รายงานสหกิจศึกษาฉบับสมบูรณ์

การจำแนกหมวดหมู่ของบทความโดยใช้โครงข่ายประสาทเทียม
Article Classification by Neural Network

นายวิศรุต กาวิดำ

ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

รายงานสหกิจศึกษาฉบับสมบูรณ์

การจำแนกหมวดหมู่ของบทความโดยใช้โครงข่ายประสาทเทียม

Article Classification by Neural Network

นายวิศรุต กาวิดำ

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

ชื่อโครงงานสหกิจศึกษา การจำแนกหมวดหมู่ของบทความโดยใช้โครงข่ายประสาทเทียม

ชื่อ - สกุล นักศึกษา นายวิศรุต กาวิดำ

คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์

ชื่อ - สกุล อาจารย์นิเทศ อาจารย์บัณฑิต พัสยา

อาจารย์จิระศักดิ์ สิทธิกร

ชื่อ - สกุล ผู้นิเทศงาน นายวิภาส สุตันตยาวลี

สถานประกอบการ บริษัท แบ็คยาร์ด จำกัด ประเทศไทย

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อสร้างเครื่องมือที่สามารถจำแนกบทความแล้วสามารถบอกได้ว่าบทความนั้น เป็นบทความที่ควรจะจัดอยู่ในหัวข้ออะไร โดยแบ่งตามหัวข้อที่ได้กำหนดไว้ แล้วจึงนำเครื่องมือนี้ไปจำแนกบทความใน Facebook Page ต่าง ๆ เพื่อที่จะบอกได้ว่าเนื้อหาของ Page นั้น ๆ พูดถึงข่าวเกี่ยวกับเรื่องอะไร แล้วนำผลลัพธ์ที่ได้ไปพัฒนาผลิตภัณฑ์หลักของบริษัทต่อไป

ทางผู้พัฒนาเลือกใช้กระบวนการทาง Deep Learning ซึ่งเป็นศาสตร์แขนงหนึ่งของ Machine Learning ที่ทำการจำลองระบบโครงข่ายเซลล์ประสาทในสมองมนุษย์ โดยใช้เครื่องมือที่ชื่อว่า PyTorch ในการสร้างโมเดลโครงข่ายประสาทเทียมขึ้นมา นอกจากเครื่องมือที่ใช้ในการสร้างโมเดลแล้วจำเป็นที่จะต้องเตรียมข้อมูลเพื่อให้เหมาะกับการนำไปเป็นอินพุตของโมเดล เพื่อให้คอมพิวเตอร์ได้เรียนรู้ได้อย่างถูกต้อง ทางผู้พัฒนาได้ใช้เครื่องการตัดคำภาษาไทยที่ชื่อว่า International Components for Unicode (ICU) ซึ่งเป็นผลิตภัณฑ์ของ IBM Corporation ที่ใช้ในการจัดการกับข้อมูลประเภท Unicode โดยผลลัพธ์ที่ได้จาก ICUจะเป็นคำในรูปแบบคำภาษาไทยที่มีความยาวของตัวอักษรในคำสั้นที่สุดและยังคงมีความหมายอยู่

Co-operative Title: Article Classification by Neural Network

Student Intern Name: Mr. Witsarut Kawidam

Faculty: Engineering **Department:** Computer Engineering

Advisor Name: Mr. Bundit Passaya

Mr. Jirasak Sittigorn

Mentor Name: Mr. Vipas Sutantayawalee

Company: Backyard Co., Ltd.

ABSTRACT

This product is intended to create a tool that can classify articles and tell what the article should be in the defined topic. Then use this tool to classify articles on Facebook Page to identify the content of the page is. Then use the outcome to develop the firm's main product.

The developer uses Deep Learning, a branch of Machine Learning, to simulate the neural network in the human brain by using a tool called PyTorch to create an artificial neural network. In addition to the tools which is used in this model, it is also necessary to provide the information that is appropriate to the input of the model so that the computer could learn correctly. The developer uses a Thai word segmentation tool called International Components for Unicode (ICU), a product of the IBM Corporation, is used to handle Unicode data. The ICU results are words in the Thai word format that have the shortest length of the letter in the word and still have meaning.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้เสร็จสมบูรณ์ได้ด้วยความช่วยเหลือจากหลายท่านทั้งในทางตรงและทางอ้อม ซึ่งจะสำเร็จไม่ได้หากปราศจากความช่วยเหลือจากบุคคลเหล่านี้

ขอขอบคุณ อาจารย์ผู้นิเทศงาน อาจารย์บัณฑิต พัสยา ซึ่งเป็นผู้ที่มานิเทศงาน ที่ให้โอกาสทำสหกิจศึกษา และช่วยให้คำแนะนำ ในการทำงาน การแก้ไขปัญหา และจุดบกพร่องของโครงงาน ทำให้โครงงานสมบูรณ์มากยิ่งขึ้น

ขอขอบคุณ อาจารย์จิระศักดิ์ สิทธิกร ซึ่งเป็นผู้ที่คอยดูแลและให้คำแนะนำในด้านของงานสหกิจศึกษาซึ่งทำให้การทำโครงงานสะดวกมากยิ่งขึ้น

ขอขอบคุณ นายวิภาส สุตันตยาวลี หัวหน้างานที่ให้คำปรึกษา แนะนำหลักการในการทำงานทั้งในเรื่องของวิชาการ และธุรกิจ อีกทั้งในส่วนของรูปแบบของงานที่เป็นมาตรฐานตามหลักการที่ถูกต้องอย่างสม่ำเสมอ ซึ่งทำให้ตัวผลิตภัณฑ์ชิ้นนี้สำเร็จลุล่วงอย่างมีคุณภาพ

ขอขอบคุณ นายกฤต นิรันดรกุลชัย ผู้ดูแลและคอยให้คำปรึกษา ในเรื่องของความรู้ทางด้านการเขียนโปรแกรม ทำให้ผลิตภัณฑ์ชิ้นนี้สำเร็จไปได้ด้วยดี

สุดท้ายนี้ขอขอบคุณ บิดา มารดา ครอบครัว เพื่อน ๆ ตลอดจนผู้เกี่ยวข้องที่ไม่ได้กล่าวนามทุกท่าน ที่เป็นกำลังใจ ให้การสนับสนุน และความช่วยเหลือในการทำโครงงานครั้งนี้จนสำเร็จลุล่วงไปได้

วิศรุต กาวิดำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

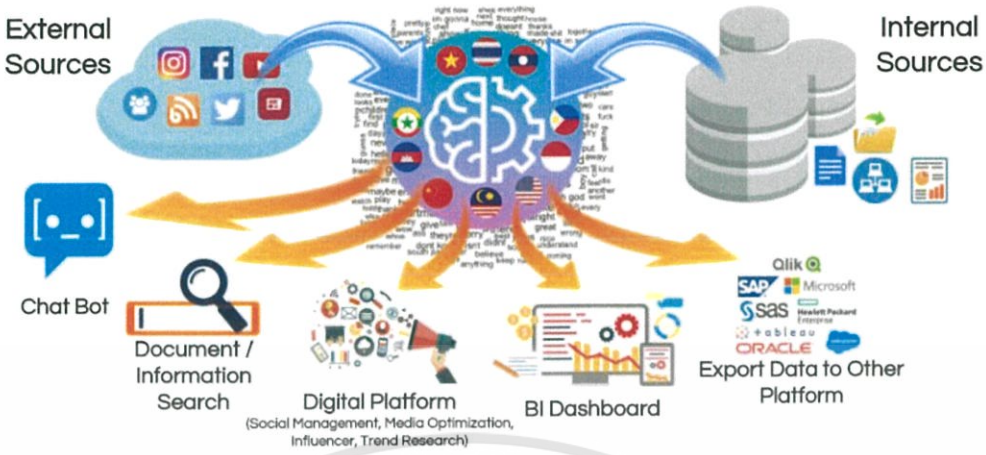
บทที่ 1
บทนำ

1.1. ความเป็นมาและความสำคัญ

จากผลการสำรวจพฤติกรรมผู้ใช้อินเทอร์เน็ตในปี 2560 โดยสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (สพธอ.) พบว่า คนไทยผู้ใช้อินเทอร์เน็ตต่อวันมากขึ้น เท่ากับ 6 ชั่วโมง 30 นาที และกิจกรรมที่นิยมทำเป็นอันดับ 1 เมื่อใช้งานอินเทอร์เน็ต คือ การใช้งานโซเชียลมีเดียสูงถึง 86.9% อีกทั้งยังเป็นครั้งแรกที่การซื้อขายสินค้าออนไลน์ติด 1 ใน 5 ของกิจกรรมที่นิยมทำเมื่อใช้งานอินเทอร์เน็ต แสดงให้เห็นถึงการยอมรับในการพาณิชย์อิเล็กทรอนิกส์ (e-commerce) มากขึ้นในสังคมไทย รวมถึงไลฟ์สไตล์ทางดิจิทัลต่าง ๆ มีการเปลี่ยนวิธีทำกิจกรรมที่เคยทำในแบบออฟไลน์มาเป็นออนไลน์มากขึ้น เช่น การจองหรือซื้อตั๋วออนไลน์ การจองห้องพัก สื่อบันเทิงต่าง ๆ เป็นต้น

จากที่กล่าวมาจะเห็นได้ว่า อินเทอร์เน็ตมีอิทธิพลมากต่อคนไทยในปัจจุบัน ไม่ว่าจะเป็นการแลกเปลี่ยนข้อมูลหรือสื่อสารกันผ่านทางโซเชียลมีเดีย การพาณิชย์อิเล็กทรอนิกส์ และไลฟ์สไตล์ ทั้งหมดนี้ต่างเกิดขึ้นบนโลกออนไลน์ทั้งสิ้น บริษัท แบ็คยาร์ด จำกัด ประเทศไทย ซึ่งเป็นบริษัทที่กระผมได้มีโอกาสไปฝึกงานและทำสหกิจศึกษาด้วย ได้เล็งเห็นความสำคัญในจุดนี้จึงได้พัฒนาระบบที่รวบรวมข้อมูลจากโซเชียลมีเดียหรือข้อมูลภายนอกองค์กร และข้อมูลภายในจากองค์กรที่เป็นลูกค้ามาสร้างความเชื่อมโยงแล้วนำไปสู่การวิเคราะห์ที่แม่นยำและมีประสิทธิภาพโดยใช้องค์ความรู้ทางเทคโนโลยีต่าง ๆ เข้ามาช่วย เช่น Big Data, Data Science, Natural Language Processing (NLP) และ Machine Learning เป็นต้น

โดยผลิตภัณฑ์และบริการของบริษัท คือ ออกแบบและติดตั้งระบบสำหรับรวบรวมและวิเคราะห์ Big Data, ให้คำปรึกษาและวิเคราะห์ข้อมูล Big Data, ให้คำปรึกษาและดำเนินการการประยุกต์ใช้ข้อมูล Big Data ผ่านช่องทางออนไลน์ ส่งผลให้เกิดผลประโยชน์ทางธุรกิจแก่ลูกค้า



ภาพที่ 1.1 ภาพรวมของระบบ

เนื่องจากบริษัทต้องการที่จะลดการทำงานแบบ Manual ของมนุษย์ลงด้วยการทำให้ระบบเป็นแบบ Automated มากขึ้นโดยการนำศาสตร์ของ Machine Learning เข้ามาช่วยและต้องการที่จะหาสื่อที่ตรงตามความต้องการของลูกค้าได้มากขึ้น จากการหาข้อมูลที่เหมือนกันระหว่างสิ่งที่สื่อกำลังพูดถึง และสิ่งที่ลูกค้าอยากจะสื่อพูดถึง ยกตัวอย่างเช่น ถ้าลูกค้าต้องการที่จะรู้ว่าสื่อต่าง ๆ บนโลกของโซเชียลมีเดีย นำเสนอข่าวหรือบทความประเภทไหนบ้างในช่วงนี้ สิ่งที่บริษัทต้องการ คือ จะมีวิธีไหนที่จะจำแนกบทความจำนวนมากเพื่อที่จะสามารถบอกแนวโน้มของโซเชียลมีเดียได้ว่า ณ เวลานั้น ๆ สื่อกำลังพูดถึงเรื่องอะไร หรือจะนำไปใช้ในอีกแง่มุมหนึ่ง เช่น กรณีที่ลูกค้าเป็นบริษัทเครื่องสำอาง แล้วอยากที่จะหา Influencer ที่พูดถึงเรื่องความสวยความงาม สิ่งที่บริษัทต้องทำ คือ อาจจะทำการลิสต์รายชื่อเพจต่าง ๆ ไม่ว่าจะเป็น Facebook, Pantip และอื่น ๆ อีกมากมาย แล้วดึงข้อมูลเหล่านั้นมาจำแนกว่า แหล่งข้อมูลไหนที่มีการพูดถึงเครื่องสำอางมากที่สุด พอรู้แล้วจึงทำการเจาะลึกลงไปอีกว่า เพจนั้นมี Keywords อะไรบ้างที่ถูกพูดถึงเยอะ ซึ่งเรียกว่า Share of Voice แล้วนำไปวิเคราะห์ต่อยอดในสิ่งที่ลูกค้าต้องการต่อ เช่น Influencer บางคนเป็นเซียนเครื่องสำอางและมีผู้ติดตามเยอะแล้วพูดถึงเรื่องลิปสติกเป็นส่วนใหญ่ หากลูกค้าอยากที่จะขายแป้งพับ ก็อาจจะไม่ตรงจุดที่จะจ้าง Influencer คนนี้ เป็นต้น จาก Requirements ดังกล่าว จึงเกิดเป็นผลิตภัณฑ์นี้ขึ้นมา นั่นคือ “เครื่องมือจำแนกหมวดหมู่ของบทความโดยใช้โครงข่ายประสาทเทียม”

1.2. วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อจำแนกบทความตามหัวข้อข่าวที่กำหนดได้อย่างถูกต้องและมีประสิทธิภาพ

1.2.2 เพื่อหาสื่อที่ตรงตามความต้องการของลูกค้าได้มากขึ้น จากการหาข้อมูลที่เหมือนกันระหว่างสิ่งที่สื่อกำลังกล่าวถึง และสิ่งที่ลูกค้าอยากจะให้สื่อกล่าวถึง

1.3. ขอบเขตของการวิจัย

เป็นเว็บแอพพลิเคชั่น (Web Application) ที่ทำให้ผู้ใช้สามารถจำแนกบทความต่าง ๆ ตามหัวข้อข่าวที่กำหนดไว้ โดยจะแบ่งระบบออกเป็น 2 ส่วนหลัก ๆ ดังนี้

1.3.1 โมเดลโครงข่ายประสาทเทียม (Neural Network Model)

1.3.2 เว็บแอพพลิเคชั่น (Web Application)

1.4. วิธีดำเนินการวิจัย

1.4.1 รับทราบความต้องการของผลิตภัณฑ์จากผู้นิเทศงาน

1.4.2 ศึกษาเครื่องมือต่าง ๆ ที่จำเป็นต่อการใช้งานในผลิตภัณฑ์

1.4.3 ออกแบบผลิตภัณฑ์

1.4.4 ลงมือเขียนโปรแกรม

1.4.5 ทดสอบผลิตภัณฑ์

1.4.6 นำผลิตภัณฑ์ที่ได้ไปนำเสนอผู้นิเทศงาน

1.4.7 แก้ไขงานหากผู้นิเทศงานไม่พึงพอใจ

1.4.8 จัดทำเล่มรายงาน

1.5. ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ผู้ใช้งานสามารถจำแนกบทความตามหัวข้อข่าวที่กำหนดได้อย่างถูกต้องและมีประสิทธิภาพ

1.5.2 ผู้ใช้งานสามารถหาสื่อที่ตรงตามความต้องการของลูกค้าได้มากขึ้น จากการหาข้อมูลเหมือนกันระหว่างสิ่งที่สื่อกำลังกล่าวถึง และสิ่งที่ลูกค้าอยากจะให้สื่อกล่าวถึง

