ระบบแนะนำหมวดหมู่ร้องเรียนของหน่วยงานราชการโดยใช้แบบจำลองการเรียนรู้เชิงลึก

COMPLAINT CATEGORY RECOMMENDATION SYSTEM OF STATE AGENCIES USING DEEP LEARNING MODEL

ธนพนธ์ วิจิตรศรีกมล

THANAPON VICHITSRIKAMOL

การค้นคว้าอิสระนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการข้อมูลและการวิเคราะห์
ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2567

KMITL-2024-SC-M-017-039

COMPLAINT CATEGORY RECOMMENDATION SYSTEM OF STATE AGENCIES USING DEEP LEARNING MODEL

THANAPON VICHITSRIKAMOL

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
2024

KMITL-2024-SC-M-017-039

หัวข้อการค้นคว้าอิสระ	ระบบแนะนำหมวดหมู่ร้องเรียนของหน่วยงานราชการโดยใช้แบบจำลองการเรียนรู้เชิงลึก
ชื่อนักศึกษา	ธนพนธ์ วิจิตรศรีกมล
รหัสประจำตัว	63605088
ปริญญา	วิทยาศาสตรมหาบัณฑิต (วิทยาการข้อมูลและการวิเคราะห์) ศูนย์วิเคราะห์ข้อมูลดิจิทัลอัจฉริยะพระจอมเกล้าลาดกระบัง
พ.ศ.	2567
อาจารย์ที่ปรึกษาการค้นคว้าอิสระ	ผู้ช่วยศาสตราจารย์ ดร.วรางคณา กิ้มปาน

บทคัดย่อ

เนื่องจากมีการส่งเรื่องร้องเรียนจากประชาชนไปยังหน่วยงานที่ไม่ถูกต้องส่งผลให้มีปัญหาคงค้างไม่ได้รับการแก้ไข เพื่อการแก้ปัญหาดังกล่าวการค้นคว้าอิสระนี้จึงนำเสนอระบบแนะนำหมวดหมู่ร้องเรียนที่มีแนวคิดการสร้างแบบจำลองการแยกประเภทหัวข้อและรายละเอียดการร้องเรียนว่าควรส่งไปยังหน่วยงานใดโดยมีรายละเอียดงานทั้งหมด 22 ประเภทงานจาก 5 หน่วยงาน ทั้งนี้ได้นำแบบจำลอง LSTM และ WangchanBERTa มาทำการเปรียบเทียบประสิทธิภาพ จากการวัดผลค่าความแม่นยำและค่าเฉลี่ยมาโคร พบว่าแบบจำลอง WangchanBERTa ได้ค่าความแม่นยำมากที่สุดคือ ร้อยละ 95.00 และค่าเฉลี่ยมาโคร ร้อยละ 87.00 จากแบบจำลอง LSTM Unbalance ได้ค่าความแม่นยำ ร้อยละ 90.73 และค่าเฉลี่ยมาโคร ร้อยละ 75.77 ถัดมาเป็นแบบจำลอง LSTM balance by class weight ได้ค่าความแม่นยำ ร้อยละ 88.64 และค่าเฉลี่ยมาโคร ร้อยละ 76.14

คำสำคัญ : ค่าความแม่นยำ ค่าเฉลี่ยมาโคร แบบจำลอง LSTM แบบจำลอง WangchanBERTa ระบบแนะนำหมวดหมู่ร้องเรียน

Independent Study Title	Complaint Category Recommendation System of State Agencies Using Deep Learning Model
Student Name	Thanapon Vichitsrikamol
Student ID	63605088
Degree	Master of Science (Data Science and Analytics)
	KMITL Digital Analytics and Intelligence Center
Year	2024
Independent Study Advisor	Asst.Prof.Dr. Warangkhana Kimpan

Abstract

The misrouting of complaints from the public to the wrong government department is a common occurrence. This can lead to problems not being solved or being delayed. To address this issue, this independent study proposes a complaint category recommendation system for classifying the topic and details of complaints to determine which department they should be sent to. The study used a dataset of 22 types of work from 5 departments. The LSTM and WangchanBERTa models were compared the efficiency based on the accuracy and macro average scores. The results showed that the WangchanBERTa model had the highest accuracy of 95.00% and a macro average of 87.00%. The LSTM Unbalance model had an accuracy of 90.73% and a macro average of 75.77%. The LSTM balance by class weight model had an accuracy of 88.64% and a macro average of 76.14%.

Keywords : accuracy, macro-Average, LSTM model, WangchanBERTa model, complaint category recommendation system

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของงานวิจัย/ปัญหา

ในหน่วยงานของทางราชการนั้นประชาชนเกิดปัญหาต่อการส่งคำร้องหรือเรื่องร้องเรียนต่าง ๆ เพื่อขอใช้บริการหรือขอความช่วยเหลือเป็นจำนวนมาก เนื่องจากเรื่องร้องเรียนถูกจัดส่งไปยังหน่วยงานที่ไม่ถูกต้อง พนักงานภายในของแต่ละหน่วยงานจึงจำเป็นต้องดำเนินการแยกประเภทคำร้องด้วยตัวเองหรือในบางกรณีคำร้องนั้นอาจไม่ได้รับการตอบสนองเพราะตกค้างจากการที่ส่งคำร้องไปผิดหน่วยงานและไม่ได้ประสานงานส่งปัญหาไปยังหน่วยงานที่รับผิดชอบ

ภายในแต่ละหน่วยงานของทางราชการ เจ้าหน้าที่มีหน้าที่รับผิดชอบต่อปัญหาที่ประชาชนเรียกร้องจำนวนมาก เกิดปัญหาในคัดแยกประเภทปัญหาที่ส่งมา จึงทำให้เกิดแนวคิดการจัดแยกประเภทข้อความ (Text Classification) ของระบบแนะนำหมวดหมู่ร้องเรียน (Complaint Category Recommendation System) โดยผู้ใช้งานที่ส่งคำร้องสามารถกรอกข้อมูลที่หน้าเว็บไซต์ซึ่งต้องเขียนหัวข้อคำร้อง (Subject) และรายละเอียดคำร้อง (Detail) แล้วคำร้องนั้นถูกทำนายออกมาว่าเป็นหน่วยงานใดต้องรับผิดชอบจากหัวข้อเรื่องใด

การพัฒนาและวิจัยในเรื่องนี้ช่วยทำนายประเภทหมวดหมู่คำร้องว่าคำร้องนี้อยู่ในหมวดหมู่ของหน่วยงานและงานประเภทใด ทำให้ลดปัญหาการส่งคำร้องหรือร้องเรียนผิดประเภทลงได้ ซึ่งการวิจัยและพัฒนาในด้านนี้มีประโยชน์ในการลดความซ้ำซ้อนและเพิ่มประสิทธิภาพในการประสานงานระหว่างหน่วยงานราชการต่าง ๆ และประชาชนที่มีความต้องการในการร้องเรียนหรือขอความช่วยเหลือจากหน่วยงานรัฐบาล

1.2 วัตถุประสงค์ของงานวิจัย

1) ศึกษาการจัดแยกประเภทข้อความเพื่อนำไปพัฒนาระบบแนะนำหมวดหมู่ร้องเรียน
2) ศึกษาแบบจำลอง (Model) ที่ให้ประสิทธิภาพดีกับการทำนายผลที่ถูกต้อง โดยการเปรียบเทียบแบบจำลองของ LSTM และ WangchanBERTa
3) ทดลองแบบจำลอง LSTM และ WangchanBERTa โดยนำแบบจำลองไปสร้างเว็บไซต์อย่างง่าย

1.3 ขอบเขตของงานวิจัย

1) หมวดหมู่ที่เลือกมามีจำนวน 22 หมวดหมู่จากประเภทงานที่ประชาชนร้องเรียน
2) โดยแต่ละหมวดหมู่ต้องเป็นประเภทหมวดหมู่ที่มีข้อมูลเกิน 50 แถว เนื่องจากแบบจำลองสามารถเรียนรู้ได้ดี และเป็นข้อร้องเรียนที่เกิน 50 แถว เป็นข้อร้องเรียนที่ประชาชนมีการส่งคำร้องเรียนเข้ามาเป็นจำนวนมาก

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ลดการส่งคำร้องไปผิดหน่วยงาน
2) ลดงานของพนักงานในแต่ละหน่วยงานที่ต้องทำการคัดแยกประเภทคำร้อง
3) ลดความผิดพลาดของพนักงานในการคัดแยกประเภทคำร้อง

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 การตัดคำ (Word Segmentation)

ภาษาไทยเป็นภาษาที่มีการเขียนติดกันไปทั้งประโยคโดยไม่มีช่องว่างเหมือนภาษาอังกฤษ ดังนั้นจึงต้องมีการตัดคำ (Word Segmentation) [1] เพื่อวัตถุประสงค์ในการใช้ประโยชน์ต่อในด้านต่าง ๆ ซึ่งการตัดคำแบบพจนานุกรมมีแนวคิดพื้นฐานคือการนำคำในพจนานุกรมไปเทียบกับข้อมูล หากพบรูปแบบตรงทำให้สามารถตัดได้ มีด้วยกัน 2 แบบคือ

1. ตัดคำแบบยาวมากที่สุด (Longest Matching) คือ การตัดคำให้ยาวที่สุด เช่น “ไปหามเหสี” ตัดออกมาเป็น “ไป|หาม|เห|สี” เป็นต้น

2. ตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) คือ การตัดคำที่น้อยที่สุดในประโยค เช่น “ไปหามเหสี” ตัดออกมาเป็น “ไป|หา|มเหสี” เป็นต้น

คำไทยส่วนใหญ่เป็นคำประสมที่เกิดจากคำมูลมาประกอบกันแล้วทำให้มีความหมาย ทำให้รูปแบบการตัดคำแบบสอดคล้องมากที่สุดนั้นเหมาะกับภาษาไทยมากกว่า และปัญหาที่เกิดจากการตัดคำคือ ไม่สามารถจัดการกับ คำที่ไม่มีในพจนานุกรม (Out of Vocabulary) ซึ่งปัญหายังคงเกิดขึ้นแม้เพิ่มคำในพจนานุกรมมากเพียงใดก็ตาม

2.2 ตัวแทนของคำ (Word Representation)

การแสดงตัวแทนของคำมีด้วยกัน 2 แบบได้แก่ [2]

1. ตัวแทนคำสัญลักษณ์ (Symbolic Representation) เช่น WordNet [3] เป็นลักษณะกราฟองค์ความรู้ (Knowledge Graph) หรือ One-hot Model [4] ข้อเสียคือ ไม่สามารถบอกคำเหมือนว่าใกล้หรือไกลกันได้

2. ตัวแทนคำการกระจาย (Distribution Representation) คือ ทำให้คำกลายเป็น เวกเตอร์ (Vector) โดยอาศัยดูบริบทคำรอบข้างได้ซึ่งมีด้วยกัน 2 แบบ

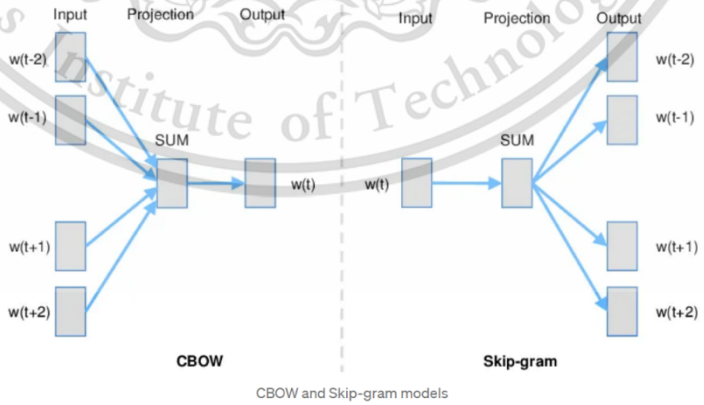
2.1 ตัวแทนคำเลขศูนย์ (Sparse Representation) คือ ตัวแทนของคำมีองค์ประกอบเป็นเลขศูนย์เป็นส่วนใหญ่ ส่วนน้อยที่ไม่เป็นเลขศูนย์ เช่น Term Frequency Inverse Document Frequency (TF-IDF) [4] เป็นต้น แต่มีข้อเสีย คือ เวกเตอร์ยาว และก่อให้เกิดเลขศูนย์จำนวนมาก

2.2 ตัวแทนคำของชั้นเชื่อมโยงแบบสมบูรณ์ คือ ตัวแทนของคำที่มีองค์ประกอบ ไม่เป็นเลขศูนย์เป็นส่วนใหญ่ เช่น Singular Value Decomposition (SVD) และ Word Embedding เป็นต้น ตัวแทนคำของชั้นเชื่อมโยงแบบสมบูรณ์ (Dense Representation) ที่น่าสนใจคือ การแปลงคำเป็นเวกเตอร์ (Word Embedding) คือการสร้างเวกเตอร์คุณลักษณะ จากคำซึ่งเป็นการแปลงชุดของคำให้เป็นตัวเลขที่ไม่ซ้ำกันโดยการคำนวนความน่าจะเป็นของคำ และบริบทภายในประโยค ซึ่งนำเสนอในรูปแบบของเวกเตอร์ เช่น Word2Vec [5], GLOVE [5] และ fastText [6]

แบบจำลอง Word2Vec สามารถตรวจจับคำที่มีความหมายเหมือนกัน และระบุระดับของความคล้ายคลึงกันทางความหมายระหว่างคำที่ถูกแทนด้วยเวกเตอร์ โดยใช้เทคนิคการวัดค่าความคล้ายเชิงมุม (Cosine Similarity) มีด้วยกัน 2 ลักษณะได้แก่

1) Continuous Bag of Words (CBOW) [7] คือ คาดการณ์คำหนึ่งคำ จากคำบริบทหลายคำรอบข้าง หรือการใช้คำหลาย ๆ คำที่อยู่ต่อเนื่องกันสำหรับการทำนายคำที่อยู่ถัดไป จากรูปที่ 2.1

2) Skip-gram [7] คือ มีการคาดเดาคำบริบทหลายคำ จากคำเดียว หรือการใช้คำหนึ่งคำในการทำนายคำอื่น ๆ ที่มีโอกาสเป็นคำถัดไปต่อจากคำปัจจุบัน แบบจำลอง CBOW และ Skip-gram แสดงดังรูปที่ 2.1



รูปที่ 2.1 แบบจำลอง CBOW และ Skip-gram [7]

เนื่องจาก Skip-gram มีการป้อนคำเพียงคำเดียว ทำให้พบคำที่พบบ่อย (Frequent Words) น้อยกว่า CBOW ที่มีการป้อนหลายคำซึ่ง CBOW มีการใช้คำที่พบบ่อยจำนวนมาก ซึ่งคำเหล่านี้มักปรากฏขึ้นบ่อยในหลายบริบท สิ่งนี้ทำให้ Skip-gram มีประสิทธิภาพที่ดีขึ้นในการจับความสัมพันธ์เชิงความหมายในเอกสารมากกว่า CBOW แต่มีข้อเสียในการฝึกสอน (Training) ที่มากกว่า

ตัวแทนของคำการฝึกล่วงหน้า (Pretrained Word2Vec) คือ แบบจำลองที่ได้รับการฝึกสอนพร้อมให้ใช้งานได้ โดยที่ไม่ต้องสร้างแบบจำลองขึ้นมาเองตั้งแต่ต้น แต่สามารถนำไปใช้งานหรือต่อยอดการใช้งานให้เหมาะสมกับงานได้ แบ่งเป็น 2 แบบ คือ

1) การแปลงคำเป็นเวกเตอร์แบบไม่สนใจบริบท (Non-contextualized Word Embeddings) คือ เวกเตอร์คงที่ (Fixed Vector) คำเดียวมีแค่เวกเตอร์เดียว เช่น Word2Vec, GLoVE และ fastText เป็นต้น

2) การแปลงคำเป็นเวกเตอร์แบบสนใจบริบท (Contextualized Word Embedding) คือ คำ ๆ เดียวมีได้หลายเวกเตอร์แม้อยู่คนละบริบท เช่น thai2fit (ULMFit) และ BERT เป็นต้น

2.3 ข้อจำกัดของการแปลงคำเป็นเวกเตอร์ (Word Embedding)

การแปลงคำย่อยเป็นเวกเตอร์ (Subword Embedding) เกิดมาจากแนวคิดการใช้ระดับคำ (Word) คือหยาบไป และใช้ระดับอักขระ (Character) ละเอียดไป จึงใช้เป็นระดับคำย่อยแทน ซึ่งมาช่วยแก้ปัญหาเรื่องคำที่เป็นคำเดียวกันแต่เขียนคนละแบบ ทำให้สามารถบอกได้ว่าเป็นคำเดียวกัน เช่น “ไม่มี” ภาษากลางของประเทศไทย กับ “บ่มี” ภาษาภาคตะวันออกเฉียงเหนือ (อีสาน) ของประเทศไทย ว่าเป็นคำเดียวกัน และช่วยแก้ปัญหาคำที่ไม่มีในพจนานุกรม แบ่งเป็น 3 แบบได้แก่

1) การเข้ารหัสจับคู่ (Byte-Pair Encoding: BPE) ยกตัวอย่างแบบจำลองที่ใช้ เช่น Transformer RoBERTa และ GPT-2

2) ชิ้นส่วนคำ (WordPiece) ยกตัวอย่างแบบจำลองที่ใช้ เช่น BERT และ DistilBERT

3) ชิ้นส่วนประโยค (SentencePiece) ยกตัวอย่างแบบจำลองที่ใช้ เช่น T5 และ WangchanBERTa

โดยใน ชิ้นส่วนประโยค ในการใช้งาน การแปลงคำเป็นเวกเตอร์ในแบบจำลอง WangchanBERTa โดยชิ้นส่วนประโยคแตกต่างกับชิ้นส่วนคำ และการเข้ารหัสจับคู่ (BPE) คือ มีการเพิ่มช่องว่าง “_”

2.4 การเรียนรู้เชิงลึก (Deep Learning)

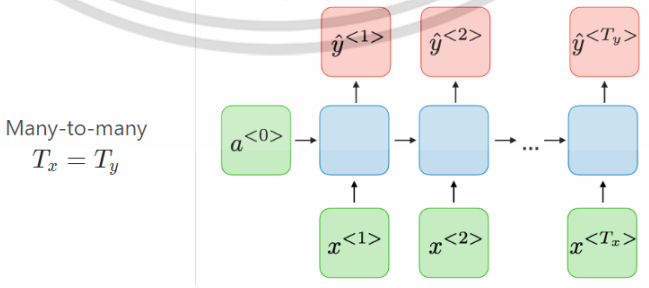
การเรียนรู้เชิงลึก (DL) เป็นส่วนหนึ่งของการเรียนรู้ของเครื่องจากโครงข่ายประสาทเทียมที่มีความคล้ายคลึงกับสมองมนุษย์ วิธีคิดหลักของการเรียนรู้เชิงลึก คือการแพร่กระจายไปข้างหน้า (Forward Propagation) เพื่อคำนวณค่าสูญเสียและการแพร่กระจายย้อนหลัง (Back Propagation) เพื่อปรับน้ำหนักเพื่อลดค่าสูญเสีย กระบวนการนี้เป็นส่วนสำคัญของการฝึกแบบจำลอง การเรียนรู้เชิงลึกเพื่อให้มีความแม่นยำในการทำนายข้อมูลที่ดีที่สุด การฝึกแบบจำลองดังกล่าวนี้ต้องผ่านการปรับน้ำหนักหลายครั้งเพื่อค้นหาค่าน้ำหนักที่ให้ความแม่นยำที่สูงที่สุดสำหรับโมเดลที่กำลังฝึกอยู่ การฝึกแบบจำลองการเรียนรู้เชิงลึก ดังกล่าวเป็นกระบวนการที่ซับซ้อน และใช้ในหลาย ๆ งาน เช่น การจำแนกประเภทภาพและการแปลภาษา และมีความสำคัญในการแก้ไขปัญหาที่ซับซ้อนในวงกว้างของอุตสาหกรรมและภาคเอกชน [8]

2.4.1 โครงข่ายระบบประสาทแบบย้อนกลับ (Recurrent Neural Network: RNN)

โครงข่ายระบบประสาทแบบย้อนกลับ (RNN) เป็นแบบจำลองสถาปัตยกรรมที่มีพื้นฐานเหมือนโครงข่ายระบบประสาท (Neural Network: NN) โดยมีผลลัพธ์ออกมาที่ไม่มีการย้อนกลับแต่โครงข่ายระบบประสาทแบบย้อนกลับ (RNN) นั้น ผลลัพธ์มีการย้อนกลับมาเป็นข้อมูลที่ป้อนขาเข้าไปใหม่ โดยใช้กับข้อมูลที่เป็นลำดับ (Sequence Data) เช่น หุ้น เสียง ภาพเคลื่อนไหว เป็นต้น แบบจำลองนี้มีปัญหาในเรื่องข้อมูลหายในประโยคที่มีความยาวมาก [9]

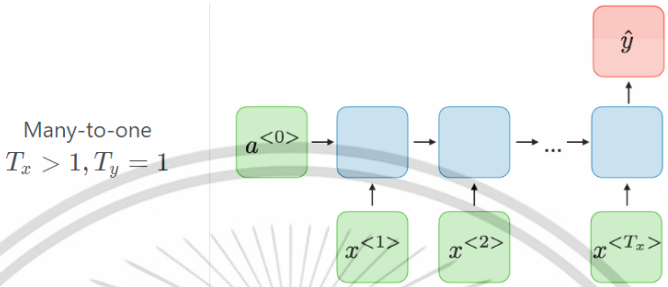
สถาปัตยกรรมโครงข่ายระบบประสาทแบบย้อนกลับ (RNN) แบ่งออกเป็น 4 แบบ ได้แก่

1) กลุ่มต่อกลุ่ม (Many to Many) ข้อมูลกลุ่มขาเข้าและขาออกเท่ากัน เหมาะกับงานการรับรู้ชื่อเฉพาะ (Name Entity Recognition) และติดป้ายไวยากรณ์ของคำศัพท์ (POS Tagging) แสดงดังรูปที่ 2.2



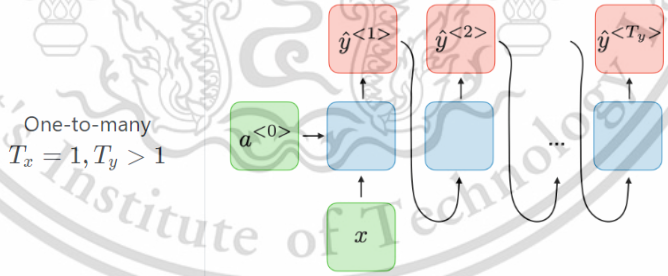
รูปที่ 2.2 กลุ่มต่อกลุ่ม (Many to Many) [10]

2) กลุ่มต่อหนึ่ง (Many to One) ข้อมูลขาออกมีหนึ่งแต่ข้อมูลกลุ่มขาเข้ามีมาก เหมาะกับงาน การวิเคราะห์ความรู้สึก (Sentiment Analysis) และการจัดแยกประเภทข้อความ (Text Classification) แสดงดังรูปที่ 2.3



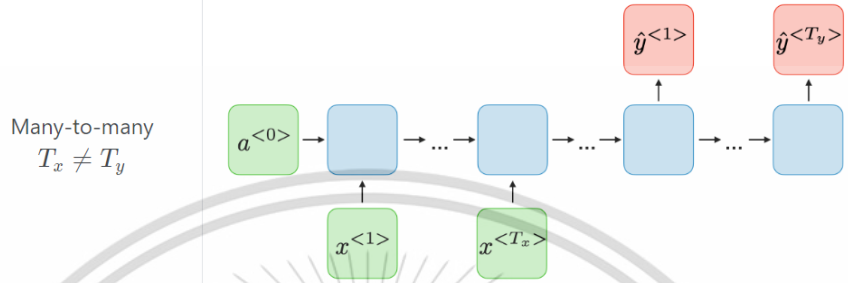
รูปที่ 2.3 กลุ่มต่อหนึ่ง (Many to One) [10]

3) หนึ่งต่อกลุ่ม (One to Many) ข้อมูลขาเข้ามีหนึ่งแต่ข้อมูลกลุ่มขาออกมีมาก เหมาะกับงาน การสร้างเนื้อร้องของเพลง (Music Generation) แสดงดังรูปที่ 2.4



รูปที่ 2.4 หนึ่งต่อกลุ่ม (One to Many) [10]

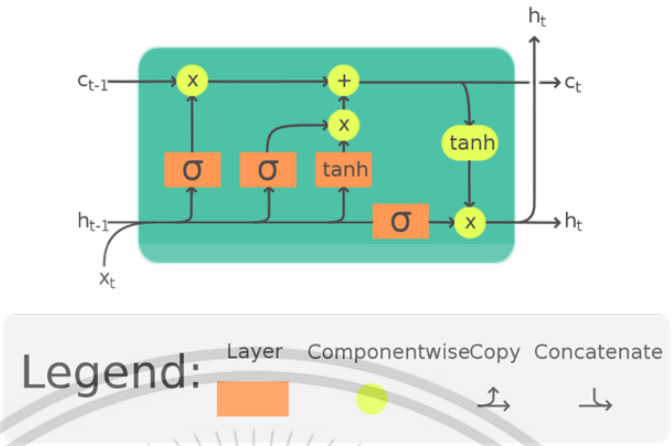
4) กลุ่มต่อกลุ่ม ตัวเข้ารหัส-ตัวถอดรหัส (Many to Many Encoder-Decoder) หรือ Sequence-to-Sequence Model (seq2seq) ข้อมูลกลุ่มขาเข้าและขาออกไม่เท่ากัน เหมาะกับงานเครื่องแปลภาษา (Machine Translation) [10, 11] แสดงดังรูปที่ 2.5



รูปที่ 2.5 กลุ่มต่อกลุ่ม ตัวเข้ารหัส-ตัวถอดรหัส (Many to Many Encoder-Decoder) [10]

2.4.2 หน่วยความจำระยะสั้นยาว (Long Short-Term Memory: LSTM)

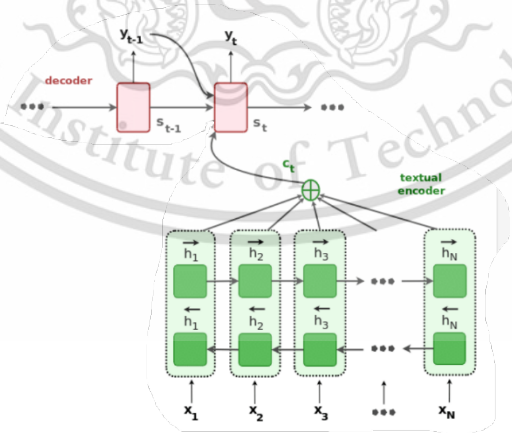
หน่วยความจำระยะสั้นยาว (LSTM) ใช้กับข้อมูลที่เป็นลำดับ (Sequence Data) และช่วยแก้ไขปัญหาข้อมูลที่อยู่ไกลกันที่ไม่มีความสัมพันธ์กัน หน่วยความจำระยะสั้นยาว (LSTM) มีส่วนประกอบหลัก 3 ประตู (Gate) ได้แก่ 1) ประตูขาเข้า (Input Gate) 2) ประตูขาออก (Output Gate) และ 3) ประตูหลงลืม (Forget Gate) ซึ่งเปรียบเสมือนวาล์วเปิดปิดท่อน้ำหรือการให้ข้อมูลผ่านหรือไม่ผ่าน และค่าที่มีการเข้าออกจากเซลล์ (Cell) ของหน่วยความจำระยะสั้นยาว (LSTM) คือ สภาวะแฝงเร้น (Hidden State) และสภาวะเซลล์ (Cell State) สุดท้ายยังช่วยแก้ไขปัญหา เกรเดียนต์ที่หายไป (Vanishing Gradient) คือการที่ค่าเกรเดียนต์ (Gradient) เล็กลงจนค่าน้ำหนัก (Weight) เข้าใกล้ศูนย์ เพราะเกิดจากการคูณของค่าน้ำหนักที่มีค่าน้อยไปจำนวนมากจนเข้าใกล้ศูนย์ เมื่อมีการแก้ปัญหาเรื่องข้อมูลหายด้วยการมีประตูเปิดปิดข้อมูลแล้ว แต่ประโยคมีความยาวมาก ข้อมูลยังคงหายไปเช่นกัน [12] หน่วยความจำระยะสั้นยาว แสดงดังรูปที่ 2.6



รูปที่ 2.6 หน่วยความจำระยะสั้นยาว (Long Short-Term Memory – LSTM) [12]

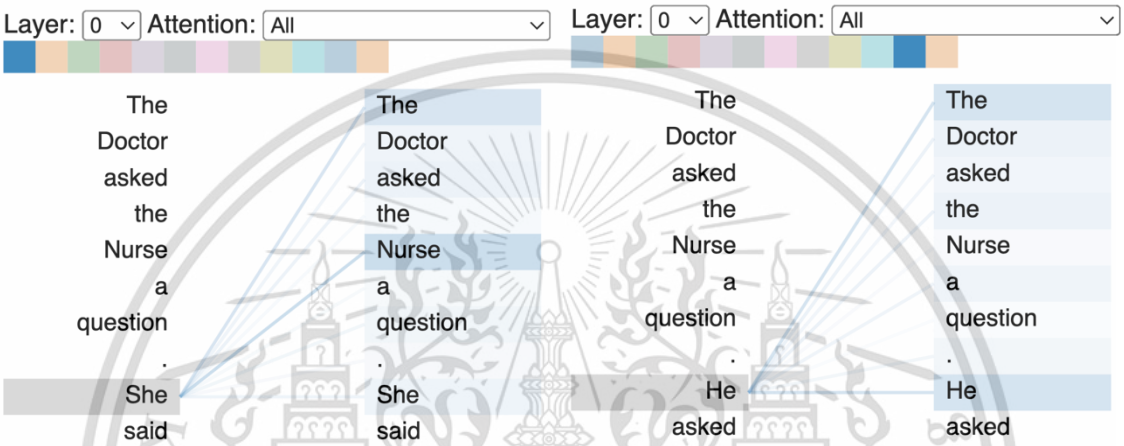
2.5 แอตเทนชัน (Attention)

จากรูปที่ 2.5 แสดงให้เห็นว่าสามารถส่งต่อข้อมูลจากฝั่ง ตัวเข้ารหัส (Encoder) ไปยังตัวถอดรหัส (Decoder) ได้ครั้งเดียวซึ่งเกิดจากการส่งข้อมูลไม่ครบถ้วน ทำให้เกิดเวกเตอร์บริบท (Context Vector) ซึ่งเกิดมาจาก ค่าเฉลี่ยถ่วงน้ำหนัก (Weighted Average) ของข้อมูลฝั่งตัวเข้ารหัส (Encoder) ทำให้เกิดการใช้ข้อมูลของทั้งฝั่งตัวเข้ารหัส และฝั่งตัวถอดรหัสเป็นการเพิ่มคุณภาพให้กับข้อมูล จึงเรียกกระบวนการนี้ว่า แอตเทนชัน (Attention) แสดงดังรูปที่ 2.7 [13]



รูปที่ 2.7 แอตเทนชัน (Attention) [14]

เซลฟ์แอตเทนชัน (Self-Attention) มีความสามารถโดดเด่นในการจับความสัมพันธ์ส่วนต่าง ๆ ของลำดับคำ ซึ่งทำให้เข้าใจไวยากรณ์ระหว่างคำได้ดีขึ้น สามารถใช้งานได้หลายครั้งอย่างอิสระ โดยการเลือกคำที่สนใจข้อมูลที่สำคัญ เช่น คำที่สำคัญในประโยค แล้วเพิ่มค่าเวทคำนั้นให้มากขึ้น ทำให้ประสิทธิภาพของแบบจำลองดีขึ้นอย่างมาก แสดงดังรูปที่ 2.8 ซึ่งแตกต่างจากแอตเทนชันที่มีการใช้งานครั้งเดียวและเป็นการถ่ายโอนข้อมูลจากตัวเข้ารหัสไปสู่ตัวถอดรหัส



รูปที่ 2.8 เซลฟ์แอตเทนชัน (Self-Attention) [15]

จากรูปที่ 2.8 สังเกตเห็นถึงความสัมพันธ์ของคำว่า “She” หรือ “เธอ” กับ “Nurse” หรือ “นางพยาบาล” ว่ามีความสัมพันธ์กันมากกว่า ความสัมพันธ์ระหว่างคำว่า “He” หรือ “เขา” กับ “Doctor” หรือ “หมอ” เพราะหมอไม่จำเป็นต้องเป็นผู้ชายแต่สามารถเป็นผู้หญิงได้ ทั้งสามคำคือ “เขา” กับ “หมอ” และ “เธอ” กับ “หมอ” จึงมีความสัมพันธ์ใกล้เคียงกันโดยสังเกตจากการเพิ่มค่าเวทจากพื้นหลังคำสีฟ้าอ่อนที่มีความเข้มน้อยกว่าสีฟ้าเข้ม ส่วนความความสัมพันธ์ของ “เธอ” กับ “นางพยาบาล” มีความสัมพันธ์ในระดับคำที่มากกว่า “เธอ” กับ “หมอ” โดยความสนใจตัวเองคือในประโยคนั้นจะจับความสัมพันธ์ของทุกคำรวมทั้งคำตัวเองเช่นคำว่า “She” หรือ “เธอ” จะมีการจับความสัมพันธ์คำว่า “She” หรือ “เธอ” ด้วย และรวมถึงคำอื่น ๆ ด้วยในประโยค [15]

2.6 การเรียนรู้แบบถ่ายโอน (Transfer Learning: TL)

เป็นเทคนิคของการเรียนรู้ของเครื่อง (Machine Learning: ML) ที่แบบจำลองที่ได้รับการฝึกล่วงหน้าในงานเดียว หรือแบบจำลองต้นน้ำ (Base Model) ได้รับการปรับแต่ง (Fine-Tuning) อย่างละเอียดสำหรับงานใหม่ที่เกี่ยวข้อง หรือเรียกว่า งานปลายน้ำ (Down-Stream Task) สรุปคือการถ่ายโอนการเรียนรู้ เป็นกระบวนการนำแบบจำลองที่ได้รับการฝึกมาแล้ว มาใช้กับชุดข้อมูลหรืองานอื่น ซึ่งนอกจะนำสถาปัตยกรรมหรือไปป์ไลน์มาใช้ใหม่แล้ว ยังนำความรู้ที่แบบจำลองมีอยู่มาใช้ในงานใหม่ด้วย

