ระบบตรวจจับคำหยาบจากหน้าจอ

โดย

นายสรทรัพย์ นาคสวัสดิ์

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
ปีการศึกษา 2568

ระบบตรวจจับคำหยาบจากหน้าจอ

โดย

นายสรทรัพย์ นาคสวัสดิ์

โครงงานพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
ปีการศึกษา 2568

SCREEN-BASED PROFANITY DETECTION SYSTEM

BY

Soratrub Nakswat

A FINAL-YEAR PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE

COMPUTER SCIENCE

FACULTY OF SCIENCE AND TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2025

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงงานพิเศษ

ของ

สรทรัพย์ นาคสวัสดิ์

เรื่อง

ระบบตรวจจับคำหยาบจากหน้าจอ

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
เมื่อ วันที่ 19 ธันวาคม พ.ศ. 2568

อาจารย์ที่ปรึกษา _____
(อ.ดร.ภัคพร เสาร์ฝั้น)

กรรมการสอบโครงงานพิเศษ _____
(ผศ. ดร.ฐาปนา บุญชู)

กรรมการสอบโครงงานพิเศษ _____
(ผศ. ดร.ประภาพร รัตนชำรง)

มหาวิทยาลัยธรรมศาสตร์
คณะวิทยาศาสตร์และเทคโนโลยี

รายงานโครงงานพิเศษ

ของ

สรทรัพย์ นาคสวัสดิ์
เรื่อง

ระบบตรวจคำหยาบจากหน้าจอ

ได้รับการตรวจสอบและอนุมัติ ให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
หลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
เมื่อ วันที่ 19 ธันวาคม พ.ศ. 2568

อาจารย์ที่ปรึกษา _____
(อ.ดร.ภัคพร เสาร์ฝั้น)

กรรมการสอบโครงงานพิเศษ _____
(ผศ. ดร.ฐาปนา บุญชู)

กรรมการสอบโครงงานพิเศษ _____
(ผศ. ดร.ประภาพร รัตนชำรง)

หัวข้อโครงงานพิเศษ	ระบบตรวจจับคำหยาบจากหน้าจอ
ชื่อผู้เขียน	สรทรัพย์ นาคสวัสดิ์
ชื่อปริญญา	วิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
สาขาวิชา/คณะ/มหาวิทยาลัย	สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์
อาจารย์ที่ปรึกษาโครงงานพิเศษ	อ.ดร.ภัคพร เสาร์ฝั้น
ปีการศึกษา	2568

บทคัดย่อ

ปัจจุบันแพลตฟอร์มสตรีมมิงเป็นที่นิยมอย่างแพร่หลาย แต่ผู้สตรีมมักประสบปัญหาในการควบคุมคำหยาบในแชทที่ปรากฏบนหน้าจอ เนื่องจากระบบกรองคำหยาบที่มีอยู่ในแพลตฟอร์มยังไม่ครอบคลุมทุกภาษาและบริบท โดยเฉพาะภาษาไทยและคำหยาบที่มีการดัดแปลงรูปแบบ

โครงงานนี้นำเสนอการพัฒนาระบบตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch โดยเชื่อมต่อกับ Twitch IRC API เพื่อดึงข้อความแชทแบบ Real-time ซึ่งให้ความแม่นยำสูง ระบบประกอบด้วย 3 ส่วนหลัก ได้แก่ โมดูลเชื่อมต่อและดึงข้อความจาก Twitch IRC โมดูลตรวจจับคำหยาบด้วยเทคนิค Dictionary-based Pattern Matching ร่วมกับ Word Segmentation สำหรับภาษาอังกฤษ และระบบแจ้งเตือนผู้ใช้แบบ Real-time ระบบใช้โครงสร้าง Multi-threading ด้วย PyQt5 QThread เพื่อประสิทธิภาพในการประมวลผล รองรับการตรวจจับคำหยาบทั้งภาษาไทยและภาษาอังกฤษ พร้อมความสามารถในการจัดการคำหยาบที่มีการดัดแปลงด้วยสัญลักษณ์พิเศษหรือการเขียนติดกัน

คำสำคัญ: ตรวจจับคำหยาบ, Pattern Matching, Word Segmentation, Twitch IRC, Real-time Chat Processing

Thesis Title	SCREEN-BASED PROFANITY DETECTION SYSTEM
Author	Soratrub Naksawat
Degree	Bachelor of Science
Major Field/Faculty/University	Computer Science Faculty of Science and Technology Thammasat University
Project Advisor	Dr.Pakkaporn Saophan
Academic Years	2025

ABSTRACT

Streaming platforms have become increasingly popular, but streamers often struggle to monitor and control inappropriate language in their chat sections, as existing profanity filtering systems do not comprehensively cover all languages and contexts, particularly Thai language and modified forms of profanity.

This project presents the development of a profanity detection system for the Twitch platform by connecting to the Twitch IRC API to retrieve chat messages in real-time, which provides high accuracy. The system consists of three main components: a module for connecting and retrieving messages from Twitch IRC, a profanity detection module using Dictionary-based Pattern Matching techniques combined with Word Segmentation for English, and a real-time user notification system. The system utilizes a Multi-threading structure with PyQt5 QThread for processing efficiency. It supports profanity detection in both Thai and English, with the capability to handle profanity modified with special characters or written as concatenated words.

Keywords : Profanity detection, Pattern Matching, Word Segmentation, Twitch IRC, Real-time Chat Processing

กิตติกรรมประกาศ

โครงงานพิเศษเรื่อง "ระบบตรวจจับคำหยาบจากหน้าจอ" นี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความกรุณาอย่างสูงจาก อ.ดร.ภัคพร เสาร์ฝั้น อาจารย์ที่ปรึกษาโครงงาน ที่กรุณาให้คำแนะนำและข้อคิดเห็นต่างๆ อันเป็นประโยชน์อย่างยิ่งในการทำโครงงาน อีกทั้งยังช่วยแก้ไขปัญหาและให้แนวทางในการพัฒนาระบบตลอดระยะเวลาการทำโครงงาน รวมถึงการตรวจสอบแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่อย่างดียิ่ง ผู้จัดทำรู้สึกซาบซึ้งในความกรุณาและขอขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบพระคุณคณะกรรมการสอบโครงงานพิเศษทุกท่าน ที่ได้กรุณาให้คำแนะนำที่มีคุณค่า และช่วยปรับปรุงแก้ไขโครงงานให้มีความสมบูรณ์ยิ่งขึ้น

สุดท้ายนี้ ขอกราบขอบพระคุณบิดา มารดา และครอบครัว ที่ให้การสนับสนุนและเป็นกำลังใจที่สำคัญยิ่ง จนทำให้การศึกษาครั้งนี้ประสบความสำเร็จลุล่วงไปได้ด้วยดี

นายสรทรัพย์ นาคสวัสดิ์

สารบัญ

		หน้า
บทคัดย่อ		2
ABSTRACT		4
กิตติกรรมประกาศ		5
สารบัญ		6
สารบัญตาราง		9
สารบัญภาพ		10
รายการสัญลักษณ์และคำย่อ		11
บทที่ 1	บทนำ	1
1.1	ความเป็นมาและความสำคัญของโครงงาน	1
1.1.1	สถานการณ์ปัจจุบันของการสตรีมมิงและการจัดการคำหยาบ	1
1.2	วัตถุประสงค์	6
1.3	ขอบเขตของโครงงาน	6
1.4	ประโยชน์ของโครงงาน	8
1.5	ข้อจำกัดของโครงงาน	9
บทที่ 2	วรรณกรรมและงานวิจัยที่เกี่ยวข้อง	11
2.1	แนวคิดทฤษฎีที่เกี่ยวข้อง	11
2.1.1	โปรโตคอล Internet Relay Chat (IRC)	11
2.1.2	การประมวลผลภาษาธรรมชาติ (Natural Language Processing)	12
2.1.3	การเขียนโปรแกรมแบบ Multi-threading	14

2.1.4 สถาปัตยกรรม GUI และการโปรแกรมเชิงเหตุการณ์ 15
2.2 งานวิจัยที่เกี่ยวข้อง 16
2.2.1 งานวิจัยด้านการตรวจจับข้อความที่ไม่เหมาะสม 16
2.2.2 การสร้างฐานข้อมูลคำหยาบภาษาไทย 16
2.2.3 การวิเคราะห์พฤติกรรมผู้ใช้ใน Twitch Chat 17
บทที่ 3 วิธีการวิจัย 18
3.1 ภาพรวมของโครงงาน 18
3.1.1 สถาปัตยกรรมของระบบ 19
3.1.2 Use case Diagram 22
3.2 การวิเคราะห์ขอบเขตและความต้องการของระบบ 23
3.2.1 ความต้องการเชิงฟังก์ชัน 23
3.2.2 ความต้องการที่ไม่ใช่เชิงฟังก์ชัน 24
3.2.3 ข้อจำกัดของระบบ 25
3.3 การดำเนินงาน 26
3.3.1 การออกแบบและพัฒนาระบบ 26
3.3.1.1 Use Case Specification 26
3.3.1.2 การออกแบบส่วนติดต่อผู้ใช้ 34
3.3.1.3 การออกแบบการเชื่อมต่อ Twitch IRC 35
3.3.1.4 การออกแบบโมดูลวิเคราะห์ภาษา 36
3.3.1.5 การออกแบบการทำงานแบบ Multi-threading 37
3.3.1.6 การออกแบบการจัดการข้อมูล 38
3.3.2 Process Flow Chart Diagram 40
3.3.3 เครื่องมือและเทคโนโลยีที่ใช้ 46
3.4 วิธีการทดสอบระบบ 47
3.4.1 การทดสอบความถูกต้องของฟังก์ชัน 47

3.4.2 การทดสอบประสิทธิภาพ 48
3.4.3 การทดสอบความแม่นยำ 48
3.5 สรุปวิธีการดำเนินงาน 49
บทที่ 4 ผลการดำเนินงาน 50
4.1 ผลการทดสอบความถูกต้องและประสิทธิภาพ 50
4.1.1 ผลการทดสอบความถูกต้องของฟังก์ชัน 50
4.1.2 ผลการทดสอบประสิทธิภาพ 52
4.2 ผลการทดสอบความแม่นยำ 53
บทที่ 5 สรุป 54
5.1 สรุปผลการดำเนินงาน 54
5.2 อภิปรายผลการทดลอง 55
5.2.1 การวิเคราะห์ความซับซ้อนของอัลกอริทึม 55
5.3 ข้อเสนอแนะสำหรับการพัฒนาต่อยอด 56
รายการอ้างอิง 58
ภาคผนวก 60
ภาคผนวก ก. ชื่อภาคผนวก 61
ภาคผนวก ข. ชื่อภาคผนวก 63
ภาคผนวก ค. ชื่อภาคผนวก 65

สารบัญตาราง

หน้า

ตารางที่ 3.1 Use Case Specification 26

ตารางที่ 4.1 สรุปผลการประเมินประสิทธิภาพของระบบตรวจจับคำหยาบ 54

สารบัญภาพ

หน้า

ภาพที่ 2.1 แผนภาพสถาปัตยกรรม Client-Server ของ IRC 12
ภาพที่ 2.2 ขั้นตอนการประมวลผลข้อความ 14
ภาพที่ 2.3 การทำงานของ Single-thread และ Multi-thread 15
ภาพที่ 3.1 สถาปัตยกรรมของระบบ 19
ภาพที่ 3.2 Use case Diagram 22
ภาพที่ 3.3 การออกแบบส่วนติดต่อผู้ใช้ 34
ภาพที่ 3.4 Flow chart ส่วนที่ 1 - การเริ่มต้นและการรับข้อความ 40
ภาพที่ 3.5 Flow chart ส่วนที่ 2 - การตรวจจับคำหยาบภาษาไทย 41
ภาพที่ 3.6 Flow chart ส่วนที่ 3 - การตรวจจับคำหยาบภาษาอังกฤษ 42
ภาพที่ 3.7 Flow chart ส่วนที่ 4 - การจัดการเมื่อพบคำหยาบและการแจ้งเตือน 44
ภาพที่ 4.1 หน้าจอหลักของโปรแกรม Twitch Bad Word Detector 50
ภาพที่ 4.2 หน้าต่าง Dashboard แสดงสถิติการตรวจจับ 51

รายการสัญลักษณ์และคำย่อ

สัญลักษณ์/คำย่อ	คำเต็ม/คำจำกัดความ
NLP	Natural Language Processing
IRC	Internet Relay Chat
GUI	Graphical User Interface
API	Application Programming Interface
CSV	Comma-Separated Values
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
TCP/IP	Transmission Control Protocol / Internet Protocol
UI	User Interface

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในยุคที่การสตรีมมิงเนื้อหาผ่านแพลตฟอร์มออนไลน์ เช่น Twitch, YouTube Live, Facebook Gaming และ TikTok Live เติบโตอย่างรวดเร็ว การมีปฏิสัมพันธ์กับผู้ชมผ่านแชทสดถือเป็นองค์ประกอบสำคัญที่สร้างความผูกพันระหว่างผู้สตรีมและชุมชนของตน อย่างไรก็ตาม แพลตฟอร์มสตรีมมิงเหล่านี้มักประสบปัญหาเกี่ยวกับการจัดการเนื้อหาที่ไม่เหมาะสมในแชท โดยเฉพาะอย่างยิ่งคำหยาบและคำที่สร้างความเกลียดชัง

ถึงแม้ว่าแพลตฟอร์มส่วนใหญ่จะมีระบบกรองคำหยาบอยู่แล้ว แต่ระบบเหล่านี้ยังมีข้อจำกัดหลายประการ ได้แก่ 1) มักรองรับเฉพาะภาษาหลักอย่างภาษาอังกฤษ ทำให้ไม่สามารถตรวจจับคำหยาบในภาษาอื่นๆ เช่น ภาษาไทยได้อย่างมีประสิทธิภาพ 2) ไม่สามารถรับมือกับการดัดแปลงคำหยาบในรูปแบบต่างๆ เช่น การสะกดผิดโดยเจตนา การใช้ตัวเลขแทรกตัวอักษร หรือการใช้สัญลักษณ์พิเศษ 3) มีข้อจำกัดเฉพาะตัวกับแพลตฟอร์มอื่น โดยฟีเจอร์ของระบบกรองแตกต่างกันส่งผลให้ผู้สตรีมที่ต้องออกอากาศบนหลายแพลตฟอร์มต้องเรียนรู้และปรับตัวกับเครื่องมือที่แตกต่างกัน

ปัญหาเหล่านี้ส่งผลกระทบโดยตรงต่อผู้สตรีมทั้ง Twitch และแพลตฟอร์มอื่น โดยเฉพาะผู้ที่มีผู้ชมจำนวนมาก การไม่สามารถควบคุมเนื้อหาที่ไม่เหมาะสมอาจส่งผลเสียต่อภาพลักษณ์ ทำให้ผู้ชมกลุ่มเป้าหมาย เช่น เยาวชนหรือครอบครัวหลีกเลี่ยงการรับชม นอกจากนี้ การปรากฏของคำหยาบหรือข้อความที่สร้างความเกลียดชังบนหน้าจอยังอาจนำไปสู่การละเมิดนโยบายของแพลตฟอร์ม ซึ่งอาจส่งผลให้ถูกตักเตือน หรือในกรณีร้ายแรง อาจถูกระงับบัญชีชั่วคราวหรือถาวร กระทบต่อรายได้และอาชีพของผู้สตรีม

1.1.1 สถานการณ์ปัจจุบันของการสตรีมมิงและการจัดการคำหยาบ

การสตรีมมิงออนไลน์กลายเป็นอุตสาหกรรมที่เติบโตอย่างรวดเร็วและมีอิทธิพลในโลกดิจิทัล จากการสำรวจในปี 2023 พบว่ามีผู้ใช้งานแพลตฟอร์ม Twitch มากกว่า 140 ล้านคนต่อเดือน YouTube Gaming มีผู้ชมมากกว่า 100 ล้านคน และ Facebook Gaming มีผู้ชมประมาณ 76 ล้านคน ในประเทศไทย จำนวนผู้สตรีมเติบโตขึ้นกว่า 200% ในช่วง 3 ปีที่

ผ่านมา โดยมีผู้สตรีมมืออาชีพมากกว่า 10,000 คน และมีผู้ชมรวมกว่า 15 ล้านคน สะท้อนให้เห็นถึงความนิยมและการเติบโตของอุตสาหกรรมนี้

นอกจากนี้ จากการสำรวจของสมาคมผู้สตรีมไทยในปี 2567 พบว่า 92% ของผู้สตรีมเคยประสบปัญหาเกี่ยวกับคำหยาบในแชทที่ไม่ได้รับการกรอง โดย 78% ระบุว่าระบบกรองอัตโนมัติของแพลตฟอร์มทำงานได้ไม่ดีกับภาษาไทย ผลการวิเคราะห์แชทจากแพลตฟอร์มสตรีมมิงหลักพบว่ามีข้อความที่มีคำหยาบภาษาไทยหลุดรอดการกรองถึง 65% เทียบกับภาษาอังกฤษที่มีเพียง 25% เท่านั้น

รายงานจากบริษัทวิเคราะห์ดิจิทัลคอนเทนต์ StreamAnalytics (2567) แสดงให้เห็นว่าในช่วงปีที่ผ่านมา มีจำนวนข้อความที่มีคำหยาบในแพลตฟอร์มสตรีมมิงเพิ่มขึ้น 43% และปัญหานี้รุนแรงกว่าในประเทศที่ใช้ภาษาที่ไม่ใช่ภาษาอังกฤษ โดยเฉพาะในประเทศไทยที่พบปัญหานี้สูงเป็นอันดับ 3 ของเอเชีย แนวโน้มการเติบโตของอุตสาหกรรมสตรีมมิงในประเทศไทยคาดว่าจะเพิ่มขึ้น 35% ต่อปีในช่วง 3 ปีข้างหน้า (รายงานดิจิทัลคอนเทนต์ไทยแลนด์, 2567) ทำให้ปัญหาการจัดการคำหยาบในแชทเป็นความท้าทายที่สำคัญมากขึ้น

(1) ปัญหาการจัดการคำหยาบในแพลตฟอร์มสตรีมมิง

การจัดการคำหยาบและเนื้อหาที่ไม่เหมาะสมในแชทเป็นความท้าทายสำคัญสำหรับผู้สตรีม โดยเฉพาะในบริบทของประเทศไทยที่ระบบกรองอัตโนมัติของแพลตฟอร์มไม่ได้รับการออกแบบมาให้รองรับภาษาไทยอย่างมีประสิทธิภาพ จากการสำรวจผู้สตรีมชาวไทยจำนวน 100 คนในปี 2023 พบว่า 85% ประสบปัญหาข้อความที่ไม่เหมาะสมในแชทอย่างน้อยสัปดาห์ละครั้ง และ 67% เคยมีประสบการณ์ที่คำหยาบหรือข้อความไม่เหมาะสมปรากฏบนหน้าจอสตรีมโดยไม่ทันสังเกต

ปัญหาดังกล่าวไม่เพียงส่งผลกระทบต่อประสบการณ์ของผู้ชมเท่านั้น แต่ยังส่งผลโดยตรงต่อผู้สตรีมในแง่ของชื่อเสียงและรายได้ ดังเห็นได้จากกรณีศึกษาในเดือนมีนาคม 2567 เมื่อผู้สตรีมชื่อดังชาวไทยรายหนึ่งถูกระงับบัญชีชั่วคราวเป็นเวลา 7 วัน เนื่องจากมีคำหยาบปรากฏบนหน้าจอสตรีมของเขาโดย

ไม่ได้ตั้งใจ แม้ว่าเขาจะใช้ระบบกรองคำหยาบของแพลตฟอร์ม แต่คำหยาบภาษาไทยที่ถูกดัดแปลงการสะกดไม่ได้ถูกตรวจจับ ส่งผลให้เขาสูญเสียรายได้ประมาณ 50,000 บาทในช่วงที่ถูกระงับบัญชี

ผลสำรวจจากผู้สตรีมมืออาชีพชาวไทย 50 คน พบว่า 35% เคยสูญเสียสัญญาการสนับสนุนจากแบรนด์ เนื่องจากมีคำหยาบปรากฏในสตรีมของพวกเขา โดย 82% ของกรณีเหล่านี้เกิดจากการที่ระบบกรองไม่สามารถตรวจจับคำหยาบภาษาไทยที่มีการดัดแปลงการสะกด จากการประเมินของสมาคมผู้ประกอบการดิจิทัลคอนเทนต์ไทย (2567) ระบุว่าผู้สตรีมไทยสูญเสียโอกาสทางธุรกิจคิดเป็นมูลค่ากว่า 150 ล้านบาทต่อปี เนื่องจากปัญหาเกี่ยวกับการจัดการคำหยาบในแชทที่ไม่มีประสิทธิภาพ

