

การจำแนกข้อความสำหรับการรายงานปัญหาในระบบ Traffy Fondue ของ
กรุงเทพมหานคร
TEXT CLASSIFICATION FOR PROBLEM REPORTING IN BANGKOK'S
TRAFFY FONDUE SYSTEM



การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

2565

KMITL-2022-SC-M-017-131

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

TEXT CLASSIFICATION FOR PROBLEM REPORTING IN BANGKOK'S
TRAFFY FONDUE SYSTEM

The seal of King Mongkut's Institute of Technology Ladkrabang is a circular emblem. It features a central five-tiered umbrella (parasol) with a sunburst at the top. Flanking the central umbrella are two smaller, three-tiered umbrellas. The entire emblem is surrounded by a circular border containing Thai text. The name 'JUFFNEE KHOTSAWAD' is printed in the center of the seal.

JUFFNEE KHOTSAWAD

AN INDEPENDENT STUDY SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN
DATA SCIENCE AND ANALYTICS

KMITL-DIGITAL ANALYTICS AND INTELLIGENCE CENTER SCHOOL OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2022

KMITL-2022-SC-M-017-131

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



COPYRIGHT 2022

SCHOOL OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อการค้นคว้าอิสระ	การจำแนกข้อความสำหรับการรายงานปัญหาในระบบ ทราฟฟิฟองด์ูว์ของกรุงเทพมหานคร
ชื่อนักศึกษา	นางสาวจีพณีย์ คชสวัสดิ์
รหัสนักศึกษา	63605073
ปริญญา	วิทยาศาสตรมหาบัณฑิต (วิทยาการข้อมูลและการวิเคราะห์) ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง
พ.ศ.	2565
อาจารย์ที่ปรึกษาการค้นคว้าอิสระ	ผู้ช่วยศาสตราจารย์ ดร.ปัทมา เจริญพร

บทคัดย่อ

งานวิจัยนี้นำเสนอการจำแนกประเภทของหมวดหมู่ของปัญหาของระบบทราฟฟิฟองด์ูว์ด้วยการใช้การจำแนกข้อความ (Text Classification) ซึ่งประเภทหมวดหมู่ของปัญหามีทั้งหมด 14 หมวดหมู่ จากข้อมูลการรายทั้งหมด 141,079 รายงาน พบว่ามีข้อมูลที่ไม่สามารถระบุหมวดหมู่ของปัญหาได้ 62,676 รายงาน หรือคิดเป็น 44.43% ของข้อมูลที่มีการรายงานทั้งหมด ส่งผลให้ปัญหาเหล่านั้นไม่มีหมวดหมู่ของปัญหาที่ชัดเจนและก่อให้เกิดความล่าช้าในการแก้ไขปัญหาเพราะไม่สามารถระบุหน่วยงานที่รับผิดชอบในการแก้ไขปัญหาได้ โดยผู้วิจัยได้ทำการจำแนกข้อความด้วยการใช้สถาปัตยกรรมโมเดล 4 รูปแบบ คือ สถาปัตยกรรมโมเดลพร้อมใช้งาน 1 รูปแบบ คือ โมเดลเข้ารหัสประโยคสากล (Universal Sentence Encoder : USE) และสถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก 3 รูปแบบคือ โมเดลหน่วยความจำระยะสั้นแบบยาว (Long Short-Term Memory : LSTM), โมเดลโครงข่ายประสาทเชิงลึก (Deep Neural Network : DNN) และ โมเดลโครงข่ายประสาทคอนโวลูชัน (Convolutional Neural Network : CNN) และสถาปัตยกรรมโมเดลทั้ง 4 รูปแบบจะถูกทดสอบวัดประสิทธิภาพด้วยมาตรวัดต่างๆ โดยสถาปัตยกรรมโมเดลที่ให้ค่าประสิทธิภาพมีความแม่นยำมากที่สุดคือโมเดลความจำระยะสั้นระยะยาว (Long Short-Term Memory : LSTM) ซึ่งมีค่าความแม่นยำ (Accuracy) อยู่ที่ 99.70% ค่าเฉลี่ยมาโคร (Marco Average) อยู่ที่ 99.60% และค่าเฉลี่ยถ่วงน้ำหนัก (Weight Average) อยู่ที่ 99.70%

คำสำคัญ: การจำแนกข้อความ, สถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก, ประเภทหมวดหมู่ของปัญหา, ทราฟฟิฟองด์ูว์, กรุงเทพมหานคร

Independent Study Title	Text Classification for Problem Reporting in Bangkok's Traffy Fondue System
Students Name	Miss. Juffnee Khotsawad
Student ID	63605073
Degree	Master of Science (Data Science and Analytics) KMITL-Digital Analytics and Intelligence Center
Year	2022
Independent Study Advisor	Asst. Prof. Dr. Pattama Charoenporn

Abstract

This research presents the classification of problem types from Bangkok's Traffy Fondue System using text classification. There are 14 categories of problem types from all 141,079 reports. The missing group of problems had 62,676 reports or 44.43% of all reported data. As a result, those problems do not have a clear category of problems and cause delays in solving problems because the agency responsible for solving the problem cannot be identified. The researcher has classified text using four model architectures: a ready-to-use model architecture: Universal Sentence Encoder (USE), and three deep learning model architectures: Long Short-Term Memory (LSTM), Deep Neural Network (DNN), and Convolutional Neural Network (CNN), and four model architectures have tested against various metrics. The most accurate model architecture is the Long Short-Term Memory (LSTM) deep learning model architecture, which has an Accuracy of 99.70%, Marco Average is 99.60% and the Weighted Average is 99.70%.

Keywords: Text classification, Deep learning, Problem types, Traffy Fondue , Bangkok

กิตติกรรมประกาศ

งานวิจัยเรื่องการจำแนกข้อความสำหรับการรายงานปัญหาในระบบกราฟฟี่ฟองคิวของ กรุงเทพมหานครโดยใช้วิธีการเรียนรู้เชิงลึกนี้สามารถสำเร็จลงไปได้ด้วยดี ผู้จัดทำขอขอบพระคุณ อาจารย์ที่ปรึกษา ผศ.ดร.ปัทมา เจริญพร ที่ได้ให้คำชี้แนะแนวทางการแก้ไขปัญหาให้ความรู้เพิ่มเติมในการประยุกต์ใช้ความรู้ให้เกิดประโยชน์สูงสุดในการพัฒนาปรับปรุงงานวิจัยนี้

ขอขอบพระคุณ ทราฟฟี่ฟองคิว และนักวิจัยที่พัฒนาแพลตฟอร์มบริหารจัดการปัญหานี้ขึ้นมา และมีการรวบรวมข้อมูลพร้อมถึงอธิบายและทำการจัดหมวดหมู่ของปัญหาไว้

สุดท้ายนี้ ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งเป็นอีกหนึ่งปัจจัยสำคัญที่ทำให้ได้พบกับคณาจารย์ที่มีศักยภาพ ในการช่วยพัฒนาทักษะและมอบความรู้แก่นักศึกษา

จีพณีย์ คชสวัสดิ์

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	ก
บทคัดย่อภาษาอังกฤษ	ข
กิตติกรรมประกาศ	ค
สารบัญ	ง
สารบัญตาราง	ฉ
สารบัญรูป	ญ
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญ	1
1.2 วัตถุประสงค์ของงานวิจัย	2
1.3 ขอบเขตของงานวิจัย	2
1.4 ประโยชน์ที่คาดว่าจะได้รับ	2
1.5 ขั้นตอนการดำเนินงาน	2
บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง	3
2.1 แพลตฟอร์มบริหารจัดการปัญหาจราจรฟฟองดูว์ (Traffy* Fondue)	3
2.1.1 ขั้นตอนการใช้งาน	3
2.1.2 แดชบอร์ด (Dashboard) ของจราจรฟฟองดูว์ (Traffy* Fondue)	3
2.1.3 ข้อมูลของจราจรฟฟองดูว์ (Traffy* Fondue)	4
2.2 การประมวลผลภาษาธรรมชาติ (Natural Language Processing: NLP)	5
2.2.1 ภาษาศาสตร์เพื่อการวิเคราะห์ข้อความ (Linguistics for text analysis)	6
2.2.2 กระบวนการทำงานของ NLP	6
2.2.2.1. Tokenization	6
2.2.2.2. Part-of-speech tagging	7
2.2.2.3. Named-Entity Recognition (NER)	8

สารบัญ (ต่อ)

	หน้า
2.2.2.5. Word Sense Disambiguation	8
2.2.2.6. Word Embedding	8
2.2.3 LTW2V: The Large Thai Word2Vec	9
2.3 โครงสร้างประสาทเทียม (Artificial Neural Network) 9	9
2.3.1 ชั้นข้อมูลเข้า (Input layer)	9
2.3.2 ชั้นซ่อน (Hidden layer)	9
2.3.3 ชั้นข้อมูลออก (Output layer)	10
2.4 ฟังก์ชันการส่งต่อ (Transfer function)	10
2.4.1 ฟังก์ชันกระตุ้นค่าแบ่ง	10
2.4.2 ฟังก์ชันกระตุ้นซิกมอยด์	10
2.4.3 ฟังก์ชันไฮเปอร์โบลิคแทงกเจนท์	11
2.4.4 Rectified Linear Unit Function	11
2.4.5 SoftMax layer	11
2.5 การเรียนรู้เชิงลึก (Deep learning)	12
2.6 Convolutional Neural Network (CNN)	12
2.7 Universal sentence encoder	13
2.8 Recurrent Neural Network (RNN)	14
2.9 Long Short-Term Memory (LSTM)	15
2.9.1 การทำงานของ LSTM	15
2.9.1.1 Forget	16
2.9.1.2 Write	17
2.9.1.3 อัปเดต cell state	17
2.9.1.4 Read	17

สารบัญ (ต่อ)

	หน้า
2.10 งานวิจัยที่เกี่ยวข้อง	18
2.10.1 Multilingual Universal Sentence Encoder for Semantic Retrieval	18
2.10.2 Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM)	18
2.10.3 Deep neural network-based classification model for Sentiment Analysis	19
บทที่ 3 การดำเนินงานวิจัย	20
3.1 ความเป็นมาและความสำคัญ	20
3.2 แนวทางการดำเนินงานวิจัย	21
3.3 การทำความเข้าใจกับข้อมูล (Data Understanding)	21
3.4 การเตรียมข้อมูล (Data Preparation)	23
3.4.1 การทำความสะอาดข้อมูล (Data Cleaning)	23
3.4.2 การเตรียมข้อมูลสำหรับการเข้าโมเดลของ Universal Sentence Encoder (USE)	27
3.4.3 การเตรียมข้อมูลสำหรับการเข้าโมเดลของการเรียนรู้เชิงลึก (Deep Learning)	28
3.4.3.1 การตัดคำ (Word Segmentation)	28
3.4.3.2 การแปลงลำดับคำของตัวแปรต้นเป็นรายการดัชนีคำ	30
3.4.3.3 การเพิ่มจำนวนคำของตัวแปรต้น (Pad Word Indices)	30
3.4.3.4 การแปลงคำของตัวแปรตามเป็นรายการดัชนีคำ	31
3.5 การสร้างรูปแบบการจำแนกประเภท (Modeling)	33
3.5.1 สถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE)	33
3.5.2 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)	34
3.5.3 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)	36

สารบัญ (ต่อ)

	หน้า
3.5.4 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)	37
3.6 การวัดประเมินผล (Evaluation)	39
บทที่ 4 ผลการทดลองและการอภิปรายผล	40
4.1 ผลการทดลองของสถาปัตยกรรมโมเดล Universal Sentence Encoder (USE)	40
4.2 ผลการทดลองของสถาปัตยกรรมโมเดล Long Short-Term Memory (LSTM)	42
4.3 ผลการทดลองของสถาปัตยกรรมโมเดล Deep Neural Network (DNN)	44
4.4 ผลการทดลองของสถาปัตยกรรมโมเดล Convolutional Neural Network (CNN)	46
บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ	53
5.1 สรุปผลการดำเนินงานวิจัย	53
5.2 ข้อเสนอแนะ	54
5.2.1 การใช้เครื่องมือในการตัดคำ	54
5.2.2 การจัดความสำคัญของปัญหา (Priority Organization)	54
บรรณานุกรม	

สารบัญตาราง

ตารางที่	หน้า
ตารางที่ 2.1 ตารางอธิบายฟิลต์ของทราฟฟี่ฟองดูว์ (Traffy* Fondue)	5
ตารางที่ 2.2 ความหมายของสาขาย่อยภาษาศาสตร์แต่ละสาขา	6
ตารางที่ 3.1 ตารางแสดงจำนวนข้อมูลทั้งหมดและข้อมูลที่เกิดการสูญหาย	24
ตารางที่ 3.2 ตารางแสดงจำนวนข้อมูลของปัญหาแต่ละประเภท	26
ตารางที่ 3.3 ค่า Hyperparameters ของสถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก	33
ตารางที่ 4.1 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Universal Sentence Encoder (USE)	41
ตารางที่ 4.2 การวัดประสิทธิภาพของโมเดล Universal Sentence Encoder (USE)	41
ตารางที่ 4.3 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Long Short-Term Memory (LSTM)	43
ตารางที่ 4.4 การวัดประสิทธิภาพของโมเดล Long Short-Term Memory (LSTM)	43
ตารางที่ 4.5 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Deep Neural Network (DNN)	45
ตารางที่ 4.6 การวัดประสิทธิภาพของโมเดล Deep Neural Network (DNN)	45
ตารางที่ 4.7 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Convolutional Neural Network (CNN)	47
ตารางที่ 4.8 การวัดประสิทธิภาพของโมเดล Convolutional Neural Network (CNN)	47
ตารางที่ 4.9 ตารางเปรียบเทียบประสิทธิภาพของสถาปัตยกรรมโมเดลและเวลา	48

สารบัญรูป

รูปที่	หน้า
2.1 แดชบอร์ด (Dashboard) ของทราฟฟี่ฟองดูว์ (Traffy* Fondue) 4	4
2.2 การทำ Word Embedding 8	8
2.3 โครงสร้างสถาปัตยกรรมประสาทเทียม 9	9
2.4 Sigmoid curve 10	10
2.5 Rectified Linear Unit Function 11	11
2.6 SoftMax layer 12	12
2.7 โครงสร้างสถาปัตยกรรม CNN 13	13
2.8 โครงสร้างสถาปัตยกรรมโมเดลการฝึกฝนของ Multilingual universal sentence encoder multi-task training over 13	13
รูปภาพ 2.9 โครงสร้าง RNN 14	14
3.1 ตัวอย่างข้อมูลของทราฟฟี่ฟองดูว์ 15	15
3.2 คำสั่งการเลือกฟิลต์ 21	21
3.3 คำสั่งการลบแถวที่มีข้อมูลมีการสูญหาย 24	24
3.4 คำสั่งการลบแถวของข้อมูลที่มีประเภทของปัญหาที่ซ้ำซ้อนกัน 25	25
3.5 คำสั่งการแบ่งข้อมูลสำหรับฝึกฝนและข้อมูลสำหรับทดสอบ 26	26
3.6 คำสั่งการเข้ารหัสชุดข้อมูลสำหรับโมเดล Universal Sentence Encoder 27	27
3.7 ตัวอย่างข้อมูลที่ถูกรหัสสำหรับโมเดล Universal Sentence Encoder 27	27
3.8 การติดตั้ง PyThaiNLP 29	29
3.9 คำสั่งตัดคำโดยใช้เครื่องมือ newmm 29	29
3.10 ตัวอย่างข้อมูลหลังการตัดคำ 29	29
3.11 การแปลงลำดับคำของตัวแปรต้นเป็นรายการดัชนีคำ 30	30
3.12 การหาจำนวนคำที่มากที่สุด 30	30
3.13 คำสั่งการเติมจำนวนคำ (Pad Word Indices) 31	31

สารบัญรูป (ต่อ)

รูปที่	หน้า
3.14 ตัวอย่างข้อมูลหลังเติมคำ	31
3.15 คำสั่งเตรียมข้อมูลสำหรับ mapping ตัวแปรตาม	31
3.16 คำสั่ง mapping ชื่อของตัวแปรตามเป็นดัชนีค่า	32
3.17 ตัวอย่างดัชนีค่าของตัวแปรตาม	32
3.18 คำสั่งแปลงค่าเป็นข้อมูลเชิงคุณภาพ (Categorical Data)	32
3.19 ตัวอย่างข้อมูลหลังแปลงเป็นข้อมูลเชิงคุณภาพ	32
3.20 คำสั่งดาวน์โหลดโมเดล Universal Sentence Encoder (USE)	33
3.21 คำสั่งการใช้งาน Logistic Regression	34
3.22 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)	34
3.23 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)	35
3.24 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)	35
3.25 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)	36
3.26 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)	36
3.27 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)	37
3.28 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)	37
3.29 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)	38
3.30 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional neural network	38
3.31 ตัวอย่างการวัดประสิทธิภาพโมเดล	39
4.1 กราฟ Confusion Matrix ของ Universal Sentence Encoder (USE)	40
4.2 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Universal Sentence Encoder (USE)	42
4.3 กราฟ Confusion Matrix ของ Long Short-Term Memory (LSTM)	42

สารบัญรูป (ต่อ)

รูปที่	หน้า
4.4 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Long Short-Term Memory	44
4.5 กราฟ Confusion Matrix ของ Deep Neural Network (DNN)	44
4.6 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Deep Neural Network (DNN)	46
4.7 กราฟ Confusion Matrix ของ Convolutional Neural Network (CNN)	46
4.8 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Convolutional Neural Network (CNN)	48



บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญ

ในกรุงเทพมหานครมีปัญหาต่างๆที่เกิดขึ้นมากมาย ทำให้บางปัญหาที่เกิดขึ้นหน่วยงานหรือภาครัฐไม่สามารถเข้าถึง หรือรับรู้ปัญหานั้นๆได้ ส่งผลให้ปัญหานั้นไม่ได้รับการแก้ไข จึงทำให้มีนักวิจัยไทยพัฒนาแพลตฟอร์มบริหารจัดการปัญหา หรือ ทราฟฟี่ฟองดูว์ (Traffy* Fondue) เพื่อรับเรื่องปัญหาจากประชาชนและส่งต่อปัญหานี้ให้กับหน่วยงานที่รับผิดชอบ ทำให้เกิดประโยชน์มากมาย ทั้งลดเวลาในการแก้ปัญหา และลดค่าใช้จ่ายมากถึง 78 ล้านบาทต่อปี

