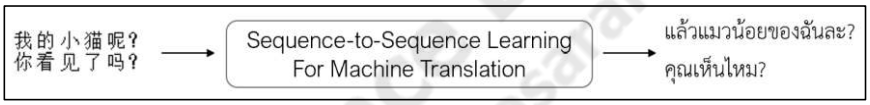
บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ความเป็นมาของโมเดลแบบทรานฟอร์มเมอร์ (History of Transformer-based Model)

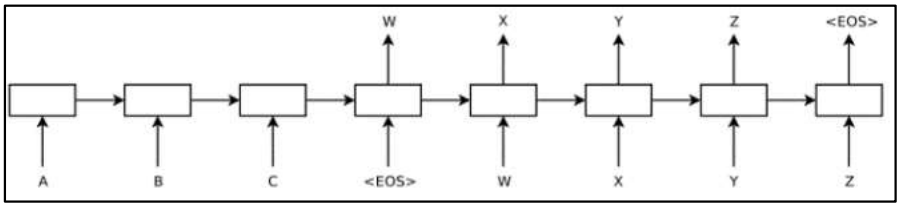
ประเภทของงานที่ประยุกต์ใช้อัลกอริทึมการเรียนรู้ของเครื่อง (Machine Learning: ML) อาจจะแบ่งตามลักษณะของข้อมูลอินพุตที่เข้ามาและเอาต์พุตที่ได้ ซึ่งโดยทั่วไปก็อาจจะเป็นงานด้านการจำแนกข้อมูล (Classification) หรือรีเกรสชัน (Regression) แต่กรณีที่ข้อมูลอินพุตเป็น Sequence แล้วให้เอาต์พุตเป็นอีก Sequence ในลักษณะนี้เรียกว่า Sequence-to-Sequence Learning ซึ่งในการประมวลผลในงานด้าน Machine Translation ก็จัดเป็นงานในลักษณะ Sequence-to-Sequence Learning และนับว่าเป็นสิ่งที่ผู้คนให้ความสำคัญมากตั้งแต่ยุคสมัยแรกๆ ของ AI มาจนถึงปัจจุบัน



ภาพประกอบที่ 2.1 ตัวอย่างการทำ Machine Translation เพื่อแปลภาษาจีนเป็นภาษาไทย

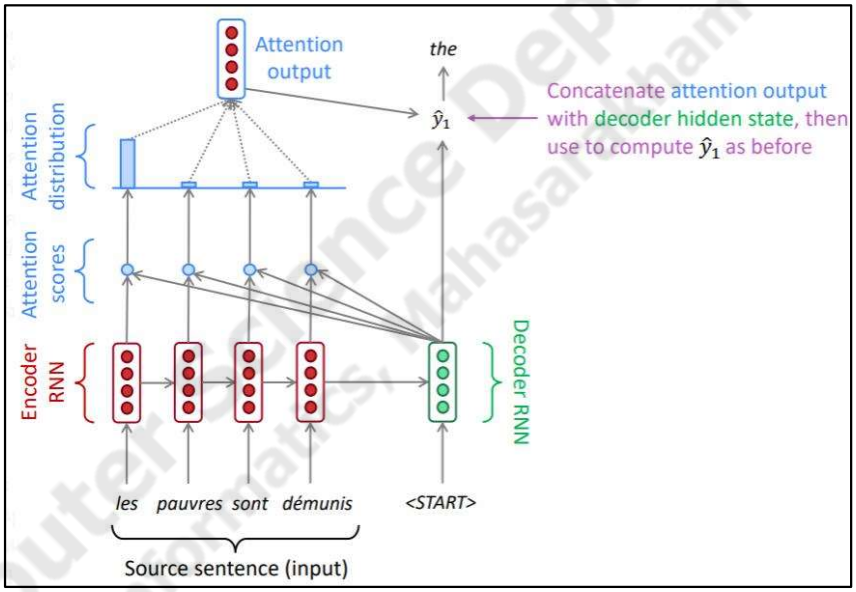
ที่มา: https://medium.com/mena-ai/

ปัจจุบันวิธีที่เป็นมาตรฐานสำหรับทำงานแบบ Sequence-to-Sequence Learning ก็คือ Sequence-to-Sequence Model (seq2seq) หรือเรียกอีกชื่อว่าโมเดล RNN Encoder–Decoder ซึ่งโมเดลนี้จะมีองค์ประกอบหลักอยู่ 2 ส่วน เรียกว่า Encoder กับ Decoder โดยส่วนของ Encoder จะรับอินพุตเข้ามาทีละหน่วยผ่านทาง RNN และเก็บสะสม Information ที่จำเป็นไว้ จากนั้นจะผ่าน Information นี้ไปยังส่วนของ Decoder ซึ่งก็จะเป็น RNN อีกตัวนึงที่ใช้ผลิตเอาต์พุตออกมาทีละหน่วย สำหรับการผลิตเอาต์พุตก็พิจารณาจาก Information ที่ได้รับมาและเอาต์พุตตัวก่อนหน้า ซึ่งแผนภาพของโมเดล seq2seq จะแสดงได้ดังภาพประกอบที่ 2.2



ภาพประกอบที่ 2.2 แผนภาพของโมเดล seq2seq

อย่างไรก็ดี โมเดล seq2seq จะมีปัญหาคอขวดเกิดขึ้น นั่นคือการส่ง Information เป็นทอดๆ ที่เป็นสายยาวๆ เช่นนี้ ก็อาจจะทำให้มี Information ที่จำเป็นบางอย่างสูญหายไประหว่างทางได้ ยกตัวอย่างเช่น จากรูปข้างบน สมมติว่าเป็นงาน Machine Translation ที่แปลจากภาษาไทยไปเป็นภาษาอังกฤษ โดย “A B C” คือคำว่า “ฉัน เลี้ยง แมว” ตามลำดับ และ “W X Y Z” เป็นคำว่า “I have a cat” ตามลำดับ จะเห็นว่าเอาต์พุตของคำว่า “cat” จะขึ้นอยู่กับอินพุตคำว่า “แมว” โดยตรง แต่ข้อมูลจาก C กว่าจะส่งมาถึง Z ต้องผ่านตัวกลางหลายทอด ทำให้อาจจะสูญหายได้ โดยเฉพาะอย่างยิ่งถ้าเป็นประโยคยาวๆ ดังนั้นจึงเกิดแนวคิดที่ว่า น่าจะดีกว่าถ้าเราให้กระบวนการของการสร้างเอาต์พุตสามารถโฟกัสไปที่อินพุตส่วนใดส่วนหนึ่งของประโยคได้โดยตรง และนี่คือที่มาของ “Attention” นั่นเอง ซึ่งภาพประกอบที่ 2.3 แสดงการทำ Attention สำหรับโมเดล seq2seq



ภาพประกอบที่ 2.3 การทำ Attention สำหรับโมเดล seq2seq

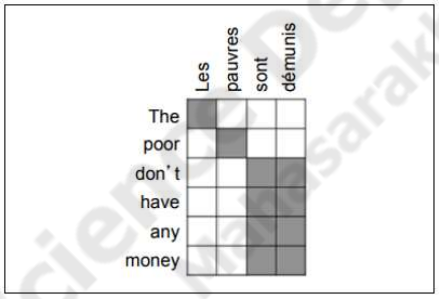
ที่มา: https://medium.com/mena-ai/

จากภาพประกอบที่ 2.3 เมื่อต้องการจะคำนวณเอาต์พุตที่ตำแหน่งใดตำแหน่งหนึ่ง ก็จะนำเวกเตอร์ของ Decoder (Q) ณ ตำแหน่งนั้น มาใช้หา Attention Score กับเวกเตอร์ของ Encoder (p) ในทุกๆ ตำแหน่ง ซึ่งถ้า Attention Score ที่ Encoder ตำแหน่งใดมีค่าสูงมากที่สุด ก็หมายความว่าเราจะให้ความสำคัญหรือใส่ใจกับตำแหน่งนั้นมาก ซึ่งการคำนวณค่านี้ก็ทำได้หลายวิธี โดยวิธีที่ง่ายที่สุดก็คือการทำ dot product ระหว่างเวกเตอร์ p กับเวกเตอร์ q เลย ซึ่งหมายความว่าเรากำลังจะใส่ใจกับตำแหน่งที่มีค่าของเวกเตอร์ p ใกล้เคียงกับค่าของเวกเตอร์ q และเมื่อได้ค่า Attention Score ออกมาแล้ว ก็จะเอาเข้าฟังก์ชัน SoftMax เพื่อแปลงให้เป็นค่าความน่าจะเป็น ซึ่งค่านี้จะเปรียบเสมือนค่า

น้ำหนัก (Weight) สำหรับเวกเตอร์ p ต่างๆ จากนั้นก็จะทำการหาค่าน้ำหนัก (Weight Average) ของเวกเตอร์ p ทั้งหมด ออกมาเป็นเวกเตอร์เดียว สมมติคือเวกเตอร์ r เพื่อนำไปใช้ในการคำนวณเอาต์ที่ควรจะได้ต่อไป ซึ่งที่เขียนมาทั้งหมดสามารถสรุปได้เป็นสมการดังนี้

$$r = \sum_i \frac{e^{p_i \cdot q}}{\sum_j e^{p_j \cdot q}} \, p_i \qquad 2.1$$

สำหรับ Attention นี้ นอกจากจะแก้ปัญหาคอขวดดังที่กล่าวมาแล้ว ยังสามารถแก้ปัญหา Vanishing Gradient (ปัญหาจากการประมวลผลข้อมูลแบบลำดับที่มียาวมากๆ ทำให้ประสิทธิภาพของอัลกอริทึมในตระกูล RNN ลดลง) ไปด้วยพร้อมกัน นอกจากนั้นถ้าพิจารณาลองดูว่าเอาต์พุตในตำแหน่งต่างๆ นั้น ให้ความใส่ใจกับอินพุตที่ตำแหน่งใด ก็เท่ากับว่าจะได้การจัดตำแหน่งของคำ (Word Alignment) มาด้วย ซึ่งการจัดตำแหน่งของคำในกรรมวิธี Machine Translation แบบดั้งเดิม ที่ไม่ใช่ Neural Machine Translation ก็จัดว่าเป็นปัญหาสำคัญเช่นกัน