บทที่ 2
แนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง

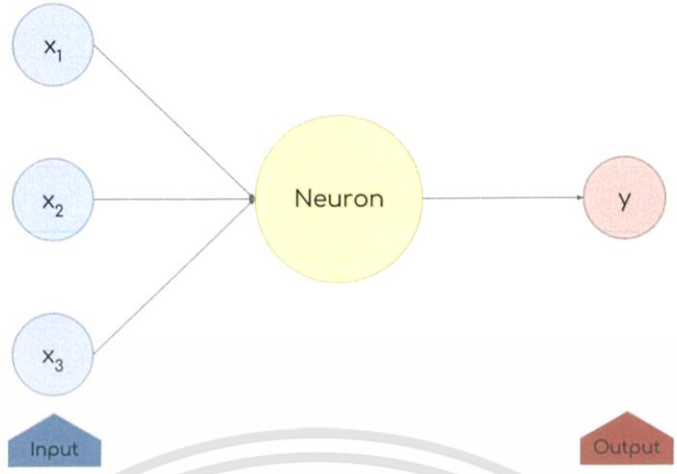
2.1. ทฤษฎีที่เกี่ยวข้อง

2.1.1 Neural Network

จากคำถามที่ว่า มนุษย์สามารถเรียนรู้สิ่งต่าง ๆ ได้อย่างไร ยกตัวอย่างเช่น เมื่อมนุษย์เห็นสิ่งของตรงหน้า มนุษย์สามารถตอบได้ทันทีว่าสิ่งนั้นคืออะไร โดยอ้างอิงจากการเรียนรู้ดูซ้ำแล้วพัฒนามาเป็นประสบการณ์ ในทางชีววิทยาได้ให้คำอธิบายการมองเห็นของมนุษย์ไว้ว่าหากมีแสงมากระทบที่ตา เซลล์ประสาทจะส่งสัญญาณต่อ ๆ กันไปเรื่อย ๆ เป็นโครงข่ายระบบประสาทจนไปถึงสมอง แล้วสมองจะบอกออกมาทันทีเลยว่าสิ่งที่เห็นนั้นคืออะไร จากองค์ความรู้ดังกล่าว นักคณิตศาสตร์และนักคอมพิวเตอร์จึงสร้างโมเดลทางคณิตศาสตร์ที่จำลองมาจากระบบประสาทในสมองของมนุษย์ขึ้นมา ทำให้เกิดโมเดลที่เรียกว่า Neural Network ขึ้นมานั่นเอง

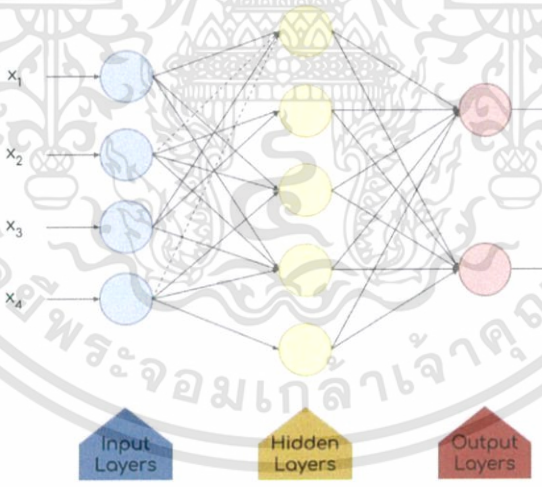
หน่วยที่เล็กที่สุดของ Neural Network เรียกว่า Neuron โดย Neuron จะมีหน้าที่คำนวนข้อมูลที่รับเข้ามา แล้วส่งผลลัพธ์ออกไป ดังภาพที่ 2.1 โดยมีองค์ประกอบที่สำคัญดังนี้

- Input หรือค่าที่ส่งเข้ามาที่ Neuron โดยจะมีเส้นทางที่เข้ามาได้หลายเส้นทางขึ้นอยู่กับการกำหนด
- Weight เป็นการให้น้ำหนักใน Neuron โดยค่าเริ่มต้นจะเป็นการสุ่มขึ้นมา เมื่อเวลาผ่านไป Neuron จะทำการเรียนรู้เรื่อย ๆ พร้อมกับปรับ weight ของตัวเองให้เหมาะสมกับ input ที่เข้ามา เพื่อให้ได้คำตอบที่ใกล้เคียงที่สุด
- Bias คือ ค่าที่จะช่วยทำให้การคำนวณออกมาถูกต้องมากขึ้น โดยจะเป็นเลขที่สุ่มขึ้นมาและปรับเปลี่ยนไปเรื่อย ๆ ทุกครั้งที่เรียนรู้ และค่านี้จะไม่ขึ้นอยู่กับ input ใด ๆ เปรียบเสมือนเป็นธรรมชาติของข้อมูล
- Output คือ ผลลัพธ์ที่ได้จากการเรียนรู้ของระบบ



ภาพที่ 2.1 Neuron

และเมื่อ Neuron หลาย ๆ โหนด มาเชื่อมต่อกันแล้วทำงานร่วมกันเป็นระบบ จึงจะเกิดเป็นโครงข่ายประสาทเทียม (Neural Network) ขึ้นมา ดังภาพที่ 2.2 โดยจะมีส่วนประกอบหลัก ๆ อยู่ 3 ส่วนดังนี้



ภาพที่ 2.2 Neural Network

- Input Layers

ชั้นที่รับข้อมูลเข้ามา โดยจำนวนของโหนดขึ้นอยู่กับจำนวนของ input ที่กำหนด เช่น ถ้า input ประกอบด้วย อายุ เพศ จังหวัดที่อาศัย รวมทั้งสิ้น 4 อย่าง input layers ก็จะมี 4 โหนด ปกติแล้วใน Machine Learning จะเรียกปัจจัยที่นำมาวิเคราะห์เหล่านี้ว่า feature

- Hidden Layers

ชั้นที่อยู่ระหว่างกลาง ซึ่งจะมีผลอย่างมากต่อประสิทธิภาพในการเรียนรู้ของโมเดล ซึ่ง hidden layers นั้นจะมีกี่ชั้นก็ได้ แล้วแต่จะกำหนด และแต่ละชั้นจะมีจำนวนของ neuron เท่าไหร่ก็ได้เช่นกัน ซึ่งการเพิ่มชั้นและจำนวน neuron ก็จะส่งผลต่อการทำงานของโมเดล ในส่วนของ hidden layer มีการทำงานเปรียบเสมือนส่วนที่เรียนรู้ข้อมูลเชิงลึก (Deep Learning)

- Output Layers

ชั้นที่จะนำเอาข้อมูลจากการคำนวณออกไปใช้ โดยจำนวนของโหนดในชั้นนี้ ขึ้นอยู่กับรูปแบบของ output ที่จะเอาไปใช้ ซึ่งถ้ายกตัวอย่างง่าย ๆ เช่น ถ้างานที่ทำเป็นการจำแนกข่าว output layers จะเท่ากับจำนวนข่าวที่สนใจที่จะให้โมเดลทำนาย เป็นต้น

จากองค์ประกอบต่าง ๆ ตั้งแต่ในส่วนของ neuron จนถึง neural network จะนำไปสู่การคำนวณทางคณิตศาสตร์เพื่อทำนายผลลัพธ์ เรียกกระบวนการนี้ว่า Forward Propagation โดยอ้างอิงจากสมการ Linear Regression ดังต่อไปนี้

$$Y_{predicted} = weight \times X + bias$$

โดย X คือ Input

$Y_{predicted}$ คือ Output ที่ได้จากการเรียนรู้ของระบบ

weight คือ น้ำหนักของเส้นทาง

bias คือ ค่าที่บวกเข้าไปเพื่อปรับให้ค่าที่คำนวณได้ถูกต้องมากขึ้น

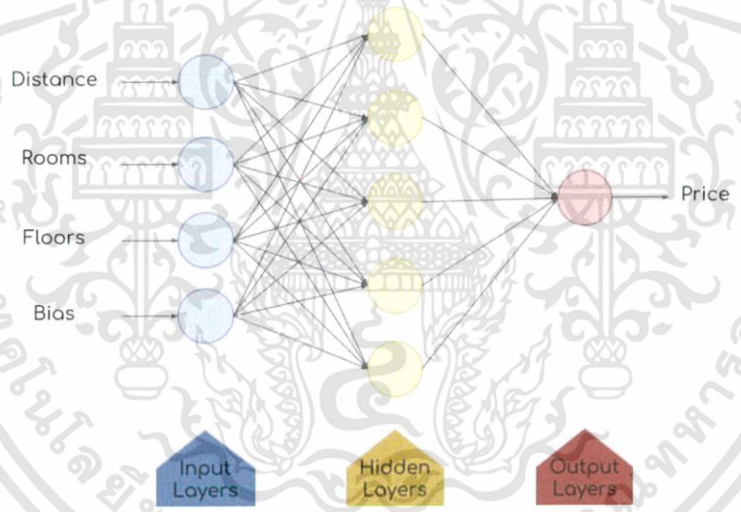
ยกตัวอย่างเช่น การทำนายมูลค่าของคอนโดมิเนียม คิดจากระยะห่างจากสถานีรถไฟฟ้าจะได้สมการดังนี้

$$Price = w \times Distance + bias$$

หากในกรณีที่มีมากกว่าหนึ่งปัจจัยที่ส่งผลต่อมูลค่าของคอนโดมิเนียม เช่น พื้นที่ในห้อง จำนวนห้อง จำนวนชั้น จะได้สมการดังนี้

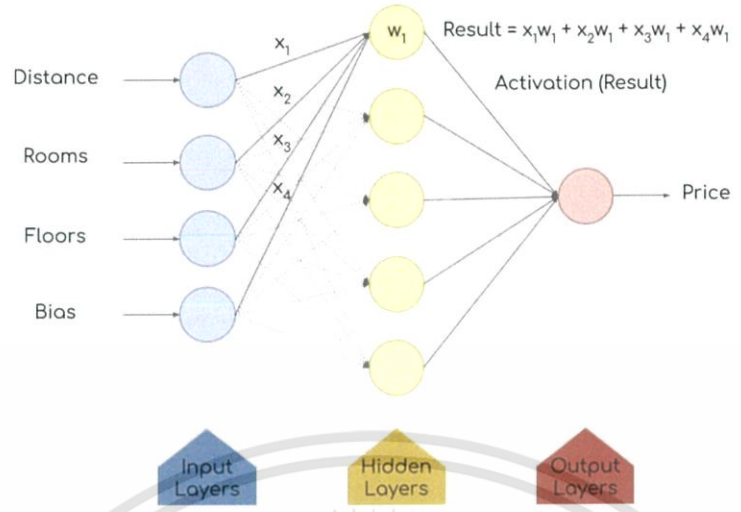
$$Price = w_1 \times Distance + w_2 \times Number\ of\ Rooms + w_3 \times Number\ of\ Floors + bias$$

จากสมการดังกล่าวสามารถวาดเป็นแผนภาพ Neural Network ได้ดังภาพที่ 2.3 ดังต่อไปนี้



ภาพที่ 2.3 Neural Network ของสมการมูลค่าของคอนโดมิเนียม

การคำนวณในช่วง Forward Propagation คือ ทุก ๆ hidden layer แต่ละโหนดจะมีสมการเชิงเส้น เพื่อคำนวณ weight ของโหนดนั้น ๆ โดย weight ของแต่ละโหนดจะมีค่าเริ่มต้นที่สุ่มขึ้นมาไม่เหมือนกัน และมีค่าของตัวมันเอง จากนั้นระบบจะนำ input แต่ละตัวคูณกับ weight ของโหนดนั้น ๆ แล้วนำผลลัพธ์มารวมกัน (Summation) และผ่าน Activation Function เพื่อกรองข้อมูลที่ไม่จำเป็นออก แล้วจึงส่งต่อไปยัง Layer ถัดไป



ภาพที่ 2.4 Forward Propagation ของโหนดแรกใน Hidden Layers

Activation Function เป็นฟังก์ชันที่ช่วยให้ผลลัพธ์ที่ออกมาถูกจำกัดอยู่ในขอบเขตที่ต้องการ เพราะ หากค่าที่เป็นไปได้อยู่ช่วง $(-\infty, \infty)$ ซึ่งหมายถึงข้อมูลจะไร้ขอบเขต ทำให้ยากต่อการคำนวณใน node ต่อ ๆ ไป และช่วยเพิ่มประสิทธิภาพให้กับการเรียนรู้ของระบบมากขึ้น โดยจะกล่าวเพิ่มเติมในส่วนถัดไป

หลังจากผ่านขั้นตอน Forward Propagation เสร็จแล้ว จะเข้าสู่ขั้นตอนการเรียนรู้ของโมเดล เรียกว่า Back Propagation โดยจะนำผลลัพธ์ที่ได้จากการคำนวณในส่วนของ Forward Propagation มาเทียบกับ output ที่เกิดขึ้นจริง เทียบจากสมการ Loss Function ดังนี้

$$Error = \frac{1}{2}\left(Y_{predicted} - Y_{actual}\right)^2$$

โดย $Error$ คือ ค่าความผิดพลาด

$Y_{predicted}$ คือ ผลลัพธ์ที่โมเดลทำนายออกมา

Y_{actual} คือ ผลลัพธ์ที่เกิดขึ้นจริง

ค่าที่ได้ออกมาจาก Loss Function จะเรียกว่า Gradient โดยโมเดลจะทำการปรับ weight ไปเรื่อย ๆ เพื่อทำให้ค่า gradient ที่ได้มีค่าน้อยที่สุด โดยใช้หลักการ Chain Rule ดังสมการนี้

$$\frac{\partial E}{\partial w} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial o_t} \frac{\partial o_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial w}$$

โดย E คือ ค่าความผิดพลาดทั้งหมด

o คือ ผลลัพธ์ที่ได้ในแต่ละโหนด

h คือ ผลรวมของค่า weight คูณกับ input แต่ละโหนด

w คือ ค่า weight

จากที่ได้กล่าวไว้เกี่ยวกับส่วนของ Activation Function เพิ่มเติม เนื่องจาก Activation Function มีอยู่หลายประเภท จึงขอยกตัวอย่าง Activation Function ที่เป็นที่รู้จักอย่างแพร่หลาย ประกอบด้วย 4 ประเภทดังนี้

- Logistic Activation Function (Sigmoid : σ)

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid Function มีข้อดีกว่า Linear Function ปกติ เนื่องจากค่าที่เป็นไปได้ทั้งหมดของ Linear Function อยู่ในช่วง $(-\infty, \infty)$ ซึ่งกว้างมาก แต่จากสมการข้างต้น จะสามารถจำกัดขอบเขตให้อยู่แค่ในช่วง $[0, 1]$ อีกทั้งยังช่วยให้ gradient มีความนุ่มนวล (smooth gradient) มากยิ่งขึ้นอีกด้วย

ด้วยเหตุผลทั้งหมดที่ได้กล่าวมา ทำให้ Sigmoid Function เป็นที่นิยมอย่างมากในการคำนวณของ Neural Network แต่ถึงอย่างนั้น Sigmoid Function ยังมีข้อเสียอยู่ นั่นคือ เป็นสาเหตุที่ทำให้เกิดปัญหาที่เรียกว่า Vanishing Gradient ในกระบวนการ Back Propagation

ค่า Gradient สามารถบอกความเปลี่ยนแปลงของค่าเริ่มต้นว่าส่งผลต่อผลลัพธ์ที่กำลังจะออกมาในอนาคตมากน้อยเพียงใด โดยถ้าค่ามีค่าสูง การเปลี่ยนแปลงก็จะมากตามไปด้วย

ดังนั้นการคำนวณย้อนหลังในกระบวนการ Back Propagation โดยใช้ Chain Rule ผ่าน Sigmoid Function ยิ่งทำให้ค่า gradient ลดลงไปเรื่อย ๆ ตามระยะทางที่ต้องผ่าน กล่าวคือ แม้ว่าค่าเริ่มต้นจะเปลี่ยนแปลงมากน้อยเพียงใด gradient ก็ไม่ได้เปลี่ยนแปลงไปจากเดิมมาก ส่งผลให้ประสิทธิภาพการเรียนรู้ของ Neural Network นั้น ไม่ดีเท่าที่ควรจะเป็น

- Tan Hyperbolic Function (Tanh)

