การจำแนกประเภทบทวิจารณ์ของผู้ใช้โมไบล์แอปพลิเคชันเพื่อการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา

นายกิตติศักดิ์ เพชรรุ่งนภา

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2561

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

Mr. Kittisak Phetrungnapha

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

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

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับสารนิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

คณะกรรมการสอบสารนิพนธ์

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

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

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

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

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2561 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6070908021 : MAJOR SOFTWARE ENGINEERING

KEYWORD:

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

Field of Study:	Software Engineering	Student's Signature .....
Academic Year:	2018	Advisor's Signature .....

กิตติกรรมประกาศ

กิตติศักดิ์ เพชรรุ่งนภา

สารบัญ

หน้า

ข้อเสนอโครงงานมหาบัณฑิต

ชื่อเรื่อง

ภาษาไทย การจำแนกประเภทบทวิจารณ์ของผู้ใช้โมไบล์แอปพลิเคชันเพื่อการสร้างทิคเก็ตสำหรับระบบติดตามปัญหา

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

1. ที่มาและความสำคัญของปัญหา

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

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

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

จากปัญหาดังกล่าวข้างต้น ทางผู้วิจัยจึงมีแนวคิดที่จะนำทฤษฎี ความรู้ต่าง ๆ ในศาสตร์ทางด้านปัญญาประดิษฐ์ [6] มาช่วยแก้ปัญหา เช่น การทำเหมืองข้อมูล [7] การเรียนรู้ของเครื่อง [8] เป็นต้น มาช่วยในการ

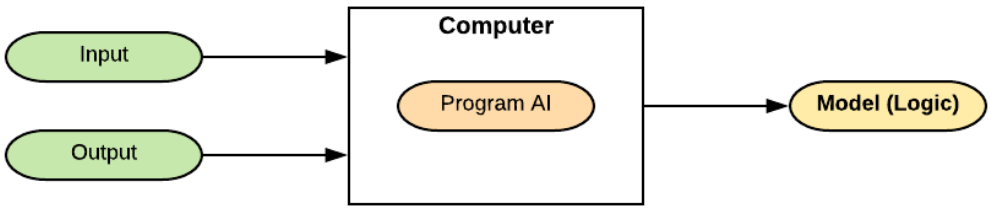
เก็บรวบรวม วิเคราะห์ และจำแนกประเภทบทวิจารณ์ของผู้ใช้ดังกล่าวว่าเป็นประเภทใด เช่น Feature Requests, Bug Reports, User Experience Improvement หรืออื่น ๆ เป็นต้น เพื่อช่วยลดระยะเวลาที่ต้องใช้คนมานั่งทำเอง ซึ่งหลังจากได้โมเดลการเรียนรู้แล้ว ทางผู้วิจัยยังมีแนวคิดที่จะสร้างเครื่องมือที่ช่วยในการสร้างทิคเก็ตบนระบบติดตามปัญหาซึ่งในโครงงานมหาบัณฑิตนี้หมายถึง ซอฟต์แวร์จิรา (Jira) โดยจะมีการแยกประเภทของทิคเก็ตตามประเภทของบทวิจารณ์ของผู้ใช้ ช่วยให้ลดระยะเวลาของงานที่ต้องทำในส่วนนี้ของการพัฒนาซอฟต์แวร์แบบเอไจล์ลง และช่วยสนับสนุนให้บริษัทพัฒนาซอฟต์แวร์และนักพัฒนาอิสระในการแก้ไขจุดบกพร่องต่าง ๆ ที่ผู้ใช้งานพบ รวมไปถึงการพัฒนาฟังก์ชันการทำงานใหม่ ๆ ที่ตอบสนองต่อความต้องการของผู้ใช้งาน ก่อนที่จะทำการอัปเดตเวอร์ชันของแอปพลิเคชันให้ผู้ใช้งานต่อไป

2. ทฤษฎีที่เกี่ยวข้อง

2.1 Supervised Learning [8]

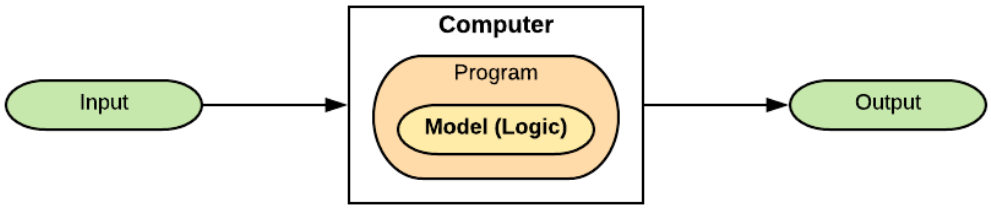
Supervised Learning หรือการเรียนรู้แบบมีผู้สอน เป็นศาสตร์หนึ่งทางด้านการเรียนรู้ของเครื่อง (Machine Learning) ที่อยู่ภายใต้ปัญญาประดิษฐ์ (Artificial Intelligence) โดยมีมานานตั้งแต่ช่วงปี 1959 แล้ว แต่ยังไม่ได้รับความนิยมเนื่องจากคอมพิวเตอร์ในสมัยนั้นมีกำลังในการประมวลผลต่ำ การคอมไพล์เพื่อให้ได้ผลลัพธ์ออกมาใช้เวลานาน

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



ภาพที่ 2.1 กระบวนการเทรนเพื่อให้ได้โมเดลที่ต้องการ

ขั้นตอนต่อมาก็คือการนำโมเดลที่ได้มาทำการทดสอบประสิทธิภาพ อธิบายโดยอิงจากภาพที่ 2.2 ในบริบทของการทำนายว่าภาพสัตว์ใด ๆ เป็นภาพแมวหรือไม่ ได้คือ ทำการนำภาพสัตว์หลากหลายประเภทมีทั้งแมวและไม่ใช่แมวใส่เข้าไปในโปรแกรมที่มีโมเดลการจำแนกประเภทอยู่ แล้วดูว่าคอมพิวเตอร์สามารถทำนายว่าเป็นภาพแมวหรือไม่ใช่ภาพแมวได้ถูกต้องหรือไม่



ภาพที่ 2.2 การนำโมเดลที่ผ่านการเรียนรู้แล้วมาลองใช้งานจริง

2.2 Naïve Bayes [9]

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

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

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

P(A $\bigcap$ B) คือ ค่า Joint Probability หรือค่าความน่าจะเป็นที่เหตุการณ์ A และเหตุการณ์ B เกิดขึ้นร่วมกัน

P(B) คือ ค่าความน่าจะเป็นที่เหตุการณ์ B เกิดขึ้น

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

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

จากทั้งสองสมการจะเห็นว่ามีค่า P(A $\bigcap$ B) ที่เหมือนกันอยู่ ดังนั้นเราสามารถเขียนสมการของ P(A $\bigcap$ B) ได้เป็นดังสมการ (3) และสมการ (4)

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

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

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

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

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

Likelihood หรือ P(A|C) คือ ค่าความน่าจะเป็นที่ข้อมูล Training Data ที่มีคลาส C และมีแอตทริบิวต์ A โดยที่ A = $a_1 \bigcap a_2 \ldots \bigcap a_m$ โดยที่ M คือจำนวนแอตทริบิวต์ใน Training Data

Prior probability หรือ P(C) คือ ค่าความน่าจะเป็นของคลาส C

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

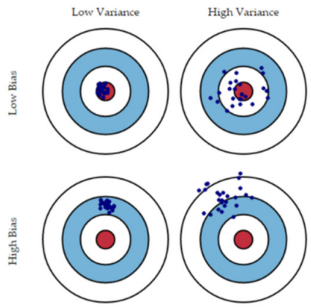
$$P(A \mid C) = P(a_1 \mid C) \times P(a_2 \mid C) \times \ldots \times P(a_m \mid C) \qquad (6)$$

2.3 Unsupervised Learning [8]

Unsupervised Learning นั้นตรงกันข้ามกับ Supervised Learning กล่าวคือเป็นการเรียนรู้แบบที่ไม่มีการระบุผล (Target Variable) ที่ต้องการไว้ก่อน แต่ให้คอมพิวเตอร์หาความสัมพันธ์จากข้อมูลเอาเอง จึงกล่าวได้ว่าการเรียนรู้ประเภทนี้เป็นการเรียนรู้แบบไม่มีผู้สอน ตัวอย่างของ Unsupervised Learning เช่น การแบ่งกลุ่มข้อมูล (Clustering) กระบวนการแบ่งกลุ่มข้อมูลนี้เป็นการจัดสิ่งของต่าง ๆ ให้อยู่ในกลุ่มที่เหมาะสม โดยของในกลุ่มเดียวกันจะคล้ายกัน และแตกต่างจากของในกลุ่มอื่น Clustering ต่างจาก Classification ตรงที่ Classification นั้นจะรู้ประเภทกลุ่มล่วงหน้า แต่ Clustering จะไม่รู้กลุ่มล่วงหน้า ตัวอย่างของ Clustering เช่น การจัดกลุ่มของเอกสารที่มีลักษณะคล้ายกันให้อยู่ด้วยกัน โดยที่ไม่จำเป็นต้องรู้ว่าแต่ละกลุ่มเอกสารนั้นเกี่ยวข้องกับเรื่องอะไร

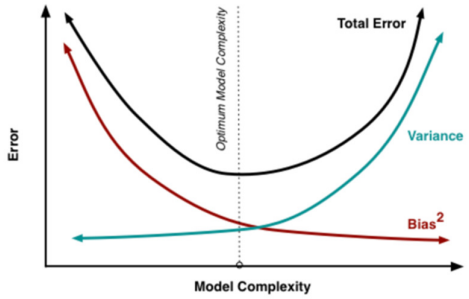
2.4 Ensemble Learning [10]

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



ภาพที่ 2.4 ผลลัพธ์การทำนายของโมเดล เมื่อมีค่า Bias และ Variance แตกต่างกันออกไป โดยจุดสีแดงคือค่าจริง จุดน้ำเงินคือค่าที่ได้จากการทำนายของโมเดล [10]

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



ภาพที 2.5 ความสัมพันธ์ระหว่าง Model Complexity และค่า Error ที่เกิดขึ้นที่เกี่ยวข้องกับค่า Bias และ Variance [10]

2.5 Jira Software

Jira Software [11] คือโปรแกรมติดตามปัญหา และจัดการโครงการพัฒนาซอฟต์แวร์ที่ใช้หลักการเอไจล์จัดทำขึ้นเพื่อใช้ในการแก้ไขจุดบกพร่องของโปรแกรม ติดตามปัญหา และใช้ในการบริหารโครงการ (Project Management) พัฒนาโดยบริษัท Atlassion ข้อดีของโปรแกรมมีดังนี้

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

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

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

4) มี Plug-ins และเปิด APIs ให้นักพัฒนาสามารถเชื่อมต่อระบบของตนเองเข้ามายัง Jira ได้ ซึ่งผู้วิจัยจะใช้ APIs ของ Jira ในการทำการสร้างทิคเก็ตจากบทวิจารณ์ของผู้ใช้ที่ได้จากการเรียนรู้ด้วยเครื่องแบบอัตโนมัติ [12]

2.6 Issue Tracking System [13]

Issue Tracking System หมายถึงระบบ ซอฟต์แวร์ หรือโปรแกรมที่เกี่ยวข้องกับการบริหารจัดการโครงการซอฟต์แวร์ต่าง ๆ ตั้งแต่เริ่มต้นการเก็บรวบรวมความต้องการของผู้ใช้งาน การวิเคราะห์ความต้องการของซอฟต์แวร์ การออกแบบระบบซอฟต์แวร์ การอิมพลีเมนต์ซอฟต์แวร์ การทดสอบซอฟต์แวร์ การจำหน่ายซอฟต์แวร์ รวมไปถึงการบำรุงรักษาซอฟต์แวร์หลังจากที่มีการใช้งานจริงแล้ว โดยสิ่งที่จะ Track ก็คือสถานะทำงานของขั้นตอนต่าง ๆ ว่าเป็นอย่างไร มีความคืบหน้าถึงส่วนไหนแล้ว โดย Issue ในที่นี้หมายถึง งานใด ๆ ที่ต้องทำซึ่งจะมีระดับ