ในทางทฤษฎี การถ่ายโอนการเรียนรู้สามารถนำไปใช้กับงานอื่น ๆ ได้ แต่โดยทั่วไปใช้เพื่อปรับปรุงประสิทธิภาพของชุดข้อมูลที่มีขนาดเล็ก โดยถ่ายโอนความรู้ที่ฝึกสอนมาจากชุดข้อมูลขนาดใหญ่ [16]

2.7 การสุ่มตัวอย่าง (Sampling)

การสุ่มตัวอย่าง (Sampling) คือการสุ่มตัวอย่างเพื่อแยกเป็นชุดสำหรับการฝึก (Trian) การตรวจสอบ (Validation) และการทดสอบ (Test) เพราะไม่สามารถเข้าถึงข้อมูลได้ทั้งหมดในโลกถึงมีข้อมูลเหล่านั้นแต่ไม่สามารถประมวลผลข้อมูลทั้งหมดได้ เพราะใช้ทรัพยากรมากและราคาแพง และจุดประสงค์เพื่อดูว่าสิ่งที่ทำต่อนั้นมีความเป็นไปได้ที่จะลงทุนเวลาและแรง จึงต้องใช้การสุ่มตัวอย่างด้วยชุดข้อมูลเล็ก ๆ หลังจากนั้นจึงนำข้อมูลชุดใหญ่มาใช้งานต่อได้ถ้าผลลัพธ์ออกมาดี

มีการแบ่งการสุ่มตัวอย่างออกเป็น 2 แบบ คือ การสุ่มตัวอย่างโดยไม่ใช้ความน่าจะเป็น (Nonprobability Sampling) และการสุ่มตัวอย่างด้วยวิธีแรนดอม (Random Sampling)

2.7.1 การสุ่มตัวอย่างโดยไม่ใช้ความน่าจะเป็น (Nonprobability Sampling)

1) การสุ่มตัวอย่างตามสะดวก ข้อมูลถูกเลือกตามความสะดวกในการพร้อมใช้งาน เป็นวิธีที่นิยมเพราะสะดวก

2) การสุ่มตัวอย่างแบบเพิ่มทวีคูณ เริ่มจากการเลือกตัวอย่างที่มีอยู่ในขณะนั้นแล้วเพิ่มทวีคูณขึ้นไปเรื่อย ๆ จนมีขนาดใหญ่

3) การสุ่มตัวอย่างจากความเห็นผู้เชี่ยวชาญ ให้ผู้เชี่ยวชาญ ตัดสินใจว่าเลือกตัวอย่างใดโดยตรง

4) การสุ่มตัวอย่างแบบโควตา เป็นการแบ่งตามโควตาโดยไม่มีการสุ่ม

2.7.2 การสุ่มตัวอย่างด้วยวิธีแรนดอม (Random Sampling)

1) การสุ่มแบ่งชั้น (Stratified Sampling)

การสุ่มแบ่งชั้น คือแบ่งให้มีทุกหมวดหมู่ครบในแต่ละชุดข้อมูลแต่ลดจำนวนข้อมูลลงตามเปอร์เซ็นต์ที่มีการแบ่งไป เช่น มีชุดข้อมูลทั้งหมด 100 แถว คลาส A มี 90 แถว และคลาส B มี 10 แถว แบ่งชุดข้อมูล การฝึก และการทดสอบเป็น 90 เปอร์เซ็นต์ ต่อ 10 เปอร์เซ็นต์ สามารถแบ่งข้อมูลจากชุดการฝึกทั้งหมด 90 แถว โดยแบ่งคลาส A มี 81 แถว คลาส B มี 9 แถว และข้อมูลการทดสอบ ทั้งหมด 10 แถว โดยแบ่งคลาส A มี 9 แถว คลาส B มี 1 แถว ซึ่งทำให้ชุดข้อมูลทุกชุดนั้นสามารถเป็นตัวแทนประชากรของข้อมูลทั้งหมดได้ เพราะถ้าวิธีสุ่มแบบอื่นอาจทำให้ข้อมูลน้อยไม่ได้ถูกสุ่มมาในชุดข้อมูลทดสอบ หรือชุดข้อมูลการตรวจสอบได้ ซึ่งส่งผลเสียต่อการทดสอบ และตรวจสอบข้อมูลได้

2) การสุ่มแบบถ่วงน้ำหนัก (Weighted Sampling)

การสุ่มแบบถ่วงน้ำหนัก คือการใช้ประโยชน์จากผู้เชี่ยวชาญโดยเฉพาะเมื่อรู้ว่าข้อมูลไหนสำคัญกับแบบจำลอง แล้วต้องการให้มีโอกาสถูกเลือกมากขึ้นซึ่งทำให้น้ำหนักกับข้อมูลใหม่นี้มากขึ้น ยกตัวอย่างเช่น สุนัข 25 เปอร์เซ็นต์ และแมว 75 เปอร์เซ็นต์ ในโลกของความเป็นจริงสมมุติว่ามีคนชอบแมวและสุนัขพอ ๆ กัน จึงมีการปรับแก้โดยเพิ่มน้ำหนักตัวอย่างของสุนัขมากกว่าแมว 3 เท่า จึงทำให้ สุนัขและแมวมีความน่าจะเป็นเท่ากัน เช่น มีสัตว์ ทั้งหมด 4 ตัว คือ สุนัข 1 ตัว และแมว 3 ตัว พอทำการปรับน้ำหนักมี สุนัข 3 ตัว เท่ากับแมว 3 ตัว เป็นต้น

3) การสุ่มจากที่เก็บ (Reservoir Sampling)

การสุ่มจากที่เก็บ (Reservoir Sampling) เป็นอัลกอริทึมเพื่อจัดการข้อมูลสตรีม เช่น มีทวีตส่งเข้ามาเรื่อย ๆ แล้วต้องการสุ่มจำนวน K ตัวอย่างของทวีตเพื่อวิเคราะห์หรือฝึกแบบจำลอง แต่ว่าไม่ทราบจำนวนทวีตที่กำลังเข้ามา เพราะมีการไหลเข้ามาเรื่อย ๆ และไม่สามารถใส่ทั้งหมดลงในหน่วยความจำได้ หมายความว่าไม่สามารถรู้ล่วงหน้าว่าค่าความน่าจะเป็นสำหรับแต่ละทวีตมีค่าเท่าไร เพื่อแก้ปัญหานี้ต้องแน่ใจว่า

ก. ทุกทวีตมีความน่าจะเป็นที่ถูกเลือกเท่ากัน

ข. หยุดอัลกอริทึมได้ตลอดเวลา และทวีตกลุ่มนั้นถูกสุ่มตัวอย่างด้วยความน่าจะเป็นที่ถูกต้อง

ทางออกสำหรับเงื่อนไขข้างต้นคือ ใช้วิธีการสุ่มจากที่เก็บจำนวน k ตัวอย่างโดยอัลกอริทึมนี้กำหนดให้เป็นอาร์เรย์ และมี 3 ขั้นตอน ดังนี้

1. กำหนดที่เก็บให้เท่ากับ k ตัวอย่าง

2. ทุก ๆ ตัวอย่าง n ที่ไหลเข้ามา กำหนดให้สุ่มตัวอย่างลำดับที่ I โดย 1 <= I <= n

3. ถ้าตัวอย่างที่สุ่ม i มีค่าระหว่าง 1 <= I <= k: ให้แทนค่า I ด้วย n แต่ถ้าไม่ใช่ไม่ต้องทำอะไร

4) การสุ่มตามความสำคัญ (Importance Sampling)

การสุ่มตามความสำคัญ (Importance Sampling) เป็นการสุ่มตัวอย่างจากชุดข้อมูล โดยอ้างอิงการแจกแจงข้อมูลของอีกชุดหนึ่ง ลองนึกภาพว่าต้องสุ่มตัวอย่าง x จากการแจกแจงแบบ P(x) แต่ P(x) มีราคาแพงมาก ช้า หรือเป็นไปไม่ได้เลยที่สุ่มตัวอย่าง อย่างไรก็ตามมีการแจกแจงแบบ Q(x) ที่สุ่มตัวอย่างได้ง่ายกว่ามาก ดังนั้นสุ่มตัวอย่าง x จาก Q(x) แทน และชั่งน้ำหนักตัวอย่างนี้ด้วย P(x)/Q(x) ค่า Q(x) เรียกว่า การแจกแจงพร็อพโพซอล (Proposal Distribution) หรือ การแจกแจงแบบอิมพอร์เทนซ์ (Importance Distribution) โดย Q(x) เป็นการแจกแจงแบบใดก็ได้ ตราบใดที่ Q(x) > 0 โดยที่ P(x) ไม่เท่ากับ 0

การสุ่มตามความสำคัญมักถูกนำมาใช้ในแบบจำลองการเรียนรู้เสริมกำลังเชิงนโยบาย (Policy-Based Reinforcement Learning) ซึ่งเป็นการเรียนรู้ด้วยการลองผิดลองถูก และปรับปรุงนโยบายการกระทำตามผลลัพธ์ที่ได้ ในการอัปเดตนโยบายใหม่นั้น มักต้องประมาณค่าฟังก์ชันของนโยบายใหม่ ซึ่งการคำนวณค่าผลตอบแทนรวม (Total Reward) จากการกระทำอาจมีค่าใช้จ่ายสูง เนื่องจากต้องพิจารณาผลลัพธ์ที่เป็นไปได้ทั้งหมด อย่างไรก็ตาม หากนโยบายใหม่มีค่าใกล้เคียงกับนโยบายเดิม เราสามารถให้น้ำหนักใหม่แก่นโยบายใหม่โดยใช้ค่าผลตอบแทนจากนโยบายเดิมเป็นตัวกำหนดการแจกแจงพร็อพโพซอล (Proposal Distribution) ซึ่งวิธีการนี้ช่วยลดค่าใช้จ่ายและเพิ่มประสิทธิภาพในการอัปเดตนโยบาย [17] โดยในการทดลองนี้ใช้วิธี การสุ่มแบ่งชั้น

2.8 การเสริมข้อมูล (Data Augmentation)

การเสริมข้อมูล (Data Augmentation) ด้วยวิธี การรบกวน (Perturbation) คือ การเพิ่มข้อมูลโดยคงป้ายกำกับเดิมไว้ แต่เพิ่มสิ่งรบกวนลงไป โดยเรียกข้อมูลล่อลวงให้แบบจำลอง ทำนายผิดพลาดว่าข้อมูลปฏิปักษ์ (Adversarial Attack) เช่น ตัวอย่างแรก การเพิ่มสิ่งรบกวนลงไปในภาพ แบบจำลองสามารถจำแนกผิดได้ เมื่อเกิดการเปลี่ยนแปลงเพียงพิกเซลเดียว ตัวอย่างสอง แบบจำลองภาษา BERT มีการสุ่มเลือก 15 เปอร์เซ็นต์ ของเนื้อหาทั้งหมด แล้ว 10 เปอร์เซ็นต์ แทนที่ด้วยคำอื่นแบบสุ่ม เช่น This is a cat. ในตอนแรกของเนื้อหาเปลี่ยนเป็น This is a book. แทน เป็นต้น อยู่ในเนื้อหาแบบจำลองภาษาที่แมสก์แล้ว (Masked Language Model: MLM) [17]

2.9 ความไม่สมดุลของคลาส (Class Imbalance)

ความไม่สมดุลของคลาส (Class Imbalance) คือจำนวนของคลาสแต่ละคลาสแตกต่างกันมาก บางคลาสมีจำนวนมาก ส่วนบางคลาสมีจำนวนน้อยมากแต่ควรมีข้อมูลเกิน 100 แถว ในแต่ละคลาส เพื่อให้แบบจำลองสามารถเรียนรู้ได้

2.9.1 การใช้เมตริกการประเมินที่ถูกต้อง

เมื่อพบงานที่มีความไม่สมดุลของคลาส สิ่งสำคัญคือต้องเลือกเมตริกการประเมิน (Evaluation Metric) ที่เหมาะสม เมตริกที่ไม่ถูกต้องทำให้วินิจฉัยผิด ทำให้พัฒนาแบบจำลองหรือเลือกแบบจำลองที่ดีไม่ได้

การประเมินประสิทธิภาพของแบบจำลองการจำแนกประเภทเป็นสิ่งสำคัญในการพัฒนาระบบปัญญาประดิษฐ์ เมทริกซ์ความสับสน (Confusion Matrix) เป็นเครื่องมือหนึ่งที่ใช้ในการประเมินผลการทำนาย โดยแสดงจำนวนของกรณีที่ถูกจำแนกประเภทอย่างถูกต้องและไม่ถูกต้องสำหรับแต่ละประเภท จากเมทริกซ์ความสับสนนี้ เราสามารถคำนวณหามาตรวัดประสิทธิภาพอื่น ๆ ได้ดังนี้

ความแม่นยำ (Accuracy) ค่าพรีซิชัน (Precision) ค่ารีคอลย์ (Recall) ค่าเอฟวัน (F1) และค่าเฉลี่ยมาโคร (Macro-Average)

การใช้ความแม่นยำ (Accuracy) เป็นตัวชี้วัดประสิทธิภาพของแบบจำลองได้ไม่ดีในงานที่มีความไม่สมดุลของคลาส เพราะประสิทธิภาพของความแม่นยำ ปฏิบัติต่อทุกคลาสเท่าเทียมกัน ค่าเมตริกที่ได้ถูกครอบงำจากประสิทธิภาพแบบจำลองที่ได้จากคลาสกลุ่มใหญ่ สิ่งนี้ยิ่งไม่ดีหากคลาสกลุ่มใหญ่ไม่ใช่สิ่งที่สนใจ

จากตารางที่ 2.1 เมทริกซ์ความสับสน คือตารางที่ใช้ประเมินประสิทธิภาพของการจัดแยกประเภท (Classification) รูปแบบต่าง ๆ ร่วมกับชุดข้อมูลทดสอบที่ทราบค่าหรือผลลัพธ์ที่แท้จริงแล้ว ด้วยการประเมินค่าจริง (Actual Value) เทียบกับค่าที่ได้จากผลลัพธ์ของแบบจำลอง หรือค่าที่คาดการณ์ไว้ (Predicted Value) แล้วสรุปค่าออกมาในรูปแบบของตารางเมทริกซ์ที่ประกอบด้วยค่า True Negative (TN), False Positive (FP), False Negative (FN) และ True Positive (TP) ซึ่งแต่ละค่ามีความหมายดังนี้ [16]

ตารางที่ 2.1 เมทริกซ์ความสับสน (Confusion Matrix)

Confusion Matrix	Actual Positive	Actual Negative
Predicted Positive	True Positive (TP)	False Positive (FP) Type I Error
Predicted Negative	False Negative (FN) Type II Error	True Negative (TN)

1) True Positive (TP) ข้อมูลจริงเป็นบวก และผลการทำนายเป็นบวก ถือว่าเป็นการทำนายที่ถูกต้อง เป็นกลุ่มข้อมูลที่สนใจ

2) True Negative (TN) ข้อมูลจริงเป็นลบ และผลการทำนายเป็นลบ ถือว่าเป็นการทำนายที่ถูกต้อง เป็นกลุ่มที่ไม่สนใจ

3) False Positive (FP) ข้อมูลจริงเป็นลบ และผลการทำนายเป็นบวก ถือว่าเป็นการทำนายที่ผิด เป็นกลุ่มข้อมูลที่ไม่สนใจ และเป็นประเภท Type I Error

4) False Negative (FN) ข้อมูลจริงเป็นบวก และผลการทำนายเป็นลบ ถือว่าเป็นการทำนายที่ผิด เป็นกลุ่มข้อมูลที่สนใจ และเป็นประเภท Type II Error

Type I Error คือ เมื่อแบบจำลองทำนายว่าลูกค้านี้เป็นกลุ่มเป้าหมายแต่ในความเป็นจริงไม่ใช่กลุ่มเป้าหมายทำให้เสียเวลาและทรัพยากรกับกลุ่มนี้ ทำให้อันตรายได้เมื่อเป็นเรื่องของกฎหมาย เช่น ผู้ต้องสงสัยคนนี้อาจไม่ได้ทำผิดแต่แบบจำลองบอกว่าเขาทำผิดทำให้ได้รับโทษค่าปรับหรือจำคุก

Type II Error คือ เมื่อแบบจำลองทำนายว่าลูกค้ากลุ่มนี้ไม่ใช่กลุ่มเป้าหมายแต่ในความเป็นจริงเป็นกลุ่มเป้าหมายจริง ๆ ทำให้เสียโอกาสในการทำการตลาด โปรโมชันเพื่อเพิ่มยอดขาย ทำให้อันตรายเมื่อเกี่ยวกับการทำนายว่าเป็นโรคร้ายแรง อาจทำให้เสียชีวิตโดยไม่ได้รับการรักษาได้

ความแม่นยำ (Accuracy) คือการแสดงประสิทธิภาพของแบบจำลอง โดยสนใจเฉพาะค่าที่ทำนายถูก เมื่อเทียบกับทุกคลาส ซึ่งใช้ได้ดีกับกรณีข้อมูลมีความสมดุล ไม่เบ้ไปทางใดทางหนึ่ง ค่าความแม่นยำ แสดงได้ดังสมการที่ (2.1)

$$\text{Accuracy} = \frac{\mathbf{TP + TN}}{\mathbf{(TP + TN+FP+FN)}} \tag{2.1}$$

ค่าพรีซิชัน (Precision) คือแสดงถึงประสิทธิภาพแบบจำลอง เมื่อสนใจค่า True Positive (TP) กับ False Positive (FP) เป็นพิเศษ จากการพยากรณ์ว่าเป็นบวกทั้งหมดมีจำนวนพยากรณ์ถูกเท่าไร หรือมองในมุมที่สนใจ Type I Error คืออยากได้ค่า ค่าพรีซิชันสูง เพื่อลดปัญหาใน Type I Error ค่าพรีซิชัน แสดงได้ดังสมการที่ (2.2)

$$\text{Precision} = \frac{\mathbf{TP}}{\mathbf{TP+FP}} \tag{2.2}$$

ค่ารีคอลย์ (Recall) คือแสดงถึงประสิทธิภาพแบบจำลอง เมื่อสนใจค่า True Positive (TP) กับ False Negative (FN) เป็นพิเศษ จากข้อมูลจริงที่เป็นบวกทั้งหมด มีจำนวนพยากรณ์ถูกเท่าไร หรือมองในมุมที่สนใจ Type II Error คืออยากได้ค่า ค่ารีคอลย์สูง เพื่อลดปัญหาใน Type II Error ค่ารีคอลย์ แสดงได้ดังสมการที่ (2.3)

$$\text{Recall} = \frac{\mathbf{TP}}{\mathbf{TP+FN}} \tag{2.3}$$

ค่าเอฟวัน (F1) คือค่าเฉลี่ยฮาร์โมนิค (Harmonic Mean) ของค่าพรีซิชัน (Precision) และค่ารีคอลย์ (Recall) แสดงถึงประสิทธิภาพแบบจำลอง เมื่อสนใจค่า True Positive และสนใจ False Positive (FP) และ False Negative (FN) เป็นพิเศษ ค่าเอฟวัน แสดงได้ดังสมการที่ (2.4)

ค่าเอฟวันสูงแสดงว่า ค่าพรีซิชัน และค่ารีคอลย์นั้นสูง แต่ถ้าค่าเอฟวันน้อยแสดงว่า ค่าพรีซิชัน และค่ารีคอลย์นั้นน้อยตามไปด้วย [17]

$$\text{F1} = \frac{\mathbf{2* Precision * Recall}}{\mathbf{(Precision+ Recall)}} \tag{2.4}$$

ค่าเฉลี่ยมาโคร (Macro-Average) คือการหาค่าเฉลี่ยเลขคณิตของ ค่าเอฟวัน (F1) แต่ละคลาส เช่น มีทั้งหมด 3 คลาส แต่ละคลาสมีค่าเอฟวัน (F1) เป็น 0.7 0.8 และ 0.9 เมื่อนำมาคำนวณหาค่าเฉลี่ยมาโคร คือ 0.7 + 0.8 + 0.9 / 3 = 0.8 เป็นต้น ค่าเฉลี่ยมาโคร แสดงได้ดังสมการที่ (2.5)

$$\text{Macro-Average} = \frac{\Sigma(\mathbf{F1})}{\mathbf{N}} \tag{2.5}$$

$\Sigma(\mathbf{F1})$ คือ จำนวนผลรวมค่าเอฟวัน (F1) แต่ละคลาสทั้งหมด
$\mathbf{N}$ คือ จำนวนคลาสทั้งหมด

โดยในการทดลองนี้ใช้ ค่าความแม่นยำ และค่าเฉลี่ยมาโครเป็นตัววัดผลประสิทธิภาพแบบจำลอง

2.9.2 วิธีแก้ไขความไม่สมดุลของคลาส

1. การสุ่มตัวอย่างใหม่ (Resampling)

การสุ่มตัวอย่างใหม่ประกอบไปด้วยการสุ่มมากขึ้น (Oversampling) โดยสุ่มตัวอย่างจากคลาสกลุ่มน้อย การทำสำเนาคลาสกลุ่มน้อยให้เพิ่มขึ้น และการสุ่มน้อยลง (Undersampling) คือการสุ่มเอาตัวอย่างในคลาสกลุ่มใหญ่ออกไปจนมีจำนวนเท่ากับคลาสกลุ่มน้อย [16]

2. น้ำหนักตัวอย่าง (Sample Weights)

น้ำหนักตัวอย่าง มีวิธีคิดคล้ายกับการสุ่มแบบถ่วงน้ำหนัก (Weighted Sampling) คือการกำหนดหรือให้ความสำคัญกับตัวอย่างเพื่อฝึกแบบจำลอง โดยตัวอย่างที่มีน้ำหนักมากส่งผลต่อลอสฟังก์ชัน (Loss Function) มาก การเปลี่ยนน้ำหนักตัวอย่างทำให้ขอบเขตการตัดสินใจของแบบจำลองมีการเปลี่ยนแปลง หลักการทำงานคือให้ความสำคัญของแต่ละคลาสอย่างเท่าเทียมกัน [17]

3. SMOTE

SMOTE หรือชื่อย่อมาจาก Synthetic Minority OverSampling Technique การทำ การสุ่มมากขึ้น (Oversampling) เป็นเทคนิคการสุ่มคลาสกลุ่มน้อยแบบสังเคราะห์ วิธีนี้สังเคราะห์ตัวอย่างใหม่ของคลาสกลุ่มน้อย ผ่านการสุ่มตัวอย่างค่าผลรวมเชิงคอนเว็กซ์ (Convex Combination) ของจุดข้อมูลที่อยู่ภายในคลาสกลุ่มน้อย หลักการคือกำหนดจำนวนด้วยแบบจำลองเพื่อนบ้านที่ใกล้เคียง

ที่สุด (K-Nearest Neighbor: KNN) จำนวน K ตัว แล้วทำการสุ่มสร้างข้อมูลใหม่บนทางที่เชื่อมระหว่างจุดข้อมูลที่กำลังพิจารณา ซึ่ง SMOTE มีข้อจำกัดคือ 1) การสุ่มตัวอย่างที่ทับซ้อนกัน 2) การสุ่มตัวอย่างทำให้มีสิ่งรบกวน และเป็นเรื่องยากที่กำหนดจำนวนเพื่อนบ้านใกล้สุด จำนวน K ตัว และทำให้มองไม่เห็นเพื่อนบ้านที่ใกล้ที่สุดในการเลือกสำหรับสร้างตัวอย่างสังเคราะห์ [18]

4. ADASYN

ADASYN หรือชื่อย่อมาจาก Adaptive Synthetic Sampling Approach การสุ่มตัวอย่างสังเคราะห์ที่ปรับเปลี่ยนได้ เป็นการทำการสุ่มมากขึ้น (Oversampling) เป็นการสร้างตัวอย่างสังเคราะห์ ที่ไม่จำเป็นต้องพิจารณาข้อมูลทุกตัวที่อยู่ในกลุ่มน้อยใช้ค่าการแจกแจงแบบถ่วงน้ำหนักของข้อมูลตัวอย่างกลุ่มน้อย โดยการสังเคราะห์ข้อมูลซึ่งขึ้นอยู่กับความสำคัญของข้อมูลนั้น ๆ ถ้าข้อมูลใดยากต่อการจำแนกให้ค่าของน้ำหนักข้อมูลนั้นมาก และสังเคราะห์ข้อมูลขึ้นมาในบริเวณนั้น ๆ [18]

โดยในการทดลองนี้ใช้วิธี การสุ่มตัวอย่างใหม่ (Resampling) และน้ำหนักตัวอย่าง (Sample Weights) และใช้ข้อมูลมากกว่า 50 แถว

2.10 ออปติไมเซอร์ (Optimizer)

2.10.1 เกรเดียนต์เดสเซนต์ (Gradient Descent)

ความเร็วในการใช้เวลาฝึกสอนขึ้นอยู่กับ 2 ปัจจัย ได้แก่ สถาปัตยกรรม (Architecture) และจำนวนตัวอย่าง (Sample) ที่ใช้ในการฝึกสอนแต่ละครั้ง ซึ่งสนใจจำนวนตัวอย่างที่ใช้

1. แบตช์เกรเดียนต์เดสเซนต์ (Batch Gradient Descent)

ใช้ทุก ๆ ตัวอย่างในการฝึกสอนต่อ 1 อีพ็อค (Epoch) ข้อดีคือความนิ่งในการลู่เข้าสู่คำตอบได้ดี ส่วนข้อเสียคือความเร็วในการใช้เวลาฝึกสอนช้า

2. สโตแคสติคเกรเดียนต์เดสเซนต์ (Stochastic Gradient Descent)

ใช้ 1 ตัวอย่างในการฝึกสอนต่อ 1 อีพ็อค ข้อดีคือความเร็วในการใช้เวลาฝึกสอนเร็ว ส่วนข้อเสียคือความนิ่งในการลู่เข้าสู่คำตอบได้ไม่ดี

3. มินิแบตช์เกรเดียนต์เดสเซนต์ (Mini-Batch Gradient Descent)

ใช้ขนาดแบตช์ 16, 32, 64 หรือ 128 ตัวอย่าง ในการฝึกสอนต่อ 1 อีพ็อค ต้องการให้แต่ละตัวอย่าง ถูกนำมาใช้ฝึกสอนแบบจำลองในจำนวนครั้งที่เท่ากัน ข้อดีและข้อเสียคือ ความเร็วในการใช้เวลาฝึกสอนเร็วกว่าแบตช์ (Batch) แต่ช้ากว่าสโตแคสติค (Stochastic) ส่วนความนิ่งในการลู่เข้าสู่คำตอบได้ดีกว่าสโตแคสติค แต่ลู่เข้าได้ไม่ดีเท่าแบตช์ ยกตัวอย่างเช่น อยากอัปเดตค่าน้ำหนัก (Weight) 1,000 ครั้ง จำนวนตัวอย่างทั้งหมด 600 ตัวอย่าง และใช้มินิแบตช์ (Mini-Batch) เท่ากับ 64 ตัวอย่าง คือ 600/64 = 10 ครั้ง (เพราะมีการปัดเศษขึ้นจาก 9.375 เป็น 10 เพื่อให้ใช้ข้อมูลครบทั้งหมด) จึงต้องมีการฝึกสอนทั้งหมด 100 อีพ็อค เพื่อให้ครบในการอัปเดตค่าน้ำหนัก 1,000 ครั้ง เป็นต้น

2.10.2 Momentum

Momentum คือพจน์เกรเดียนต์ (Gradient) ถูกพิจารณาในรูปของการสะสมเกรเดียนต์ ของอีพ็อค ก่อนหน้ารวมกับเกรเดียนต์ของอีพ็อค ในปัจจุบัน

$$\theta_t = \theta_{t-1} - V_t \quad (2.6)$$

$$V_t = \gamma V_{t-1} + \propto g_t \quad (2.7)$$

$$g_t = \nabla Cost_{t-1} \quad (2.8)$$

รูปที่ 2.9 สูตร Momentum

จากสมการที่ (2.6), (2.7) และ (2.8) คือสูตรของ Momentum โดย ทีตา (θ) คือค่าน้ำหนัก V_t คือ ตัวเฉลี่ยเกรเดียนต์ มีการเฉลี่ยแบบ ค่าเฉลี่ยเคลื่อนที่มีการถ่วงน้ำหนัก (Exponential Moving Average) เวลา ณ จุดเริ่มต้น $V_0 = 0$ แกมมา (γ) คือค่าคงที่ใช้เพื่อถ่วงน้ำหนักของเกรเดียนต์ในรอบก่อนหน้า (โดยปกติมีค่าเท่ากับ 0.9)

โดยเมื่อดูจากสูตร V_t สังเกตว่าค่าความชันก่อนหน้า V_{t-1} มีการคูณกับแกมมาเพื่อลดความสำคัญลงไม่ให้มีความสำคัญเท่ากับค่าความชัน g_t ที่เป็นค่าปัจจุบัน

2.10.3 Adagrad

Adagrad ย่อมาจาก Adaptive Gradient Algorithm เป็นออปติไมเซอร์ที่มีการปรับอัตราการเรียนรู้ (Learning Rate) ให้อัตโนมัติ คือยิ่งผ่านประสบการณ์การสะสมเกรเดียนต์มากยิ่งให้ค่าอัตราการเรียนรู้ต่ำ หลักการคือปรับลดอัตราการเรียนรู้ตามเกรเดียนต์ที่มีการเคลื่อนที่ผ่านมาแล้ว

$$\theta_t = \theta_{t-1} - S_t \odot g_t \tag{2.9}$$

$$S_t = \frac{\propto}{\sqrt{G_t + \varepsilon}} \tag{2.10}$$

$$G_t = G_{t-1} + (g_t \odot g_t) \tag{2.11}$$

$$g_t = \nabla Cost_{t-1} \tag{2.8}$$

รูปที่ 2.10 สูตร Adagrad

จากสมการที่ (2.9), (2.10), (2.11) และ (2.8) คือสูตรของ Adagrad โดย ทีตา (θ) คือค่าน้ำหนักแอลฟา (α) คืออัตราการเรียนรู้เริ่มต้น (Initial Learning Rate) S_t คืออัตราการเรียนรู้แบบปรับได้ (Adaptive Learning Rate) G_t คือผลรวมความชันยกกำลังสองในแต่ละมิติ เวลา ณ จุดเริ่มต้น $G_0 = 0$ และเอปไซลอน (ε) เท่ากับ 10^{-8} (ป้องกันการหารด้วย 0) เมื่อสังเกต S_t พบว่ายิ่งเกรเดียนต์ หรือ G_t ยิ่งมากส่งผลให้อัตราการเรียนรู้แบบปรับได้ลดต่ำลง

ข้อเสียคือเกิดปัญหาเมื่อทำการใช้เวลาฝึกสอนแบบจำลองไปนาน ๆ อัตราการเรียนรู้ลู่เข้าสู่ 0 ถึงแม้ว่าแบบจำลองยังฝึกสอนไม่เสร็จ

2.10.4 RMSprop

RMSprop ย่อมาจาก Root Mean Square Propagation เป็นออปติไมเซอร์ที่มาช่วยแก้ปัญหาการหยุดนิ่งท้าย ๆ ของ Adagrad เมื่ออัตราการเรียนรู้ เท่ากับ 0

$$\theta_t = \theta_{t-1} - S_t \odot g_t \tag{2.9}$$

$$S_t = \frac{\propto}{\sqrt{G_t + \varepsilon}} \tag{2.10}$$

$$G_t = \beta G_{t-1} + (1-\beta)(g_t \odot g_t) \tag{2.12}$$

$$g_t = \nabla Cost_{t-1} \tag{2.8}$$

รูปที่ 2.11 สูตร RMSprop

จากสมการที่ (2.9), (2.10), (2.12) และ (2.8) คือสูตรของ RMSprop โดย ทีตา (θ) คือ ค่าน้ำหนัก แอลฟา (α) คืออัตราการเรียนรู้เริ่มต้น S_t คืออัตราการเรียนรู้แบบปรับได้ เอปไซลอน (ε) เท่ากับ 10^{-8} (ป้องกันการหารด้วย 0) และ G_t ค่าเฉลี่ยของเกรเดียนต์ แบบค่าเฉลี่ยเคลื่อนที่มีการถ่วงน้ำหนัก เวลา ณ จุดเริ่มต้น $G_0 = 0$

ในสูตรของ Adagrad ก่อนหน้า G_t คือผลรวมความชันยกกำลังสองในแต่ละมิติ เปลี่ยนเป็นมีการเพิ่มเติมการคำนวณค่าเฉลี่ยเคลื่อนที่มีการถ่วงน้ำหนัก กลายเป็นหาค่าเฉลี่ยของเกรเดียนต์แทนทำให้มีการปรับตัว ณ ตำแหน่งของ Cost Landscape ได้ดี (Cost Landscape คือแผนผังหรือภูมิทัศน์ที่เกี่ยวข้องกับค่าความคลาดเคลื่อนหรือค่าฟังก์ชันความสูญเสีย (Cost Function) ในพื้นที่ของพารามิเตอร์ของแบบจำลองที่กำหนด)

2.10.5 Adam

Adam ย่อมาจาก Adaptive Moment Estimation เป็นออปติไมเซอร์ที่มีการใช้แนวคิดจาก Momentum และ RMSprop มาใช้ร่วมกันโดย Momentum มีการสะสมความเร็วในการเคลื่อนที่ในแต่ละมิติ และ RMSprop มีการลดขนาดอัตราการเรียนรู้ในแต่ละมิติด้วยการสะสมเกรเดียนต์โดยใช้ค่าเฉลี่ยเคลื่อนที่มีการถ่วงน้ำหนัก

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{v_t+\varepsilon}} \odot m_t \quad (2.13)$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (2.14)$$

$$V_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t \odot g_t) \quad (2.15)$$

$$g_t = \nabla Cost_{t-1} \quad (2.8)$$

รูปที่ 2.12 สูตร Adam

จากสมการที่ (2.13), (2.14), (2.15) และ (2.8) คือสูตรของ Adam โดยทีตา (θ) คือค่าน้ำหนัก V_t คือ ส่วนของ RMSprop เวลา ณ จุดเริ่มต้น $V_0 = 0$ M_t คือส่วนของ Momentum เวลา ณ จุดเริ่มต้น $G_0 = 0$ และเอปไซลอน (ε) เท่ากับ 10^{-8} (ป้องกันการหารด้วย 0)

