOP03, RBOP04
70.	RPOP70	*rpop-70* ⟵ (“Plan” | “idea” | “aim”) “is” *something*[Target];	RBOP02, RBOP03, RBOP04
71.	RPOP71	*rpop-71* ⟵ *someone* “recommends” *something*[Target];	RBOP02, RBOP03, RBOP04
72.	RPOP72	*rpop-72* ⟵ *something*[Agent] (“provides” | “supports”)[Activity] *something*[Target];	RBOP02, RBOP04, RBOP05
73.	RPOP73	*rpop-73* ⟵ “There are efforts” (“to” | “for”) *something*[Target];	RBOP09
74.	RPOP74	*rpop-74* ⟵ *someone* “should create[Activity]” *something*[Target];	RBOP02, RBOP03, RBOP04, RBOP05
75.	RPOP75	*rpop-75* ⟵ *something*[Target] “is removed”;	RBOP02, RBOP03, RBOP04
76.	RPOP76	*rpop-76* ⟵ *someone* “is looking” (“to” | “for”) VERB[Activity];	RBOP02, RBOP03, RBOP05
77.	RPOP77	*rpop-77* ⟵ *someone* “expects” *something*[Target];	RBOP02, RBOP03, RBOP04
78.	RPOP78	*rpop-78* ⟵ *someone* “wants to know” *something*[Target];	RBOP04, RBOP11
79.	RPOP79	*rpop-79* ⟵ (“Is” | “Are”) “there” *something*[Target] “?”;	RBOP02, RBOP03, RBOP04

ตารางที่ 6-3 User Review Patterns ที่สอดคล้องกับลักษณะของ Operability User Reviews ในรูปของ EBNF Syntax (ต่อ)

No.	Review Pattern Code	Operability User Review Pattern	Requirement Boilerplate Code
80.	RPOP80	*rpop-80* ⟵ “Let me know if” *something*[Target];	RBOP11
81.	RPOP81	*rpop-81* ⟵ “Does” *something*[Agent] “provide[Activity]” *something*[Target] “?”;	RBOP02, RBOP04, RBOP05
82.	RPOP82	*rpop-82* ⟵ “Can” *someone* (“get” | “do” | “use”)[Activity] *something*[Target] “?”;	RBOP02, RBOP03, RBOP04, RBOP05
83.	RPOP83	*rpop-83* ⟵ “Can I find[Activity]” *something*[Target] “?”;	RBOP02, RBOP03, RBOP04, RBOP05
84.	RPOP84	*rpop-84* ⟵ “Please share[Activity]” *something*[Target];	RBOP02, RBOP05
85.	RPOP85	*rpop-85* ⟵ (“could” | “would” | “can”) “not” VERB[Activity] *something*[Target];	RBOP02, RBOP03, RBOP05
86.	RPOP86	*rpop-86* ⟵ “Is” *something*[Target] “possible?”;	RBOP02, RBOP03, RBOP04

ตารางที่ 6-4 โครงสร้างไวยากรณ์ทางภาษาของ Non-terminals ในรูปของ EBNF Syntax

Non-terminal	Grammar Structure
SOMEONE	*someone* ⟵ *someone-noun* {*someone-noun*}; *someone-noun* ⟵ PROPER NOUN | PERSONAL PRONOUN;
SOMETHING	*something* ⟵ *premodifier something-noun* {*something-noun*} [*postmodifier-prepositional-phrase* | *postmodifier-to-infinitives*]; *postmodifier-to-infinitives* ⟵ “to” VERB; *postmodifier-prepositional-phrase* ⟵ (“with” | “in” | “on”) *premodifier something-noun* {*something-noun*}; *premodifier* ⟵ [DETERMINER] [NUMERAL] [ADJECTIVE]; *something-noun* ⟵ PROPER NOUN | NOUN;
SOMEHOW	*somehow* ⟵ *something* VERB [*something*];

6.2 Information Extraction for Agent, Activity, Target, and Way

ในขั้นตอนนี้ เมื่อบทวิจารณ์ของผู้ใช้งานใดได้ถูกจับคู่กับ User Review Patterns ที่กำหนดไว้ วิทยานิพนธ์จะสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน เพื่อนำมาใช้สร้างความต้องการ

จากงานวิจัย [42] ประโยคความต้องการในรูปแบบภาษาธรรมชาติระดับเบื้องต้นอยู่ในรูปแบบ Requirement ⟵ Agent . Activity . Target . [Way] โดยที่ Agent หมายถึงเอนทิตีซึ่งเป็นผู้กระทำพฤติกรรมตามที่กิจกรรมได้กำหนดไว้, Activity หมายถึงการกระทำที่ถูกทำโดย Agent, Target หมายถึงเอนทิตีทางกายภาพหรือแนวความคิด, Way หมายถึงแนวทางการทำกิจกรรม ทั้ง Agent, Activity, Target คือข้อมูลที่จำเป็นต้องมี ในขณะที่ Way คือข้อมูลทางเลือก ซึ่งมีหรือไม่มีข้อมูลก็ได้

ดังนั้นวิทยานิพนธ์ได้กำหนดตำแหน่งของ Agent, Activity, Target, Way ลงในแต่ละ User Review Patterns ดังตารางที่ 6-1 ถึงตารางที่ 6-3 เพื่อสกัดข้อมูล ณ ตำแหน่งที่กำหนด ซึ่งวิทยานิพนธ์คาดว่าเป็นข้อมูลที่จำเป็นสำหรับการสร้างประโยคความต้องการ

ยกตัวอย่างการกำหนดตำแหน่งลงใน Functional User Review Patterns รหัส RPFN01

rpfn-01 ⟵ *someone* (“could” | “should”) (“add” | “provide” | “offer” | “integrate”)$^{\text{Activity}}$ *something*$^{\text{Target}}$;

หมายเหตุ Functional User Review Patterns รหัส RPFN01 วิทยานิพนธ์ได้กำหนดตำแหน่งแค่เพียง Activity และ Target เท่านั้น จากตัวอย่างก่อนหน้านี้หลังจากที่บทวิจารณ์ของผู้ใช้งานได้ถูกจับคู่กับ Functional User Review Patterns รหัส RPFN01 แล้ว วิทยานิพนธ์สกัดข้อมูล Activity และ Target ตามตำแหน่งที่กำหนดใน Functional User Review Patterns ดังนั้น จะได้ “add” คือข้อมูลตำแหน่ง Activity และ “a like button in the comment section” คือข้อมูลตำแหน่ง Target

6.3 Requirement Generation

วิทยานิพนธ์ใช้ Requirement Boilerplates สำหรับสร้างประโยคความต้องการ วิทยานิพนธ์สร้าง Requirement Boilerplates ตามแนวทางการเขียนความต้องการที่มีรูปแบบที่ดีในภาษาธรรมชาติ (Individual Requirement) จาก ISO/IEC/IEEE 29148 [9] แบ่งออกเป็น Functional Requirement Boilerplates จำนวน 10 Requirement Boilerplates ดังตารางที่ 6-5, Availability Requirement Boilerplates จำนวน 7 Requirement Boilerplates ดังตารางที่ 6-6, Operability Requirement Boilerplates จำนวน 11 Requirement Boilerplates ดังตารางที่ 6-7

ตารางที่ 6-5 Functional Requirement Boilerplates

No.	Requirement Boilerplate Code	Functional Requirement Boilerplate
1.	RBFN01	The <Agent> shall be able to <Activity> <Target>.
2.	RBFN02	The system shall allow <Target> to be in <Agent>.
3.	RBFN03	The <Agent> shall be able to <Activity> <Target> with <Way>.
4.	RBFN04	The <Agent> shall allow a user to be able to <Activity> <Target>.
5.	RBFN05	The <Agent> shall have <Target>.
6.	RBFN06	The <Agent> shall <Activity> <Target>.
7.	RBFN07	The system shall have <Target> fixed.
8.	RBFN08	The system shall have <Agent> fix <Target>.
9.	RBFN09	The <Target> shall be able to <Activity>.
10.	RBFN10	The system shall notify a user about <Target>.

ตารางที่ 6-6 Availability Requirement Boilerplates

No.	Requirement Boilerplate Code	Availability Requirement Boilerplate
1.	RBA01	The system shall be able to <Activity>.
2.	RBA02	The system shall have <Target> fixed.
3.	RBA03	The <Target> shall be able to <Activity>.
4.	RBA04	The <Target> shall be available for use.
5.	RBA05	The system shall <Activity> <Target>.
6.	RBA06	The system shall have <Agent> fix <Target>.
7.	RBA07	The <Agent> shall be able to <Activity> <Target>.

ตารางที่ 6-7 Operability Requirement Boilerplates

No.	Requirement Boilerplate Code	Operability Requirement Boilerplate
1.	RBOP01	A user shall be able to successfully use the <Target> by having <Way>.
2.	RBOP02	The <Agent> shall be able to <Activity> <Target>.
3.	RBOP03	The <Agent> shall allow a user to be able to <Activity> <Target>.
4.	RBOP04	The <Agent> shall have <Target>.
5.	RBOP05	The <Agent> shall <Activity> <Target>.
6.	RBOP06	The system shall allow <Target> to be in <Agent>.
7.	RBOP07	The <Agent> shall be able to <Activity> <Target> with <Way>.
8.	RBOP08	The <Target> shall be able to <Activity>.
9.	RBOP09	The system shall have <Target> fixed.
10.	RBOP10	The system shall have <Agent> fix <Target>.
11.	RBOP11	The system shall notify a user about <Target>.