ภาพประกอบที่ 2.4 ตัวอย่างการจัดตำแหน่งของคำที่ได้จาก Attention

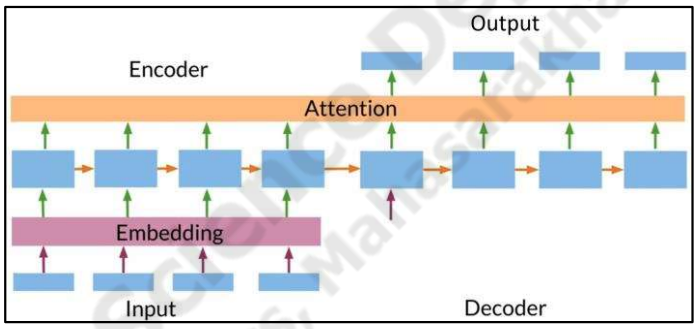
ที่มา: https://medium.com/mena-ai/

ซึ่งเทคนิค Attention ได้ถูกนำเสนอในงานวิจัยเรื่อง “Attention Is All You Need” [7] เมื่อปี ค.ศ. 2017 โดยนักวิจัยจาก Google โดยในงานวิจัยนี้กล่าวถึง “Attention Mechanism” และอธิบายถึงการสร้างโมเดลแบบโครงข่ายประสาทเทียมขึ้นมาอีก 1 ตัว ที่เรียกว่า “ทรานสฟอร์มเมอร์ (Transformer)” ซึ่งใช้ Attention Mechanism เป็นเครื่องมือหลักของโมเดล ใจความสำคัญในงานวิจัยนี้ที่ได้กล่าวไว้ได้แก่

การเรียนรู้เชิงลึก (Deep Learning) ในปัจจุบันนั้น อัลกอริทึมที่ใช้ส่วนใหญ่เป็นโครงข่ายประสาทเทียมแบบวนกลับ (Recurrent Neural Network: RNN) หรือโครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network: CNN) สำหรับ RNN ส่วนใหญ่จะใช้ในงานด้านการประมวลผลภาษา ในขณะที่ CNN จะใช้ในงานด้านคอมพิวเตอร์วิทัศน์ (Computer Vision) ซึ่งงานวิจัยนี้เล็งเห็นทั้งจุดอ่อนของทั้ง RNN และ CNN นั่นคืออัลกอริทึมเหล่านี้มักจะประสบปัญหาเรื่อง Vanishing Gradient อันเนื่องมาจากการประมวลผลข้อมูลแบบลำดับที่มียาวมากๆ ทำให้ประสิทธิภาพของอัลกอริทึมใน

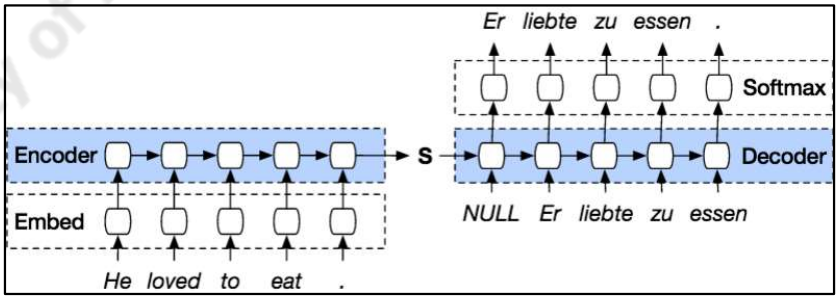
ตระกูล RNN จะลดลง โดยเฉพาะงานใน NLP แล้วหันมาใช้สิ่งที่เรียกว่า “Attention Mechanism” เพียงอย่างเดียว โดยมีการขยายความสามารถของ Attention ขึ้นไป แล้วเรียกสถาปัตยกรรมใหม่นี้ว่า “ทรานสฟอร์มเมอร์ (Transformer)”

หัวใจหลักของทรานสฟอร์มเมอร์คือกระบวนการที่เรียกว่า Self-Attention โดยกระบวนการนี้นอกจากจะเป็นสิ่งที่ทดแทน RNN และ CNN ได้แล้ว ยังแสดงถึงความข้องเกี่ยวกันของคำต่างๆ ในข้อความ ทำให้สามารถแก้ปัญหา Coreference Resolution ได้อย่างน่าสนใจ Coreference Resolution คือปัญหาของการหานิพจน์ทั้งหมดที่อ้างถึงสิ่ง ๆ เดียวกันในข้อความที่กำหนด ยกตัวอย่าง เช่น “*It was founded by the Romans, who named it Londinium.*” ซึ่งเมื่อมนุษย์อ่านประโยคนี้ ก็จะสามารถเข้าใจได้อย่างง่ายดายว่า “*It*” หมายถึง “*Londinium*” เป้าหมายของการแก้ปัญหา Coreference Resolution คือหาคำทั้งหมดที่อ้างอิงถึงเอนทิตีเดียวกันนั่นเอง ซึ่งการแก้ปัญหานี้มีความสำคัญอย่างมากต่องาน NLP หลายประเภท เช่น Machine Translation



ภาพประกอบที่ 2.5 ภาพรวมของสถาปัตยกรรมแบบทรานสฟอร์มเมอร์

ที่มา: https://www.youtube.com/watch?v=EFkbT-1VGTQ



ภาพประกอบที่ 2.6 ตัวอย่างการใช้งานโมเดลแบบทรานสฟอร์มเมอร์ใน Machine Translator

ที่มา: https://iq.opengenus.org/transformer-network-replace-gans/

นอกจากนี้ในงานวิจัยนี้ยังแสดงให้เห็นว่าทรานสฟอร์มเมอร์ให้ผลการทดลองในชุดข้อมูลมาตรฐานที่เหนือกว่าวิธีการอื่นอย่างชัดเจน ตัวอย่างเช่นในงาน Machine Translation ซึ่งใช้ชุดข้อมูล newstest2014 ที่ทำการแปลจากภาษาอังกฤษเป็นภาษาเยอรมัน และภาษาอังกฤษเป็นภาษาฝรั่งเศส แล้ววัดผลโดยใช้ค่า BLEU (Bilingual Evaluation Understudy) ซึ่งเป็นค่าที่แสดงความแตกต่างระหว่าง Automatic Translation และ Human-created Reference และผลการเปรียบเทียบจะเป็นดังตารางที่ 2.1

ตารางที่ 2.1 เปรียบเทียบประสิทธิภาพของโมเดล Transformer กับโมเดล state-of-the-art

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [18]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [16]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [19]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [20]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + posUnk Ensemble [18]		40.4		$8.0 \cdot 10^{20}$
GNMK + RL Ensemble [16]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [19]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

หมายเหตุ – EN = English, DE = German, FR = French

จากตารางที่ 2.1 จะเห็นว่าภาษาอังกฤษเป็นภาษาฝรั่งเศสจะดีกว่าการแปลภาษาอังกฤษเป็นภาษาเยอรมัน นั่นเป็นเพราะว่าข้อมูลของภาษาฝรั่งเศสมีมากกว่าหลายเท่า และภาษาเยอรมันจะมีโครงสร้างของคำ (Morphology) ที่ซับซ้อนกว่านั่นเอง อย่างไรก็ตามทราสฟอร์มเมอร์สามารถเพิ่มประสิทธิภาพของการแปลภาษาอังกฤษเป็นภาษาเยอรมันได้มากกว่าวิธีอื่น

ภาพประกอบที่ 2.7 ตัวอย่างการใช้งานโมเดลแบบทรานสฟอร์มเมอร์ใน Google Translate

2.2 ประเภทของโมเดลทรานฟอร์มเมอร์

โมเดลทรานฟอร์มเมอร์สามารถแบ่งออกได้เป็น 4 ประเภท คือ

2.2.1 Encoder-only (ในเพจ Huggingface เรียก Autoencoder Models) [21]

โมเดลทรานฟอร์มเมอร์ที่มี Encoder เพียงอย่างเดียวนั้นจะเหมาะกับปัญหาพื้นฐาน เช่น การจำแนกข้อมูล (Classification) การวิเคราะห์ถดถอย (Regression) การถาม-ตอบแบบระบบปิด (Closed-domain Question-Answering) ตัวอย่างโมเดลทรานฟอร์มเมอร์ในกลุ่มนี้ เช่น BERT, Roberta, XLM-Roberta, Longformer, และ DPR เป็นต้น

2.2.2 Decoder-only (ในเพจ Huggingface เรียก Autoregressive Models) [21]

โมเดลทรานฟอร์มเมอร์ที่มี Decoder เพียงอย่างเดียวก็จะเหมาะกับการงานด้าน Story Generation ตัวอย่างโมเดลทรานฟอร์มเมอร์ในกลุ่มนี้ เช่น GPT, GPT-2 และ GPT-3 รวมทั้ง BertGeneration กลุ่มนี้จะไม่ได้สร้างเอาต์พุตจากอินพุตที่เจาะจงโดยตรง เพราะไม่มีส่วนของ Encoder นั่นเอง