ทราฟฟี่ฟองดูว์ (Traffy* Fondue) เป็นแพลตฟอร์มในการรับเรื่องปัญหาที่เกิดขึ้นจากประชาชน โดยการพิมพ์ปัญหาเป็นข้อความ ถ่ายรูป ประเภทปัญหา พิกัด ตำแหน่ง GPS ผ่านทางแอปพลิเคชันไลน์ แหบทบอทโดยที่เมื่อประชาชนพิมพ์ปัญหาเข้าสู่ระบบ แหบทบอทจะทำการใช้ Text classification เพื่อจำแนกประเภทของปัญหาให้ประชาชนเลือก หรือถ้าเป็นการถ่ายรูปก็จะทำการใช้ Image classification เพื่อจำแนกประเภทของปัญหาให้เช่นกัน

ซึ่งแพลตฟอร์มนี้ได้ทำการเก็บรวบรวมข้อมูลปัญหาต่างๆไว้ และเปิดเผยข้อมูลให้ประชาชนสามารถเข้าถึงได้ แต่เนื่องจากประชาชนเป็นผู้แจ้งเหตุด้วยตนเองผ่านระบบจึงส่งผลให้ประเภทของปัญหา มีความไม่สมบูรณ์ ไม่สามารถระบุประเภทของปัญหาที่ชัดเจน หรือมีการระบุประเภทของปัญหาที่มากเกินไปจริง ส่งผลให้ยากต่อการประสานงานเพื่อแก้ไขปัญหาต่างๆ ดังนั้นงานวิจัยชิ้นนี้จะศึกษาการจำแนกประเภทของปัญหา โดยใช้การประมวลผลภาษาธรรมชาติ หรือ Natural Language Processing (NLP) เพื่อพัฒนาออลิเทียมของการจำแนกข้อความ(Text classification) ให้มีประสิทธิภาพมากขึ้น และสามารถนำไปใช้แก้ปัญหการระบุประเภทของปัญหา

1.2 วัตถุประสงค์ของงานวิจัย

1. เพื่อศึกษาการจำแนกประเภทปัญหาโดยใช้การประมวลผลภาษาธรรมชาติ
2. เพื่อจำแนกปัญหาที่ไม่สามารถระบุประเภทของปัญหาจากทราฟฟี่ฟองดูว์ (Traffy* Fondue)
3. เพื่อเปรียบเทียบประสิทธิภาพของสถาปัตยกรรมโมเดล

1.3 ขอบเขตของงานวิจัย

1. ชุดข้อมูลนี้เป็นข้อมูลเปิด (Open Source) จากทราฟฟี่ฟองดูว์ (Traffy* Fondue) ที่รวบรวมการแจ้งปัญหาจากประชาชนในกรุงเทพมหานคร
2. ชุดข้อมูลนี้เก็บข้อมูลตั้งแต่วันที่ วันที่ 1 กันยายน พ.ศ. 2564 ถึงวันที่ 8 กันยายน พ.ศ. 2565

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถจำแนกประเภทของหมวดหมู่ปัญหาที่หายสาบสูญหรือไม่สมบูรณ์จากทราฟฟี่ฟองดูว์ (Traffy* Fondue)
2. สามารถเปรียบเทียบประสิทธิภาพของโมเดลและหาโมเดลที่มีประสิทธิภาพมากที่สุด
3. หน่วยงานสามารถรับเรื่องได้เร็วขึ้นจากการใช้โมเดลทำนายประเภทหมวดหมู่ปัญหา

1.5 ขั้นตอนการดำเนินงาน

1. การทำความเข้าใจกับข้อมูล (Data Understanding)
2. การเตรียมข้อมูล (Data Preparation)
3. การสร้างรูปแบบการทำนาย (Modeling)
4. การประเมินผล (Evaluation)
5. การสรุปผลการทดลอง (Conclusion)

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 แพลตฟอร์มบริหารจัดการปัญหาทราฟฟี่ฟองดูว์ (Traffy* Fondue)

แพลตฟอร์มบริหารจัดการปัญหา หรือ Traffy* Fondue เป็นแพลตฟอร์มที่ถูกสร้างขึ้นมาเพื่อรับรายงานปัญหาจากประชาชน โดยใช้แอปพลิเคชันไลน์แชทบอทเป็นตัวกลางในการรับเรื่องและส่งต่อไปยังหน่วยงานต่างๆ ได้อย่างรวดเร็ว

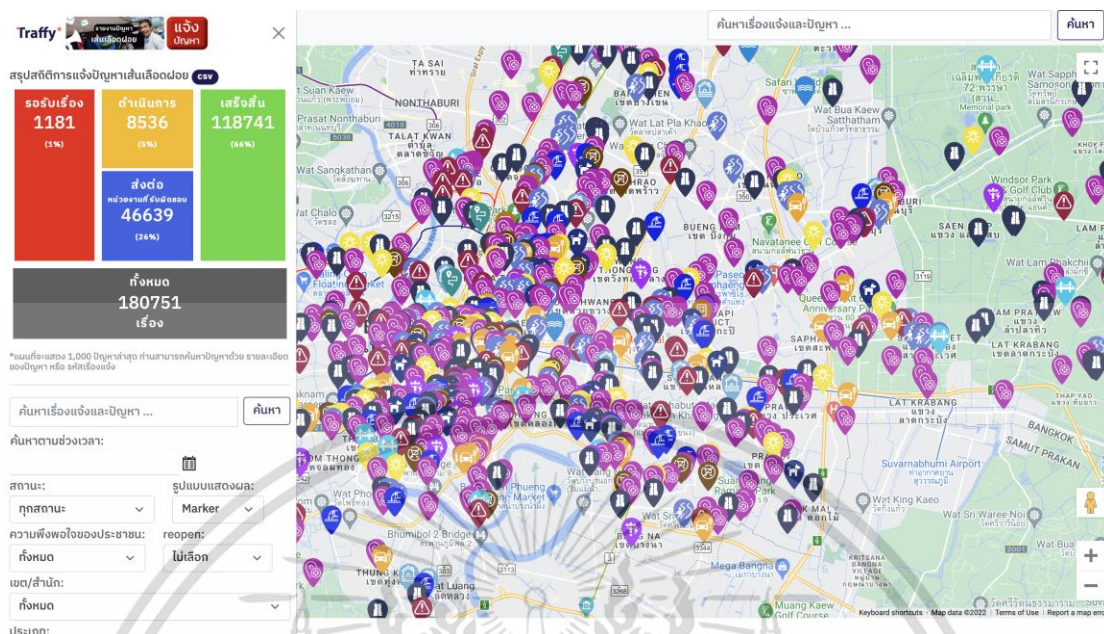
2.1.1 ขั้นตอนการใช้งาน

1. เพิ่ม Traffy Fondue เป็นเพื่อนในไลน์ค้นหาไอดี @traffyfondue
2. พิมพ์ปัญหาที่ต้องการแจ้ง
3. ถ่ายรูปปัญหา
4. ระบุประเภทปัญหา
5. แשרตำแหน่งที่เกิดปัญหา
6. เลือกหน่วยงานที่ต้องการให้เข้าไปแก้ไขปัญหา

โดยปัญหาที่พิมพ์และรูปภาพที่ส่งไป จะมี AI ประมวลผลและส่งกลับมาเป็นประเภทปัญหา และให้ประชาชนเลือกประเภทของปัญหาในขั้นตอนที่ 4 หลังจากทำครบทุกขั้นตอน รอหน่วยงานที่รับผิดชอบดำเนินงานต่อไป

2.1.2 แดชบอร์ด (Dashboard) ของทราฟฟี่ฟองดูว์ (Traffy* Fondue)

นอกจากทราฟฟี่ฟองดูว์ (Traffy* Fondue) จะเป็นแพลตฟอร์มที่รายงานปัญหา ยังมีการทำแดชบอร์ด ดังรูปที่ 2.1 ซึ่งเป็นการแสดงผลสรุปสถิติการแจ้งปัญหา แผนที่ระบุตำแหน่งปัญหาที่เกิดขึ้น และยังสามารถฟิลเตอร์ (Filter) หรือค้นหาเรื่องที่แจ้งและปัญหา ค้นหาตามช่วงเวลา ตามสถานะ ความพึงพอใจ เขตที่เกิดขึ้น หรือประเภทของปัญหา



รูปที่ 2.1 แดชบอร์ด (Dashboard) ของ Traffy* Fondue

โดยจากรูปที่ 2.1 ข้อมูลต่างๆที่อยู่ในแดชบอร์ดจะแสดงผลสรุปของสถานะเรื่องที่ทำการแจ้งซึ่งจะมีตั้งแต่ การรอรับเรื่อง ดำเนินการ ส่งต่อหน่วยงานที่เกี่ยวข้อง และเสร็จสิ้นการแก้ปัญหา และมีการปักหมุดของปัญหาที่แจ้งตามหมวดหมู่ลงในแผนที่ประเทศไทย และข้อมูลที่อยู่ในแดชบอร์ดยังเป็นข้อมูลเปิดสาธารณะเพื่อให้ประชาชนหรือ หน่วยงานต่างๆสามารถดาวน์โหลดเพื่อนำข้อมูลเหล่านี้มาศึกษาต่อได้

2.1.3 ข้อมูลของ Traffy* Fondue

ข้อมูลที่ได้จาก Traffy* Fondue จะเป็นข้อมูลที่มีโครงสร้าง (Structure Data) ซึ่งมีนามสกุลของไฟล์เป็น .csv โดยเป็นข้อมูลที่มาจากการแจ้งปัญหาจากประชาชนผ่านการพิมพ์อธิบายปัญหาที่พบเจอหรือถ่ายรูปปัญหานั้นๆเพื่อเก็บรายละเอียดของข้อมูลเป็นข้อมูลแบบมีโครงสร้าง (Structure Data) และยังมีการเก็บข้อมูลของวันเวลาที่แจ้งหมวดหมู่ของปัญหา รวมไปถึงสถานะของการแจ้งปัญหาว่าอยู่ขั้นตอนไหนแล้วเนื้อหาของปัญหาจะมีฟิลด์ (Field) และคำอธิบายตามตารางที่ 2.1

ตารางที่ 2.1 ตารางอธิบายฟิลด์ของทราฟฟี่ฟองดูว์ (Traffy* Fondue)

Field	คำอธิบาย
ticket_id	รหัสของการแจ้งปัญหา 1 เรื่องต่อ 1 รหัส
type	ประเภทของปัญหาที่ทำการแจ้ง
organization	ชื่อขององค์กรที่อยู่ในเขตรับผิดชอบ
comment	ข้อมูลที่อธิบายปัญหาที่แจ้งจากประชาชน
coords	พิกัดของสถานที่ที่เกิดปัญหา
photo	รูปภาพของปัญหาที่เกิดขึ้น
address	ที่อยู่ของปัญหาที่เกิดขึ้น
district	เขต หรืออำเภอของปัญหาที่เกิดขึ้น
subdistrict	แขวง หรือตำบลของปัญหาที่เกิดขึ้น
province	จังหวัดของปัญหาที่เกิดขึ้น
Timestamp	วันและเวลาที่แจ้งพบปัญหา
state	สถานะของปัญหาที่ประชาชนทำการแจ้งไปแล้ว

2.2 การประมวลผลภาษาธรรมชาติ (Natural Language Processing: NLP)

การประมวลผลภาษาธรรมชาติ เป็นสาขาหนึ่งของวิทยาการคอมพิวเตอร์ (Computer Science) และเป็นสาขาเฉพาะเจาะจงกับแขนง ปัญญาประดิษฐ์ (Artificial Intelligence) หรือ AI โดยวัตถุประสงค์หลักของ NLP คือการให้คอมพิวเตอร์มีความสามารถในการเข้าใจข้อความและคำพูดในลักษณะเดียวกับที่มนุษย์เข้าใจ

NLP เป็นการผสมผสานภาษาศาสตร์คอมพิวเตอร์ หรือ ภาษาศาสตร์เชิงคำนวณ การสร้างแบบจำลองตามกฎของภาษามนุษย์ มารวมเข้ากับสถิติ การเรียนรู้ของเครื่อง (Machine Learning) และแบบจำลองการเรียนรู้เชิงลึก (Deep learning) NLP เทคโนโลยีเหล่านี้ช่วยให้คอมพิวเตอร์สามารถประมวลผลภาษามนุษย์ในรูปแบบของข้อมูลข้อความหรือเสียง และ "เข้าใจ" ความหมายทั้งหมด พร้อมด้วยเจตนาและความรู้สึกนึกคิดของผู้พูดหรือผู้เขียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.1 ภาษาศาสตร์เพื่อการวิเคราะห์ข้อความ (Linguistics for text analysis)

Briscoe (2013) ให้คำแนะนำที่เป็นประโยชน์เกี่ยวกับภาษาศาสตร์และวิธีที่มันติดกันกับสาขาการคำนวณเชิงปฏิบัติของการประมวลผลภาษาธรรมชาติ ภาษาศาสตร์สาขากว้างรวมถึงสาขาย่อยที่เน้นด้านต่างๆ ของภาษา ซึ่งค่อนข้างเป็นลำดับชั้น ดังตารางที่ 2.2 โดยสาขาย่อยของภาษาศาสตร์ ย้ายจากโครงสร้างที่เล็กไปสู่โครงสร้างที่กว้างขึ้น

ตารางที่ 2.2 ความหมายของสาขาย่อยภาษาศาสตร์แต่ละสาขา

สาขาย่อยภาษาศาสตร์	ความหมาย
Phonetics	เสียงที่คนใช้ในภาษา
Phonology	ระบบเสียงในภาษาเฉพาะ
Morphology	วิธีสร้างคำ
Syntax	วิธีสร้างประโยคจากคำ หรือ ไวยากรณ์
Semantics	ความหมายของประโยค
Pragmatics	วิธีใช้ภาษาในบริบท

2.2.2 กระบวนการทำงานของ NLP

การทำงานพื้นฐานของ NLP ประกอบด้วยการทำงานดังต่อไปนี้

2.2.2.1. Tokenization

การแบ่งทำให้เป็น token มีความหมายให้คนเข้าใจ Tokenization เป็นกระบวนการทำงานที่สำคัญต่อ NLP ที่ใช้ในการแบ่งตัวอักษรให้เป็นคำที่มีความหมาย โดยจะมี Sentence tokenization จะเป็นตัวแบ่งประโยคให้เป็นคข้อความ ส่วน word tokenization จะเป็นค้วแบ่งคำในประโยค โดยทั่วไปในภาษาอังกฤษ Word tokens จะสามารถแยกได้โดยใช้เว้นวรรค และ sentence tokens จะแยกโดยใช้มหัพภาค (stops) หรือเรียกว่า ขอบเขตของคำ (word boundary) แต่ในภาษาไทยที่มีความซับซ้อนไม่มีช่องว่าง ขอบเขตของคำ และไม่มีมหัพภาค มีความกำกวมในการจบประโยคเหมือนในภาษาอังกฤษ แต่เราสามารถปรับแต่งให้เป็น high-level tokenization โดยการตัดคำจะมีการตัดคำ 2 แบบด้วยกันคือ Dictionary-based และ Machine Learning-Based

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดย Dictionary-based เป็นอัลกอริทึม และมีฐานข้อมูลเป็นคำศัพท์ หรือมีเป็นพจนานุกรม ซึ่งมี 2 ประเภทคือ Longest matching และ Maximal matching

1. Longest matching

เป็นการเขียนโปรแกรมแบบการสแกนจากซ้ายไปขวา พยายามให้เจอคำใน dictionary ที่ยาวที่สุดแล้วตัด (Until no word matches, longest) เช่น

ป้ายกลับรถ เริ่มสแกนด้วย ป

ป้าย / ยกลับรถ

ป้าย/ กลับรถ / ไม่มีคำไหนยาวกว่าคำว่า ป้าย เปลี่ยนเป็นคำใหม่

ป้าย/ กลับ / รถ

2. Maximal matching

เป็นการตัดให้มีจำนวนคำน้อยที่สุด (Fewest words) เช่น

ไปหามเหสี

ไป / หาม / เห สี

ไป / หา / มเหสี

ส่วน Machine learning-based เป็นการทำการเรียนรู้ของเครื่องโดยมีการสร้างโมเดลขึ้นมาใช้ และมีการเรียนรู้ของข้อมูล (Training Data)

2.2.2.2. Part-of-speech tagging

Part-of-speech tagging คือการบอกประเภทของคำในประโยค ได้แก่ คำนาม nouns, คำสรรพนาม pronouns, คำคุณศัพท์ adjectives, คำกริยา verbs, คำกริยาวิเศษณ์ adverbs, คำบุพบท prepositions, คำสันธาน conjunctions และ คำอุทาน interjections และใน Part-of-speech จะมี PoS tagging ซึ่งเป็นสิ่งที่มีประโยชน์ในการระบุความสัมพันธ์ระหว่างคำ และทำความเข้าใจความหมายของประโยค

2.2.2.3. Named-Entity Recognition (NER)

NER หรือ Named Entity Tagging คือ กระบวนการในการหาตำแหน่ง และ แปะป้ายชื่อเฉพาะของสิ่งต่างๆ ที่อยู่ในเอกสาร เช่น ชื่อคน ชื่อองค์กร สถานที่ ตัวเลข จำนวนเงิน วันเวลา persons, locations, organizations, products, medical codes, time expressions, quantities, monetary values, percentage

2.2.2.4. Parsing

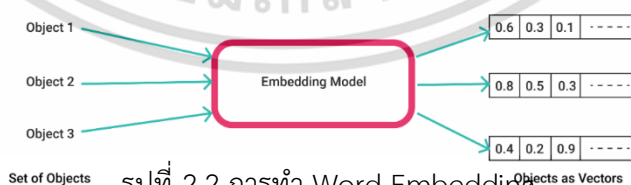
Parsing คือ กระบวนการในการระบุโครงสร้างของข้อความ โดยการวิเคราะห์คำที่เป็นส่วนประกอบ ด้วยหลักไวยากรณ์ของภาษา ผลลัพธ์ที่ได้ออกมาจะเป็นโครงสร้างแบบต้นไม้ เรียกว่า Parse Tree

2.2.2.5. Word Sense Disambiguation

Word Sense Disambiguation คือ ความสามารถที่จะระบุความหมายของคำในทางคอมพิวเตอร์ โดยบ่อยครั้งจะมีการใช้ Corpus ภายนอก เช่น WordNet หรือ Wikipedia มาเป็นตัวอ้างอิงไว้ ตัวอย่างเช่น การพัฒนาอัลกอริทึมที่ระบุได้ว่า คำว่า แอปเปิล ในข้อความนั้นอ้างถึงบริษัท หรือ ผลไม้

