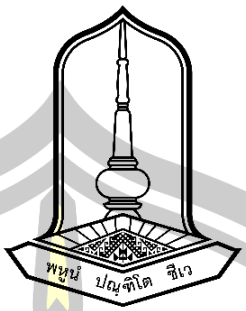


การจำแนกโรคซึมเศร้าจากทวิตเตอร์โดยขั้นตอนวิธีการเรียนรู้เชิงลึก

วิทยานิพนธ์

ของ

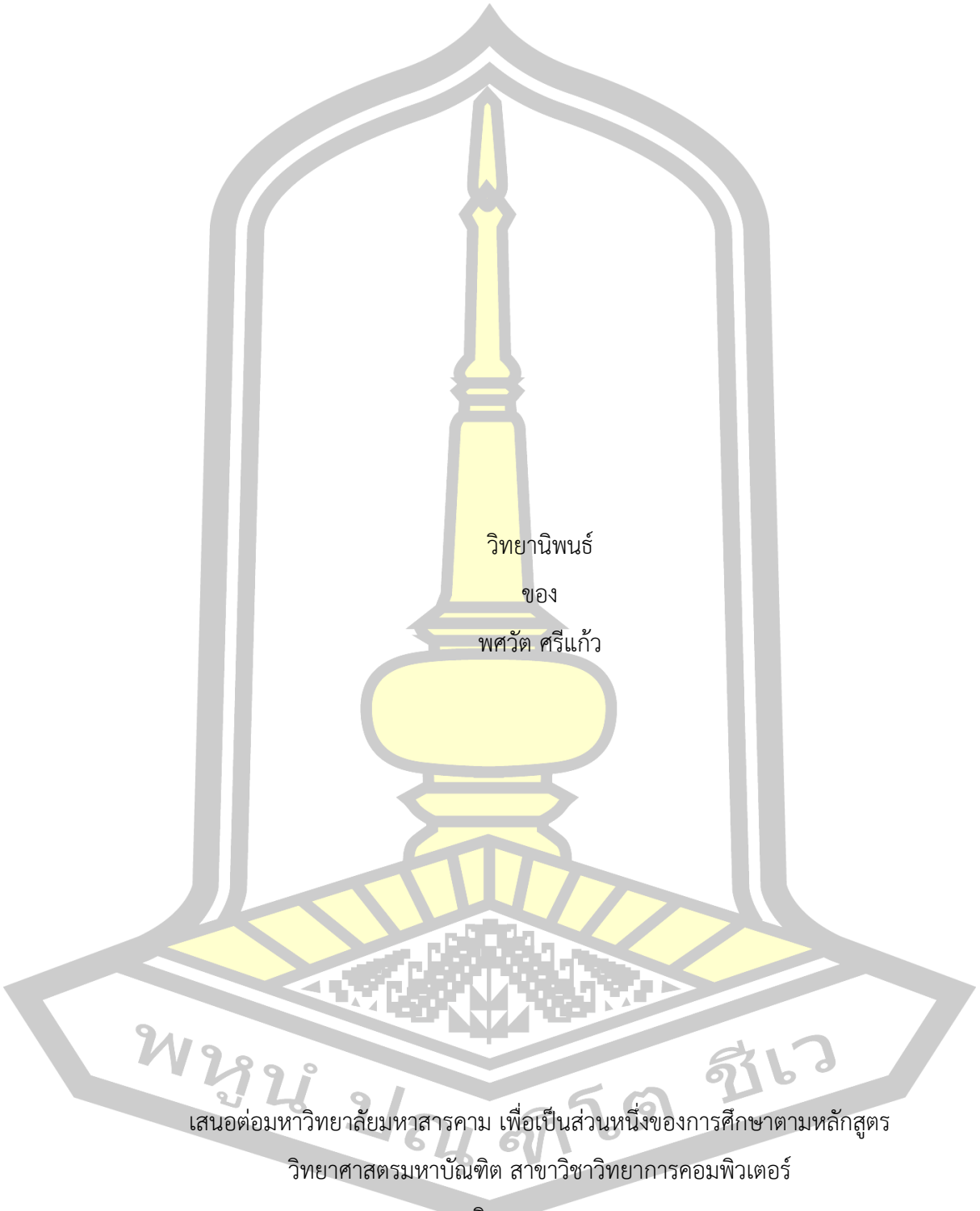
พศวัต ศรีแก้ว

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

มิถุนายน 2568

การจำแนกโรคซึมเศร้าจากทวิตเตอร์โดยขั้นตอนวิธีการเรียนรู้เชิงลึก



วิทยานิพนธ์

ของ

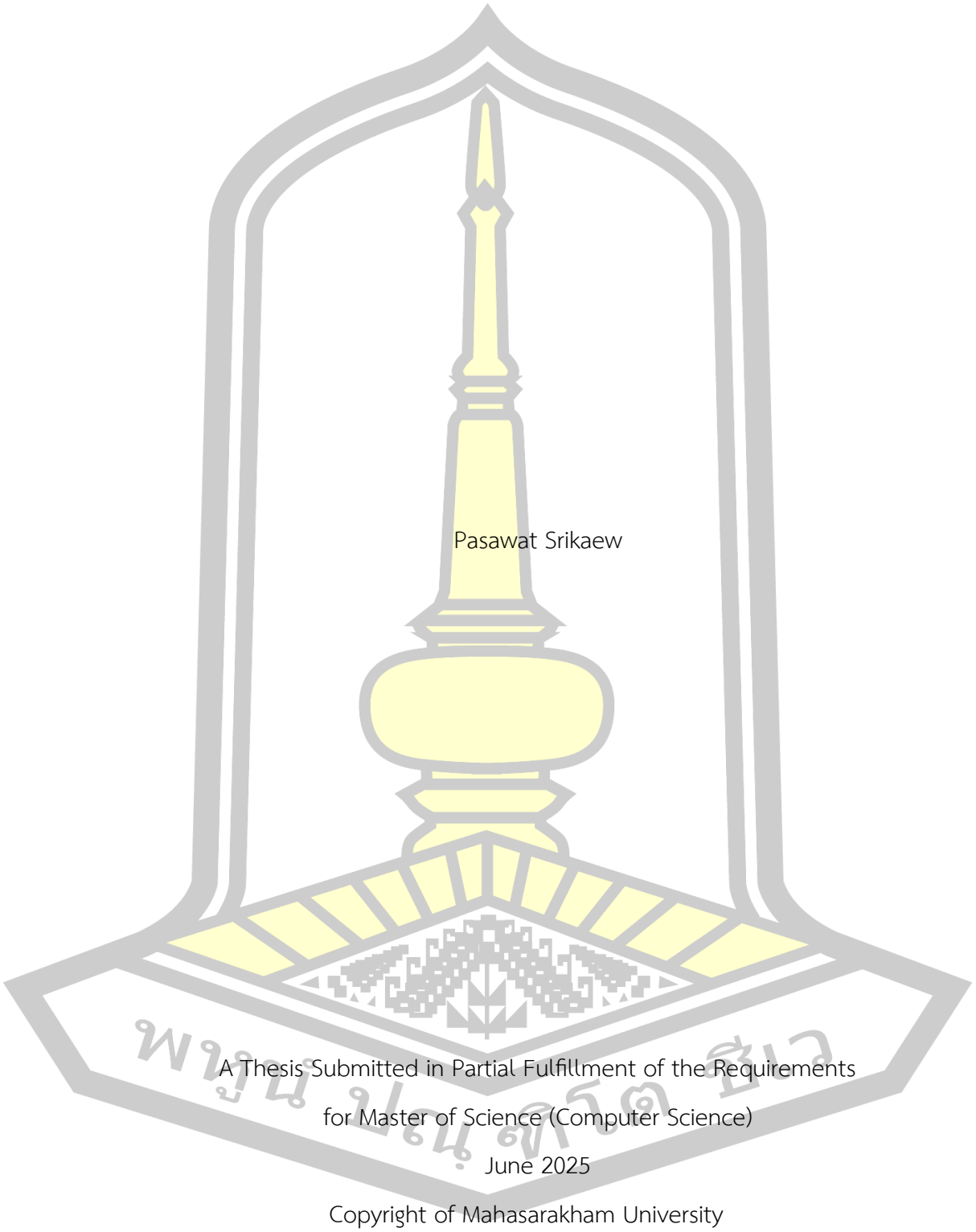
พศวัต ศรีแก้ว

เสนอต่อมหาวิทยาลัยมหาสารคาม เพื่อเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

มิถุนายน 2568

Depressive disorder classification from twitter using deep learning algorithms.



Pasawat Srikaew

A Thesis Submitted in Partial Fulfillment of the Requirements

for Master of Science (Computer Science)

June 2025

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาวิทยานิพนธ์ของ นายพศวัต ศรีแก้ว แล้วเห็นสมควรรับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(รศ. ดร.กฤษณพงศ์ สมสุข)

.....อาจารย์ที่ปรึกษา

(ผศ. ดร.ฉัตรเกล้า เจริญผล)

.....กรรมการ

(รศ. ดร.พนิดา ทรงรัมย์)

.....กรรมการ

(รศ. ดร.สุชาติ คุ้มมะณี)

มหาวิทยาลัยอนุมัติให้รับวิทยานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ของมหาวิทยาลัยมหาสารคาม

.....

(รศ. ดร.จันทิมา พลพินิจ)

คณบดีคณะวิทยาการสารสนเทศ

.....

(ผศ. ดร.พลเดช เชาวรัตน์)

คณบดีบัณฑิตวิทยาลัย

ชื่อเรื่อง การจำแนกโรคซึมเศร้าจากทวิตเตอร์โดยขั้นตอนวิธีการเรียนรู้เชิงลึก
ผู้วิจัย พศวัต ศรีแก้ว
อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.ฉัตรเกล้า เจริญผล
ปริญญา วิทยาศาสตรมหาบัณฑิต **สาขาวิชา** วิทยาการคอมพิวเตอร์
มหาวิทยาลัย มหาวิทยาลัยมหาสารคาม **ปีที่พิมพ์** 2567

บทคัดย่อ

การศึกษานี้มุ่งเน้นไปที่การจำแนกภาวะซึมเศร้าจากโพสต์บน Twitter โดยใช้เทคนิคการเรียนรู้เชิงลึก โดยเฉพาะโมเดลการเรียนรู้เชิงลึก (Deep Learning Model) ภาวะซึมเศร้าเป็นปัญหาสำคัญที่ส่งผลกระทบต่อทั้งบุคคลและสังคม ซึ่งอาจนำไปสู่ผลลัพธ์ที่ร้ายแรง รวมถึงอัตราการเสียชีวิต การศึกษานี้ใช้ข้อมูลจากโซเชียลมีเดียเพื่อระบุสัญญาณของภาวะซึมเศร้า นำเสนอแนวทางทางเทคโนโลยีเพื่อทำความเข้าใจและคาดการณ์พฤติกรรมซึมเศร้า โดยรวบรวมข้อมูลข้อความภาษาอังกฤษ รวมถึงทวีตที่ติดแฮชแท็กซึ่งบ่งบอกถึงอาการซึมเศร้า และจำแนกข้อมูลออกเป็น 9 ประเภทของอาการซึมเศร้า ระบบการจำแนกประเภทที่นำเสนอมีเป้าหมายเพื่อสนับสนุนความพยายามในการระบุและให้การช่วยเหลือเบื้องต้นแก่บุคคลที่เสี่ยงต่อภาวะซึมเศร้า

คำสำคัญ: การเรียนรู้เชิงลึก, การจำแนกภาวะซึมเศร้า, ทรานส์ฟอร์เมอร์

TITLE Depressive disorder classification from twitter using deep learning algorithms.

AUTHOR Pasawat Srikaew

ADVISORS Assistant Professor Chatklaw Jareanpon, Ph.D.

DEGREE Master of Science **MAJOR** Computer Science

UNIVERSITY Mahasarakham University **YEAR** 2024

ABSTRACT

This study focuses on the classification of depression from Twitter posts using deep learning techniques, particularly deep learning models. Depression is a significant issue that affects both individuals and society and can lead to severe outcomes, including increased mortality rates. This research utilizes data from social media to detect signs of depression and proposes a technological approach to understand and predict depressive behavior. The study collects English-language text data, including tweets containing hashtags that indicate symptoms of depression, and classifies them into nine categories of depressive symptoms. The proposed classification system aims to support efforts in early detection and intervention for individuals at risk of depression.

Keywords: Deep learning, Depressive Classification, Tranformer

กิตติกรรมประกาศ

การดำเนินการในครั้งนี้สำเร็จได้ด้วยความช่วยเหลือจากหลาย ๆ ท่านที่อาจจะกล่าวได้ไม่หมดโดยท่านแรกคือ ท่านอาจารย์ ผศ. ดร.ฉัตรเกล้า เจริญผล ที่ได้ให้คำแนะนำและ ช่วยเหลือตรวจทานในหลาย ๆ ขั้นตอน และให้ความช่วยเหลือในด้านอื่น ๆ อีกด้วย ทางผู้จัดทำขอกล่าวขอบพระคุณอย่างสูง รวมถึงทั้งคณะกรรมการที่ได้ให้คำแนะนำและให้แก้ไขให้ดียิ่งขึ้น

สุดท้ายนี้ทางผู้จัดทำขอขอบพระคุณทุกท่านที่อาจจะไม่ได้กล่าวถึงมา ณ ที่นี้ด้วย

พศวัต ศรีแก้ว

สารบัญ

หน้า

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ปัญหาโรคซึมเศร้านับว่าเป็นปัญหาที่ส่งผลทั้งทางตรงและทางอ้อม ก่อให้เกิดผลกระทบในหลายรูปแบบและอาจส่งผลร้ายแรงถึงขั้นเสียชีวิต ซึ่งส่งผลเสียต่อทั้งตนเองและบุคคลรอบข้างรวมถึงสังคมด้วย จากผลการวิจัยของ Tonsai Kaewsawang [1] กล่าวไว้ว่าการเกิดโรคซึมเศร้าจะเริ่มต้นจากการเกิดความเศร้าแบบปกติซึ่งเป็นอาการที่เกิดขึ้นจากอารมณ์ด้านลบที่เกิดจากเหตุการณ์ต่าง ๆ ที่เป็นผลกระทบทางด้านที่ไม่ดีต่อบุคคล โดยจะเกิดเป็นครั้งคราวและบรรเทาลงตามระยะเวลา อย่างไรก็ตามกรณีที่เกิดภาวะที่รุนแรงขึ้นมาจะเรียกว่าภาวะซึมเศร้า โดยจะต่างกับความเศร้าแบบปกติตรงที่เมื่อมีผลกระทบด้านลบเกิดขึ้นจะเกิดเป็นเวลานานและบางครั้งก็ไม่สามารถบรรเทาลงได้ โดยจะมีผลกระทบเกิดขึ้น 3 ด้านหลัก คือ 1.ด้านสภาพร่างกาย เช่น มีอาการนอนไม่หลับ หรือเกิดอาการเบื่ออาหาร เป็นต้น 2.ด้านสภาวะทางอารมณ์ โดยจะมีอาการ เศร้าตลอดเวลา ไร้ชีวิตชีวา เกิดความเบื่อหน่าย รู้สึกไร้ค่า 3.ด้านสังคม ยกตัวอย่างเช่น เกิดการเก็บตัวไม่อยากออกไปพบปะผู้คน เป็นต้น ซึ่งทำให้เกิดเป็นโรคซึมเศร้าที่จะมีสภาวะซึมเศร้าเป็นอาการหลักประกอบกับอาการอื่น ๆ ร่วมด้วย โดยในสังคมทั่วไปของประเทศไทยการที่คนทั่วไปจะไปปรึกษาจิตแพทย์นั้นเป็นไปได้ค่อนข้างยากเนื่องจากค่านิยมที่เกิดขึ้นมาตั้งแต่สมัยก่อน ที่มักจะคิดว่าคนที่ไปรักษาอาการทางจิตเวชว่าคือคนที่ผิดปกติทางจิต ทำให้ปัญหาทางด้านโรคซึมเศร้านั้นได้รับการแก้ไขปัญหาแบบถูกวิธีจะเกิดขึ้นน้อยกว่าที่ควรจะเป็น

ซึ่งในปัจจุบันได้มีวิธีและกระบวนการต่าง ๆ มากมายเกิดขึ้นในการช่วยตรวจสอบและเฝ้าระวังการเกิดปัญหาโรคซึมเศร้าโดยได้มีงานวิจัย ที่นำมาใช้ในการตรวจจับสังเกตพฤติกรรมของบุคคลในโลกโซเชียล ด้วยวิธีการใช้การทำเหมืองข้อมูลเพื่อช่วยในการตรวจจับวิเคราะห์[2] โอกาสของบุคคลที่จะสามารถเกิดโรคซึมเศร้า ดังเช่นที่ งานวิจัยของดำรงเดช เดินรีบรัมย์ [3] ได้ใช้เทคนิคการทำเหมืองความคิดเห็น โดยใช้อัลกอริทึม Bayes ในการพัฒนาแบบจำลองและจำแนกวิเคราะห์โอกาสที่บุคคลที่ทำการโพสต์ข้อความต่าง ๆ ลงบนโลกออนไลน์จะเกิดเป็นโรคซึมเศร้า และในงานวิจัยของวงษ์ปัญญา นวนแก้ว [4] ได้ประยุกต์ใช้วิธีการเอ็นเซมเบิลในการปรับปรุงค่าน้ำหนักให้เหมาะสมเพื่อช่วยเพิ่มความสามารถในการทำนาย โดยที่ Wongpanya Nuankaew ได้ใช้ตัวจำแนกที่นิยมมาทำเป็น

แบบจำลอง โดยใช้ 7 ตัวจำแนก ซึ่งได้นำมาใช้เพื่อกำหนดค่าน้ำหนักในการทำนายคลาสคำตอบร่วมกับ Binary term occurrences มาใช้ในการสกัดคุณลักษณะ และใช้วิธีการคัดเลือกคุณลักษณะด้วยวิธีการ Information gain เพื่อหาคุณลักษณะที่เหมาะสม และช่วยในการลดเวลาการประมวลผลที่มีประสิทธิภาพ โดยใช้ข้อมูลการแสดงความคิดเห็นจากการใช้สื่อสังคมออนไลน์(Social Media)

ดังนั้นในงานวิจัยนี้จะใช้วิธีการจำแนกประเภทด้วยวิธีการ BERT มาทำการจัดหมวดหมู่และจำแนกโอกาสที่บุคคลจะเกิดการเป็นโรคซึมเศร้าได้ โดยที่มีสมมุติฐานว่าจะสามารถเพิ่มความถูกต้องของข้อมูลและเพิ่มความแม่นยำให้มากขึ้นและจะสามารถนำงานวิจัยนี้ไปช่วยคัดแยกและทำการช่วยเหลือบุคคลที่จะสามารถเกิดเป็นโรคซึมเศร้าได้ในอนาคต

1.2 ปัญหาวิจัย

การพัฒนาแบบจำลอง Tranformer จะสามารถทำให้ประสิทธิภาพของการจำแนกการเกิดโรคซึมเศร้าจากการโพสต์บนสื่อสังคมออนไลน์สูงขึ้นหรือทำงานได้ดีขึ้นกว่าเดิมได้หรือไม่

1.3 วัตถุประสงค์ของการวิจัย

พัฒนาแบบจำลองการจำแนกโอกาสเกิดโรคซึมเศร้าจากการโพสต์ข้อความบนสื่อสังคมออนไลน์ด้วยวิธีการการเรียนรู้เชิงลึกที่จะสามารถเพิ่มประสิทธิภาพในการทำงานได้ดียิ่งขึ้น

1.4 ความสำคัญของการวิจัย

1.4.1 ข้อมูลสำหรับการสร้างแบบจำลองเป็นข้อความภาษาอังกฤษ ซึ่งได้จากการรวบรวมผ่านการติด Hashtag บน Twitter ของผู้ใช้งานทั่วไป โดยข้อความเหล่านี้ถูกแบ่งออกเป็น 9 กลุ่มตามลักษณะของอาการที่สื่อถึงโรคซึมเศร้าโดยชุดข้อมูลที่ใช้ในการศึกษามาจาก Kaggle ของ ผู้ช่วยศาสตราจารย์ ดร.ฉัตร เกล้า เจริญผล และข้อมูลอีกส่วนหนึ่งจาก ดร.วงษ์ปัญญา นวนแก้ว โดยเก็บข้อมูลมาใช้ในงานวิจัย ณ วันที่ 15 กุมภาพันธ์ 2567

1) ข้อความเกี่ยวกับอารมณ์ซึมเศร้า

2) ข้อความเกี่ยวกับการขาดความสนใจลดลง

3) ข้อความเกี่ยวกับน้ำหนักผิดปกติ

4) ข้อความเกี่ยวกับการนอนผิดปกติ

5) ข้อความเกี่ยวกับร่างกายอ่อนเพลีย

6) ข้อความเกี่ยวกับการรู้สึกตนเองไร้ค่า

7) ข้อความเกี่ยวกับสมาธิสั้น

8) ข้อความเกี่ยวกับการเคลื่อนไหลช้า

9) ข้อความเกี่ยวกับการคิดฆ่าตัวตาย

10) ข้อความที่แสดงถึงอาการปกติ

1.4.2 ข้อมูลที่ใช้ทดสอบแบบจำลองเป็นข้อความข้อมูลที่ใช้ทดสอบแบบจำลองเป็นข้อความภาษาอังกฤษ โดยนำมาจากผู้มีชื่อเสียงที่มีอาการเป็นโรคซึมเศร้าจำนวน 15 คน และนำมาจากผู้มีชื่อเสียงที่ไม่มีอาการเป็นโรคซึมเศร้าจำนวน 15 คน [2] และชุดข้อมูลที่ทำการสร้างขึ้นมาใหม่ด้วย Chat GPT (GPT-4) จำนวน 200 ข้อความ

1.4.3 การวิเคราะห์อาการของโรคซึมเศร้าตามคู่มือการวินิจฉัยและสถิติสำหรับความผิดปกติทางจิตฉบับที่ 5ของสมาคมจิตเวชศาสตร์สหรัฐอเมริกา(American Psychiatric Association) [4] ตารางที่ 1 คู่มือการวินิจฉัยและสถิติสำหรับความผิดปกติทางจิตฉบับที่ 5 ของสมาคมจิตเวชศาสตร์สหรัฐอเมริกา [4]

No	Symptoms	None	Someday	Frequently	Everyday
1	อารมณ์ซึมเศร้า	0	1	2	3
2	ความสนใจลดลง	0	1	2	3
3	น้ำหนักผิดปกติ	0	1	2	3
4	การนอนผิดปกติ	0	1	2	3
5	สมาธิสั้น	0	1	2	3
6	รู้สึกไร้ค่า	0	1	2	3
7	ร่างกายอ่อนเพลีย	0	1	2	3
8	กระวนกระวาย หรือเชื่องช้า	0	1	2	3
9	การอยากฆ่าตัวตาย	0	1	2	3

การวิเคราะห์ผู้ที่ทำแบบสอบถามเกี่ยวกับอาการในระยะเวลา 2 สัปดาห์จะใช้ระบบการกำหนดคะแนนตามความถี่ของอาการที่เกิดขึ้น โดยแบ่งเป็น 3 ระดับดังนี้

1. Someday: มีอาการในระหว่าง 2-4 วัน คะแนนเท่ากับ 1 คะแนน

2. Frequently: มีอาการในระหว่าง 6-8 วัน คะแนนเท่ากับ 2 คะแนน

3. Everyday: มีอาการในระหว่าง 10-14 วัน คะแนนเท่ากับ 3 คะแนน

จากนั้นนำคะแนนทั้ง 9 อาการมาหาผลรวมเพื่อใช้ในการประเมินโรคซึมเศร้า โดยแบ่งเป็น 4 ระดับ ดังนี้

1. ระดับที่ 1: คะแนนรวมน้อยกว่า 7 คะแนน - ปกติ

2. ระดับที่ 2: คะแนนรวม 8-12 คะแนน - มีอาการโรคซึมเศร้าน้อย

3. ระดับที่ 3: คะแนนรวม 13-18 คะแนน - มีอาการโรคซึมเศร้าปานกลาง

4. ระดับที่ 4: คะแนนรวมมากกว่า 19 คะแนน – มีอาการโรคซึมเศร้ามีอาการรุนแรง

1.4.4 การวัดประสิทธิภาพ

ในงานวิจัยฉบับนี้จะใช้การวัดประสิทธิภาพด้วยค่า

การใช้ Accuracy, F1-score, Precision, และ Recall เป็นเครื่องมือในการวัดประสิทธิภาพของโมเดลการจำแนกประเภท (classification model) มีความสำคัญอย่างมากเพราะสามารถให้ข้อมูลเกี่ยวกับประสิทธิภาพของโมเดลได้อย่างครอบคลุม

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในส่วนของบทนี้จะกล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องเกี่ยวกับโรคซึมเศร้าเพื่อนำไปประกอบในการสร้างแบบจำลอง มีรายละเอียดดังนี้

ทฤษฎีที่เกี่ยวข้อง

ส่วนของทฤษฎีที่เกี่ยวข้องได้ทำการศึกษาเกี่ยวกับอาการของโรคซึมเศร้าหลักเกณฑ์และวิธีการวินิจฉัย รวมถึงในด้านของการจัดการข้อมูล

โรคซึมเศร้า

โรคซึมเศร้าคือความผิดปกติของการหลั่งสารเคมีในสมองส่งผลให้เกิดความผิดปกติทางอารมณ์ ทำให้พฤติกรรมของผู้ป่วยเปลี่ยนไปจนส่งผลกระทบต่อการใช้ชีวิตประจำวัน กลายเป็นคนมองโลกในแง่ลบ เศร้า หม่นหมอง หดหู่ เก็บเนื้อเก็บตัว รู้สึกเบื่อหน่ายกับสิ่งที่เคยสนุกหรือสบายใจไม่มีความสุข [5]

แสดงให้เห็นว่าโรคซึมเศร้าเป็นโรคที่เกี่ยวกับทางอารมณ์และความรู้สึกทางด้านลบส่งผลให้เกิดกับร่างกายและส่งผลเสียในระยะยาว สาเหตุการเกิดโรคซึมเศร้าโรคซึมเศร้าถือเป็นโรคทางด้านจิตเวชที่พบมากเป็นอันดับต้นๆ ของเมืองไทย เกิดจาก 2 สาเหตุหลักคือ ปัจจัยทางชีวภาพหรือพันธุกรรมและปัจจัยด้านจิตใจหรือสิ่งแวดล้อม ปัจจัยทางชีวภาพหรือพันธุกรรม เนื่องจากมีการเปลี่ยนแปลงของสารสื่อประสาทในสมอง หรือความผันผวนของระดับฮอร์โมนที่สำคัญ ส่วนปัจจัยด้านจิตใจหรือสิ่งแวดล้อมที่เรียกว่าปัจจัยทางอารมณ์ เป็นผลมาจากสถานการณ์ความตึงเครียดทางอารมณ์ เช่น หากเป็นภาวะซึมเศร้าในวัยเด็กอาจมีสาเหตุจากความตึงเครียดในครอบครัว เหตุการณ์สะเทือนใจอย่างรุนแรง ภาวะซึมเศร้าในโรคประสาทซึ่งอาจพบร่องรอยว่าถูกบีบคั้นอย่างมากในวัยเด็ก แล้วปะทุออกมาในช่วงชีวิตภายหลัง ภาวะซึมเศร้าเพราะความชราเกิดเพราะความสามารถในการปรับตัวลดน้อยลง มีชีวิตโดดเดี่ยว ปัญหาช่องว่างระหว่างวัย หรือปัญหาที่เรียกกันว่าภาวะสะเทือนใจหลังเกษียณ (สูญเสียคุณค่าในตน ไม่มีงาน มีความรู้สึกว่าไร้สมรรถภาพ) [6]

ตามเกณฑ์บัญชีจำแนกทางสถิติ ระหว่างประเทศของโรคและปัญหาสุขภาพที่เกี่ยวข้อง ฉบับที่ 10 (International Classification of Diseases – 10) ในหมวด F32, F33, F34.1, F38, F39 ได้จำแนกการวินิจฉัยอาการที่เกี่ยวกับโรคซึมเศร้าเอาไว้ [1]

ตารางที่ 2 เกณฑ์การวินิจฉัย Depressive episode ตามเกณฑ์ของ ICD – 10 [1]

อาการหลัก	อาการร่วม	อาการทางกาย
1. มีอารมณ์เศร้า	1. สมาธิลดลง	1. เบื่อหน่าย ไม่สนุกสนานในกิจกรรมที่เคยเป็น
2. ความสนุกสนานเพลิดเพลินหรือ ความสนใจในกิจกรรมลดลง	2. ความมั่นใจและความภาคภูมิใจในตนเองลดลง	2. ไร้อารมณ์ต่อสิ่งรอบข้างที่เคยทำให้เพลิดเพลินใจ
3. อ่อนเปลี้ยเพลีย แรง มีกิจกรรม น้อยลง	3. รู้สึกผิดและไร้ค่า	3. ตื่นเช้ากว่าปกติ ≥ 2 ชม.
	4. มองอนาคตในทางลบ	4. อาการซึมเศร้าเป็นมากในช่วงเช้า
	5. คิดฆ่าตัวตายหรือทำร้ายตนเอง หรือฆ่าตนเอง	5. ทำอะไรช้า เคลื่อนไหวช้าลง หรือ กระสับกระส่าย
	6. มีความผิดปกติในการนอนหลับ	6. เบื่ออาหารอย่างมาก
	7. เบื่ออาหาร	7. น้ำหนักลดลง (5%ใน 1 เดือน)
		8. ความต้องการทางเพศลดลง

การประมวลผลภาษาธรรมชาติ (NLP)

เป็นเทคโนโลยีแมชชีนเลิร์นนิ่งที่ช่วยให้คอมพิวเตอร์สามารถตีความ จัดการ และทำความเข้าใจภาษามนุษย์ได้ องค์กรในปัจจุบันมีข้อมูลเสียงและข้อความจำนวนมากจากช่องทางการสื่อสารต่างๆ เช่น อีเมล ข้อความ ฟีดข่าวโซเชียลมีเดีย วิดีโอ เสียง และอื่นๆ โดยใช้ซอฟต์แวร์ NLP เพื่อประมวลผลข้อมูลนี้โดยอัตโนมัติ วิเคราะห์เจตนาหรือความเชื่อมั่นในข้อความ และตอบสนองการสื่อสารของมนุษย์แบบเรียลไทม์ [7]

แนวทางการประมวลผลข้อมูลในเชิงภาษาธรรมชาติ (Natural Language Processing - NLP) มักจะประกอบด้วยสามขั้นตอนหลัก คือ Preprocessing, Modeling), และ Results ดังนี้

1) Preprocessing:

Tokenization: การแบ่งข้อความเป็นหน่วยย่อยที่เรียกว่า Token เช่น คำ, ประโยค, หรือตัวอักษร เพื่อให้สามารถประมวลผลได้ง่ายขึ้น

การลบตัวอักษรพิเศษและเครื่องหมาย: เช่น เครื่องหมายวรรคตอน, เครื่องหมายคำพูด, ตัวอักษรพิเศษ เป็นต้น

การแยกคำ:สำหรับภาษาที่ไม่มีการเว้นวรรคชัดเจน เช่น ภาษาไทยหรือภาษาจีน

การแปลงตัวหนังสือเป็นตัวพิมพ์เล็กหรือตัวพิมพ์ใหญ่: เพื่อป้องกันการแยกแยะคำต่างกันเนื่องจากตัวหนังสือมีการใช้ตัวพิมพ์ใหญ่หรือตัวพิมพ์เล็กแตกต่างกัน

Part-of-Speech Tagging (POS Tagging): กระบวนการในการระบุและทำเครื่องหมายประเภทของคำในประโยค ซึ่งประเภทของคำเหล่านี้มักเรียกว่า "Part of Speech" (POS) เช่น คำนาม (noun), คำกริยา (verb), คำคุณศัพท์ (adjective), คำวิเศษณ์ (adverb) เป็นต้น

Lemmatization: เป็นกระบวนการแปลงคำให้อยู่ในรูปแบบพื้นฐาน (Lemma) หรือรูปแบบพจนานุกรม ซึ่งต่างจาก Stemming ที่เป็นการตัดคำเพื่อให้ได้รากคำ (Root form) โดย Lemmatization จะพิจารณารูปแบบของคำและความหมายทางไวยากรณ์ เช่น tense และ part of speech เพื่อแปลงคำให้ถูกต้องตามความหมาย

2) Modeling:

Word Embeddings (การฝังคำ): การแปลงคำให้อยู่ในรูปแบบเวกเตอร์ที่มีความหมาย ซึ่งช่วยเพิ่มประสิทธิภาพในการจัดการคำศัพท์และความหมายของประโยค