ในแต่ละ Requirement Boilerplate มี Placeholders สำหรับ Agent, Activity, Target หรือ Way ซึ่งสอดคล้องกันกับตำแหน่งของข้อมูลที่วิทยานิพนธ์ได้สกัดจากขั้นตอนที่ 6.2 วิทยานิพนธ์แบ่งขั้นตอนการสร้างความต้องการออกเป็น 2 ขั้นตอนประกอบด้วย 1) การเลือก Requirement Boilerplate ที่เหมาะสม และ 2) การแทนที่ข้อมูลตามตำแหน่งของ Placeholders ใน Requirement Boilerplate

1) การเลือก Requirement Boilerplate ที่เหมาะสม

การเลือก Requirement Boilerplate ที่เหมาะสมสำหรับข้อมูลบทวิจารณ์ของผู้ใช้งานนั้น วิทยานิพนธ์วิเคราะห์จาก User Review Pattern ซึ่งจะถูกจับคู่กับข้อมูลบทวิจารณ์ของผู้ใช้งาน วิทยานิพนธ์ได้กำหนดการจับคู่กันระหว่าง Requirement Boilerplate และ User Review Pattern ของข้อมูลแต่ละประเภท ดังตารางที่ 6-1 ถึงตารางที่ 6-3 แต่ละ User Review Pattern จะได้รับการจับคู่จำนวนตั้งแต่ 1 หรือมากกว่า 1 กับ Requirement Boilerplate ยกตัวอย่าง Functional User

Review Patterns รหัส RPFN01 จะถูกจับคู่กับ 3 Functional Requirement Boilerplates ได้แก่ รหัส RBFN01, RBFN05, และ RBFN06

2) การแทนที่ข้อมูลตามตำแหน่งของ Placeholders ใน Requirement Boilerplate

หลังจากได้กำหนด Requirement Boilerplate สำหรับใช้เป็นโครงสร้างการสร้างประโยคความต้องการแล้ว ข้อมูล Agent, Activity, Target, Way ที่สกัดได้จากบทวิจารณ์ของผู้ใช้งานจะถูกนำมาแทนที่ตามตำแหน่งของ Placeholders ที่กำหนดไว้ใน Requirement Boilerplate

สำหรับข้อมูล Activity คือการกระทำที่ถูกทำโดย Agent ซึ่งมีโครงสร้างไวยากรณ์ในภาษาอังกฤษคือ Verb จะถูกนำมาทำ Lemmatization ก่อนนำข้อมูลมาแทนที่ตามตำแหน่งของ Placeholders ที่กำหนดไว้ เพื่อปรับโครงสร้างไวยากรณ์ในภาษาอังกฤษของข้อมูล

ในกรณีที่มีข้อมูลที่ต้องการใน Requirement Boilerplate มากกว่าที่สกัดได้จากบทวิจารณ์ของผู้ใช้งาน วิทยานิพนธ์ได้กำหนดค่าโดยปริยาย (Default) ของข้อมูล <Agent>, <Activity>, <Target> ซึ่งจะถูกนำมาใช้แทนที่ใน Requirement Boilerplate แทน โดยมีค่าโดยปริยายของข้อมูลดังนี้

a) Functional Requirement Boilerplates และ Operability Requirement Boilerplates มีค่าโดยปริยายของข้อมูล ได้แก่ <Agent> คือ “system”, <Activity> คือ “use”, และ <Target> คือไม่มีค่าใด ๆ
b) Availability Requirement Boilerplates มีค่าโดยปริยายของข้อมูล ได้แก่ <Agent> คือ “system”

ยกตัวอย่าง Functional User Reviews: “you should add a like button in the comment section” ถูกจับคู่กับ Functional User Review Patterns รหัส RPFN01 และสามารถสกัดข้อมูล ณ ตำแหน่ง Activity และ Target ได้จากบทวิจารณ์ของผู้ใช้งาน ซึ่ง Functional User Review Patterns รหัส RPFN01 สามารถสร้างประโยคความต้องการได้จาก Functional Requirement Boilerplates รหัส RBFN01, RBFN05, RBFN06 ดังนั้นประโยคความต้องการที่ได้จากแนวทางของวิทยานิพนธ์ คือ

RBFN01: “ The system[Agent] shall be able to add[Activity] a like button in the comment section[Target].”

RBFN05: “ The system[Agent] shall have a like button in the comment section[Target].”

RBFN06: “ The system[Agent] shall add[Activity] a like button in the comment section[Target].”

สาเหตุที่ทำให้สามารถสร้างประโยคความต้องการได้มากกว่า 1 ประโยคความต้องการ เนื่องจากว่านักพัฒนาสามารถตัดสินใจเลือกใช้ประโยคความต้องการที่เหมาะสมที่สุดสำหรับบริบทของข้อมูลได้

6.4 Evaluation of Requirement Quality

ในขั้นตอนนี้เป็นการประเมินคุณภาพของความต้องการที่ได้จากแนวทางการสร้างความต้องการของวิทยานิพนธ์ จากแนวทางการทดลองผู้วิจัยได้นำบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews และ Operability User Reviews มาใช้ทดลองมีจำนวน 3,475, 1,589 และ 2,969 บทวิจารณ์ของผู้ใช้งานตามลำดับ ได้ผลลัพธ์ประโยคความต้องการประเภท Functional, Availability, Operability ซึ่งมีรายการของข้อมูลแต่ละประเภทดังต่อไปนี้

a) ความต้องการประเภท Functional

บทวิจารณ์ของผู้ใช้งานที่นำมาทดสอบถูกจับคู่ได้กับ 6 ใน 10 รูปแบบของ Functional Requirement Boilerplates ได้แก่ RBFN01, RBFN02, RBFN04, RBFN05, RBFN06, RBFN10 โดยมาจาก Functional User Review Patterns ทั้งหมด 20 ใน 84 แบบรูป ได้แก่ RPFN01, RPFN02, RPFN06, RPFN07, RPFN12, RPFN15, RPFN17, RPFN18, RPFN21, RPFN22, RPFN23, RPFN28, RPFN29, RPFN30, RPFN33, RPFN59, RPFN62, RPFN74, RPFN76, RPFN83

b) ความต้องการประเภท Availability

บทวิจารณ์ของผู้ใช้งานที่นำมาทดสอบถูกจับคู่ได้กับ 4 ใน 6 รูปแบบของ Availability Requirement Boilerplates ได้แก่ RBA02, RBA03, RBA04, RBA05 โดยมาจาก Availability User Review Patterns ทั้งหมด 10 ใน 37 แบบรูป ได้แก่ RPA09, RPA10, RPA12, RPA13, RPA14, RPA18, RPA20, RPA24, RPA36, RPA37

c) ความต้องการประเภท Operability

บทวิจารณ์ของผู้ใช้งานที่นำมาทดสอบถูกจับคู่ได้กับ 6 ใน 11 รูปแบบของ Operability Requirement Boilerplates ได้แก่ RBOP02, RBOP03, RBOP04, RBOP05, RBOP06, RBOP11 โดยมาจาก Operability User Review Patterns ทั้งหมด 12 ใน 86 แบบรูป ได้แก่ RPOP06, RPOP15, RPOP17, RPOP22, RPOP27, RPOP33, RPOP44, RPOP63, RPOP71, RPOP75, RPOP78, RPOP85

จากการทดลองพบว่ามี User Review Patterns ที่ไม่ได้รับการจับคู่กับบทวิจารณ์ของผู้ใช้งาน บาง User Review Patterns ได้รับการจับคู่กับ 1 บทวิจารณ์ของผู้ใช้งาน หรือมากกว่า 1 บทวิจารณ์ของผู้ใช้งาน ดังรูปที่ 6-1 ข้างต้น เมื่อบทวิจารณ์ของผู้ใช้งานใดได้รับการจับคู่กับ User Review Patterns แล้ว ในลำดับถัดมาบทวิจารณ์ของผู้ใช้งานนั้นจะถูกนำมาสกัดข้อมูลและสร้างความต้องการซอฟต์แวร์ แต่หากว่าบทวิจารณ์ของผู้ใช้งานนั้นไม่ได้รับการจับคู่กับ User Review Patterns บทวิจารณ์ของผู้ใช้งานนั้นจะไม่ถูกนำมาสร้างความต้องการซอฟต์แวร์

เนื่องจากความต้องการซอฟต์แวร์ที่ได้จากการทดลองมีจำนวนมาก ดังนั้นเพื่อให้เหมาะสมกับการประเมินคุณภาพโดยใช้การพิจารณาโดยผู้ประเมิน ผู้วิจัยจึงได้สุ่มเลือกบทวิจารณ์ของผู้ใช้งานจำนวน 1-2 รายการที่ได้รับการจับคู่กับแต่ละ User Review Patterns เนื่องจากว่าบาง User Review Pattern ได้รับการจับคู่กับเพียง 1 บทวิจารณ์ของผู้ใช้งาน และบาง User Review Pattern ได้รับการจับคู่มากกว่า 1 บทวิจารณ์ของผู้ใช้งาน การประเมินจะครอบคลุม User Review Patterns ทั้งหมดที่จับคู่ได้กับบทวิจารณ์ของผู้ใช้งานที่ใช้ในการทดสอบ

วิทยานิพนธ์นำประโยคความต้องการทั้งหมดที่ถูกสร้างขึ้นจากการจับคู่กันระหว่างบทวิจารณ์ของผู้ใช้งานที่สุ่มเลือกมากับ User Review Patterns มาประเมินคุณภาพ ซึ่งประกอบด้วย ความต้องการประเภท Functional จำนวน 85 ประโยค ความต้องการประเภท Availability จำนวน 23 ประโยค และความต้องการ Operability จำนวน 41 ประโยค