2.2.2.6. Word Embedding

Word Embedding คือ การทำตัวอักษรหรือคำ ให้อยู่ในรูปของเวกเตอร์หรือ Embedded Vector เพื่อเป็น Input ของการเข้าสถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก โดย Word Embedding จะเกิดจากการนำคำที่ผ่านการตัดคำแล้ว นำเข้า Embedding Model เช่น Word2Vec, Continuous Bag of Words (CBOW) และ Skip-gram เพื่อให้เกิด Embedded Vector ดังรูปที่ 2.2



หมายเหตุ จาก <https://www.pinecone.io/learn/vector-embeddings/>

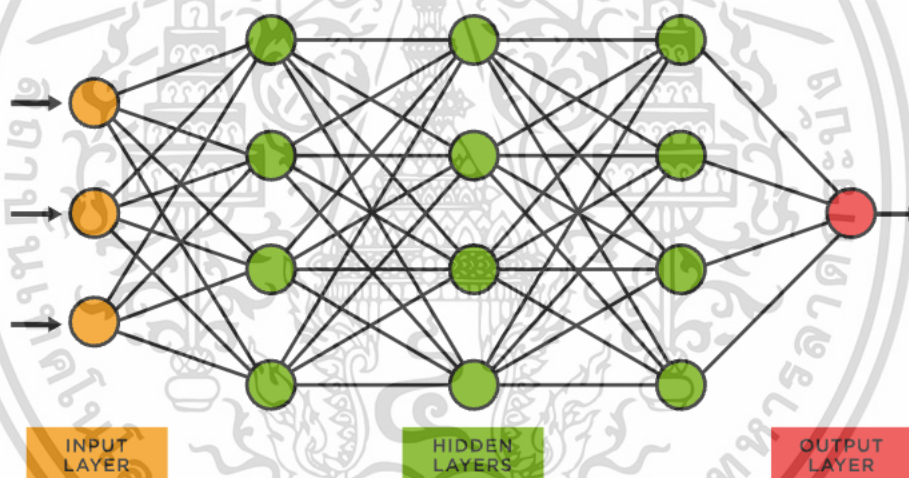
2.2.3 LTW2V: The Large Thai Word2Vec

LTW2V: The Large Thai Word2Vec เป็น Word2Vec ที่สร้างจาก OSCAR Corpus ซึ่งเป็นคลังข้อมูลที่น่าข้อมูลบนเว็บจาก common crawl มาประมวลผลแยกตามภาษา โดยผมได้ train word2vec ด้วย Gensim กับ oxidized-thainlp 1.0 (ตัวตัดคำ newmm ของ pythainlp ที่นำไป port ให้ทำงานบน Rust)

มีจำนวนบรรทัดทั้งหมด 3,749,826 บรรทัด จำนวนคำ 1,739,705,916 คำ (รวมคำซ้ำ) และจำนวนของมิติ 400 มิติ

2.3 โครงสร้างประสาทเทียม (Artificial Neural Network)

โครงสร้างประสาทเทียม (Artificial Neural Network: ANN) เป็นเทคนิคที่เรียนแบบมาจากการทำงานของสมองมนุษย์ หรือสามารถเรียกได้เป็น Fully connected networks หรือ deep neural networks ซึ่ง ANN จะมีประกอบไปด้วยชั้นข้อมูลเข้า (Input layer) ชั้นซ่อน (Hidden layer) และชั้นข้อมูลออก (Output layer) ดังรูปที่ 2.3



รูปที่ 2.3 โครงสร้างสถาปัตยกรรมประสาทเทียม

2.3.1 ชั้นข้อมูลเข้า (Input layer)

โดยชั้นนี้จำนวนของโหนดจะขึ้นอยู่กับจำนวนของข้อมูลที่จะนำมาใช้ในแบบจำลองนี้

2.3.2 ชั้นซ่อน (Hidden layer)

ชั้นนี้เป็นชั้นระหว่างกลางซึ่งสำคัญมากเพราะจะส่งผลประสิทธิภาพของแบบจำลองนี้

2.3.3 ชั้นข้อมูลออก (Output layer)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ขั้นนี้จะเอาข้อมูลจากการคำนวณไปใช้ และแต่ละโหนดจะขึ้นอยู่กับรูปแบบข้อมูลที่จะนำไปใช้

2.4 ฟังก์ชันการส่งต่อ (Transfer function)

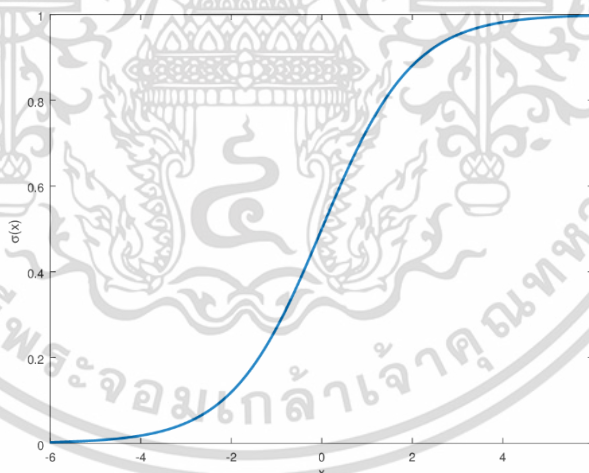
ฟังก์ชันการส่งต่อ หรือฟังก์ชันกระตุ้น เป็นฟังก์ชันในการคำนวณเพื่อการทำนายค่าของผลลัพธ์หรือข้อมูลออก (Output) ฟังก์ชันมีหลายรูปแบบทั้งเป็นเส้นตรง (Linearity) และไม่เป็นเส้นตรง (Non-linearity)

2.4.1 ฟังก์ชันกระตุ้นค่าแบ่ง

ฟังก์ชันกระตุ้นค่าแบ่ง (Threshold Activation Function) ฟังก์ชันนี้เรียกอีกชื่อว่า ฟังก์ชันไบนารีสเตป (Binary step function) ซึ่งพิจารณาว่าข้อมูลเข้าว่ามากกว่าหรือน้อยกว่าค่าแบ่งที่กำหนดไว้ (Threshold) หรือไม่ เพื่อส่งค่าไปต่ออย่างขั้นถัดไป

2.4.2 ฟังก์ชันกระตุ้นซิกมอยด์

ฟังก์ชันกระตุ้นซิกมอยด์ (Sigmoid Activation Function) เป็นฟังก์ชันทางคณิตศาสตร์ที่มีลักษณะเป็นตัวเอส “S-curve” หรือเรียกว่า “Sigmoid curve” ดังรูปที่ 2.4 โดยจะมีค่าระหว่าง 0 และ 1 ฟังก์ชันนี้จะใช้เมื่อต้องการทำนายความน่าจะเป็น (Probability) ของข้อมูล



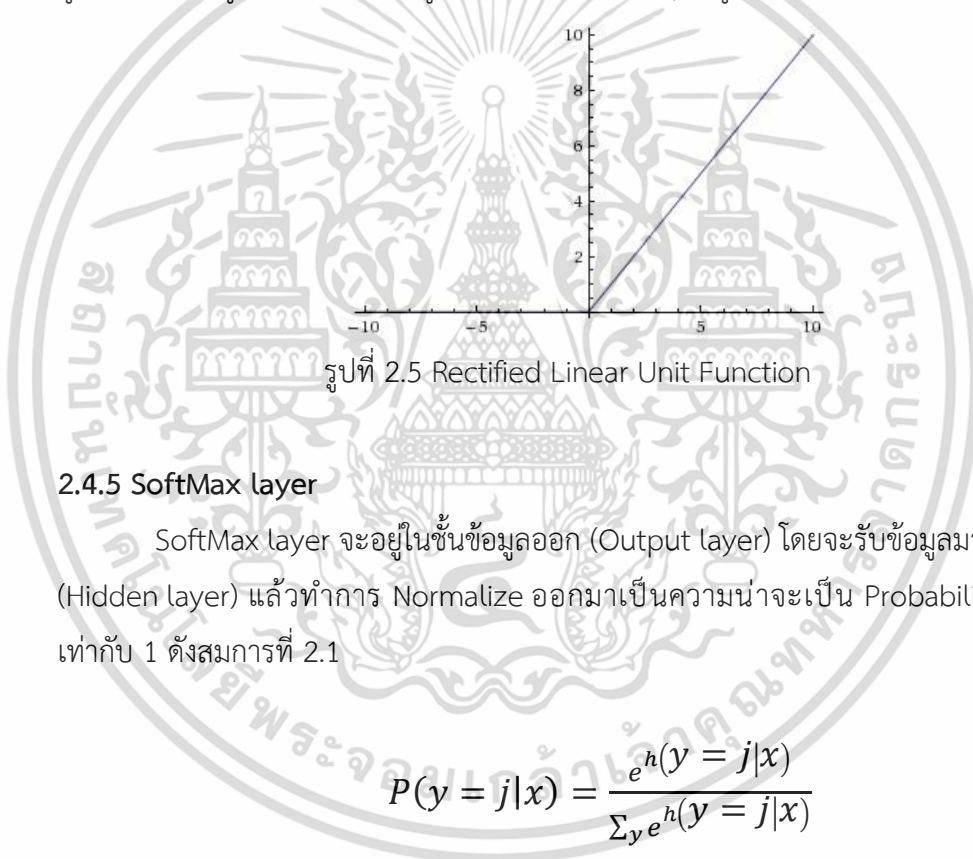
รูปที่ 2.4 Sigmoid curve

2.4.3 ฟังก์ชันไฮเพอร์โบลิกแทนกเจนท์

ฟังก์ชันไฮเพอร์โบลิกแทนกเจนท์ (Hyperbolic Tangent Function: tanh) มีการทำงานคล้ายฟังก์ชันกระตุ้นซิกมอยด์ แต่มีประสิทธิภาพดีกว่าจะมีค่าระหว่าง $[-1, 1]$ ข้อดีของฟังก์ชันไฮเพอร์โบลิกแทนกเจนท์คือ สามารถแปลงค่าข้อมูลที่มีค่าเป็นลบมากๆ ให้ข้อมูลออกมาติดลบได้ และข้อมูลที่ค่าเป็นศูนย์จะถูกแปลงเป็นข้อมูลที่มีค่าเข้าใกล้ศูนย์ (Near-zero output)

2.4.4 Rectified Linear Unit Function

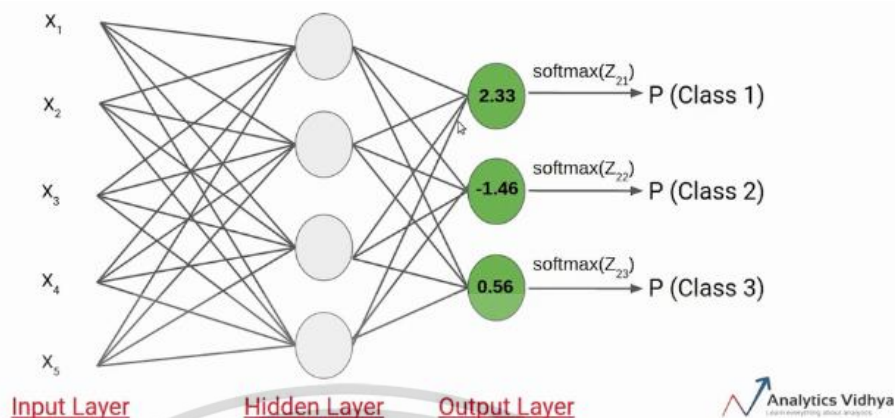
Rectified Linear Unit (ReLU) เป็นฟังก์ชันกระตุ้นที่นิยมใช้มากที่สุด โดยฟังก์ชันนี้จะมีค่าอยู่ระหว่างศูนย์ถึงจำนวนบวกใดๆ โดยที่ถ้าข้อมูลเข้า (Input) เป็นค่าลบข้อมูลออกจะเป็นศูนย์และถ้าข้อมูลเข้าเป็นบวกข้อมูลออกก็จะเป็นค่านั้นๆ ดังรูปที่ 2.5



2.4.5 SoftMax layer

SoftMax layer จะอยู่ในชั้นข้อมูลออก (Output layer) โดยจะรับข้อมูลมาจากชั้นซ่อน (Hidden layer) แล้วทำการ Normalize ออกมาเป็นความน่าจะเป็น Probability ที่ผลรวมเท่ากับ 1 ดังสมการที่ 2.1

$$P(y = j|x) = \frac{e^{h(y = j|x)}}{\sum_y e^{h(y = j|x)}} \quad (2.1)$$



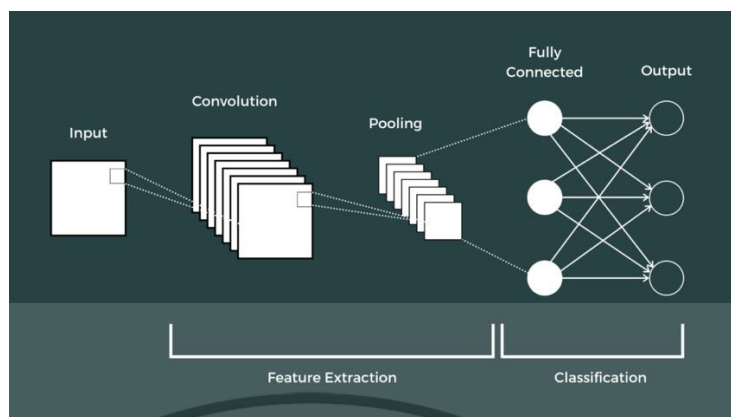
รูปที่ 2.6 SoftMax layer

2.5 การเรียนรู้เชิงลึก (Deep learning)

การเรียนรู้เชิงลึกเป็นส่วนย่อยของการเรียนรู้ของเครื่อง (Machine Learning) ซึ่งโดยพื้นฐานแล้วจะเป็นโครงข่ายประสาทเทียม (Neural network) ที่มีสามชั้นขึ้นไป โครงข่ายเทียมเหล่านี้พยายามจำลองพฤติกรรมของสมองมนุษย์ ทำให้สามารถ เรียนรู้ จากข้อมูลจำนวนมากได้ แม้ว่าโครงข่ายประสาทเทียมที่มีเพียงหนึ่งชั้นยังสามารถคาดการณ์ได้โดยประมาณ แต่จำนวนชั้นที่มากขึ้นและซับซ้อนขึ้นจะช่วยเพิ่มเติมประสิทธิภาพและปรับแต่งความแม่นยำได้ ซึ่งข้อมูลดังกล่าวจะถูกนำไปใช้ในการตรวจจบบรูปแบบ หรือจำแนกข้อมูล

2.6 Convolutional Neural Network (CNN)

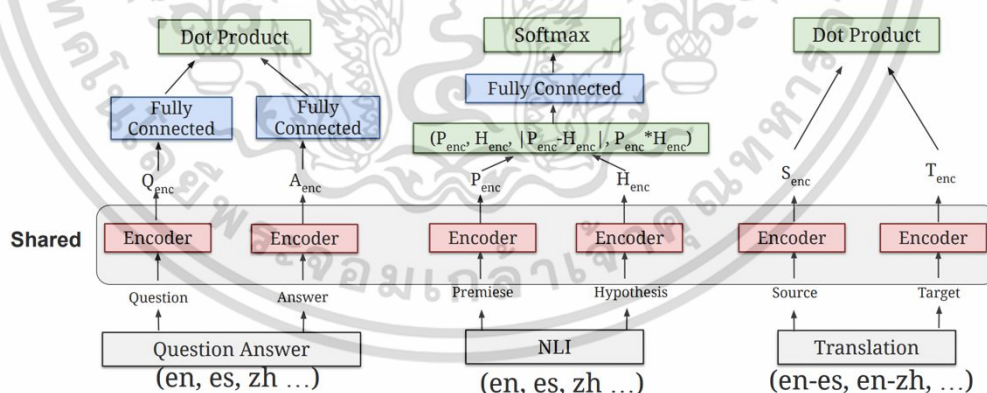
CNN เป็นการเรียนรู้เชิงลึกประเภทหนึ่งที่ถูกนำมาใช้อย่างกว้างขวาง โดยทั่วไปโครงสร้างสถาปัตยกรรมของ CNN จะประกอบไปด้วย Convolution, Pooling และ Fully connected (FC) ซึ่งสองชั้นแรกคือชั้น Convolution และ Pooling จะทำการแยกคุณลักษณะ ในขณะที่ชั้นที่สาม Fully connected เป็นชั้นที่มีโครงสร้างเหมือนกับ ANN จะจับคู่คุณลักษณะที่แยกออกมา เป็นผลลัพธ์สุดท้ายเช่น การจำแนกประเภท (Classification) ดังรูปที่ 2.7



รูปที่ 2.7 โครงสร้างสถาปัตยกรรม CNN

2.7 Universal sentence encoder

Universal sentence encoder (USE) เป็นแบบจำลองโมเดลที่มีพื้นฐานมาจาก CNN โดยสามารถใช้ได้กับ 16 ภาษาคือ อาหรับ (ar) จีน (PRC) (zh) จีน (ไต้หวัน) (zh-tw) ดัตช์ (nl) อังกฤษ (en) เยอรมัน (de) ฝรั่งเศส (fr) อิตาลี (it) โปรตุเกส (pt) สเปน (es) ญี่ปุ่น (ja) เกาหลี (ko) รัสเซีย (ru) โปแลนด์ (pl) ไทย (th) และ ตุรกี (tr) ซึ่งการฝึกฝนของ USE จะเป็นแบบ Multilingual universal sentence encoder ดังรูปที่ 2.8 ซึ่งประกอบไปด้วย Retrieval question-answering (ReQA) natural language inference (NLI) และ translation ranking การฝึกฝนเช่นนี้ทำให้ USE สามารถนำไปใช้งานได้หลายรูปแบบ



รูปที่ 2.8 โครงสร้างสถาปัตยกรรมโมเดลการฝึกฝนของ Multilingual universal sentence encoder multi-task training over

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 Recurrent Neural Network (RNN)

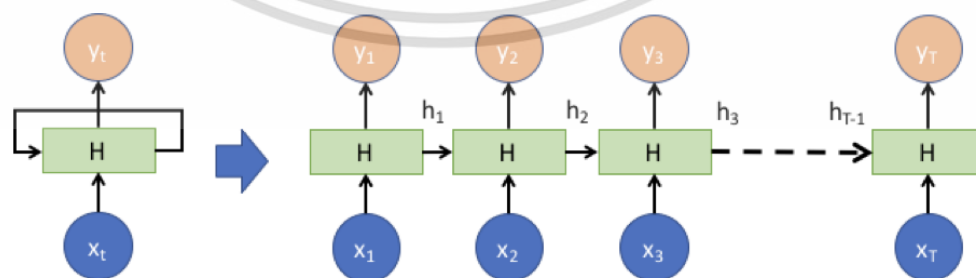
RNN คือเป็นการเรียนรู้เชิงลึกที่ใช้ในงานเกี่ยวกับข้อมูลที่เป็นลำดับ (Sequence) เช่น video (sequence of images) หรือ ข้อความ Text (sequence of words) โดย RNN จะใช้หลักการเดียวกันกับการอ่านจากซ้ายไปขวา คือ การปรับรูปแบบของ ANN เดิม เพื่อให้สามารถเอา state หรือ ความรู้ก่อนหน้า มาบวกกับข้อมูลเข้า(input data)ตัวใหม่ที่เข้ามา เพื่อทำความเข้าใจอะไรสักอย่างไปเรื่อยๆ ดังนั้น 2 ส่วนสำคัญของ RNN คือ