โมเดลการจำแนกประเภท (Classification Models): โมเดลที่ใช้ในการจำแนกประเภทข้อความออกเป็นหมวดหมู่หรือประเภทต่าง ๆ เช่น การวิเคราะห์ทฤษฎีบท, การจำแนกหมวดหมู่ข่าว, หรือการจำแนกอารมณ์ (sentiment analysis)

โมเดลการแปลภาษา (Translation Models): โมเดลที่ใช้ในการแปลภาษาจากภาษาหนึ่งไปยังอีกภาษาหนึ่ง เช่น Google Translate, โมเดลการแปลข้อความเฉพาะเรื่อง

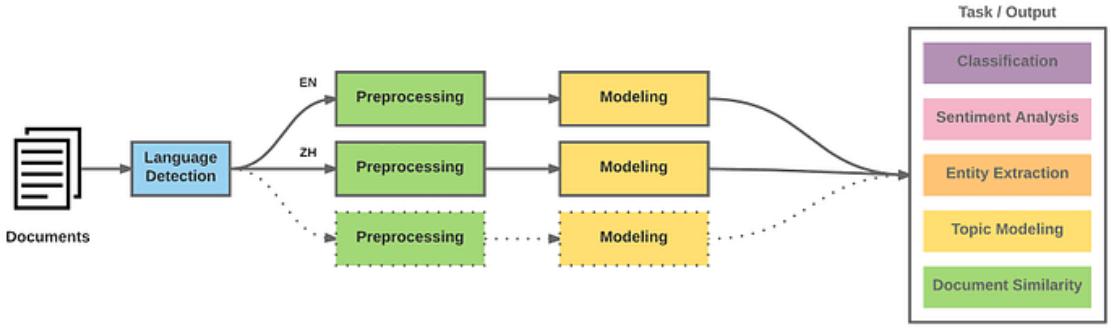
โมเดลการสร้างประโยค (Text Generation Models): โมเดลที่ใช้สร้างประโยคหรือข้อความใหม่โดยอ้างอิงจากข้อมูลเรียนรู้ เช่น โมเดลที่ใช้สร้างข้อความโฆษณาหรือบทความ

3) Results:

การประเมินผลประสิทธิภาพของโมเดล (Model Evaluation): การใช้ชุดข้อมูลทดสอบเพื่อวัดประสิทธิภาพของโมเดล โดยใช้เมตริกที่เหมาะสม เช่น Accuracy, F1-score, Precision, Recall

การนำไปใช้ (Application): การนำโมเดลที่ได้ผลลัพธ์ดีไปใช้ในงานจริง เช่น การปรับปรุงการบริการลูกค้าออนไลน์, การจัดการข้อมูลสื่อสารภายในองค์กร, หรือการพัฒนาแอปพลิเคชันสนับสนุนภาษาธรรมชาติ

การประมวลผลข้อมูลในเชิง NLP มีความสำคัญในการทำให้คอมพิวเตอร์เข้าใจและประมวลผลข้อมูลที่เป็นภาษาธรรมชาติได้อย่างมีประสิทธิภาพและแม่นยำ ซึ่งมีการนำไปใช้ในหลากหลายด้าน เช่น การวิเคราะห์ข้อมูลทางการแพทย์, การวิเคราะห์เนื้อหาในสื่อสังคมออนไลน์, หรือการพัฒนาเทคโนโลยีด้านการแปลภาษาและการสื่อสารระหว่างมนุษย์กับเครื่อง

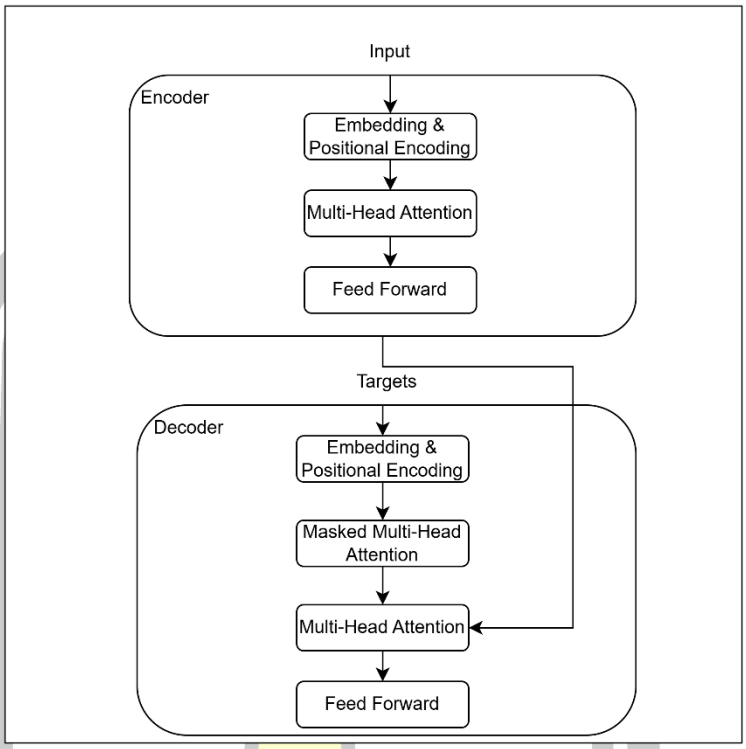


ภาพที่ 1 ขั้นตอนการทำงานของ NLP [15]

การเรียนรู้เชิงลึกแบบทรานส์ฟอร์เมอร์(Transformer)

คือโครงสร้างแบบพิเศษที่ถูกพัฒนาขึ้นสำหรับการประมวลผลภาษาธรรมชาติ โมเดล Transformer ได้รับความนิยมมากในงานประมวลผลภาษาธรรมชาติ เช่น text classification, machine translation, summarization, และการตอบคำถาม เนื่องจากความสามารถในการจัดการกับข้อมูลที่มีลำดับและมีความเกี่ยวเนื่องกัน

ขั้นตอนหลักใน Transformer ประกอบด้วย:



ภาพที่ 2 ขั้นตอนการทำงานของ Tranformer

Tokenization: ใช้เพื่อช่วยแบ่งประโยคเป็น token และแปลงให้เป็นรหัส (token IDs) การแบ่งประโยคเป็น Token Tokenization ช่วยแบ่งประโยคเป็นหน่วยที่เล็กลงที่เรียกว่า "token" เนื้อหาข้อความจะถูกแบ่งเป็นคำหรือส่วนย่อยที่เรียกว่า token ซึ่งเป็นหน่วยที่นำไปใช้ในการเรียนรู้ของโมเดลแปลง Token เป็นรหัส (Token IDs) หลังจากที่ได้ token แล้ว จะแปลงแต่ละ token เป็นรหัสที่ใช้ในการให้โมเดลเรียนรู้ ช่วยให้โมเดลสามารถทำนายหรือเรียนรู้จากข้อมูลทางภาษาได้ การจัดการกับข้อความที่ซับซ้อนในภาษาธรรมชาติมีข้อความที่ซับซ้อน ประกอบไปด้วยคำที่ต่อกันไปเรื่อย ๆ โดยไม่มีตัวเว้นวรรคหรือตัวเว้นวรรคเพียงพอ Tokenization ช่วยให้สามารถจัดการกับข้อความที่ซับซ้อนนี้ได้ง่ายขึ้นการจัดการกับคำพวกเครื่องหมายวรรคตอนและตัวอักษรพิเศษ Tokenization ช่วยในการแยกคำพวกเครื่องหมายวรรคตอนและตัวอักษรพิเศษออกจากข้อความ เพื่อให้โมเดลเห็นภาพของข้อความที่สะอาด ป้องกัน Overflow การใช้ Tokenization ช่วยลดโอกาสที่ข้อมูลจะเกินความจุที่โมเดลรองรับ. โดยทั่วไปข้อมูลที่มีขนาดใหญ่จะถูกตัดเป็นส่วนๆ (tokens) เพื่อป้องกันปัญหาที่เกี่ยวกับขนาดของข้อมูลที่เข้ามา

Embedding Layer: หน้าที่หลักๆ คือการแปลงข้อมูลที่เป็น discrete (ไม่ต่อเนื่อง) เช่น คำศัพท์หรือ token เป็น vector ที่มีความหลากหลายและมีความหมาย Embedding Layer

ช่วยในเรื่องต่างๆ เช่นการแปลงคำหรือ Token เป็น Vector การใช้ Embedding Layer ทำให้สามารถแปลงคำหรือ token ในภาษาธรรมชาติเป็น vector ที่มีความหลากหลายและมีความหมาย. นี้ช่วยให้โมเดลเรียนรู้ความสัมพันธ์และลักษณะของคำในบริบท ความสามารถในการเรียนรู้ความหมาย Embedding Layer ช่วยโมเดลเรียนรู้ความหมายของคำต่างๆ โดยการพจน์ความสัมพันธ์ในพื้นที่ vector space คำที่มีความหมายคล้ายกันจะมี vector ที่ใกล้เคียงกันในพื้นที่นี้การลดมิติ Embedding Layer ส่วนใหญ่ทำให้มีการลดมิติของข้อมูลจากข้อมูล discrete ที่มีขนาดใหญ่ไปเป็น vector ที่มีขนาดเล็กกว่า. นี้ช่วยลดการซับซ้อนในการประมวลผลข้อมูล ปรับปรุงความสามารถในการฝึกฝน Embedding Layer ช่วยลดข้อจำกัดของการใช้ข้อมูลที่เป็น discrete ในการฝึกฝนโมเดล โดยที่การใช้ vector ที่ต่อเนื่องช่วยให้ Gradient Descent ทำงานได้ต่อเนื่อง Embedding Layer เป็นส่วนสำคัญที่มีบทบาทในการเปลี่ยนข้อมูล discrete ใน Deep Learning เป็น vector ที่มีความหมายและมีความหลากหลาย

Self-Attention Mechanism: Transformer ใช้การกำหนดน้ำหนักในการสร้างรูปแบบของความสัมพันธ์ระหว่างคำศัพท์ที่ต่างกันในประโยค ทำให้สามารถรับรู้ความเกี่ยวเนื่องกันได้.

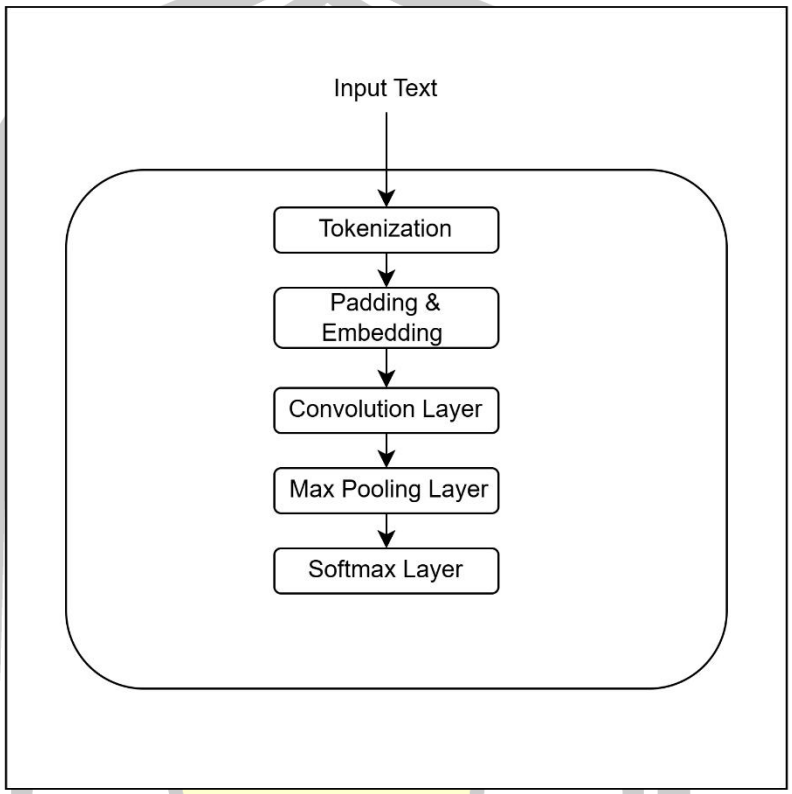
Multi-Head Attention: Self-attention mechanism ถูกขยายต่อเนื่องในหลายๆ ตัว โดยที่แต่ละตัวมีการเรียนรู้และค้นหาลักษณะต่างๆ ของข้อมูล จากนั้นเชื่อมต่อผลลัพธ์จากทุกตัวเข้าด้วยกัน.

Positional Encoding: เนื่องจาก Transformer ไม่รับรู้ลำดับของข้อมูล ดังนั้นต้องมีการเพิ่มข้อมูลลำดับ (positional information) เข้าไปในข้อมูล ซึ่งจะทำให้ Transformer สามารถจับความเกี่ยวเนื่องกันได้.

Feedforward Neural Networks: หลังจากขั้นตอน self-attention แล้ว, Transformer ยังมี layer ของ feedforward neural networks ที่ทำหน้าที่ในการประมวลผลข้อมูลในแต่ละตำแหน่ง.

ในงาน text classification, Transformer สามารถนำมาใช้โดยการให้ข้อมูลข้อความเป็น input และตัวแบบนี้สามารถเรียนรู้ความสัมพันธ์ระหว่างคำและทำนายหมวดหมู่ของข้อความได้. การใช้ Transformer ในงาน text classification มีประสิทธิภาพสูงและสามารถเรียนรู้จากข้อมูลทั่วไปได้ดี เนื่องจากความสามารถในการจับความเกี่ยวเนื่องกันและการทำงานที่เร็ว

การเรียนรู้เชิงลึกแบบโครงข่ายประสาทเทียมแบบคอนโวลูชัน CNN (Convolutional Neural Network)



ภาพที่ 3 ขั้นตอนการทำงานของ CNN

1D Convolutional Layer: คอนโวลูชัน CNN ใช้ 1D Convolution แทนที่จะเป็น 2D เพราะข้อความเป็นลำดับเชิงเส้น ฟิลเตอร์ (Kernel) ใน CNN ทำหน้าที่เหมือนตัวดึงลักษณะเด่น (features) ของกลุ่มคำ เช่น ฟิลเตอร์ขนาด 2 อาจจับคู่ bi-grams ฟิลเตอร์ขนาด 3 อาจจับคู่ tri-grams ฟิลเตอร์หลายตัวช่วยให้ CNN สามารถเข้าใจรูปแบบของข้อความที่ซับซ้อนได้ดี

Stride: ควบคุมว่าฟิลเตอร์จะเลื่อนไปตามข้อมูลทีละกี่ตำแหน่ง

Padding: ช่วยให้ฟีเจอร์แมพมีขนาดที่เหมาะสม (เช่น same padding หรือ valid padding)

ใช้ ReLU (Rectified Linear Unit) เพื่อทำให้โมเดลสามารถเรียนรู้ฟีเจอร์ที่ซับซ้อนได้ดีขึ้น ช่วยป้องกันปัญหา vanishing gradient ที่เกิดจาก sigmoid หรือ tanh

Pooling Layer: การลดขนาดข้อมูล หลังจากการใช้ Convolution แล้ว ผลลัพธ์จะมีขนาดใหญ่มาก

Pooling Layer ช่วยลดขนาดข้อมูลลง ทำให้โมเดลเรียนรู้ได้เร็วขึ้น และลด overfitting

Global Poolingแทนที่การใช้ Fully Connected Layer ด้วย Global Max Pooling เพื่อลดจำนวนพารามิเตอร์

Fully Connected Layer (FC Layer)

แปลงข้อมูลที่ถูกลดขนาดแล้วให้เป็นเวกเตอร์เพื่อใช้ในการจำแนกประเภท

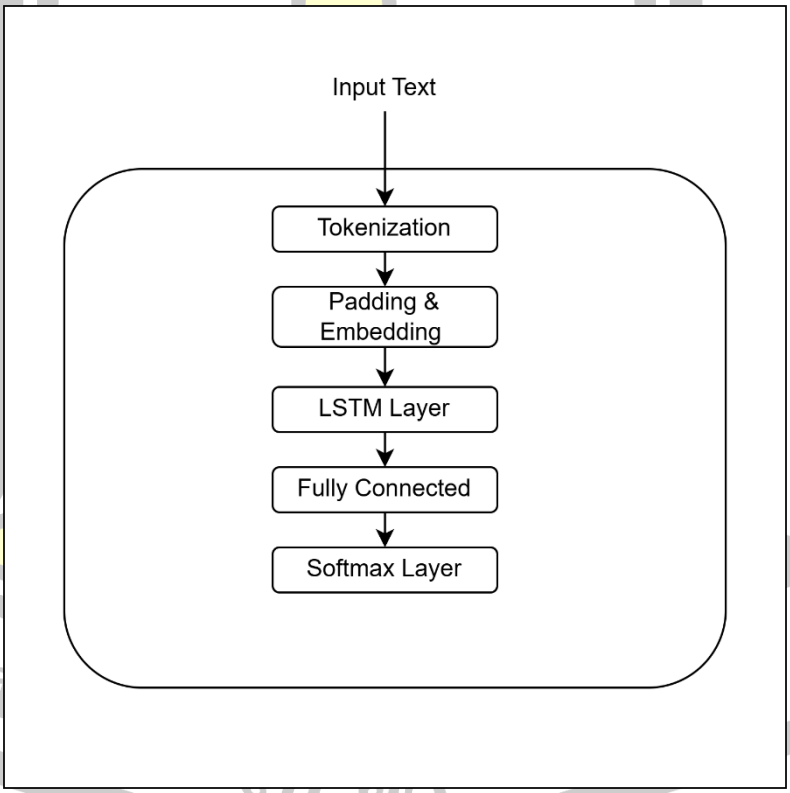
ใช้ Dropout เพื่อลด overfitting

สามารถเพิ่ม Batch Normalization เพื่อช่วยให้โมเดลเรียนรู้ได้เร็วขึ้น

ฟังก์ชัน Output Layer (Softmax) เป็นฟังก์ชันที่ใช้ในการแปลงค่าคะแนน (logits) ให้อยู่ในช่วง 0 ถึง 1 และผลรวมของค่าทั้งหมดจะเท่ากับ 1 ซึ่งทำให้สามารถใช้เป็นความน่าจะเป็นของแต่ละคลาสในปัญหาการจำแนกประเภท (classification) ได้

Softmax Activation Function: ใช้สำหรับ Multi-class Classification (เช่นวิเคราะห์อารมณ์เป็น 3 หรือ 5 ระดับ)

การเรียนรู้เชิงลึกแบบหน่วยความจำระยะยาว-สั้น LSTM (Long Short-Term Memory)



ภาพที่ 4 ขั้นตอนการทำงานของ LSTM

LSTM ใช้โครงสร้างที่มี 3 Gates เพื่อควบคุมการไหลของข้อมูลภายในเครือข่าย ได้แก่:

Forget Gate (ลืมข้อมูลที่ไม่สำคัญ) ตัดสินใจว่าจะลบข้อมูลเก่าออกจากเซลล์สเตทหรือไม่ โดยใช้ค่าจากอินพุตและสถานะก่อนหน้า

Input Gate (อัปเดตข้อมูลใหม่) ตัดสินใจว่าข้อมูลใหม่ควรถูกเพิ่มเข้าไปในเซลล์สเตทหรือไม่

Output Gate (กำหนดค่าที่จะส่งออก) ควบคุมว่าข้อมูลใดควรนำมาใช้เพื่อสร้างสถานะใหม่

ขั้นตอนการทำงานหลัก

รับข้อมูลข้อความ ⟶ แปลงเป็นเวกเตอร์ เช่น Word2Vec หรือใช้ Embedding Layer

ผ่านข้อมูลเข้า LSTM layer ⟶ จัดการข้อมูลทีละ timestep (คำในประโยค)

LSTM คำนวณค่าผ่าน Forget, Input และ Output Gate

ได้ Hidden State สุดท้าย (h_t) ⟶ นำไปใช้สำหรับ classification

ผ่าน Fully Connected Layer (Dense Layer) และ Softmax ⟶ ทำนาย class

การวัดประสิทธิภาพ

1. Accuracy

- คำนวณจากจำนวนตัวอย่างที่ถูกต้องทั้งหมด (True Positives + True Negatives) หารด้วยจำนวนทั้งหมดของตัวอย่าง (True Positives + False Positives + True Negatives + False Negatives)

- ช่วยในการวัดประสิทธิภาพโดยรวมของโมเดล แต่อาจไม่เหมาะสมในกรณีที่ความสมดุลของคลาสไม่เท่ากัน

2. Validation Accuracy

- คือค่าความถูกต้องของโมเดลเมื่อทดสอบกับชุดข้อมูล validation ซึ่งเป็นข้อมูลที่แยกไว้ต่างหากจากข้อมูลที่ใช้ในการฝึก (training data) ค่านี้จะบอกได้ว่าโมเดลสามารถทำนายข้อมูลใหม่ที่ไม่เคยเห็นได้ถูกต้องแค่ไหน โดยแสดงผลในรูปของเปอร์เซ็นต์

3. Loss Function

- คือค่าที่ใช้วัดความแตกต่างระหว่างค่าที่โมเดลคาดการณ์ (predicted values) กับค่าจริง (actual values) จากข้อมูลที่เรามี โดยการคำนวณค่านี้จะบอกได้ว่าโมเดลนั้นทำนายได้ดีแค่ไหน ค่า loss ที่น้อยแสดงว่าโมเดลมีการทำนายที่ใกล้เคียงกับค่าจริงมากขึ้น ส่วนค่า loss ที่มากแสดงว่าโมเดลทำนายได้ไม่แม่นยำเท่าที่ควร

4. Validation Loss

- คือค่าการสูญเสีย (loss) ที่คำนวณจากชุดข้อมูล validation ซึ่งเป็นข้อมูลที่ไม่ถูกใช้ในการฝึกโมเดลโดยตรง แต่จะถูกใช้เพื่อตรวจสอบว่าโมเดลที่ฝึกได้ทำงานดีแค่ไหนกับข้อมูลที่โมเดลยังไม่เคยเห็น

- เมื่อค่า Validation Loss มีแนวโน้มลดลง แสดงว่าโมเดลกำลังเรียนรู้และมีประสิทธิภาพดีขึ้นกับข้อมูลใหม่ ๆ แต่ถ้า Validation Loss เริ่มเพิ่มขึ้นหลังจากฝึกไปสักพัก อาจเป็นสัญญาณว่าโมเดลกำลัง "overfitting" หรือเรียนรู้เฉพาะข้อมูลฝึก (training data) มากเกินไป

ทฤษฎีที่เกี่ยวข้อง

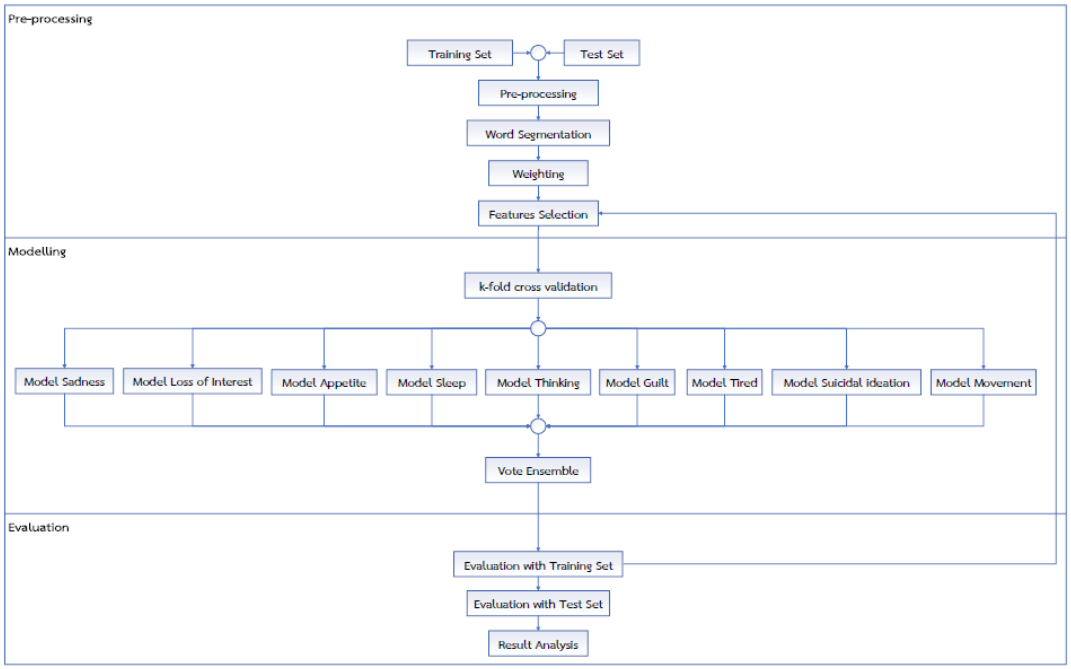
ในส่วนของงานวิจัยที่เกี่ยวข้องได้ทำการศึกษางานวิจัยที่เกี่ยวข้องกับการทำการจำแนกข้อมูล จากงานวิจัยดังนี้

งานวิจัยของดำรงเดช เดินรีบรัมย์ [2] ในงานวิจัยนี้ได้บ่งชี้ถึงปัญหาโรคซึมเศร้าซึ่งเป็นสาเหตุหลักของการเสียชีวิตก่อนวัยอันควร และเป็นสาเหตุอันดับสองของการฆ่าตัวตายในคนอายุระหว่าง 15-29 ปี ปัจจัยสำคัญที่ก่อให้เกิดโรคนี้เกิดจากความเครียดและความผิดหวังที่สะสมมานาน โดยโซเชียลมีเดีย (Social Media) มีผลต่อจิตใจของมนุษย์ผู้ใช้งาน จากปัญหานี้จึงได้นำเทคนิคการทำเหมืองความคิดเห็นมาประยุกต์ใช้งานโดยสร้างแบบจำลองตรวจสอบความคิดเห็นบนโซเชียลมีเดียเพื่อหาผู้ที่เข้าข่ายเป็นโรคซึมเศร้าเพื่อช่วยแก้ปัญหาดังแสดงในตารางที่ 3 โดยทำการกรองข้อมูลจาก Twitter โดยการลบข้อความที่เป็นการ Retweet, การลบข้อความที่เป็น URL เข้าใช้งานเว็บไซต์, การลบข้อความที่เกี่ยวข้องกับชื่อบุคคลที่ถูกแทกในโพสต์ และการลบ Stopword สำหรับการให้น้ำหนักเลือกใช้งานวิธีการ Binary term occurrence แล้วคัดเลือกแอตทริบิวต์ด้วย Information Gain ที่ทำการหาค่า Entropy

ตารางที่ 3 ตัวอย่างการให้น้ำหนักแอตทริบิวต์ด้วยวิธี Information Gain

Attribute	Weight
depression	1
back	0.384
best	0.382

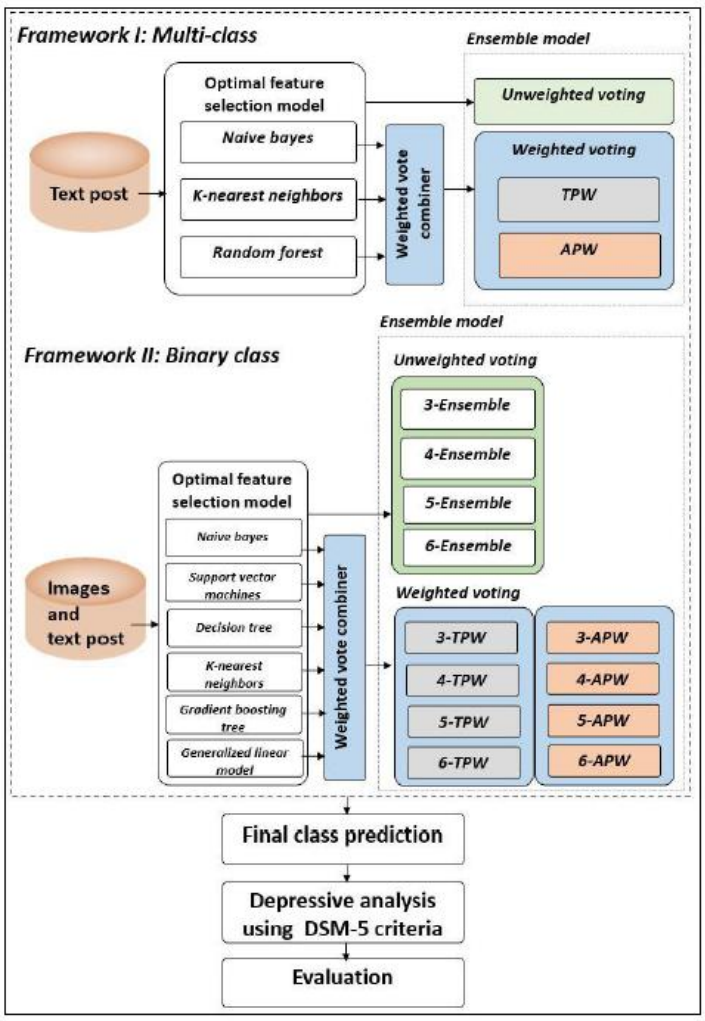
ที่กำหนด Top k = 2,000 4,000 6,000 และเลือกคุณลักษณะทั้งหมด ในการสร้างแบบจำลองผู้วิจัยเลือกใช้การแบ่งข้อมูลโดยใช้วิธีการ 10 – Fold Cross Validation โดยผู้วิจัยเลือกใช้การสร้างแบบจำลองตามลักษณะอารมณ์ที่เกี่ยวข้องกับโรคซึมเศร้า โดยแบบจำลอง 1 แบบจำลองมีคำตอบเป็น เข้าข่ายเป็นอาการ (Yes) และ ไม่เข้าข่ายเป็นอาการ (No)ซึ่งใช้งานอัลกอริทึม Bayes ในการสร้างแบบจำลอง Information Gain ในการคัดเลือกคุณลักษณะ โดยกำหนด Top-K คือ 2,000 4,000 6,000 และเลือกคุณลักษณะทั้งหมด โดยมีการกำหนด Boundary เพื่อหาช่วงการคัดเลือกค่าความน่าจะเป็นที่เหมาะสมโดยผลลัพธ์ที่ได้ปรากฏว่าช่วงค่าความน่าจะเป็นที่เหมาะสมอยู่ที่ 80 – 90 ได้ผลลัพธ์ที่ดีที่สุด คือ Accuracy = 80.00%, Precision Yes = 100.00%, Precision No = 71.43%, Recall Yes = 60.00%, Recall No = 100.00% และ F-1 = 75.00% จากการวิเคราะห์ผลการทดลองทำให้ทราบว่า การคัดเลือกแอตทริบิวต์ด้วย Information Gain มีผลทำให้เวลาในการสร้างแบบจำลองลดน้อยลง และการให้น้ำหนักแบบ Binary term occurrence เหมาะสำหรับการทำนายข้อความที่มีลักษณะสั้น เนื่องจากข้อความที่ได้หลังจากการกรอกข้อมูลมีคุณลักษณะสำคัญเพียงไม่กี่คำ



ภาพที่ 5 ขั้นตอนการดำเนินงานวิจัยของ ดำรงเดช เดินรีบรัมย์[2]

งานวิจัยของวงษ์ปัญญา นวนแก้ว [4] ได้กล่าวถึงโรคซึมเศร้าเป็นโรคทางจิตเวชที่เกิดจากการไม่สมดุลของสารเคมีในสมอง ซึ่งส่งผลต่ออารมณ์ ความคิด และความรู้สึก ผู้ป่วยโรคซึมเศร้าจะแสดงอาการต่างกันขึ้นอยู่กับความรุนแรง ตั้งแต่การทำร้ายตนเองไปจนถึงการฆ่าตัวตาย โดยเฉพาะผู้ป่วยที่อายุ

