การสรุปข้อความข่าวภาษาไทย
โดยใช้แบบจำลองการเรียนรู้เชิงลึก

Thai News Text Summarization
Using Deep Learning Model

ธีรพงศ์ แต่งเมือง

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

Thai News Text Summarization

Using Deep Learning Model

TEERAPONG TAENGMUANG

A SPECIAL PROBLEM SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR

THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)

DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2018

หัวข้อปัญหาพิเศษ การสรุปข้อความข่าวภาษาไทยโดยใช้แบบจำลองการเรียนรู้เชิงลึก
Thai News Text Summarization Using Deep Learning Model

ชื่อนักศึกษา นายธีรพงศ์ แต่งเมือง รหัสนักศึกษา 58050290

ปริญญา วิทยาศาสตรบัณฑิต

ภาควิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษา ดร.กุลสวัสดิ์ จิตขจรวานิช

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.) อนุมัติให้ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2561

คณะกรรมการสอบ	ลายมือชื่อ
ผศ.ดร.สายชล ใจเย็น ประธานกรรมการ	
ผศ.ดร.อนันตพร หรรษคุณาฒัย กรรมการ	
ดร.กุลสวัสดิ์ จิตขจรวานิช กรรมการและอาจารย์ที่ปรึกษา	

หัวข้อปัญหาพิเศษ การสรุปข้อความข่าวภาษาไทยโดยใช้แบบจำลองการเรียนรู้เชิงลึก
ชื่อนักศึกษา นายธีรพงศ์ แต่งเมือง รหัสนักศึกษา 58050290
ปริญญา วิทยาศาสตรบัณฑิต
ภาควิชา วิทยาการคอมพิวเตอร์
คณะ วิทยาศาสตร์
มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)
ปีการศึกษา 2561
อาจารย์ที่ปรึกษา ดร.กุลสวัสดิ์ จิตขจรวานิช

บทคัดย่อ

เนื่องด้วยการศึกษาและพัฒนาเทคนิคการสรุปข้อความสำหรับภาษาไทยยังมีข้อจำกัด ปัญหาพิเศษนี้มีวัตถุประสงค์เพื่อแก้ไขปัญหาดังกล่าวโดยการประยุกต์ใช้เทคนิคการเรียนรู้เชิงลึกซึ่งเป็นศาสตร์ที่กำลังได้รับความนิยม ข้อมูลที่นำมาใช้ในการศึกษาครั้งนี้ได้รวบรวมมาจากข้อมูลข่าวจากสถานีโทรทัศน์ไทยพีบีเอสโดยเลือกเฉพาะส่วนของหัวข้อข่าวและข่าวย่อ จำนวน 29,170 ข่าว มาใช้ในการสร้างแบบจำลองการเรียนรู้ของเครื่อง จากการดำเนินงานพบว่าเป็นการเริ่มต้นศึกษาและมีแนวโน้มในการพัฒนาไปในทางที่ดี อย่างไรก็ตามผลการวัดประสิทธิภาพด้วยตัวชี้วัด ROUGE พบว่ายังมีความถูกต้องน้อย ซึ่งคาดว่าต้องใช้ข้อมูลข่าวอีกจำนวนมากในการสร้างแบบจำลองการเรียนรู้เชิงลึก อีกทั้งยังต้องมีการศึกษาและทดลองนำอัลกอริทึมใหม่ๆ มาประยุกต์ใช้กับการสรุปข้อความข่าวภาษาไทยอีกจำนวนมากเพื่อให้ได้ผลลัพธ์ที่ดียิ่งขึ้น

คำสำคัญ: การประมวลผลภาษาธรรมชาติ การเรียนรู้ของเครื่อง การเรียนรู้เชิงลึก การสรุปข้อความ การสรุปข่าว

Title	Thai News Text Summarization Using Deep Learning Model
Students	Mr. Teerapong Taengmuang Student ID 58050290
Degree	Bachelor of Science
Department	Computer Science
Faculty	Science
University	King Mongkut's Institute of Technology Ladkrabang (KMITL)
Academic Year	2018
Advisor	Dr. Kulsawasd Jitkajornwanich

Abstract

Designing and developing an efficient text summarization tool for Thai language is still considered to be a challenge. The goal of this special problem is to overcome such challenge by applying deep learning technique, one of the increasingly popular fields of study in the era of big data. The Thai news dataset used in this work is collected from Thai PBS TV station, consisting of 29,170 news in which only news headlines and short news were used in developing a machine learning model. The results showed that the proposed model has a good potential evidenced by a performance evaluation, called ROUGE. However, there is still room for improvements especially for the accuracy aspect. In future work, we expect that with more and larger news datasets, our deep learning-based model will perform better. In addition, applying more diverse/new algorithms of text summarization should also be considered in improving the overall performance of the Thai news text summarization system.

Keywords: Natural Language Processing, Machine Learning, Deep Learning, Text Summarization, News Summarization

กิตติกรรมประกาศ

โครงงานพิเศษฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี เนื่องจากความช่วยเหลือและการสนับสนุนจากบุคคลหลายๆ ท่าน ขอขอบคุณอาจารย์ที่ปรึกษา ดร.กุลสวัสดิ์ จิตขจรวานิช ผู้ซึ่งให้คำปรึกษาและคำแนะนำอันเป็นประโยชน์ต่อการปรับปรุงข้อบกพร่องในการจัดทำโครงงานพิเศษ ขอขอบคุณ ผศ.ดร.สายชล ใจเย็น และ ผศ.ดร.อนันตพร หรรษคุณาฒัย ประธานกรรมการและกรรมการ ที่ให้ข้อคิดเห็นและข้อเสนอแนะในการจัดทำโครงงานพิเศษให้สำเร็จลุล่วงไปได้ด้วยดี

ขอกราบขอบพระคุณครูบาอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ให้แก่ข้าพเจ้าตั้งแต่เยาว์วัยจนข้าพเจ้ามีความรู้ความสามารถในการจัดทำโครงงานพิเศษนี้ได้

และสุดท้ายข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และบุคคลในครอบครัว ที่ให้โอกาสในการศึกษาอย่างเต็มที่ ตลอดจนเลี้ยงดู อบรมสั่งสอน พร้อมทั้งให้กำลังใจและความรักเสมอมา

ธีรพงศ์ แต่งเมือง

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหาพิเศษ

ในปัจจุบันมีข้อมูลในรูปแบบข้อความจำนวนมากซึ่งผู้อ่านต้องใช้เวลาในการอ่านและทำความเข้าใจกับเนื้อหา แต่ในบางโอกาสผู้อ่านเพียงต้องการทราบถึงใจความสำคัญหรือสาระสำคัญเท่านั้น

จากปัญหาดังกล่าวจึงมีการพยายามพัฒนาเครื่องมือและวิธีการสำหรับสรุปข้อความตลอดมาเพื่อลดเวลาในการอ่านและสามารถเข้าใจถึงสาระสำคัญได้อย่างรวดเร็ว

การเรียนรู้เชิงลึก (Deep Learning) เป็นสาขาวิชาที่ได้รับความนิยมมากขึ้นเรื่อย ๆ เนื่องจากสามารถแก้ปัญหาที่วิธีการอื่นไม่สามารถแก้ปัญหาได้หรือให้ความถูกต้องมากกว่าวิธีการอื่นในบางปัญหา ด้วยความสามารถดังกล่าวทำให้แบบจำลองการเรียนรู้เชิงลึกได้รับการศึกษาและพัฒนาอย่างแพร่หลาย

ในปัญหาพิเศษนี้ในได้นำแบบจำลองการเรียนรู้เชิงลึกสำหรับการประมวลผลธรรมชาติ (Natural Language Processing) คือแบบจำลอง Sequence-to-Sequence (Seq2Seq) มาปรับใช้กับข้อความข่าวภาษาไทยเพื่อสรุปข้อความข่าวให้สั้นลงและเป็นแนวทางในการพัฒนาการสรุปข้อความในภาษาไทยต่อไป

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1) เพื่อลดเวลาในการอ่านข่าว
2) เพื่อทดลองนำแบบจำลองประเภท Sequence-to-Sequence มาปรับใช้กับการสรุปข้อความข่าวภาษาไทย
3) เพื่อพัฒนาองค์ความรู้ในสาขาวิชาการประมวลผลภาษาธรรมชาติในภาษาไทย

1.3 ขอบเขตของปัญหาพิเศษ

1) ใช้ Bahdanau ในการสร้างแบบจำลองการเรียนรู้ของเครื่อง
2) ใช้ Word2Vec ในการสกัดคุณลักษณะจากข้อความข่าว
3) รวบรวมข่าวจากสถานีโทรทัศน์ไทยพีบีเอส (Thai Public Broadcasting Service: ThaiPBS)
4) ตัวตัดคำภาษาไทย คือ newmm

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ลดเวลาในการอ่านข่าว
2) เป็นแนวทางในการศึกษาการสรุปข้อความข่าวภาษาไทยโดยใช้แบบจำลองการเรียนรู้เชิงลึกและแบบจำลองประเภท Sequence-to-Sequence
3) พัฒนาองค์ความรู้ในสาขาวิชาการประมวลผลภาษาธรรมชาติในภาษาไทย

1.5 ขั้นตอนการดำเนินงาน

1) ศึกษางานวิจัยและทฤษฎีที่เกี่ยวข้อง
2) ศึกษาการสกัดคุณลักษณะด้วยเทคนิค Word2Vec
3) ศึกษาการแบบจำลองประเภท Sequence-to-Sequence
4) ศึกษาหาเครื่องมือที่เหมาะสมในการทำปัญหาพิเศษ
5) จัดหาข้อมูลและจัดเตรียมข้อมูลให้เหมาะสม
6) สร้างแบบจำลองการเรียนรู้ของเครื่องและทดสอบประสิทธิภาพ
7) สรุปผลและแนะนำแนวทางการพัฒนาต่อ

1.6 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

1) ฮาร์แวร์

   - NVIDIA GeForce GTX 1060 หน่วยความจำ 6 GB

2) ซอฟแวร์

   - ระบบปฏิบัติการ Windows 10 Home 64-bit
   - Python 3.6.5
   - TensorFlow-GPU 1.13.1
   - PyThaiNLP 2.0.5
   - NLTK 3.3
   - CUDA Toolkit 10.0

บทที่ 2
ทฤษฎีและเอกสารที่เกี่ยวข้อง

2.1 การตัดคำ

การตัดคำ (Word Segmentation) คือกระบวนการหาขอบเขตของคำ ซึ่งวิธีการตัดคำในแต่ละภาษามีขั้นตอนวิธีต่างกันออกไป ขึ้นอยู่กับโครงสร้างและลักษณะเฉพาะของแต่ละภาษา ตัวอย่างเช่น ภาษาอังกฤษมีการเว้นระหว่างคำทำให้ง่ายต่อการตัดคำ สำหรับภาษาไทยเนื่องจากมีลักษณะการเขียนคำที่ติดต่อกันไปจนจบประโยคหรือวลีจึงทำให้ยากต่อการตัดคำ ดังนั้นการตัดคำภาษาไทย (Thai Word Segmentation) จึงมีการพัฒนาขั้นตอนวิธีต่าง ๆเพื่อสร้างตัวตัดคำภาษาไทย (Thai Tokenizer) โดยมีความแตกต่างกันออกไปตามกาลเวลาและเพื่อให้เหมาะสมต่อการใช้งาน ซึ่งเกณฑ์การตัดคำภาษาไทย มีดังนี้

ตารางที่ 2.1 เกณฑ์การตัดคำภาษาไทย

เกณฑ์	การตัดคำ
อิงตามคำมูล	ฉัน|นำ|ดอก|ไม้|ไป|ไหว้|ศาล|พระ|ภูมิ|ที่|โรง|เรียน|ประจำ|
อิงตามคำมูลที่รวมกันแน่น	ฉัน|นำ|ดอกไม้|ไป|ไหว้|ศาลพระ|ภูมิ|ที่|โรงเรียน|ประจำ|
อิงตามคำประสมเปลี่ยนความหมาย	ฉัน|นำ|ดอก|ไม้|ไป|ไหว้|ศาล|พระภูมิ|ที่|โรง|เรียน|ประจำ|
อิงตามหน่วยความหมาย	ฉัน|นำ|ดอกไม้|ไป|ไหว้|ศาลพระ|ภูมิ|ที่|โรงเรียนประจำ|

จากตารางที่ 2.1 แสดงเกณฑ์การตัดคำภาษาไทยและผลที่ได้จากการตัดคำตามเกณฑ์นั้น ๆ ของประโยค “ฉันนำดอกไม้ไปไหว้ศาลพระภูมิที่โรงเรียนประจำ” ซึ่งการพิจารณาในการใช้เกณฑ์ขึ้นอยู่กับความเหมาะสมต่อการใช้การ ตัวอย่างเช่น การประยุกต์การตัดคำกับการแปลภาษาจะใช้เกณฑ์การตัดคำอิงตามหน่วยความหมาย เป็นต้น

2.1.1 วิธีการตัดคำ

ขั้นตอนวิธีสำหรับตัดคำมีหลากหลายวิธี ซึ่งในแต่ละขั้นตอนวิธีจะให้ประสิทธิภาพที่แตกต่างกันออกไป โดยส่วนใหญ่แล้วจะแบ่งวิธีการตัดคำภาษาไทยออกเป็น 3 ประเภท ได้แก่ ใช้กฎ (Rule-based) ใช้พจนานุกรม (Dictionary-based) และใช้คลังข้อความ (Corpus-based)

2.2.1.1 ใช้กฎ

วิธีการตัดคำโดยใช้กฎ (Rule-based) คือขั้นตอนวิธีที่พัฒนาขึ้นโดยการอาศัยหลักไวยากรณ์ภาษาไทยเพื่อสร้างกฎขึ้น ซึ่งถูกพัฒนาขึ้นในยุคแรกเริ่มของการพัฒนาตัวตัดคำภาษาไทย โดยงานวิจัยเน้นไปที่การวิเคราะห์ในระดับคำหรือพยางค์เป็นส่วนใหญ่

2.2.1.2 ใช้พจนานุกรม

วิธีการตัดคำโดยใช้พจนานุกรม (Dictionary-based) คือขั้นตอนวิธีที่พัฒนาขึ้นโดยใช้สายอักขระ (String) เปรียบเทียบกับคำที่จัดเก็บในพจนานุกรม

2.2.1.3 ใช้คลังข้อความ

วิธีการตัดคำโดยใช้คลังข้อความ (Corpus-based) คือขั้นตอนวิธีที่พัฒนาขึ้นโดยการสร้างแบบจำลองภาษา (Language Model) ด้วยเทคนิคต่าง ๆ ด้วยคลังข้อความ เช่น วิธีการทางสถิติ การเรียนรู้ของเครื่องจักร เป็นต้น

ตารางที่ 2.2 การเปรียบเทียบข้อดีและข้อเสียของวิธีการตัดคำภาษาไทย

วิธีการ	ข้อดี	ข้อเสีย
ใช้กฎ	1) สามารถตัดคำระดับพยางค์ได้ดี	1) กฎสามารถสร้างได้ยาก
ใช้พจนานุกรม	1. สามารถเพิ่มคำศัพท์ใหม่ในพจนานุกรมได้	1) การรวบรวมพจนานุกรมต้องใช้ทรัพยากรมนุษย์ในการจัดการสูง
ใช้คลังข้อความ	1) สามารถสร้างกฎต่าง ๆ ได้โดยไม่ต้องวิเคราะห์หลักภาษา 2) สามารถตัดคำศัพท์ที่ไม่รู้จักหรือคำผิดได้บางกรณี	1) ระยะเวลาในการเรียนรู้สูงเนื่องจากความซับซ้อนของขั้นตอนวิธี 2) ต้องการคลังข้อความที่ถูกต้องและเหมาะสมจำนวนมาก

2.1.2 เทคนิคการตัดคำ

เทคนิคสำหรับการตัดคำในภาษาไทยถูกพัฒนามาอย่างต่อเนื่อง แต่ละเทคนิคให้ผลลัพธ์ที่แตกต่างกันดังตัวอย่างการตัดคำภาษาไทยในตารางที่ 2.1 โดยในกาวิจัยนี้มีเทคนิคที่เกี่ยวข้องกับการตัดคำภาษาไทย ดังนี้

2.1.2.1 เทคนิคการตัดคำด้วยกลุ่มตัวอักษรไทย [1]

เทคนิคการตัดคำด้วยกลุ่มตัวอักษรไทย (Thai Character Cluster: TCC) [1] เป็นขั้นตอนวิธีโดยใช้การแยกแต่ละอักขระเพื่อคำการวิเคราะห์ก่อนทำการกำหนดตำแหน่งขอบเขตของคำ ซึ่งอาศัยแนวคิดว่ามีตัวอักษรบางตัวที่ทราบว่าไม่ควรตัดคำตำแหน่งนั้น ตัวอย่างเช่น Cา จะไม่สามารถตัดคำหน้าสระอาได้ ซึ่งสามารถแก้ปัญหาคำที่ไม่รู้จัก (Unknown Word) ได้ในบางกรณี

2.1.2.2 เทคนิคการตัดคำแบบเลือกคำยาวที่สุด

เทคนิคการตัดคำแบบเลือกคำยาวที่สุด (Longest Matching) เป็นเทคนิคที่ใช้หลักการตัดคำโดยใช้พจนานุกรม โดยใช้สายอักขระ (String) เปรียบเทียบกับคำที่จัดเก็บในพจนานุกรม โดยเลือกคำที่มีความยาวมากที่สุดในพจนานุกรม ตัวอย่างเช่น

ตามหามเหสี = ตาม|หาม|เห|สี|

เธอนั่งตากลมที่ชายหาด = เธอ|นั่ง|ตาก|ลม|ที่|ชายหาด|

2.1.2.3 เทคนิคการตัดคำแบบสอดคล้องมากที่สุด

เทคนิคการตัดคำแบบสอดคล้องมากที่สุด (Maximal Matching) เป็นเทคนิคที่พัฒนาขึ้นเพื่อแก้ปัญหาของเทคนิคการตัดคำแบบเลือกคำยาวที่สุด ซึ่งมีข้อผิดพลาด

จากการเลือกคำที่มีความยาวมากที่สุด เช่น ตาม|หาม|เห|สี|

เทคนิคการตัดคำแบบสอดคล้องมากที่สุดจะทำการค้นหาเส้นทางของการแบ่งคำที่เป็นไปได้ทั้งหมด และทำการเลือกเส้นทางที่มีคำนวณคำที่น้อยที่สุด ตัวอย่างเช่น

ตามหามเหสี = ตาม|หาม|เห|สี| = 4 คำ
= ตาม|หา|มเหสี| = 3 คำ (ถูกเลือก)

ในกรณีที่เส้นทางการแบ่งคำทั้งหมดมีจำนวนของคำเท่ากันจะใช้เทคนิคการตัดคำแบบยาวที่สุด ตัวอย่างเช่น

เธอนั่งตากลมที่ชายหาด = เธอ|นั่ง|ตาก|ลม|ที่|ชายหาด| = 7 คำ (ถูกเลือก)
= เธอ|นั่ง|ตา|กลม|ที่|ชายหาด| = 7 คำ

2.2 Word2Vec [2]

Word2Vec คือการนำเสนอคำ (Word Represent) ในรูปแบบเวกเตอร์ โดยจะมีคุณสมบัติเหมือนกับค่าเวกเตอร์ เช่น ค่าของเวกเตอร์มีความใกล้เคียงกันมาก หมายความว่าคำนั้น ๆมีความสัมพันธ์กันมาก

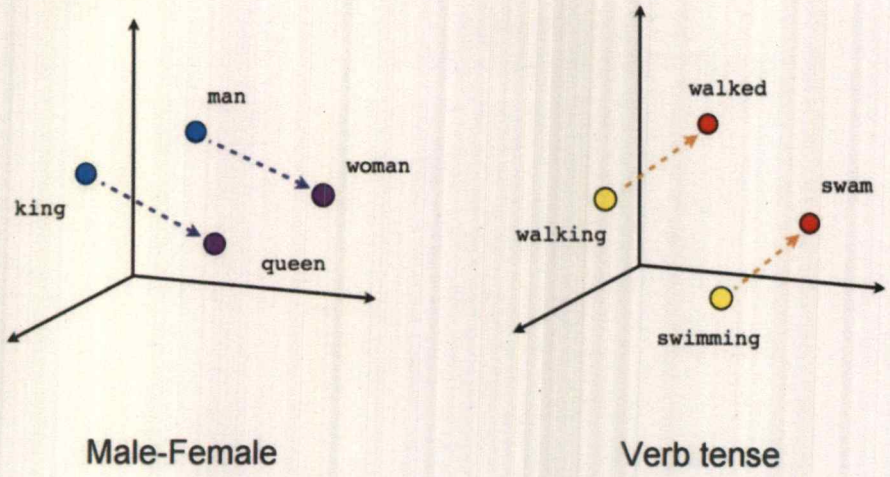
การสร้างค่าเวกเตอร์ของคำสามารถทำได้โดยการใช้โครงข่ายประสาทเทียมร่วมกับแบบจำลองทางภาษา (Language Model) ซึ่งมี 2 แบบจำลอง คือ Skip-Gram model และ CBOW (Continuous Bag-of-Words) ซึ่งมีจุดประสงค์ในการใช้งานแบบจำลองที่แตกต่างกันดังนี้

การสร้างค่าเวกเตอร์ด้วยแบบจำลอง CBOW ในการประยุกต์ใช้งานจะใช้เพื่อการทำนายคำที่ขาดหายไปในบริบท ตัวอย่างเช่น

ประโยคที่ 1: อยากไปเที่ยว___ที่มีริมชายหาดสวยๆ
ประโยคที่ 2: อากาศร้อนมากอยาก___อะไรที่มันเย็นๆ

จากประโยคที่ 1 และประโยคที่ 2 ในช่องว่างมีคำที่ขาดหายไปซึ่งเมื่อพิจารณาจากบริบทแล้วสามารถมีคำที่นำมาเติมได้หลายคำ ซึ่งการสร้างแบบจำลอง Word2Vec ด้วยแบบจำลอง CBOW จะสามารถค้นหาคำที่เหมาะสมต่อบริบทรอบข้างได้

การสร้างค่าเวกเตอร์ด้วยแบบจำลอง Skip-Gram ในการประยุกต์ใช้งานจะใช้เพื่อการทำนายบริบทของคำว่ามีความเกี่ยวข้องกับบริบทใด ซึ่งบริบทคือคำหรือข้อความแวดล้อมเพื่อช่วยให้เข้าใจความหมาย กล่าวคือ การประยุกต์ใช้งานจะใช้เพื่อการทำนายว่าคำหนึ่งๆมีความเกี่ยวข้องกับคำใดบ้าง และจากหลักสำคัญนี้เองทำให้สามารถทราบได้ว่าคำใด ๆในภาษามีความสัมพันธ์กันอย่างไร เช่น มีความคล้ายคลึงกันมากน้อยเพียงใด และด้วยความสามารถของแบบจำลองนี้ ทำให้ได้รับความนิยมและนำประยุกต์ใช้งานอย่างแพร่หลาย เช่น การจำแนกประเภทข้อความ (Text Classification) เป็นต้น



รูปที่ 2.1 แสดงความสัมพันธ์ของคำ

(ที่มา: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c)

จากรูปที่ 2.1 ด้านซ้ายแสดงความสัมพันธ์ของคำว่า “king”, “man”, “queen”, “woman” โดยแสดงความสัมพันธ์ของคำว่า “king”, “man” เป็นเพศชาย “queen”, “woman” เป็นเพศหญิง หรือ “king”, “queen” จะใช้ในบริบทเดียวกันและ “man”, “woman” จะใช้ในบริบทเดียวกัน

จากรูปที่ 2.1 ด้านขวาแสดงความสัมพันธ์ของคำว่า “walking”, “walked”, “swimming”, “swam” โดยแสดงความสัมพันธ์ของคำว่า “walking”, “walked” มีรากศัพท์เป็นคำเดียวกัน “swimming”, “swam” มีรากศัพท์เป็นคำเดียวกัน หรือ “walking”, “swimming” ใช้ในรูปแบบของกริยา (Tense) เดียวกัน และ “walked”, “swam” ใช้ในรูปแบบของกริยาเดียวกัน

2.3 ขนาดของแบตช์ (Batch Size)

คือการแบ่งข้อมูลออกเป็นส่วนๆ เพื่อการจำกัดการใช้ทรัพยากรการประมวลผล เนื่องจากจำนวนข้อมูลในการสร้างแบบจำลองมีจำนวนมาก แต่ในการสร้างแบบจำลองมีทรัพยากรไม่เพียงพอสำหรับนำข้อมูลเข้าสู่หน่วยคำจำเพื่อประมวลผล ดังนั้นจึงต้องแบ่งข้อมูลออกเป็นส่วนๆ เช่น ขนาดของแบตช์เท่ากับ 1,000 แปลว่า นำข้อมูลเข้าสู่หน่วยคำจำครั้งละ 1,000 แถวและแบบจำลองมีการเรียนรู้ข้อมูลครั้งละ 1,000 ข้อมูล

ทั้งนี้ในการทำงานครบ 1 แบตช์ยังนิยมให้มีการคำนวณข้อผิดพลาดของแบบจำลอง (Loss Value) และปรับพารามิเตอร์ (Parameter) ของแบบจำลองให้มีความถูกต้องมากยิ่งขึ้น ซึ่งความถี่ในการปรับค่าพารามิเตอร์จะส่งผลต่อความถูกต้องของแบบจำลอง ดังนั้นแล้วจึงต้องมีการกำหนดค่าความถี่ในการปรับค่าพารามิเตอร์ให้เหมาะสม

2.4 Epoch

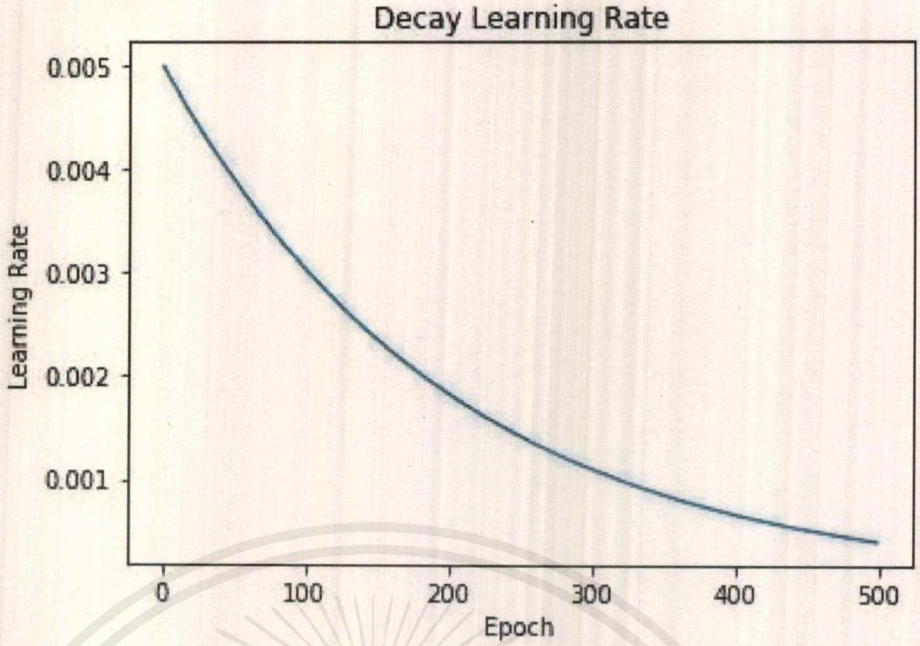
คือ 1 รอบในการเรียนรู้ของแบบจำลองกับข้อมูลทั้งหมด เช่น จำนวน Epoch เท่ากับ 5 คือ กำหนดให้แบบจำลองมีการเรียนรู้กับข้อมูลทั้งหมดจำนวน 5 รอบ

2.5 อัตราการเรียนรู้ (Learning Rate)

คือ ค่าที่บ่งบอกถึงความรวดเร็วในการปรับตัวตามข้อมูลการสอนของแบบจำลอง หากมีอัตราการเรียนรู้น้อยเกินไปจะทำให้แบบจำลองเข้าใกล้คำตอบที่ถูกต้องช้า แต่หากมีอัตราการเรียนรู้ที่มากเกินไปจะทำให้แบบจำลองจำเพาะเจาะจงกับข้อมูลฝึกสอนล่าสุดมากจนเกินไป ดังนั้นแล้วต้องมีการกำหนดค่าอัตราการเรียนรู้ให้มีความเหมาะสมกับแต่ละแบบจำลอง