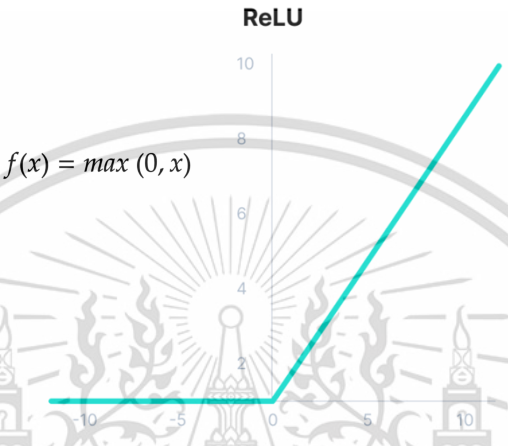
สังเกตจากสูตรว่าทั้งอัตราการเรียนรู้ ในส่วนของ RMSprop และเกรเดียนต์ ในส่วนของ Momentum มีการคำนวณแบบค่าเฉลี่ยเคลื่อนที่มีการถ่วงน้ำหนักทั้งคู่ [19] โดยในการทดลองนี้เลือกใช้ตัวอย่างแบบ Mini-Batch Gradient Descent และออปติไมเซอร์ คือ Adam

2.11 แอ็กทิเวชันฟังก์ชัน (Activation Function)

การรวมค่าน้ำหนักนั้นเป็นเพียงการเรียนรู้ความสัมพันธ์ในรูปแบบเชิงเส้น (Linear) แต่นำผลรวมน้ำหนักมาผ่านแอ็กทิเวชันฟังก์ชันเพื่อทำให้เป็นการเรียนรู้ความสัมพันธ์ในรูปแบบไม่เชิงเส้น (Non-Linear) ซึ่งทำให้สามารถจับความสัมพันธ์ข้อมูลไม่เป็นแบบเชิงเส้นได้ดีซึ่งข้อมูลส่วนใหญ่มีความสัมพันธ์แบบไม่เชิงเส้น แอ็กทิเวชันฟังก์ชันมีหลากหลายชนิด เช่น ซิกมอยด์ (Sigmoid - Logistic Activation Function), tanh (Hyperbolic Tangent Function), ReLU (Rectified Linear Unit) เป็นต้น

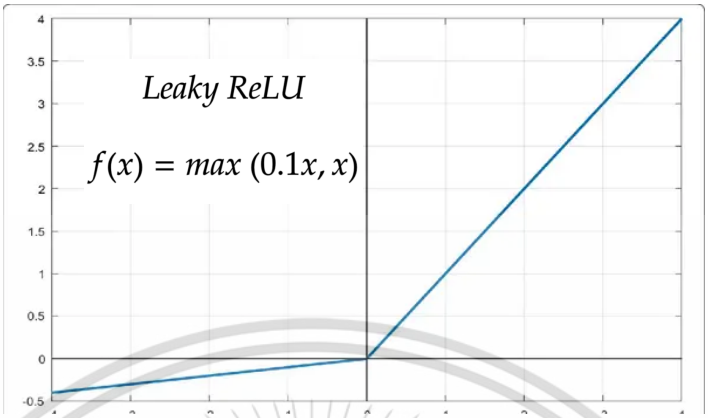
แอ็กทิเวชันฟังก์ชัน ReLU เป็นที่นิยมในการใช้งานเพราะไม่ซับซ้อนและเข้าใจง่าย จากรูปที่ 2.9 เป็นกราฟของแอ็กทิเวชันฟังก์ชัน ReLU โดยผลลัพธ์ผลรวมค่าน้ำหนักที่น้อยกว่า 0 เป็น 0 ทั้งหมด ส่วน

ค่าผลรวมค่าน้ำหนักที่มากกว่า 0 ได้ค่าผลลัพธ์ตัวมันเอง คือไม่เกิดการอิ่มตัวค่าบวกเหมือนซิกมอยด์ และคำนวณผลลัพธ์ได้อย่างรวดเร็วแต่มักเกิดปัญหา คือนิวรอนบางส่วนตาย (Dying ReLU) เพราะ ReLU ส่งสัญญาณ 0 เท่านั้น โดยพบว่านิวรอนบางส่วนตายเมื่อน้ำหนักบวกผลรวมไบแอสมีค่าเป็นลบ



รูปที่ 2.9 แอ็กทิเวชันฟังก์ชัน ReLU [20]

เพื่อแก้ปัญหานี้จึงเกิดเป็น ReLU เวอร์ชันที่พัฒนาขึ้นมา เช่น Leaky Rectified Linear Unit, Parametric Rectified Linear Unit, Exponential Linear Unit, Scaled Exponential Linear Unit, Gaussian Error Linear Unit, Swish, Mish ซึ่งขออธิบายแค่บางส่วน คือ Leaky Rectified Linear Unit (Leaky ReLU), Exponential Linear Unit (ELU), Gaussian Error Linear Unit (GELU)



รูปที่ 2.10 แอ็กทิเวชันฟังก์ชัน Leaky ReLU [21]

ReLU (Rectified Linear Unit) เป็นฟังก์ชันเรียกใช้งานพื้นฐานที่ใช้กันมาก แต่มีข้อบกพร่องคือ ถ้าค่าน้ำหนักรวมกับไบแอสติดลบ ผลลัพธ์จะเป็น 0 ซึ่งอาจทำให้การเรียนรู้ของนิวรอนนั้นหยุดชะงัก

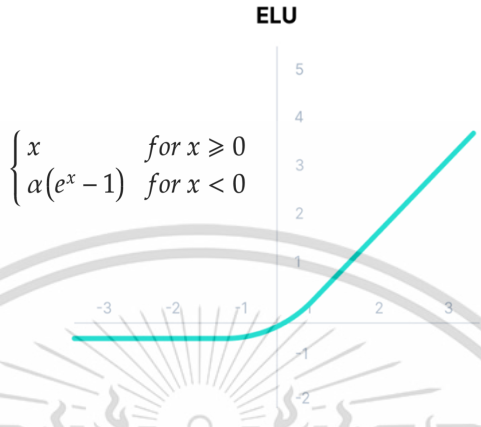
จากรูปที่ 2.10 เป็นกราฟของแอ็กทิเวชันฟังก์ชัน Leaky Rectified Linear Unit (Leaky ReLU) จึงถูกคิดค้นขึ้นมาในปี ค.ศ. 2015 โดย Bing Xu และคณะ เพื่อแก้ปัญหานี้ โดยให้มีการ "รั่วไหล" ในช่วงค่าลบ แทนที่จะให้เป็น 0 เลย โดยค่ารั่วไหลนี้ (ตัวเลข 0.1 ในสมการ) สามารถปรับค่าได้ เช่น 0.01 หมายถึงรั่วไหลน้อย หรือ 0.2 รั่วไหลมาก

รั่วไหลน้อย มีความลาดชันที่เล็กกว่า อนุญาตให้สัญญาณผ่านเข้ามาได้น้อยลงสำหรับค่าอินพุตเป็นลบ และฟังก์ชันจะทำงานคล้ายกับ ReLU มากขึ้น

รั่วไหลมาก มีความลาดชันที่ใหญ่กว่า อนุญาตให้สัญญาณผ่านเข้ามาได้มากขึ้นสำหรับค่าอินพุตเป็นลบ และอาจป้องกันไม่ให้นิวรอนตายสนิท

จากนั้นจึงมีการพัฒนาฟังก์ชันเรียกใช้งานอื่นๆ ที่ปรับปรุงข้อบกพร่องเหล่านี้ เช่น Exponential Linear Unit และ Scaled Exponential Linear Unit ซึ่งเป็นฟังก์ชันที่ต่อเนื่องและสามารถแก้ปัญหาการรั่วไหลได้ดีกว่า

สรุปคือ เป็นการแก้ไขข้อบกพร่องของฟังก์ชันก่อนหน้า เพื่อให้โครงข่ายประสาทเทียมสามารถเรียนรู้ได้อย่างมีประสิทธิภาพมากขึ้น

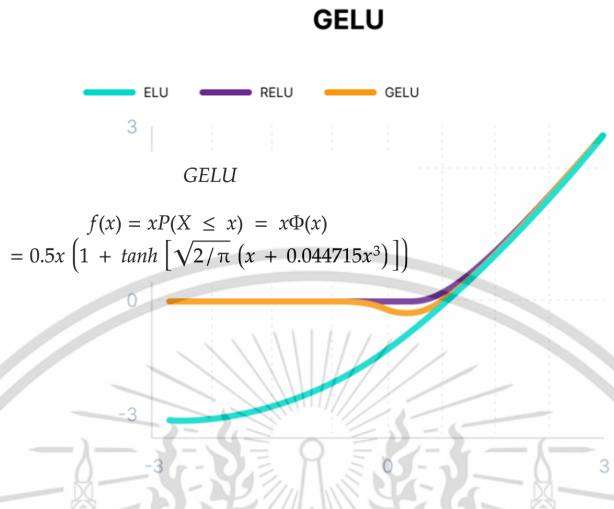


รูปที่ 2.11 แอ็กทิเวชันฟังก์ชัน ELU [20]

จากรูปที่ 2.11 เป็นกราฟของแอ็กทิเวชันฟังก์ชัน ELU ซึ่งถูกค้นพบโดย Djork-Arné Clevert และคณะในปี ค.ศ. 2015 หรืออีกชื่อคือ Exponential Linear Unit (ELU) ซึ่งข้อดีของ Leaky ReLU ตามมาคือเกรเดียนต์ไม่เป็น 0 เมื่อ $x < 0$ เพื่อหลีกเลี่ยงปัญหานิวรอนตาย

โดยปกติค่าแอลฟา (α) เท่ากับ 1 แต่สามารถปรับได้ เป็นฟังก์ชันที่ราบรื่นต่อเนื่องที่ทุกจุด รวมทั้งรอบจุด x=0 ช่วยให้การคำนวณอนุพันธ์ราบรื่นแม้จะคำนวณช้ากว่า ReLU เล็กน้อย แต่ช่วยให้การเรียนรู้คอนเวิร์จเร็วขึ้น ข้อดีของ ELU คือ แก้ปัญหานิวรอนตายของ ReLU ขณะเดียวกันก็ลดปัญหาการไม่ราบรื่นรอบจุด x=0 เพื่อให้การเรียนรู้มีประสิทธิภาพมากขึ้น แต่ก็ยังมีข้อจำกัดบางประการ เช่นคำนวณช้ากว่า ReLU เล็กน้อย

สรุปคือ ELU เป็นทางเลือกที่ดีกว่า ReLU ในการนำไปใช้กับโครงข่ายประสาทเทียม เนื่องจากปัญหาที่พบในการใช้ ReLU ถูกแก้ไข ทำให้ประสิทธิภาพการเรียนรู้ดีขึ้น



รูปที่ 2.12 แอ็กทิเวชันฟังก์ชัน ELU ReLU และ GELU [20]

จากรูปที่ 2.12 เป็นกราฟของแอ็กทิเวชันฟังก์ชัน GELU ซึ่งถูกค้นพบโดย Dan Hendrycks และ Kevin Gimpel ในปี ค.ศ. 2016 หรืออีกชื่อคือ Gaussian Error Linear Unit (GELU) ซึ่งค่า ฟาย (Φ) เป็นฟังก์ชันการแจกแจงสะสมแบบเกาส์เซียนมาตรฐาน (Gaussian Cumulative Distribution Function) และ Φ(x) หมายถึงความน่าจะเป็นตัวแปรสุ่มจากการแจกแจงแบบปกติที่มีค่าเฉลี่ย 0 และความแปรปรวน 1 ซึ่งมีต่ำกว่า x

จากกราฟ GELU นั้นคล้ายกับ ReLU ลักษณะกราฟเริ่มจากด้านซ้ายไปขวา เริ่มจากวิ่งตรง แล้วลดลงจนถึงจุดหนึ่งแล้วจบลงด้วยการพุ่งสูงขึ้นไปด้านขวาบน ด้วยรูปร่างที่ค่อนข้างซับซ้อนจึงทำงานได้ดีกับงานที่ซับซ้อน เพราะเกรเดียนต์ไล่ระดับง่ายขึ้นในการฝึกแบบจำลองซับซ้อนในทางปฏิบัติ ฟังก์ชันนี้มีประสิทธิภาพดีกว่าทุกฟังก์ชันที่กล่าวมา [22] โดยในการทดลองนี้เลือกใช้แอ็กทิเวชันฟังก์ชัน ReLU ในแบบจำลอง LSTM และ GELU ในแบบจำลอง WangchanBERTa

2.12 แบตซ์นอร์มัลไลเซชัน (Batch Normalization: BN)

เป็นงานวิจัยปี ค.ศ. 2015 ของ Sergery Ioffe และ Christian Szegedy ได้เสนอเทคนิคเรียกว่า แบตซ์นอร์มัลไลเซชัน เพื่อแก้ปัญหานี้ ด้วยการเพิ่มการดำเนินการในแบบจำลอง ก่อนหรือหลังแอ็กทิเวชันฟังก์ชันของแต่ละฮิดเดนเลเยอร์ การดำเนินการนี้เพียงแค่เปลี่ยนค่าข้อมูลให้ค่าเฉลี่ยใหม่เป็น 0 และนอร์มัลไลซ์แต่ละอินพุตคือการรีสเกลให้ข้อมูลอยู่ในช่วง 0 กับ 1 จากนั้นปรับขนาด และเลื่อนผลลัพธ์โดยใช้พารามิเตอร์เวกเตอร์ใหม่ 2 ตัวต่อเลเยอร์ ตัวแรกสำหรับการปรับขนาด อีกตัวสำหรับการเลื่อน เพื่อทำให้แบบจำลองเรียนรู้ขนาดและค่าเฉลี่ยที่เหมาะสมจากอินพุตของแต่ละเลเยอร์ หากใช้เลเยอร์ BN เป็นเลเยอร์แรกของนิวรอลเน็ตเวิร์ค ไม่ต้องทำสแตนดาร์ดไดซ์ชุดฝึกก่อน คือไม่ต้องใช้ StandardScaler หรือ Normalization เลเยอร์ BN ทำหน้าที่นี้แทน

ข้อเสียคือการฝึกค่อนข้างช้าในแต่ละอีพ็อค (Epoch) ใช้เวลามากขึ้นเมื่อใช้ BN แต่ข้อดีคือลู่เข้าสู่คำตอบได้เร็วขึ้นมาก และใช้จำนวนอีพ็อคน้อยลงเพื่อประสิทธิภาพที่เท่ากัน ซึ่งส่งผลให้เวลาโดยรวมมักสั้นลง [22]

2.13 ที่มาแบบจำลอง WangchanBERTa

งานวิจัยที่ได้นำเสนอเป็นลักษณะของการพัฒนาแบบจำลองภาษาขนาดใหญ่ (Large Language Model: LLM) ว่ามีพัฒนาการเป็นมาอย่างไรจากจุดเริ่มต้นด้วยนักวิจัย Ashish Vaswani และคณะ ในปี ค.ศ. 2017 ได้พัฒนาแบบจำลองภาษา Transformer ที่สามารถแปลภาษาอังกฤษเป็นฝรั่งเศสได้ กล่าวโดยย่อสถาปัตยกรรมของแบบจำลองภาษา Transformer ประกอบด้วย 2 ส่วน ฝั่งซ้าย คือตัวเข้ารหัส (Encoder) และฝั่งขวา คือตัวถอดรหัส (Decoder) ซึ่งมีนักวิจัยกลุ่มหนึ่งได้นำฝั่งขวา หรือตัวถอดรหัสไปวิจัยต่อจนออกมาเป็นแบบจำลอง GPT (Generative Pre-trained Transformer) หรือเป็นรู้จักในชื่อ ChatGPT ที่ใช้งานกันอยู่ในปัจจุบัน)

ถัดมาในปี ค.ศ. 2018 นักวิจัย Jacob Devlin และคณะ ได้ต่อยอดแบบจำลองภาษา Transformer โดยคิดค้นแบบจำลอง BERT คือการนำฝั่งซ้าย คือตัวเข้ารหัสมาพัฒนาต่อขึ้นมาซึ่งมีงานหลากหลายที่ แบบจำลอง BERT สามารถทำได้หลายงานและเป็นที่มาของงานวิจัยมากมายตามมาในปีเดียวกันโดยนำแบบจำลอง BERT ไปพัฒนาต่อ แต่การทดลองที่งานวิจัยนี้สนใจคือ งานการจัดแยกประเภท (Classification) ต่อมาได้มีนักวิจัย Yinhan Liu และคณะ จากทีมงาน Facebook ในปี 2019 ที่ต่อยอดแบบจำลอง BERT โดยคิดค้นแบบจำลอง RoBERTa ซึ่งมีการปรับปรุงแบบจำลองให้ดีขึ้นกว่า

แบบจำลอง BERT ซึ่งแบบจำลองที่กล่าวมาได้มีการพัฒนาและมีการใช้งานในหลากหลายภาษา หรือเรียกว่าแบบจำลองหลากหลายภาษา (Multi-Lingual Language Model)

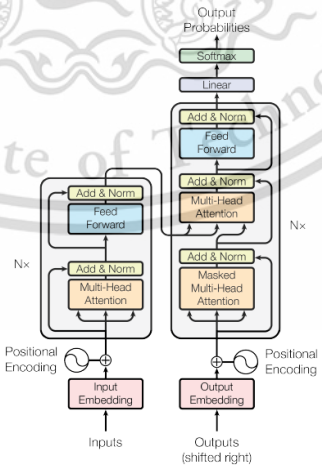
ถัดมาในปี ค.ศ. 2021 นักวิจัยนักวิจัย Lalita Lowphansirikul และคณะ จากทีมงาน VISTEC-depa Thailand Artificial Intelligence Research Institute ได้ต่อยอดแบบจำลองภาษา RoBERTa โดยคิดค้นแบบจำลอง WangchanBERTa ซึ่งมีการเรียนรู้ภาษาไทยภาษาเดียว หรือเรียกว่าแบบจำลองภาษาเดียว (Mono-Lingual Language Model) ส่วนถัดมาเป็นการอธิบายแต่ละแบบจำลองตามลำดับการพัฒนาการ

ในปี ค.ศ. 2017 นักวิจัย Ashish Vaswani และคณะ จากทีมงาน Google ได้นำเสนอแบบจำลองภาษา Transformer ในชื่องานวิจัยว่า “Attention Is All You Need” [23] แบบจำลองภาษา Transformer เป็นสถาปัตยกรรม ในเรื่อง การประมวลภาษาธรรมชาติ (Natural language processing: NLP) ในงานแปลภาษา และ โดยภายในแบบจำลองแบ่งออกเป็น 2 ส่วนคือฝั่งตัวเข้ารหัสและฝั่งตัวถอดรหัส แสดงดังรูปที่ 2.13

1) ฝั่งตัวเข้ารหัส คือมีการแปลงรูปแบบประโยคภาษาที่ต้องการแปลเข้ารหัสให้อยู่ในรูปแบบของเวกเตอร์

2) ฝั่งตัวถอดรหัส คือมีการแปลงเวกเตอร์จากตัวเข้ารหัสให้กลับมาเป็นประโยคภาษาที่ถูกแปล

โดยที่แบบจำลอง Transformer ได้ใช้เซลฟ์แอตเทนชัน (Self-Attention) ในการแปลแต่ละคำแบบจำลองสนใจเฉพาะคำที่เกี่ยวข้องและข้อมูลที่มีความสำคัญ โดยสามารถประมวลผลของคำพร้อม ๆ กันได้โดยอิสระต่อกันและสามารถจัดการกับข้อมูลประโยคที่ยาวได้โดยไม่ลืมข้อมูล

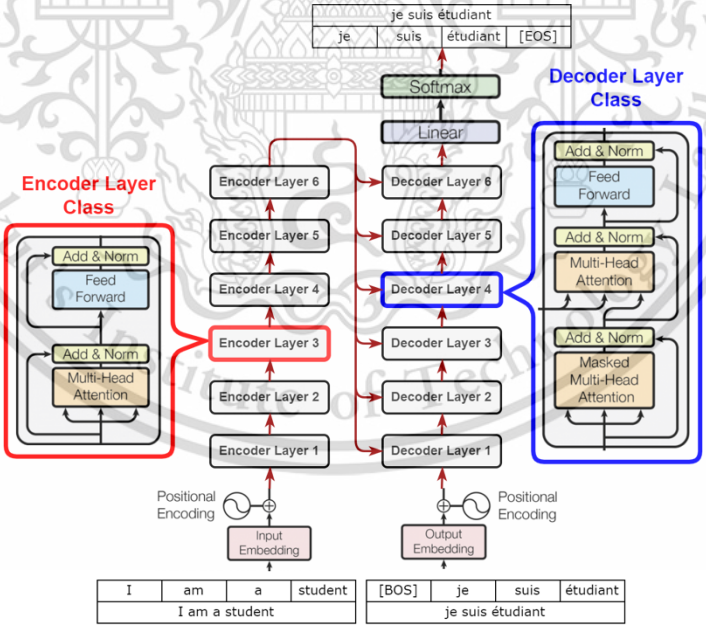


รูปที่ 2.13 สถาปัตยกรรมแบบจำลอง Transformer [23]

1) ตัวเข้ารหัส (Encoder)

จากรูปที่ 2.13 เป็นขาเข้ารหัสฝั่งซ้ายของแบบจำลอง Transformer ซึ่งมีการสร้าง กุญแจ (Key) และคำตอบ (Value) ในส่วนของ มัลติแฮดแอตแทนชัน (Multi-Head Attention) โดยฝั่งตัวเข้ารหัส (Encoder) ประกอบไปด้วยจำนวนชั้นที่เหมือนกัน Nx = 6 ชั้น (Layers) เช่นเดียวกับฝั่งตัวถอดรหัส (Decoder) มีจำนวน Nx = 6 ชั้น เหมือนกัน

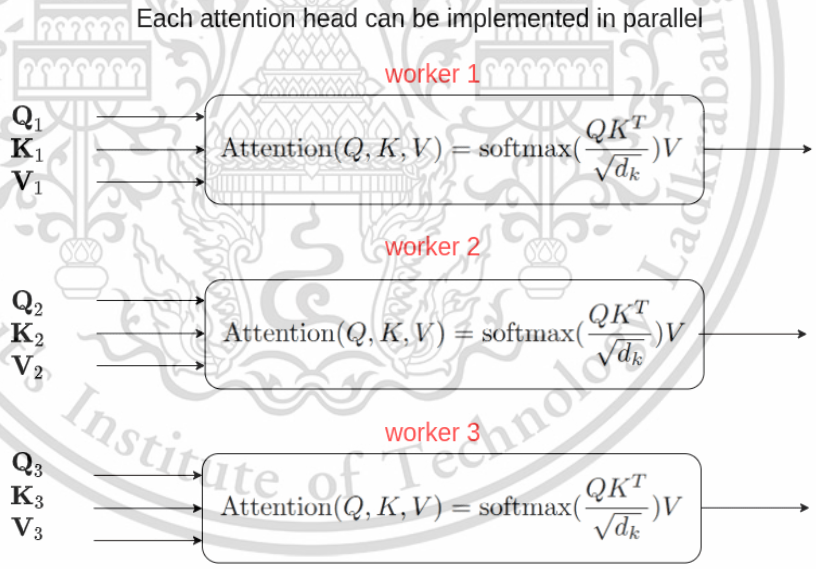
1.1 อินพุตเอ็มแบ็ดดิ้ง (Input Embedding) คือ การแปลงข้อมูลขาเข้าเป็นเวกเตอร์ หรือการแปลงคำเป็นเวกเตอร์ (Word Embedding) ด้วยวิธีการเข้ารหัสจับคู่ (Byte-pair Embedding) คือการเปลี่ยนจากคำที่เป็นตัวแทนตัวเลขของแต่ละเวกเตอร์ให้เป็นคำย่อย (Subword) ซึ่งช่วยแก้ปัญหาคำที่เฉพาะหรือคำหายาก และคำที่ไม่มีในพจนานุกรม จากการแปลงคำเป็นเวกเตอร์แบบเดิมเพราะคำไทยส่วนใหญ่เป็นคำประสมที่เกิดจากคำมูลมาประกอบกันแล้วทำให้มีความหมาย อีกทั้งยังช่วยเรื่องลดพื้นที่การจัดเก็บข้อมูลแบบเดิมที่ใช้คำในพจนานุกรมอีกด้วย เช่น ต้องการแปลภาษาจากภาษาอังกฤษไปเป็นฝรั่งเศส ให้ใส่คู่ภาษาลงไปให้แบบจำลองเรียนรู้หลาย ๆ คู่ประโยค จากรูปที่ 2.14 ประโยคภาษาอังกฤษจากรูปคือ “I am a student” เข้าไปแปลงคำเป็นเวกเตอร์



รูปที่ 2.14 ฝั่งตัวเข้ารหัส (Encoder) และฝั่งตัวถอดรหัส (Decoder) [23]

1.2 การเข้ารหัสตำแหน่ง (Positional Encoding) เป็นการเข้ารหัสตำแหน่งแบบคงที่ ซึ่งบอกลำดับของคำ โดยใช้กราฟไซน์ (Sin) และโคไซน์ (Cos) โดยดูจากช่วงขึ้นลงของกราฟสามารถบ่งบอกได้ว่าคำมาก่อนหรือคำใดมาทีหลัง

1.3 มัลติแฮดแอตแทนชัน (Multi-Head Attention) คือมีหลาย แอตเทนชัน (Attention) ทำงานพร้อมกันช่วยกันสกัดความรู้จากเนื้อหาออกมา แอตเทนชันภายในของตัวเข้ารหัส (Encoder Self-Attention) ด้วยการสนใจประโยคตัวเองทีละคำแต่บวกทุกคำรวมกันเป็นเวกเตอร์เฉลี่ย โดยมีการใช้งานทั้งหมด 8 แอตเทนชัน หรือคนงาน (Worker) ทั้งหมด 8 คน โดยแต่ละแอตเทนชันมีการคำนวณค่าคิวรี กุญแจ และคำตอบ และสามารถทำงานคู่ขนานไปพร้อมกันซึ่งทำให้การ์ดจอส่งผลต่อความเร็วในการประมวลผลแบบจำลองประเภทนี้ แสดงดังรูปที่ 2.15 โดยจะมีการอธิบายรายละเอียดเพิ่มเติมในหัวข้อ 2.4

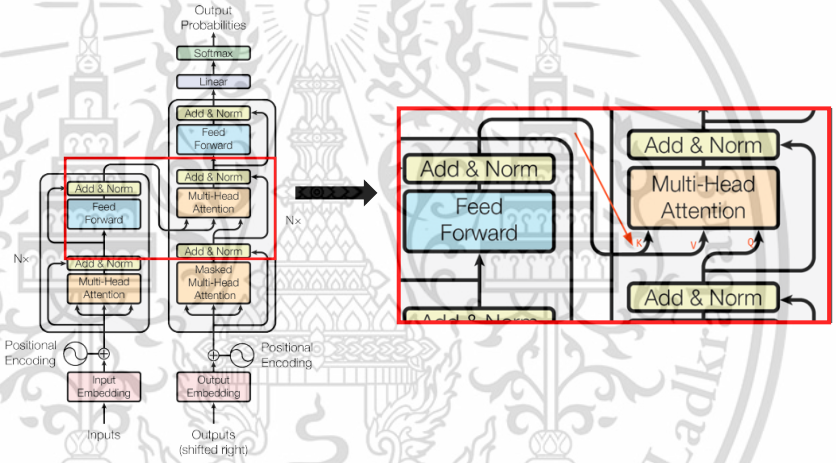


รูปที่ 2.15 มัลติแฮดแอตแทนชัน (Multi-Head Attention) ทำงานแบบคู่ขนาน [24]

1.4 การป้อนข้อมูลไปข้างหน้า (Feed Forward) การผ่านโครงข่ายระบบประสาท (NN) มีการปรับน้ำหนัก (Weight) ความรู้ที่สกัดได้ออกมาจากเส้นตรงหลาย ๆ เส้นให้เป็นรูปร่างทางวัตถุหรือเรขาคณิตที่ซับซ้อนขึ้นโดยมีทั้งหมด 2 ชั้น โดยชั้นแรกมีโหนดทั้งหมด 2,048 โหนด และชั้นที่สองมีโหนดทั้งหมด 512 โหนด ซึ่งแต่ละโหนดคูณค่าน้ำหนักที่เท่ากันโดยใช้ แอ็กทิเวชันฟังก์ชันเป็น ReLU

2) ตัวถอดรหัส (Decoder)

ฝั่งขวาของแบบจำลอง Transformer คือฝั่งตัวถอดรหัสมีการรับค่ากุญแจ และคำตอบเข้ามาจากฝั่งตัวเข้ารหัส เข้ามาในมัลติแฮดแอตแทนชันในส่วนของฝั่งตัวถอดรหัส และรับค่าคิวรี (Query) มาจากมัลติแฮดแอตแทนชันที่แมสก์แล้ว (Masked Multi-Head Attention) แสดงดังรูปที่ 2.16



รูปที่ 2.16 คิวรี (Query) กุญแจ (Key) คำตอบ (Value) [23]

2.1 ขาออกเอ็มแบ็ดดิ้ง (Output Embedding) คือการแปลงคำเป็นเวกเตอร์ (Word Embedding) ด้วยวิธีการเข้ารหัสจับคู่ (Byte-pair Embedding) เช่นเดียวกับหัวข้อที่ 1.1 เช่น ต้องการแปลภาษาจากภาษาอังกฤษไปเป็นฝรั่งเศส ให้ใส่คู่ภาษาที่จะต้องการแปล คือ ภาษาฝรั่งเศส จากรูปที่ 2.13 ประโยคฝรั่งเศส จากรูปคือ “ja suis étudiant” เข้าไปแปลงคำเป็นเวกเตอร์ ซึ่งมีความหมายเดียวกันกับภาษาอังกฤษคือ “I am a student”

2.2 การเข้ารหัสตำแหน่ง คือการบอกลำดับของคำจากช่วงขึ้นลงของกราฟสามารถบ่งบอกได้ว่าคำใดมาก่อนหรือคำใดมาหลังเหมือนกับหัวข้อ 1.2

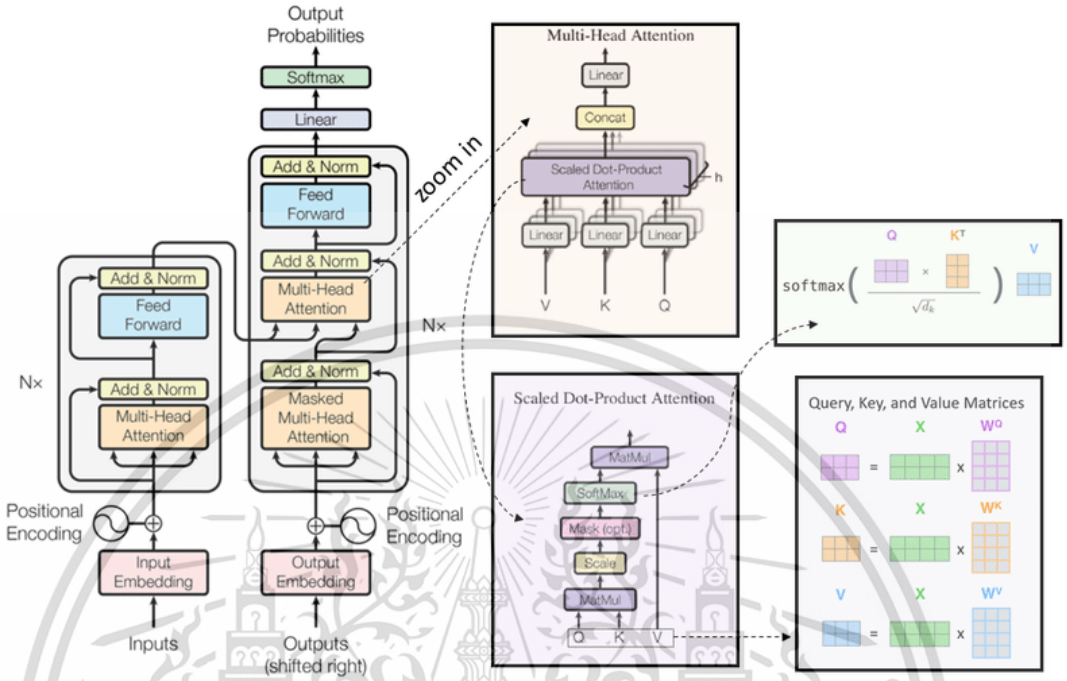
2.3 มัลติแฮดแอตแทนชันที่แมสก์แล้ว (Masked Multi-Head Attention) สร้างคิวรี

(Decoder Self-Attention) เรียนรู้ฝั่งขาออกเฉพาะส่วนที่ควรรู้ คำอื่น ๆ ถูกปิดไว้ไม่ให้รู้คำตอบเพราะเป็นการรู้คำตอบล่วงหน้า และให้เดาคำอนาคตโดยที่ไม่รู้คำตอบ คำที่ไม่ให้รู้คำตอบมีค่า ลบอนันต์ (-inf) ซึ่งหลังจากนำค่า $\frac{\mathrm{QK}^{T}}{\sqrt{d_k}}$ ที่เป็น ลบอนันต์ (-inf) มาผ่านซอฟต์แม็กซ์ (Softmax) ทำให้ได้ค่า 0 ออกมา ทำให้ไม่เกิดกระบวนการแอตเทนชันในคำที่ถูกปิดไว้ ไม่มีการตีความหมายเป็นบริบทใด ๆ ออกมาได้ เช่น “This is a cat” ถ้าสนใจคำว่า “is” อยู่เป็นคำที่ 2 ทำให้คำที่ 3 และ 4 คือ “a” และ “cat” ถูกปิดไว้ไม่ให้เห็น หรือสนใจคำว่า “a” อยู่เป็นคำที่ 3 ทำให้คำที่ 4 คือ“cat” ถูกปิดไว้ไม่ให้เห็น ถูกปิดไว้คือทำให้เป็นค่า ลบอนันต์ (-inf) และกลายเป็นแอตเทนชันมีค่า 0 ในเวลาต่อมา