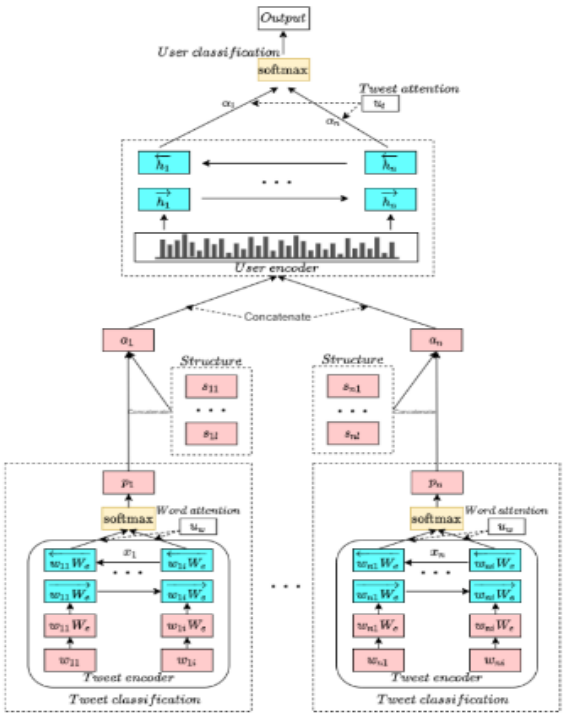
น้อยและผู้ป่วยที่อยู่ในช่วงวัยทำงานมีความเสี่ยงในการฆ่าตัวตายมากขึ้น การระบุโรคซึมเศร้าในช่วงเริ่มแรกจำเป็นมาก เพื่อให้ได้การดูแลและการรักษาที่เหมาะสมและทันเวลา หลายๆ งานวิจัยได้ใช้ข้อมูลโพสต์จากโซเชียลมีเดียเพื่อหาข้อบ่งชี้และตีความอารมณ์ ความรู้สึก และความคิดของผู้ใช้โซเชียลมีเดียแต่ละคน ซึ่งส่งผลให้เกิดงานวิจัยนี้ที่ได้เสนอวิธีการเอ็นเซมเบิลในการปรับปรุงค่าน้ำหนักให้เหมาะสมเพื่อช่วยเพิ่มความสามารถในการทำนาย โดยที่ Wongpanya Nuankaew ได้ใช้ตัวจำแนกที่นิยมมาทำเป็นแบบจำลอง โดยใช้ 7 ตัวจำแนก ได้แก่ 1) นาอีฟเบย์ (Naive bayes: NB) 2) ซัพพอร์ตเวกเตอร์แมชชีน (Support vector machines: SVM) 3) ต้นไม้ตัดสินใจ (Decision tree: DT) 4) ขั้นตอนวิธีการเพื่อนบ้านใกล้ที่สุด (K-nearest neighbors: KNN) 5) แรนดอมฟอเรสต์ (Random forest: RF) 6)กราเดียนบูทติ้งทรี (Gradient boosting tree: GBT)และ 7) ตัวแบบเชิงเส้นนัยทั่วไป (Generalized linear model: GLMs) ซึ่งได้นำมาใช้เพื่อกำหนดค่าน้ำหนักในการทำนายคลาสคำตอบร่วมกับ Binary term occurrences มาใช้ในการสกัดคุณลักษณะ และใช้วิธีการคัดเลือกคุณลักษณะด้วยวิธีการ Information gain เพื่อหาคุณลักษณะที่เหมาะสม และช่วยในการลดเวลาการประมวลผลที่มีประสิทธิภาพ โดยใช้ข้อมูลการแสดงความคิดเห็นจากการใช้โซเชียลมีเดีย โดยวัดประสิทธิภาพของแบบจำลองด้วย 1) ตัวจำแนกประเภทแบบเดี่ยว (Single classifier) 2) วิธีการเอ็นเซมเบิลแบบไม่กำหนดค่าน้ำหนัก และ 3)วิธีการเอ็นเซมเบิลแบบกำหนดค่าน้ำหนัก จากผลการทดลองทำให้ทราบได้ว่า ประสิทธิภาพเอ็นเซมเบิลแบบกำหนดค่าน้ำหนักมีความถูกต้องมากกว่าแบบไม่กำหนดค่าน้ำหนักทั้งสองวิธีการและมีเสถียรภาพมากกว่า โดยวิธีการวิธีการค่าเฉลี่ยความน่าจะเป็นของการเกิดคลาสคำตอบ มีค่า accuracy 66.67% มี Attention score สูงสุดที่ 80.00% และค่า F1-score 70.59% และการให้ค่าน้ำหนักแก่คุณลักษณะด้วยวิธีการ Binary term occurrence คัดเลือกคุณลักษณะด้วยวิธีการ Information gain ตาม Top-k ทำให้ได้คุณลักษณะที่เหมาะสมในการสร้างแบบจำลองการเรียนรู้ที่มีประสิทธิภาพสามารถ ช่วยลดเวลาการประมวลผลได้



ภาพที่ 6 ขั้นตอนการดำเนินงานวิจัยของ วงษ์ปัญญา นวนแก้ว[4]

งานวิจัยของ Akkapon Wongkoblap [8] และคณะได้กล่าวถึงปัญหาสุขภาพจิตเป็นปัญหาสาธารณสุขที่เป็นที่รู้จักกันอย่างแพร่หลายทั่วโลก ตามข้อมูลจากองค์การอนามัยโลก มีผู้ป่วยโรคซึมเศร้าทั่วโลก 264 ล้านคนในปี 2020 โรคจิตเวชเป็นหนึ่งในสาเหตุหลักของภาระโรคระดับโลก โดยประเมินว่าในประเทศอังกฤษมีการใช้จ่าย 105 พันล้านปอนด์อังกฤษ (145 พันล้านดอลลาร์สหรัฐ) สำหรับบริการและการรักษาสุขภาพจิตหรือการสูญเสียความสามารถในการทำงานในปี 2018 และค่าใช้จ่ายทั่วโลกคาดว่าจะเพิ่มขึ้นเป็น 6 ล้านล้านดอลลาร์สหรัฐในปี 2030 จึงได้ได้ทำวิจัยที่ช่วยในการ วิเคราะห์และวินิจฉัยตัวบุคคลที่มีโอกาสเกิดการเป็นโรคซึมเศร้า ด้วยการใช้ Multipleinstance learning (MIL) เป็นแบบจำลอง และนำวิธีการ Anaphora ที่เป็นการประมวลผลภาษาธรรมชาติ (NLP) มาช่วยในการวิเคราะห์และพิจารณาว่าอาจจะมีการกล่าวถึงบุคคลอื่นเนื่องด้วยเป็นสื่อโซเชียลมีเดียจึงทำให้มีความเป็นไปได้ว่าจะจะกล่าวถึงบุคคลอื่นที่ไม่ใช่ผู้โพสต์ได้ ซึ่งได้ผลจากการใช้ MIL เป็น

maximum accuracy (92%), precision (92%), recall (92%), and F1 score (92%), immediately followed by the MIL-SocNet. The MIL-SocNet yielded an accuracy, precision, recall, and F1 score of 90%, 91%, 90%, and 90% ซึ่งจากผลการทดลองที่กล่าวมาแสดงให้เห็นว่าตัวของ MIL สามารถแสดงประสิทธิภาพได้ดีและมีความคาดหวังว่าจะมีการนำมาใช้ให้เป็นประโยชน์ และได้การยอมรับมากขึ้นในอนาคตดังแสดงในภาพที่ 7

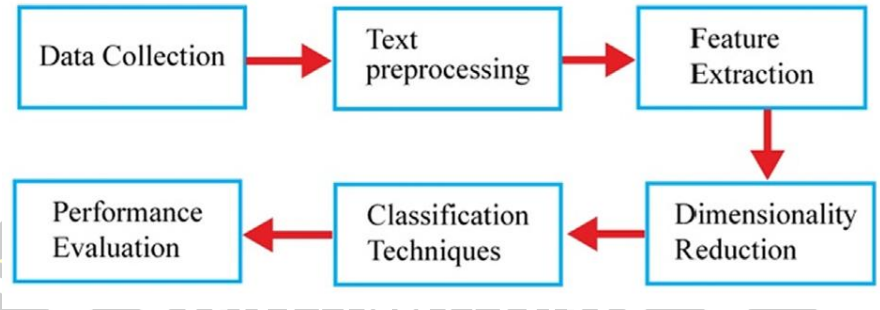


ภาพที่ 7 ขั้นตอนการดำเนินงานวิจัยของ Akkapon Wongkoblap และคณะ[8]

งานวิจัยของ Xiaoyu Luo [9] ได้กล่าวถึงการทำงานขององค์กรณ์ที่แม้ว่าจะมีการจัดการในการจัดเก็บข้อมูลที่ดีแต่ก็อาจจะเกิดปัญหาขึ้นในกรณีที่ทำการเก็บข้อมูลในจำนวนมากทั้งข้อมูลที่มาในรูปแบบ ไฟล์เอกสารรวมถึงอีเมลต่าง ๆ จึงได้ทำการนำวิธีการแก้ปัญหาโดยใช้ Machine Learning มาช่วยในการจำแนกข้อมูลที่มีปริมาณมากนี้ให้ได้ตามต้องการ ซึ่งเป็นการทำนายหมวดหมู่ของข้อความที่ใหม่ๆ ตามโมเดลที่ได้สร้างขึ้นมา การจัดหมวดหมู่นี้มักจะใช้กับเอกสารและข้อความที่มีจำนวนมาก โดยมีขั้นตอนหลักๆ ได้แก่ การเตรียมข้อมูลก่อน (Pre-processing) ซึ่งรวมถึงการลบคำทั้งซ้ำและลบคำที่ไม่มีความสำคัญ (Stop-words) และการทำให้คำเหล่านั้นมีรูปแบบเดียวกัน (Stemming) นอกจากนี้ยังมีการใช้การวิเคราะห์สถิติในการเลือกคุณลักษณะ (Features) ที่สำคัญในการจัดหมวดหมู่ โดยใช้เทคนิคต่างๆ อาทิเช่น [9] Naïve Bayes, Decision Tree, Neural Network,

Support Vector Machines และเทคนิค Hybrid ในการทำนายหมวดหมู่ของข้อความ นอกจากนี้ยังพูดถึงปัญหาที่อาจเกิดขึ้น เช่น การจัดการกับข้อความที่ถูกจัดหมวดหมู่ผิด การจัดการกับจำนวนมากของคุณลักษณะข้อความ และการเลือกเทคนิคการเรียนรู้ที่มีประสิทธิภาพ

การจัดหมวดหมู่ข้อความนี้มีความสำคัญมากในการแยกแยะข้อมูลที่มีจำนวนมาก เช่นในการกรองข้อความสแปมในอีเมล การระบุเงินปลอม การตรวจสอบข่าวปลอม และเทคนิคการทำการวิเคราะห์ความคิดเห็น (Opinion mining) รวมถึงการอธิบายถึงวิธีการทำ Text Mining ที่เป็นกระบวนการหลักในการค้นพบความรู้จากข้อความ (Knowledge Discovery in Text) และการจัดหมวดหมู่ข้อมูลที่มีลักษณะคล้ายกัน ในทางปฏิบัติ Text Mining เกี่ยวข้องกับการค้นหาข้อมูลที่เกี่ยวข้อง การหาความสัมพันธ์ การจัดกลุ่ม และการตรวจสอบความเปลี่ยนแปลงในข้อมูลที่มีโครงสร้างไม่เรียงรายละเอียด การทำ Text Mining มีการใช้เทคนิคในการหาข้อมูลที่เกี่ยวข้อง โดยมีการเปรียบเทียบว่า Support Vector Machines และ Naïve Bayes เป็นเทคนิคที่มีความแม่นยำสูงที่สุด และถูกนำมาใช้ในการจัดหมวดหมู่ข้อความ จากผลของการทดลองทำให้ได้ทราบว่า SVM เป็นเทคนิคการจัดหมวดหมู่ข้อความภาษาอังกฤษที่มีประสิทธิภาพสูงกว่า NB และ LR สำหรับชุดข้อมูลที่ใช้ Xiaoyu Luo ยังแนะนำว่าในอนาคตจะใช้ SVM ในการจัดหมวดหมู่ข้อความ BBC ซึ่งต้องใช้ hardware specification ที่สูงขึ้นและ tensors flow หรือ python เพื่อจำลองผลดังแสดงในภาพที่ 8



ภาพที่ 8 ขั้นตอนการดำเนินงานวิจัยของ Xiaoyu Luo[9]

งานวิจัยของ Hongping Wu และคณะ [10] ได้อธิบายถึงความสำคัญของการจัดหมวดหมู่ข้อความในการประมวลผลภาษาธรรมชาติ และแสดงตัวอย่างการประยุกต์ใช้ เช่น การวิเคราะห์อารมณ์, การจัดหมวดหมู่ข่าว และการจัดหมวดหมู่หัวข้อ ได้ทำการวิเคราะห์และสรุปข้อเสียของวิธีการจัดหมวดหมู่ข้อความที่ใช้การเรียนรู้เชิงเครื่อง ซึ่งใช้การแทนข้อความด้วยแบบจำลองที่ไม่สามารถรักษาความหมายและบริบทได้ เช่น one-hot, Bag-of-Words, TF-IDF และ N-Gram รวมถึงได้แนะนำการจัด

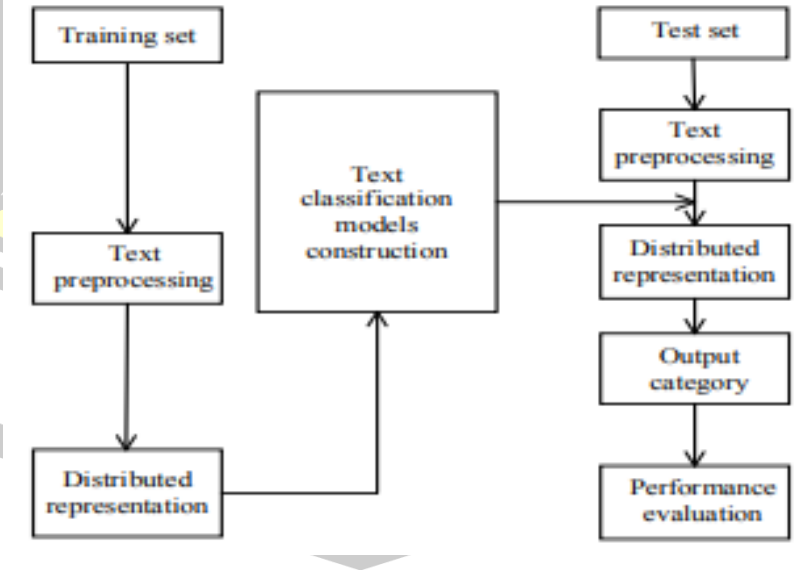
หมวดหมู่ข้อความที่ใช้โครงข่ายประสาทเทียม (Neural Network) ซึ่งสามารถป้องกันปัญหาในการแยกแยะลักษณะเด่น และสามารถทำนายผลได้แม่นยำกว่า โดยใช้การแทนข้อความด้วย Word Embedding และโครงข่ายประสาทเทียม(Neural Network) ต่างๆ เช่น CNN, RNN, Attention Mechanism โดยตัวของ Hongping Wu ได้ใช้ Hierarchical Attention

Network (HAN) มาใช้ในการทดลองดังแสดงในภาพที่ 9 โดยได้ผลการทดลองเปรียบเทียบคือ

Model	Precision	Training time
CNN	0.8534	77 s/epoh×2 epoh
RNN	0.8273	164 s/epoh×2 epoh
HAN	0.8456	179 s/epoh×2 epoh

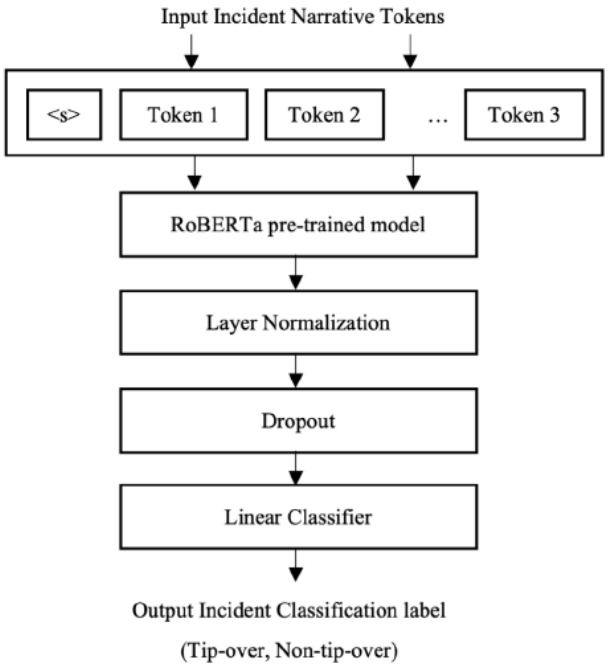
ภาพที่ 9 ผลการทดลองเปรียบเทียบของ Hongping Wu และคณะ[10]

สรุปได้ว่าถึงแม้ว่าการทดลองจะทำได้ค่อนข้างดีแต่ก็มีปัญหาอยู่ในบางส่วนที่ต้องไปแก้ไขคือ การจัดหมวดหมู่ข้อความที่ใช้การเรียนรู้เชิงลึก เช่น ความไม่เป็นที่ยอมรับ, ความยากในการปรับพารามิเตอร์, ความยุ่งยากในการได้รับข้อมูลฝึก, ความต้องการทรัพยากรที่สูง เป็นต้น



ภาพที่ 10 ขั้นตอนการดำเนินงานวิจัยของ Hongping Wu และคณะ[10]

งานวิจัยของ Ben Rodrawangpai และ Witawat Daungjaiboon ได้กล่าวถึง [11] สรุปรายงานของ Consumer Product Safety Commission (CPSC) ปี 2020 ระบุว่า อุบัติการณ์ความเสี่ยงจากการล้มของเฟอร์นิเจอร์ในสหรัฐฯ ทำให้เกิดอุบัติเหตุเฉลี่ยประมาณ 25,000 ครั้งต่อปีที่ต้องรักษาในห้องฉุกเฉินของโรงพยาบาล รายงานยังระบุว่ามีการตรวจพบว่ามีผู้เสียชีวิต 571 รายจากอุบัติการณ์ล้มของเฟอร์นิเจอร์ระหว่างปี 2000 ถึง 2019 โดยเด็กอายุต่ำกว่า 18 ปีเป็นส่วนใหญ่ (82%) ของผู้เสียชีวิต จากที่กล่าวมาคณะกรรมการความปลอดภัยของผู้บริโภคในสหรัฐ (CPSC) ได้เผยแพร่ข้อมูลที่เกี่ยวข้องกับการบาดเจ็บจากผลิตภัณฑ์ผู้บริโภค ปริมาณข้อมูลเพิ่มขึ้นอย่างรวดเร็ว และกระบวนการที่ต้องทำการตรวจสอบและจัดประเภทเหตุการณ์แต่ละรายการด้วยมือกลายเป็นการใช้ทรัพยากรมากมาย จึงได้ทำให้มีการนำเสนอวิธีการที่ดีขึ้นที่ใช้ร่วมกันระหว่างเทคนิคการประมวลผลภาษาธรรมชาติ (NLP) และอัลกอริทึมเรียนรู้ของเครื่อง (ML) เพื่อจัดประเภทข้อมูลที่เป็นข้อความ โมเดลเรียนรู้ของเครื่องสามารถช่วยลดเวลาและความพยายามโดยการประสานงานในการจัดประเภทเนื้อหาข้อความดังแสดงในภาพที่ 11



ภาพที่ 11 ขั้นตอนการดำเนินงานวิจัยของ Ben Rodrawangpai และคณะ[11]

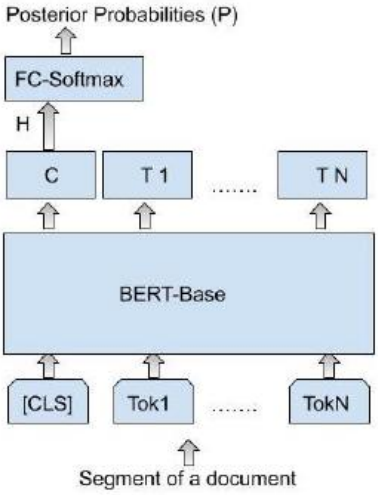
โดยใช้โมเดลการจัดประเภท default อย่าง BERT และโมเดลอื่น ๆ จาก Transformer เพื่อเปรียบเทียบกับวิธีการที่ Ben Rodrawangpai และ Witawat Daungjaiboon เสนอ โดยใช้ pretrained language models อย่าง BERT, RoBERTa, DeBERTa, ALBERT, DistilBERT, และ

MPNet เพื่อทำการ preprocess ข้อความของเหตุการณ์และทำ fine tune ให้เหมาะสำหรับงาน text classification ใช้ชุดข้อมูลจากข้อมูล CPSC ที่นำมาใช้ในการทดลอง

Model	Macro			Weighted		
	Precision	Recall	F1	Precision	Recall	F1
Proposed model	0.96	0.96	0.96	0.98	0.98	0.98
DeBERTa	0.91	0.92	0.92	0.96	0.96	0.96
BERT	0.93	0.90	0.91	0.96	0.97	0.96
DistilBERT	0.92	0.90	0.91	0.96	0.96	0.96
MPNet	0.93	0.88	0.90	0.96	0.96	0.96
RoBERTa	0.90	0.90	0.90	0.96	0.96	0.96
ALBERT	0.90	0.88	0.89	0.95	0.95	0.95

ภาพที่ 12 ผลการทดลองเปรียบเทียบของ Ben Rodrawangpai และคณะ[11]

งานวิจัยของ Raghavendra Pappagari และคณะ [12] ได้กล่าวถึงการสนทนาในสาย Call Center ซึ่งโดยทั่วไปมีความสั้นและมักมีตัวแทนพยายามแก้ไขปัญหาที่ซับซ้อนของลูกค้า ทำให้บางครั้งการโทรอาจใช้เวลาหลายชั่วโมง สำหรับการวิเคราะห์เสียงสิ่งที่เรียกว่าการ automatic speech recognition (ASR) มักนำไปใช้ในการทำ transcript [12]เสียงเป็นข้อความที่มีการประมวลผลต่อไปในกระบวนการ NLP ข้อมูลที่ถูก transcribe บางครั้งมีความยาวเกิน 5000 คำ นอกจากนี้ข้อมูลทางเวลาอาจเป็นสิ่งที่สำคัญในงานเช่น CSAT (Customer Satisfaction) เช่นลูกค้าอาจโกรธเมื่อเริ่มต้นการโทรแต่หลังจากที่ปัญหาของเขาได้รับการแก้ไขเขาอาจจะรู้สึกพอใจมาก ๆ กับวิธีการจัดการ เพราะฉะนั้นโมเดลที่ใช้โครงสร้าง bag of words หรือโมเดลที่ไม่รวมถึงความขึ้นต่อตราบรรจบระหว่างข้อมูล input อาจไม่เหมาะสมกับการทำงานในหมวดหมู่นี้ นี่เป็นแรงบันดาลใจในการใช้โมเดลเช่น BERT ในงานนี้ โดยทาง Raghavendra Pappagari และคณะ ได้ใช้โมเดล 2 ประเภทคือ Transformer over BERT และ Recurrence over BERT มาทำการวัดประสิทธิภาพและทำการเปรียบเทียบกัน



ภาพที่ 13 ขั้นตอนการดำเนินงานวิจัยของ Raghavendra Pappagari และคณะ[12]

โดยการทดลองได้ใช้ดาต้าเซ็ต 3 ชนิดคือ

- CSAT dataset for CSAT prediction, consisting of spoken transcripts (automatic via ASR).
- 20 newsgroups for topic identification task, consistingof written text;
- Fisher Phase 1 corpus for topic identification task, consisting of spoken transcripts(manual)

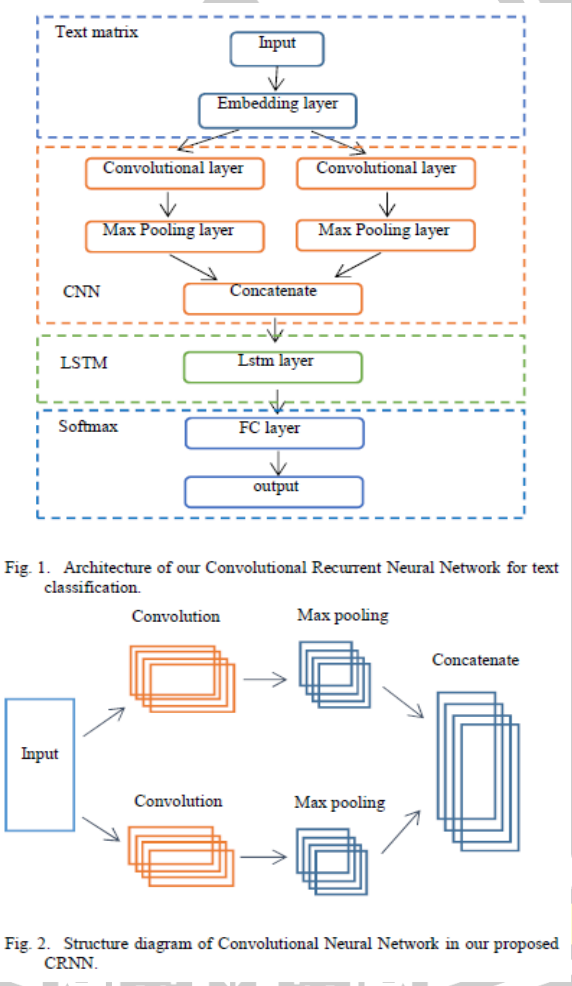
ตารางที่ 4 Results using segment representations (H) from a pre-trained BERT (without fine-tuning)

	RoBERT	ToBERT
CSAT	71.16	74.77
20newsgroups	60.75	65.04
Fisher	38.04	80.68

ตารางที่ 5 Results using segment representations (H) from a fine-tuned BERT

	RoBERT	ToBERT
CSAT	83.65	83.48
20newsgroups	84.71	85.52
Fisher	82.28	95.48

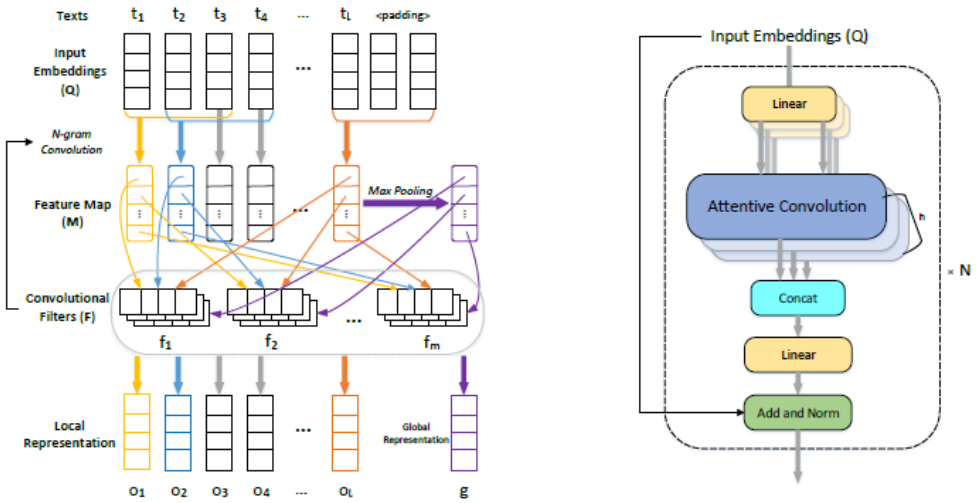
งานวิจัยของ Ruishuang Wang และคณะ [13] ได้ทำการนำเสนอ convolutional recurrent neural network สำหรับการจัดประเภทข้อความทำการทดลองบนชุดข้อมูลภาษาจีน 2 ชุดข้อมูล และชุดข้อมูลภาษาอังกฤษ 5 ชุด และเปรียบเทียบวิธีการจัดประเภทอื่น ๆ เพื่อแสดงให้เห็นว่าผลการทดลองแสดงให้เห็นว่าวิธีการที่Ruishuang Wang และคณะเสนอสามารถทำได้ดีกว่าในงานจัดประเภทข้อความ



ภาพที่ 14 ขั้นตอนการดำเนินงานวิจัยของ Ruishuang Wang และคณะ[13]

งานวิจัยของ Pengfei Li และคณะ [14] ได้ทำการนำเสนอ Attentive Convolutional Transformer (ACT) ที่ใช้ประโยชน์จากทั้ง Transformer และ CNN สำหรับการจัดประเภทข้อความอย่างมีประสิทธิภาพโดยเฉพาะ โดยPengfei Li และคณะสังเกตได้ว่าโมเดลที่ใช้ attention มักมีประสิทธิภาพดีกว่าโมเดลที่ใช้ RNN, เนื่องจากการใช้ attention mechanisms มีความสามารถดีในการจับ dependencies ไปไกลๆโดยเฉพาะ self-attention ที่ใช้ใน Transformer โมเดล ACT ที่เราเสนอดีกว่าโมเดล attentive ทั้งหมดรวมถึง Transformer

encoder สาเหตุที่ ACT ดีกว่าเป็นเพราะมีความสามารถในการสกัดลักษณะ n-gram ดีขึ้นด้วยกลไก attentive convolution



ภาพที่ 15 ขั้นตอนการดำเนินงานวิจัยของ Pengfei Li และคณะ[14]

วิธีการวัดประสิทธิภาพ

Actual \ Predicted	Positive (P')	Negative (N')
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

ภาพที่ 16 confusion matrix ของการวัดประสิทธิภาพ

Accuracy

คืออัตราส่วนของตัวอย่างที่ถูกต้องทั้งหมด ซึ่งเป็นตัววัดที่ง่ายและเป็นที่นิยมสำหรับงานที่เกี่ยวกับการจำแนกประเภทข้อความหรืองานที่มีการทำนายผลลัพธ์เป็นค่าดิบ (raw output) สูตรการคำนวณ Accuracy

$$Accuracy = \frac{TruePositives+TrueNegatives}{TruePositives+FalsePositives+TrueNegatives+FalseNegatives} \quad (1)$$

F1-score

เป็นการวัดประสิทธิภาพที่ใช้ค่า Precision และ Recall โดยคำนวณจากค่าเฉลี่ยความแม่นยำระหว่าง Precision และ Recall ซึ่งเหมาะสำหรับงานที่มีความสำคัญเท่ากันระหว่าง Accuracy และ Recall

สูตรการคำนวณ F1-score

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2)$$

Precision

คืออัตราส่วนของตัวอย่างที่ถูกต้องที่ระบบจำแนกหรือค้นหาได้เทียบกับตัวอย่างที่ระบบทำนายว่าเป็นไปได้ นั่นคือ จำนวนตัวอย่างที่ถูกต้องที่ระบบทำนายว่าเป็นไปได้ (True Positives) หารด้วยจำนวนตัวอย่างที่ระบบทำนายว่าเป็นไปได้ (True Positives + False Positives)

สูตรการคำนวณ precision

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (3)$$

Recall

คืออัตราส่วนของตัวอย่างที่ถูกต้องที่ระบบจำแนกหรือค้นหาได้เทียบกับจำนวนตัวอย่างทั้งหมดที่เป็นไปได้ในชุดข้อมูล นั่นคือ จำนวนตัวอย่างที่ถูกต้องที่ระบบทำนายว่าเป็นไปได้ (True Positives) หารด้วยจำนวนตัวอย่างทั้งหมดที่เป็นไปได้ในชุดข้อมูล (True Positives + False Negatives)