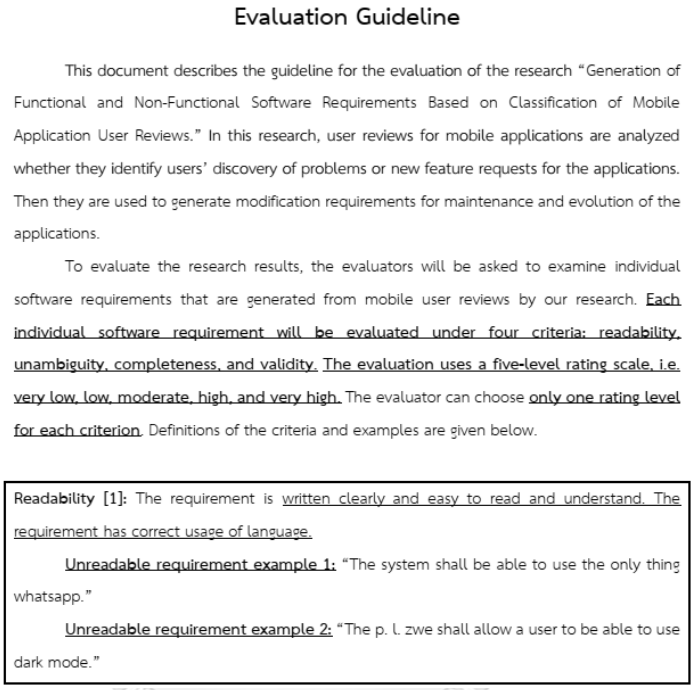
วิทยานิพนธ์ประเมินคุณภาพของความต้องการซอฟต์แวร์ที่สร้างจากแนวทางของวิทยานิพนธ์ทั้งหมด 4 เกณฑ์คุณภาพ ได้แก่ 1) Readability [43] คือความต้องการถูกเขียนอย่างชัดเจน สามารถอ่านเข้าใจได้ง่าย และถูกต้องตามหลักภาษาศาสตร์ 2) Unambiguity [9] คือความต้องการสามารถตีความความหมายได้เพียงทางเดียวเท่านั้น 3) Completeness [9] คือความต้องการอธิบายข้อมูลที่จำเป็นด้านความสามารถ ลักษณะ ข้อจำกัดหรือคุณภาพได้อย่างเพียงพอ โดยไม่ต้องการข้อมูลใดเพิ่มเติมอีก และ 4) Validity [9] คือความต้องการตรงกับเจตนาของผู้มีส่วนได้ส่วนเสีย (Stakeholder) หมายเหตุ เฉพาะเกณฑ์ Validity ผู้ประเมินจะต้องเปรียบเทียบระหว่างความต้องการที่ได้กับบทวิจารณ์ของผู้ใช้งานต้นฉบับ

สำหรับแนวทางการประเมินความต้องการซอฟต์แวร์ที่ได้จากแนวทางของวิทยานิพนธ์มีรายละเอียดดังนี้

a) ประเมินโดยนิสิตระดับปริญญาโท สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย จำนวน 6 คน

b) ประเมินความต้องการทั้งหมด 4 เกณฑ์คุณภาพด้วยแบบสอบถาม มีระดับการประเมิน 5 ระดับ ได้แก่ ต่ำมาก (Very low) ต่ำ (Low) กลาง (Moderate) สูง (High) สูงมาก (Very high)

c) ก่อนการประเมิน ผู้ประเมินจะได้รับการอธิบายถึงวิธีการประเมิน พร้อมทั้งคู่มือการประเมิน ดังรูปที่ 6-3 ซึ่งอธิบายถึงนิยามของเกณฑ์คุณภาพและตัวอย่างประกอบ



Evaluation Guideline

This document describes the guideline for the evaluation of the research "Generation of Functional and Non-Functional Software Requirements Based on Classification of Mobile Application User Reviews." In this research, user reviews for mobile applications are analyzed whether they identify users' discovery of problems or new feature requests for the applications. Then they are used to generate modification requirements for maintenance and evolution of the applications.

To evaluate the research results, the evaluators will be asked to examine individual software requirements that are generated from mobile user reviews by our research. Each individual software requirement will be evaluated under four criteria: readability, unambiguity, completeness, and validity. The evaluation uses a five-level rating scale, i.e. very low, low, moderate, high, and very high. The evaluator can choose only one rating level for each criterion. Definitions of the criteria and examples are given below.

Readability [1]: The requirement is written clearly and easy to read and understand. The requirement has correct usage of language.

Unreadable requirement example 1: "The system shall be able to use the only thing whatsapp."

Unreadable requirement example 2: "The p. l. zwe shall allow a user to be able to use dark mode."

รูปที่ 6-3 ตัวอย่างคู่มือการประเมิน

ผลการประเมินความต้องการซอฟต์แวร์ประเภท Functional, Availability, Operability แสดงดังตารางที่ 6-8 เกณฑ์คุณภาพ Readability ได้ระดับ High ในทุกประเภทความต้องการซอฟต์แวร์ซึ่งแสดงถึงความต้องการซอฟต์แวร์มีความสามารถในการอ่านและสามารถเข้าใจได้ ไม่ค่อยพบปัญหาด้านการใช้ภาษา ในขณะที่เกณฑ์คุณภาพ Unambiguity, Completeness, Validity ได้ระดับ Moderate แสดงถึงความต้องการไม่ค่อยมีความกำกวม ค่อนข้างมีความสมบูรณ์ของข้อมูลและค่อนข้างตรงตามเจตนาของผู้มีส่วนได้ส่วนเสีย ยกเว้นความต้องการซอฟต์แวร์ประเภท Operability ที่ได้รับระดับ Low ในเกณฑ์คุณภาพ Completeness และ Validity ตัวอย่างความต้องการซอฟต์แวร์และระดับของเกณฑ์คุณภาพที่ได้จากการประเมินแสดงดังตารางที่ 6-9

ตารางที่ 6-8 ผลการประเมินความต้องการซอฟต์แวร์ประเภท Functional, Availability, Operability ทั้ง 4 เกณฑ์คุณภาพ ได้แก่ Readability, Unambiguity, Completeness, Validity

เกณฑ์คุณภาพ	**ระดับ (Functional)**	**ระดับ (Availability)**	**ระดับ (Operability)**
Readability	High (4.08)	High (4.13)	High (4.13)
Unambiguity	Moderate (3.28)	Moderate (3.38)	Moderate (3.15)
Completeness	Moderate (3.28)	Moderate (3.16)	Low (2.85)
Validity	Moderate (3.24)	Moderate (3.48)	Low (2.54)

หมายเหตุ ภายในวงเล็บคือค่าเฉลี่ยระดับการประเมิน โดยช่วงของค่า [1 , 2.33] คือระดับ Low, (2.33, 3.66] คือระดับ Moderate, (3.66, 5] คือระดับ High

จากการวิเคราะห์ประโยคความต้องการซอฟต์แวร์ที่ได้รับเกณฑ์คุณภาพ Readability ที่ระดับ High พบว่าบทวิจารณ์ของผู้ใช้งานมีคุณภาพดี ให้ข้อมูลที่ชัดเจนและ User Review Pattern สามารถสกัดข้อมูลจากบทวิจารณ์ของผู้ใช้งานได้ครบถ้วน นอกจากนี้ Requirement Boilerplate มีโครงสร้างทางภาษาที่ชัดเจน ซึ่งมีส่วนทำให้ความต้องการซอฟต์แวร์มีความสามารถในการอ่านและสามารถเข้าใจได้ สำหรับเกณฑ์คุณภาพ Unambiguity และ Completeness ที่ได้ผลการประเมินระดับ Moderate พบว่าลักษณะข้อมูลบทวิจารณ์ของผู้ใช้งานเป็นข้อมูลในมุมมองของผู้ใช้งาน ซึ่งทำให้ได้ข้อมูลไม่เพียงพอต่อการสร้างความต้องการซอฟต์แวร์ที่ชัดเจนและเฉพาะเจาะจงมากพอ นอกจากนี้เกิดจากข้อจำกัดในการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน ซึ่งทำให้ได้รับข้อมูลไม่ครบถ้วน ข้อมูลบางส่วนขาดหาย แม้ว่าบทวิจารณ์ของผู้ใช้งานจะให้ข้อมูลที่ครบถ้วนก็ตาม ซึ่งส่งผลต่อระดับเกณฑ์คุณภาพของ Readability และ Validity ด้วย ในบางครั้งข้อมูลที่สกัดได้จากบทวิจารณ์ของผู้ใช้งานเช่นคำว่า “some”, “that”, “this” จัดอยู่ในประเภทของ Determiner ซึ่งเป็นคำที่ไม่จำเพาะเจาะจง สามารถตีความได้หลากหลาย ไม่ชัดเจน จึงทำให้ประโยคความต้องการมีความกำกวมเกิดขึ้น ความสมบูรณ์ของเนื้อหาลดลง ส่งผลกระทบต่อเกณฑ์คุณภาพทั้ง Unambiguity และ Completeness ได้รับการประเมินอยู่ในระดับ Low สำหรับเกณฑ์คุณภาพ Validity ที่ระดับ Low เนื่องจากว่าแนวทางการสกัดข้อมูลจากบทวิจารณ์ของผู้ใช้งานไม่ได้คำนึงถึงบริบททางความหมาย ซึ่งทำให้บางครั้งความต้องการไม่ตรงตามเจตนาของผู้ใช้งานที่ระบุในบทวิจารณ์ของผู้ใช้งาน