2.6 การลดอัตราการเรียนรู้ (Decay Learning Rate)

จากปัญหาในการปรับค่าความเหมาะสมของอัตราการเรียน การลดอัตราการเรียนรู้จะสามารถลดปัญหาการกำหนดค่าอัตราการเรียนรู้ที่เหมาะสมได้ โดยนิยมกำหนดให้มีการลดอัตราการเรียนรู้คือ เมื่อเริ่มต้นจะกำหนดให้อัตราการเรียนรู้มีค่ามากเพื่อให้แบบจำลองมีการเรียนรู้ที่รวดเร็ว จากนั้นทำการปรับลดค่าอัตราการเรียนรู้ลงเรื่อย ๆ เพื่อให้แบบจำลองมีการเรียนรู้ที่ช้าลงเพื่อเข้าใกล้คำตอบที่ถูกต้องมากที่สุด



รูปที่ 2.2 แสดงการลดลงของอัตราการเรียนรู้

2.7 One-Hot Encoding

คือชุดของบิท (1 หรือ 0) ที่นำเสนอแทนข้อมูลต่าง ๆ โดยมีค่า 1 เพียง 1 ตำแหน่งและตำแหน่งอื่น ๆมีค่าเท่ากับ 0 ตัวอย่างเช่น “ฉัน|รัก|เมือง|ไทย” สามารถแปลงค่าเป็น One-Hot ของแต่ละคำได้ดังนี้

ฉัน	=	[1, 0, 0, 0]
รัก	=	[0, 1, 0, 0]
เมือง	=	[0, 0, 1, 0]
ไทย	=	[0, 0, 0, 1]

2.8 ฟังก์ชันก์ความสูญเสีย (Loss Function)

ฟังก์ชันก์ความสูญเสีย คือฟังก์ชันที่ใช้ในวัดความแตกต่างระหว่างค่าจริงกับค่าที่แบบจำลองทำนาย

2.9 Categorical Cross-Entropy [3]

คือฟังก์ชันที่ใช้ค่าครอสเอนโทรปี (Cross-Entropy) ในการวัดความสูญเสียจากการทำนายของแบบจำลอง

เอนโทรปี คือการวัดความไม่แน่นอนในการทำนายผลลัพธ์ของแบบจำลอง

$$H(p) = -\sum_{i=0}^{N} p_i \log p_i \tag{2.1}$$

เมื่อ p คือความน่าจะเป็นที่จะเกิดคลาสต่าง ๆ จากทั้งหมด N คลาส

H คือค่า Entropy ของ p

Cross-Entropy คือการวัดค่าความต่างของการแจกแจงความน่าจะเป็นในการทำนายผลลัพธ์ของแบบจำลอง

$$H(p, q) = -\sum_{i=0}^{N} p_i \log q_i \tag{2.2}$$

เมื่อ p คือความน่าจะเป็นของผลลัพธ์ในแต่ละคลาส (Class) จากคลาสทั้งหมด Nคลาส

q คือผลลัพธ์ในการทำนายคลาส (Class) ซึ่งเป็นรูปแบบของ One-Hot Encoding

H คือค่า Cross-Entropy ของ p และ q

2.10 Optimizer

Optimizer คืออัลกอริทึมสำหรับปรับค่าถ่วงน้ำหนัก (Weight) เพื่อให้แบบจำลองการเรียนรู้ด้วยเครื่องมีค่าสูญเสีย (Loss) ความถูกต้องน้อยที่สุด

2.11 ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [4]

คือตัวชี้วัดสำหรับประเมินผลการสรุปข้อความอัตโนมัติ (Automatic Summarization) และการแปลภาษาด้วยเครื่อง (Machine Translation) โดยการอ้างอิงจากการสรุปหรือการแปลที่มนุษย์สร้างขึ้น โดยมีสมการดังนี้

$$Recall = \frac{\text{จำนวนคำที่คาบเกี่ยวกัน (Overlap)}}{\text{จำนวนคำในประโยคสรุปข่าวที่ถูกต้อง}} \quad (2.3)$$

ค่า Recall คือ การวัดความสามารถในการดึงข้อมูลที่เกี่ยวข้องออกมา หรือสร้างข้อความผลลัพธ์ที่ถูกต้อง

$$Precision = \frac{\text{จำนวนคำที่คาบเกี่ยวกัน (Overlap)}}{\text{จำนวนคำในประโยคสรุปข่าวที่แบบจำลองสร้างขึ้น}} \quad (2.4)$$

ค่า Precision คือการวัดความสามารถในการขจัดข้อมูลที่ไม่เกี่ยวข้องออกไป หรือไม่สร้างข้อความผลลัพธ์ที่ไม่ถูกต้อง

2.11.1 ROUGE-1

คือการวัดความถูกต้องของการสรุปข้อความหรือการแปลภาษาโดยวัดจากความคาบเกี่ยวกัน (Overlap) ของคำด้วยความยาว 1 คำ ตัวอย่างเช่น

ประโยคสรุปข่าวที่แบบจำลองสร้างขึ้น: the cat was found under the bed

ประโยคสรุปข่าวที่ถูกต้อง: the cat was under the bed

มีค่า Recall เท่ากับ

$$Recall = \frac{6}{6} = 1.0$$

มีค่า Precision เท่ากับ

$$Precision = \frac{6}{7} = 0.86$$

2.11.2 ROUGE-2

คือการวัดความถูกต้องของการสรุปข้อความหรือการแปลภาษาโดยวัดจากความคาบเกี่ยวกัน (Overlap) ของคำด้วยความยาวจำนวน 2 คำ จากประโยคสรุปข่าวที่แบบจำลองสร้างขึ้นและประโยคสรุปข่าวที่ถูกต้อง จะได้ลำกับคู่ดังนี้

the	cat
cat	was
was	found
found	under
under	the
the	bed

the	cat
cat	was
was	under
under	the
the	bed

มีค่า Recall เท่ากับ

$$ROUGE2_{Recall} = \frac{4}{5} = 0.8$$

มีค่า Precision เท่ากับ

$$ROUGE2_{Precision} = \frac{4}{6} = 0.67$$

2.11.3 ROUGE-L

คือการวัดความถูกต้องของการสรุปข้อความหรือการแปลภาษา ซึ่งมีความคล้ายคลึงกับ ROUGE-1 และ ROUGE-2 ดังที่ได้กล่าวมาในข้างต้น แต่สำหรับ ROUGE-L จะใช้สมการ LCS (Longest Common Subsequence) ในการคำนวณ ซึ่งข้อดีของการใช้ LCS คือไม่ต้องใช้การจับคู่แบบต่อเนื่อง แต่เป็นการจับคู่เรียงลำดับที่สะท้อนถึงการเรียงลำดับคำในระดับประโยค ซึ่งมีสมการดังนี้

$$Recall = \frac{LCS(\text{ประโยค}_{\text{แบบจำลองสร้างขึ้น}}, \text{ประโยค}_{\text{เป้าหมาย}})}{\text{จำนวนคำในประโยคสรุปข่าวที่ถูกต้อง}} \quad (2.5)$$

$$Precision = \frac{LCS(\text{ประโยค}_{\text{แบบจำลองสร้างขึ้น}}, \text{ประโยค}_{\text{เป้าหมาย}})}{\text{จำนวนคำในประโยคสรุปข่าวที่แบบจำลองสร้างขึ้น}} \quad (2.6)$$

2.12 F-measure

F-measure (F_1 Score หรือ F Score) คือค่าเฉลี่ยประสิทธิโดยรวมของแบบจำลองการเรียนรู้ของเครื่อง

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.7)$$

2.13 Longest Common Subsequence Problem (LCS) [5]

เมื่อต้องการเปรียบเทียบชุดอักษร 2 ชุด ได้แก่ “AGCAT” และ “GAC” ด้วย LCS จะมีขั้นตอนดำเนินการ ดังนี้

ตารางที่ 2.3 เริ่มต้นสร้างตาราง LCS

	Ø	A	G	C	A	T
Ø	Ø	Ø	Ø	Ø	Ø	Ø
G	Ø					
A	Ø					
C	Ø					

1. เริ่มต้นนำคำที่ต้องการคำนวณมาเขียนเป็นตารางดังเช่นตารางที่ 2.3
2. เริ่มเปรียบเทียบอักษรในแถวที่ 2 คือ G คือ กับอักษรคอลัมน์ที่ 2 คือ A ว่าอักษรเหมือนกันหรือไม่ ซึ่งได้ผลลัพธ์คือ ไม่เหมือนกัน และนำ Ø พร้อมทั้งลูกศรชี้ขึ้นและลงเพิ่มในตารางดังรูปอักษร
3. จากนั้นดำเนินการต่อกับคอลัมน์ที่ 3 จะได้ผลลัพธ์คืออักษรเหมือนกัน ให้ทำการเพิ่มอักษร G และลูกศรชี้ขึ้นเอียงไปทางด้านซ้ายมือในตาราง
4. จากนั้นดำเนินการต่อกับคอลัมน์ที่ 4 จะได้ผลลัพธ์คืออักษรไม่เหมือนกัน ให้ทำการเพิ่มชุดอักษรตัวก่อนหน้าที่เหมือนกัน คือ G และลูกศรเข้าหาชุดอักษรที่เหมือนกันก่อนหน้าคือชี้ไปทางซ้ายเนื่องจากชุดอักษรที่เหมือนกันกันก่อนหน้าอยู่ทางด้านซ้าย
5. ดำเนินการเช่นไปเรื่อย ๆ จนจบทุกตัวอักษร จะได้ผลลัพธ์ดังนี้ตารางที่ 2.4

ตารางที่ 2.4 การคำนวณ LCS กับข้อมูลแถวที่ 2

	Ø	A	G	C	A	T
Ø	Ø	Ø	Ø	Ø	Ø	Ø
G	Ø	↑ ←Ø	↖(G)	←(G)	←(G)	←(G)
A	Ø					
C	Ø					

6. ดำเนินการกับแถวที่ 3 คืออักษร A กับคอลัมน์ที่ 2 คือ A จะได้ผลลัพธ์คืออักษรเหมือนกัน ให้ทำการเพิ่มอักษร A และลูกศรชี้ขึ้นเอียงไปทางด้านซ้ายมือในตาราง
7. จากนั้นดำเนินการต่อกับคอลัมน์ที่ 3 จะได้ผลลัพธ์คืออักษรไม่เหมือนกัน ให้ทำการเพิ่มชุดอักษรตัวก่อนหน้าที่เหมือนกัน คือ A และ G ซึ่งจะลูกศรชี้เข้าหาชุดอักษรที่เหมือนกันก่อนหน้า คือทางด้านซ้ายเพื่อชี้เข้าหา A และทางด้านบน เพื่อชี้เข้าทาง G และสำหรับการดำเนินการกับคอลัมน์ที่ 4 จะได้ผลลัพธ์เช่นกับกับการดำเนินการกับคอลัมน์ที่ 3
8. ในการดำเนินการกับคอลัมน์ที่ 5 จะได้ผลลัพธ์คือมีอักษรเหมือนกัน ให้ทำการเพิ่มอักษร GA และลูกศรชี้ขึ้นเอียงไปทางด้านซ้ายมือในตาราง เนื่องมาจากก่อนหน้านี้มีรูปแบบอักษรเหมือนกันคือ G และในการดำเนินการนี้มีอักษรที่เหมือนกันคือ A ดังนั้น ชุดอักษรที่ต้องเพิ่มในตารางคือ GA
9. ดำเนินการเช่นนี้ไปเรื่อย ๆ จนจบทุกตัวอักษร จะได้ผลลัพธ์ดังนี้ตารางที่ 2.5

ตารางที่ 2.5 การคำนวณ LCS กับข้อมูลแถวที่ 3

	Ø	A	G	C	A	T
Ø	Ø	Ø	Ø	Ø	Ø	Ø
G	Ø	↑←Ø	↖(G)	←(G)	←(G)	←(G)
A	Ø	↖(A)	↑←(A) & (G)	↑←(A) & (G)	↖(GA)	←(GA)
C	Ø					

10. เมื่อดำเนินการครบทุกตัวอักษรแล้ว จะได้ผลลัพธ์ดังนี้

ตารางที่ 2.6 การคำนวณ LCS กับข้อมูลแถวที่ 4

	Ø	A	G	C	A	T
Ø	Ø	Ø	Ø	Ø	Ø	Ø
G	Ø	↑←Ø	↖(G)	←(G)	←(G)	←(G)
A	Ø	↖(A)	↑←(A) & (G)	↑←(A) & (G)	↖(GA)	←(GA)
C	Ø	↑(A)	↑←(A) & (G)	↖(AC) & (GC)	↑←(AC) & (GC) & (GA)	↑←(AC) & (GC) & (GA)

11. จากนั้นทำการนับจำนวนความยาวสูงสุดของชุดอักษรในแตกละช่องตาราง จะได้ผลลัพธ์ดังเช่นในตาราง 2.7

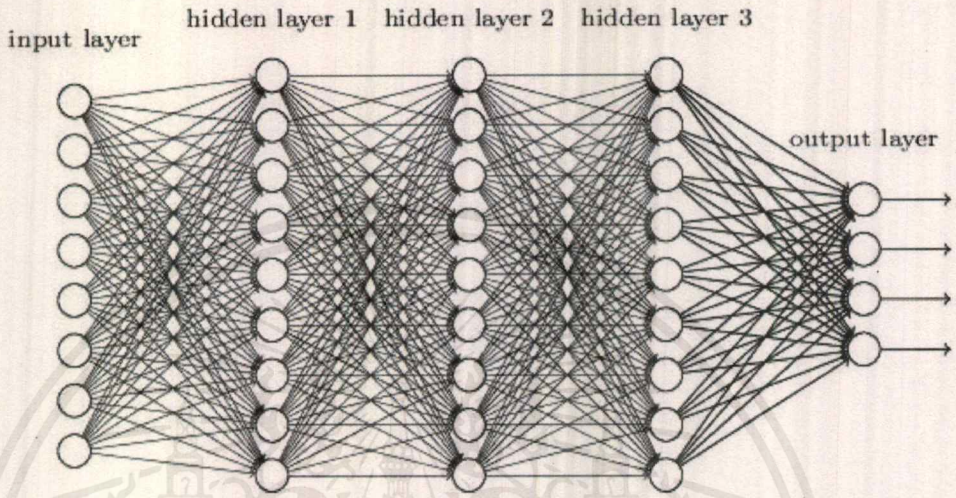
ตารางที่ 2.7 จำนวนความยาวสูงสุดของชุดอักษรในแตกละช่องตาราง

	Ø	A	G	C	A	T
Ø	0	0	0	0	0	0
G	0	↑←0	↖1	←1	←1	←1
A	0	↖1	↑←1	↑←1	↖2	←2
C	0	↑1	↑←1	↖2	↑←2	↑←2

จากตารางจะได้ผลลัพธ์การดำนำเนินการของฟังก์ชัน LCS เท่ากับ 2 (ค่าที่ด้านล่างมุมขวาของตาราง) ซึ่งมีความหมายว่า มีชุดอักษรที่เหมือนกันยาวที่สุดเท่ากับ 2 ตัวอักษร ได้แก่ AC, GC และ GA

2.14 การเรียนรู้เชิงลึก (Deep Learning) [6]

การเรียนรู้เชิงลึกเป็นสาขาของการเรียนรู้ของเครื่อง หลักสำคัญของการเรียนรู้เชิงลึก คือ อัลกอริทึมที่พยายามสร้างแบบจำลองเพื่อแทนรูปแบบและความสัมพันธ์ของข้อมูลที่ซับซ้อน โดยการสร้างสถาปัตยกรรมข้อมูลและประกอบไปด้วยโครงสร้างย่อย ๆหลายอันหรือหลายชั้นเชื่อมต่อกัน



รูปที่ 2.3 โครงข่ายประสาทเทียมแบบลึก

(ที่มา: https://www.kdnuggets.com/2017/05/deep-learning-big-deal.html)

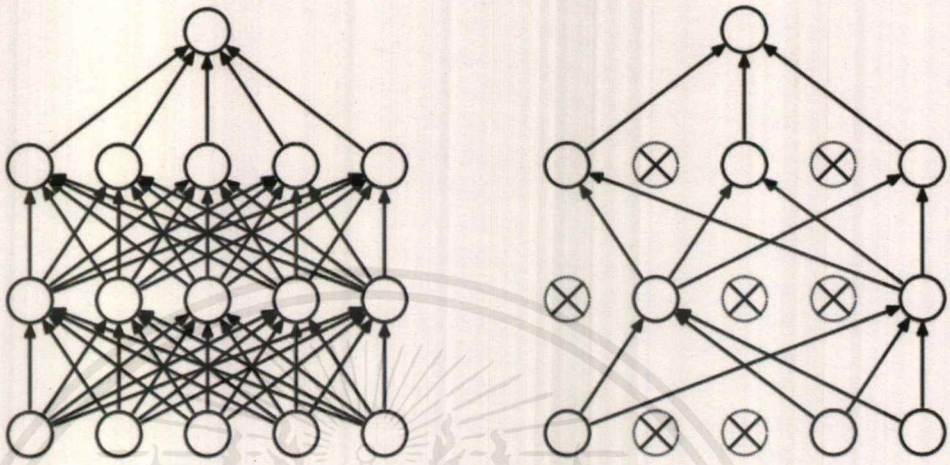
จากรูป คือ โครงข่ายประสาทเทียมแบบลึก (Deep Artificial Neuron Networks) ประกอบไปด้วยชั้นอินพุต, ชั้นเอาต์พุตและชั้นซ่อนจำนวน 2 ชั้น ซึ่งเป็นหนึ่งในสถาปัตยกรรมของการเรียนรู้เชิงลึก (Deep Learning)

2.15 การเข้ากันมากเกินไป (Overfitting)

Overfitting คือ แบบจำลองสามารถจดจำข้อมูลชุดฝึกสอนได้เป็นอย่างดี แต่เมื่อนำแบบจำลองไปทดสอบกับข้อมูลชุดทดสอบแล้วจะได้ผลลัพธ์ที่มีประสิทธิภาพต่ำ เนื่องจากแบบจำลองสามารถเข้ากัน (Fitting) กับข้อมูลชุดฝึกสอนได้เป็นอย่างดีแต่ไม่เข้ากันหรือไม่สามารถคาดการณ์ (Prediction) ผลลัพธ์ที่ถูกต้องของข้อมูลชุดทดสอบได้

2.16 Dropout [7]

คือการปิดบังนิวรอนของโครงข่ายประสาทเทียมในบางส่วน เพื่อเพิ่มความยากในการเรียนรู้ของแบบจำลอง ซึ่งวิธีการ Dropout นี้ใช้เพื่อลดปัญหา Overfitting ดังที่ได้กล่าวมาในข้างต้น



รูปที่ 2.4 แสดงการ Dropout ของแบบจำลองโครงข่ายประสาทเทียม
(ที่มา: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf)

จากรูปที่ 2.4 ทางด้านซ้ายคือแบบจำลองโครงข่ายประสาทเทียมโดยปกติที่ไม่มีการ Dropout และรูปทางด้านขวาคือ แบบจำลองโครงข่ายประสาทเทียมที่มีการ Dropout โดยจำนวนของการปิดบังนิวรอนขึ้นอยู่กับการกำหนดค่าเปอร์เซ็นต์การ Dropout ของผู้พัฒนา ซึ่งจะส่งผลต่อประสิทธิภาพการทำงานของแบบจำลองการเรียนรู้ของเครื่อง

2.17 การเรียนแบบวนซ้ำ (Recurrent Neural Network: RNN) [8]

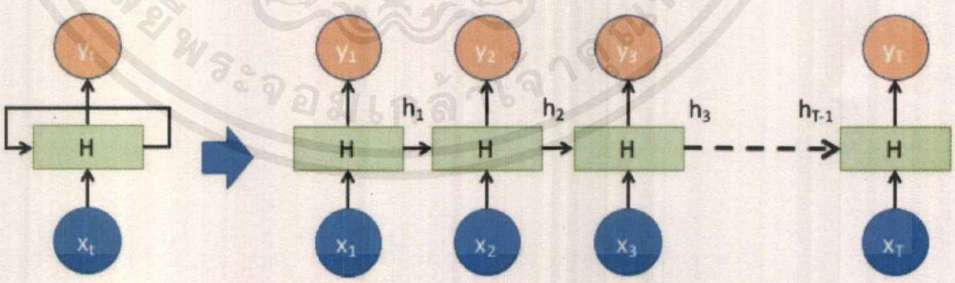
สถาปัตยกรรมของ RNN ถูกออกแบบมาเพื่อใช้งานกับข้อมูลที่มีลักษณะเป็นลำดับ (Sequential Data) เช่น ข้อความ (Sequence of Words) เมื่อพิจารณาพฤติการณ์อ่านหนังสือของคนเราจะได้ว่า เราจะอ่านทีละคำจากซ้ายไปขวา (สำหรับภาษาไทยหรือภาษาอังกฤษ) และการที่เราสามารถเข้าใจได้ว่าประโยคเหล่านั้นมีความเกี่ยวข้องกับเรื่องใด เนื่องจากเราได้นำเอาเรื่องราวจากสิ่งที่ได้อ่านผ่านไปแล้ว (สถานะซ่อนหรือสถานะก่อนหน้า) มาผสมกับคำที่เพิ่งอ่านไป (ข้อมูลอินพุตหรือคำที่เรากำลังอ่าน ณ เวลานั้น) ทำให้สามารถเข้าใจความหมายในส่วนตรงที่กำลังอ่านได้ ซึ่งในสถาปัตยกรรม RNN จะใช้หลักการเดียวกันกับการอ่านหนังสือนี้ คือการปรับรูปแบบของโครงข่ายประสาทเทียมเดิมเพื่อให้สามารถนำเอาสถานะ (State) หรือความรู้ก่อนหน้ามารวมกับข้อมูลอินพุตตัวใหม่ที่เพื่อทำความเข้าใจข้อมูลใหม่ไปเรื่อย ๆ

ดังนั้นสถาปัตยกรรม RNN มี 2 ส่วนที่สำคัญคือ

1. สถานะซ่อน (Hidden State) ก่อนหน้า
2. ข้อมูลอินพุต ณ เวลานั้น

สำหรับการอธิบาถึงการทำงานต่อไป กำหนดให้มีตัวแปรดังนี้

- ข้อมูลอินพุต คือ $x_1, x_2, x_3, ..., x_t$
- สถานะซ่อนที่เวลา t จะใช้ตัวแปรว่า x_h
- ถ้าข้อมูลอินพุตคือประโยคว่า “ฉันกินข้าว” จะได้ว่า x_1 = “ฉัน”, x_2 = “กิน” และ x_3 = “ข้าว”



รูปที่ 2.5 สถาปัตยกรรม RNN

(ที่มา: https://medium.com/@sinart.t/long-short-term-memory-lstm-e6cb23b494c6)

โดยที่

H = Hidden Layer

y_t = เอาต์พุต (Output) จาก RNN ที่เวลา t

x_t = ข้อมูลอินพุต (Input Data) ที่เวลา t

h_t = สถานะซ่อน (Hidden State) ที่เวลา t

จากรูป 2.5 ทางซ้ายแสดงให้เห็นว่ามีการวนซ้ำ (Loop) กลับเข้ามาที่ชั้นซ่อน (Hidden Layer) ของโครงข่ายประสาทเทียมดังที่ได้กล่าวไปในข้างต้นแล้วว่าสิ่งสำคัญสิ่งขอสถาปัตยกรรมง RNN คือสถานะก่อนหน้าข้อมูลอินพุต ณ เวลานั้น ซึ่งหลักสำคัญของการวนซ้ำนี้เพื่อนำสถานะซ่อนก่อนหน้ากลับเข้ามาใช้นั่นเอง หรืออาจมองได้ว่า สถาปัตยกรรม RNN คือโครงข่ายผระสารทเทียมที่มีหน่วยความจำ (Memory) เพิ่มขึ้นมาเพื่อเก็บค่าสถานะซ่อนที่คำนวณเอาไว้ก่อนหน้า)

จากรูป 2.5 ทางขวาเป็นรูปที่คลี่รูปทางซ้ายออกเพื่อแสดงการทำงานทีละขั้นตอนของสถาปัตยกรรม RNN

$$h_t = f_h(U_h h_{t-1} + W_h x_t + b_h) \tag{2.8}$$

$$y_t = f_y(W_y h_y + b_y) \tag{2.9}$$

โดยที่

f_h คือฟังก์ชันก์กระตุ้น (Activation Function) ของชั้นซ่อน เช่น Tanh function [9], ReLU Function [10], Sigmoid Function [11]

f_y คือฟังก์ชันก์กระตุ้นของชั้นเอาต์พุต เช่น Softmax Function [12]

W_h คือเมทริกซ์ค่าถ่วงน้ำหนัก (Weight Matrix) ของชั้นซ่อน

U_h คือเมทริกซ์ที่เชื่อมระหว่างสถานะซ่อน (Hidden-State-to-Hidden-State Matrix หรือ Transition Matrix)

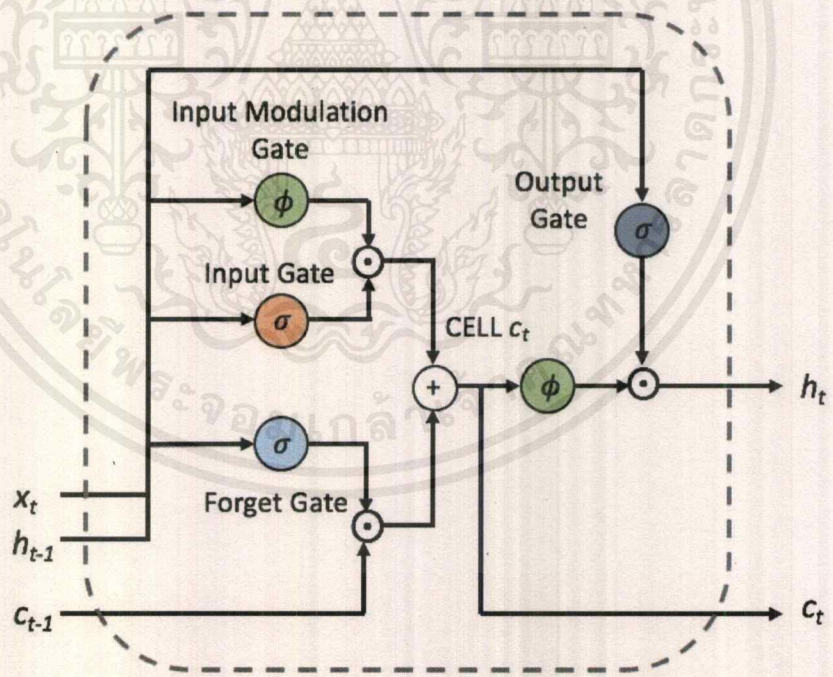
จากสมการจะเห็นว่าการคำนวณสถานะซ่อนที่เวลา t ได้นั้น (h_t) จำเป็นต้องใช้ 2 ตัวแปรสำคัญ คือสถานะซ่อนก่อนหน้าก่อนหน้า (h_{t-1}) และข้อมูลอินพุต ณ เวลานั้น (x_t)

2.17.1 ปัญหาของ RNN

ในการปรุงปรุงค่าถ่วงน้ำหนักของโครงข่ายประสาทเทียมจะอาศัยอัลกอริทึม Backpropagation [13] สำหรับสถาปัตยกรรม RNN จะมีการคำนวณที่ความซับซ้อนมาก เนื่องจากเอาต์พุต y_t ไม่ได้ถูกคำนวณจากช่วงเวลา t=t เพียงอย่างเดียว แต่ถูกคำนวณต่อเนื่องจาก t=t-1, t=t-2, t=t-3, ..., t=1 ซึ่งจะทำให้เกิดปัญหาการสูญหายของค่าเกรเดียน (Vanishing Gradient Problem) [14] และหากมีความยาวของข้อมูลมากจะทำให้มีโอกาสเกิดปัญหาการสูญหายของค่าเกรเดียนมาก โดยจะส่งผลให้แบบจำลองไม่เกิดการเรียนรู้ เนื่องจากไม่มีปรับปรุงค่าถ่วงน้ำหนักตามที่ได้กล่าวมาในข้างต้น หรือกล่าวสรุปได้ว่าสถาปัตยกรรม RNN เกิดปัญหากับข้อมูลที่มีขนาดของลำดับ (Sequential) ยาวเกินไปนั่นเอง