แบบจำลองภาษาที่แมสก์แล้ว (MLM) มีลักษณะเหมือนกับเป็นการเพิ่มสิ่งรบกวนลงไปในข้อความซึ่งเป็นการเสริมข้อมูลด้วยวิธีการรบกวน คล้ายกับข้อมูลที่เป็นรูปภาพแมวที่มีการหมุน ย่อ ขยาย ให้ภาพเดียวเกิดเป็นหลายภาพซึ่งทำให้แบบจำลองมีความทนทานต่อข้อมูลที่ไม่เคยพบมาก่อนและมีประสิทธิภาพต่อความแม่นยำเพิ่มมากขึ้น

2.4 มัลติแฮดแอตแทนชัน (Multi-Head Attention) คือมีหลาย แอตเทนชัน (Attention) ทำงานพร้อมกันช่วยกันสกัดความรู้จากเนื้อหาออกมา แอตเทนชันภายในของตัวเข้ารหัส-ตัวถอดรหัส (Encoder Decoder Attention) ด้วยการสนใจประโยคทีละคำรวมถึงคำตัวเองแต่บวกทุกคำรวมกันเป็นเวกเตอร์เฉลี่ย โดยมีการใช้งานทั้งหมด 8 แอตเทนชัน ทำงานคู่ขนานพร้อมกัน

โดยวิธีการคำนวณ เซลฟ์แอตเทนชัน (Self-Attention) คือการนำอินพุตเอ็มแบ็ดดิ้งที่รวมกับการเข้ารหัสตำแหน่งแล้วมาคูณกับน้ำหนัก (Weight: W) ค่าสุ่มเริ่มต้นของแต่ละ น้ำหนักคิวรี (W-query) น้ำหนักกุญแจ (W-key) น้ำหนักคำตอบ (W-value) โดยค่าน้ำหนักมีการเปลี่ยนไปทุกรอบที่มีการคำนวณใหม่ หลังการคำนวณเสร็จได้ค่า คิวรี กุญแจ และคำตอบ ค่าใหม่ ต่อไปทำการตรวจสอบคิวรีกับกุญแจโดยวิธีผลคูณจุด (Dot Product) ว่าเวกเตอร์ใดมีความใกล้เคียงกันที่สุดแล้วนำค่าคำตอบไปใช้ โดยค่า QK^T ที่ได้ถูกหารด้วยความยาวของกุญแจ คือรากที่สองของ 64 เท่ากับ 8 (สาเหตุที่ต้องมีการหารด้วยรากที่สองความยาวของกุญแจ ทำให้ผลคูณ QK^T คงที่ และลดความแปรปรวนของผลคูณ QK^T และทำให้ผลคูณค่า QK^T มีความแปรปรวนเข้าใกล้ 1 เหมือนกับค่า Q และค่า K^T) ถัดไปทำซอฟต์แม็กซ์ (Softmax) คือการหาค่าความน่าจะเป็นสูงสุดมาเป็นคำตอบและคูณกับค่า คำตอบ ทำให้ได้ค่า เซลฟ์แอตเทนชันออกมาเพื่อไปสู่ขั้นตอนถัดไป แสดงดังรูปที่ 2.17

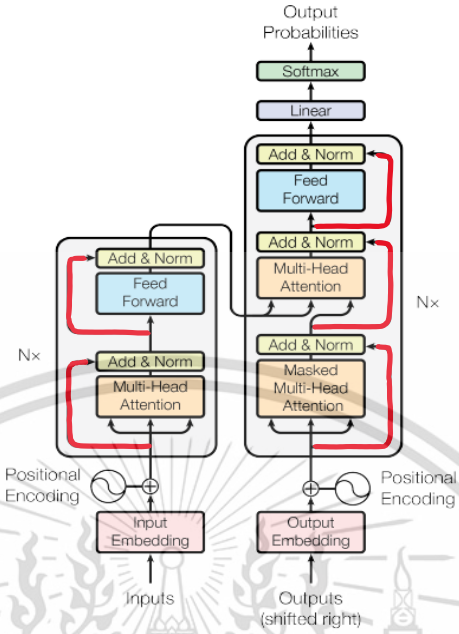


รูปที่ 2.17 มัลติแฮดแอตแทนชัน (Multi-Head Attention) และคิวรี (Query) กุญแจ (Key) คำตอบ (Value) [23]

2.5 การป้อนข้อมูลไปข้างหน้า (Feed Forward) การผ่านโครงข่ายระบบประสาท (NN) มีการปรับน้ำหนัก เหมือนหัวข้อ 1.4 แล้วนำไปผ่านการทำ linear และทำซอฟต์แม็กซ์ (Softmax) เพื่อให้ได้คำตอบออกมาเป็นอีกภาษาหนึ่ง ของคู่ภาษาที่ต้องการแปล

3) ส่วนอื่น ๆ

3.1 การเชื่อมส่วนข้อมูลที่หายไป (Residual) เป็นการแก้ปัญหาเกรเดียนต์ที่หายไป (Vanishing gradient) ที่ทำให้เกรเดียนต์มีขนาดเล็กลงจนเท่ากับ 0 ทำให้ค่าน้ำหนักไม่ถูกอัปเดต โดยใช้วิธีส่งต่อข้อมูลข้ามชั้น (Skip Connection) ได้นำข้อมูลสุดท้ายมารวมกับข้อมูลก่อนหน้า แสดงดังรูปที่ 2.18 ที่ถูกวาดด้วยเส้นสีแดงทั้ง 5 เส้น



รูปที่ 2.18 การเชื่อมส่วนข้อมูลที่หายไป (Residual) [23]

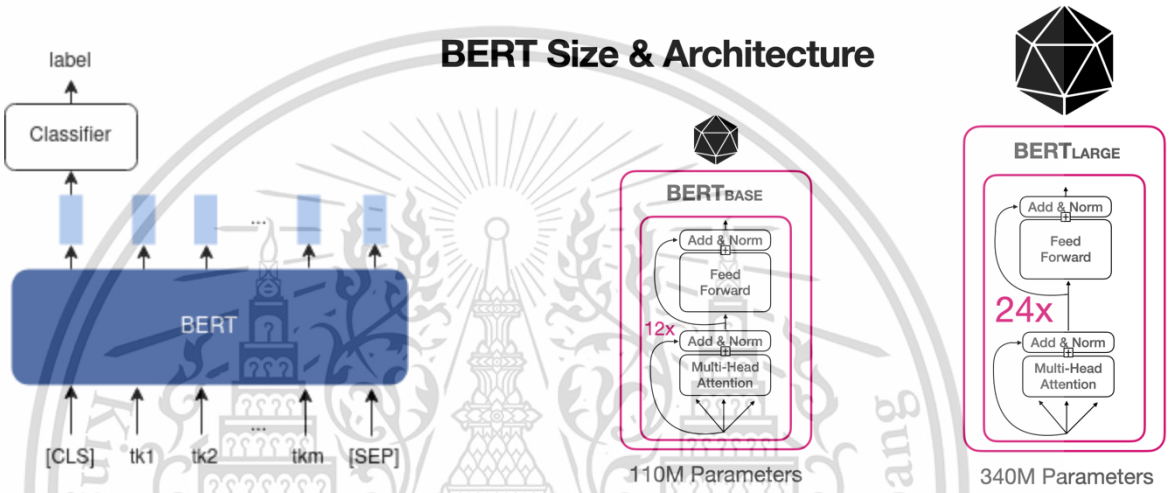
3.2 เลเยอร์นอร์มัลไลเซชัน (Layer Normalization) เป็นวิธีการปรับช่วงข้อมูลให้เป็นมาตรฐาน (Normalization) รูปแบบหนึ่งที่ช่วยให้แบบจำลองมีการลู่เข้าสู่คำตอบได้เร็วขึ้น

3.3 Dropout เป็นการสุ่มไม่คิดคำนวณบางโหนดในทุก ๆ ขั้นตอนของแบบจำลอง Transformer โดยมี Dropout Rate ที่ 0.1 หรือ 10 เปอร์เซ็นต์ ที่มีการปิดโหนดไว้ ทำให้แบบจำลองมีความทนทานต่อข้อมูลที่ไม่เคยเจอมาก่อนและมีประสิทธิภาพต่อความแม่นยำเพิ่มมากขึ้น

สรุปหลักการทำงานของแบบจำลอง Transformer มี 4 ส่วนที่สำคัญได้แก่ 1) การแปลงคำเป็นเวกเตอร์ 2) การเข้ารหัสตำแหน่ง 3) เซลฟ์แอตเทนชัน 4) การเชื่อมส่วนข้อมูลที่หายไป

สุดท้ายผลวัดค่าการแปลภาษาด้วยตัววัดแบบ BLEU Score โดยมีการให้คะแนนความถูกต้องของการแปลภาษาในระดับของคำ และความยาวของรูปประโยค การแปลภาษาจากภาษาอังกฤษไปเป็นฝรั่งเศส ได้คะแนน 41.8/100 และภาษาอังกฤษไปเป็นภาษาเยอรมัน ได้คะแนน 28.4/100

ในปี ค.ศ. 2018 นักวิจัย Jacob Devlin และคณะ จากทีมงาน Google ได้นำเสนอ แบบจำลองภาษา BERT [25] เป็นสถาปัตยกรรมที่ต่อยอดมาจาก Transformers ในชื่องานวิจัยว่า “BERT” ซึ่งย่อมาจาก Bidirectional Encoder Representations from Transformers โดยมีการนำ ฝั่งตัวเข้ารหัสของแบบจำลอง Transformer มาต่อเพิ่มส่วนขยาย (Classifier) แสดงดังรูปที่ 2.19



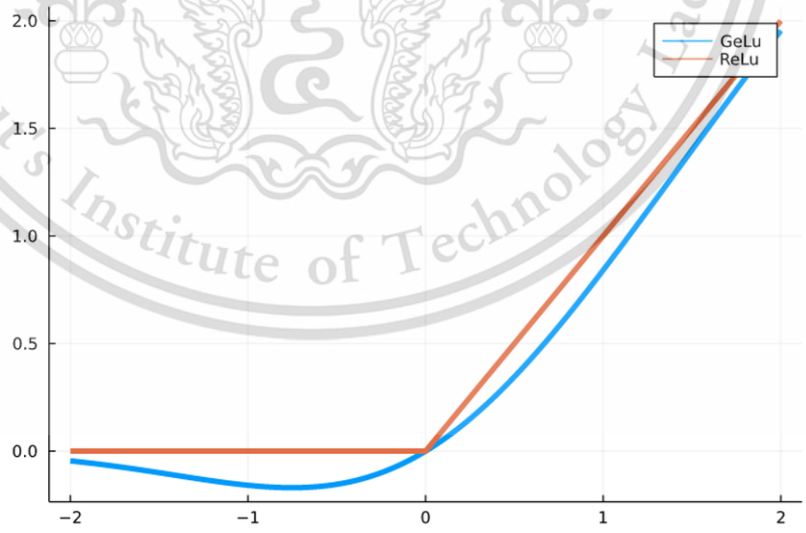
รูปที่ 2.19 สถาปัตยกรรมแบบจำลอง BERT [26, 27]

พร้อมทั้งมีการมอง 2 ทิศทางจากอดีตไปอนาคตหรือฝั่งซ้ายไปขวาของประโยค และอนาคตกลับมาอดีตหรือฝั่งขวาไปซ้ายของประโยค คือเซลฟ์แอตเทนชัน เหมือนกับวิธีแบบจำลอง Transformer และได้มีการเพิ่มขนาดชั้น (Layers) เอ็มแบ็ดดิ้ง (Embedding) และ แอตเทนชัน (Attention) โดยมีการสร้างแบบจำลอง BERT 2 แบบ คือขนาดปกติ และขนาดใหญ่ แสดงดังตารางที่ 2.2

ตารางที่ 2.2 เปรียบเทียบความแตกต่างของแบบจำลอง BERT ทั้ง 2 แบบ [25]

แบบจำลอง BERT	**ชั้น (layers)**	**เอ็มแบ็ดดิ้ง (Embedding)**	**แอตเทนชัน (Attention)**	**พารามิเตอร์ ล้าน**
ขนาดปกติ (Base)	12	768	12	110
ขนาดใหญ่ (Large)	24	1024	16	340

ส่วนที่แบบจำลอง BERT แตกต่างกับแบบจำลอง Transformer คือแบบจำลอง Transformer มีการใช้งานแอ็กทิเวชันฟังก์ชันแบบ RELU แต่ในส่วนของ BERT มีการใช้งานแอ็กทิเวชันฟังก์ชันแบบ GELU ซึ่งช่วยให้ช่วงที่มีค่าน้ำหนักน้อยกว่า 0 นั้นมีการปรับปรุงและทำให้แบบจำลอง เรียนรู้ได้ดีขึ้น แสดงดังรูปที่ 2.20



รูปที่ 2.20 กราฟเปรียบเทียบ RELU และ GELU [28]

ใน BERT นั้นมีขั้นตอนทั้งหมด 2 ขั้นตอน ได้แก่ ขั้นตอนการฝึกล่วงหน้า (Pre-train) และขั้นตอนการปรับแต่ง (Fine-Tuning)

1. ขั้นตอนการฝึกล่วงหน้า (Pre-train) เป็นการสอนแบบจำลองให้เรียนรู้โครงสร้างทางภาษาโดยรวมก่อน

1.1 แบบจำลองภาษาที่แมสก์แล้ว (Masked Language Model: MLM) มีการสุ่มแมสก์ไป 15 เปอร์เซ็นต์ ของชิ้นส่วนคำ (WordPiece) โดยสัดส่วนการแมสก์คำนั้นมีดังนี้

1) 80 เปอร์เซ็นต์ ในการสุ่มแมสก์คำ
2) 10 เปอร์เซ็นต์ แทนที่ด้วยคำอื่นแบบสุ่ม
3) 10 เปอร์เซ็นต์ ไม่มีการเปลี่ยนแปลง

ซึ่งเป็นสัดส่วนที่ดีที่สุด 80-10-10 แต่อาจพบปัญหาเพราะมีการแมสก์คำในขั้นตอนการฝึกล่วงหน้า แต่ในขั้นตอนการปรับแต่งนั้นไม่มีการใช้ MLM ทำให้ถ้ามีโครงสร้างทางภาษาต่างกันอาจทำให้แบบจำลองเรียนรู้ได้ไม่ดี

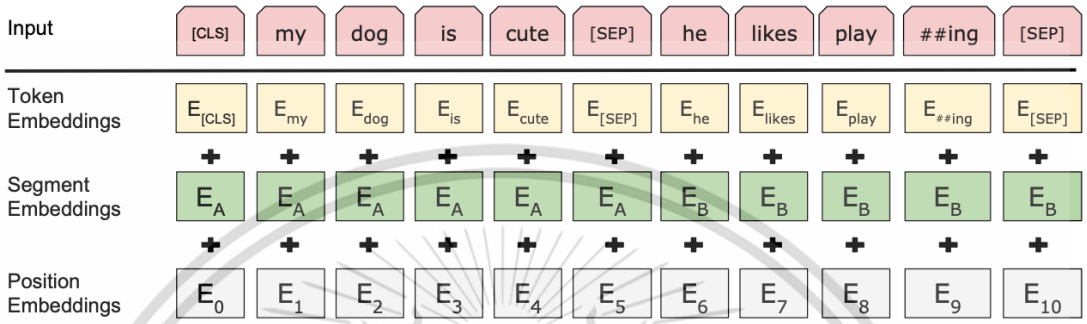
1.2 การทำนายประโยคถัดไป (Next Sentence Prediction: NSP) เนื่องจาก MLM นั้นเป็นการทำความเข้าใจ ในระดับคำในการใช้ MLM แต่ NSP เป็นการเข้าใจในระดับประโยคโดยนำคู่ประโยคมาทำนายว่าอยู่ติดกันหรือไม่ โดยแบ่งข้อมูลครึ่งแรกเป็นประโยคที่อยู่ติดกันและครึ่งหลังเป็นประโยคแบบสุ่ม

1.3 ข้อมูลการฝึกสอนใช้ข้อมูลจาก Wikipedia 2,500 ล้านคำและ BooksCorpus 800 ล้านคำ

การแปลงคำเป็นเวกเตอร์ (Word Embedding) ของ BERT ประกอบด้วยผลรวมทั้ง 3 ชนิด ได้แก่

1) Token Embeddings เป็นตัวแทนข้อมูลเกี่ยวกับคำ
2) Positional Embeddings เป็นตัวแทนข้อมูลเกี่ยวกับตำแหน่ง
3) Segment Embeddings เป็นตัวแทนข้อมูลของคำนั้นอยู่ในประโยคที่ 1 หรือ 2

เพิ่มเติมโทเค็น (Token) พิเศษเข้ามาคือ [CLS] ใช้แทนตำแหน่งเริ่มต้น และ [SEP] เป็นตัวคั้นประโยคที่ 1 และประโยคที่ 2 แสดงดังรูปที่ 2.21



รูปที่ 2.21 การแปลงคำเป็นเวกเตอร์ (Word Embedding) ขาเข้าของ BERT [25]

การทำไฮเปอร์พารามิเตอร์ (Hyperparameters) ของ BERT ในช่วงขั้นตอนการฝึกล่วงหน้านั้นมีการกำหนดค่าต่าง ๆ แสดงดังตารางที่ 2.3

ตารางที่ 2.3 การปรับแต่งพารามิเตอร์ของแบบจำลอง BERT ในขั้นตอนการฝึกล่วงหน้า [25]

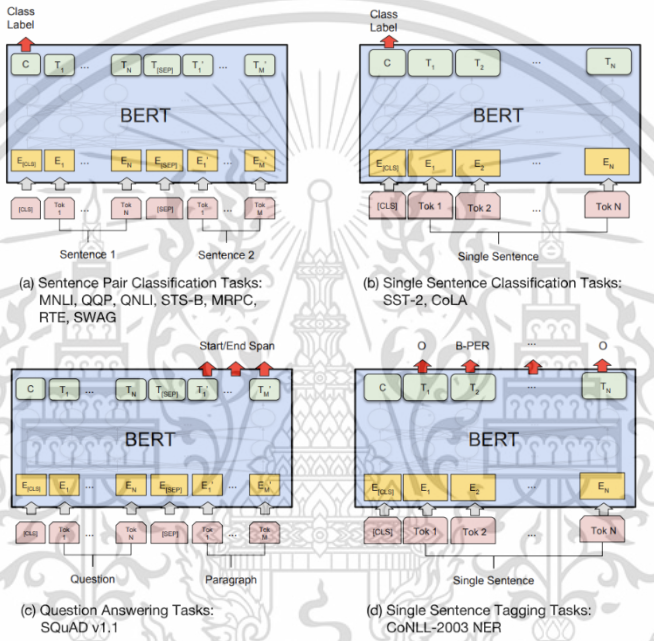
พารามิเตอร์	ผลลัพธ์การปรับแต่ง
ความยาวลำดับ (Sequence Length)	256
ขั้นตอนการฝึกสอน (Training Steps)	1 ล้านก้าว (steps) หรือ 40 อีพ็อค (epochs)
ออปติไมเซอร์ (Optimizer)	Adam ($\beta 1 = 0.9$ $\beta 2 = 0.999$)
อัตราการเรียนรู้ (Learning Rate)	10^{-4}

2. ขั้นตอนการปรับแต่ง (Fine-Tuning) เป็นการสอนแบบจำลองให้เรียนรู้เพิ่มเติม เพื่อนำ BERT ไปใช้งานตามวัตถุประสงค์การใช้งานแบบจำลองทางภาษาของนักพัฒนา

เป็นขั้นตอนในการปรับแบบจำลองให้เหมาะสมกับงานของผู้พัฒนาซึ่งต้องมีการสอนแบบจำลองอีกรอบให้เหมาะสมกับงานที่นำไปใช้มีงานหลากหลายตามประเภทชุดข้อมูลที่มีการอธิบายไว้ช่วงท้ายของแบบจำลอง BERT แสดงดังรูปที่ 2.22

2.1 งานจำแนกชนิดคำในข้อความ (Token Classification) เช่น งานหาชื่อเฉพาะ (Named Entity Recognition: NER) ประเภทงาน (d) หรือ งานการจัดแยกประเภท (Classification) ประเภทงาน (b)

2.2 งานจำแนกข้อความ (Sequence Classification) เช่น งานหาความเหมือนกันของความหมายในประโยค (Semantic Textual Similarity) ประเภทงาน (a) หรือ งานด้านดูคู่ประโยคว่าเป็นเหตุเป็นผลกันหรือไม่ (Natural Language Inference: NLI) ประเภทงาน (a)



รูปที่ 2.22 การปรับแต่ง (Fine-Tuning) แบบจำลอง BERT ในงานที่แตกต่างกัน [25]

จากการใช้งาน BERT จากการทดลองการปรับแต่งโดยมีค่าที่แนะนำการใช้งาน แสดงดังตารางที่ 2.4

ตารางที่ 2.4 การปรับแต่งพารามิเตอร์ของแบบจำลอง BERT ในขั้นตอนการปรับแต่ง [25]

พารามิเตอร์	ผลลัพธ์การปรับแต่ง
การสุ่มลบ (Dropout)	0.1
ขั้นตอนการฝึกสอน (Training Steps)	3 - 4 อีพ็อค (epochs)
ออปติไมเซอร์ (Optimizer)	Adam (β1 = 0.9 β 2 = 0.999)
อัตราการเรียนรู้ (Learning Rate)	$2x10^{-5}$, $3x10^{-5}$, $5x10^{-5}$
ขนาดแบตช์ (Batch Size)	16 , 32

การประเมินด้วย การเข้าใจทางภาษาทั่วไป (General Language Understanding Evaluation: GLUE) เป็นแบบทดสอบโดยมีการคัดเลือกชุดข้อมูลหลากหลายโจทย์ทั้ง 9 ชุดไว้แล้วใช้ค่าเฉลี่ยทั้ง 9 โจทย์เป็นตัววัดความฉลาดของแบบจำลองทางภาษายกตัวอย่างแค่บางชุดข้อมูล [29]

1) MNLI ย่อมาจาก The Multi-Genre Natural Language Inference Corpus ข้างในชุดข้อมูลนี้ประกอบด้วยการถอดเสียงออกมาจากนวนิยาย รายงานของรัฐบาล เป็นต้น ข้อมูลรวมคู่ประโยคกว่า 550,000 ตัวอย่าง เป็นโจทย์ด้านดูคู่ประโยคว่าเป็นเหตุเป็นผลกันหรือไม่ (NLI) โดยมี 3 ประเภท การเกี่ยวข้อง (Entailment) ขัดแย้ง (Contradiction) เป็นกลาง (Neutral) และมีการแยกการประเมินที่เป็นประเภทตรงกัน (Match) (MNLI-m) และไม่ตรงกัน (Mismatch) (MNLI-mm)

2) SST-2 ย่อมาจาก The Stanford Sentiment Treebank ข้างในชุดข้อมูลนี้ประกอบด้วยคำวิจารณ์ภาพยนตร์ และคำอธิบายความรู้สึกของมนุษย์เป็นโจทย์การทำนายความรู้สึก 2 ประเภทด้านบวกและด้านลบของประโยคเดี่ยว

3) MRPC ย่อมาจาก The Microsoft Research Paraphrase Corpus ข้างในชุดข้อมูลนี้ประกอบด้วยคู่ประโยค ที่ดึงมาจากข่าวออนไลน์พร้อมคำอธิบายจากมนุษย์เป็นโจทย์ดูความคล้ายคลึงกันของคู่ประโยค

การประเมินด้วย SQuAD v1.1 เป็นชุดคู่คำถามเนื้อเรื่อง 100,000 คู่ เป็นโจทย์บททดสอบด้านถามตอบ โดยมีคำถามและเนื้อเรื่องให้มา และให้ตอบคำถามจากเนื้อเรื่องที่ให้มา และ SQuAD v2.0 เป็นบททดสอบด้านถามตอบ โดยบางคำถามไม่ได้มีคำตอบในเนื้อเรื่องที่ให้มาทำให้ปัญหานั้นสมจริงมากขึ้น

การประเมินด้วย SWAG เป็นบททดสอบมีคำถามและมีคำตอบให้เลือก 4 ตัวเลือก

ในปีเดียวกัน นักวิจัย Yinhan Liu และคณะ จากทีมงาน Facebook ได้นำเสนอ แบบจำลองภาษา RoBERTa [30] เป็นสถาปัตยกรรมที่ต่อยอดมาจาก BERT ในชื่องานวิจัยว่า “RoBERTa” ซึ่งย่อมาจาก Robustly Optimized BERT Pretraining Approach ซึ่งได้ปรับปรุงให้ดีขึ้นโดย ปรับการฝึกล่วงหน้าของแบบจำลองภาษาที่แมสก์แล้ว (Pretrain MLM) ให้เป็นแบบไดนามิกแทนแบบคงที่ ลบการใช้การทำนายประโยคถัดไป (NSP) เพราะไม่ได้ช่วยเพิ่มประสิทธิภาพ เพิ่มความยาวลำดับ (Sequence Length) จาก 256 เป็น 512 รวมทั้งการสอนแบบจำลองให้นานขึ้นโดยการเพิ่มขนาดแบตช์มีการเพิ่มขนาดอัตราการเรียนรู้ให้สูงขึ้น และเพิ่มชุดข้อมูลฝึกล่วงหน้า (Pretrained Dataset) ให้ใหญ่กว่าเดิมมาก ทำให้ได้ผลลัพธ์ดีกว่า BERT โดยภาพรวม

ชุดข้อมูลดั้งเดิมของ BERT มี Wikipedia และ BooksCorpus รวมขนาดข้อมูล 16 กิกะไบต์เท่านั้น และ RoBERTa เพิ่ม ชุดข้อมูล CC-New ขนาด 76 กิกะไบต์ ชุดข้อมูล OpenWebText ขนาด 38 กิกะไบต์ และชุดข้อมูล STORIES ขนาด 31 กิกะไบต์ รวมชุดข้อมูลทั้งหมด 161 กิกะไบต์ และใช้วิธีการตัดคำย่อยแบบการเข้ารหัสจับคู่ (BPE)

ปรับการฝึกล่วงหน้าของแบบจำลองภาษาที่แมสก์แล้ว (Pretrain MLM) ให้เป็นแบบไดนามิกแทนแบบคงที่ หมายความว่า แบบคงที่ ทุกรอบแมสก์ (Masked) เหมือนเดิมทุกรอบ ส่วนแบบไดนามิกในรอบแรกเป็นแบบสุ่มพอพบประโยคเดิมให้เปลี่ยนคำแมสก์

แบบจำลอง RoBERTa ใช้สถาปัตยกรรมเดียวกันกับ แบบจำลอง BERT ขนาดใหญ่ (Bert-Large) : 24 ชั้น 1024 เอ็มแบ็ดดิ้ง 16 แอตเทนชัน

เมื่อมีการใช้ข้อมูลขนาด 160 กิกะไบต์ จำนวนก้าวตั้งแต่ 1 แสน 3 แสน และ 5 แสนตามลำดับ ได้ค่าที่ดีที่สุดเมื่อใช้จำนวนก้าว 5 แสนก้าว เมื่อมีการประเมินจากหลายชุดข้อมูล ได้แก่ (SQuAD v.1.1/v.2.0) 94.6/89.4 (MNLI-m) 90.2 และ (SST-2) 96.4 ได้ประสิทธิภาพดีกว่าเมื่อเทียบกับ BERT ขนาดใหญ่ และ XLNet ขนาดใหญ่ แสดงดังตารางที่ 2.5

ตารางที่ 2.5 เปรียบเทียบประสิทธิภาพของทั้ง 3 แบบจำลอง [30]

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

ขนาดของแบบจำลอง RoBERTa นั้นมี 2 ขนาดเช่นเดียวกับแบบจำลอง BERT คือมี RoBERTa ขนาดปกติ (RoBERTa Base) และ RoBERTa ขนาดใหญ่ (RoBERTa Large) แสดงดังตารางที่ 2.6

ตารางที่ 2.6 ไฮเปอร์พารามิเตอร์ (Hyperparameters) สำหรับฝึกล่วงหน้าของ RoBERTa ขนาดปกติและขนาดใหญ่ [30]

Hyperparam	$\text{RoBERTa}_{\text{LARGE}}$	$\text{RoBERTa}_{\text{BASE}}$
Number of Layers	24	12
Hidden size	1024	768
FFN inner hidden size	4096	3072
Attention heads	16	12
Attention head size	64	64
Dropout	0.1	0.1
Attention Dropout	0.1	0.1
Warmup Steps	30k	24k
Peak Learning Rate	4e-4	6e-4
Batch Size	8k	8k
Weight Decay	0.01	0.01
Max Steps	500k	500k
Learning Rate Decay	Linear	Linear
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.98
Gradient Clipping	0.0	0.0

ต่อมาในปี ค.ศ. 2021 โดยนักวิจัย Lalita Lowphansirikul และคณะ จากทีมงาน VISTEC-depa Thailand Artificial Intelligence Research Institute ได้นำเสนอ แบบจำลองภาษา WangchanBERTa [31] เป็นสถาปัตยกรรมที่ต่อยอดมาจาก RoBERTa ในชื่องานวิจัยว่า “WangchanBERTa”

มีการใช้ข้อมูลภาษาไทยขนาด 78.48 กิกะไบต์ ประกอบด้วยข้อมูลเปิด 6 เปอร์เซ็นต์ ข้อมูลจาก Pantip 28 เปอร์เซ็นต์ และข้อมูลจาก Wisesight-large 66 เปอร์เซ็นต์ และผ่านการทำความสะอาดข้อมูลจนได้เป็นชุดข้อมูล Assorted Thai Texts ขนาด 78.5 กิกะไบต์ และอีกชุดข้อมูลคือวิกิพีเดียขนาด 0.57 กิกะไบต์

ได้นำแบบจำลองของ RoBERTa ขนาดปกติ (RoBERTa Base) มาต่อยอดและใช้วิธีการตัดคำย่อยแบบชิ้นส่วนประโยค (SentencePiece) แต่พบปัญหาตรงการจัดการช่องว่างเพราะการตัดคำย่อยแบบชิ้นส่วนประโยคลบช่องว่างออก ทำให้ต้องเพิ่มโทเค็น (Token) พิเศษคือ ตัวช่องว่าง <_>

เพื่อรักษาช่องว่างที่เป็นการแบ่งคำและประโยคซึ่งสำคัญในภาษาไทยเอาไว้ นำข้อมูลทั้งหมด 15 ล้านข้อความมาฝึกสอนจนได้ขนาดคำศัพท์ทั้งหมด 25,000 คำย่อย

สำหรับในงานวิจัยนี้เลือกใช้งาน RoBERTa เพราะสามารถใช้วิธีการแบบจำลองภาษาที่แมสก์แล้ว (MLM) ได้โดยไม่ใช้วิธีการทำนายประโยคถัดไป (NSP) โดยแบบจำลองภาษาที่แมสก์แล้วมีการสุ่มแมสก์ 15 เปอร์เซ็นต์ ของชิ้นส่วนประโยค โดยสัดส่วนการแมสก์คำ (80 เปอร์เซ็นต์ สุ่มแมสก์คำ : 10 เปอร์เซ็นต์ แทนที่ด้วยคำอื่นแบบสุ่ม : 10 เปอร์เซ็นต์ ไม่มีการเปลี่ยนแปลง)

RoBERTa ขนาดปกติ (RoBERTa Base) ที่มีการปรับไฮเปอร์พารามิเตอร์บางส่วน RoBERTa : 12 ชั้น 768 เอ็มแบ็ดดิ้ง 12 แอตเทนชัน ความยาวลำดับ 416 จากปกติ 512 ใช้ขนาดแบตช์ 4,092 จากปกติ 8,000 และใช้อัตราการเรียนรู้ $3x10^{-4}$ จากปกติ $6x10^{-4}$ เมื่อเทียบกับแบบจำลองภาษา RoBERTa ขนาดปกติ แสดงดังตารางที่ 2.6 และ 2.7

ในแบบจำลอง WangchanBERTa ได้มีการฝึกล่วงหน้าด้วยกัน 5 แบบจำลอง โดยมีสถาปัตยกรรมของ RoBERTa ทั้ง 5 แบบจำลองแต่มีการใช้ชุดข้อมูล หรือโทเค็นไนเซอร์ (Tokenizer) คือตัวตัดคำ แตกต่างกัน เช่น ใช้ตัวตัดคำ หรือพยางค์จาก newmm ซึ่งเป็นวิธีตัดคำแบบสอดคล้องมากที่สุด และมีการใช้วิธีการตัดคำแบบ SEFR หรือ ชิ้นส่วนประโยคในส่วนของชุดข้อมูลมีการใช้ข้อมูลจากวิกิพีเดีย หรือ ใช้ข้อมูลจากชุดข้อมูล Assorted Thai Texts แสดงดังตารางที่ 2.8

ตารางที่ 2.7 ไฮเปอร์พารามิเตอร์ (Hyperparameters) ของ RoBERTa ขนาดปกติ (RoBERTa Base) ชุดข้อมูลวิกิพีเดีย และ ชุดข้อมูล Assorted Thai Texts [31]

Hyperparameters	**RoBERTa$_{BASE}$** (Wikipedia-only Dataset)	**RoBERTa$_{BASE}$** (Assorted Thai Texts Dataset)
Number of Layers	12	12
Hidden size	768	768
FFN hidden size	3,072	3,072
Attention heads	12	12
Dropout	0.1	0.1
Attention dropout	0.1	0.1
Max sequence length	512	416
Effective batch size	8,192	4,092
Warmup steps	1,250	24,000
Peak learning rate	7e-4	3e-4
Learning rate decay	Linear	Linear
Max steps	31,250	500,000
Weight decay	0.01	0.01
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.98	0.999
FP16 training	True	True

ตารางที่ 2.8 ชื่อแบบจำลองของ WangchanBERTa [31]

	Architecture	Dataset	Tokenizer
wangchanberta-base-wiki-spm	RoBERTa-base	Wikipedia-only	SentencePiece
wangchanberta-base-wiki-newmm	RoBERTa-base	Wikipedia-only	word (newmm)
wangchanberta-base-wiki-syllable	RoBERTa-base	Wikipedia-only	syllable (newmm)
wangchanberta-base-wiki-sefr	RoBERTa-base	Wikipedia-only	SEFR
wangchanberta-base-att-spm-uncased	RoBERTa-base	Assorted Thai Texts	SentencePiece

ขั้นตอนการปรับแต่งในงานจำแนกข้อความโดยเทียบกับแบบจำลองอื่น ๆ ที่ทำการปรับแต่ง เช่น NBSVM, ULMFit, XLMR และ mBERT สุดท้ายแนะนำชุดข้อมูลที่นำมาใช้ในแบบจำลองทั้งหมด 4 ชุด ได้แก่