นอกจากนี้ในบางครั้งพบว่าแนวทางการตัดประโยคของวิทยานิพนธ์ด้วยเครื่องหมายมหัพภาค (Full Stop) มีประสิทธิภาพไม่เพียงพอเพื่อแยกประโยคออกจากกัน ทำให้ในการสกัดข้อมูลได้รับข้อมูลที่ไม่มีประโยชน์ติดมาด้วย ส่งผลทำให้ประโยคความต้องการไม่มีความหมายและไม่สามารถนำไปใช้งานต่อได้

จากการวิเคราะห์สาเหตุที่ความต้องการซอฟต์แวร์ประเภท Operability ได้รับผลการประเมินเกณฑ์คุณภาพ Completeness และ Validity ในระดับ Low ในขณะที่ความต้องการซอฟต์แวร์ประเภท Functional และ Availability ได้รับผลการประเมินระดับ Moderate พบว่าจากลักษณะของข้อมูล Operability User Reviews ซึ่งอธิบายถึงประสบการณ์การใช้งานแอปพลิเคชันของผู้ใช้งาน เช่น ระดับความยากหรือง่ายในการใช้งานและควบคุม โดยส่วนใหญ่ข้อมูล Operability User Reviews จะมีลักษณะเป็นการเล่าถึงสถานการณ์ต่าง ๆ ซึ่งต้องอาศัยการวิเคราะห์และทำความเข้าใจถึงสิ่งที่ผู้ใช้งานต้องการ ตัวอย่างเช่น Operability User Reviews จากแอปพลิเคชัน Messenger “now the user is required to search out the same stickers every time they use them.” ซึ่งเล่าถึงความไม่สะดวกในการใช้งานเมื่อผู้ใช้งานต้องค้นหาสติกเกอร์เดิมทุกครั้งที่ใช้งาน จากลักษณะของข้อมูลดังกล่าวทำให้แนวทางการสกัดข้อมูลของวิทยานิพนธ์ ซึ่งสกัดข้อมูลที่คาดว่าจะสำคัญบางส่วนจากประโยคบทวิจารณ์ของผู้ใช้งานโดยตรงเท่านั้น ไม่ได้คำนึงบริบททางความหมาย หรือสังเคราะห์ข้อมูลใหม่จากการวิเคราะห์ประโยคบทวิจารณ์ของผู้ใช้งาน ทำให้ข้อมูลที่สกัดได้จากบทวิจารณ์ของผู้ใช้งานไม่ใช่ใจความสำคัญที่ผู้ใช้งานต้องการสื่อความหมาย แม้ว่าบทวิจารณ์ของผู้ใช้งานจะให้ข้อมูลที่มีประโยชน์ต่อการพัฒนาและปรับปรุงแอปพลิเคชันก็ตาม ส่งผลให้ความต้องการซอฟต์แวร์ประเภท Operability ที่สร้างจากแนวทางของวิทยานิพนธ์ได้รับผลการประเมินเกณฑ์คุณภาพ Completeness และ Validity ที่ระดับ Low ในขณะที่ลักษณะของข้อมูล Functional User Reviews และ Availability User Reviews จะรายงานถึงสิ่งที่ต้องการโดยตรง ตัวอย่างเช่น Functional User Reviews จากแอปพลิเคชัน Wattpad “wattpad needs a dark mood.” ซึ่งเล่าถึงความต้องการอารมณ์มืดจากแอปพลิเคชัน Wattpad อย่างชัดเจน ทำให้แนวทางการสกัดข้อมูลของวิทยานิพนธ์สามารถสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งานได้ตรงกับที่ผู้ใช้งานต้องการสื่อความหมาย แต่เนื่องจากข้อจำกัดของแนวทางการสกัดข้อมูลของวิทยานิพนธ์ ทำให้ได้รับข้อมูลที่สำคัญไม่ครบถ้วนและบทวิจารณ์ของผู้ใช้งานเป็นข้อมูลในมุมมองของผู้ใช้งานที่ไม่เฉพาะเจาะจงมากพอสำหรับการสร้างความต้องการซอฟต์แวร์ ส่งผลให้ความต้องการซอฟต์แวร์ประเภท Functional และ Availability ที่สร้างจากแนวทางของวิทยานิพนธ์ได้รับผลการประเมินเกณฑ์คุณภาพ Completeness และ Validity ที่ระดับ Moderate

หมายเหตุ ผลการประเมินความต้องการซอฟต์แวร์ประเภท Functional, Availability, Operability สามารถเกิดความคลาดเคลื่อนได้จากความเข้าใจด้านภาษาและการตีความหมายประโยคความต้องการของผู้ประเมินด้วย รวมถึงการสุ่มคุณภาพของประโยคความต้องการที่นำมาประเมิน ซึ่งวิทยานิพนธ์พยายามลดความคลาดเคลื่อนของผลการประเมินให้น้อยที่สุดด้วยการอธิบายถึงวิธีการประเมิน พร้อมทั้งให้คู่มือประกอบการประเมิน

ตารางที่ 6-9 ตัวอย่างความต้องการซอฟต์แวร์และระดับของเกณฑ์คุณภาพที่ได้จากการประเมิน

เกณฑ์คุณภาพ	ระดับ	ประเภทความต้องการ	Review Pattern Code	Requirement Boilerplate Code	บทวิจารณ์ของผู้ใช้งาน	ความต้องการซอฟต์แวร์
Readability	High	Availability	RPA09	RBA04	the edit button[Target] is missing, so books cannot be deleted	The edit button[Target] shall be available for use.
Readability	Low	Functional	RPFN23	RBFN01	i think adding[Activity] the option to become[Target] a patron instead of paying for coins is a better option in my opinion	The system[Agent] shall be able to add[Activity] the option to become[Target].
Unambiguity, Completeness	High	Availability	RPA24	RBA05	whenever i click on a story, it cannot load[Activity] the description page[Target]	The system shall load[Activity] the description page[Target].
Unambiguity, Completeness	Low	Functional	RPFN07	RBFN05	all that is needed is a list[Target] of books in the library	The system[Agent] shall have a list[Target].
Validity	High	Operability	RPOP85	RBOP03	you cannot edit[Activity] an individual picture[Target] after you post it very inconvenient	The system[Agent] shall allow a user to be able to edit[Activity] an individual picture[Target].

ตารางที่ 6-9 ตัวอย่างความต้องการซอฟต์แวร์และระดับของเกณฑ์คุณภาพที่ได้จากการประเมิน (ต่อ)

เกณฑ์คุณภาพ	ระดับ	ประเภทความต้องการ	Review Pattern Code	Requirement Boilerplate Code	บทวิจารณ์ของผู้ใช้งาน	ความต้องการซอฟต์แวร์
Validity	Low	Operability	RPOP27	RBOP05	you really got to stop adding[Activity] ads in every chapter[Target] that am reading cause it really getting annoying and pissing me off	The system[Agent] shall add[Activity] ads in every chapter[Target].

บทที่ 7
การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันจากข้อมูลบทวิจารณ์ของผู้ใช้งานที่ไม่เคยเห็นมาก่อน

เฟสนี้คือการทดลองนำแนวทางวิจัยของวิทยานิพนธ์มาประยุกต์ใช้งาน โดยนำแนวทางที่ดีที่สุดของทุกเฟสมารวมกัน ประกอบด้วย 2 ขั้นตอนหลัก คือ 7.1) ขั้นตอนการเตรียมข้อมูล (Data Preparation) ซึ่งบทวิจารณ์ของผู้ใช้งานที่ไม่เคยเห็นมาก่อน (Unseen User Reviews) จะถูกจำแนกประเภทและตรวจสอบความซ้ำซ้อนกันของบทวิจารณ์ของผู้ใช้งานหากตรวจสอบแล้วพบว่าบทวิจารณ์ของผู้ใช้งานนั้นซ้ำซ้อนกันกับบทวิจารณ์ของผู้ใช้งานเดิมที่มีอยู่แล้ว บทวิจารณ์ของผู้ใช้งานนั้นจะถูกคัดออก บทวิจารณ์ของผู้ใช้งานที่คงเหลืออยู่จะถูกนำมาสร้างประโยคความต้องการ และ 7.2) ขั้นตอนการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์

วิทยานิพนธ์ทดลองพัฒนาโปรแกรมเบื้องต้นตามแนวทางขั้นตอนของวิทยานิพนธ์โดยใช้ภาษา Python ด้วยเครื่องมือ Google Colab Pro มีข้อมูลอินพุตของโปรแกรมซึ่งคือข้อมูล Unseen User Reviews เก็บอยู่ในรูปแบบไฟล์นามสกุล CSV และข้อมูลผลลัพธ์ของโปรแกรมคือประโยคความต้องการตามแต่ละประเภทของข้อมูลเก็บอยู่ในรูปแบบไฟล์นามสกุล CSV เช่นกัน สำหรับข้อมูลที่ใช้ทดสอบแนวทางของวิทยานิพนธ์เป็นข้อมูล Unseen User Reviews ที่เก็บรวบรวมมาด้วยวิธี Screen Scraping จาก 2 แอปพลิเคชันคือ Wattpad และ Messenger รายละเอียดจำนวนบทวิจารณ์ของผู้ใช้งานในแต่ละขั้นตอน ดังตารางที่ 7-1