2.18 Long Short-Term Memory (LSTM)

จากปัญหาของสถาปัตยกรรม RNN ที่ได้กล่าวไปข้างต้น มีการเสนอการใช้สถาปัตยกรรม LSTM โดย Sepp Hochreiter และ Juergen Schmidhuber



รูปที่ 2.6 สถาปัตยกรรม LSTM

(แหล่งที่มา: https://medium.com/@sinart.t/long-short-term-memory-lstm-e6cb23b494c6)

จากที่ได้กล่าวไปในข้างต้นแล้วว่า RNN เป็นเหมือนโครงข่ายประสาทเทียมที่มีหน่วยความจำเพื่อบันทึกสถานะซ่อนก่อนหน้า และสำหรับ LSTM จะหน่วยความจำอยู่ภายในเช่นกัน แต่หลักสำคัญของสถาปัตยกรรม LSTM คือมีการทำงานที่บ่งบงได้ว่าเมื่อใดควรจะจดจำข้อมูลในหน่วยความจำ, ลืม (ลบ) ข้อมูลในหน่วยความจำหรืออนุญาตให้แบบจำลองรับข้อมูลอินพุตได้

2.18.1 การทำงานของ LSTM มีตัวแปรที่สำคัญดังนี้

1. เซลล์สถานะ (Cell State) คือตัวเก็บสถานะของเซลล์หน่วยความจำ (Memory Cell) ใน LSTM
2. Gate คือตัวที่ควบคุมการไหลของข้อมูล ซึ่งเปรียบเมือนประตูที่คอยควบคุมว่าเมื่อไหร่ควรเปิดให้ข้อมูลไหลเข้า ไหลออก หรือไหลหายไปเลย (Forget)

ส่วนการทำงานต่าง ๆ ของ LSTM สามารถแบ่งได้ดังนี้

2.18.1.1 การลืม (Forget)

การลืม (Forget) คือการลบเซลล์สาถานะเดิมออกไปเพื่อเตรียมพื้นที่รับข้อมูลใหม่ ซึ่งส่วนการทำงานนี้เป็นหน้าที่ตัดสินใจว่าจะลบหรือไม่ เป็นหน้าที่ของ Forget Gate

การสร้าง Forget Gate จะพิจารณาจากข้อมูลอินพุตประกอบกับสถานะซ่อนก่อนหน้าประกอบการตัดสินใจ โดยจะใช้ Sigmoid Function ในการตัดสินดังสมการ

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \qquad (2.10)$$

2.18.1.2 การเขียน (Write)

เมื่อมีข้อมูลอินพุตใหม่เข้ามา จะมีการพิจารณาอยู่ 2 ส่วนการทำงาน คือ 1. จะมีการปรับปรุงค่าเซลล์สถานะด้วย ข้อมูลอินพุตใหม่หรือไม่ 2. หากมีการปรับปรุงค่า จะปรับปรุงด้วยค่าใดจึงจะเหมาะสม

พิจารณาการปรับปรุงค่าเซลล์สถานะ โดยมีการควบคุมส่วนงานนี้โดย Input Gate ซึ่งใช้ Sigmoid Function และใช้ข้อมูลอินพุตประกอบกับสถานะซ่อนก่อนหน้าประกอบการตัดสินใจว่าจะอนุญาตให้ปรับปรุงค่าหรือไม่

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \qquad (2.11)$$

ค่าที่เหมาะสมเมื่อมีการปรับปรุงค่าเซลล์สถานะ ซึ่งจะ Input Modulation gate เป็นตัวจัดการ โดยสมการก็จะคล้ายคลึงกับ Input Gate แต่ว่าจะใช้เป็น Tanh Function ในการคำนวณ ซึ่งค่าที่ได้จากการคำนวณเปรียบเสมือนว่าเป็น Cell State Candidate

$$g_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \qquad (2.12)$$

2.18.1.3 การปรับปรุงเซลล์สถานะ (Cell state)

จากการคำนวณของ Forget Gate, Input Gate และ Input Modulation Gate แล้ว เมื่อนำไปปรับปรุงเซลล์สถานะได้ตามสมการนี้

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \qquad (2.13)$$

จากสมการทางด้านฝั่งขวาในส่วนแรกของสมการจะเห็นได้ว่า ถ้า Forget Gate จะกำหนดให้ลบเซลล์สถานะเดิมทิ้ง เมื่อ f_t มีค่าเป็น 0 และมีการเก็บเซลล์สถานะ c_{t-1} ไว้ประกอบการพิจารณาการปรับปรุง เมื่อ f_t มีค่าเป็น 1

ในส่วนหลังของสมการ จะเป็นส่วนการทำงานปรับปรุงเซลล์สถานะจากข้อมูลใหม่ ซึ่งจะพิจารณาจากค่าที่ได้จาก Input Modulation Gate (g_t) ร่วมกับค่าผลลัพธ์จาก Input Gate (i_t)ว่ามีความเหมาะสมหรือไม่ หาก i_t มีค่าเป็น 1 จะใช้ค่า g_t ในการปรับปรุงเซลล์สถานะ แต่หาก i_t มีค่าเป็น 0 จะไม่ใช้ค่า g_t ที่มิติ (Dimension) นั้นในการปรับปรุงเซลล์สถานะ

2.18.1.4 การอ่าน (Read)

สำหรับผลลัพธ์ ณ เวลานั้น ๆของสถาปัตยกรรม LSTM จะมี 2 ส่วนคือ y_t และ h_t ในส่วนของ h_t จะมีการส่งต้องค่าไปยังเซลล์สถานะถัดไป ซึ่งจะมีการพิจารณาจะมีการส่งค่า h_t ไปยังเซลล์สถานะ h_{t+1} หรือไม่ หรือกล่าวว่า ยอมเซลล์สถานะที่ t=t+1 อ่านค่าสถานะซ่อนหรือไม่ โดยในส่วนการทำงานนี้ Output Gate ในการตัดสินใจ ซึ่งจะยังคงใช้สมการเดิมกับเหมือนกับการคำนวณ Forget Gate และ Input Gate ดังนี้

$$o_t = \sigma(W_{x^o} x_t + W_{h^o} h_{t-1} + b_o) \quad (2.14)$$

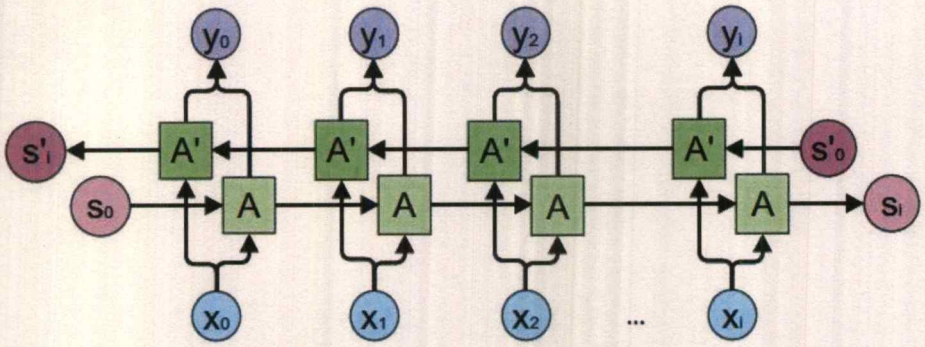
สมการที่ใช้ในการพิจารณาการส่งออกค่า h_t ไปยังเซลล์สถานะถัดไป คือ

$$h_t = o_t \odot \tanh(c_t) \quad (2.15)$$

จากสมการจะเห็นได้ว่าหาก Output Gate มีผลลัพธ์ค่า o_t เป็น 0 จะทำให้ได้ค่าผลลัพธ์ของ h_t มีค่าเป็น 0 ซึ่งมีความหมายว่าจะไม่ส่งค่าใด ๆ ออกไปไปเซลล์สถานะถัดไป และในขณะเดียวกันหาก o_t มีค่าเป็น 1 มีความหมายว่าจะคำนวณค่า h_t และส่งค่าผลลัพธ์ไปยังเซลล์สถานะถัดไป

2.19 Bi-directional LSTM [15]

Bi-directional LSTM หรือ LSTM แบบสองทิศทาง ถูกพัฒนาต่อยอดจากสถาปัตยกรรม LSTM ซึ่งจะมีการการทำงานเหมือนกับ LSTM ทุกประการ แต่เมื่อทำการจนถึงเซลล์สถานะสุดท้ายแล้วจะทำการคำนวณย้อนกลับไปยังข้อมูลอินพุตลำดับแรกดังรูป



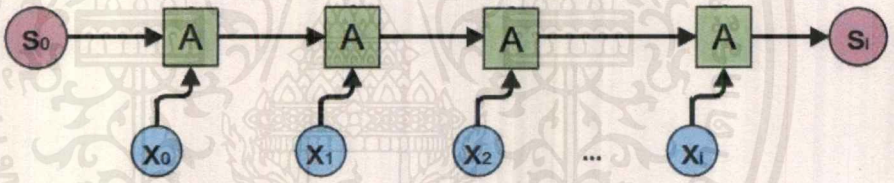
รูปที่ 2.7 สถาปัตยกรรม Bi-directional LSTM

(ที่มา: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66)

2.19.1 ขั้นตอนการทำงาน Bi-directional LSTM

สำหรับการขั้นตอนการดำเนินการของ Bi-directional LSTM มีดังนี้

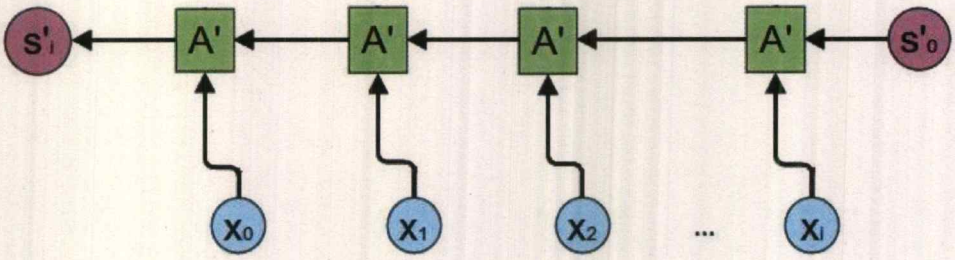
ขั้นตอนที่ 1: ดำเนินการเช่นเดียวกับสถาปัตยกรรม LSTM ทุกประการ



รูปที่ 2.8 Bi-directional LSTM ขั้นตอนไปข้างหน้า

(แก้ไขรูปจาก: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66)

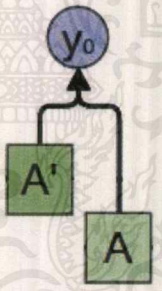
ขั้นตอนที่ 2: ดำเนินการเช่นเดียวกับสถาปัตยกรรม LSTM ทุกประการ แต่เริ่มต้นที่ข้อมูลอินพุตลำดับสุดท้ายไปยังข้อมูลอินพุตลำดับเริ่มต้น



รูปที่ 2.9 Bi-directional LSTM ขั้นตอนย้อนกลับ

(แก้ไขรูปจาก: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66)

ขั้นตอนที่ 3: นำค่าเวกเตอร์ของเซลล์สถานะที่มีข้อมูลลำดับเดียวกันมาต่อเรียงกัน (Concatenated) เช่น เวกเตอร์ A = [1, 2, 3] และเวกเตอร์ A' = [4, 5, 6] เมื่อดำเนินการ A concat A' จะได้ผลลัพธ์คือ [1, 2, 3, 4, 5, 6]

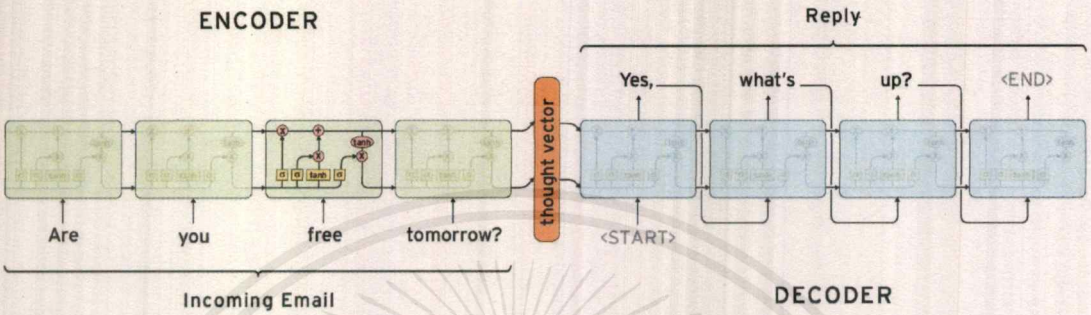


รูปที่ 2.10 การ Concatenate

(แก้ไขรูปจาก: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66)

2.20 แบบจำลอง Sequence-to-Sequence [16]

แบบจำลอง Sequence-to-Sequence เป็นการพัฒนาสถาปัตยกรรม LSTM เพื่อนำมาแก้ปัญหาที่มีข้อมูลอินพุตและเอาต์พุตในรูปแบบข้อมูลเป็นลำดับ (Sequential Data) เช่น การแปลภาษาด้วยเครื่อง, แชทบอท เป็นต้น



รูปที่ 2.11 แบบจำลอง Seq2Seq

(ที่มา: https://towardsdatascience.com/sequence-to-sequence-tutorial-4fde3ee798d8)

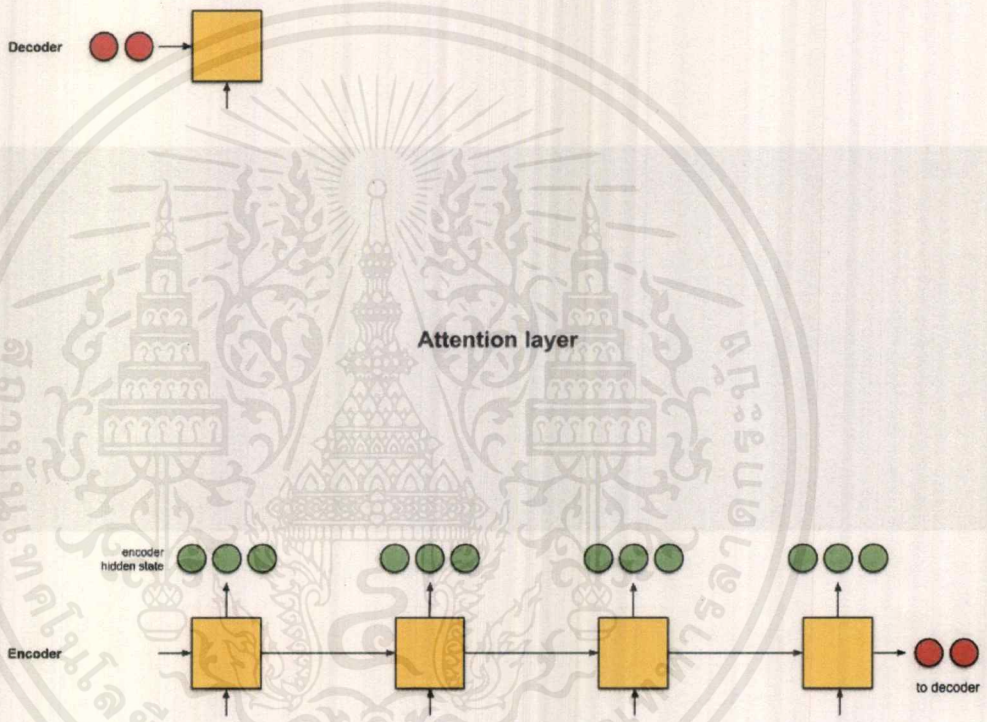
สำหรับการทำงานของแบบ Sequence-to-Sequence จะถูกแบ่งเป็น 2 ส่วนการทำงาน คือ Encoder และ Decoder ซึ่งมีรายละเอียดการทำงานดังนี้

1. Encoder จะทำงานร่วมกับข้อมูลอินพุต โดยมีการทำงานตามสถาปัตยกรรม LSTM ทุกประการ
2. Decoder จะทำงานร่วมกับข้อมูลเอาต์พุต จากรูป 2.11 จะเห็นได้ว่าในส่วนของเซลล์สถานะเริ่มต้นจะรับค่าเวกเตอร์มาจากส่วนการทำงาน Encoder และมีข้อมูลอินพุตเริ่มต้นคือคำว่า “<START>” จากนั้นจะดำเนินการทำนายผลลัพธ์และนำไปเป็นข้อมูลอินพุตในเซลล์ถัดไป โดยมีการดำเนินการต่าง ๆตามการทำงานของสถาปัตยกรรม LSTM ทุกประการ

2.21 Bahdanau Attention [17]

จากแบบจำลอง Sequence-to-Sequence ได้มีการพัฒนาเพื่อให้มีการทำงานที่ดีขึ้นและเหมาะสมในแต่ละปัญหา ซึ่งวิธีการที่ได้รับความนิยมคือทำการเพิ่ม Attention Layer เข้ามาในการประมวลผล อยู่ระหว่าง Encoder และ Decoder ดังรูป 2.12

Bahdanau Attention เป็น Attention Layer ที่ได้รับความนิยมอย่างกว้างขวางในการนำไปประยุกต์ใช้ในปัญหาต่าง ๆ เช่น การแปลภาษาด้วยเครื่อง, แชทบอท, การสรุปข้อความ เป็นต้น เนื่องจากมีความง่ายต่อการศึกษาและให้ผลลัพธ์ที่ดีในระดับหนึ่ง

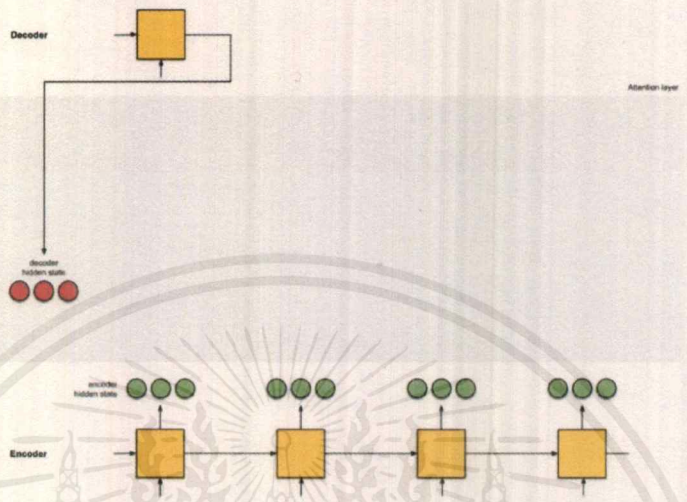


รูปที่ 2.12 Attention Layer

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

2.21.1 ขั้นตอนการทำงานของ Attention Layer

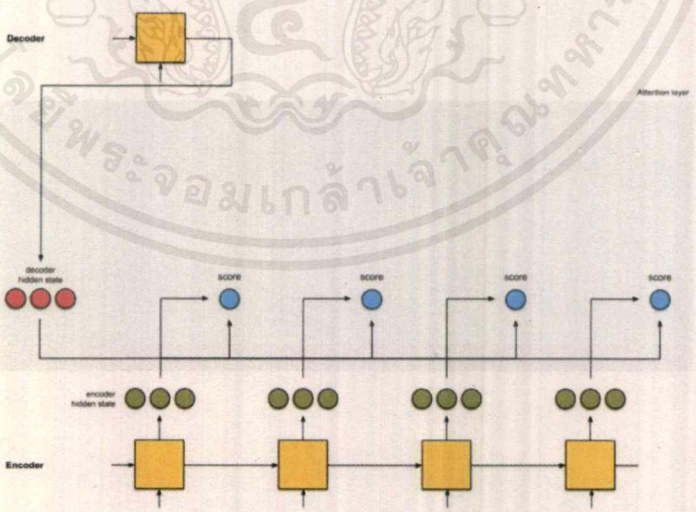
ขั้นตอนที่ 1: คำนวณค่าต่างๆจากเซลล์สถานะตามการทำการของสถาปัตยกรรม LSTM ทุกประการ



รูปที่ 2.13 Attention Layer การคำนวณขั้นตอนที่ 1

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

ขั้นตอนที่ 2: คำนวณคะแนนโดยใช้เซลล์สถานะของ Decode กับ เซลล์สถานะในแต่ละลำดับเวลา ดังรูป 2.14



รูปที่ 2.14 Attention Layer การคำนวณขั้นตอนที่ 2

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

โดยมีตัวอย่างการคำนวณดังนี้

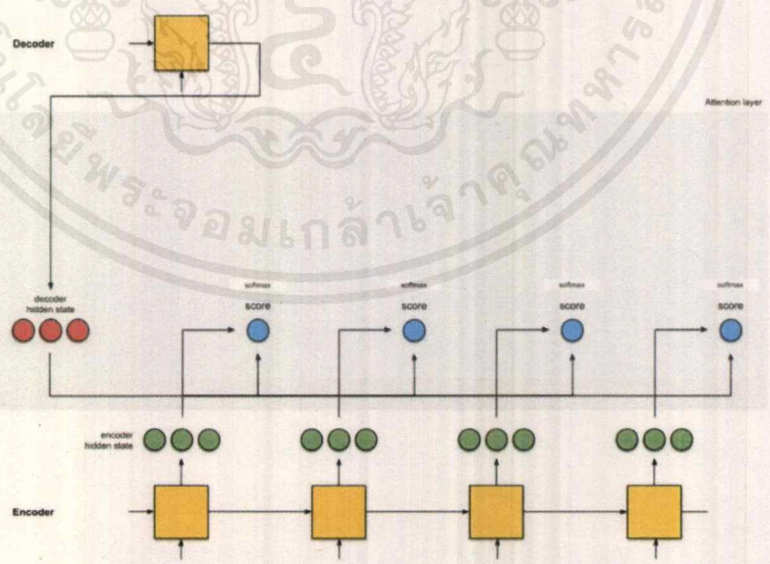
กำหนดให้ Decoder Hidden State = [10, 5, 10]

ดำเนินการคำนวณคะแนนของเซลล์สถานะในแต่ละลำดับด้วยการทำ Dot Product ระหว่าง Decoder Hidden State และ Encoder Hidden State จะได้ผลลัพธ์ดังนี้

ตารางที่ 2.8 แสดงผลลัพธ์การทำ Dot Product ระหว่าง Decoder Hidden State และ Encoder Hidden State

Encoder Hidden State	Score
[0, 1, 1]	15 = (10×0) + (5×1) + (10×1)
[5, 0, 1]	60 = (10×5) + (5×0) + (10×1)
[1, 1, 0]	15 = (10×1) + (5×1) + (10×0)
[0, 5, 1]	35 = (10×0) + (5×5) + (10×1)

ขั้นตอนที่ 3: นำผลลัพธ์จากการคำนวณในขั้นตอนที่ 2 ผ่านฟังก์ชัน Softmax โดยฟังก์ชันก์จะทำการเลือกเวกเตอร์ที่มีคะแนนสูงสุด คือ 60 ดังนั้นในการดำเนินการขั้นตอนนี้จะได้ผลลัพธ์ คือ เวกเตอร์ [5, 0, 1]



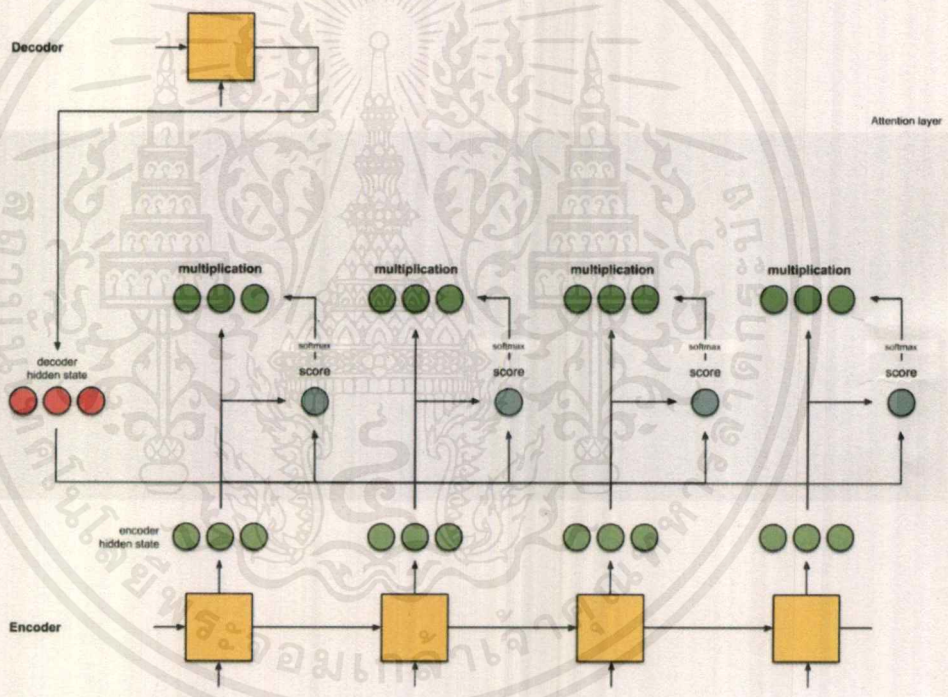
รูปที่ 2.15 Attention Layer การคำนวณขั้นตอนที่ 3

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

ตารางที่ 2.9 ผลลัพธ์จากการผ่านฟังก์ชัน Softmax

Encoder Hidden State	Score	Score'
[0, 1, 1]	15	0
[5, 0, 1]	60	1
[1, 1, 0]	15	0
[0, 5, 1]	35	0

ขั้นตอนที่ 4: คูณคะแนนจากขั้นตอนที่ 4 กับค่าเวกเตอร์เซลล์สถานะในแต่ละลำดับเวลา



รูปที่ 2.16 Attention Layer การคำนวณขั้นตอนที่ 4

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

ตารางที่ 2.10 ผลลัพธ์จากการคูณ Encoder Hidden State และ Score'

Encoder Hidden State	Score	Score'	Alignment
[0, 1, 1]	15	0	[0, 0, 0]
[5, 0, 1]	60	1	[5, 0, 1]
[1, 1, 0]	15	0	[0, 0, 0]
[0, 5, 1]	35	0	[0, 0, 0]

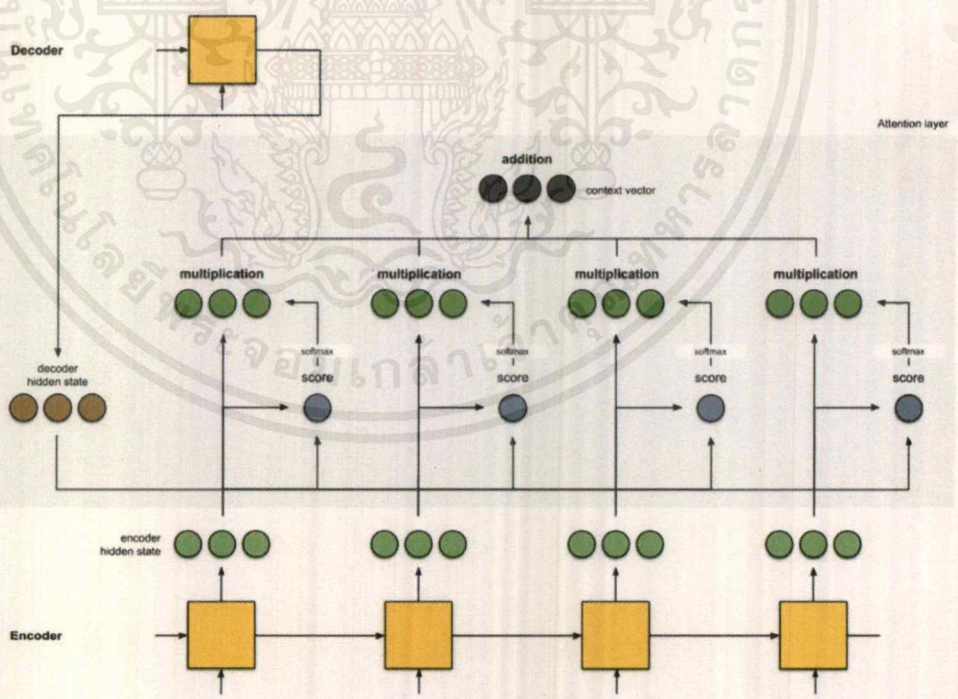
ขั้นตอนที่ 5: รวมผลลัพธ์เวกเตอร์ Alignment จากขั้นตอนที่สี่เข้าด้วยกัน โดยมีการรวมค่าตามตำแหน่งของเวกเตอร์ดังนี้