(2) ความไม่มีประสิทธิภาพของระบบกรองอัตโนมัติสำหรับภาษาไทย

ระบบกรองอัตโนมัติของแพลตฟอร์มสตรีมมิงหลักมักถูกพัฒนาโดยบริษัทต่างชาติและออกแบบมาสำหรับภาษาอังกฤษเป็นหลัก ทำให้มีข้อจำกัดในการตรวจจับคำหยาบภาษาไทย ข้อมูลจากการศึกษาของศูนย์วิจัยดิจิทัลคอนเทนต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (2023) แสดงให้เห็นว่าระบบกรองอัตโนมัติของแพลตฟอร์มหลักสามารถตรวจจับคำหยาบภาษาไทยได้เพียง 43-56% ในขณะที่สามารถตรวจจับคำหยาบภาษาอังกฤษได้ถึง 78-85%

ภาษาไทยมีความท้าทายพิเศษในการตรวจจับคำหยาบด้วยระบบอัตโนมัติ เนื่องจากลักษณะเฉพาะของภาษา ได้แก่:

1. การไม่มีช่องว่างระหว่างคำทำให้การแบ่งคำทำได้ยาก
2. ความซับซ้อนของการวางตำแหน่งสระและวรรณยุกต์
3. วัฒนธรรมการสร้างคำใหม่และการเล่นคำที่มีพลวัตสูง

4. รูปแบบการดัดแปลงคำที่หลากหลายเพื่อหลบเลี่ยงการตรวจจับ เช่น การใช้ตัวเลขแทนตัวอักษร การสลับตำแหน่งสระ หรือการแทรกอักษรที่ไม่ออกเสียงระหว่างพยัญชนะ

การศึกษาจากศูนย์วิจัยเทคโนโลยีภาษาไทย มหาวิทยาลัยเกษตรศาสตร์ (2566) พบว่า คำหยาบภาษาไทยมีรูปแบบการดัดแปลงได้มากกว่า 20 รูปแบบ ทำให้ระบบกรองทั่วไปซึ่งใช้วิธีการตรวจจับแบบพื้นฐานไม่สามารถรองรับได้อย่างมีประสิทธิภาพ

(3) การหลบเลี่ยงระบบกรองโดยการดัดแปลงคำ

ปัญหาอีกประการหนึ่งคือการดัดแปลงคำหยาบเพื่อหลบเลี่ยงระบบกรอง เช่น การสะกดผิดโดยเจตนา การใช้ตัวเลขแทรกตัวอักษรบางตัว การใช้สัญลักษณ์พิเศษคั่นระหว่างตัวอักษร หรือการใช้คำพ้องเสียง ระบบกรองทั่วไปไม่สามารถตรวจจับวิธีการเหล่านี้ได้อย่างมีประสิทธิภาพ โดยเฉพาะในภาษาไทยที่มีความซับซ้อนทางภาษาและมีวิธีการดัดแปลงคำได้หลากหลาย

ระบบกรองอัตโนมัติของแพลตฟอร์มสตรีมมิงหลัก เช่น Twitch มีข้อจำกัดในการตรวจจับคำหยาบภาษาไทย ตามการศึกษาของศูนย์วิจัยดิจิทัลคอนเทนต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี (2023) พบว่าระบบ AutoMod ของ Twitch สามารถตรวจจับคำหยาบภาษาไทยได้เพียง 43-56% ในขณะที่ภาษาอังกฤษได้ 78-85%

ที่ผ่านมา ผู้สตรีมมักแก้ไขปัญหาด้วยการจ้างผู้ดูแลแชท (moderators) หรือใช้บอทสำเร็จรูป เช่น Night Bot หรือ Stream labs Chatbot อย่างไรก็ตาม วิธีการเหล่านี้ยังมีข้อจำกัด ผู้ดูแลแชทที่เป็นมนุษย์ไม่สามารถตรวจสอบข้อความทั้งหมดได้ในช่วงที่มีการแชทหนาแน่น โดยเฉพาะในกรณีของสตรีมเมอร์ที่มีผู้ชมหลักพันหรือหลักหมื่นคน ซึ่งอาจมีข้อความในแชทเกิดขึ้นหลายร้อยข้อความต่อนาที ส่วนบอทสำเร็จรูปส่วนใหญ่ทำงานผ่าน API ของแพลตฟอร์มนั้นๆ ทำให้ไม่

สามารถทำงานข้ามแพลตฟอร์มได้อย่างมีประสิทธิภาพ และมักไม่รองรับการตรวจจับคำหยาบในภาษาไทยอย่างครอบคลุม

ความเร่งด่วนในการแก้ไขปัญหานี้เพิ่มขึ้นอย่างมากในปัจจุบัน เนื่องจากปัจจัยหลายประการ:

1. การเข้มงวดมากขึ้นของนโยบายแพลตฟอร์มต่อเนื้อหาที่ไม่เหมาะสม โดยมีบทลงโทษที่รุนแรงขึ้น
2. การแข่งขันที่สูงขึ้นในวงการสตรีมที่ทำให้ผู้สตรีมต้องรักษาภาพลักษณ์ที่เหมาะสม
3. ความคาดหวังที่เพิ่มขึ้นจากผู้สนับสนุนและพันธมิตรทางธุรกิจเกี่ยวกับคุณภาพของเนื้อหา
4. การขยายกลุ่มผู้ชมไปสู่กลุ่มวัยที่หลากหลายมากขึ้น รวมถึงเยาวชนและครอบครัว

นอกจากนี้ ผลกระทบทางสังคมจากการปรากฏของคำหยาบในสตรีมยังรวมถึงการสร้างสภาพแวดล้อมที่เป็นพิษในชุมชนออนไลน์ การสร้างตัวอย่างที่ไม่ดีให้กับผู้ชมที่เป็นเยาวชน และการลดทอนคุณค่าของเนื้อหาที่มีคุณภาพ การศึกษาจากสำนักงานคณะกรรมการดิจิทัลเพื่อเศรษฐกิจและสังคมแห่งชาติ (2566) พบว่า 62% ของผู้ปกครองมีความกังวลเกี่ยวกับการที่บุตรหลานจะได้รับอิทธิพลจากคำหยาบในการสตรีมออนไลน์

โครงงานนี้จึงนำเสนอแนวทางใหม่ในการแก้ไขปัญหาดังกล่าวด้วยการพัฒนาระบบตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch โดยเชื่อมต่อกับ Twitch IRC (Internet Relay Chat) API โดยตรง เพื่อดึงข้อความแชทแบบ Real-time ซึ่งให้ความแม่นยำสูงสุดและมีประสิทธิภาพสูง ระบบใช้เทคนิค Dictionary-based Pattern Matching ร่วมกับเทคนิค Word Segmentation สำหรับการวิเคราะห์ข้อความภาษาอังกฤษ เพื่อให้สามารถตรวจจับคำหยาบที่ถูกดัดแปลงรูปแบบต่างๆ เช่น การใช้ตัวเลขแทนตัวอักษร (shi1t) หรือการเขียนติดกัน (thisisabadword) ที่ผู้ใช้สร้างขึ้นเพื่อหลบเลี่ยงการกรองได้อย่างมีประสิทธิภาพ

ด้วยประสิทธิภาพสูงในการตรวจจับและการประมวลผลแบบ Real-time ระบบตรวจจับคำหยาบสำหรับ Twitch นี้จะช่วยให้ผู้สตรีมสามารถ

จัดการกับเนื้อหาที่ไม่เหมาะสมในแชทได้อย่างมีประสิทธิภาพมากขึ้น ลดภาระในการสังเกตแชทด้วยตนเอง และลดความเสี่ยงในการแสดงเนื้อหาที่ไม่เหมาะสมบนหน้าจอ ซึ่งจะช่วยปกป้องภาพลักษณ์ของผู้สตรีม สร้างสภาพแวดล้อมที่ปลอดภัยและเหมาะสมสำหรับชุมชนผู้ชม และป้องกันการละเมิดนโยบายของแพลตฟอร์ม Twitch ที่อาจนำไปสู่การถูกระงับบัญชี ทั้งนี้ ระบบยังมีความสำคัญในการช่วยแก้ปัญหาข้อจำกัดของระบบ AutoMod ของ Twitch โดยให้ประสิทธิภาพในการตรวจจับคำหยาบภาษาไทยที่สูงกว่า ระบบ AutoMod ของ Twitch

1.2 วัตถุประสงค์

โครงงานนี้มีเป้าหมายเพื่อพัฒนาระบบตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch ที่มีประสิทธิภาพสำหรับผู้สตรีม ช่วยแก้ปัญหาข้อจำกัดของระบบกรองที่มีอยู่ในปัจจุบัน และเพิ่มประสิทธิภาพในการตรวจจับคำหยาบในภาษาไทย เพื่อให้บรรลุเป้าหมายดังกล่าว จึงกำหนดวัตถุประสงค์ของโครงงานดังต่อไปนี้

1. พัฒนาระบบต้นแบบในการตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch โดยเชื่อมต่อ กับ Twitch IRC API เพื่อดึงข้อความแชทแบบ Real-time
2. พัฒนาและปรับแต่งระบบตรวจจับคำหยาบโดยใช้เทคนิค Dictionary-based Pattern Matching ร่วมกับ Word Segmentation สำหรับการวิเคราะห์ข้อความจากแชท Twitch
3. พัฒนาส่วนติดต่อผู้ใช้ (User Interface) ที่ใช้งานง่าย ไม่ซับซ้อน และสามารถปรับแต่งได้ตามความต้องการของผู้สตรีม
4. ทดสอบประสิทธิภาพและความแม่นยำของระบบที่พัฒนาขึ้นกับแพลตฟอร์ม Twitch โดยตรง

1.3 ขอบเขตของโครงงาน

โครงงานนี้มุ่งเน้นการพัฒนาระบบตรวจจับคำหยาบ**สำหรับแพลตฟอร์ม Twitch** สำหรับผู้สตรีม โดยมีขอบเขตดังนี้

1. ด้านระบบและเทคโนโลยี
    - เชื่อมต่อกับ Twitch IRC API เพื่อดึงข้อความแชทแบบ Real-time
    - ใช้เทคนิค Dictionary-based Pattern Matching ร่วมกับ Word Segmentation
    - ใช้ Regular Expression สำหรับทำความสะอาดข้อความภาษาไทย
    - ใช้ PyQt5 Multi-threading สำหรับการประมวลผลแบบ Real-time
2. ด้านภาษาและเนื้อหา
    - รองรับการตรวจจับคำหยาบในภาษาไทยและภาษาอังกฤษเป็นหลัก
    - ความสามารถในการตรวจจับรูปแบบการดัดแปลงคำ เช่น การสะกดผิดโดยเจตนา การใช้ตัวเลขแทรกตัวอักษร
    - จัดทำฐานข้อมูลคำหยาบภาษาไทยและภาษาอังกฤษที่ครอบคลุมและปรับปรุงได้
3. ด้านการใช้งานและแพลตฟอร์ม
    - รองรับการทำงานกับแพลตฟอร์ม Twitch เป็นหลัก
    - พัฒนาส่วนติดต่อผู้ใช้ที่ใช้งานง่าย ปรับแต่งได้ และไม่รบกวนการสตรีม
    - ระบบแจ้งเตือนที่ผู้ใช้สามารถเลือกรูปแบบได้ เช่น เสียงเตือน
    - มี Dashboard สำหรับติดตามสถิติการตรวจจับแบบ Real-time
    - รองรับการ Export ข้อมูลเป็น CSV
    - รองรับ Dark Mode
4. ด้านการทดสอบและประเมินผล
    - ทดสอบประสิทธิภาพและความแม่นยำในการตรวจจับคำหยาบภาษาไทยและภาษาอังกฤษ
    - วัดผลกระทบต่อประสิทธิภาพของเครื่องคอมพิวเตอร์ระหว่างการใช้งานระบบร่วมกับการสตรีม

1.4 ประโยชน์ของโครงงาน

1. ประโยชน์ต่อผู้สตรีม
    - ช่วยลดภาระในการสังเกตและคัดกรองข้อความไม่เหมาะสมในแชทด้วยตนเอง
    - ลดความเสี่ยงในการแสดงเนื้อหาที่ไม่เหมาะสมบนหน้าจอ ซึ่งอาจส่งผลเสียต่อภาพลักษณ์
    - ป้องกันการละเมิดนโยบายของแพลตฟอร์มที่อาจนำไปสู่การถูกตักเตือนหรือระงับบัญชี
    - เพิ่มประสิทธิภาพในการสตรีม เนื่องจากผู้สตรีมสามารถโฟกัสกับเนื้อหาได้มากขึ้น
2. ประโยชน์ต่อชุมชนผู้ชม
    - สร้างสภาพแวดล้อมที่ปลอดภัยและเหมาะสมสำหรับผู้ชมทุกวัย
    - ส่งเสริมวัฒนธรรมการสื่อสารที่สร้างสรรค์และเคารพซึ่งกันและกัน
    - ลดความเสี่ยงที่ผู้ชมโดยเฉพาะเยาวชนจะได้รับผลกระทบจากการเห็นเนื้อหาที่ไม่เหมาะสม
3. ประโยชน์ในเชิงเทคโนโลยีและการพัฒนา
    - สร้างฐานข้อมูลสำหรับการวิเคราะห์และเรียนรู้รูปแบบคำหยาบในภาษาไทย
    - เปิดโอกาสในการพัฒนาต่อยอดเป็นซอฟต์แวร์เชิงพาณิชย์ หรือประยุกต์ใช้ในบริบทอื่น เช่น ระบบการเรียนการสอนออนไลน์ หรือแพลตฟอร์มการสื่อสารสังคมออนไลน์
4. ประโยชน์ในเชิงวิชาการและการศึกษา
    - สนับสนุนการวิจัยและพัฒนาเกี่ยวกับการประมวลผลภาษาธรรมชาติสำหรับภาษาไทย
    - สร้างองค์ความรู้เกี่ยวกับการพัฒนาระบบที่เชื่อมต่อกับบริการภายนอกผ่าน API และการประมวลผลข้อความแบบ Real-time

5. ประโยชน์ต่อแพลตฟอร์มสตรีมมิ่ง
    - เสริมระบบกรองที่มีอยู่เดิมให้มีประสิทธิภาพมากขึ้น โดยเฉพาะสำหรับภาษาไทย
    - ลดภาระในการตรวจสอบและจัดการกับเนื้อหาที่ละเมิดนโยบาย
    - ช่วยส่งเสริมภาพลักษณ์ของแพลตฟอร์มในฐานะพื้นที่ปลอดภัยสำหรับทุกคน

1.5 ข้อจำกัดของโครงงาน

ถึงแม้ว่าโครงงานนี้จะมีประโยชน์มากมาย แต่ก็มีข้อจำกัดในด้านต่างๆ ดังนี้

1. ข้อจำกัดด้านเทคนิค
    - ความแม่นยำขึ้นอยู่กับความเสถียรของการเชื่อมต่อ Twitch IRC การประมวลผลแบบเรียลไทม์ต้องใช้ทรัพยากรของระบบ ซึ่งอาจส่งผลต่อประสิทธิภาพของคอมพิวเตอร์ที่ใช้ในการสตรีม
    - ระบบจัดเก็บเฉพาะ 200 ข้อความล่าสุด ต้องอาศัยการ Export หากต้องการเก็บข้อมูลแบบยาวนาน
    - ยังไม่รองรับการตรวจจับคำหยาบในรูปแบบของรูปภาพ สติกเกอร์ หรือไฟล์มีเดียอื่นๆ
2. ข้อจำกัดด้านความแม่นยำ
    - ความแม่นยำในการตรวจจับรูปแบบการดัดแปลงแบบซับซ้อนมีข้อจำกัด เช่น การใช้ตัวเลขและสัญลักษณ์แทนตัวอักษรนั้นเลย ต้องอาศัยการปรับปรุง ฐานข้อมูลอย่างต่อเนื่อง
    - ยังไม่สามารถตรวจจับความหมายแฝงหรือนัยยะที่ไม่เหมาะสมในบางบริบทได้อย่างสมบูรณ์
3. ข้อจำกัดด้านภาษา
    - รองรับเฉพาะภาษาไทยและภาษาอังกฤษเป็นหลัก การตรวจจับภาษาอื่นจะมีข้อจำกัด

- ความหลากหลายของคำศัพท์และสำนวนในแต่ละภูมิภาคของประเทศไทยอาจทำให้เกิดความท้าทายในการตรวจจับ
- ภาษาที่มีการเปลี่ยนแปลงอย่างรวดเร็ว เช่น คำสแลงใหม่ๆ อาจต้องมีการอัปเดตฐานข้อมูลอยู่เสมอ

4. ข้อจำกัดด้านการใช้งาน
    - ผู้ใช้ต้องมีความรู้พื้นฐานในการตั้งค่าและปรับแต่งระบบให้เหมาะสมกับความต้องการ
5. ข้อจำกัดด้านความเป็นส่วนตัวและจริยธรรม
    - การระบุว่าคำหรือเนื้อหาใดเป็นคำหยาบมีความซับซ้อนและอาจแตกต่างกันตามบริบททางวัฒนธรรมและความเชื่อส่วนบุคคล
    - การจัดเก็บข้อมูลเกี่ยวกับข้อความที่ตรวจจับต้องคำนึงถึงความเป็นส่วนตัวและการปกป้องข้อมูล
    - ระบบไม่สามารถตัดสินความเหมาะสมของเนื้อหาในทุกบริบทได้ และยังคงต้องอาศัยการตัดสินใจของมนุษย์ในบางกรณี

บทที่ 2
วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดทฤษฎีที่เกี่ยวข้อง

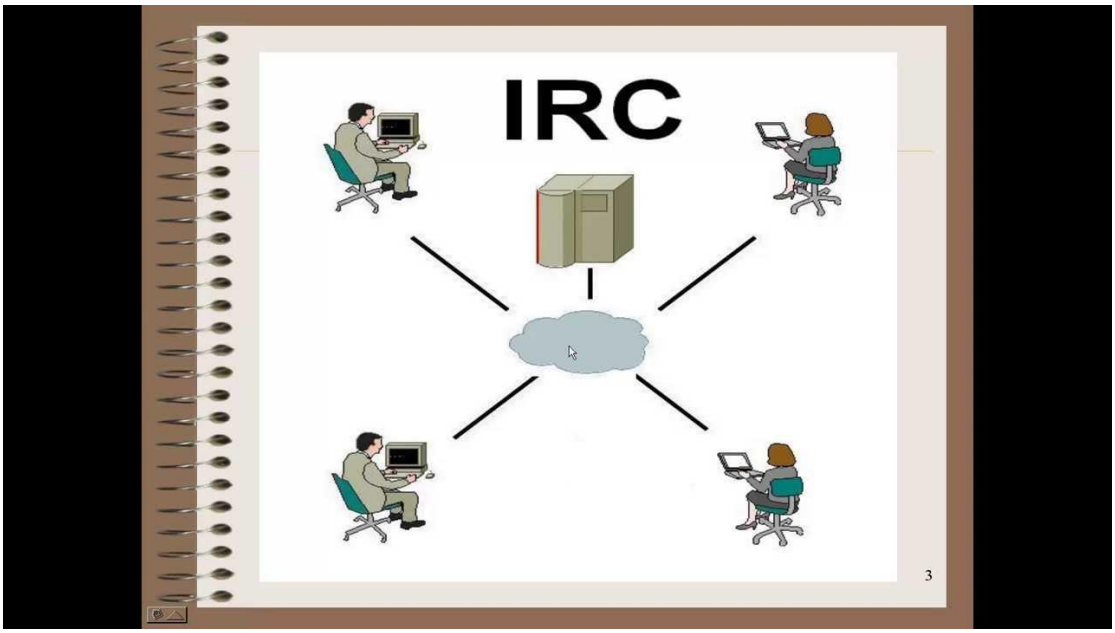
2.1.1 โปรโตคอล Internet Relay Chat (IRC)

Internet Relay Chat หรือ IRC เป็นโปรโตคอลสำหรับการสื่อสารด้วยข้อความแบบทันที (Real-time) บนเครือข่ายอินเทอร์เน็ต ถูกออกแบบมาสำหรับการสนทนากลุ่มในห้องสนทนาที่เรียกว่า "แชนแนล" (Channel) แต่ก็รองรับการสนทนาแบบหนึ่งต่อหนึ่งเช่นกัน สถาปัตยกรรมของ IRC เป็นแบบ Client-Server โดยผู้ใช้ (Client) จะต้องใช้โปรแกรม IRC Client เชื่อมต่อไปยังเซิร์ฟเวอร์ IRC (IRC Server) เพื่อเข้าร่วมแชนแนลและสื่อสารกับผู้ใช้คนอื่นๆ ที่เชื่อมต่ออยู่กับเซิร์ฟเวอร์เดียวกัน

แพลตฟอร์ม Twitch ได้นำโปรโตคอล IRC มาใช้เป็นเทคโนโลยีพื้นฐานสำหรับระบบแชทของตนเอง โดยเปิดให้นักพัฒนาสามารถเชื่อมต่อโปรแกรมภายนอกเข้ากับเซิร์ฟเวอร์แชทของ Twitch (irc.chat.twitch.tv) ได้โดยตรง การเชื่อมต่อนี้ทำให้สามารถรับข้อความแชททั้งหมดที่เกิดขึ้นในแชนแนลใดๆ ได้แบบ Real-time ซึ่งเป็นหัวใจสำคัญของโปรแกรมในโครงงานนี้ การทำงานประกอบด้วย:

1. การเชื่อมต่อ (Connection): โปรแกรมสร้างการเชื่อมต่อแบบ TCP/IP ไปยังเซิร์ฟเวอร์ของ Twitch ผ่านพอร์ต 6667 ซึ่งเป็นพอร์ตมาตรฐานสำหรับ IRC
2. การยืนยันตัวตน (Authentication): โปรแกรมส่งคำสั่ง NICK เพื่อตั้งชื่อผู้ใช้ (ในโครงงานนี้ใช้ชื่อแบบไม่ระบุตัวตน เช่น justinfan12345) และคำสั่ง JOIN เพื่อเข้าร่วมแชนแนลที่ต้องการติดตาม
3. การรับ-ส่งข้อความ: หลังจากเชื่อมต่อสำเร็จ เซิร์ฟเวอร์จะส่งข้อความทั้งหมดในแชนแนลมายังโปรแกรม ข้อความที่ผู้ใช้พิมพ์จะอยู่ในรูปแบบของคำสั่ง PRIVMSG

4. การรักษาการเชื่อมต่อ (Keep-alive): เซิร์ฟเวอร์จะส่งคำสั่ง PING มาเป็นระยะ ๆ โปรแกรมจะต้องตอบกลับด้วยคำสั่ง PONG เพื่อยืนยันว่าการเชื่อมต่อยังคงทำงานอยู่ มิฉะนั้นจะถูกตัดการเชื่อมต่อ



รูปที่ 2.1: แผนภาพสถาปัตยกรรม Client-Server ของ IRC

2.1.2 การประมวลผลภาษาธรรมชาติ (Natural Language Processing)

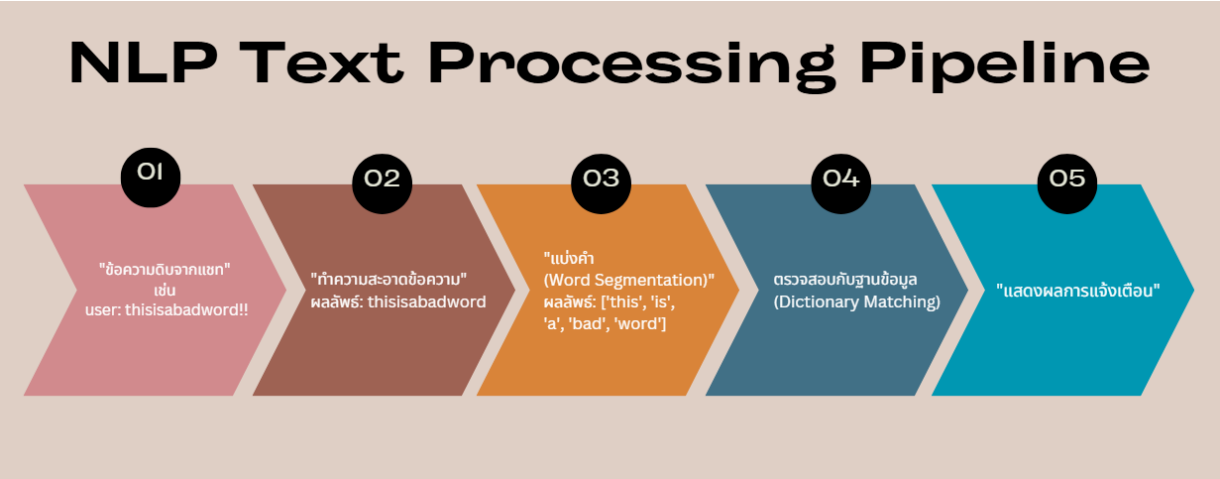
การประมวลผลภาษาธรรมชาติ เป็นสาขาหนึ่งของปัญญาประดิษฐ์ที่มุ่งเน้นการทำให้คอมพิวเตอร์สามารถเข้าใจและประมวลผลภาษามนุษย์ได้ ในโครงงานนี้ได้ประยุกต์ใช้เทคนิค NLP หลายอย่างในการตรวจจับคำหยาบ

(1) การใช้รายการคำต้องห้าม (Dictionary-based / Blacklist) เป็นวิธีการพื้นฐานที่ใช้เป็นแกนหลักของระบบ โดยสร้างฐานข้อมูลหรือรายการคำหยาบ (ในโครงงานนี้คือไฟล์ badwords.txt สำหรับภาษาไทย และ badwords_en.txt สำหรับภาษาอังกฤษ) จากนั้นนำข้อความที่ได้รับจากแชทมาเปรียบเทียบกับทุกคำในรายการ หากพบคำที่ตรงกัน ระบบ

จะแจ้งเตือนทันที วิธีนี้มีความรวดเร็ว แต่ประสิทธิภาพจะขึ้นอยู่กับความครอบคลุมของฐานข้อมูล

(2) การทำความสะอาดข้อความ (Text Cleaning) ด้วย Regular Expressions ก่อนนำข้อความไปตรวจสอบจำเป็นต้องผ่านกระบวนการทำความสะอาดเพื่อทำให้ข้อความอยู่ในรูปแบบมาตรฐานและง่ายต่อการเปรียบเทียบ ในโครงงานนี้มีการใช้ Regular Expressions ซึ่งเป็นรูปแบบนิพจน์สำหรับค้นหาและจัดการสตริง เพื่อลบอักขระพิเศษ สัญลักษณ์ หรือช่องว่างที่ไม่จำเป็นออกจากข้อความ เช่น ข้อความ "f..u@c#k" จะถูกแปลงเป็น "fuck" ก่อนนำไปตรวจสอบ

(3) การแบ่งคำ (Word Segmentation) เป็นเทคนิคขั้นสูงที่นำมาใช้เพื่อแก้ปัญหาการหลีกเลี่ยงระบบตรวจจับโดยการพิมพ์คำติดกันโดยไม่มีการเว้นวรรค เช่น "thisisabadword" ผู้ใช้มักใช้วิธีนี้เพื่อทำให้ระบบที่ตรวจจับคำแบบตรงไปตรงมาไม่สามารถทำงานได้ ในโครงงานนี้ได้ใช้ไลบรารี wordsegment สำหรับภาษาอังกฤษ ซึ่งใช้อัลกอริทึมที่อิงตามความน่าจะเป็นทางสถิติในการแบ่งข้อความที่ติดกันให้กลายเป็นคำที่มีความหมาย เช่น "thisisabadword" จะถูกแบ่งเป็น ['this', 'is', 'a', 'bad', 'word'] จากนั้นจึงนำแต่ละคำไปเปรียบเทียบกับรายการคำต้องห้ามต่อไป ซึ่งช่วยเพิ่มความแม่นยำในการตรวจจับได้อย่างมาก



รูปที่ 2.2: ขั้นตอนการประมวลผลข้อความ (Text Processing Pipeline)

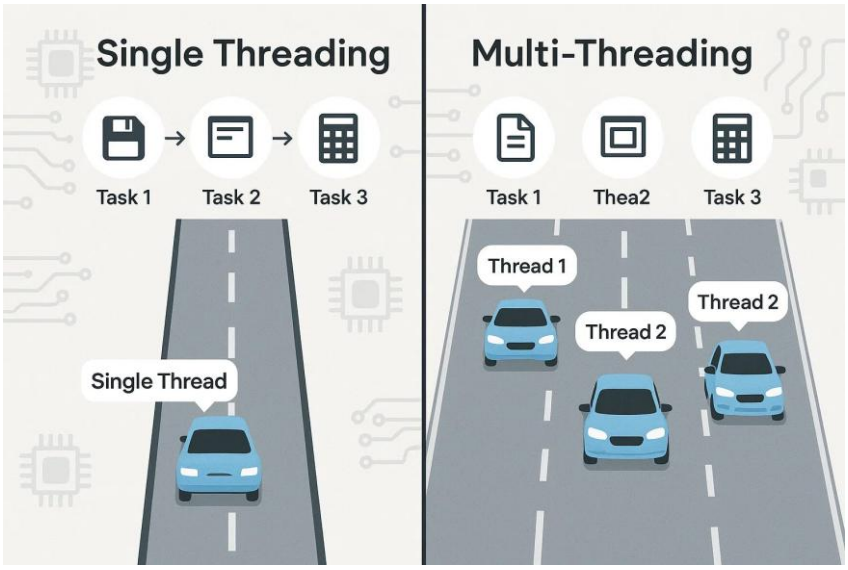
2.1.3 การเขียนโปรแกรมแบบ Multi-threading

ในแอปพลิเคชันที่มีส่วนติดต่อผู้ใช้ (GUI) การทำงานที่ใช้เวลานาน เช่น การรอรับข้อมูลจากเครือข่ายอินเทอร์เน็ต หากทำงานอยู่ใน Thread หลักเดียวกันกับ GUI จะทำให้โปรแกรมค้าง (Not Responding) และผู้ใช้ไม่สามารถโต้ตอบกับโปรแกรมได้ เพื่อแก้ปัญหานี้ จึงมีการนำหลักการ Multi-threading มาใช้

ในโครงงานนี้ได้ใช้ไลบรารี PyQt5 ซึ่งมีคลาส QThread และ QObject สำหรับจัดการ Thread โดยเฉพาะ การทำงานของระบบถูกแบ่งออกเป็น 2 ส่วนหลัก:

1. GUI Thread (Thread หลัก): ทำหน้าที่จัดการหน้าต่างโปรแกรม ปุ่ม และการแสดงผลทั้งหมด
2. Worker Thread (Thread รอง): คลาส TwitchChatWorker ถูกสร้างขึ้นเพื่อทำงานใน Thread นี้โดยเฉพาะ มีหน้าที่เชื่อมต่อกับ Twitch IRC, รอรับข้อความแชท, ประมวลผลและตรวจจับคำหยาบ เมื่อพบคำหยาบหรือได้รับข้อความใหม่ Worker Thread จะส่งสัญญาณ (Signal) กลับมายัง GUI Thread เพื่ออัปเดตการแสดงผล

การออกแบบสถาปัตยกรรมเช่นนี้ทำให้โปรแกรมสามารถรับและประมวลผลข้อความแชทได้อย่างต่อเนื่องในเบื้องหลัง โดยที่ผู้ใช้ยังคงสามารถใช้งานหน้าต่างโปรแกรมได้อย่างราบรื่น



รูปที่ 2.3 : การทำงานของ Single-thread และ Multi-thread

2.1.4 สถาปัตยกรรม GUI และการโปรแกรมเชิงเหตุการณ์ (GUI Architecture & Event-Driven Programming)

โครงงานนี้ได้เลือกใช้ PyQt5 ซึ่งเป็นไลบรารีสำหรับสร้างส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) การทำงานของโปรแกรมจึงเป็นลักษณะของ การโปรแกรมเชิงเหตุการณ์ (Event-Driven Programming) ซึ่งโปรแกรมจะรอการกระทำจากผู้ใช้ (เช่น การคลิกปุ่ม 'เชื่อมต่อ') ซึ่งเรียกว่า 'เหตุการณ์ (Event)' เมื่อเหตุการณ์เกิดขึ้น โปรแกรมจะส่ง 'สัญญาณ (Signal)' ไปยังฟังก์ชันที่กำหนดไว้ (เรียกว่า 'สล็อต (Slot)') ให้ทำงานตามคำสั่ง เช่น การเรียกใช้ฟังก์ชัน connect_twitch() เมื่อปุ่ม twitch_connect_btn ถูกคลิก กลไก Signal-Slot นี้เป็นหัวใจสำคัญที่ทำให้ส่วนติดต่อผู้ใช้สามารถโต้ตอบกับส่วนการทำงานเบื้องหลังได้อย่างเป็นระบบ

2.2 งานวิจัยที่เกี่ยวข้อง

2.2.1 งานวิจัยด้านการตรวจจับข้อความที่ไม่เหมาะสม

การตรวจจับคำหยาบและข้อความแสดงความเกลียดชัง (Hate Speech) เป็นหัวข้อวิจัยที่ได้รับความสนใจอย่างมากในปัจจุบัน มีงานวิจัยจำนวนมากที่พยายามพัฒนาระบบกรองเนื้อหาสำหรับโซเชียลมีเดียและแพลตฟอร์มสตรีมมิ่ง ตัวอย่างเช่น งานวิจัยของ K.M. Hartl และคณะ (2022) ได้ทำการวิเคราะห์ข้อความแชทจากแพลตฟอร์ม Twitch จำนวนมากเพื่อสร้างแบบจำลองในการจำแนกประเภทของข้อความที่เป็นพิษ (Toxic Messages) โดยพบว่าการใช้โมเดลการเรียนรู้เชิงลึก (Deep Learning) สามารถตรวจจับข้อความที่มีความหมายเชิงลบได้แม่นยำกว่าการใช้เพียงรายการคำต้องห้าม ซึ่งชี้ให้เห็นถึงแนวทางการพัฒนาต่อยอดจากโครงงานนี้ในอนาคต

2.2.2 การสร้างฐานข้อมูลคำหยาบภาษาไทย

ประสิทธิภาพของระบบที่ใช้วิธี Dictionary-based ขึ้นอยู่กับความสมบูรณ์ของฐานข้อมูลคำหยาบโดยตรง ในประเทศไทยยังมีงานวิจัยด้านการรวบรวมและสร้างฐานข้อมูลคำหยาบภาษาไทยอย่างเป็นระบบค่อนข้างจำกัด อย่างไรก็ตาม มีโครงการอย่าง "ThaiTox" โดย สุรเดช และคณะ (2019) ที่พยายามสร้างฐานข้อมูลคำหยาบภาษาไทยที่ครอบคลุมและมีการจัดหมวดหมู่ตามระดับความรุนแรง ซึ่งฐานข้อมูลจากงานวิจัยลักษณะนี้สามารถนำมาปรับปรุงและต่อยอดฐานข้อมูลที่ใช้ในโครงงานนี้ (badwords.txt) ให้มีประสิทธิภาพสูงขึ้นได้

2.2.3 การวิเคราะห์พฤติกรรมผู้ใช้ใน Twitch Chat

นอกจากการตรวจจับคำหยาบแล้ว ยังมีงานวิจัยที่มุ่งเน้นการวิเคราะห์ปฏิสัมพันธ์และพฤติกรรมของผู้ใช้ใน Twitch Chat เพื่อทำความเข้าใจวัฒนธรรมชุมชนออนไลน์ เช่น งานวิจัยของ A. B. Massanari (2017) ที่ศึกษาการใช้ Emote และคำสแลงเฉพาะกลุ่มใน Twitch ซึ่งแสดงให้เห็นว่าการสื่อสารในแชทมีความซับซ้อนและเปลี่ยนแปลงอย่างรวดเร็ว งานวิจัยเหล่านี้ชี้ให้เห็นถึงความท้าทายในการพัฒนาระบบตรวจจับคำหยาบที่ต้องปรับตัวให้ทันกับบริบทและคำศัพท์ใหม่ๆ ที่เกิดขึ้นในชุมชนอยู่เสมอ

บทที่ 3
วิธีการวิจัย

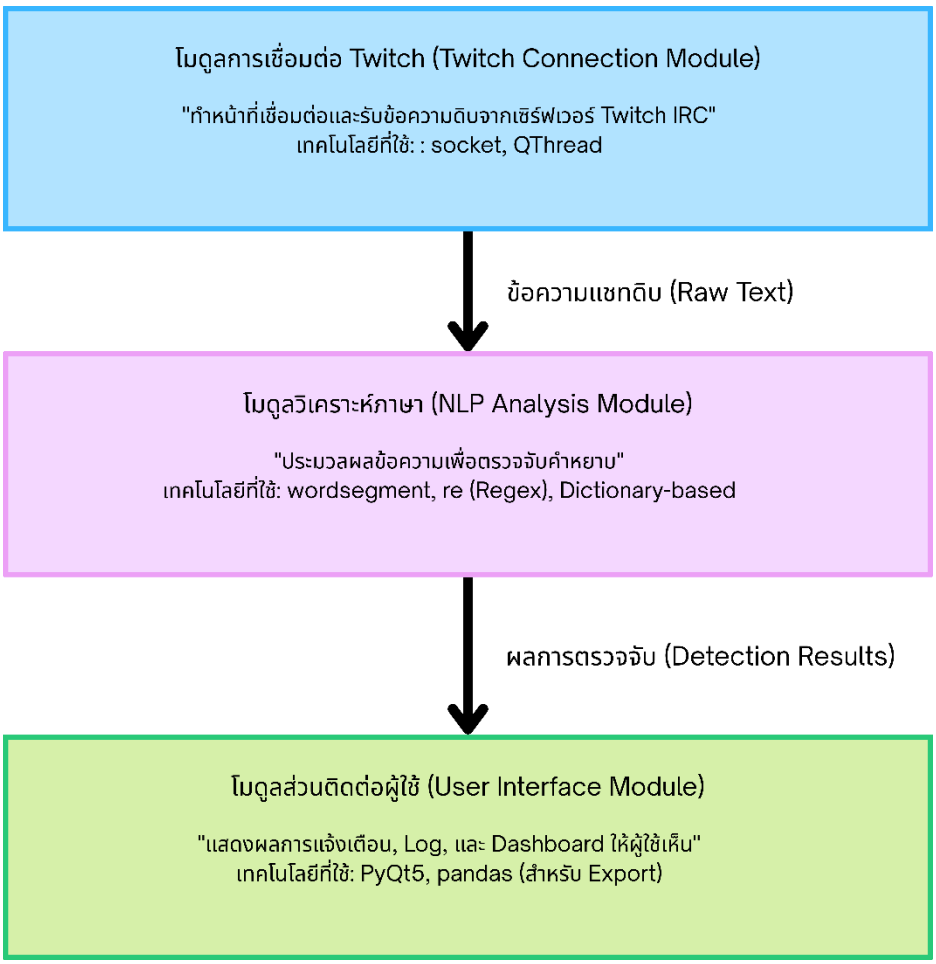
3.1 ภาพรวมของโครงงาน

โครงงานนี้มุ่งพัฒนาระบบตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch โดยเชื่อมต่อกับเซิร์ฟเวอร์แชทโดยตรงเพื่อดึงข้อความแบบ Real-time จากนั้นนำมาวิเคราะห์ด้วยเทคนิคการประมวลผลภาษาธรรมชาติ (NLP) และแสดงผลการแจ้งเตือนผ่านส่วนติดต่อผู้ใช้ (GUI) ที่พัฒนาขึ้นโดยเฉพาะ การทำงานของระบบประกอบด้วย 4 ขั้นตอนหลัก ดังนี้:

1. การเชื่อมต่อกับ Twitch IRC (Twitch IRC Connection): โปรแกรมจะสร้างการเชื่อมต่อแบบ Socket ไปยังเซิร์ฟเวอร์ irc.chat.twitch.tv ของ Twitch ทำการยืนยันตัวตนแบบไม่ระบุชื่อ และเข้าร่วมแชนแนลที่ผู้ใช้กำหนด
2. การรับและถอดรหัสข้อความ (Message Reception & Decoding): เมื่อเชื่อมต่อสำเร็จ โปรแกรมจะเข้าสู่สถานะรอรับข้อมูล เมื่อมีข้อความใหม่ในแชท เซิร์ฟเวอร์จะส่งข้อมูลมายังโปรแกรม ซึ่งจะถูกถอดรหัสและสกัดออกมาเฉพาะชื่อผู้ใช้และเนื้อหาข้อความ
3. การวิเคราะห์และตรวจจับคำหยาบด้วย NLP (NLP Analysis & Detection): ข้อความที่ได้รับจะถูกส่งไปประมวลผลตามขั้นตอนของ NLP ได้แก่ การทำความสะอาดข้อความ, การแบ่งคำ (สำหรับภาษาอังกฤษ) และการเปรียบเทียบกับฐานข้อมูลคำหยาบ
4. การแสดงผลและการแจ้งเตือน (Display & Notification): หากตรวจพบคำหยาบ ระบบจะส่งสัญญาณกลับมายังส่วนติดต่อผู้ใช้เพื่อแสดงการแจ้งเตือน, บันทึกข้อมูลลงใน Log และอัปเดตสถิติต่างๆ บน Dashboard

3.1.1 สถาปัตยกรรมของระบบ

ระบบได้รับการออกแบบโดยแบ่งการทำงานออกเป็น 3 โมดูลหลักที่ทำงานประสานกัน เพื่อให้ง่ายต่อการพัฒนาและบำรุงรักษา ดังแสดงในภาพที่ 3.1