1) Wisesight Sentiment เป็นชุดข้อมูลเกี่ยวกับโซเชียลเน็ตเวิร์คเป็นโจทย์ทำนายอารมณ์ ลบ บวก กลาง หรือคำถาม

2) Wongnai Review เป็นชุดข้อมูลเกี่ยวกับการรีวิวร้านอาหารเป็นโจทย์เกี่ยวกับทำนายดาว ตั้งแต่ 1 ดาวไปจนถึง 5 ดาว

3) Generated Review (EN-TH) เป็นชุดข้อมูลเกี่ยวกับการรีวิวสินค้าโดยทำการแปลจากภาษาอังกฤษเป็นภาษาไทยเป็นโจทย์ทำนายดาวตั้งแต่ 1 ดาวไปจนถึง 5 ดาว

4) Prachathai67k เป็นชุดข้อมูลเกี่ยวกับข่าวโดยมีแท็กทั้งหมด 12 แท็กเป็นโจทย์ทำนายแท็กว่าเรื่องอะไรใน 12 แท็กโดยสามารถใส่ได้มากกว่าหนึ่งแท็ก

ตารางที่ 2.9 ผลลัพธ์เทียบในงานจำแนกข้อความ (Sequence Classification) ของแบบจำลอง RoBERTa ขนาดปกติ เทียบกับ NBSVM ULMFit XLMR และmBERT โดยใช้ตัวชี้วัดค่าเฉลี่ยไมโคร (Micro-Average F1) และค่าเฉลี่ยมาโคร (Macro-Average F1) ตามลำดับ [31]

Model	Wisesight	Wongnai	Generated Reviews (EN–TH) (Review rating)	Prachathai
Existing multilingual models:				
XLMR [Conneau et al., 2019]	73.57 / 62.21	62.57 / 52.75	**64.91 / 60.29**	68.18 / 63.14
mBERT [Devlin et al., 2018b]	70.05 / 57.81	47.99 / 12.97	62.14 / 57.20	66.47 / 60.11
Our baseline models:				
Naive Bayes SVM	72.03 / 54.67	58.38 / 39.75	59.68 / 52.17	66.77 / 60.73
ULMFit (thai2fit)	70.95 / 60.62	61.79 / 48.04	64.33 / 59.33	66.21 / 60.21
Our pretrained $\text{RoBERTa}_{\text{BASE}}$ *models*:				
wangchanberta-base-wiki-spm	73.94 / 60.13	60.60 / 48.17	63.43 / 58.43	68.85 / 63.46
wangchanberta-base-wiki-newmm	72.74 / 55.87	59.81 / 45.75	63.70 / 58.41	68.78 / 63.50
wangchanberta-base-wiki-syllable	73.42 / 59.12	60.36 / 46.68	63.53 / 58.73	68.90 / 63.59
wangchanberta-base-wiki-sefr	70.80 / 59.51	59.83 / 48.21	63.31 / 58.85	67.45 / 61.14
wangchanberta-base-att-spm-uncased	**76.19 / 67.05**	**63.05 / 52.19**	64.66 / 59.54	**69.78 / 64.90**

สำหรับอีกโจทย์ขั้นตอนการปรับแต่งเป็นงานการจำแนกชนิดคำในข้อความโดยมีชุดข้อมูลทั้งหมด 2 ชุดได้แก่

1) Thainer เป็นชุดข้อมูล เกี่ยวกับโจทย์หาชื่อเฉพาะ (NER) แบ่งแท็กทั้งหมดเป็น 13 แท็ก ขนาด 6,456 ข้อความ

2) LST20 เป็นชุดข้อมูล เกี่ยวกับโจทย์ งานหาหน้าที่ของคำ (POS) 16 แท็ก และหาชื่อเฉพาะ (NER) 10 แท็ก ขนาด 78,931 ข้อความ

ตารางที่ 2.10 ผลลัพธ์เทียบในงานการจำแนกชนิดคำในข้อความ (Token Classification) ของแบบจำลอง RoBERTa ขนาดปกติ เทียบกับ CRF XLMR และmBERT โดยใช้ตัวชี้วัดค่าเฉลี่ยไมโคร (Micro-Average F1) และค่าเฉลี่ยมาโคร (Macro-Average F1) ตามลำดับ [31]

Model	ThaiNER	LST20	
	NER	POS	NER
Existing multilingual models:			
XLMR [Conneau et al., 2019]	83.25 / 67.23	96.57 / 85.00	73.61 / 68.67
mBERT [Devlin et al., 2018b]	81.48 / 73.97	96.44 / **85.86**	75.05 / 68.25
Our baseline models:			
Conditional Random Fields (CRF)	78.98 / **81.85**	96.28 / 81.28	75.94 / 72.13
Our pretrained $\text{RoBERTa}_{\text{BASE}}$ *models*:			
wangchanberta-base-wiki-spm	56.64 / 55.34	96.18 / 83.99	77.12 / 71.32
wangchanberta-base-wiki-newmm	58.54 / 47.71	96.14 / 83.11	76.59 / 70.57
wangchanberta-base-wiki-syllable	83.23 / 76.64	96.06 / 83.98	76.45 / 70.37
wangchanberta-base-wiki-sefr	85.04 / 77.73	96.36 / 85.24	76.25 / 69.34
wangchanberta-base-att-spm-uncased	**86.49** / 79.29	**96.62** / 85.44	**78.01 / 72.25**

จากตารางที่ 2.9 และ 2.10 สังเกตเห็นว่า wangchanberta-base-att-spm-uncased สามารถทำผลลัพธ์ออกมาได้ดี มีคะแนนน้อยกว่าในงานจำแนกข้อความของชุดข้อมูล Generated Review (EN-TH) ให้กับแบบจำลอง XLMR ไป 0.25 ในค่าเฉลี่ยไมโคร (Micro-Average F1) เท่านั้น สาเหตุมาจากแบบจำลอง XLMR เป็นแบบจำลองหลากหลายภาษา (Multi-Lingual Language Model) ทำให้ได้เปรียบเล็กน้อย

ทำให้สรุปได้ว่าแบบจำลองภาษาเดียว (Mono-Lingual Language Model) นั้นสามารถทำงานได้ดีกว่าแบบจำลองหลากหลายภาษา เมื่อใช้งานกับภาษาที่ออกแบบมาเฉพาะภาษานั้นอย่างเดียว

2.14 งานวิจัยที่เกี่ยวข้อง

2.14.1 การจัดการข้อมูลไม่สมดุล

ในปี ค.ศ. 2019 นักวิจัย Suphamongkol Akkaradamrongrat และคณะ ได้ศึกษาเรื่อง การสร้างข้อความสำหรับการจำแนกประเภทข้อความที่ไม่สมดุล คือข้อมูลไม่สมดุลพบได้บ่อยครั้งในข้อมูลจริง ส่งผลให้เกิดการเอนเอียงของแบบจำลองการจำแนกประเภท นั่นคือแบบจำลองจะทำนายตัวอย่างส่วนใหญ่เป็นหมวดหมู่หลักซึ่งมักจะเป็นประเภทผลลบ ในงานวิจัยนี้ใช้เทคนิคการสร้างข้อความสังเคราะห์เพื่อสร้างตัวอย่างประเภทกลุ่มน้อยขึ้นมาทำให้ชุดข้อมูลมีความสำคัญมากขึ้น โดยมีเทคนิคการสร้างข้อความ 2 วิธี การสร้างข้อความด้วยห่วงโซ่มาร์คอฟ (Markov Chains) และการสร้างข้อความด้วยแบบจำลอง LSTM เพื่อเปรียบเทียบความสามารถในการปรับปรุงประสิทธิภาพของการจำแนกประเภทข้อความที่ไม่สมดุล เทคนิคการเพิ่มตัวอย่างแบบดั้งเดิมถูกใช้เป็นแนวทางเบสไลน์ (Baseline) งานวิจัยนี้ได้ศึกษาชุดข้อมูลภาษาไทยที่เกี่ยวกับโฆษณาใน Facebook จากเกิดเพิ่มขึ้นของค่ารีคอลย์ (Recall) การประยุกต์ใช้เหล่านี้แสดงให้เห็นถึงการปรับปรุงความสามารถในการสร้างแบบจำลองที่สามารถทำนายตัวอย่างบวกซึ่งเป็นตัวอย่างหมวดหมู่น้อยได้แม่นยำมากขึ้น ผลการวิจัยพบว่าห่วงโซ่มาร์คอฟ ให้ผลดีกว่าการเพิ่มตัวอย่างแบบดั้งเดิมและการสร้างข้อความโดยใช้แบบจำลอง LSTM [32]

2.14.2 การจำแนกประเภทด้วยแบบจำลอง LSTM GRU หรือ WangchanBERTa

ในปี ค.ศ. 2019 นักวิจัย Chayapol Piyaphakdeesakun และคณะ ได้ศึกษาเรื่อง การวิเคราะห์ความรู้สึกของความคิดเห็นภาษาไทยในเครือข่ายสังคมออนไลน์โดยใช้วิธีการเรียนรู้เชิงลึก คือ การวิเคราะห์ความรู้สึกเป็นหนึ่งในภารกิจของการประมวลผลภาษาธรรมชาติ ในการจำแนกประเภทข้อความความรู้สึกอารมณ์จากเอกสารที่กำหนด ในงานวิจัยนี้แก้ไขสองปัญหาคือ ปัญหาแรกแก้ไขปัญหาคำกำกวม หรือคำคลุมเครือ และปัญหาที่สองคือจำแนกประเภทความรู้สึกอารมณ์อัตโนมัติ งานวิจัยแสดงกระบวนการทำความสะอาดเอกสารออนไลน์ภาษาไทยเพื่อแก้ไขความไม่แน่นอนในภาษาไทย นอกจากนี้ยังศึกษาอัลกอรึทึมการเรียนรู้เชิงลึกที่เหมาะสมในการจำแนกประเภทความรู้สึกของเอกสารออนไลน์ภาษาไทย โดยเปรียบเทียบประสิทธิภาพหลายวิธีการ ผลลัพธ์แสดงให้เห็นว่าการใช้แบบจำลอง GRU ร่วมกับกลไกแอตเทนชัน (BGRU + ATTN) ให้ประสิทธิภาพที่ดีที่สุดในการจำแนกประเภทความรู้สึก (เมื่อมีการเปรียบเทียบกับ 11 แบบจำลอง Naive Bayes / CNN / CNN + ATTN / BLSTM / BLSTM + ATTN / CNN + BLSTM / CNN + BLSTM + ATTN / BGRU / CNN + BGRU / CNN + BGRU + ATTN) ATTN คือ แอตเทนชัน / CNN คือ Convolution Neural Network โครงข่ายประสาทเทียมที่ใช้งานเกี่ยวกับรูปภาพ / BLSTM คือ Bi-directional แบบจำลอง LSTM สองทิศทาง เช่นเดียวกับ GRU [33]

ในปี ค.ศ. 2022 นักวิจัย Pongsatorn Harnmetta และ Taweesak Samanchuen ได้ศึกษาเรื่อง การวิเคราะห์ความรู้สึกของบทวิจารณ์หุ้นโดยใช้แบบจำลอง Transformer คือตลาดหุ้นมักได้รับผลกระทบจากปัจจัยต่าง ๆ ในระยะยาว เช่น การเมือง เศรษฐกิจ การเงิน ซึ่งปัจจัยเหล่านี้สะท้อนออกมาผ่านทางสื่อออนไลน์ที่ผู้คนสามารถเข้าถึงได้ง่าย นอกจากนี้ข้อมูลมีแนวโน้มเติบโตแบบทวีคูณ ข้อมูลหลายล้านถูกสร้างขึ้นผ่านแพลตฟอร์มออนไลน์ต่าง ๆ บนอินเทอร์เน็ต เพื่อให้สามารถใช้ประโยชน์จากข้อมูลได้ จึงได้มีการนำเสนอระบบวิเคราะห์ความรู้สึกจากหุ้นผสานกับแบบจำลอง Transformer ในงานชิ้นนี้แบบจำลอง Transformer ได้ก้าวข้ามข้อจำกัดการประมวลผลภาษาธรรมชาติ นอกจากนี้ยังมีการรวบรวมข้อมูลการวิเคราะห์พื้นฐานในเนื้อหาการเงินไทยในสถาบันการเงิน อย่างไรก็ตามเมื่อเปรียบเทียบผลลัพธ์ระหว่างเทคนิคเอ็มแบ็ดดิ้งด้วยเบสไลน์ (Baseline) มีการใช้การถดถอยแบบมัลติโนเมียลลอจิสติกของแบบจำลองการทำนายและใช้เบสไลน์ คือ TF-IDF ผลการทดลองแสดงให้เห็นว่าแบบจำลอง WangchanBERTa และ BERT ได้ความแม่นยำสูงถึง 92.52 และ 89.12 เปอร์เซ็นต์ตามลำดับ ในขณะที่ค่าเบสไลน์คือ 85.03 เปอร์เซ็นต์ สรุปได้ว่าระบบที่นำเสนอสามารถทำนายความรู้สึกของหุ้นในภาษาไทยด้วยความแม่นยำสูง [34]

ในปี ค.ศ. 2023 นักวิจัย Nattawat Khamphakdee และ Pusadee Seresangtakul ได้ศึกษาเรื่อง การเรียนรู้เชิงลึกที่มีประสิทธิภาพสำหรับการวิเคราะห์ความรู้สึกในภาษาไทย คือจำนวนรีวิวจากลูกค้าในเว็บไซต์และแพลตฟอร์มการท่องเที่ยวกำลังเพิ่มขึ้นอย่างรวดเร็ว พวกเขาให้โอกาสผู้คนในการเขียนรีวิวเกี่ยวกับประสบการณ์ของตนเองในด้านคุณภาพการบริการ ที่ตั้ง ห้องพัก และความสะอาด ซึ่งช่วยให้ผู้อื่นตัดสินใจก่อนจองโรงแรม หลายคนไม่สามารถตัดสินใจเรื่องการจองที่พักเนื่องจากรีวิวมีจำนวนมากและต้องใช้เวลานานในการอ่าน และหลายรีวิวเป็นภาษาที่ไม่ใช่ภาษาแม่ ดังนั้น ธุรกิจโรงแรมจึงจำเป็นต้องมีกระบวนการที่มีประสิทธิภาพในการวิเคราะห์และจำแนกความโน้มเอียงของรีวิวว่าเป็นบวก ลบ หรือกลาง โดยเฉพาะอย่างยิ่งสำหรับภาษาที่มีทรัพยากรน้อย เช่น ภาษาไทย ซึ่งมีข้อจำกัดมากขึ้นในด้านทรัพยากรสำหรับการจำแนกความโน้มเอียงของความรู้สึก

ในงานวิจัยนี้ เสนอวิธีการวิเคราะห์ความรู้สึกสำหรับการจำแนกความรู้สึกภาษาไทยในธุรกิจโรงแรม ขั้นแรก ใช้เทคนิค Word2Vec การประมวลผลแบบ CBOW และ skip-gram ในการสร้างการแปลงคำเป็นเวกเตอร์ในมิติต่าง ๆ ขั้นต่อมา นำแบบจำลองการแปลงคำเป็นเวกเตอร์มารวมกับแบบจำลองการเรียนรู้เชิงลึก เพื่อสังเกตผลกระทบของมิติเวกเตอร์คำแต่ละค่า เปรียบเทียบประสิทธิภาพของแบบจำลองการเรียนรู้เชิงลึกเก้าแบบจำลอง ได้แก่ CNN, LSTM, Bi-LSTM, GRU, Bi-GRU, CNN-LSTM, CNN-BiLSTM, CNN-GRU และ CNN-BiGRU ที่มีจำนวนชั้นแตกต่างกันในการประเมินประสิทธิภาพการจำแนกความโน้มเอียง ชุดข้อมูลได้รับการจำแนกประเภทโดยใช้แบบจำลองการฝึกล่วงหน้า FastText และ BERT เพื่อดำเนินการจำแนกความโน้มเอียงของความรู้สึก

สุดท้าย ผลการทดลองแสดงให้เห็นว่าแบบจำลอง WangchanBERTa ปรับปรุงค่าความแม่นยำเล็กน้อยโดยมีค่าเท่ากับ 0.9225 และการผสานระหว่างการแทนคำแบบ skip-gram และโมเดล CNN ให้ผลดีกว่าแบบจำลองการเรียนรู้เชิงลึกอื่นๆ โดยมีค่าความแม่นยำ 0.9170 จากการทดลอง เราพบว่ามิติของเวกเตอร์คำ ค่าไฮเปอร์พารามิเตอร์ และจำนวนชั้นของแบบจำลองการเรียนรู้เชิงลึกมีผลต่อประสิทธิภาพของการจำแนกประเภทความรู้สึก งานวิจัยให้แนวทางในการตั้งค่าไฮเปอร์พารามิเตอร์ที่เหมาะสมเพื่อปรับปรุงความแม่นยำในการจำแนกประเภทความรู้สึกสำหรับภาษาไทยในธุรกิจโรงแรม [35]

ในปี ค.ศ. 2023 นักวิจัย Nichaphan Noiyoo และ Jessada Thutkawkornpin ได้ศึกษาเรื่อง การเปรียบเทียบอัลกอริทึมการเรียนรู้ของเครื่องและเครือข่ายประสาทเทียมสำหรับระบบตรวจสอบคุณภาพงานเรียงความภาษาไทยอัตโนมัติ คือ การตรวจสอบคุณภาพการเขียนเรียงความภาษาไทยยังคงเป็นงานที่ซับซ้อน เนื่องจากภาษาไทยมีความซับซ้อนในแง่ของเครื่องหมายวรรคตอน โครงสร้างประโยค คำซ้ำ การสะกด การแสดงความคิดเห็น และการใช้เหตุผลในเนื้อหา ดังนั้นการตรวจสอบคุณภาพของเรียงความและการให้คะแนนจึงต้องอาศัยทักษะของผู้ตรวจในการอ่านและตีความ ซึ่งส่งผลให้ใช้เวลานานในการตรวจ หากมีผู้ตรวจมากกว่าหนึ่งคนในกระบวนการตรวจสอบ อาจทำให้มีมาตรฐานการตรวจสอบคุณภาพที่แตกต่างกัน งานวิจัยนี้ได้รวบรวมเรียงความภาษาไทยที่เขียนโดยนักศึกษาที่ลงทะเบียนเรียนการเขียน จากสถาบันภาษาไทยสิรินธร มหาวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย งานวิจัยนี้ได้ใช้แบบจำลอง LSTM, CNN, BERT และ WangchanBERTa มาใช้เพื่อเปรียบเทียบประสิทธิภาพในการตรวจสอบคุณภาพของการเขียนเรียงความภาษาไทย ผลการทดลองแสดงให้เห็นว่าการวิเคราะห์การจำแนกประเภทร่วมกับแบบจำลอง WangchanBERTa สามารถให้ความแม่นยำสูงถึง 90% อย่างไรก็ตาม แบบจำลอง CNN ร่วมกับการวิเคราะห์การจำแนกประเภทสามารถให้ความแม่นยำสูงถึง 87% ในขณะที่ร่วมกับการวิเคราะห์การถดถอยสามารถให้ความแม่นยำสูงในช่วง 90% สรุปได้ว่าระบบที่นำเสนอสามารถพยากรณ์คุณภาพของเรียงความภาษาไทยได้อย่างแม่นยำสูง ดังนั้นจึงแนะนำให้ใช้แบบจำลอง Wangchanberta สำหรับปัญหาการจำแนกประเภท และใช้แบบจำลอง CNN สำหรับปัญหาการถดถอย [36]

บทที่ 3
วิธีการดำเนินงานวิจัย

ข้อมูลที่ได้มาจากสำนักงานพัฒนารัฐบาลดิจิทัล (องค์การมหาชน) (สพร.) หรือชื่อในภาษาอังกฤษคือ Digital Government Development Agency (Public Organization) (DGA) โดยข้อมูลที่ได้มาจากสำนักงานพัฒนารัฐบาลดิจิทัล (องค์การมหาชน) มีทั้งหมด 41 ประเภทงานของทั้ง 8 หน่วยงาน ประกอบไปด้วยหน่วยงาน กองช่าง กองสาธารณสุขฯ กองการศึกษาฯ กองคลัง กองสวัสดิการสังคม กองวิชาการและแผน สำนักปลัด และศูนย์รับเรื่องราวร้องทุกข์ ได้มีการจำแนกทั้งหน่วยงานและประเภทงานจากข้อมูลที่ได้มา แสดงดังตารางที่ 3.1

ตารางที่ 3.1 จำนวนข้อมูลจำแนกหน่วยงานและประเภทของงานแต่ละหน่วยงาน

ลำดับ	หน่วยงาน	ประเภทงาน	จำนวนแถว
1	กองช่าง	ไฟฟ้าสาธารณะ(ไฟกิ่ง)	2970
2	กองสาธารณสุขฯ	รับส่งผู้ป่วย	2592
3	กองสวัสดิการสังคม	ยืมวัสดุอุปกรณ์	1065
4	สำนักปลัด	สนับสนุนน้ำอุปโภค บริโภค	535
5	กองช่าง	ถนน/ไหล่ทาง	427
6	กองสาธารณสุขฯ	จัดเก็บกิ่งไม้ใบไม้	367
7	กองสาธารณสุขฯ	จัดเก็บขยะมูลฝอย	348
8	กองช่าง	คำร้องทั่วไปกองช่าง	337
9	กองสาธารณสุขฯ	เหตุเดือดร้อนรำคาญ	261
10	สำนักปลัด	ป้องกันและบรรเทาสาธารณภัย	257
11	กองสาธารณสุขฯ	ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข	241
12	สำนักปลัด	คำร้องทั่วไปสำนักปลัด	227
13	กองช่าง	ท่อระบายน้ำ	193
14	กองสาธารณสุขฯ	หนังสือรับรองการแจ้งสะสมอาหาร หรือสถานที	192
15	กองช่าง	รางระบายน้ำ	165

ตารางที่ 3.1 (ต่อ) จำนวนข้อมูลจำแนกหน่วยงานและประเภทของงานแต่ละหน่วยงาน

ลำดับ	หน่วยงาน	ประเภทงาน	จำนวนแถว
16	กองสาธารณสุขฯ	ตัดต้นไม้	164
17	กองสาธารณสุขฯ	ตัดหญ้าที่สาธารณะ	150
18	กองช่าง	ขออนุญาตก่อสร้างอาคารประเภทบ้านพักอาศัย	138
19	กองสาธารณสุขฯ	คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม	134
20	กองสาธารณสุขฯ	การป้องกันและควบคุมโรค	84
21	กองสวัสดิการสังคม	คำร้องทั่วไปกองสวัสดิการสังคม	74
22	ศูนย์รับเรื่องราวร้องทุกข์	เรื่องร้องเรียน/ร้องทุกข์	72
23	กองการศึกษาฯ	จองสนามฟุตบอลหญ้าเทียม	46
24	สำนักปลัด	ขออนุญาตติดตั้งป้าย	43
25	กองสาธารณสุขฯ	สัตว์เลี้ยงไร้เจ้าของ	36
26	กองช่าง	ขออนุญาตก่อสร้างอาคาร ประเภทควบคุมการใช้	29
27	กองสาธารณสุขฯ	ใบอนุญาตสะสมอาหารหรือสถานที่จำหน่ายอาหาร	26
28	กองวิชาการและแผน	เครื่องเสียงประกาศตามสาย	24
29	กองช่าง	ขออนุญาตใช้น้ำบาดาล	22
30	กองสาธารณสุขฯ	ขอต่อใบอนุญาตการเก็บ ขนสิ่งปฏิกูลหรือขยะ	19
31	สำนักปลัด	ยืมวัสดุอุปกรณ์	19
32	กองคลัง	คำร้องทั่วไปกองคลัง	10
33	กองช่าง	สะพาน	9
34	กองการศึกษาฯ	คำร้องทั่วไปกองการศึกษาฯ	6
35	กองวิชาการและแผน	คำร้องทั่วไปกองวิชาการและแผน	6
36	กองวิชาการและแผน	ขุดลอกลำเหมือง	5
37	กองสาธารณสุขฯ	ใบประกอบกิจการตลาด	4
38	กองการศึกษาฯ	รับสมัครนักเรียน	3
39	กองการศึกษาฯ	ขอใช้ลานกีฬา/โรงยิม	2
40	กองสาธารณสุขฯ	การขอใช้บริการรถสุขาเคลื่อนที่	2
41	สำนักปลัด	ยืมวัสดุอุปกรณ์จราจร	2

3.1 การสำรวจข้อมูลเบื้องต้น

ข้อมูลที่ได้มานั้นบางประเภทงานของแต่ละหน่วยงานมีข้อมูลที่น้อยเกินไปที่ให้แบบจำลองเรียนรู้ได้ จึงจำเป็นต้องตัดข้อมูลบางส่วนออกไปโดยให้เกณฑ์การตัดที่มีจำนวนข้อมูลน้อยกว่า 50 แถว เป็นเส้นสีแดงระหว่างลำดับที่ 22 และ 23 แสดงดังตารางที่ 3.1 หมายความว่าจาก 41 ประเภทงาน 11,306 แถว ที่ต้องฝึกสอน ตรวจสอบ และทำนาย เหลือ 22 ประเภทงานของทั้ง 5 หน่วยงาน 10,993 แถว งานที่ต้องฝึกสอน ตรวจสอบ และทำนาย ข้อมูลที่หายไปคิดเป็น 2.77 เปอร์เซ็นต์ หรือ 313 แถว จากข้อมูลทั้งหมด

3.2 การเตรียมสิ่งแวดล้อม (Setup Environment) และการติดตั้งไลบรารี่ (Library)

ถัดมาได้ทำการเตรียมสิ่งแวดล้อมของโปรแกรมในเครื่อง Macbookก่อนตามแหล่งอ้างอิง ซึ่งเป็นหนึ่งในหลาย ๆ วิธี [37] แต่ในอนาคตอาจมีการเปลี่ยนแปลงเวอร์ชันและวิธีการติดตั้งโปรแกรมในสิ่งแวดล้อมสำหรับการพัฒนาแบบเบ็ดเสร็จ (Integrated Development Environment: IDE) คือ Jupyter Notebook

มีการติดตั้งไลบรารี่ Json, Pandas, Numpy, Re, Tensorflow, Tensorflow_addons, Pythainlp, Sklearn, Time, Datetime, Pickle, Plotly, Seaborn และ Matplotlib ซึ่งบางไลบรารี่ถูกนำไปใช้ในแบบจำลองถัดมาด้วย เช่น Sklearn Pickle Json Numpy และ Pandas

1) Numpy เพื่อช่วยในการคำนวณตัวเลขเมทริกซ์

2) Json และ Pandas เพื่ออ่านข้อมูลมาจากนามสกุล .json และจัดการข้อมูลแบบตาราง

3) Tensorflow_Addons เพื่อติดตามการฝึกสอนจากแถบแสดงความคืบหน้า

4) Time และ Datetime เพื่อบอกระยะเวลาในการฝึกสอน

5) Pickle เพื่อบันทึกค่าน้ำหนักของแบบจำลองที่ดีที่สุด และค่าตัวแปรที่จำเป็นต่อการนำแบบจำลองไปใช้งาน (Deployment)

6) Plotly Seaborn และ Matplotlib เพื่อแสดงรูปและกราฟต่าง ๆ

7) PythaiNLP เพื่อจัดการภาษาไทย เช่น คลังคำศัพท์ไทย คลังคำศัพท์ฟุ่มเฟือย และวิธีการตัดคำแบบ newmm

8) Tensorflow เพื่อจัดการแบบจำลอง LSTM เช่น การเตรียมรูปแบบข้อมูลให้สามารถเข้าไปฝึกสอนได้ หรือการประมวลผลข้อมูลล่วงหน้า (Data Preprocessing) และการจัดการสถาปัตยกรรมแบบจำลองต่าง ๆ

9) Sklearn เพื่อจัดการแยกชุดข้อมูลออกเป็นทั้ง 3 ชุด และสร้างรายงานผลค่าความแม่นยำ และค่าเฉลี่ยมาโคร

3.3 การทำความสะอาดข้อมูลในแบบจำลอง LSTM

จากนั้นนำข้อมูลมาทำความสะอาดก่อนนำเข้าแบบจำลอง LSTM หลังจากที่ลบประเภทงานจนเหลือ 22 งานแล้ว ถัดมาได้นำหัวข้อร้องเรียนกับรายละเอียดมารวมกันโดยมีวิธีการดังนี้

1) การกำหนดรูปแบบตัวอักษรด้วย นิพจน์ทั่วไป (Regular Expression) เป็นรูปแบบ [^ก-๙a-zA-Z0-9.] หมายความว่าในวงเล็บเหลี่ยม ([ ]) หลังแคเรต (^) คือการเก็บตัวอักษรที่ระบุไว้ส่วนที่ไม่ได้ระบุไว้ทำการลบออกไปจากข้อความในที่นี้คือ เก็บภาษาไทย เลขไทย ภาษาอังกฤษตัวใหญ่ ภาษาอังกฤษตัวเล็ก เลขอารบิก และจุด

2) นำคำไทยจากคลังข้อมูลของ PyThaiNLP และเพิ่มคำกำหนดเองเพิ่มเติมเพื่อให้ตรงกับข้อมูลที่แบบจำลองเรียนรู้ เช่น วันจันทร์ ถึง วันอาทิตย์ เดือนมกราคม ถึง เดือนธันวาคม อ. ต. จ. ชื่อเขต/อำเภอ และแขวง/ตำบล เป็นต้น

3) ลบช่องว่างหน้าและหลังข้อความ ทำให้ภาษาอังกฤษเป็นตัวเล็กทั้งหมด

4) ตัดคำแบบ newmm คือการตัดคำแบบสอดคล้องมากที่สุด

5) นำคลังข้อมูลคำฟุ่มเฟือย (Stopword) ภาษาไทยของ PyThaiNLP และเพิ่มคำฟุ่มเฟือยที่กำหนดเองเข้าไป เช่น ลบวันจันทร์ ถึงวันอาทิตย์ เดือนมกราคม ถึง เดือนธันวาคม เป็นต้น และทำการลบทิ้ง

6) สุดท้ายลบคำที่มีระดับอักขระ หรือสระต่าง ๆ ที่น้อยกว่า 3 ออก เพื่อทำให้ข้อมูลสะอาดขึ้นก่อนนำไปใช้งานในแบบจำลอง LSTM แสดงดังรูปที่ 3.1

```
df2['subject_detail'] = df2['subject'] + '_' + df2['detail']

df2['claen_sd'] = df2['subject_detail'].str.replace(r'[^ก-๙a-zA-Z0-9.]', "", regex=True)

from pythainlp.corpus.common import thai_words

custom_words_list = set(thai_words())

with open(THAI_CUSTOM_WORDS, encoding='utf8') as f:

    words = [line.strip() for line in f]

custom_words_list.update(words)

df2['claen_sd'] = df2['claen_sd'].str.strip()

df2['claen_sd'] = df2['claen_sd'].fillna('').apply(lambda x: x.lower())

df2['claen_sd'] = df2['claen_sd'].apply(lambda x: custom_tokenizer.word_tokenize(x))

from pythainlp.corpus import thai_stopwords

stopwords = thai_stopwords()

with open(THAI_STOP_WORDS, encoding='utf8') as f:

    new_stopwords_list = [line.strip() for line in f]

stopwords = stopwords.union(new_stopwords_list)

df2['claen_sd'] = df2['claen_sd'].apply(lambda x: [item for item in x if item not in stopwords])

df2['claen_sd'] = df2['claen_sd'].apply(lambda x: [item for item in x if len(item) > 3])
```

รูปที่ 3.1 การทำความสะอาดชุดข้อมูลก่อนเข้าแบบจำลอง LSTM

3.4 การจัดการข้อมูลที่ไม่สมดุลของแต่ละคลาส และแยกชุดข้อมูลออกเป็น 3 ชุด ของแบบจำลอง LSTM

วิธีการจัดการความไม่สมดุลของคลาสในแต่ละประเภทงาน คือต้องทำให้น้ำหนักของข้อมูลเท่ากัน โดยการกระทำนี้ทำให้แบบจำลองไม่ได้ไปทำนายประเภทงานที่มีจำนวนแถวมากเพียงอย่างเดียวและละเลยการทำนายจำนวนแถวน้อย โดยวิธีการทำให้น้ำหนักของข้อมูลเท่ากันมีการทดลอง 2 ประเภท ได้แก่ 1.วิธีสุ่มตัวอย่างใหม่ (Resampling) 2.วิธีน้ำหนักตัวอย่าง (Sample Weights)

1) วิธีสุ่มตัวอย่างใหม่ (Resampling) คือแถวแต่ละหมวดหมู่มีจำนวนข้อมูลเท่ากัน โดยข้อมูลมีการหาค่าเฉลี่ยจากทุกหมวดหมู่และใช้ค่าเฉลี่ยนั้นในการกำหนดตัวเลข และหมวดหมู่ใดมีจำนวนน้อยกว่าค่าเฉลี่ยมีการทำสำเนาซ้ำแถวเดิม และหมวดหมู่ใดมีจำนวนมากกว่าค่าเฉลี่ยให้เลือกจำนวนแถวเท่ากับค่าเฉลี่ยแล้วตัดข้อมูลที่เหลือทิ้ง

2) วิธีน้ำหนักตัวอย่าง (Sample Weights) คือแถวแต่ละหมวดหมู่มีข้อมูลไม่เท่ากัน แต่ให้น้ำหนักแต่ละหมวดหมู่ไม่เท่ากัน โดยปกติบางหมวดหมู่มีจำนวนข้อมูลมากทำให้แบบจำลองให้ความสำคัญมาก และจำนวนข้อมูลน้อยทำให้แบบจำลองไม่ให้ความสำคัญ แต่วิธีน้ำหนักตัวอย่างให้ความสำคัญกับหมวดหมู่ที่มีจำนวนข้อมูลมากลดลง และให้น้ำหนักความสำคัญกับหมวดหมู่ที่มีจำนวนน้อยให้มีความสำคัญมากขึ้น