ตารางที่ 7-1 จำนวนบทวิจารณ์ของผู้ใช้งานในแต่ละขั้นตอน

แอปพลิเคชัน	**ขั้นตอนเก็บรวบรวมข้อมูล**	**ขั้นตอน Data Preparation**		**ขั้นตอนการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชัน**
		ขั้นตอนการจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน	หลังผ่านขั้นตอนการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน	
Wattpad	6,562	6,543	674	24
Messenger	3,831	3,826	579	16
ทั้งหมด	10,393	10,369	1,253	40

7.1 Data Preparation

ขั้นตอนนี้คือการเตรียมข้อมูลบทวิจารณ์ของผู้ใช้งานก่อนนำไปสร้างประโยคความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์ในขั้นตอนถัดไป ประกอบด้วย 2 ขั้นตอน คือ

7.1.1) การจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน โดยจะจำแนกบทวิจารณ์ของผู้ใช้งานออกเป็นประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ด้วยโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพดีที่สุดสำหรับการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานที่ได้จากการทดลองของวิทยานิพนธ์ในบทที่ 4 และขั้นตอน 7.1.2) การตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน โดยใช้แนวทาง Pairwise Text Similarity ซึ่งเป็นแนวทางที่มีประสิทธิภาพดีที่สุดสำหรับการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนจากการทดลองของวิทยานิพนธ์ในบทที่ 5

7.1.1 การจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน

สำหรับโมเดลการเรียนรู้ของเครื่อง Feature และเทคนิคที่นำมาใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้งานแต่ละประเภท วิทยานิพนธ์เลือกโมเดลการเรียนรู้ของเครื่องที่มีประสิทธิภาพสูงที่สุดจากการทดลองในขั้นตอนที่ 4 มีรายละเอียดดังนี้

a) ประเภท Functional User Reviews ใช้อัลกอริทึมการเรียนรู้ของเครื่อง คือ Logistic Regression ซึ่งประกอบด้วย Feature คือ BoW, Sentiment, POS และใช้เทคนิค MinMaxScaler และ SVMSMOTE
b) ประเภท Availability User Reviews ใช้อัลกอริทึมการเรียนรู้ของเครื่อง คือ Random Forest ซึ่งประกอบด้วย Feature คือ TF-IDF, Stemming, n_tokens, Sentiment และใช้เทคนิค SVMSMOTE และ GridSearchCV มีการปรับค่าพารามิเตอร์ของโมเดลการเรียนรู้ ได้แก่ ค่า max_depth เท่ากับ 100, ค่า n_estimates เท่ากับ 1,000, ค่า criterion คือ gini
c) ประเภท Operability User Reviews ใช้อัลกอริทึมการเรียนรู้ของเครื่อง คือ Logistic Regression ซึ่งประกอบด้วย Feature คือ BoW, n_tokens, Sentiment, POS และใช้เทคนิค SVMSMOTE และ GridSearchCV มีการปรับค่าพารามิเตอร์ของโมเดลการเรียนรู้ ได้แก่ ค่า c เท่ากับ 10, ค่า penalty คือ L2

วิทยานิพนธ์ได้นำออกโมเดลการเรียนรู้ของเครื่องที่ได้จากการทดลองเป็นไฟล์นามสกุล SAV และได้นำเข้าโมเดลการเรียนรู้มาใช้จำแนกประเภทบทวิจารณ์ของผู้ใช้งาน โดยการใช้ไลบรารี Pickle [44] ของภาษา Python ตัวอย่าง Source Code ของวิธีการนำออกของโมเดลการเรียนรู้ของเครื่องดังรูปที่ 7-1 และวิธีการนำเข้าของโมเดลการเรียนรู้ของเครื่องดังรูปที่ 7-2

```
import pickle
#export machine learning model
filename = "{0}.sav".format(model_name)
file_path = "{0}{1}".format("/content/models/", filename)
pickle.dump(model, open(file_path, 'wb'))
```

รูปที่ 7-1 Source Code วิธีการนำออกของโมเดลการเรียนรู้ของเครื่องด้วยไลบรารี Pickle

```
import pickle
#import machine learning model
func_model_filname = "Functional_Logistic.sav"
with open(func_model_filname,'rb') as file:
  func_model = pickle.load(file)
```

รูปที่ 7-2 Source Code วิธีการนำเข้าของโมเดลการเรียนรู้ของเครื่องด้วยไลบรารี Pickle

Unseen User Reviews ที่ได้รวบรวมมา จะถูกนำมาทำ Data Cleaning และ Feature Extraction เช่นเดียวกันกับขั้นตอนการทดลองจำแนกประเภทบทวิจารณ์ของผู้ใช้งานของวิทยานิพนธ์ในบทที่ 4 ก่อนจะนำมาจำแนกประเภทบทวิจารณ์ของผู้ใช้งานด้วยโมเดลการเรียนรู้ของเครื่องตามรายละเอียดที่ได้กล่าวไว้ข้างต้น จำนวนบทวิจารณ์ของผู้ใช้งานหลังผ่านขั้นตอนการเตรียมข้อมูลบทวิจารณ์ของผู้ใช้งานดังตารางที่ 7-1 หากผลการทำนายประเภทของบทวิจารณ์ของผู้ใช้งานที่ได้จากโมเดลการเรียนรู้ของเครื่องสามารถจำแนกประเภทได้มากกว่า 1 ประเภท วิทยานิพนธ์พิจารณาเลือกประเภทบทวิจารณ์ของผู้ใช้งานจากค่าความน่าจะเป็น (Probability) ของการทำนายประเภทที่สูงที่สุดแทน ซึ่งบทวิจารณ์ของผู้ใช้งานจะสามารถถูกจัดอยู่ได้เพียงประเภทเดียวเท่านั้น หากบทวิจารณ์ของผู้ใช้งานไม่ได้ถูกจัดอยู่ในประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews จะถือว่าเป็นประเภท Others ซึ่งบทวิจารณ์ของผู้ใช้งานจะไม่ถูกนำมาประมวลผลต่อ ผลลัพธ์จากการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานอยู่ในรูปแบบไฟล์นามสกุล CSV ตัวอย่างดังรูปที่ 7-3 โดยนำมาใช้เป็นข้อมูลอินพุตในขั้นตอนการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกัน

	A	B	C	D
1	ID	SOURCE	TEXT	TARGET
2	1	Wattpad	I loved this app bec	OPERABILITY
3	2	Wattpad	The update only let	OTHERS
4	3	Wattpad	I was glad without t	OTHERS
5	4	Wattpad	The update makes s	OPERABILITY
6	5	Wattpad	Also, it can discoura	FUNCTIONAL
7	6	Wattpad	If they read the revi	FUNCTIONAL
8	7	Wattpad	It can simply stop th	OTHERS
9	8	Wattpad	Along with other W	OTHERS
10	9	Wattpad	Hi Im Yelling_Nothir	OTHERS
11	10	Wattpad	When I first got into	OTHERS
12	11	Wattpad	I was able to grow a	FUNCTIONAL
13	12	Wattpad	I learned through W	OTHERS
14	13	Wattpad	I learned communic	OTHERS
15	14	Wattpad	Now Im currently a	OTHERS

รูปที่ 7-3 ตัวอย่างบทวิจารณ์ของผู้ใช้งานหลังการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานในรูปแบบไฟล์นามสกุล CSV

หลังจากการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานแล้ว พบว่ามีจำนวนบทวิจารณ์ของผู้ใช้งานของประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ดังตารางที่ 7-2

ตารางที่ 7-2 จำนวนบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ของแอปพลิเคชัน Wattpad และ Messenger

แอปพลิเคชัน	Functional	Availability	Operability
Wattpad	1,606	275	1,542
Messenger	974	356	670
ทั้งหมด	2,580	631	2,212

7.1.2 การตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน

ในขั้นตอนนี้มีข้อมูลอินพุตคือบทวิจารณ์ของผู้ใช้งานที่ได้จำแนกประเภทแล้วจากขั้นตอน 7.1.1 วิทยานิพนธ์ใช้วิธี Pairwise Text Similarity ซึ่งเป็นวิธีที่มีประสิทธิภาพดีที่สุดสำหรับการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกันจากการทดลองในบทที่ 5 มีรายละเอียดดังนี้

บทวิจารณ์ของผู้ใช้งานถูกนำมาเตรียมข้อมูลเช่นเดียวกันกับการทดลองในบทที่ 5 ก่อนการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน วิทยานิพนธ์ทำ Representation ด้วย Lemmatization และโมเดล Bert-large-nil ที่ 300 Dimension สำหรับ Functional User Reviews และ Operability User Reviews มี Similarity Threshold ที่ 0.73 และ 0.60 ตามลำดับ สำหรับ

Availability User Reviews ถูกทำ Representation ด้วย Lemmatization และโมเดล Paraphrase-mini ที่ 300 Dimension และมี Similarity Threshold ที่ 0.60

หลังจากตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อนกันแล้ว บทวิจารณ์ของผู้ใช้งานที่ยังคงเหลืออยู่จะถูกนำมาสร้างประโยคความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์ โดยบทวิจารณ์ของผู้ใช้งานที่ยังคงเหลืออยู่จะถูกเก็บอยู่ในรูปแบบไฟล์นามสกุล CSV ซึ่งมีจำนวนบทวิจารณ์ของผู้ใช้งานที่คงเหลืออยู่ดังตารางที่ 7-1