ความสำคัญประกอบไว้ นอกจากนี้ใน Issue ยังมีรายละเอียดเกี่ยวกับกำหนดวันที่ต้องส่ง รายละเอียดของงานที่ต้องทำหรือปัญหาที่ต้องแก้ไข วิธีการแก้ปัญหาที่ใช้ เป็นต้น Issue สามารถเป็นไปได้หลายประเภท เช่น Features Request, Bug Reports, User Experience Improvement เป็นต้น

2.7 Ticket [13]

ทิคเก็ต หมายถึง สิ่งที่ระบุ (Artifact) ถึงงานที่ต้องทำในรอบการทำใด ๆ ของการพัฒนาซอฟต์แวร์แบบเอไจล์ โดยสามารถแบ่ง ประเภทของทิคเก็ต (Issue Type) ได้ดังนี้

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

2.8.2 งาน (Epic) หมายถึง รายละเอียดใด ๆ ซึ่งสามารถนำไปแตกเป็นงานย่อย ๆ ได้

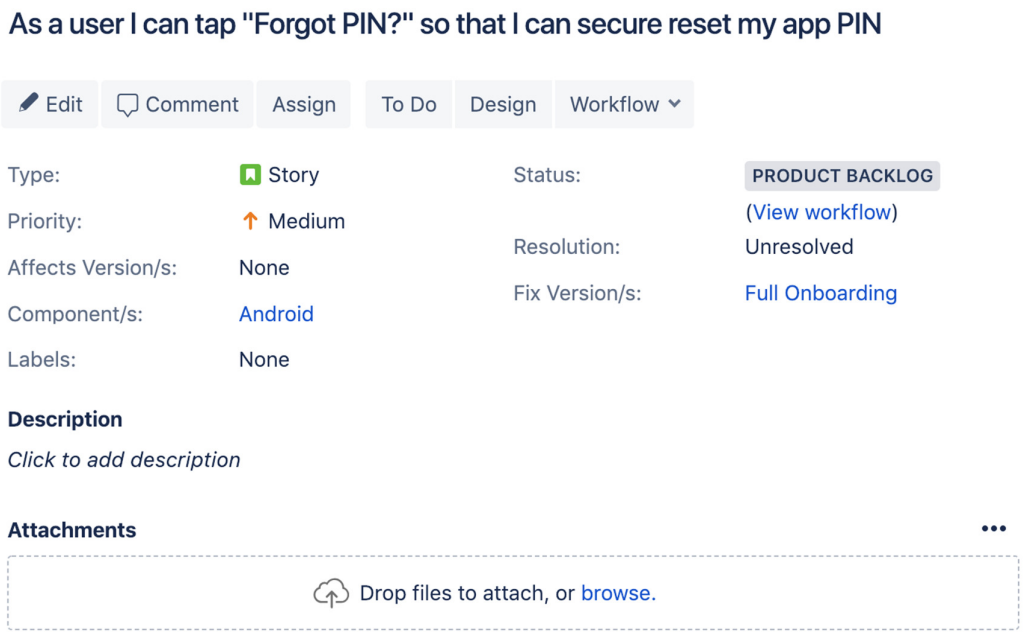
2.8.3 งานย่อย (Task) หมายถึง สิ่งที่ต้องทำเพื่อให้บรรลุสตอรีที่อยู่ภายใต้งานใหญ่

2.8.4 จุดบกพร่อง (Bug) หมายถึง สิ่งที่แสดงออกมาบ่งบอกว่าซอฟต์แวร์ไม่สามารถทำงานได้อย่างถูกต้องตามความต้องการที่ระบุไว้

2.8.5 การปรับปรุง (Improvement) หมายถึง การพัฒนาให้ดีขึ้น

2.8.6 ความสามารถ (Feature) หมายถึง ฟังก์ชันการทำงานใด ๆ ที่ซอฟต์แวร์สามารถทำงานเพื่อตอบสนองความต้องการของผู้ใช้ได้

ตัวอย่างสตอรีดังภาพที่ 2.6



ภาพที่ 2.6 สตอรีที่บอกว่าผู้ใช้งานสามารถกดปุ่มลืมรหัสผ่านเพื่อทำการรีเซ็ตรหัสผ่านใหม่ได้

2.8 Product Backlog [14]

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

2.9 Agile Board [15]

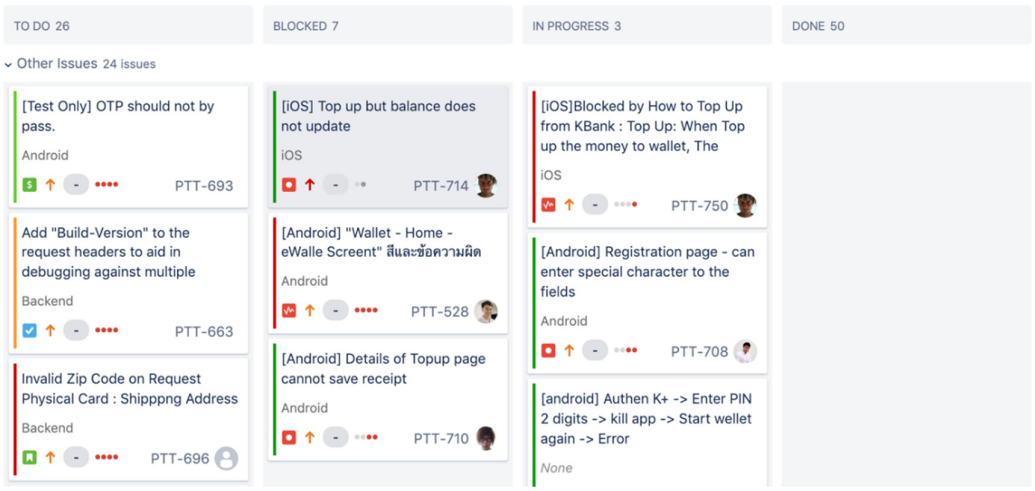
Agile Board คือ สิ่งที่ระบุถึงภาพรวมของการทำงานในรอบการทำงานปัจจุบัน โดยทั่วไปแล้วจะอยู่ในรูปแบบตารางที่บรรจุไปด้วยงานย่อยต่าง ๆ ปกติตารางจะมีสามคอลัมน์ แต่อาจจะมีมากกว่านี้ก็ได้ขึ้นอยู่กับการตกลงกันของผู้ที่เกี่ยวข้องในโครงการซอฟต์แวร์ใด ๆ ซึ่งสามคอลัมน์หลัก เป็นดังนี้

1) To Do หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่ยังไม่ได้ทำในรอบการทำงานปัจจุบัน

2) In Progress หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่กำลังทำอยู่ในรอบการทำงานปัจจุบัน

3) Done หมายถึง คอลัมน์ที่บ่งบอกถึงจำนวนงานย่อยที่ทำเสร็จสิ้นแล้วในรอบการทำงานปัจจุบัน

โดยในระหว่างรอบการทำงานปัจจุบันนั้น นักพัฒนาซอฟต์แวร์สามารถที่จะทำการเคลื่อนย้ายทิคเก็ตไปตามคอลัมน์ต่าง ๆ ตามสถานะการทำงานได้ ตัวอย่างเอไจล์บอร์ด ดังภาพที่ 2.7



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

3. งานวิจัยที่เกี่ยวข้อง

3.1 Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews [16]

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

1) รายงานจุดบกพร่อง (Bug Reports)

2) รายการความต้องการทางฟังก์ชันการทำงาน (Feature Requests)

3) ประสบการณ์การใช้งานแอปพลิเคชัน (User Experiences)

4) ความพึงพอใจในการใช้งาน (Ratings)

โดยเพื่อให้มั่นใจว่าผลลัพธ์การแบ่งกลุ่มที่ได้มีความถูกต้องสูง ทางกลุ่มผู้วิจัยได้เลือกบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ มาอย่างละ 2,200 ความคิดเห็นตามลำดับ โดยมีทั้งเลือกแบบสุ่ม และเลือกแบบเจาะจง มาให้ผู้มีประสบการณ์ด้านเขียนโปรแกรมบอกว่าบทวิจารณ์ของผู้ใช้นี้เป็นบทวิจารณ์ประเภทใด โดยหนึ่งบทวิจารณ์ของผู้ใช้จะถูกระบุโดยคนสองคน แล้วเอาผลลัพธ์มาเปรียบเทียบกันว่าได้ประเภทของบทวิจารณ์ของผู้ใช้ตรงกันหรือไม่ ถ้าไม่ตรงกันเพราะสาเหตุใด โดยทำผ่านเครื่องมือการระบุ Label ที่ทางผู้วิจัยได้พัฒนาขึ้น และ Guideline สำหรับการรีวิว ซึ่งหลังจากได้บทวิจารณ์ของผู้ใช้ที่ถูกระบุประเภทเรียบร้อยมาทั้งสิ้น 4,400 ความคิดเห็นแล้ว ผู้วิจัยได้ทำการแบ่งข้อมูล 70% มาเป็น Training Set เพื่อทำการเทรนโมเดลด้วยเทคนิคต่าง ๆ ที่กล่าวไปแล้วข้างต้น และข้อมูล 30% เป็น Test Set เพื่อใช้สำหรับวัดผลโมเดลที่ผ่านการเทรนมาว่ามีประสิทธิภาพมากน้อยเพียงใด ซึ่งผลลัพธ์ที่ได้คือ ค่า Precision ประมาณ 70-95% และค่า Recall ประมาณ 80-90% นอกจากนี้ยังมีการนำค่า F-Measure มาช่วยในการเปรียบเทียบประสิทธิภาพของโมเดลอีกด้วย ดังตารางที่ 3.1

ตารางที่ 3.1 การเปรียบเทียบประสิทธิภาพของโมเดลโดยใช้เทคนิคของ Naïve Bayes ต่าง ๆ กับบทวิจารณ์ของผู้ใช้บน แอปสโตร์และเพลย์สโตร์ที่ได้เลือกมา [16]

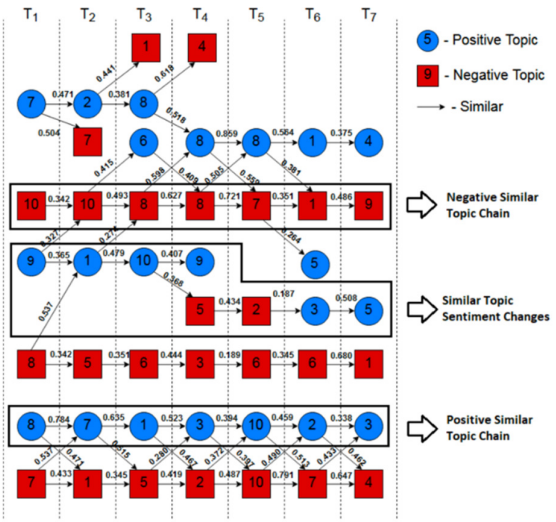
Classification techniques	Bug Reports			Feature Requests			User Experiences			Ratings		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Basic (string matching)	0.58	0.24	0.33	0.39	0.55	0.46	0.27	0.12	0.17	0.74	0.56	0.64
Document Classification (& NLP)												
Bag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

ผู้วิจัยจะนำข้อมูลบทวิจารณ์ของผู้ใช้ที่งานวิจัยนี้ใช้มาเป็น Training Set เนื่องจากมีการทำ Label ไว้แล้ว ข้อมูลมีความน่าเชื่อถือเนื่องจากมี Methodology ในการติด Label ที่รัดกุม เพื่อใช้เป็นข้อมูลนำเข้าไปในการเรียนรู้แบบ Supervised Learning ให้กับโมเดล และใช้เป็นประเภทของบทวิจารณ์ของผู้ใช้ที่โครงงานมหาบัณฑิตจะทำการแยกประเภท รวมทั้งเลือกวิธีการจัดประเภทบทวิจารณ์ของผู้ใช้ที่มีประสิทธิภาพสูงที่สุด เพื่อนำมาเปรียบเทียบกับวิธีการจัดประเภทของบทวิจารณ์ของผู้ใช้ของโครงงานมหาบัณฑิต