ภาพที่ 3.1 : สถาปัตยกรรมของระบบ

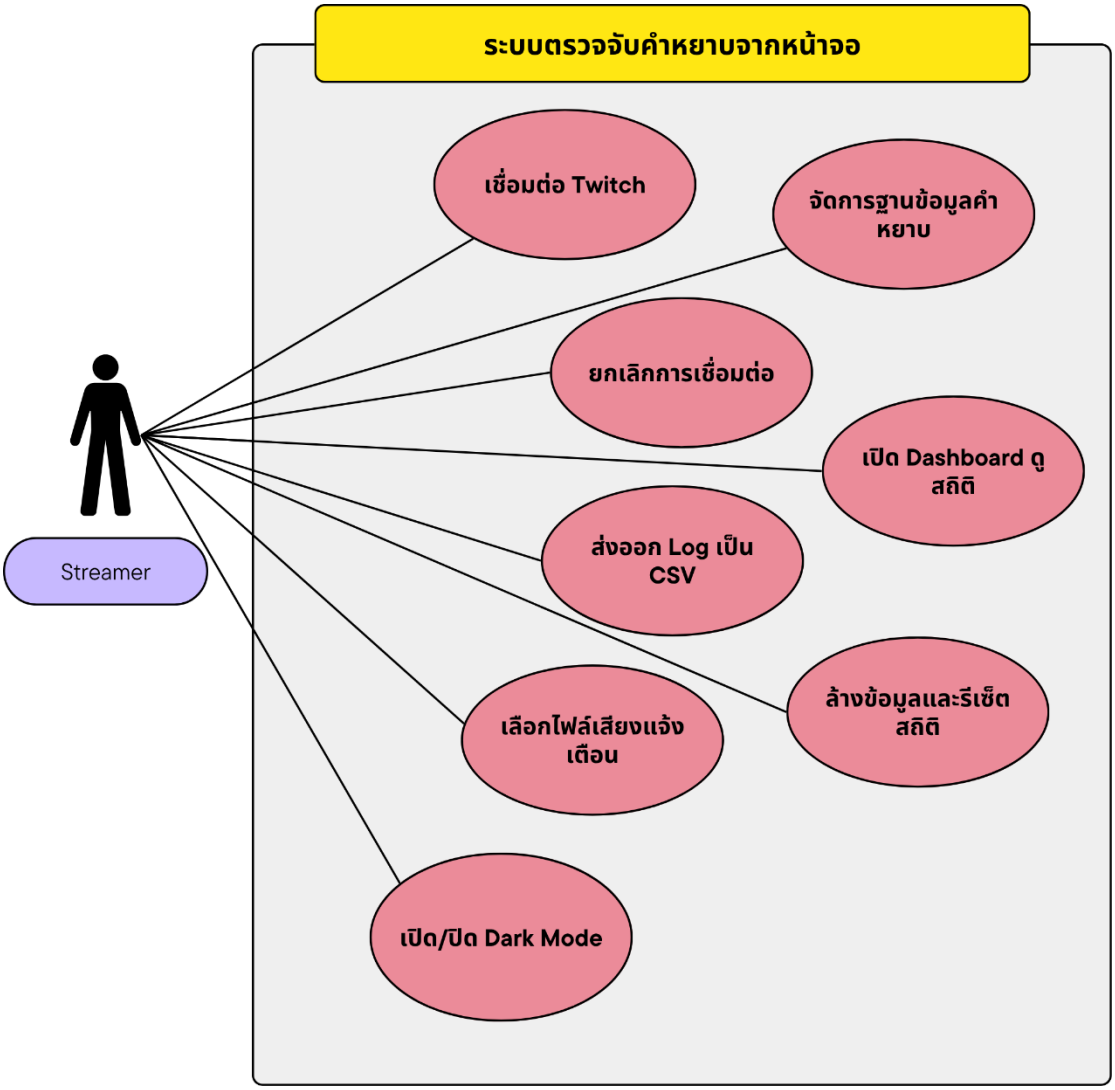
1. โมดูลการเชื่อมต่อ Twitch (Twitch Connection Module):
   - หน้าที่: รับผิดชอบการสื่อสารทั้งหมดกับเซิร์ฟเวอร์ Twitch IRC จัดการการเชื่อมต่อ, การยืนยันตัวตน, การรับส่งข้อมูล, และการรักษาสถานะการเชื่อมต่อ (PING/PONG)
   - ส่วนที่เกี่ยวข้องในโค้ด: คลาส TwitchChatWorker และการใช้ไลบรารี socket

- การทำงานของโมดูลนี้เกิดขึ้นทั้งหมดใน Worker Thread ที่แยกออกมาต่างหาก (คลาส TwitchChatWorker) เพื่อป้องกันไม่ให้ส่วนติดต่อผู้ใช้ (GUI) ค้างขณะรอรับข้อมูล. ภายใน Thread นี้ โปรแกรมจะทำงานในลูป while self.running เพื่อ "ดักฟัง" ข้อมูลที่ถูกส่งมาจากเซิร์ฟเวอร์อย่างต่อเนื่อง. กลไกที่สำคัญคือการจัดการกับคำสั่ง PING ที่เซิร์ฟเวอร์ส่งมาเป็นระยะๆ ซึ่งโมดูลนี้จะต้องตอบกลับด้วย PONG ทันที เพื่อยืนยันว่าการเชื่อมต่อยังคงทำงานอยู่และป้องกันไม่ให้เซิร์ฟเวอร์ตัดการเชื่อมต่อ

2. โมดูลวิเคราะห์ภาษา (NLP Analysis Module):
    - หน้าที่: เป็นสมองของระบบ รับข้อความดิบมาจากโมดูลเชื่อมต่อ แล้วนำมาประมวลผลเพื่อตรวจจับคำหยาบ ประกอบด้วยการทำความสะอาดข้อความ, การใช้ wordsegment สำหรับภาษาอังกฤษ, และการเปรียบเทียบกับฐานข้อมูล badwords.txt และ badwords_en.txt
    - ส่วนที่เกี่ยวข้องในโค้ด: ฟังก์ชัน detect_thai_profanity(), detect_english_profanity() และ optimized_detect_bad_words()
    - สำหรับภาษาไทย (ฟังก์ชัน detect_thai_profanity): ใช้วิธีการค้นหาสายอักขระ (Substring Matching) โดยตรง คือการตรวจสอบว่าคำหยาบจากฐานข้อมูล เป็นส่วนหนึ่งของประโยค หรือไม่ (badword in message). วิธีนี้เหมาะกับภาษาไทยที่มักไม่มีการเว้นวรรคระหว่างคำ
    - สำหรับภาษาอังกฤษ (ฟังก์ชัน detect_english_profanity): ใช้วิธีการที่ซับซ้อนกว่า คือ การแบ่งคำ (Tokenization) ก่อน แล้วจึงนำแต่ละคำไปตรวจสอบ. โดยจะทำการแยกประโยคออกเป็นคำๆ ด้วย wordsegment และ split() จากนั้นจึงนำ คำที่แยกได้ไปตรวจสอบว่ามีอยู่ในฐานข้อมูลคำหยาบ หรือไม่ (word in self.badwords_en). วิธีนี้ป้องกันการเกิดผลบวกลวง (False Positive) ได้อย่างมีประสิทธิภาพ เช่น การตรวจเจอคำว่า ass ในคำว่า cl-ass-ic

3. โมดูลส่วนติดต่อผู้ใช้ (User Interface Module):
    - หน้าที่: เป็นส่วนที่ผู้ใช้โต้ตอบด้วยโดยตรง รับข้อมูลจากผู้ใช้ (เช่น ชื่อแชนแนล), แสดงผลลัพธ์การตรวจจับ, แสดงข้อความแชทล่าสุด, และจัดการหน้าต่างต่างๆ เช่น Dashboard และหน้าจัดการคำหยาบ
    - ส่วนที่เกี่ยวข้องในโค้ด: คลาส BadWordDetectorApp, DashboardWindow, BadWordManagerDialog และการใช้ไลบรารี PyQt5 ทั้งหมด
    - การสื่อสารระหว่าง Worker Thread (โมดูลเชื่อมต่อ) และ GUI Thread (โมดูลส่วนติดต่อผู้ใช้) อาศัยกลไก Signal-Slot ของ PyQt5. เมื่อ TwitchChatWorker ตรวจพบคำหยาบหรือได้รับข้อความใหม่ มันจะทำการ "ส่งสัญญาณ" (emit signal) เช่น bad_word_detected หรือ message_received. ในขณะเดียวกัน คลาสหลัก BadWordDetectorApp จะมีฟังก์ชัน (เรียกว่า "Slot") เช่น on_twitch_bad_word() และ on_twitch_message() ที่ถูกเชื่อมต่อ (connect) ไว้กับสัญญาณเหล่านั้น. เมื่อมีสัญญาณถูกส่งมา สล็อตที่เชื่อมต่อไว้จะถูกเรียกใช้งานโดยอัตโนมัติเพื่ออัปเดตข้อมูลบนหน้าจอ เช่น การเพิ่มข้อความในกล่อง Log หรือการเพิ่มค่าในตัวนับสถิติ. สถาปัตยกรรมนี้ทำให้การทำงานเบื้องหลังและการแสดงผลแยกออกจากกันอย่างชัดเจนและปลอดภัย

3.1.2 Use case Diagram



ภาพที่ 3.2 Use case Diagram

3.2 การวิเคราะห์ขอบเขตและความต้องการของระบบ

การวิเคราะห์ขอบเขตและความต้องการของระบบเป็นขั้นตอนสำคัญ เพื่อกำหนดแนวทางการพัฒนาระบบตรวจจับคำหยาบให้ตอบสนองต่อปัญหาและความต้องการของผู้ใช้งาน (สตรีมเมอร์) ได้อย่างมีประสิทธิภาพ โดยคำนึงถึงเทคโนโลยีที่เลือกใช้เป็นหลัก คือการเชื่อมต่อกับ Twitch IRC API โดยตรง

3.2.1 ความต้องการเชิงฟังก์ชัน (Functional Requirements)

ความต้องการเชิงฟังก์ชันคือสิ่งที่ระบบ ต้องทำได้ เพื่อให้บรรลุวัตถุประสงค์ของโครงงาน:

- **FR-01:** ระบบต้องสามารถเชื่อมต่อกับเซิร์ฟเวอร์ Twitch IRC (irc.chat.twitch.tv) และเข้าร่วมแชนแนลที่ผู้ใช้ระบุได้
- **FR-02:** ระบบต้องสามารถรับข้อความแชท (คำสั่ง PRIVMSG) จากแชนแนลที่เชื่อมต่อได้แบบ Real-time
- **FR-03:** ระบบต้องสามารถตรวจจับคำหยาบภาษาไทย โดยเปรียบเทียบข้อความกับฐานข้อมูล badwords.txt ได้
- **FR-04:** ระบบต้องสามารถตรวจจับคำหยาบภาษาอังกฤษ โดยเปรียบเทียบคำที่แยกแล้วกับฐานข้อมูล badwords_en.txt ได้
- **FR-05:** ระบบต้องมีความสามารถในการตรวจจับคำหยาบภาษาอังกฤษที่ถูกดัดแปลงโดยการพิมพ์ติดกัน (อาศัยไลบรารี wordsegment)
- **FR-06:** ระบบต้องมีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) ที่ใช้งานง่าย สำหรับการกรอกชื่อแชนแนล, ควบคุมการเชื่อมต่อ, และแสดงผลลัพธ์
- **FR-07:** ระบบต้องสามารถแสดงผลการตรวจจับคำหยาบและข้อความแชทล่าสุดบนหน้าจอหลักได้
- **FR-08:** ระบบต้องสามารถแจ้งเตือนผู้ใช้เมื่อตรวจพบคำหยาบ (เช่น เสียงเตือน)
- **FR-09:** ระบบต้องมีหน้าต่าง Dashboad สำหรับแสดงสถิติการตรวจจับแบบ Real-time

- **FR-10:** ระบบต้องสามารถส่งออก (Export) ประวัติการตรวจจับคำหยาบที่เก็บในหน่วยความจำเป็นไฟล์ CSV ได้
- **FR-11:** ผู้ใช้ต้องสามารถจัดการ (เพิ่ม/ลบ/บันทึก) ฐานข้อมูลคำหยาบ (ไฟล์ .txt) ผ่าน GUI ได้
- **FR-12:** ระบบต้องรองรับการเปลี่ยนธีมสี (Dark Mode)

3.2.2 ความต้องการที่ไม่ใช่เชิงฟังก์ชัน (Non-Functional Requirements)

ความต้องการที่ไม่ใช่เชิงฟังก์ชันคือคุณลักษณะหรือข้อกำหนดเกี่ยวกับ "วิธีการทำงาน" ของระบบ:

- **NFR-01 (Performance):** ระบบต้องสามารถประมวลผลและแจ้งเตือนคำหยาบได้แบบ **Real-time** โดยมีความล่าช้า (Latency) น้อยที่สุด
- **NFR-02 (Resource Usage):** ระบบต้องใช้ทรัพยากร **CPU และ Memory ต่ำ** เพื่อไม่ให้กระทบต่อประสิทธิภาพการสตรีมของผู้ใช้
- **NFR-03 (Usability):** ส่วนติดต่อผู้ใช้ต้อง ใช้งานง่าย ไม่ซับซ้อน และไม่รบกวนการทำงานหลักของผู้ใช้ (สตรีมเมอร์)
- **NFR-04 (Reliability):** การทำงานของระบบขึ้นอยู่กับ ความเสถียรของการเชื่อมต่ออินเทอร์เน็ต และสถานะของเซิร์ฟเวอร์ Twitch IRC
- **NFR-05 (Maintainability):** ฐานข้อมูลคำหยาบต้อง สามารถปรับปรุงและอัปเดตได้ง่าย โดยผู้ใช้เอง
- **NFR-06 (Platform):** ระบบมุ่งเน้นการทำงานกับ แพลตฟอร์ม Twitch เป็นหลัก

3.2.3 ข้อจำกัดของระบบ (System Constraints)

จากที่ระบุไว้ในบทที่ 1 ข้อจำกัดเหล่านี้เป็นกรอบสำคัญในการกำหนดความต้องการ:

- ระบบ ไม่รองรับ การตรวจจับคำหยาบในรูปแบบ รูปภาพ, สติกเกอร์, หรือไฟล์มีเดียอื่นๆ
- ความแม่นยำในการตรวจจับ ขึ้นอยู่กับความสมบูรณ์ของฐานข้อมูลคำหยาบ เป็นหลัก และอาจไม่สามารถตรวจจับคำสแลงใหม่ๆ หรือความหมายแฝงได้
- ระบบจัดเก็บประวัติการตรวจจับไว้ในหน่วยความจำ จำกัดที่ 200 ข้อความล่าสุด เท่านั้น หากต้องการเก็บถาวรต้องทำการ Export เป็นระยะ

3.3 การดำเนินงาน

ส่วนนี้จะกล่าวถึงรายละเอียดขั้นตอนการดำเนินงานทั้งหมด ตั้งแต่การออกแบบระบบในส่วนต่างๆ การพัฒนาระบบตามที่ได้ออกแบบไว้ ไปจนถึงการวางแผนทดสอบระบบเพื่อให้มั่นใจว่าโปรแกรมสามารถทำงานได้ตามวัตถุประสงค์

3.3.1 การออกแบบและพัฒนาระบบ (System Design and Development

3.3.1.1 ตาราง Use Case Specification

ชื่อยูสเคส(Use Case Name)	เชื่อมต่อ Twitch
รหัสยูสเคส(Use Case ID)	UC-001
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้สั่งให้โปรแกรมเริ่มการตรวจจับคำหยาบจากแชทแนล Twitch ที่ต้องการ
เงื่อนไขก่อนหน้า(Pre-conditions)	1. ผู้ใช้ต้องกรอกชื่อแชทแนลที่ต้องการ 2. ระบบต้องยังไม่ได้เชื่อมต่อ
เงื่อนไขภายหลัง (Post-conditions)	1. ระบบเริ่มทำงานเบื้องหลังเพื่อดักฟังแชท 2. หน้าจอแสดงสถานะว่า "เชื่อมต่อสำเร็จ" 3. ปุ่ม "เชื่อมต่อ" จะถูกปิด และปุ่ม "ยกเลิกการเชื่อมต่อ" จะใช้งานได้
กระแสหลัก (Basic Flow)	1. ผู้ใช้พิมพ์ชื่อแชทแนลในช่องที่กำหนด 2. ผู้ใช้กดปุ่ม "เชื่อมต่อ Twitch" 3. ระบบเริ่มกระบวนการเชื่อมต่อในเบื้องหลัง (เพื่อไม่ให้หน้าจอค้าง)

	4. เมื่อเชื่อมต่อสำเร็จ ระบบจะอัปเดตสถานะบนหน้าจอ และเริ่มดักฟังข้อความแชท
กระแสรอง (Alternative Flow)	2a. ผู้ใช้ไม่กรอกชื่อแชนแนล: 1. ระบบจะแสดงหน้าต่างแจ้งเตือนว่า "กรุณาใส่ชื่อ channel" 2b. ผู้ใช้กรอกชื่อแชนแนลผิดรูปแบบ (เช่น มีสัญลักษณ์พิเศษ): 1. ระบบจะแสดงหน้าต่างแจ้งเตือนว่า "ชื่อ channel ไม่ถูกต้อง" 4a. การเชื่อมต่อล้มเหลว (เช่น อินเทอร์เน็ตมีปัญหาหรือไม่มีแชนแนลนี้): 1. ระบบจะแสดงสถานะเป็น "การเชื่อมต่อล้มเหลว" (มักจะเป็นสีแดง) 2. ปุ่ม "เชื่อมต่อ" จะกลับมากดได้ตามเดิม

ชื่อยูสเคส(Use Case Name)	ยกเลิกการเชื่อมต่อ
รหัสยูสเคส(Use Case ID)	UC-002
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้สั่งให้โปรแกรมหยุดการตรวจจับคำหยาบ และตัดการเชื่อมต่อจาก Twitch
เงื่อนไขก่อนหน้า(Pre-conditions)	ระบบต้องอยู่ในสถานะ "เชื่อมต่อแล้ว"
เงื่อนไขภายหลัง (Post-conditions)	1. ระบบหยุดการดักฟังแชท และตัดการเชื่อมต่อจาก Twitch 2. ปุ่ม "ยกเลิกการเชื่อมต่อ" จะถูกปิด และปุ่ม "เชื่อมต่อ" จะกลับมาใช้งานได้
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "ยกเลิกการเชื่อมต่อ" 2. ระบบส่งคำสั่งไปหยุดการทำงานเบื้องหลัง และตัดการเชื่อมต่อ 3. หน้าจออัปเดตสถานะเป็น "ยกเลิกการเชื่อมต่อ" และเปลี่ยนสถานะปุ่มกลับเป็นปกติ
กระแสรอง (Alternative Flow)	-

ชื่อยูสเคส(Use Case Name)	จัดการฐานข้อมูลคำหยาบ
รหัสยูสเคส(Use Case ID)	UC-003
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้ เพิ่ม, ลบ, หรือค้นหา คำหยาบในฐานข้อมูล (ไฟล์ .txt) ของโปรแกรม
เงื่อนไขก่อนหน้า(Pre-conditions)	โปรแกรมต้องทำงานอยู่
เงื่อนไขภายหลัง (Post-conditions)	1. ฐานข้อมูลคำหยาบ (ไฟล์ .txt) ถูกอัปเดตด้วยข้อมูลใหม่ 2. (ถ้ากำลังเชื่อมต่อ) ระบบจะเริ่มใช้ฐานข้อมูลคำหยาบที่อัปเดตใหม่ทันที
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "จัดการคำหยาบ (ไทย)" หรือ "(อังกฤษ)" 2. ระบบเปิดหน้าต่างใหม่สำหรับจัดการคำหยาบ 3. ผู้ใช้สามารถพิมพ์คำใหม่ในช่องและกด "เพิ่ม" เพื่อเพิ่มคำ 4. ผู้ใช้สามารถเลือกคำที่มีอยู่และกด "ลบ" เพื่อลบคำ 5. ผู้ใช้สามารถพิมพ์ในช่อง "ค้นหา" เพื่อกรองรายการคำ 6. เมื่อผู้ใช้กด "บันทึก" ระบบจะนำรายการคำใหม่ทั้งหมดไปเก็บไว้ในไฟล์ฐานข้อมูล
กระแสรอง (Alternative Flow)	**6a. บันทึกไฟล์ไม่สำเร็จ (เช่น ไฟล์อาจถูกเปิดค้างไว้):** 1. ระบบจะแจ้งเตือนข้อผิดพลาดว่า "ไม่สามารถบันทึกได้"

	6b. ผู้ใช้ปิดหน้าต่างโดยไม่บันทึก: 1. การเปลี่ยนแปลงทั้งหมดที่ทำในหน้าต่างนั้นจะถูกยกเลิก

ชื่อยูสเคส(Use Case Name)	เปิด Dashboard ดูสถิติ
รหัสยูสเคส(Use Case ID)	UC-004
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้เปิดหน้าต่างใหม่เพื่อดูสถิติการตรวจจับแบบ Real-time
เงื่อนไขก่อนหน้า(Pre-conditions)	โปรแกรมต้องทำงานอยู่
เงื่อนไขภายหลัง (Post-conditions)	หน้าต่างสถิติแสดงขึ้นมา และข้อมูลจะอัปเดตตัวเองอัตโนมัติ
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "เปิด Dashboard" 2. ระบบเปิดหน้าต่างสถิติใหม่ 3. หน้าต่างนี้จะอัปเดตข้อมูล (เช่น จำนวนคำหยาบที่พบ, จำนวนข้อความทั้งหมด, เวลาการทำงาน) โดยอัตโนมัติทุกๆ 1 วินาที
กระแสรอง (Alternative Flow)	**3a. ผู้ใช้กด "ล้างสถิติ" ในหน้า Dashboard:** 1. ระบบจะรีเซ็ตค่าสถิติทั้งหมดในโปรแกรมให้เป็น 0 และแจ้งเตือนผู้ใช้