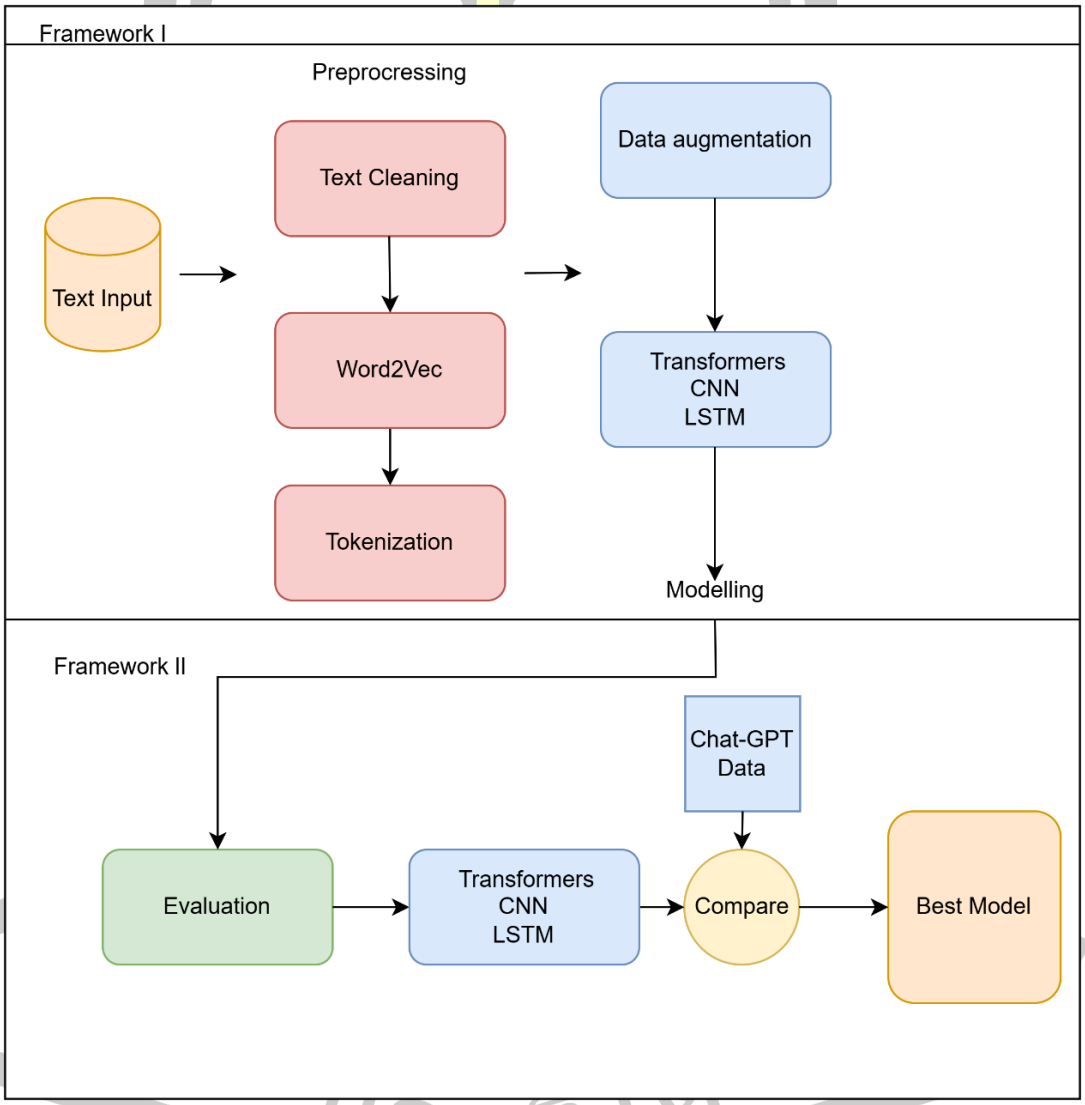
สูตรการคำนวณ recall

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4)$$

จากการศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง เราพบว่าโรคซึมเศร้าเป็นอาการที่มีผลกระทบทั้งต่อชีวิตของผู้ป่วยเองและคนรอบข้างที่มีความสัมพันธ์กับผู้ป่วยด้วย โรคซึมเศร้าสามารถส่งผลกระทบต่อคุณภาพชีวิตของผู้ป่วยในด้านต่าง ๆ เช่น สุขภาพร่างกายและจิตใจ, ความสัมพันธ์กับคนรอบข้าง, การทำงานและการเรียน, และคุณภาพชีวิตทั่วไป ดังนั้นการเลือกใช้วิธีการที่มีประสิทธิภาพและความแม่นยำในการวิเคราะห์ข้อมูลเป็นสิ่งสำคัญ โดยเฉพาะในกรณีของโรคซึมเศร้าที่มีความซับซ้อนและหลากหลาย หลังจากศึกษาและวิเคราะห์ข้อมูลอย่างละเอียด เราจึงตัดสินใจเลือกใช้วิธีการจำแนกข้อมูลที่มีประสิทธิภาพสูงอย่าง BERT เพื่อทำการพัฒนาและปรับปรุงความแม่นยำและประสิทธิภาพของการวิเคราะห์ข้อมูลในงานวิจัย

บทที่ 3
วิธีดำเนินการวิจัย

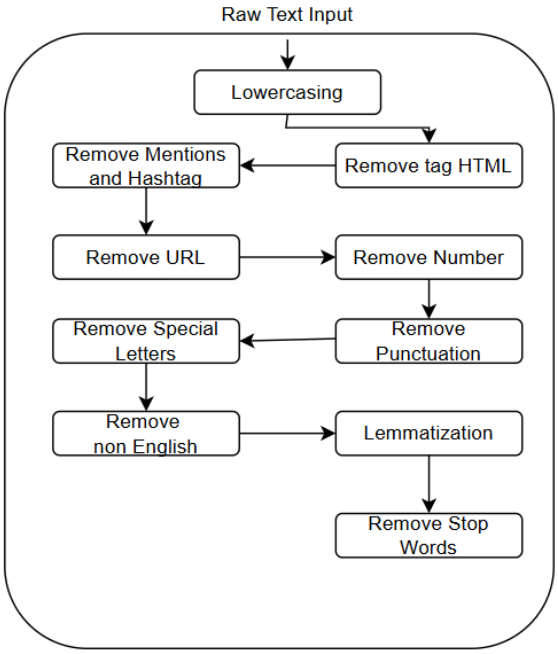
การดำเนินการวิจัยนี้ได้ทำการปฏิบัติตามวัตถุประสงค์ เพื่อทำการดำเนินการพยากรณ์โอกาสการเกิดโรคซึมเศร้า โดยจะแบ่งการดำเนินการเป็นขั้นตอนเริ่มจากการทำการเตรียมข้อมูล การสร้างแบบจำลอง และทำการวัดประสิทธิภาพของแบบจำลอง



ภาพที่ 17 ขั้นตอนการดำเนินงานวิจัย

จากที่แสดงในภาพที่ 17 แสดงถึงขั้นตอนการทำงาน 2 ส่วนคือส่วนแรกใน Framework I ในขั้นตอนนี้จะทำการเตรียมข้อมูลจัดการกับคำที่ไม่เรียบร้อย หลังจากนั้นในส่วนของ Framework II จะทำหน้าที่ในการวิเคราะห์โรคซึมเศร้าจากข้อมูลที่ได้จากขั้นตอนของ Framework I

การทำการเตรียมข้อมูล



ภาพที่ 18 ขั้นตอนการเตรียมข้อมูล

การรวบรวมข้อมูล

ทำการรวบรวมข้อมูลเพื่อนำมาทำการฝึกแบบจำลองจากงานวิจัยของ การจำแนกโรคซึมเศร้า จากพฤติกรรมการโพสต์ข้อความบน Twitter [2] โดยมีข้อมูลดังตารางที่ 3.1

ตารางที่ 6 จำนวนข้อมูลจากแฮชแท็กตามลักษณะอาการชุดข้อมูลสำหรับเรียนรู้แบบจำลองแบบ Imbalance

ลักษณะอาการ	**จำนวนข้อความที่ถูกต้องตามอาการ**
1) ข้อความเกี่ยวกับอารมณ์ซึมเศร้า	3,000 ข้อความ
2) ข้อความเกี่ยวกับการขาดความสนใจลดลง	3,000 ข้อความ
3) ข้อความเกี่ยวกับน้ำหนักผิดปกติ	3,000 ข้อความ
4) ข้อความเกี่ยวกับการนอนผิดปกติ	3,000 ข้อความ
5) ข้อความเกี่ยวกับร่างกายอ่อนเพลีย	3,000 ข้อความ
6) ข้อความเกี่ยวกับการรู้สึกตนเองไร้ค่า	3,000 ข้อความ
7) ข้อความเกี่ยวกับสมาธิสั้น	3,000 ข้อความ
8) ข้อความเกี่ยวกับการเคลื่อนไหวช้า	3,000 ข้อความ
9) ข้อความเกี่ยวกับการคิดฆ่าตัวตาย	3,000 ข้อความ
10) ข้อความที่แสดงถึงอาการปกติ	8,000 ข้อความ

ตารางที่ 7 จำนวนข้อมูลจากแฮชแท็กตามลักษณะอาการชุดข้อมูลสำหรับเรียนรู้แบบจำลองแบบ balance

ลักษณะอาการ	**จำนวนข้อความที่ถูกต้องตามอาการ**
1) ข้อความเกี่ยวกับอารมณ์ซึมเศร้า	3,000 ข้อความ
2) ข้อความเกี่ยวกับการขาดความสนใจลดลง	3,000 ข้อความ
3) ข้อความเกี่ยวกับน้ำหนักผิดปกติ	3,000 ข้อความ
4) ข้อความเกี่ยวกับการนอนผิดปกติ	3,000 ข้อความ
5) ข้อความเกี่ยวกับร่างกายอ่อนเพลีย	3,000 ข้อความ
6) ข้อความเกี่ยวกับการรู้สึกตนเองไร้ค่า	3,000 ข้อความ
7) ข้อความเกี่ยวกับสมาธิสั้น	3,000 ข้อความ
8) ข้อความเกี่ยวกับการเคลื่อนไหวช้า	3,000 ข้อความ
9) ข้อความเกี่ยวกับการคิดฆ่าตัวตาย	3,000 ข้อความ
10) ข้อความที่แสดงถึงอาการปกติ	3,000 ข้อความ

ตารางที่ 8 ตัวอย่างข้อมูลที่มาจากการโพสต์ข้อความ

ชื่อตัวแปร	**คำอธิบาย**	**ตัวอย่างของข้อมูล**
Id	รหัสของโพสต์นั้นๆ	832146985768769748
Created_At	วันเวลาที่โพสต์ข้อความ	2017-01-18 08:49
Text	ข้อความ	i don't deserve this sadness https://t.co/69oyK3fkE1

ขั้นตอนการเตรียมข้อมูล:

```
FUNCTION preprocess_text(text):
    FUNCTION get_wordnet_pos(tag):
        IF tag starts with 'J':
            RETURN ADJECTIVE
        ELSE IF tag starts with 'V':
            RETURN VERB
        ELSE IF tag starts with 'N':
            RETURN NOUN
        ELSE IF tag starts with 'R':
            RETURN ADVERB
        ELSE:
            RETURN NOUN
    text = LOWERCASE(text)
    text = REPLACE(text, '<[^\s]*>', ' ')
    text = REPLACE(text, '(rt\s|rt)?[@]\w+\s*', ' ', IGNORECASE)
    text = REPLACE(text, 'https?:[^\s]+', ' ')
    text = REPLACE(text, '\d+', ' ')

    punctuation_map = CREATE_TRANSLATION_MAP(punctuation characters -> spaces)
    text = TRANSLATE(text, punctuation_map)
    text = REPLACE(text, '[\n\t\b\r]', '')
    text = REPLACE(text, '[^a-zA-Z]', ' ')
    tokens = TOKENIZE(text)
    lemmatizer = INITIALIZE_LEMMATIZER()
    tagged_tokens = TAG_PARTS_OF_SPEECH(tokens)
```

```
FOR EACH (word, tag) IN tagged_tokens:
    lemma = LEMMATIZE(word, get_wordnet_pos(tag))
    ADD lemma TO tokens

# Remove stopwords
stop_words = GET_ENGLISH_STOPWORDS()
tokens = REMOVE_STOPWORDS(tokens, stop_words)

# Join tokens back into a single string
```

ทำการแบ่งโดยแยกประเภทเป็น 10 อย่างคือ ชุดที่ข้อมูล 1 ชุดที่เป็น positive คือชุดที แสดงถึงบุคคลที่ไม่เป็นอาการของโรคซึมเศร้าและ negative ที่แสดงถึงบุคคลที่แสดงอาการของโรค ซึมเศร้าทั้ง 9 อาการ หลังจากนั้นทำการใช้การทำความสะอาดข้อมูลด้วยวิธีการ การลบสัญลักษณ์ พิเศษ การกล่าวถึง ลิงก์ แท็ก ตัวเลข และตัวอักษรที่ไม่ใช่ภาษาอังกฤษ ลบเครื่องหมายวรรคตอน นำข้อความไปผ่านการทำ lemmatize เพื่อให้ได้คำในรากศัพท์ทำการลบ stop words เพื่อลบคำที่พบบ่อยที่ไม่ช่วยในการสื่อความหมาย ข้อความที่ได้จะเป็นคำในรากศัพท์ที่ผ่านการคัดกรองคำที่ไม่จำเป็นออกไปแล้ว

1) ตัวอย่างข้อมูลหลังจากการทำลบสัญลักษณ์พิเศษ การกล่าวถึง ลิงก์ แท็ก ตัวเลข และตัวอักษรที่ไม่ใช่ภาษาอังกฤษ

ตารางที่ 9 ตัวอย่างข้อมูลหลังจากการทำลบข้อต้องห้ามต่าง ๆ

ตัวอย่างข้อความ	**ตัวอย่างข้อความหลังจากการแปลงข้อมูล**
Part of my 6 chair will be shown tonight!	part of my chair will be shown tonight
RT @BuildStrongUSA: Wildfires and hurricanes	wildfires and hurricanes
We seek happiness in external things	we seek happiness in external things

2) ตัวอย่างข้อมูลหลังจากการทำ tokenizer โดยจะทำการตัดโดยดูจาก เครื่องหมายวรรคตอนหรือช่องว่างเป็นหลัก

ตารางที่ 10 ตัวอย่างข้อมูลหลังจากการทำ tokenizer

ตัวอย่างข้อความ	ตัวอย่างข้อความหลังจากการแปลงข้อมูล
part of my chair will be shown tonight	[part, of, my, chair, will, be, shown, tonight]
wildfires and hurricanes	[wildfires, and, hurricanes]
we seek happiness in external things	[we, seek, happiness, in, external, things]

3) ตัวอย่างข้อมูลหลังจากการทำการระบุชนิดของคำ (Part-of-Speech Tagging) โดยจะแทนการระบุชนิดของคำแต่ละโทเค็น เช่น verb แทนด้วย VB,นามแทนด้วย NN,adjective แทนด้วย JJ เป็นต้น

ตารางที่ 11 ตัวอย่างข้อมูลหลังจากการทำ Part-of-Speech Tagging

ตัวอย่างข้อความ	ตัวอย่างข้อความหลังจากการแปลงข้อมูล
[part, of, my, chair, will, be, shown, tonight]	[(part,NN), (of,IN), (my, PRP), (chair,NN), (will,MD), (be,VB), (shown,VBN), (tonight,NN)]
[wildfires, and, hurricanes]	[(wildfires,NNS), (and, CC), (hurricanes,NNS)]
[we, seek, happiness, in, external, things]	[(we,PRP), (seek,VB), (happiness,NN), (in,IN) (external,JJ), (things,NN)]

4) ตัวอย่างข้อมูลหลังจากการทำ Lemmatization แปลงคำให้อยู่ในรูปแบบพื้นฐาน หรือรูปแบบรากศัพท์ที่ถูกต้องทางไวยากรณ์เลยมีการทำ Part-of-Speech Tagging เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด

ตารางที่ 12 ตัวอย่างข้อมูลหลังจากการทำ Lemmatization

ตัวอย่างข้อความ	ตัวอย่างข้อความหลังจากการแปลงข้อมูล
[(part,NN), (of,IN), (my, PRP), (chair,NN), (will,MD), (be,VB), (shown,VBN), (tonight,NN)]	[part, of, my, chair, will, be, show, tonight]
[(wildfires,NNS), (and, CC), (hurricanes,NNS)]	[wildfire, and, hurricane]
[(we,PRP), (seek,VB), (happiness,NN), (in,IN) (external,JJ), (things,NN)]	[we, seek, happiness, in, external, thing]

5) ตัวอย่างข้อมูลหลังจากการทำ (Stop Words Removal) ทำการลบคำที่ไม่มีผลต่อการประมวลผลข้อความมักใช้บ่อยในรูปประโยคของภาษาอังกฤษ เช่น is, am, are, on, the เป็นต้น

ตารางที่ 13 ตัวอย่างข้อมูลหลังจากการทำ Stop Words Removal

ตัวอย่างข้อความ	ตัวอย่างข้อความหลังจากการแปลงข้อมูล
[part, of, my, chair, will, be, show, tonight]	[part, chair, show, tonight]
[wildfire, and, hurricane]	[wildfire, hurricane]
[we, seek, happiness, in, external, thing]	[seek, happiness, external, thing]

หลังจากการทำขั้นตอนข้างต้นมาแล้วได้ทำการตรวจสอบข้อความเพื่อให้มั่นใจว่าข้อความที่นำไปใช้จะไม่ซ้ำกัน เพื่อให้โมเดลเรียนรู้ได้ดีขึ้น

```
Initialize same_classes as a defaultdict of defaultdict(int)

For each row in the DataFrame:
  Get the sentence from the 'text' column
  Get the class from the 'class' column
  Increment the count of this class for the sentence in same_classes

Initialize an empty list filtered_sentences
Initialize count to 0

For each sentence in same_classes:
  If the sentence has more than one class:
    Add the sum of class counts to count
    Append a dictionary with 'Sentence' and 'Count' to filtered_sentences
Print the value of count
Create a DataFrame filtered_df from filtered_sentences
```

1) รวบรวมข้อมูลที่ซ้ำกัน: หาแถวที่ข้อความซ้ำกันในประเภทอาการมากกว่าสองประเภท และรวบรวมข้อมูลเหล่านั้นไว้ด้วยกัน

2) นับจำนวนการพบข้อความ: นับจำนวนครั้งที่ข้อความถูกพบในแต่ละประเภทอาการ และเก็บข้อมูลเหล่านี้ไว้สำหรับการวิเคราะห์เพิ่มเติม

3) ลบแถวที่ซ้ำกันออก: ลบแถวที่มีข้อความซ้ำออกจากชุดข้อมูลเดิม

4) รวมข้อมูลเข้าด้วยกัน: สร้างแถวใหม่ที่รวบรวมข้อมูลจากแถวที่ซ้ำกัน โดยระบุประเภทอาการที่พบและจำนวนการพบในแถวเดียว ส่วนของ Imbalance หลังจากทำการลบข้อมูลที่ซ้ำกันแล้วพบว่ามีข้อความที่ซ้ำกัน 33 ชุดข้อความซ้ำกันรวมทั้งหมด 3817 ครั้ง และส่วนของ Balance หลังจากทำการลบข้อมูลที่ซ้ำกันแล้วพบว่ามีข้อความที่ซ้ำกัน 25 ชุดข้อความซ้ำกันรวมทั้งหมด 3888 ครั้ง ตัวอย่างเช่น

ตารางที่ 14 ตัวอย่างข้อมูลที่เช็คจำนวนคำซ้ำ

ลำดับที่	ตัวอย่างข้อความ	อยู่ที่คลาส: จำนวนที่ซ้ำ
15	care grade tire go sleep idc	{4: 32, 7: 18}
16	go sleep even tire sec later	{4: 148, 7: 185}
17	go sleep every night thinking	{4: 1, 5: 2}

หลังจากทำการลบข้อมูลของ Imbalance ที่มีการซ้ำกันจะได้ชุดข้อความจำนวน 30,779 คำและแต่ละประเภทอาการที่ซ้ำกันจะเหลือ 18,805 คำ และส่วนของ Balance จะได้ชุดข้อความจำนวน 26,023 คำและเมื่อทำการรวมข้อความที่อยู่ในแต่ละประเภทอาการที่ซ้ำกันจะเหลือ 15,698 คำ แบ่งแต่ละประเภทได้ดังนี้

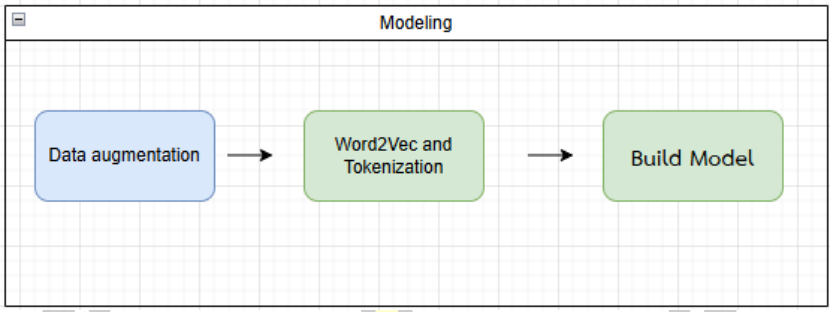
ตารางที่ 15 จำนวนคำในแต่ละประเภทหลังจากการลบข้อมูลของ Imbalance

กลุ่ม	คำ
0	5635
1	1341
2	462
3	1352
4	1732
5	1634
6	1472
7	1854
8	1826
9	1499

ตารางที่ 16 จำนวนคำในแต่ละประเภทหลังจากการลบข้อมูลของ Balance

กลุ่ม	คำ
0	2525
1	1341
2	462
3	1352
4	1732
5	1634
6	1472
7	1854
8	1826
9	1499

การสร้างแบบจำลอง



ภาพที่ 19 ขั้นตอนการทำโมเดล

การเพิ่มชุดข้อมูล

หลังจากทำการเตรียมข้อมูลจะเห็นได้ว่ามีข้อมูลในประเภทที่ 2 ที่มีจำนวนน้อยเกินไปและอาจจะส่งผลกระทบต่อการนำไปใช้ จึงทำการใช้โมเดล GoogleNews-vectors-negative300.bin ที่จะช่วยการเพิ่มจำนวนคำโดยวิธีการเพิ่มชุดข้อมูลของประเภทที่ 2 ขึ้นมาอีก 1 ชุดและแทรกคำที่มีความหมายคล้ายคำเดิมเข้าไปในชุดข้อความตัวยอย่างเช่น

1) คำที่ทำการเตรียมมาแล้ว marrall brexits ribosomal result either loss

2) หลังจากการเพิ่มคำ marrall brexits deet result either loss control

หลังจากทำการเพิ่มแล้วทำการรวมเข้ากับชุดข้อความประเภทที่ 2 จากจำนวน 462 คำ เป็น 924 คำ

Word2Vec และ Tokenzation

1) Word2Vec ทำการสร้างโมเดลโดยวิธีการ Word Embedding โดยใช้งานตัว Word Embedding หลาย ๆ ชั้นมาสร้างเป็นตัวโมเดลแล้วแปลงข้อความเป็นเวกเตอร์

ขั้นตอนการสร้างต้นแบบ Word2Vec:

```
# ตั้งค่าพารามิเตอร์สำหรับการสร้างโมเดล Word2Vec
dimension = 300  # ขนาดของเวกเตอร์ที่ต้องการ


# สร้างโมเดล Word2Vec ด้วยพารามิเตอร์ที่กำหนด
model_w2v = Word2Vec(
documents,          # ข้อมูลข้อความที่ใช้ในการฝึกโมเดล
vector_size = dimension,  # ขนาดของเวกเตอร์ที่ต้องการ
min_count = 1,          # จำนวนขั้นต่ำของคำที่ต้องการเพื่อฝึกโมเดล
window = 3,             # ขนาดของหน้าต่างการคาดการณ์
```

2) Tokenzation เป็นวิธีการที่จะทำการแปลงข้อความให้เป็นตัวเลขเพื่อให้โมเดลสามารถเรียนรู้และทำการคาดการณ์ได้อย่างมีประสิทธิภาพ

การติดตำแหน่ง (Indexing) ให้แต่ละคำ:

- Tokenizer จะทำการแปลงคำทั้งหมดในข้อความให้เป็นหมายเลข (index) โดยใช้ฟังก์ชัน fit_on_texts เพื่อสร้างพจนานุกรมที่มีการจับคู่คำกับตำแหน่ง
- พารามิเตอร์ oov_token ถูกใช้เพื่อกำหนดค่าตัวแทนสำหรับคำที่ไม่อยู่ในพจนานุกรม (Out of Vocabulary, OOV) เช่น "<OOV>"

```
tokenizer=Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(df.text)
```

หลังจากการใช้ fit_on_texts, Tokenizer จะสร้างพจนานุกรม (dictionary) ของคำและตำแหน่งของแต่ละคำ ตัวอย่างผลลัพธ์: {'<OOV>': 1, 'think': 2, 'get': 3,'like': 4, 'go': 5, 'interest': 6}

เมื่อมีพจนานุกรมแล้วสามารถใช้ฟังก์ชัน texts_to_sequences เพื่อแปลงข้อความเป็นลิสต์ของหมายเลขตามพจนานุกรมที่สร้างขึ้น ตัวอย่างผลลัพธ์ของ Sequences:

think get like go interest = [2,3,4,5,6] ที่แสดงถึงตำแหน่งของคำที่แปลงเป็น Tokenizer

โดย ตำแหน่งของ think คือ 2

ตำแหน่งของ get คือ 3

ตำแหน่งของ like คือ 4

ตำแหน่งของ go คือ 5

ตำแหน่งของ interest คือ 6

สร้างโมเดล

1) การทำ embedding matrix โดยจะใช้เป็นค่าน้ำหนักของคำเพื่อนำไปใช้ฝึกโมเดล

สร้าง embedding matrix:

(1)สร้าง array ที่มีค่า 0 ขนาด 300 (ขนาดของเวกเตอร์) สำหรับคำทั้งหมดที่อยู่ใน Tokenizer

(2)นำ index ของคำจาก Tokenizer มาหาเวกเตอร์จากโมเดล Word2Vec

```
dimension = 300


embedding_matrix=np.zeros((vocab_size, dimension))
for word, i in tokenizer.word_index.items():
if word in model_w2v.wv:
embedding_matrix[i]=model_w2v.wv[word]
print(embedding_matrix.shape)
```

(3)หากคำไม่มีในโมเดล Word2Vec จะใช้เวกเตอร์ที่เป็นค่า 0

ตัวอย่าง

```
(array([[   0,    0,    0, ...,   66,  531, 2414],
        [   0,    0,    0, ...,    0,   85, 3126],
        [   0,    0,    0, ...,   85, 10184, 1523],
                                                ...,
        [   0,    0,    0, ...,  360,   37,  230],
        [   0,    0,    0, ...,    2,   22,   11],
        [   0,    0,    0, ...,  202,  417,  471]]),
```

2) การทำ Text to sequences ในการเตรียมข้อมูลสำหรับการฝึกโมเดล Deep Learning คุณจำเป็นต้องทำให้ความยาวของ array ที่แทนตำแหน่งของคำ (word indices) ในแต่ละข้อความมีขนาดเท่ากัน สำหรับกรณีนี้ เราจะกำหนดความยาวสูงสุดเป็น 300 ดังนี้:

(1) แปลงข้อความเป็นตำแหน่งของคำ: ใช้ Tokenizer เพื่อแปลงข้อความเป็นตัวเลขตำแหน่งของคำ

(2) ปรับขนาดความยาว: ทำให้ความยาวของ array ตำแหน่งของคำในแต่ละข้อความเท่ากัน

```
sequences = tokenizer.texts_to_sequences(df['text'].values)
X = pad_sequences(sequences, maxlen=300)
X, X.shape
```

ที่ 300 โดยการ padding

3) การทำการแบ่งข้อมูลเพื่อทดลอง

```
x_train, x_test, y_train, y_test = train_test_split(X , Y, test_size=0.3, random_state=42)
x_test, x_val, y_test, y_val = train_test_split(x_test , y_test, test_size=0.5,
random_state=42)
x_train.shape, y_train.shape, x_test.shape, y_test.shape, x_val.shape, y_val.shape
```

แบ่งชุดข้อมูลเพื่อฝึกโมเดลเป็น 70% ของชุดข้อมูล ทดสอบ 15% และตรวจสอบ15%

4) โมเดล Tranformer

โครงสร้างของโมเดล:

Class TokenAndPositionEmbedding (inherits from Layer):

Function __init__(maxlen, vocab_size, embed_dim):

- Call the parent class constructor (super)

- Initialize token embedding layer (token_emb) with vocab_size as input dimension and embed_dim as output dimension

- Initialize positional embedding layer (pos_emb) with maxlen as input dimension and embed_dim as output dimension

Function call(input_sequence):

- Get the sequence length (maxlen) from the shape of the input_sequence

- Create a range of positions (from 0 to maxlen-1)

- Pass the positions through the positional embedding layer (pos_emb)

- Pass the input_sequence through the token embedding layer (token_emb)

- Add the token embeddings to the positional embeddings

- Return the result (combined token and positional embeddings)

ส่วนประกอบ:

เมธอด __init__:

(1) maxlen: ความยาวสูงสุดของลำดับ (จำนวนโทเคน) ที่โมเดลจะรับเข้าไป ซึ่งเป็นตัวกำหนดจำนวนตำแหน่งที่ต้องใช้ใน Position Embedding

(2) vocab_size: ขนาดของพจนานุกรม (จำนวนโทเคนทั้งหมดในข้อมูล) ที่ใช้ใน Token Embedding

(3) embed_dim: ขนาดของเวกเตอร์ฝังโทเคน (embedding vector) โดยแต่ละโทเคนจะถูกแทนด้วยเวกเตอร์ที่มีขนาดเท่ากันตามค่าที่กำหนดนี้

(4) self.token_emb = Embedding(input_dim=vocab_size, output_dim=embed_dim): เป็นชั้นฝังโทเคน (Token Embedding) ซึ่งจะทำการแปลงโทเคนแต่ละตัวในข้อมูลให้เป็นเวกเตอร์ที่มีขนาด embed_dim

(5) self.pos_emb = Embedding(input_dim=maxlen, output_dim=embed_dim): เป็นชั้นฝังตำแหน่ง (Position Embedding) โดยตำแหน่งของโทเคนในลำดับจะถูกแทนด้วยเวกเตอร์ตำแหน่งที่มีขนาด embed_dim เช่นเดียวกัน

เมธอด call:

(1) maxlen = tf.shape(x)[-1]: ดึงค่าความยาวของลำดับโทเคน (จำนวนโทเคน) ที่รับเข้ามาในอินพุต x

(2) positions = tf.range(start=0, limit=maxlen, delta=1): สร้างลำดับของตำแหน่ง (0, 1, 2, 3, ...) ที่สอดคล้องกับแต่ละโทเคนในลำดับ เพื่อใช้ในการฝังตำแหน่ง (Position Embedding)

(3) positions = self.pos_emb(positions): นำตำแหน่งเหล่านี้ไปผ่านชั้น Embedding เพื่อแปลงตำแหน่งแต่ละตัวเป็นเวกเตอร์ตำแหน่งที่มีขนาด embed_dim

(4) x = self.token_emb(x): แปลงโทเคนในอินพุต x ให้เป็นเวกเตอร์ฝังโทเคนผ่านชั้น Token Embedding

(5) return x + positions: บวกเวกเตอร์ฝังโทเคนกับเวกเตอร์ฝังตำแหน่งเพื่อให้ข้อมูลในลำดับมีทั้งข้อมูลเนื้อหา (จาก Token Embedding) และข้อมูลตำแหน่ง (จาก Position Embedding) ซึ่งช่วยให้โมเดลเข้าใจตำแหน่งของแต่ละโทเคนในลำดับ

Class TransformerBlock (inherits from Layer):

Function __init__(embed_dim, num_heads, ff_dim, rate=0.1):

- Call the parent class constructor (super)

- Store the embed_dim, num_heads, and ff_dim as attributes

- Initialize Multi-Head Attention layer (att) with num_heads and key_dim set to embed_dim

- Initialize a feed-forward network (ffn) with two Dense layers:

- First Dense layer has ff_dim units and uses "relu" activation

- Second Dense layer has embed_dim units

- Initialize two LayerNormalization layers (layernorm1 and layernorm2)

- Initialize two Dropout layers (dropout1 and dropout2) with the rate specified

Function call(inputs, training):

- Apply the Multi-Head Attention layer (att) on inputs (self-attention)

- Apply Dropout (dropout1) to the attention output (training=True during training)

- Add inputs to attention output (residual connection) and normalize using LayerNormalization (layernorm1)

- Pass the normalized result through the feed-forward network (ffn)

- Apply Dropout (dropout2) to the feed-forward output (training=True during training)

- Add the previous normalized result to the feed-forward output (residual connection) and normalize again using LayerNormalization (layernorm2)

- Return the final output

Function get_config():

- Return the configuration of the block, including stored attributes for embed_dim, num_heads, ff_dim, and dropout rate

ส่วนประกอบ:

เมธอด __init__:

(1) embed_dim: ขนาดของเวกเตอร์ที่ใช้แทนคำหรือโทเคน ขนาดนี้จะถูกใช้ตลอดโมเดลเพื่อกำหนดว่าการฝังแต่ละโทเคนจะมีมิติกี่มิติ

(2) num_heads: จำนวนหัวใน Attention กลไก Multi-Head Attention จะใช้หัวหลายๆ หัวเพื่อประมวลผลความสัมพันธ์ระหว่างโทเคนต่างๆ ในลักษณะขนานกัน ทำให้โมเดลสามารถเรียนรู้หลายรูปแบบความสัมพันธ์ได้

(3) ff_dim: ขนาดของ Dense layer ใน feed-forward network ซึ่งใช้ประมวลผลข้อมูลหลังจากการคำนวณ Attention โดยปกติจะใหญ่กว่า embed_dim เพื่อเพิ่มความสามารถในการเรียนรู้ของโมเดล

(4) MultiHeadAttention: กลไกที่ทำให้โมเดลสามารถโฟกัสกับโทเคนที่สำคัญในลำดับเดียวกันหรือในลำดับที่ต่างกันได้ในเวลาเดียวกัน

(5) ffn (Feed-Forward Network): เครือข่ายประสาทเทียมแบบฟีดฟอร์เวิร์ดสองชั้นที่ใช้คำนวณผลลัพธ์จากข้อมูลที่ได้จาก Attention เป็นการแปลงข้อมูลให้ซับซ้อนขึ้น