ตำแหน่งที่ 1: $0 + 5 + 0 + 0 = 5$

ตำแหน่งที่ 2: $0 + 0 + 0 + 0 = 0$

ตำแหน่งที่ 3: $0 + 1 + 0 + 0 = 1$

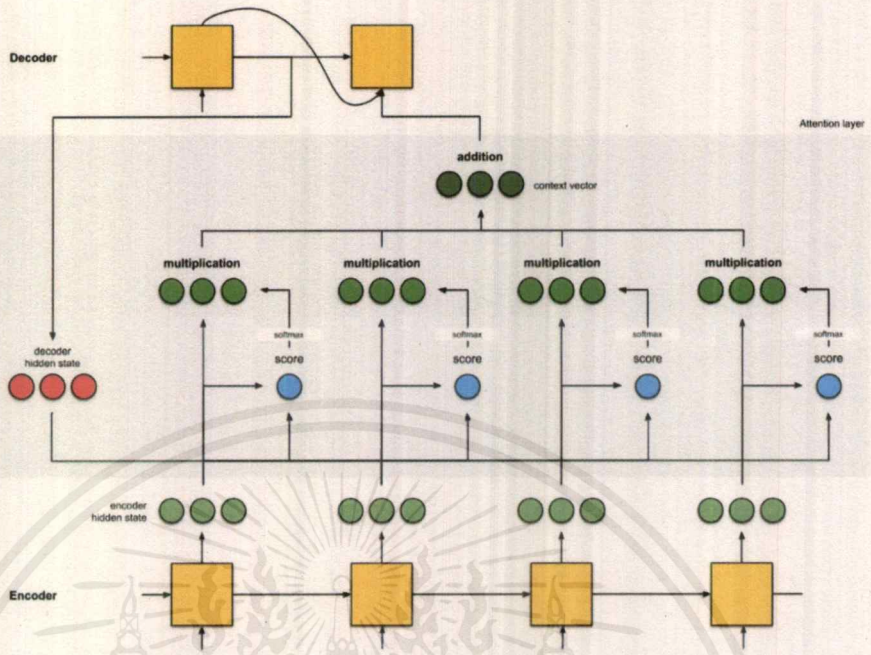
ดังนั้นในขั้นตอนนี้จะได้ผลลัพธ์คือ [5, 0, 1] ซึ่งเรียกว่า เวกเตอร์บริบท (Context Vector)



รูปที่ 2.17 Attention Layer การคำนวณขั้นตอนที่ 5

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

ขั้นตอนที่ 6: นำเวกเตอร์บริบทไปยังอินพุตในส่วนของ Decoder



รูปที่ 2.18 Attention Layer การคำนวณขั้นตอนที่ 6

(ที่มา: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

2.22 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.22.1 เอกสารที่เกี่ยวข้อง

2.22.1.1 Neural Abstractive Text Summarization with Sequence-to-Sequence Models [18]

รายงานฉบับนี้ได้รวบรวมแบบจำลองและเทคนิคต่าง ๆ ที่เกี่ยวข้องกับการสรุปข้อความประเภท Abstractive หรือที่เรียกว่า Neural Abstractive Text Summarizer (NATS) ตั้งแต่ปี พ.ศ. 2558 ถึงปี พ.ศ. 2561 ซึ่งแสดงให้เห็นถึงการให้ความสนใจจำนวนมากต่อการพัฒนาแบบจำลองการเรียนรู้ของเครื่องและเทคนิคต่าง ๆ เพื่อการสรุปข้อความให้มีความถูกต้องมากยิ่งขึ้น ทำให้ง่ายต่อการศึกษาและง่ายต่อการค้นหางานวิจัยที่เกี่ยวข้องกับปัญหาการสรุปข้อความ โดยมีการแสดงรายชื่อแบบจำลองการเรียนรู้ด้วยเครื่องหรือหรือสถาปัตยกรรมหรือเทคนิค ชื่อผู้วิจัย ซึ่งจัดเรียงลำดับตามช่วงเวลาในการพัฒนา

2.22.1.2 Abstractive Text Summarization [19]

เอกสารฉบับนี้กล่าวถึงที่มาและความสำคัญ ความรู้เบื่องต้น สถาปัตยกรรม และเทคนิคในการสรุปข้อความ โดยมีเนื้อหาโดยสรุปดังนี้

การสรุปข้อความถูกแบ่งออกเป็น 2 ประเภท คือ Extractive และ Abstractive

1. Extractive เปรียบเสมือนการอ่านหนังสือแล้วสกัดเนื้อหาสำคัญออกมาหรือคือการไฮไลท์ (Highlight)
2. Abstractive เปรียบเสมือนการอ่านและทำความเข้าใจเนื้อหาของข้อความและเขียนข้อความขึ้นมาใหม่ให้มีแต่เนื้อหาที่สำคัญ

เทคนิคที่มีความน่าสนใจและได้รับความนิยม มี 2 เทคนิค คือ 1.การนำแบบจำลอง Attention Mechanism มาร่วมประมวลผลกับแบบจำลอง Sequence-to-Sequence และ 2. การสรุปข้อความโดยการการเรียนรู้ด้วยเครื่องประเภทการเรียนรู้แบบเสริมแรง (Reinforcement Learning)

1. แบบจำลอง Attention Mechanism นิยมถูกใช้ในการแก้ปัญหาการแปลภาษาด้วยเครื่อง และมีการนำมาปรับใช้กับปัญหาอื่น ๆ ที่มีรูปแบบปัญหาที่ใกล้เคียง คือข้อมูลอินพุตและเอาต์พุตมีรูปแบบเป็นข้อมูลลำดับ ซึ่งในเอกสารฉบับนี้ได้กล่าวแนะนำแบบจำลอง Attention Mechanism ด้วย Bahdanau ซึ่งเป็นแบบจำลองที่สามารถศึกษาและทำความเข้าใจได้ง่ายในการเริ่มต้นเรียนรู้

2. การใช้การเรียนรู้แบบเสริมแรง (Reinforcement Learning) โดยกล่าวถึงการเรียนรู้แบบเสริมแรงโดยใช้ Policy Gradient-based โดยการประมวลผลในระดับคำ (Word Level) มีการคำนวณคะแนน (Reward) ด้วยตัวชี้วัด ROUGE สำหรับข้อสังเกตุของวิธีการนี้คือ เป้าหมายของแบบจำลองไม่ควรถูกจำกัดให้แสดงผลลัพธ์ที่อ้างอิงความถูกจากเพียงข้อความเป้าหมายเท่านั้น เนื่องจากในความเป็นจริงแล้ว ประโยคที่แตกต่างกันอาจจะมีความหมายที่สื่อถึงสิ่งเดียวกันได้ หรือกล่าวคือในการสื่อ

ความหมายหนึ่งๆ สามารถถูกเขียนได้ในหลายรูปแบบ ซึ่งจากการคำนวณคะแนนด้วยตัวชี้วัด ROUGE จะทำให้ได้ผลลัพธ์คะแนนที่น้อย ซึ่งมีความไม่สมเหตุสมผลนั่นเอง

นอกจากนี้ยังมีการกล่าวถึงตัวชี้วัดในการวัดประสิทธิภาพในการสรุปข้อความของแบบแบบจำลองการเรียนรู้ด้วยเครื่อง ซึ่งตัวชี้วัดที่ได้รับความนิยมสูง คือ ROUGE-L และ ROUGE-N

2.22.2 งานวิจัยที่เกี่ยวข้อง

2.22.2.1 Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau) [20]

จากปัญหาของแบบจำลอง Sequence-to-Sequence ที่จะนำเพียงเวกเตอร์บริบทหรือเวกเตอร์ของชั้นซ่อนลำดับสุดท้ายในส่วนของ Encoder ไปทำนายผลลัพธ์ในส่วนของ Decoder ซึ่งจากการดำเนินการนี้ทำให้เวกเตอร์ในชั้นซ่อนอื่น ๆ มีความสำคัญลดน้อยลงทำให้สูญเสียความถูกต้องของผลลัพธ์ในชั้นซ่อน Decoder ลำดับยาว (ลำดับที่อยู่ไกล)

จากปัญหาในข้างต้นในงานวิจัยนี้นำเสนอสถาปัตยกรรมสำหรับการแปลภาษาด้วยเครื่อง โดยมีการนำเวกเตอร์ของชั้นซ่อน Encoder ในทุก ๆ ลำดับไปประมวลผลร่วมกับส่วนการทำงาน Decoder

ผลลัพธ์จากงานวิจัยนี้ ทำให้ได้แบบจำลองที่สามารถสร้างประโยคที่มีความหมายคล้ายคลึงกัน กล่าวคือในการสื่อความหมายหนึ่งๆ สามารถถูกเขียนได้ในหลายรูปแบบ หรือมีการเขียนรูปแบบคำที่แตกต่างกันแต่สามารถสื่อความหมายเดียวกันได้

บทที่ 3
วิธีการดำเนินงาน

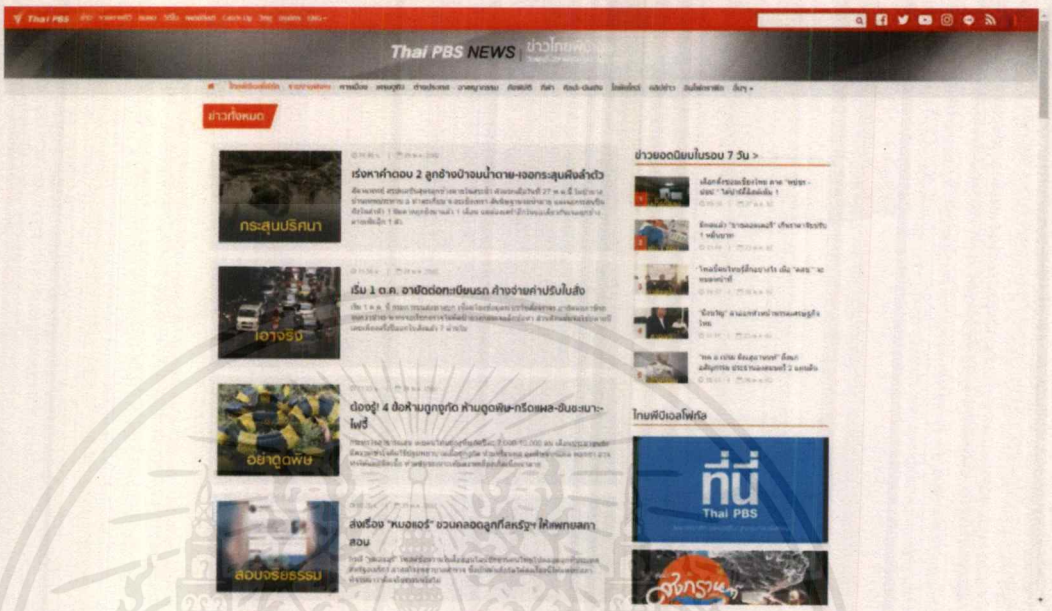
3.1 กระบวนการจัดเตรียมข้อมูล

ในการจัดเตรียมข้อมูลเพื่อใช้ในปัญหาพิเศษและเหมาะสมต่อการฝึกสอนแบบจำลองของเครื่อง มีขั้นตอนดังต่อไปนี้

1. เก็บรวบรวมข้อมูล
2. แปลงคำศัพท์ย่อ (Contractions)
3. ตัดคำ (Word Segmentation)
4. กำจัดคำหยุด (Stop Words)
5. กำจัดเครื่องหมายสัญลักษณ์
6. แปลงคำที่ไม่มีในพจนานุกรมเวกเตอร์เป็น “<UNK>”
7. เพิ่ม “<PAD>”, “<EOS>”, “<GO>” ในชุดข้อมูลเพื่อให้เหมาะสมต่อการสร้างแบบจำลองการเรียนรู้ของเครื่องประเภท Sequence-to-Sequence
8. แปลงคำศัพท์เป็นดัชนีของคำศัพท์ (Index)
9. แปลงดัชนีของคำศัพท์เป็นค่าเวกเตอร์ของคำศัพท์

3.1.1การเก็บรวบรวมข้อมูล (Data Integration)

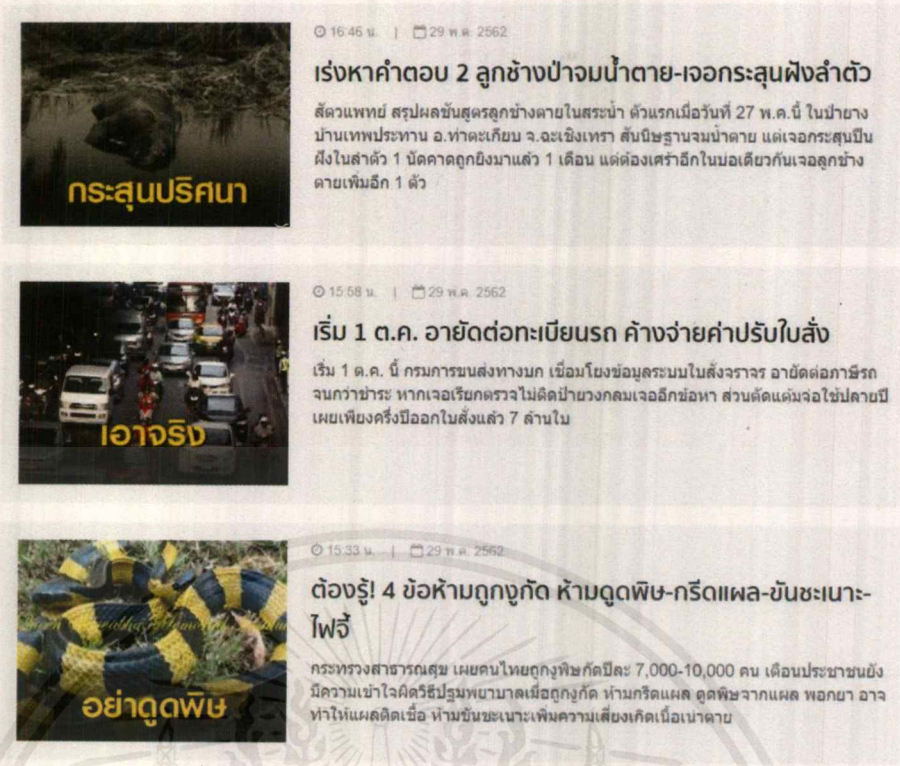
ข้อมูลที่นำมาใช้ในปัญหาพิเศษนี้รวบรวมมากจากเว็ปไซต์สถานีโทรทัศน์ไทยพีบีเอส ในส่วนของเมนูข่าวทั้งหมด ซึ่งจะแสดงข่าวทั้งหมดทุกประเภท เรียงลำดับตามเวลาในการเขียนข่าว



รูปที่ 3.1 เว็ปไซต์ ThaiPBS

(ที่มา: https://news.thaipbs.or.th/archive)

ข้อมูลที่นำมาสร้างแบบจำลองคือข้อมูลในส่วนของหัวข้อข่าวและข้อมูลข่าวอย่างสั้น จำนวน 29,170 ข่าว



รูปที่ 3.2 ส่วนของข้อมูลข่าวที่นำมาประมวลผล

3.1.2 แปลงคำศัพท์ย่อ (Contractions)

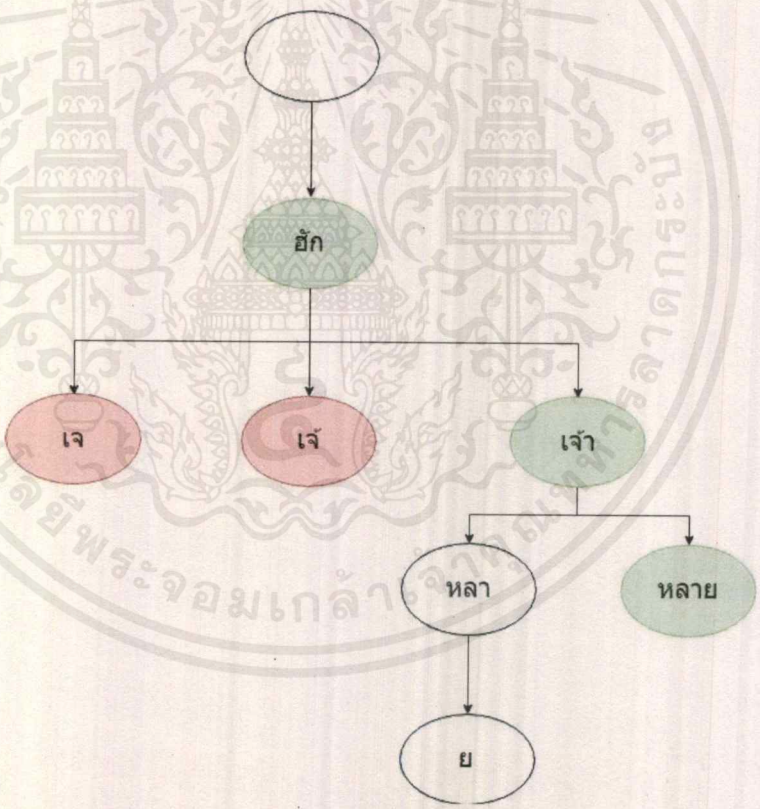
ในภาษาอังกฤษมีการลดรูปของคำอยู่จำนวนมาก เช่น it's, you're ซึ่งมีความหมายเหมือน it is และ you are ตามลำดับ ดังนั้นในขั้นตอนนี้จะดำเนินการเปลี่ยนแปลงคำเขียนย่อให้อยู่ในรูปคำเขียนเต็ม

3.1.3 ตัดคำ (Word Segmentation)

ในปัญหาพิเศษนี้ใช้ตัวตัดคำ newmm [21] ซึ่งตัวตัดคำนี้ได้ถูกพัฒนาขึ้นโดยใช้เทคนิคการตัดคำแบบสอดคล้องมากที่สุดร่วมกับเทคนิคการตัดคำด้วยกลุ่มตัวอักษรไทย (Thai Character Cluster: TCC) ซึ่งมีหลักการทำงานดังนี้

1. ค้นหาความเป็นไปได้ทั้งหมดในการกำหนดขอบเขตคำ ด้วยการพิจารณาจากคลังคำศัพท์ว่ามีคำใดบ้างที่เป็นส่วนประกอบ (Subset) ของชุดอักษรที่ต้องการกำหนดขอบเขตคำ โดยเริ่มต้นที่อักษรตัวแรก

2. พิจารณาตำแหน่งของผลลัพธ์ที่กำหนดขอบเขตคำจากขั้นตอนที่ 1 โดยคัดเลือกการกำหนดขอบเขตคำที่สอดคล้องกับกฎของ TCC
   - หากมีจำนวนผลลัพธ์จากการคัดเลือกมากกว่า 0 ดำเนินการจัดเก็บคำเหล่านั้นในโครงสร้างข้อมูลประเภทคิว (Queue)
   - หากจำนวนผลลัพธ์จากการคัดเลือกเท่ากับ 0 จะดำเนินการกำหนดขอบเขตคำตามกฎของ TCC และจัดเก็บชุดอักษรในโครงสร้างข้อมูลประเภทคิวเช่นเดียวกัน
3. ดำเนินการซ้ำในขั้นตอนที่ 1 โดยเริ่มต้นที่ตำแหน่งการค้นหาที่ตำแหล่งต่อจากผลลัพธ์ในขั้นตอนที่ 2 เรียงตามลำดับการจัดเก็บข้อมูล (คิว)
4. เมื่อสิ้นสุดการกำหนดขอบเขตคำแล้ว พิจารณาเลือกชุดอักษรที่ยาวที่สุดตามลำดับการดำเนินการ (คิว)



รูปที่ 3.3 ขั้นตอนการดำเนินการกำหนดขอบเขตคำด้วยเทคนิค newmm กับคำว่า “ฮักเจ้าหลาย”

จากรูปที่ 3.3 แสดงขั้นตอนการดำเนินการกำหนดขอบเขตคำด้วยเทคนิค newmm กับคำว่า “ฮักเจ้าหลาย” ดังนี้

1. พิจารณาความเป็นไปได้ในการกำหนดขอบเขตคำและได้ผลลัพธ์เพียง 1 คำ คือ “ฮัก”
2. พิจารณาความเป็นไปได้ในการกำหนดขอบเขตคำต่อจากคำว่า “ฮัก” และได้ผลลัพธ์ 3 คำ ได้แก่ “เจ”, “เจ้” และ “เจ้า”
3. พิจารณาผลลัพธ์ทั้ง 3 คำสอดคล้องตามกฎของ TCC หรือไม่ ซึ่งได้ผลลัพธ์ว่า “เจ”, “เจ้” ไม่สอดคล้องตามกฎ เพราะเหตุผลว่า หากกำหนดขอบเขตคำที่คำว่า “เจ” ในลำดับการดำเนินการต่อไป จะเริ่มต้นการค้นหาที่อักษร “้” ซึ่งไม่ถูกต้องตามกฎของ TCC และหากกำหนดขอบเขตที่คำว่า “เจ้” จะเริ่มต้นการค้นหาที่อักษร “า” ไม่ถูกต้องตามกฎของ TCC ด้วยเช่นกัน ดังนั้นผลลัพธ์ในขั้นตอนนี้มีเพียง 1 ผลลัพธ์คือคำว่า “เจ้า”
4. พิจารณาความเป็นไปได้ในการกำหนดขอบเขตคำต่อจากคำว่า “เจ้า” และได้ผลลัพธ์ 2 คำ ได้แก่ “หลา” และ “หลาย”
5. พิจารณาความเป็นไปได้ในการกำหนดขอบเขตคำต่อจากคำว่า “หลาย” ซึ่งมีเพียงอักษรเดียวคือ “ย” ซึ่งไม่พบในคลังคำศัพท์ จึงดำเนินการกำหนดขอบเขตคำด้วย TCC และได้ผลลัพธ์คือ “ย”
6. ดำเนินการคัดเลือกผลลัพธ์ทั้งหมดโดยเริ่มพิจารณาจากโหนดราก (Root Node) และเลือกชุดอักษรที่มีความยาวมากที่สุดในแต่ละลำดับชั้น (Level) ของแผนภาพต้นไม้ จะได้ผลลัพธ์คือ “ฮัก|เจ้า|หลาย”

คลังพจนานุกรมที่ใช้ในการในการตัดคำนำมาจากโมดูล PyThaiNLP [22] ร่วมกับคลังคำศัพท์จากโมดูล NLTK [23] รวมทั้งหมด 298,756 คำ และเพิ่มคำศัพท์ในเหมาะสมกับปัญหาพิเศษนี้ ได้แก่ กยศ., ซีพี, ดิวตี้, ทีซีดีซีเฟซบุค, บูโร, ฟูลมูน, มธ., รีเมค, ส.ส.ท., สดร., อีอีซี, อีโอดี, เอฟซี, แบงค์อก, แฮชแท็ก, ไอซ์, “, ”

3.1.4 กำจัดคำหยุด (Stop Words)

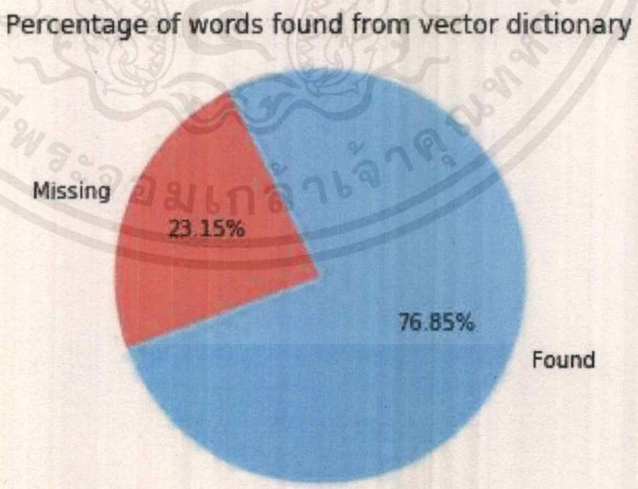
ในขั้นตอนนี้จะทำการลบคำในส่วนของอินพุต (Input) เพื่อกำจัดคำไม่มีความหมายออกไป เนื่องจากเป็นคำที่ไม่มีความสำคัญต่อการประมวลผล เช่น คำเชื่อม, ประติชญาวิเศษณ์ เป็นต้น แต่ไม่กำจัดคำหยุดในเอาต์พุต (Output) เนื่องจากต้องการให้ผลลัพธ์ที่ได้มีความเป็นธรรมชาติ เช่น “กินข้าวกับปลา” หากกำจัดคำหยุดจะได้ผลลัพธ์คือ “กินข้าวปลา” โดยใช้คลังคำศัพท์ Stop words จาก NLTK และ PyThaiNLP

3.1.5 กำจัดเครื่องหมายสัญลักษณ์

เนื่องจากในบางเครื่องหมายไม่มีความสำคัญต่อการประมวลผลจึงทำการกำจัดออกไป เช่น {, }, ~, # เป็นต้น ในกรณีเครื่องหมาย “.” จะไม่ทำการกำจัดออกเนื่องมีการใช้เครื่องหมายดังกล่าวในคำย่อต่าง ๆ เช่น i.e., U.S., กกต., กศน., ธ.ค. เป็นต้น

3.1.6 แปลงคำที่ไม่มีในพจนานุกรมเวกเตอร์เป็น “<UNK>”

สำหรับข้อมูลอินพุตของแบบจำลองจะเป็นค่าเป็นเวกเตอร์ของคำ โดยใช้พจนานุกรมเวกเตอร์ของ PyThaiNLP ซึ่งฝึกสอนด้วยแบบจำลอง Word2Vec ดังนั้นจึงต้องตรวจสอบก่อนว่ามีคำใดอยู่ในพจนานุกรมเวกเตอร์บ้าง และหากว่าอยู่ในรายการของพจนานุกรมเวกเตอร์จะทำการเปลี่ยนคำนั้น ๆเป็น Unknow ซึ่งจะใช้สัญลักษณ์คือ “<UNK>” ในกรณีพบคำศัพท์คำหนึ่งๆ ในชุดข้อมูลจำนวนมากแต่ไม่พบในรายการของพจนานุกรมเวกเตอร์ จะทำการสุ่มค่าของเวกเตอร์ขึ้นมาเพื่อเพิ่มในพจนานุกรม ซึ่ง



รูปที่ 3.4 แสดงอัตราส่วนของคำที่พบและไม่พบในพจนานุกรมเวกเตอร์

จากรูป 3.4 แสดงอัตราส่วนของคำที่พบและไม่พบในพจนานุกรมเวกเตอร์ ซึ่งมีจำนวนคำศัพท์ที่ไม่พบจำนวน 5,881 คำ คิดเป็นร้อยละ 23.15 และคำศัพท์ที่พบจำนวน 19,518 คำ คิดเป็นร้อยละ 76.85 จะได้ว่ามีคำในชุดข้อมูลทั้งหมดจำนวน 25,399 คำ

ในปัญหาพิเศษนี้กำหนดให้หากพบจำนวนคำหนึ่งๆปรากฏในชุดข้อมูลมากกว่า 20 ครั้งแต่ไม่มีค่าในพจนานุกรมเวกเตอร์ จะทำการสุ่มค่าของเวกเตอร์ให้คำนั้น ซึ่งมีจำนวน 359 คำ

3.1.7 เพิ่ม "<PAD>", "<EOS>", "<GO>" ในชุดข้อมูลเพื่อให้เหมาะสมต่อการสร้างแบบจำลองประเภท Sequence-to-Sequence

2. "<EOS>"

เพิ่มสัญลักษณ์ "<EOS>" ในข้อความอินพุตและข้อความเอาต์พุตเพื่อให้เหมาะสมต่อการฝึกสอนแบบจำลอง เช่น

[วิกฤต, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง]

จะได้ [วิกฤต, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง, <EOS>]

3. "<GO>"

เพิ่มสัญลักษณ์ "<GO>" ในข้อความเอาต์พุตเพื่อให้เหมาะสมต่อการฝึกสอนแบบจำลอง เช่น

[วิกฤต, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง]

จะได้ [<GO>, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง]