ชื่อยูสเคส(Use Case Name)	ส่งออก Log เป็น CSV
รหัสยูสเคส(Use Case ID)	UC-005
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้บันทึกประวัติการตรวจจับคำหยาบ (ที่อยู่ในหน่วยความจำชั่วคราว) ลงเป็นไฟล์ที่เปิดด้วย Excel ได้
เงื่อนไขก่อนหน้า(Pre-conditions)	1. ผู้ใช้ต้องเคยเชื่อมต่อ Twitch อย่างน้อยหนึ่งครั้ง 2. ต้องมีข้อมูลประวัติการตรวจจับหลงเหลืออยู่
เงื่อนไขภายหลัง (Post-conditions)	ไฟล์ CSV ที่มีข้อมูลประวัติ (เวลา, ผู้ใช้, ข้อความ, คำหยาบที่พบ) ถูกบันทึกในเครื่องของผู้ใช้
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "Export Log (CSV)" 2. ระบบแสดงหน้าต่างให้ผู้ใช้เลือกว่าจะบันทึกไฟล์ไว้ที่ไหนและตั้งชื่อไฟล์ 3. ผู้ใช้เลือกที่บันทึกและกด "ตกลง" 4. ระบบรวบรวมประวัติการตรวจจับทั้งหมด และสร้างเป็นไฟล์ CSV บันทึกไปยังตำแหน่งที่ผู้ใช้เลือก 5. ระบบแจ้งเตือนผู้ใช้ว่า "บันทึกสำเร็จ"
กระแสรอง (Alternative Flow)	**1a. ยังไม่มีข้อมูลให้บันทึก (เช่น เพิ่งเปิดโปรแกรม):** **1. ระบบจะแจ้งเตือนว่า "ยังไม่มีข้อมูลประวัติการตรวจจับ"** **3a. ผู้ใช้กด "ยกเลิก" ในหน้าต่างเลือกที่บันทึก:** **1. ระบบจะยกเลิกการบันทึกไฟล์**

ชื่อยูสเคส(Use Case Name)	ล้างข้อมูลและรีเซ็ตสถิติ
รหัสยูสเคส(Use Case ID)	UC-006
ผู้ใช้(Actor)	สตรีมเมอร์
คำอธิบาย(Description)	ผู้ใช้ล้างข้อมูลประวัติที่แสดงบนหน้าจอและรีเซ็ตค่าสถิติทั้งหมดเป็น 0
เงื่อนไขก่อนหน้า(Pre-conditions)	โปรแกรมต้องทำงานอยู่
เงื่อนไขภายหลัง (Post-conditions)	1. ประวัติการแชทและประวัติคำหยาบบนหน้าจอถูกล้าง 2. ข้อมูลในหน่วยความจำชั่วคราว (สำหรับ Export) ถูกล้าง 3. ค่าสถิติต่างๆ (เช่น จำนวนคำหยาบที่พบ) ถูกตั้งค่าเป็น 0
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "ล้างข้อมูลทั้งหมด" (ในแท็บข้อความแชท) 2. ระบบจะล้างข้อความประวัติทั้งหมดบนหน้าจอ 3. ระบบจะสั่งล้างข้อมูลในหน่วยความจำชั่วคราว (ที่ใช้สำหรับ Export Log) 4. ระบบจะรีเซ็ตค่าสถิติการนับทั้งหมดให้เป็น 0
กระแสรอง (Alternative Flow)	-

ชื่อยูสเคส(Use Case Name)	เลือกไฟล์เสียงแจ้งเตือน
รหัสยูสเคส(Use Case ID)	UC-007
ผู้ใช้(Actor)	ผู้ใช้งาน (สตรีมเมอร์)
คำอธิบาย(Description)	ผู้ใช้เปลี่ยนเสียงแจ้งเตือนเริ่มต้น (เสียง Beep) เป็นไฟล์เสียง .wav ที่ต้องการ
เงื่อนไขก่อนหน้า(Pre-conditions)	โปรแกรมต้องทำงานอยู่
เงื่อนไขภายหลัง (Post-conditions)	1. ระบบจดจำไฟล์เสียงที่ผู้ใช้เลือก 2. ข้อความบนปุ่มเปลี่ยนเป็นชื่อไฟล์ที่เลือก 3. เมื่อพบคำหยาบครั้งถัดไป ระบบจะเล่นเสียงจากไฟล์นี้
กระแสหลัก (Basic Flow)	1. ผู้ใช้กดปุ่ม "เลือกไฟล์เสียง (.wav)" 2. ระบบแสดงหน้าต่างให้ผู้ใช้เลือกไฟล์เสียงจากในเครื่องคอมพิวเตอร์ 3. เมื่อผู้ใช้เลือกไฟล์ .wav และกดยืนยัน, ระบบจะจดจำไฟล์นั้น และเปลี่ยนข้อความบนปุ่มเป็นชื่อไฟล์ 4. (ครั้งต่อไปที่พบคำหยาบ ระบบจะเล่นเสียงจากไฟล์นี้แทน)
กระแสรอง (Alternative Flow)	**3a. ผู้ใช้กด "ยกเลิก" หรือไม่เลือกไฟล์:** 1. ระบบจะยังคงใช้เสียง Beep พื้นฐานตามเดิม

3.3.1.2 การออกแบบส่วนติดต่อผู้ใช้ (User Interface Design)

ภาพที่ 3.3 : การออกแบบส่วนติดต่อผู้ใช้ (User Interface Design) ด้วย Figma

ก่อนเริ่มการพัฒนาส่วนติดต่อผู้ใช้ด้วยโค้ด PyQt5, ได้มีการออกแบบหน้าตาโปรแกรม (Mockup) และวางแผนประสบการณ์การใช้งาน (User Experience) เบื้องต้นโดยใช้เครื่องมือ Figma ดังแสดงในภาพ เพื่อให้มั่นใจว่าองค์ประกอบต่างๆ เช่น ปุ่มควบคุม, การแสดงผล, และหน้าต่าง Dashboard ถูกจัดวางอย่างเหมาะสม ใช้งานง่าย และไม่รบกวนการทำงานหลักของผู้ใช้งาน (สตรีมเมอร์) การออกแบบนี้จึงเป็นเสมือนพิมพ์เขียว (Blueprint) สำหรับการพัฒนา GUI ในขั้นตอนต่อไป

3.3.1.3 การออกแบบการเชื่อมต่อ Twitch IRC (Twitch IRC Connection Design)

โมดูลนี้รับผิดชอบการสื่อสารทั้งหมดกับเซิร์ฟเวอร์แชทของ Twitch โดยทำงานอยู่ภายใน Worker Thread (คลาส TwitchChatWorker) เพื่อไม่ให้กระทบต่อการทำงานของส่วนติดต่อผู้ใช้. การทำงานหลักประกอบด้วย:

(1) การสร้างการเชื่อมต่อ (Establishing Connection): ใช้ไลบรารี socket ซึ่งเป็นไลบรารีมาตรฐานของ Python ในการสร้างการเชื่อมต่อแบบ TCP/IP ไปยังเซิร์ฟเวอร์ irc.chat.twitch.tv ที่พอร์ต 6667. มีการตั้งค่า Timeout (socket.settimeout()) เพื่อป้องกันการค้างหากเซิร์ฟเวอร์ไม่ตอบสนอง.

(2) การยืนยันตัวตนและการเข้าร่วมแชนแนล (Authentication and Joining Channel): หลังจากเชื่อมต่อสำเร็จ โปรแกรมจะส่งคำสั่งมาตรฐานของ IRC คือ NICK เพื่อตั้งชื่อผู้ใช้ (ใช้ชื่อแบบไม่ระบุตัวตน justinfan ตามด้วยตัวเลขเวลาเพื่อเลี่ยงชื่อซ้ำ) และคำสั่ง JOIN เพื่อระบุว่าจะเข้าร่วมแชทของแชนแนลใด (ตามที่ผู้ใช้กรอก).

(3) การรับและประมวลผลข้อความ (Receiving and Processing Messages): โปรแกรมเข้าสู่ลูป (while self.running) เพื่อรอรับข้อมูลจากเซิร์ฟเวอร์อย่างต่อเนื่องโดยใช้ socket.recv(). ข้อมูลที่ได้รับจะถูกถอดรหัส (decode) จาก UTF-8. หากเป็นข้อความแชท (ระบุด้วยคำสั่ง PRIVMSG) จะถูกส่งต่อไปยังฟังก์ชัน process_chat_message() เพื่อแยกชื่อผู้ใช้และเนื้อหาข้อความ.

(4) การรักษาการเชื่อมต่อ (Maintaining Connection - PING/PONG): โปรแกรมถูกออกแบบมาให้ตรวจจับคำสั่ง PING ที่เซิร์ฟเวอร์ส่งมาเป็นระยะๆ และจะตอบกลับด้วยคำสั่ง PONG โดยอัตโนมัติ เพื่อยืนยันว่าการเชื่อมต่อยังคงทำงานอยู่และป้องกันการถูกตัดการเชื่อมต่อ.

3.3.1.4 การออกแบบโมดูลวิเคราะห์ภาษา (NLP Analysis Module Design)

โมดูลนี้ทำหน้าที่วิเคราะห์ข้อความที่ได้รับจาก Twitch IRC เพื่อตรวจจับคำหยาบ โดยมีการทำงานหลักอยู่ในฟังก์ชัน optimized_detect_bad_words() ซึ่งจะเรียกใช้ฟังก์ชันย่อยสำหรับแต่ละภาษาตามลำดับ.

(1) การทำความสะอาดข้อความ (Text Cleaning): ก่อนการตรวจจับข้อความจะถูกเตรียมให้พร้อม โดยทั่วไปจะมีการแปลงเป็นตัวพิมพ์เล็ก (.lower()) และใช้ Regular Expressions (re.sub()) เพื่อลบอักขระพิเศษที่ไม่ต้องการออก. สำหรับภาษาไทย จะมีการลบช่องว่างออกทั้งหมด (.replace(' ', '')) เพื่อให้การตรวจจับแบบ Substring Matching ทำงานได้ดีขึ้น.

(2) การตรวจจับคำหยาบภาษาไทย (Thai Profanity Detection): ใช้ฟังก์ชัน detect_thai_profanity() ซึ่งทำงานโดยใช้วิธี Substring Matching. โปรแกรมจะวนลูปอ่านคำหยาบแต่ละคำจาก self.badwords_th (ที่โหลดมาจาก badwords.txt) แล้วตรวจสอบว่าคำหยาบนั้น เป็นส่วนหนึ่งของ ข้อความที่ทำความสะอาดแล้วหรือไม่ (badword in message_clean_no_space).

(3) การตรวจจับคำหยาบภาษาอังกฤษ (English Profanity Detection): ใช้ฟังก์ชัน detect_english_profanity() ซึ่งทำงานซับซ้อนกว่าเพื่อป้องกัน False Positive:

- การแบ่งคำ (Tokenization): โปรแกรมจะตรวจสอบว่าไลบรารี wordsegment พร้อมใช้งานหรือไม่ (WORDSEGMENT_AVAILABLE).
    - หากมี wordsegment: จะทำการลบช่องว่างทั้งหมดในข้อความ แล้วใช้ segment() เพื่อแยกคำที่พิมพ์ติดกัน

(เช่น Areyoustupid -> ['Are', 'you',’stupid’]) จากนั้นรวมผลลัพธ์นี้กับคำที่ได้จากการแยกด้วยช่องว่างปกติ (split()) เพื่อให้ได้รายการคำที่ครอบคลุมที่สุด (words_to_check).

- หากไม่มี wordsegment: จะใช้แผนสำรองคือการแยกคำด้วยช่องว่างปกติเท่านั้น (cleaned_message.split()).

- การจับคู่ (Matching): โปรแกรมจะวนลูปอ่าน คำที่แยกได้จากข้อความ ทีละคำ (word ใน words_to_check) แล้วตรวจสอบว่าคำนั้น มีอยู่ใน ฐานข้อมูลคำหยาบภาษาอังกฤษ (self.badwords_en ที่โหลดมาจาก badwords_en.txt) หรือไม่ (word in self.badwords_en).

- (4) การรวบรวมผลลัพธ์: ฟังก์ชัน optimized_detect_bad_words() จะรวบรวมคำหยาบที่พบจากทั้งสองภาษา แล้วใช้ set() เพื่อลบคำที่ซ้ำซ้อนกันออก ก่อนส่งคืนผลลัพธ์สุดท้าย.

3.3.1.5 การออกแบบการทำงานแบบ Multi-threading (Multi-threading Design)

เพื่อให้ส่วนติดต่อผู้ใช้ (GUI) สามารถตอบสนองต่อผู้ใช้ได้ตลอดเวลา แม้ในขณะที่กำลังรอรับข้อมูลจากเครือข่าย ระบบจึงถูกออกแบบให้ใช้ Multi-threading โดยอาศัยความสามารถของไลบรารี PyQt5:

- (1) การแบ่งเธรด (Thread Separation): การทำงานถูกแบ่งออกเป็น 2 เธรดหลัก:

  - GUI Thread (Main Thread): เป็นเธรดที่ PyQt5 สร้างขึ้นโดยอัตโนมัติ ทำหน้าที่จัดการหน้าต่างโปรแกรม, ปุ่ม, การแสดงผล และรับ Event จากผู้ใช้ทั้งหมด.

  - Worker Thread (Background Thread): สร้างขึ้นโดยใช้คลาส QThread (ในที่นี้คือ TwitchChatThread) ทำหน้าที่รันการทำงานที่อาจใช้เวลานานหรือไม่แน่นอน เช่น การเชื่อมต่อ Socket, การรอรับข้อมูล (socket.recv()) และการประมวลผลข้อความ ซึ่ง

ทั้งหมดนี้ถูกจัดการภายในออบเจกต์ TwitchChatWorker ที่ถูกย้ายไปทำงานบนเธรดนี้ (worker.moveToThread(self)).

- (2) การสื่อสารระหว่างเธรด (Inter-Thread Communication - Signal-Slot): การส่งข้อมูลระหว่าง Worker Thread และ GUI Thread จะทำผ่านกลไก Signal-Slot ของ PyQt5 ซึ่งมีความปลอดภัย (Thread-safe):
    - Signals: ในคลาส TwitchChatWorker มีการประกาศ Signals ต่างๆ (เช่น message_received, bad_word_detected, connection_status) โดยใช้ pyqtSignal(). เมื่อมีข้อมูลใหม่หรือสถานะเปลี่ยนแปลง Worker Thread จะทำการ "ส่ง" (emit) สัญญาณเหล่านี้พร้อมข้อมูลที่เกี่ยวข้อง.
    - Slots: ในคลาส BadWordDetectorApp (ซึ่งทำงานบน GUI Thread) จะมีฟังก์ชัน (Slots) ที่เตรียมไว้รอรับสัญญาณเหล่านี้ (เช่น on_twitch_message(), on_twitch_bad_word(), on_twitch_connection_status()). การเชื่อมต่อระหว่าง Signal และ Slot จะถูกสร้างขึ้นในฟังก์ชัน connect_twitch() โดยใช้ .connect(). เมื่อ Worker Thread ส่ง Signal ออกมา PyQt5 จะจัดการเรียก Slot ที่เชื่อมต่อไว้ให้ทำงานบน GUI Thread โดยอัตโนมัติ เพื่ออัปเดตหน้าจออย่างปลอดภัย.

3.3.1.6 การออกแบบการจัดการข้อมูล (Data Handling Design)

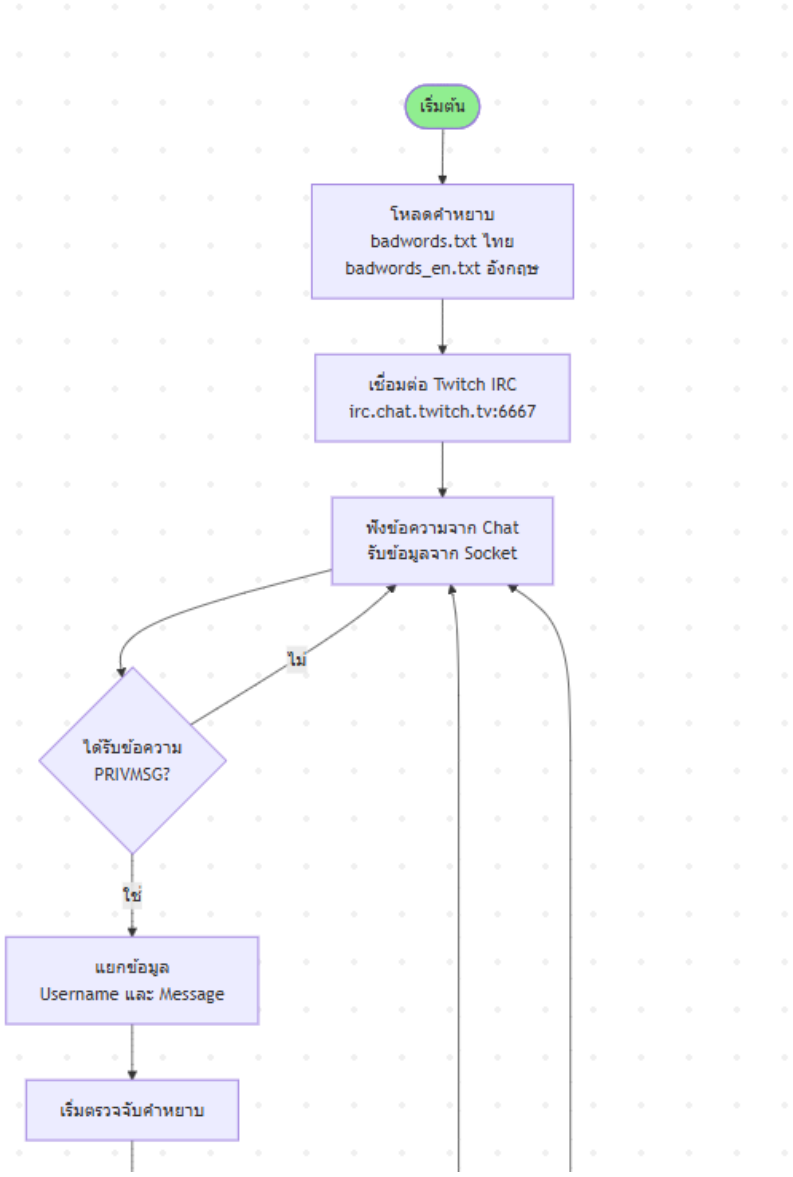
ระบบมีการจัดการข้อมูลหลักๆ 2 ส่วน คือ การเก็บประวัติการตรวจจับชั่วคราว และการส่งออกข้อมูล:

- (1) การเก็บประวัติชั่วคราว (Temporary Log Storage): เพื่อแสดงผล Log คำหยาบและข้อความแชทล่าสุดบน GUI และเพื่อเก็บข้อมูลไว้สำหรับการ Export ระบบใช้โครงสร้างข้อมูลแบบ deque (มาจากไลบรารี collections) ซึ่งเป็นลิสต์ที่มีประสิทธิภาพสูงในการเพิ่มและลบข้อมูลจากปลายทั้งสองด้าน.
    - จำกัดขนาด: deque ถูกกำหนดขนาดสูงสุดไว้ที่ 200 รายการ (maxlen=200). เมื่อมีข้อมูลใหม่เข้ามาเกินขนาด ข้อมูลที่เก่าที่สุดจะถูกลบออกไปโดยอัตโนมัติ ทำให้ระบบใช้หน่วยความจำคงที่ ไม่เพิ่มขึ้นเรื่อยๆ.

- Thread Safety: การเข้าถึง deque (ทั้งการเพิ่มข้อมูลใน process_chat_message และการอ่าน/ล้างข้อมูลใน get_chat_messages, clear_memory_messages) จะมีการใช้ QMutex (self.chat_mutex.lock() และ self.chat_mutex.unlock()) เพื่อป้องกันปัญหาข้อมูลเสียหายที่อาจเกิดขึ้นจากการเข้าถึงพร้อมกันจากหลายเธรด (แม้ในกรณีนี้จะเข้าถึงจาก Worker เป็นหลัก แต่การมี Mutex ถือเป็นการออกแบบที่ปลอดภัย).