(5.1) ชั้นแรกใช้ฟังก์ชัน relu เพื่อเพิ่ม non-linearity ให้โมเดล

(5.2) ชั้นที่สองปรับขนาดผลลัพธ์กลับไปที่ embed_dim เพื่อให้ตรงกับมิติของข้อมูลต้นฉบับ

(6) LayerNormalization: การนอร์มัลไลซ์ค่าในชั้นต่างๆ ของโมเดล ช่วยให้กระบวนการฝึกฝนมีความเสถียร โดยไม่ทำให้ข้อมูลที่ผ่านมามีการเบี่ยงเบนมากเกินไป

(7) Dropout: ชั้น Dropout ใช้ในการป้องกันการเกิด overfitting โดยสุ่มปิดนิวรอนบางส่วนระหว่างการฝึกฝนโมเดล

เมธอด call:

(1) attn_output = self.att(inputs, inputs): คำนวณ Multi-Head Attention ระหว่างโทเคนในอินพุต

(2) attn_output = self.dropout1(attn_output, training=training): ใช้ Dropout กับผลลัพธ์จาก Attention (ทำงานเฉพาะเมื่ออยู่ในโหมดการฝึกฝน)

(3) out1 = self.layernorm1(inputs + attn_output): เพิ่มค่า Attention กับอินพุตต้นฉบับ (Residual Connection) แล้วนอร์มัลไลซ์ผลลัพธ์

(4) ffn_output = self.ffn(out1): ส่งผลลัพธ์ผ่าน Feed-Forward Network

(5) ffn_output = self.dropout2(ffn_output, training=training): ใช้ Dropout กับผลลัพธ์จาก Feed-Forward Network

(6) return self.layernorm2(out1 + ffn_output): เพิ่มผลลัพธ์ของ Feed-Forward กับค่าเดิมอีกครั้ง (Residual Connection) แล้วนอร์มัลไลซ์ผลลัพธ์เพื่อส่งออก

เมธอด get_config:

บันทึกค่าพารามิเตอร์ที่ใช้ในชั้น TransformerBlock เพื่อให้สามารถบันทึกโมเดลหรือเรียกใช้ซ้ำได้

5) โมเดล CNN

Input Layer:

- Convert words to vector representations using a pre-trained embedding matrix
- Set embeddings to non-trainable

Dropout Layer (50%):

- Randomly drop out 50% of the neurons to prevent overfitting

Convolutional Layer (1D CNN):

- Apply 300 filters with a kernel size of 5
- Use ReLU activation to extract important features
- Use 'same' padding to maintain input size

Global Max Pooling Layer:

- Reduce dimensionality by selecting the maximum value from each filter

Fully Connected Dense Layer:

- Apply 300 neurons with ReLU activation
- Learn deeper patterns from extracted features

Dropout Layer (50%):

- Further reduce overfitting by dropping out 50% of neurons

Output Layer (Softmax Activation):

- Classify text into one of 10 categories
- Convert final feature vector into probability distribution

8. **Return Predicted Class**

การทำงานของโมเดล CNN สำหรับ Text Classification

โมเดลนี้ใช้ Convolutional Neural Network (CNN) เพื่อ จัดประเภทข้อความ โดยดึงฟีเจอร์ที่สำคัญออกจากข้อความผ่าน Convolutional และ Pooling Layers

(1) Embedding Layer

เปลี่ยน คำ เป็น เวกเตอร์เชิงความหมาย ใช้ Pretrained Word Embeddings (Word2Vec) ค่าที่ได้คือ เมทริกซ์ที่แทนข้อความในรูปเวกเตอร์

(2) Dropout Layer (50%)

ลด Overfitting โดยปิดบางส่วนของเวกเตอร์ป้องกันโมเดลจากการพึ่งพาข้อมูลบางส่วนมากเกินไป

(3) Convolutional Layer (1D CNN)

ใช้ Conv1D ขนาด 300 filters กับ Kernel Size 5 ดึง ฟีเจอร์สำคัญจากข้อความ เช่น N-grams ใช้ ReLU Activation เพื่อให้โมเดลเรียนรู้ฟีเจอร์ที่ซับซ้อน

(4) Global Max Pooling Layer

ลดขนาดของข้อมูลโดยเลือก ค่าที่มากที่สุดของแต่ละฟีเจอร์ ช่วยให้โมเดลเก็บฟีเจอร์ที่เด่นที่สุด

(5) Fully Connected Layer (Dense Layer)

ใช้ ReLU Activation เพื่อเพิ่มความสามารถในการเรียนรู้ฟีเจอร์ มีขนาด 300 Neurons

(6) Dropout Layer (50%)

ลด Overfitting เพิ่มเติม

(7) Softmax Output Layer

แปลงค่าฟีเจอร์ให้เป็น ค่าความน่าจะเป็น ของแต่ละคลาส ใช้กับปัญหาที่มี 10 คลาส

Input Layer:

- Convert words into vector representations using a pre-trained embedding matrix
- Set embeddings to non-trainable

Spatial Dropout Layer (20%):

- Randomly drop out 20% of input embeddings to improve generalization

Bidirectional LSTM Layer (300 units, return sequences = True):

- Process input text in both forward and backward directions
- Capture long-range dependencies in the text
- Return sequences for the next LSTM layer

Bidirectional LSTM Layer (300 units, last output only):

- Further refine feature extraction from text sequences
- Output a single feature vector representing the entire sequence

Dropout Layer (20%):

- Prevent overfitting by randomly deactivating 20% of neurons

Fully Connected Dense Layer (64 neurons, ReLU Activation):

- Learn deeper representations from the LSTM output

Dropout Layer (20%):

- Further prevent overfitting

Output Layer (Softmax Activation):

- Classify text into one of `CLASS_NUM` categories
- Convert the final feature vector into a probability distribution

Return Predicted Class

6) โมเดล LSTM

การทำงานของโมเดล LSTM สำหรับ Text Classification

โมเดลนี้ออกแบบมาเพื่อจัดประเภทข้อความโดยใช้ LSTM (LSTM) และ Pretrained Embedding โดยมีขั้นตอนหลักดังนี้:

(1) Embedding Layer

ทำหน้าที่แปลงคำเป็น เวกเตอร์เชิงความหมายใช้ Pretrained Word Embeddings (Word2Vec) ช่วยให้โมเดลเข้าใจความสัมพันธ์ของคำในเชิงความหมาย

(2) Spatial Dropout

ป้องกัน Overfitting โดยสุ่มปิดบางส่วนของเวกเตอร์คำ ทำให้โมเดลเรียนรู้ข้อมูลที่หลากหลายขึ้น

(3) Bidirectional LSTM (ชั้นแรก)

เป็น LSTM แบบสองทาง (Forward และ Backward) วิเคราะห์ข้อความทั้งจาก ต้นทางไปปลายทาง และ ย้อนกลับ ช่วยให้เข้าใจบริบทของข้อความในทุกมิติ

(4) Bidirectional LSTM (ชั้นที่สอง)

ใช้ LSTM อีกชั้นเพื่อสกัดข้อมูลที่ซับซ้อนขึ้น ไม่มีการส่งคืนลำดับข้อมูล (return sequences = False) ค่าที่ได้จาก LSTM สุดท้ายนี้เป็นตัวแทนของข้อความทั้งหมด

(5) Dropout Layer

ลดปัญหา Overfitting โดยสุ่มปิดบาง Neurons ทำให้โมเดลมีความยืดหยุ่นและลดการพึ่งพาข้อมูลเดิมมากเกินไป

(6) Fully Connected Layer (Dense Layer)

เป็นชั้นที่ทำหน้าที่ แยกประเภทข้อมูล ใช้ ReLU Activation Function เพื่อช่วยให้โมเดลสามารถเรียนรู้ฟีเจอร์ที่ซับซ้อนได้

(7) Dropout Layer (เพิ่มเติม)

ลดการเกิด Overfitting เพิ่มเติม

(8) Softmax Output Layer

แปลงค่าที่ได้จาก Fully Connected Layer เป็น ค่าความน่าจะเป็น ใช้เลือก คลาสของข้อความ ที่มีค่าความน่าจะเป็นสูงสุด

การประเมินประสิทธิภาพ

Evaluation:

- ใช้ข้อมูล evaluation ที่เตรียมไว้ในขั้นตอน pre-processing.

- ประเมินประสิทธิภาพของโมเดลที่ถูก fine-tune โดยใช้ตัววัดที่เหมาะสม (accuracy,loss, val_accuracy, val_loss).

การประเมินประสิทธิภาพด้วย DSM-5

BEGIN

Prepare Data:

- Convert `Created-At` column to datetime format
- Sort data by `Created-At`
- Merge predictions with original dataset

Process Data in 14-Day Intervals:

- Set `start_date` to the earliest `Created-At`
- Set `end_date` to `start_date + 14 days`
- WHILE `start_date` is within dataset range:
  - Extract rows where `Created-At` falls within the current 14-day interval
  - Group predictions by date and store them in a dictionary
  - Save interval summary (start date, end date, daily predictions)
  - Shift `start_date` and `end_date` forward by 14 days

Calculate Depression Score for Each Interval:

- FOR each 14-day interval:
  - Initialize `total_score = 0`
  - Create a counter to track occurrences of each symptom
  - FOR each day in interval:
    - COUNT occurrences of each predicted class
  - FOR each symptom type:
    - Convert the symptom occurrence count into a score:
      - 2-4 days → score = 1
      - 6-8 days → score = 2
      - 10-14 days → score = 3
      - Otherwise → score = 0
    - ADD score to `total_score`

- Determine depression level based on `total_score`:
  - `<7` → Normal (1)
  - `8-12` → Mild depression (2)
  - `13-18` → Moderate depression (3)
  - `>18` → Severe depression (4)
- Store results for this interval

Display Results:

- FOR each 14-day interval:
  - PRINT start date, end date
  - PRINT total depression score
  - PRINT corresponding depression level

END

(1) เตรียมข้อมูล (Data Preparation)

สร้าง DataFrame test_result

นำค่าที่โมเดลพยากรณ์ (test_predict) มาแปลงเป็นคลาสที่คาดการณ์ (Class Predict)

เปรียบเทียบผลลัพธ์กับค่าจริง (Actuals)

รวม test_result กับข้อมูลต้นฉบับ df

เพิ่มผลพยากรณ์ (Predict และ Class Predict) ลงใน DataFrame

แปลง Created-At เป็น Datetime format และเรียงลำดับตามวันที่

(2) แบ่งข้อมูลออกเป็นช่วงเวลา 14 วัน (Processing 14-Day Intervals)

กำหนดช่วงเวลาเริ่มต้น (start_date) และสิ้นสุด (end_date)

start_date = วันที่ที่เก่าที่สุดในข้อมูล

end_date = start_date + 14 วัน

วนลูปแบ่งข้อมูลออกเป็นช่วงๆ

ดึงข้อมูลที่อยู่ในช่วงวันที่กำหนด (Created-At อยู่ระหว่าง start_date และ end_date)

นับจำนวนครั้งที่แต่ละคลาสถูกพยากรณ์ในแต่ละวัน

เก็บผลลัพธ์ไว้ใน dictionary (Daily_Results)

ขยับช่วงเวลาไปข้างหน้า (เลื่อนไป 14 วันถัดไป)

(3) คำนวณคะแนนภาวะซึมเศร้า (Depression Score Calculation)

วนลูปทุกช่วงเวลา 14 วัน

ใช้ Counter() เพื่อนับจำนวนวันที่แต่ละคลาสถูกพยากรณ์แปลงจำนวนวันที่พยากรณ์เป็นคะแนน

หากอาการเกิดขึ้น 2-4 วัน ⟶ score = 1

หากอาการเกิดขึ้น 6-8 วัน ⟶ score = 2

หากอาการเกิดขึ้น 10-14 วัน ⟶ score = 3

หากอาการเกิดขึ้น ต่ำกว่า 2 วัน หรือ มากกว่า 8 วัน ⟶ score = 0

รวมคะแนนทั้งหมด (Total Score) ในช่วงเวลา 14 วัน

ถ้าคะแนนรวม < 7 ⟶ ระดับ ปกติ (Normal)

ถ้าคะแนนรวม 8-12 ⟶ ระดับ ซึมเศร้าเล็กน้อย (Mild Depression)

ถ้าคะแนนรวม 13-18 ⟶ ระดับ ซึมเศร้าปานกลาง (Moderate Depression)

ถ้าคะแนนรวม >18 ⟶ ระดับ ซึมเศร้ารุนแรง (Severe Depression)

(4) แสดงผลลัพธ์ (Display Results)

วนลูปแสดงผลข้อมูลแต่ละช่วง 14 วัน แสดง คะแนนรวม (Total Score) และระดับภาวะซึมเศร้า (Depression Level)

บทที่ 4
ผลการวิจัยและการอภิปราย

ในส่วนของงานวิจัยนี้ได้มีจุดมุ่งหมายในการจัดทำเพื่อทำการจำแนกประเภทของบุคคลที่มีโอกาสเกิดโรคซึมเศร้า โดยทำการจำแนกจากการวิเคราะห์ข้อมูลของตัวบุคคลที่ทำการโพส โดยใช้วิธีการทาง NLP มาช่วยในการวิเคราะห์โดยได้ผลการทดลอง เป็นผลการดำเนินงานวิจัย และทำการอภิปรายการทดลอง ดังนี้

ผลการจัดเตรียมข้อมูล

หลังจากทำการเตรียมข้อมูล (Preprocessing Data) จากชุดข้อมูลของ Imbalance ทั้งหมดจำนวน 35,000 ข้อความแล้ว เหลือข้อความที่ใช้ได้จำนวน 18,805ข้อความ คิดเป็น 56.37% ของข้อมูลทั้งหมดและส่วนของ Balance ทั้งหมดจำนวน 30,000 ข้อความแล้ว เหลือข้อความที่ใช้ได้จำนวน 15,698 ข้อความ คิดเป็น 52.32% ของข้อมูลทั้งหมด นอกจากนี้ ได้ทำการเพิ่มชุดข้อมูลด้วยวิธีการขยายข้อมูล (Data Augmentation) ในหมวดอาการที่ 2 โดยเพิ่มข้อความจำนวน 924 ข้อความของ Imbalance และข้อความจำนวน 924 ข้อความของ Balance เพื่อปรับปรุงประสิทธิภาพของโมเดลที่ใช้ จากนั้นได้ทำการฝึกสอนโมเดล ได้แก่ Transformer, LSTM และ CNN ด้วยชุดข้อมูลที่เตรียมไว้ ดังนี้

- ชุดข้อมูลที่ทำการเพิ่มจำนวนชุดข้อมูลของส่วนของ Imbalance (Data augmentation) จำนวน 19,729 ข้อความ โดยแบ่งเป็นชุดข้อมูลที่ให้โมเดลใช้ในการเรียนรู้ 13,810 ข้อความ ชุดข้อมูลทดสอบ 2,959 ข้อความ ชุดข้อมูลตรวจสอบ 2,960 ข้อความ

ตารางที่ 17 จำนวนคำในแต่ละประเภทหลังจากการลบข้อมูลของ Imbalance

ตัวเลขแสดงประเภทอาการ	ข้อมูลเดิม	ข้อมูลที่เพิ่ม	ข้อมูลหลังจากเพิ่ม
0	3945	0	3945
1	989	0	989
2	323	647	970
3	946	0	946
4	1212	0	1212
5	1144	0	1144
6	1026	0	1029
7	1300	0	1300
8	1278	0	1278
9	1049	0	1049
รวม	13212	647	13859

ตารางที่ 18 จำนวนข้อมูลที่ใช้ในการตรวจสอบของ Imbalance

ตัวเลขแสดงประเภทอาการ	ข้อมูลเดิม	ข้อมูลที่เพิ่ม	ข้อมูลหลังจากเพิ่ม
0	845	0	845
1	201	0	201
2	69	139	208
3	203	0	203
4	260	0	260
5	245	0	245
6	221	0	221
7	278	0	278
8	274	0	274
9	225	0	225
รวม	2821	139	2960

นอกจากนี้ ยังมีการนำชุดข้อมูลจาก Chat-GPT ซึ่งเป็นข้อความจำลองสถานการณ์อาการต่างๆ มาใช้เพื่อเปรียบเทียบประสิทธิภาพของโมเดลเพิ่มเติมจากข้อมูลที่มีอยู่แล้ว โดยจะมีชุดข้อมูลทดสอบทั้งหมด 3 ชุด ดังนี้:

- ชุดข้อมูลทดสอบจากข้อมูลเดิม จำนวน 2,821 ข้อความ
- ชุดข้อมูลทดสอบจากการขยายชุดข้อมูล (Data Augmentation) จำนวน 2,960 ข้อความ
- ชุดข้อมูลทดสอบจาก Chat-GPT จำนวน 200 ข้อความ

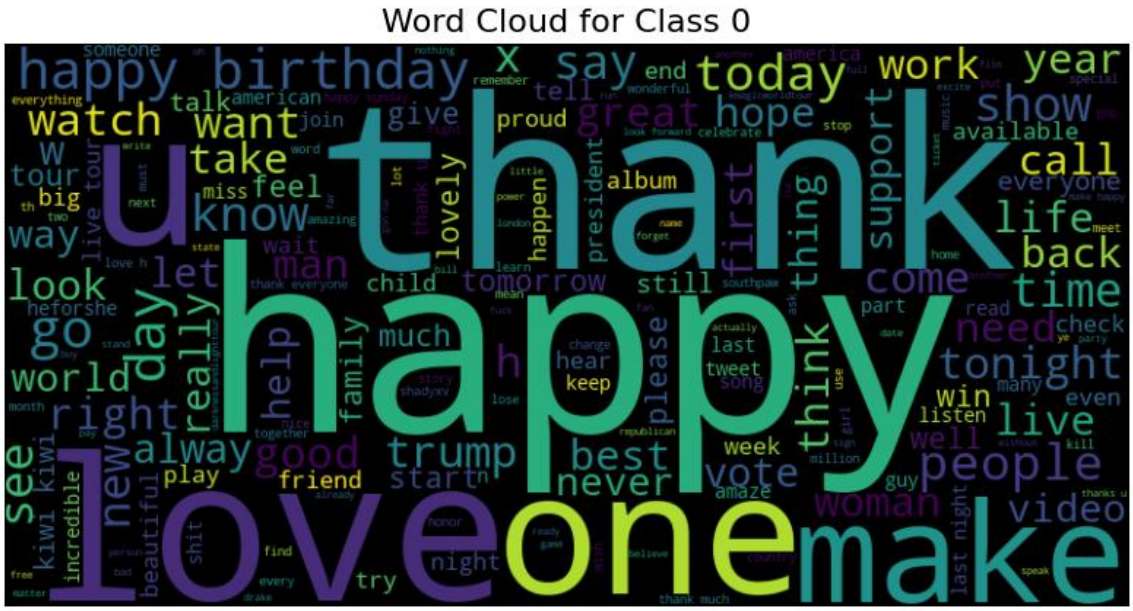
ดังนั้น จำนวนข้อความที่ใช้สำหรับการทดสอบทั้งหมดจะแสดงดังในตาราง

ตารางที่ 19 จำนวนข้อมูลที่ใช้ในการตรวจสอบทั้งหมดของ Imbalance

ตัวเลขแสดงประเภทอาการ	ข้อมูลเดิม	ข้อมูลหลังจากเพิ่ม	ข้อมูลจากChat-GPT
0	845	845	20
1	201	201	20
2	69	208	20
3	203	203	20
4	260	260	20
5	245	245	20
6	221	221	20
7	278	278	20
8	274	274	20
9	225	225	20
รวม	2821	2960	200

การทำแผนภาพคำ (Word Cloud)

แผนภาพคำถูกสร้างขึ้นเพื่อแสดงคำที่มีความถี่สูงในรูปแบบกราฟิก ซึ่งช่วยให้เห็นภาพรวมของข้อมูลคำได้อย่างชัดเจนและเข้าใจง่าย คำที่ปรากฏบ่อยจะถูกแสดงด้วยขนาดใหญ่กว่าคำที่ปรากฏน้อยกว่า โดยทำการสร้างแผนภาพคำสำหรับชุดข้อมูลของแต่ละประเภทอาการทั้ง 10 ประเภทดังนี้

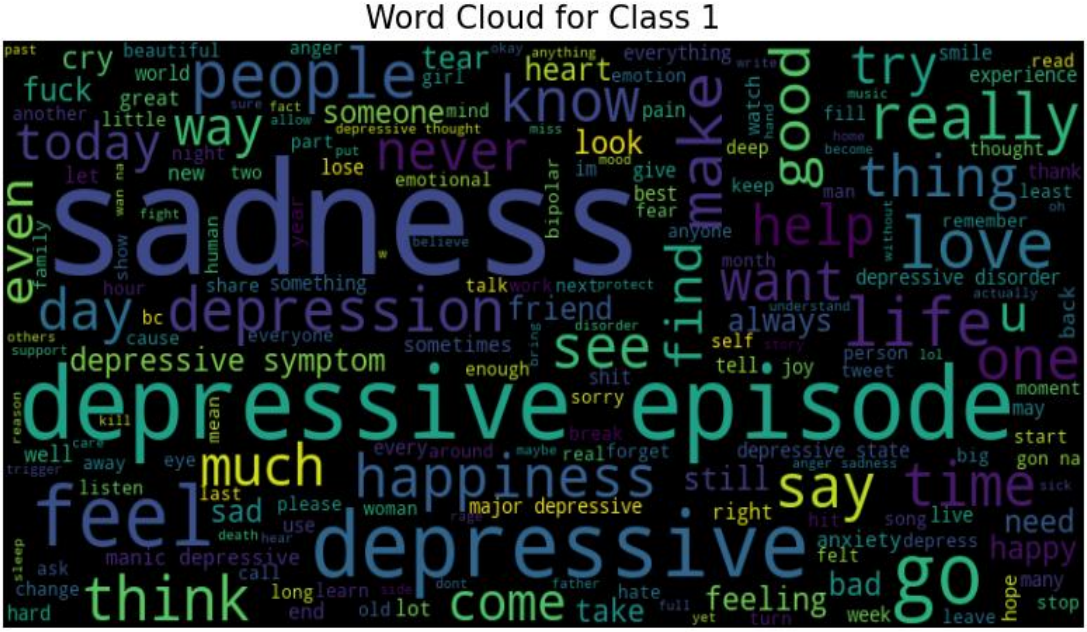


ภาพที่ 20 Word Cloud ของอาการปกติของ Imbalance

(1) ข้อมูลประเภทอาการปกติ จากรูปที่ 20 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการปกติได้แก่

ตารางที่ 20 จำนวนข้อมูลในคลาสอาการปกติของ Imbalance

ลำดับที่	คำ	จำนวน
1	happy	838
2	thank	522
3	love	417
4	u	407
5	get	330
6	make	296
7	see	268
8	birthday	257
9	like	257
10	one	246

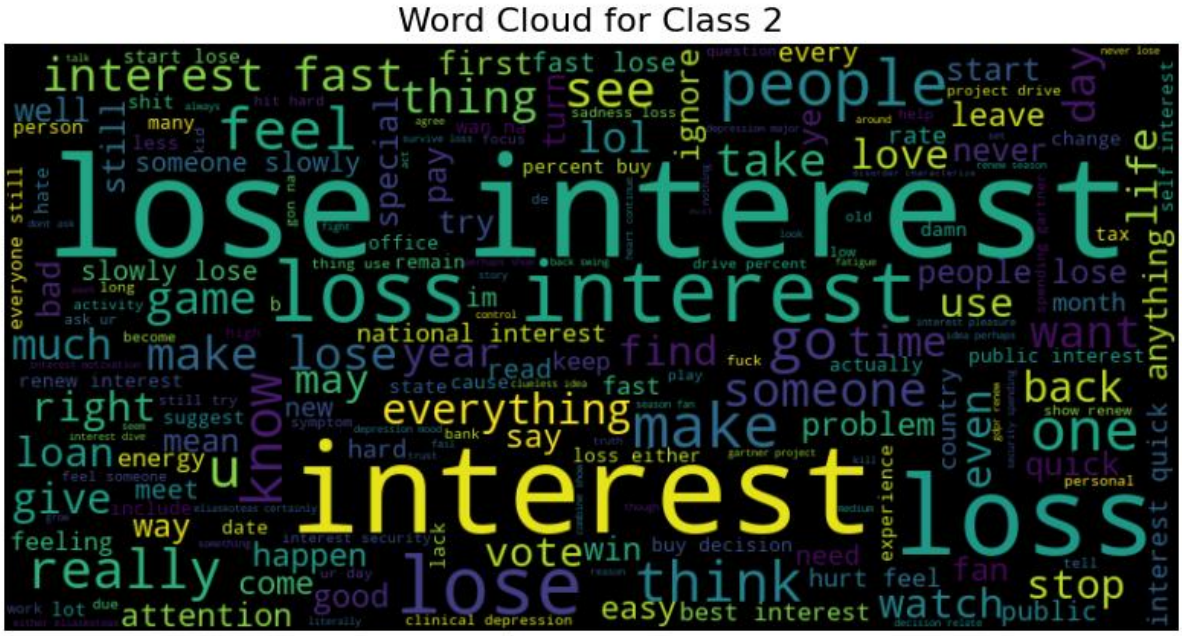


ภาพที่ 21 Word Cloud ของอารมณ์ซึมเศร้าของ Imbalance

(2) ข้อมูลประเภทอารมณ์ซึมเศร้า จากรูปที่ 21 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการอารมณ์ซึมเศร้าได้แก่

ตารางที่ 21 จำนวนข้อมูลในคลาสอารมณ์ซึมเศร้าของ Imbalance

ลำดับที่	คำ	จำนวน
1	sadness	684
2	depressive	552
3	episode	200
4	like	147
5	feel	145
6	go	126
7	get	123
8	life	97
9	love	92
10	make	90



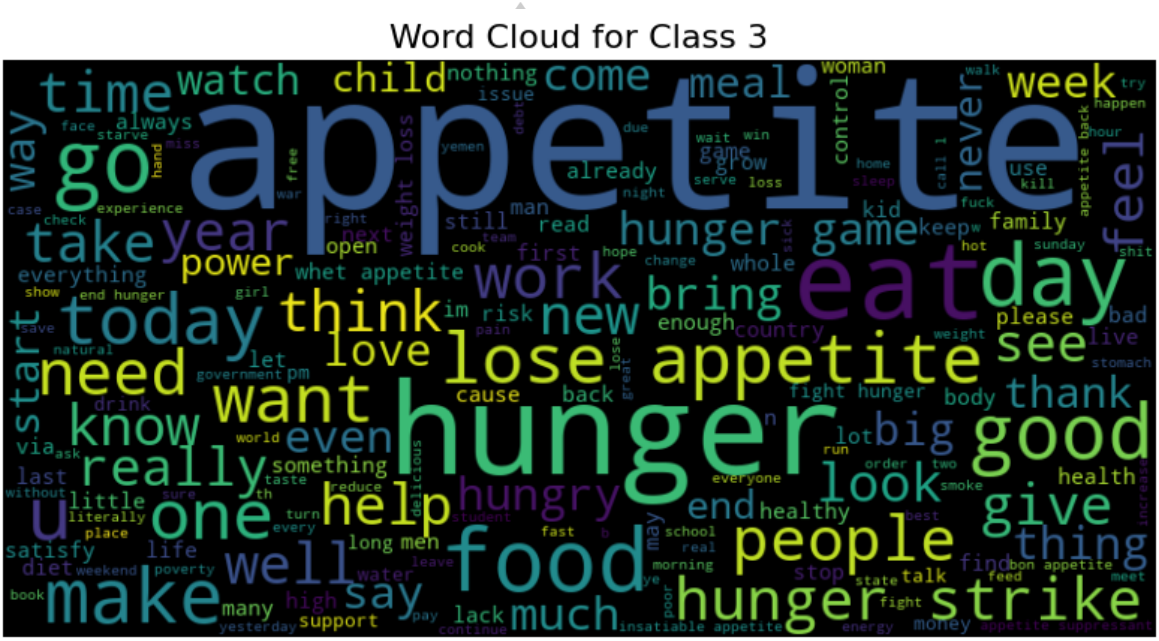
ภาพที่ 22 Word Cloud ของการขาดความสนใจลดลงของ Imbalance

(3) ข้อมูลประเภทอารมณ์ซึมเศร้า จากรูปที่ 22 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการขาดความสนใจได้แก่

ตารางที่ 22 จำนวนข้อมูลในคลาสการขาดความสนใจลดลงของ Imbalance

ลำดับที่	คำ	จำนวน
1	interest	1274
2	lose	789
3	loss	543
4	get	166
5	make	149
6	people	138
7	like	134
8	feel	123
9	thing	105
10	go	102

ภาพที่ 23 Word Cloud ของการที่น้ำหนักผิดปกติของ Imbalance

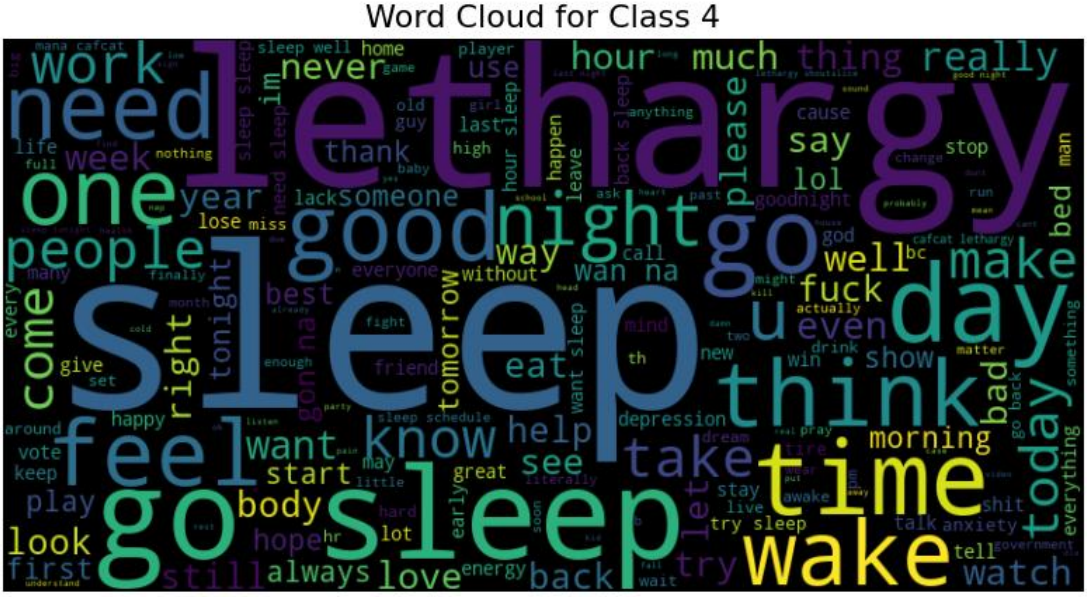


(4) ข้อมูลประเภทน้ำหนักผิดปกติจากรูปที่ 23 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการน้ำหนักผิดปกติได้แก่

ตารางที่ 23 จำนวนข้อมูลในคลาสน้ำหนักผิดปกติของ Imbalance

ลำดับที่	คำ	จำนวน
1	appetite	839
2	hunger	360
3	eat	152
4	get	112
5	food	97
6	lose	91
7	day	90
8	go	89
9	like	86
10	u	72

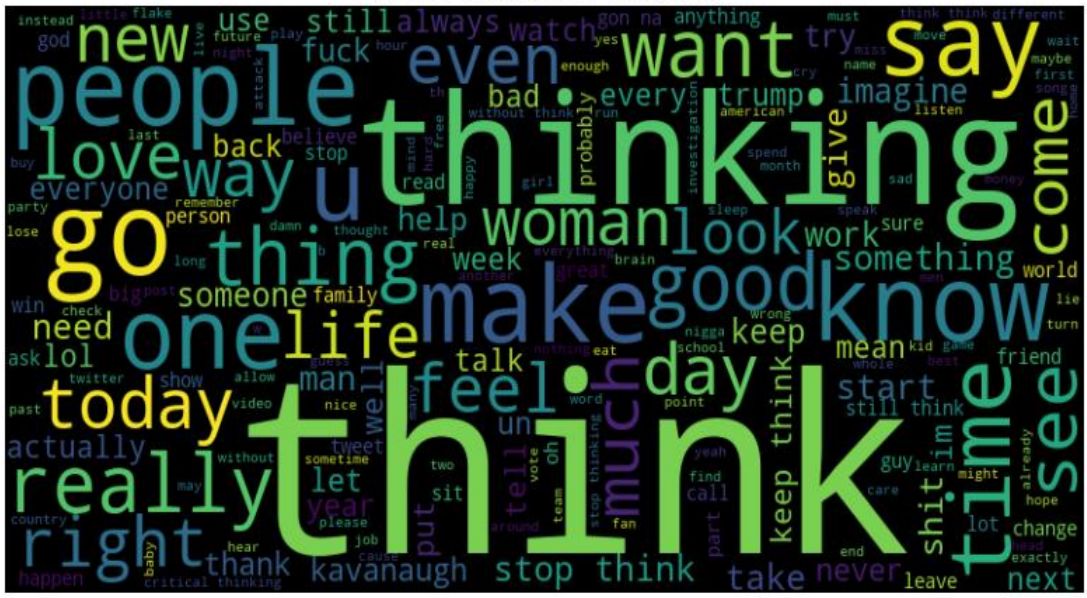
ภาพที่ 24 Word Cloud ของการนอนผิดปกติของ Imbalance