4. การ Padding

สำหรับข้อมูลอินพุตจำเป็นต้องทำให้ข้อมูลมีความยาวเท่ากันหรือมีจำนวนคำที่เท่ากัน โดยกำหนดให้อิงตามจำนวนคำในข้อความข่าวที่มีความยาวมากที่สุด ซึ่งในปัญหาพิเศษนี้มีความยาวเท่ากับ 49 คำ ดังนั้นแล้วหากข้อความข่าวใดมีความยาวน้อยกว่า 49 จะทำการเพิ่มสัญลักษณ์ "<PAD>" เข้าไปในด้านหลัง เช่น

[วิกฤต, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง, <EOS>,<PAD>, <PAD>, <PAD>, <PAD>, ..., <PAD>]

จากตัวอย่างในข้างต้นข้อความข่าวมีจำนวน 10 คำ (ไม่นับรวม "<EOS>") และทำการเพิ่ม "<PAD>" อีก 39 คำ เพื่อให้ได้ความยาวของข้อความครบ 49 คำ

นอกจากนี้มีการดำเนินการกับข้อมูลเอาต์พุตเช่นเดียวกัน แต่ความยาวในการเพิ่ม “<PAD>” จะอิงตามจำนวนคำในข้อความสรุปข่าวที่มีความยาวมากที่สุด ซึ่งในปัญหาพิเศษนี้มีความยาวเท่ากับ 26 คำ

ดังนั้นในการดำเนินการขั้นตอนนี้จะได้ผลลัพธ์เช่นตัวอย่างดังต่อไปนี้

กำหนดให้ ข้อความอินพุตมีความยาวสูงสุด 20 คำ

ข้อความเอาต์พุตมีความยาวสูงสุด 10 คำ

ไม่มีราการคำว่า “เปี้ยน” ในพจนานุกรมเวกเตอร์

ข้อความก่อนดำเนินการ

ข้อความข่าว: [โปร, ช้าง, ธงชัย, ใจดี, โชว์, ฟอร์ม, เยี่ยม, คว้าแชมป์, กอล์ฟ, ยูโร, เปี้ยน, ทัวร์, ฝรั่งเศส, ครอง, สำเร็จ]

ข้อความสรุปข่าว: [โปร, ช้าง, คว้าแชมป์, กอล์ฟ, ยูโร, เปี้ยน, ทัวร์, ที่, ฝรั่งเศส]

ผลลัพธ์จากการดำเนินการในขั้นตอนนี้

ข้อความข่าว: [โปร, ช้าง, ธงชัย, ใจดี, โชว์, ฟอร์ม, เยี่ยม, คว้าแชมป์, กอล์ฟ, ยูโร, <UNK>, ทัวร์, ฝรั่งเศส, ครอง, สำเร็จ, <EOS>, <PAD>, <PAD>, <PAD>, <PAD>, <PAD>]

ข้อความสรุปข่าว: [โปร, ช้าง, คว้าแชมป์, กอล์ฟ, ยูโร, <UNK>, ทัวร์, ที่, ฝรั่งเศส, <EOS>, <PAD>]

3.1.8 แปลงคำศัพท์เป็นดัชนีของคำศัพท์ (Index)

ในปัญหาพิเศษนี้ใช้เครื่องมือสำหรับการเรียนรู้เชิงลึกคือ TensorFlow ซึ่งเป็นโมดูลในการฝึกสอนและพัฒนาแบบจำลองการเรียนรู้ของเครื่องในภาษาไพธอน (Python) และในส่วนงานนี้ TensorFlow จะช่วยในการจัดการทรัพยากรของหน่วยความจำเพื่อให้เพียงพอต่อการอินพุตข้อมูลดังเช่นตัวอย่างต่อไปนี้

[วิกฤต, ไฟป่า, ใน, แคลิฟอร์เนีย, เผา, ทำลาย, อาคาร, บ้าน, 1500, หลัง]

จากประโยคในข้างต้น หากต้องการนำข้อมูลไปฝึกสอนกับแบบจำลอง จะต้องดำเนินการแปลงข้อมูลในรูปแบบของข้อความให้เป็นข้อมูลในรูปแบบของเวกเตอร์ดังที่ได้กล่าวนำในข้างต้นแล้ว ซึ่งคำหนึ่งๆนั้นจะถูกนำเสนอด้วยค่าเวกเตอร์จำนวน 300 มิติ ดังนั้นแล้วในประโยคข้างต้นจะมีมิติของข้อมูลคือ 10x300 ต่อข้อมูล 1 แถว หากมีข้อมูลจำนวน 1,000 แถว จะได้ว่า

1000x10x300 ซึ่งจากการดำเนินการนี้หากเป็นชุดข้อมูลที่มีปริมาณมากจะใช้พื้นที่หน่วยความจำมากและอาจไม่เพียงสำหรับผู้ที่มีทรัพยากรอย่างจำกัด

จากประโยคในข้างต้น กำหนดให้มีดัชนีเป็นดังนี้ [7, 9, 250, 88, 6, 5, 67, 65, 4500, 66] ซึ่งจะมีมิติของข้อมูลเพียง 1x10 ต่อข้อมูล 1 แถว หากมีข้อมูลจำนวน 1,000 แถว จะได้ว่ามีจำนวนมิติของข้อมูลเท่ากับ 1000x10

ด้วยเหตุนี้เอง TensorFlow จึงได้จัดเตรียมความช่วยเหลือในการแปลงคำศัพท์เป็นค่าเวกเตอร์ ซึ่ง TensorFlow จะรองรับข้อมูลประเภทดัชนีของคำศัพท์ ดังนั้นผู้พัฒนาโปรแกรมจึงจำเป็นต้องแปลงคำศัพท์เป็นดัชนีของคำศัพท์เพื่อให้ตรงตามที่ TensorFlow รองรับ

3.1.9 แปลงดัชนีของคำศัพท์เป็นค่าเวกเตอร์ของคำศัพท์

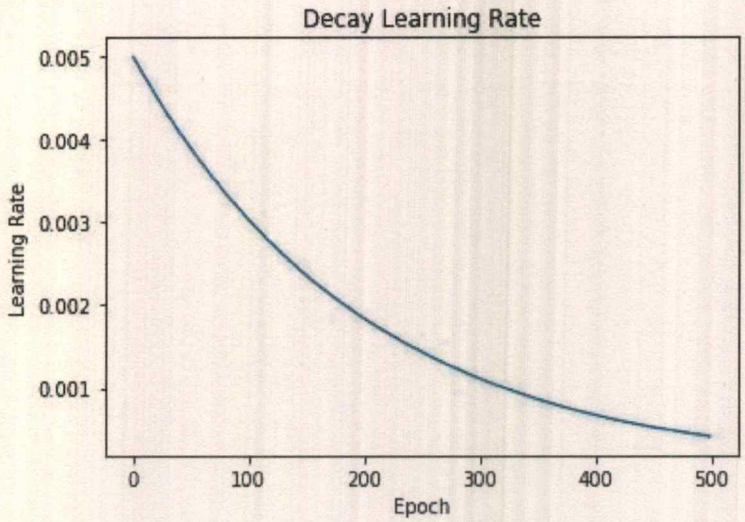
ในส่วนการทำงานนี้จะถูกนำเนินการในขั้นตอนการอินพุตข้อมูลไปยังแบบจำลอง ซึ่ง TensorFlow จะดำเนินการทั้งหมดให้โดยอัตโนมัติ สำหรับผู้พัฒนาโปรแกรม จะต้องจัดเตรียมให้ดัชนีของค่าเวกเตอร์ตรงตามดัชนีของคำศัพท์ตามที่ได้กล่าวไว้ในขั้นตอนที่ 8

3.2 สร้างแบบจำลองการเรียนรู้ของเครื่อง

3.2.1 การกำหนดค่า Hyperparameter

- จำนวน Epoch เท่ากับ 500
- Optimizer คือ Adam [24] และ RMSProp [25] ซึ่งเป็น Optimizer ที่ได้รับความนิยมสูงเนื่องจากได้ปรับปรุงในส่วนด้อยของ Optimizer ตัวอื่น ๆให้ดียิ่งขึ้น
- Loss Fuction คือ Categorical Cross-Entropy
- ขนาดแบทช์ เท่ากับ 256
- จำนวนของ Units ใน LSTM เซลล์ เท่ากับ 256
- จำนวน LSTM เลเยอร์ เท่ากับ 2
- มีการคำนวณปรับค่าถ่วงน้ำหนักทุก ๆ 20 แบทช์
- Dropout ใน LSTM เซลล์ เท่ากับ 0 %, 20 % และ 35 %
- อัตราการเรียนรู้ (Learning Rate) กำหนดให้ค่าเริ่มต้นเท่ากับ 0.005 และมีการลดลงของอัตราการเรียนรู้ในแต่ละ Epoch คือ

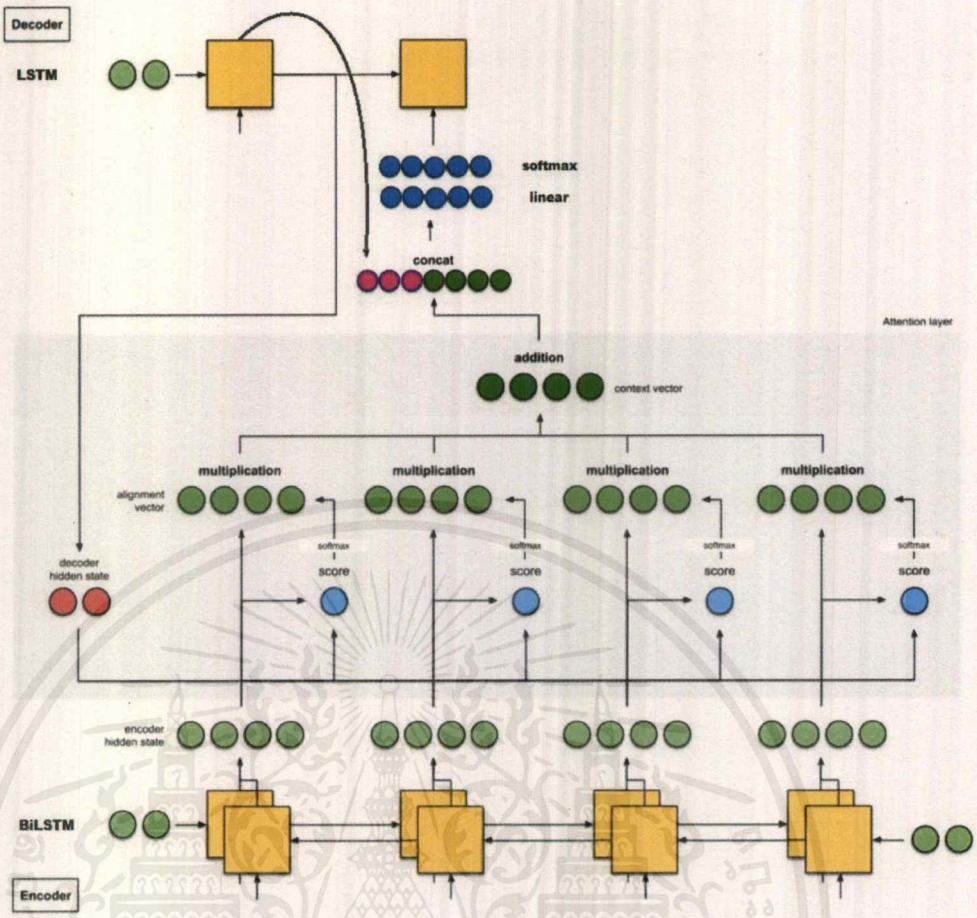
$$\text{Learning Rate}_{Epoch(i)} = \text{Learning Rate}_{Epoch(i-1)} * 0.995 \quad (3.1)$$



รูปที่ 3.5 แสดงการลดลงของอัตราการเรียนรู้ที่คำนวณได้จากสมการ 3.1 เมื่อกำหนดให้อัตราการเรียนรู้เริ่มต้นเท่ากับ 0.005

3.2.2 การออกแบบจำลองการเรียนรู้ของเครื่อง

1. Encoder
   - ใช้ Bi-directional LSTM
   - มีข้อมูลอินพุต คือ เวกเตอร์ของคำ
   - ใช้ Bahdanau ในการสร้าง Attention Layer
2. Decoder
   - ใช้ LSTM
   - ข้อมูลอินพุตคือเวกเตอร์ของคำ ซึ่งในการทำนายผลลัพธ์ว่าเป็นคำใดนั้นจะใช้โครงข่ายประสาทเทียม โดยมีรายละเอียดดังนี้
     - ชั้นอินพุต (Input Layer) มีข้อมูลอินพุต คือเวกเตอร์ผลลัพธ์ของ Decoder ในเซลล์สถานะก่อนหน้าและเวกเตอร์ของผลลัพธ์มาเรียงต่อกัน (Concatenate) โดยจำนวนนิวรอนเท่ากับ 768
     - ชั้นซ่อน (Hidden Layer) มีฟังก์ชันกระตุ้น คือฟังก์ชันเชิงเส้น (Linear Function) [26] โดยมีจำนวนนิวรอนเท่ากับ 19,878
     - ชั้นเอาต์พุต (Output Layer)) มีฟังก์ชันกระตุ้น คือฟังก์ชัน Softmax โดยมีจำนวนนิวรอนเท่ากับ 19,878



รูปที่ 3.6 แบบจำลอง Bahdanau

(แก้ไขรูปจาก: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3)

3.3 การวัดประสิทธิภาพของแบบจำลองการเรียนรู้ของเครื่อง (Model Evaluation)

สำหรับปัญหาพิเศษนี้ใช้ตัวชี้วัด ROUGE-1, ROUGE-2 และ ROUGE-L ในการวัดประสิทธิภาพของแบบจำลองการเรียนรู้ของเครื่อง ซึ่งในแต่ละตัวชี้วัดประกอบไปด้วยค่าวัดประสิทธิภาพ ได้แก่ Recall, Precision และ F-measure

บทที่ 4

ผลการทดลองและอภิปรายผล

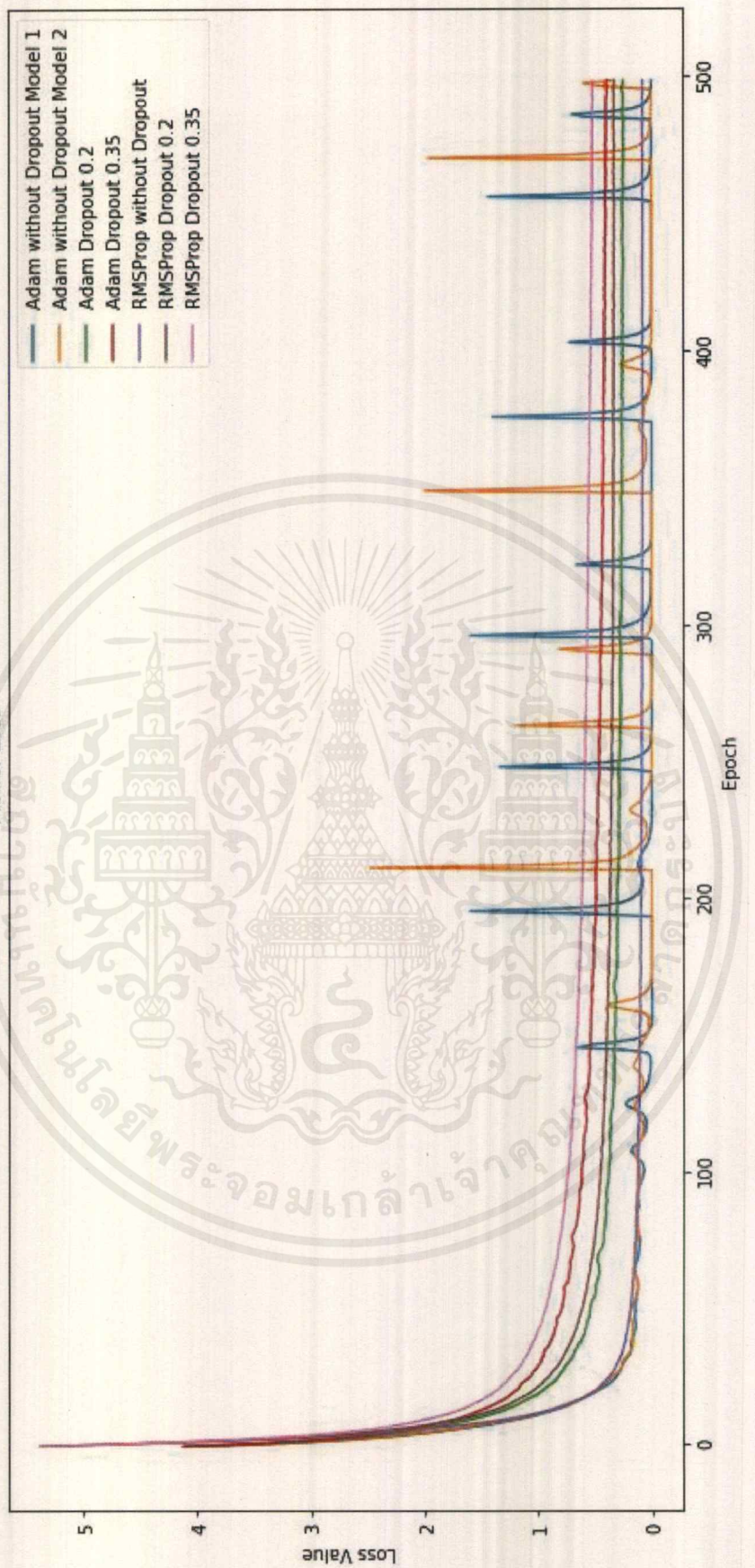
4.1 ผลการทดลอง

ในปัญหาพิเศษนี้มีการสร้างแบบจำลองทั้งหมด 7 แบบจำลอง ได้แก่

1. Adam without Dropout Model 1
2. Adam without Dropout Model 2
3. Adam Dropout 0.2
4. Adam Dropout 0.35
5. RMSProp without Dropout
6. RMSProp Dropout 0.2
7. RMSProp Dropout 0.35

ในการสร้างแบบจำลองโดยใช้ Adam Optimizer และไม่มีการ Dropout มีการสร้างแบบจำลอง 2 แบบจำลอง เนื่องจากพิจารณาพฤติกรรมของค่าความสูญเสีย (Loss) มีความผิดปกติ คือมีการเพิ่มขึ้นและลดลงในบางช่วง ได้แก่ แบบจำลองที่ 1: Adam without Dropout Model 1 และแบบจำลองที่ 2: Adam without Dropout Model 2

จากการทดลองสร้างแบบจำลองโดยการปรับเปลี่ยน Optimizer และค่าการ Dropout ได้ผลลัพธ์ค่าความสูญเสียดังรูปที่ 4.1



รูปที่ 4.1 แสดงค่าความสูญเสียของแบบจำลองในแต่ละ Epoch

ตารางที่ 4.1 แสดงผลลัพธ์ค่าความถูกต้องโดยตัวชี้วัด ROUGE-1, ROUGE-2 และ ROUGE-L ประกอบด้วยค่าชี้วัด Recall, Precision และ F-measure

Evaluation metric	Model	Precision	Recall	F-measure
ROUGE-1	Adam without Dropout Model 1	0.2456	0.2171	0.2237
	Adam without Dropout Model 2	**0.2518**	0.2213	**0.2285**
	Adam Dropout 0.2	0.2104	0.2131	0.2047
	Adam Dropout 0.35	0.2171	0.2241	0.2138
	RMSProp without Dropout	0.2462	0.2075	0.2187
	RMSProp Dropout 0.2	0.2425	**0.2247**	0.2268
	RMSProp Dropout 0.35	0.2298	0.216	0.216
ROUGE-2	Adam without Dropout Model 1	**0.0752**	0.073	**0.0715**
	Adam without Dropout Model 2	0.0739	0.0714	0.07
	Adam Dropout 0.2	0.0564	0.0635	0.0574
	Adam Dropout 0.35	0.0605	0.0705	0.0627
	RMSProp without Dropout	0.0695	0.0645	0.0647
	RMSProp Dropout 0.2	0.0729	**0.0749**	0.0714
	RMSProp Dropout 0.35	0.0628	0.0666	0.0624
ROUGE-L	Adam without Dropout Model 1	0.2003	0.2031	0.1855
	Adam without Dropout Model 2	**0.2027**	0.2079	**0.189**
	Adam Dropout 0.2	0.1685	0.1989	0.165
	Adam Dropout 0.35	0.1714	0.2086	0.1698
	RMSProp without Dropout	0.1975	0.194	0.1809
	RMSProp Dropout 0.2	0.1964	**0.2114**	0.1878
	RMSProp Dropout 0.35	0.1825	0.203	0.1759

4.2 อภิปรายผลการทดลอง

จากตารางที่ 4.1 แสดงคะแนนถูกต้องในการสรุปเนื้อหาข่าว ซึ่งมีการคำนวณด้วยตัวชี้วัด ROUGE-1, ROUGE-2 และ ROUGE-L แบบจำลองที่ใช้ Adam Optimizer และไม่มีการ Dropout จะให้ผลลัพธ์ค่า Recall และ F-measure ดีกว่าแบบจำลองอื่น ๆ ในทุก ๆ ตัวชี้วัด และแบบจำลองที่ใช้ RMSProp Optimizer และมีการ Dropout 20 เปอร์เซ็นต์ จะมีค่า Precision ดีกว่าแบบจำลองอื่น ๆ ในทุก ๆ ตัวชี้วัด

จากรูปที่ 4.1 แสดงกราฟการค่าความสูญเสียในการเรียนรู้ในแต่ละ Epoch ของแบบจำลอง ซึ่งจากพฤติกรรมของค่าความสูญเสีย แสดงให้เห็นว่ามีค่าค่อนข้างคงที่ตั้งแต่ก่อนจะมีการเรียนรู้ครบจำนวน 100 Epoch เสียอีก เว้นแต่พฤติกรรมค่าความสูญเสียของแบบจำลอง ที่ใช้ Adam Optimizer และไม่มีการ Dropout จะมีการเพิ่มขึ้นของค่าความสูญเสียอย่างรวดเร็วและลดลงอย่างรวดเร็วในบางช่วง ซึ่งคาดการณ์ว่าอาจเป็นผลมาจากการมีข้อมูลที่เป็นข้อมูลรบกวน (Noise) หรืออาจมีการจัดเตรียมข้อมูลสำหรับการเรียนรู้ที่ไม่ถูกต้อง จึงได้มีการทดลองสร้างแบบจำลองจำนวน 2 แบบจำลองด้วยการกำหนดค่าต่าง ๆ ที่เหมือนกัน เพื่อศึกษาว่าจะเกิดพฤติกรรมค่าความสูญเสียเช่นนี้อีกหรือไม่ ซึ่งจากผลลัพธ์ว่าทั้ง 2 แบบจำลองมีพฤติกรรมค่าความสูญเสียในทำนองเดียว คือจะมีการเพิ่มขึ้นของค่าความสูญเสียอย่างรวดเร็วและลดลงอย่างรวดเร็วในบางช่วง และเมื่อนำแบบจำลองทั้ง 2 ไปวัดประสิทธิภาพการสรุปข้อความข่าว พบว่าแบบจำลองทั้ง 2 ให้ผลลัพธ์ค่าความถูกต้องที่ใกล้เคียงกัน

นอกจากนี้กราฟดังกล่าวยังแสดงให้เห็นว่าการ Dropout ที่มากขึ้น ทำให้เกิดค่าความสูญเสียที่มากขึ้น และเมื่อพิจารณาตารางที่ 4.2 จะเห็นได้ว่าเมื่อมีการ Dropout ที่มากขึ้นไม่ได้ทำให้มีค่าความถูกต้องในการสรุปเนื้อหาข่าวที่มากขึ้นตาม

บทที่ 5
สรุปผลการทดลองและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

ปัญหาพิเศษนี้เป็นการศึกษาและนำเอาแบบจำลอง Bahdanau มาประยุกต์ใช้กับการสรุปข้อความข่าวภาษาไทย โดยข้อมูลที่นำมาใช้ในการสรุปข้อความข่าวคือข้อความข่าวจากสถานีโทรทัศน์ไทยพีบีเอส ซึ่งเป็นข้อมูลในส่วนของหัวข้อข่าวและข้อมูลข่าวอย่างสั้นจำนวน 29,170 ข่าว และใช้ตัวชี้วัด ROUGE ในการวัดประสิทธิภาพ

จากการทดลองสร้างแบบจำลองการเรียนรู้ด้วยเครื่องและวัดประสิทธิภาพได้ผลลัพธ์ว่ามีค่าความถูกต้องน้อย เนื่องจากมีจำนวนข้อมูลสำหรับการเรียนรู้ที่จำกัด ซึ่งไม่เหมาะสมต่อการเรียนรู้เชิงลึกที่ต้องการปริมาณข้อมูลจำนวนมาก แต่จากการพิจารณาผลลัพธ์การสร้างข้อความสรุปข่าวของแบบจำลองพบว่าผลลัพธ์มีความสนใจ คือแบบจำลองสามารถสร้างคำสรุปข่าวที่แตกต่างจากคำในเนื้อหาข่าวได้ โดยคำเหล่านั้นจะมีการสื่อความหมายที่เหมือนหรือคล้ายคลึงกัน

5.2 ข้อเสนอแนะ

1. หากต้องการใช้การเรียนรู้เชิงลึกในการสร้างแบบจำลองการเรียนรู้ของเครื่อง ควรจะมีข้อมูลจำนวนมากจึงจะเหมาะสม
2. ในการจัดสร้างแบบจำลองควรมีทรัพยากรในการประมวลผลที่สูง ซึ่งในปัจจุบันนิยมใช้หน่วยประมวลผลกราฟิกส์ เนื่องจากมีกำลังในการประมวลผลสูงและมีซอฟแวร์รองรับในการพัฒนาแบบจำลอง
3. ตัวตัดคำภาษาไทยยังไม่สามารถตัดคำศัพท์เฉพาะได้ดีเท่าที่ควร เช่น ชื่อของบุคคล ชื่อของสถานที่ เป็นต้น ซึ่งสามารถแก้ไขได้โดยการเพิ่มคำศัพท์ในพจนานุกรม ดังนั้นจึงควรมีการจัดสร้างพจนานุกรมดังกล่าวให้ครอบคลุมคำศัพท์เฉพาะมากที่สุด
4. ค่าเวกเตอร์ของคำในภาษาไทยยังจำเป็นต้องมีการพัฒนาต่ออีก เพื่อให้มีคลังคำศัพท์ที่มากขึ้นและเพื่อให้มีความถูกต้องของค่าเวกเตอร์ที่มากขึ้น
5. ในปัจจุบันมีการพัฒนาเทคนิคต่าง ๆ จำนวนมากสำหรับการแก้ปัญหาการสรุปข้อความ และยังมีเทคนิคในการแก้ปัญหาอื่น ๆ ที่มีความเกี่ยวข้องกัน เช่น การแปลภาษาด้วยเครื่อง ดังนั้นจำเป็นต้องมีการทดลองอีกมากสำหรับการนำมาประยุกต์ใช้กับภาษาไทย

เอกสารอ้างอิง

[1] Thanaruk Theeramunkong, Virach Sornlertlamvanich, Thanasan Tanhermhong, Wirat Chinnan. 2000. "Character cluster based Thai information retrieval"

[2] Tomas Mikolov. 2013. "Distributed Representations of Words and Phrases and their Compositionality"

[3] "Categorical Cross-Entropy" [Online] Available from: https://en.wikipedia.org/wiki/Cross_entropy Retrieved April 6, 2019

[4] Chin-Yew Lin. 2004. "ROUGE: A Package for Automatic Evaluation of Summaries"

[5] "Longest Common Subsequence Problem" [Online] Available from: https://en.wikipedia.org/wiki/Longest_common_subsequence_problem Retrieved June 1, 2019

[6] "Deep learning" [Online] Available from: https://en.wikipedia.org/wiki/Deep_learning Retrieved April 7, 2019

[7] "Dropout (neural networks)" [Online] Available from: https://en.wikipedia.org/wiki/Dropout_(neural_networks) Retrieved April 7, 2019

[8] "Long Short-Term Memory (LSTM)" [Online] Available from: https://medium.com/@sinart.t/long-short-term-memory-lstm-e6cb23b494c6 Retrieved April 9, 2019

[9] "Hyperbolic function" [Online] Available from: https://en.wikipedia.org/wiki/Hyperbolic_function Retrieved April 12, 2019