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

จากสมมติฐานที่ว่า ช่วงของค่าที่เป็นไปได้ของ Sigmoid Function นั้น มีความแคบเกินไป ทำให้เห็นความเปลี่ยนแปลงของ gradient ได้น้อย จึงได้เกิด Tanh Function ขึ้น เพื่อแก้ปัญหาดังกล่าว โดยจากสมการข้างต้น ทำให้ช่วงของค่าที่เป็นไปได้ จากเดิมที่เป็น Sigmoid Function ค่าจะอยู่ในช่วง $[0, 1]$ เป็น $[-1, 1]$ ซึ่งช่วงที่กว้างขึ้นนั้น ทำให้ gradient ที่เกิดจาก Tanh Function สังเกตและเห็นความเปลี่ยนแปลงได้ดีกว่า Sigmoid Function ส่งผลให้ Tanh Function ถูกใช้เป็น Activation Function อย่างแพร่หลายนั่นเอง

- Rectified Linear Unit (ReLU)

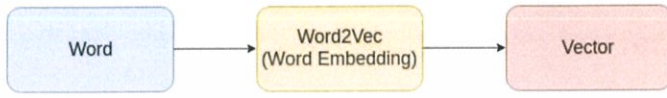
$$f(x) = max(0, x)$$

นอกจาก Tanh Function แล้ว ยังมีอีกหนึ่งฟังก์ชันที่น่าสนใจและเป็นที่นิยมกว่า โดยมีชื่อว่า Rectified Linear Unit (ReLU) โดยมีหลักการที่ง่ายมาก นั่นคือ ฟังก์ชันจะตรวจสอบค่าที่ส่งเข้ามา หากค่า ๆ นั้น มีค่ามากกว่าหรือเท่ากับ 0 ให้ทำการส่งค่าไปยัง node ต่อไปตามปกติ แต่ถ้าน้อยกว่า 0 จะเซ็ตให้ค่านั้นเป็น 0 แล้วส่งไปยัง node ต่อไปแทน

จากสมการ จะได้ช่วงของค่าที่เป็นไปได้ทั้งหมดจะอยู่ในช่วง $[0, \infty)$ โดยถ้าเทียบกับ Tanh Function จะมีข้อได้เปรียบตรงที่การคำนวณเป็นไปได้ง่ายกว่าหลายเท่า แต่ข้อเสียของ ReLU คือ ค่าของข้อมูลที่อยู่ในช่วงติดลบหรือเท่ากับศูนย์ การคำนวณจะไม่ดีเท่าที่ควร

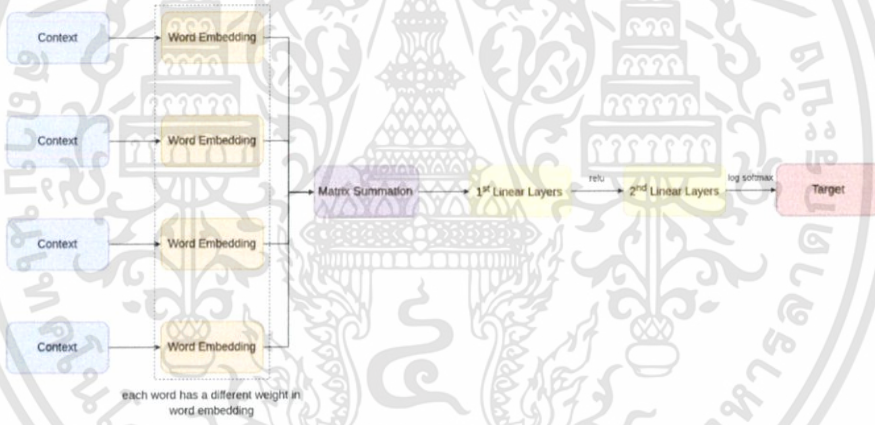
2.1.2 Word2Vec

กระบวนการนี้ มีจุดประสงค์เพื่อแปลงคำให้เป็นเวกเตอร์ เพื่อทำให้คอมพิวเตอร์เข้าใจในคำนั้น ๆ โดยมีวิธีการใช้งานดังภาพที่ 2.5



ภาพที่ 2.5 วิธีใช้งาน Word2Vec

ทางผู้จัดทำได้เลือกใช้ Library ที่ชื่อว่า Gensim ในส่วนของ models.word2vec และเลือกใช้อัลกอริทึมที่มีชื่อว่า Continuous Bag-of-Words (CBOW) มาทำการ Implement ให้ใช้งานกับโมเดล โดยมีกระบวนการการทำงานดังภาพที่ 2.6



ภาพที่ 2.6 กระบวนการเทรนโมเดลโดยใช้อัลกอริทึม Continuous Bag-of-Words

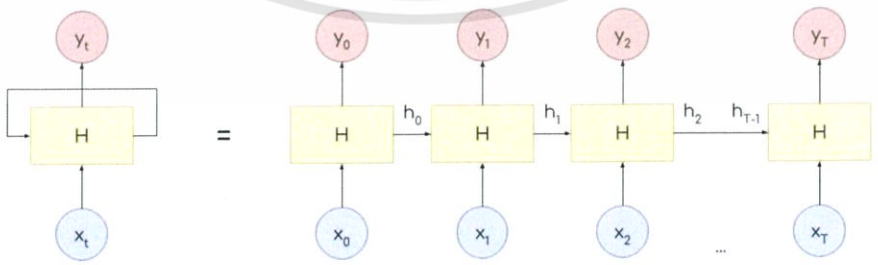
อัลกอริทึม Continuous Bag-of-Words (CBOW) มีแนวคิดและกระบวนการหลัก คือ นำคำรอบข้าง (context) มาทำการทำนายว่าคำตรงกลาง (target) ควรจะเป็นคำว่าอะไร ยกตัวอย่าง เช่น กำหนดให้ประโยคที่สนใจคือ “นายกฯ ปิดเยือนเมืองเบียร์ชวนเอกชนเยอรมนีลงทุนในไทย” หากนำประโยคดังกล่าว มาผ่านกระบวนการเตรียมข้อมูล (Data Preprocessing) จะได้ผลลัพธ์ออกมา คือ ['นา', 'ยกฯ', 'ปิด', 'เยือน', 'เมือง', 'เบียร์', 'ชวน', 'เอกชน', 'เยอรมนี', 'ลงทุน'] จากนั้นจะกำหนด context size ซึ่งทำหน้าที่เป็นค่าที่บอกว่า context ฝั่งซ้ายและขวาจะมีกี่คำ โดยผู้จัดทำกำหนดให้ context size เท่ากับ 2 จากนั้นจะได้ training set ดังนี้

context: ['นา', 'ยกฯ', 'เยือน', 'เมือง'] -> target: 'ปิด'
context: ['ยกฯ', 'ปิด', 'เมือง', 'เบียร์'] -> target: 'เยือน'
context: ['ปิด', 'เยือน', 'เบียร์', 'ชวน'] -> target: 'เมือง'
context: ['เยือน', 'เมือง', 'ชวน', 'เอกชน] -> target: 'เบียร์'
context: ['เมือง', 'เบียร์', 'เอกชน', 'เยอรมนี'] -> target: 'ชวน'
context: [' 'เบียร์', 'ชวน', 'เยอรมนี', 'ลงทุน'] -> target: 'เอกชน'

จาก training set ข้างต้น จะเข้าสู่กระบวนการต่อไป คือ นำไปเข้ากระบวนการเทรนดังภาพที่ 2.6 เพื่อทำการหา Word Embedding ที่เก็บค่า Weight ที่เหมาะสมกับคำนั้น ๆ เพื่อนำไปใช้ในขั้นตอนการเทรนของโมเดล

2.1.3 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) เป็นหนึ่งใน Neural Network ประเภทหนึ่ง ที่นำผลลัพธ์ที่ได้จากการคำนวณย้อนกลับมาใช้เป็น input อีกครั้ง ดังภาพที่ 2.7 เพื่อจะใช้งานกับข้อมูลที่มีลักษณะเป็นลำดับ (sequential data) เช่น วิดีโอ (sequence of images) หรือ ข้อความ (sequence of words) ยกตัวอย่างเช่น การอ่านหนังสือ โดยปกติแล้วสำหรับภาษาไทยและภาษาอังกฤษ มนุษย์จะอ่านหนังสือจากซ้ายไปขวาทีละคำต่อ ๆ กันไป การที่มนุษย์สามารถเข้าใจเรื่องราวและความหมายของประโยคนั้น ต้องอาศัยข้อความที่อ่านผ่านไปแล้วเปรียบเสมือนเป็น state ก่อนหน้า (Hidden State) แล้วนำมารวมกับคำที่กำลังอ่าน ณ เวลาปัจจุบัน ซึ่งเปรียบเสมือนเป็น input ด้วยองค์ประกอบทั้ง 2 สิ่งนี้ จึงทำให้มนุษย์สามารถเข้าใจความหมายของข้อความที่กำลังอ่านได้นั่นเอง



ภาพที่ 2.7 Recurrent Neural Network (RNN)

จากภาพที่ 2.7 ในฝั่งซ้ายแสดงให้เห็น loop ที่รับ x (input) ผ่าน H (hidden layer) และได้ h (ค่า hidden state ณ เวลานั้น ๆ) ออกมา แล้วนำค่า h ดังกล่าวกลับมาคิดในรอบถัดไป เปรียบเสมือนฝั่งขวาที่แสดงการทำงานเป็นลำดับ (sequence) โดยจะเห็นได้ว่า y_1 เป็น ค่าที่ออกมาจาก hidden layer ที่รับ x_1 และ h_0 มาใช้ในการคำนวณ และ h_1 จะไปเป็นค่าที่ส่งผลต่อ y ของ hidden layer ต่อ ๆ ไปอีก อ้างอิงจากสมการดังนี้

$$h_t = f_h(U_h h_{t-1} + W_h x_t + b_n)$$
$$y_t = f_y(W_y h_t + b_y)$$

โดย f_h คือ activation function ของ hidden Layer

f_y คือ activation function ของ output Layer

W_h คือ weight matrix ของ hidden Layer

U_h คือ hidden-state-to-hidden-state matrix

จากที่กล่าวมาข้างต้นจะเห็นได้ว่า Recurrent Neural Network ดูสมเหตุสมผลและไม่น่ามีปัญหา แต่ยังมีปัญหาหลักที่ถูกเรียกว่า Vanishing Gradient ซึ่งเกิดในช่วงอัพเดท weight ของกระบวนการ Back Propagation ซึ่งจะคำนวณ gradient หรือ loss โดยสามารถอธิบายได้ด้วยสมการดังต่อไปนี้

$$\frac{\partial E}{\partial w} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial o_t} \frac{\partial o_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial w}$$

โดย E คือ ค่าความผิดพลาดทั้งหมด

o คือ ผลลัพธ์ที่ได้ในแต่ละโหนด

h คือ ผลรวมของค่า weight คูณกับ input แต่ละโหนด

w คือ Weight

จากสมการดังกล่าวตัวแปรแต่ละตัวมีความหมายดังนี้

$\frac{\partial E_t}{\partial o_t}$ คือ การคำนวณว่า ค่าความผิดพลาดที่เกิดขึ้นมีผลต่อค่าของผลลัพธ์ที่ได้มากน้อยเพียงใด

$\frac{\partial o_t}{\partial h_t}$ คือ การคำนวณความเปลี่ยนแปลงของผลรวมของค่า weight คูณกับ input ในผลลัพธ์ที่ออกไปในแต่ละโหนด

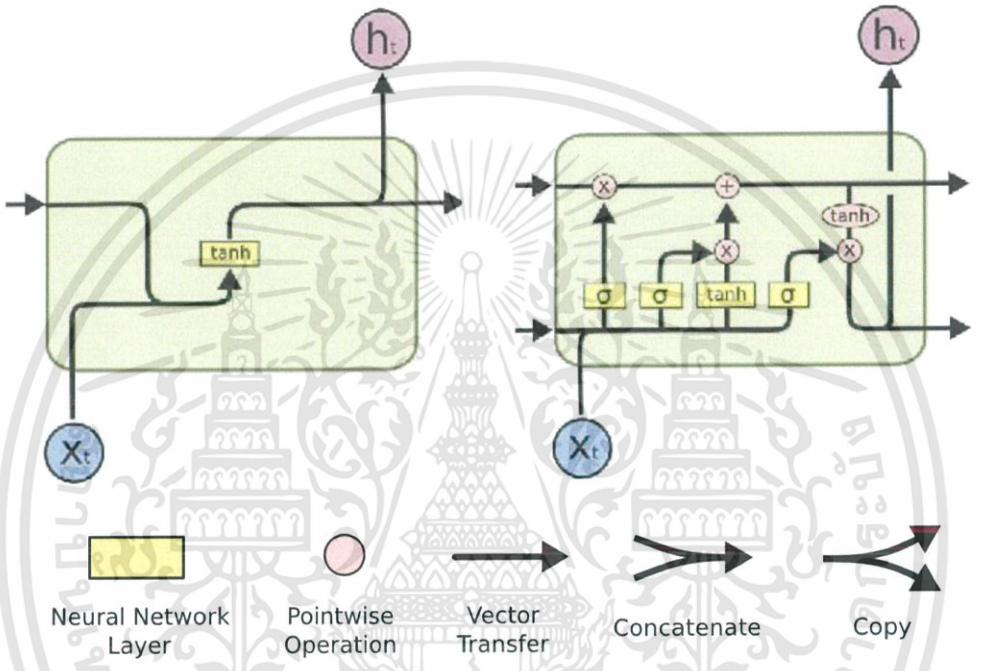
$\frac{\partial h_t}{\partial h_k}$ คือ การคำนวณการเปลี่ยนแปลงของผลรวมของค่า weight คูณกับ input ในโหนดหนึ่ง ซึ่งอาจจะส่งผลต่อผลรวมของค่า weight คูณกับ input ในอีกโหนดหนึ่ง

$\frac{\partial h_k}{\partial w}$ คือ การคำนวณผลรวมของค่า weight คูณกับ input ที่อาจจะส่งผลต่อการเปลี่ยนแปลงในน้ำหนัก

จากสูตรดังกล่าวจะเห็นได้ว่า ค่า gradient นั้นเมื่อถูกคำนวณไปนาน ๆ ค่าจะยิ่งลดน้อยลงจนแทบจะไม่เห็นการเปลี่ยนแปลงใด ๆ แม้ข้อมูลที่เข้ามาจะมีการเปลี่ยนแปลงไปมากมายหรือยิ่งใหญ่แค่ไหนก็ตาม

2.1.4 Long Short-Term Memory Networks (LSTMs)

จากปัญหา Vanishing Gradient ของ Recurrent Neural Network ที่ได้กล่าวไปจึงมีการเสนอใช้ Recurrent Neural Network แบบพิเศษที่เรียกว่า Long Short Term Memory Networks (LSTMs) โดย *Sepp Hochreiter & Juergen Schmidhuber (1997)* ซึ่งมีโครงสร้างที่แตกต่างออกไปจาก Recurrent Neural Network ดังภาพที่ 2.8



ภาพที่ 2.8 เปรียบเทียบโครงสร้างระหว่าง RNN และ LSTM

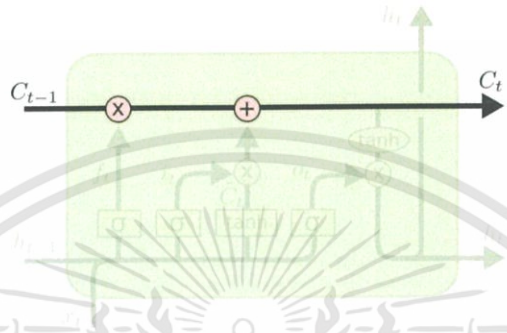
ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

หาก RNN เปรียบเสมือน Neural Network ที่มีหน่วยความจำอยู่ภายในเพื่อบันทึกค่า hidden state ก่อนหน้า LSTM ก็มีหน่วยความจำอยู่ภายในเช่นกัน แต่ที่ดีกว่าเนื่องจากตัวหน่วยความจำของ LSTM สามารถบอกได้ด้วยว่า เมื่อไหร่ที่ควรจะ read, write หรือ forget (delete) โดยตัวควบคุมการกระทำเหล่านี้ ไม่ใช่บิต 0 กับ 1 แต่เป็นค่า analog แทน

LSTMs มีองค์ประกอบสำคัญที่ต้องรู้จักก่อนจะนำไปสู่กระบวนการทำงานอยู่ด้วยกันทั้งหมด 2 ประการดังนี้