2.2.3 Encoder-Decoder (หรือ Sequence-to-Sequence Models) [21]

โมเดลทรานฟอร์มเมอร์ที่มี Encoder-Decoder ซึ่งจะสร้างเอาต์พุตจากอินพุตจะเหมาะกับปัญหาทางด้านการแปลภาษา (Language Translation) การตอบคำถามแบบทั่วไป (Free-form Question-Answering) หรือการย่อความ (Text Summarization) ถึงแม้ว่าเอาต์พุตของโมเดลกลุ่มนี้จะขึ้นกับอินพุตโดยตรง แต่เอาต์พุตก็จะมีความ "อิสระ" หรือ "สร้างสรรค์" ได้ เช่นในการตอบคำถามแบบทั่วไป เมื่อเจอคำถามประเภท "ทำไม?" หรือ "อย่างไร?" โมเดลจะสามารถอธิบายและให้แง่คิดเกี่ยวกับคำถามได้อย่างอิสระ ตัวอย่างโมเดลทรานฟอร์มเมอร์ในกลุ่มนี้ เช่น T5, Bart และ RAG

2.2.4 Multi-modal Encoder-Decoder [21]

โมเดลทรานฟอร์มเมอร์ในกลุ่ม Multi-modal encoder-decoder เป็นกลุ่มพิเศษซึ่งเพิ่งเริ่มเข้ามาใน Huggingface ปี 2020 นี่เอง นั่นคืออินพุตนอกจากจะเป็น sequential data ได้แล้ว ยังสามารถรับอินพุตที่เป็นรูปภาพ (Image) ได้ด้วย เช่นปัญหา Image Captioning ที่โมเดลทรานฟอร์มเมอร์สามารถ “เล่าเรื่องราว” จากรูปภาพได้เป็นต้น ตัวอย่างโมเดลทรานฟอร์มเมอร์ในกลุ่มนี้ เช่น LXMert

2.3 ความเป็นมาของ BERT

2.4.1 ปัญหาในการแทนคำหรือการแทนเอกสารข้อความ (Problem in Word or Text Representation)

ปัญหาหลักของ Word หรือ Text Representation ที่พบใน One Hot Encoding, Bag of Word (BOW), tf-idf, และ Word Embedding คือไม่สามารถแสดงโครงสร้างและความหมายของคำได้

One Hot Encoding [8] - ในการประมวลผลทางด้านภาษา เราจะเริ่มจากการตัดคำ (Tokenization) แล้วแปลงคำให้เป็นตำแหน่งของคำในพจนานุกรม (Dictionary) เช่น ประโยคที่ว่า “I really love my dog.” ก็จะเปลี่ยนเป็น Word Sequence คือ “4 <OOV> 2 1 3” โดย OOV คือ Out of Vocabulary มักจะถูกแทนด้วยศูนย์ (0) จากนั้นทำการแปลงเป็นเวกเตอร์ One Hot โดย 1 คำ ก็คือ 1 แถว ดังแสดงในภาพประกอบที่ 2.8

	I	Really	Love	My	dog
I	1	0	0	0	0
Really	0	1	0	0	0
Love	0	0	1	0	0
My	0	0	0	1	0
dog	0	0	0	0	1

ภาพประกอบที่ 2.8 แสดงตัวอย่าง One Hot Vector ของประโยค “I really love my dog.”

ความยาวของแถวก็เท่ากับจำนวนคำที่มีในประโยค จากนั้นก็จะเอาไปสร้างโมเดล เช่น LSTM หรือ GRU ปัญหาคือจำนวน dimension มีจำนวนมาก แต่เวกเตอร์ที่ได้มี 1 ไม่มาก ขณะที่ 0 มีเป็นจำนวนมากกระจายในเวกเตอร์ ซึ่งปัญหานี้เรียกว่า Sparse Matrix ทำให้ยากต่อการสร้าง Pattern เพราะว่าข้อมูลกระจายตัวมากเกินไป จึงจำเป็นต้องทำ Dimensional Reduction ก่อนเอาไปสร้างโมเดล นอกจากนั้น One Hot Vector ยังไม่สามารถแสดงโครงสร้างและความหมายของคำได้

Bag of Word (*BOW*) [8] – BOW เป็นโมเดลในรูปแบบเวกเตอร์ที่ใช้กันแพร่หลายในงานด้านการจำแนกเอกสารข้อความ ซึ่งในโมเดลของ BOW จะเป็นกลุ่มของคำที่สกัดมาจากคลังเอกสาร โดยไม่ได้คำนึงถึงหลักไวยากรณ์ และลำดับของคำ โดยคำในเวกเตอร์จะใช้เป็นฟีเจอร์สำหรับใช้ในการเรียนรู้ตัวจำแนกเอกสารข้อความ (Text Classifier) หากคำๆ นั้นพบในเอกสารก็จะแสดงค่าของคำนั้นเป็นจำนวนครั้งที่พบ ในขณะที่หากคำๆ นั้นไม่พบในเอกสารก็จะแสดงค่าของคำนั้นเป็น 0 ตัวอย่างของ BOW สามารถแสดงได้ดังภาพประกอบที่ 2.9

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

ภาพประกอบที่ 2.9 ตัวอย่าง BOW

ที่มา: https://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

อย่างไรก็ตาม BOW ก็ยังมีข้อเสีย นั่นคือ BOW ไม่ได้แสดงตำแหน่งของ “คำ” ในเอกสาร ซึ่งตำแหน่งของ “คำ” ในเอกสารจะมีความสำคัญมาก เพราะ “คำ” ที่อยู่ในตำแหน่งที่แตกต่างอาจจะมีความหมายไม่เหมือนกัน นอกจากนั้น แม้ว่าจะมีการเข้ารหัส “คำ” โดยการนับจำนวนครั้งของการพบคำๆ นั้นในเอกสาร วิธีการดังกล่าวไม่ได้ช่วยให้สามารถแยกแยะ “คำทั่วไป (Common Words)” กับ “คำเฉพาะ (Specific Words)”

Term Frequency - Inverse Document Frequency (*tf-idf*) [9] – *tf-idf* เป็นวิธีเวกเตอร์ของการนับคำ (Count Vectorization) ในการคัดแยกคำตามความสำคัญโดยการให้น้ำหนักคำในแต่ละคำด้วยการพิจารณาจาก 2 ปัจจัยคือ *tf* และ *idf* ซึ่งแนวคิดของ *tf* คือถ้าหาก “คำ” ไหนถูกพูดถึงอยู่บ่อยๆ ในเอกสารนั้นๆ จะมีความเป็นไปได้สูงว่าคำนั้นมีความสำคัญกับเอกสารนั้นๆ ในขณะที่ *idf* คือเป็นการคำนวณค่าน้ำหนักเพื่อแสดงความสำคัญของแต่ละ “คำ” โดยคำที่พบเจอได้บ่อยในเอกสารหลายเอกสารที่อยู่ในคลังเอกสาร หาก “คำ” ใดมีค่า *idf* ต่ำ แสดงว่า “คำ” นั้นจะไม่สามารถดึงเอาจุดเด่นของเอกสารที่มี “คำ” นั้นอยู่ออกมาได้ดี โดยการคำนวณค่า *tf-idf* ของคำในเอกสารสามารถแสดงได้ดังต่อไปนี้

$$tf(w,d) = \frac{freq(w,d)}{total\ of\ words\ containing\ in\ a\ document} \quad 2.2$$

$$idf(w,corpus) = log\left(1 + \frac{total\ documents\ in\ the\ corpus}{total\ of\ documents\ containing\ word\ w}\right) \quad 2.3$$

$$tf - idf\ (w) = tf(w,d) \times idf(w) \quad 2.4$$

โดยตัวอย่างของ *tf-idf* สามารถแสดงได้ดังภาพประกอบที่ 2.10 ข้อดีของ *tf-idf* คือวิธีการนี้สามารถกรองคำที่ไม่เกี่ยวข้องและรักษาคำที่มีนัยสำคัญเอาไว้ได้ อย่างไรก็ตาม *tf-idf* ก็มีพื้นฐานมาจาก BOW ดังนั้น *tf-idf* จึงไม่สนใจตำแหน่งของคำ ความหมายของคำ และคำที่เกิดร่วมในเอกสารที่แตกต่าง

จากสาเหตุนี้ *tf-idf* จึงมีประโยชน์ต่อการวิเคราะห์เอกสารข้อความในระดับคำ (Lexical Features) แต่ไม่สามารถแสดงความหมายของคำเหมือนกับ Word Embedding

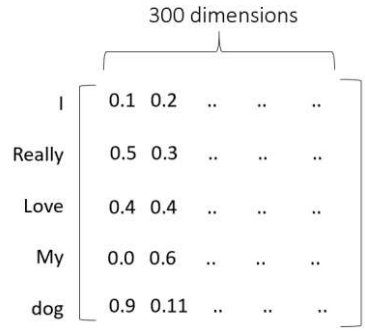
	key	tfidf_tmp	tf_7783482	idf
4256	พรรค	0.161791	0.069959	2.312661
3848	7	0.094287	0.028807	3.273123
6966	เรา	0.090792	0.028807	3.151762
254	นาย	0.086348	0.061728	1.398843
3801	จะ	0.085512	0.069959	1.222313
5490	ยึดมั่น	0.080911	0.016461	4.915351
773	หารือ	0.076366	0.020576	3.711378
6769	หัวหน้า	0.070824	0.028807	2.458615
1471	เจตนารมณ์	0.069241	0.012346	5.608498
3467	ภูมิธรรม	0.069241	0.012346	5.608498