[10] "Rectifier (neural networks)" [Online] Available from: https://en.wikipedia.org/wiki/Rectifier_(neural_networks) Retrieved April 12, 2019

[11] "Sigmoid function" [Online] Available from: https://en.wikipedia.org/wiki/Sigmoid_function Retrieved April 12, 2019

[12] "Softmax function" [Online] Available from: https://en.wikipedia.org/wiki/Softmax_function Retrieved April 12, 2019

[13] "Backpropagation" [Online] Available from: https://en.wikipedia.org/wiki/Backpropagation Retrieved April 13, 2019

[14] "Vanishing gradient problem" [Online] Available from: https://en.wikipedia.org/wiki/Vanishing_gradient_problem Retrieved April 13, 2019

[15] "Understanding Bidirectional RNN in PyTorch" [Online] Available from: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66 Retrieved April 15, 2019

[16] "Sequence-to-sequence Models" [Online] Available from: https://nlp.stanford.edu/~johnhew/public/14-seq2seq.pdf Retrieved April 16, 2019

[17] "Attn: Illustrated Attention" [Online] Available from: https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3 Retrieved April 21, 2019

[18] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, Chandan K. Reddy. 2018. "Neural Abstractive Text Summarization with Sequence-to-Sequence Models"

[19] "Abstractive Text Summarization" [Online] Available from: http://home.iitk.ac.in/~soumye/cs498a/report.pdf Retrieved May 4, 2019

[20] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. 2014. "Neural Machine Translation by Jointly Learning to Align and Translate"

[21] "newmm algorithm" [Library] Available from: https://github.com/PyThaiNLP/pythainlp/blob/dev/pythainlp/tokenize/newmm.py

[22] "PyThaiNLP" [Library] Available from: https://pythainlp.github.io/

[23] "NLTK" [Library] Available from: https://www.nltk.org/

[24] Diederik P. Kingma, Jimmy Ba. 2017. "Adam: A Method for Stochastic Optimization"

[25] "Neural Networks for Machine Learning – Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude"
[Online] Available from:
https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf
Retrieved May 15, 2019

[26] "Linear predictor function" [Online] Available from:
https://en.wikipedia.org/wiki/Linear_predictor_function
Retrieved April 12, 2019

ภาคผนวก

ภาคผนวก ก

รายการคำศัพท์ย่อ (Constructions)

ain't: am not	aren't: are not	can't: can not
can't've: cannot have	'cause: because	could've: could have
couldn't: could not	couldn't've: could not have	didn't: did not
doesn't: does not	don't: do not	hadn't: had not
hadn't've: had not have	hasn't: has not	haven't: have not
he'd: he would	he'd've: he would have	he'll: he will
he's: he is	how'd: how did	how'll: how will
how's: how is	i'd: i would	i'll: i will
i'm: i am	i've: i have	isn't: is not
it'd: it would	it'll: it will	it's: it is
let's: let us	ma'am: madam	mayn't: may not
might've: might have	mightn't: might not	must've: must have
mustn't: must not	needn't: need not	sha'n't: shall not
she'd: she would	she'll: she will	she's: she is
should've: should have	that's: that is	there'd: there had
there's: there is	they'd: they would	they'll: they will
they're: they are	they've: they have	wasn't: was not
we'd: we would	we'll: we will	we're: we are
we've: we have	weren't: were not	what'll: what will
what're: what are	what's: what is	what've: what have
where'd: where did	where's: where is	who'll: who will
who's: who is	won't: will not	wouldn't: would not
you'd: you would	you'll: you will	you're: you are
%: เปอร์เซ็นต์	\xa0: (ช่องว่าง 1 ช่อง)	

oughtn't: ought not shan't: shall not

shouldn't: should not that'd: that would

ภาคผนวก ข

ตัวอย่างผลลัพธ์การสรุปข้อความข่าว

ตัวอย่างการสรุปข้อความข่าวของแบบจำลองต่าง ๆ โดยมีลำดับผลลัพธ์ดังนี้

1. Adam without Dropout Model 1
2. Adam without Dropout Model 2
3. Adam Dropout 0.2
4. Adam Dropout 0.35
5. RMSProp without Dropout
6. RMSProp Dropout 0.2
7. RMSProp Dropout 0.35

สวนดุสิตโพลเผยประชาชนรู้สึกเบื่อหน่ายการจัดตั้งรัฐบาล โดยมองว่าเกิดจากผลประโยชน์ทางการเมืองไม่ลงตัว-แย่งชิงเก้าอี้รัฐมนตรี จึงอยากให้ยึดผลประโยชน์ของประเทศชาติเป็นหลัก ขณะที่นิด้าโพลเผยสำรวจ 10 สาเหตุที่ทำให้พรรคประชาธิปัตย์ "แพ้เลือกตั้ง-แตกแยก"

1. โพล|นิด้า|ช่วย|ร้อง|รัฐบาล|แก้ปัญหา|อดีต|นายทหาร|เกษียณ
2. โพล|ชี้|ผู้ประกันตน|12|ปี|เหตุ|แพ้|เลือกตั้ง
3. โพล|ชี้|ต้อง|มี|ผลกระทบ|คน|วิจารณ์|เลือกตั้ง
4. โพล|ชี้|อนาคต|ต้อง|เป็น|ของ|ทักษิณ|หยุด|ทุจริต|จาก|เหตุ|เกษียณ
5. ผล|โพล|โพล|หลัง|ผล|ทดสอบ|กลุ่ม|การเมือง
6. โพล|ชี้|จับ|10|เก้าอี้|กลุ่ม|การเมือง
7. เลือกตั้ง|2562|เช็ก|ด่วน|ปชช.|ยัง|ไม่|ล้มเหลว|ทางการเมือง

เจ้าหน้าที่ท่าอากาศยานสุวรรณภูมิ ควบคุมตัวชาวจีน 2 คน พร้อมของกลาง เป็นทรัพย์สินที่ขโมยจากผู้โดยสารต่างชาติระหว่างเดินทางเข้าไทย เจ้าหน้าที่เตรียมเสนอเพิกถอนการอนุญาตให้อยู่ต่อในราชอาณาจักร พร้อมทั้งดำเนินคดี หลังเห็นว่ามีพฤติกรรมเป็นภัยสังคม

1. คุมตัว|ชาวจีน|ใน|สุวรรณภูมิ|เร่ง|รื้อ|ตั๋ว|วัน|แรก
2. รวบ|ผู้|เข้า|ออ|ฮอลลีวูด|เพิ่ม|อำนาจ|ผู้|รุก|ใน|ไทย
3. จนท.|ที่|ใช้|เหยี่ยว|โดยสาร|เหยื่อ|ถูก|เรียก|แป๊ะเจี๊ยะ
4. จับ|ชาวจีน|ประสาน|ชาวจีน|ต้อง|ออกจาก|ไทย|จับ|เพิ่ม
5. นัก|ดำน้ำ|เกาหลีเหนือ|เตรียม|ออกหมายจับ|เพิ่ม|คน|ก่อน|ขโมย|ทรัพย์สิน|มา|แกะรอย
6. จนท.|เร่ง|แก๊ง|หลัง|ขโมย|กระเป๋า|ใน|ไทย
7. จนท.|จับ|ชาวจีน|ค้า|คำสั่ง|ลักทรัพย์|ใน|สนามบิน|สุวรรณภูมิ

พล.อ.สุรยุทธ์ จุลานนท์ รักษาการประธานองคมนตรี เป็นผู้แทนพระองค์ ไปในการบำเพ็ญพระราชกุศลทักษิณานุปทาน 7 วัน พระราชทานศพ พล.อ.เปรม ติณสูลานนท์ อดีตประธานองคมนตรีและรัฐบุรุษ โดยมีบุคคลสำคัญและประชาชนเข้าร่วมพิธี

1. จาก|พล|อ.|เปรม|ชัย|นั่ง|บัญชี|ให้|ล.|อ.|เปรม|เป็น|ประธาน
2. ล.|อ.|เปรม|นั่ง|หัวหน้าพรรค|300|ตำแหน่ง|300|เปอร์เซ็นต์
3. พล|อ.|เปรม|เปิด|พิธี|แสดง|ความอาลัย|อดีต|ประธาน|องคมนตรี
4. ล.|อ.|เปรม|เชื่อ|เป็น|นักธุรกิจ|องคมนตรี|ทำบุญ|นำ|ไป|กลับ|พระ|ราช|กุศล
5. องคมนตรี|วิ่ง|เปรม|ลาออก|พล|อ.|เปรม|เป็น|พล|อ.|เปรม|เป็น|พล|อ.|เปรม
6. พล|อ.|เปรม|ชัย|เผย|ประชาชน|ร่วม|แสดง|ขอให้|บรรจุ|ศพ|ล.|อ.|เปรม
7. ล.|อ.|เปรม|รับ|พิธี|เป็น|ประธาน|องคมนตรี

คปภ.แจ้งสิทธิค่ารักษาพยาบาลและค่าสินไหมทดแทนตาม พ.ร.บ. ให้กับผู้บาดเจ็บและผู้เสียชีวิตแล้ว กรณีรถเบนซ์พุ่งชน บริเวณถนนงามวงศ์วาน พร้อมเร่งติดตามบริษัทประกันภัย

1. ติดตาม|ข้อมูล|ขรก.|คาด|ว่า|เร่ง|ติดตาม|ผู้ก่อเหตุ|ใน|อดีต|ผู้ใหญ่บ้าน
2. คปภ.|เตรียม|ประเมิน|20|ม.ค.|นี้|เร่ง|ค้นหา|รถ|เสียงดัง|เบ|จ.|งาม
3. ตั้ง|ข้อกล่าวหา|สาเหตุ|รถ|เบนซ์|ออก|โฉนดที่ดิน|ถ.|พหลโยธิน
4. คปภ.|แจ้ง|ข้อกล่าวหา|คน|แอบอ้าง|ทำ|ญาติ|ออก|เบรก|การคุม|เข้ม|เยียวยา|3000|หมื่น|คน
5. คปภ.|งัด|กรม|ศิ|ไกร|เพชรบูรณ์|เสียชีวิต|ตร.|เร่ง|ตรวจสอบ|เพิ่ม|ตาม|ควบคุม|น้อง|ตาย|ควบคุม
6. คปภ.|เผย|เรียกร้อง|ตรวจ|เค|ดา|ปา|เข้า|ก่อสร้าง|อิตาเลียน
7. คปภ.|เรียกร้อง|รอ|การ|ใช้|สิทธิ์|นิ่ง

เกิดเหตุผู้ก่อเหตุ 2 คนขี่จักรยานยนต์ ปาระเบิดไม่ทราบชนิดใส่ป้อมตำรวจที่ อ.บันนังสตา จ.ยะล วัยรุ่นชาย อายุ 17 ปี โดนลูกหลงบาดเจ็บ 1 คน

1. ระทึก|พลัดตก|บาดเจ็บ|ก่อน|ปา|ระเบิด|ใส่|ประทัด
2. รถกระบะ|ใหม่|ปิ้น|จาก|ปา|แก๊ส|ใส่|นักรบ|ตร.|สังเวย|บาดเจ็บ|คน
3. เกิดเหตุ|ขว้าง|ระเบิด|ใส่|บลู|ปา|แก้ว|ใส่|ฟิลิปปินส์|พุ่ง|เจ็บ|สาหัส
4. มือ|ปา|ระเบิด|ใส่|หัว|ตร.|ใส่|จาก|เหตุ|ใส่|ใส่|บาง|สะพาน|ตร.|เสียชีวิต|คน|ใส่|เร็ว
5. พิษ|ปา|ถล่ม|ใส่|ระเบิด|ใน|จ.|ยะ|ไม่มี|เป็น|18|สาหัส
6. หลักฐาน|ชัด|อีก|ศพ|แก๊ง|ปา|ระเบิด|ใส่|รถกระบะ|ใส่|พลิกคว่ำ|ปา|บึก|เจ็บ
7. เกิดเหตุ|ปา|ระเบิด|ใส่|ร้าน|สี|ปา|โฮ|ใส่|ร้าน|ตำรวจ|จ.|กาญจนบุรี|บาดเจ็บ|คน

ภาคผนวก ค

โค้ดที่ใช้ในการสร้างแบบจำลอง

1. โค้ดสร้างแบบจำลอง Adam without Dropout Model 1

```
from tensorflow.layers import Dense
from tensorflow.nn.rnn_cell import BasicLSTMCell
import numpy as np
import tensorflow as tf
import re
import time
import json

print('TensorFlow Version: {}'.format(tf.__version__))
```

```
TensorFlow Version: 1.13.1
```

```
import json


def read_json(fname, key_int=False):
    with open(fname, 'r') as file:
        data = file.read()
        json_data = json.loads(data)

        if not key_int:
            return json_data

        json_data = {int(key): value for key, value in json_data.items()}
        return json_data


def to_json(fname, data):
    with open(fname, 'w') as file:
        json_data = json.dumps(data)
        file.write(json_data)


def to_txt(fname, data):
    with open(fname, 'w', encoding='utf-8') as file:
        data = [str(d) for d in data]
        file.write('\n'.join(data))
```

Load value

```
vocab_to_int = read_json('../../vocab_to_int.json')
int_to_vocab = read_json('../../int_to_vocab.json', key_int=True)
word_embedding_matrix = np.load('../../word_embedding_matrix.npy') # load

# dataset
dataset = read_json('../../dataset.json')
X_train = dataset['X_train']
X_test = dataset['X_test']
y_train = dataset['y_train']
y_test = dataset['y_test']
```

```
print('Train: ', len(X_train))
print('Test: ', len(X_test))
```

```
Train:  21393
Test:  2377
```

Building the Model

```
def model_inputs():
    '''Create palceholders for inputs to the model'''

    input_data = tf.placeholder(tf.int32, [None, None], name='input')
    targets = tf.placeholder(tf.int32, [None, None], name='targets')
    lr = tf.placeholder(tf.float32, name='learning_rate')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')
    summary_length = tf.placeholder(tf.int32, (None,), name='summary_length')
    max_summary_length = tf.reduce_max(summary_length, name='max_dec_len')
    text_length = tf.placeholder(tf.int32, (None,), name='text_length')

    return input_data, targets, lr, keep_prob, summary_length, max_summary_length, text_length
```

```
def process_encoding_input(target_data, vocab_to_int, batch_size):
    '''Remove the last word id from each batch and concat the <GO> to the begining of each batch'''

    ending = tf.strided_slice(target_data, [0, 0], [batch_size, -1], [1, 1])
    dec_input = tf.concat([tf.fill([batch_size, 1], vocab_to_int['<GO>']), ending], 1)

    return dec_input
```

```
def encoding_layer(rnn_size, sequence_length, num_layers, rnn_inputs, keep_prob):
    '''Create the encoding layer'''

    for layer in range(num_layers):
        with tf.variable_scope('encoder_{}'.format(layer)):
            cell_fw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            cell_fw = tf.contrib.rnn.DropoutWrapper(cell_fw,
                                                    input_keep_prob = keep_prob)

            cell_bw = tf.contrib.rnn.LSTMCell(rnn_size,
                                              initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            cell_bw = tf.contrib.rnn.DropoutWrapper(cell_bw,
                                                    input_keep_prob = keep_prob)

            enc_output, enc_state = tf.nn.bidirectional_dynamic_rnn(cell_fw,
                                                                    cell_bw,
                                                                    rnn_inputs,
                                                                    sequence_length,
                                                                    dtype=tf.float32)
    # Join outputs since we are using a bidirectional RNN
    enc_output = tf.concat(enc_output,2)

    return enc_output, enc_state
```

```
def training_decoding_layer(dec_embed_input, summary_length, dec_cell, initial_state, output_layer,
                            vocab_size, max_summary_length):
    '''Create the training logits'''

    training_helper = tf.contrib.seq2seq.TrainingHelper(inputs=dec_embed_input,
                                                        sequence_length=summary_length,
                                                        time_major=False)

    training_decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell,
                                                       training_helper,
                                                       initial_state,
                                                       output_layer)

    training_logits, _, _ = tf.contrib.seq2seq.dynamic_decode(training_decoder,
                                                              output_time_major=False,
                                                              impute_finished=True,
                                                              maximum_iterations=max_summary_length)

    return training_logits
```

```python
def inference_decoding_layer(embeddings, start_token, end_token, dec_cell, initial_state, output_layer,
                             max_summary_length, batch_size):
    '''Create the inference logits'''

    start_tokens = tf.tile(tf.constant([start_token], dtype=tf.int32), [batch_size], name='start_tokens')

    inference_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embeddings,
                                                                start_tokens,
                                                                end_token)

    inference_decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell,
                                                        inference_helper,
                                                        initial_state,
                                                        output_layer)

    inference_logits, _, _ = tf.contrib.seq2seq.dynamic_decode(inference_decoder,
                                                               output_time_major=False,
                                                               impute_finished=True,
                                                               maximum_iterations=max_summary_length)

    return inference_logits
```

```python
def decoding_layer(dec_embed_input, embeddings, enc_output, enc_state, vocab_size, text_length, summary_length,
                   max_summary_length, rnn_size, vocab_to_int, keep_prob, batch_size, num_layers):
    '''Create the decoding cell and attention for the training and inference decoding layers'''

    for layer in range(num_layers):
        with tf.variable_scope('decoder_{}'.format(layer)):
            lstm = tf.contrib.rnn.LSTMCell(rnn_size,
                                           initializer=tf.random_uniform_initializer(-0.1, 0.1, seed=2))
            dec_cell = tf.contrib.rnn.DropoutWrapper(lstm,
                                                     input_keep_prob = keep_prob)

    # projection_layer
    output_layer = Dense(units=vocab_size,
                         activation='linear',
                         kernel_initializer = tf.truncated_normal_initializer(mean = 0.0, stddev=0.1))

    attn_mech = tf.contrib.seq2seq.BahdanauAttention(rnn_size,
                                                     enc_output,
                                                     text_length,
                                                     normalize=False,
                                                     name='BahdanauAttention')

    dec_cell = tf.contrib.seq2seq.AttentionWrapper(dec_cell,
                                                   attn_mech,
                                                   rnn_size)

    initial_state = dec_cell.zero_state(dtype=tf.float32, batch_size=batch_size)

    with tf.variable_scope("decode"):
        training_logits = training_decoding_layer(dec_embed_input,
                                                  summary_length,
                                                  dec_cell,
                                                  initial_state,
                                                  output_layer,
                                                  vocab_size,
                                                  max_summary_length)
    with tf.variable_scope("decode", reuse=True):
        inference_logits = inference_decoding_layer(embeddings,
                                                    vocab_to_int['<GO>'],
                                                    vocab_to_int['<EOS>'],
                                                    dec_cell,
                                                    initial_state,
                                                    output_layer,
                                                    max_summary_length,
                                                    batch_size)

    return training_logits, inference_logits
```

```python
def seq2seq_model(input_data, target_data, keep_prob, text_length, summary_length, max_summary_length,
                  vocab_size, rnn_size, num_layers, vocab_to_int, batch_size):
    '''Use the previous functions to create the training and inference logits'''

    # Use Numberbatch's embeddings and the newly created ones as our embeddings
    embeddings = word_embedding_matrix

    enc_embed_input = tf.nn.embedding_lookup(embeddings, input_data)
    enc_output, enc_state = encoding_layer(rnn_size, text_length, num_layers, enc_embed_input, keep_prob)

    dec_input = process_encoding_input(target_data, vocab_to_int, batch_size)
    dec_embed_input = tf.nn.embedding_lookup(embeddings, dec_input)

    training_logits, inference_logits  = decoding_layer(dec_embed_input,
                                                        embeddings,
                                                        enc_output,
                                                        enc_state,
                                                        vocab_size,
                                                        text_length,
                                                        summary_length,
                                                        max_summary_length,
                                                        rnn_size,
                                                        vocab_to_int,
                                                        keep_prob,
                                                        batch_size,
                                                        num_layers)

    return training_logits, inference_logits
```

```python
def find_max(data):
    return max([len(d) for d in data])


def pad_sentence_batch(sentence_batch, max_sentence=None):
    """Pad sentences with <PAD> so that each sentence of a batch has the same length"""
    if max_sentence is None:
        max_sentence = find_max(sentence_batch)
    else:
        pass

    sent_pad = []
    for sentence in sentence_batch:
        sentence = sentence + [vocab_to_int['<PAD>']] * (max_sentence - len(sentence))
        sentence = sentence[:max_sentence]
        sent_pad.append(sentence)
    return sent_pad
```

```python
def get_batches(summaries, texts, batch_size):
    """Batch summaries, texts, and the lengths of their sentences together"""
    len_ = len(texts) // batch_size
    if len_ is 0:
        raise "Error: dataset size < batch size, performance of model to low"

    for batch_i in range(0, len_):
        start_i = batch_i * batch_size
        summaries_batch = summaries[start_i:start_i + batch_size]
        texts_batch = texts[start_i:start_i + batch_size]
        pad_summaries_batch = np.array(pad_sentence_batch(summaries_batch))
        pad_texts_batch = np.array(pad_sentence_batch(texts_batch))

        # Need the lengths for the _lengths parameters
        pad_summaries_lengths = []
        for summary in pad_summaries_batch:
            pad_summaries_lengths.append(len(summary))

        pad_texts_lengths = []
        for text in pad_texts_batch:
            pad_texts_lengths.append(len(text))

        yield pad_summaries_batch, pad_texts_batch, pad_summaries_lengths, pad_texts_lengths
```

Hyperparameters

```
# Set the Hyperparameters
batch_size = 256
rnn_size = 256
num_layers = 2
learning_rate = 0.005
epochs = 500

# Train the Model
learning_rate_decay = 0.995
min_learning_rate = 0.0001
display_step = 20 # Check training loss after every 20 batches
per_epoch = 3 # Make 3 update checks per epoch
update_check = (len(X_train)//batch_size//per_epoch)-1
stopwatch = 0
stop_early = 0
update_loss = 0
batch_loss = 0
summary_update_loss = [] # Record the update losses for saving improvements in the model
```

dropout
[x] 0%
[ ] 20%
[ ] 35%

optimizer
[ ] RMSProp
[x] Adam

```
keep_probability = 1.0 # dropout
optimizer = tf.train.AdamOptimizer(learning_rate)  # optimizer
```

```python
# Build the graph
train_graph = tf.Graph()
# Set the graph to default to ensure that it is ready for training
with train_graph.as_default():

    # Load the model inputs
    input_data, targets, lr, keep_prob, summary_length, max_summary_length, text_length = model_inputs()

    # Create the training and inference logits
    training_logits, inference_logits = seq2seq_model(tf.reverse(input_data, [-1]),
                                                      targets,
                                                      keep_prob,
                                                      text_length,
                                                      summary_length,
                                                      max_summary_length,
                                                      len(vocab_to_int)+1,
                                                      rnn_size,
                                                      num_layers,
                                                      vocab_to_int,
                                                      batch_size)

    # Create tensors for the training logits and inference logits
    training_logits = tf.identity(training_logits.rnn_output, 'logits')
    inference_logits = tf.identity(inference_logits.sample_id, name='predictions')

    # Create the weights for sequence_loss
    masks = tf.sequence_mask(summary_length, max_summary_length, dtype=tf.float32, name='masks')

    with tf.name_scope("optimization"):
        # Loss function
        # the last layer is FC NET as vocab_size units,
        # and use activation as softmax function
        cost = tf.contrib.seq2seq.sequence_loss(
            training_logits,
            targets,
            masks)

        # Gradient Clipping
        gradients = optimizer.compute_gradients(cost)
        capped_gradients = [(tf.clip_by_value(grad, -5., 5.), var) for grad, var in gradients if grad is not None]
        train_op = optimizer.apply_gradients(capped_gradients)

print("Graph is built.")
```

Train the Model

```
checkpoint = "best_model.ckpt"
with tf.Session(graph=train_graph) as sess:
    sess.run(tf.global_variables_initializer())

    for epoch_i in range(1, epochs+1):
        print('=' * 70)
        print('Epoch {:>3}/{}'.format(epoch_i, epochs))
        print('=' * 70)

        update_loss = 0
        batch_loss = 0
        for batch_i, (summaries_batch, texts_batch, summaries_lengths, texts_lengths) in enumerate(
                get_batches(y_train, X_train, batch_size)):
            start_time = time.time()

            _, loss = sess.run([train_op, cost],
                               {input_data: texts_batch,
                                targets: summaries_batch,
                                lr: learning_rate,
                                summary_length: summaries_lengths,
                                text_length: texts_lengths,
                                keep_prob: keep_probability})

            batch_loss += loss
            update_loss += loss
            end_time = time.time()
            batch_time = end_time - start_time
            stopwatch += batch_time

            if (batch_i % display_step == 0) and (batch_i is not 0):
                print('Epoch {:>3}/{} Batch {:>4}/{} - Loss: {:>6.3f}, Seconds: {:>4.2f}'
                      .format(epoch_i,
                              epochs,
                              batch_i,
                              len(X_train) // batch_size,
                              batch_loss / display_step,
                              batch_time * display_step))
                batch_loss = 0

            if batch_i % update_check == 0:
                print("Average loss for this update:", round(update_loss/update_check,3))
                summary_update_loss.append(update_loss)

                # If the update loss is at a new minimum, save the model
                if update_loss <= min(summary_update_loss):
                    print('\nNew Record!')
                    stop_early = 0
                    saver = tf.train.Saver()
                    saver.save(sess, checkpoint)

                else:
                    stop_early += 1
                    print("No Improvement.  " + str(stop_early))
                update_loss = 0


        # Reduce learning rate, but not below its minimum value
        learning_rate *= learning_rate_decay
        if learning_rate < min_learning_rate:
            learning_rate = min_learning_rate

print('\n--- Finish ---')
print(f'Stopwatch: {stopwatch:.2f} sec.')
```

Loss Plot

```
import matplotlib.pyplot as plt
```

```
num_update_check_per_epoch = (len(X_train) // batch_size // update_check) +1  # step update check
step_add = 1 / num_update_check_per_epoch
len_data_plot = len(summary_update_loss)
len_data_x_axis = len_data_plot * step_add
data_x_axis = np.arange(step_add, len_data_x_axis+step_add, step_add)
```

```
summary_update_loss = np.array(summary_update_loss) / update_check
```

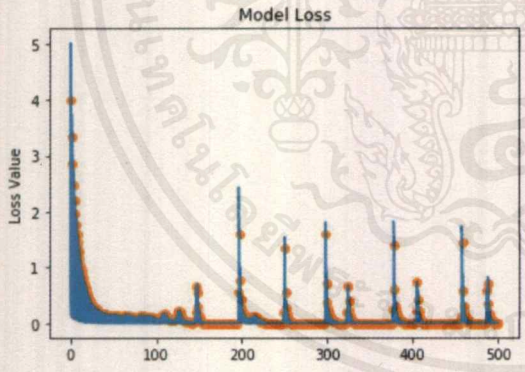
```
highlight_x = []
highlight_y = []
for i, (x, y) in enumerate(zip(data_x_axis, summary_update_loss)):
    if (i+1) % 4 == 0 and x > 0:
        highlight_x.append(x)
        highlight_y.append(y)
```

```
print('number update check every epoch: ', num_update_check_per_epoch)
print('update check: ', update_check)
```

```
number update check every epoch:  4
update check:  26
```

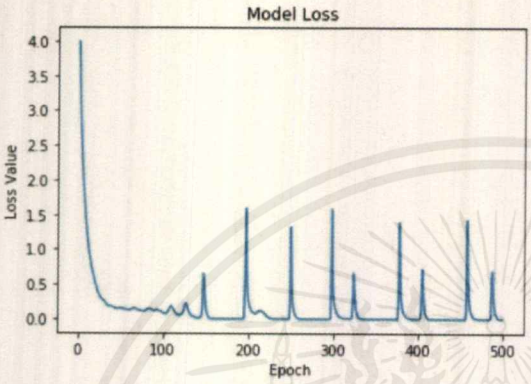
Full plot

```
# point
plt.plot(highlight_x, highlight_y, 'bo', color='C1')

# line
plt.plot(data_x_axis, summary_update_loss, 'k', color='C0')

plt.title('Model Loss')
plt.ylabel('Loss Value')
plt.xlabel('Epoch')

plt.savefig('model_loss_full.png')  # save to image
plt.show()
```