1) Hidden state ก่อนหน้า

2) Input data ณ ตอนนั้น

ดังภาพที่ 2.9 โดยที่

- input data คือ $x_1, x_2, \dots, x_t$
- hidden state ที่เวลา t จะใช้ตัวแปรว่า h_t
- H = hidden layer
- y_t = output จาก RNN ที่เวลา t
- x_t = input data ที่เวลา t
- h_t = hidden state ที่เวลา t

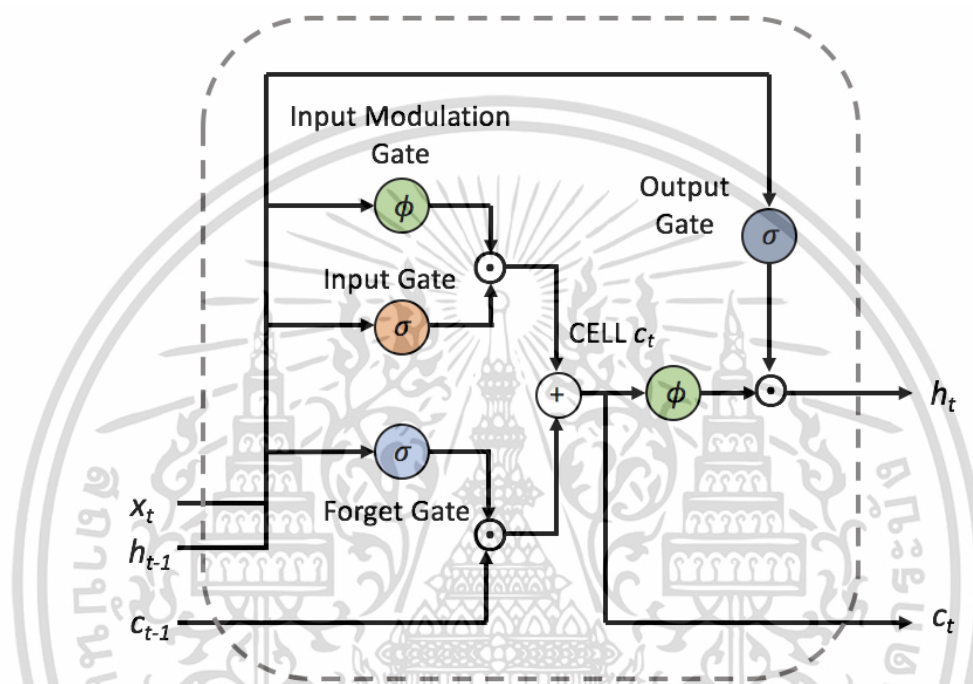


รูปภาพ 2.9 โครงสร้าง RNN

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.9 Long Short-Term Memory (LSTM)

LSTM เป็นการเรียนรู้เชิงลึกที่แก้ไขปัญหของ RNN ที่เกิดจากลำดับของข้อมูลมีความยาวมากๆ โดยโครงสร้างหลักจะมีลักษณะคล้าย RNN แต่มีรายละเอียดเยอะกว่า ดังรูปภาพที่ 2.10



รูปภาพที่ 2.10 โครงสร้างสถาปัตยกรรม LSTM

2.9.1 การทำงานของ LSTM

จะมี Cell state เป็นตัวเก็บ state ของ memory cell ใน LSTM และมี Gate เป็นตัวที่ควบคุมการไหลของข้อมูล ซึ่งก็คือ ค่า analog ที่คอยควบคุมว่าเมื่อไหร่ควรจะ read, write หรือ forget ซึ่งมันก็เหมือนกับประตูที่จะดูว่า เมื่อไหร่ควรเปิดให้ข้อมูลไหลเข้า ไหลออก หรือไหลหายไปเลย (forget) (หลักการของ gate ก็คล้ายๆ node ใน neural network ที่จะดูว่าจะ activate หรือ ไม่ activate จากค่าความแรงของสัญญาณที่เข้ามาใน node นั้น (ที่ผ่านการ weight แล้ว)

2.9.1.1 Forget

การ forget ก็เหมือนการล้าง cell state เดิมออกไป คือ จงลืมนั่นไปซะ เตรียมเคลียร์พื้นที่รับข้อมูลใหม่ แต่คนที่จะตัดสินใจว่าจะลบดี หรือไม่ลบดี ก็จะเป็นหน้าที่ของสิ่งที่เรียกว่า forget gate (อย่างที่บอกไว้ gate จะเป็นตัวควบคุมการไหลไปมาของข้อมูลข้างใน) ซึ่งเป็นเหมือนตัวตัดสินใจ ถ้า forget gate ให้ค่าเป็น 0 ก็จงลบ cell state เดิมออกไปซะ แต่ถ้า forget gate ให้ค่าเป็น 1 ก็ยินดีด้วย เราจะยังเก็บ cell state นี้ต่อไป

การสร้าง forget gate นี้ เราจะดู input data ที่เข้ามา ประกอบกับ hidden state ก่อนหน้า (ตามสูตรของ RNN) ประกอบการตัดสินใจ โดยจะใช้ sigmoid function เป็นตัวตัดสินใจ ดังสมการที่ 2.2

$$f_t = \sigma(W_x f x_t + W_h f h_t + b_f) \quad (2.2)$$

2.9.1.2 Write

เมื่อมี input data ใหม่เข้ามา เราก็ต้องมานั่งคิดว่า 1) เราอัปเดต cell state ของเราด้วย input data ใหม่ดีมั๊ย และ 2) ถ้าเราจะอัปเดตเราจะอัปเดตด้วยค่าอะไรดี มากล่าวกันทีละอย่างเลยเรื่องแรก เราจะอัปเดต cell state ของเราดีมั๊ย อันนี้จะควบคุมโดยสิ่งที่เรียกว่า input gate ตรงนี้เราก็จะยังคงใช้ sigmoid function เป็นตัวตัดสินใจว่าจะอนุญาตให้อัปเดตหรือไม่แน่นอนว่าการคำนวณนี้ใช้ค่า input data ที่เข้ามา กับ hidden state ก่อนหน้านั้น (อีกแล้ว)

$$i_t = \sigma(W_x i x_t + W_h i h_t + b_i) \quad (2.3)$$

มาถึงปัญหาที่สอง คือ แล้วถ้าเราจะอัปเดตจริงๆ เราจะอัปเดตด้วยค่าอะไรดี ซึ่งรอบนี้จะใช้สิ่งที่เรียกว่า Input modulation gate เป็นตัวจัดการ โดยสมการก็จะคล้ายๆ กับ input gate แต่ว่าจะใช้เป็น tanh function แทน ซึ่งค่าที่ได้นั้น จะมองว่าเป็น cell state candidate ก็ได้

$$g_t = \tanh(W_x c x_t + W_h c h_t + b_c) \quad (2.4)$$

2.9.1.3 อัปเดต cell state

ก่อนจะไปถึงตัวที่บอกว่าจะให้ read ขอค้นด้วยการอัปเดตค่า cell state ก่อน ตอนนี้เราได้ข้อมูลจาก forget gate, input gate และ input modulation gate แล้ว ซึ่งก็เพียงพอต่อการอัปเดต cell state ได้แล้ว ทีนี้เราก็มารวมทุกสิ่งทุกอย่างเข้าด้วยกันเลย ดังนี้

$$C_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (2.5)$$

มาดูทางฝั่งขวามือของสมการกัน เริ่มจากส่วนแรกของสมการ จะเห็นว่า ถ้า forget gate บอกว่า ให้ลบ cell state เดิมทิ้ง (f_t มีค่าเป็น 0) เราก็จะไม่เอาค่า c_{t-1} มาประกอบการอัปเดต cell state เลย คือ ลืมมันไปซะ แต่ถ้า f_t มีค่าเป็น 1 เราก็จะยังคงค่า c_{t-1} เอาไว้ประกอบการพิจารณาการอัปเดตอยู่

มาในส่วนหลังของสมการบ้าง ส่วนนี้จะเป็นส่วนของการอัปเดต cell state จากข้อมูลใหม่ ตอนนี้เรามีค่าที่จะอัปเดตเตรียมรอไว้แล้วจาก input modulation gate หรือ g_t ทีนี้เราก็จะมาดูกันว่า ค่า g_t ที่เตรียมไว้เนี่ย จะได้ใช้ หรือไม่ได้ใช้ อันนี้ก็จะใช้ output จาก input gate หรือ i_t มาตัดสิน ถ้า i_t เป็น 1 ก็เลยเอาค่า g_t อัปเดตไปเลย แต่ถ้า i_t เป็น 0 ก็เสียใจด้วย เราจะไม่ใช้ g_t ที่ dimension นั้นอัปเดตนะ จากค่าทั้งหมดที่ได้มา เราก็จะได้ค่า c_t ตัวใหม่เป็นที่เรียบร้อย

2.9.1.4 Read

จากเรื่อง RNN ตัวเดิม สิ่งที่เราต้องผลิตออกไปก็คือ hidden state ณ เวลาที่ t หรือ h_t ออกไป ซึ่งเมื่อตอนที่เวลา $t+1$ ตัว LSTM นี้จะได้อาณา h_t นี้ไปคำนวณด้วย (ตามสูตร) ดังนั้นคำว่า read จริงๆ ในที่นี้แล้ว ถ้าบอกให้ถูก มันก็คือ การที่จะอนุญาตให้คนข้างนอกมา read ตัว h_t ได้หรือเปล่า (จริงๆ ก็เหมือนตั้ง permission อย่างนึงแหละ ว่าจะมาเห็นได้มัย) หรือว่าจะเก็บเอาไว้ในใจเงียบๆ ไม่บอกใคร ไม่ส่งค่า h_t ต่อ ตรงนี้เราก็จะมี output gate มาช่วยตัดสิน โดยจะยังคงใช้สูตรเดิมกับ forget gate และ input gate อยู่ก็คือ ใช้ sigmoid function กับค่า hidden state ตัวก่อนหน้า กับ input data ที่เข้ามาในตอนนั้นนั่นเอง

$$O_t = \sigma(W_o x_t + W_o h_t + b_o) \quad (2.6)$$

นี้ ค่าจะเราจะ output เป็นค่า h_t สำหรับ sequence ถัดไป ก็จะเป็นว่า

$$h_t = o_t \Theta \tanh(c_t) \quad (2.5)$$

จะเห็นว่า ถ้า output gate ให้ตัว o_t มีค่าเป็น 0 แล้วละก็ ค่าของ h_t ก็จะมีค่าเป็น 0 คือไม่ส่งค่าใดๆ ออกไป ในขณะเดียวกัน ถ้า o_t มีค่าเป็น 1 เราก็จะคำนวณค่า h_t และส่งออกไปข้างนอก หรือก็คือ อนุญาตให้คนอื่นๆ สามารถดูค่า h_t ได้นั่นเอง

2.10 งานวิจัยที่เกี่ยวข้อง

2.10.1 Multilingual Universal Sentence Encoder for Semantic Retrieval

งานวิจัยที่เกี่ยวข้องเรื่องนี้เป็น เป็นงานวิจัยที่นำเสนอและพัฒนาโมเดลการฝังประโยคหลายภาษา ที่สามารถเรียกใช้ได้ง่ายและมีให้ใช้งานบน TensorFlow Hub และครอบคลุมถึง 16 ภาษาที่สามารถหาความหมายที่ใช้ร่วมกันได้โดยใช้การฝึกฝนข้อมูลแบบ Multi-task trained dual encoder ซึ่งเรียนรู้การแปลข้ามภาษาจะเชื่อมโยงผ่าน งานบริดจ์การแปล (Translation bridge tasks) แบบจำลองนี้ได้รับการปรับปรุงใหม่ในด้านประสิทธิภาพการแปลได้หลายภาษา และสามารถแปลข้ามภาษากันได้ (Monolingual and Cross-lingual semantic retrieval: SR) ได้รับประสิทธิภาพการแข่งขันในงานที่เกี่ยวข้องของคู่การแปลการดึงข้อมูลบิตเท็กซ์ (Bitext retrieval :BR) และการตอบคำถาม (retrieval question-answering: ReQA) โดยการนำไปใช้งานสามารถนำไปใช้ได้หลายภาษา แต่มีบางฟังก์ชันเท่านั้นที่สามารถใช้งานเฉพาะภาษาอังกฤษได้อย่างมีประสิทธิภาพ [1]

2.10.2 Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM)

งานวิจัยฉบับนี้ เป็นกระบวนการอัตโนมัติในการวิเคราะห์ความคิดเห็นข้อความจาก Twitter ที่วิเคราะห์ความคิดเห็นหรือความรู้สึกจากข้อความที่ถูกทวีต (Tweet) บนสาธารณะจากเฟลด์ (Feld) ต่างๆ เช่นการตลาด ทวีตจำนวนมากทางการเมืองกำลังโพสต์ด้วยแฮชแท็กทุกช่วงเวลาผ่านอินเทอร์เน็ตจากผู้ใช้อย่างหนึ่งไปยังอีกรายหนึ่ง การวิเคราะห์ความรู้สึกนี้เป็นงานที่ทำหายสำหรับนักวิจัย โดยส่วนใหญ่จะต้องแก้ไขการตีความบริบทซึ่งคำทวีตบางคำยากที่จะประเมินว่าอะไรคือข้อความเชิงลบและเชิงบวกจากคลังข้อมูลทวีตขนาดใหญ่ ปัญหานี้ละเมิดความสมบูรณ์ของระบบและความน่าเชื่อถือของผู้ใช้อาจลดลงอย่างมาก ในบทความนี้ จะระบุค่า

ทวิตแต่ละคำและกำลังกำหนดความหมายให้กับคำนั้นการทำงานของคุณสมบัตินี้ร่วมกับ tweet words, word2vec, stop words และรวมเข้ากับเทคนิคการเรียนรู้เชิงลึกของโมเดล Convolution neural network (CNN) และ Long short Term Memor(LSTM) อัลกอริทึมเหล่านี้สามารถระบุรูปแบบการหยุดจำนวนคำด้วยกลยุทธิ์ของมันเอง โมเดลทั้งสองนี้ได้รับการฝึกฝนมาอย่างดีและนำไปใช้กับชุดข้อมูล IMDB ซึ่งมีบทวิจารณ์ภาพยนตร์ 50,000 รายการ ด้วยการประมวลผลข้อมูล Twitter จำนวนมากเพื่อคาดการณ์ทวีตอารมณ์สำหรับการจัดประเภทด้วยวิธีการที่เสนอ กลุ่มตัวอย่างจะถูกทดลอง รวบรวมจากสภาพแวดล้อมตามเวลาจริงสามารถแยกแยะได้ดีและปรับปรุงประสิทธิภาพของระบบ ผลลัพธ์ของอัลกอริทึมการเรียนรู้เชิงลึกมีจุดประสงค์เพื่อให้คะแนนทวีตรีวิวและยังสามารถระบุบทวิจารณ์ภาพยนตร์ด้วยความแม่นยำในการทดสอบที่ 87.74% และ 88.02% [3]

2.10.3 Deep neural network-based classification model for Sentiment Analysis

งานวิจัยนี้เป็นเรื่องการจัดการประเภทอารมณ์และความรู้สึกของผู้ใช้ออนไลน์จากข้อความ โดยการวิจัยเกี่ยวกับแบบจำลองการจำแนกความรู้สึก จะอาศัยโครงข่ายประสาทเทียมเชิงลึก (Deep neural network) แบบจำลองการจำแนกตาม deep fully connected neural network (DNN), long short-term memory (LSTM), bidirectional long short-term memory (Bi-LSTM) และ Convolutional Neural Network (CNN) ถูกสร้างขึ้นเพื่อตัดสินแนวโน้มของข้อความแสดงความรู้สึกโดยนัยของผู้ใช้ จากแบบจำลอง Bi-LSTM ได้ศึกษาแบบจำลองการจำแนกประเภทของกลไกความสนใจระดับคำ ผลการทดลองในชุดข้อมูลสาธารณะแสดงให้เห็นว่าแบบจำลองการจำแนกประเภทซีรีส์ LSTM และแบบจำลองการจำแนกประเภท CNN ที่สร้างขึ้นสามารถบรรลุผลการจำแนกความรู้สึกที่ดีและผลการจำแนกประเภทนั้นดีกว่าแบบจำลอง DNN อย่างมีนัยสำคัญ แบบจำลองการจำแนกตามกลไกความสนใจของ Bi-LSTM ได้รับค่า R ที่เหมาะสมที่สุดในการระบุหมวดหมู่ที่เป็นบวก [5]

บทที่ 3

การดำเนินงานวิจัย

3.1 ความเป็นมาและความสำคัญ

ทราฟฟี่ฟองดูว์ (Traffy* Fondue) คือ แพลตฟอร์มบริหารจัดการปัญหาโดยนักวิจัยไทย สามารถช่วยหน่วยงานบริหารจัดการปัญหาได้ทันที่ พร้อมแสดงข้อมูลรายละเอียดของปัญหา ภาพหน้าจอ และพิกัดตำแหน่ง เพื่อประกอบการตัดสินใจให้เจ้าหน้าที่พร้อมเข้าแก้ไขปัญหาได้อย่างรวดเร็ว

โดยผู้พบปัญหาเพียงพูดคุยกับ LINE @traffyfondue หรือสแกน QR code หรือคลิกที่ลิงค์ <https://lin.ee/nwxfnHw> จากนั้นระบบ AI จะวิเคราะห์ข้อมูลและส่งปัญหาไปยังผู้รับผิดชอบในพื้นที่โดยตรง และมีประเภทปัญหาที่รับแจ้ง ดังนี้

1. ความสะอาด ชยะ
2. ไฟฟ้า ประปา
3. ไฟถนนเสีย
4. ถนน ทางเท้า
5. อาคารสถานที่ชำรุด
6. อุปกรณ์ครุภัณฑ์ชำรุด
7. จุดเสี่ยง
8. สาธารณภัย: น้ำท่วม ไฟไหม้ เผลาไหม้
9. ต้นไม้ กลิ่น เสีย สัตว์
10. ขึ้นทะเบียน ประชาสัมพันธ์
11. ความช่วยเหลือ
12. สุขภาพ
13. เบาะแสทุจริต
14. อื่นๆ