ภาพประกอบที่ 2.10 ตัวอย่างโมเดล tf-idf

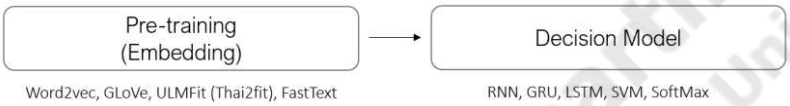
ที่มา: https://www.softnix.co.th/2019/05/28/tf-idf-ทำงานยังไง/

Word Embedding [10] – คือการแทนคำ (Word Representation) ที่เก็บลักษณะทางโครงสร้าง (Syntactics) และความหมาย (Semantics) ของคำด้วย สมมติกำหนดให้ “คำ” หนึ่งคำมีขนาดของ dimension เท่ากับ 300 นั่นคือ คำหนึ่งคำจะแทนด้วยตัวเลข 300 ตัว เช่น จากประโยค “I really love my dog.” ประโยคนี้มีคำอยู่ 5 คำ ก็จะได้เมทริกซ์ขนาด 5 x 300 (ดังภาพประกอบที่ 2.11) ซึ่งตัวเลขที่อยู่ในเมทริกซ์นั้นจะเป็นการแสดง Characteristic ของคำนั้นๆ โดยที่ไม่สนใจว่าประโยคนั้นจะคืออะไร และไม่สนใจลำดับของการสอน (Do not care sequence in Training) ซึ่งการสอนในลักษณะเช่นนี้ก็คือมีความเป็นอิสระจากบริบทของเอกสารนั่นเอง (Context Independence)

ปัจจุบัน Word Embedding มีหลายตัว เช่น Word2vec, GLoVe (Global Vectors for Word Representation), ULMFit (Thai2Fit) และ FastText ซึ่ง Word Embedding ก็คือ Pre-training ด้วยการ Embedding นั่นเอง เมื่อได้ผลลัพธ์จาก Word Embedding แล้ว ก็สามารถนำไปใช้ร่วมกับอัลกอริทึมอื่นๆ ได้ เช่น RNN, GRU, LSTM, SoftMax หรือ SVM ในขั้นของ Decision Model ได้ การที่จะเลือกใช้อัลกอริทึมตัวใดนั้นก็ขึ้นกับวัตถุประสงค์ของการทำงาน

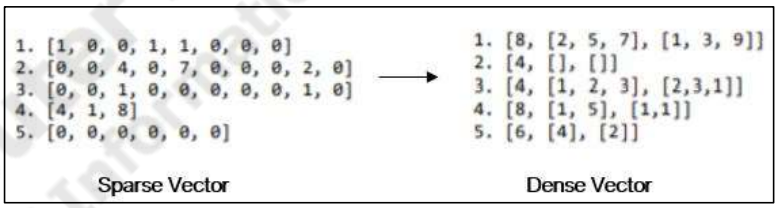


ภาพประกอบที่ 2.11 แสดงตัวอย่าง Word Embedding ของประโยค “I really love my dog.”



ภาพประกอบที่ 2.12 แสดงการใช้งาน Word Embedding

ในที่นี้จะกล่าวถึง Word2Vec ซึ่งเป็นโมเดลที่ พัฒนาโดยทีมนักวิจัยของ Google นำโดย Tomas Mikolov [10] ในปี ค.ศ. 2013 โดย Word2vec เป็นเทคนิคในการประมวลผลทางด้านภาษา เพื่อให้ได้ Language Model โดยเรียนรู้จากข้อมูลวิกิพีเดีย ซึ่ง โมเดลดังกล่าวจะที่ใช้ในการแปลง “คำ” ให้อยู่ในรูปแบบของเวกเตอร์ที่ไม่มีปัญหาเรื่องของ Sparse Vector เพราะจะเป็นลักษณะ Dense Vector ซึ่งตัวเลขในเวกเตอร์ไม่มี 0 มากจนเกินไป (ดังแสดงในภาพประกอบที่ 2.13)



ภาพประกอบที่ 2.13 ตัวอย่าง Sparse vector และ Dense Vector

ซึ่งเวกเตอร์ของคำ (Word Vector) ต่างๆ จะถูกคำนวณจากบริบทรอบข้างด้วยการใช้เทคนิคโครงข่ายประสาทเทียมแบบ Encoder-Decoder ที่มี 2 เลเยอร์ โดยจะมีหลักการในการเปรียบเทียบเวกเตอร์ทางความหมายของคำ 2 คำ แล้วคืนค่าออกมาเป็นตัวเลขตั้งแต่ -1 ถึง 1 เพื่อแสดงให้เห็นถึงความคล้ายคลึงหรือความสอดคล้องของคำทั้งสอง นั่นคือ Word2Vec จะมองว่าคำที่มีบริบทคล้ายกัน ควรเป็นคำที่มีความหมายคล้ายกันด้วย ซึ่งวิธีการนี้ทำให้สามารถทำให้การแทนเอกสารข้อความด้วย Word2Vec นอกจากจะแสดงคำที่เป็นฟีเจอร์ ยังสามารถแสดงความหมายของคำได้ด้วย เพียงแต่จะมีมิติของฟีเจอร์มากขึ้น

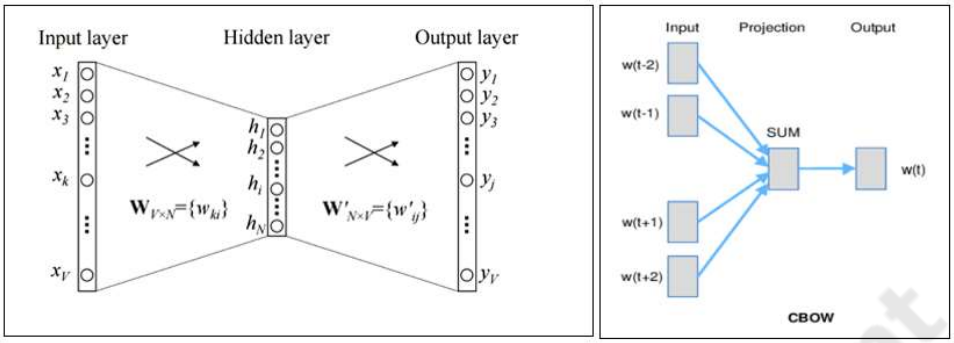
Word2Vec จะประกอบด้วยองค์ประกอบ 2 ส่วนคือ Continuous Bag of Words (CBOW) และ Skip-gram ซึ่งโมเดลเหล่านี้ก็ใช้โครงข่ายประสาทเทียมในการสร้างเพื่อให้ได้เวกเตอร์ของคำ (Word Vector) โดยโมเดลเหล่านี้จะใช้บริบทของเอกสารในการวิเคราะห์ความสัมพันธ์ของคำ นั่นคือ Word Embedding จะเรียนรู้ด้วยการพิจารณาคำที่รอบๆ หากพบว่ากลุ่มคำใดมีความใกล้เคียง (หรือเกิดร่วม) กับคำเดียวกันเสมอ กลุ่มคำเหล่านั้นก็จะมี Embedding ที่คล้ายกัน

เพื่อแสดงความเหมือนของคำกลุ่มหนึ่ง อันดับแรกจะต้องมีการกำหนด window-size เพื่อแสดงขอบเขตของบริบท (Context) ที่กำลังพิจารณา ตัวอย่างเช่น หากกำหนดให้ window-size = 2 สำหรับทุกคำ นั่นคือ Word2Vec จะพิจารณาคำที่ละ 2 คำ คือพิจารณา2 คำด้านซ้าย และ 2 คำด้านขวาของคำที่กำลังพิจารณา (ดังแสดงในภาพประกอบที่ 2.14)

ภาพประกอบที่ 2.14 การพิจารณาความเหมือนของคำด้วย Word2Vec ที่ใช้ window-size = 2

จากนั้นจะใช้หลักการของ Text Classification เพื่อเรียนรู้ค่าความน่าจะเป็นของคำหรือบริบทที่อยู่รอบๆ คำว่า “วิกฤต” ว่าลาเบลที่อยู่รอบๆ คำนี้ ซึ่งระหว่างการเรียนรู้ก็จะมีการปรับค่าต่างๆ จนกว่าจะเกิดความแม่นยำในการทำนาย ซึ่งโมเดลที่ใช้ในการทำนายจะใช้อัลกอริทึม Logistic Regression

สำหรับโมเดลแบบ CBOW นั้น เราจะต้องทำการสร้างโครงข่ายสมองแบบตื้น (Shallow Neural Network) โดยจะใช้งาน Word Vector เป็นชั้นอินพุต ซึ่งชั้นนี้จะเชื่อมต่อเข้ากับชั้นซ่อน (Hidden Layer) จำนวน 1 ชั้น โดยที่เราสามารถกำหนดจำนวนโหนด (Node) ในชั้นซ่อนนี้ได้ตามต้องการ แต่โดยปกติควรจะมีจำนวนโหนดน้อยกว่าจำนวนมิติของเวกเตอร์ที่เป็นอินพุต ซึ่งก็คือ Word Vector นั่นเอง ท้ายที่สุดเวกเตอร์ที่เป็นผลลัพธ์ของโมเดลแบบ CBOW ที่แสดงในชั้นเอาต์พุต ควรจะมีจำนวนมิติเท่ากับ Word Vector ของชั้นอินพุต โมเดลแบบ CBOW สามารถแสดงได้ดังภาพประกอบที่ 2.15