(5) ข้อมูลประเภทการนอนผิดปกติ จากรูปที่ 24 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการนอนผิดปกติ ได้แก่

ตารางที่ 24 จำนวนข้อมูลในคลาสการนอนผิดปกติของ Imbalance

ลำดับที่	คำ	จำนวน
1	sleep	1043
2	lethargy	551
3	go	291
4	get	228
5	like	134
6	night	110
7	day	110
8	need	95
9	time	94
10	good	92

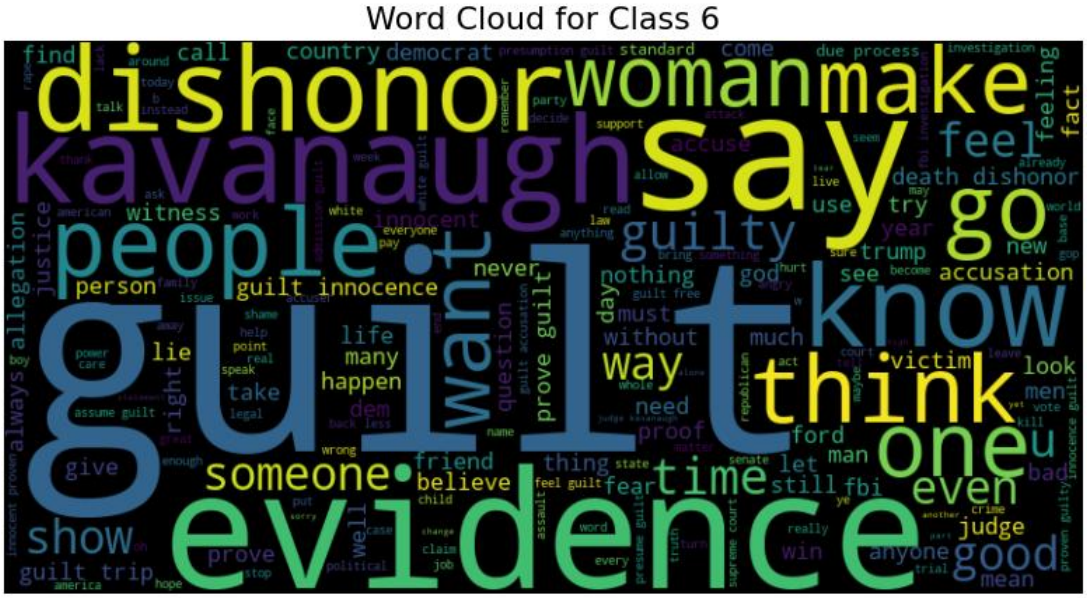


ภาพที่ 25 Word Cloud ของร่างกายอ่อนเพลียของ Imbalance

(6) ข้อมูลประเภทร่างกายอ่อนเพลีย จากรูปที่ 25 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการร่างกายอ่อนเพลีย ได้แก่

ตารางที่ 25 จำนวนข้อมูลในคลาสร่างกายอ่อนเพลียของ Imbalance

ลำดับที่	คำ	จำนวน
1	think	1298
2	thinking	315
3	get	213
4	like	175
5	go	139
6	make	115
7	know	113
8	people	109
9	u	102
10	say	96

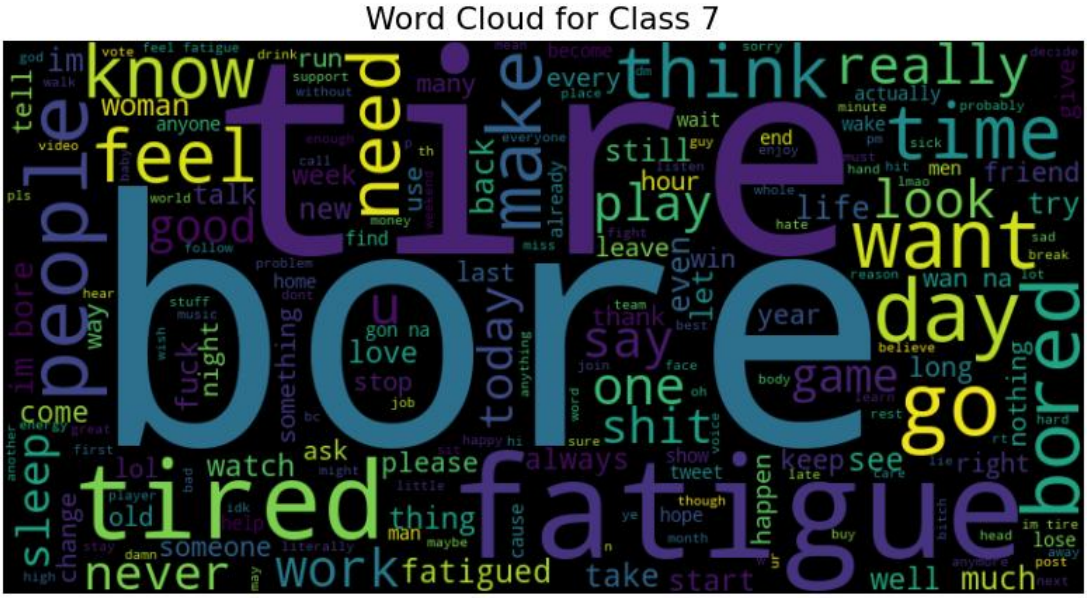


ภาพที่ 26 Word Cloud ของการรู้สึกตนเองไร้ค่าของ Imbalance

(7) ข้อมูลประเภทการรู้สึกตนเองไร้ค่า จากรูปที่ 26 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการรู้สึกตนเองไร้ค่า ได้แก่

ตารางที่ 26 จำนวนข้อมูลในคลาสการรู้สึกตนเองไร้ค่าของ Imbalance

ลำดับที่	คำ	จำนวน
1	guilt	1098
2	dishonor	193
3	evidence	153
4	kavanaugh	141
5	like	129
6	say	126
7	innocence	124
8	get	122
9	would	113
10	people	111

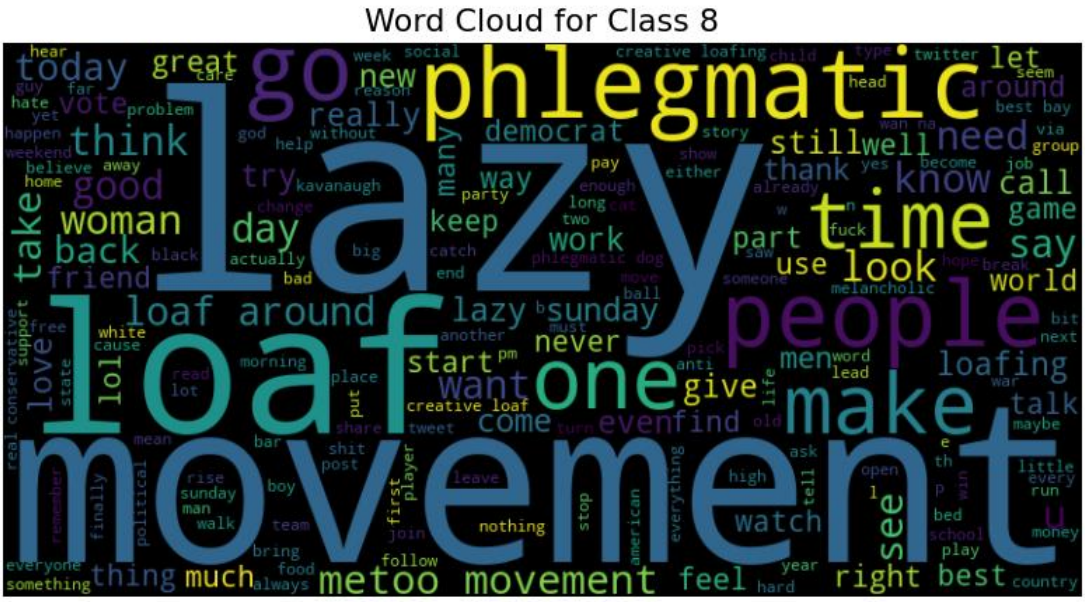


ภาพที่ 27 Word Cloud ของการรู้สึกสมาธิสั้นของ Imbalance

(8) ข้อมูลประเภทการรู้สึกสมาธิสั้น จากรูปที่ 27 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการสมาธิสั้น ได้แก่

ตารางที่ 27 จำนวนข้อมูลในคลาสสมาธิสั้นของ Imbalance

ลำดับที่	คำ	จำนวน
1	bore	630
2	tire	469
3	get	414
4	fatigue	392
5	tired	194
6	like	159
7	go	126
8	im	121
9	time	121
10	people	113

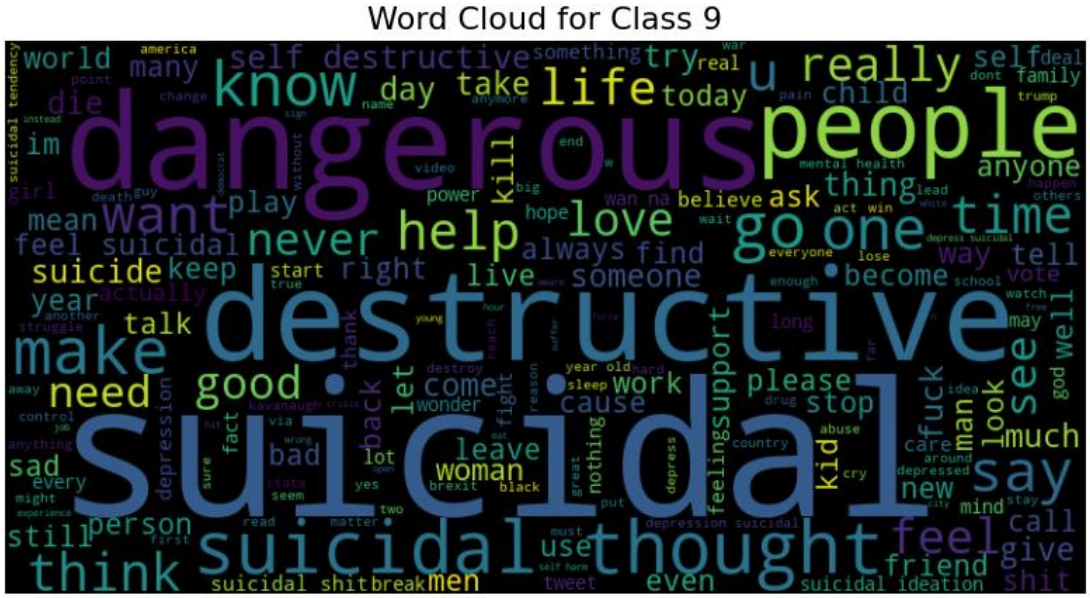


ภาพที่ 28 Word Cloud ของการเคลื่อนไหวช้าของ Imbalance

(9) ข้อมูลประเภทการเคลื่อนไหวช้าจากรูปที่ 28 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการเคลื่อนไหวช้า ได้แก่

ตารางที่ 28 จำนวนข้อมูลในคลาสการเคลื่อนไหวช้าของ Imbalance

ลำดับที่	คำ	จำนวน
1	movement	450
2	lazy	450
3	loaf	371
4	get	193
5	phlegmatic	142
6	like	128
7	loafing	116
8	go	111
9	people	109
10	around	108



ภาพที่ 29 Word Cloud ของการคิดฆ่าตัวตายของ Imbalance

(10) ข้อมูลประเภทการคิดฆ่าตัวตาย จากรูปที่ 29 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการคิดฆ่าตัวตาย ได้แก่

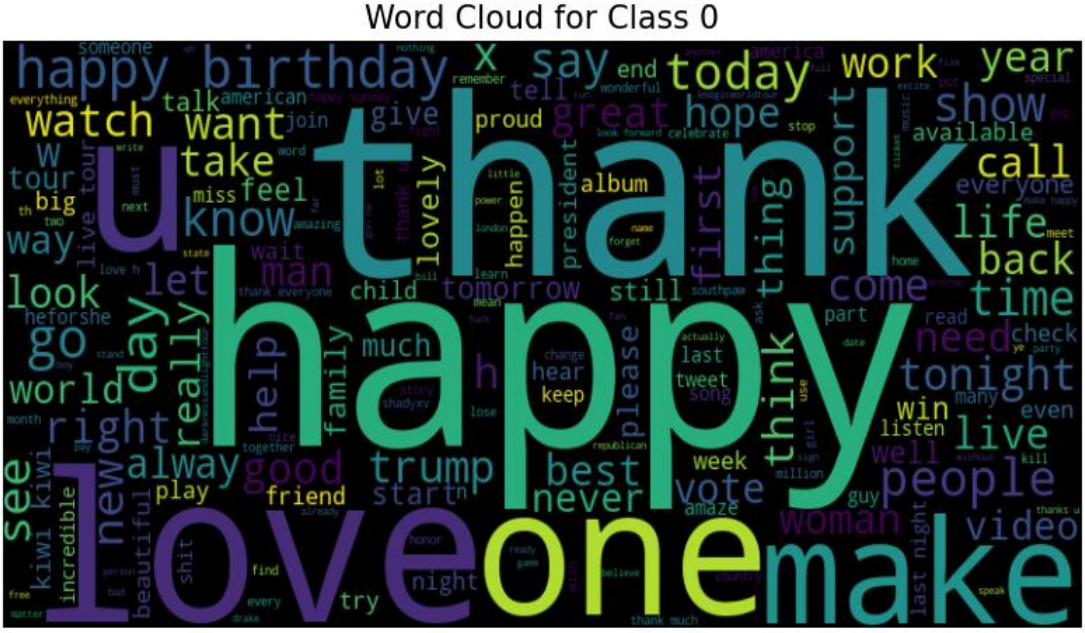
ตารางที่ 29 จำนวนข้อมูลในคลาสการคิดฆ่าตัวตายของ Imbalance

ลำดับที่	คำ	จำนวน
1	suicidal	795
2	destructive	255
3	dangerous	200
4	thought	153
5	people	145
6	like	132
7	go	120
8	make	119
9	get	118
10	self	106

- ชุดข้อมูลที่ทำการเพิ่มจำนวนชุดข้อมูลของส่วนของ Balance (Data augmentation) จำนวน 16,622ข้อความ โดยแบ่งเป็นชุดข้อมูลที่ให้โมเดลใช้ในการเรียนรู้ 11,355 ข้อความ ชุดข้อมูลทดสอบ 2,433 ข้อความ ชุดข้อมูลตรวจสอบ 2,434 ข้อความ

ตารางที่ 30 จำนวนข้อมูลที่ใช้ในการตรวจสอบทั้งหมดของ Balance

ตัวเลขแสดงประเภทอาการ	จำนวนข้อมูลเดิม	จำนวนข้อมูลหลังจากเพิ่ม	จำนวนข้อมูลจาก Chat-GPT
0	845	845	20
1	201	201	20
2	69	208	20
3	203	203	20
4	260	260	20
5	245	245	20
6	221	221	20
7	278	278	20
8	274	274	20
9	225	225	20
รวม	2821	2960	200

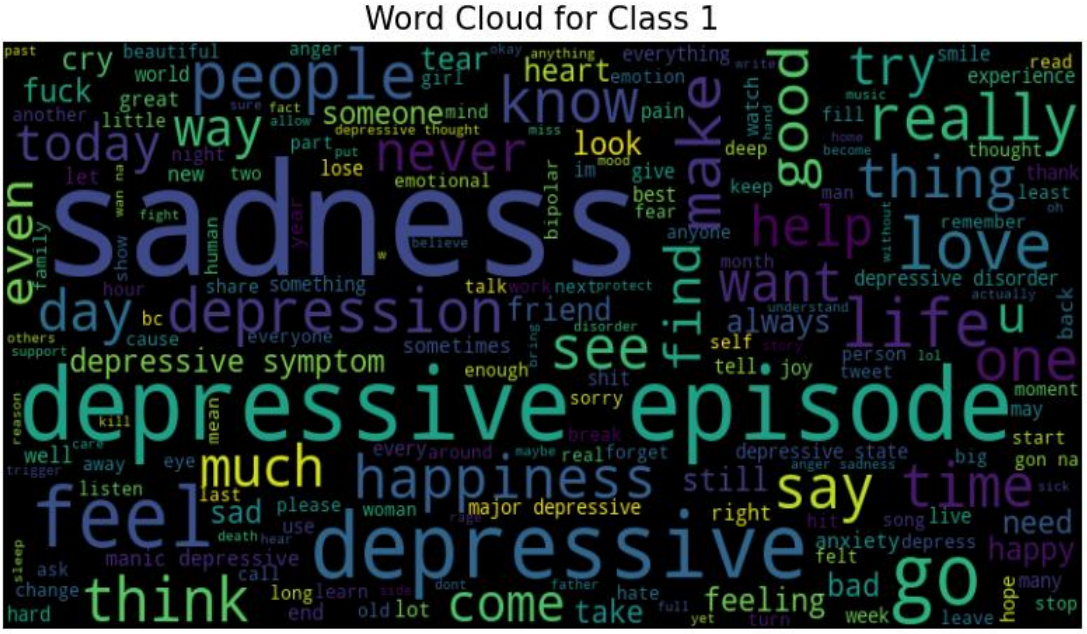


ภาพที่ 30 Word Cloud ของอาการปกติของ Balance

(1) ข้อมูลประเภทอาการปกติ จากรูปที่ 30 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการปกติได้แก่

ตารางที่ 31 จำนวนข้อมูลในคลาสอาการปกติของ Balance

ลำดับที่	คำ	จำนวน
1	happy	745
2	birthday	210
3	u	199
4	love	196
5	get	162
6	thank	152
7	make	150
8	day	140
9	one	127
10	see	121

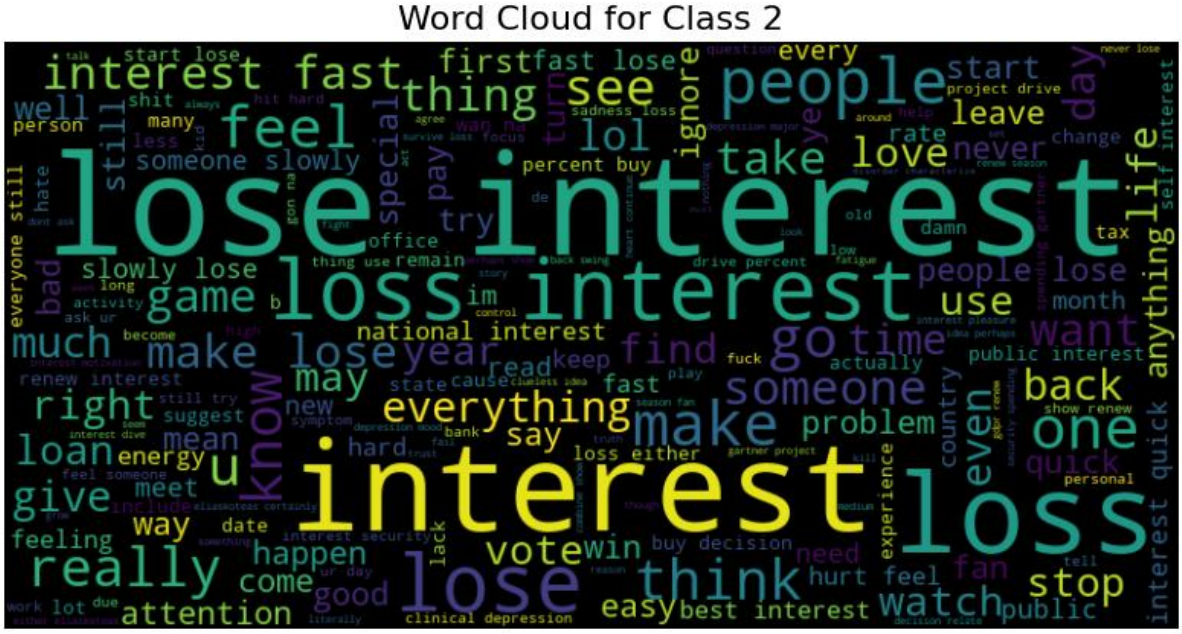


ภาพที่ 31 Word Cloud ของอารมณ์ซึมเศร้าของ Balance

(2) ข้อมูลประเภทอารมณ์ซึมเศร้า จากรูปที่ 31 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการอารมณ์ซึมเศร้าได้แก่

ตารางที่ 32 จำนวนข้อมูลในคลาสอารมณ์ซึมเศร้าของ Imbalance

ลำดับที่	คำ	จำนวน
1	sadness	684
2	depressive	552
3	episode	200
4	like	147
5	feel	145
6	go	126
7	get	123
8	life	97
9	love	92

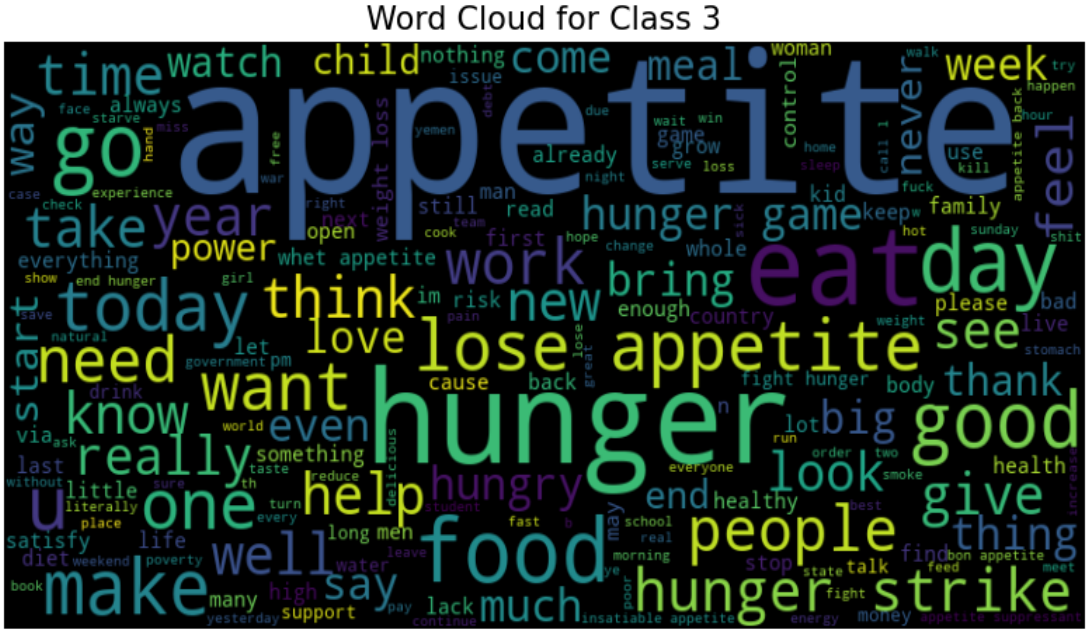


ภาพที่ 32 Word Cloud ของการขาดความสนใจลดลงของ Balance

(3) ข้อมูลประเภทอารมณ์ซึมเศร้า จากรูปที่ 32 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการขาดความสนใจได้แก่

ตารางที่ 33 จำนวนข้อมูลในคลาสการขาดความสนใจลดลงของ Imbalance

ลำดับที่	คำ	จำนวน
1	interest	1274
2	lose	789
3	loss	543
4	get	166
5	make	149
6	people	138
7	like	134
8	feel	123
9	thing	105
10	go	102

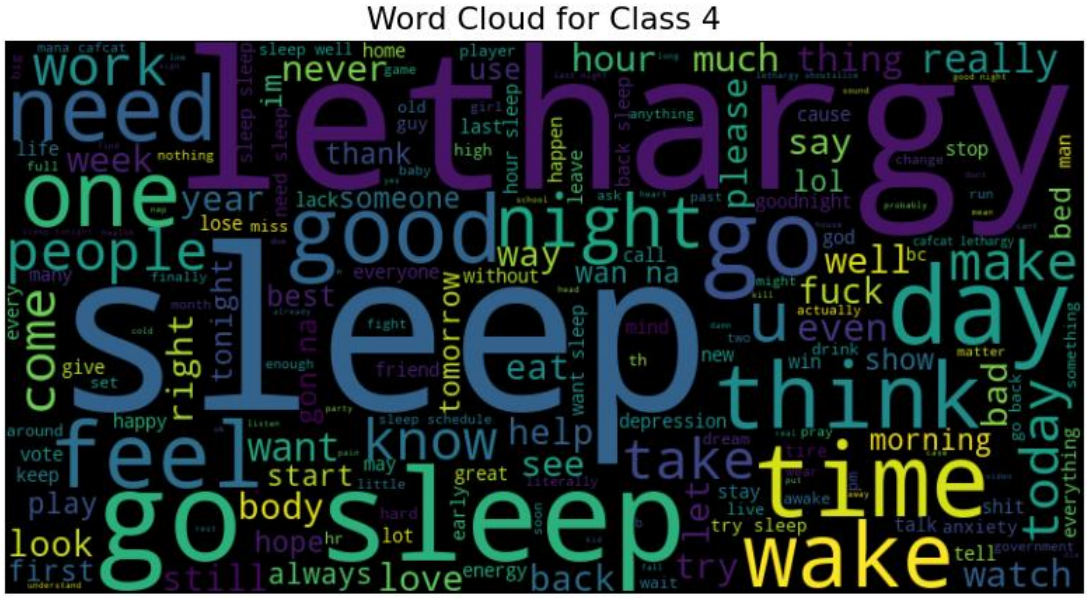


ภาพที่ 33 Word Cloud ของการที่น้ำหนักผิดปกติของ Balance

(4) ข้อมูลประเภทน้ำหนักผิดปกติจากรูปที่ 33 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการน้ำหนักผิดปกติได้แก่

ตารางที่ 34 จำนวนข้อมูลในคลาสน้ำหนักผิดปกติของ Balance

ลำดับที่	คำ	จำนวน
1	appetite	839
2	hunger	360
3	eat	152
4	get	112
5	food	97
6	lose	91
7	day	90
8	go	89
9	like	86
10	u	72

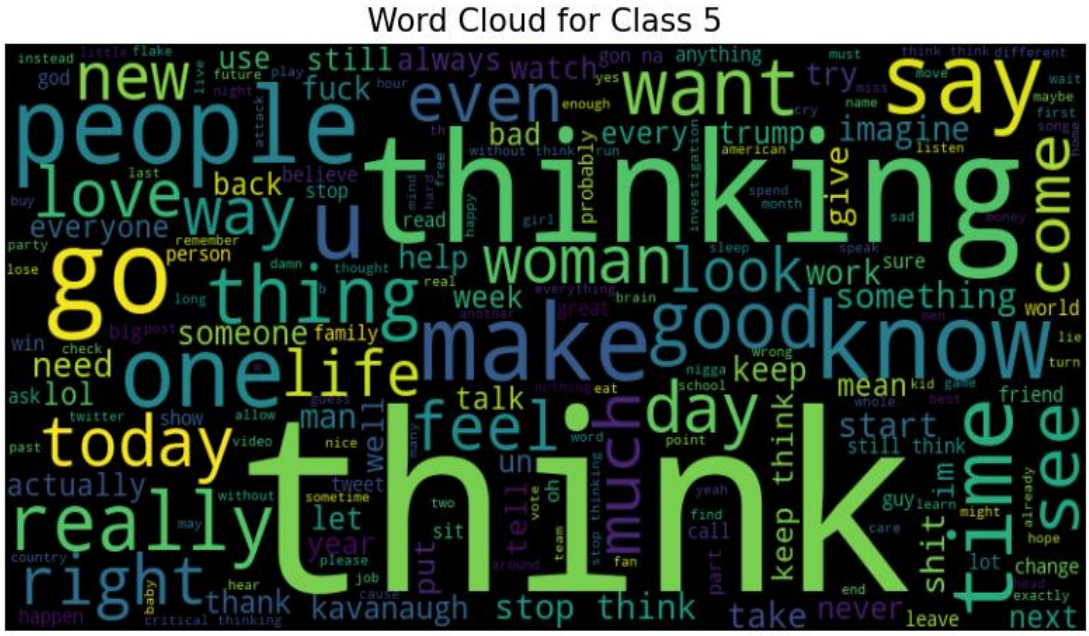


ภาพที่ 34 Word Cloud ของการนอนผิดปกติของ Balance

(5) ข้อมูลประเภทการนอนผิดปกติ จากรูปที่ 34 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการนอนผิดปกติ ได้แก่

ตารางที่ 35 จำนวนข้อมูลในคลาสการนอนผิดปกติของ Balance

ลำดับที่	คำ	จำนวน
1	sleep	1043
2	lethargy	551
3	go	291
4	get	228
5	like	134
6	night	110
7	day	110
8	need	95
9	time	94
10	good	92

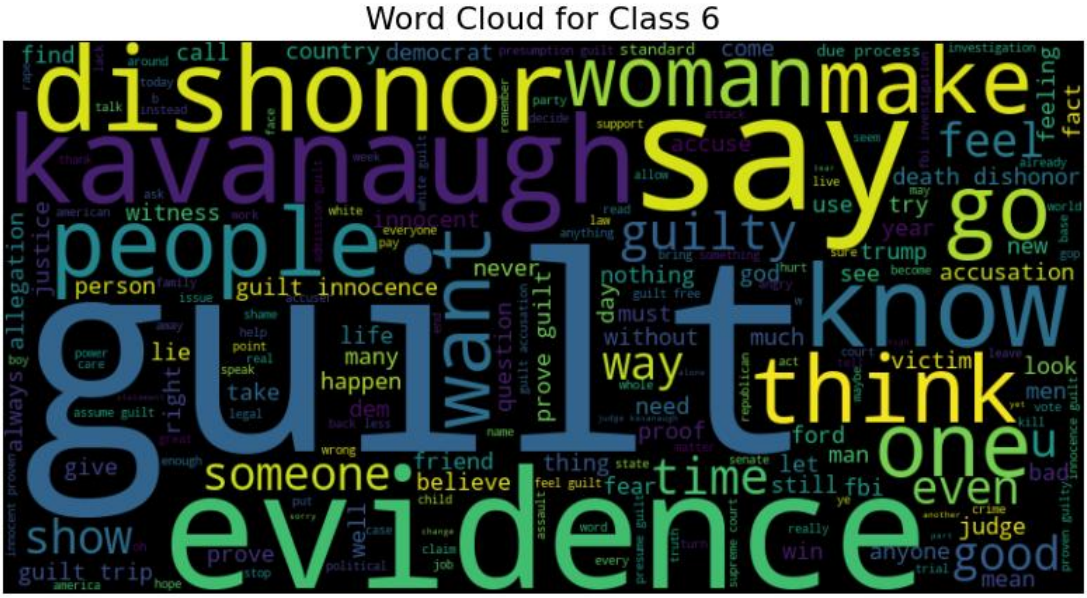


ภาพที่ 35 Word Cloud ของร่างกายอ่อนเพลียของ Balance

(6) ข้อมูลประเภทร่างกายอ่อนเพลีย จากรูปที่ 35 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการร่างกายอ่อนเพลีย ได้แก่

ตารางที่ 36 จำนวนข้อมูลในคลาสร่างกายอ่อนเพลียของ Balance

ลำดับที่	คำ	จำนวน
1	think	1298
2	thinking	315
3	get	213
4	like	175
5	go	139
6	make	115
7	know	113
8	people	109
9	u	102
10	say	96