ข้อแตกต่างระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้มุ่งเน้นไปที่การจัดประเภทของบทวิจารณ์ของผู้ใช้ว่ามีประเภทอะไรบ้าง โดยใช้เทคนิค วิธีการ และอัลกอรึทึมต่าง ๆ ทาง Data Mining และ Machine Learning แล้วนำผลลัพธ์ที่ได้มาเปรียบเทียบกันว่าวิธีการใดให้ผลลัพธ์ได้มีประสิทธิภาพมากที่สุดในการจัดประเภทบทวิจารณ์ของผู้ใช้ นอกจากนี้ยังแนะนำถึงประโยชน์ของการนำผลลัพธ์ที่จัดประเภทได้ไปใช้งานในการสร้างเครื่องมือวิเคราะห์การใช้งานแอปพลิเคชันของผู้ใช้งาน เช่น การเลือกจำนวน Bug Reports ทั้งหมดออกมาเพื่อส่งต่อให้นักพัฒนาทำการแก้ไข เป็นต้น ส่วนโครงงานมหาบัณฑิตมุ่งเน้นไปที่การจัดประเภทของบทวิจารณ์ของผู้ใช้ ให้อยู่ในกลุ่ม Features Requests, Bug Reports, User Experience Improvement ให้ได้อย่างแม่นยำ มีประสิทธิภาพสูง เพื่อนำไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของ Issue ในทิคเก็ตจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้ที่จัดกลุ่มได้

3.2 Mobile App Evolution Analysis based on User Reviews [17]

งานวิจัยนี้ได้ศึกษาเกี่ยวกับการนำบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันจากบนสโตร์มาทำการวิเคราะห์ถึงแนวโน้มความคิดเห็นของผู้ใช้งานที่มีต่อแอปพลิเคชัน ผู้ใช้งานชอบหรือไม่ชอบแอปพลิเคชันในเรื่องใดบ้าง เช่น ฟังก์ชันการทำงาน ความยากง่ายในการใช้งาน จุดบกพร่องต่าง ๆ ที่พบในแอปพลิเคชัน รวมไปถึงฟังก์ชันการทำงานที่ต้องการให้มีการเพิ่มหรือปรับปรุงให้ดีขึ้น เป็นต้น โดยวิเคราะห์ในระดับแต่ละเวอร์ชันของแอปพลิเคชัน และนำผลที่ได้มาเปรียบเทียบกันว่า ก่อนอัปเดตแอปพลิเคชันและหลังอัปเดตแอปพลิเคชัน ผู้ใช้งานมีความชอบหรือไม่ชอบในเรื่องใดเปลี่ยนแปลงไปบ้าง และได้เลือกแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์ที่ชื่อว่า Whatsapp จากเพลย์สโตร์มาทำการทดลอง เวอร์ชันของแอปพลิเคชันที่ใช้สังเกตมีการอัปเดตแบบ Major ทั้งสิ้น 7 ครั้ง และ Minor 39 ครั้ง ระยะเวลาอยู่ระหว่างเดือนกันยายนปี 2016 ไปจนถึงเดือนสิงหาคมปี 2017 ดังภาพที่ 3.1



ภาพที่ 3.1 ตัวอย่างกราฟการจัดกลุ่มความคิดเห็นของผู้ใช้งานโดยใช้โมเดลของผู้วิจัยของแอปพลิเคชันแอนดรอยด์ Whatsapp ที่มีการอัปเดต Major ทั้งสิ้น 7 ครั้ง และ Minor 39 ครั้ง [17]

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

ผู้วิจัยจะนำทฤษฎี Naïve Bayes ที่งานวิจัยนี้ใช้ในการจำแนกประเภทของความคิดเห็นว่าเป็นความคิดเห็นเชิงบวก หรือความคิดเห็นเชิงลบ มาใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้ในโครงงานมหาบัณฑิตเช่นเดียวกัน

สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้สนใจศึกษาเกี่ยวกับภาพรวมและแนวโน้มของบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันแอนดรอยด์ที่ชื่อว่า Whatapps ก่อนและหลังอัปเดตเวอร์ชันซอฟต์แวร์ ในขณะที่โครงงานมหาบัณฑิตสนใจเกี่ยวกับการจัดประเภทของบทวิจารณ์ของผู้ใช้แอปพลิเคชันใด ๆ และนำไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของ Issue ในทิคเก็ตจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้

3.3 What Do Mobile App Users Complain About? [18]

งานวิจัยนี้ทำการศึกษาเกี่ยวกับประเภทของบทวิจารณ์เชิงลบของผู้ใช้งานแอปพลิเคชัน iOS บนแอปสโตร์ว่ามีอะไรบ้าง โดยทำการจัดลำดับความสำคัญเอาไว้ด้วยเพื่อประโยชน์ในการเลือกพิจารณาบทวิจารณ์เชิงลบกลุ่มที่มีผลต่อภาพรวมของแอปพลิเคชันมากที่สุด เพื่อจะได้รีบทำการแก้ไขโดยเร็ว เพราะบางบริษัทอาจจะมีทรัพยากรหรือบุคลากรทางด้าน Quality Assurance จำกัด จึงต้องบริหารจัดการให้ดี

งานวิจัยนี้เลือกแอปพลิเคชันบนแอปสโตร์ที่ครอบคลุมทุกหมวดหมู่ของแอปพลิเคชันมากที่สุดมา 20 แอปพลิเคชัน แบ่งเป็นกลุ่มที่ Rating เฉลี่ยมากกว่าหรือเท่ากับ 3.5 จำนวน 10 แอปพลิเคชัน และกลุ่มที่ Rating เฉลี่ยน้อยกว่า 3.5 จำนวน 10 แอปพลิเคชัน หลังจากนั้นทำการเก็บรวบรวมเฉพาะบทวิจารณ์ของผู้ใช้งานแต่ละแอปพลิเคชันที่ให้หนึ่งหรือสองดาว เพราะบทวิจารณ์ที่ได้หนึ่งหรือสองดาว มักจะมีคำอธิบายหรือการพูดถึงที่ค่อนข้างมีประโยชน์ในการนำมาจัดกลุ่ม กลุ่มผู้วิจัยก็จะทำการตรวจสอบบทวิจารณ์โดยอาศัยการอ่าน ทำความเข้าใจ และแบ่งประเภทของบทวิจารณ์ด้วยตนเอง ซึ่งจะทำไล่ทีละแอปพลิเคชันไปเรื่อย ๆ ถ้าพบหมวดหมู่ใหม่ ก็จะมีการบันทึกเพิ่มเข้าไป สุดท้ายได้ผลลัพธ์ทั้งสิ้น 13 ประเภทเรียงตามลำดับความถี่ที่พบ ดังตารางที่ 3.2

ตารางที่ 3.2 ประเภทบทวิจารณ์เชิงลบของผู้ใช้งานแอปพลิเคชันบนแอปสโตร์ โดยจัดลำดับตามความถี่ของการพบปัญหาจากมากไปน้อยทั้งสิ้น 13 ประเภท [18]

TABLE 3

The most frequent and impactful complaint types.*

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

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

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

จากตารางที่ 3.2 ด้านบน จะพบว่า Functional Error, Feature Request, App Crashing เป็นความคิดเห็นเชิงลบสามลำดับแรกที่ผู้ใช้งานพูดถึงมากที่สุด โดยรวมกันแล้วครอบคลุมกว่า 50% ของบทวิจารณ์เชิงลบทั้งหมด

งานวิจัยนี้ไม่ได้ทำการ Train Classification Model แต่อย่างใด เพียงแต่ใช้วิธีการเก็บข้อมูลปกติ จากนั้นนำหลักการทางสถิติมาจัดอันดับเรื่องที่คนพูดถึงในแง่ลบมากที่สุด 13 อันดับแรก ซึ่งกระบวนการตั้งแต่ต้นจนจบกระทำโดยกลุ่มผู้วิจัยแบบ Manual เอง เนื่องจากงานวิจัยนี้ได้แบ่งประเภทของบทวิจารณ์ของผู้ใช้ไว้ค่อนข้างละเอียด

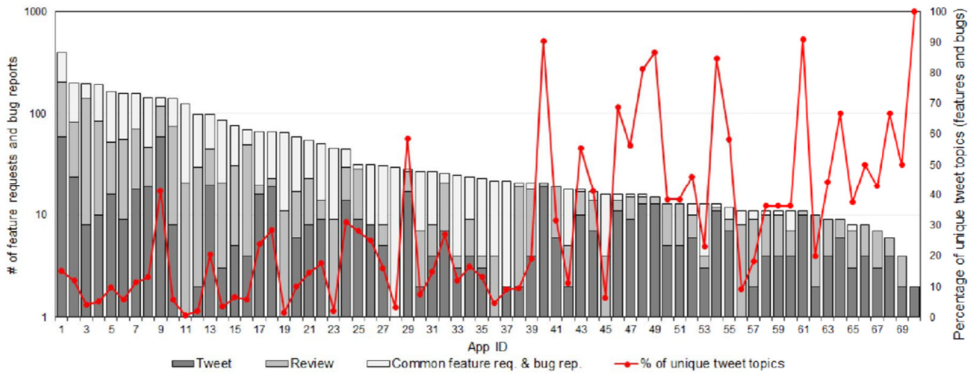
สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้ไม่ได้ใช้ศาสตร์ทางด้าน Data Mining หรือ Machine Learning มาจัดประเภทของบทวิจารณ์ของผู้ใช้ ไม่ได้มีการนำองค์ความรู้ที่วิเคราะห์ได้ไปต่อยอดในเชิงอุตสาหกรรม ซึ่งโครงงานมหาบัณฑิตจะครอบคลุมทั้งในสองส่วนนี้ คือ การจัดประเภทของบทวิจารณ์ของผู้ใช้โดยใช้ศาสตร์ทางด้าน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

3.4 App Store Mining is Not Enough [19]

งานวิจัยนี้ได้สำรวจเกี่ยวกับบทวิจารณ์ของผู้ใช้งานแอปพลิเคชัน iOS ที่ไม่ได้อยู่บนแอปสโตร์ แต่อยู่บน Social Media อื่น ๆ และเพลย์สโตร์ ตัวอย่าง Social Media ที่มีจำนวนผู้ใช้งานและจำนวนบทวิจารณ์สูง คือ ทวิตเตอร์ (Twitter) เป็นต้น โดยผู้วิจัยระบุว่า การสำรวจบทวิจารณ์ของผู้ใช้งานแอปพลิเคชันเพื่อดูถึงสิ่งที่ผู้ใช้งานต้องการเพิ่ม หรือสิ่งที่ต้องการให้แก้ไขบนแอปสโตร์อย่างเดียวนั้นไม่เพียงพอ เนื่องจากผู้ใช้งานบางคนไม่ได้มาให้คะแนนและแสดงความคิดเห็นในแอปสโตร์ แต่ได้ไปโพสต์ผ่านทวิตเตอร์แทน จึงมีความเป็นไปได้สูงว่าข้อมูลการ Tweets จะมีความสำคัญและสัมพันธ์กับข้อมูลที่อยู่บนแอปสโตร์ด้วย