- Cell state

  ทำหน้าที่เก็บ state ของ memory cell ใน LSTM

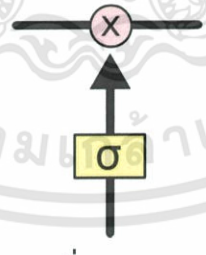


ภาพที่ 2.9 Cell State

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

- Gate

  ทำหน้าที่ควบคุมการไหลของข้อมูล โดยใช้ค่า analog ที่กล่าวไปข้างต้น ซึ่งค่า analog จะคอยควบคุมว่าควรจะ read, write หรือ forget (delete) เปรียบเสมือนประตูที่จะบอกว่า ควรให้ข้อมูลไหลเข้า ไหลออก หรือไหลหายไปจากระบบ



ภาพที่ 2.10 Gate

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

หลักการทำงานของ LSTMs จะมีอยู่ 4 ขั้นตอนดังนี้

2.1.4.1 Forget

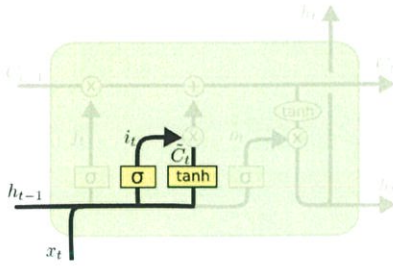
ขั้นตอนแรกเป็นการตัดสินใจว่าข้อมูลใด ที่ระบบจะทำการลืมหรือลบออกจาก cell state โดยมี forgot gate layer ดังภาพที่ 2.11 เป็นตัวตัดสิน ซึ่งการตัดสินดังกล่าว วัดจาก input (x_t) ที่เข้ามา และ ค่า hidden state (h_{t-1}) ก่อนหน้า แล้วผ่าน sigmoid function (σ) เพื่อตัดสินตามสมการหาก forget gate (f_t) ให้ค่าเป็น 0 หมายความว่าให้ลบ cell state เดิมออก แต่ถ้าให้ค่าเป็น 1 หมายความว่า ระบบจะยังคงเก็บค่า cell state ต่อไป

ภาพที่ 2.11 Forget Gate Layer

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

2.1.4.2 Write

เมื่อระบบรับ input ใหม่เข้ามา สิ่งที่ต้องคำนึงถึงมี 2 อย่าง ได้แก่ จะอัพเดท cell state ด้วย input ใหม่หรือไม่ และ หากอัพเดท จะอัพเดทด้วยค่าอะไร ในส่วนของการอัพเดทหรือไม่ จะควบคุมโดย input gate (i_t) ซึ่งสมการจะเหมือนกับขั้นตอนแรกในส่วนของการ Forget ต่อมาในส่วนที่สอง หากจะอัพเดทค่า ที่จะทำการอัพเดทจะได้มาจาก input modulation gate ($\tilde{C}_t$) โดยคิดจากสมการดังภาพที่ 2.12 จะเห็นว่าสมการจะคล้ายกับ input gate ต่างกันตรงที่เปลี่ยนจาก sigmoid function เป็น tanh function แทน โดยค่าที่ได้ออกมา จะเรียกว่า cell state candidate



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

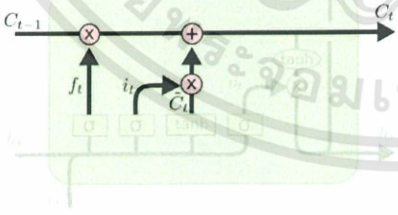
ภาพที่ 2.12 Input Gate and Input Modulation Gate Layer

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

2.1.4.3 Update Cell State

จากสมการดังภาพที่ 2.13 สังเกตฝั่งขวาของสมการ เริ่มจากส่วนแรก ($f_t * C_{t-1}$) จะเห็นว่า หาก forgot gate บอกให้ลบ cell state เดิมทิ้งไป ($f_t = 0$) จะทำให้ส่วนแรกนั้น ไม่มีผลต่อการอัพเดท cell state ในทางกลับกัน หาก forgot gate บอกให้เก็บ cell state เดิมไว้ ($f_t = 1$) จะทำให้ส่วนแรกนั้น มีผลต่อการอัพเดท cell state อยู่นั่นเอง

ในส่วนหลังของสมการ ($i_t * \tilde{C}_t$) จะเป็นส่วนของการอัพเดท cell state จากข้อมูลใหม่ จากขั้นตอนที่ผ่านมา ระบบจะมีค่าที่เตรียมจะอัพเดทไว้แล้ว ($\tilde{C}_t$) และค่าที่บ่งบอกว่าจะอัพเดทหรือไม่อัพเดทซึ่งมาจาก input gate (i_t) โดยถ้าหาก $i_t = 1$ หมาความว่าให้อัพเดท แต่ถ้า $i_t = 0$ หมายความว่า จะไม่ใช้ค่าที่เตรียมอัพเดทไว้ และหลังจากเสร็จสิ้นกระบวนการทั้งหมดในการอัพเดท cell state แล้ว จะได้ค่า C_t ออกมาเพื่อนำไปใช้ในขั้นตอนถัดไป



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

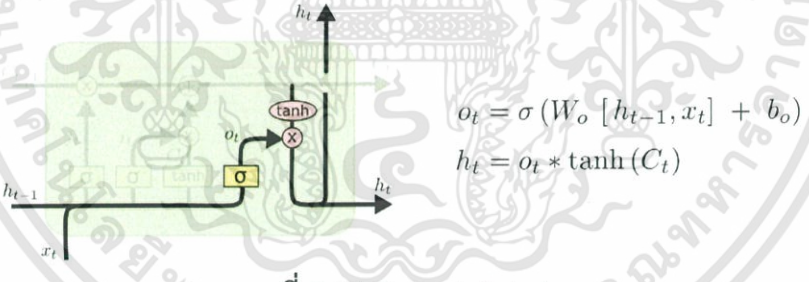
ภาพที่ 2.13 Update Cell State

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

2.1.4.4 Read

เป็นส่วนที่ใช้บอกว่า ข้อมูลนี้ พร้อมที่จะเป็นข้อมูลขาออกแล้วหรือไม่ ซึ่งข้อมูลขาออกของแต่ละ node มีด้วยกันทั้งหมด 2 ค่า ได้แก่ ค่าที่เกิดจากการคำนวณโดยมีการ update สถานะ ซึ่งถูกคำนวณจากขั้นตอน Update cell state ค่านี้จะถูกส่งต่อไปยัง node ต่อไปทันทีโดยไม่ผ่านฟังก์ชั่นใด ๆ และค่าของข้อมูลขาเข้าที่ถูกดัดแปลงผ่านการคำนวณใน output gate layer ค่านี้จะถูกส่งต่อไปเป็นข้อมูลขาเข้าของ node ถัดไป และยังถูกส่งไปเป็นผลลัพธ์ของ node นั้น ๆ ด้วย

โดยค่านี้ก็เช่นเดียวกับส่วนอื่น ๆ คือการนำข้อมูลจาก node ที่แล้วที่เข้ามา รวมกับข้อมูลขาเข้าใน node นั้น ๆ ผ่านฟังก์ชัน sigmoid เช่นเดียวกับส่วนอื่น ๆ ตามสมการแรกดังภาพที่ 2.14 แต่สิ่งที่แตกต่างกันก็คือ หลังจากที่มันผ่านการคำนวณค่า sigmoid แล้ว มันจะถูกนำมา pointwise กับค่าสถานะปัจจุบันของ node ที่ได้ถูกคำนวณมาเรียบร้อยแล้วจากส่วนที่ผ่านมา ซึ่งค่าสถานะนั้น จะถูกนำไปเข้าฟังก์ชั่น tanh ก่อน แล้วจึงนำผลลัพธ์ที่ได้หลังจากเข้าฟังก์ชันนั้นแล้วมา pointwise กับค่าที่ออกมาจาก sigmoid function ตามสมการที่สองดังภาพที่ 2.14

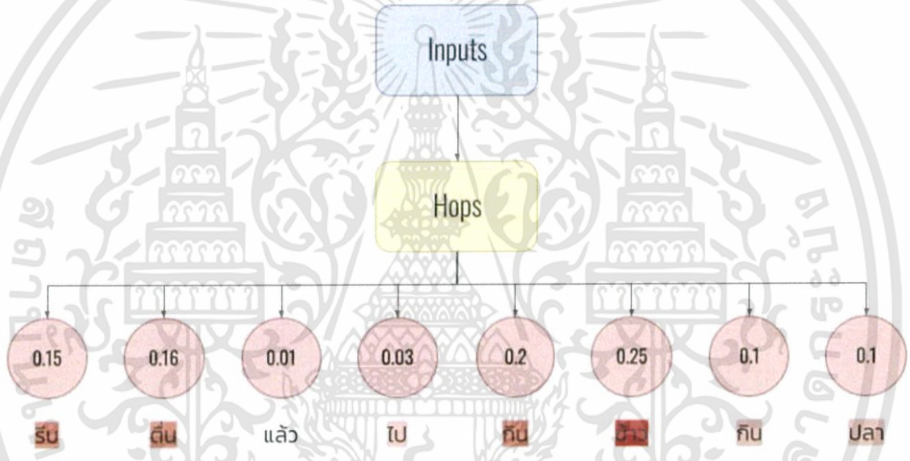


ภาพที่ 2.14 Output Gate Layer

ที่มา : colah.github.io/posts/2015-08-Understanding-LSTMs

2.1.5 Self-Attention

แนวคิดของ Self-Attention นั้น ถูกใช้ทั่วไปในหลาย ๆ โมเดลที่เป็น Recurrent Neural Network (RNN) ยกตัวอย่างเช่น Language Models, Machine Translation ซึ่งถือว่า เป็น Sequence-to-Sequence Model (seq2seq) หรือเรียกอีกชื่อว่า RNN Encoder–Decoder เป็นต้น เนื่องจาก Sequence-to-Sequence Model มีปัญหาที่สืบเนื่องมาจาก RNN นั่นคือ การส่งข้อมูลต่อกันไปเป็นสายยาว อาจจะมีข้อมูลที่จำเป็นบางอย่างสูญหายไประหว่างทางได้ จึงมีแนวคิดว่า จะดีกว่าไหมถ้าให้กระบวนการสร้าง output สามารถโฟกัสไปที่ input ส่วนใดส่วนหนึ่งได้โดยตรง และนี่คือที่มาของ Self-Attention นั่นเอง



ภาพที่ 2.15 การทำงานของ Attention

กระบวนการทำ Self-Attention มีกระบวนดังภาพที่ 2.15 โดยในขั้นแรก คือ ในแต่ละ timestep (t) จะรับ inputs เข้ามาใน hops ซึ่งเป็น Neural Network แล้วตัว hops จะทำการคิด score ของแต่ละ node โดยหาก score ของตำแหน่งนั้นสูง หมายความว่า โมเดลจะให้ความสำคัญ หรือให้ความใส่ใจกับตำแหน่งนั้นมาก โดยคิดจากสมการดังต่อไปนี้

$$e^d_{tt'} = h^{dT}_t W^d_{\text{attn}} h^d_{t'}$$

เมื่อได้ค่า score ออกมาแล้ว โมเดลจะเอาเข้าฟังก์ชัน softmax เพื่อแปลงเป็นค่าความน่าจะเป็น ซึ่งค่านี้จะเปรียบเสมือน weight สำหรับ node ต่าง ๆ โดยคิดจากสมการดังต่อไปนี้

$$\alpha_{tt'}^{d} = \frac{exp(e_{tt'}^{d})}{\sum_{j=1}^{t-1} exp(e_{tj}^{d})}$$

จากนั้นก็จะทำการหา weight average ของออกทุก node มาเป็นเวกเตอร์เดียว เพื่อนำไปใช้ในการคำนวณ output ต่อไป โดยคิดจากสมการดังต่อไปนี้

$$c_t^d = \sum_{j=1}^{t-1} \alpha_{tj}^d h_j^d$$

ในทางปฏิบัติก็พบว่าเมื่อใช้ Attention เข้ามาช่วยแล้ว ได้ผลลัพธ์ที่ดีกว่าเมื่อไม่ใช้แทบจะแน่นอน เนื่องจาก Attention จะแก้ปัญหาคอขวดดังที่กล่าวมาข้างต้นแล้ว ยังถือว่าสามารถแก้ปัญหา vanishing gradient ไปด้วยพร้อมกันอีกด้วย โดยในปัจจุบัน Attention จึงเปรียบเสมือนท่าบังคับพื้นฐานของการทำ sequence-to-sequence learning ไปแล้ว ถ้าหากใช้ซอฟต์แวร์สำหรับทำงานด้านนี้ เช่น OpenNMT จะพบว่ามี Attention ถูกตั้งค่าเป็น default ให้เลย และใน OpenNMT-py ตอนนี้ยังไม่มี option ให้เอา Attention ออกได้ด้วย

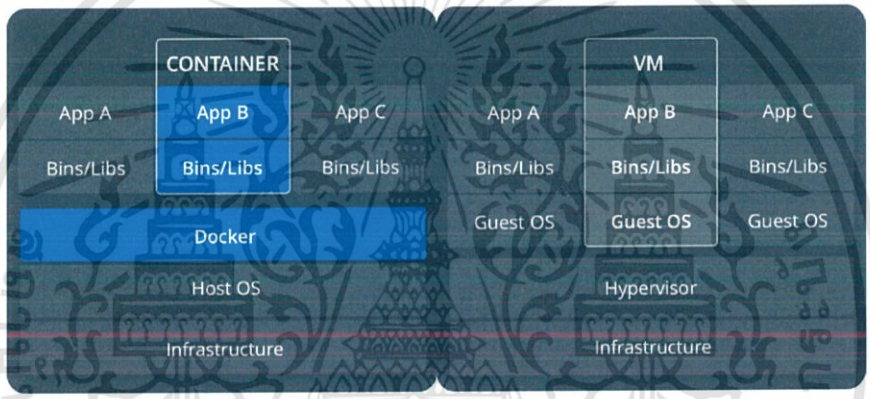
2.1.6 PyTorch

PyTorch เป็น Framework ที่ถูกพัฒนาขึ้นโดย Facebook ซึ่งดัดแปลงมาจาก Library ที่มีชื่อว่า torch ที่ถูกใช้มาก่อนในภาษา Lua โดยมีจุดประสงค์หลักเพื่อเป็น Framework สำหรับการสร้างโครงข่ายประสาทเทียม (Neural Network) เขียนโดยใช้ภาษา Python ทำให้สามารถเข้าใจได้ง่าย เหมาะสำหรับการศึกษาค้นคว้าและปรับแต่งตามที่ผู้ใช้ต้องการ

โดยทั่วไปแล้วโครงข่ายประสาทเทียมจะถูกสร้างจากการนำองค์ประกอบต่าง ๆ ในแต่ละชั้นของระบบโครงข่ายประสาทเทียมมาประกอบกันเปรียบเสมือนจิ๊กซอว์ ความสะดวกของ PyTorch คือ มีองค์ประกอบพื้นฐานทุกอย่างเก็บไว้ภายในโมดูลแล้ว ทำให้ผู้ใช้ไม่ต้องไปลงมือเขียนโปรแกรมใหม่ให้ยุ่งยากเสียเวลา อีกทั้งเมื่อเทียบกับ Tensorflow ซึ่งเป็น Framework ที่ใช้สำหรับสร้างโครงข่ายประสาทเทียมเหมือนกันแล้ว PyTorch จะใช้งานง่ายกว่า เนื่องจาก มีลักษณะที่ค่อนข้างสำเร็จรูป และมีเอกสารคู่มือการใช้ที่เข้าใจง่ายกว่ามาก ส่งผลให้ง่ายสำหรับผู้ใช้ใหม่ หรือหากเป็นผู้ใช้ที่คุ้นเคยอยู่แล้ว PyTorch ก็เหมาะเช่นกัน เพราะ มีคำสั่งและการใช้งานที่คล้าย numpy ซึ่งเป็นโมดูลส่วนเสริมของ Python ที่ใช้ในการคำนวณทางคณิตศาสตร์อยู่มาก ทำให้ไม่ต้องเสียเวลาไปเรียนรู้ใหม่ตั้งแต่เริ่มนั่นเอง