Last loss on epoch (avg)

```
loss_per_epoch = [summary_update_loss[i-1] for i in range(1,
                                  len(summary_update_loss)+1) if i % num_update_check_per_epoch is 0]
data_x_axis = list(range(1, len(loss_per_epoch)+1))

# line
plt.plot(data_x_axis, loss_per_epoch, 'k', color='C0')

plt.title('Model Loss')
plt.ylabel('Loss Value')
plt.xlabel('Epoch')

plt.savefig('model_loss_epoch.png')  # save to image
plt.show()
```




```
to_txt('summary_update_loss.txt', summary_update_loss)
```

Model Evaluate

```
def text_to_seq(text):
    '''Prepare the text for the model'''
    return [vocab_to_int.get(word, vocab_to_int['<UNK>']) for word in text]
```

```
# set generate summaries length
generate_len = find_max(y_train)

print('# Train set')
print("Text length: ", find_max(X_train))
print("Summaries length: ", generate_len)
print('-----------------------------------------------------------------------')
print('# Test set')
print("Text length: ", find_max(X_test))
print("Summaries length: ", find_max(y_test))
```

```
# Train set
Text length:  49
Summaries length:  26
------------------------------------------------------------------------
# Test set
Text length:  43
Summaries length:  22
```

```
texts = X_test
checkpoint = "./best_model.ckpt"

loaded_graph = tf.Graph()
with tf.Session(graph=loaded_graph) as sess:
    # Load saved model
    loader = tf.train.import_meta_graph(checkpoint + '.meta')
    loader.restore(sess, checkpoint)

    input_data = loaded_graph.get_tensor_by_name('input:0')
    logits = loaded_graph.get_tensor_by_name('predictions:0')
    text_length = loaded_graph.get_tensor_by_name('text_length:0')
    summary_length = loaded_graph.get_tensor_by_name('summary_length:0')
    keep_prob = loaded_graph.get_tensor_by_name('keep_prob:0')

    outputs = []
    for text in texts:
        #Multiply by batch_size to match the model's input parameters
        answer_logits = sess.run(logits,
                                 {input_data: [text]*batch_size,
                                  summary_length: [generate_len],
                                  text_length: [len(text)]*batch_size,
                                  keep_prob: keep_probability})[0]
        outputs.append(answer_logits)
```

```
print('Number of output: ', len(outputs))
```

```
Number of output:  2377
```

- **ROUGE** : metrics for Text summarization

```
import sys
sys.path.append('../../')

from evaluate import rouge

def clean_pad(lists, pad_int=vocab_to_int['<PAD>']):
    result = []
    for list_ in lists:
        without_pad = [l for l in list_ if l != pad_int]
        result.append(without_pad)
    return result
```

```
# remove padding in outputs (generate) for ROUGE
outputs_without_pad = clean_pad(outputs)

# covert to vocab
y_test_text = [[int_to_vocab[i] for i in y] for y in y_test]
outputs_text = [[int_to_vocab[i] for i in output] for output in outputs_without_pad]

# join vocab to sentent separate by '|' charactor
y_test_text = ["|".join(text) for text in y_test_text]
outputs_text = ["|".join(text) for text in outputs_text]
```

```
# compute ROUGE score
report = rouge(hypotheses=outputs_text, references=y_test_text)
print(report)
```

```
{'rouge_1/f_score': 0.22370522924266248,
 'rouge_1/r_score': 0.21714887438573277,
 'rouge_1/p_score': 0.245603127273964,
 'rouge_2/f_score': 0.07147199573550303,
 'rouge_2/r_score': 0.07299463691164472,
 'rouge_2/p_score': 0.07520423450945052,
 'rouge_l/f_score': 0.1854591933014478,
 'rouge_l/r_score': 0.20306315305196806,
 'rouge_l/p_score': 0.200269883421336}
```

```
to_json('report.json', report)
```

Summary Result

```
# covert to vocab
X_test_text = [[int_to_vocab[i] for i in x] for x in X_test]

# join vocab to sentent separate by '|' charactor, without <EOS>
X_test_text = ["|".join(text[:-1]) for text in X_test_text]
```

```
for text, summ_true, summ_gen in zip(X_test_text, y_test_text, outputs_text):
    print('Text  ---> ', text)
    print('True summary          ---> ', summ_true)
    print('Genterate summary  ---> ', summ_gen)
    print('-' * 70)
```

```
Text  ---> ผู้บัญชาการ|ตำรวจ|แห่งชาติ|ขานรับ|คำสั่ง|นายกรัฐมนตรี|ห้าม|ผู้ต้องหา|แถลงข่าว|สื่อมวลชน|ชี้|ปฏิบัติตาม|พิจารณา|ข้อบกพร่อง|วินัย|อาญา
True summary          --->  ผบ.|ตร.|รับ|ลูก|นายกฯ|ห้าม|นำ|ผู้ต้องหา|มา|นั่ง|แถลงข่าว
Genterate summary  --->  ผบ.|ตร.|สั่ง|ให้|น้ำเกลือ|ดู|คำสั่ง|คสช.|ผิดวินัย|หลัง|ถูก|ตัดสิน
----------------------------------------------------------------------
Text  ---> ทช.|เตือน|แมงกะพรุน|กล่อง|จำนวนมาก|บริเวณ|ชายหาด|เกาะ|หมาก|จ.|ตราด|นักท่องเที่ยว|สัมผัส|โดน|บริเวณ|ขา|ปฐมพยาบาล|เบื้องต้น|ชี้|แมงกะพรุน|พิษ|ปฐมพยาบาล|วิธี|เสี่ยง|ตาย|แนะ|ผู้ประกอบการ|แจ้ง|เตือน|นักท่องเที่ยว|เตรียม|น้ำส้มสายชู|ห้าม
True summary          --->  แมงกะพรุน|กล่อง|โผล่|เกาะ|หมาก|เตือน|พิษ|ถึงตาย
Genterate summary  --->  เตือน|เล่น|น้ำ|หาด|เกาะ|ก้อน|ห่วง|ชบเชา|โดน|พิษ
----------------------------------------------------------------------
Text  ---> อัคร|กิตติ์|วร|โรจน์|เจริญ|เดช|เบนซ์|เร|ซิ่ง|มารดา|หลักฐาน|การครอบครอง|รถยนต์|หรู|ทรัพย์สิน|ตำรวจ|ปราบปราม|ยาเสพติด|ตรวจสอบ
True summary          --->  เบนซ์|เร|ซิ่ง|หอบ|หลักฐาน|ครอบครอง|รถ|หรู|ให้|ตำรวจ|ปส.
Genterate summary  --->  เบนซ์|เร|ซิ่ง|ส่ง|ห้อง|ยึดทรัพย์|ทาง|ทรัพย์|ดอกไม้
```

โค้ดสร้างแบบจำลองประเภทอื่น ๆ มีความเหมือนกันกับโค้ดสร้างแบบจำลอง Adam without Dropout Model 1 ซึ่งจะมีความแตกต่างเพียงการกำหนดค่า Hyperparameter ที่อยู่ในส่วนของการทำกรอบสีแดงของโค้ดดังกล่าว

2. โค้ดสร้างแบบจำลอง Adam without Dropout Model 2 เหมือนกันกับโค้ดสร้างแบบจำลอง Adam without Dropout Model 1

3. โค้ดกำหนดค่า Hyperparameter ของ Adam Dropout 0.2

dropout
[ ] 0%
[x] 20%
[ ] 35%

optimizer
[ ] RMSProp
[x] Adam

```
keep_probability = 0.80  # dropout
optimizer = tf.train.AdamOptimizer(learning_rate)
```

4. โค้ดกำหนดค่า Hyperparameter ของ Adam Dropout 0.35

dropout
[ ] 0%
[ ] 20%
[x] 35%

optimizer
[ ] RMSProp
[x] Adam

```
keep_probability = 0.65 # dropout
optimizer = tf.train.AdamOptimizer(learning_rate)  # optimizer
```

5. โค้ดกำหนดค่า Hyperparameter ของ RMSProp without Dropout

dropout
[x] 0%
[ ] 20%
[ ] 35%

optimizer
[x] RMSProp
[ ] Adam

```
keep_probability = 1.0  # dropout
optimizer = tf.train.RMSPropOptimizer(learning_rate)
```

6. โค้ดกำหนดค่า Hyperparameter ของ RMSProp Dropout 0.2

dropout
[ ] 0%
[x] 20%
[ ] 35%

optimizer
[x] RMSProp
[ ] Adam

```
keep_probability = 0.80  # dropout
optimizer = tf.train.RMSPropOptimizer(learning_rate)
```

7. โค้ดกำหนดค่า Hyperparameter ของ RMSProp Dropout 0.35

dropout
[ ] 0%
[ ] 20%
[x] 35%

optimizer
[x] RMSProp
[ ] Adam

```
keep_probability = 0.65  # dropout
optimizer = tf.train.RMSPropOptimizer(learning_rate)
```

8. โค้ดสร้าง Dataset

```
from nltk.corpus import words as nltk_words
from nltk.corpus import stopwords
import pythainlp
from pythainlp import tokenize
from pythainlp.corpus import thai_stopwords
from num2words import num2words
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import re
import time

print('PyThaiNLP Version: {}'.format(pythainlp.__version__))
```

```
PyThaiNLP Version: 2.0.5
```

Inspecting the Data

```
dataset_path = '../dataset/thaipbs_news/scraping/news_list/thaipbs_news_dataset.csv'
language = 'thai'
tk_engine = 'newmm'
```

```
# dtype: dict
custom_contractions_dict = {
    '%': 'เปอร์เซ็นต์',
    '\xa0': ' ',
}

# dtype: set
custom_dictionary = {
    '"',
    "'",
    'ซีพี',
    'ดิวตี้',
    'อีอีซี',
    'มธ.',
    'แฮชแท็ก',
    'ส.ส.ท.',
    'อีโอดี',
    'ฟุลบูน',
    'ไอซี',
    'รีเมค',
    'แบงค์อก',
    'เอฟซี',
    'ทีซีดีซี',
    'เฟซบุค',
    'สคร.',
    'นูโร',
    'กยศ.',
}

# dtype: set
custom_stopwords = {
    '\r\n',
}
```

```
df = pd.read_csv(dataset_path)
```

```
display(df.shape)
display(df.head())
```

(29170, 2)

	highlights	story
0	ล้วงข้อมูล WhatsApp โยงฆาตกรรมนักข่าวชาอุฯ	ผู้ใช้งานแอปพลิเคชัน "วอทสแอพ" ทั่วโลกประมาณ 1...
1	สภานักศึกษา มธ.ยื่นจดหมายขอยกเลิกบังคับใส่เครื่...	สภานักศึกษามหาวิทยาลัยธรรมศาสตร์ศูนย์รังสิต ยื...
2	สรรพสามิตแจงเพิ่มภาษีดีเซล หวังหนุนใช้ปาล์มผลิ...	กรมสรรพสามิตปรับเพิ่มอัตราภาษีน้ำมันดีเซล หวัง...
3	ท่าแผนฆ่าสามีฝังดิน อ้างถูกผู้ฆ่า-ทำร้ายลูก	ผู้ต้องหาอ้างยิงสามีเสียชีวิตและนำศพไปฝังดิน ท...
4	"ไดโซเดียม" ในอาหารกึ่งสำเร็จรูปไม่ใช่ผงชูรส	แพทย์ยืนยัน ไดโซเดียม ส่วนประกอบในบะหมี่และโจ๊...

```
# Check for any nulls values
df.isnull().sum()
```

```
highlights    0
story         0
dtype: int64
```

```
from sklearn.utils import shuffle
```

```
df.columns = ['Summary', 'Text']

# Remove null values and unneeded features
df = df.drop_duplicates().dropna() # drop nan & duplicates
df = shuffle(df)  # shuffle data
df = df.reset_index(drop=True)
```

```
df.head()
```

	Summary	Text
0	จีนค้านสหรัฐฯ ขายอาวุธให้ไต้หวัน	จีนเตือนสหรัฐฯ ว่าความสัมพันธ์ทวิภาคีอาจสั่นคล...
1	"พล.ต.อ.สมยศ-โค้ชเฮง-ชิโก้" ร่วมประชุมเตรียมที...	สมาคมกีฬาฟุตบอลฯวางแผนเตรียมทีมฟุตบอลทีมชาติไท...
2	อนาคต "วัดพระธรรมกาย" หากไร้ "พระธัมมชโย"	อนาคต "วัดพระธรรมกาย" หากไม่มี "พระธัมมชโย" ศร...
3	" จิสด้า"ใช้ดาวเทียม 10 ดวงติดตามไฟป่าดั้งเป้า...	กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม จัด2 มาต...
4	เลือกตั้ง2562: ทอ.แจงปมเบรกพรรคอนาคตใหม่หาเสีย...	โฆษกกองทัพอากาศ ยืนยันไม่ได้กีดกันนายนิกม์ แสง...

Sample Data

```
# Inspecting some of the news
for i in range(5):
    print("Review #",i+1)
    print(df.Summary[i])
    print(df.Text[i])
    print()
```

Review # 1
จีนค้านสหรัฐฯ ขายอาวุธให้ไต้หวัน
จีนเดือนสหรัฐฯ ว่าความสัมพันธ์ทวิภาคีอาจสั่นคลอนหากสหรัฐฯ ขายอาวุธให้ไต้หวัน พร้อมย้ำถึงนโยบายจีนเดียวและจะไม่ยอมให้มีใครมาแยกดินแดนอย่างเด็ดขาด

Review # 2
"พล.ต.อ.สมยศ-โค้ชเฮง-ซิโก้" ร่วมประชุมเตรียมทีมลุยศึกฟุตบอลโลก รอบคัดเลือก 12 ทีมสุดท้ายโซนเอเชีย
สมาคมกีฬาฟุตบอลฯวางแผนเตรียมทีมฟุตบอลทีมชาติไทยก่อนลุยศึกฟุตบอลโลก รอบคัดเลือก 12 ทีมสุดท้ายโซนเอเชีย

Review # 3
อนาคต "วัดพระธรรมกาย" หากไร้ "พระธัมมชโย"
อนาคต "วัดพระธรรมกาย" หากไม่มี "พระธัมมชโย" ศรัทธายัง "แข็งแกร่ง" หรือ "สั่นคลอน" สำนักงานพระพุทธศาสนาแห่งชาติ เชื่อผลที่เกิดขึ้นกับพระธัมมชโยไม่กระทบกับกิจกรรมของวัด ขณะที่อดีตลูกศิษย์เชื่อสถานภาพของวัดอาจถูกสั่นคลอน โดยเฉพาะการบริหารจัดการภายในวัด

Review # 4
" จิสด้า"ใช้ดาวเทียม 10 ดวงติดตามไฟป่าตั้งเป้าลดร้อยละ 30
กระทรวงทรัพยากรธรรมชาติและสิ่งแวดล้อม จัด2 มาตรการสังคมประนามมือเผาป่าต้นเหตุหมอกควันพิษในภาคเหนือ พร้อมคาดโทษจนท.รับผิดชอบในการลดจุดความร้อนระดับพื้นที่ให้ได้ร้อยละ30 เดือนม.ค.นี้ถก 5 ชาติแก้หมอกควันข้ามแดน

Review # 5
เลือกตั้ง2562: ทอ.แจงปมเบรกพรรคอนาคตใหม่หาเสียงเขต รร.
โฆษกกองทัพอากาศ ยืนยันไม่ได้กีดกันนายนิกม์ แสงศิรินาวิน ผู้สมัคร ส.ส.กรุงเทพมหานคร พรรคอนาคตใหม่ ที่ออกมาระบุว่าถูกสารวัตรทหารกองทัพอากาศ (สห.) ไม่ให้เข้าไปหาเสียงในโรงเรียนฤทธิยะวรรณาลัย ชี้อยู่ในพื้นที่กองทัพอากาศ หากหาเสียงต้องขออนุญาต

Preparing the Data

```
import json

def to_json(fname, data):
    with open(fname, 'w') as file:
        json_data = json.dumps(data)
        file.write(json_data)

def read_json(fname):
    with open(fname, 'r') as file:
        data = file.read()
        json_data = json.loads(data)
        return json_data

def to_txt(fname, data):
    with open(fname, 'w', encoding='utf-8') as file:
        data = [str(d) for d in data]
        file.write('\n'.join(data))

def read_txt(fname):
    with open(fname, 'r', encoding='utf-8') as file:
        data = file.read()
        string_data = data.split('\n')
        try:
            float_data = [float(s) for s in string_data]
            return float_data
        except ValueError:
            return string_data
```

```python
# A list of contractions from:
# http://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python

contractions_dict = read_json('../contractions.json')
contractions_dict.update(custom_contractions_dict)


def contractions(text):
    # replace contractions with their longer forms
    for word1, word2 in contractions_dict.items():
        text = text.replace(word1, word2)
    return text
```

```python
contractions_dict.update(custom_contractions_dict)
th_dictionary = set(tokenize.DEFAULT_DICT_TRIE)  # load defeault dictionary
english_dictionary = set(nltk_words.words())
DICTIONARY = th_dictionary.union(english_dictionary).union(custom_dictionary)
TK = tokenize.Tokenizer(DICTIONARY, engine=tk_engine)
```

```python
# stopwords
english_stopwords = set(stopwords.words('english'))
thai_stopwords_ = set(thai_stopwords())
stopwords_ = english_stopwords.union(thai_stopwords_).union(custom_stopwords)
```

```python
to_txt('stopwords.txt', stopwords_)
to_txt('dictionary.txt', DICTIONARY)
to_json('contractions_dict.json', contractions_dict)
```

```python
def clean_token(tokens, stopword_dict={}, use_remove_symbol=True):
    clean = []
    for token in tokens:
            token = token.strip(' ')  # remove white space
            if token in stopword_dict:
                continue

            if use_remove_symbol:
                token = remove_symbol(token)
                if len(token) <= 1:
                    continue

            clean.append(token)
    return clean
```

```python
def build_vocab(text, unk_method='replace', use_norm_num=True, **kwargs):
    '''
    unk_method: unknow vocab method
    replace: replace know vocab to <unk>
    remove: pass vocab (remove vocab)
    ignore: can not operate
    '''

    # to lower-case
    text = text.lower()
    # replace contractions with their longer forms
    text = contractions(text)
    # word tokenize
    tokens = TK.word_tokenize(text)

    segments = []
    for token in tokens:
        if use_norm_num:
            # convert number to word
            token_list = norm_num(token, **kwargs)
        else:
            token_list = [token]
        # append to segments list
        segments += token_list
    return segments
```

```python
def norm_num(string, use_num2word=False, lang='thai'):
    "'return dtype: list"'

    ignore_string = ['infinity']

    if string in ignore_string:
        return string

    try:
        new_string = string.replace(',', '')
        num = float(new_string)
        try:
            num = int(new_string)
        except ValueError:
            pass

        if use_num2word:
            word = num2words(num, lang=lang)  # to word
            tokens = TK.word_tokenize(word)
        else:
            tokens = [str(num)]
        return tokens  # dtyep: list
    except ValueError:
        return [string]
```

```python
def remove_symbol(word):
    # Format words and remove unwanted characters
    word = re.sub(r'https?:\/\/.*[\r\n]*', '', word, flags=re.MULTILINE)
    word = re.sub(r'\<a href', '', word)
    word = re.sub(r'&amp;', '', word)
#     word = re.sub(r'[_ "|-;%()|+&=*%.,!?:#$@\[\]/]', '', word)  # with . symbol
    word = re.sub(r'[_"\-;%()|+&=*%,!?:#$@\[\]/]', '', word)  # without . symbol
    word = re.sub(r'<br />', '', word)
    word = re.sub(r'\', '', word)
    return word
```

We will remove the **stopwords from the texts** because they do not provide much use for training our model. However, we will **keep them for our summaries** so that they sound more like natural phrases.

```python
clean_texts = [build_vocab(text, lang=language) for text in df.Text]

# clean token with stopword
clean_texts = [clean_token(tokens, stopword_dict=stopwords_) for tokens in clean_texts]
print("Texts are complete.")

clean_summaries = [build_vocab(text, lang=language) for text in df.Summary]

# clean token without stopword
clean_summaries = [clean_token(tokens, stopword_dict={}) for tokens in clean_summaries]
print("Summaries are complete.")
```

```
Texts are complete.
Summaries are complete.
```

```
# Inspect the cleaned summaries and texts to ensure they have been cleaned well
for i in range(5):
    print("Build & Clean Review #", i+1)
    print(clean_summaries[i])
    print(clean_texts[i], end='\n\n')
```

```
Build & Clean Review # 1
['จีน', 'ค้าน', 'สหรัฐฯ', 'ขาย', 'อาวุธ', 'ให้', 'ไต้หวัน']
['จีน', 'เตือน', 'สหรัฐฯ', 'ความสัมพันธ์', 'ทวิภาคี', 'สั่นคลอน', 'สหรัฐฯ', 'ขาย', 'อาวุธ', 'ไต้หวัน', 'ย้ำ', 'นโยบาย', 'จีน', 'ยอมให้', 'แยก', 'ดินแดน', 'อย่างเด็ดขาด']

Build & Clean Review # 2
['พล.ต.อ.', 'สม', 'ยศ', 'โค้ช', 'เฮง', 'ซิ', 'โก่', 'ร่วม', 'ประชุม', 'เตรียม', 'ทีม', 'ลุย', 'ศึก', 'ฟุตบอล', 'โลก', 'รอบ', 'คัดเลือก', '12', 'ทีม', 'สุดท้าย', 'โซน', 'เอเชีย']
['สมาคม', 'กีฬา', 'ฟุตบอล', 'วางแผน', 'เตรียม', 'ทีม', 'ฟุตบอล', 'ทีม', 'ชาติ', 'ไทย', 'ลุย', 'ศึก', 'ฟุตบอล', 'โลก', 'รอบ', 'คัดเลือก', '12', 'ทีม', 'สุดท้าย', 'โซน', 'เอเชีย']

Build & Clean Review # 3
['อนาคต', 'วัด', 'พระ', 'ธรรมกาย', 'หาก', 'ไร้', 'พระ', 'ธัมม', 'ชโย']
['อนาคต', 'วัด', 'พระ', 'ธรรมกาย', 'ไม่มี', 'พระ', 'ธัมม', 'ชโย', 'ศรัทธา', 'แข็งแกร่ง', 'สั่นคลอน', 'สำนักงาน', 'พระพุทธศาสนา', 'แห่งชาติ', 'ที่เกิด', 'ชื่นกับ', 'พระ', 'ธัมม', 'ชโย', 'กระทบ', 'กิจกรรม', 'ของวัด', 'ลูกศิษย์', 'สถานภาพ', 'ของวัด', 'สั่นคลอน', 'โดยเฉพาะ', 'การบริหาร', 'วัด']

Build & Clean Review # 4
['จี', 'สด้า', 'ใช้', 'ดาวเทียม', '10', 'ดวง', 'ติดตาม', 'ไฟป่า', 'ตั้งเป้า', 'ลด', 'ร้อยละ', '30']
['กระทรวง', 'ทรัพยากรธรรมชาติ', 'สิ่งแวดล้อม', 'จัด', 'มาตรการ', 'สังคม', 'ประ', 'นาม', 'มือ', 'เผา', 'ป่า', 'ต้นเหตุ', 'หมอก', 'ควันพิษ', 'ภาคเหนือ', 'คาดโทษ', 'จนท.', 'รับผิดชอบ', 'ลด', 'จุด', 'ความร้อน', 'ระดับ', 'พื้นที่', 'ให้ได้', 'ร้อยละ', '30', 'เดือน', 'ม.ค.', 'ถก', 'ชาติ', 'แก้', 'หมอก', 'ควัน', 'ข้ามแดน']

Build & Clean Review # 5
['เลือกตั้ง', '2562', 'ทอ.', 'แจง', 'ปม', 'เบรก', 'พรรค', 'อนาคต', 'ใหม่', 'หาเสียง', 'เขต', 'รร.']
['โฆษก', 'กองทัพอากาศ', 'ไม่ได้', 'กีดกัน', 'นิ', 'กม', 'แสง', 'คี', 'ริ', 'นาวิน', 'ผู้สมัคร', 'ส.ส.', 'กรุงเทพมหานคร', 'พรรค', 'อนาคต', 'ออกมา', 'ระบุ', 'สารวัตรทหาร', 'กองทัพอากาศ', 'สห.', 'เข้าไป', 'หาเสียง', 'โรงเรียน', 'ฤทธิยะวรรณาลัย', 'ขี่', 'พื้นที่', 'กองทัพอากาศ', 'หาเสียง', 'ขออนุญาต']
```

```
def count_words(count_dict, text):
    '''Count the number of occurrences of each word in a set of text'''
    for sentence in text:
        for word in sentence:
            if word not in count_dict:
                count_dict[word] = 1 # new key-case
            else:
                count_dict[word] += 1
```

```
# Find the number of times each word was used and the size of the vocabulary
word_counts = {}

count_words(word_counts, clean_summaries)
count_words(word_counts, clean_texts)

print("Size of Vocabulary:", len(word_counts))
```

```
Size of Vocabulary: 25399
```