ผู้วิจัยได้ทำการเลือกแอปพลิเคชันบนแอปสโตร์มา 70 แอปพลิเคชัน หลังจากนั้นจึงไปทำการ Mining Tweets และบทวิจารณ์บนเพลย์สโตร์ที่เกี่ยวข้องกับทั้ง 70 แอปพลิเคชัน ภายในระยะเวลาหกสัปดาห์ เพื่อดูว่ามีบทวิจารณ์ของผู้ใช้ใดบ้าง ที่มีความเกี่ยวข้องกับบทวิจารณ์ของผู้ใช้งานบนแอปสโตร์ หรือบทวิจารณ์ของผู้ใช้ใดบ้างที่ปรากฏบนทวิตเตอร์หรือเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์ โดยใช้หลักการ Cosine Similarity วิเคราะห์บทวิจารณ์ของผู้ใช้สองบทวิจารณ์ใด ๆ บนทวิตเตอร์ แอปสโตร์ และเพลย์สโตร์ ว่าถ้ามีค่ามากกว่า 0.6 แปลว่า บทวิจารณ์ของผู้ใช้อันนี้ปรากฏจากทั้งสองแหล่งข้อมูลใด ๆ ที่นำมาเปรียบเทียบกัน หลังจากนั้นจึงทำการตรวจสอบโดยให้คนมาทำการวิเคราะห์แบบ Manual อีกครั้งหนึ่ง จากภาพที่ 3.2 พบว่า

1) มีจำนวน Feature Requests 22.4% และ Bug Reports 12.89% ที่ปรากฏบนทวิตเตอร์หรือเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์
2) Feature Requests 43.51% และ Bug Reports 56.61% ที่ปรากฏบนทวิตเตอร์และเพลย์สโตร์ แต่ไม่ปรากฏบนแอปสโตร์ตามลำดับ จากข้อมูลดังกล่าวจึงทำให้ผู้วิจัยสรุปว่าบทวิจารณ์ของผู้ใช้งานที่อยู่บนแอปสโตร์เพียงอย่างเดียวไม่เพียงพอ จำเป็นต้องใช้ข้อมูลจากแหล่งอื่นเพื่อทำให้การปรับปรุงคุณภาพของแอปพลิเคชันดียิ่งขึ้น



ภาพที่ 3.2 เปรียบเทียบจำนวน Feature Requests และ Bug Reports ของทั้ง 70 แอปพลิเคชันที่ผู้วิจัยได้เลือกมา แยกตามแหล่งที่มาของข้อมูล คือ ทวิตเตอร์ สโตร์ ทั้งทวิตเตอร์และสโตร์ และ % แสดง Feature Requests กับ Bug Reports ที่พบเฉพาะบนทวิตเตอร์ [19]

งานวิจัยนี้ได้ใช้ Naïve Bayes ในการแยกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์ เพลย์สโตร์ และทวิตเตอร์ ว่าเป็น Features Request หรือ Bug Reports หรืออื่น ๆ ซึ่งโครงงานมหาบัณฑิตก็จะใช้ Naïve Bayes เช่นเดียวกัน เนื่องจากมีประสิทธิภาพสูง ในขณะที่ใช้ Training Data ในปริมาณที่ไม่ได้มากจนเกินไป นอกจากนี้งานวิจัยนี้ได้ใช้ Support Vector Machine เนื่องจากมีประสิทธิภาพดีกว่า Decision Tree สามารถกรองบทวิจารณ์ของผู้ใช้บนทวิตเตอร์ที่ไม่เกี่ยวข้องออกไปได้ดีกว่า

สิ่งที่แตกต่างกันระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิตคือ งานวิจัยนี้มุ่งเน้นเรื่องเกี่ยวกับว่ามีบทวิจารณ์ของผู้ใช้ใด ๆ ที่ปรากฏบนแหล่งอื่น ๆ นอกจากแอปสโตร์ เช่น ทวิตเตอร์ เพลย์สโตร์ เป็นต้น ไม่ได้มีการทำเครื่องมือช่วยในการจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์ เพลย์สโตร์ หรือทวิตเตอร์แต่อย่างใด แต่โครงงานมหาบัณฑิตจะมุ่งเน้นที่การจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ และการสร้างทิคเก็ตของ Jira โดยประเภทของ Issue จะขึ้นอยู่กับประเภทของบทวิจารณ์ของผู้ใช้

3.5 AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace [20]

งานวิจัยนี้ได้กล่าวถึงประเด็นที่ตลาดแอปพลิเคชันนั้นเติบโตอย่างรวดเร็วมาก ในแต่ละปีมีผู้ใช้งานสมาร์ตโฟน เพิ่มขึ้นเป็นจำนวนมาก จำนวนแอปพลิเคชันบนสโตร์ที่ถูกดาวน์โหลดจากผู้ใช้งานก็เพิ่มขึ้น ซึ่งหลังจากผู้ใช้งานได้ทดลองใช้งานแอปพลิเคชันแล้ว ก็จะมาให้คะแนนและแสดงความคิดเห็นในช่องทางบนสโตร์ ซึ่งคะแนนและความคิดเห็นเหล่านี้เป็นสิ่งที่มีประโยชน์ต่อนักพัฒนาแอปพลิเคชันเป็นอย่างมาก เพราะช่วยให้ทราบถึงสิ่งที่ดี สิ่งที่ไม่ดีที่ต้องปรับปรุงเพื่อให้ผู้ใช้งานพึงพอใจมากที่สุด

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

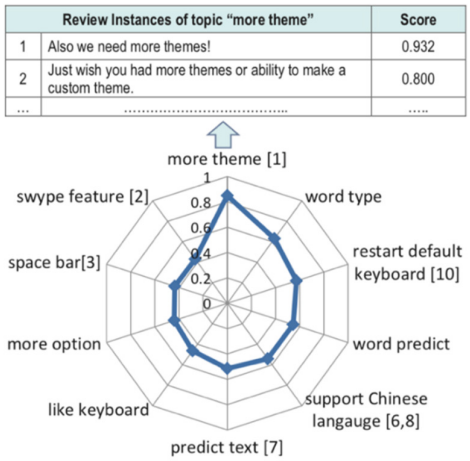
1) ใช้ Expectation Maximization for Naive Bayes (EMNB) ซึ่งเป็น Semi-Supervised Learning ประเภทหนึ่ง ทำการกรองข้อมูลที่ไม่สมบูรณ์ และไม่เกี่ยวข้องออก โดยจำแนกประเภทของบทวิจารณ์ออกเป็นสองกลุ่ม คือ Informative และ Uninformative เพื่อนำโมเดลที่ได้มากรองให้เหลือเฉพาะบทวิจารณ์ที่มีประโยชน์ต่อการพัฒนาซอฟต์แวร์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

2) ใช้ K-Means เพื่อทำการจัดกลุ่มประเภทของบทวิจารณ์ที่มีลักษณะคล้ายกันไว้ด้วยกันบนสมมติฐานที่ว่าหนึ่งบทวิจารณ์ของผู้ใช้ใด ๆ จะถูกจัดกลุ่มให้อยู่ได้เพียงแค่กลุ่มเดียวเท่านั้น

3) ใช้ LDA เพื่อทำการจัดกลุ่มประเภทของบทวิจารณ์ที่มีลักษณะคล้ายกันไว้ด้วยกัน โดยหนึ่งบทวิจารณ์ใด ๆ สามารถถูกจัดให้อยู่ได้มากกว่าหนึ่ง Topic ได้ เนื่องจากบางครั้งบทวิจารณ์ของผู้ใช้ที่มีลักษณะยาว จะมีการอธิบายผสมกันทั้ง Features Request และ Bug Reports หรืออื่น ๆ ได้

4) จัดลำดับความสำคัญของแต่ละกลุ่มความคิดเห็นและทำการวัดผล

5) แสดงผลกลุ่มของความคิดเห็นที่มีความสำคัญเรียงตามลำดับจากมากไปน้อยในรูปแบบที่สวยงาม ง่ายต่อนักพัฒนาในการทำความเข้าใจ ดังภาพที่ 3.3



ภาพที่ 3.3 การแสดงผลของ AR-Miner ของกลุ่มความคิดเห็น 10 กลุ่มแรกที่มีความสำคัญมากที่สุดของแอปพลิเคชัน SwiftKey โดย Keyword "more theme" ถูกจัดเป็นอันดับหนึ่ง [20]

กลุ่มผู้วิจัยได้ใช้เครื่องมือ AR-Miner ที่พัฒนาขึ้น มาทดลองเพื่อที่จะทำการวัดประสิทธิภาพของเครื่องมือกับแอปพลิเคชัน Android ที่มีจำนวนบทวิจารณ์ของผู้ใช้มากทั้งสิ้น 4 แอปพลิเคชัน คือ SwiftKey, Facebook, TempleRun2, TapFish โดยใช้ค่า Precision-Recall รวมทั้ง F-Measure เป็นตัววัดประสิทธิภาพ และนอกจากนี้ยังได้นำไปเทียบกับ Forum การแสดงความคิดเห็นออนไลน์ของ SwiftKey ที่เปิดให้ผู้ใช้งานสามารถแสดงความคิดเห็นต่อแอปพลิเคชันได้ โดยผลการทดลองที่ได้คือ เครื่องมือ AR-Miner นั้นช่วยลดงานที่ต้องใช้คนทำลงได้เป็นอย่างมาก รวมทั้งให้ผลลัพธ์ที่มีประสิทธิภาพสูงกว่า นอกจากนี้ยังพบ Feature Request บางอย่างที่ไม่พบจากการรวบรวมข้อมูลแบบ Manual อีกด้วย

โครงงานมหาบัณฑิตจะนำแนวคิดของงานวิจัยนี้ในเรื่องต่อไปนี้ไปใช้งาน

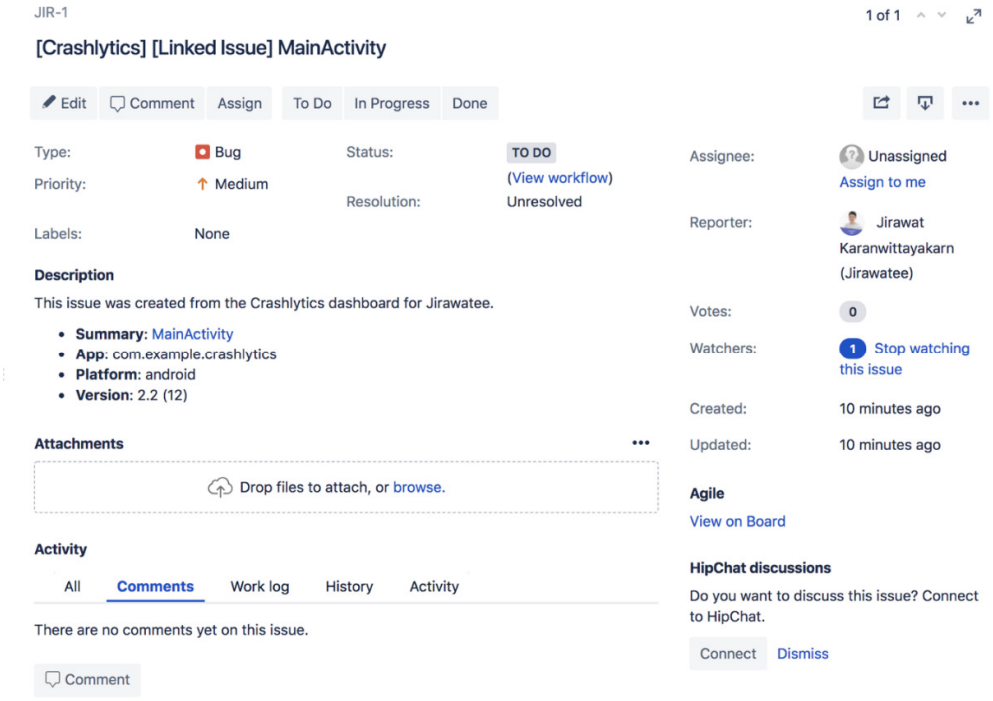
1) ใช้ Expectation Maximization for Naive Bayes เพื่อกรองบทวิจารณ์ที่ไม่มีประโยชน์ต่อการพัฒนาซอฟต์แวร์ออก
2) แนวคิดการทำเครื่องมือเกี่ยวกับบทวิจารณ์ของผู้ใช้มาจากงานวิจัยนี้ แต่แตกต่างกันตรงที่งานวิจัยนี้สร้างเครื่องมือวิเคราะห์บทวิจารณ์ของผู้ใช้ แล้วนำเสนอออกมาในรูปแบบที่ง่ายต่อการทำความเข้าใจ แต่เครื่องมือของโครงงานมหาบัณฑิตที่จะพัฒนาขึ้นนี้จะเป็นการสร้างทิคเก็ต Jira จากบทวิจารณ์ของผู้ใช้ โดยประเภทของ Issue ขึ้นอยู่กับประเภทของบทวิจารณ์ของผู้ใช้