2.1.7 Docker

Docker คือ engine ตัวหนึ่งที่มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการ run service ที่ต้องการ มีการทำงานคล้ายคลึงกับ Virtual Machine เช่น VMWare, VirtualBox, XEN, KVM แต่ข้อแตกต่างที่ชัดเจน คือ Virtual Machine ที่รู้จักกันก่อนหน้านี้นั้น เป็นการจำลองทั้ง OS เพื่อใช้งานและหากต้องการใช้งาน service ใด ๆ จึงทำการติดตั้งเพิ่มเติมบน OS นั้น ๆ แต่สำหรับ docker แล้วจะใช้ container ในการจำลองสภาพแวดล้อมขึ้นมา เพื่อใช้งานสำหรับ 1 service ที่ต้องการใช้งานเท่านั้น โดยไม่ต้องมีส่วนของ OS เข้าไปเกี่ยวข้องเหมือน Virtual Machines อื่น ๆ ดังภาพที่ 2.16



ภาพที่ 2.16 การเปรียบเทียบระหว่าง Containers และ Virtual Machines

Docker image เป็นเหมือนตัวต้นแบบของ container ซึ่งภายในจะประกอบด้วย application ต่างๆ ที่มีการติดตั้งไว้เพื่อใช้งานสำหรับ service นั้น ๆ รวมทั้งมีการ config ค่าต่างๆ ไว้เรียบร้อยแล้ว จากนั้นก็นำมาสร้างเป็น docker image บน registry เพื่อนำใช้งาน ทั้งนี้ผู้ใช้งานสามารถยังสร้าง docker image สำหรับใช้งานเองได้อีกด้วย

Docker container สามารถมองได้เสมือนกล่อง ซึ่งนำ docker image มาติดตั้ง เพื่อให้สามารถใช้งาน service ที่ต้องการจาก image นั้นๆ ได้ โดยใน container แต่ละตัวจะมีการใช้งาน RAM, CPU, ไฟล์ config ต่าง ๆ เป็นของแต่ละ container เอง และยังสามารถสั่ง start, stop ได้ที่ container นั้น ๆ อีกด้วย

ประโยชน์ของ Docker หลัก ๆ มีทั้งหมดดังต่อไปนี้

- Docker engine สามารถใช้งานได้บนหลาย platform ทั้งบน Linux, Mac และ Windows
- Docker มีขนาดเล็ก สามารถใช้งาน และติดตั้งได้อย่างรวดเร็ว และสะดวกในการ start / stop หรือแม้แต่การย้ายไปใช้งานสำหรับเครื่อง server อื่นที่มีการ run docker engine ก็สามารถทำได้โดยไม่ซับซ้อน
- ผู้ใช้งาน docker ไม่จำเป็นต้องติดตั้ง OS อีกครั้งเพื่อติดตั้ง container รวมทั้งไม่จำเป็นต้อง config เพิ่มเติมในส่วนที่ไม่จำเป็นอีกด้วย
- Docker มีความต้องการในการใช้ CPU, RAM และพื้นที่น้อยกว่า Virtual Machine ทั้งนี้ในทรัพยากรที่มีเท่ากัน docker สามารถใช้งาน container ได้มากกว่า Virtual Machine
- เนื่องจากผู้ใช้งาน สามารถสร้าง docker image ได้เอง จาก dockerfile ดังนั้นการใช้งาน docker ยังช่วยลดปัญหาสภาพแวดล้อมที่ต่างกัน ที่มักพบเมื่อบาง application สามารถทำงานได้บน development server แต่ไม่สามารถใช้งานบน production server ได้
- Docker ยังมี docker registry ซึ่งผู้ใช้งานสามารถเลือก pull image ต่าง ๆ ที่มีการสร้างไว้ให้แล้วมาใช้งาน โดยมี Docker Hub เป็น registry หลักในการเรียกใช้ image

2.1.8 Flask

Flask เป็น Micro Web Framework ที่เขียนโดยใช้ภาษา Python สาเหตุที่ Flask ถูกจัดเป็น Micro Framework เนื่องจาก ไม่จำเป็นที่จะต้องมีเครื่องมือหรือ Library อื่น ๆ และไม่มีส่วนของ Database Layers มีข้อดีตรงที่ใช้ทรัพยากรที่น้อยกว่า Web Framework ทั่วไป แต่ถ้าหากอยากจะเพิ่มความสามารถหรือ Feature ที่ผู้ใช้ต้องการ Flask ก็สามารถจะเพิ่มส่วนขยายเหล่านี้เข้าไปได้

2.1.9 Git

Git คือ Version Control แบบ Distributed ตัวหนึ่ง เป็นระบบที่ใช้จัดเก็บและควบคุมการเปลี่ยนแปลงที่เกิดขึ้นกับไฟล์ชนิดใดก็ได้ ไม่ว่าจะเป็น Text File หรือ Binary File โดยจะเรียกรวมกันว่า Source Code

เหตุผลที่ต้องใช้ Git มีอยู่ 2 อย่างหลัก ๆ ได้แก่ สามารถ Track Version ของ Source Code ย้อนกลับได้ กล่าวคือ เมื่อจัดเก็บไฟล์เข้าไปในระบบของ Git จะเรียกว่า Git Repository ซึ่งเก็บสำรองข้อมูลและการเปลี่ยนแปลงของ Source Code ทำให้สามารถย้อนกลับไปที่เวอร์ชั่นใด ๆ ก่อนหน้า และดูรายละเอียดการเปลี่ยนแปลงของแต่ละเวอร์ชั่นได้ นอกจากนั้นยังสามารถดูได้ว่าใครเป็นคนแก้ไขเพื่อง่ายต่อการจัดการ ทำให้เกิดเหตุผลข้อต่อไป ซึ่งก็คือ ช่วยในการพัฒนาซอฟต์แวร์เป็นทีม เนื่องจาก Developer นั้น ไม่สามารถที่จะทำงานด้วยตัวคนเดียวได้ และจากเหตุผลข้อที่แล้วที่กล่าวว่า Git สามารถเก็บบันทึกการเปลี่ยนแปลงของ Source Code เวอร์ชั่นล่าสุดไว้ที่ Local Repository ซึ่งสามารถทำงานได้โดยที่ไม่ต้องต่อกับอินเตอร์เน็ต ทำให้การ Update เพื่อเปลี่ยนแปลงของ Source Code เวอร์ชั่นล่าสุดให้กับเพื่อนร่วมทีมก็สามารถที่จะ Push ขึ้นไปเก็บที่ Remote Repository (Git Hosting) และเพื่อนร่วมทีมสามารถ Pull เวอร์ชั่นล่าสุดนั้นมารวม (Auto Merge) ที่เครื่องของตนได้ ทำให้ Source Code ที่พัฒนาร่วมกันกับคนภายในทีมเป็นเวอร์ชั่นล่าสุดเสมอ

การทำงานของ Git ค่อนข้างที่จะเรียบง่าย เพียงแค่ติดตั้งจาก https://git-scm.com ซึ่งเป็นเว็บไซต์อย่างเป็นทางการของ Git จากนั้นพิมพ์ชุดคำสั่งลงไปใน Terminal เพียงเท่านี้ก็สามารถใช้งานได้เลย และเนื่องจากคำสั่งที่ใช้ในการทำงานของ Git มีค่อนข้างมาก จึงยกตัวอย่างคำสั่งที่ใช้งานบ่อย ๆ ดังนี้

Git Config

เป็นคำสั่งที่ใช้แสดงและกำหนดข้อมูลของผู้ใช้เพื่อระบุตัวตน และคุณสมบัติอื่นของ Git โดยมีคำสั่งดังนี้

```
$git config --global --list
$git config --list
```

คำสั่งแรกใช้ในการแสดงคุณสมบัติของ Git ทั้งหมด และในส่วนคำสั่งที่สองใช้แสดงคุณสมบัติ Git เฉพาะ Repository นั้น ๆ

```
$git config --global user.name "Your Name"
$git config --global user.email
"example@email.com"
$git config --global -list
```

ในส่วนที่สองจะเป็นการกำหนดค่าเริ่มต้นต่าง ๆ จะเริ่มตั้งแต่การกำหนดชื่อผู้ใช้ กำหนดอีเมล์ จากนั้นให้ใช้คำสั่งที่แสดงคุณสมบัติเพื่อตรวจสอบอีกครั้งหลังจากกำหนดค่าเสร็จเรียบร้อยแล้ว

Git Init

เป็นคำสั่งที่ใช้สร้างระบบของ Git ขึ้นมาภายใต้โฟลเดอร์หรือ Path นั้น โดยจะสร้างโฟลเดอร์ .git ขึ้นมาเพื่อใช้เก็บ สำรองข้อมูล การเปลี่ยนแปลงและคุณสมบัติอื่น ๆ ของ Git โดยมีคำสั่งดังนี้

```
$git init
```

Git Status

เป็นคำสั่งที่ใช้ตรวจสองสถานะของ Source Code ในระบบของ Git ซึ่งจะแสดงสถานะดังที่ได้อธิบายข้างต้นไปแล้วโดยมีคำสั่งดังนี้

```
$git status
```

Git Add

เป็นคำสั่งที่ใช้เพิ่มการเปลี่ยนแปลงของ Source Code เข้าไปที่สถานะ Staged โดยมีคำสั่งดังนี้

```
$git add <file_name>
$git add README.md
$git add .
```

คำสั่งแรกคือ การ Add ส่วนบรรทัดถัดไปคือการเพิ่มชื่อไฟล์ README.md โดยไฟล์นี้จะเปรียบเสมือนคู่มือการใช้งานที่แสดงให้ผู้ใช้เข้าใจสิ่งที่ผู้พัฒนาจะสื่อ แล้วในส่วนบรรทัดสุดท้ายจะใช้ในกรณีที่มีหลาย ๆ ไฟล์และต้องการเพิ่มเข้าไปบน Git ทั้งหมด

Git Commit

เป็นคำสั่งที่ใช้ยืนยัน Source Code ที่อยู่ในสถานะ Staged เข้าไปเก็บไว้ที่ Local Repository โดยมีคำสั่งดังนี้

```
$git commit -m "message"
$git commit
```

ความแตกต่างระหว่างสองคำสั่งดังกล่าว คือ คำสั่งแรกเป็นการยืนยันการเปลี่ยนแปลงพร้อมกับใส่ข้อความไปพร้อมกันเลย ซึ่งข้อความจะเปรียบเสมือนเป็นคอมเม้นท์ ต่างจากบรรทัดถัดไปที่หลังจากพิมพ์คำสั่งแล้ว จะทำการเปิดโปรแกรม Vim ซึ่งเป็น Editor แบบหนึ่งที่ใช้งานใน Terminal แล้วให้ผู้ใช้เพิ่มข้อความตรงนั้นแทน

Git Push

เป็นคำสั่งที่ใช้ส่งการเปลี่ยนแปลงของ Source Code ที่เก็บอยู่บน Local Repository ขึ้นไปยัง Remote Repository โดยมีคำสั่งดังนี้

```
$git push origin master
```

คำสั่งดังกล่าว คือ การส่งการเปลี่ยนแปลง Branch master ไปยัง Remote ที่มีชื่อว่า origin

Git Clone

เป็นคำสั่งที่ใช้ดึงประวัติทั้งหมดบน Remote Repository ของเพื่อนร่วมทีม ของคนอื่นหรือของผู้ใช้เองที่มีอยู่แล้วบน Git Hosting มาที่เครื่องของผู้ใช้ คำสั่งนี้จะคล้าย ๆ Git Init ที่ใช้สร้างระบบ Git ขึ้นมาตอนเริ่มต้น แต่ผู้ใช้จะได้ประวัติเดิมของ Repository มาด้วย ทำให้สามารถเริ่มพัฒนาต่อจากตรงจุดนี้ได้เลย โดยมีคำสั่งดังนี้

```
$git clone https://github.com/NewGame0/
Android_HelloWorld.git
```

Git Pull

เป็นคำสั่งที่ใช้รับการเปลี่ยนแปลงของ Source Code ล่าสุดที่อยู่บน Remote Repository ลงมายัง Local Repository และทำการ Auto Merge โดยมีคำสั่งดังนี้

```
$git pull origin master
```

2.2. งานวิจัยที่เกี่ยวข้อง

Attention Is All You Need

ในโลกของ Deep Learning ปัจจุบัน โครงสร้างของ Neural Network ที่เป็นที่นิยมใช้กันอย่างแพร่หลายมีอยู่แค่ 2 โครงสร้างเท่านั้น คือ Recurrent Neural Network (RNN) และ Convolutional Neural Network (CNN) สำหรับ RNN โดยส่วนใหญ่จะใช้ในงานด้าน Language หรือ งานจำพวก Sequential Data เช่น ข้อความต่าง ๆ และ CNN จะใช้ในงานด้าน Vision เช่น ภาพและวิดีโอ แต่ทั้งนี้ทั้งนั้น CNN ก็เอามาใช้ในงานด้าน Language ส่วน RNN ก็สามารถนำไปใช้ในงานด้าน Vision ได้เช่นกัน ซึ่งงานวิจัยนี้เล็งเห็นทั้งจุดอ่อนของทั้ง RNN และ CNN จึงปฏิเสธที่จะใช้ทั้งสองอย่างนี้ในการทำ NLP แล้วหันมาใช้สิ่งที่เรียกว่า Attention เพียงอย่างเดียว โดยไม่ใช่แค่ Attention ธรรมดา แต่จะขยายขอบเขตและความสามารถของ Attention ขึ้นไป แล้วเรียก Architecture ใหม่นี้ว่า Transformer

หัวใจหลักของ Transformer คือกระบวนการที่เรียกว่า Self-Attention โดยกระบวนการนี้นอกจากจะเป็นสิ่งที่ทดแทน RNN และ CNN ได้แล้ว ยังแสดงถึงความข้องเกี่ยวกันของคำต่าง ๆ ในข้อความ ทำให้สามารถแก้ปัญหา Coreference Resolution หรือก็คือ การที่ทำให้คอมพิวเตอร์เข้าใจว่าสรรพนามนั้น ๆ อ้างอิงถึงคำ ๆ ไหนในประโยค ไปได้ในระดับที่ถือว่าค่อนข้างดี ซึ่งการแก้ปัญหานี้มีความสำคัญอย่างมากต่องาน NLP หลายประเภท เช่น Machine Translation เป็นต้น

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	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**	$2.3 \cdot 10^{19}$	