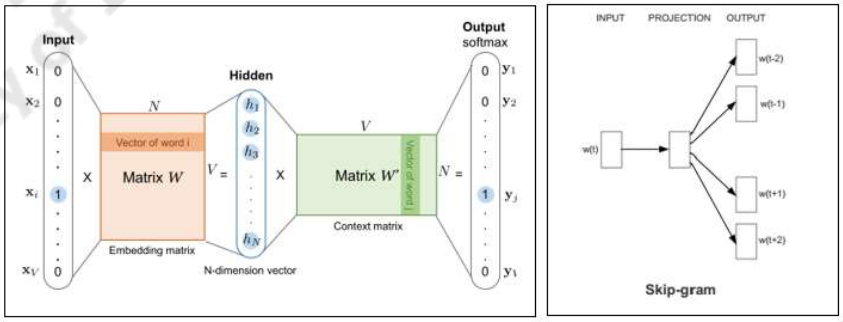
ภาพประกอบที่ 2.15 โมเดลแบบ CBOW

ที่มา: https://bigdata.go.th/big-data-101/word2vec/

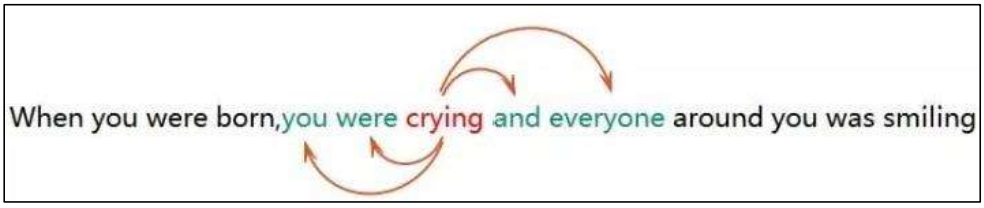
โมเดลแบบ CBOW จะใช้ในคำในบริบทเพื่อการทำนาย “คำ” สำคัญหรือคำที่สนใจที่เรียกว่า “Central Word” (แสดงได้ดังภาพประกอบที่ 2.16)

ภาพประกอบที่ 2.16 ตัวอย่างการทำนาย “คำ” จากบริบทรอบๆ ด้วยโมเดลแบบ CBOW

สำหรับโมเดลแบบ Skip-gram นั้น จะใช้คำสำคัญที่เป็น Central Word เพื่อทำนายบริบทรอบๆ นั่นคือ สำหรับแต่ละคำที่อยู่ในประโยคหรือข้อความที่นำมาพิจารณานั้น การฝึกฝนโมเดลนี้จะนำเอา Word Vectors ของคำนั้นมาใช้เป็นคำกลาง (Central Word) จากนั้นจะทำนายการกระจายตัวของคำ (Probability Distribution) ที่น่าจะเป็นบริบทของคำๆ นี้



ภาพประกอบที่ 2.17 โมเดลแบบ Skip-gram



ภาพประกอบที่ 2.18 ตัวอย่างการทำนายบริบทที่อยู่รอบๆ “คำ” ด้วยโมเดลแบบ Skip-gram

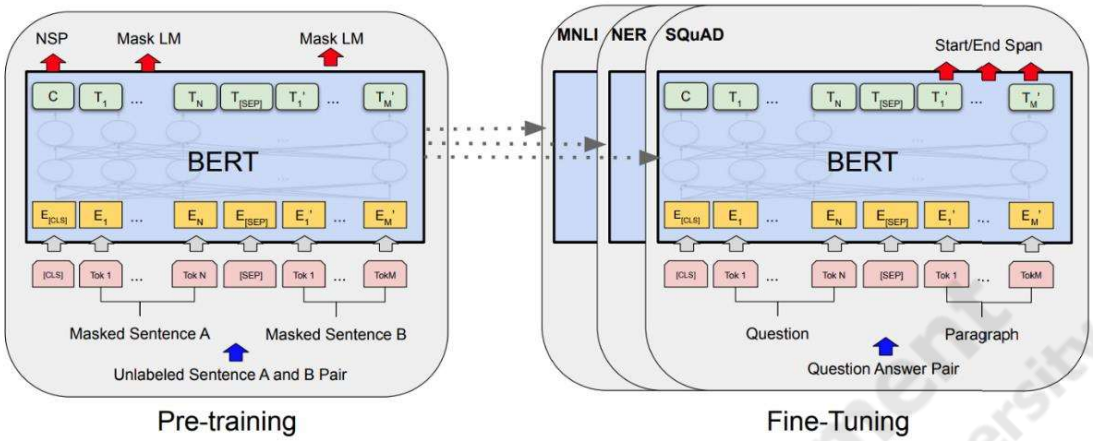
แม้ว่า Word Embedding แบบ Word2Vec จะให้ประสิทธิภาพที่ดีในการทำ Word Representation อย่างไรก็ตาม คำและเวกเตอร์เป็นความสัมพันธ์แบบหนึ่งต่อหนึ่งซึ่งไม่สามารถแก้ปัญหาคำที่มีหลายความหมาย (Polysemous Words) ได้ นอกจากนี้ Word2Vec ยังเป็นโมเดลแบบคงที่ (Static Model) แม้ว่าจะประยุกต์ใช้ในงานได้หลากหลาย แต่ Word2Vec ก็ไม่ง่ายที่จะปรับให้เหมาะสมโดยเฉพาะกับงานแบบเฉพาะได้

2.4.2 BERT (Bidirectional Encoder Representations from Transformer)

จากปัญหาของการแสดงเอกสารข้อความ (Text Representation) ที่อธิบายข้างต้น โดยเฉพาะในเรื่องของการพิจารณาตำแหน่งของคำ และการให้ความหมายของคำ จึงมีความพยายามที่จะพัฒนาโมเดลอื่นๆ เพื่อปรับปรุงประสิทธิภาพของงานด้าน NLP ซึ่งโมเดลที่ได้มีการนำเสนอขึ้นมาคือโมเดลแบบทรานสฟอร์มเมอร์ ซึ่งโมเดลที่เด่นๆ เช่น BERT ซึ่ง BERT คือ Google (AI) Algorithm ที่ในเวอร์ชันล่าสุดนี้มันได้พัฒนาบนพื้นฐานของเทคโนโลยี AI Neural Network เพื่อให้ระบบอัลกอริทึมเข้าใจภาษามนุษย์มากขึ้น

BERT เป็น Transfer Learning ที่ประกอบด้วย 3 ส่วนหลัก คือ Pre-training, Fine-tuning และ Decision Model โดย Pre-training สถาปัตยกรรมของ BERT จะเป็นโมเดลที่มีการใช้ Encoder อย่างเดียว แสดงได้ดังภาพประกอบที่ 2.19 และภาพประกอบที่ 2.20

ภาพประกอบที่ 2.19 กรอบการดำเนินงานใน BERT



ภาพประกอบที่ 2.20 Pre-training และ Fine-tuning ของ BERT

ที่มา: https://towardsdatascience.com/what-exactly-happens-when-we-fine-tune-bert-f5dc32885d76

ใน BERT คือการสร้างโมเดลด้วยข้อมูลเอกสารจำนวนมากๆ ซึ่งจะเป็นเอกสารอะไรก็ได้ (ข้อมูลไม่จำเป็นต้องมีคลาสลาเบล) เพื่อให้โมเดลเกิดความเข้าใจใน Language Model นั่นคือเข้าใจในลักษณะของภาษาหรือการใช้ภาษา จากนั้นเมื่อจะนำโมเดลที่ได้ไปใช้ในงานที่ต้องการก็ต้องทำ Fine-tuning ซึ่งก็คือ การนำข้อมูลที่เกี่ยวข้องกับงานที่ต้องการประมวลผลมาทำการปรับค่าในโมเดลที่ได้จาก Pre-training โดยข้อมูลที่นำมา Fine-tuning ควรเป็นข้อมูลที่มีคลาสลาเบล เพื่อใช้ในการปรับ weight เพื่อให้กลายเป็น Decision Model ที่ต่อยอดมาจาก BERT จากนั้นจะนำ Decision Model ที่ได้จาก BERT ไปใช้ร่วมกับอัลกอริทึมตัวอื่น เช่น Logistic Regression, SoftMax, หรือ Support Vector Machines (SVM) ในงานเฉพาะที่ต้องการประยุกต์ใช้ BERT

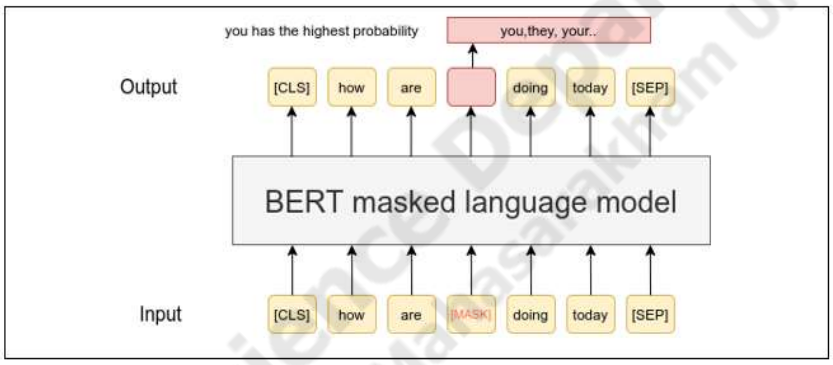
จากเทคนิคที่กล่าวมาก่อนหน้า เช่น One Hot Encoding, Bag of Word (BOW), tf-idf, และ Word Embedding จะเห็นว่าล้วนเป็นการใช้หน่วยคำในการแสดงฟีเจอร์ของเอกสาร ทำให้ขั้นตอนการตัดคำจะมีความหลากหลาย แต่ใน BERT จะใช้รูปแบบ “การตัดแบ่งหน่วยคำย่อย (Sub-word Level Tokenization)” ซึ่ง “sub-word” ที่ได้จะมีขนาดเล็กกว่า “คำ” และจะถูกใช้เป็นฟีเจอร์ เช่น คำว่า “management” อาจจะตัดคำได้เป็น “ma”, “na”, “ge”, “me” และ “nt” สาเหตุที่ทำเช่นนี้ก็เพราะต้องการลดจำนวนคำในพจนานุกรมและจะได้ไม่ต้องมาเรียนรู้ว่าคำในภาษาต่างๆ เป็นอย่างไร จากนั้นจะนับความถี่ของแต่ละตัวอักษรเพื่อพิจารณาว่าตัวอักษรใดอยู่คู่กันและเกิดบ่อย โดยจะเรียงค่าของการเกิดร่วมจากมากไปน้อย แล้วแสดง sub-word เหล่านี้ในรูปแบบของเวกเตอร์ วิธีการนี้เรียกว่า “Sub-word2Vec” ด้วยวิธีการเช่นนี้ การตัดคำของ BERT จึงไม่ขึ้นกับภาษา ดังนั้นไม่ว่าจะเป็นภาษาอะไร BERT ก็สามารถตัดคำในภาษานั้นได้ ซึ่ง BERT จะกำหนดจำนวน sub-word ไว้ที่ 512 sub-word