3.6 Automatically Create Jira Issue from Firebase Crashlytics [21]

บทความนี้ไม่ใช่งานวิจัยตีพิมพ์ แต่กล่าวถึงการนำ Jira มาใช้งานร่วมกับ Firebase Crashlytics ซึ่งเป็นความร่วมมือกันทางธุรกิจระหว่าง Atlassian ซึ่งเป็นเจ้าของ Jira และ Google ที่เป็นเจ้าของ Firebase Crashlytics ทั้งนี้ Firebase Crashlytics คือ ไลบรารีที่สามารถนำไปติดตั้งไว้ในโมไบล์แอปพลิเคชันและเว็บไซต์ได้ เมื่อโมไบล์แอปพลิเคชันหรือเว็บไซต์เกิดการปิดตัวลงอย่างกะทันหัน (Crash) ตัว Firebase Crashlytics จะทำการสร้างบันทึกรายงานจุดบกพร่องที่ระบุรายละเอียดเกี่ยวกับจุดที่เกิดปัญหาขึ้น เช่น ชื่อไฟล์ และหมายเลขบรรทัดของโค้ดที่เกิด

ปัญหา เป็นต้น เก็บไว้ใน Firebase Console ซึ่งนักพัฒนาสามารถเข้ามาดูรายละเอียดในภายหลังได้ พร้อมทั้งมีการแจ้งเตือนทางอีเมลด้วย

ทาง Atlassian และ Google ต้องการให้การทำงานของบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระนั้นราบรื่นมากยิ่งขึ้น โดยเล็งเห็นว่าส่วนใหญ่บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระนิยมใช้ Jira เป็นเครื่องมือในการบริหารจัดการโครงการซอฟต์แวร์แบบเอไจล์อยู่แล้ว ดังนั้นเมื่อเกิดจุดบกพร่องจนทำให้โมไบล์แอปพลิเคชันหรือเว็บไซต์ปิดตัวลงอย่างกะทันหัน Google จะทำการเรียก APIs ของ Jira เพื่อทำการสร้าง Issues ขึ้นในเอไจล์บอร์ด โดยกระบวนการทั้งหมดเป็นไปอย่างอัตโนมัติ เพียงแค่นักพัฒนาติดตั้งไลบรารี Firebase Crashlytics ลงไป และไปทำการตั้งค่าใน Jira กับ Firebase Console เล็กน้อย ตัวอย่าง Issue ที่ถูกสร้างจาก Firebase Crashlytics เข้าไปที่ Jira เป็นดังภาพที่ 3.4



ภาพที่ 3.4 ตัวอย่างทิคเก็ตประเภท Bug ที่ถูกสร้างใน Jira จาก Firebase Crashlytics ที่ทำการเชื่อมต่อกับ APIs ของ Jira [21]

ผู้วิจัยจะนำแนวทางการเชื่อมต่อระหว่าง Jira กับ Firebase Crashlytics มาสร้างเป็นเครื่องมือที่สามารถสร้างทิคเก็ตแยกตามประเภทของบทวิจารณ์ของผู้ใช้ โดยทำการเรียกใช้งาน APIs ของ Jira เพื่อส่งค่าพารามิเตอร์เข้าไปเช่นเดียวกับที่ Firebase Crashlytics ทำ

3.7 Recommending and Localizing Change Requests for Mobile Apps based on User Reviews [22]

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

1) แยกประเภทของบทวิจารณ์ของผู้ใช้ประเภท Maintenance ออกมาได้
2) จัดกลุ่มบทวิจารณ์ของผู้ใช้ที่มีลักษณะคล้ายกันให้อยู่กลุ่มเดียวกันได้
3) บอกได้ว่าไฟล์ และฟังก์ชันใด ที่ทำให้เกิดจุดบกพร่องขึ้น โดยอิงข้อมูลจากบทวิจารณ์ของผู้ใช้ และซอร์ซโค้ดที่ต้องอัปโหลดเข้าไปในเครื่องมือ

ผู้วิจัยจะนำวิธีการทำ Data Cleansing ของงานวิจัยนี้มาใช้ในโครงงานมหาบัณฑิต เนื่องจากมีหลายขั้นตอน และมีความละเอียดสูง ซึ่งจะกล่าวต่อไปในหัวข้อขั้นตอนและวิธีการดำเนินงาน

ข้อแตกต่างระหว่างงานวิจัยนี้กับโครงงานมหาบัณฑิต คือ งานวิจัยนี้สร้างเครื่องมือที่ช่วยระบุได้ว่าโค้ดไฟล์ใด ฟังก์ชันอะไรที่ทำให้เกิดจุดบกพร่องขึ้นโดยอิงข้อมูลจากบทวิจารณ์ของผู้ใช้และซอร์ซโค้ดที่อัปโหลดให้กับเครื่องมือ ส่วนเครื่องมือที่จะพัฒนาในโครงงานมหาบัณฑิตจะทำหน้าที่สร้างทิคเก็ต Jira แยกประเภท Issue ตามประเภทของบทวิจารณ์ของผู้ใช้แบบอัตโนมัติ

4. แนวคิดและวิธีดำเนินการ

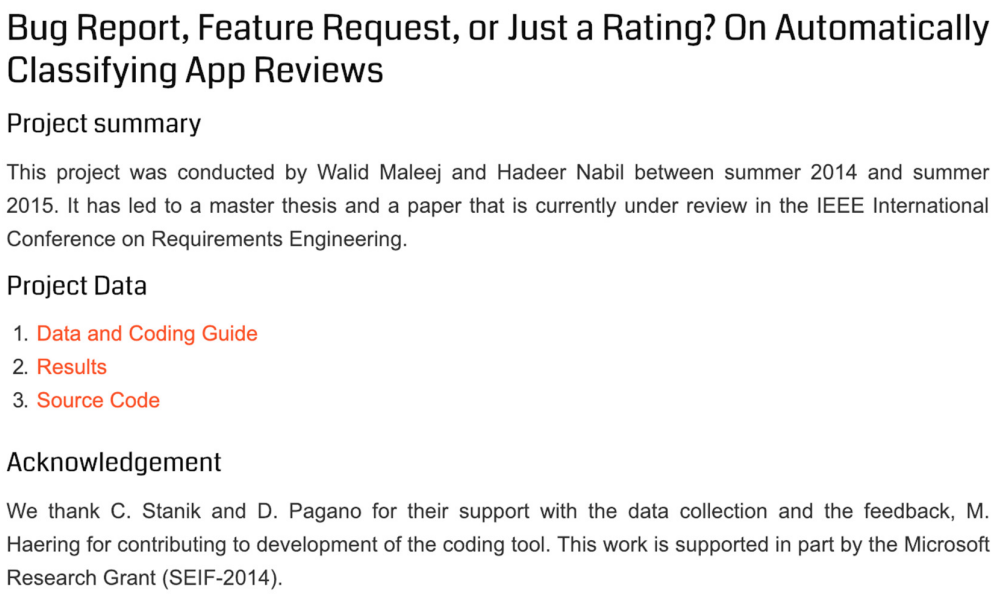
แนวคิดงานวิจัยนี้เป็นการนำองค์ความรู้ทางด้าน Data Mining, Machine Learning ต่าง ๆ มาทำการจำแนกบทวิจารณ์ของผู้ใช้บนแอปสโตร์หรือเพลย์สโตร์ของแอปพลิเคชันใด ๆ ว่าเป็นประเภทใด โดยแบ่งเป็น Features Request, Bug Reports, User Experience Improvement, และประเภท Others ซึ่งหมายถึงบทวิจารณ์ของผู้ใช้ที่ไม่เข้าพวกในสามกลุ่มแรก ซึ่งหลังจากแบ่งกลุ่มบทวิจารณ์ของผู้ใช้เรียบร้อยแล้ว ก็จะนำข้อมูลสามกลุ่มแรกไปเชื่อมต่อกับ APIs ของ Jira เพื่อทำการสร้างทิคเก็ตลงใน Product Backlog ให้กับโปรเจค เพื่อให้นักพัฒนาทำการปรับปรุงคุณภาพของแอปพลิเคชันให้ดีขึ้นในรอบการทำงานถัดไป โดยภาพรวมขั้นตอนการดำเนินงานทั้งหมดแบ่งได้ทั้งสิ้น 9 ขั้นตอน ดังนี้

4.1 ขั้นตอนการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ตัวอย่าง
4.2 ขั้นตอนการทำ Data Cleansing และ Text Processing
4.3 ขั้นตอนการพัฒนาโมเดลเพื่อจำแนกประเภทของบทวิจารณ์ของผู้ใช้
4.4 ขั้นตอนการทดสอบโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้
4.5 ขั้นตอนการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้
4.6 ขั้นตอนการปรับปรุงประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้
4.7 ขั้นตอนการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้
4.8 ขั้นตอนการเชื่อมต่อกับ APIs ของ Jira เพื่อทำการสร้างทิคเก็ต

4.9 ขั้นตอนการตรวจสอบทิคเก็ตที่ถูกสร้างในเอไจล์บอร์ดของ Jira
โดยสามารถอธิบายแต่ละขั้นตอนการดำเนินงานได้ ดังนี้

4.1 ขั้นตอนการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ตัวอย่าง

ขั้นตอนนี้จะเป็นการเก็บรวบรวมบทวิจารณ์ของผู้ใช้ตัวอย่างที่ได้ทำการแยกประเภทไว้แล้วมาเป็น Training Set เพื่อเป็นการประหยัดเวลาในการติด Label ของบทวิจารณ์ของผู้ใช้แบบ Manual รวมไปถึงเรื่องของความน่าเชื่อถือของข้อมูล ทำให้ผู้วิจัยจะใช้ข้อมูลชุดเดียวกับงานวิจัยที่ 3.1 ซึ่งได้เปิดให้สามารถดาวน์โหลดข้อมูลบทวิจารณ์ของผู้ใช้ได้ฟรี มีประมาณ 4,000 บทวิจารณ์อยู่ในรูปโครงสร้างแบบ JSON โดยสามารถโหลดได้จาก https://mast.informatik.uni-hamburg.de/app-review-analysis/ ดังภาพที่ 4.1



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

Project summary

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

Project Data

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

Acknowledgement

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

ภาพที่ 4.1 หน้าเว็บไซต์ของงานวิจัยที่ 3.1 ที่เปิดให้สามารถดาวน์โหลดข้อมูลได้ฟรี

หลักการจำแนกประเภทบทวิจารณ์ของผู้ใช้จะอ้างอิงตามเปเปอร์ที่ 3.1 ซึ่งได้นำเสนอคู่มือการจำแนกประเภทบทวิจารณ์ของผู้ใช้โดยสามารถดาวน์โหลดได้จาก https://mobis.informatik.uni-hamburg.de/wp-content/uploads/2015/03/Data-and-Coding-Guide-.zip ซึ่งมีคำอธิบายของแต่ละประเภทบทวิจารณ์ของผู้ใช้ดังนี้

1) User Experience [16] ผู้ใช้งานมีการอ้างอิงถึงฟังก์ชันการทำงานใด ๆ ที่มีอยู่แล้วในแอปพลิเคชัน ซึ่งช่วยอำนวยความสะดวกในการดำเนินชีวิตประจำวัน เช่น

“I use this app almost every weekend while exploring back roads and trails by motorcycle. Functionality is excellent while on the road as well as using the data to review later using Google Earth”

“I have used for this app for a few years now. It just keeps getting better. I Tavel a lot so it is very useful”

“Great for daily readings. I love it.”