3.2 แนวทางการดำเนินงานวิจัย

แนวทางของการดำเนินงานวิจัยนี้ ประกอบไปด้วยขั้นตอนดังต่อไปนี้

1. การทำความเข้าใจกับข้อมูล (Data Understanding)
2. การเตรียมข้อมูล (Data Preparation)
3. การสร้างรูปแบบการทำนาย (Modeling)
4. การประเมินผล (Evaluation)
5. การสรุปผลการทดลอง (Conclusion)

3.3 การทำความเข้าใจกับข้อมูล (Data Understanding)

ข้อมูลของทราฟฟี่ฟองดูว์ เป็นข้อมูลที่เกิดขึ้นมาจากการแจ้งปัญหาจากประชาชนผ่านทาง Line@ traffyfondue โดยประชาชนจะทำการถ่ายภาพ หรือพิมพ์ปัญหาที่เจอเป็นภาษาไทย เลือกหมวดหมู่ปัญหา และส่งที่อยู่ของปัญหานั้นที่เจอ เพื่อให้ทางระบบของทราฟฟี่ฟองดูว์ จัดหมวดหมู่เพื่อแจ้งต่อไปยังองค์กรที่เกี่ยวข้อง ให้รับรู้แล้ววางแผนดำเนินการแก้ไข โดยข้อมูลที่เปิดเผยให้ประชาชนรับรู้ได้จะเป็น ข้อมูลแบบมีโครงสร้าง (Structure Data) และมีนามสกุลของไฟล์ คือ .csv ดังรูปที่ 3.1

ticket_id	type	organization	comment	coords	photo	address	district	subdistrict	province	timestamp	state
2022-HTVAVL	นำท่วม	กรุงเทพมหานคร	น้ำท่วมฝั่ง	100.63834,13.71169	https://storage.googleapis.com/traffy_public_b...	8862 ซ. ลอนเชน 39 แขวง สวนหลวง แขวงสวนหลวง ก...	สวนหลวง	สวนหลวง	กรุงเทพมหานคร	2022-09-08 14:44:51.840044+00	รอรับ เชื้อ
2022-NW68GG	คลอง	กรุงเทพมหานคร	เสนอแนะเรื่องการปล่อยน้ำคลองแปรประมาณการจากเขื่อน...	100.53606,13.79978	https://storage.googleapis.com/traffy_public_b...	363/28 ทรูรามพ์ 6 ซอย 96 แขวง บางซื่อ เขตบางซื่อ...	บางซื่อ	บางซื่อ	กรุงเทพมหานคร	2022-09-08 14:43:09.893101+00	รอรับ เชื้อ
2022-BUWEZE	ถนน, น้ำท่วม	กรุงเทพมหานคร	ถนนเป็นหลุม น้ำท่วมขัง	100.60685,13.67319	https://storage.googleapis.com/traffy_public_b...	ทางคู่ขนาน ถนน นานา-ลาดพร้าว แขวง นานา เขตบางนา ก...	บางนา	บางนาใต้	กรุงเทพมหานคร	2022-09-08 14:43:04.637022+00	รอรับ เชื้อ
2022-3FUZK3	พอร์ช, น้ำ, คลอง, น้ำท่วม	กรุงเทพมหานคร	1.น้ำเป็นน้ำขุ่น สีเทาเหม็นสุดห่วยในซอย 12 ค...	100.68132,13.68229	https://storage.googleapis.com/traffy_public_b...	17 เลี้ยวพระเกียรติ 19 ซอย 49 แขวง 10 แขวง ดลกล...	ปทุมวัน	ดลกล	กรุงเทพมหานคร	2022-09-08 14:42:24.947619+00	รอรับ เชื้อ
2022-K442H2	น้ำท่วม	กรุงเทพมหานคร	น้ำท่วมขัง เป็นเวลานานไม่ได้รับการแก้ไข	100.60789,13.67322	https://storage.googleapis.com/traffy_public_b...	6/341-4 Soi Ramkamhang 2 Bangna-Trad Rd., Do...	บางนา	บางนาใต้	กรุงเทพมหานคร	2022-09-08 14:40:29.152482+00	รอรับ เชื้อ
...	...	...	...	...	...	...	...	...	...	...	...
2021-CQ7EGC	ถนน, คลอง	เพื่อชนชาติ	ธรรมชาติของคลองมีน้ำขุ่นขี้โคลน เพราะมี...	100.61709,13.73008	https://storage.googleapis.com/traffy_public_b...	ถนนพัฒนาการ คลองตัน หมู่บ้านสวนเทวทอง 2 สวนหลวง	สวนหลวง	สวนหลวง	กรุงเทพมหานคร	2021-09-01 09:37:57.19743+00	ส่งเรื่อง แล้ว
2021-BUB6LG	พอร์ช, น้ำ	เพื่อชนชาติ	น้ำขุ่น, สีเทา, กลิ่นเหม็น, น้ำไหลเชี่ยว	100.56645,13.81663	https://storage.googleapis.com/traffy_public_b...	ลาดพร้าว ซ. 4, 3, 15 แขวงบางเขน เขตจตุจักร	จตุจักร	จตุจักร	กรุงเทพมหานคร	2021-09-01 09:37:37.867086+00	ส่งเรื่อง แล้ว
2021-ASBHV8	น้ำท่วม	กรุงเทพมหานคร	จุดเสี่ยงน้ำท่วมจุดน้ำท่วมขังบริเวณคลอง...	100.44531,13.78057	https://storage.googleapis.com/traffy_public_b...	ถนนจันทน์ คลองจันทน์ คลองจันทน์	จันทน์	จันทน์	กรุงเทพมหานคร	2021-09-01 09:33:44.447887+00	เสร็จสิ้น
2021-BUTJNC	ถนน, น้ำท่วม	กรุงเทพมหานคร	มีน้ำท่วมมาตั้งแต่เช้าจนถึงเย็น...	100.56436,13.80310	https://storage.googleapis.com/traffy_public_b...	ซอยรามคำแหง 20 จตุจักร	จตุจักร	จตุจักร	กรุงเทพมหานคร	2021-09-01 09:33:39.568655+00	เสร็จสิ้น
2021-BK3BQU	พอร์ช, น้ำ	กรุงเทพมหานคร	ขาดการระบายน้ำ...	100.51286,13.73761	https://storage.googleapis.com/traffy_public_b...	เขารวม สันติราษฎร์	สันติราษฎร์	สันติราษฎร์	กรุงเทพมหานคร	2021-09-01 09:27:37.146614+00	เสร็จสิ้น

รูปที่ 3.1 ตัวอย่างข้อมูลของทราฟฟี่ฟองดูว์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 3.1 จะพบว่าข้อมูลของกราฟฟี่ฟองดูว์ จะมีฟิลด์ (Fields) หรือคอลัมน์ (Columns) เพื่อบอกความเฉพาะหรือกำหนดเรื่องที่จะเก็บข้อมูลทั้งหมด 12 ฟิลด์ ดังนี้

1. ticket_id คือ รหัสของการแจ้งปัญหา 1 เรื่องต่อ 1 รหัส
2. type คือ ประเภทของปัญหาที่ทำการแจ้ง
3. organization คือ ชื่อขององค์กรที่อยู่ในเขตรับผิดชอบ
4. comment คือ ข้อมูลที่อธิบายปัญหาที่แจ้งจากประชาชน
5. coords คือ พิกัดของสถานที่ ที่เกิดปัญหา
6. photo คือ รูปภาพของปัญหาที่เกิดขึ้น
7. address คือ ที่อยู่ของปัญหาที่เกิดขึ้น
8. district คือ เขต หรืออำเภอของปัญหาที่เกิดขึ้น
9. subdistrict คือ แขวง หรือตำบลของปัญหาที่เกิดขึ้น
10. province คือ จังหวัด ของปัญหาที่เกิดขึ้น
11. timestamp คือ วันและเวลาที่แจ้งพบปัญหา
12. state คือ สถานะของปัญหาที่ประชาชนทำการแจ้งไปแล้ว

โดยปัญหาที่ประชาชนที่การแจ้งมา มีบางปัญหาที่ไม่มีหมวดหมู่ของปัญหา หรือมีการเลือกหมวดหมู่ของปัญหามากกว่า 1 หมวดหมู่ ซึ่งส่งผลต่อการจัดกลุ่มของปัญหาเพื่อส่งต่อให้หน่วยงานที่รับผิดชอบได้ยากลำบาก ส่งผลให้เกิดความล่าช้าในการแก้ไข ดังนั้นจึงจำเป็นต้องทำการจำแนกหมวดหมู่จาก ข้อมูลที่ประชาชนกรอกอธิบายปัญหา (comment) เพื่อให้อยู่ในหมวดหมู่ที่ถูกต้องและสามารถกำหนดขอบเขตของหน่วยงานที่รับผิดชอบได้ เพื่อให้ปัญหาที่ประชาชนแจ้งเข้าได้รับการแก้ไขจากหน่วยงานที่เกี่ยวข้องอย่างรวดเร็วมากขึ้น

โดยในชุดข้อมูลนี้เราจะสนใจ 2 ฟิลด์ (Fields) คือ ประเภทของปัญหาที่ทำการแจ้ง (type) และ ข้อมูลที่อธิบายปัญหาที่แจ้งจากประชาชน (comment) เพื่อทำการจำแนกประเภทของปัญหาจากข้อมูลที่เป็นตัวอักษร (Text Classification)

3.4 การเตรียมข้อมูล (Data Preparation)

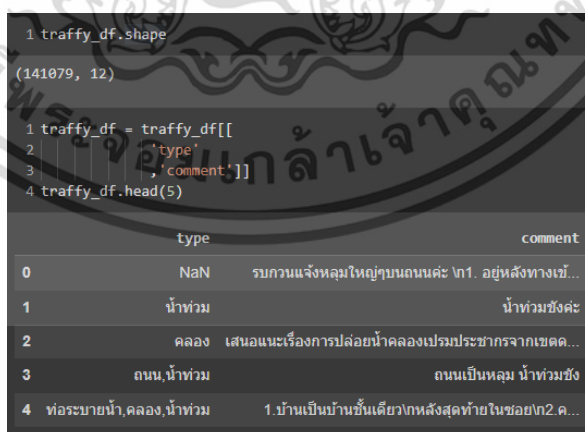
ในการเตรียมข้อมูลลักษณะประเภทตัวอักษร (Text) เพื่อทำการจำแนกประเภทหมวดหมู่ของการอธิบายในเชิงลักษณะตัวอักษร จำเป็นต้องการทำทำความสะอาดข้อมูลหรือการจัดการข้อมูลให้เหมาะสม (Data Cleaning) เพื่อให้ส่งผลที่ดีต่อโมเดล (Model) ที่จะทำการจำแนก โดยมีขั้นตอนการจัดเตรียมข้อมูลดังนี้

1. การทำความสะอาดข้อมูล (Data Cleaning)
2. การเตรียมข้อมูลสำหรับการเข้าโมเดลของ Universal Sentence Encoder (USE)
3. การเตรียมข้อมูลสำหรับการเข้าโมเดลของ การเรียนรู้เชิงลึก (Deep Learning)

3.4.1 การทำความสะอาดข้อมูล (Data Cleaning)

ในการทำความสะอาดข้อมูลเพื่อให้พร้อมสำหรับการเตรียมข้อมูลสำหรับเข้าโมเดล จำเป็นต้องศึกษาข้อมูลต้นทาง (Input) ว่ามีข้อมูลที่ขาดหายไป หรือไม่สมบูรณ์ (Missing Values) ซึ่งจะมีผลอย่างมากในเรื่องของการจำแนกประเภทจากข้อมูลลักษณะเชิงตัวอักษร

จากข้อมูลที่นำมาจากทราฟฟี่ฟองดูว์ จะพบว่าตัวข้อมูลมีทั้งหมด 141,079 แถว (Records) และ 12 ฟิลด์ (Fields) โดยเราจะเลือก 2 ฟิลด์ คือ ประเภทของปัญหาที่ทำการแจ้ง (type) และ ข้อมูลที่อธิบายปัญหาที่แจ้งจากประชาชน (comment) ดังรูปที่ 3.2 เพื่อเตรียมสำหรับการเข้าโมเดล โดย ประเภทของปัญหาที่ทำการแจ้ง (type) เป็นตัวแปรตาม (Label) และ ข้อมูลที่อธิบายปัญหาที่แจ้งจากประชาชน (comment) เป็นตัวแปรต้น (Features)



```

1 traffic_df.shape
(141079, 12)

1 traffic_df = traffic_df[[
2     'type',
3     'comment']]
4 traffic_df.head(5)

```

	type	comment
0	NaN	รบกวนแจ้งหลุมใหญ่บนถนนละ ๓๑. อยู่หลังทางเข้า...
1	น้ำท่วม	น้ำท่วมขังคะ
2	คลอง	เสนอแนะเรื่องการปล่อยน้ำคลองเปรมประชากรจากเขต...
3	ถนน, น้ำท่วม	ถนนเป็นหลุม น้ำท่วมขัง
4	ท่อระบายน้ำ, คลอง, น้ำท่วม	1.บ้านเป็นบ้านชั้นเดียวทาสีสดท้ายในซอย๓2 ค...

รูปที่ 3.2 คำสั่งการเลือกฟิลด์

หลังจากเลือกฟิลด์ที่ต้องการแล้วทำการสำรวจข้อมูลเพื่อหาข้อมูลที่ไม่ครบถ้วนหรือข้อมูลที่เกิดการสูญหาย (Missing Values) ดังตารางที่ 3.1

ตารางที่ 3.1 ตารางแสดงจำนวนข้อมูลทั้งหมดและข้อมูลที่เกิดการสูญหาย

	การนับจำนวนข้อมูล (Count)
ข้อมูลทั้งหมด	141,079
ข้อมูลที่เกิดการสูญหายของ type	62,676
ข้อมูลที่เกิดการสูญหายของ comment	221

หลังจากการสำรวจข้อมูล จะเห็นว่าข้อมูลที่เกิดการสูญหายของฟิลด์ type จะมีจำนวนที่สูงมากถึง 44.43% ของข้อมูลทั้งหมด เพื่อเป็นการเตรียมข้อมูลให้พร้อมสำหรับการเตรียมเข้าโมเดลจึงจำเป็นต้องลบแถวที่มีการสูญหายของข้อมูล ดังรูปที่ 3.3

```
1 df = traffy_df.dropna(subset=['type', 'comment']).reset_index(drop = True)
2 df
```

	type	comment
0	น้ำท่วม	น้ำท่วมขังคะ
1	คลอง	เสนอแนะเรื่องการปล่อยน้ำคลองเปรมประชากรจากเขต...
2	ถนน,น้ำท่วม	ถนนเป็นหลุม น้ำท่วมวัง
3	ท่อระบายน้ำ,คลอง,น้ำท่วม	1.บ้านเป็นบ้านชั้นเดียวทาสีสุดท้ายในซอยก2.ค...
4	น้ำท่วม	น้ำท่วมขัง เป็นเวลานานไม่ได้รับการแก้ไข
...	...	...
78390	ถนน,คลอง	ารระบายน้ำลงคลองบริเวณนั้น ช้ามาก เพราะปัจจุบัน...
78391	ท่อระบายน้ำ	ไม่มีการ ลอก ท่อ แคว้งท่อระบายน้ำให้ใหญ่ขึ้น
78392	น้ำท่วม	จุดเสี่ยงน้ำท่วม จุดเฝ้าระวัง กรณีเขตนกเครื่องบิน...
78393	ถนน,น้ำท่วม	มีน้ำท่วมบ้าง เพราะพื้นถนนสูงกว่าถนนวิภาวดี ปร...
78394	ท่อระบายน้ำ	ขาดการจัดการขยะที่มีประสิทธิภาพ ส่งผลให้เกิดกา...

78395 rows x 2 columns

รูปที่ 3.3 คำสั่งการลบแถวที่มีข้อมูลมีการสูญหาย

จากรูปที่ 3.3 จะเห็นว่าในฟิลด์ type จะมีหมวดหมู่ที่ซ้อนทับกัน เช่น ถนนกับน้ำท่วม ซึ่งส่งผลให้การส่งต่อหน่วยงานที่รับผิดชอบนั้น ซ้ำซ้อนมากขึ้นจนส่งผลทำให้หน่วยงานทำงานได้ลำบาก ดังนั้นจึงต้องทำการลบข้อมูลของแถวที่มีการใส่ประเภทของปัญหา ที่มากกว่า 1 ประเภท ดังรูปที่ 3.4

```

1 drop_values = [' ','']
2 new_df=traffy[~traffy['type'].str.contains(' '.join(drop_values))].reset_index(drop = True)
3 new_df

```

	type	comment
0	น้ำท่วม	น้ำท่วมขังคะ
1	คลอง	เสนอแนะเรื่องการปล่อยน้ำคลองเปรมประชากรจากเขต...
2	น้ำท่วม	น้ำท่วมขัง เป็นเวลานานไม่ได้รับการแก้ไข
3	กีดขวาง	รถจอดกีดขวางป้ายรถเมล์
4	น้ำท่วม	น้ำท่วมขัง
...	...	...
57988	ทอระบายน้ำ	ไม่มีทอระบายน้ำ หลังบ้านติดมิง ดอนนี้ป่าในเมิง...
57989	ถนน	ฝั่งของถนนด้านนี้พื้นที่ต่ำ อาจจะเพราะไม่ได้ล่อ...
57990	ทอระบายน้ำ	ไม่มีการ ลอก ท่อ แคววางทอระบายน้ำให้ใหญ่ขึ้น
57991	น้ำท่วม	จุดเสี่ยงน้ำท่วม จุดเฝ้าระวัง กรณีเขดยกเครื่อง...
57992	ทอระบายน้ำ	ขาดการจัดการขยะที่มีประสิทธิภาพ ส่งผลให้เกิดกา...