- *Pre-training of BERT*

ในการเรียนรู้โมเดลของภาษา (Language Model Learning) สมมติว่า การทำนาย “คำ” ในประโยค เช่น “The child came home from ___.” ในส่วนที่ละไว้ หลายๆ โมเดลจะทำนาย “คำ” ถัดไปในลำดับของประโยค (Sequence) ซึ่งวิธีการที่พิจารณาลำดับของประโยคแบบ Directional Model จึงเป็นวิธีการที่มักจะมีข้อจำกัดของการเรียนรู้จากบริบท (Context Learning) ซึ่ง BERT ได้ใช้กลยุทธ์ 2 อย่างในการแก้ปัญหานี้

1. Masked Language Model (MLM)

กรอบการดำเนินงานในส่วนของ MLM สามารถแสดงได้ดังภาพประกอบที่ 2.21



ภาพประกอบที่ 2.21 กรอบการดำเนินงานในส่วนของ Masked Language Model

ที่มา: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

จากภาพประกอบที่ 2.21 ก่อนที่จะนำ word sequence เข้าสู่ BERT จะมีการแทน “คำ” ในแต่ละ sequence ด้วยโทเค็น [MASK] เรียกว่า “Masked Word” จำนวน 15% หลังจากนั้นโมเดลจะทำการทำนายค่าดั้งเดิม (Original Value) ของ Masked Word ด้วยคำที่ไม่มี MASK ที่เรียกว่า “Non-masked Word” ซึ่งเป็นบริบทที่อยู่รอบ “Masked Word” ใน word sequence นั้นๆ สำหรับการทำนายคำที่เป็นเอาต์พุต จะมีการดำเนินการดังนี้

(1) เพิ่มเลเยอร์ของการจำแนกข้อมูลในส่วนบนของ Encoder ส่วนที่เป็นเอาต์พุต

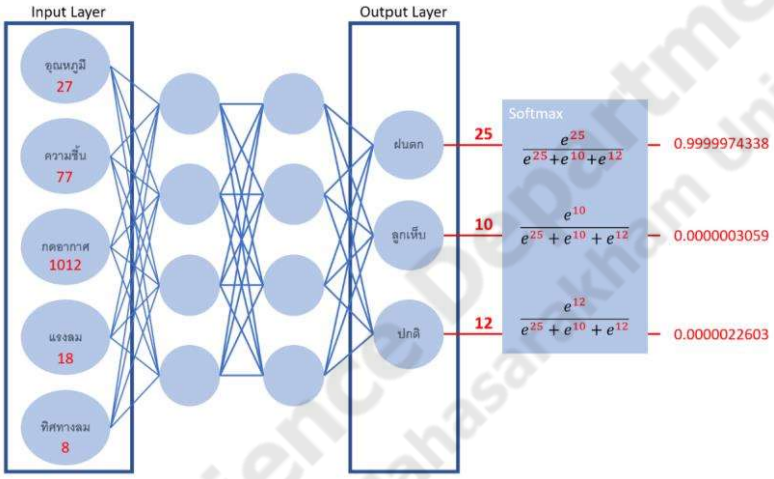
(2) นำเอาเวกเตอร์ที่เป็นเอาต์พุตมาคูณกับ Embedding Metrix จากนั้นแปลงผลลัพธ์ที่ได้ให้เป็นมิติของคำศัพท์ (Vocabulary Dimension) หรือเวกเตอร์ของคำ

(3) การคำนวณความน่าจะเป็นของแต่ละคำในเวกเตอร์ของคำด้วย SoftMax (SoftArgMax Function หรือ Normalized Exponential Function: σ) คือ ฟังก์ชันที่จะรับอินพุต

เป็นเวกเตอร์ของตัวเลขจำนวนจริง แล้วนอร์มอลไลซ์ (Normalize) ให้ออกมาเป็นความน่าจะเป็น (Probability) ที่ผลรวมที่ค่าเท่ากับ 1 โดยมีสมการคือ

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{K=1}^{K} e^{z_k}} \qquad 2.5$$

เมื่อ σ คือค่าผลลัพธ์ที่ได้จาก SoftMax ส่วน e = 2.71828 เสมอ และ z คือค่าที่ได้จากชั้นเอาต์พุต ซึ่งตัวอย่างการคำนวณค่าความน่าจะเป็นด้วยฟังก์ชัน SoftMax สามารถแสดงได้ดังภาพประกอบที่ 2.22



ภาพประกอบที่ 2.22 ตัวอย่างการคำนวณค่าความน่าจะเป็นด้วยฟังก์ชัน SoftMax

ที่มา: https://medium.com/super-ai-engineer/softmax-function-คืออะไร-eae1f1bbef63

BERT loss function จะพิจารณาเฉพาะการทำนายค่าของ “Masked Word” และจะละเว้นทำนายค่าของคำที่ไม่ปิดบัง “Non-masked Word” ด้วยเหตุนี้ การรวมโมเดลจึงค่อนข้างช้ากว่า Directional Model แต่ BERT จะมีการเรียนรู้ในบริบทเพิ่มขึ้น

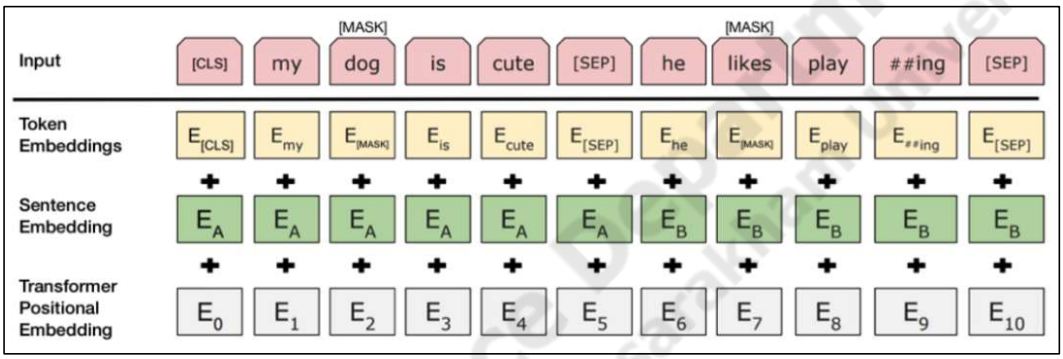
2. Next Sentence Prediction (NSP)

ในขั้นตอนการเรียนรู้ของ BERT โมเดลแบบ BERT จะรับประโยคเข้ามาเป็นคู่ และจะเรียนรู้เพื่อทำนาย ถ้าประโยคที่สองในคู่ประโยคที่รับเข้ามา เป็น subsequence ในเอกสารต้นฉบับ ระหว่างการเรียนรู้นั้น 50% ของข้อมูลอินพุตจะเป็นคู่ประโยค (Pairs of Sentences) โดยประโยคที่สองที่ตามหลังประโยคแรกต้องเป็น subsequent sentence ในเอกสารต้นฉบับ ในขณะที่ อีก 50% ของข้อมูลอินพุตที่เหลือ จะเป็น random sentence จากคลังเอกสารที่ถูกเลือกมาเป็นประโยคที่สอง ดังนั้นเพื่อช่วยให้โมเดลแยกแยะระหว่างสองประโยคในการเรียนรู้ได้ ข้อมูลอินพุตจะถูกประมวลผลด้วยวิธีต่อไปนี้ก่อนเข้าสู่โมเดล:

(1) ทำการแทรกโทเค็น [CLS] ที่จุดเริ่มต้นของประโยคแรก และแทรกโทเค็น [SEP] ที่ส่วนท้ายของแต่ละประโยค

(2) Sentence Embedding ของประโยค A หรือประโยค B จะถูกเพิ่มเข้าไปในแต่ละโทเค็น Sentence Embedding ของประโยคใดๆ ที่คล้ายคลึงกัน จะเป็นการ Embeddings ด้วยคำศัพท์จำนวน 2 คำ

(3) เพิ่ม Positional Embedding ในแต่ละโทเค็น เพื่อใช้ในการระบุตำแหน่ง (Position) ใน sequenceการดำเนินงานตามที่อธิบายไว้ข้างต้น สามารถแสดงดังภาพประกอบที่ 2.23



ภาพประกอบที่ 2.23 กรอบการทำงานใน Next Sentence Prediction

ที่มา: https://medium.com/super-ai-engineer/softmax-function-คืออะไร-eae1f1bbef63