สุดท้ายได้ฝึกสอนแบบจำลอง LSTM ทั้ง 3 แบบจำลอง แสดงดังตารางที่ 3.2

ตารางที่ 3.2 วิธีการจัดการข้อมูลแต่ละแบบของแบบจำลอง LSTM

แบบจำลอง	การ์ดจอ	วิธีจัดการข้อมูล	เวลา
LSTM Unbalance	M2	-	15 นาที
LSTM balance by sample	M2	สุ่มตัวอย่างใหม่ (Resampling)	15 นาที
LSTM balance by class weight	M2	น้ำหนักตัวอย่าง (Sample Weights)	14 นาที

แบบจำลองแรกคือแบบจำลอง LSTM ที่ไม่ได้มีการจัดการความไม่สมดุลของคลาสจึงทำให้มีข้อมูลแต่ละหมวดหมู่ไม่เท่ากันส่วนอีกสองแบบจำลองมีการจัดการความไม่สมดุลของคลาสด้วยวิธีสุ่มตัวอย่างใหม่ และวิธีน้ำหนักตัวอย่าง

1) LSTM Unbalance คือมีจำนวนแถวไม่เท่ากันมี และโครงสร้างแบบจำลองของทั้ง 3 มีลักษณะเหมือนกัน แสดงดังรูปที่ 3.2

ข้อมูลทั้งหมด 10,993 แถว แบ่งข้อมูลเป็น 3 ชุด ได้แก่ ชุดฝึกสอน (Trian) 80 เปอร์เซ็นต์ หรือ 8,794 แถว ชุดตรวจสอบ (Validation) 10 เปอร์เซ็นต์ หรือ 1,099 แถว และชุดทดสอบ (Test) 10 เปอร์เซ็นต์ หรือ 1,099 แถว และได้ทำการสุ่มแบ่งชั้น (Stratified Sampling) คือแบ่งให้มีทุกหมวดหมู่ครบในแต่ละชุดข้อมูลแต่ลดจำนวนข้อมูลลงตามเปอร์เซ็นต์ที่มีการแบ่งไป แสดงดังรูปที่ 3.3 และจำนวนพารามิเตอร์ทั้งหมดประมาณ 1,497,000 ตัว แสดงดังตารางที่ 3.3

```python
def create_model(vocab_size, max_length):
  model = Sequential()

  model.add(Embedding(vocab_size, 128, input_length = max_length,  trainable = True))
  model.add(Bidirectional(LSTM(128),merge_mode="concat"))
  model.add(Dense(128, activation = "relu"))
  model.add(Dropout(0.5))
  model.add(Dense(64, activation = "relu"))
  model.add(Dropout(0.5))
  model.add(BatchNormalization())
  model.add(Dense(num_class, activation = "softmax"))

  return model
```

รูปที่ 3.2 โครงสร้างแบบจำลอง LSTM ทั้ง 3 แบบ

```python
from sklearn.model_selection import train_test_split
```

```python
x_train,x_val_test,y_train,y_val_test = train_test_split(padded_doc,output_one_hot,test_size = 0.2,random_state=12,stratify = output_one_hot)
```

```python
x_val, x_test, y_val, y_test = train_test_split(x_val_test, y_val_test, test_size = 0.5, random_state=12, stratify = y_val_test)
```

```python
x_train.shape, x_val.shape, x_test.shape, y_train.shape, y_val.shape, y_test.shape
```

```
((8794, 124), (1099, 124), (1100, 124), (8794, 22), (1099, 22), (1100, 22))
```

รูปที่ 3.3 การแบ่งข้อมูลเป็น 3 ชุด ฝึกสอน : ตรวจสอบ : ทดสอบ (80:10:10 เปอร์เซ็นต์)

ตารางที่ 3.3 โครงสร้างแบบจำลอง LSTM และจำนวนพารามิเตอร์ทั้งหมด

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 124, 128)          1191552

 bidirectional (Bidirectiona  (None, 256)              263168
 l)

 dense (Dense)               (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_2 (Dense)             (None, 22)                1430

=================================================================
Total params: 1,497,558
Trainable params: 1,497,430
Non-trainable params: 128
_________________________________________________________________
```

2) LSTM balance by sample คือ การสุ่มข้อมูลให้ถึงเกณฑ์ที่กำหนด โดยมีการปรับแต่งเหมือนกับ LSTM Unbalance

โดยใช้เกณฑ์ค่าเฉลี่ยของทุกหมวดหมู่ คือ 10,993/22 = 500 แถว หมวดหมู่ใดมีจำนวนแถวน้อยจึงทำการสุ่มทำสำเนาซ้ำแถวเดิมจนครบ 500 แถว และหมวดหมู่ใดมีแถวเกิน 500 แถว จึงสุ่มเลือกแถวมาแค่ 500 แถว จากข้อมูลทั้งหมด แสดงดังรูปที่ 3.4

การแบ่งข้อมูลเป็น 3 ชุด (80:10:10) และโครงสร้างแบบจำลอง เหมือนกับแบบจำลอง LSTM Unbalance จำนวนพารามิเตอร์ทั้งหมด 1,160,000 ตัว

```python
df2 = pd.concat([
    df2[df2['type_name']=='สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)'].sample(n=500,replace=True),
    df2[df2['type_name']=='สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_รางระบายน้ำ'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_ถนน/ไหล่ทาง'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_การป้องกันและควบคุมโรค'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_เหตุเดือดร้อนรำคาญ'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_ท่อระบายน้ำ'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_รับส่งผู้ป่วย'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้'].sample(n=500,replace=True),
    df2[df2['type_name']=='สำนักปลัด_คำร้องทั่วไปสำนักปลัด'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองช่าง_คำร้องทั่วไปกองช่าง'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_ตัดต้นไม้'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_หรือสถานที่'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม'].sample(n=500,replace=True),
    df2[df2['type_name']=='กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์'].sample(n=500,replace=True),
    df2[df2['type_name']=='ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์'].sample(n=500,replace=True),
],axis=0) # ต่อท้าย
```

รูปที่ 3.4 การทำวิธีสุ่มตัวอย่างใหม่ (Resampling)

3) LSTM balance by class weight คือ การใช้วิธีน้ำหนักตัวอย่าง (Sample Weights) โดยให้ทุกประเภทงานนั้นมีน้ำหนักเท่ากัน โดยมีการปรับแต่งเหมือนกับ LSTM Unbalance

```python
class_weight = compute_class_weight(
                                    class_weight = 'balanced',
                                    classes = unique_cat,
                                    y = df2['type_name']
                                    )
```

```
{0: 6.752457002457002,
 1: 3.7289687924016284,
 2: 5.948593073593074,
 3: 2.2012414897877455,
 4: 0.19277847923681257,
 5: 2.589024964672633,
 6: 0.9339847068819032,
 7: 2.0733685401735196,
 8: 3.028374655647383,
 9: 1.3615308397324746,
 10: 1.482735365524683,
 11: 1.9442872302794483,
 12: 1.1702150308707686,
 13: 2.6025094696969697,
 14: 1.9144897248345525,
 15: 0.4691848058045241,
 16: 3.331212121212121,
 17: 0.1682430364248546,
 18: 1.4358672936259143,
 19: 6.940025252525253,
 20: 3.620882740447958,
 21: 3.046840354767184}
```

```
{'กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม': 1,
 'กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม': 2,
 'กองสาธารณสุขฯ_การป้องกันและควบคุมโรค': 3,
 'สำนักปลัด_คำร้องทั่วไปสำนักปลัด': 4,
 'กองสาธารณสุขฯ_รับส่งผู้ป่วย': 5,
 'กองช่าง_ท่อระบายน้ำ': 6,
 'สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค': 7,
 'กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข': 8,
 'กองช่าง_รางระบายน้ำ': 9,
 'กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้': 10,
 'กองช่าง_คำร้องทั่วไปกองช่าง': 11,
 'สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย': 12,
 'กองช่าง_ถนน/ไหล่ทาง': 13,
 'กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_หรือสถานที่': 14,
 'กองสาธารณสุขฯ_เหตุเดือดร้อนรำคาญ': 15,
 'กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์': 16,
 'กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ': 17,
 'กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)': 18,
 'กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย': 19,
 'ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์': 20,
 'กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย': 21,
 'กองสาธารณสุขฯ_ตัดต้นไม้': 22}
```

รูปที่ 3.5 การทำวิธีน้ำหนักตัวอย่าง (Sample Weights)

จากรูปที่ 3.5 สังเกตว่า กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) ให้น้ำหนัก 0.1682 ซึ่งเป็นหมวดหมู่หรือประเภทงานที่มีจำนวนแถวสูงที่สุดเท่ากับ 2,970 แถว หลังจากมีการแบ่งข้อมูลฝึกสอนเป็น 80 เปอร์เซ็นต์ มีข้อมูล 2,376 แถว และเมื่อทำการเทียบกับ ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์ ที่มีข้อมูลน้อยสุดเท่ากับ 72 แถว หลังจากมีการแบ่งข้อมูลฝึกสอนมีข้อมูล 57 แถว มีการให้น้ำหนัก 6.940

3.5 การปรับแต่ง (Fine-Tuning) พารามิเตอร์ของแบบจำลอง LSTM

การปรับแต่งของแบบจำลอง LSTM ทั้ง 3 แบบจำลอง แสดงดังตารางที่ 3.4

ตารางที่ 3.4 การปรับแต่งพารามิเตอร์ของแบบจำลอง LSTM ทั้ง 3 แบบจำลอง

พารามิเตอร์	การปรับแต่ง	ผลลัพธ์การปรับแต่ง
จำนวนรอบ (Epoch)	40	40
การสุ่มลบ (Dropout)	0.4 – 0.6	0.5
ขนาดแบตช์ (Batch Size)	8, 16, 32	32
ออปติไมเซอร์ (Optimizer)	Adam	Adam
อัตราการเรียนรู้ (Learning Rate)	10^{-7} - 10^{-4}	10^{-4}, 10^{-7}

หมายเหตุ อัตราการเรียนรู้เริ่มต้นที่ 10^{-4} และถ้า validation loss ไม่ลดใน 3 อีพ็อค อัตราการเรียนรู้ลดเป็น 10^{-5} และอัตราการเรียนรู้ลดลงไปจนหยุดที่ 10^{-7} หรือหยุดการฝึกสอนที่ 40 อีพ็อคก่อน

ถัดไปเป็นการทำแบบจำลอง WangchanBERTa บน Google Colaboratory ซึ่งมีความสะดวกเนื่องจากไม่ต้องเตรียมสิ่งแวดล้อม (Setup Environment) และสามารถติดตั้งไลบรารี่ โปรแกรมสำเร็จรูปต่าง ๆ ใน IDE ของ Google Colaboratory ได้ นอกจากนี้ยังสามารถจ่ายเงินเพื่อเพิ่มประสิทธิภาพของการ์ดจอเพื่อให้สามารถฝึกสอนแบบจำลองได้มีประสิทธิภาพมากขึ้น

3.6 การติดตั้งไลบรารี่ (Library) ของแบบจำลอง WangchanBERTa

ติดตั้งไลบรารี่ ตัวแรกคือ datasets เพื่อใช้จัดการชุดข้อมูล (Dataset) ประเภท DatasetDict ถัดมาติดตั้ง accelerate เพื่อใช้ในการเพิ่มประสิทธิภาพในการใช้หลายการ์ดจอ (GPU) ติดตั้ง transformers [sentencepiece] เป็นเครื่องมือสำหรับแปลข้อความเป็นส่วนคำย่อย ติดตั้ง pythainlp emoji เพื่อใช้สำหรับจัดการตัดคำ และการจัดการ emoji ของภาษาไทย แสดงดังรูปที่ 3.6

```
!pip install -q datasets
!pip install -q accelerate -U
!pip install -q transformers[sentencepiece]
!pip install -q pythainlp emoji
```

รูปที่ 3.6 ติดตั้งไลบรารี่ (library) โปรแกรมสำเร็จรูป

3.7 แยกชุดข้อมูลออกเป็น 3 ชุด ของแบบจำลอง WangchanBERTa

แบบจำลอง WangchanBERTa มีการสำรวจข้อมูลเบื้องต้นในหัวข้อ 3.1 เหมือนกับแบบจำลอง LSTM จึงทำให้มีข้อมูลทั้งหมด 10,993 แถว แบ่งข้อมูลเป็น 3 ชุด ได้แก่ ชุดฝึกสอน 80 เปอร์เซ็นต์ หรือ 8,794 แถว ชุดตรวจสอบ 10 เปอร์เซ็นต์ หรือ 1,100 แถว และชุดทดสอบ 10 เปอร์เซ็นต์ หรือ 1,099 แถว และได้ทำการสุ่มแบ่งชั้น คือแบ่งให้มีทุกหมวดหมู่ครบในแต่ละชุดข้อมูลแต่ลดจำนวนข้อมูลลงตามเปอร์เซ็นต์ที่มีการแบ่งไป และมีการเปลี่ยนชุดข้อมูลจาก DataFrames เป็น DatasetDict

3.8 การทำความสะอาดข้อมูล และการประมวลผลข้อมูลล่วงหน้า (Data Preprocessing) ในแบบจำลอง WangchanBERTa

โหลดข้อมูลไฟล์ นามสกุล .py จากคำสั่ง !wget และเรียกใช้ซึ่งเป็นไลบรารี่เกี่ยวกับวิธีการทำความสะอาดข้อความซึ่งเหมือนกับการทำความสะอาดที่นักวิจัยที่คิดค้นแบบจำลอง WangchanBERTa ใช้ในการทำความสะอาดก่อนนำข้อมูลไปประมวลผลข้อมูลล่วงหน้า เนื่องจากต้องการให้ข้อมูลมีการกระจายแบบละเอียดเพื่อให้ตรงกับข้อความการฝึกล่วงหน้ามากที่สุด โดยมีการทำความสะอาด [38] แสดงดังรูปที่ 3.7 และประกอบไปด้วยขั้นตอนดังนี้

1) แปลง HTML tag และตัวอักษรพิเศษ เช่น \n, nbsp; เป็นช่องว่าง

2) ลบวงเล็บเปล่า เช่น (), {}, [] ที่มักพบได้บ่อยในข้อมูล Wikipedia และเว็บอื่น ๆ

3) ลบตัวอักษรที่ซ้ำกันเกิน 3 ตัว เช่น อ้ยน้อนนนน เป็น อ้ยน้อน

4) ลบคำที่ซ้ำ เช่น เต็มไปหมดเลยอะอะอะอะอะ เป็น เต็มไปหมดเลยอะ (คำตัดโดย newmm ของ PyThaiNLP)

5) ตัดและแยกประโยคที่ยาวเกิน 300 คำ (คำตัดโดย newmm ของ PyThaiNLP) เพื่อไม่ให้ใหญ่เกินไปสำหรับแบบจำลองภาษา (Language Model)

6) ลบข้อความที่ซ้ำออกทั้งหมด (Deduplication)

```
!wget https://raw.githubusercontent.com/vistec-
AI/thai2transformers/master/thai2transformers/preprocess.py

from preprocess import process_transformers
```

รูปที่ 3.7 ฟังก์ชันการทำความสะอาดชุดข้อมูลก่อนเข้าแบบจำลอง

ฟังก์ชันการประมวลผลข้อมูลล่วงหน้าขออธิบายเป็นทั้งหมด 4 ส่วน ได้แก่

1) ขั้นตอนการสร้างโทเค็นไนเซอร์ (Tokenizer) คือตัวตัดคำแบบ ชิ้นส่วนประโยค (SentencePiece) จากแบบจำลอง wangchanberta-base-att-spm-uncased และความยาวสูงสุดของแบบจำลองกำหนดไว้ที่ 510 คำ การตัดเป็นท่อน (truncation) = true คือการตัดข้อความที่ยาวเกิน 510 คำที่กำหนดไว้ use_fast = True เป็นการทำให้โทเค็นไนเซอร์ หรือตัวตัดคำมีประสิทธิภาพเพิ่มมากขึ้น padding = 'max_length' คือการเติมข้อความที่สั้นกว่าที่แบบจำลองกำหนดไว้ให้เต็มคือ 510 คำ แต่ให้สนใจส่วนที่มีแต่ข้อความเท่านั้น

2) การสร้างฟังก์ชันตัวเข้ารหัส (Encode) โดยนำ โทเค็นไนเซอร์ที่สร้างมาแล้วมาตัดคำแบบชิ้นส่วนประโยคแล้วทำการเข้ารหัสข้อมูลของคำที่ตัดไว้ หรือการเปลี่ยนชิ้นส่วนประโยค ให้กลายเป็นตัวเลข ซึ่งเรียกว่าเวกเตอร์นี้ว่า input_ids และสร้างอีกเวกเตอร์ที่สองขึ้นมาคือ attention_mask เป็นการระบุเลข 0 กับ 1 เพื่อบ่งบอกว่าเลข 1 คือข้อความที่สนใจที่ถูกเปลี่ยนจากคำมาเป็นตัวเลข ส่วนเลข 0 คือข้อความที่มีการ padding เพิ่มเข้ามาและไม่ให้แบบจำลองสนใจ ซึ่งทั้งเทนเซอร์ (Tensor) ทั้งสองนั้นคือ input_ids และ attention_mask ต่างมีขนาดที่กำหนดไว้เท่ากันคือ 510 ตัวเลข

3) การสร้างฟังก์ชันของข้อความที่มีการทำความสะอาดข้อความ เพื่อให้ตรงกับข้อความการฝึกล่วงหน้ามากที่สุด โดยมี 6 ข้อที่กล่าวมาแล้วข้างต้น

4) เริ่มจากนำชุดข้อมูลมาทำการทำความสะอาดข้อความในหัวข้อ 3) ถัดมาคือขั้นตอนประมวลผลข้อมูลล่วงหน้า (Data Preprocessing) แล้วนำตัวตัดคำ โทเค็นไนเซอร์ที่สร้างในหัวข้อ 1) มาเข้ารหัสในหัวข้อ 2) และสุดท้ายเป็นการจัดรูปแบบข้อมูลให้ตรงกับ PyTorch ในรูปแบบที่เหมาะสมก่อนนำไปใช้ในแบบจำลอง

ซึ่งตัวแบบจำลอง wangchanberta-base-att-spm-uncased มีจำนวนพารามิเตอร์ทั้งหมด 105,261,334 ตัว และกระบวนการทั้ง 4 ส่วน แสดงดังรูปที่ 3.8

```
# create tokenizer

tokenizer = AutoTokenizer.from_pretrained(args.model_name,
model_max_length=args.max_length, truncation=True,use_fast=False,
padding='max_length')


# encode dataset

def encode_function(examples):

    return tokenizer(examples[args.feature_col],
max_length=args.max_length, truncation=True, padding='max_length')


# preprocess text

def preprocess_example(examples):

    examples[args.feature_col] =
process_transformers(examples[args.feature_col])

    return examples


processed_dataset = dataset_dict.map(preprocess_example)

encoded_dataset = processed_dataset.map(encode_function,
batched=True)

encoded_dataset.set_format('torch', columns=['input_ids',
'attention_mask', 'label'])
```

รูปที่ 3.8 การประมวลผลล่วงหน้าและการเข้ารหัสข้อความของชุดข้อมูลก่อนเข้าแบบจำลอง

3.9 การปรับแต่ง (Fine-Tuning) พารามิเตอร์ของแบบจำลอง WangchanBERTa

ใช้การ์ดจอ V100 ซึ่งต้องเสียค่าใช้จ่ายให้ทางบริษัท Google เพื่อปรับระดับเป็นผู้ใช้งาน Google Colaboratory Pro ทำให้ใช้เวลาฝึกสอน 22 นาที แต่ถ้าใช้การ์ดจอ Tesla T4 ซึ่งเป็นการ์ดจอในระดับผู้ใช้งานเริ่มต้นและไม่เสียค่าใช้จ่าย ทำให้ใช้เวลาฝึกสอน 1 ชั่วโมง 20 นาที มีการกำหนดพารามิเตอร์ต่าง ๆ แสดงดังตารางที่ 3.5

ตารางที่ 3.5 การปรับแต่งพารามิเตอร์ของแบบจำลอง WangchanBERTa

พารามิเตอร์	การปรับแต่ง	ผลลัพธ์การปรับแต่ง
จำนวนรอบ (Epoch)	3 - 5	5
ขนาดแบตช์ (Batch Size)	8, 16	16
Warmup Percent	0.1	0.1
Weight Decay	0.01	0.01
Max Length	510	510
อัตราการเรียนรู้ (Learning Rate)	$2x10^{-5}$, $3x10^{-5}$, $5x10^{-5}$	$3x10^{-5}$

บทที่ 4
ผลการวิจัยและการอภิปรายผล

เนื่องจากข้อมูลที่ได้มาจากสำนักงานพัฒนารัฐบาลดิจิทัล (องค์การมหาชน) (สพร.) มีข้อจำกัดด้านจำนวนข้อมูลต่อแถวน้อยเกินกว่าที่ให้แบบจำลองเรียนรู้หรือฝึกสอนได้ในบางหมวดหมู่ จึงทำให้มีการตัดข้อมูลบางส่วน จาก 41 ประเภทงานเหลือ 22 ประเภทงาน โดยตั้งเกณฑ์จำนวนข้อมูลไว้ที่จำนวน 50 แถว ขึ้นไปในการให้แบบจำลองเรียนรู้ ซึ่งข้อมูลที่ลบไปเป็นจำนวน 313 แถว ข้อมูลที่ได้ทำการตัดออกถือว่าเป็นจำนวนน้อยมากเมื่อเทียบกับจำนวนข้อมูลทั้งหมด 97.23 เปอร์เซ็นต์ โดยข้อมูลที่เหลืออยู่สามารถทำนายประเภทงานที่มีการร้องเรียนส่วนใหญ่ได้ครอบคลุมในเนื้องาน โดยมีการเปรียบเทียบแบบจำลอง LSTM 3 แบบจำลอง และ WangchanBERTa 1 แบบจำลอง ว่าแบบจำลองใดให้ผลลัพธ์ที่ดีกว่าในการทำนายหมวดหมู่ประเภทงานทั้ง 22 ประเภทงาน

4.1 ผลการวิจัย และการประเมินผลแบบจำลอง

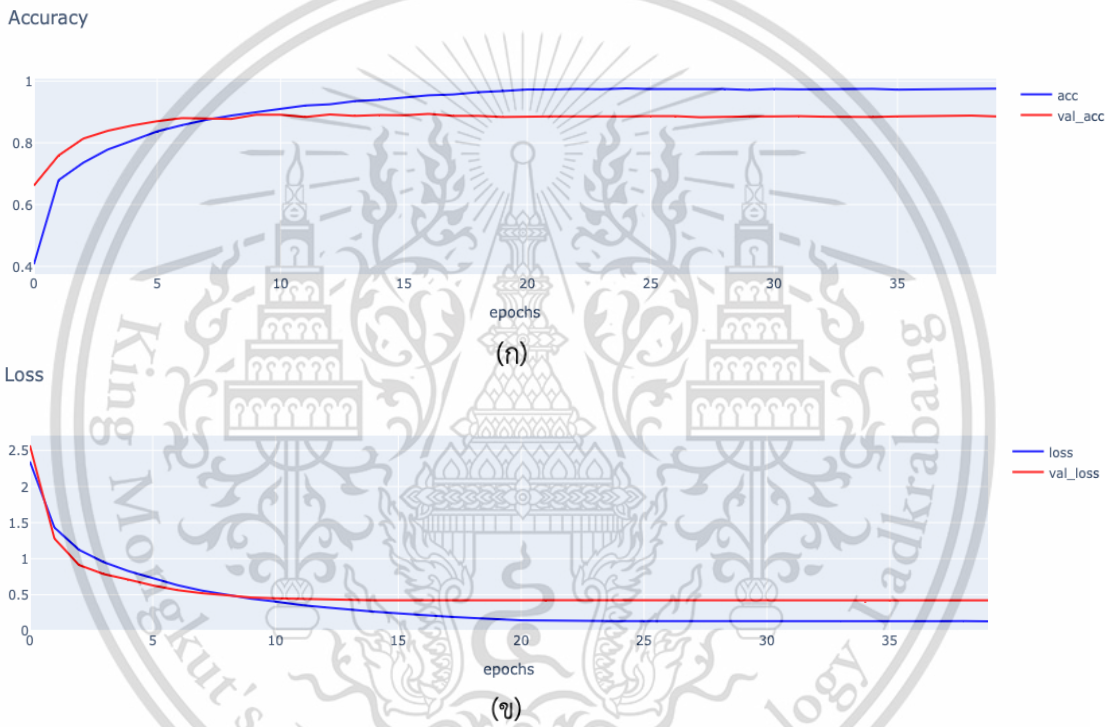
ในการทดลองได้ผลลัพธ์จากแบบจำลอง LSTM ทั้ง 3 แบบจำลอง ได้แก่ 1) แบบจำลอง LSTM Unbalance 2) แบบจำลอง LSTM balance by sample 3) แบบจำลอง LSTM balance by class weight

4.1.1 แบบจำลอง LSTM Unbalance

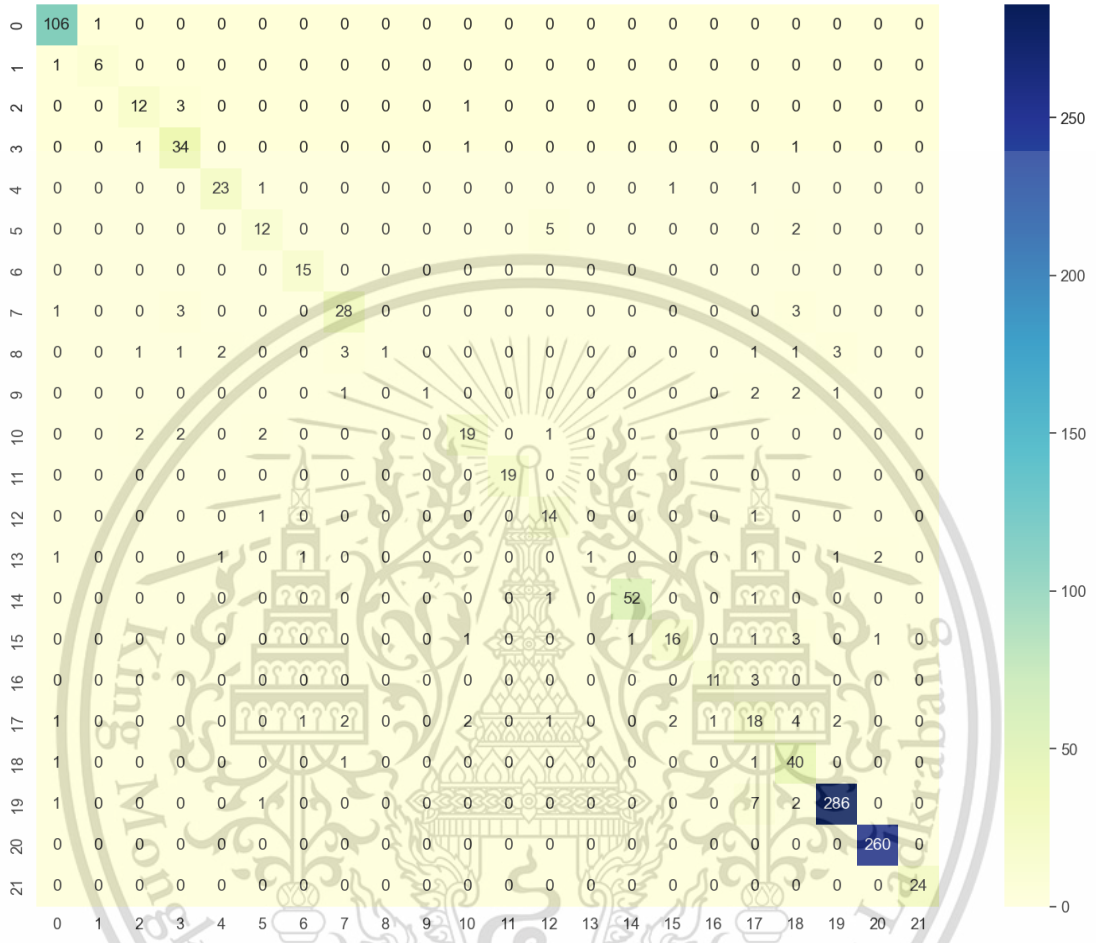
กราฟแบบจำลอง LSTM ที่ไม่ได้มีการจัดการความไม่สมดุลของคลาสจึงทำให้มีข้อมูลแต่ละหมวดหมู่ไม่เท่ากัน

จากรูปที่ 4.1 (ก) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 7 อีพ็อค เกิดจุดตัดค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) ส่วนค่าความแม่นยำเพิ่มขึ้นอย่างต่อเนื่องจนถึงรอบที่ 20 อีพ็อค ในส่วนค่าความแม่นยำตรวจสอบเริ่มนิ่งที่จุดตัดที่ 5 อีพ็อค

จากรูปที่ 4.1 (ข) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 8 อีพ็อค เกิดจุดตัดค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss) ส่วนค่าความสูญเสียลดลงอย่างต่อเนื่องจนถึงรอบที่ 20 อีพ็อค ในส่วนค่าความสูญเสียตรวจสอบเริ่มนิ่งในรอบที่ 10 อีพ็อค



รูปที่ 4.1 กราฟแบบจำลอง LSTM Unbalance (ก) ค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) (ข) ค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss)



รูปที่ 4.2 เมทริกซ์ความสับสน (Confusion Matrix) ของแบบจำลอง LSTM Unbalance

	precision	recall	f1-score	support
0	0.9464	0.9907	0.9680	107
1	0.8571	0.8571	0.8571	7
2	0.7500	0.7500	0.7500	16
3	0.7907	0.9189	0.8500	37
4	0.8846	0.8846	0.8846	26
5	0.7059	0.6316	0.6667	19
6	0.8824	1.0000	0.9375	15
7	0.8000	0.8000	0.8000	35
8	1.0000	0.0769	0.1429	13
9	1.0000	0.1429	0.2500	7
10	0.7917	0.7308	0.7600	26
11	1.0000	1.0000	1.0000	19
12	0.6364	0.8750	0.7368	16
13	1.0000	0.1250	0.2222	8
14	0.9811	0.9630	0.9720	54
15	0.8421	0.6957	0.7619	23
16	0.9167	0.7857	0.8462	14
17	0.4865	0.5294	0.5070	34
18	0.6897	0.9302	0.7921	43
19	0.9761	0.9630	0.9695	297
20	0.9886	1.0000	0.9943	260
21	1.0000	1.0000	1.0000	24
accuracy			0.9073	1100
macro avg	0.8603	0.7568	0.7577	1100
weighted avg	0.9148	0.9073	0.9003	1100