57993 rows x 2 columns

รูปที่ 3.4 คำสั่งการลบแถวของข้อมูลที่มีประเภทของปัญหาที่ซ้ำซ้อนกัน

หลังจากทำการลบแถวที่มีประเภทหมวดหมู่ของปัญหาที่ซ้ำซ้อนกัน หรือมีประเภทหมวดหมู่ของปัญหาที่มากกว่า 1 ประเภท จะทำการสำรวจข้อมูลแต่ละประเภทเพื่อให้ทราบถึงจำนวนข้อมูลแต่ละหมวดหมู่ก่อนนำข้อมูลไปทำการแบ่งข้อมูลสำหรับข้อมูลสำหรับฝึกฝน (Training Set) และข้อมูลสำหรับการทดสอบ (Testing Set) โดยประเภทของหมวดหมู่ของปัญหา จะมีทั้งหมด 14 ประเภท และมีจำนวนข้อมูลแต่ละประเภทดังตารางที่ 3.2

ตารางที่ 3.2 ตารางแสดงจำนวนข้อมูลของปัญหาแต่ละประเภท

ประเภทของปัญหา	จำนวนข้อมูล
ถนน	17,807
น้ำท่วม	7,473
ทางเท้า	5,925
แสงสว่าง	5,156
ความสะอาด	4,411
สายไฟ	3,541
ความปลอดภัย	2,629
กีดขวาง	2,469
ท่อระบายน้ำ	2,242
จราจร	2,070
สะพาน	1,820
คลอง	1,540
สัตว์จรจัด	623
เสนอแนะ	199

หลังจากสำรวจข้อมูลแล้วจะทำการแบ่งข้อมูลสำหรับฝึกฝน(Training Set) และ ข้อมูลสำหรับทดสอบ (Testing Set) โดยจะทำการแบ่งเป็น ข้อมูลสำหรับฝึกฝน(Training Set) 80% และข้อมูลสำหรับทดสอบ (Testing Set) 20% และจัดเก็บเป็น 2 ไฟล์เพื่อนำไปใช้ต่อไปสำหรับการเตรียมข้อมูลสำหรับเข้าโมเดล โดยใช้คำสั่งดังรูปที่ 3.5

```

1 # Get cleaned input text as a list
2 input_text = clean_df["comment"].tolist()
3 # Get labels from the cleaned dataset
4 labels = clean_df["type"].tolist()
5 # Split train test
6 train_text, test_text, train_labels, test_labels = train_test_split(input_text, labels, train_size=0.8, random_state=12345)

1 with open('train_set.pkl', 'wb') as f:
2 | pickle.dump([train_text, train_labels], f)
3 with open('test_set.pkl', 'wb') as f:
4 | pickle.dump([test_text, test_labels], f)

```

รูปที่ 3.5 คำสั่งการแบ่งข้อมูลสำหรับฝึกฝนและข้อมูลสำหรับทดสอบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.2 การเตรียมข้อมูลสำหรับการเข้าโมเดลของ Universal Sentence Encoder (USE)

การเตรียมข้อมูลสำหรับการเข้าโมเดล Universal Sentence Encoder จะต้องทำการเข้ารหัส (Encode) ข้อมูลสำหรับใช้กับโมเดล โดยจะทำการเข้ารหัสกับข้อมูลสำหรับฝึกฝน (Training Set) และข้อมูลสำหรับทดสอบ (Testing Set) ใช้คำสั่งดังรูปที่ 3.6 และตัวอย่างข้อมูลที่ถูกรหัส ดังรูปที่ 3.7

```
1 def batch_feed(texts, batch_size=32):
2     """Feed the text as batches to avoid an OOM problem"""
3     results = []
4     for i in range(0, len(texts), batch_size):
5         vectors = use_model(texts[i:i+batch_size]) # Feed to USE
6         vectors = vectors.numpy() # Convert from tf.Tensor to numpy array
7         results.append(vectors)
8     results = np.concatenate(results, axis=0)
9     return results

1 encoded_train_text = batch_feed(train_text)
2 encoded_test_text = batch_feed(test_text)
```

รูปที่ 3.6 คำสั่งการเข้ารหัสชุดข้อมูลสำหรับโมเดล Universal Sentence Encoder

```
1 encoded_train_text
array([[ -0.02191951,  0.06160262, -0.05702358, ...,  0.08492932,
        -0.05834782,  0.04123144],
       [ -0.0337523 ,  0.04193281, -0.03044064, ...,  0.03353939,
        -0.08679815,  0.079267  ],
       [ -0.0717233 ,  0.05651521, -0.00424928, ...,  0.04445213,
        -0.03607543,  0.0769323 ],
       ...,
       [ -0.07386637, -0.05067023, -0.03087322, ..., -0.02435911,
         0.00420448,  0.02933631],
       [ -0.05704545,  0.0236774 ,  0.04176341, ..., -0.0204719 ,
         0.01174034,  0.10580344],
       [ 0.03919629,  0.05118721,  0.02556839, ...,  0.06094627,
        -0.08177039,  0.06438911]], dtype=float32)
```

รูปที่ 3.7 ตัวอย่างข้อมูลที่ถูกรหัสสำหรับโมเดล Universal Sentence Encoder

3.4.3 การเตรียมข้อมูลสำหรับการเข้าโมเดลของการเรียนรู้เชิงลึก (Deep Learning)

การเตรียมข้อมูลสำหรับการเข้าโมเดลของ การเรียนรู้เชิงลึกจะต้องทำการป้อนข้อมูลขาเข้า (Input Data) เป็นข้อมูล Embedding Vector โดยจะมีขั้นตอนดังนี้

1. การตัดคำ (Word Segmentation)
2. การแปลงลำดับคำของตัวแปรต้นเป็นรายการดัชนีคำ
3. การเติมจำนวนคำของตัวแปรต้น (Pad Word Indices)
4. การแปลงคำของตัวแปรตามเป็นรายการดัชนีคำ

3.4.3.1 การตัดคำ (Word Segmentation)

ก่อนจะไปเข้าโมเดลเพื่อจำแนกประเภทของปัญหา จำเป็นจะต้องมีการตัดคำ เพื่อให้สามารถหาความน่าจะเป็นของคำแต่ละคำว่าใกล้เคียงกับความหมายของตัวแปรตามเพื่อระบุประเภทของปัญหา ดังนั้นการตัดคำจึงมีบทบาทสำคัญในการจำแนกประเภทของปัญหา แต่ในภาษาไทยนั้นการจะตัดคำเป็นไปได้อย่างมากจึงจำเป็นต้องใช้เครื่องมือเฉพาะทาง คือ PyThaiNLP โดยต้องทำการติดตั้ง library ดังรูปที่ 3.8 ซึ่งเครื่องมือนี้จะช่วยให้สามารถตัดคำได้ โดยใช้คำสั่ง ดังรูปที่ 3.9 เพื่อทำการตัดคำซึ่งจะทำการตัดคำกับชุดข้อมูลสำหรับฝึกฝน (Training Set) และชุดข้อมูลสำหรับทดสอบ (Testing Set) ซึ่งเครื่องมือที่ใช้ในการตัดคำจะมีเครื่องมือที่หลากหลายมากเช่น newmm, logest, deepcut, attacut ซึ่งแต่ละเครื่องมือจะมีความสามารถที่ต่างกันเช่นสามารถตัดคำได้แม่นยำมากแต่อาจจะแลกด้วยกับเวลานานในการตัด โดยจะเน้นใช้ในกรณีที่โมเดลนำข้อมูลที่เรานำเข้าไปที่โมเดลแล้วส่งผลให้ประสิทธิภาพของโมเดลลดลงหรือประสิทธิภาพไม่เพียงพอกับการใช้งาน แต่ในงานวิจัยนี้ จะใช้เครื่องมือชื่อ newmm ในการตัดคำเพราะสามารถตัดคำได้เร็วและค่อนข้างนิยมใช้กับข้อมูลที่มีขนาดใหญ่หรือปริมาณมาก และมีตัวอย่างผลลัพธ์หลังการตัดคำ ดังรูปที่ 3.10

```

1 !pip install pythainlp

Looking in indexes: https://pypi.org/simple, https://us-
Collecting pythainlp
  Downloading pythainlp-3.1.1-py3-none-any.whl (9.6 MB)
    | 9.6 MB 4.2 MB/s
Requirement already satisfied: requests>=2.22.0 in /usr/
Requirement already satisfied: idna<3,>=2.5 in /usr/locat
Requirement already satisfied: urllib3!=1.25.0,!1.25.1,
Requirement already satisfied: certifi>=2017.4.17 in /usr
Requirement already satisfied: chardet<4,>=3.0.2 in /usr
Installing collected packages: pythainlp
Successfully installed pythainlp-3.1.1

```

รูปที่ 3.8 การติดตั้ง PyThaiNLP

```

1 from pythainlp import word_tokenize
2
3 tokenized_train_text = [word_tokenize(text) for text in train_text]
4 tokenized_test_text = [word_tokenize(text) for text in test_text]

```

รูปที่ 3.9 คำสั่งตัดคำโดยใช้เครื่องมือ newmm

```

1 tokenized_train_text[0]

['บุคคล',
 'เรื่อน',
 ',',
 'นอน',
 'ริม',
 'ทางเท้า',
 ',',
 'และ',
 'จอด',
 'รถเข็น',
 'ซอย',
 'สุขุมวิท',
 ',',
 '24']

```

รูปที่ 3.10 ตัวอย่างข้อมูลหลังการตัดคำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.4.3.2 การแปลงลำดับคำของตัวแปรต้นเป็นรายการดัชนีคำ

ทำการแปลงลำดับคำ (Word Sequences) ให้อยู่ในรูปแบบดัชนีคำ (Indices) เพื่อให้ง่ายต่อการเติมเต็มช่องว่างของตัวอักษร (Padding) โดยใช้คำสั่งดังรูปที่ 3.11 โดยจะทำกับตัวแปรต้นของชุดข้อมูลสำหรับฝึกฝน (Training Set) และตัวแปรต้นของชุดข้อมูลสำหรับทดสอบ (Testing Set)

```
1 def map_word_index(word_seq):
2     """
3     mapping word sequence to list of indices
4     """
5     indices = []
6     for word in word_seq:
7         if word in word2vec_model.vocab:
8             indices.append(word2vec_model.vocab[word].index + 1)
9         else:
10            # consider unknown word as '' (empty string); map to index 1
11            indices.append(1)
12    return indices

1 train_word_indices = [map_word_index(word_seq) for word_seq in tokenized_train_text]
2 test_word_indices = [map_word_index(word_seq) for word_seq in tokenized_test_text]
```

รูปที่ 3.11 การแปลงลำดับคำของตัวแปรต้นเป็นรายการดัชนีคำ

3.4.3.3 การเติมจำนวนคำของตัวแปรต้น (Pad Word Indices)

การเติมคำเพื่อให้ข้อมูล เพื่อให้ข้อมูลที่แจ้งปัญหานั้นมีจำนวนคำที่เท่ากันทั้งชุดข้อมูล โดรนจะใช้จำนวนคำในชุดข้อมูลที่มากที่สุด เพื่อให้รู้จำนวนคำที่แน่นอนก่อนเข้าโมเดล โดยใช้คำสั่งดังรูปที่ 3.12

```
1 # Find maximum word length of the sentences
2 max_len = 0
3 for word_seq in tokenized_train_text:
4     if len(word_seq) > max_len:
5         max_len = len(word_seq)

1 print("Maximum word length:", max_len)

Maximum word length: 1456
```

รูปที่ 3.12 การหาจำนวนคำที่มากที่สุด

จากรูปที่ 3.12 จะเห็นว่าจำนวนคำที่มากที่สุดคือ 1,456 คำ ดังนั้นจะใช้จำนวนของคำที่มากที่สุดมาเป็นเกณฑ์ในการเพิ่มคำ โดยใช้คำสั่งดังรูปที่ 3.13 และตัวอย่างของข้อมูลหลังเติมคำดังรูปที่ 3.14

```
1 # pad sequence using keras library
2 from tensorflow.keras.preprocessing.sequence import pad_sequences
3
4 train_padded_wordinds = pad_sequences(train_word_indices, maxlen=max_len, value=0)
5 test_padded_wordinds = pad_sequences(test_word_indices, maxlen=max_len, value=0)
```

รูปที่ 3.13 คำสั่งการเติมจำนวนคำ (Pad Word Indices)

```
1 print("Word sequence:", tokenized_train_text[0])
2 print("Prepared word indices:", train_padded_wordinds[0])

Word sequence: ['มด', 'เรื้อน', ' ', 'นอน', 'ริม', 'ทางเท้า', ' ', 'และ', 'จอด', 'รถเข็น', ' ', 'ชอบ', 'สุนัข', ' ', '24']
Prepared word indices: [ 0  0  0 ... 2540  1 520]
```

รูปที่ 3.14 ตัวอย่างข้อมูลหลังเติมคำ

3.4.3.4 การแปลงคำของตัวแปรตามเป็นรายการดัชนีคำ

ทำการแปลงตัวแปรตาม (Label) ให้อยู่ในรูปแบบดัชนีคำ (Indices) เพื่อให้ง่ายต่อการทำข้อมูลเป็นข้อมูลเชิงคุณภาพ (Categorical Data) เพื่อให้เป็นตัวแปรตามของการจำแนกประเภทของปัญหา ซึ่งทำการแปลงทั้งชุดข้อมูลสำหรับฝึกฝน (Training Set) และชุดข้อมูล (Testing Set) ซึ่งจำเป็นต้องเตรียมข้อมูลก่อนการแปลงเป็นข้อมูลเชิงคุณภาพ (Categorical Data) โดยใช้คำสั่งดังรูปที่ 3.15 ซึ่งจะเลือกตัวแปรตามหรือประเภทของปัญหามาแต่ละประเภทและคำสั่งดังรูป 3.16 จะทำการ mapping ชื่อของตัวแปรตาม กับดัชนีของตัวแปรตาม และคำสั่งแสดงตัวอย่างของดัชนีของตัวแปรตามในแต่ละประโยค ดังรูปที่ 3.17

```
1 # Prepare map between label index and label name
2 unique_labels = set(train_labels)
3 index_to_label = [label for label in sorted(unique_labels)]
4 label_to_index = {label:i for i, label in enumerate(sorted(unique_labels))}
```

รูปที่ 3.15 คำสั่งเตรียมข้อมูลสำหรับ mapping ตัวแปรตาม

```
1 # Mapping label to index
2 train_label_indices = [label_to_index[label] for label in train_labels]
3 test_label_indices = [label_to_index[label] for label in test_labels]
```

รูปที่ 3.16 คำสั่ง mapping ชื่อของตัวแปรตามเป็นดัชนีคำ

```
1 # Label indices of first 10 sentences
2 train_label_indices[:10]

[6, 5, 5, 6, 2, 8, 5, 13, 7, 5]
```

รูปที่ 3.17 ตัวอย่างดัชนีคำของตัวแปรตาม

หลังจาก mapping แล้วจะทำการแปลงดัชนีคำของตัวแปรตามเป็นข้อมูลเชิงคุณภาพ (Categorical Data) โดยใช้คำสั่งดังรูปที่ 3.18 และตัวอย่างข้อมูลหลังแปลงดัชนีคำเป็นข้อมูลเชิงคุณภาพ (Categorical Data) ดังรูปที่ 3.19

```
1 # Create dummy code of label index
2 from tensorflow.keras.utils import to_categorical
3
4 train_dummy_label = to_categorical(train_label_indices, num_classes=len(unique_labels))
5 test_dummy_label = to_categorical(test_label_indices, num_classes=len(unique_labels))
```

รูปที่ 3.18 คำสั่งแปลงคำเป็นข้อมูลเชิงคุณภาพ (Categorical Data)

```
1 train_dummy_label[:10]

array([[0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.]])
dtype=float32)
```

รูปที่ 3.19 ตัวอย่างข้อมูลหลังแปลงเป็นข้อมูลเชิงคุณภาพ

3.5 การสร้างรูปแบบการจำแนกประเภท (Modeling)

ในงานวิจัยนี้จะทำการจำแนกประเภทของปัญหาจากข้อความของผู้แจ้งปัญหาเพื่อจัดหมวดหมู่ที่เหมาะสม โดยใช้เทคนิคการจำแนกประเภทข้อความ (Text Classification) โดยใช้รูปแบบสถาปัตยกรรมโมเดลพร้อมใช้งาน 1 รูปแบบ คือ Universal Sentence Encoder (USE) และสถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก 3 รูปแบบคือ Long Short-Term Memory (LSTM), Deep Neural Network (DNN) และ Convolutional Neural Network (CNN) ซึ่งมี Hyperparameters ตารางที่ 3.3

ตารางที่ 3.3 ค่า Hyperparameters ของสถาปัตยกรรมโมเดลของการเรียนรู้เชิงลึก

Hyperparameters	Value
Input Value	1,456
Embedding	(1456 , 400)
Activation Function	Softmax
Optimizer Algorithm	Adam
Learning Rate	0.001
Epochs	30
Batch SIZE	32

3.5.1 สถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE)

สถาปัตยกรรมโมเดล Universal Sentence Encoder (USE) เป็นโมเดลที่รองรับภาษาถึง 16 ภาษา ซึ่งรวมถึงภาษาไทย และสามารถใช้งานได้ง่ายแต่จำเป็นต้องทำการดาวน์โหลดโมเดลมาก่อน โดยมีคำสั่งดังรูปที่ 3.20

```

1 # The 16-language multilingual module is the default but feel free
2 # to pick others from the list and compare the results.
3 model_url = 'https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3' #@p.
4 use_model = hub.load(model_url)

```

รูปที่ 3.20 คำสั่งดาวน์โหลดโมเดล Universal Sentence Encoder (USE)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลังจากดาวน์โหลดโมเดลแล้ว จะทำการใช้เข้ารหัสข้อมูลสำหรับ Universal Sentence Encoder (USE) และนำมาใช้กับสถาปัตยกรรมโมเดล Logistic Regression โดยใช้คำสั่งดังรูปที่ 3.21

```
1 from sklearn.linear_model import LogisticRegression
2
3 model = LogisticRegression(C=4, max_iter=1000, random_state=42)

1 model.fit(encoded_train_text, train_labels)
```

รูปที่ 3.21 คำสั่งการใช้งาน Logistic Regression

3.5.2 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)

จะมีการสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก โดยใช้คำสั่งดังรูปที่ 3.22 และแสดงสรุปของสร้างสถาปัตยกรรมโมเดล ดังรูปที่ 3.23 หลังจากนั้นจะใช้คำสั่งสำหรับการฝึกฝนสถาปัตยกรรมโมเดล ดังรูปที่ 3.24

```
1 # Init the model
2 K.clear_session()
3 # def get_object_nn(word2vec_model, max_leng, num_labels):
4 def get_object_nn(max_leng, num_labels):
5     input1 = Input(shape=(max_leng,))
6     # get the embedding from pretrained word vectors
7     x = Embedding(input_dim=len(word2vec_model.vocab)+1,
8                   output_dim=word2vec_model.vector_size,
9                   weights=[ embedding_matrix ],
10                  trainable=False)(input1)
11     x = Bidirectional(LSTM(64, return_sequences=True))(x)
12     x = Bidirectional(LSTM(32, return_sequences=False))(x)
13     out = Dense(num_labels, activation='softmax')(x)
14
15     model_lstm = Model(inputs=input1, outputs=out)
16     model_lstm.compile(optimizer=Adam(0.001), loss=CategoricalCrossentropy(), metrics=[CategoricalAccuracy()])
17     return model_lstm
18
19 model_lstm = get_object_nn(max_leng=max_leng, num_labels=len(unique_labels))
20 model_lstm.summary()
```