7.2 การสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์

บทวิจารณ์ของผู้ใช้งานที่คงเหลืออยู่จากขั้นตอนที่ 7.1.2 ถูกนำมาสร้างประโยคความต้องการ ซึ่งคือข้อมูลอินพุตสำหรับในขั้นตอนนี้ โดยสร้างประโยคความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันด้วยวิธีเช่นเดียวกับการทดลองในบทที่ 6 ผลลัพธ์ประโยคความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันอยู่ในรูปแบบไฟล์นามสกุล CSV ดังรูปที่ 7-4

	A	B	C	D	E	F	G
	PATTERN	TEXT	BOILERPLATE_ID	REG	TARGET	SOURCE	ID
	RPFN2	this applic	RBFN04	The system shall allow a user to be able to use more customization.	FUNCTIONAL	Messenger	1515
	RPFN2	this applic	RBFN05	The system shall have more customization.	FUNCTIONAL	Messenger	1515
	RPFN22	i believe lo	RBFN01	The system shall be able to leave location services.	FUNCTIONAL	Messenger	3062
	RPFN22	i believe lo	RBFN06	The system shall leave location services.	FUNCTIONAL	Messenger	3062
	RPFN22	i seriously	RBFN01	The system shall be able to turn crap.	FUNCTIONAL	Messenger	3514
	RPFN22	i seriously	RBFN06	The system shall turn crap.	FUNCTIONAL	Messenger	3514
	RPFN23	it is not us	RBFN01	The system shall be able to add stuff.	FUNCTIONAL	Messenger	3323
	RPFN23	it is not us	RBFN05	The system shall have stuff.	FUNCTIONAL	Messenger	3323
	RPFN23	it is not us	RBFN06	The system shall add stuff.	FUNCTIONAL	Messenger	3323
	RPFN23	it would al	RBFN01	The system shall be able to add backgrounds.	FUNCTIONAL	Messenger	1517
	RPFN23	it would al	RBFN05	The system shall have backgrounds.	FUNCTIONAL	Messenger	1517
	RPFN23	it would al	RBFN06	The system shall add backgrounds.	FUNCTIONAL	Messenger	1517

รูปที่ 7-4 ตัวอย่างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันในรูปแบบไฟล์นามสกุล CSV

จำนวน Unseen User Reviews ที่ถูกนำมาสร้างประโยคความต้องการมีจำนวนทั้งหมด 40 บทวิจารณ์ของผู้ใช้งานแบ่งตามแอปพลิเคชัน ดังตารางที่ 7-1 จากแอปพลิเคชัน Wattpad จำนวน 24 บทวิจารณ์ของผู้ใช้งาน และแอปพลิเคชัน Messenger จำนวน 16 บทวิจารณ์ของผู้ใช้งาน โดยความต้องการประเภท Functional, Availability, Operability ที่ได้จาก Unseen User Reviews มีรายละเอียดของข้อมูลแต่ละประเภทดังต่อไปนี้

1) Unseen User Reviews จากแอปพลิเคชัน Wattpad

   a) ความต้องการประเภท Functional พบว่ามีบทวิจารณ์ของผู้ใช้งานจำนวนทั้งหมด 12 บทวิจารณ์ของผู้ใช้งานถูกจับคู่กับ 7 Functional User Review Patterns ที่กำหนดไว้ คือ RPFN01, RPFN22, RPFN23, RPFN28, RPFN29, RPFN59, RPFN83 ได้ประโยคความต้องการจาก 5 รูปแบบของ Functional

Requirement Boilerplates ได้แก่ RBFN01, RBFN02, RBFN04, RBFN05, RBFN06

b) ความต้องการประเภท Availability พบว่ามีบทวิจารณ์ของผู้ใช้งานจำนวนทั้งหมด 12 บทวิจารณ์ของผู้ใช้งานถูกจับคู่กับ 4 Availability User Review Patterns ที่กำหนดไว้ คือ RPA10, RPA14, RPA20, RPA24 ได้ประโยคความต้องการจาก 3 รูปแบบของ Availability Requirement Boilerplates ได้แก่ RBA02, RBA04, RBA05

c) ความต้องการประเภท Operability พบว่าไม่มี Unseen User Reviews ประเภท Operability ที่ถูกจับคู่กับ Operability User Review Patterns ที่กำหนดไว้

2) Unseen User Reviews จากแอปพลิเคชัน Messenger

a) ความต้องการประเภท Functional พบว่ามีบทวิจารณ์ของผู้ใช้งานจำนวนทั้งหมด 8 บทวิจารณ์ของผู้ใช้งานถูกจับคู่กับ 5 Functional User Review Patterns ที่กำหนดไว้ คือ RPFN02, RPFN22, RPFN23, RPFN60, RPFN83 ได้ประโยคความต้องการจาก 4 รูปแบบของ Functional Requirement Boilerplates ได้แก่ RBFN01, RBFN04, RBFN05, RBFN06

b) ความต้องการประเภท Availability พบว่ามีบทวิจารณ์ของผู้ใช้งานจำนวนทั้งหมด 7 บทวิจารณ์ของผู้ใช้งานถูกจับคู่กับ 3 Availability User Review Patterns ที่กำหนดไว้ คือ RPA12, RPA20, RPA24 ได้ประโยคความต้องการจาก 3 รูปแบบของ Availability Requirement Boilerplates ได้แก่ RBA02, RBA03, RBA05

c) ความต้องการประเภท Operability พบว่ามีบทวิจารณ์ของผู้ใช้งานจำนวนทั้งหมด 1 บทวิจารณ์ของผู้ใช้งานถูกจับคู่กับ 1 Operability User Review Patterns ที่กำหนดไว้ คือ RPOP85 ได้ประโยคความต้องการจาก 3 รูปแบบของ Operability Requirement Boilerplates ได้แก่ RBOP02, RBOP03, RBOP05

จากการวิเคราะห์แนวทางการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์โดยอัตโนมัติจากข้อมูลบทวิจารณ์ของผู้ใช้งานเมื่อนำแนวทางที่ดีที่สุดของวิทยานิพนธ์มารวมกันพบว่า มีโอกาสที่แนวทางอัตโนมัติจะจำแนกประเภทของบทวิจารณ์ของผู้ใช้งานผิดพลาดได้ เช่น หากนำบทวิจารณ์ของผู้ใช้งานซึ่งอันที่จริงเป็นประเภท Others แต่ได้รับการทำนายเป็นประเภท

Functional, Availability, หรือ Operability ซึ่งบทวิจารณ์ของผู้ใช้งานนี้อาจจะไม่ให้ข้อมูลที่สำคัญสำหรับการปรับปรุงและพัฒนาแอปพลิเคชัน การสร้างความต้องการซอฟต์แวร์จากบทวิจารณ์ของผู้ใช้งานที่ทำนายผิดประเภทหรือมีเนื้อหาที่จำเป็นไม่เพียงพอ จะทำให้ทีมนักพัฒนาไม่สามารถนำความต้องการซอฟต์แวร์ที่สร้างขึ้นมาไปใช้ปรับปรุงและพัฒนาแอปพลิเคชันต่อไปได้

นอกจากนี้การสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งานเพื่อนำมาสร้างประโยคความต้องการด้วยวิธีการจับคู่บทวิจารณ์ของผู้ใช้งานกับ User Review Patterns รูปแบบต่าง ๆ นั้นพบว่าความต้องการซอฟต์แวร์ที่ได้มีจำนวนน้อย จากตารางที่ 7-1 จำนวนบทวิจารณ์ของผู้ใช้งานหลังผ่านขั้นตอนการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่ซ้ำซ้อน ซึ่งคือข้อมูลอินพุตของขั้นตอนการสร้างความต้องการ มีจำนวนมากถึง 1,253 บทวิจารณ์ของผู้ใช้งาน แต่ส่วนมากจะยังไม่สามารถจับคู่ได้กับ User Review Patterns ที่มีอยู่ โดยมีเพียง 40 บทวิจารณ์ของผู้ใช้งานเท่านั้นที่สามารถจับคู่ได้และถูกนำมาสร้างประโยคความต้องการซอฟต์แวร์ โดยมีสาเหตุมาจากแนวทางการสร้างความต้องการซอฟต์แวร์ของวิทยานิพนธ์ โดยวิทยานิพนธ์มีแนวทางการสร้างความต้องการจากการสกัดข้อมูลที่คาดว่าจะสำคัญต่อการสร้างความต้องการโดยอาศัยการจับคู่กันระหว่างบทวิจารณ์ของผู้ใช้งานกับ User Review Patterns ที่กำหนดไว้และสกัดข้อมูลที่สำคัญตามตำแหน่งที่กำหนดไว้บน User Review Patterns โดยนำข้อมูลที่สกัดได้มาสร้างความต้องการซอฟต์แวร์ด้วยการใช้ Requirement Boilerplates หากว่าบทวิจารณ์ของผู้ใช้งานนั้นได้รับการจับคู่กับ User Review Patterns ที่มีอยู่ บทวิจารณ์ของผู้ใช้งานนั้นจะถูกนำมาสร้างความต้องการซอฟต์แวร์ แต่หากบทวิจารณ์ของผู้ใช้งานนั้นไม่ถูกจับคู่กับ User Review Patterns จะไม่ถูกนำมาสร้างความต้องการซอฟต์แวร์ แม้ว่าบทวิจารณ์ของผู้ใช้งานนั้นจะให้ข้อมูลที่อาจเป็นประโยชน์ต่อการบำรุงรักษาซอฟต์แวร์ก็ตาม ซึ่งจากโครงสร้างไวยากรณ์ทางภาษาของ User Review Patterns ที่วิทยานิพนธ์ได้กำหนดไว้มีความซับซ้อนของโครงสร้างทางไวยากรณ์ไม่เพียงพอต่อแบบรูปการเขียนบทวิจารณ์ของผู้ใช้งานทำให้แม้จะมีข้อมูลบทวิจารณ์ของผู้ใช้งานจำนวนมากแต่ได้รับการจับคู่กับ User Review Patterns ในจำนวนน้อย ซึ่งแนวทางในการปรับปรุงประสิทธิภาพเพื่อเพิ่มจำนวนการจับคู่กันระหว่างบทวิจารณ์ของผู้ใช้งานกับ User Review Patterns คือการเพิ่มความซับซ้อนให้กับโครงสร้างทางไวยากรณ์ของ User Review Patterns ตัวอย่างเช่น การเพิ่มความซับซ้อนให้กับโครงสร้างทางไวยากรณ์ของ Non-terminals การเพิ่มรายการ Key words ที่ระบุใน User Review Patterns เพื่อให้รองรับแบบรูปการเขียนบทวิจารณ์ของผู้ใช้งานได้มากขึ้น และการเพิ่มจำนวน User Review Patterns ให้มากขึ้น