ภาพที่ 2.17 ตารางเปรียบเทียบผลของ Transfomer กับวิธีอื่น ๆ

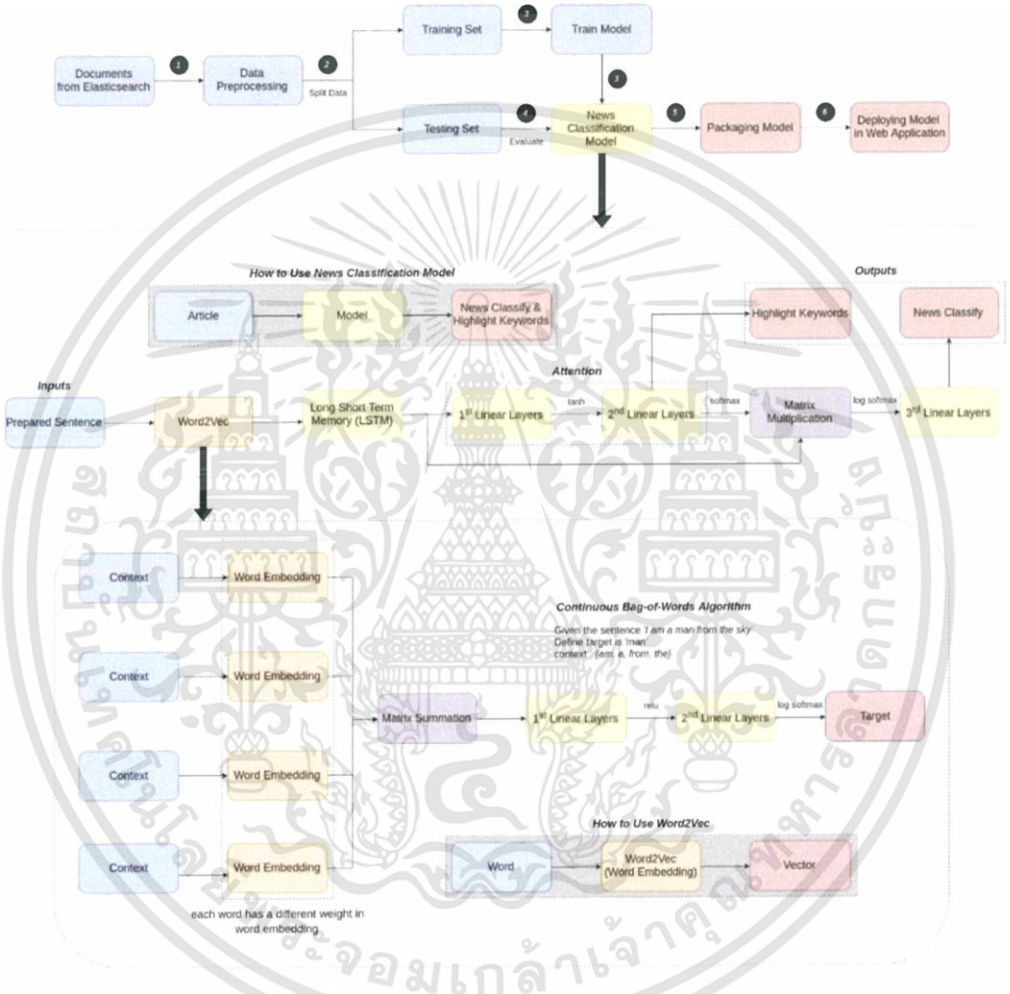
Transformer ให้ผลการทดลองในชุดข้อมูลมาตรฐานที่เหนือกว่าวิธีการอื่นอย่างชัดเจน ตัวอย่างเช่นในงาน Machine Translation ซึ่งใช้ชุดข้อมูล newstest2014 ทำการแปลจากภาษาอังกฤษเป็นภาษาเยอรมัน และภาษาอังกฤษเป็นภาษาฝรั่งเศส แล้ววัดผลโดยใช้ค่า BLEU ผลการเปรียบเทียบจะเป็นดังภาพที่ 2.17 ซึ่งนำมาจากงานวิจัยดังกล่าวโดยตรง จะเห็นว่าการแปลไปภาษาฝรั่งเศสให้ค่าดีกว่าการแปลไปภาษาเยอรมัน นั่นเป็นเพราะว่าข้อมูลของภาษาฝรั่งเศสมีเยอะกว่าหลายเท่า และภาษาเยอรมันจะมีโครงสร้างทางภาษาที่ซับซ้อนกว่า ซึ่ง Transformer สามารถอัพสกอร์ของภาษาเยอรมันได้มากกว่าวิธีอื่นถึงขนาดนี้โดยที่ใช้การคำนวณน้อยกว่าก็นับว่าให้ผลลัพธ์ที่ดีเลยทีเดียว

จากทั้งหมดที่ได้กล่าวมา ทางผู้จัดทำจึงได้เล็งเห็นว่ากระบวนการ Self-Attention น่าจะสามารถนำมาประยุกต์ใช้ได้ในโมเดลจำพวก Text Classification โดยเดิมทีจะใช้ LSTM ซึ่งเป็นหนึ่งในประเภทของ RNN ทำให้ผลลัพธ์ที่ได้ออกมาแม่นยำและมีประสิทธิภาพมากยิ่งขึ้น

บทที่ 3

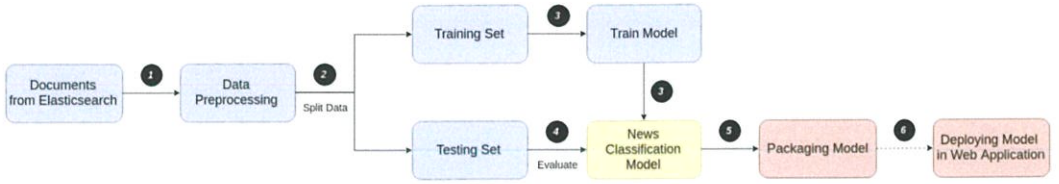
วิธีดำเนินการวิจัย

3.1. สถาปัตยกรรมของระบบ



ภาพที่ 3.1 สถาปัตยกรรมของระบบ

3.2. การทำงานของระบบ



ภาพที่ 3.2 ขั้นตอนการทำงานของระบบ

3.2.1 รวบรวมข้อมูลมาจากฐานข้อมูล

ตารางที่ 3.1 ข้อมูลทั้งหมดที่ใช้ในการเรียนรู้และวัดผลของโมเดล

	Blognone	Thairath	Dailynews	Sanook	Posttoday	Matichon	รวม
เศรษฐกิจ	0	20,277	11,382	12,371	6,689	16,748	67,467
บันเทิง	0	35,195	8,054	12,549	20,239	10,422	86,459
การเมือง	0	34,341	19,833	35,959	29,107	28,546	147,786
กีฬา	0	40,791	15,294	15,485	0	18,441	90,011
ต่างประเทศ	0	24,142	10,355	10,938	6,736	12,117	64,288
อาชญากรรม	0	0	9,793	11,914	22,988	0	44,695
ไอที	75,242	0	3,320	0	0	0	78,562
รวม	75,242	154,746	78,031	99,216	85,759	86,274	**579,268**

ข้อมูลที่นำมาใช้ในการสร้างโมเดลประกอบด้วย 579,268 โพสต์จากเว็บไซต์ข่าวต่าง ๆ ตามตารางที่ 3.1 ซึ่งนำมาจากฐานข้อมูลของบริษัท ซึ่งเก็บข้อมูลโดยใช้เทคนิคที่เรียกว่า Web Crawling ซึ่งเป็นการดึงข้อมูลจากหน้าเว็บไซต์อ้างอิงจากรูปแบบของ HTML ของเว็บไซต์นั้น ๆ ผ่าน Application Framework ที่มีชื่อว่า Scrapy แล้วเก็บในฐานข้อมูลที่ชื่อว่า Elasticsearch ซึ่งเป็นฐานข้อมูลแบบหนึ่งที่มีจุดเด่นในเรื่องของความสามารถในการค้นหาและสรุปข้อมูลที่มีขนาดใหญ่ได้อย่างรวดเร็วและมีประสิทธิภาพ โดยข้อมูลจะอยู่ในรูปแบบของ JSON Format เพื่อให้ง่ายต่อการใช้งาน โดยมีตัวอย่างข้อมูลดังต่อไปนี้

{

“title”: “นายกฯ ปิดเยือนเมืองเบียร์ ชวนเอกชนเยอรมนีลงทุนในไทย”,

“detail”: “นายกฯ ปิดเยือนเมืองเบียร์สวยงาม ชวนภาคเอกชนเยอรมนีลงทุนในไทย หวังขยายลู่ทางการลงทุนกลุ่มอียู ภูมิภาคยุโรป ชู ขับเคลื่อนแผนยุทธศาสตร์ชาติ 20 ปี หวังดันโครงสร้างพื้นฐาน นำไทยสู่ระเบียงเศรษฐกิจแห่งเอเชีย”,

“category”: “การเมือง”

}

3.2.2 การเตรียมข้อมูล (Data Preprocessing)

เนื่องจากข้อมูลที่ได้มานั้น โดยส่วนใหญ่จะไม่ใช่ข้อมูลที่พร้อมที่จะนำไปสร้างโมเดล จึงต้องมีการเตรียมข้อมูลให้พร้อม ไม่ว่าจะเป็นการตัดส่วนที่ไม่จำเป็นออก เช่น สัญลักษณ์ต่าง ๆ , เครื่องหมายวรรคตอน (Punctuation), url เป็นต้น เพื่อทำให้ข้อมูลเป็นไปอย่างที่ผู้พัฒนาต้องการ โดยแบ่งเป็น 3 ขั้นตอนดังนี้

3.2.2.1 การทำความสะอาดข้อมูล (Cleaning Data)

เป็นขั้นตอนที่จะทำการลบสิ่งที่ไม่จำเป็นออกดังนี้

- ลบ nbsp (normal blank space)
- ลบ html tags ได้แก่ < ... >
- ลบ http เช่น https://www.google.com เป็นต้น
- ลบ sign ได้แก่ _ \ \ / : - ; - , ~ =
- ลบ emoji
- ลบ Stopwords ภาษาไทย เช่น ที่ ซึ่ง อัน การ บาท หลัง ได้ เป็นต้น

3.2.2.2 การตัดคำ (Word Segmentation)

ในส่วนของการตัดคำ ทางผู้จัดทำได้เลือกใช้ Library ที่ชื่อว่า International Components for Unicode (ICU) ซึ่งเป็นผลิตภัณฑ์ของ IBM Corporation เพื่อใช้ในการตัดคำภาษาไทย เนื่องจากผลลัพธ์ที่ได้นั้น จะออกมาเป็นหน่วยคำที่เล็กที่สุด ยกตัวอย่าง เช่น “นายกฯ ปิดเยือนเมืองเบียร์ ชวนเอกชนเยอรมนีลงทุนในไทย” จะตัดคำได้เป็น นา|ยกฯ|ปิด|เยือน|เมือง|เบียร์|ชวน|เอกชน|เยอรมนี|ลงทุน|ใน|ไทยทำให้ง่ายในการทำ Word Embedding ในขั้นตอนต่อไป

3.2.2.3 การสร้างอินเด็กซ์

ทำการเปลี่ยนหมวดหมู่จากที่เป็น String ให้กลายเป็น Integer เพื่อทำให้คอมพิวเตอร์สามารถเข้าใจความหมายของ category นั้น ๆ ได้ โดยอ้างอิงจาก Dictionary ดังต่อไปนี้

```
{
    “เศรษฐกิจ”: 0, “บันเทิง”: 1, “การเมือง”: 2, “กีฬา”: 3,
    “ต่างประเทศ”: 4, “อาชญากรรม”: 5, “ไอที”: 6
}
```

หลังจากทำกระบวนการดังกล่าวเรียบร้อยแล้ว จะได้ข้อมูลออกมาดังนี้

```
{
    “data”: ['นา', 'ยกฯ', 'ปิด', 'เยือน', 'เมือง', 'เบียร์', 'ชวน', 'เอกชน', 'เยอรมนี', 'ลงทุน'],
    “category”: 2
},
{
    “data”: ['นา', 'ยกฯ', 'ปิด', 'เยือน', 'เมือง', 'เบียร์', 'สวยงาม', 'ชวน', 'ภาค', 'เอกชน',
    'เยอรมนี', 'ลงทุน', 'หวัง', 'ขยาย', 'ลู่ทาง', 'ลงทุน', 'กลุ่ม', 'อี', 'ยู', 'ภูมิภาค', 'ยุโรป', 'ชู', 'ขับ',
    'เคลื่อน', 'แผน', 'ยุทธศาสตร์', 'ชาติ', 'หวัง', 'ดัน', 'โครงสร้าง', 'พื้น', 'ฐาน', 'นำไทย', 'สู่',
    'ระเบียง', 'เศรษฐกิจ', 'เอเชีย'],
    “category”:  2
}
```

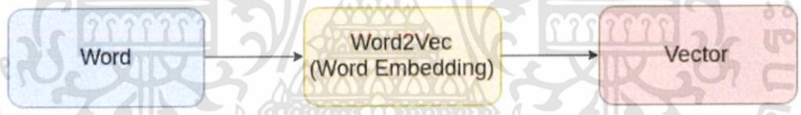
3.2.3 การแบ่งข้อมูลเพื่อเรียนรู้และทดสอบ (Split Data for Training and Testing Set)

ขั้นตอนนี้จะทำการแบ่งข้อมูลที่จะใช้ในการสร้างโมเดลออกเป็น 2 ส่วน ด้วยอัตราส่วนของ training set ต่อ testing set มีค่าเท่ากับ 90:10 เนื่องจากข้อมูลทั้งหมดที่รวบรวมมาจากฐานข้อมูลค่อนข้างมาก เพียงแค่ 10% ของข้อมูลทั้งหมดซึ่งเท่ากับ 57,927 ข้อมูล ถือว่าเพียงพอแล้วต่อการประเมินผล

ในแต่ละส่วนของข้อมูลที่แบ่ง จะนำไปใช้ต่างกัน โดย training set เป็นส่วนที่เอาไว้ใช้ในการเทรนโมเดล และ testing set เป็นส่วนที่เอาไว้ใช้ในการประเมินผล (evaluation) เพื่อดูว่าโมเดลที่ผ่านการเทรนมา มีประสิทธิภาพมากน้อยเพียงใด

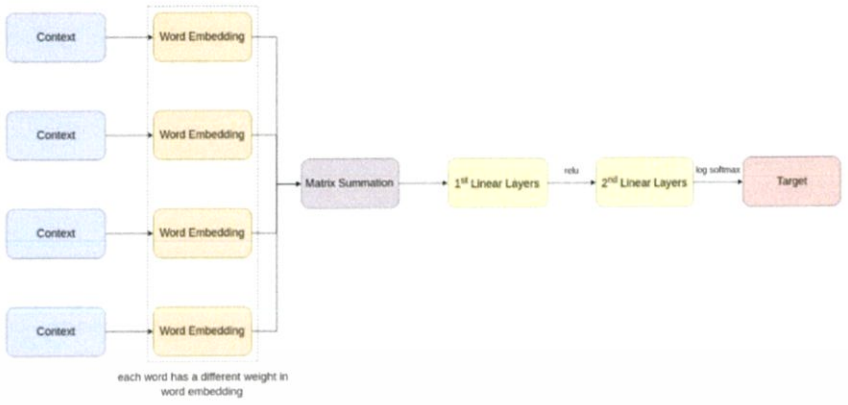
3.2.4 การแปลงคำเป็นเวกเตอร์ (Word2Vec)

กระบวนการนี้ มีจุดประสงค์เพื่อแปลงคำให้เป็นเวกเตอร์ เพื่อทำให้คอมพิวเตอร์เข้าใจในคำนั้น ๆ แล้วนำค่าเวกเตอร์ที่ได้มาเป็นอินพุตของโมเดลต่อไป โดยมีวิธีการใช้งานเบื้องต้นดังภาพที่ 3.3



ภาพที่ 3.3 วิธีใช้งาน Word2Vec

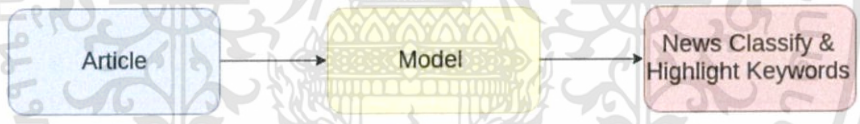
ทางผู้จัดทำได้เลือกใช้ Library ที่ชื่อว่า Gensim ในส่วนของ models.word2vec และเลือกใช้อัลกอริทึมที่มีชื่อว่า Continuous Bag-of-Words (CBOW) มาทำการ Implement ใช้งานกับโมเดล โดยมีกระบวนการการทำงานดังภาพที่ 3.4



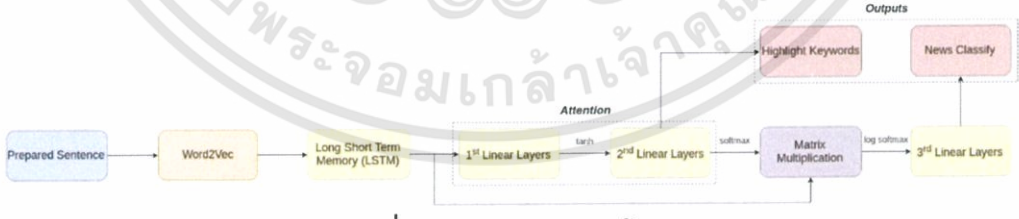
ภาพที่ 3.4 กระบวนการเทรนโมเดลโดยใช้อัลกอริทึม Continuous Bag-of-Words

3.2.5 โมเดลจำแนกบทความ (Article Classification Model)

เป็นโมเดลที่ใช้กระบวนการ Deep Learning มาทำการจำแนกประเภทของบทความ (article) ที่ใส่เข้าไปและทำการเน้นคำที่สำคัญของบทความนั้น ๆ โดยนำเสนอในรูปแบบของความเข้มของสี ดังภาพที่ 3.5 และมีส่วนของ Model มีกระบวนการทำงานดังภาพที่ 3.6 ซึ่งจะอธิบายแต่ละ Layers ของโมเดลดังนี้



ภาพที่ 3.5 วิธีใช้งาน Article Classification Model



ภาพที่ 3.6 การทำงานของโมเดล

ในส่วนของโมเดล มีการเซ็ตค่าพารามิเตอร์ ทั้งในส่วนของชั้น Layers ต่าง ๆ ของโมเดล และการเทรน ดังตารางที่ 3.1