รูปที่ 3.22 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 1456)]	0
embedding (Embedding)	(None, 1456, 400)	292474400
bidirectional (Bidirectional)	(None, 1456, 128)	238080
bidirectional_1 (Bidirectional)	(None, 64)	41216
dense (Dense)	(None, 14)	910
Total params: 292,754,606		
Trainable params: 280,280		
Non-trainable params: 292,474,400		

รูปที่ 3.23 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)

```

1 # Uncomment this section when training
2 lstm_weight_path_object_model = '/content/drive/MyDrive/traffy/Model/lstm_model.h5'
3 # Add callback method for model checkpoint (save a checkpoint when get the best accuracy)
4 callbacks_list_object_model = [
5     ModelCheckpoint(
6         lstm_weight_path_object_model,
7         save_best_only=True,
8         save_weights_only=True,
9         monitor='val_categorical_accuracy',
10        mode='max',
11        verbose=1
12    )
13 ]
14
15 verbose = 1
16 epochs = 30
17 batch_size = 32
18 print("train with {} epochs and {} batch size".format(epochs, batch_size))
19 history = model_lstm.fit(train_padded_wordinds, train_dummy_label, epochs=epochs, batch_size=batch_size, verbose=verbose,
20                          callbacks=callbacks_list_object_model,
21                          validation_data=(test_padded_wordinds, test_dummy_label))

```

รูปที่ 3.24 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.3 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)

จะมีการสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก โดยใช้คำสั่งดังรูปที่ 3.25 และแสดงสรุปของสร้างสถาปัตยกรรมโมเดล ดังรูปที่ 3.26 หลังจากนั้นจะใช้คำสั่งสำหรับการฝึกฝนสถาปัตยกรรมโมเดล ดังรูปที่ 3.27

```

1 # Init the model
2 K.clear_session()
3 # def get_object_nn(word2vec_model, max_leng, num_labels):
4 def get_object_nn(max_leng, num_labels):
5     input1 = Input(shape=(max_leng,))
6     # get the embedding from pretrained word vectors
7     x = Embedding(
8         input_dim=len(word2vec_model.vocab)+1,
9         output_dim=word2vec_model.vector_size,
10        weights=[ embedding_matrix ], # Pretrained matrix
11        input_length=max_leng,
12        trainable=False # Freeze embedding
13    )(input1)
14    x = GlobalMaxPooling1D()(x)
15    x = Dense(200, activation="tanh")(x)
16    out = Dense(num_labels, activation="softmax")(x)
17
18    model_max = Model(inputs=input1, outputs=out)
19    model_max.compile(optimizer=Adam(0.001), loss=CategoricalCrossentropy(), metrics=[CategoricalAccuracy()])
20    return model_max
21
22 model_max = get_object_nn(max_leng=max_leng, num_labels=len(unique_labels))
23
24 model_max.summary()

```

รูปที่ 3.25 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 1456)]	0
embedding (Embedding)	(None, 1456, 400)	292474400
global_max_pooling1d (GlobalMaxPooling1D)	(None, 400)	0
dense (Dense)	(None, 200)	80200
dense_1 (Dense)	(None, 14)	2814
=====		
Total params: 292,557,414		
Trainable params: 83,014		
Non-trainable params: 292,474,400		

รูปที่ 3.26 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)

```

1 cnn_weight_path_object_model='/content/drive/MyDrive/traffy/Model/Max_Pooling.h5'
2 # Add callback method for model checkpoint (save a checkpoint when get the best accuracy)
3 callbacks_list_object_model = [
4     ModelCheckpoint(
5         cnn_weight_path_object_model,
6         save_best_only=True,
7         save_weights_only=True,
8         monitor='val_categorical_accuracy',
9         mode='max',
10        verbose=1
11    )
12 ]
13
14 verbose = 1
15 epochs = 30
16 batch_size = 32
17 print("train with {} epochs and {} batch size".format(epochs, batch_size))
18 history = model_max.fit(train_padded_wordinds, train_dummy_label, epochs=epochs, batch_size=batch_size, verbose=verbose,
19                        callbacks=callbacks_list_object_model,
20                        validation_data=(test_padded_wordinds, test_dummy_label))

```

รูปที่ 3.27 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN)

3.5.4 สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)

จะมีการสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก โดยใช้คำสั่งดังรูปที่ 3.28 และแสดงสรุปของสร้างสถาปัตยกรรมโมเดล ดังรูปที่ 3.29 หลังจากนั้นจะใช้คำสั่งสำหรับการฝึกฝนสถาปัตยกรรมโมเดล ดังรูปที่ 3.30

```

1 # Init the model
2 K.clear_session()
3 # def get_object_nn(word2vec_model, max_len, num_labels):
4 def get_object_nn(max_len, num_labels):
5     input1 = Input(shape=(max_len,))
6     # get the embedding from pretrained word vectors
7     x = Embedding(
8         input_dim=len(word2vec_model.vocab)+1,
9         output_dim=word2vec_model.vector_size,
10        weights=[ embedding_matrix ], # Pretrained matrix
11        input_length=max_len,
12        trainable=False # Freeze embedding
13    )(input1)
14     x = Conv1D(128, 3, padding="valid", activation="tanh", strides=1)(x)
15     x = Conv1D(64, 3, padding="valid", activation="tanh", strides=1)(x)
16     x = GlobalMaxPooling1D()(x)
17     out = Dense(num_labels, activation="softmax")(x)
18
19     model_cnn = Model(inputs=input1, outputs=out)
20     model_cnn.compile(optimizer=Adam(0.001), loss=CategoricalCrossentropy(), metrics=[CategoricalAccuracy()])
21     return model_cnn
22
23 model_cnn = get_object_nn(max_len=max_len, num_labels=len(unique_labels))
24 model_cnn.summary()

```

รูปที่ 3.28 คำสั่งสร้างสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 1456)]	0
embedding (Embedding)	(None, 1456, 400)	292474400
conv1d (Conv1D)	(None, 1454, 128)	153728
conv1d_1 (Conv1D)	(None, 1452, 64)	24640
global_max_pooling1d (GlobalMaxPooling1D)	(None, 64)	0
dense (Dense)	(None, 14)	910
Total params: 292,653,678		
Trainable params: 179,278		
Non-trainable params: 292,474,400		

รูปที่ 3.29 สรุปสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)

```

1 # Uncomment this section when training
2 cnn_weight_path_object_model='/content/drive/MyDrive/traffy/Model/cnn_model_weight_object.h5'
3 # Add callback method for model checkpoint (save a checkpoint when get the best accuracy)
4 callbacks_list_object_model = [
5     ModelCheckpoint(
6         cnn_weight_path_object_model,
7         save_best_only=True,
8         save_weights_only=True,
9         monitor='val_categorical_accuracy',
10        mode='max',
11        verbose=1
12    )
13 ]
14
15 verbose = 1
16 epochs = 30
17 batch_size = 32
18 print("train with {} epochs and {} batch size".format(epochs, batch_size))
19 history = model_cnn.fit(train_padded_wordinds, train_dummy_label, epochs=epochs, batch_size=batch_size, verbose=verbose,
20                        callbacks=callbacks_list_object_model,
21                        validation_data=(test_padded_wordinds, test_dummy_label))

```

รูปที่ 3.30 คำสั่งฝึกฝนสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.6 การวัดประเมิณผล (Evaluation)

การทดสอบประสิทธิภาพโมเดลในงานวิจัยนี้จะใช้มาตรวัดเพื่อวัดประสิทธิภาพของโมเดล มาตรวัดที่ถูกใช้เพื่อทำทดสอบประสิทธิภาพโมเดลในแต่ละโมเดลมีทั้งหมด 6 มาตรวัด ได้แก่ Accuracy, Precision, Recall, F1-Score, Marco Average และ Weighted Average โดยมีตัวอย่างผลการประเมินประสิทธิภาพโมเดล ดังรูปที่ 3.15

	precision	recall	f1-score	support
กีดขวาง	0.984	0.243	0.389	511
คลอง	0.920	0.268	0.416	298
ความปลอดภัย	0.840	0.079	0.145	530
ความสะอาด	0.899	0.352	0.506	836
จราจร	0.989	0.213	0.350	423
ถนน	0.397	0.999	0.568	3546
ทางเท้า	0.974	0.411	0.578	1164
ท่อระบายน้ำ	0.948	0.406	0.569	448
น้ำท่วม	0.996	0.367	0.536	1508
สะพาน	1.000	0.173	0.295	375
สิ่งแวดล้อม	1.000	0.292	0.452	120
สายไฟ	0.996	0.383	0.553	723
เลนอเนาะ	0.000	0.000	0.000	37
แสงสว่าง	0.952	0.318	0.477	1062
accuracy			0.527	11581
macro avg	0.850	0.322	0.417	11581
weighted avg	0.785	0.527	0.500	11581

รูปที่ 3.31 ตัวอย่างการวัดประสิทธิภาพโมเดล

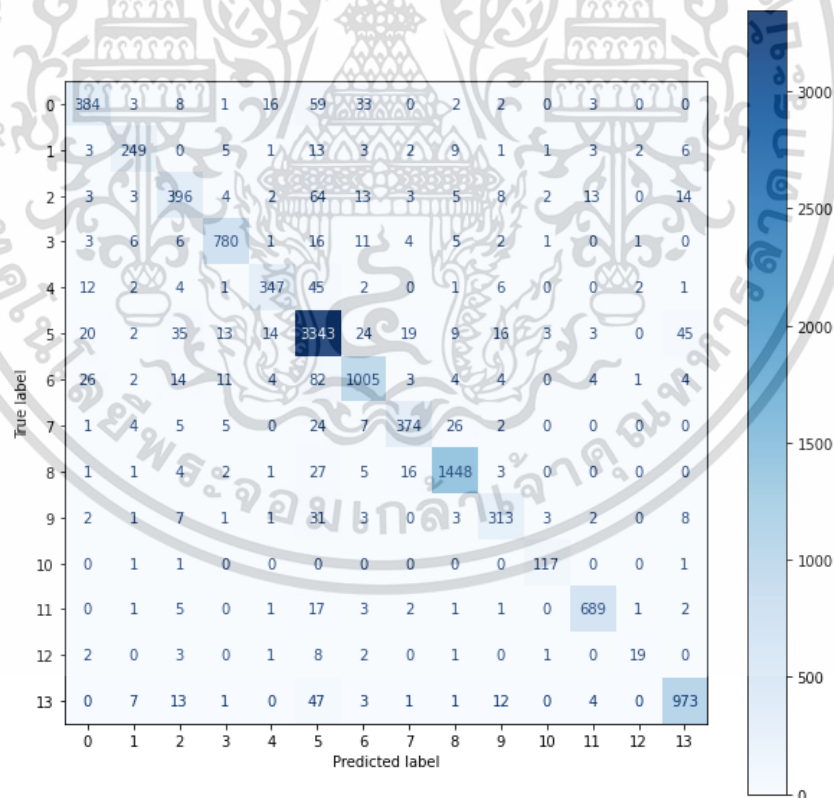
บทที่ 4

ผลการทดลองและการอภิปรายผล

ในงานวิจัยนี้จะมีการทดลองสร้างโมเดลจำแนกประเภทของหมวดหมู่ปัญหา ด้วยสถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN) และสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN) โดยจะมีทั้งหมด 4 การทดลอง

4.1 ผลการทดลองของสถาปัตยกรรมโมเดล Universal Sentence Encoder (USE)

ผลการทดลองนี้ประกอบไปด้วยกราฟ Confusion Matrix ดังรูปที่ 4.1 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหา ดังตารางที่ 4.1 และการวัดประสิทธิภาพของโมเดล ดังตารางที่ 4.2 โดยมีตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาดังรูปที่ 4.2



รูปที่ 4.1 กราฟ Confusion Matrix ของ Universal Sentence Encoder (USE)

ตารางที่ 4.1 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Universal Sentence Encoder (USE)

	Precision	Recall	F1-Score	Support
กีดขวาง	0.840	0.751	0.793	511
คลอง	0.883	0.836	0.859	298
ความปลอดภัย	0.790	0.747	0.768	530
ความสะอาด	0.947	0.933	0.940	836
จราจร	0.892	0.820	0.855	423
ถนน	0.885	0.943	0.913	3546
ทางเท้า	0.902	0.863	0.882	1164
ท่อระบายน้ำ	0.882	0.835	0.858	448
น้ำท่วม	0.956	0.960	0.958	1508
สะพาน	0.846	0.835	0.840	375
สัตว์จรจัด	0.914	0.975	0.944	120
สายไฟ	0.956	0.953	0.954	723
เสนอแนะ	0.731	0.514	0.603	37
แสงสว่าง	0.923	0.916	0.920	1062

ตารางที่ 4.2 การวัดประสิทธิภาพของโมเดล Universal Sentence Encoder (USE)

Accuracy	90.10%
Marco Average	86.30%
Weighted Average	90.00%

จากตารางที่ 4.3 และ 4.4 พบว่าโมเดล Universal Sentence Encoder (USE) มีความแม่นยำ 90.10% โดยสามารถจำแนกประเภทของ “น้ำท่วม” ได้แม่นยำมากที่สุดจากค่า F1-Score ที่ 0.958 และจำแนกประเภทของ “เสนอแนะ” ได้แม่นยำน้อยที่สุดจากค่า F1-Score ที่ 0.603

```

1 %%time
2 text = "มีคาน้ำมันคดกลั่นบนถนน เพลินจิตซาเข้าช่องทางด้านซ้าย บริเวณแยกชิดลม รบกวณหน่วยเกี่ยวข้องดำเนินการ มีจักรยานยนต์ ส้มหลายคันแล้วครับ"
3 encoded_text = use_model([text])
4 predict = model.predict(encoded_text)[0]
5 print(predict)

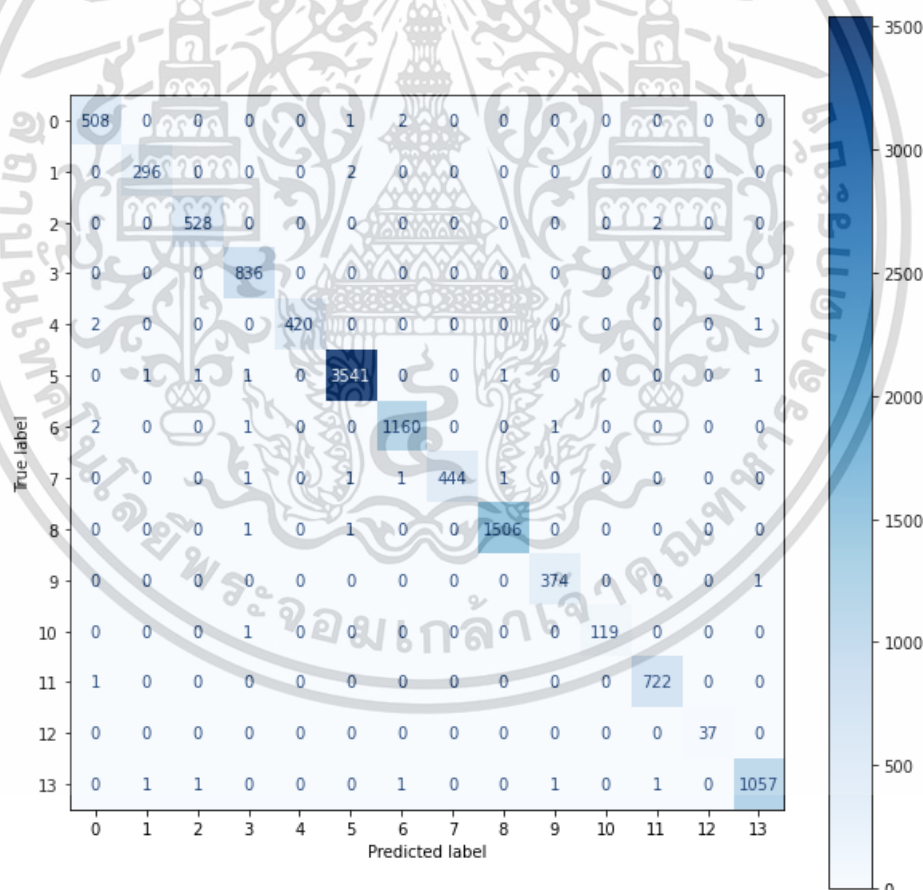
```

ถนน
CPU times: user 29.6 ms, sys: 1.08 ms, total: 30.7 ms
Wall time: 20.8 ms

รูปที่ 4.2 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Universal Sentence Encoder (USE)

4.2 ผลการทดลองของสถาปัตยกรรมโมเดล Long Short-Term Memory (LSTM)

ผลการทดลองนี้ประกอบไปด้วยกราฟ Confusion Matrix ดังรูปที่ 4.3 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหา ดังตารางที่ 4.3 และการวัดประสิทธิภาพของโมเดล ดังตารางที่ 4.4 โดยมีตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาดังรูปที่ 4.4



รูปที่ 4.3 กราฟ Confusion Matrix ของ Long Short-Term Memory (LSTM)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.3 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Long Short-Term Memory (LSTM)

	Precision	Recall	F1-Score	Support
กีดขวาง	0.990	0.994	0.992	511
คลอง	0.993	0.993	0.993	298
ความปลอดภัย	0.996	0.996	0.996	530
ความสะอาด	0.994	1.000	0.997	836
จราจร	1.000	0.993	0.996	423
ถนน	0.999	0.999	0.999	3546
ทางเท้า	0.997	0.997	0.997	1164
ท่อระบายน้ำ	1.000	0.991	0.996	448
น้ำท่วม	0.999	0.999	0.999	1508
สะพาน	0.995	0.997	0.996	375
สัตว์จรจัด	1.000	0.992	0.996	120
สายไฟ	0.996	0.999	0.997	723
เสนอแนะ	1.000	1.000	1.000	37
แสงสว่าง	0.997	0.995	0.996	1062

ตารางที่ 4.4 การวัดประสิทธิภาพของโมเดล Long Short-Term Memory (LSTM)