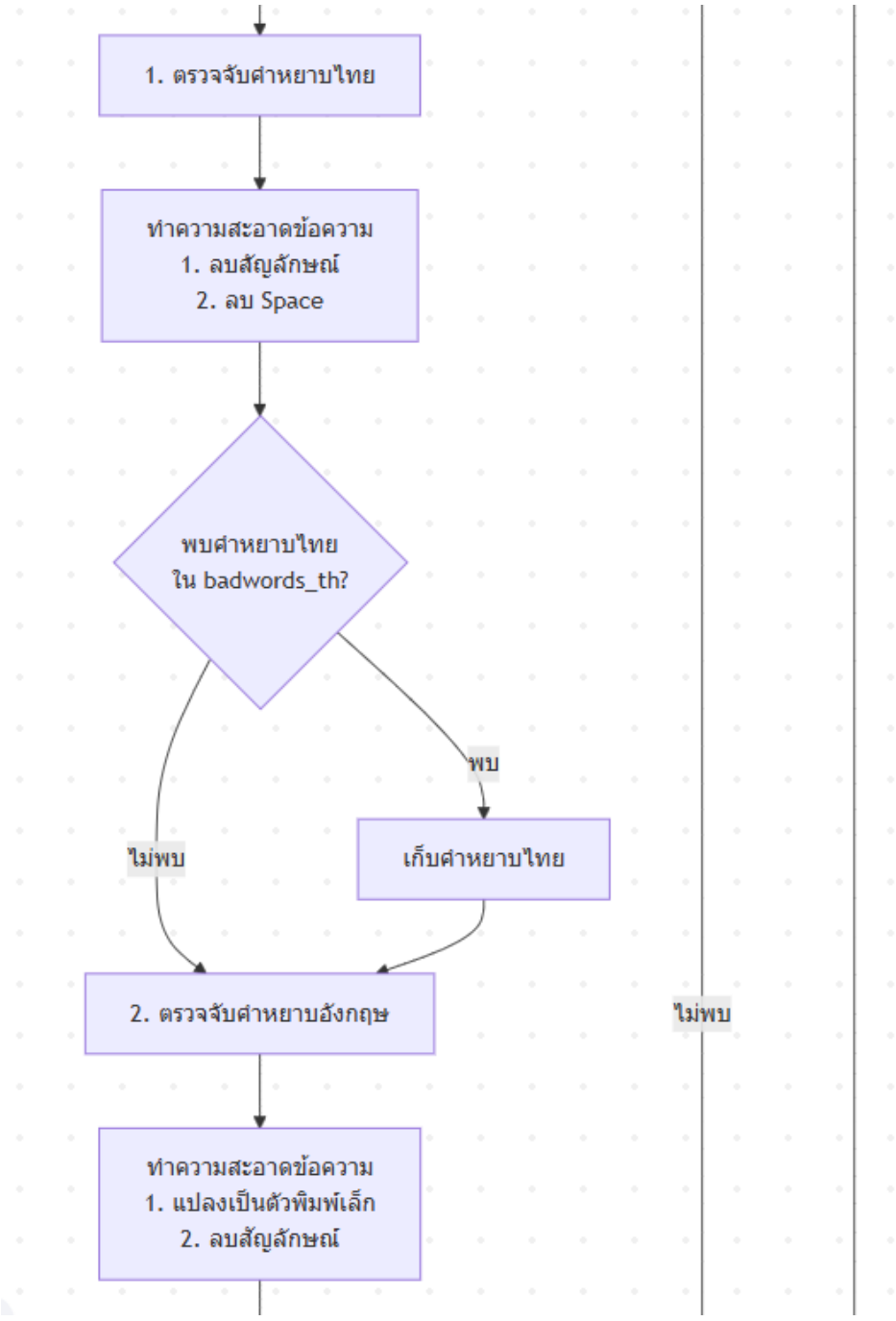
- (2) การส่งออกข้อมูลเป็น CSV (CSV Exporting): ระบบมีฟังก์ชัน export_log() สำหรับให้ผู้ใช้บันทึกประวัติการตรวจจับที่เก็บอยู่ใน deque ลงไฟล์.
    - ใช้ Pandas: กระบวนการนี้ใช้ไลบรารี Pandas ซึ่งเป็นเครื่องมือมาตรฐานสำหรับการจัดการข้อมูล. โปรแกรมจะดึงข้อมูลทั้งหมดจาก deque (worker.get_chat_messages()) มาสร้างเป็น DataFrame ซึ่งเป็นโครงสร้างข้อมูลแบบตารางของ Pandas จากนั้นใช้เมธอด .to_csv() เพื่อบันทึก DataFrame นั้นลงเป็นไฟล์ CSV ได้อย่างง่ายดายและถูกต้องตามรูปแบบมาตรฐาน. มีการระบุ encoding เป็น utf-8-sig เพื่อให้สามารถเปิดอ่าน
    - ภาษาไทยใน Excel ได้โดยไม่เกิดปัญหาตัวอักษรผิดเพี้ยน

3.3.2 Process Flow Chart Diagram



ภาพที่ 3.4: Flow Chart ส่วนที่ 1 - การเริ่มต้นและการรับข้อความ

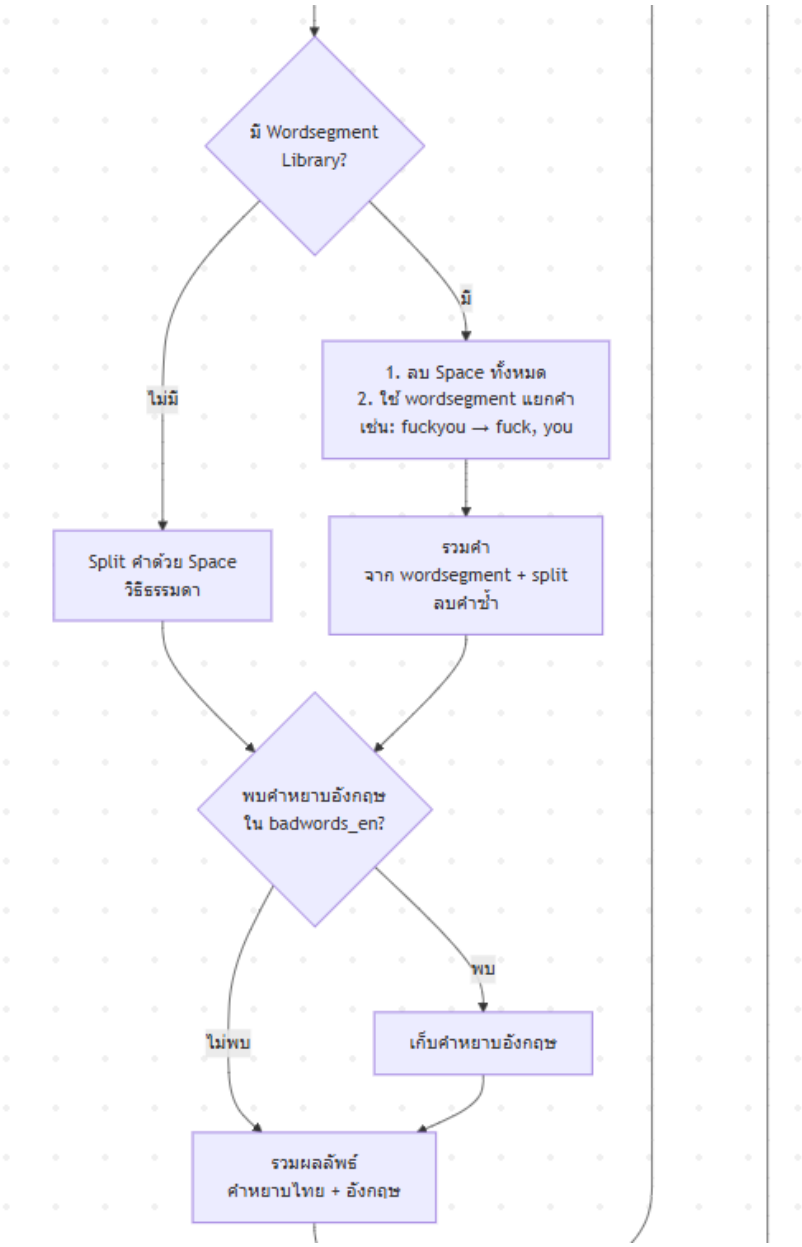
ภาพนี้แสดงจุดเริ่มต้นการทำงานของระบบ ตั้งแต่การโหลดฐานข้อมูลคำหยาบภาษาไทย และภาษาอังกฤษ, การเชื่อมต่อไปยังเซิร์ฟเวอร์ Twitch IRC, และเข้าสู่กระบวนการหลักคือการรอรับข้อมูลจาก Socket. เมื่อได้รับข้อมูล ระบบจะตรวจสอบว่าเป็นข้อความแชท (PRIVMSG) หรือไม่ หากใช่ จะทำการแยกชื่อผู้ใช้และเนื้อหาข้อความออกมา เพื่อส่งต่อไปยังกระบวนการตรวจจับคำหยาบในส่วนถัดไป.



ภาพที่ 3.5: Flow Chart ส่วนที่ 2 - การตรวจจับคำหยาบภาษาไทย

ส่วนนี้แสดงขั้นตอนการตรวจจับคำหยาบภาษาไทย ซึ่งจะทำงานเป็นลำดับแรก. ข้อความที่ได้รับจะถูกทำความสะอาดก่อน โดยลบสัญลักษณ์พิเศษ, และลบช่องว่างทั้งหมด. จากนั้น ระบบจะตรวจสอบว่ามีคำใดในฐานข้อมูล badwords_th เป็นส่วนหนึ่งของข้อความที่ทำ

ความสะอาดแล้วหรือไม่ (Substring Matching). หากพบ จะทำการบันทึกคำหยาบที่พบไว้. ไม่ว่าจะพบหรือไม่พบคำหยาบภาษาไทย กระบวนการจะดำเนินต่อไปยังขั้นตอนการตรวจจับคำหยาบภาษาอังกฤษ.



ภาพที่ 3.6: Flow Chart ส่วนที่ 3 - การตรวจจับคำหยาบภาษาอังกฤษ

ถัดมาเป็นการตรวจจับคำหยาบภาษาอังกฤษ. ข้อความจะถูกทำความสะอาด (แปลงตัวพิมพ์เล็ก, ลบสัญลักษณ์). จากนั้น ระบบจะตรวจสอบว่าไลบรารี wordsegment พร้อมใช้งานหรือไม่.

- **หากมี:** ระบบจะลบช่องว่างทั้งหมดในข้อความ แล้วใช้ wordsegment เพื่อแยกคำที่พิมพ์ติดกัน จากนั้นนำผลลัพธ์มารวมกับคำที่ได้จากการ split ด้วยช่องว่างปกติ และลบคำซ้ำ.
- **หากไม่มี:** ระบบจะใช้แผนสำรองคือการแยกคำด้วยช่องว่างปกติ (split()) เท่านั้น. สุดท้าย ระบบจะนำคำที่แยกได้แต่ละคำไปตรวจสอบว่ามีอยู่ในฐานข้อมูล badwords_en หรือไม่. หากพบ จะบันทึกคำหยาบที่พบไว้ แล้วจึงนำผลลัพธ์ทั้งหมด (ทั้งไทยและอังกฤษ) ไปรวมกัน.

ภาพที่ 3.7: Flow Chart ส่วนที่ 4 - การจัดการเมื่อพบคำหยาบและการแจ้งเตือน

ส่วนสุดท้ายคือการจัดการเมื่อกระบวนการตรวจจับเสร็จสิ้น. ระบบจะตรวจสอบว่ามีคำหยาบถูกพบหรือไม่ (จากผลรวมในขั้นตอนก่อนหน้า).

- **หากไม่พบ:** ระบบจะกลับไปรอรับข้อความใหม่ (Listen).
- **หากพบ:** ระบบจะดำเนินการตามลำดับดังนี้:

    1. เพิ่มค่าตัวนับคำหยาบ (bad_word_count).

2. ใช้ Mutex เพื่อ Lock การเข้าถึงหน่วยความจำชั่วคราว (deque) เพื่อความปลอดภัยของข้อมูล (Thread Safety).
3. บันทึกข้อมูลการตรวจจับ (เวลา, ผู้ใช้, ข้อความ, คำหยาบ) ลงใน deque.
4. Unlock Mutex.
5. ส่งสัญญาณ (emit signal) เพื่ออัปเดตส่วนติดต่อผู้ใช้ (แสดง Log คำหยาบ, แสดงข้อความแชท, อัปเดตสถิติ).
6. เล่นเสียงแจ้งเตือน (Beep หรือไฟล์ .wav ที่เลือกไว้).
7. แสดง Notification บน System Tray ของ Windows. หลังจากแจ้งเตือนเสร็จสิ้น ระบบจะกลับไปรอรับข้อความใหม่ (Listen).

3.3.3 เครื่องมือและเทคโนโลยีที่ใช้ (Tools and Technologies Used)

ในการพัฒนาระบบตรวจจับคำหยาบสำหรับ Twitch นี้ ได้มีการเลือกใช้เครื่องมือและเทคโนโลยีต่างๆ ที่เหมาะสมเพื่อให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพ ดังนี้:

- Python: ใช้เป็น ภาษาโปรแกรมหลัก ในการพัฒนาโครงงานทั้งหมด เนื่องจากมีความยืดหยุ่นสูง มีไลบรารีสนับสนุนจำนวนมาก และเหมาะสมกับการพัฒนาแอปพลิเคชันหลากหลายรูปแบบ
- PyQt5: เป็น เฟรมเวิร์กหลักสำหรับสร้างส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) ทั้งหมดของโปรแกรม. นอกจากนี้ยังใช้ความสามารถของ PyQt5 ในการจัดการ Multi-threading (QThread) เพื่อแยกการทำงานเบื้องหลังออกจากการแสดงผล และใช้กลไก Signal-Slot ในการสื่อสารระหว่างเธรดอย่างปลอดภัย.
- Socket: (ไลบรารีมาตรฐานของ Python) ใช้ในการสร้าง การเชื่อมต่อเครือข่ายระดับต่ำ (TCP/IP) เพื่อสื่อสารกับเซิร์ฟเวอร์ Twitch IRC (irc.chat.twitch.tv) โดยตรง ทำให้สามารถรับ-ส่งข้อความแชทได้แบบ Real-time.
- wordsegment: ไลบรารีสำหรับการประมวลผลภาษาธรรมชาติ (NLP) ที่มีความสำคัญ ใช้สำหรับ แบ่งคำภาษาอังกฤษที่ถูกพิมพ์ติดกัน (เช่น thisisbadword -> this is bad word) เพื่อเพิ่มความสามารถในการตรวจจับคำหยาบที่ถูกดัดแปลงรูปแบบ.
- Pandas: ไลบรารีสำหรับการจัดการข้อมูล ใช้ในฟังก์ชัน ส่งออกประวัติการตรวจจับ (Log) ที่สะสมอยู่ในหน่วยความจำชั่วคราว (deque) ไปเป็นไฟล์รูปแบบ CSV ซึ่งง่ายต่อการนำไปวิเคราะห์ต่อ.
- Regular Expressions (re): (ไลบรารีมาตรฐานของ Python) ใช้สำหรับ ทำความสะอาดข้อความ (Text Cleaning) ในขั้นตอนการประมวลผลภาษาธรรมชาติ เช่น การลบสัญลักษณ์พิเศษต่างๆ หรือตัวเลขที่ไม่ต้องการออกจากข้อความแชทก่อนนำไปวิเคราะห์.
- Collections (deque): (ไลบรารีมาตรฐานของ Python) ใช้โครงสร้างข้อมูล deque ในการ เก็บประวัติข้อความแชทและ Log การตรวจจับล่าสุด ไว้ในหน่วยความจำ โดยจำกัดจำนวนไว้ที่ 200 รายการ (maxlen=200) เพื่อควบคุมการใช้หน่วยความจำ.

3.4 วิธีการทดสอบระบบ (System Testing Methodology)

เพื่อให้มั่นใจว่าระบบตรวจจับคำหยาบที่พัฒนาขึ้น สามารถทำงานได้อย่างถูกต้อง มีประสิทธิภาพ และมีความแม่นยำตามที่ออกแบบไว้ ได้มีการวางแผนและดำเนินการทดสอบในหลายมิติ โดยใช้เครื่องมือทดสอบอัตโนมัติ (pytest) เพื่อให้ได้ผลลัพธ์ที่น่าเชื่อถือและวัดผลได้จริง โดยแบ่งการทดสอบออกเป็นด้านต่างๆ ดังนี้

3.4.1 การทดสอบความถูกต้องของฟังก์ชัน (Functional Testing)

การทดสอบนี้มุ่งเน้นตรวจสอบว่าส่วนประกอบหลักๆ ของโปรแกรมทำงานได้ถูกต้องตามหน้าที่หรือไม่ โดยดำเนินการผ่าน Unit Testing และ Integration Testing ดังนี้:

- ทดสอบการตรวจจับคำ: สร้างข้อความตัวอย่างหลายๆ แบบ ทั้งภาษาไทยและภาษาอังกฤษ เพื่อป้อนเข้าสู่ระบบตรวจจับ:
    - กรณีปกติ: ทดสอบว่าระบบเจอคำหยาบง่ายๆ ที่มีในฐานข้อมูลหรือไม่ (เช่น Input: "ไอ้สัส", "you stupid")
    - กรณีดัดแปลง: ทดสอบว่าระบบยังเจอคำหยาบหรือไม่ แม้จะมีการแทรกช่องว่าง (เช่น Input: "คุณ ไ อ้ ส ั สว์") หรือสัญลักษณ์พิเศษ (เช่น Input: "คุณมัน!@#ไอ้$%สัสว์")
    - กรณีคำติดกัน (อังกฤษ): ทดสอบว่าระบบสามารถแยกและเจอคำหยาบที่พิมพ์ติดกันได้หรือไม่ (เช่น Input: "stupidperson") (เพื่อทดสอบ wordsegment)
    - กรณีไม่มีคำหยาบ: ทดสอบว่าระบบไม่แจ้งเตือนผิดพลาดเมื่อเจอข้อความปกติ (เช่น Input: "สวัสดีครับ", "hello everyone")
    - กรณีผสมภาษา: ทดสอบว่าระบบเจอคำหยาบได้ทั้งไทยและอังกฤษในประโยคเดียวกัน (เช่น Input: "hello ไอ้สัส stupid")
    - กรณีข้อความว่าง: ทดสอบว่าระบบจัดการกับ Input ที่เป็นข้อความว่างได้โดยไม่เกิดข้อผิดพลาด (Input: "")
- ทดสอบการทำงานพื้นฐาน: ตรวจสอบว่าระบบสามารถโหลดฐานข้อมูลคำหยาบจากไฟล์ได้ถูกต้อง (test_load_badwords_files), สามารถเก็บข้อมูล Log ไว้ในหน่วยความจำชั่วคราว (deque) ได้ (test_chat_message_storage), และสามารถล้างข้อมูลนั้นทิ้งได้ (test_clear_messages)

3.4.2 การทดสอบประสิทธิภาพ (Performance Testing)

การทดสอบนี้วัดความเร็วและการใช้ทรัพยากร เพื่อให้แน่ใจว่าโปรแกรมไม่ทำให้เครื่องช้าลงขณะใช้งานจริง

- วัดความเร็ว (Detection Speed): สร้างข้อความจำนวนมาก (500 ข้อความ) (โดยนำข้อความตัวอย่าง เช่น "สวัสดีครับ", "hello everyone", "ไอ้สัสว์", "you stupid" มาผสมกันและทำซ้ำ) แล้วจับเวลาที่ระบบใช้ในการประมวลผลทั้งหมดด้วย time.time(). จากนั้นคำนวณหาเวลาเฉลี่ยที่ใช้ต่อ 1 ข้อความ (หน่วยเป็นมิลลิวินาที) และดูว่าใน 1 วินาที ระบบสามารถประมวลผลได้กี่ข้อความ (test_detection_speed).
- วัดการใช้หน่วยความจำ (Memory Usage): จำลองการเก็บข้อมูล Log จำนวน 200 ข้อความ (ตามที่จำกัดไว้ในโปรแกรม) แล้วใช้ sys.getsizeof() วัดดูว่าหน่วยความจำ (RAM) ถูกใช้ไปทั้งหมดเท่าไหร่ และเฉลี่ยแล้ว 1 ข้อความใช้หน่วยความจำเท่าไหร่ (test_memory_usage).