```
{'กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์': 1,
 'กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม': 2,
 'กองสาธารณสุขฯ_ตัดต้นไม้': 3,
 'กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้': 4,
 'กองสาธารณสุขฯ_เหตุเดือดร้อนรำคาญ': 5,
 'กองช่าง_ท่อระบายน้ำ': 6,
 'กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ': 7,
 'กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย': 8,
 'กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม': 9,
 'ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์': 10,
 'สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย': 11,
 'กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_หรือสถานที': 12,
 'กองช่าง_รางระบายน้ำ': 13,
 'กองสาธารณสุขฯ_การป้องกันและควบคุมโรค': 14,
 'สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค': 15,
 'สำนักปลัด_คำร้องทั่วไปสำนักปลัด': 16,
 'กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย': 17,
 'กองช่าง_คำร้องทั่วไปกองช่าง': 18,
 'กองช่าง_ถนน/ไหล่ทาง': 19,
 'กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)': 20,
 'กองสาธารณสุขฯ_รับส่งผู้ป่วย': 21,
 'กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข': 22}
```

รูปที่ 4.3 รายงานค่าความแม่นยำ (Accuracy) และค่าเฉลี่ยมาโคร (Macro-Average) ของแบบจำลอง LSTM Unbalance

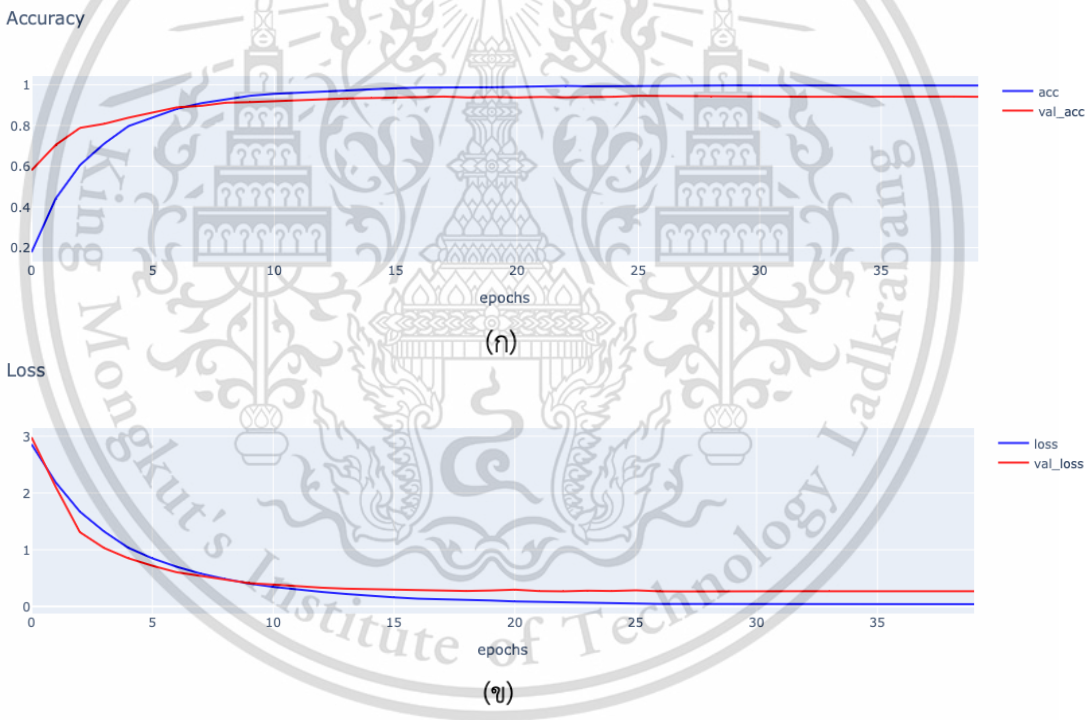
จากรูปที่ 4.2 และ 4.3 จะสังเกตได้ว่าเมทริกซ์ความสับสน (Confusion Matrix) มีการทำนายถูกมากที่สุด 3 อันดับแรกของ 1) กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) 286 แถว 2) กองสาธารณะสุขฯ_รับส่งผู้ป่วย 260 แถว และ 3) กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์ 106 แถว รวม 652 แถว ในส่วนการทำนายจำนวนข้อมูลแถวผิดมากที่สุด 3 อันดับแรก สำนักปลัด_คำร้องทั่วไปสำนักปลัด 16 แถว กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม 12 แถว และกองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) 11 แถว และ จากชุดข้อมูลทดสอบ 1,099 แถว ผลลัพธ์รายงานการจัดหมวดหมู่จากแบบจำลอง LSTM Unbalance ได้ค่าความแม่นยำ (Accuracy) 90.73 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร (Macro-Average) 75.77 เปอร์เซ็นต์ แสดงผลลัพธ์ดังรูปที่ 4.3

4.1.2 แบบจำลอง LSTM balance by sample

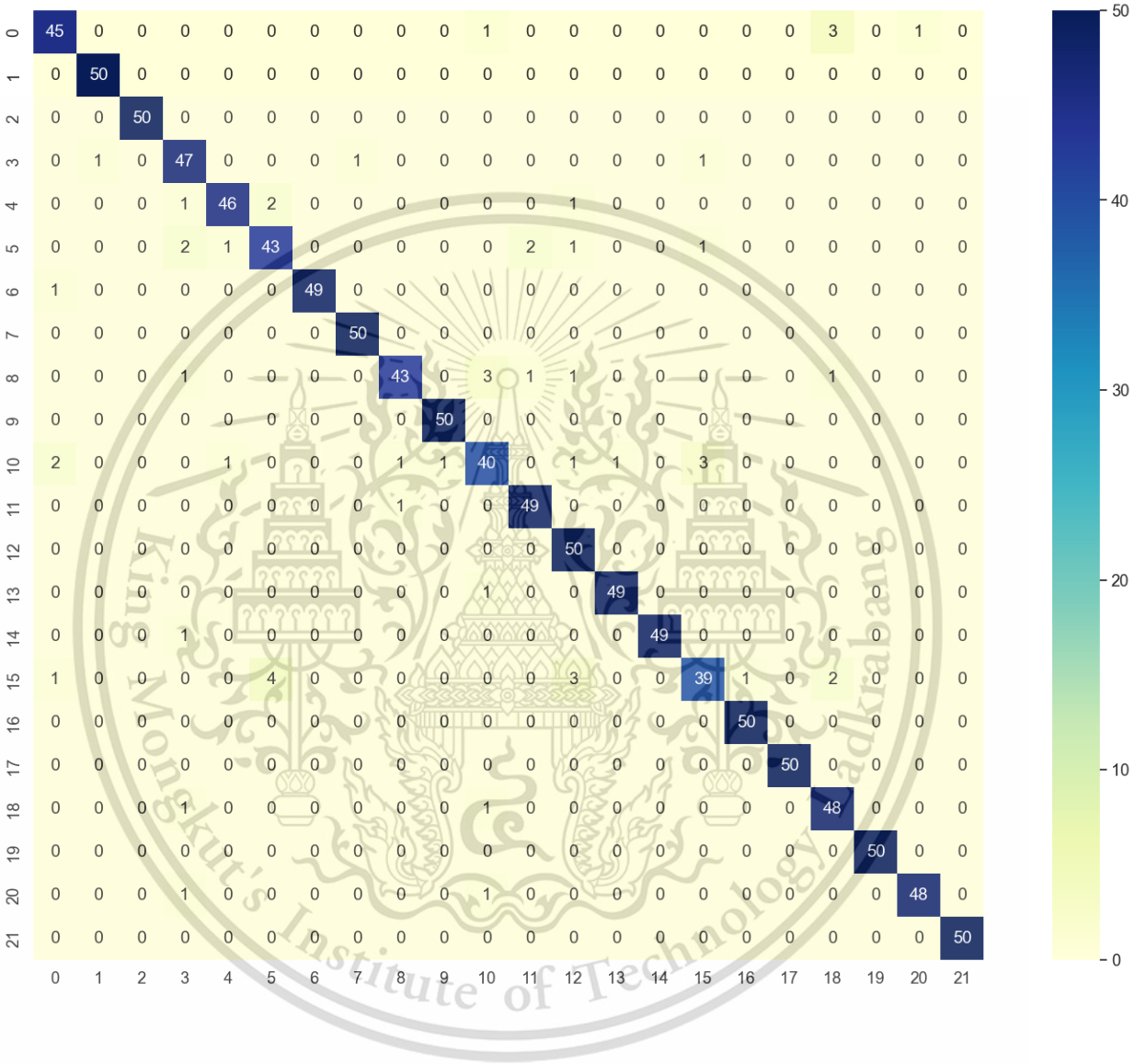
กราฟแบบจำลอง LSTM มีการจัดการความไม่สมดุลของคลาสด้วยวิธีสุ่มตัวอย่างใหม่จึงทำให้มีข้อมูลแต่ละหมวดหมู่เท่ากัน

จากรูปที่ 4.4 (ก) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 7 อีพ็อค เกิดจุดตัดค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) ส่วนค่าความแม่นยำเพิ่มขึ้นอย่างต่อเนื่องจนถึงรอบที่ 15 อีพ็อค ในส่วนค่าความแม่นยำตรวจสอบเริ่มนิ่งที่จุดตัดที่ 10 อีพ็อค

จากรูปที่ 4.4 (ข) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 9 อีพ็อค เกิดจุดตัดค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss) ส่วนค่าความสูญเสียลดลงอย่างต่อเนื่องจนถึงรอบที่ 20 อีพ็อค ในส่วนค่าความสูญเสียตรวจสอบเริ่มนิ่งในรอบที่ 15 อีพ็อค



รูปที่ 4.4 กราฟแบบจำลอง LSTM balance by sample (ก) ค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) (ข) ค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss)



รูปที่ 4.5 เมทริกซ์ความสับสน (Confusion Matrix) ของแบบจำลอง LSTM balance by sample

	precision	recall	f1-score	support
0	0.9184	0.9000	0.9091	50
1	0.9804	1.0000	0.9901	50
2	1.0000	1.0000	1.0000	50
3	0.8704	0.9400	0.9038	50
4	0.9583	0.9200	0.9388	50
5	0.8776	0.8600	0.8687	50
6	1.0000	0.9800	0.9899	50
7	0.9804	1.0000	0.9901	50
8	0.9556	0.8600	0.9053	50
9	0.9804	1.0000	0.9901	50
10	0.8511	0.8000	0.8247	50
11	0.9423	0.9800	0.9608	50
12	0.8772	1.0000	0.9346	50
13	0.9800	0.9800	0.9800	50
14	1.0000	0.9800	0.9899	50
15	0.8864	0.7800	0.8298	50
16	0.9804	1.0000	0.9901	50
17	1.0000	1.0000	1.0000	50
18	0.8889	0.9600	0.9231	50
19	1.0000	1.0000	1.0000	50
20	0.9796	0.9600	0.9697	50
21	1.0000	1.0000	1.0000	50
accuracy			0.9500	1100
macro avg	0.9503	0.9500	0.9495	1100
weighted avg	0.9503	0.9500	0.9495	1100