บทที่ 8
สรุปผลการวิจัยและข้อเสนอแนะ

8.1 สรุปผลการวิจัย

วิทยานิพนธ์นำเสนอแนวทางการสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์โดยอัตโนมัติจากข้อมูลบทวิจารณ์ของผู้ใช้งานโมไบล์แอปพลิเคชันจากแอปสโตร์และเพลย์สโตร์ ซึ่งวิทยานิพนธ์มีจุดมุ่งหมายเพื่อช่วยจัดหาข้อมูลที่เป็นประโยชน์ให้แก่ทีมนักพัฒนาในกระบวนการปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชัน ประกอบด้วย 3 ขั้นตอนหลัก เริ่มจากการจำแนกประเภทบทวิจารณ์ของผู้ใช้งานออกเป็นประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews โดยใช้แนวทาง Text Classification ขั้นตอนถัดมา คือ การตรวจสอบความซ้ำซ้อนกันของบทวิจารณ์ของผู้ใช้งานด้วยแนวทาง Text Similarity หากพบว่าบทวิจารณ์ของผู้ใช้งานซ้ำซ้อนกันกับข้อมูลที่มีอยู่แล้ว บทวิจารณ์ของผู้ใช้งานนั้นจะถูกคัดออก ในขั้นตอนสุดท้ายบทวิจารณ์ของผู้ใช้งานที่ยังคงเหลืออยู่จะถูกนำมาสร้างความต้องการเชิงฟังก์ชันและที่ไม่ใช่เชิงฟังก์ชันของซอฟต์แวร์จากการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน โดยการใช้ User Review Patterns และสร้างประโยคความต้องการด้วยการเติมข้อมูลลงใน Requirement Boilerplates ที่วิทยานิพนธ์ได้เตรียมไว้ นอกจากนี้วิทยานิพนธ์ได้ประเมินประสิทธิภาพของแต่ละแนวทางที่นำเสนอ รวมถึงประเมินคุณภาพความต้องการซอฟต์แวร์ที่สร้างจากแนวทางของวิทยานิพนธ์ ทั้งหมด 4 เกณฑ์คุณภาพ ได้แก่ Readability, Unambiguity, Completeness, Validity

ผลการทดลองจากแนวทางการวิจัยของวิทยานิพนธ์ พบว่าอัลกอริทึมการเรียนรู้ Logistic Regression มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews และ Operability User Reviews โดยมีค่า F1-Score อยู่ที่ 75.4% และ 62.1% ตามลำดับ อัลกอริทึมการเรียนรู้ Random Forest มีประสิทธิภาพสูงที่สุดสำหรับการจำแนกบทวิจารณ์ของผู้ใช้งานประเภท Availability User Reviews โดยมีค่า F1-Score อยู่ที่ 73.5% ซึ่ง Feature ที่มีความสำคัญต่อการจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน ได้แก่ BoW, TF-IDF, n_tokens, POS, Sentiment สำหรับการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งาน แนวทาง Pairwise Text Similarity มีประสิทธิภาพดีกว่าแนวทาง Clustering โดยมีค่าเฉลี่ย Accuracy คือ 0.72, 0.85, 0.67 สำหรับบทวิจารณ์ของผู้ใช้งานประเภท Functional User Reviews, Availability User Reviews, Operability User Reviews ตามลำดับ ในส่วนของผลการประเมินความต้องการซอฟต์แวร์ประเภท Functional, Availability, Operability ที่สร้างจากแนวทางของวิทยานิพนธ์ มีค่าเฉลี่ยผลการประเมินของเกณฑ์คุณภาพ Readability ที่ระดับ High

แสดงถึงความต้องการซอฟต์แวร์มีความสามารถในการอ่านและสามารถเข้าใจได้ ในส่วนของเกณฑ์คุณภาพ Unambiguity, Completeness, Validity อยู่ในระดับ Moderate ซึ่งความต้องการซอฟต์แวร์ค่อนข้างมีข้อมูลที่สมบูรณ์ ค่อนข้างไม่กำกวมและตรงตามเจตนาของผู้ใช้งานตามที่ระบุในบทวิจารณ์ของผู้ใช้งาน ยกเว้นความต้องการซอฟต์แวร์ประเภท Operability ที่ได้ระดับ Low ในเกณฑ์คุณภาพ Completeness และ Validity

8.2 ข้อจำกัดของงานวิจัย

1) บทวิจารณ์ของผู้ใช้งานรองรับเพียงภาษาอังกฤษเท่านั้น
2) จำนวนบทวิจารณ์ของผู้ใช้งานที่ใช้ฝึกสอนโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกประเภทบทวิจารณ์ของผู้ใช้งานมีจำนวนไม่มากนัก
3) แนวทางการตรวจสอบความซ้ำซ้อนกันของบทวิจารณ์ของผู้ใช้งานยังไม่แม่นยำในการตรวจสอบบทวิจารณ์ของผู้ใช้งานที่มีใจความสำคัญทางความหมายที่ซ้ำซ้อนกัน
4) แนวทางการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งานเพื่อนำมาสร้างประโยคความต้องการซอฟต์แวร์ยังไม่สามารถสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งานได้ครบถ้วนและยังไม่เข้าใจถึงบริบททางความหมายของบทวิจารณ์ของผู้ใช้งาน ทำให้ในบางครั้งความต้องการซอฟต์แวร์ที่ได้ไม่ตรงตามเจตนาของผู้ใช้งาน นอกจากนี้ User Review Patterns ที่วิทยานิพนธ์นำเสนอยังไม่ครอบคลุมกับรูปแบบการเขียนบทวิจารณ์ของผู้ใช้งานมากพอ ทำให้ไม่สามารถจับคู่กับบทวิจารณ์ของผู้ใช้งานได้ และทำให้ความต้องการซอฟต์แวร์ที่ได้รับมีจำนวนน้อยกว่าจำนวนข้อมูลบทวิจารณ์ของผู้ใช้งานที่เป็นข้อมูลอินพุตอยู่มาก
5) บทวิจารณ์ของผู้ใช้งานเป็นข้อมูลในมุมมองของผู้ใช้งาน ซึ่งมีข้อมูลที่สำคัญไม่เพียงพอต่อการนำมาสร้างความต้องการซอฟต์แวร์ที่ชัดเจนและเฉพาะเจาะจงมากพอสำหรับปรับปรุงและวิวัฒนาการโมไบล์แอปพลิเคชัน ยังคงต้องอาศัยทีมนักพัฒนาพิจารณาถึงข้อมูลที่สำคัญบางส่วนเพิ่มเติม
6) วิทยานิพนธ์พิจารณาบทวิจารณ์ของผู้ใช้งานเพียงประโยคเดียวเท่านั้น ไม่ได้นำประโยคข้างเคียงหรือบทวิจารณ์ของผู้ใช้งานที่มีบริบทของข้อมูลใกล้เคียงกันมาประมวลผลด้วย ซึ่งทำให้ได้รับข้อมูลที่มีประโยชน์ลดลง

8.3 ข้อเสนอแนะ

1) ส่วนการจำแนกประเภทบทวิจารณ์ของผู้ใช้งาน
   a) เพิ่มจำนวน Training Set และ Test Set เพื่อเพิ่มประสิทธิภาพการเรียนรู้ของโมเดลการเรียนรู้ของเครื่อง

b) ทำ Representation บทวิจารณ์ของผู้ใช้งานด้วย SBERT เพื่อให้โมเดลการเรียนรู้ของเครื่องสามารถเข้าใจบริบทของคำได้ดีขึ้นและเพิ่มประสิทธิภาพการเรียนรู้ของโมเดลการเรียนรู้ของเครื่อง