ในการทำนายว่าประโยคที่สองเชื่อมต่อกับประโยคแรกจริงหรือไม่ ให้ดำเนินการตามขั้นตอนต่อไปนี้:

(1) นำอินพุตทั้งหมดเข้าสู่โมเดลทรานสฟอร์มเมอร์

(2) เอาต์พุตของโทเค็น [CLS] จะถูกแปลงเป็นเวกเตอร์ขนาด 2×1 โดยใช้เลเยอร์การจำแนกข้อมูลอย่างง่าย เมทริกซ์ที่ได้จากเรียนรู้จะแสดงค่าน้ำหนัก (Weight) และค่าเอนเอียง (Bias)

(3) การคำนวณความน่าจะเป็นของ IsNextSequence ด้วยฟังก์ชัน softmax

ในขั้นตอนของการเรียนรู้โมเดล BERT นั้น Masked LM และ Next Sentence Prediction จะได้รับการเรียนร่วมกัน โดยมีเป้าหมายในการลด Loss Function ที่รวมกันจากทั้งสองกลยุทธ์

- *Fine-tuning of BERT*

การใช้ BERT สำหรับงานเฉพาะนั้นค่อนข้างง่าย และ BERT สามารถใช้กับงานด้าน NLP ได้หลากหลาย ในการประยุกต์ใช้งานก็เพียงแต่เพิ่มเพิ่มเลเยอร์ขนาดเล็กลงในโมเดลหลักเท่านั้น

สำหรับงานด้านการจำแนกเอกสารข้อความ (Text Classification) เช่น การวิเคราะห์ความรู้สึก (Sentiment Analysis) ทำได้คล้ายกับ Next Sentence Classification ดังนั้นจะมีการเพิ่มเลเยอร์ของการจำแนกเอกสารที่ด้านบนของเอาต์พุตของทรานสฟอร์มเมอร์สำหรับโทเค็น [CLS]

สำหรับงานด้านการถาม-ตอบ (Question-Answering) นั้น โมเดลจะรับคำถามที่เป็น text sequence และต้องมีการใส่เครื่องหมายกำกับ (Mark) ในคำตอบที่อยู่ใน sequence โดย BERT จะถูกสอนด้วยการเรียนรู้เวกเตอร์ที่เพิ่มเข้ามา 2 ตัวที่มีการทำเครื่องหมายจุดเริ่มต้นและจุดสิ้นสุดของคำตอบ

สำหรับงานด้านการรู้จำคำนามเฉพาะ (Named Entity Recognition: NER) โมเดลจะรับ text sequence ที่มีการใส่เครื่องหมายกำกับ หรือ Mark สำหรับแสดงเอนติตี้ประเภทต่างๆ (Person, Organization, Date, และอื่นๆ) ที่ปรากฏอยู่ในข้อความ โดย BERT จะถูกสอนด้วย feeding เอาเวกเตอร์เอาต์พุตของแต่ละโทเค็นเข้าสู่เลเยอร์ของการจำแนกเอกสารที่ทำนายลาเบลของคำนามเฉพาะ

- *ประเภทของ BERT*

สถาปัตยกรรมของ BERT จะเป็นโมเดลที่มีการใช้ Encoder อย่างเดียว ซึ่ง BERT มี 2 ประเภทคือ (1) BERT Base และ (2) BERT Large

โดย BERT Base จะใช้พารามิเตอร์ทั้งหมด 110 ล้านพารามิเตอร์ ซึ่งใช้จำนวนเลเยอร์ทั้งสิ้น 12 เลเยอร์ ซึ่งจำนวนเลเยอร์ก็คือจำนวน Transformer Blocks ในขณะที่ขนาดของ Hidden คือ 768 และจำนวนของ Self-attention Heads เท่ากับ 12 ส่วน ในการประมวลผลต้องการ 1 GPU

สำหรับ BERT Large จะใช้พารามิเตอร์ทั้งหมด 340 ล้านพารามิเตอร์ ซึ่งใช้จำนวนเลเยอร์ทั้งสิ้น 24 เลเยอร์ ซึ่งจำนวนเลเยอร์ก็คือจำนวน Transformer Blocks ในขณะที่ขนาดของ Hidden คือ 1024 และจำนวนของ Self-attention Heads เท่ากับ 16 ในการประมวลผลต้องการ 1 TPU โดยทั่วไปแล้ว โมเดลที่ใหญ่กว่าก็จะให้ผลลัพธ์ที่ดีกว่า

2.4 เทคนิคสำหรับการประเมินโมเดล

โดยพื้นฐาน ตารางคอนฟิวชั่นเมทริกซ์จะใช้ในปัญหาของการจำแนกข้อมูลแบบ 2 กลุ่ม ซึ่งเป็นการแสดงผลลัพธ์การทำนายที่ได้จากโปรแกรมและเปรียบเทียบกับผลลัพธ์จริงที่ได้จากมนุษย์หรือตามความเป็นจริง ซึ่งตารางดังกล่าวสามารถใช้ในประเมินแนวทางที่ดีที่สุดในกระบวนการของการเรียนรู้เพื่อสร้างตัวจำแนกข้อมูล โดยตารางคอนฟิวชั่นเมทริกซ์สามารถแสดงได้ดังภาพประกอบที่ 2.24

		Predicted results: **yes**	Predicted results: **no**
Actual results	**yes**	$true\ positive\ (tp)$	$false\ negative\ (fn)$
	no	$false\ positive\ (fp)$	$true\ negative\ (tn)$

ภาพประกอบที่ 2.24 ตารางคอนฟิวชั่นเมทริกซ์

สำหรับความหมายของค่าในตารางคอนฟิวชั่นเมทริกซ์สามารถอธิบายได้ดังนี้

- True Positive (TP) คือ สิ่งที่ตัวจำแนกเอกสารข้อมูลทำนายว่า**จริง** และมัน**จริง**
- True Negative (TN) คือ สิ่งที่ตัวจำแนกเอกสารข้อมูลทำนายว่า**ไม่จริง** และมัน**ไม่จริง**
- False Positive (FP) คือ สิ่งที่ตัวจำแนกเอกสารข้อมูลทำนายว่า**จริง** แต่ว่ามัน**ไม่จริง**
- False Negative (FN) คือ สิ่งที่ตัวจำแนกเอกสารข้อมูลทำนายว่า**ไม่จริง** แต่มันเป็น**จริง**

ค่าความถูกต้อง (Accuracy: Acc) จะใช้ประเมินการทำนายกลุ่มหรือคลาสที่ถูกต้องจากจำนวนข้อมูลชุดทดสอบทั้งหมด ค่าความถูกต้องจะเป็นตัววัดที่นิยมใช้ในการจำแนกข้อมูล โดยเฉพาะในการประเมินเพื่อเลือกตัวจำแนกข้อมูลที่ดีที่สุดไปใช้งาน เพราะค่าความถูกต้องจะพิจารณาประสิทธิภาพของโมเดลจากอัตราส่วนของข้อมูลที่ทำนายกลุ่มที่ถูกต้องจากจำนวนข้อมูลทั้งหมดในข้อมูลชุดทดสอบ

$$Acc = \frac{tp + tn}{tp + fp + tn + fn} \qquad 2.6$$

ค่าความระลึก (Recall) หรืออัตราผลบวกจริง (True Positive Rate: TPR) คือ ค่าที่พิจารณาว่าข้อมูลที่มนุษย์หรือในความเป็นจริงบอกว่า “จริง” และเมื่อถูกพิจารณาด้วยโมเดลแล้ว โมเดลจะทำนายผล “จริง” ได้ถูกต้องและตรงกับคำตอบที่กระทำโดยมนุษย์หรือในความเป็นจริงมากน้อยเพียงใด

$$Recall, TPR = \frac{tp}{tp + fn} \qquad 2.7$$

ค่าความแม่นยำ (Precision) คือค่าของการทำนายกลุ่มข้อมูลที่โมเดลพิจารณาจาก จำนวนข้อมูลที่มีทำนายกลุ่มว่าเป็น “จริง” ทั้งหมดนั้น แท้ที่จริงแล้วถูกต้องมากน้อยเพียงใด เมื่อเปรียบเทียบกับจำนวนข้อมูลทั้งหมดในกลุ่มที่เป็น “จริง”

$$Precision = \frac{tp}{tp + fp} \qquad 2.8$$

ค่าเอฟ (F1, F-score) เป็นตัววัดที่เป็นการสมดุลค่า (Balanced Score) ระหว่างค่าความระลึกและค่าความแม่นยำด้วยค่าเฉลี่ยแบบฮาร์โมนิค (Harmonic Mean)

$$F1 = 2 * \frac{recall\ x\ precision}{recall\ +\ precision} \quad 2.9$$

2.5 Google Colaboratory (Google Colab)

Google Colab เป็นโฮสต์โปรแกรม Jupyter notebook บน Cloud ของ Google ชื่อเต็มคือ Google Colaboratory โดยใช้ภาษา Python3 เป็นภาษาหลักที่ใช้ในการเขียนและรันงานบน Google Colab นี้

Google Colab จะช่วยให้การสร้างโมเดลด้วย Machine Learning และ Deep Learning ทำได้สะดวกขึ้น เพราะบนการสร้างโมเดลเหล่านั้นจะต้องใช้คอมพิวเตอร์ที่มีความเร็วหรือมีประสิทธิภาพสูง และลดระยะเวลาในการประมวลผลของเครื่อง