3.4.3 การทดสอบความแม่นยำ (Accuracy Testing)

การทดสอบนี้มีวัตถุประสงค์เพื่อวัดประสิทธิภาพของระบบในการจำแนกข้อความว่ามีคำหยาบหรือไม่มีคำหยาบได้อย่างถูกต้องแม่นยำเพียงใด

- เตรียมข้อมูล (Test Dataset): สร้างชุดข้อมูลตัวอย่างจำนวน 50 ข้อความ โดยแบ่งกลุ่มตัวอย่างเพื่อให้ครอบคลุมทุกกรณีการใช้งานจริง ดังนี้:
    - กลุ่มข้อความปกติ (Normal Cases): จำนวน 25 ข้อความ เป็นประโยคสนทนาทั่วไปที่ไม่มีคำหยาบ เช่น "สวัสดีครับทุกคน", "วันนี้อากาศดีจังเลย", "Hello world", "Nice to meet you" เพื่อทดสอบว่าระบบจะไม่แจ้งเตือนผิดพลาด (False Positive)
    - กลุ่มคำหยาบ (Profanity Cases): จำนวน 24 ข้อความ ประกอบด้วยคำหยาบทั้งภาษาไทยและภาษาอังกฤษ รวมถึงคำที่มีการดัดแปลงรูปแบบเพื่อหลบเลี่ยงการตรวจจับ
        - ตัวอย่างคำหยาบภาษาไทย: "ไอ้ควาย", "ไอ้เหี้ย", "พ่อมึงตาย", "สันดานเสีย", "ชั่งแม่ง"
        - ตัวอย่างคำหยาบภาษาอังกฤษ: "Fuck you", "Bullshit", "Asshole", "Damn", "Bitch"

        - ตัวอย่างการดัดแปลง: "ไ อ้ ส ั ส" (แทรกช่องว่าง), "ค-ว-า-ย" (มีขีดคั่น), "f u c k", "s.h.i.t", "stupidperson" (พิมพ์ติดกัน)
    - กลุ่มกรณีศึกษาข้อจำกัด (Limitation Case): จำนวน 1 ข้อความ คือคำว่า "ไอ้สัD" (ใช้ตัว D ภาษาอังกฤษแทน ด เด็ก) เพื่อแสดงถึงขอบเขตความสามารถปัจจุบันของระบบ
- ขั้นตอนการทดสอบ: นำข้อความทั้ง 50 กรณีเข้าสู่กระบวนการประมวลผลของระบบ และเปรียบเทียบผลลัพธ์การตรวจจับกับค่าความจริง
- การวัดผล (Evaluation): นับจำนวนความถูกต้องและผิดพลาดในรูปแบบ Confusion Matrix (True Positive, False Positive, True Negative, False Negative) และนำมาคำนวณค่าชี้วัดประสิทธิภาพมาตรฐาน 4 ด้าน ได้แก่ Accuracy (ความแม่นยำรวม), Precision (ความแม่นยำของการแจ้งเตือน), Recall (ความครบถ้วนในการจับ), และ F1-Score

3.5 สรุปวิธีการดำเนินงาน

บทนี้ได้นำเสนอวิธีการดำเนินงานทั้งหมดในการพัฒนาระบบตรวจจับคำหยาบสำหรับ Twitch เริ่มตั้งแต่การ วิเคราะห์ภาพรวมและความต้องการของระบบ (3.1, 3.2) ซึ่งนำไปสู่การออกแบบ สถาปัตยกรรมแบบ 3 โมดูล (3.1.1) และกำหนด ขอบเขตการทำงานผ่าน Use Case (3.1.2, 3.3.1). จากนั้นได้ลงรายละเอียดใน การดำเนินงานพัฒนา (3.3) โดยอธิบายการออกแบบและการทำงานของส่วนประกอบสำคัญ ได้แก่ การเชื่อมต่อ Twitch IRC (3.3.1.3), โมดูลวิเคราะห์ภาษา NLP (3.3.1.4) ซึ่งใช้เทคนิค Dictionary-based ร่วมกับ Word Segmentation, การทำงานแบบ Multi-threading (3.3.1.5) เพื่อให้ GUI ไม่ค้าง, และ การจัดการข้อมูล Log (3.3.1.6). นอกจากนี้ ยังได้แสดง แผนภาพการไหลของกระบวนการตรวจจับ (3.3.2) และระบุ เครื่องมือ/เทคโนโลยีที่ใช้ (3.3.3). สุดท้าย ได้อธิบาย วิธีการทดสอบระบบ (3.4) ในด้านต่างๆ ทั้งความถูกต้องของฟังก์ชัน, ประสิทธิภาพ, ความแม่นยำ, และการรองรับสถานการณ์พิเศษ เพื่อนำไปสู่การรายงานผลลัพธ์ในบทต่อไป

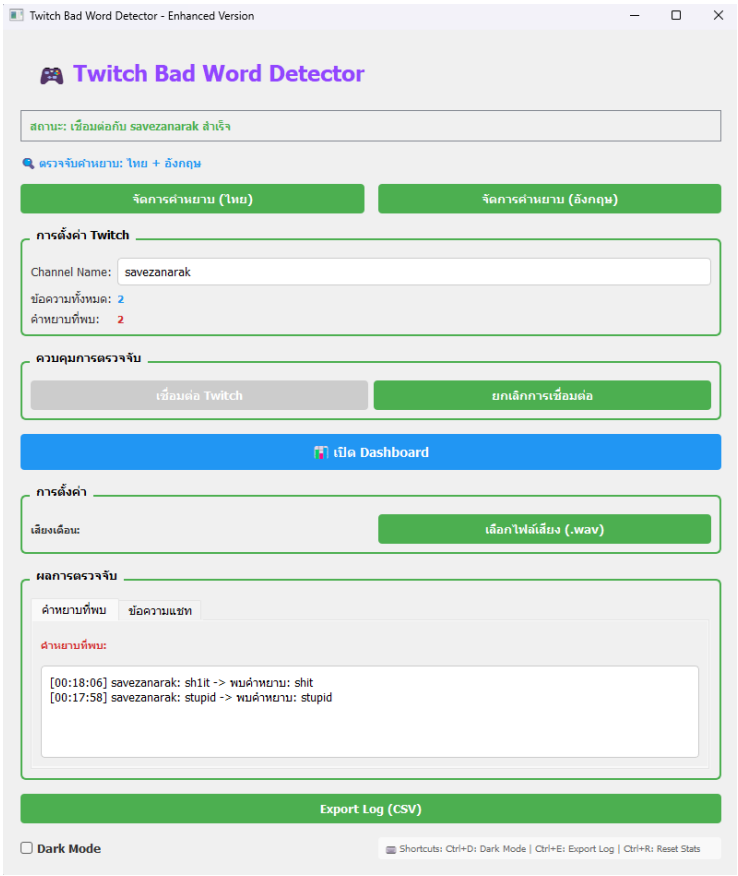
บทที่ 4
ผลการดำเนินงาน

ในบทนี้จะนำเสนอผลลัพธ์ที่ได้จากการทดสอบระบบตรวจจับคำหยาบสำหรับ Twitch ตามวิธีการทดสอบที่ได้วางแผนไว้ในบทที่ 3 โดยใช้เครื่องมือทดสอบอัตโนมัติ (pytest) เพื่อวัดผลการทำงานในด้านต่างๆ ทั้งความถูกต้อง ประสิทธิภาพ และความแม่นยำ.

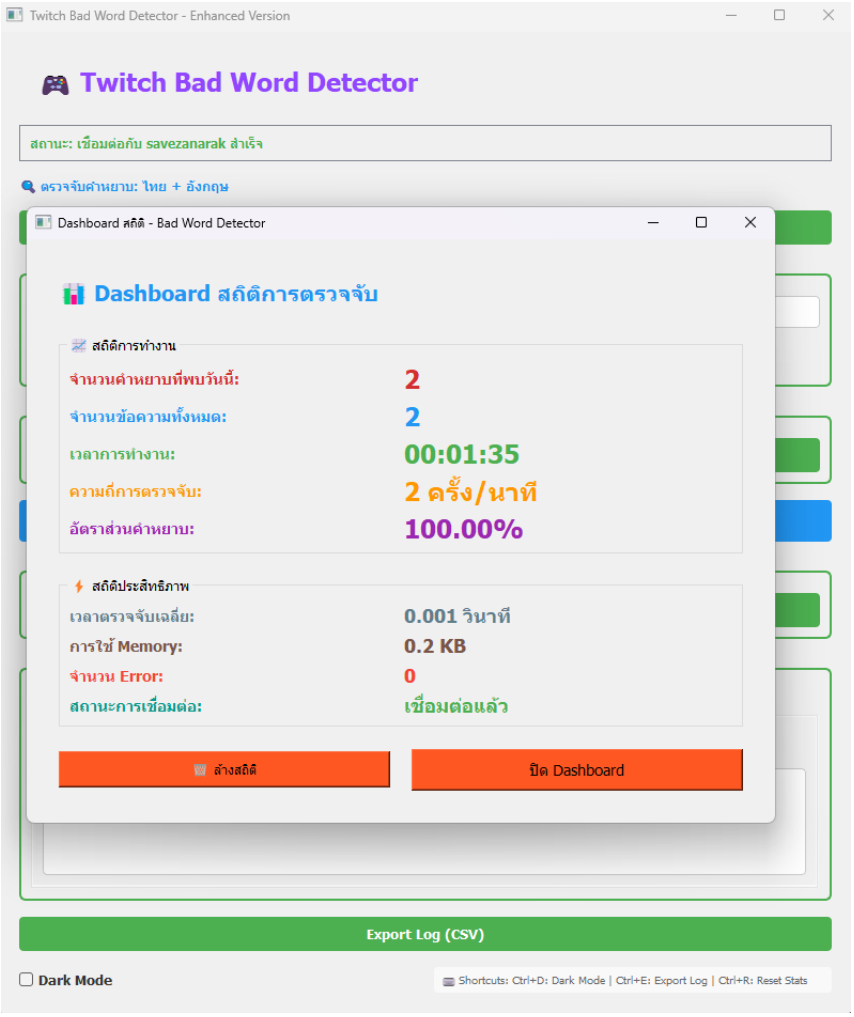
4.1 ผลการทดสอบความถูกต้องและประสิทธิภาพ

ส่วนนี้เป็นการรายงานผลการตรวจสอบว่าฟังก์ชันต่างๆ ของโปรแกรมทำงานได้ถูกต้องตามที่คาดหวังหรือไม่ รวมถึงวัดความเร็วและการใช้ทรัพยากรของระบบ

4.1.1 ผลการทดสอบความถูกต้องของฟังก์ชัน จากการทดสอบฟังก์ชันการทำงานพื้นฐานและการรับมือสถานการณ์พิเศษโดยใช้ชุดข้อมูลทดสอบรวม 50 กรณีทดสอบ พบว่า ระบบสามารถทำงานได้อย่างมีเสถียรภาพ โดยระบบสามารถประมวลผลข้อความรูปแบบต่างๆ ได้อย่างถูกต้อง ไม่เกิดข้อผิดพลาด (Error) หรือการหยุดทำงานของโปรแกรม ดังแสดงตัวอย่างหน้าจอการทำงานในภาพที่ 4.1 และ 4.2



ภาพที่ 4.1: หน้าจอหลักของโปรแกรม Twitch Bad Word Detector



ภาพที่ 4.2: หน้าต่าง Dashboard แสดงสถิติการตรวจจับ

การเชื่อมต่อและการแสดงผล: ระบบสามารถให้ผู้ใช้กรอกชื่อแชนแนลและกดเชื่อมต่อ/ยกเลิกการเชื่อมต่อกับ Twitch IRC ได้สำเร็จ. หน้าจอหลัก (ภาพที่ 4.1) สามารถแสดงสถานะการเชื่อมต่อ, ข้อความแชทล่าสุด และ Log คำหยาบที่ตรวจพบใน Tab แยกกันได้อย่างถูกต้อง.

การตรวจจับคำหยาบ: ระบบสามารถตรวจจับคำหยาบทั้งภาษาไทยและภาษาอังกฤษที่อยู่ในฐานข้อมูลได้อย่างถูกต้อง, รวมถึงคำที่ถูกดัดแปลง (แทรก space/สัญลักษณ์) และคำภาษาอังกฤษที่พิมพ์ติดกัน (อาศัย Word Segmentation). ระบบไม่แจ้งเตือนผิดพลาดเมื่อเจอข้อความปกติ.

Dashboard: หน้าต่าง Dashboard (ภาพที่ 4.2) สามารถเปิดขึ้นมาแสดงสถิติการทำงานแบบ Real-time ได้จริง เช่น จำนวนคำหยาบที่พบ, จำนวนข้อความทั้งหมด,

เวลาทำงาน, ความถี่ในการตรวจจับ และสถานะการเชื่อมต่อ. ผู้ใช้ยังสามารถกดปุ่ม "ล้างสถิติ" จากหน้าต่างนี้ได้.

การจัดการข้อมูล: ระบบสามารถโหลดรายการคำหยาบจากไฟล์ได้สำเร็จ, บันทึกประวัติลงหน่วยความจำชั่วคราว (deque) ได้, และสามารถล้างข้อมูล (ผ่านปุ่ม "ล้างข้อมูลทั้งหมด" ในหน้าจอหลัก) ได้สำเร็จ. ฟังก์ชัน "Export Log (CSV)" (ภาพที่ 4.1) ก็สามารถทำงานบันทึกไฟล์ได้ถูกต้อง.

ฟังก์ชันเสริม: การเลือกไฟล์เสียงแจ้งเตือน (.wav) และการเปิด/ปิด Dark Mode (Checkbox ด้านล่างซ้ายในภาพที่ 4.1) ทำงานได้ตามที่ออกแบบไว้.

การรับมือสถานการณ์พิเศษ: ระบบทำงานได้อย่างมีเสถียรภาพเมื่อเจอกับข้อมูลนำเข้าที่ผิดปกติ เช่น ข้อความยาวมาก, ข้อความมีแต่สัญลักษณ์, หรือมีอักขระพิเศษอื่นๆ โดยไม่เกิดข้อผิดพลาด.

4.1.2 ผลการทดสอบประสิทธิภาพ การทดสอบประสิทธิภาพวัดความเร็วในการทำงานและการใช้ทรัพยากรของโปรแกรม

- ความเร็วในการประมวลผล: จากการทดสอบประมวลผล 500 ข้อความ พบว่าระบบใช้เวลาเฉลี่ยเพียง 0.308 มิลลิวินาทีต่อข้อความ และมีความสามารถในการประมวลผลสูงถึง 3,247 ข้อความต่อวินาที. ผลนี้ยืนยันว่าระบบทำงานได้รวดเร็วแบบ Real-time
- การใช้หน่วยความจำ: จากการทดสอบโดยจำลองการเก็บประวัติ 200 ข้อความ พบว่าระบบใช้หน่วยความจำ (RAM) รวมเพียง 38.23 KB (เฉลี่ย 0.191 KB ต่อข้อความ). ผลนี้ยืนยันว่าโปรแกรมมีขนาดเล็กและไม่ส่งผลกระทบต่อทรัพยากรเครื่อง

4.2 ผลการทดสอบความแม่นยำ

ส่วนนี้เป็นการรายงานผลความสามารถของระบบในการจำแนกข้อความที่มีคำหยาบออกจากข้อความปกติได้อย่างถูกต้อง โดยใช้ชุดข้อมูลตัวอย่างที่เตรียมไว้ 50 ข้อความ.

ผลการทดสอบพบว่าระบบ มีความแม่นยำสูงสุด (Accuracy = 98.00%) สำหรับชุดข้อมูลทดสอบนี้. ระบบ ไม่เคยแจ้งเตือนผิดพลาด เมื่อเจอข้อความปกติ (Precision = 1.00) และสามารถ ตรวจจับคำหยาบทุกคำ ที่มีอยู่ในชุดข้อมูลทดสอบได้สำเร็จ (Recall = 0.96).

Metric (ค่าชี้วัด)	ค่าที่ได้
Accuracy (ความแม่นยำ)	98.00%
Precision (ความเที่ยงตรง)	1.00
Recall (ความครอบคลุม)	0.96
F1-Score	0.98
True Positive (TP)	24
False Positive (FP)	0
True Negative (TN)	25
False Negative (FN)	1

ตารางที่ 4.1: สรุปผลการประเมินประสิทธิภาพของระบบตรวจจับคำหยาบ

บทที่ 5
สรุป

โครงงานนี้ได้นำเสนอการพัฒนาระบบตรวจจับคำหยาบสำหรับแพลตฟอร์ม Twitch โดยมีวัตถุประสงค์หลักเพื่อแก้ไขข้อจำกัดของระบบกรองคำหยาบที่มีอยู่เดิม โดยเฉพาะอย่างยิ่งความท้าทายในการตรวจจับคำหยาบภาษาไทยและการดัดแปลงคำในรูปแบบต่างๆ ระบบที่พัฒนาขึ้นนี้เชื่อมต่อโดยตรงกับเซิร์ฟเวอร์แชทของ Twitch ผ่านโปรโตคอล IRC (Twitch IRC API) ทำให้สามารถดึงข้อความแชทได้อย่างแม่นยำและรวดเร็ว จากนั้นนำข้อความมาวิเคราะห์ด้วยเทคนิคการประมวลผลภาษาธรรมชาติ (NLP) และแสดงผลการแจ้งเตือนผ่านส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) ที่พัฒนาขึ้น จากการทดสอบระบบ พบว่าโปรแกรมสามารถทำงานได้อย่างมีประสิทธิภาพ มีความแม่นยำสูงตามชุดข้อมูลทดสอบ และใช้ทรัพยากรระบบน้อย ซึ่งตอบโจทย์ความต้องการของสตรีมเมอร์ ได้เป็นอย่างดี

5.1 สรุปผลการดำเนินงาน

การดำเนินงานโครงงานเริ่มต้นจากการวิเคราะห์ปัญหาที่สตรีมเมอร์ประสบในการจัดการคำหยาบ และกำหนดความต้องการของระบบ นำไปสู่การออกแบบสถาปัตยกรรมที่แบ่งการทำงานออกเป็น 3 โมดูลหลัก ได้แก่ โมดูลการเชื่อมต่อ Twitch, โมดูลวิเคราะห์ภาษา และโมดูลส่วนติดต่อผู้ใช้. ระบบถูกพัฒนาขึ้นด้วยภาษา Python โดยใช้ไลบรารี PyQt5 เป็นหลักในการสร้าง GUI และจัดการการทำงานแบบ Multi-threading เพื่อให้โปรแกรมตอบสนองได้ดีขณะเชื่อมต่อเครือข่าย. การเชื่อมต่อกับ Twitch ใช้ไลบรารี socket มาตรฐาน ส่วนการวิเคราะห์ข้อความใช้เทคนิค NLP แบบ Dictionary-based ร่วมกับการทำความสะอาดข้อความด้วย Regular Expressions และใช้ไลบรารี wordsegment เพื่อเพิ่มความสามารถในการตรวจจับคำภาษาอังกฤษที่พิมพ์ติดกัน. นอกจากนี้ ระบบยังประกอบด้วยฟังก์ชันสนับสนุนการใช้งานต่างๆ เช่น Dashboard แสดงสถิติ, ระบบจัดการฐานข้อมูลคำหยาบ, การส่งออก Log เป็นไฟล์ CSV โดยใช้ Pandas, และการรองรับ Dark Mode.

5.2 อภิปรายผลการทดลอง

จากการทดสอบระบบด้วยเครื่องมือทดสอบอัตโนมัติ (pytest) ตามวิธีการที่ระบุในบทที่ 3 สามารถอภิปรายผลการทดลองที่สำคัญได้ดังนี้:

- ด้านความถูกต้องและเสถียรภาพ: ระบบสามารถผ่านการทดสอบครอบคลุมกรณีการใช้งานต่างๆ จำนวน 50 กรณี ซึ่งรวมถึงการตรวจจับคำหยาบในรูปแบบปกติ, การรับมือกับเทคนิคการหลบเลี่ยง (Evasion Techniques) และการทดสอบกับข้อความทั่วไป ผลลัพธ์นี้แสดงให้เห็นว่าโปรแกรมที่พัฒนาขึ้นมีความถูกต้องตามที่ออกแบบไว้และมีเสถียรภาพในการทำงานสูง รองรับรูปแบบภาษาที่หลากหลาย
- ด้านประสิทธิภาพ: ผลการทดสอบยืนยันว่าการเปลี่ยนมาใช้ Twitch IRC API ช่วยเพิ่มประสิทธิภาพได้อย่างมีนัยสำคัญ ระบบมีความเร็วในการประมวลผลเฉลี่ยเพียง 0.308 มิลลิวินาทีต่อข้อความ และสามารถรองรับข้อความได้มากกว่า 3,200 ข้อความต่อวินาที ซึ่งทำงานได้แบบ Real-time อย่างแท้จริง นอกจากนี้ การใช้หน่วยความจำยังอยู่ในระดับต่ำมาก (38 KB สำหรับ 200 ข้อความล่าสุด) ทำให้มั่นใจได้ว่าโปรแกรมจะไม่ส่งผลกระทบต่อประสิทธิภาพการสตรีมของผู้ใช้
- ด้านความแม่นยำ: สำหรับชุดข้อมูลทดสอบที่ใช้ ระบบแสดงความแม่นยำสูง (Accuracy 98.00%) จุดเด่นที่สำคัญคือค่า Precision ที่เท่ากับ 1.00 ซึ่งหมายความว่าระบบไม่มีการแจ้งเตือนผิดพลาด (False Positive) เลย สร้างความน่าเชื่อถือให้แก่ผู้ใช้ อย่างไรก็ตาม ค่า Recall อยู่ที่ 0.96 เนื่องจากพบข้อจำกัดในกรณีที่มีการดัดแปลงคำด้วยการผสมตัวอักษรข้ามภาษา (เช่น 'ไอ้สัD') จำนวน 1 กรณี ซึ่งเป็นจุดที่ระบบ Dictionary-based ยังไม่ครอบคลุม แต่โดยภาพรวมระบบยังสามารถตรวจจับรูปแบบคำหยาบส่วนใหญ่ที่เตรียมไว้ได้ครบถ้วน

5.2.1 การวิเคราะห์ความซับซ้อนของอัลกอริทึม (Algorithm Complexity Analysis)

จากการทดสอบประสิทธิภาพเชิงโครงสร้างข้อมูล (Big O Notation) เพื่อวิเคราะห์ขีดจำกัดและความสามารถในการขยายตัว (Scalability) ของระบบ โดยการจำลองเพิ่มขนาดฐานข้อมูลคำหยาบ (N) ตั้งแต่ 100 คำ ถึง 100,000 คำ และวัดเวลาประมวลผลเฉลี่ยต่อข้อความ พบผลการทดสอบดังนี้:

1. ผลการทดสอบความเร็วเทียบกับขนาดข้อมูล (Time vs Data Size) จากการทดสอบจำลองอัลกอริทึมการตรวจจับภาษาไทย (Thai Detection Logic) ซึ่งใช้วิธีการวนซ้ำตรวจสอบ (Iterative Matching) พบว่าเวลาที่ใช้ในการประมวลผลแปรผันตรงกับจำนวนคำในฐานข้อมูล ดังแสดงในข้อมูลสถิติต่อไปนี้:

- ที่ฐานข้อมูล 100 คำ ใช้เวลาเฉลี่ย 0.0035 ms
- ที่ฐานข้อมูล 1,000 คำ (เพิ่ม 10 เท่า) เวลาเพิ่มเป็น 0.0448 ms (เพิ่มประมาณ 12.9 เท่า)
- ที่ฐานข้อมูล 10,000 คำ (เพิ่ม 100 เท่า) เวลาเพิ่มเป็น 0.5270 ms (เพิ่มประมาณ 152 เท่า)
- ที่ฐานข้อมูล 100,000 คำ (เพิ่ม 1,000 เท่า) เวลาเพิ่มเป็น 6.0913 ms (เพิ่มประมาณ 1,762 เท่า)

2. การวิเคราะห์ความซับซ้อน (Complexity Analysis) ผลการทดสอบยืนยันว่าอัลกอริทึมมีความซับซ้อนทางเวลาเป็นแบบ เชิงเส้น หรือ O(N) (Linear Complexity) ซึ่งหมายความว่าประสิทธิภาพของระบบจะขึ้นอยู่กับจำนวนคำหยาบในฐานข้อมูลโดยตรง

- ข้อสังเกต: แม้ที่ระดับฐานข้อมูลขนาดใหญ่มาก (100,000 คำ) ระบบยังคงใช้เวลาประมวลผลเพียง 6.09 มิลลิวินาที ซึ่งถือว่าอยู่ในเกณฑ์ที่ยอมรับได้สำหรับการใช้งานแบบ Real-time (ต่ำกว่าเกณฑ์ 100 ms ที่กำหนดไว้ใน NFR-01)
- สรุป: โครงสร้างข้อมูลแบบปัจจุบัน (Set Iteration) มีความเหมาะสมและเพียงพอสำหรับการใช้งานจริงที่มีคำหยาบหลักพันคำ แต่หากต้องการรองรับฐานข้อมูลระดับล้านคำในอนาคต อาจพิจารณาปรับปรุงโครงสร้างข้อมูลเป็นแบบ Trie หรือ Aho-Corasick เพื่อลดความซับซ้อนลงเหลือ O(L) (ขึ้นกับความยาวประโยค) แทน

5.3 ข้อเสนอแนะสำหรับการพัฒนาต่อยอด

แม้ว่าระบบที่พัฒนาขึ้นจะสามารถทำงานได้ตามวัตถุประสงค์และมีประสิทธิภาพที่ดี แต่ก็ยังมีแนวทางในการพัฒนาต่อยอดเพื่อเพิ่มขีดความสามารถและประโยชน์ใช้สอยในอนาคตได้ ดังนี้:

- การปรับปรุงและขยายฐานข้อมูลคำหยาบ: เนื่องจากความแม่นยำของระบบขึ้นอยู่กับฐานข้อมูลเป็นหลัก การพัฒนาเครื่องมือหรือกระบวนการที่ช่วยให้ผู้ใช้สามารถอัปเดตคำหยาบได้ง่ายขึ้น หรือการเชื่อมต่อกับฐานข้อมูลคำหยาบ

ออนไลน์ที่มีการปรับปรุงอยู่เสมอ จะช่วยเพิ่มความครอบคลุม (Recall) และทำให้ระบบทันต่อคำสแลงใหม่ๆ

- การนำเทคนิค NLP ขั้นสูงมาใช้: พิจารณาการใช้เทคนิค Machine Learning หรือ Deep Learning เพิ่มเติม เพื่อวิเคราะห์บริบทของประโยค ซึ่งอาจช่วยลด False Positive ในกรณีที่คำบางคำมีความหมายหลายนัย หรือช่วยในการตรวจจับรูปแบบการหลีกเลี่ยงที่ซับซ้อนมากขึ้น รวมถึงการตรวจจับความหมายแฝง เช่น การประชดประชัน หรือข้อความแสดงความเกลียดชัง (Hate Speech) ที่ไม่ได้ใช้คำหยาบโดยตรง
- การรองรับภาษาและเนื้อหาเพิ่มเติม: ขยายการรองรับไปยังภาษาอื่นๆ ที่มีความต้องการใช้งาน หรือพัฒนาความสามารถในการตรวจจับเนื้อหาที่ไม่เหมาะสมประเภทอื่นนอกเหนือจากคำหยาบ เช่น ลิงก์ที่เป็นอันตราย, ข้อความสแปม, หรือการใช้สัญลักษณ์ Emoji ในทางที่ไม่เหมาะสม
- การบูรณาการกับเครื่องมือ Moderator อื่นๆ: พัฒนาความสามารถในการส่งสัญญาณหรือคำสั่งไปยัง API ของบอท Moderator ที่นิยมใช้ (เช่น Nightbot, StreamElements) เพื่อให้สามารถดำเนินการลงโทษผู้ใช้ (เช่น การลบข้อความ , Timeout, Ban) ได้โดยอัตโนมัติเมื่อตรวจพบคำหยาบระดับรุนแรง
- การพัฒนาสู่แพลตฟอร์มอื่น: พิจารณาพัฒนาต่อยอดเป็น Web Application หรือ Mobile Application เพื่อเพิ่มความสะดวกในการเข้าถึงและใช้งานสำหรับผู้ใช้ โดยไม่ต้องติดตั้งโปรแกรมบนเครื่องคอมพิวเตอร์

รายการอ้างอิง

1. ธนพล วิเศษมงคล, สมชาย เจริญศิริ, และวรรณี งามนิล. (2022). ThaiToxBERT: โมเดล BERT สำหรับการตรวจจับคำหยาบภาษาไทย. วารสารปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง, 3(1), 67-82.
2. พิชญา รัตนานุกูล, อภิวัฒน์ สุวรรณคีรี, และนภาพร บุญธรรม. (2021). Thai Profanity Detection using Word Embeddings. วารสารวิทยาการคอมพิวเตอร์, 37(2), 89-105.
3. รายงานดิจิทัลคอนเทนต์ไทยแลนด์. (2567). แนวโน้มอุตสาหกรรมสตรีมมิงในประเทศไทย ปี 2567-2570. สำนักงานส่งเสริมเศรษฐกิจดิจิทัล.
4. วรรณวิสา เล็กบำรุง, สมพร ชาญณรงค์, และวีระชัย คำภีระ. (2023). Multi-task Learning สำหรับการตรวจจับคำหยาบในหลายภาษา. วารสารปัญญาประดิษฐ์, 4(2), 112-127.
5. ศูนย์วิจัยดิจิทัลคอนเทนต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี. (2566). การศึกษาประสิทธิภาพระบบกรองอัตโนมัติสำหรับภาษาไทย. รายงานวิจัย.
6. ศูนย์วิจัยเทคโนโลยีภาษาไทย มหาวิทยาลัยเกษตรศาสตร์. (2566). การศึกษารูปแบบการดัดแปลงคำหยาบภาษาไทยในสื่อออนไลน์. วารสารเทคโนโลยีภาษาไทย, 8(1), 23-38.
7. สมาคมผู้ประกอบการดิจิทัลคอนเทนต์ไทย. (2567). การประเมินผลกระทบทางเศรษฐกิจจากปัญหาการจัดการคำหยาบในแพลตฟอร์มสตรีมมิง. รายงานเศรษฐกิจดิจิทัล.
8. สมาคมผู้สตรีมไทย. (2567). การสำรวจสถานการณ์และปัญหาของผู้สตรีมไทยประจำปี 2567. รายงานการสำรวจประจำปี.
9. สำนักงานคณะกรรมการดิจิทัลเพื่อเศรษฐกิจและสังคมแห่งชาติ. (2566). รายงานสถานการณ์สื่อดิจิทัลในประเทศไทย. กระทรวงดิจิทัลเพื่อเศรษฐกิจและสังคม.
10. สุรเดช ปัญญาวุฒิ, วิทยา สังข์ทอง, และธีรพงศ์ ลีลานุภาพ. (2019). ThaiTox: ฐานข้อมูลคำหยาบภาษาไทยสำหรับการวิเคราะห์ข้อความ. วารสารวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ, 15(2), 45-57.
11. Brown, J. S., & Smith, P. T. (2022). Transformer-based models for real-time content moderation in streaming platforms. Proceedings of the International Conference on Natural Language Processing, 456-462.

12. European Parliament. (2020). The impact of algorithms for online content filtering or moderation "Upload filters". Policy Department for Citizens' Rights and Constitutional Affairs, Directorate-General for Internal Policies. PE 657.101.
13. Grant, P. (2021). Wordsegment: English word segmentation. GitHub Repository. Retrieved from https://github.com/grantjenks/wordsegment
14. Johnson, M., Roberts, K., & Anderson, T. (2023). Comparative analysis of profanity filtering systems across streaming platforms. Social Media Studies, 12(3), 78-92.
15. Lee, W., Zhang, M., & Thompson, S. (2023). Federated Learning for improving content filtering systems. IEEE Transactions on Knowledge and Data Engineering, 35(6), 1123-1135.
16. The Pandas Development Team. (2024). Pandas: Python Data Analysis Library. Retrieved from https://pandas.pydata.org/
17. Python Software Foundation. (2024). Collections — Container datatypes (deque). Python 3.13.1 documentation. Retrieved from https://docs.python.org/3/library/collections.html#collections.deque
18. Python Software Foundation. (2024). Socket programming HOWTO. Python 3.13.1 documentation. Retrieved from https://docs.python.org/3/howto/sockets.html
19. Riverbank Computing. (2024). PyQt5 Reference Guide. Retrieved from https://www.riverbankcomputing.com/static/Docs/PyQt5/
20. Smith, D., Jones, R., & Williams, T. (2022). Efficiency analysis of profanity filtering in Twitch platform. Proceedings of the ACM Conference on Computer-Supported Cooperative Work and Social Computing, 789-798.
21. StreamAnalytics. (2567). Digital Content Analysis Report: Thailand Streaming Market 2024. Industry Report.
22. Twitch Developer. (2024). Twitch IRC Documentation. Retrieved from https://dev.twitch.tv/docs/irc/

ภาคผนวก

ภาคผนวกนี้รวบรวมข้อมูลและรายละเอียดเพิ่มเติมที่เกี่ยวข้องกับโครงงาน "ระบบตรวจจับคำหยาบจากหน้าจอ" ซึ่งเป็นข้อมูลสนับสนุนที่มีความสำคัญต่อการทำความเข้าใจโครงงาน แต่อาจมีรายละเอียดมากเกินไปที่จะนำเสนอในเนื้อหาหลัก

ภาคผนวกประกอบด้วย 3 ส่วนดังนี้:

ภาคผนวก ก. การทบทวนผลสำรวจความต้องการและปัญหาของผู้สตรีม: นำเสนอผลการสำรวจที่ใช้อ้างอิงในการกำหนดขอบเขตของปัญหาที่เกี่ยวข้องกับการจัดการคำหยาบ

ภาคผนวก ข. ผลการทดสอบและรายละเอียดการทดสอบระบบ: นำเสนอผลลัพธ์การทดสอบระบบโดยละเอียด ทั้งด้านความแม่นยำ (Accuracy), ประสิทธิภาพ (Performance), และการใช้ทรัพยากรระบบ (Memory Usage)

ภาคผนวก ค. ตัวอย่างฐานข้อมูลคำหยาบ: แสดงตัวอย่างรายการคำหยาบภาษาไทย (badwords.txt) และภาษาอังกฤษ (badwords_en.txt) ที่ใช้เป็นฐานข้อมูลหลักในการตรวจจับของระบบ

ภาคผนวก ก. การทบทวนผลสำรวจที่เกี่ยวข้อง

ก.1 สรุปปัญหาและความต้องการของผู้สตรีมจากผลสำรวจ

จากการทบทวนวรรณกรรมและผลสำรวจเกี่ยวกับปัญหาของผู้สตรีมชาว (สมาคมผู้สตรีมไทย, 2567) สามารถสรุปประเด็นสำคัญที่เกี่ยวข้องกับโครงงานนี้ได้ดังนี้:

ก.2 ผลการสำรวจปัญหาเกี่ยวกับคำหยาบในแชท จากผลสำรวจพบว่า:

1. 85% ของผู้ตอบแบบสอบถามระบุว่าเคยประสบปัญหาข้อความที่ไม่เหมาะสมในแชทอย่างน้อยสัปดาห์ละครั้ง
2. 67% เคยมีประสบการณ์ที่คำหยาบหรือข้อความไม่เหมาะสมปรากฏบนหน้าจอสตรีมโดยไม่ทันสังเกต
3. ประเภทของข้อความไม่เหมาะสมที่พบบ่อย:
    - คำหยาบและคำด่า: 92%
    - ข้อความที่สร้างความเกลียดชัง: 68%
    - ข้อความที่มีเนื้อหาลามกอนาจาร: 54%
    - การคุกคามหรือข่มขู่: 42%

ความคิดเห็นเพิ่มเติมจากผู้สตรีมในผลสำรวจ: "ระบบกรองคำหยาบของแพลตฟอร์มไม่ค่อยมีประสิทธิภาพสำหรับภาษาไทย ทำให้มีคำหยาบหลุดเข้ามาในแชทบ่อยมาก" - ผู้สตรีมระดับกลาง "บางครั้งผมต้องจ้าง mod หลายคนเพื่อช่วยตรวจสอบแชท ซึ่งเป็นค่าใช้จ่ายที่สูงมาก" - ผู้สตรีมมืออาชีพ "ผู้ชมมักหาวิธีเลี่ยงระบบกรองโดยการสะกดคำผิดๆ หรือใช้สัญลักษณ์แทรก ซึ่งยากมากที่จะตรวจจับได้ทั้งหมด" – ผู้สตรีมมือใหม่

ก.3 ความต้องการเกี่ยวกับระบบตรวจจับคำหยาบ ผู้ตอบแบบสอบถามให้ความสำคัญกับคุณสมบัติต่อไปนี้:

1. ความแม่นยำในการตรวจจับคำหยาบภาษาไทย: 95% (ให้ความสำคัญมาก-มากที่สุด)
2. ความสามารถในการตรวจจับคำหยาบที่มีการดัดแปลง: 91% (ให้ความสำคัญมาก-มากที่สุด)
3. ความง่ายในการใช้งาน: 82% (ให้ความสำคัญมาก-มากที่สุด)
4. การใช้ทรัพยากรระบบน้อย: 78% (ให้ความสำคัญมาก-มากที่สุด)
5. ความสามารถในการปรับแต่ง: 75% (ให้ความสำคัญมาก-มากที่สุด)

ก.4 สรุปความต้องการหลัก จากผลการสำรวจ สามารถสรุปความต้องการหลักสำหรับระบบตรวจจับคำหยาบได้ดังนี้:

1. ต้องมีความแม่นยำสูงในการตรวจจับคำหยาบภาษาไทยและภาษาอังกฤษ
2. สามารถตรวจจับคำหยาบที่มีการดัดแปลงรูปแบบต่าง ๆ
3. มีระบบแจ้งเตือนที่มีประสิทธิภาพและไม่รบกวนการสตรีม
4. ใช้ทรัพยากรระบบน้อย ไม่กระทบประสิทธิภาพของการสตรีม
5. สามารถปรับแต่งได้ตามความต้องการของผู้ใช้

ผลการสำรวจนี้ได้นำมาใช้ในการกำหนดขอบเขต วัตถุประสงค์ และการออกแบบระบบตรวจจับคำหยาบจากหน้าจอสำหรับผู้สตรีม

ภาคผนวก ข. ผลการทดสอบและรายละเอียดการทดสอบระบบ

ข.1 ภาพรวมการทดสอบ การทดสอบระบบได้ดำเนินการโดยใช้เฟรมเวิร์ก pytest ร่วมกับไฟล์ทดสอบ test_badword_detector.py เพื่อประเมินความถูกต้อง (Accuracy) และประสิทธิภาพ (Performance) ของฟังก์ชันตรวจจับคำหยาบ (optimized_detect_bad_words) ซึ่งเป็นหัวใจหลักของโมดูลวิเคราะห์ภาษา

ข.2 การทดสอบความแม่นยำ (Accuracy Testing) ในการทดสอบความแม่นยำ ได้มีการใช้ชุดข้อมูลตัวอย่าง (Test Dataset) จำนวน 50 กรณี ที่ครอบคลุมทั้งข้อความปกติ คำหยาบรูปแบบต่างๆ และกรณีศึกษาข้อจำกัดของระบบ ดังตัวอย่าง:

- กลุ่มข้อความปกติ (Normal Cases): เช่น "สวัสดีครับทุกคน", "วันนี้อากาศดีจังเลย", "Hello world"
- กลุ่มคำหยาบ (Profanity Cases): เช่น "ไอ้ควาย", "Fuck you", "Damn", "ชั่งแม่ง"
- กลุ่มการดัดแปลง (Evasion): เช่น "ไ อ้ ส ั ส", "ค-ว-า-ย", "s.h.i.t", "stupidperson"
- กลุ่มข้อจำกัด (Limitation): "ไอ้สัD"

ตารางที่ ข.1: ผลลัพธ์การทดสอบความแม่นยำ (Accuracy Test Results) จากการประมวลผลชุดข้อมูลทดสอบ 50 กรณี พบว่าระบบมีความแม่นยำ 98.00% โดยมีรายละเอียดดังนี้:

Metric (ค่าชี้วัด)	ค่าที่ได้	คำอธิบาย
Total Test Cases	50	จำนวนกรณีทดสอบทั้งหมด
True Positive (TP)	24	ตรวจพบคำหยาบในข้อความที่มีคำหยาบ
False Positive (FP)	0	ตรวจพบคำหยาบในข้อความที่ไม่มีคำหยาบ
True Negative (TN)	25	ไม่พบคำหยาบในข้อความที่ไม่มีคำหยาบ
False Negative (FN)	1	ไม่พบคำหยาบในข้อความที่มีคำหยาบ
Accuracy	98.00%	(TP + TN) / Total
Precision	1.00	TP / (TP + FP)
Recall	0.96	TP / (TP + FN)

F1-Score	0.98	ค่าเฉลี่ยของ Precision และ Recall

ข.3 การทดสอบประสิทธิภาพ (Performance Testing) การทดสอบนี้วัดความเร็วในการทำงานและการใช้ทรัพยากรของระบบ

1. ความเร็วในการประมวลผล (Detection Speed) ทดสอบโดยการประมวลผล 500 ข้อความผสมกัน (ข้อความปกติและคำหยาบ) เพื่อหาค่าเฉลี่ย:

- Total Messages: 500
- Total Time: 0.154 วินาที (ผลลัพธ์จากการรันเทสต์จริง)
- Average Time: 0.308 ms/message (เวลาเฉลี่ยต่อข้อความ)
- Messages/Second: ~3,247 (ความสามารถในการประมวลผลต่อวินาที)

2. การใช้หน่วยความจำ (Memory Usage) ทดสอบโดยจำลองการเก็บข้อมูลประวัติการแชท 200 ข้อความ (ตามที่จำกัด deque(maxlen=200) ไว้ในโค้ด):

- Messages Stored: 200
- Memory Usage: ~38.23 KB
- Memory per Message: ~0.191 KB (หรือ 191 ไบต์)

ผลการทดสอบประสิทธิภาพยืนยันว่าระบบทำงานได้รวดเร็วแบบ Real-time และใช้ทรัพยากรระบบ (RAM) ต่ำมาก

ภาคผนวก ค. ชื่อภาคผนวก

ระบบใช้ฐานข้อมูลคำหยาบแบบไฟล์ข้อความ (.txt) 2 ไฟล์ แยกตามภาษา เพื่อให้ง่ายต่อการจัดการและอัปเดตโดยผู้ใช้งานผ่านฟังก์ชัน "จัดการคำหยาบ"

ค.1 ฐานข้อมูลคำหยาบภาษาไทย (badwords.txt) ตัวอย่างส่วนหนึ่งของไฟล์ badwords.txt ที่ใช้ในการจับคู่แบบ Substring Matching:
สัด
สันดาน
สันตีน
หน้าควย
หน้าด้าน
หน้าตัวเมีย
หน้าสันตีน

ค.2 ฐานข้อมูลคำหยาบภาษาอังกฤษ (badwords_en.txt) ตัวอย่างส่วนหนึ่งของไฟล์ badwords_en.txt ซึ่งใช้ร่วมกับไลบรารี wordsegment เพื่อตรวจจับคำที่พิมพ์ติดกัน:
2 girls 1 cup
2g1c
4r5e
5h1t
5hit
5ht
@$$
a s s
a s shole
a55
a55hole
a_s_s
abbo
abeed
abuse