2) ส่วนการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งาน
   a) ทำ Text Summarization ก่อนการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งานเพื่อเพิ่มความแม่นยำให้กับบทวิจารณ์ของผู้ใช้งานที่มีใจความสำคัญเหมือนกันแต่มีรูปแบบการเขียนบทวิจารณ์ของผู้ใช้งานค่อนข้างแตกต่างกัน
   b) ทดลองใช้ตัววัดค่าความคล้ายคลึงกันทางความหมายระหว่างข้อมูลตัวอื่น ๆ เช่น Jaccard Index
   c) สร้างคู่มือแนวทางการติดป้ายความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้งาน
3) ส่วนการสร้างความต้องการ
   a) เพิ่มจำนวน User Review Patterns สำหรับการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน
   b) ปรับปรุงแนวทางการตัดประโยค เช่น เพิ่มการตัดประโยคด้วยจุลภาคและเลขหัวข้อเป็นต้น
   c) ปรับปรุงแนวทางการสกัดข้อมูลที่สำคัญจากบทวิจารณ์ของผู้ใช้งาน เช่น เพิ่มความซับซ้อนให้แก่โครงสร้างทางไวยากรณ์ของ User Review Pattern เพิ่มรายการ Key words ใน User Review Patterns รวมถึงนำบทวิจารณ์ของผู้ใช้งานประโยคข้างเคียงหรือบทวิจารณ์ของผู้ใช้งานที่มีบริบทใกล้เคียงกันมาประมวลผลด้วย
   d) ปรับปรุงความสอดคล้องของไวยากรณ์ทางภาษาหลังจากสร้างประโยคความต้องการซอฟต์แวร์แล้ว ตัวอย่างเช่น เติม s, es ให้กับประธาน เปลี่ยนคำกริยา is, am, are ตามประธานของประโยค เป็นต้น
   e) สร้างประโยคความต้องการซอฟต์แวร์จากแนวทางอื่น เช่น แนวทางการสร้างประโยคด้วยอัลกอริทึม Deep Learning

บรรณานุกรม

1. Panichella, S., et al., *How can i improve my app? classifying user reviews for software maintenance and evolution*, in 2015 *IEEE international conference on software maintenance and evolution (ICSME)*. 2015, IEEE: Bremen, Germany. p. 281-290.
2. 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, IEEE: Ottawa, ON, Canada. p. 116-125.
3. Lu, M. and P. Liang, *Automatic classification of non-functional requirements from augmented app user reviews*, in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. 2017: Karlskrona, Sweden. p. 344-353.
4. 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, ICSE* 2014. 2014: Hyderabad India. p. 767-778.
5. Villarroel, L., et al., *Release planning of mobile apps based on user reviews*, in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. 2016: Austin, TX, USA. p. 14-24.
6. Mitchell, T.M., *Machine Learning*. 1997: McGraw-Hill. 414.
7. Jurafsky, D. and J.H. Martin, *Speech and language processing (draft)*. 3 ed. Available from: https://web.stanford.edu/~jurafsky/slp3. 2021.
8. Dick, J., E. Hull, and K. Jackson, *Requirements Engineering*. 4 ed. 2017: Springer International Publishing. 239.
9. *ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering*, in *ISO/IEC/IEEE* 29148:2018(*E*). 2018. p. 1-104.
10. *IEEE Standard Glossary of Software Engineering Terminology*, in *IEEE Std* 610.12-1990. 1990. p. 1-84.

11. Bourque, P. and R.E. Fairley, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version* 3.0. 2014: IEEE Computer Society Press.
12. *ISO/IEC* 25010 : 2011 *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. 2013.
13. Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Vol. 2. 2009: Springer-Verlag New York. 745.
14. Bengfort, B., R. Bilbro, and T. Ojeda, *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. 2018: O'Reilly Media, Inc.
15. Han, J., M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. 3 ed. 2011: Morgan Kaufmann Publishers Inc.
16. Reimers, N. and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks.* arXiv preprint arXiv:1908.10084, 2019.
17. Devlin, J., et al., *Bert: Pre-training of deep bidirectional transformers for language understanding.* arXiv preprint arXiv:1810.04805, 2018.
18. Chilakapati, A. *BoW to BERT*. 2019 [cited 2021 24 Sep 2021]; Available from: https://xplordat.com/2019/09/23/bow-to-bert/.
19. Reimers, N. *Pretrained Models*. 2021 [cited 2021 24 Sep 2021]; Available from: https://www.sbert.net/docs/pretrained_models.html.
20. Reimers, N. *SentenceTransformers Documentation*. 2021 [cited 2021 24 Sep 2021]; Available from: https://www.sbert.net/.
21. Aysolmaz, B., et al., *A semi-automated approach for generating natural language requirements documents based on business process models.* Information and Software Technology, 2018. 93: p. 14-29.
22. Georgiades, M.G. and A.S. Andreou, *Automatic generation of a software requirements specification (SRS) document*, in 2010 10*th International Conference on Intelligent Systems Design and Applications*. 2010, IEEE: Cairo, Egypt. p. 1095-1100.
23. facundoolano. *app-store-scraper*. [cited 2021 24 Sep 2021]; Available from: https://www.npmjs.com/package/app-store-scraper.

24. facundoolano. *google-play-scraper*. [cited 2021 24 Sep 2021]; Available from: https://www.npmjs.com/package/google-play-scraper.
25. bsolomon1124. *demoji* 1.1.0. 30 Aug 2021 [cited 2021 24 Sep 2021]; Available from: https://pypi.org/project/demoji/#description.
26. neelindresh. *ContractedText.py*. 2018 [cited 2021 24 Sep 2021]; Available from: https://gist.github.com/neelindresh/39b3b4c3113d30a6e796697ff9e5fc12.
27. Kleinschmdit, T. *gingerit* 0.8.2. 6 Jul 2021 [cited 2021 24 Sep 2021]; Available from: https://pypi.org/project/gingerit/.
28. Řehůřek, R. *gensim.parsing.preprocessing – Functions to preprocess raw text*. [cited 2021 24 Sep 2021]; Available from: https://radimrehurek.com/gensim/parsing/preprocessing.html#gensim.parsing.preprocessing.remove_stopwords.
29. Hutto, C. and E. Gilbert, *Vader: A parsimonious rule-based model for sentiment analysis of social media text*, in *Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media*. 2014. p. 216-225.
30. Nguyen, H.M., E.W. Cooper, and K. Kamei, *Borderline over-sampling for imbalanced data classification.* International Journal of Knowledge Engineering and Soft Data Paradigms (IJKESDP), 2011. 3: p. 4-21.
31. Forman, G., *An extensive empirical study of feature selection metrics for text classification.* Journal of Machine Learning Research (JMLR), 2003. 3: p. 1289-1305.
32. Müller, A.C. and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. 2016: O'Reilly Media, Inc. 392.
33. Binkhonain, M. and L. Zhao, *A review of machine learning algorithms for identification and classification of non-functional requirements.* Expert Systems with Applications: X, 2019. 1.
34. *scikit-learn machine learning in Python*. [cited 2021 24 Sep 2021]; Available from: https://scikit-learn.org/stable/.
35. *Natural Language Toolkit (NLTK)*. 20 Sep 2021 [cited 2021 24 Sep 2021]; Available from: https://www.nltk.org/.

36. *Scipy.org*. [cited 2021 24 Sep 2021]; Available from: https://www.scipy.org/.
37. Banerjee, A. and R.N. Dave, *Validating clusters using the Hopkins statistic*, in 2004 *IEEE International Conference on Fuzzy Systems*. 2004. p. 149-153.
38. Wilson, B., *Chapter* 1: *Clustering for dataset exploration in Unsupervised Learning in Python Course*. 2020, Datacamp.
39. Oghbaie, M. and M.M. Zanjireh, *Pairwise document similarity measure based on present term set.* Journal of Big Data, 2018. 5(1): p. 52.
40. Rekabsaz, N., M. Lupu, and A. Hanbury, *Exploration of a threshold for similarity based on uncertainty in word embedding*, in 39*th European Conference on Information Retrieval Research (ECIR)*. 2017, Springer, Cham. p. 396-409.
41. Honnibal, M. and I. Montani. *Rule-based matching*. [cited 2021 24 Sep 2021]; Available from: https://spacy.io/usage/rule-based-matching#matcher.
42. Fatwanto, A., *Natural language requirements specification analysis using Part-of-Speech Tagging*, in *Second International Conference on Future Generation Communication Technologies (FGCT* 2013). 2013, IEEE: London, UK. p. 98-102.
43. Collins Dictionary [cited 2021 24 Sep 2021]; Available from: https://www.collinsdictionary.com/.
44. *pickle — Python object serialization*. [cited 2021 24 Sep 2021]; Available from: https://docs.python.org/3/library/pickle.html.

ประวัติผู้เขียน

ชื่อ-สกุล	ธนัชชา พันธ์ธรรม
วัน เดือน ปี เกิด	12 พฤศจิกายน 2536
สถานที่เกิด	กรุงเทพมหานคร, ประเทศไทย
วุฒิการศึกษา	ปีการศึกษา 2557 หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ จากคณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์ ปีการศึกษา 2560 เข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ที่อยู่ปัจจุบัน	144 ซอยพหลโยธิน 59 แยก 1 แขวงอนุสาวรีย์ เขตบางเขน กรุงเทพมหานคร 10220
ผลงานตีพิมพ์	ปี พ.ศ. 2564 "Generating Functional Requirements Based on Classification of Mobile Application User Reviews" โดย ธนัชชา พันธ์ธรรม และ รศ.ดร. ทวิตีย์ เสนีวงศ์ ณ อยุธยา ในงานประชุมวิชาการ 2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA) ระหว่างวันที่ 20-22 มิถุนายน พ.ศ. 2564 ณ เมืองคานาซาว่า ประเทศญี่ปุ่น
รางวัลที่ได้รับ	ปี พ.ศ. 2557 รางวัลชมเชยจากการแข่งขันพัฒนาเว็บแอปพลิเคชัน โครงการ SCB YOUNG TALENT ครั้งที่ 1 จัดโดยธนาคารไทยพาณิชย์