2) Bug Reports [16] ผู้ใช้งานอธิบายถึงปัญหาที่พบในการใช้งานแอปพลิเคชัน พฤติกรรมการทำงานไม่เหมาะสมของฟังก์ชันการทำงานใด ๆ หรือทั้งแอปพลิเคชันโดยรวม เช่น

“Uploading is not working with the iOS6”

“Every time I launch the app, it crashes”

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

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

3) Features Request [16] ผู้ใช้งานอธิบายถึงฟังก์ชันการทำงาน รายละเอียดของแอปพลิเคชันที่ขาดหายไปหรือฟังก์ชันเดิมที่มีอยู่แล้ว แต่ต้องการให้มีการปรับปรุงให้ดียิ่งขึ้น เช่น

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

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

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

4) Rating [16] ผู้ใช้งานกล่าวชื่นชมหรือตำหนิแอปพลิเคชันหรือฟังก์ชันการทำงานใด ๆ เช่น

“Great app I love itttt”,

“Fire the developer created this app it’s the worst app ever !!”,

“I cannot believe how amazing the new changes are.”,

“Buy it !!”

โดยประเภท Rating นี้ จะเทียบเคียงได้กับประเภท Others ในโครงงานมหาบัณฑิต

สำหรับประเภท User Experience Improvement นั้นทางผู้วิจัยจะอ้างอิงความหมายของคำว่า “Improvement” จากดิกชันนารีออนไลน์ของมหาวิทยาลัยเคมบริดจ์ ซึ่งหมายถึง “an occasion when something gets better or when you make it better” หรือการทำให้สิ่งใด ๆ นั้นให้มีประสิทธิภาพดีมากขึ้น ซึ่งเมื่อนำมารวมกับนิยามของคำว่า User Experience จากงานวิจัย 3.1 นั้นก็จะได้ว่า User Experience Improvement หมายถึง ผู้ใช้งานมีการอ้างอิงถึงฟังก์ชันการทำงานใด ๆ ที่มีอยู่แล้วในแอปพลิเคชัน แต่ผู้ใช้งานยังต้องการให้มีการปรับปรุงฟังก์ชันการทำงานนั้น ๆ ให้ดีมากยิ่งขึ้น เช่น

“Love filtering news type but it could be better if it can be filtered by customization topic from user”,

“The list layout of the app is very impressive. However, pagination for list would be appreciate”

ซึ่งผู้วิจัยจะทำการ Label บทวิจารณ์ของผู้ใช้ประเภท User Experience และ Features Request จากข้อมูลบทวิจารณ์ของผู้ใช้ของงานวิจัยที่ 3.1 เพื่อแยกออกมาเป็นประเภท User Experience Improvement ด้วยตนเอง เนื่องจาก User Experience และ Features Request จากงานวิจัยที่ 3.1 นั้นมีส่วนที่คาบเกี่ยวกับการ Improvement โดยจำนวนบทวิจารณ์ของผู้ใช้ประเภท User Experience Improvement จะใกล้เคียงกับสี่ประเภทดังที่กล่าวไปแล้วข้างต้น ซึ่งหลังจากแยกประเภทบทวิจารณ์ของผู้ใช้ User Experience Improvement เรียบร้อยแล้ว ก็จะนำข้อมูลที่จำแนกได้ไปให้บุคคลอื่นอย่างน้อยสองคนทำการอ่าน และยืนยันประเภทบทวิจารณ์ของผู้ใช้อีกครั้ง เพื่อเป็นการยืนยันความน่าเชื่อถือของการจำแนกประเภทบทวิจารณ์ของผู้ใช้ User Experience Improvement

4.2 ขั้นตอนการทำ Data Cleansing และ Text Processing

ขั้นตอนนี้ถือเป็นขั้นตอนสำคัญของการทำ Data Mining และ Machine Learning เพราะส่งผลกระทบโดยตรงกับข้อมูลนำเข้าเพื่อนำไปทำการพัฒนาโมเดลการเรียนรู้ เนื่องจากข้อมูลเกี่ยวกับบทวิจารณ์ของผู้ใช้ที่เก็บรวบรวมได้จากขั้นตอนก่อนหน้านั้นมีลักษณะเป็นข้อความ ตัวอักษร ซึ่งมนุษย์สามารถอ่านและทำความเข้าใจได้จากรูปประโยค และบริบทที่กำลังพูดถึงอยู่ แต่สำหรับคอมพิวเตอร์แล้วเป็นเรื่องยากมากที่จะให้เข้าใจประโยคเหล่านั้นได้เหมือนกับมนุษย์ การทำ Data Cleansing จะช่วยกำจัดคำที่ไม่มีนัยสำคัญต่อการประมวลผลของคอมพิวเตอร์ออกไป เช่น คำสามัญทั่วไป a, an, the เป็นต้น ส่วนการทำ Text Processing คือการแยกประโยคใหญ่ ๆ ออกเป็นคำย่อย ๆ โดยแต่ละคำ สามารถนำมาวิเคราะห์ถึงสิ่งที่กำลังพูดถึงได้ เช่น Rhythm Sathorn Condominium is one of the most beautiful condominiums that stays next to Chao Phraya River. ก็จะได้เป็น Rhythm | Sathorn | Condominium | is | one | of | the | most | beautiful | condominiums | that | stays | next | to | Chao | Phraya | River | . ซึ่งช่วยให้คอมพิวเตอร์สามารถเข้าใจความหมายจากการวิเคราะห์คำแต่ละคำได้ง่ายขึ้น โดยเทคนิคเกี่ยวกับ Data Cleansing และ Text Processing ที่คาดว่าจะนำมาใช้มีดังนี้

4.2.1 **Spelling Correction** ทำการแก้ไขคำที่สะกดผิดของผู้ใช้งานให้ถูกต้องโดยใช้ไลบรารี PYENCHANT [23] เช่น beautifal แก้ไขเป็น beautiful เป็นต้น

4.2.2 **Contractions Expansion** ทำการแทนที่คำย่อ หรือคำที่ลดรูปด้วยคำเต็ม เช่น don’t จะถูกแทนที่ด้วย do not เป็นต้น

4.2.3 **Nouns and Verbs Filtering** ใช้หลักการ Part-of-Speech Tagging [24] เพื่อเลือกเอาเฉพาะคำนามและคำกริยาในประโยคเนื่องจากส่วนใหญ่จะเป็นคำที่มีความหมาย และสื่อถึงบริบทของประโยคได้

4.2.4 **Tokenization** ทำการแยกประโยคออกเป็นคำย่อย ๆ โดยจะมีการกรองตัวเลข และเครื่องหมายวรรคตอน (,) ออก เนื่องจากไม่ค่อยมีความสำคัญในบริบทของประโยค

4.2.5 **Singularization** ใช้ฟังก์ชัน singularization ของไลบรารี TEXTBLOB [25] ทำการแปลงคำต่าง ๆ ให้อยู่ในรูปเอกพจน์ทั้งหมด เช่น talks เป็น talk เป็นต้น

4.2.6 **Stopword Removal** ทำการเอาคำทั่วไปที่ปรากฏบ่อย ๆ เช่น a, an, the ออก

4.2.7 **Stemming** ทำการเปลี่ยนรูปคำศัพท์ให้อยู่ในรูปรากศัพท์ เช่น pushed กลายเป็น push เพื่อลดจำนวน Token ที่ต้องใช้ในการประมวลผลลง

4.2.8 **Short Tokens Removal** ทำการนำ Token ที่มีน้อยกว่าสามตัวอักษรออก เพราะคำจำพวกนี้มักจะเป็นคำเชื่อม หรือคำทั่วไปที่ปรากฏอยู่ในประโยค ซึ่งมักจะไม่มีประโยชน์ต่อการวิเคราะห์

4.2.9 **Short Documents Removal** ทำการตัดบทวิจารณ์ของผู้ใช้ที่มี Token น้อยกว่าสามคำออก เนื่องจากไม่สามารถสื่อถึงความหมายของผู้ใช้งานได้เพียงพอ

ซึ่งผลลัพธ์ที่ได้จากขั้นตอนนี้ จะเป็นเซ็ตของ Tokens ที่เรียกว่า Bag-of-Words ของแต่ละบทวิจารณ์ของผู้ใช้เพื่อเตรียมนำไปเป็นข้อมูลนำเข้าของโมเดลการเรียนรู้แบบ Supervised Learning

4.3 ขั้นตอนการพัฒนาโมเดลเพื่อจำแนกประเภทของบทวิจารณ์ของผู้ใช้

ขั้นตอนนี้จะเป็นการนำข้อมูลบทวิจารณ์ของผู้ใช้ที่ผ่านกระบวนการ Data Cleansing และ Text Processing จำนวน 4,000 บทวิจารณ์ของผู้ใช้ในขั้นตอนก่อนหน้ามาทำการสร้างโมเดลการเรียนรู้แบบ Supervised Learning เพื่อใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้ว่าเป็น Features Request, Bug Reports, User Experience Improvement, หรืออื่น ๆ ด้วยเทคนิคต่าง ๆ พร้อมทั้งเหตุผลที่เลือกใช้ ดังต่อไปนี้

1) Multinomial Naïve Bayes เพราะมีประสิทธิภาพสูง รวมไปถึงใช้ Training Data Set ในปริมาณน้อย เปเปอร์ที่เกี่ยวข้องหลาย ๆ เปเปอร์ก็ใช้เทคนิคนี้เป็น Baseline
2) Linear Support Vector Machine เป็นหนึ่งในเทคนิคพื้นฐานที่มีประสิทธิภาพดีที่สุดในการทำ Text Classification
3) Logistic Regression เป็น Classification Algorithm ที่สามารถทำการ Predict Multiple Classes ได้ดี อัลกอรึทึมง่ายต่อการทำความเข้าใจ
4) BoW with Keras เป็นไลบรารี Deep Learning ของภาษา Python ที่ทำงานบน TensorFlow ของ Google ซึ่งผู้วิจัยคาดว่าจะมีประสิทธิภาพสูง จึงได้เลือกมาทำการทดลองด้วย

Features ที่จะใช้เป็นข้อมูลในการเทรนโมเดลอย่างน้อย มีดังต่อไปนี้

1) Rating หมายถึง คะแนนที่ผู้ใช้ให้กับแอปพลิเคชัน ซึ่งมีค่าตั้งแต่ 1 คือ ไม่พึงพอใจมากที่สุด ถึง 5 คือ พึงพอใจมากที่สุด
2) Title หมายถึง ข้อความที่เป็นหัวข้อของบทวิจารณ์ของผู้ใช้ (ถ้ามี)
3) Text หมายถึง คำอธิบายของบทวิจารณ์ของผู้ใช้
4) User Review Sentiment หมายถึง ลักษณะของบทวิจารณ์ของผู้ใช้ว่าเป็นเชิงบวก (Positive) หรือเชิงลบ (Negative)

โดย Features เหล่านี้ได้มาจากงานวิจัยที่ 3.1 แต่ทางผู้วิจัยจะทำ Feature Extraction เพิ่มเติมจากบทวิจารณ์ของผู้ใช้เพื่อให้ได้ Features ใหม่ ๆ มาเทรนโมเดลโดยใช้ Training Data ชุดเดียวกับงานวิจัยที่ 3.1 แต่ Features แตกต่างกันออกไป โดยคาดหวังว่าจะได้โมเดลที่มีประสิทธิภาพสูงกว่าโมเดลที่ดีที่สุดของงานวิจัยที่ 3.1

4.4 ขั้นตอนการทดสอบโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

ในการทดสอบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ของแต่ละโมเดลที่ได้จากการใช้ Classification Algorithm แตกต่างกันนั้น จะใช้ค่า Precision, Recall, F-Measure มาเป็นตัววัด