Google Colab มีความเร็วหลายเท่าถ้าเทียบกับคอมพิวเตอร์ที่ใช้ทั่วไป ตัวอย่างเช่น คอมพิวเตอร์ทั่วไปจะใช้เวลาในการเรียนรู้โมเดล 1 ครั้ง ใช้เวลาประมาณ 2-3 ชั่วโมง แต่หากดำเนินการบน Google Colab จะใช้เวลาในการเรียนรู้โมเดล 1 ครั้ง จะใช้เวลาประมาณ 30-35 นาที

2.6 งานวิจัยที่เกี่ยวข้อง (Related Works)

เนื่องจาก BERT เป็นโมเดล Transformer แบบ Encoder only ที่ถูกนำไปวิจัยต่อยอดในงานทางด้าน Sentiment Analysis อย่างกว้างขวาง จึงได้ยกตัวอย่างงานวิจัยที่ศึกษาต่อยอด BERT ดังนี้

2.6.1 Fine-Tuning BERT for Sentiment Analysis of Vietnamese Reviews [10]

ในปี 2020 Nguyen และคณะ ได้ทำการวิจัยโดยใช้ BERT ในการทำ sentiment Analysis กับภาษาเวียดนาม โดยทดลองแบ่งการ Fine-tuning เป็นสองวิธี วิธีที่ 1 ใช้โทเค็น [CLS] เพิ่มไปที่ตำแหน่งเริ่มต้นของประโยค จากนั้นเวกเตอร์ output ของโทเค็นนี้จะถูกส่งผ่าน feed-forward neural network เพื่อจำแนกข้อความ วิธีที่ 2 ใช้ผลลัพธ์ที่ได้จาก BERT รวมไปถึงโทเค็น [CLS] มาสร้างเป็นเมทริกซ์ SEQ_LEN×h โดยที่ SEQ_LEN คือความยาวสูงสุดของลำดับอินพุต และ h คือความยาวของ hidden vectors จากนั้นนำ output matrix ที่ได้จากกระบวนการข้างต้นไปใช้เป็น input ไปยังโมเดลการจำแนกประเภท 3 โมเดล ได้แก่ LSTM, TextCNN, RCNN โดยพวกเขาใช้ชุดข้อมูลจากสองแหล่ง ชุดข้อมูลแรกคือ Ntc-sv2 เป็นชุดข้อมูลรีวิวร้านอาหารและอาหารบน Foody ประกอบด้วยชุดข้อมูลตัวอย่าง 50,000 ชุดเอกสาร label จะถูกกำหนดจากคะแนนเฉลี่ย (avg_score) โดย คะแนนที่สูงกว่า 8.5 เป็น positive และ คะแนนที่น้อยกว่า 5 เป็น Negative ชุดข้อมูลที่ 2 คือ Vreview เป็นชุดข้อมูลบทวิจารณ์ผลิตภัณฑ์บนเว็บไซต์อีคอมเมิร์ซต่างๆ และยังมีบทวิจารณ์บางส่วนจากที่เกี่ยวกับอาหารและร้านอาหารบน Foody4 label จะถูกกำหนดจากคะแนนเฉลี่ย (avg_score)

โดย คะแนนที่สูงกว่า 7.5 เป็น positive และ คะแนนที่น้อยกว่า 5 เป็น Negative จากการทดลองพบว่าในการทำ Sentiment Analysis กับภาษาเวียดนาม พบว่าโมเดล BERT-RCNN ได้ประสิทธิภาพที่ดีที่สุด โดยค่า F1 88.22% แม้โมเดลนี้จะได้ไม่ได้มีประสิทธิภาพสูงกว่าโมเดลอื่นมากนัก แต่สำหรับงานในอนาคต พวกเขาตั้งเป้าที่จะขยายวิธีการที่เสนอสำหรับการวิเคราะห์ความรู้สึกตามแง่มุมต่างๆ และอาจมีการทดลองที่กว้างขวางมากขึ้นในชุดข้อมูลต่างๆ

2.6.2 Comparative study of Twitter Sentiment on COVID - 19 Tweets [11]

ในปี 2021 Nair และคณะ กล่าวว่า เมื่อเร็ว ๆ นี้ จำนวนทวีตเกี่ยวกับ COVID-19 เพิ่มขึ้นในอัตราที่ไม่เคยมีมาก่อน โดยรวมถึงทวีตเชิงบวก เชิงลบ และเป็นกลางลักษณะทวีตที่หลากหลายนี้ดึงดูดให้นักวิจัยทำการวิเคราะห์ความรู้สึกและวิเคราะห์อารมณ์ที่หลากหลายของประชาชนจำนวนมากที่มีต่อโควิด-19 งานวิจัยนี้จึงได้นำเสนอการวิเคราะห์ความรู้สึกจากความคิดเห็นใน Twitter ที่มีต่อโควิด-19 โดยใช้เดลที่ 3 ตัวคือ Logistic Regression sentiment analysis, VADER sentiment analysis, BERT sentiment analysis ชุดข้อมูลที่ผู้วิจัยนำมาใช้คือ ความคิดเห็นที่มีต่อโควิด-19 จำนวน 20,000 ชุดเอกสาร โดยแบ่งออกเป็น 3 label ได้แก่ positive, neutral, negative แบ่งข้อมูลเป็นข้อมูลชุดสอน(train) ต่อชุดข้อมูลทดสอบ(Test) เป็น 75 : 25 จากการทดลองเปรียบเทียบผลลัพธ์จากทั้ง 3 โมเดลพบว่า BERT เป็นโมเดลที่มีค่า Accuracy สูงที่สุดคือ 92%

2.6.3 BERT-Based Stock Market Sentiment Analysis [12]

ในปี 2020 Lee และคณะ งานวัยจัยของพวกเขาคือการวิเคราะห์ความรู้สึกจากความคิดเห็นของผู้คนเกี่ยวของหุ้นในหุ้นสหรัฐจากเว็บไซต์ Stocktwits โดยใช้ BERT-based เป็น pre-trained และ Fine-tuned ด้วยชุดข้อมูลภาษาอังกฤษโดยใช้ 2 ชุดคือ Bullish และ bearish เป็นชุดข้อมูลที่มี label ที่บอกความรู้สึกของข้อความเป็น positive และ Negative แต่ชุดข้อมูลจาก Bullish มีขนาดใหญ่กว่า bearish อยู่มาก พวกเขาจึงสุ่มเลือกข้อมูลจาก Bullish มา 1.3 เท่า ของชุดข้อมูล bearish เพื่อหลีกเลี่ยงปัญหาการกระจายข้อมูลที่ไม่สมดุล จากนั้นแบ่งข้อมูลออกเป็นชุดข้อมูลสอน (Training) และ ชุดข้อมูลตรวจสอบ (Validation) อัตราส่วนอยู่ที่ 80 : 20 ตามลำดับ โดยผลลัพธ์ที่พวกเขาได้จากการทดลอง ได้ค่า accuracy : 87.3% สุดท้ายพวกเขากล่าวว่า แบบจำลอง BERT ได้รับผลลัพธ์ที่น่าทึ่งในการจำแนกความคิดเห็น สิ่งนี้จะช่วยในการศึกษาเกี่ยวกับเกี่ยวกับความสัมพันธ์ระหว่างโซเชียลมีเดียของหุ้นกับราคาหุ้นจริง

2.6.4 Sentiment Analysis of Reviews in Kazakh With Transfer Learning Techniques [13]

ในปี 2022 Nugumanova และคณะ ได้ศึกษาเกี่ยวกับการวิเคราะห์ความรู้สึกกับภาษา Kazakh โดยการทดลองของพวกเขาเลือกใช้การ transfer learning สองแบบคือ zero-shot learning และ fine-tuning สำหรับโมเดล Pre-training จากเขาเลือกใช้โมเดล BERT-base จาก HuggingFace's Transformers Library [14] มา 3 แบบ ได้แก่ NLPtown Savacy และ BERTurk ชุดข้อมูลที่ใช้ในการทดลองเป็นชุดข้อมูลที่รวบรวมจาก Facebook เว็บไซต์ร้องเรียนของผู้บริโภค zhalobi.kz เว็บไซต์แอปพลิเคชัน 2GIS ข้อมูลมี label แบ่งเป็นสองแบบคือ positive และ Negative แบ่งชุดข้อมูลออกเป็น 3 ส่วน ชุดข้อมูลทดสอบ (Training) ชุดข้อมูลทดสอบ (Testing) ชุดข้อมูลตรวจสอบ (Validation) แบ่งเป็น 30 : 30 : 100 ชุดข้อมูล ตามลำดับ จากการทดลองพวกเขาพบว่าค่า Accuracy สูงที่สุดคือ NLPtown ที่ทำการ transfer learning แบบ fine-tuning ได้ค่า Accuracy 0.73

2.6.5 Comparison of BERT Models and Machine Learning Methods for Sentiment Analysis on Turkish Tweets [15]

ในปี 2021 Guven ได้ศึกษาการวิเคราะห์ความคิดเห็นบนทวีตโดยใช้โมเดล BERT และ นอกจากนี้ ยังมีการเปรียบเทียบโมเดล BERT ที่ผ่านการฝึกอบรมและวิธีการเรียนรู้ของเครื่อง ในบรรดาวิธีการเรียนรู้ของเครื่อง Random Forest, Naive Bayes และ Logistic Regression นั้น Logistic Regression เป็นวิธีที่ประสบความสำเร็จมากที่สุดถึง 98.4% ในอีกด้านหนึ่ง โมเดล BERT มีความแม่นยำถึง 98.75% และแซงหน้าวิธีการเรียนรู้ของเครื่อง ผลเชิงบวกของแบบจำลอง BERT ในการวิเคราะห์ความรู้สึกได้แสดงให้เห็นในการศึกษานี้