Accuracy	99.70%
Marco Average	99.60%
Weighted Average	99.70%

จากตารางที่ 4.3 และ 4.4 พบว่าโมเดล Long Short-Term Memory (LSTM) มีความแม่นยำ 99.70% โดยสามารถจำแนกประเภทของ “เสนอแนะ” ได้แม่นยำมากที่สุดจากค่า F1-Score ที่ 1.000 และจำแนกประเภทของ “กีดขวาง” ได้แม่นยำน้อยที่สุดจากค่า F1-Score ที่ 0.992

```

1 %%time
2 text = "มีคาน้ำมันดอกกลีบมนถนน เพลินจิตเข้าช่องทางด้านซ้าย บริเวณแยกชิดลม รบกันหน่วยเกี่ยวข้องดำเนินการ มีจักรยานยนต์ ล้มหลายคันแล้วครับ"
3 # tokenize
4 word_seq = word_tokenize(text)
5 # map_index
6 word_indices = map_word_index(word_seq)
7 # padded to max_len
8 padded_wordindices = pad_sequences([word_indices], maxlen=max_len, value=0)
9 # predict to get logit
10 logit = model_lstm.predict(padded_wordindices, batch_size=32)
11 # get prediction
12 predict = [ index_to_label[pred] for pred in np.argmax(logit, axis=-1) ][0]
13 print(predict)

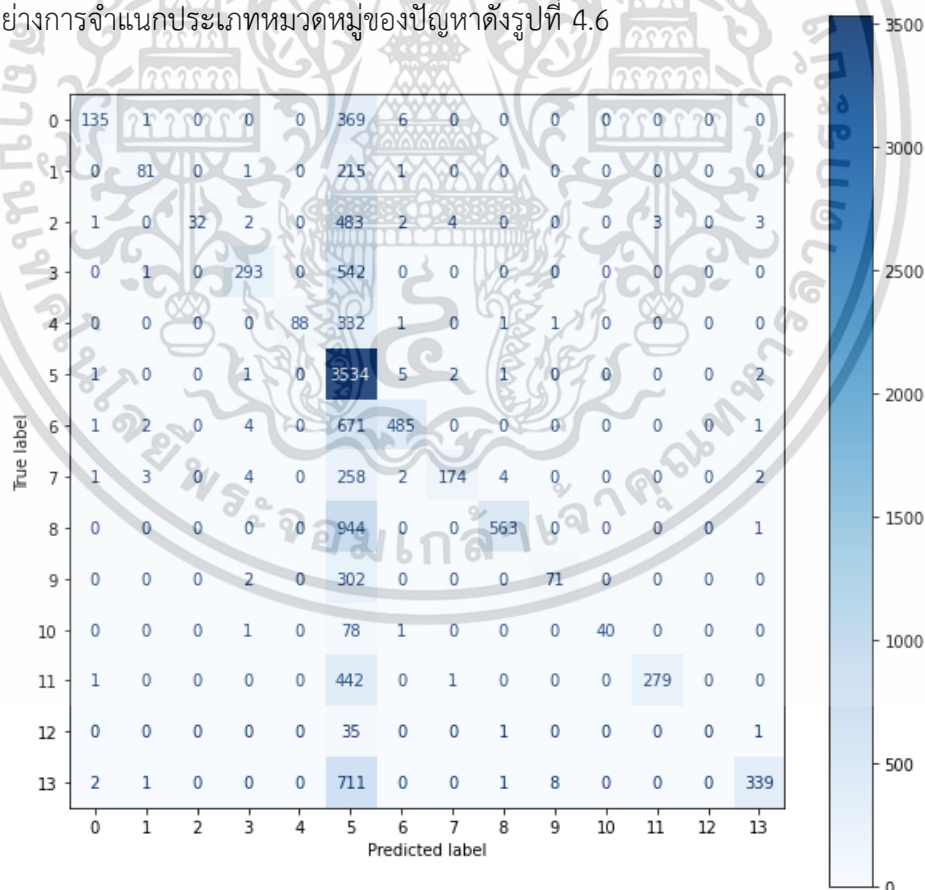
1/1 [=====] - 0s 136ms/step
ก็ต๋วาว
CPU times: user 175 ms, sys: 4.82 ms, total: 179 ms
Wall time: 174 ms

```

รูปที่ 4.4 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Long Short-Term Memory (LSTM)

4.3 ผลการทดลองของสถาปัตยกรรมโมเดล Deep Neural Network (DNN)

ผลการทดลองนี้ประกอบไปด้วยกราฟ Confusion Matrix ดังรูปที่ 4.5 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหา ดังตารางที่ 4.5 และการวัดประสิทธิภาพของโมเดล ดังตารางที่ 4.6 โดยมีตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาดังรูปที่ 4.6



รูปที่ 4.5 กราฟ Confusion Matrix ของ Deep Neural Network (DNN)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.5 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Deep Neural Network (DNN)

	Precision	Recall	F1-Score	Support
กีดขวาง	0.951	0.264	0.413	511
คลอง	0.910	0.272	0.419	298
ความปลอดภัย	1.000	0.060	0.114	530
ความสะอาด	0.951	0.350	0.512	836
จราจร	1.000	0.208	0.334	423
ถนน	0.396	0.997	0.567	3546
ทางเท้า	0.964	0.417	0.582	1164
ท่อระบายน้ำ	0.961	0.338	0.553	448
น้ำท่วม	0.986	0.373	0.542	1508
สะพาน	0.887	0.189	0.312	375
สัตว์จรจัด	1.000	0.333	0.500	120
สายไฟ	0.989	0.386	0.555	723
เสนอแนะ	0.000	0.000	0.000	37
แสงสว่าง	0.971	0.319	0.481	1062

ตารางที่ 4.6 การวัดประสิทธิภาพของโมเดล Deep Neural Network (DNN)

Accuracy	52.80%
Marco Average	42.10%
Weighted Average	50.20%

จากตารางที่ 4.5 และ 4.6 พบว่าโมเดล Deep Neural Network (DNN) มีความแม่นยำ 52.80% โดยสามารถจำแนกประเภทของ “ทางเท้า” ได้แม่นยำมากที่สุดจากค่า F1-Score ที่ 0.582 และจำแนกประเภทของ “เสนอแนะ” ได้แม่นยำน้อยที่สุดจากค่า F1-Score ที่ 0.000

```

1 %%time
2 text = "มีคาน้ำมันคอกหล่นบนถนน เหลลนจิตขาเข้าช่องทางด่วนซ้าย บริเวณแยกชิดลม รบกวหน่วยเกี่ยวข้องดำเนินการ มีจักรยานยนต์ ส้มหลายคันแล้วครับ"
3 # tokenize
4 word_seq = word_tokenize(text)
5 # map index
6 word_indices = map_word_index(word_seq)
7 # padded to max_leng
8 padded_wordindices = pad_sequences([word_indices], maxlen=max_leng, value=0)
9 # predict to get logit
10 logit = model_max.predict(padded_wordindices, batch_size=32)
11 # get prediction
12 predict = [ index_to_label[pred] for pred in np.argmax(logit, axis=1) ][0]
13 print(predict)

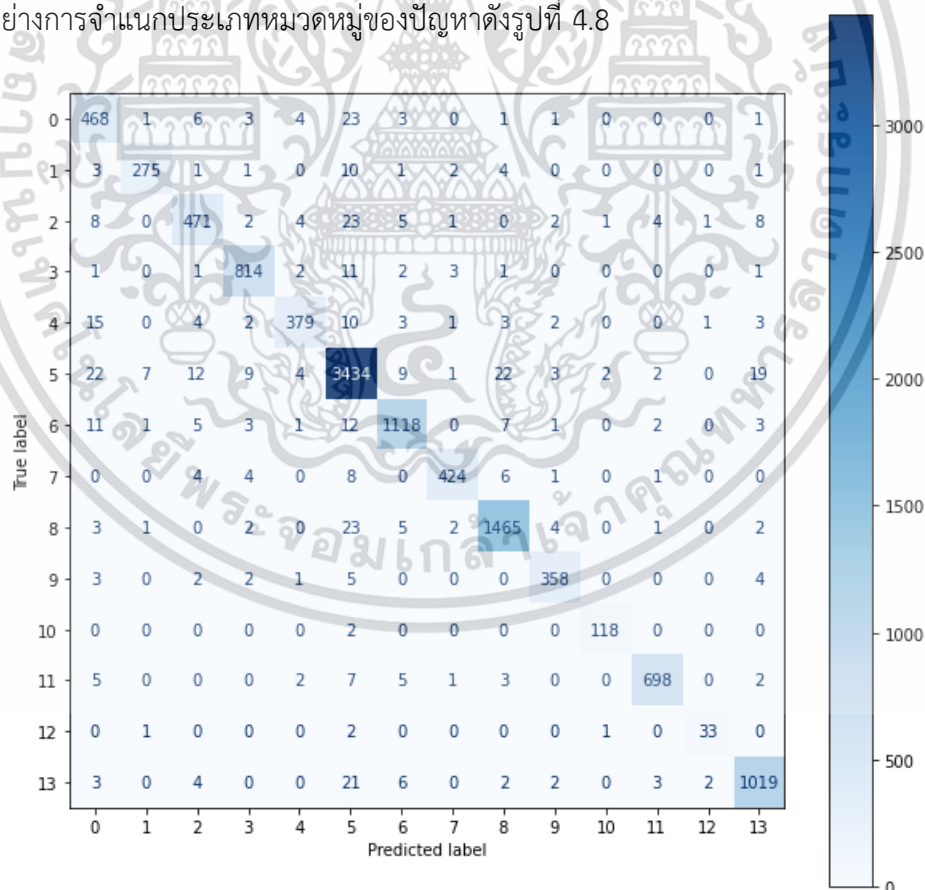
1/1 [=====] - 0s 18ms/step
ณน
CPU times: user 61.7 ms, sys: 7.11 ms, total: 68.8 ms
Wall time: 63.5 ms

```

รูปที่ 4.6 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Deep Neural Network (DNN)

4.4 ผลการทดลองของสถาปัตยกรรมโมเดล Convolutional Neural Network (CNN)

ผลการทดลองนี้ประกอบไปด้วยกราฟ Confusion Matrix ดังรูปที่ 4.7 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหา ดังตารางที่ 4.7 และการวัดประสิทธิภาพของโมเดล ดังตารางที่ 4.8 โดยมีตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาดังรูปที่ 4.8



รูปที่ 4.7 กราฟ Confusion Matrix ของ Convolutional Neural Network (CNN)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 4.7 การวัดประสิทธิภาพการจำแนกประเภทหมวดหมู่ของปัญหาของ Convolutional Neural Network (CNN)

	Precision	Recall	F1-Score	Support
กีดขวาง	0.863	0.916	0.889	511
คลอง	0.962	0.923	0.942	298
ความปลอดภัย	0.924	0.889	0.906	530
ความสะอาด	0.967	0.974	0.970	836
จราจร	0.955	0.896	0.924	423
ถนน	0.956	0.968	0.962	3546
ทางเท้า	0.966	0.960	0.963	1164
ท่อระบายน้ำ	0.975	0.946	0.960	448
น้ำท่วม	0.968	0.971	0.970	1508
สะพาน	0.957	0.955	0.956	375
สัตว์จรจัด	0.967	0.983	0.975	120
สายไฟ	0.982	0.965	0.974	723
เสนอแนะ	0.892	0.892	0.892	37
แสงสว่าง	0.959	0.960	0.959	1062

ตารางที่ 4.8 การวัดประสิทธิภาพของโมเดล Convolutional Neural Network (CNN)

Accuracy	95.60%
Marco Average	94.60%
Weighted Average	95.60%

จากตารางที่ 4.7 และ 4.8 พบว่าโมเดล Convolutional Neural Network (CNN) มีความแม่นยำ 95.60% โดยสามารถจำแนกประเภทของ “สัตว์จรจัด” ได้แม่นยำมากที่สุดจากค่า F1-Score ที่ 0.975 และจำแนกประเภทของ “กีดขวาง” ได้แม่นยำน้อยที่สุดจากค่า F1-Score ที่ 0.889

```

1 %%time
2 text = "มีคาน้ำร้อนคอกหล่นบนถนน เพลินจิตเข้าช่องทางด้านซ้าย บริเวณแยกชิดลม รบกวนหน่วยเกี่ยวข้องดำเนินการ มีจักรยานยนต์ ล้มหลายคันแล้วครับ"
3 # tokenize
4 word_seq = word_tokenize(text)
5 # map index
6 word_indices = map_word_index(word_seq)
7 # padded to max_leng
8 padded_wordindices = pad_sequences([word_indices], maxlen=max_leng, value=0)
9 # predict to get logit
10 logit = model_cnn.predict(padded_wordindices, batch_size=32)
11 # get prediction
12 predict = [ index_to_label[pred] for pred in np.argmax(logit, axis=1) ][0]
13 print(predict)

1/1 [=====] - 0s 19ms/step
จากร
CPU times: user 57 ms, sys: 3.11 ms, total: 60.1 ms
Wall time: 57.3 ms

```

รูปที่ 4.8 ตัวอย่างการจำแนกประเภทหมวดหมู่ของปัญหาโดย Convolutional Neural Network (CNN)

จากตารางที่ 4.2, 4.4, 4.6 และ 4.8 สามารถสร้างตารางเพื่อเปรียบเทียบค่าประสิทธิภาพของสถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN) และสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN) ดังตารางที่ 4.9

ตารางที่ 4.9 ตารางเปรียบเทียบประสิทธิภาพของสถาปัตยกรรมโมเดลและเวลา

	Accuracy	Marco Average	Weighted Average	Time
USE	90.10%	86.30%	90.00%	5 min
LSTM	99.70%	99.60%	99.70%	168 min
DNN	52.80%	42.10%	50.20%	10 min
CNN	95.60%	94.60%	95.60%	20 min

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงานวิจัย

จากผลการทดลองการเปรียบเทียบประสิทธิภาพของการจำแนกประเภทหมวดหมู่ของปัญหาจากกราฟฟี่ฟองดูว์ (Traffy* Fondue) ระหว่างสถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM), สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN) และ สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN) จะมีค่าความแม่นยำและเวลาที่แตกต่างกัน โดยสถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE) มีค่า Accuracy 90.10%, Marco Average 86.30% และ Weighted Average 90.00% โดยใช้เวลาดำเนินการทั้งสิ้น 5 นาที สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM) มีค่า Accuracy 99.70%, Marco Average 99.60% และ Weighted Average 99.70% โดยใช้เวลาดำเนินการทั้งสิ้น 168 นาที หรือ 2 ชั่วโมง 48 นาที สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN) มีค่า Accuracy 52.80%, Marco Average 42.10% และ Weighted Average 50.20% โดยใช้เวลาดำเนินการทั้งสิ้น 10 นาที และสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Convolutional Neural Network (CNN) มีค่า Accuracy 95.60%, Marco Average 94.60% และ Weighted Average 95.60% โดยใช้เวลาดำเนินการทั้งสิ้น 20 นาที

จากการทดลองเปรียบเทียบการเปรียบเทียบสถาปัตยกรรมโมเดลทั้ง 4 รูปแบบ จะพบว่าสถาปัตยกรรมโมเดลที่ให้ความแม่นยำมากที่สุดคือ สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM) และสถาปัตยกรรมโมเดลที่ให้ความแม่นยำน้อยที่สุดคือ สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Deep Neural Network (DNN) และในเชิงความเร็วในการฝึกฝนข้อมูลจะพบว่าสถาปัตยกรรมโมเดลที่ฝึกฝนได้เร็วที่สุดคือ สถาปัตยกรรมโมเดลพร้อมใช้งาน Universal Sentence Encoder (USE) และสถาปัตยกรรมโมเดลที่ฝึกฝนได้ช้าที่สุดคือ สถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM)

โดยในงานวิจัยนี้จะเลือกสถาปัตยกรรมโมเดลการเรียนรู้เชิงลึก Long Short-Term Memory (LSTM) ไปจำแนกหมวดหมู่ของปัญหาจากกราฟฟี่ฟองดูว์ (Traffy* Fondue) ที่ไม่สมบูรณ์เพื่อระบุให้หน่วยงานสามารถเข้าถึงปัญหาได้รวดเร็วมากขึ้น

5.2 ข้อเสนอแนะ

เนื่องจากข้อจำกัดด้านเวลา และทรัพยากร เลยส่งผลให้งานวิจัยนี้สามารถนำไปต่อยอดในอนาคตได้ โดยผู้วิจัยมีความเห็นว่าสามารถปรับปรุงและพัฒนาได้ ดังนี้

5.2.1 การใช้เครื่องมือในการตัดคำ

จากการวิจัยนี้ ผู้วิจัยใช้เครื่องมือในการตัดคำชื่อ newmm ซึ่งเป็นเครื่องมือที่ใช้ตัดคำได้เร็วที่สุดแต่จะให้ผลลัพธ์ของการตัดคำไม่ดีเท่าที่ควร เมื่อเทียบกับเครื่องมือ attacut และ deepcut ที่จะสามารถตัดคำได้มีประสิทธิภาพมากขึ้น แต่แลกมาด้วยการใช้เวลาที่มากขึ้นในการตัดคำ

5.2.2 การจัดความสำคัญของปัญหา (Priority Organization)

จากการวิจัยพบว่า ปัญหาต่างๆที่เกิดขึ้นนั้นเป็นปัญหาจริง ซึ่งมีมากมายเกินกว่าหน่วยงานจะจัดการได้อย่างรวดเร็ว ผู้วิจัยเห็นว่าสามารถพัฒนาโดยใช้ข้อมูลจากหน่วยงานอื่นๆและสื่อสาธารณะ (Social Listening) เพื่อมากำหนดความสำคัญของปัญหาที่แจ้งจากประชาชน หรือดูความถี่ของปัญหาที่แจ้ง ณ ช่วงเวลานั้น เพื่อเร่งแก้ปัญหาที่จำเป็นเพื่อช่วยเหลือประชาชนได้รวดเร็วขึ้น

บรรณานุกรม

- [1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. (2020). Multilingual Universal Sentence Encoder for Semantic Retrieval . doi: [10.18653/v1/2020.acl-demos.12](https://doi.org/10.18653/v1/2020.acl-demos.12) –
- [2] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. (2018). Universal sentence encoder for English.. doi: [10.18653/v1/D18-2029](https://doi.org/10.18653/v1/D18-2029) -
- [3] Usha Devi Gandhi · ,Priyan Malarvizhi Kumar · ,Gokulnath Chandra Babu · , and Gayathri Karthick.(2021). Sentiment Analysis on Twitter Data by Using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM). doi: [10.1007/s11277-021-08580-3](https://doi.org/10.1007/s11277-021-08580-3) –
- [4] Jianqiang, Z. H. A. O., & Xiaolin, G. U. I., and Zhang XUEJUN. (2021), deep convolution neural networks for twitter sentiment analysis. doi: [10.1109/ACCESS.2017.2776930](https://doi.org/10.1109/ACCESS.2017.2776930)–
- [5] [Donghang Pan](#), [Jingling Yuan](#), [Lin Li](#), and [Deming Sheng](#) .(2021). Deep neural network-based classification model for Sentiment Analysis. doi: [10.48550/arXiv.1907.02046](https://doi.org/10.48550/arXiv.1907.02046)