ประสิทธิภาพ โดยในส่วนของ Test Data จะใช้ K-Fold Cross Validation ทำการแบ่ง Training Data บางส่วนมาใช้เป็น Test Data เช่น 5-Fold Cross Validation ก็จะทำการแบ่ง Training Data ออกเป็น 5 ส่วน แต่ละส่วนมีจำนวนข้อมูลเท่า ๆ กัน โดยใช้ข้อมูลหนึ่งส่วนเป็นตัววัดประสิทธิภาพของโมเดล และใช้ข้อมูลสี่ส่วนที่เหลือเป็นข้อมูลการเรียนรู้ให้กับโมเดล ทำเช่นนี้ไปจนครบ 5 รอบ เป็นต้น

4.5 ขั้นตอนการเปรียบเทียบประสิทธิภาพของโมเดลการจำแนกประเภทของบทวิจารณ์ของผู้ใช้

หลังจากที่วัดประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้ครบทุกโมเดลที่นำมาใช้ในการทดลองแล้ว ก็จะนำผลลัพธ์ของแต่ละโมเดลมาเปรียบเทียบกัน โดยดูจากค่า Precision, Recall, และ F-Measure หลังจากนั้นเลือกโมเดลที่มีประสิทธิภาพดีที่สุดเพื่อนำไปเทียบกับผลลัพธ์ของงานวิจัยที่ 3.1 อันที่ดีที่สุด โดยประสิทธิภาพของโมเดลใหม่จะต้องมีประสิทธิภาพสูงกว่างานวิจัยที่ 3.1 เพื่อนำมาใช้เป็นโมเดลในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ ที่จะนำไปสร้างเป็นทิคเก็ต Jira

4.6 ขั้นตอนการปรับปรุงประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้

ในการปรับปรุงประสิทธิภาพของโมเดลการจำแนกประเภทบทวิจารณ์ของผู้ใช้นั้น จะใช้ Ensemble Learning ในการนำ Classification Model ที่ได้จากขั้นตอนก่อนหน้าหลาย ๆ แบบ มาใช้งานร่วมกัน หลังจากนั้นทำการวัดประสิทธิภาพของโมเดลการเรียนรู้ใหม่อีกครั้ง โดยคาดหวังว่าโมเดลที่ได้จะมีประสิทธิภาพสูงขึ้น และต้องสูงกว่างานวิจัยที่ 3.1 เช่นเดิม

4.7 ขั้นตอนการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้

หลังจากได้โมเดลที่มีประสิทธิภาพดีที่สุดในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ ก็จะนำบทวิจารณ์ของผู้ใช้ที่ยังไม่เคยถูกแยกประเภทมาแยกประเภทผ่านโมเดล ซึ่งจะได้ผลลัพธ์ออกมาเป็นบทวิจารณ์ของผู้ใช้จำนวนหนึ่งที่ระบุประเภทว่าเป็น Features Request, Bug Reports, User Experience Improvement, หรือ Others ซึ่งก่อนที่นำบทวิจารณ์ของผู้ใช้เหล่านี้ไปสร้างเป็นทิคเก็ต Jira จะทำการตรวจสอบความซ้ำซ้อนของบทวิจารณ์ของผู้ใช้ก่อนโดยใช้วิธีทางด้าน Text Similarity หลาย ๆ วิธี เช่น

1) Syntactic Similarity คือ การตรวจสอบความคล้ายคลึงกันของเอกสารสองเอกสารใด ๆ โดยใช้คำที่ปรากฏอยู่ในเอกสารมาคำนวณ ตัวอย่างวิธีการในกลุ่มนี้ เช่น Jaccard Similarity, Cosine Similarity, Euclidean Distance
2) Semantic Similarity คือ การตรวจสอบความคล้ายคลึงกันของเอกสารสองเอกสารใด ๆ โดยการแปลความหมายของเอกสารแต่ละเอกสารว่าสื่อถึงเรื่องเดียวกันหรือไม่ ตัวอย่างวิธีการในกลุ่มนี้ เช่น Latent Semantic Analysis, Siamese-LSTM

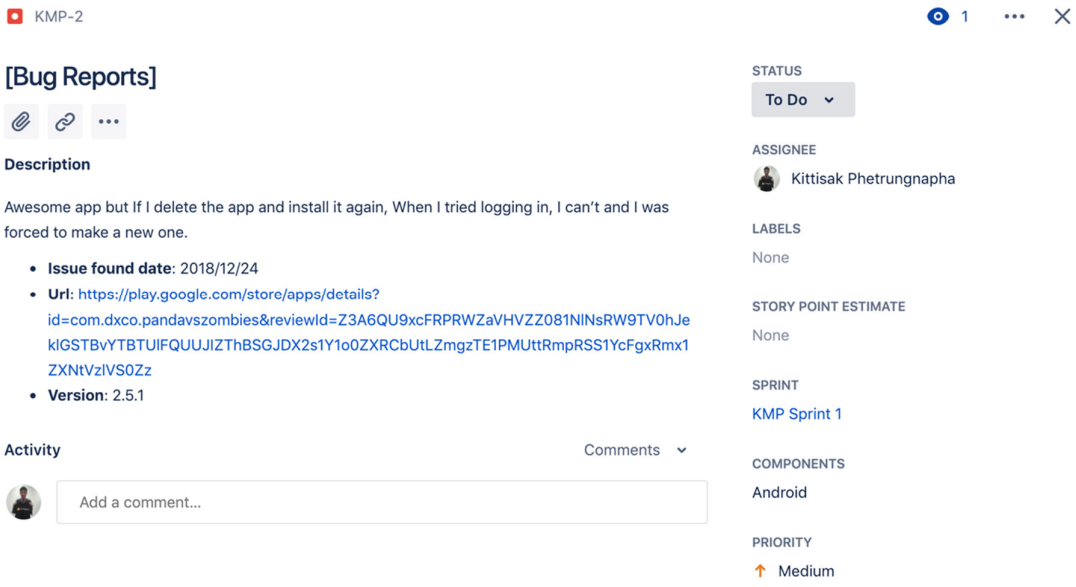
ตัวอย่างวิธีการวัดความคล้ายคลึงกันของบทวิจารณ์ของผู้ใช้สองอันใด ๆ โดยใช้ Cosine Similarity ทำได้โดยเตรียมข้อมูลอีกชุดหนึ่งสำหรับการวัด Cosine Similarity เพื่อทำการหาว่าค่า Threshold ที่ดีที่สุดที่ทำให้ค่า Precision, Recall, และ F-Measure สูงมากที่สุดคือเท่าใด หลังจากนั้นจะนำค่า Threshold ดังกล่าวมาเป็น Baseline โดยถ้าค่า Cosine Similarity มีค่ามากกว่าหรือเท่ากับค่านี้ก็จะถือว่าบทวิจารณ์ที่อยู่ในประเภทเดียวกันมี

ความซ้ำซ้อนกัน ซึ่งจะเลือกบทวิจารณ์ของผู้ใช้ที่เป็นประเภทเดียวกันและมีค่า Cosine Similarity มากกว่าหรือเท่ากับค่า Threshold เพียงอันเดียวไปสร้างเป็นทิคเก็ต Jira เพื่อป้องกันปัญหาทิคเก็ตซ้ำซ้อนบน Jira Dashboard

ส่วน Semantic Similarity ใช้ประโยชน์ได้ในกรณีที่เอกสารทั้งสองเอกสารใด ๆ มีความยาวแตกต่างกันมาก หรือความยาวใกล้เคียงกัน แต่มีการใช้คำที่มีความหมายเหมือนกันแต่เขียนต่างกันมาสร้างประโยค เช่น Car กับ Vehicle ซึ่งถ้าใช้ Syntactic Similarity เพียงอย่างเดียวจะพบว่าเอกสารทั้งสองมีโอกาสที่จะแตกต่างกันสูงมาก ทั้งที่ในความเป็นจริงเอกสารทั้งสองนั้นอาจจะสื่อถึงเรื่องเดียวกัน

4.8 ขั้นตอนการเชื่อมต่อกับ APIs ของ Jira เพื่อทำการสร้างทิคเก็ต

ขั้นตอนนี้จะเลือกแอปพลิเคชันที่ดาวน์โหลดฟรีและมีจำนวนผู้ใช้งานสูงบนแอปสโตร์หรือเพลย์สโตร์ มาเป็นกรณีศึกษาหนึ่งแอปพลิเคชันมาทำการแยกประเภทบทวิจารณ์ของผู้ใช้เป็น Features Request, Bug Reports, User Experience Improvement, และ Others จากนั้นทำการสร้าง Jira Dashboard ขึ้นมา และนำบทวิจารณ์ของผู้ใช้ที่จำแนกประเภทแล้วไปเชื่อมต่อกับ APIs ของ Jira เพื่อทำการสร้างทิคเก็ตแบบอัตโนมัติ โดยประเภทของทิคเก็ตจะแยกตามประเภทของบทวิจารณ์ของผู้ใช้ซึ่งจะมีสามประเภทคือ Features Request, Bug Reports, User Experience Improvement ส่วนประเภท Others จะไม่นำมาสร้างทิคเก็ต เนื่องจากไม่ได้ก่อให้เกิดประโยชน์ต่อบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาในกระบวนการทำงานแบบเอไจล์ ตัวอย่าง Bug Reports ที่สามารถนำมาสร้างทิคเก็ตที่เป็น Issue ประเภท Bug Reports ได้ดังภาพที่ 4.2



ภาพที่ 4.2 ตัวอย่างทิคเก็ต Issue ประเภท Bug Reports ที่จะถูกสร้างขึ้นจากเครื่องมือในโครงงานมหาบัณฑิต

ซึ่งเครื่องมือที่จะทำการสร้างทิคเก็ตจากบทวิจารณ์ของผู้ใช้ที่จะพัฒนาขึ้นในโครงงานมหาบัณฑิตนี้ จะอยู่ในรูปของ Command Line Program ซึ่งสามารถทำงานได้ทั้งบน Window, OS X, Linux โดย Input ที่ต้องใส่เข้า

ไปมีเพียงแค่ Bundle ID ใน iOS หรือ Package Name ใน Android เท่านั้น ซึ่งบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระต้องรู้ค่าเหล่านี้อยู่แล้ว จากนั้นเครื่องมือจะใช้ Bundle ID หรือ Package Name เป็นพารามิเตอร์ทำการดึงบทวิจารณ์ของผู้ใช้บนแอปสโตร์หรือเพลย์สโตร์มา แล้วใช้โมเดลการเรียนรู้ทำการจำแนกประเภทของบทวิจารณ์ของผู้ใช้ หลังจากนั้นตัวโปรแกรมจะให้ทำการใส่ Credential ของ Jira เพื่อทำการ Authentication เข้าไปสร้างทิคเก็ตแยกตามประเภทของบทวิจารณ์ของผู้ใช้ให้แบบอัตโนมัติ หลังจากนั้นบริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระก็สามารถที่จะเข้าไปตรวจสอบใน Jira Dashboard เพื่อดูรายละเอียดของทิคเก็ตที่ถูกสร้างขึ้นจากบทวิจารณ์ของผู้ใช้แยกตามประเภท Features Request, Bug Reports, User Experience Improvement ได้เพื่อจะได้หยิบทิคเก็ตไปทำการอิมพลีเมนต์ต่อไป

4.9 ขั้นตอนการตรวจสอบทิคเก็ตที่ถูกสร้างในเอไจล์บอร์ดของ Jira

ขั้นตอนนี้จะเป็นการตรวจสอบทิคเก็ตที่ถูกสร้างจากข้อมูลบทวิจารณ์ของผู้ใช้ที่ได้จำแนกประเภทไปแล้วในข้อ 4.8 ที่อยู่บน Jira Dashboard โดยจำนวนทิคเก็ตที่สร้างขึ้นต้องเท่ากับจำนวนบทวิจารณ์ของผู้ใช้ที่แตกต่างกันและต้องสร้างได้ถูกต้อง 100% ตามประเภทของบทวิจารณ์ของผู้ใช้นั้น ๆ ด้วย โดยวิธีการตรวจสอบจะทำโดยผู้วิจัยเอง

5. วัตถุประสงค์

5.1 เพื่อพัฒนาโมเดลที่ใช้จำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์และเพลย์สโตร์ของแอปพลิเคชันที่ต้องการได้โดยมีประสิทธิภาพสูง