ตารางที่ 3.2 ค่าพารามิเตอร์

ชื่อพารามิเตอร์	ค่าของพารามิเตอร์	หน้าที่ของพารามิเตอร์
batch_size	100	ค่าที่บอกว่าในแต่ละการ feed forward จะรับกี่ประโยค
max_len	300	จำนวนคำในแต่ละประโยคที่รับเข้ามาเทรน
emb_dim	100	Dimensions ของเวกเตอร์ที่ผ่านกระบวนการทำ Word Embedding
lstm_dim	600	LSTM Dimensions
att_dim	3000	Attention Dimensions
att_hops	10	ค่าที่บอกว่าในแต่ละ attention จะมองคำที่มีความสำคัญชั้นละกี่คำ
outputs_size	7	ประเภทของข่าวที่จะจำแนก
epoch_size	20	จำนวนรอบในการเทรนโมเดล
lr	0.06	Learning Rate

3.2.6 ตัวอย่างข้อมูลในแต่ละขั้นตอน

3.2.6.1 ตัวอย่างประโยค

เนื่องจากในแต่ละกระบวนการมีความซับซ้อนค่อนข้างมาก จึงขอยกตัวอย่างประโยคง่าย ๆ เพื่อเพิ่มความเข้าใจในแต่ละกระบวนการ โดยประโยคตัวอย่างเป็นพาดหัวข่าวที่นำมาจากเว็บไซต์ข่าว มีเนื้อความว่า “ข้าวโพดหวานไทยครองแชมป์ส่งออกอันดับ 1 ของโลกต่อเนื่อง 10 ปีซ้อน” และจัดอยู่ในหมวดเศรษฐกิจ

3.2.6.2 ประโยคหลังจากทำการเตรียมข้อมูล (Data Preprocessing)

หลังจากผ่านขั้นตอนการเตรียมข้อมูล (Data Preprocessing) ซึ่งมีด้วยกันอยู่ 3 ขั้นตอนตามที่ได้อธิบายไว้ในส่วนของข้อ 3.2.2 จะได้ข้อมูลออกมาดังต่อไปนี้

['ข้าวโพด', 'หวาน', 'ไทย', 'ครอง', 'แชมป์', 'ส่ง', 'ออก', 'อันดับ', ' ', '1', ' ', 'ของ', 'โลก', 'ต่อ', 'เนื่อง', ' ', '10', ' ', 'ปี', 'ซ้อน']

ข้อมูลข้างต้นเป็นข้อมูลที่ผ่านกระบวนการตัดคำแล้ว แต่ยังไม่ได้ทำการลบ Stopwords ภาษาไทยออก

['ข้าวโพด', 'หวาน', 'ครอง', 'แชมป์', 'อันดับ', 'โลก', 'เนื่อง', 'ซ้อน']

ข้อมูลข้างต้นเป็นข้อมูลที่ผ่านกระบวนการเตรียมข้อมูล (Data Preprocessing) เรียบร้อยแล้ว พร้อมที่นำไปสู่กระบวนการต่อไปนั่นคือ การแปลงคำให้เป็นเวกเตอร์ (Word2Vec)

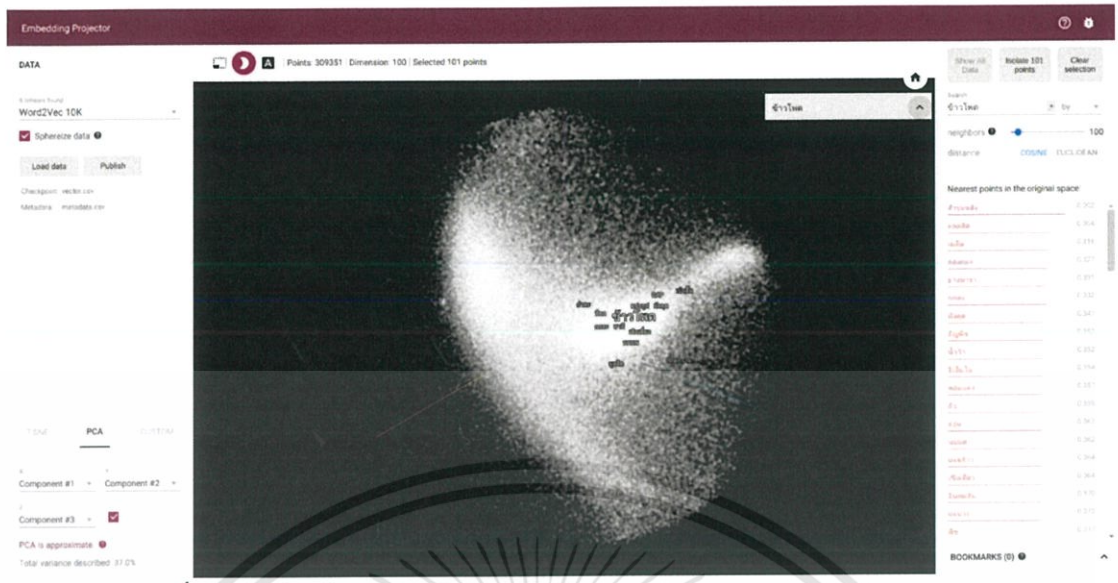
3.2.6.3 ข้อมูลหลังจากผ่านกระบวนการการแปลงคำเป็นเวกเตอร์ (Word2Vec)

หลังจากผ่านกระบวนการเตรียมข้อมูล (Data Preprocessing) เรียบร้อยแล้ว จะนำข้อมูลดังกล่าวมาทำการแปลงคำให้เป็นเวกเตอร์โดยอ้างอิงจากโมเดลที่ได้ทำการเรียนรู้ไว้แล้วในส่วนของข้อ 3.2.4 จะได้ข้อมูลออกมาดังต่อไปนี้

```
tensor([[-3.8977,  0.4165, -6.7135,  ..., -2.1381,  1.7662, -3.6179],
        [ 0.8091,  1.7815,  1.7168,  ..., -1.7170,  0.6998, -0.0141],
        [-5.7347, -1.2043, -3.8402,  ...,  1.2774, -1.5798,  2.3015],
        ...,
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000]])
```

จากข้อมูลดังกล่าว จะเห็นได้ว่า ข้อมูลมีลักษณะเป็น tensor ซึ่งเป็นชนิดตัวแปรแบบหนึ่งใน PyTorch โดยจะคล้ายกับ numpy array ที่ทำหน้าที่เป็นเหมือนเมทริกซ์ที่เก็บค่าต่าง ๆ ซึ่งแทนด้วยตัวเลข 100 ตัวต่อคำ 1 คำ ยกตัวอย่างคำว่า ข้าวโพด จะมีค่าเท่ากับ [-3.8977, 0.4165, -6.7135, ..., -2.1381, 1.7662, -3.6179] เป็นต้น โดยสามารถกำหนดค่าพารามิเตอร์ต่าง ๆ ได้ตามตารางที่ 3.2 ซึ่งค่าที่ใช้อ้างอิงทั้งหมดในกระบวนการนี้ ได้แก่ emb_dim ซึ่งจะเป็นค่าที่กำหนดว่า จะใช้ตัวเลขกี่ตัวแทนค่าคำ 1 คำ โดยทางผู้พัฒนาจะให้เท่ากับ 100 และ max_len จะเป็นค่าที่จะบอกว่า โมเดลจะรับคำทั้งหมดกี่คำมาประมวลผล โดยจะเห็นได้ว่าประโยคตัวอย่างที่ผ่านการเตรียมข้อมูลเสร็จแล้ว จะมีทั้งหมด 8 คำ แต่โมเดลจะต้องรับคำทั้งหมด 300 คำ โดยอ้างอิงจากตารางที่ 3.2 จึงต้องทำการเติม tensor ที่เก็บค่า 0 เป็นจำนวน 100 ตัวเปรียบเสมือนคำ 1 คำ ที่จะไม่มีผลต่อความหมายของข้อมูล แต่ต้องทำเพื่อให้ระบบสามารถทำงานได้เพิ่มเข้าไปด้านหลังของประโยคจนครบ 300 คำ ซึ่งจะเรียกกระบวนการดังกล่าวว่า Padding

โดยหากนำเอาค่าเหล่านี้มาทำการพล็อตกราฟผ่านวิธีการที่เรียกว่า Principal Component Analysis (PCA) ซึ่งกระบวนการนี้จะทำหน้าที่เสมือนบีบอัดตัวเลขทั้ง 100 ตัวให้เหลือน้อยลงโดยไม่เสียความสำคัญของคำ ๆ นั้นไป เพื่อนำไปแสดงให้เห็นในรูปของกราฟ 3 มิติ บนเว็บไซต์ projector.tensorflow.org ซึ่งเป็นเว็บที่ถูกพัฒนาโดย Tensorflow เพื่อใช้ในการแสดงผลลัพธ์ของการทำ Word Embedding ดังภาพที่ 3.7 ดังต่อไปนี้

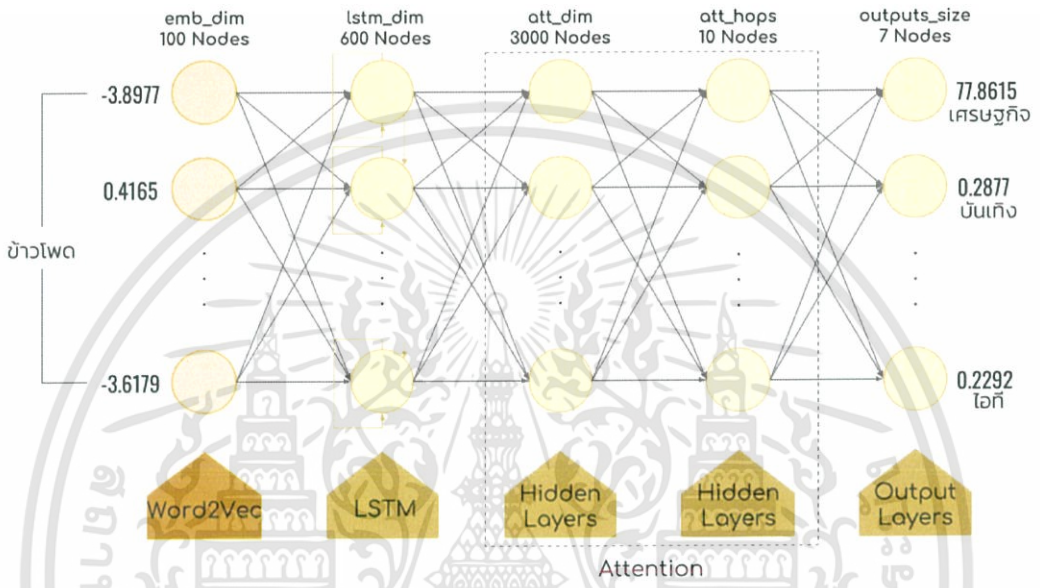


ภาพที่ 3.7 กราฟแสดงผลของการทำ Word2Vec ผ่าน Embedding Projector

จะภาพที่ 3.7 ทางฝั่งขวา จะแสดงให้เห็นว่า ระบบจะมองคำว่า ข้าวโพด มีความหมายใกล้เคียงกับคำว่า สำปะหลัง, ผลผลิต, เมล็ด, ลองกอง และอื่น ๆ ตามลำดับ เปรียบเสมือนมนุษย์ที่สามารถเรียนรู้ได้นั่นเอง

3.2.6.4 ข้อมูลหลังจากผ่านโมเดลจำแนกบทความ

หลังจากผ่านกระบวนการแปลงคำเป็นเวกเตอร์ (Word2Vec) เรียบร้อยแล้ว จะได้ผลลัพธ์เป็นคำทั้งหมดในประโยคในรูปของเวกเตอร์ จากนั้นกระบวนการต่อไปจะเป็นการเรียนรู้ของโมเดลเพื่อทำให้โมเดลประมวลผลให้ได้ผลลัพธ์ที่ถูกต้องและแม่นยำ โดยวิธีการเรียนรู้นั้น จะอธิบายดังภาพที่ 3.8 ดังต่อไปนี้



ภาพที่ 3.8 Neural Network ของโมเดลจำแนกบทความ

จากภาพที่ 3.8 เพื่อให้ง่ายต่อการเข้าใจ จึงยกตัวอย่างการทำงานเป็นแต่ละคำในประโยคต่อ ๆ กันไปเรื่อย ๆ เริ่มจากคำแรกนั่น คือ ข้าวโพด จากการแปลงคำเป็นเวกเตอร์จะได้เวกเตอร์ของคำว่า ข้าวโพด เท่ากับ [-3.8977, 0.4165, -6.7135, ..., -2.1381, 1.7662, -3.6179] ทำให้ส่วนของ Embedding Layers ต้องมีโหนดทั้งหมด 100 โหนดตามจำนวนตัวเลขใน 1 คำ โดยโหนดแรกมีค่าเท่ากับตัวแรกของเวกเตอร์ซึ่งเท่ากับ -3.8977 เป็นแบบนี้ไปเรื่อย ๆ ตามลำดับนั่นเอง จากนั้นจะเป็นการเรียนรู้ในส่วนของ Long Short-Term Memory (LSTM) และ Attention ตามลำดับ

จะเห็นได้ว่าการเรียนรู้ในส่วนของ LSTM นั้น มีความแตกต่างออกไปจาก Neural Network ปกติ เนื่องจาก LSTM จัดว่าเป็น Recurrent Neural Network (RNN) ประเภทหนึ่ง โดยวิธีการคิดและเรียนรู้นั้น หลัก ๆ แล้วจะมีการนำ output ที่ได้กลับมาคิดในส่วนของ input ของตัวถัดไปด้วย ซึ่งผู้พัฒนาได้อธิบายเพิ่มเติมอย่างละเอียดไว้ในส่วนของบทที่ 2 ทั้งในส่วนของ Long Short-Term Memory (LSTM) และ Attention แล้ว

```
{
        “เศรษฐกิจ”: 0, “บันเทิง”: 1, “การเมือง”: 2, “กีฬา”: 3,
        “ต่างประเทศ”: 4, “อาชญากรรม”: 5, “ไอที”: 6
}
```

จาก Dictionary ดังกล่าว จะเห็นได้ว่ามีหมวดของบทความอยู่ทั้งหมด 7 แบบ จึงกำหนดจำนวนของโหนดใน output layers ให้เท่ากับ 7 โดยโหนดแรกจะบอกความน่าจะเป็นที่บทความจะเป็นหมวดเศรษฐกิจในรูปของเปอร์เซ็นต์ซึ่งจะปรับเปลี่ยนไปเรื่อย ๆ ตามข้อมูลที่เราใส่เข้าไปเป็น input ของโมเดล แล้วเป็นแบบนี้ในทุก ๆ โหนด ตามลำดับของหมวดใน Dictionary นั่นเอง

3.2.7 Packaging Model

เป็นการแพ็กโมเดลเป็นแพ็กเกจเพื่อให้ง่ายต่อการใช้งาน โดยจะทำการ push ขึ้นไปที่ Version-Control System และ ถ้าหากจะนำไปใช้งานเพียงแค่ทำการ pull แล้วทำการติดตั้ง โดยขั้นตอนที่ทำการติดตั้งและใช้งานให้ทำตามขั้นตอนที่เขียนไว้ในไฟล์ README.md ดังภาพที่ 3.9

README.md

Backyard News Classification : nouvelle

Requirements

- Need to install PyICU first

```
$ sudo apt install libicu-dev
$ pip3 install pyicu
```

Package Build

```
$ python3 setup.py sdist bdist_wheel
```

Package Installation

```
$ pip3 install dist/nouvelle-1.0.0.tar.gz
```

How to Used

```
import nouvelle

word_vector = nouvelle.load_wordvector()
parameters = nouvelle.load_model_params()
model = nouvelle.load_model(parameters)
txt = '"นักตบสาวไทย" คว้า "อาเซอร์ไบจาน" ฉลุยรอบ 2 ลูกยางชิงแชมป์โลก'
nouvelle.news_classify(txt, word_vector, model ,parameters)
```

Result

```
result or news_classfiy pattern is (list of ranking classify, list of attention weight)
([('กีฬา', '99.9189'), ('ต่างประเทศ', '0.0772'), ('การเมือง', '0.0028'), ('บันเทิง', '0.0007'), ('เศรษฐกิจ', '0.0003'),
[('"', 0), ('นัก', 0), ('ตบ', 0.006077303551137447), ('สาว', 0.0059795081615448), ('ไทย', 0), ('"', 0), (' ', 0),
```