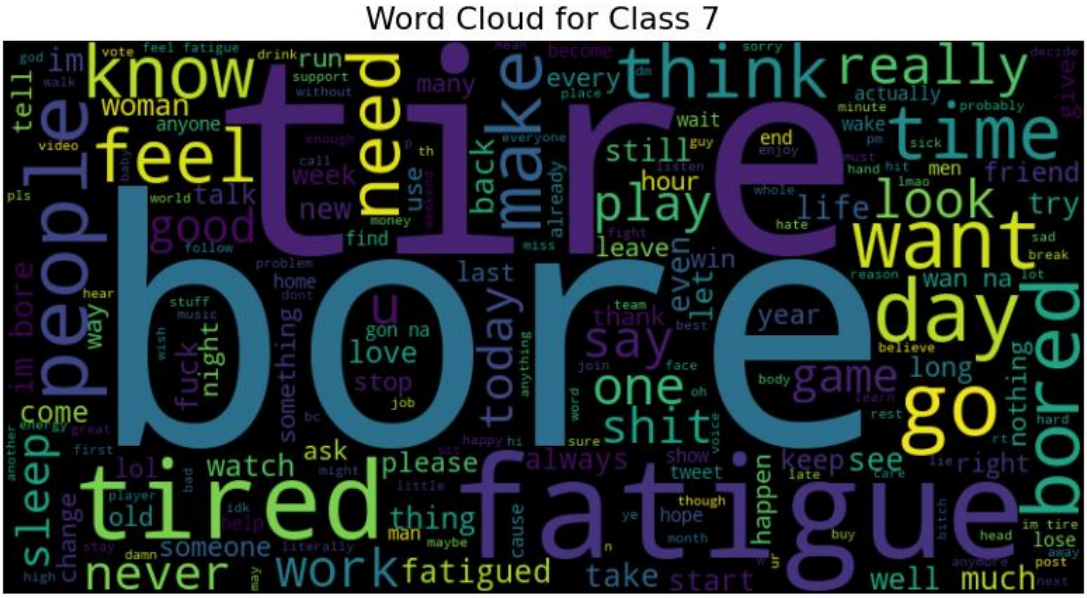


ภาพที่ 36 Word Cloud ของการรู้สึกตนเองไร้ค่าของ Balance

(7) ข้อมูลประเภทการรู้สึกตนเองไร้ค่า จากรูปที่ 36 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการรู้สึกตนเองไร้ค่า ได้แก่

ตารางที่ 37 จำนวนข้อมูลในคลาสการรู้สึกตนเองไร้ค่าของ Balance

ลำดับที่	คำ	จำนวน
1	guilt	1098
2	dishonor	193
3	evidence	153
4	kavanaugh	141
5	like	129
6	say	126
7	innocence	124
8	get	122
9	would	113
10	people	111

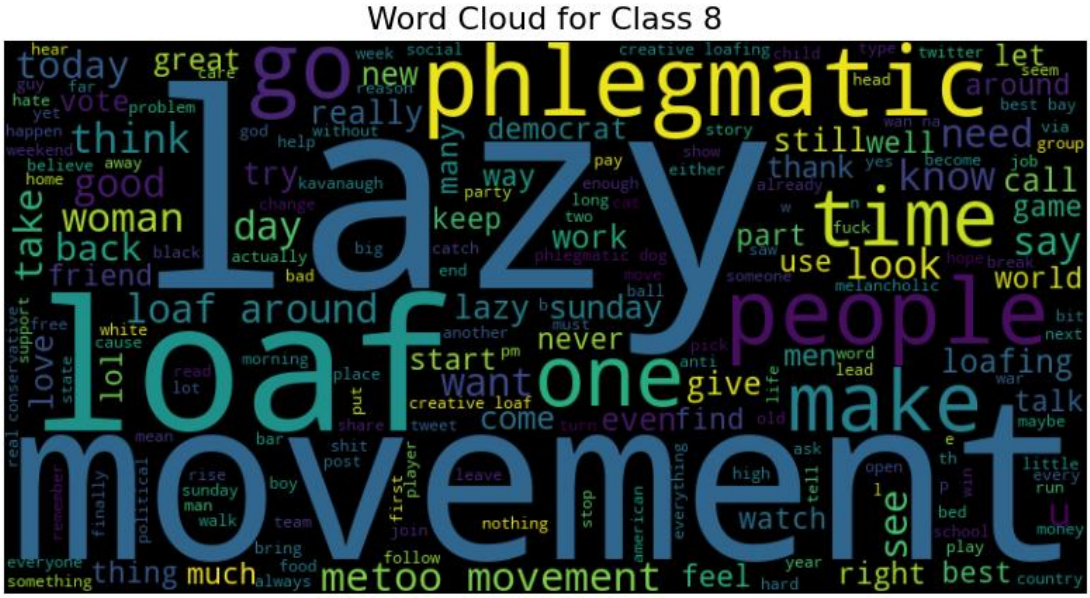


ภาพที่ 37 Word Cloud ของการรู้สึกสมาธิสั้นของ Balance

(8) ข้อมูลประเภทการรู้สึกสมาธิสั้น จากรูปที่ 37 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการสมาธิสั้น ได้แก่

ตารางที่ 38 จำนวนข้อมูลในคลาสสมาธิสั้นของ Balance

ลำดับที่	คำ	จำนวน
1	bore	630
2	tire	469
3	get	414
4	fatigue	392
5	tired	194
6	like	159
7	go	126
8	im	121
9	time	121
10	people	113

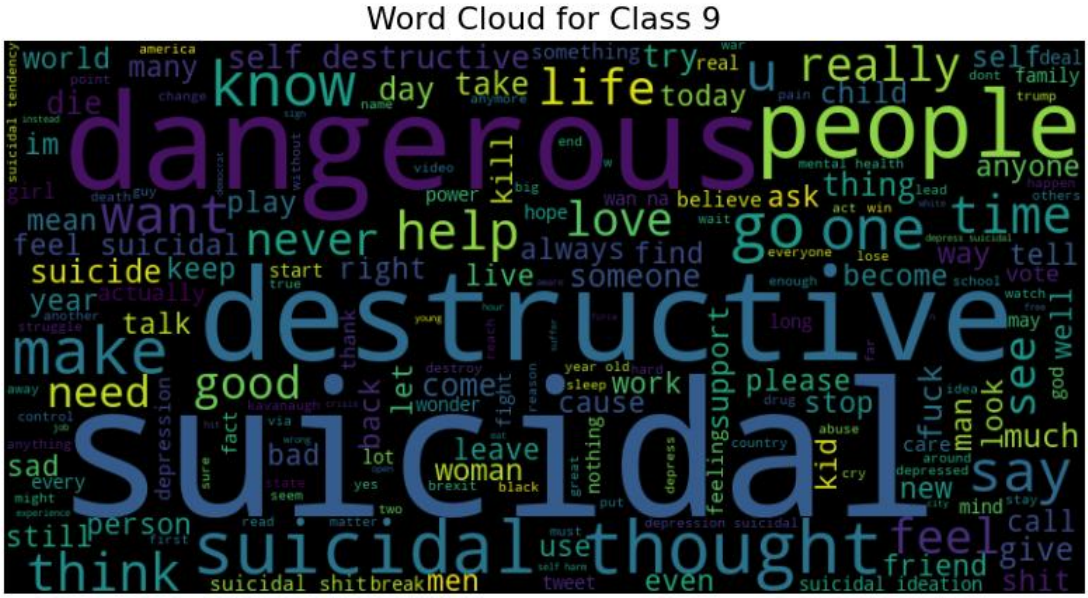


ภาพที่ 38 Word Cloud ของการเคลื่อนไหวช้าของ Balance

(9) ข้อมูลประเภทการเคลื่อนไหวช้าจากรูปที่ 38 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการเคลื่อนไหวช้า ได้แก่

ตารางที่ 39 จำนวนข้อมูลในคลาสการเคลื่อนไหวช้าของ Balance

ลำดับที่	คำ	จำนวน
1	movement	450
2	lazy	450
3	loaf	371
4	get	193
5	phlegmatic	142
6	like	128
7	loafing	116
8	go	111
9	people	109
10	around	108



ภาพที่ 39 Word Cloud ของการคิดฆ่าตัวตายของ Balance

(10) ข้อมูลประเภทการคิดฆ่าตัวตาย จากรูปที่ 39 แสดงให้เห็นถึงปริมาณของจำนวนคำที่ปรากฏมากที่สุดในอาการการคิดฆ่าตัวตาย ได้แก่

ตารางที่ 40 จำนวนข้อมูลในคลาสการคิดฆ่าตัวตายของ Balance

ลำดับที่	คำ	จำนวน
1	suicidal	795
2	destructive	255
3	dangerous	200
4	thought	153
5	people	145
6	like	132
7	go	120
8	make	119
9	get	118
10	self	106

จากตารางแสดงจำนวนคำที่ใช้บ่อยจะมีคำบางประเภทที่ไม่ได้แสดงถึงอาการในประเภทแต่ละประเภทเช่นคำว่า say ,go และ get ซึ่งอาจจะส่งผลถึงการทำนายและอาจจะทำให้คาดเคลื่อน ได้รวมถึงคำบางประเภทที่อาจอยู่ในรูปแบบย่อความเช่นคำว่า u ที่มาจาก you ก็อาจจะส่งผลกระทบเช่นกัน

ตารางที่ 41 จำนวนคำที่ปรากฏมากที่สุดในแต่ละคลาส

คลาส	คำ
0	happy
1	sadness
2	interest
3	appetite
4	sleep
5	think
6	guilt
7	bore
8	movement
9	suicidal

ผลการสร้างโมเดล

หลังจากทำการเตรียมข้อมูลแล้วในลำดับต่อไปจะเป็นขั้นตอนการจัดทำโมเดลซึ่งได้ผลการทดลองเป็นพารามิเตอร์ดังต่อไปนี้

ตารางที่ 42 พารามิเตอร์ของโมเดล Tranformer

เลเยอร์	พารามิเตอร์	พารามิเตอร์ที่กำหนด
Input	Input shape	300
Embedding	input dimension	25,507
	output dimension	300

เลเยอร์	พารามิเตอร์	พารามิเตอร์ที่กำหนด
	weights	embedding_matrix
Transformer	Embedding dimension	300
	Number of attention heads	6
	Feedforward network dimension	256
	Dropout rate	0.2
ในแต่ละบล็อกของ Transformer	Multi-Head Attention	Self-attention mechanism with residual connections
	Layer Normalization	Applied after attention and feedforward steps
	Feedforward Network	Dense layer with ReLU activation followed by another Dense layer
	Residual Connections	After both attention and feedforward layers
Global Average Pooling	Global Average Pooling	Reduces the sequence dimension by taking the average over the time steps of the Transformer block outputs
Dropout	Dropout rate	0.5
Output	Dense layer with softmax activation	10

ตารางที่ 43 พารามิเตอร์ของโมเดล CNN

เลเยอร์	พารามิเตอร์	พารามิเตอร์ที่กำหนด
Embedding	input size	25,507
	output dimension	300
	weights	embedding_matrix
Dropout	Dropout rate	0.5
Conv1D	filters	300
	Kernel	5
	Padding	input/output size the same
	Activation function	ReLU
GlobalMaxPooling1D	max-pooling	max-pooling across each filter
Dense	units	300
	Activation function	ReLU
Dropout	Dropout rate	0.5
Output	Number of units	10
	Activation function	Softmax

ตารางที่ 44 พารามิเตอร์ของโมเดล LSTM

เลเยอร์	พารามิเตอร์	พารามิเตอร์ที่กำหนด
Embedding	input size	25,507
	output dimension	300
	weights	embedding_matrix
SpatialDropout1D	Dropout rate	0.2
Bidirectional LSTM (First)	Units	300
	Dropout	0.2
	Return sequences	True
	Bidirectional	LSTM
Bidirectional LSTM (Second)	Units	300
	Bidirectional	LSTM
Dropout	Dropout rate	0.2
Dense	units	64
	Activation function	ReLU
Dropout	Dropout rate	0.2
Output	Number of units	10
	Activation function	Softmax

ผลการวัดประสิทธิภาพ

ผลการทดลองทั้งหมดที่เกิดจากการทดลองอัลกอริทึมทั้ง 3 แบบคือ Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) โดยทำการ

เรียนรู้จากชุดข้อมูล และทำการแบ่งข้อมูลโดย มีชุดข้อมูลเรียนรู้ร้อยละ 70 ชุดทดสอบร้อยละ 15 และชุดตรวจสอบร้อยละ 15 โดยได้ผลการทดลองดังนี้

ตารางที่ 45 ผลการวัดประสิทธิภาพโมเดล Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ของ Imbalance ที่เรียนรู้ด้วยชุดข้อมูลปกติ และทดสอบด้วยชุดข้อมูลปกติ

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
0 Happy	Precision	0.7597	0.7718	0.7740
	Recall	0.9688	0.9675	0.9591
	F1-score	0.8516	0.8587	0.8567
1 Depressive	Precision	0.9722	0.9476	0.9427
	Recall	0.8495	0.8786	0.8786
	F1-score	0.9067	0.9118	0.9095
2 Loss of interest	Precision	0.9352	0.9809	0.9505
	Recall	0.8978	0.9111	0.9378
	F1-score	0.9161	0.9447	0.9441
3 Appetite	Precision	1.0000	0.9938	1.0000
	Recall	0.8073	0.8385	0.8438
	F1-score	0.8934	0.9096	0.9153
4 Sleep	Precision	0.9537	0.9626	0.9575
	Recall	0.8374	0.8374	0.8252
	F1-score	0.8918	0.8957	0.8865
5 Thinking	Precision	0.8594	0.8770	0.8500
	Recall	0.9129	0.9170	0.9170

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
	F1-score	0.8853	0.8966	0.8822
6 Guilt	Precision	0.9951	0.9904	0.9406
	Recall	0.8826	0.9000	0.8957
	F1-score	0.9355	0.9431	0.9176
7 Tired	Precision	0.9848	0.9777	0.9615
	Recall	0.8810	0.8946	0.8503
	F1-score	0.9300	0.9343	0.9025
8 Movement	Precision	0.9598	0.9817	0.9683
	Recall	0.7847	0.7810	0.7810
	F1-score	0.8635	0.8699	0.8646
9 Suicidal	Precision	0.9837	0.9635	0.9722
	Recall	0.8265	0.8447	0.7991
	F1-score	0.8983	0.9002	0.8772
Accuracy		0.8861	0.8949	0.8858

ตารางที่ 46 ผลการวัดประสิทธิภาพโมเดล Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ของ Imbalance ที่เรียนรู้ด้วยชุดข้อมูลปกติ และทดสอบด้วยชุดข้อมูลจาก Chat-GPT โดยที่แต่ละคลาสที่ทำการทดสอบมีจำนวนคลาสละ 20 ชุดข้อความที่นำมาทดสอบ

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
0 Happy	Precision	0.4000	0.3846	0.3750
	Recall	1.0000	0.7500	0.7500
	F1-score	0.5714	0.5085	0.5000
	Precision	1.0000	0.8000	0.7143

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
1 Depressive	Recall	1.0000	1.0000	1.0000
	F1-score	1.0000	0.8889	0.8333
2 Loss of interest	Precision	0.9524	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000
	F1-score	0.9756	1.0000	1.0000
3 Appetite	Precision	1.0000	1.0000	1.0000
	Recall	0.9500	1.0000	1.0000
	F1-score	0.9744	1.0000	1.0000
4 Sleep	Precision	0.9091	0.9167	0.8462
	Recall	0.5000	0.5500	0.5500
	F1-score	0.6452	0.6875	0.6667
5 Thinking	Precision	1.0000	1.0000	1.0000
	Recall	0.7000	0.7000	0.7000
	F1-score	0.8235	0.8235	0.8235
6 Guilt	Precision	1.0000	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000
	F1-score	1.0000	1.0000	1.0000
7 Tired	Precision	0.5833	0.6923	0.6364
	Recall	0.3500	0.4500	0.3500
	F1-score	0.4375	0.5455	0.4516
8 Movement	Precision	0.9333	1.0000	1.0000
	Recall	0.7000	0.7000	0.7000

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
	F1-score	0.8000	0.8235	0.8235
9 Suicidal	Precision	1.0000	0.8696	0.9500
	Recall	0.9000	1.0000	0.9500
	F1-score	0.9474	0.9302	0.9500
Accuracy		0.8100	0.8150	0.8000

ตารางที่ 47 ผลการวัดประสิทธิภาพโมเดล Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ของ Balance ที่เรียนรู้ด้วยชุดข้อมูลปกติ และทดสอบด้วยชุดข้อมูลปกติ

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
0 Happy	Precision	0.6033	0.6504	0.6974
	Recall	0.9534	0.9301	0.9016
	F1-score	0.7390	0.7655	0.7864
1 Depressive	Precision	0.9492	0.9769	0.9819
	Recall	0.9130	0.9185	0.8859
	F1-score	0.9307	0.9468	0.9314
2 Loss of interest	Precision	1.0000	0.9810	0.9593
	Recall	0.8789	0.9238	0.9507
	F1-score	0.9356	0.9515	0.9550
3 Appetite	Precision	0.9600	0.9698	0.9512
	Recall	0.8930	0.8977	0.9070
	F1-score	0.9253	0.9324	0.9286
	Precision	0.9505	0.9364	0.8939

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
4 Sleep	Recall	0.8373	0.8770	0.8690
	F1-score	0.8903	0.9057	0.8813
5 Thinking	Precision	0.9336	0.8975	0.8765
	Recall	0.8242	0.8555	0.8594
	F1-score	0.8755	0.8760	0.8679
6 Guilt	Precision	0.8950	0.9372	0.9323
	Recall	0.8565	0.8565	0.8565
	F1-score	0.8753	0.8950	0.8928
7 Tired	Precision	0.9762	0.9570	0.9416
	Recall	0.9044	0.9007	0.8897
	F1-score	0.9389	0.9280	0.9149
8 Movement	Precision	0.9776	0.9821	0.8952
	Recall	0.8195	0.8271	0.8346
	F1-score	0.8916	0.8980	0.8638
9 Suicidal	Precision	0.9947	0.9135	0.8995
	Recall	0.8087	0.8261	0.8174
	F1-score	0.8921	0.8676	0.8565
Accuracy		0.8724	0.8829	0.8777

ตารางที่ 48 ผลการวัดประสิทธิภาพโมเดล Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ของ Balance ที่เรียนรู้ด้วยชุดข้อมูลปกติ และทดสอบด้วยชุดข้อมูลจาก Chat-GPT โดยที่แต่ละคลาสที่ทำการทดสอบมีจำนวนคลาสละ 20 ชุดข้อความที่นำมาทดสอบ

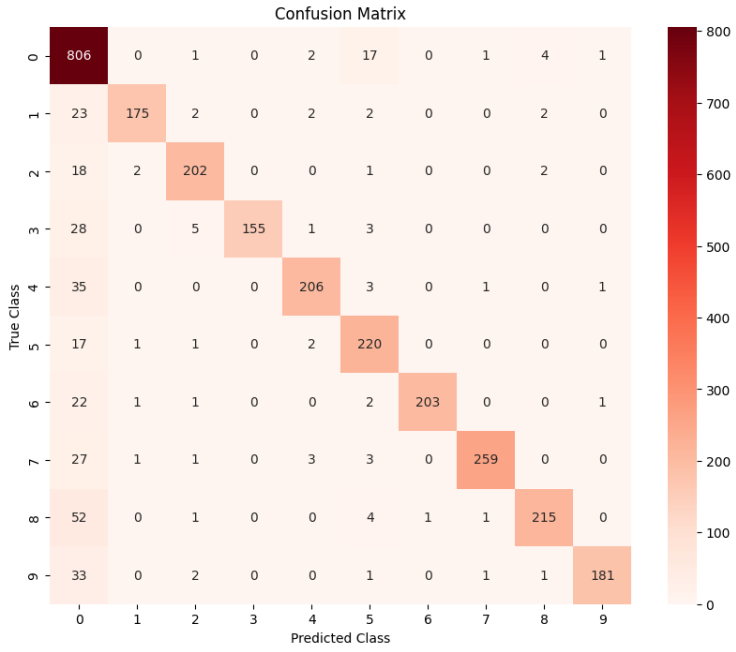
Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
0 Happy	Precision	0.4082	0.4737	0.6316
	Recall	1.0000	0.4500	0.6000
	F1-score	0.5797	0.4615	0.6154
1 Depressive	Precision	1.0000	0.6452	0.8333
	Recall	1.0000	1.0000	1.0000
	F1-score	1.0000	0.7843	0.9091
2 Loss of interest	Precision	1.0000	0.9091	0.9524
	Recall	1.0000	1.0000	1.0000
	F1-score	1.0000	0.9524	0.9756
3 Appetite	Precision	0.9524	0.9091	0.8333
	Recall	1.0000	1.0000	1.0000
	F1-score	0.9756	0.9524	0.9091
4 Sleep	Precision	0.9091	0.6875	0.6500
	Recall	0.5000	0.5500	0.6500
	F1-score	0.6452	0.6111	0.6500
5 Thinking	Precision	1.0000	1.0000	1.0000
	Recall	0.6000	0.7000	0.8000
	F1-score	0.7500	0.8235	0.8889
6 Guilt	Precision	0.8000	1.0000	0.9524
	Recall	1.0000	1.0000	1.0000
	F1-score	0.8889	1.0000	0.9756

Class	ประสิทธิภาพ	Tranformer	CNN	LSTM
7 Tired	Precision	0.5385	0.5000	0.5000
	Recall	0.3500	0.4000	0.3500
	F1-score	0.4242	0.4444	0.4118
8 Movement	Precision	1.0000	1.0000	1.0000
	Recall	0.6500	0.6500	0.6500
	F1-score	0.7879	0.7879	0.7879
9 Suicidal	Precision	0.9375	0.7407	0.7143
	Recall	0.7500	1.0000	1.0000
	F1-score	0.8333	0.8511	0.8333
Accuracy		0.7850	0.7750	0.8000

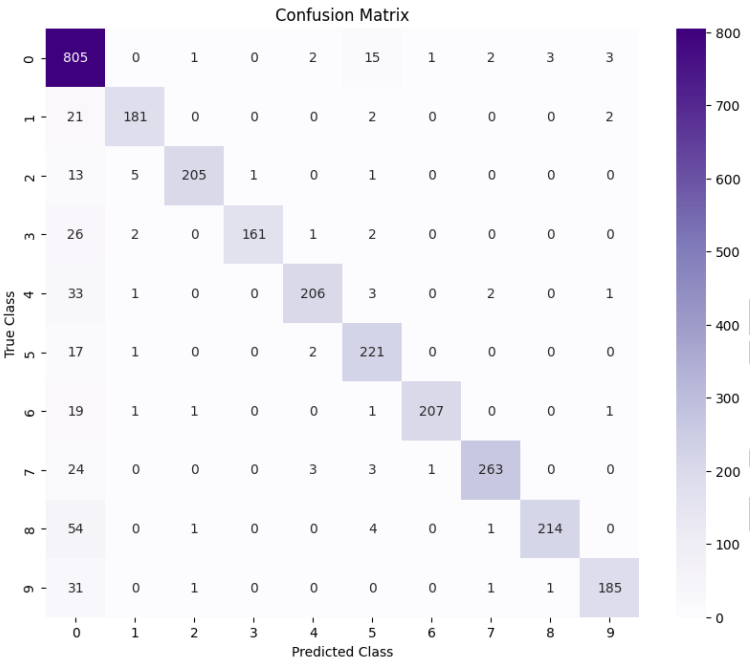
ผลการวัดประสิทธิภาพด้วย Confusion Matrix

ผลการวัดประสิทธิภาพของ Confusion Matrix

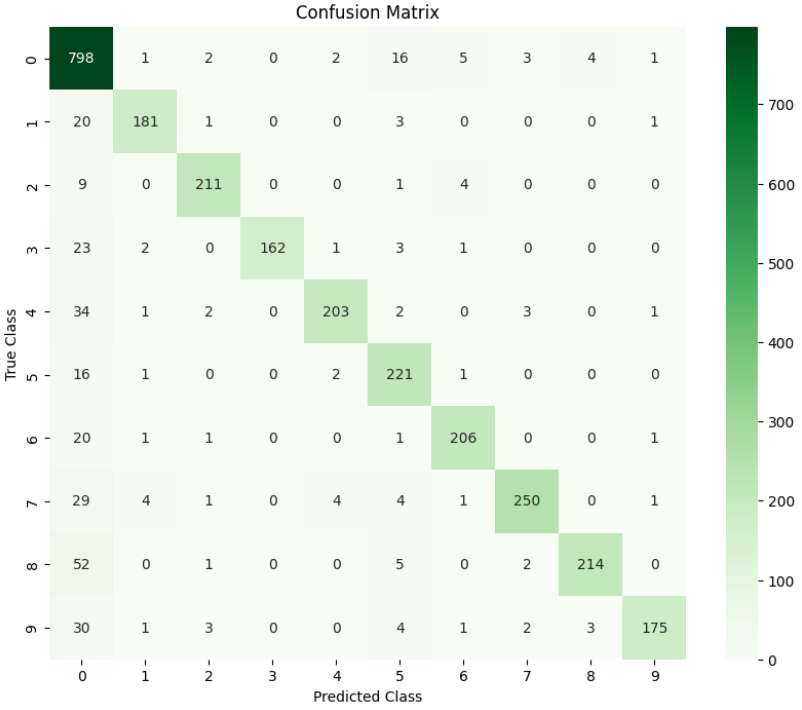
1) ทดสอบด้วยชุดข้อมูลทดสอบจากชุดข้อมูลปกติของ Imbalance จำนวน 2,959 ข้อความ



ภาพที่ 40 Confusion Matrix ของโมเดล Tranformer ของ Imbalance

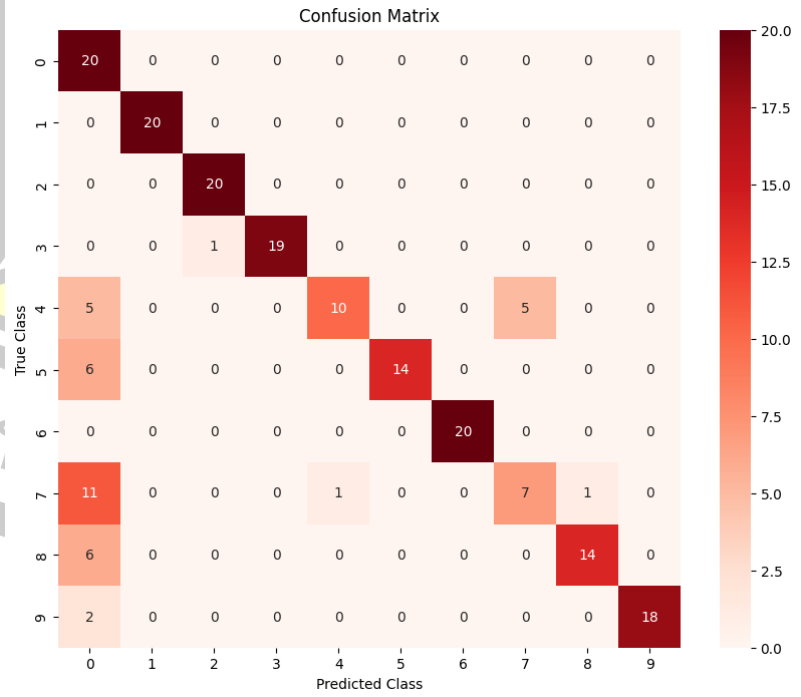


ภาพที่ 41 Confusion Matrix ของโมเดล CNN ของ Imbalance

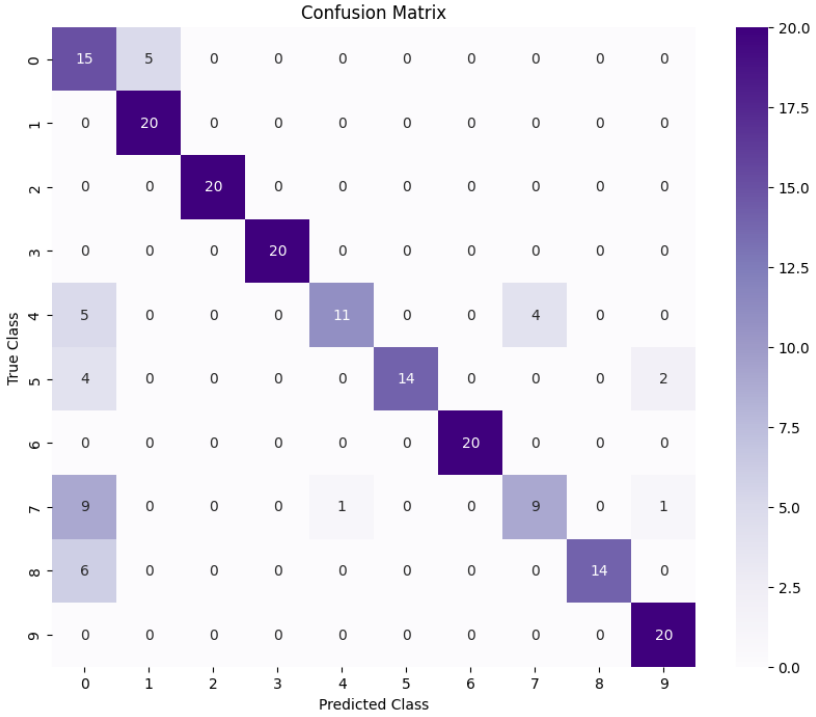


ภาพที่ 42 Confusion Matrix ของโมเดล LSTM ของ Imbalance

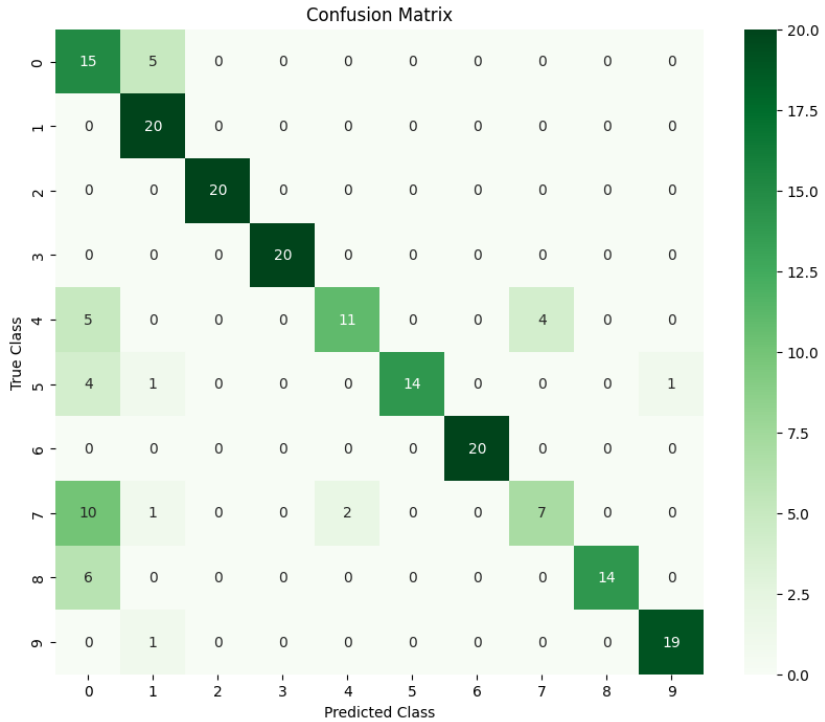
2) ทดสอบด้วยชุดข้อมูลทดสอบของ Imbalance จากชุดข้อมูลจาก Chat-GPT จำนวน 200 ข้อความ