5.2 เพื่อพัฒนาเครื่องมือที่นำบทวิจารณ์ของผู้ใช้ที่จำแนกประเภทเป็น Features Request, Bug Reports, User Experience Improvement ไปสร้างเป็นทิคเก็ตบน Jira โดยประเภทของทิคเก็ตที่ถูกสร้างจะเป็นไปตามประเภทของบทวิจารณ์ของผู้ใช้

6. ขอบเขตการดำเนินงาน

6.1 สร้างโมเดลที่สามารถจำแนกประเภทของบทวิจารณ์ของผู้ใช้บนแอปสโตร์หรือเพลย์สโตร์ ออกเป็นประเภท Bug Reports, Features Request, User Experience Improvement, และ Others ได้ โดยประเภท Others นั้นจะไม่ได้ระบุว่าเกี่ยวข้องกับอะไร เพียงแต่แยกบทวิจารณ์ของผู้ใช้ที่ไม่เข้าพวกในสามกลุ่มแรกมาไว้ในกลุ่มนี้เท่านั้น

6.2 โมเดลที่จำแนกประเภทของบทวิจารณ์ของผู้ใช้รองรับแค่ภาษาอังกฤษเท่านั้น

6.3 เครื่องมือที่ใช้สร้าง Jira Ticket จากบทวิจารณ์ของผู้ใช้ต้องสามารถสร้างทิคเก็ตที่แยกตามประเภทของบทวิจารณ์ของผู้ใช้ที่ได้จากโมเดลอย่างถูกต้อง เช่น ทิคเก็ตประเภท Bug Reports, ทิคเก็ตประเภทของ Features Request, ทิคเก็ตประเภท User Experience Improvement เป็นต้น

6.4 Classification Model ที่จะใช้ คือ Multinomial Naïve Bayes, Linear Support Vector Machine, Logistic Regression, และ BoW with Keras

6.5 Features ที่จะใช้ในการเรียนรู้โมเดลอย่างน้อย คือ Rating, Title & Text (Bags of Word), และ User Review Sentiment และ Features ใหม่ ๆ ที่ได้จาก Feature Extraction จากบทวิจารณ์ของผู้ใช้ที่ผู้วิจัยจะเพิ่มเติมเข้ามา

6.6 บทวิจารณ์ของผู้ใช้ใด ๆ จะถูกจัดให้อยู่ในประเภทใดประเภทหนึ่งได้เพียงประเภทเดียวนั้น

6.7 เครื่องมือที่ใช้สร้าง Jira Ticket จากบทวิจารณ์ของผู้ใช้สามารถรองรับการตรวจสอบทิคเก็ตซ้ำซ้อนได้

6.8 Field บางอันที่อยู่ภายในทิคเก็ตเช่น ความสำคัญของทิคเก็ต (Priority) นั้นจะมีค่าเริ่มต้นเป็น ปานกลาง (Medium) เท่านั้น

6.9 ประเมินประสิทธิภาพของโมเดล โดยใช้ค่า Precision, Recall, และ F-Measure

6.10 ในการประเมินความถูกต้องของเครื่องมือสร้าง Jira Ticket ที่ใช้บทวิจารณ์ของผู้ใช้แบบอัตโนมัติเป็นข้อมูลนำเข้า ผู้วิจัยจะทำการตรวจสอบแบบ Manual ใน Dashboard ของ Jira ด้วยตนเอง

7. ขั้นตอนการดำเนินงาน

7.1 ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการทำ User Reviews Data Mining บนแอปสโตร์หรือเพลย์สโตร์

7.2 ศึกษาทฤษฎีที่เกี่ยวข้องกับการทำ Automate Jira Ticket, APIs ของ Jira ที่เปิดให้ใช้งาน

7.3 ศึกษาภาษา Python และไลบรารีที่เกี่ยวข้องกับการทำ Data Mining, Machine Learning

7.4 ทำการรวบรวมข้อมูลบทวิจารณ์ของผู้ใช้ที่แบ่งประเภทเรียบร้อยแล้วจากงานวิจัยที่ 3.1 มาเป็น Training Set

7.5 ทำการเทรนโมเดลโดยใช้ข้อมูลบทวิจารณ์ของผู้ใช้ที่ได้จากขั้นตอนก่อนหน้า

7.6 ทดสอบการจำแนกประเภทบทวิจารณ์ของผู้ใช้ของโมเดลที่ได้สร้างขึ้นโดยใช้ K-Fold Cross Validation

7.7 วัดผลประสิทธิภาพในการจำแนกประเภทบทวิจารณ์ของผู้ใช้ โดยใช้ค่า Precision, Recall, และ F-Measure

7.8 ปรับปรุงโมเดลให้มีประสิทธิภาพในการจำแนกประเภทบทวิจารณ์ของผู้ใช้มากขึ้น

7.9 ทำการคำนวณความคล้ายคลึงกันของบทวิจารณ์ของผู้ใช้เพื่อกรองบทวิจารณ์ที่ซ้ำซ้อนออก

7.10 พัฒนาเครื่องมือที่ใช้ในการสร้าง Jira Ticket จากบทวิจารณ์ของผู้ใช้ที่ผ่านการจำแนกประเภทเรียบร้อยแล้ว

7.11 ประเมินความถูกต้อง ครบถ้วน สมบูรณ์ของ Jira Ticket ใน Jira dashboard

7.12 จัดทำและนำเสนอบทความทางวิชาการ

7.13 สรุปผลแนวทางการวิจัย ข้อเสนอแนะ และจัดทำโครงงานฉบับสมบูรณ์

8. ประโยชน์ที่คาดว่าจะได้รับ

8.1 ได้โมเดลการเรียนรู้ที่สามารถจำแนกบทวิจารณ์ของผู้ใช้ของแอปพลิเคชันใด ๆ บนแอปสโตร์หรือเพลย์สโตร์ได้อย่างมีประสิทธิภาพ โดยแยกเป็นประเภทต่าง ๆ ได้แค่ Bug Reports, Features Request, User Experience Improvement, และ Others

8.2 ได้เครื่องมือสำหรับทำการสร้าง Jira Tickets จากบทวิจารณ์ของผู้ใช้ของแอปพลิเคชันใด ๆ บนแอปสโตร์หรือ เพลย์สโตร์ แยกตามประเภทของบทวิจารณ์ของผู้ใช้

8.3 ช่วยลดระยะเวลาในการเก็บรวบรวมความคิดเห็น รายการความต้องการของผู้ใช้งานที่อยู่บนแอปสโตร์หรือเพลย์สโตร์ในรูปแบบของบทวิจารณ์ของผู้ใช้

8.4 ช่วยลดระยะเวลาในการสร้าง Jira Tickets ลง ส่งผลให้บริษัทพัฒนาซอฟต์แวร์หรือนักพัฒนาอิสระสามารถมุ่งเน้นที่การทำการปรับปรุง แก้ไข เพิ่มเติมแอปพลิเคชันของตนเองให้ดียิ่งขึ้น

9. รายการอ้างอิง

[1] Wikipedia. Smartphone [Online]. Available: https://en.wikipedia.org/wiki/สมาร์ตโฟน. Last Accessed: 23 Dec 2018.

[2] Apple. App Store [Online]. Available: https://www.apple.com/th/ios/app-store/. Last Accessed: 23 Dec 2018.

[3] Google. Play Store [Online]. Available: https://play.google.com/store?hl=en. Last Accessed: 23 Dec 2018.

[4] Wikipedia. User Reviews [Online]. Available: https://en.wikipedia.org/wiki/User_review. Last Accessed: 23 Dec 2018.

[5] Atlassian. Agile [Online]. Available: https://www.atlassian.com/agile/. Last Accessed: 15 Jan 2019.

[6] Berkeley. Artificial Intelligence [Online]. Available: http://aima.cs.berkeley.edu/. Last Accessed: 15 Jan 2019.

[7] ACM. Data Mining [Online]. Available: https://dl.acm.org/citation.cfm?id=2778285/. Last Accessed: 15 Jan 2019.

[8] Carnegie Mellon University. Machine Learning [Online]. Available: http://www.cs.cmu.edu/~tom/mlbook.html/. Last Accessed: 15 Jan 2019.

[9] Eakasit Pacharawongsakda, Ph.D. Naïve Bayes introduction [Online]. Available: http://dataminingtrend.com/2014/naive-bayes/. Last Accessed: 20 Dec 2018.

[10] TAVISH SRIVASTAVA. Basics of Ensemble Learning Explained in Simple English [Online]. Available: https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/. Last Accessed: 1 Jan 2019.

[11] Atlassian. JIRA Software [Online]. Available: https://www.atlassian.com/software/jira/. Last Accessed: 15 Jan 2019.

[12] Atlassian. Jira Cloud platform Developer [Online]. Available: https://developer.atlassian.com/cloud/jira/platform/rest/v3/. Last Accessed: 22 Dec 2018.

[13] Wikipedia. Issue Tracking System [Online]. Available: https://en.wikipedia.org/wiki/Issue_tracking_system/. Last Accessed: 7 Jan 2019.

[14] Scrum.org. What is a Product Backlog? [Online]. Available: https://www.scrum.org/resources/what-is-a-product-backlog/. Last Accessed: 7 Jan 2019.

[15] Atlassian. What is a board? [Online]. Available: https://www.scrum.org/resources/what-is-a-product-backlog/. Last Accessed: 9 Jan 2019.

[16] Walid Maalej, Hadeer Nabil, “Bug Report, Feature Request, or Simply Praise? On Automatically Classifying App Reviews”, 2015 IEEE 23rd International Requirements Engineering Conference (RE), pages: 116-125, 2016.

[17] Xiaozhou LI, Zheying ZHANG, and Kostas STEFANIDIS, “Mobile App Evolution Analysis based on User Reviews”, September 2018.

[18] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E. Hassan, “What Do Mobile App Users Complain About?”, IEEE Journals & Magazines, pages: 70-77, 2015

[19] Maleknaz Nayebi, Homayoon Farrahi, Guenther Ruhe, and Henry Cho, “App Store Mining is Not Enough”, 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pages: 152-154, 2017.

[20] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang. “Ar-miner: mining informative reviews for developers from mobile app marketplace”. In Proceedings of the 36th International Conference on Software Engineering, 2014.

[21] Jirawat Karanwittayakarn. Automatically create Jira Issue from Firebase Crashlytics [Online]. Available: https://medium.com/firebasethailand/%E0%B8%A1%E0%B8%B2%E0%B8%AA%E0%B8%A3%E0%B9%89%E0%B8%B2%E0%B8%87-jira-issues-%E0%B8%88%E0%B8%B2%E0%B8%81-crash-%E0%B8%97%E0%B8%B5%E0%B9%88%E0%B9%80%E0%B8%81%E0%B8%B4%E0%B8%94%E0%B8%82%E0%B8%B6%E0%B9%89%E0%B8%99%E0%B9%83%E0%B8%99-firebase-crashlytics-%E0%B8%81%E0%B8%B1%E0%B8%99%E0%B9%80%E0%B8%96%E0%B8%AD%E0%B8%B0-4b9d979e0212. Last Accessed: 31 Dec 2018.

[22] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. “Recommending and Localizing Change Requests for Mobile Apps based on User Reviews”. In Proceedings of the 39th IEEE International Conference on Software Engineering (ICSE 2017), Buenos Aires, Argentina, 20 May 2017 - 25 May 2017.

[23] Ryan Kelly. Pyenchant [Online]. Available: https://github.com/rfk/pyenchant. Last Accessed: 28 Dec 2018.

[24] Sachin Malhotra, and Divya Godayal. An introduction to part-of-speech tagging and the Hidden Markov Model [Online]. Available: https://medium.freecodecamp.org/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24. Last Accessed: 28 Dec 2018.

[25] Steven Loria. TextBlob [Online]. Available: https://github.com/sloria/textblob. Last Accessed: 28 Dec 2018.

บรรณานุกรม

ประวัติผู้เขียน

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