```
{'กองช่าง_ถนน/ไหล่ทาง': 1,
 'กองสาธารณสุขฯ_การป้องกันและควบคุมโรค': 2,
 'กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์': 3,
 'สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย': 4,
 'กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ': 5,
 'กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้': 6,
 'กองสาธารณสุขฯ_รับส่งผู้ป่วย': 7,
 'กองช่าง_ท่อระบายน้ำ': 8,
 'สำนักปลัด_คำร้องทั่วไปสำนักปลัด': 9,
 'ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์': 10,
 'กองช่าง_คำร้องทั่วไปกองช่าง': 11,
 'กองสาธารณสุขฯ_ตัดต้นไม้': 12,
 'กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม': 13,
 'กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย': 14,
 'สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค': 15,
 'กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย': 16,
 'กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม': 17,
 'กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_หรือสถานที่': 18,
 'กองสาธารณสุขฯ_เหตุเตือดร้อนรำคาญ': 19,
 'กองช่าง_รางระบายน้ำ': 20,
 'กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)': 21,
 'กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข': 22}
```

รูปที่ 4.6 รายงานค่าความแม่นยำ (Accuracy) และค่าเฉลี่ยมาโคร (Macro-Average) ของแบบจำลอง LSTM balance by sample

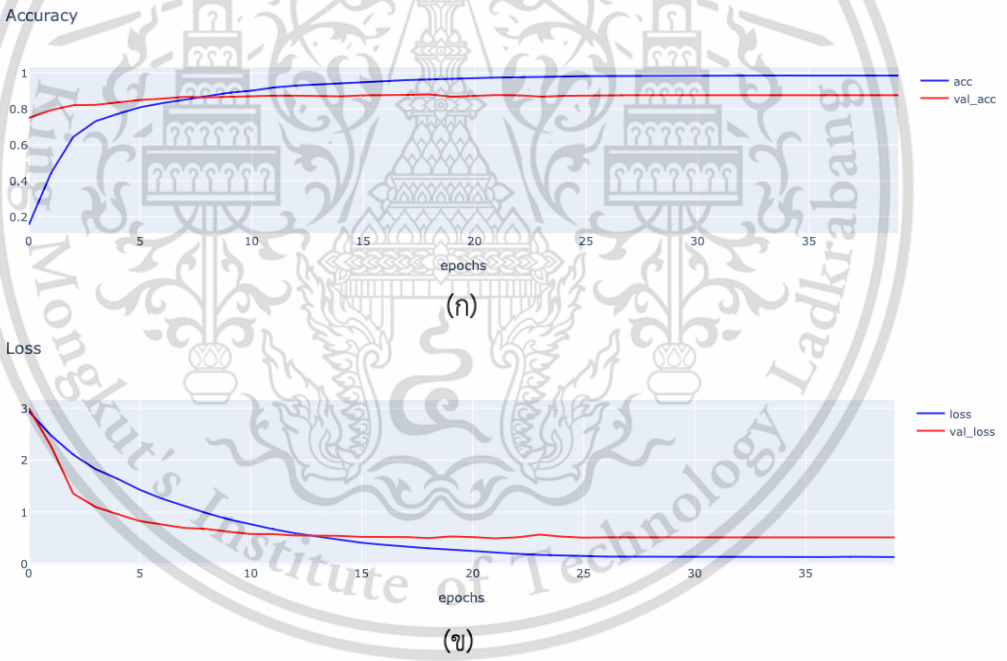
จากรูปที่ 4.5 และ 4.6 จะสังเกตได้ว่าเมทริกซ์ความสับสน มีการทำนายถูกต้อง ส่วนใหญ่ได้ทุกประเภทงานโดยคะแนนทำนายถูกเต็มอยู่ที่ 45 – 50 แถว โดยมีประเภททำนายผิดบ่อยครั้งได้แก่ กองสาธารณะสุข_จัดเก็บขยะมูลฝอย 11 แถว กองช่าง_คำร้องทั่วไปกองช่าง 10 แถว กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้ 7 แถว และสำนักปลัด_คำร้องทั่วไปสำนักปลัด 7 แถว จากชุดข้อมูลทดสอบ 1,100 แถว ผลลัพธ์รายงานการจัดหมวดหมู่จากแบบจำลอง LSTM balance by sample ได้ค่าความแม่นยำ 95.00 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 94.95 เปอร์เซ็นต์ แสดงผลลัพธ์ดังรูปที่ 4.6

4.1.3 แบบจำลอง LSTM balance by class weight

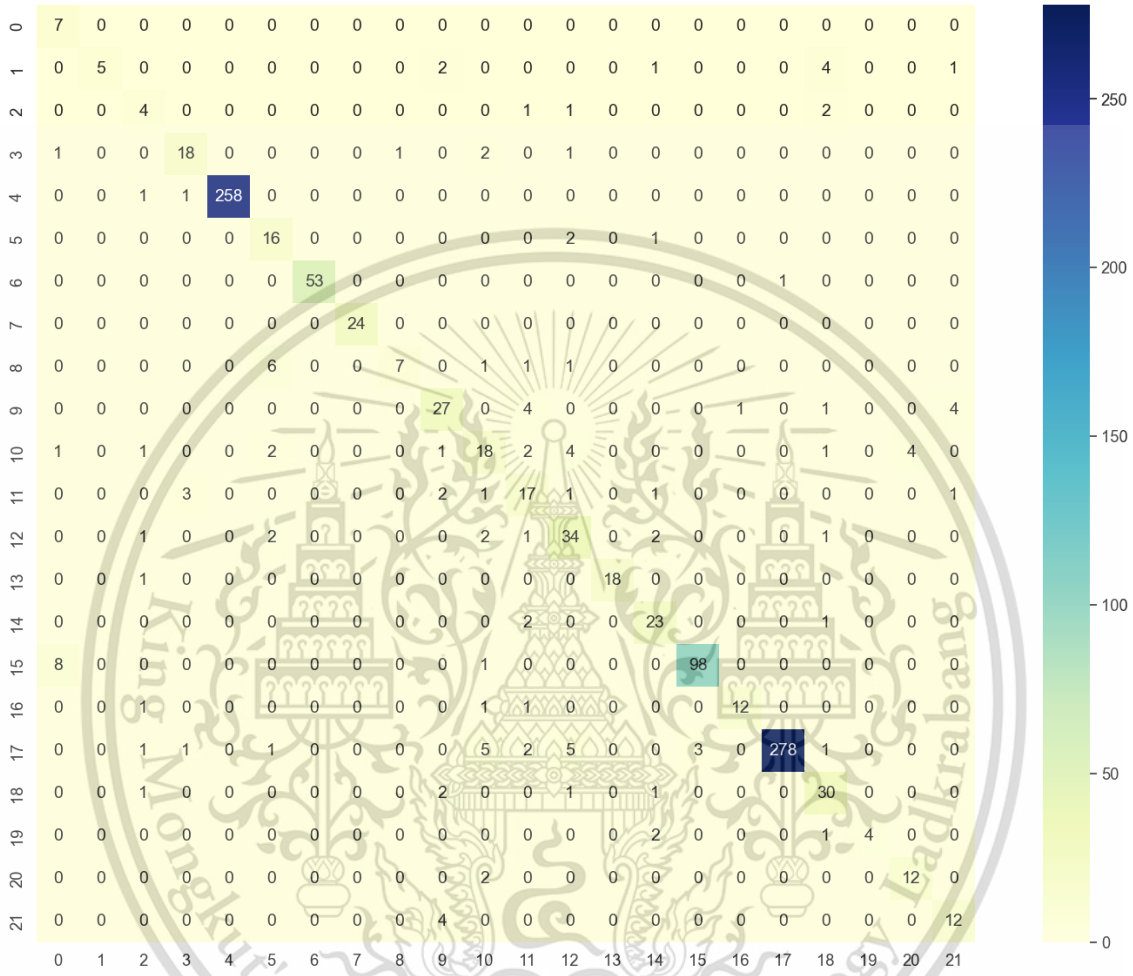
กราฟแบบจำลอง LSTM มีการจัดการความไม่สมดุลของคลาสด้วยวิธีน้ำหนักตัวอย่างจึงให้น้ำหนักความสำคัญกับหมวดหมู่ที่มีจำนวนข้อมูลน้อยให้มีความสำคัญมากขึ้น

จากรูปที่ 4.7 (ก) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 8 อีพ็อค เกิดจุดตัดค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) ส่วนค่าความแม่นยำเพิ่มขึ้นอย่างต่อเนื่องจนถึงรอบที่ 20 อีพ็อค ในส่วนค่าความแม่นยำตรวจสอบเริ่มนิ่งที่จุดตัดที่ 5 อีพ็อค

จากรูปที่ 4.7 (ข) เมื่อทำการสอนแบบจำลองไปจนถึงรอบที่ 13 อีพ็อค เกิดจุดตัดค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss) ส่วนค่าความสูญเสียลดลงอย่างต่อเนื่องจนถึงรอบที่ 25 อีพ็อค ในส่วนค่าความสูญเสียตรวจสอบเริ่มนิ่งในรอบที่ 10 อีพ็อค



รูปที่ 4.7 กราฟแบบจำลอง LSTM balance by class weight (ก) ค่าความแม่นยำ (Accuracy) และค่าความแม่นยำตรวจสอบ (Validation Accuracy) (ข) ค่าความสูญเสีย (Loss) และค่าความสูญเสียตรวจสอบ (Validation Loss)



รูปที่ 4.8 เมทริกซ์ความสับสน (Confusion Matrix) ของแบบจำลอง LSTM balance by class weight

	precision	recall	f1-score	support
0	0.4118	1.0000	0.5833	7
1	1.0000	0.3846	0.5556	13
2	0.3636	0.5000	0.4211	8
3	0.7826	0.7826	0.7826	23
4	1.0000	0.9923	0.9961	260
5	0.5926	0.8421	0.6957	19
6	1.0000	0.9815	0.9907	54
7	1.0000	1.0000	1.0000	24
8	0.8750	0.4375	0.5833	16
9	0.7105	0.7297	0.7200	37
10	0.5455	0.5294	0.5373	34
11	0.5484	0.6538	0.5965	26
12	0.6800	0.7907	0.7312	43
13	1.0000	0.9474	0.9730	19
14	0.7419	0.8846	0.8070	26
15	0.9703	0.9159	0.9423	107
16	0.9231	0.8000	0.8571	15
17	0.9964	0.9360	0.9653	297
18	0.7143	0.8571	0.7792	35
19	1.0000	0.5714	0.7273	7
20	0.7500	0.8571	0.8000	14
21	0.6667	0.7500	0.7059	16
accuracy			0.8864	1100
macro avg	0.7851	0.7793	0.7614	1100
weighted avg	0.9031	0.8864	0.8893	1100

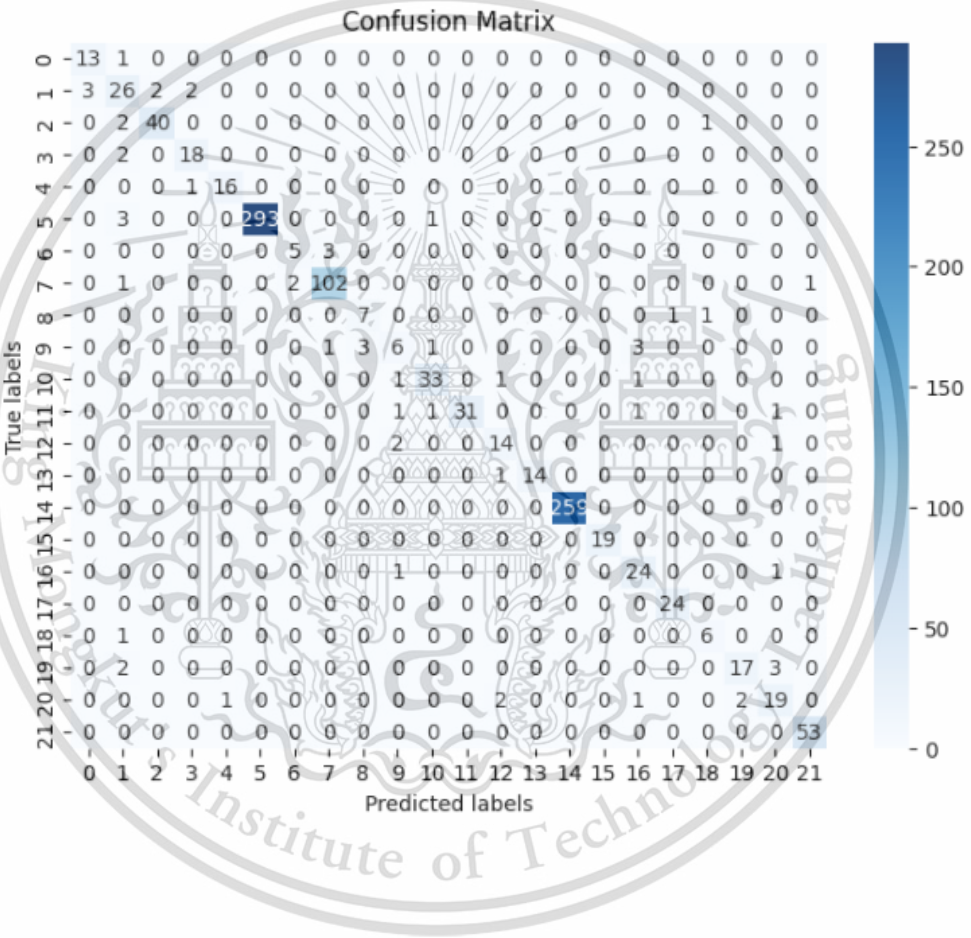
```
{'กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม': 1,
 'กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม': 2,
 'กองสาธารณสุขฯ_การป้องกันและควบคุมโรค': 3,
 'สำนักปลัด_คำร้องทั่วไปสำนักปลัด': 4,
 'กองสาธารณสุขฯ_รับส่งผู้ป่วย': 5,
 'กองช่าง_ท่อระบายน้ำ': 6,
 'สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค': 7,
 'กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตรายต่อสุข': 8,
 'กองช่าง_รางระบายน้ำ': 9,
 'กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้': 10,
 'กองช่าง_คำร้องทั่วไปกองช่าง': 11,
 'สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย': 12,
 'กองช่าง_ถนน/ไหล่ทาง': 13,
 'กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_หรือสถานที': 14,
 'กองสาธารณสุขฯ_เหตุเดือดร้อนรำคาญ': 15,
 'กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์': 16,
 'กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ': 17,
 'กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)': 18,
 'กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย': 19,
 'ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์': 20,
 'กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย': 21,
 'กองสาธารณสุขฯ_ตัดต้นไม้': 22}
```

รูปที่ 4.9 รายงานค่าความแม่นยำ (Accuracy) และค่าเฉลี่ยมาโคร (Macro-Average) ของแบบจำลอง LSTM balance by class weight

จากรูปที่ 4.8 และ 4.9 จะสังเกตได้ว่าเมทริกซ์ความสับสนมีความคล้ายกับแบบจำลอง LSTM Unbalance มีการทำนายถูกมากที่สุด 3 อันดับแรก กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) 278 แถว กองสาธารณะสุขฯ_รับส่งผู้ป่วย 258 แถว และกองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์ 98 แถว รวม 634 แถว ในส่วนการทำนายจำนวนข้อมูลแถวผิดมากที่สุด 3 อันดับแรก กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) 19 แถว กองช่าง_คำร้องทั่วไปกองช่าง 16 แถว กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์ 9 แถว และสำนักปลัด_ป้องกันและบรรเทาสาธารณภัย 9 แถว จากชุดข้อมูลทดสอบ 1,100 แถว ผลลัพธ์รายงานการจัดหมวดหมู่จากแบบจำลอง LSTM balance by class weight ได้ค่าความแม่นยำ 88.64 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 76.14 เปอร์เซ็นต์ แสดงผลลัพธ์ดังรูปที่ 4.9

4.1.4 แบบจำลอง WangchanBERTa

แบบจำลอง WangchanBERTa ใช้ชื่อแบบจำลอง wangchanberta-base-att-spm-uncased ซึ่งเป็น แบบจำลองภาษาขนาดใหญ่ (LLM) ที่ได้ผลลัพธ์ความแม่นยำของแบบจำลองดีที่สุดโดยใช้ ชุดข้อมูลจาก Assorted Thai Texts และใช้โทเค็นไนเซอร์ คือการตัวตัดคำแบบชิ้นส่วนประโยคในการทดลองได้ผลลัพธ์ของแบบจำลอง WangchanBERTa



รูปที่ 4.10 เมทริกซ์ความสับสน (Confusion Matrix) ของแบบจำลอง WangchanBERTa

	precision	recall	f1-score	support
0	0.81	0.93	0.87	14
1	0.68	0.79	0.73	33
2	0.95	0.93	0.94	43
3	0.86	0.90	0.88	20
4	0.94	0.94	0.94	17
5	1.00	0.99	0.99	297
6	0.71	0.62	0.67	8
7	0.96	0.96	0.96	106
8	0.70	0.78	0.74	9
9	0.55	0.43	0.48	14
10	0.92	0.92	0.92	36
11	1.00	0.89	0.94	35
12	0.78	0.82	0.80	17
13	1.00	0.93	0.97	15
14	1.00	1.00	1.00	259
15	1.00	1.00	1.00	19
16	0.80	0.92	0.86	26
17	0.96	1.00	0.98	24
18	0.75	0.86	0.80	7
19	0.89	0.77	0.83	22
20	0.76	0.76	0.76	25
21	0.98	1.00	0.99	53
accuracy			0.95	1099
macro avg	0.86	0.87	0.87	1099
weighted avg	0.95	0.95	0.95	1099

type_name	label
กองช่าง_ขออนุญาตก่อสร้างอาคาร_ประเภทบ้านพักอาศัย	0
กองช่าง_คำร้องทั่วไปกองช่าง	1
กองช่าง_ถนน/ไหล่ทาง	2
กองช่าง_ท่อระบายน้ำ	3
กองช่าง_รางระบายน้ำ	4
กองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง)	5
กองสวัสดิการสังคม_คำร้องทั่วไปกองสวัสดิการสังคม	6
กองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์	7
กองสาธารณสุขฯ_การป้องกันและควบคุมโรค	8
กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแ...	9
กองสาธารณสุขฯ_จัดเก็บกิ่งไม้ใบไม้	10
กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย	11
กองสาธารณสุขฯ_ตัดต้นไม้	12
กองสาธารณสุขฯ_ตัดหญ้าที่สาธารณะ	13
กองสาธารณสุขฯ_รับส่งผู้ป่วย	14
กองสาธารณสุขฯ_หนังสือรับรองการแจ้งสะสมอาหาร_ทร...	15
กองสาธารณสุขฯ_เหตุเดือดร้อนรำคาญ	16
กองสาธารณสุขฯ_ใบอนุญาตประกอบกิจการที่เป็นอันตร...	17
ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้อ...	18
สำนักปลัด_คำร้องทั่วไปสำนักปลัด	19
สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย	20
สำนักปลัด_สนับสนุนน้ำอุปโภค_บริโภค	21

รูปที่ 4.11 รายงานค่าความแม่นยำ (Accuracy) และค่าเฉลี่ยมาโคร (Macro-Average) ของแบบจำลอง WangchanBERTa

จากรูปที่ 4.10 และ 4.11 จะสังเกตได้ว่าเมทริกซ์ความสับสนมีการทำนายถูกมากที่สุด 3 อันดับแรก ของกองช่าง_ไฟฟ้าสาธารณะ(ไฟกิ่ง) 293 แถว กองสาธารณะสุขฯ_รับส่งผู้ป่วย 259 แถว และกองสวัสดิการสังคม_ยืมวัสดุอุปกรณ์ 102 แถว รวม 654 แถว ในส่วนการทำนายจำนวนข้อมูลแถวผิดมากที่สุด 3 อันดับแรก กองสาธารณสุขฯ_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม 8 แถว กองช่าง_คำร้องทั่วไปกองช่าง 7 แถว และสำนักปลัด_ป้องกันและบรรเทาสาธารณภัย 6 แถว จากชุดข้อมูลทดสอบ 1,099 แถว ผลลัพธ์รายงานการจัดหมวดหมู่จากแบบจำลอง WangchanBERTa ได้ค่าความแม่นยำ 95.00 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 87.00 เปอร์เซ็นต์ แสดงผลลัพธ์ดังรูปที่ 4.11

ถัดมาจึงได้ทดสอบประสิทธิภาพโดยนำแบบจำลองไปสร้างเว็บไซต์อย่างง่ายใช้งานบน Streamlit และเก็บข้อมูลไว้บน Github โดยใช้ LSTM balance by class weight เป็นตัวทดสอบแรกและแบบจำลอง WangchanBERTa ทำเว็บไซต์อย่างง่ายด้วย Streamlit เช่นกันแต่เก็บข้อมูลไว้บน Hugging Face Space เนื่องจากข้อมูลแบบจำลองมีขนาดใหญ่จึงไม่สามารถวางบน Github ได้

4.2 การสร้างเว็บไซต์อย่างง่าย

ต่อมาจึงได้ทดสอบประสิทธิภาพโดยนำแบบจำลองไปสร้างเว็บไซต์อย่างง่ายใช้งานบน Streamlit โดยใช้ LSTM balance by class weight เป็นตัวทดสอบแรกและแบบจำลองที่สอง WangchanBERTa ซึ่งเป็นตัวทดสอบสองที่ให้ค่าความแม่นยำ และค่าเฉลี่ยมาโครที่ดีกว่า

ปัญหาที่พบเมื่อนำโค้ดจากโปรแกรมในสิ่งแวดล้อมสำหรับการพัฒนาแบบเบ็ดเสร็จ (IDE) เช่น Jupyter Notebook และ Google Colaboratory มาสร้างสร้างเว็บไซต์อย่างง่าย ควรเก็บเวอร์ชันแพ็คเกจของไลบรารี่ ในเวอร์ชันที่ใช้แล้วในเครื่องนั้น หรือกำหนดไว้ใน requirement.txt และควรใช้ Python เวอร์ชันเดียวกันด้วยเพื่อลดปัญหาการอัพเดตของแต่ละเวอร์ชันในแพ็คเกจของไลบรารี่ และเวอร์ชันใน Python

หน่วยงานและประเภทงานที่ตรวจสอบคือสิ่งที่หน่วยงาน กองสาธารณสุขฯ ประเภทงานการป้องกันและควบคุมโรค โดยยกตัวอย่างที่ 1 คือ

หัวข้อ (Subject) : ขอความอนุเคราะห์ทางเทศบาล พ่นยาฆ่าหญ้า บริเวณรอบอาคาร SML

เนื้อเรื่อง (Detail) : ขอความอนุเคราะห์ทางเทศบาล พ่นยาฆ่าหญ้า บริเวณรอบอาคาร SML หมู่ที่ 1 บริเวณหลังวัดตำหนัก เข้าทางวัด เข้าซอย 8 ร้านตัดผมลุงช่วย ซึ่งบริเวณนี้ มียุงชุกชุมตอนหัวค่ำ 5 โมงเป็นต้นไป และสถานที่แห่งนี้ ชมรถผู้สูงอายุหมู่ที่ 1 จะมีการออกกำลังการ ขอไปดำเนินการ

ใช้หัวข้อและเนื้อเรื่องมาทำนายโดยใช้เว็บไซต์อย่างง่าย ทำนายหมวดหมู่ร้องเรียนของทั้งสองแบบจำลอง โดยแบบจำลอง LSTM balance by class weight ทำนายผลลัพธ์ กองสาธารณสุข_ตัดหญ้าที่สาธารณะ ซึ่งไม่ถูกต้อง แต่แบบจำลอง WangchanBERTa ทำนาย กองสาธารณสุขฯ_การป้องกันและควบคุมโรค ซึ่งถูกต้องตามที่แบบจำลองควรทำนายได้ แสดงดังรูปที่ 4.12

รูปที่ 4.12 เว็บไซต์อย่างง่ายทำนายหมวดหมู่ร้องเรียนของ LSTM balance by class weight และ WangchanBERTa หน่วยงาน กองสาธารณสุขฯ ประเภทงาน การป้องกันและควบคุมโรค ตัวอย่างที่ 1

รูปที่ 4.13 เว็บไซต์อย่างง่ายทำนายหมวดหมู่ร้องเรียนของ LSTM balance by class weight และ WangchanBERTa หน่วยงาน กองสาธารณสุขฯ ประเภทงาน การป้องกันและควบคุมโรค ตัวอย่างที่ 2

หน่วยงานและประเภทงานที่ตรวจสอบคือสิ่งที่หน่วยงาน กองสาธารณสุขฯ ประเภทงานการป้องกันและควบคุมโรค โดยยกตัวอย่างที่ 2 คือ

หัวข้อ (Subject) : ขอความอนุเคราะห์พ่นหมอกควันไข้เลือดออก หมู่ 5
เนื้อเรื่อง (Detail) : ข้าพเจ้ามีความประสงค์ขอความอนุเคราะห์พ่นหมอกควันไข้เลือดออก บริเวณ ซอย 6

ใช้หัวข้อและเนื้อเรื่องมาทำนายโดยใช้เว็บไซต์อย่างง่าย ทำนายหมวดหมู่ร้องเรียนของทั้งสองแบบจำลอง โดยแบบจำลอง LSTM balance by class weight ทำนายผลลัพธ์กองสาธารณสุข_คำร้องทั่วไปกองสาธารณสุขและสิ่งแวดล้อม ซึ่งไม่ถูกต้อง แต่แบบจำลอง WangchanBERTa ทำนาย กองสาธารณสุขฯ_การป้องกันและควบคุมโรค ซึ่งถูกต้องตามที่แบบจำลองควรทำนายได้ แสดงดังรูปที่ 4.13

ต่อมาเป็นการคิดหัวข้อและเนื้อเรื่องขึ้นมาเองว่าทั้งสองแบบจำลองสามารถทำนายได้ถูกต้องหรือไม่ โดยยกตัวอย่างที่ 3 คือ

หัวข้อ (Subject) : งู ตัวเงินตัวทอง เข้าบ้าน
เนื้อเรื่อง (Detail) : ข้าพเจ้าขอความช่วยเหลือเนื่องจากมีงู ตัวเงินตัวทอง เข้าบ้าน ช่วยส่งเจ้าหน้าที่มาดูแลหน่อยครับ

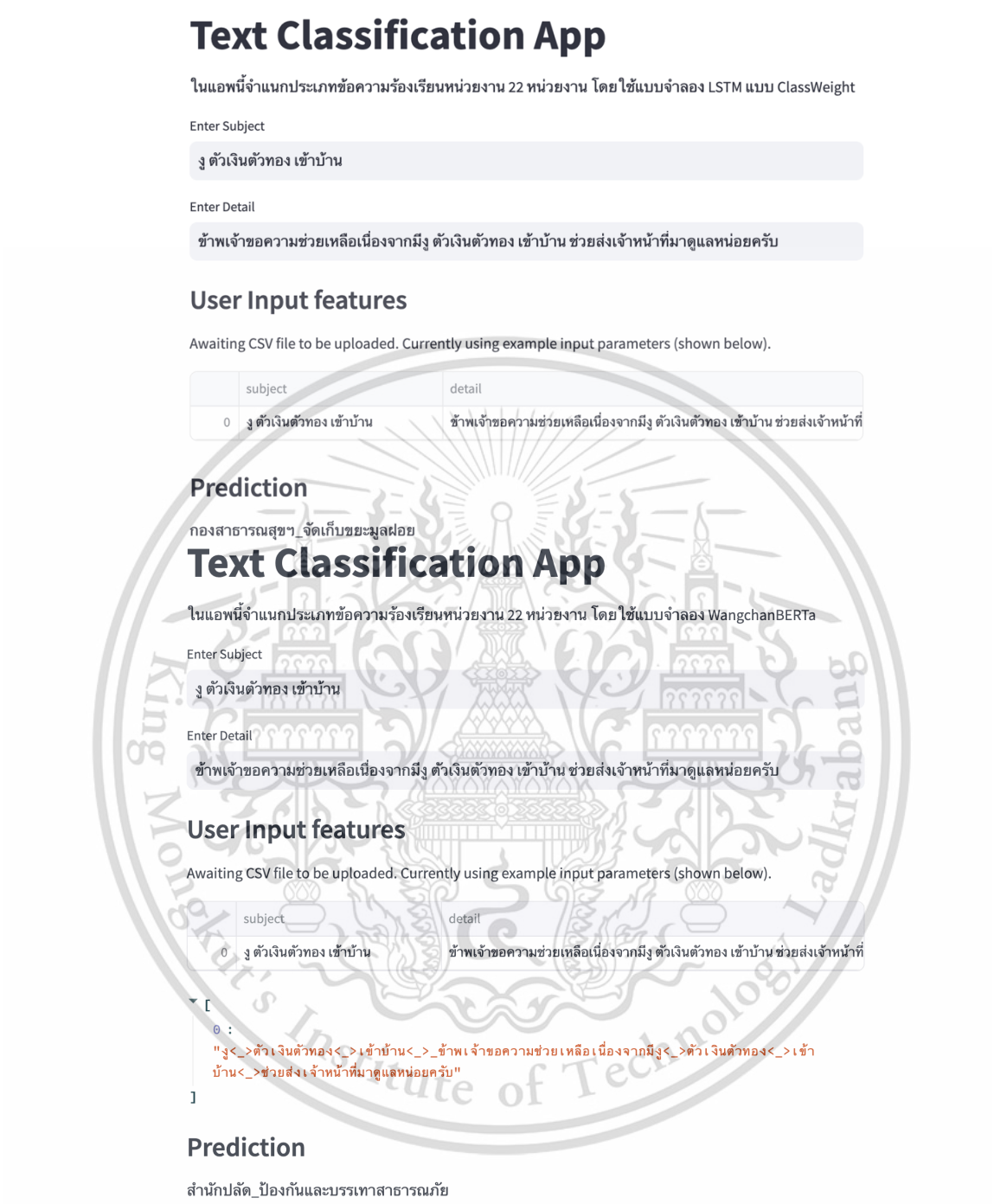
ใช้หัวข้อและเนื้อเรื่องมาทำนายโดยใช้เว็บไซต์อย่างง่าย ทำนายหมวดหมู่ร้องเรียนของทั้งสองแบบจำลอง โดยแบบจำลอง LSTM balance by class weight ทำนายผลลัพธ์ กองสาธารณสุขฯ_จัดเก็บขยะมูลฝอย ซึ่งไม่ถูกต้อง แต่แบบจำลอง WangchanBERTa ทำนาย สำนักปลัด_ป้องกันและบรรเทาสาธารณภัย ซึ่งแบบจำลอง WangchanBERTa มีการทำนายถูกต้องตามเนื้อหามากกว่า แสดงดังรูปที่ 4.14

สุดท้ายเป็นการคิดหัวข้อและเนื้อเรื่องขึ้นมาเองว่าทั้งสองแบบจำลองสามารถทำนายได้ถูกต้องหรือไม่ โดยยกตัวอย่างที่ 4 คือ

หัวข้อ (Subject) : พบเจอคนสติไม่สมประกอบ

เนื้อเรื่อง (Detail) : พบเจอคนสติไม่สมประกอบ ทำร้ายเด็ก บริเวณ 7-11 ใกล้กับสวนสาธารณะหมู่บ้าน อยากให้เจ้าหน้าที่เข้ามาดูแลด้วยค่ะ

ใช้หัวข้อและเนื้อเรื่องมาทำนายโดยใช้เว็บไซต์อย่างง่าย ทำนายหมวดหมู่ร้องเรียนของทั้งสองแบบจำลอง โดยแบบจำลอง LSTM balance by class weight ทำนายผลลัพธ์ กองช่าง_ถนน/ไหล่ทาง ซึ่งไม่ถูกต้อง แต่แบบจำลอง WangchanBERTa ทำนาย ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์ ซึ่งแบบจำลอง WangchanBERTa มีการทำนายถูกต้องตามเนื้อหามากกว่า แสดงดังรูปที่ 4.15



รูปที่ 4.14 เว็บไซต์อย่างง่ายทำนายหมวดหมู่ร้องเรียนของ LSTM balance by class weight และ WangchanBERTa โดยคิดหัวข้อและเนื้อเรื่องขึ้นมา ตัวอย่างที่ 3

รูปที่ 4.15 เว็บไซต์อย่างง่ายทำนายหมวดหมู่ร้องเรียนของ LSTM balance by class weight และ WangchanBERTa โดยคิดหัวข้อและเนื้อเรื่องขึ้นมา ตัวอย่างที่ 4

4.3 การอภิปรายผล

ในแบบจำลองแรกคือ แบบจำลอง LSTM มีการลบคำฟุ่มเฟือยในคลังคำศัพท์ PythaiNLP และคำที่กำหนดขึ้นมาเอง เช่น วันจันทร์ ถึงวันอาทิตย์ และเดือนมกราคม ถึง เดือนธันวาคม เข้าคลังคำศัพท์ฟุ่มเฟือยเพิ่มเติม เพราะบริบทเหล่านี้ไม่ได้บ่งบอกว่าควรทำนายเข้าหมวดหมู่ใดใน 22 หมวดหมู่ จึงไม่มีความจำเป็นที่จะให้แบบจำลองเรียนรู้ แล้วก่อนลบคำฟุ่มเฟือยควรกำหนดคำเหล่านี้เพิ่มเติมไปในคลังคำศัพท์ของ PyThaiNLP เพื่อให้มีคำเหล่านี้อยู่ในคลังคำศัพท์ของพจนานุกรมก่อนที่จะนำการตัดคำและสุดท้ายนำไปลบทิ้งในช่วงคลังคำศัพท์ฟุ่มเฟือย

ถัดมาการใช้งานแบตช์นอร์มัลไลเซชัน (BN) คือ เทคนิคที่ถูกใช้เพื่อเพิ่มประสิทธิภาพในการฝึกสอนโครงข่ายระบบประสาทเชิงลึก โดยการปรับลำดับข้อมูลอินพุตในแอ็กทิเวชันเลเยอร์ ของโครงข่ายเชิงลึก ให้มีค่าทางสถิติใกล้เคียงกันในแต่ละแบตช์ ของข้อมูลที่ถูกนำเข้าให้มีค่าทางสถิติที่เฉลี่ยเป็น 0 และรีสเกลให้ข้อมูลอยู่ในช่วง 0 กับ 1 วิธีนี้ช่วยลดการกระโดดของน้ำหนักขณะฝึกสอนแบบจำลองด้วยข้อมูลในแต่ละแบตช์ซึ่งทำให้แบบจำลองปรับตัวได้เพื่อเข้าสู่จุดที่ทำงานได้เหมาะสมเร็วขึ้น และมีการใช้ Dropout คือทำการสุ่มลบบางจำนวนของโหนดในชั้นที่ซ่อนอยู่ฮิดเดนเลเยอร์ (Hidden Layer) ขณะฝึกสอน Dropout ถูกออกแบบมาเพื่อลดปัญหา Overfitting ในแบบจำลองซึ่งเกิดจากการที่แบบจำลองจดจำข้อมูลการฝึกสอนมากเกินไปและไม่สามารถทำนายข้อมูลใหม่ได้ดี

ตารางที่ 4.1 เปรียบเทียบแบบจำลอง LSTM Unbalance กับแบบจำลอง LSTM balance ในเรื่องการ์ดจอ เวลา พารามิเตอร์ ความแม่นยำ และค่าเฉลี่ยมาโคร

แบบจำลอง (Model)	**การ์ดจอ (GPU)**	**เวลา (Time)**	**พารามิเตอร์ (Parameter)**	**ความแม่นยำ (Accuracy)**	**ค่าเฉลี่ยมาโคร (Macro-Average)**
LSTM Unbalance	M2	15 นาที	1,497,558 ตัว	90.73 %	75.77 %
LSTM balance by sample	M2	15 นาที	1,160,790 ตัว	95.00 %	94.95 %
LSTM balance by class weight	M2	14 นาที	1,497,558 ตัว	88.64 %	76.14 %

ได้ทดลองการจัดการความไม่สมดุลของคลาส (Class Imbalance) เพื่อเทียบประสิทธิภาพทั้ง 3 แบบจำลองในสถาปัตยกรรม LSTM โดยแบ่งเป็น 1) LSTM Unbalance 2) LSTM balance by sample 3) LSTM balance by class weight เพื่อเทียบประสิทธิภาพจากผลลัพธ์รายงานการจัดหมวดหมู่ในค่าความแม่นยำ และค่าเฉลี่ยมาโคร แสดงผลลัพธ์ดังตารางที่ 4.1

สังเกตได้ว่า LSTM balance by sample ได้ค่าความแม่นยำสูงที่สุด แต่การที่แบบจำลองทำนายถูกมากกว่าอีกสองแบบ นั้นเพราะข้อมูลบางหมวดหมู่น้อยมีผลให้เกิดการทำสำเนาซ้ำแถวเดิมทำให้ในรายงานได้ค่าความแม่นยำสูงเกินความเป็นจริงเนื่องจากเป็นการเรียนรู้รูปแบบคำเหมือนซ้ำ ๆ หลายรอบและทำนายสิ่งที่เคยเห็นมาแล้ว ดังนั้นแบบจำลอง LSTM balance by sample จึงมีความฉลาดมากกว่าอีกสองแบบเพราะเคยเห็นข้อมูลมาก่อนล่วงหน้าแล้ว เช่น ศูนย์รับเรื่องราวร้องทุกข์_เรื่องร้องเรียน/ร้องทุกข์ มีข้อมูล 72 แถว หาค่าเฉลี่ยจากข้อมูลทุกหมวดหมู่ 10,993/22 = 500 แถว ทำข้อมูลจาก 72 แถว ให้เป็น 500 แถวนั้นต้องมีการทำสำเนาซ้ำเดิมถึง 428 แถว หรือทำสำเนาซ้ำเดิมมากกว่า 5-6 ครั้งพอแบ่งข้อมูลออกเป็น 3 ชุด มีข้อมูลที่เคยเห็นมาแล้วอยู่ใน ข้อมูลฝึกสอน ตรวจสอบ และข้อมูลทดสอบด้วย สรุปคือข้อมูลทดสอบนั้นเป็นข้อความเดียวกันกับข้อมูลฝึกสอน ซึ่งเปรียบเสมือนการรู้คำตอบล่วงหน้า

เมื่อเปรียบเทียบ LSTM balance by class weight กับ LSTM Unbalance จะสังเกตได้ว่าได้ค่าความแม่นยำน้อยกว่า 2.09 เปอร์เซ็นต์ แต่ในส่วนค่าเฉลี่ยมาโครได้มากกว่า 0.37 เปอร์เซ็นต์ ซึ่งเป็นค่าเฉลี่ยที่บ่งบอกถึงการทำนายถูกต้องของแต่ละหมวดหมู่ทำนายได้ถูกต้องมากกว่า ซึ่งการทำนายถูกของแบบจำลอง LSTM Unbalance ทำนายถูกเฉพาะหมวดหมู่ที่มีจำนวนมากเท่านั้นแต่แบบจำลอง LSTM balance by class weight ทำนายถูกในหมวดหมู่ที่มีข้อมูลจำนวนน้อยด้วยซึ่งได้มีการถ่วงน้ำหนักให้มีความสำคัญเพิ่มมากขึ้นในการฝึกสอนเพราะฉะนั้นในกรณีข้อมูลชุดนี้จึงมีความเหมาะสมกับการใช้แบบจำลอง LSTM balance by class weight มากกว่า แบบจำลอง LSTM balance by sample และแบบจำลอง LSTM Unbalance อาจเห็นความแตกต่างของค่าเฉลี่ยมาโครสูงมากกว่านี้ในแบบจำลอง LSTM balance by class weight ถ้าใช้ประเภทงานที่มีจำนวนแถวน้อยกว่า 50 แถว

ส่วนต่อมาคือลักษณะของกราฟทั้ง 3 แบบจำลองมีลักษณะคล้ายกัน คือ ค่าความแม่นยำมีแนวโน้มสูงขึ้น และค่าความสูญเสียมีแนวโน้มลดลง แต่มีช่องว่างระหว่างค่าความแม่นยำ และค่าความแม่นยำตรวจสอบโดยค่าความแม่นยำมีค่ามากกว่าค่าความแม่นยำตรวจสอบ รวมทั้งมีช่องว่างระหว่างค่าความสูญเสีย และค่าความสูญเสียตรวจสอบโดยค่าความสูญเสียมีค่าน้อยกว่าค่าความสูญเสียตรวจสอบ

สาเหตุที่ได้กราฟรูปที่ 4.1, 4.4 และ 4.7 บ่งบอกว่า ชุดข้อมูลฝึกสอนไม่สามารถเป็นตัวแทน (Unrepresentative Train Dataset) ซึ่งแสดงถึงว่ามีชุดข้อมูลฝึกสอนน้อยไปที่ให้แบบจำลองในการเรียนรู้อาจต้องเพิ่มข้อมูลมากขึ้น หรือปรับสัดส่วนของแบ่งข้อมูลเป็น 3 ชุด จากข้อมูล ฝึกสอน-

ตรวจสอบ-ทดสอบ ทั้งนี้วิธีที่แก้ปัญหาได้ดีที่สุดคือควรเก็บข้อมูลเพิ่มมากขึ้นเพื่อให้ชุดข้อมูลฝึกสอนสามารถเป็นตัวแทนของทุกหมวดหมู่ประเภทงานได้

แบบจำลองที่สองคือ แบบจำลอง WangchanBERTa โดยใช้ชื่อแบบจำลอง wangchanberta-base-att-spm-uncased ซึ่งเป็น แบบจำลองภาษาขนาดใหญ่ (LLM) ที่ได้ผลลัพธ์ความแม่นยำของแบบจำลองดีที่สุดโดยใช้ ชุดข้อมูลจาก Assorted Thai Texts และใช้โทเค็นไนเซอร์ คือการตัวตัดคำแบบชิ้นส่วนประโยค และใช้ Google Colaboratory แทนซึ่งสามารถเช่าการ์ดจอ (GPU) ที่แรงขึ้นเพื่อให้สามารถฝึกสอนแบบจำลองได้เร็วขึ้น โดยใช้การ์ดจอ V100 ในการฝึกสอนได้มีการกำหนดค่าพารามิเตอร์ดังนี้

1) ขนาดแบตช์ (Batch Size) = 16 เพราะ แรมการ์ดจอ (VRAM) ของ Google Colaboratory นั้นมีขนาด 16 กิกะไบต์ (GB) ทำให้การฝึกสอนแต่ละแบตช์ไม่พียงพอต่อการใช้งานที่ขนาดแบตช์ 16 ขึ้นไป เช่น 32 เป็นต้น

2) Learning Rate = $3x10^{-5}$ มีการปรับค่าตามแนะนำของการปรับแต่งในแบบจำลอง BERT ($2x10^{-5}$, $3x10^{-5}$, $5x10^{-5}$)

3) Num Train Epochs = 5 อีพ็อค การปรับแต่งใน BERT แนะนำช่วง 3-4 อีพ็อค ซึ่งในงานตระกูล BERT ไม่ควรฝึกสอนเกิน 5 อีพ็อค ตามคำแนะนำของ BERT

4) Weight Decay = 0.01 ใช้เท่ากับแบบจำลอง RoBERTa

5) Warmup Percent = 0.1 ใช้ Warmup Steps 10 เปอร์เซ็นต์

6) Max Length = 510 สาเหตุที่ไม่สามารถใช้ 256 เหมือน BERT และต้องทำการเลือกใช้ 510 เพราะหลังการตัดคำภาษาไทยมีคำย่อยมากกว่าภาษาอังกฤษ

Warmup Steps คือการเพิ่มอัตราการเรียนรู้ด้วยอัตราคงที่ในช่วงแรกเพื่อให้ช่วงแรกแบบจำลองสามารถเรียนรู้ได้มากแล้วจึงตามมาด้วย Decay Steps ในช่วงท้ายเพื่อลดอัตราการเรียนรู้เพื่อให้แบบจำลองสามารถหาจุดที่ต่ำที่สุด (Global Minimum)

ตารางที่ 4.2 เปรียบเทียบแบบจำลอง LSTM balance by class weight กับแบบจำลอง WangchanBERTa ในเรื่องการ์ดจอ เวลา พารามิเตอร์ ความแม่นยำ และค่าเฉลี่ยมาโคร

แบบจำลอง (Model)	การ์ดจอ (GPU)	เวลา (Time)	พารามิเตอร์ (Parameter)	ความแม่นยำ (Accuracy)	ค่าเฉลี่ยมาโคร (Macro-Average)
LSTM balance by class weight	M2	14 นาที	1,497,558 ตัว	88.64 %	76.14 %
WangchanBERTa	V100	22 นาที	105,261,334 ตัว	95.00 %	87.00 %

จากตารางที่ 4.2 แบบจำลอง WangchanBERTa ได้ค่าความแม่นยำ 95.00 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 87.00 เปอร์เซ็นต์ โดยมีค่าพารามิเตอร์ทั้งหมด 105,261,334 ตัว โดยเมื่อเทียบกับ LSTM balance by class weight ได้ค่าความแม่นยำ 88.64 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 76.14 เปอร์เซ็นต์ โดยมีค่าพารามิเตอร์ทั้งหมด 1,497,558 ตัว จึงเห็นว่าได้ความแม่นยำ และค่าเฉลี่ยมาโครที่ดีกว่าโดยค่าความแม่นยำดีกว่า 6.36 เปอร์เซ็นต์ และค่าเฉลี่ยมาโครดีกว่า 10.86 เปอร์เซ็นต์

ในการศึกษาแบบจำลองทำให้รู้ว่าแบบจำลอง WangchanBERTa นั้นมีประสิทธิภาพมากกว่าแบบจำลอง LSTM เพราะแบบจำลอง WangchanBERTa เป็นแบบจำลองที่ได้รับกระบวนการฝึกมาแล้ว หรือเรียกว่า การเรียนรู้แบบถ่ายโอน (TL) เพราะแบบจำลองเคยได้รับการฝึกฝนเรียนรู้ภาษาไทยด้วยชุดข้อมูล Assorted Thai Texts ขนาด 78.5 กิกะไบต์ ล่วงหน้าเป็นจำนวนมากพร้อมทั้งยังมีการเพิ่มโทเค็นพิเศษคือ ตัวช่องว่าง <_> ซึ่งภาษาไทยไม่มีสัญลักษณ์การจบประโยคเหมือนภาษาอังกฤษ ภาษาไทยจึงมีการเว้นช่องว่างในการจบประโยคแทน

นอกจากนี้ยังมีการทำแบบจำลองภาษาที่แมสก์แล้ว (MLM) เป็นการเพิ่มสิ่งรบกวนลงไปในข้อความซึ่งเป็นการเสริมข้อมูลด้วยวิธีการรบกวน คล้ายกับข้อมูลที่เป็นรูปภาพแมวที่มีการหมุน ย่อ ขยาย ให้ภาพเดียวเกิดเป็นหลายภาพซึ่งทำให้แบบจำลองมีความทนทานต่อข้อมูลที่ไม่เคยเจอมาก่อนและมีประสิทธิภาพต่อความแม่นยำเพิ่มมากขึ้น

ส่วนต่อมาเป็นการแก้ปัญหานิวรอนบางส่วนตายเพราะ ReLU ส่งสัญญาณเป็น 0 ในกรณีเมื่อน้ำหนักผลรวมมีค่าเป็นลบ จึงเลือกใช้แอ็กทิเวชันฟังก์ชัน GELU แทน ReLU ทำให้จัดการข้อมูลน้ำหนักผลรวมมีค่าเป็นลบ ได้ดีกว่า

ถัดมาเป็น เซลฟ์แอตเทนชันมีความสามารถโดดเด่นในการจับความสัมพันธ์ส่วนต่าง ๆ ของลำดับคำ ซึ่งทำให้เข้าใจไวยากรณ์ระหว่างคำได้ดีขึ้น สามารถใช้งานได้หลายครั้งอย่างอิสระ โดยการเลือกคำที่สนใจข้อมูลที่สำคัญ เช่น คำที่สำคัญในประโยค แล้วเพิ่มค่าเวทคำนั้นให้มากขึ้น ซึ่งทำให้สามารถจดจำจำนวนคำที่ถูกตัดคำแล้วได้มากกว่าแบบจำลอง LSTM ทำให้ประสิทธิภาพของแบบจำลองดีขึ้นอย่างมาก

สุดท้ายคือขนาดความใหญ่ของแบบจำลอง WangchanBERTa มีขนาดจำนวนพารามิเตอร์ 105,261,334 ตัว ส่วนในแบบจำลอง LSTM มีจำนวนพารามิเตอร์ 1,497,558 ตัว ซึ่งมีขนาดความต่างของพารามิเตอร์กันถึง 70 เท่า

จากการศึกษาเรื่องการทำความสะอาดข้อมูลสังเกตว่ามีการทำความสะอาด และการประมวลผลข้อมูลล่วงหน้าไม่เหมือนกันเพราะแบบจำลอง LSTM นั้นเป็นแบบจำลองที่เริ่มต้นเรียนรู้ข้อมูลตั้งแต่ต้นจึงมีการใช้คลังคำศัพท์ PythaiNLP และคำฟุ่มเฟือยใน PythaiNLP เป็นคลังคำศัพท์ตั้งต้น พร้อมทั้งเพิ่มคำที่กำหนดเองลงในคลังคำศัพท์ที่คิดว่ามีผลต่อความสัมพันธ์ของคำและบริบทการทำนายหมวดหมู่ได้ดีขึ้น และเพิ่มคำฟุ่มเฟือยที่กำหนดเองลงในคลังคำศัพท์ฟุ่มเฟือยที่ไม่ได้มีส่วนช่วยเพิ่มความสัมพันธ์ของคำและบริบทการทำนายหมวดหมู่ แต่ในแบบจำลอง WangchanBERTa นั้นเป็นแบบจำลองแบบการเรียนรู้แบบถ่ายโอน (TL) ซึ่งได้รับการฝึกฝนเรียนรู้ภาษาไทยด้วยชุดข้อมูล Assorted Thai Texts ขนาด 78.5 กิกะไบต์ จึงไม่จำเป็นต้องใช้คลังคำศัพท์ PythaiNLP และมีการทำความสะอาดเหมือนกับข้อมูลเก่าที่ผู้สร้างแบบจำลองเคยทำความสะอาดมาก่อนจึงมีความแตกต่างกันของวิธีการทำความสะอาดข้อมูล

จำนวนพารามิเตอร์ที่มีขนาดใหญ่ของแบบจำลอง WangchanBERTa มีข้อเสียคือใช้ทรัพยากรในการฝึกฝนแบบจำลองมากกว่า LSTM เพื่อแลกกับประสิทธิภาพการทำนายที่แม่นยำมากขึ้น ถ้าต้องนำแบบจำลองไปใช้ในอุปกรณ์ที่มีหน่วยความจำน้อยอาจทำให้ไม่สามารถนำแบบจำลองไปใช้งานได้ และต้องเตรียมทรัพยากรอย่างการ์ดจอที่มีประสิทธิภาพในการประมวลผลสูง เพื่อให้ใช้ลดเวลาในการฝึกฝนลงหรือลดเวลาในการฝึกฝนปรับเพิ่มข้อมูลชุดใหม่ และสามารถปรับแต่งพารามิเตอร์อย่างละเอียดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดได้ และจำนวนแรมการ์ดจอเพื่อเพิ่มจำนวนข้อมูลในแต่ละแบตช์ได้มากขึ้น ซึ่งไม่ได้รวมวิธีการเหล่านี้ในการทดลองครั้งนี้ด้วย

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากที่ได้นำข้อมูลการร้องเรียน มาสร้างแบบจำลองทำนาย 22 ประเภทงาน โดยใช้แบบจำลอง LSTM และแบบจำลอง WangchanBERTa ถัดมาได้นำข้อมูลมาทำความสะอาด เพื่อนำมาเข้าแบบจำลอง LSTM แบบ Unbalance ซึ่งยังไม่ได้แก้ไขความไม่สมดุลของคลาส และ LSTM แบบ Class Weight ได้มีการจัดการความไม่สมดุลของคลาสด้วยวิธี น้ำหนักตัวอย่าง (Sample Weights) ผลความแม่นยำ และค่าเฉลี่ยมาโครมีความใกล้เคียงกันแต่การเรียนรู้การถ่ายโอนของแบบจำลอง WangchanBERTa เป็นการสร้างมาจากแบบจำลองภาษาเดียว (Mono-Lingual Language Model) คือภาษาไทยโดยเฉพาะ ทำให้ได้ค่าผลความแม่นยำ 95.00 เปอร์เซ็นต์ และค่าเฉลี่ยมาโคร 87.00 เปอร์เซ็นต์ ดีกว่าแบบจำลอง LSTM แต่ต้องใช้ทรัพยากรของการ์ดจอที่มากกว่าในการปรับแต่งอย่างละเอียด และได้สร้างเว็บไซต์อย่างง่ายเพื่อให้ผู้ที่สนใจ และประชาชนที่เกี่ยวข้องมีความสะดวกต่อการใช้งานสามารถเข้ามาทดลองใช้งานได้

5.2 ข้อเสนอแนะ

1) ทดสอบชุดข้อมูลแบบอื่นเพื่อดูว่าแบบจำลอง WangchanBERTa นั้นสามารถทำผลความแม่นยำ และค่าเฉลี่ยมาโครได้มากกว่าโครงข่ายระบบประสาทแบบย้อนกลับ (RNN) เช่น Gated Recurrent Unit (GRU)

2) ทดสอบแบบจำลองที่พัฒนามาจากแบบจำลองภาษา Transformer เช่น BERT ว่าแบบจำลองภาษาเดียวสามารถทำงานได้ดีกว่าแบบจำลองหลากหลายภาษา

3) ทดสอบการทำความสะอาดในแบบจำลอง LSTM ด้วยวิธีการทำความสะอาดเหมือนกับแบบจำลอง WangchanBERTa

5.3 ข้อจำกัด

1) เนื่องจากแบบจำลอง WangchanBERTa เป็นแบบจำลองภาษาขนาดใหญ่ (LLM) ทำให้มีข้อจำกัดในการนำไปใช้งานในอุปกรณ์ที่มีข้อจำกัดด้านหน่วยความจำ และแบบจำลองใช้ทรัพยากรในการฝึกสอนที่มากกว่าแบบจำลอง LSTM

2) ข้อมูลไม่เพียงพอต่อการฝึกสอนเนื่องจากในหมวดหมู่บางประเภทงานที่มีจำนวนข้อมูลน้อยทำให้ไม่สามารถเป็นตัวแทนข้อมูลของประชากรทั้งหมดได้

เอกสารอ้างอิง

1. Aroonmanakun, W. 2561. การตัดคำภาษาไทย : ความเป็นมา. [Online]. Available : https://awirote.medium.com/การตัดคำภาษาไทย-ความเป็นมา-97fbb2bb4897#:~:text=ความเป็นมา-,การตัดคำภาษาไทยเป็นงานพื้นฐานของ%20Thai,พยางค์มากกว่าตัดคำ%20เช่น
2. Ferrone, L. and Zanzotto, F. M. 2020. Symbolic, Distributed, and Distributional Representations for Natural Language Processing in the Era of Deep Learning: A Survey. [Online]. Available : https://www.frontiersin.org/articles/10.3389/frobt.2019.00153/full
3. Wikipedia contributors. 2024. WordNet. [Online]. Available : https://en.wikipedia.org/wiki/WordNet
4. Rakhmanberdieva, N. 2018. Word Representation in Natural Language Processing Part I. [Online]. Available : https://towardsdatascience.com/word-representation-in-natural-language-processing-part-i-e4cd54fed3d4
5. Rakhmanberdieva, N. 2018. Word Representation in Natural Language Processing Part II. [Online]. Available : https://towardsdatascience.com/word-representation-in-natural-language-processing-part-ii-1aee2094e08a
6. Wikipedia contributors. 2024. FastText. [Online]. Available : https://en.wikipedia.org/wiki/FastText
7. Kumararatnam, K. 2019. Word Representations in Natural Language Processing. [Online]. Available : https://medium.com/tech-sauce/word-representations-in-natural-language-processing-73f0cad0a02
8. Wikipedia contributors. 2021. Deep learning. [Online]. Available : https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1054241808
9. Wikipedia contributors. 2024. Recurrent neural network. [Online]. Available : https://en.wikipedia.org/wiki/Recurrent_neural_network
10. Amidi, A. and Amidi, S. 2023. Recurrent Neural Networks cheatsheet. [Online]. Available : https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

11. Zivkovic S. 2020. #003 RNN – Architectural Types of Different Recurrent Neural Networks. [Online]. Available : https://datahacker.rs/003-rnn-architectural-types-of-different-recurrent-neural-networks/
12. Wikipedia contributors. 2021. Long short-term memory. [Online]. Available : https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1054851044
13. SyncedReview. 2017. A Brief Overview of Attention Mechanism. [Online]. Available : https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129
14. Calixto, I. Stein, D. Matusov, E. Lohar, P. Castilho, S. and Way, A. 2017. Using images to improve machine-translating e-commerce product listings. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers (pp. 637-643). Valencia, Spain: Association for Computational Linguistics.
15. Morgan A. 2023. Explainable AI: Visualizing Attention in Transformers. [Online]. Available : https://www.comet.com/site/blog/explainable-ai-for-transformers/
16. Ameisen, E. 2020. Building machine learning powered applications. O'Reilly Media.
17. Huyen, C. 2022. Designing machine learning systems. O'Reilly Media.
18. วิทยา ปัญญา และวฤษาย์ ร่มสายหยุด. 2565. “วิธีการสร้างแบบจำลองเชิงทำนายพฤติกรรมการผิดเงื่อนไขการปล่อยชั่วคราวของศาลจากชุดข้อมูลที่ไม่สมดุลโดยใช้เทคนิคการเรียนรู้ของเครื่อง”. วารสารวิทยาศาสตร์ และเทคโนโลยี มหาวิทยาลัยมหาสารคาม. 42(2) : 7-17.
19. Nakwijit P. 2563. ทำความเข้าใจ Optimizer. [Online]. Available : https://medium.com/@chameleontk/ทำความเข้าใจ-optimizer-a44455615c32
20. Baheti P. 2021. Activation Functions in Neural Networks [12 Types & Use Cases]. [Online]. Available : https://www.v7labs.com/blog/neural-networks-activation-functions
21. Keshav. 2022. Leaky ReLU Activation Function [with python code]. [Online]. Available : https://vidyasheela.com/post/leaky-relu-activation-function-with-python-code

22. Géron, A. 2019. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O'Reilly Media.
23. Vaswani, A. Shazeer, N. Parmar, N. Uszkoreit, J. Jones, L. Gomez, A. N. Kaiser, Ł. and Polosukhin, I. 2017. Attention is all you need. In Advances in neural information processing systems, pp. 5998–6008.
24. Adaloglou, N. 2021. Why multi-head self attention works: math, intuitions and 10+1 hidden insights. [Online]. Available : https://theaisummer.com/self-attention/
25. Devlin, J. Chang, M. Lee, K. and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Vol. 1, pp. 4171-4186).
26. Press, E. 2024. The Data-Driven Decision Making Blog. [Online]. Available : https://d3mlabs.de/?p=1169
27. Muller, B. 2022. BERT 101 State Of The Art NLP Model Explained. [Online]. Available : https://huggingface.co/blog/bert-101
28. ResearchGate. 2023. Figure 3 - available via license: Creative Commons Attribution 4.0 International. [Online]. Available : https://www.researchgate.net/figure/Comparison-of-the-ReLu-and-GeLu-activation-functions-ReLu-is-simpler-to-compute-but_fig3_370116538
29. Wang, A. Singh, A. Michael, J. Hill, F. Levy, O. and Bowman, S. R. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, pp. 353–355.
30. Liu, Y. Ott, M. Goyal, N. Du, J. Joshi, M. Chen, D. Levy, O. Lewis, M. Zettlemoyer, L. and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.
31. Lowphansirikul, L. Polpanumas, C. Jantrakulchai, N. and Nutanong, S. 2021. WangchanBERTa: Pretraining transformer-based Thai Language Models. arXiv preprint arXiv:2101.09635.

32. Akkaradamrongrat, S. Kachamas, P. and Sinthupinyo, S. 2019. Text Generation for Imbalanced Text Classification.
33. Piyaphakdeesakun, C. Facundes, N. and Polvichai, J. 2019. Thai Comments Sentiment Analysis on Social Networks with Deep Learning Approach.
34. Harnmetta, P. and Samanchuen, T. 2022. Sentiment Analysis of Thai Stock Reviews Using Transformer Models.
35. Khamphakdee, N. and Seresangtakul, P. 2023. An Efficient Deep Learning for Thai Sentiment Analysis.
36. Noiyoo, N. and Thutkawkornpin, J. 2023. A Comparison of Machine Learning and Neural Network Algorithms for An Automated Thai Essay Quality Checking.
37. it24hrs. 2566. วิธีติดตั้ง TensorFlow บนเครื่อง Mac ชิป M1, M2. [Online]. Available : https://www.it24hrs.com/2023/how-to-install-tensorflow-on-m1-m2-mac/
38. Medium of VISTEC-depa AI Research Institute of Thailand. 2564. WangchanBERTa โมเดลประมวลผลภาษาไทยที่ใหญ่และก้าวหน้าที่สุดในขณะนี้. [Online]. Available : https://medium.com/airesearch-in-th/wangchanberta-โมเดลประมวลผลภาษาไทยที่ใหญ่และก้าวหน้าที่สุดในขณะนี้-d920c27cd433