ภาพที่ 3.9 README.md

ผลลัพธ์ที่ได้จากฟังก์ชัน Classify ซึ่งเป็น คำสั่งที่เรียกใช้งาน Module ที่ทางผู้พัฒนาได้เขียนโปรแกรมไว้ หากป้อน input ตามตัวอย่างประโยคในข้อที่ 3.2.6.1 จะได้ผลลัพธ์ออกมาดังต่อไปนี้

([('เศรษฐกิจ', '77.8615'), ('การเมือง', '10.3279'), ('ต่างประเทศ', '10.2002'), ('กีฬา', '1.0847'), ('บันเทิง', '0.2877'), ('ไอที', '0.2292'), ('อาชญากรรม', '0.0089')], [('ข้าวโพด', 0.6813256070017815), ('หวาน', 0.5037288647145033), ('ไทย', 0), ('ครอง', 0.16056385356932878), ('แชมป์', 0.16375029226765037), ('ส่งออก', 0), ('อันดับ', 0.1602252246811986), ('1 ของ', 0), ('โลก', 0.46420274302363396), ('ต่อ', 0), ('เนื่อง', 0.3232695162296295), ('10 ปี', 0), ('ซ้อน', 0.16220981488004327)])

จากผลลัพธ์ดังกล่าว จะเห็นได้ว่าจะมี 2 ส่วน ได้แก่ ส่วนแรกเป็นส่วนที่บอกความน่าจะเป็นว่าบทความที่ป้อนเข้าไปจะมีโอกาสเป็นบทความประเภทไหนกี่เปอร์เซ็นต์ และส่วนที่สองเป็นส่วนที่จะบอกว่าคำ ๆ ไหนถูกให้ความใส่ใจมากในประโยคที่ป้อนเข้าไป ถ้ามีค่ามาก หมายถึงมีความสำคัญมาก โดยผู้พัฒนาจะนำค่านี้ไปแสดงผลผ่านความเข้มของสี (Opacity) โดยยิ่งค่าของคำไหนมาก สีที่พื้นหลังของคำนั้นก็จะมีความเข้มมากตามไปด้วย

3.2.8 เว็บแอพพลิเคชั่นจำแนกบทความ (Article Classification Web Application)

หลังจากที่ทำกระบวนการข้างต้นเสร็จสิ้นแล้ว จึงนำมาสู่ขั้นตอนที่ทำการสร้าง Web Application โดยวิธีการใช้งาน คือ User สามารถนำข่าวหรือบทความที่สนใจมาป้อนในช่อง Textfield แล้วกดปุ่ม submit จากนั้น Application จะแสดงผลลัพธ์ออกเป็น 2 ส่วน โดยส่วนแรกจะเป็นส่วนของการจำแนกข้อมูลที่ User ป้อนเข้ามา ว่ามีแนวโน้มที่บทความนั้นจะเป็นข่าวเกี่ยวกับเรื่องอะไร และส่วนที่สองจะแสดงให้เห็นว่าคำ ๆ ไหนในบทความที่สำคัญและเป็นเสมือนคีย์เวิร์ดที่ทำให้ระบบคิดว่าเป็นข่าวประเภทอะไรในส่วนแรก โดยแสดงให้เห็นในรูปแบบความเข้มของสี ซึ่งจะแปรผันตรงกับความสำคัญ กล่าวคือ ยิ่งสีเข้มมากเท่าใด หมายความว่า คำ ๆ นั้นยิ่งมีความสำคัญมากเท่านั้น

ภาพที่ 3.10 ตัวอย่างผลการจำแนกบทความ

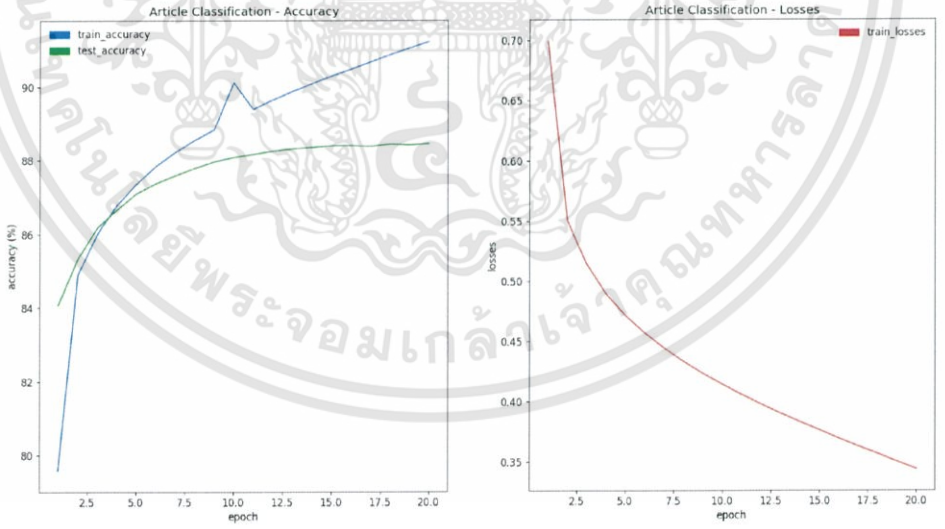
บทที่ 4
ผลการวิจัย

4.1. การวัดผลโดยใช้ความแม่นยำจากข้อมูลสำหรับการวัดผล (Split Test)

จากการทำงานของระบบ จะสังเกตได้ว่าก่อนขั้นตอนการเรียนรู้ของโมเดล จะมีการแบ่งข้อมูลทั้งหมดออกเป็น 2 ส่วนด้วยอัตราส่วน 90 ต่อ 10 แบ่งออกเป็น ข้อมูลที่นำไปให้โมเดลเรียนรู้ (Training Set) 521,341 ข้อมูล และ ข้อมูลสำหรับการวัดผล (Testing Set) 57,927 ข้อมูล โดยทำการวัดผลตามสมการดังนี้

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

จากสมการดังกล่าวจะได้ผลลัพธ์ที่นำเสนอในรูปของกราฟ โดยแกนนอน คือ รอบของการเรียนรู้ของโมเดล และ แกนตั้ง คือ ความแม่นยำทั้งในส่วนของการเทรนและการวัดผลของโมเดล และค่า Average Loss ของรอบการเทรนนั้น ๆ ดังภาพที่ 4.1



ภาพที่ 4.1 กราฟแสดงผลความแม่นยำและค่า Losses

จากภาพที่ 4.1 Accuracy สูงสุดของข้อมูลที่นำไปให้โมเดลเรียนรู้ (Training Set) และข้อมูลสำหรับการวัดผล (Testing Set) โดยใช้รอบการเทรน (epoch) 20 รอบ มีค่าเท่ากับ 91.22% ในรอบที่ 20 และ 88.45% ในรอบที่ 20 ตามลำดับ ในส่วนของค่า Losses มีการลดลงเรื่อย ๆ โดยต่ำสุดอยู่ที่รอบที่ 20 มีค่าเท่ากับ 0.3436

4.2. การวัดผลโดยใช้วิธี Confusion Matrix และ F-Measure

ก่อนจะทำการวัดผลโดยใช้วิธี Confusion Matrix และ F-Measure จำเป็นที่จะต้องรู้จักองค์ประกอบที่ใช้ในการวัดผลดังกล่าว โดยมีองค์ประกอบดังนี้

- True Positive (TP) คือ โมเดลทำนายว่าผลลัพธ์เป็นจริง และผลลัพธ์จริงเป็นจริง
- True Negative (TN) คือ โมเดลทำนายว่าผลลัพธ์เป็นเท็จ และผลลัพธ์จริงเป็นเท็จ
- False Positive (FP) คือ โมเดลทำนายว่าผลลัพธ์เป็นจริง และผลลัพธ์จริงเป็นเท็จ
- False Negative (FN) คือ โมเดลทำนายว่าผลลัพธ์เป็นเท็จ และผลลัพธ์จริงเป็นจริง
- Precision คือ ค่าที่บอกว่าโมเดลทำนายว่าผลลัพธ์เป็นจริง มีความถูกต้องเท่าไร โดยคิดจากสมการดังต่อไปนี้

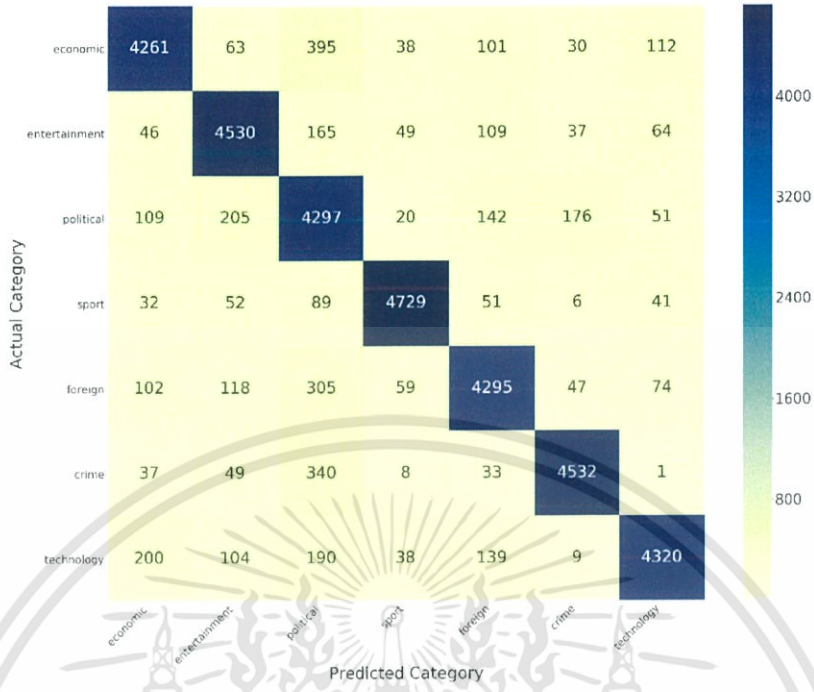
$$Precision = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)}$$

- Recall คือ ค่าที่บอกว่าโมเดลทำนายว่าผลลัพธ์เป็นจริง มีอัตราส่วนเท่าไรจากผลลัพธ์จริงทั้งหมด โดยคิดจากสมการดังต่อไปนี้

$$Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)}$$

- F - Measure คือ ค่าเฉลี่ยของ Precision และ Recall คิดจากสมการดังต่อไปนี้

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



ภาพที่ 4.2 Confusion Matrix

จากภาพที่ 4.2 Confusion Matrix ดังกล่าว สร้างมาจากการสุ่มเลือกข้อมูลในแต่ละหมวดของข้อมูลใน Testing Set หมวดหมู่ละ 5,000 ข้อมูล เพื่อเป็นการ Normalization ข้อมูล ทำให้ข้อมูลที่นำมาวัดผลนั้น ไม่เอนเอียงไปทางใดทางหนึ่งจนเกินไป จากนั้นจึงจะนำมาคิดค่า Precision, Recall และ F-Measure โดยจากสูตรการคำนวณที่ได้อธิบายไปข้างต้น จะได้ค่าทางสถิติต่าง ๆ ดังตารางที่ 4.1

ตารางที่ 4.1 ค่า Precision, Recall และ F - Measure

	PRECISION	RECALL	F - MEASURE
เศรษฐกิจ	0.8901	0.8522	0.8707
บันเทิง	0.8846	0.906	0.8952
การเมือง	0.7433	0.8594	0.7971
กีฬา	0.9571	0.9458	0.9514
ต่างประเทศ	0.8819	0.859	0.8703
อาชญากรรม	0.9369	0.9064	0.9214
ไอที	0.9264	0.864	0.8941
ค่าเฉลี่ย	**0.8886**	**0.8847**	**0.8858**

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1. สรุปผลการวิจัย

ระบบสามารถจำแนกบทความตามหัวข้อข่าวที่กำหนดได้อย่างถูกต้องและมีประสิทธิภาพ และสามารถหาสื่อที่ตรงตามความต้องการของลูกค้าได้มากขึ้น จากการหาข้อมูลที่เหมือนกันระหว่างสิ่งสื่อกำลังกล่าวถึง และสิ่งที่ลูกค้าอยากจะให้สื่อกล่าวถึง แต่ผลลัพธ์ที่ได้อาจจะยังไม่ถูกต้องทั้งหมด เนื่องจากความกำกวมของภาษาไทย ทำให้ไม่สามารถจำแนกได้ ต่อให้มนุษย์จะเป็นคนอ่านแล้วจำแนกก็ยังถือว่ายาก

5.2. ปัญหาอุปสรรคและข้อเสนอแนะ

- การตัดคำของภาษาไทยนั้นยังไม่มีเครื่องมือตัวไหนที่สามารถตัดคำได้อย่างถูกต้องทั้งหมด เนื่องจากความกำกวมของภาษาไทย เช่น คำว่า ตากลม อาจจะตัดเป็นคำว่า ตา|กลม หรือ ตาก|ลม หรือ ตากลม ทั้ง 3 รูปแบบนี้ ตัวโมเดลจะมองเป็นคนละตัวแปรกันทั้งสิ้น อีกทั้งยังมีจำนวนคำที่ไม่เท่ากัน ทำให้เกิดโอกาสผิดพลาดในการเรียนรู้ของโมเดล สามารถแก้ไขปัญหาได้ด้วยการเลือกเครื่องมือการตัดคำที่เหมาะสมทั้งในส่วนของข้อมูลที่นำมาใช้ และผลลัพธ์ที่ต้องการให้มากที่สุด
- การที่จะให้โมเดลนั้นเรียนรู้อย่างมีประสิทธิภาพในช่วงเวลาที่จำกัด จำเป็นที่จำต้องใช้ทรัพยากรทางคอมพิวเตอร์จำนวนมาก ไม่ว่าจะเป็นในส่วนของ RAM ที่ใช้ในการเก็บข้อมูลในช่วงเวลาที่เตรียมข้อมูลและให้โมเดลเรียนรู้ และหน่วยประมวลผลต่าง ๆ ที่มีผลต่อความเร็วในกระบวนการต่าง ๆ ของโมเดล ทั้งในส่วนของ CPU ที่ใช้การเตรียมข้อมูล และ GPU ที่ใช้การเรียนรู้และวัดผลของโมเดล สามารถแก้ปัญหาได้ด้วยการเพิ่มทรัพยากรดังที่กล่าวมา หรือเลือกใช้บริการเช่าเครื่องเซิฟเวอร์แบบออนไลน์ประเภท Deep Learning Platform เพื่อทดแทนทรัพยากรที่ไม่เพียงพอ
- เนื่องจากโมเดลดังกล่าวยังไม่มีการนำไปใช้งานจริงในทางธุรกิจ ณ เวลาปัจจุบัน ทำให้การปรับปรุงโมเดลโดยดูจากความพึงพอใจของผู้ใช้งานนั้น ยังไม่สามารถทำได้ สามารถแก้ไขได้โดยการทดลองและวัดผลจากผู้พัฒนาและทีมงานของทางบริษัท ทำให้สามารถปรับปรุงข้อบกพร่องส่วนนี้ได้ในระดับหนึ่ง

บรรณานุกรม

[1] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. (2013). *Efficient Estimation of Word Representations in Vector Space*. Retrieved October 15, 2018

[2] Sepp Hochreiter, Jurgen Schmidhuber. (1997). *Long Short-Term Memory. Neural Computation, 9(8): 1735–1780*. Retrieved September 30, 2018

[3] *Understanding LSTM Networks.* Retrieved September 24, 2018, from http://colah.github.io/posts/2015-08-Understanding-LSTMs

[4] Richard Socher. (2018). *Natural Language Processing with Deep Learning CS224N/Ling284 Lecture 11: Paying attention to attention and Tips and Tricks for large MT*. Retrieved August 22, 2018

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. (2017). *Attention Is All You Need*. Retrieved October 31, 2018

[6] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, Yoshua Bengio. (2017). *A Structured Self-Attentive Sentence Embedding*. Retrieved October 24, 2018

[7] *Docker Documentation – Get Started*. Retrieved August 1, 2018, from https://docs.docker.com/get-started

[8] *Welcome to Flask.* Retrieved November 12, 2018, from http://flask.pocoo.org/docs/1.0/

[9] Scott Chacon, Ben Straub. (2018). *Pro Git.* Retrieved August 6, 2018