ภาพที่ 43 Confusion Matrix ของโมเดล Tranformer ของ Imbalance

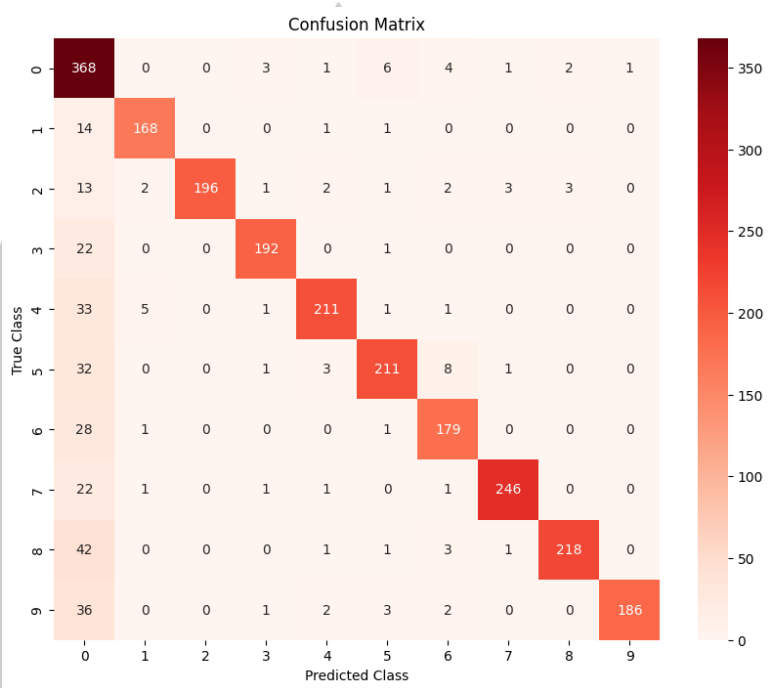


ภาพที่ 44 Confusion Matrix ของโมเดล CNN ของ Imbalance

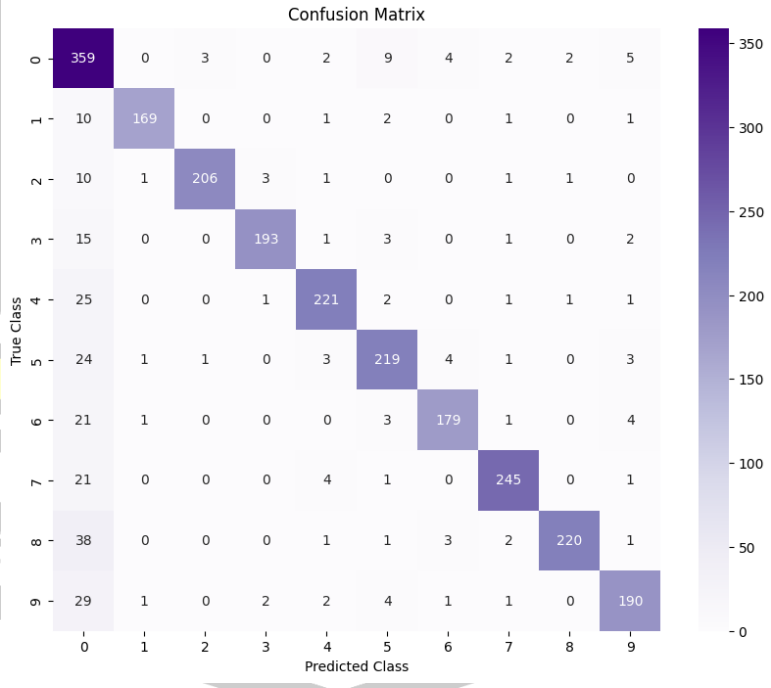


ภาพที่ 45 Confusion Matrix ของโมเดล LSTM ของ Imbalance

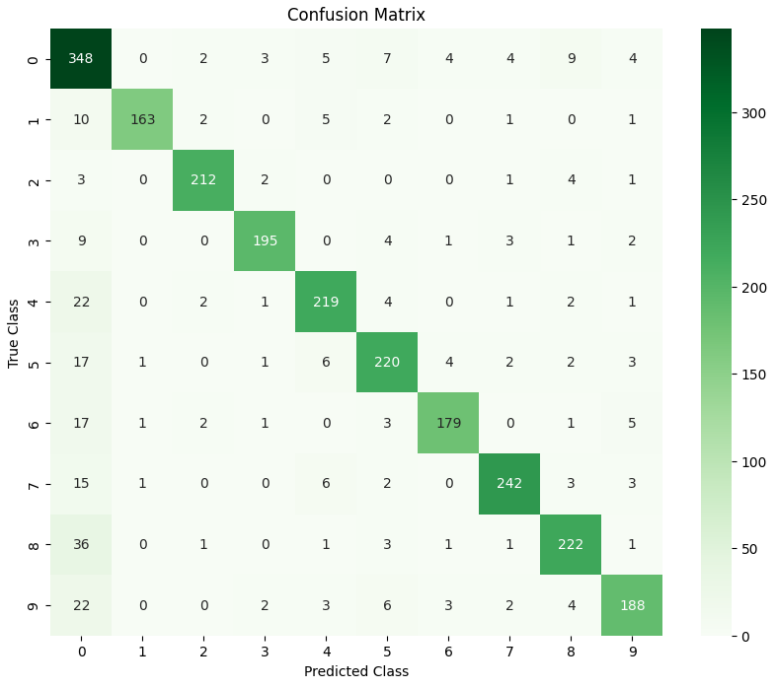
3) ทดสอบด้วยชุดข้อมูลทดสอบจากชุดข้อมูลปกติของ Imbalance จำนวน 2,959 ข้อความ



ภาพที่ 46 Confusion Matrix ของโมเดล Tranformer ของ Balance

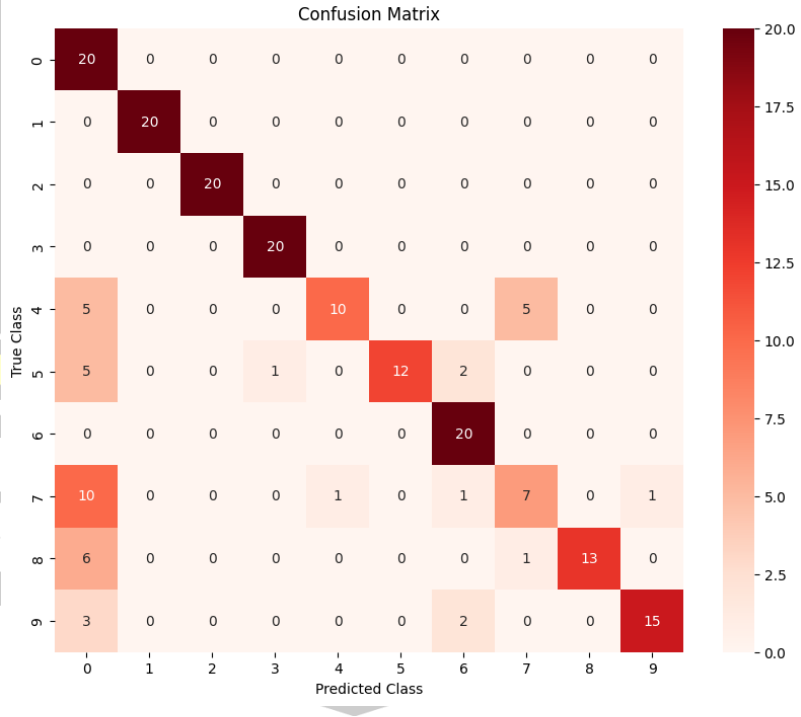


ภาพที่ 47 Confusion Matrix ของโมเดล CNN ของ Balance

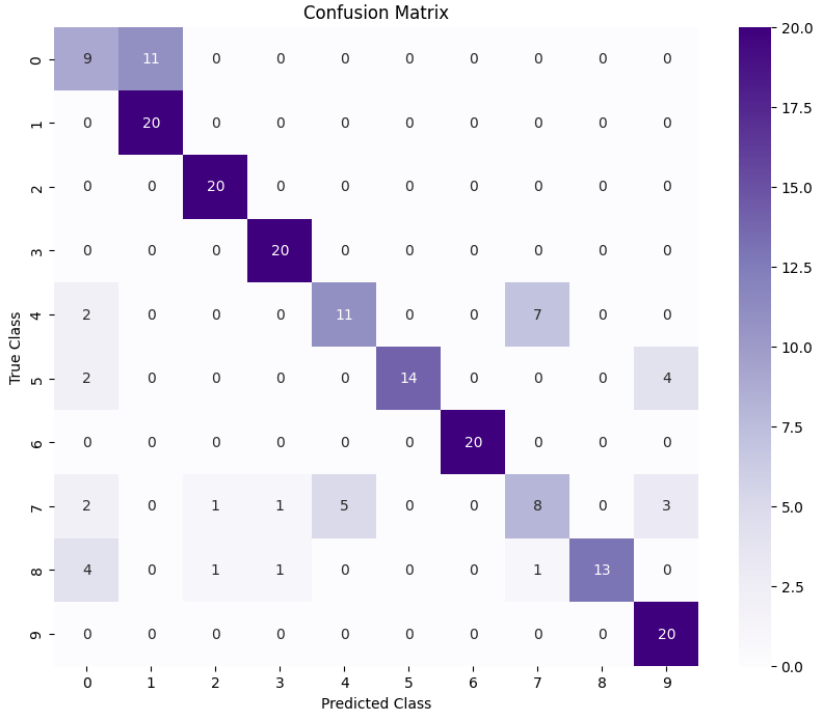


ภาพที่ 48 Confusion Matrix ของโมเดล LSTM ของ Balance

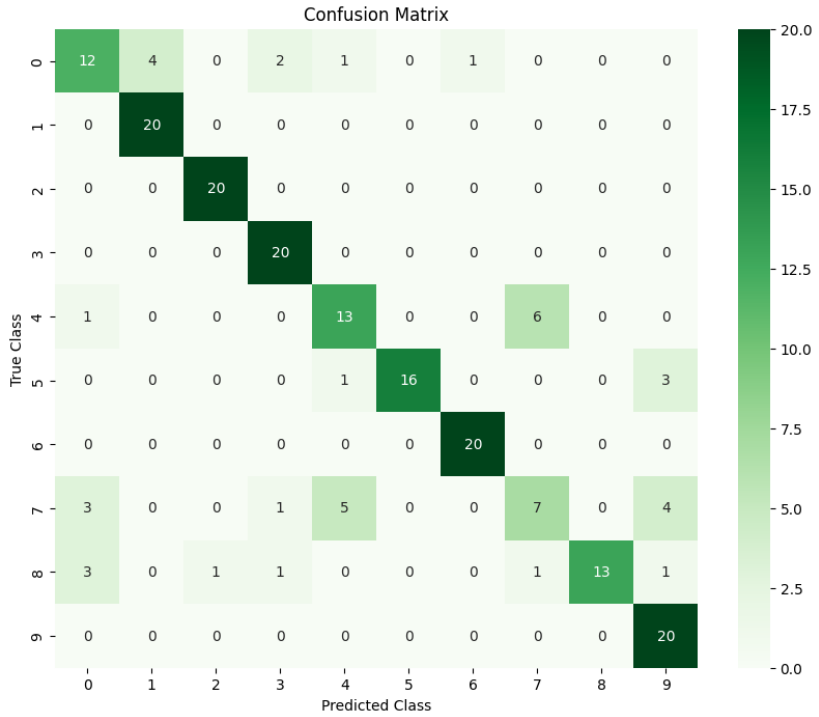
4) ทดสอบด้วยชุดข้อมูลทดสอบของ Balance จากชุดข้อมูลจาก Chat-GPT จำนวน 200 ข้อความ



ภาพที่ 49 Confusion Matrix ของโมเดล Tranformer ของ Balance



ภาพที่ 50 Confusion Matrix ของโมเดล CNN ของ Balance



ภาพที่ 51 Confusion Matrix ของโมเดล LSTM ของ Balance

วิเคราะห์ประสิทธิภาพของ Confusion Matrix

ตารางที่ 49 ผลการทดสอบโมเดล Imbalance ที่เรียนรู้ด้วยชุดข้อมูลปกติจำนวน 2,959 ข้อความ

คลาส	จำนวนที่ทดสอบ	จำนวนที่ถูกต้อง		
		Tranformer	CNN	LSTM
0	832	806	805	798
1	206	175	181	181
2	225	202	205	211
3	192	155	161	162
4	246	206	206	203
5	241	220	221	221
6	230	203	207	206
7	294	259	263	250
8	274	215	214	214
9	219	181	185	175
รวม	2,959	2,622	2,648	2,621

ตารางที่ 50 ผลการทดสอบโมเดล Imbalance ที่เรียนรู้ด้วยชุดข้อมูลจาก Chat-GPT จำนวน 200 ข้อความ

คลาส	จำนวนที่ทดสอบ	จำนวนที่ถูกต้อง		
		Tranformer	CNN	LSTM
0	20	20	15	15
1	20	20	20	20
2	20	20	20	20

คลาส	จำนวนที่ทดสอบ	จำนวนที่ถูกต้อง		
		Tranformer	CNN	LSTM
3	20	19	20	20
4	20	10	11	11
5	20	14	14	14
6	20	20	20	20
7	20	7	9	7
8	20	14	14	14
9	20	18	20	19
รวม	200	162	163	160

จากการทดสอบทั้งหมดจากชุดข้อมูลปกติจำนวน 2,959 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.89489) ,Tranformer (0.88611) และ LSTM (0.88577) ตามลำดับ และจากชุดข้อมูลChat-GPT จำนวน 200 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.81499) ,Tranformer (0.81000) และ LSTM (0.80000) ตามลำดับ

วิเคราะห์ประสิทธิภาพของด้านเวลา

ตารางที่ 51 ผลการทดสอบโมเดล Imbalance ด้านเวลา

ข้อมูลทดสอบ	จำนวนข้อมูล	เวลาที่ใช้ (วินาที)		
		Tranformer	CNN	LSTM
ชุดข้อมูลปกติ	2,959	2.646017	0.561143	4.311373
ชุดข้อมูลจาก Chat-GPT	200	0.436017	0.097381	1.602529

จากการทดสอบด้านเวลาทั้งหมดจากชุดข้อมูลปกติจำนวน 2,959 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.561143 วินาที) ,Tranformer (2.646017 วินาที) และ LSTM (4.311373

วินาที) ตามลำดับ และจากชุดข้อมูลChat-GPT จำนวน 200 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.561143 วินาที) ,Tranformer (0.436017 วินาที) และ LSTM (1.602529 วินาที) ตามลำดับ

ตารางที่ 52 ผลการทดสอบโมเดล Balance ที่เรียนรู้ด้วยชุดข้อมูลปกติจำนวน 2,492 ข้อความ

คลาส	จำนวนที่ทดสอบ	จำนวนที่ถูกต้อง		
		Tranformer	CNN	LSTM
0	386	368	359	348
1	184	168	169	163
2	223	196	206	212
3	215	192	193	195
4	252	211	221	219
5	256	211	219	220
6	209	179	179	179
7	272	246	245	242
8	266	218	220	222
9	230	186	190	188
รวม	2,492	2,175	2,198	2,188

ตารางที่ 53 ผลการทดสอบโมเดล Balance ที่เรียนรู้ด้วยชุดข้อมูลจาก Chat-GPT จำนวน 200 ข้อความ

คลาส	จำนวนที่ทดสอบ	จำนวนที่ถูกต้อง		
		Tranformer	CNN	LSTM
0	20	20	11	12
1	20	20	20	20
2	20	20	20	20
3	20	20	20	20
4	20	10	11	13
5	20	12	14	16
6	20	20	20	20
7	20	7	8	7
8	20	13	13	13
9	20	15	20	20
รวม	200	155	157	161

จากการทดสอบทั้งหมดจากชุดข้อมูลปกติจำนวน 2,492 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.88287), LSTM (0.87765) และ Tranformer (0.87244) ตามลำดับ และจากชุดข้อมูลChat-GPT จำนวน 200 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ LSTM (0.80500), Tranformer (0.78500), CNN (0.77499) และ ตามลำดับ

วิเคราะห์ประสิทธิภาพของด้านเวลา

ตารางที่ 54 ผลการทดสอบโมเดล Balance ด้านเวลา

ข้อมูลทดสอบ	จำนวนข้อมูล	เวลาที่ใช้ (วินาที)		
		Tranformer	CNN	LSTM
ชุดข้อมูลปกติ	2,959	2.011279	0.450929	3.128241
ชุดข้อมูลจาก Chat-GPT	200	0.265181	0.144292	0.328518

จากการทดสอบด้านเวลาทั้งหมดจากชุดข้อมูลปกติจำนวน 2,492 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0.450929 วินาที) ,Tranformer (2.011279 วินาที) และ LSTM (3.128241 วินาที) ตามลำดับ และจากชุดข้อมูลChat-GPT จำนวน 200 ข้อความ เรียงลำดับโมเดลที่ดีที่สุดได้คือ CNN (0. 144292 วินาที) ,Tranformer (0.265181 วินาที) และ LSTM (0.328518 วินาที) ตามลำดับ

ผลการวัดประสิทธิภาพด้วย DSM-5

เมื่อทำการประเมินโดยใช้ชุดข้อมูลรองที่สร้างโดย ChatGPT ซึ่งประกอบด้วยข้อความเพียง 14 ตัวอย่าง พบว่าประสิทธิภาพของทุกโมเดลลดลงเมื่อเทียบกับชุดข้อมูลหลัก

ชุดข้อมูลรองนี้ แม้จะมีขนาดจำกัด แต่ถูกออกแบบให้รวมถึงอาการที่ไม่ใช่ภาวะซึมเศร้าและอาการของภาวะซึมเศร้า กระจายอยู่ในช่วงเวลา 14 วัน เพื่อให้สอดคล้องกับเกณฑ์ DSM-5 สำหรับการประเมินภาวะซึมเศร้า

ผลการประเมินมีดังนี้:

Transformer ทำได้ดีที่สุดด้วยความแม่นยำ 86.67% ตามมาด้วย CNN และ LSTM ที่มีความแม่นยำ 83.33%

Precision, Recall และ F1-Score แสดงให้เห็นว่า Transformer มีประสิทธิภาพดีกว่า CNN และ LSTM โดยเฉพาะในด้านการรักษาสมดุลระหว่าง Precision และ Recall

การวิเคราะห์ชุดข้อมูลรอง (DSM-5 Score)

ชุดข้อมูลรองที่ใช้ Transformer ประกอบด้วย 14 ข้อความที่ถูกสร้างขึ้นและประเมินในช่วงเวลา 14 วัน (ตั้งแต่ 2023-12-04 16:46:00 ถึง 2023-12-18 16:45:59)

ข้อมูลถูกวิเคราะห์โดยใช้เกณฑ์ DSM-5 ซึ่งจำแนกข้อความเป็น "Happy" (มีความสุข) หรือเชื่อมโยงกับอาการของภาวะซึมเศร้า

ตารางที่ 55 ผลการทดสอบอาการผิดปกติในแต่ละวัน

วันที่	**อาการที่คาดเดา**	**อาการจริง**
2023-12-04	Happy, Happy	Happy, Happy
2023-12-05	Happy, Happy, Depressive Sadness, Depressive Sadness	Happy, Happy, Depressive Sadness, Depressive Sadness
2023-12-06	Loss of Interest, Loss of Interest	Loss of Interest, Loss of Interest
2023-12-07	Appetite, Appetite, Happy	Appetite, Appetite, Sleep disturbance
2023-12-08	Sleep disturbance, Happy	Sleep disturbance, Guilt
2023-12-09	Difficulty concentrating, Depressive Sadness	Difficulty concentrating, Depressive Sadness
2023-12-10	Depressive Sadness, Psychomotor changes	Depressive Sadness, Psychomotor changes
2023-12-11	Psychomotor change, Psychomotor change, Psychomotor change	Psychomotor change, Psychomotor change, Psychomotor change

วันที่	อาการที่คาดเดา	อาการจริง
2023-12-12	Guilt, Guilt	Guilt, Guilt
2023-12-13	Guilt, Guilt	Guilt, Guilt
2023-12-14	Difficulty concentrating	Difficulty concentrating
2023-12-15	Difficulty concentrating, Happy	Difficulty concentrating, Difficulty concentrating
2023-12-16	Happy, Appetite, Appetite	Difficulty concentrating, Appetite, Appetite

ตารางที่ 56 ผลการวัดประสิทธิภาพโมเดล Tranformer, Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ที่ประสิทธิภาพด้วย DSM-5

โมเดล	ตัวชี้วัดการประเมิน			
	Accuracy (%)	*Precision (%)*	*Recall (%)*	*F1-Score (%)*
Transformer	86.67	83.33	78.89	78.70
CNN	83.33	80.32	76.11	76.12
LSTM	83.33	80.32	76.11	76.12

การประเมินตามเกณฑ์ DSM-5:

คะแนนรวม: 8

ระดับภาวะซึมเศร้า: 2

จากการให้คะแนนตาม DSM-5 ใน ตาราง ll และการวิเคราะห์อาการ พบว่า ชุดข้อมูลรองสะท้อนถึงแนวโน้มของภาวะซึมเศร้าในระดับเล็กน้อย ที่เกิดขึ้นตลอดช่วงเวลา 14 วัน โดยมีรูปแบบของอาการที่เกิดซ้ำ ๆ

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

ในการวิจัยครั้งนี้ ผู้วิจัยได้ทำการทดลองเพื่อพัฒนากระบวนการจำแนกคัดแยกโรคซึมเศร้าจากการโพสต์ข้อความบนทวิตเตอร์ โดยใช้การสร้างแบบจำลองด้วยอัลกอริทึมต่าง ๆ เพื่อคัดกรองโรคซึมเศร้าจากข้อความ ซึ่งมุ่งหวังช่วยในการวินิจฉัยของแพทย์ ช่วยในการสังเกตุจากคนรอบข้างและสามารถนำไปประยุกต์ใช้ในหน่วยงานที่เกี่ยวข้องได้ในอนาคต

สรุปผลและอภิปรายผล

การจำแนกข้อความที่มีโอกาสเข้าข่ายเป็นโรคซึมเศร้าที่มาจากการโพสต์ข้อความลงบนทวิตเตอร์โดยใช้อัลกอริทึมต่าง ๆ มาช่วยในการคัดแยกข้อความและวิเคราะห์โอกาสการเป็นโรคซึมเศร้าโดยมีการใช้อัลกอริทึม 3 ตัวในการทดลองคือ Tranformer, CNN และ LSTM และใช้ชุดข้อมูลที่แบ่งเป็นข้อความแสดงอาการโรคซึมเศร้า 9 ประเภท และชุดข้อความที่แสดงอาการปกติอีก 1 ประเภทดังนี้

1) ข้อความเกี่ยวกับอารมณ์ซึมเศร้า
2) ข้อความเกี่ยวกับการขาดความสนใจลดลง
3) ข้อความเกี่ยวกับน้ำหนักผิดปกติ
4) ข้อความเกี่ยวกับการนอนผิดปกติ
5) ข้อความเกี่ยวกับร่างกายอ่อนเพลีย
6) ข้อความเกี่ยวกับการรู้สึกตนเองไร้ค่า
7) ข้อความเกี่ยวกับสมาธิสั้น
8) ข้อความเกี่ยวกับการเคลื่อนไหลช้า
9) ข้อความเกี่ยวกับการคิดฆ่าตัวตาย
10) ข้อความที่แสดงถึงอาการปกติ

โดยข้อมูลมีจำนวนทั้งหมด 35,000 ข้อความ แบ่งเป็นชุดข้อความที่แสดงอาการซึมเศร้าประเภทละ 3,000 ข้อความ และชุดข้อความที่แสดงอาการปกติอีก 8,000 ข้อความรวมถึงอีกชุดข้อมูลคือ มีจำนวนทั้งหมด 30,000 ข้อความ แบ่งเป็นชุดข้อความที่แสดงอาการซึมเศร้าประเภทละ 3,000 ข้อความ และชุดข้อความที่แสดงอาการปกติอีก 3,000 ข้อความ แล้วนำไปผ่านการประมวลผลข้อมูล

โดยใช้กระบวนการ ได้แก่ 1)การลบสัญลักษณ์พิเศษ 2)การลบการกล่าวถึง (Mentions) 3)ลบการกล่าวถึงผู้ใช้งานอื่น ๆ ที่มีสัญลักษณ์ @ ตามด้วยชื่อ 4)ลบ URL หรือลิงก์ที่ปรากฏในข้อความ 5)ลบแท็กที่มีสัญลักษณ์ # 6)ลบตัวเลขทั้งหมด 7)ลบตัวอักษรหรือคำที่ไม่ใช่ภาษาอังกฤษ 8)การลบเครื่องหมายวรรคตอน 9)การทำ Lemmatization 10)แปลงคำให้อยู่ในรูปแบบรากศัพท์ 11)ลบคำที่มักพบบ่อยแต่ไม่ช่วยสื่อความหมาย หลังจากการเตรียมข้อมูลจะเหลือชุดข้อมูลทั้งหมด 18,805 ข้อความ และนำชุดข้อความไปทำการเพิ่มคำที่มีความหายใกล้เคียงกันด้วยวิธีการ Augmentation โดยเน้นเพิ่มเฉพาะในประเภทที่ 2 ที่มีข้อความจำนวนน้อยที่สุดจาก 462 เป็น 924 ข้อความ หลังจากการทำ Augmentation จะได้ชุดข้อความทั้งหมด 19,729 ข้อความกับอีกชุดคือเพิ่มเฉพาะในประเภทที่ 2 ที่มีข้อความจำนวนน้อยที่สุดจาก 462 เป็น 924 ข้อความ หลังจากการทำ Augmentation จะได้ชุดข้อความทั้งหมด 16,622 ข้อความ และนำชุดข้อมูลที่ได้ไปใช้งานกับโมเดล Tranformer, CNN และ LSTM โดยนำมาใช้ทดลองกับชุดข้อมูล 2 แบบคือชุดข้อมูลปกติที่นำมาใช้กันการฝึกฝนของโมเดลแยกของส่วน Imbalance มาจำนวน 2,959 ข้อความส่วนโมเดลแยกของส่วน Imbalance มาจำนวน 2,492 ข้อความ และชุดข้อมูลจาก Chat-GPT จำนวน 200 ข้อความ โดยการทดลองจากชุดข้อมูลปกติได้ผลการทดลองคือ

ในส่วนของ Imbalance ได้ผลการทดลอง CNN ร้อยละ 89.489 ,Tranformer ร้อยละ 88.611 และ LSTM ร้อยละ 88.577 โดยใช้เวลาคือ CNN 0.561143 วินาที ,Tranformer 2.646017 วินาที และ LSTM 4.311373 วินาที หลังจากการทดลองกับชุดข้อมูลปกติแล้วเพื่อเป็นการพิสูจน์และพยายามหาข้อแตกต่างจึงได้ทำการทดลองเพิ่มอีกจากข้อมูลประเภทต่าง ๆ ที่เกี่ยวกับอาการซึมเศร้าและอาการปกติได้ผลการทดลองคือ CNN ร้อยละ 81.499 ,Tranformer ร้อยละ81.000 และ LSTM ร้อยละ 80.000

โดยใช้เวลาคือ CNN 0.561143 วินาที ,Tranformer 0.436017 วินาที และ LSTM 1.602529 วินาที และในส่วนของ Balance ได้ผลการทดลอง CNN ร้อยละ 0.88287, LSTM ร้อยละ 0.87765 และ Tranformer ร้อยละ 0.87244 โดยใช้เวลาคือ CNN 0.450929 วินาที ,Tranformer 2.011279 วินาที และ LSTM 3.128241 วินาที หลังจากการทดลองกับชุดข้อมูลปกติแล้วเพื่อเป็นการพิสูจน์และพยายามหาข้อแตกต่างจึงได้ทำการทดลองเพิ่มอีกจากข้อมูลประเภทต่าง ๆ ที่เกี่ยวกับอาการซึมเศร้าและอาการปกติได้ผลการทดลอง LSTM ร้อยละ 0.80500, Tranformer ร้อยละ 0.78500, CNN ร้อยละ 0.77499

โดยใช้เวลาคือ CNN 0.144292 วินาที ,Tranformer 0.265181 วินาที และ LSTM 0.328518 วินาที จากผลการทดลองของ Imbalance ทั้งจากชุดข้อมูลปกติและชุดข้อมูลจาก Chat-GPT โมเดล CNN ทำได้ดีกว่าทั้งด้านความแม่นยำและทั้งด้านเวลาที่ใช้น้อยกว่า รองลงมาคือ Tranformer และ LSTM ตามลำดับในส่วนของ Balance จากชุดข้อมูลปกติโมเดล CNN ทำได้ดีที่สุดด้านประสิทธิภาพ รองลงมาคือ LSTM และ Tranformer ตามลำดับ ส่วนของด้านเวลาคือ CNN ทำได้ดีที่สุดรองลงมาคือ Tranformer และ LSTM ตามลำดับ ชุดข้อมูลจาก Chat-GPT โมเดล LSTM ทำได้ดีที่สุดด้านประสิทธิภาพ รองลงมาคือ Tranformer และ CNN ตามลำดับ ในส่วนของเวลาคือ CNN ทำได้ดีที่สุด รองลงมาคือ Tranformer และ LSTM ตามลำดับ

ปัญหาและอุปสรรคในการดำเนินงาน

1) ชุดข้อมูลที่สามารถนำมาใช้ได้หลังจากกระบวนการเตรียมข้อมูลหายไปเป็นจำนวนมาก ส่งผลถึงการเรียนรู้ของข้อมูล และอาจเรียนรู้ได้ไม่มีประสิทธิภาพเท่าที่ควร

2) ชุดข้อมูลเป็นข้อความที่ไม่สามารถนำมาใช้การได้เลยจึงจำเป็นต้องนำมาผ่านกระบวนการเตรียมข้อมูลก่อน

ข้อเสนอแนะ

1) ทำการเพิ่มจำนวนข้อมูลให้มีมากขึ้นเพื่อเพิ่มประสิทธิภาพและความแม่นยำในการทำนายให้มากยิ่งขึ้น

2) ด้วยการทำการทดลองไม่ได้เจาะจงไปที่เอกลักษณ์หรือลักษณะใดของผู้ถูกทำนายเป็นพิเศษ จึงอาจจะมีปัญหาในการนำไปใช้ในความเป็นจริง เช่น เพศ อายุ หรือเอกลักษณ์อื่น ๆ ที่อาจจะไม่ได้กล่าวถึง

บรรณานุกรม

[1] แก้วสว่าง ต. องค์ความรู้โรคซึมเศร้าและจิตวิทยาเชิงบวกสำหรับนักจิตวิทยา. วารสารจิตวิทยาคลินิกไทย (Online) 2020; 51[1]: 49-65. [2025/03/17]; https://so03.tci-thaijo.org/index.php/tci-thaijclinicpsy/article/view/251796

[2] amazon a. การทำเหมืองข้อมูลคืออะไร. 18 สิงหาคม]; https://aws.amazon.com/th/what-is/data-mining/.

[3] เดินรีบรัมย์ ด, เจริญผล ฉ, จิรานุกูล จ. การเปรียบเทียบประสิทธิภาพโครงสร้างเหมืองข้อมูลเพื่อจำแนกโรคซึมเศร้าจากพฤติกรรมการโพสต์ชัอ ความบนทวิตเตอร์. Journal of Science & Technology MSU 2020; 39[3]:

[4] Nuankaew W, Jareanpon C. Improvement of prediction technique for adolescent depression disorder. Mahasarakham University; 2022.

[5] โรงพยาบาลมนารมย์. โรคซึมเศร้า. 18 สิงหาคม]; https://www.manarom.com/blog/depression_disorder.html.

[6] โรงพยาบาลศิครินทร์. เรียนรู้และเข้าใจ 'โรคซึมเศร้า'. 18 สิงหาคม]; https://www.sikarin.com/health/โรคซึมเศร้า-depression.

[7] amazon a. การประมวลผลภาษาธรรมชาติ (NLP) คืออะไร. 18 สิงหาคม]; https://aws.amazon.com/th/what-is/nlp/.

[8] Wongkoblap A, Vadillo MA, Curcin V. Deep learning with anaphora resolution for the detection of tweeters with depression: Algorithm development and validation study. JMIR mental health 2021; 8[8]: e19824.

[9] Luo X. Efficient English text classification using selected Machine Learning Techniques. Alexandria Engineering Journal 2021; 60[3]: 3401-3409. https://www.sciencedirect.com/science/article/pii/S1110016821000806

[10] Wu H, Liu Y, Wang J. Review of Text Classification Methods on Deep Learning. Computers, Materials & Continua 2020; 63[3]:

[11] Rodrawangpai B, Daungjaiboon W. Improving text classification with transformers and layer normalization. Machine Learning with Applications 2022; 10100403. https://www.sciencedirect.com/science/article/pii/S2666827022000792

[12] Pappagari R, Zelasko P, Villalba J, Carmiel Y, Dehak N. Hierarchical Transformers for Long Document Classification. 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU); 14-18 Dec. 2019; 838-844.

[13] Wang R, Li Z, Cao J, Chen T, Wang L. Convolutional Recurrent Neural Networks for Text Classification. 2019 International Joint Conference on Neural Networks (IJCNN); 14-19 July 2019; 1-6.

[14] Li P, Zhong P, Mao K, Wang D, Yang X, Liu Y, et al. Act: an attentive convolutional transformer for efficient text classification. Proceedings of the AAAI conference on artificial intelligence; 13261-13269

ประวัติผู้เขียน

ชื่อ	**พศวัต ศรีแก้ว**
วันเกิด	5 กรกฎาคม 2543
สถานที่เกิด	อำเภอเมือง จังหวัดสระแก้ว
สถานที่อยู่ปัจจุบัน	222 หมู่ 9 ตำบลขนวน อำเภอหนองนาคำ จังหวัดขอนแก่น 40150
ตำแหน่งหน้าที่การงาน	นักศึกษา
สถานที่ทำงานปัจจุบัน	มหาวิทยาลัยมหาสารคาม
ประวัติการศึกษา	พ.ศ. 2555 ประถมศึกษาโรงเรียนอนุบาลภูเวียง พ.ศ. 2561 มัธยมศึกษาโรงเรียนภูเวียงวิทยาคม พ.ศ. 2565 ปริญญาตรีมหาวิทยาลัยขอนแก่น คณะวิทยาลัยการคอมพิวเตอร์ สาขาวิทยาการคอมพิวเตอร์ พ.ศ. 2567 ปริญญาโทมหาวิทยาลัยมหาสารคาม คณะวิทยาการสารสนเทศ สาขาวิทยาการคอมพิวเตอร์