Word2Vec: PyThaiNLP

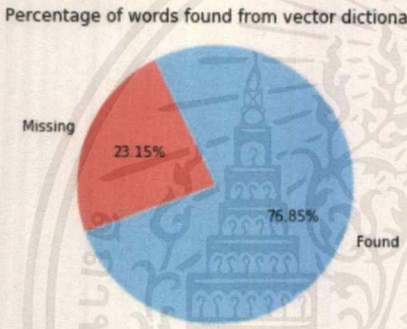
```
from pythainlp.word_vector import get_model

wv_model = get_model()
embeddings_index = {vocab: wv_model[vocab] for vocab in wv_model.vocab}  # wv_dict

print('Word embeddings:', len(embeddings_index))
```

```
Word embeddings: 51358
```

```
# Find the number of words that are missing from CN, and are used more than our threshold.
missing_words = 0
threshold = 20

for word, count in word_counts.items():
    if word not in embeddings_index:
        missing_words += 1

missing_ratio = round(missing_words/len(word_counts),4)*100

print('Word counts: ', len(word_counts))
print("Number of words found: ", len(word_counts) - missing_words)
print("Number of words missing:", missing_words)
print("Percent of words that are missing from vocabulary: {}%".format(missing_ratio))
```

```
Word counts:  25399
Number of words found:  19518
Number of words missing: 5881
Percent of words that are missing from vocabulary: 23.150000000000002%
```

```
labels = ['Found', 'Missing']
sizes = [len(word_counts) - missing_words, missing_words]
plt.title('Percentage of words found from vector dictionary')
plt.pie(sizes, labels=labels, autopct='%1.2f%%', startangle=200, colors=['lightskyblue', 'lightcoral'])
plt.axis('equal')
plt.show()
```



vocab2int, int2vocab

```
#dictionary to convert words to integers
vocab_to_int = {}

value = 0
for word, count in word_counts.items():
    if count >= threshold or word in embeddings_index:
        vocab_to_int[word] = value
        value += 1

# Special tokens that will be added to our vocab
codes = ["<UNK>","<PAD>","<EOS>","<GO>"]

# Add codes to vocab
for code in codes:
    vocab_to_int[code] = len(vocab_to_int)

# Dictionary to convert integers to words
int_to_vocab = {}
for word, value in vocab_to_int.items():
    int_to_vocab[value] = word
```

```python
# Need to use 300 for embedding dimensions
embedding_dim = 300
nb_words = len(vocab_to_int)

# Create matrix with default values of zero
word_embedding_matrix = np.zeros((nb_words, embedding_dim), dtype=np.float32)
new_word_embedding = []

for word, i in vocab_to_int.items():
    if word in embeddings_index:
        word_embedding_matrix[i] = embeddings_index[word]
    else:
        # If word not in thai word2vec vector dict, create a random embedding for it
        new_embedding = np.array(np.random.uniform(-1.0, 1.0, embedding_dim))
        embeddings_index[word] = new_embedding
        word_embedding_matrix[i] = new_embedding

        new_word_embedding.append(word)

# Check if value matches len(vocab_to_int)
print('number of word embedding matrix: ', len(word_embedding_matrix))

print('number of new word embedding: ', len(new_word_embedding))
print('new word embedding: ', new_word_embedding)
```

```
number of word embedding matrix:  19878
number of new word embedding:  360
new word embedding:  ['ท.', 'ยกครัว', 'ขุนน้ำ', 'นายกฯ', 'สหรัฐฯ', 'ใน', 'ดอนเมือง', 'ดอลลาร์สหรัฐ', 'อุดฯ', 'ผู้ถูกกล่าวหา', 'คป', 'เฟซบุ
ก', 'ผู้ว่าฯ', 'ร่วมใน', 'ซีพี', 'ผิดวินัย', 'ปชช.', 'ลุมพินี', 'สป', 'เข้าให้', 'ปี', 'โดนัลด์', 'ทรัมป์', 'เมียนมา', 'ผู้ต้องขัง', 'กรุงเทพฯ', 'อุทยานฯ', 'โ
ดรน', 'เน', 'ธัมม', 'สตร.', 'โรดแมป', 'สนง.', 'สต', 'รัชต', 'ธร', 'อินโดฯ', 'ส.', 'ลมกระโชก', 'ล้มทับ', 'เฟส', 'โรฮิงญา', 'ไบค์', 'สวล.',
'ส.ส.ท.', 'เต่ามะเฟือง', 'ลับง', 'พะเนิน', 'โปรดเกล้าฯ', 'ลอนดอน', 'บ', 'เอฟซี', 'คพ.', 'รับซื้อ', 'จุฬาฯ', 'กรมชล', 'อินทนนท์', '40000', 'สม
เด็จพระสังฆราช', 'แป๊ะเจี๊ยะ', 'สุไหง', 'จยย.', 'โตะ', 'รอมฎอน', 'ภูมิพล', 'น้ำบาดาล', 'โจชัว', 'หว่อง', 'ตุน', 'ซูเปอร์', 'เรีย', 'วิทย์', 'ทค.', 'ก
ยท.', 'ทัต', 'มาร์ก', 'ธา', 'กองปราบฯ', 'กรมอุทยานฯ', 'กรุงเทพ', 'ชญ', 'ตรละ', 'จช', 'วธ', '100000', 'ยังมี', 'ดี', 'ทีดีอาร์ไอ', '12000', 'เงิน
ทอน', 'hia', 'อร', 'ธิป', 'คอลเซ็นเตอร์', 'ตั้ง', 'บร', 'ณ', 'เบลเยียม', 'ณัฐ', 'แอดมิน', 'สระแก้ว', 'สวีเดน', 'คปภ.', 'สพฉ.', 'แอพพลิเคชั่น', 'วั
นมาฆบูชา', 'เปอร์', 'ธนา', 'เอ็นจีวี', 'ช.', 'แชมป์', '300000', 'สจ', 'แกร็บ', 'รับงาน', 'ทับซ้อน', 'สเปน', 'นญา', 'สปส.', 'ภูฏาน', 'สมยศ', 'กร
ะลอก', 'ไซเบอร์', 'รีวิว', 'เรือโดยสาร', 'อัล', 'มธ.', 'สพฐ.', 'สคบ.', 'คอมฯ', 'ถา', 'ท์', 'ทปอ.', 'ผู้ตรวจการแผ่นดิน', 'ล้อมคอก', 'สี', 'ตำรวจท
างหลวง', '60000', 'ปภ.', 'เลสเต', 'สังข', 'อี้', 'แพทยสภา', 'สุรเชษฐ์', 'กทพ.', 'พะยูง', 'เว็บ', 'gat', 'ทช.', 'ช.ภ.', 'ธนช', 'สตา', 'เจ้าท่า', 'เ
ลขาฯ', '13000', 'หลอดเลือด', 'น์', 'ทรู', 'ตู้เอทีเอ็ม', 'ณป', 'ณี', 'หนองบัวลำภู', 'ปานามา', 'แอล', 'กยศ.', 'มท.1', 'อิ', 'พงษ์', 'ชิตี้', 'อิสตัน
บูล', 'ท่ายเขื่อน', 'เฮอริเคน', 'พปช', 'เชิน', 'กรมบังคับคดี', 'สอบพยาน', 'เบ', 'อัค', 'โรล', 'ปวีณา', 'ออง ซาน ซู จี', 'บึงกาฬ', 'ทวิตเตอร์',
'อุบลฯ', 'วันเฉลิมพระชนมพรรษา', 'ผือ', 'เซลฟี่', 'เมจิก', 'โปเกมอน', 'สห', 'ไล่ล่า', 'วีระ', 'กฤษณ์', 'ย้อนศร', 'หัวคิว', 'ลาสเวกัส', 'ทร', 'ศ
ป', 'มหิดล', 'ประ', 'สก', '50000', 'ภาษีสรรพสามิต', '70000', 'สิดอ', 'ซาอุฯ', 'กัน', 'ยังอยู่', 'เอช', 'นี', 'กส', 'ทัย', 'นาปรัง', 'ป่ายหาเสียง',
'วังช้าง', 'ฟ้องกลับ', 'ต่อยอด', 'ogle', 'ไอซ์', 'สุว', 'วสท.', 'นัม', 'นค', 'เช', '80000', 'ช้างแอฟริกา', 'จันทน์หอม', 'ธงชาติไทย', '35000', 'ร์
ช', 'กส', 'พายุไซโคลน', 'เทพย', 'วัฒน', '15000', 'ออกโฉนด', 'ซูจิ', 'อดุลยเดช', 'สุนัขจรจัด', 'หมายค้น', 'พรากผู้เยาว์', 'ให้ความเป็นธรร
ม', 'ธี', 'โบ๊ท', 'ตม', 'สุร', 'รัตน', 'สต็อก', 'พัฒน', 'เอกวาดอร์', 'สตรีท', 'แอพพลิเคชัน', 'อัพ', 'โอบามา', 'สทนช.', '14.0', 'ปิดอ่าว', '2000
00', 'แจ็ค', 'จตุพร', 'วีร', 'เนสตี้', 'แคนดิเดต', 'โทษทางวินัย', 'กระสอบทราย', 'พญาไท', 'อีอีซี', 'ลันดา', 'โบอิ้ง', 'ที่วัด', 'ปูติน', 'ไทรโยค',
'สถานการณ์ฉุกเฉิน', 'แคมป์', 'ซัมซุง', 'ดัดประเด็น', 'ยะไข่', 'ธิ', 'ดีเบต', 'บรัสเซลส์', 'เกีย', 'คลื่นความถี่', 'คืนให้', 'กรมการจัดหางาน', 'วัน
สงกรานต์', 'ไลโก้', '22.0', 'พาร์', 'ซาอุ', 'อาระเบีย', 'ราค', 'ชั่น', 'ซัปโปโร', 'สายไฟฟ้า', 'นาคะ', 'เป็นราย', 'นอมินี', 'ผวจ.', 'ผู้ประกันตน',
'ชินวัตร', 'ธัญบุรี', 'เสด็จสวรรคต', 'วางมาตรการ', 'ตัญ', 'จุฬาลงกรณ์', 'hail', 'อินสตาแกรม', 'น้ัน', 'บารัค', 'ให้ได้', 'ป่าสงวนแห่งชาติ', 'ออ
กของ', 'ถึงชีวิต', 'สท', '500000', 'คช', 'โรยตัว', 'วัชร', 'ไม้หวงห้าม', '18.0', 'เก่', 'ธ์', 'ส่วนราชการ', 'ปากท่อ', 'ท่าเป็น', 'ยอดน้ำ', 'ล้านน
า', 'ของตลาด', 'รับของโจร', '600000', 'กระทรวงเกษตรฯ', 'ให้น้ำ', 'ละมุง', 'ศีลอด', '400000', 'ลำห้วย', 'เปรม ติณสูลานนท์', 'การออกเสีย
งประชามติ', 'อยู่แล้ว', 'ผู้ดำรงตำแหน่งทางการเมือง', '\r\r', 'พระราชบัญญัติประกอบรัฐธรรมนูญ', 'เพีย', 'ทฤษฎี สหวงษ์', 'สมเด็จพระเท
พรัตนราชสุดาฯ สยามบรมราชกุมารี', 'พราหมณ', 'นายสุเทพ เทือกสุบรรณ', '17.0', 'ว่าไม่ได้', 'ตลอด 24 ชั่วโมง', 'ต่าง ๆ', 'กิตติ์', '15.0',
'โดยที่', '16.0', 'นายอภิสิทธิ์ เวชชาชีวะ', '<UNK>', '<PAD>', '<EOS>', '<GO>']
```

```python
to_json('vocab_to_int.json', vocab_to_int)
to_json('int_to_vocab.json', int_to_vocab)
np.save('word_embedding_matrix.npy', word_embedding_matrix) # save
```

```python
def convert_to_ints(text, word_count, unk_count, eos=False):
    '''Convert words in text to an integer.
       If word is not in vocab_to_int, use UNK's integer.
       Total the number of words and UNKs.
       Add EOS token to the end of texts'''
    ints = []
    for sentence in text:
        sentence_ints = []
        for word in sentence:

            word_count += 1
            if word in vocab_to_int:
                sentence_ints.append(vocab_to_int[word])
            else:
                sentence_ints.append(vocab_to_int["<UNK>"])
                unk_count += 1
        if eos:
            sentence_ints.append(vocab_to_int["<EOS>"])
        ints.append(sentence_ints)
    return ints, word_count, unk_count
```

```
# Apply convert_to_ints to clean_summaries and clean_texts
word_count = 0
unk_count = 0

int_summaries, word_count, unk_count = convert_to_ints(clean_summaries, word_count, unk_count)
int_texts, word_count, unk_count = convert_to_ints(clean_texts, word_count, unk_count, eos=True)
```

```
def create_lengths(text):
    '''Create a data frame of the sentence lengths from a text'''
    lengths = []
    for sentence in text:
        lengths.append(len(sentence))
    return pd.DataFrame(lengths, columns=['counts'])
```

```
lengths_summaries = create_lengths(int_summaries)
lengths_texts = create_lengths(int_texts)

print("Summaries:")
print(lengths_summaries.describe())
print()
print("Texts:")
print(lengths_texts.describe())
```

```
Summaries:
             counts
count  29170.000000
mean      10.748166
std        2.980412
min        3.000000
25%        9.000000
50%       10.000000
75%       13.000000
max       26.000000

Texts:
             counts
count  29170.000000
mean      23.781796
std        6.073380
min        5.000000
25%       19.000000
50%       23.000000
75%       28.000000
max       51.000000
```

```
# Inspect the length of texts
print(np.percentile(lengths_texts.counts, 90))
print(np.percentile(lengths_texts.counts, 95))
print(np.percentile(lengths_texts.counts, 99))
```

```
32.0
34.0
39.0
```

```
# Inspect the length of summaries
print(np.percentile(lengths_summaries.counts, 90))
print(np.percentile(lengths_summaries.counts, 95))
print(np.percentile(lengths_summaries.counts, 99))
```

```
15.0
16.0
19.0
```

```
def unk_counter(sentence):
    '''Counts the number of time UNK appears in a sentence.'''
    unk_count = 0
    for word in sentence:
        if word == vocab_to_int["<UNK>"]:
            unk_count += 1
    return unk_count
```

```
# Sort the summaries and texts by the length of the texts, shortest to longest
# Limit the length of summaries and texts based on the min and max ranges.
# Remove news that include too many UNKs

max_text_length = 1000
max_summary_length = 100
min_length = 2

unk_text_limit = 1
unk_summary_limit = 0

y = []  # summaries dataset
X = []  # textsy dataset

for length in range(min(lengths_texts.counts), max_text_length):
    for count, words in enumerate(int_summaries):
        if (len(int_summaries[count]) >= min_length and
            len(int_summaries[count]) <= max_summary_length and
            len(int_texts[count]) >= min_length and
            unk_counter(int_summaries[count]) <= unk_summary_limit and
            unk_counter(int_texts[count]) <= unk_text_limit and
            length == len(int_texts[count])
           ):
            y.append(int_summaries[count])
            X.append(int_texts[count])

# Compare lengths to ensure they match
print('y size: ', len(y))
print('X size: ', len(X))
```

```
y size:  23770
X size:  23770
```

Train-Test Split

train 90 %
test 10 %

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1)

print('Dataset size: ', len(X))
print('Train size:', len(X_train))
print('Test size:', len(X_test))
```

```
Dataset size:  23770
Train size: 21393
Test size: 2377
```

```
def find_max(data):
    return max([len(d) for d in data])

def find_min(data):
    return min([len(d) for d in data])
```

```
# data sory by X but without y sorted, maual y sort
print('# Dataset')
print("The shortest text length: ", find_min(X))
print("The longest text length: ", find_max(X))
print()
print("The shortest summary length: ", find_min(y))
print("The longest summary length : ", find_max(y))

print('--------------------------------------------------------------------------------')
print('# Train set')
print("The shortest text length: ", find_min(X_train))
print("The longest text length: ", find_max(X_train))
print()
print("The shortest summary length: ", find_min(y_train))
print("The longest summary length: ", find_max(y_train))

print('--------------------------------------------------------------------------------')
print('# Test set')
print("The shortest text length: ", find_min(X_test))
print("The longest text length: ", find_max(X_test))
print()
print("The shortest summary length: ", find_min(y_test))
print("The longest summary length: ", find_max(y_test))
```

```
# Dataset
The shortest text length:  5
The longest text length:  49

The shortest summary length:  3
The longest summary length :  26
--------------------------------------------------------------------------------
# Train set
The shortest text length:  5
The longest text length:  49

The shortest summary length:  3
The longest summary length:  26
--------------------------------------------------------------------------------
# Test set
The shortest text length:  5
The longest text length:  45

The shortest summary length:  3
The longest summary length:  22
```

Save Dataset

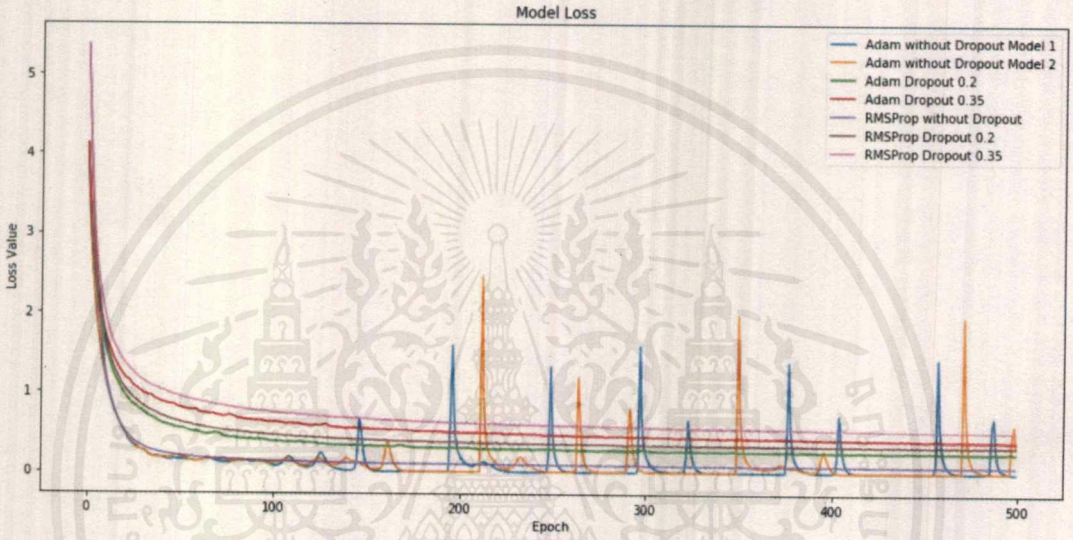
```
dataset = {
    'data': X, 'target': y,
    'X_train': X_train , 'X_test': X_test,
    'y_train': y_train, 'y_test': y_test
}

to_json('dataset.json', dataset)
```

9. โค้ดสร้างรายงาน

```python
from matplotlib import pyplot as plt
import json

# config
batch_size = 256
update_check = 26
num_update_check_per_epoch = 4


def read_txt(fname):
    with open(fname, 'r', encoding='utf-8') as file:
        data = file.read()
        string_data = data.split('\n')
        try:
            float_data = [float(s) for s in string_data]
            return float_data
        except ValueError:
            return string_data


def read_json(fname):
    with open(fname, 'r') as file:
        data = file.read()
        json_data = json.loads(data)
        return json_data


def get_data(loss, limit=500):
    return [loss[i-1] for i in range(1, len(loss)+1) if i % num_update_check_per_epoch is 0][:limit]
```

```python
model_dict = {
    'Adam without Dropout Model 1': 'Adam/0_1',
    'Adam without Dropout Model 2': 'Adam/0_2',
    'Adam Dropout 0.2': 'Adam/20',
    'Adam Dropout 0.35': 'Adam/35',
    'RMSProp without Dropout': 'RMSProp/0',
    'RMSProp Dropout 0.2': 'RMSProp/20',
    'RMSProp Dropout 0.35': 'RMSProp/35',
}
```

```python
# load loss value
plot_dict = {}
for model_name, path in model_dict.items():
    loss = read_txt(f'{path}/summary_update_loss.txt')
    plot_dict[model_name] = get_data(loss)


# plot
plt.figure(figsize=(15, 7))

_ = [plt.plot(loss) for loss in plot_dict.values()]

plt.title('Model Loss')
plt.ylabel('Loss Value')
plt.xlabel('Epoch')

plt.legend(plot_dict.keys())
plt.savefig('model_loss.png')  # save to image
plt.show()
```




```python
import pandas as pd
# load report value
df = pd.DataFrame()
for model_name, path in model_dict.items():
    report = read_json(f'{path}/report.json')
    report = pd.DataFrame([report])
    df = df.append(report)

df.index = model_dict
```

```python
df.round(4).to_csv('report.csv')
```

10. โค้ดการนำแบบจำลองไปใช้งาน

```
import pythainlp
from pythainlp import tokenize
from num2words import num2words
import numpy as np
import tensorflow as tf
import re
import time
import json
```

```
print('PyThaiNLP Version: {}'.format(pythainlp.__version__))
print('TensorFlow Version: {}'.format(tf.__version__))
```

```
PyThaiNLP Version: 2.0.5
TensorFlow Version: 1.13.1
```

```
def read_json(fname, key_int=False):
    with open(fname, 'r') as file:
        data = file.read()
        json_data = json.loads(data)

        if not key_int:
            return json_data

        json_data = {int(key): value for key, value in json_data.items()}
        return json_data

```

```
def read_txt(fname):
    with open(fname, 'r', encoding='utf-8') as file:
        data = file.read()
        string_data = data.split('\n')
        try:
            float_data = [float(s) for s in string_data]
            return float_data
        except ValueError:
            return string_data
```

```
def text_to_seq(text):
    '''Prepare the text for the model'''
    return [vocab_to_int.get(word, vocab_to_int['<UNK>']) for word in text]
```

```
def contractions(text):
    # replace contractions with their longer forms
    for word1, word2 in contractions_dict.items():
        text = text.replace(word1, word2)
    return text
```

```
def clean_token(tokens, stopword_dict={}, use_remove_symbol=True):
    clean = []
    for token in tokens:
            token = token.strip(' ')  # remove white space
            if token in stopword_dict:
                continue

            if use_remove_symbol:
                token = remove_symbol(token)
                if len(token) <= 1:
                    continue

            clean.append(token)
    return clean
```

```python
def build_vocab(text, unk_method='replace', use_norm_num=True, **kwargs):
    '''
    unk_method: unknow vocab method
    replace: replace know vocab to <unk>
    remove: pass vocab (remove vocab)
    ignore: can not operate
    '''

    # to lower-case
    text = text.lower()
    # replace contractions with their longer forms
    text = contractions(text)
    # word tokenize
    tokens = TK.word_tokenize(text)

    segments = []
    for token in tokens:
        if use_norm_num:
            # convert number to word
            token_list = norm_num(token, **kwargs)
        else:
            token_list = [token]
        # append to segments list
        segments += token_list
    return segments
```

```python
def norm_num(string, use_num2word=False, lang='thai'):
    '''return dtype: list'''

    ignore_string = ['infinity']

    if string in ignore_string:
        return string

    try:
        new_string = string.replace(',', '')
        num = float(new_string)
        try:
            num = int(new_string)
        except ValueError:
            pass

        if use_num2word:
            word = num2words(num, lang=lang)  # to word
            tokens = TK.word_tokenize(word)
        else:
            tokens = [str(num)]
        return tokens  # dtyep: list
    except ValueError:
        return [string]
```

```python
def remove_symbol(word):
    # Format words and remove unwanted characters
    word = re.sub(r'https?:\/\/.*[\r\n]*', '', word, flags=re.MULTILINE)
    word = re.sub(r'\<a href', '', word)
    word = re.sub(r'&amp;', '', word)
#     word = re.sub(r'[_"\-;%()|+&=*%.,!?:#$@\[\]/]', '', word)  # with . symbol
    word = re.sub(r'[_"\-;%()|+&=*%,!?:#$@\[\]/]', '', word)  # without . symbol
    word = re.sub(r'<br />', '', word)
    word = re.sub(r'\'', '', word)
    return word
```

```python
tk_engine = 'newmm'
batch_size = 256
```

Load Value

```
vocab_to_int = read_json('vocab_to_int.json')
int_to_vocab = read_json('int_to_vocab.json', key_int=True)
word_embedding_matrix = np.load('word_embedding_matrix.npy') # load

stopwords_ = set(read_txt('stopwords.txt'))
DICTIONARY = set(read_txt('dictionary.txt'))
contractions_dict = read_json('contractions_dict.json')

TK = tokenize.Tokenizer(DICTIONARY, engine=tk_engine)
```

```
print('number of word output space: ', len(int_to_vocab))
```

```
number of word output space:  19878
```

```
def textsumm(input_sentence, checkpoint):
    build_input_sentence = [build_vocab(text) for text in input_sentence]
    clean_input_sentence = [clean_token(tokens, stopword_dict={}) for tokens in build_input_sentence]
    texts = [text_to_seq(text) for text in clean_input_sentence] # text for input

    loaded_graph = tf.Graph()
    with tf.Session(graph=loaded_graph) as sess:
        # Load saved model
        loader = tf.train.import_meta_graph(checkpoint + '.meta')
        loader.restore(sess, checkpoint)

        input_data = loaded_graph.get_tensor_by_name('input:0')
        logits = loaded_graph.get_tensor_by_name('predictions:0')
        text_length = loaded_graph.get_tensor_by_name('text_length:0')
        summary_length = loaded_graph.get_tensor_by_name('summary_length:0')
        keep_prob = loaded_graph.get_tensor_by_name('keep_prob:0')

        outputs = []
        for text in texts:
            #Multiply by batch_size to match the model's input parameters
            answer_logits = sess.run(logits, {input_data: [text]*batch_size,
                                              summary_length: [26],
                                              text_length: [len(text)]*batch_size,
                                              keep_prob: 1.0})[0]
            outputs.append(answer_logits)

    # Remove the padding
    pad = vocab_to_int["<PAD>"]
    summs = [[int_to_vocab[i] for i in output if i != pad] for output in outputs]

    return summs
```

APP

```
input_sentence = [
    'สวนดุสิตโพลเผยประชาชนรู้สึกเบื่อหน่ายการจัดตั้งรัฐบาล โดยมองว่าเกิดจากผลประโยชน์ทางการเมืองไม่ลงตัว-แย่งชิงเก้าอี้รัฐมน
    'เจ้าหน้าที่ทำอากาศยานสุวรรณภูมิ ควบคุมตัวชาวจีน 2 คน พร้อมของกลาง เป็นทรัพย์สินที่ขโมยจากผู้โดยสารต่างชาติระหว่างเดิน
    'พล.อ.สุรยุทธ์ จุลานนท์ รักษาการประธานองคมนตรี เป็นผู้แทนพระองค์ ไปในการบำเพ็ญพระราชกุศลทักษิณานุปทาน 7 วัน พระรา
    'คปภ.แจ้งสิทธิค่ารักษาพยาบาลและค่าสินไหมทดแทนตาม พ.ร.บ. ให้กับผู้บาดเจ็บและผู้เสียชีวิตแล้ว กรณีรถเบนซ์พุ่งชน บริเวณถ
    'เกิดเหตุผู้ก่อเหตุ 2 คนขี่จักรยานยนต์ ปาระเบิดไม่ทราบชนิดใส่ป้อมตำรวจที่ อ.บันนังสตา จ.ยะลา วัยรุ่นชาย อายุ 17 ปี โดนลูกหลง
]
```

```
checkpoints = {
    'Adam without Dropout Model 1': 'Adam/0_1',
    'Adam without Dropout Model 2': 'Adam/0_2',
    'Adam Dropout 0.2': 'Adam/20',
    'Adam Dropout 0.35': 'Adam/35',
    'RMSProp without Dropout': 'RMSProp/0',
    'RMSProp Dropout 0.2': 'RMSProp/20',
    'RMSProp Dropout 0.35': 'RMSProp/35',
}
```

```
for model_name, checkpoint_path in checkpoints.items():
    checkpoint_path = f'{checkpoint_path}/best_model.ckpt'
    summs = textsumm(input_sentence, checkpoint=checkpoint_path)

    print(model_name)
    print('='* 70)

    for text_input, text_output in zip(input_sentence, summs):
        print(text_input)
        print(f'  ---> {"|".join(text_output)}')
        print('-' * 70)
    print()
```

```
INFO:tensorflow:Restoring parameters from Adam/0_1/best_model.ckpt
Adam without Dropout Model 1
======================================================================
สวนดุสิตโพลเผยประชาชนรู้สึกเบื่อหน่ายการจัดตั้งรัฐบาล โดยมองว่าเกิดจากผลประโยชน์ทางการเมืองไม่ลงตัว-แย่งชิงเก้าอี้รัฐมนตรี จึง
อยากให้ยึดผลประโยชน์ของประเทศชาติเป็นหลัก ขณะที่นิด้าโพลเผยสำรวจ 10 สาเหตุที่ทำให้พรรคประชาธิปัตย์ "แพ้เลือกตั้ง-แตกแย
ก"
  ---> โพล|นิด้า|ช่วย|ร้อง|รัฐบาล|แก้ปัญหา|อดีต|นายทหาร|เกษียณ
----------------------------------------------------------------------
เจ้าหน้าที่ทำอากาศยานสุวรรณภูมิ ควบคุมตัวชาวจีน 2 คน พร้อมของกลาง เป็นทรัพย์สินที่ขโมยจากผู้โดยสารต่างชาติระหว่างเดินทางเข้
าไทย เจ้าหน้าที่เตรียมเสนอเพิกถอนการอนุญาตให้อยู่ต่อในราชอาณาจักร พร้อมทั้งดำเนินคดี หลังเห็นว่ามีพฤติกรรมเป็นภัยสังคม
  ---> คุมตัว|ชาวจีน|ใน|สุวรรณภูมิ|เร่ง|รื้อ|ด้า|วัน|แรก
----------------------------------------------------------------------
```

11. โค้ด ROUGE metric

```
"""ROUGe metric implementation.
This is a modified and slightly extended verison of
https://github.com/google/seq2seq.
"""

import itertools
import numpy as np


def _get_ngrams(n, text):
  """Calcualtes n-grams.
  Args:
    n: which n-grams to calculate
    text: An array of tokens
  Returns:
    A set of n-grams
  """
  ngram_set = set()
  text_length = len(text)
  max_index_ngram_start = text_length - n
  for i in range(max_index_ngram_start + 1):
    ngram_set.add(tuple(text[i:i + n]))
  return ngram_set


def _split_into_words(sentences):
  """Splits multiple sentences into words and flattens the result"""
  return list(itertools.chain(*[_.split("|") for _ in sentences]))


def _get_word_ngrams(n, sentences):
  """Calculates word n-grams for multiple sentences.
  """
  assert len(sentences) > 0
  assert n > 0

  words = _split_into_words(sentences)
  return _get_ngrams(n, words)
def _len_lcs(x, y):
  """
  Returns the length of the Longest Common Subsequence between sequences x
  and y.
  Source: http://www.algorithmist.com/index.php/Longest_Common_Subsequence
  Args:
    x: sequence of words
    y: sequence of words
  Returns
    integer: Length of LCS between x and y
  """
  table = _lcs(x, y)
  n, m = len(x), len(y)
  return table[n, m]
```

```
def _lcs(x, y):
    """
    Computes the length of the longest common subsequence (lcs) between two
    strings. The implementation below uses a DP programming algorithm and runs
    in O(nm) time where n = len(x) and m = len(y).
    Source: http://www.algorithmist.com/index.php/Longest_Common_Subsequence
    Args:
      x: collection of words
      y: collection of words
    Returns:
      Table of dictionary of coord and len lcs
    """
    n, m = len(x), len(y)
    table = dict()
    for i in range(n + 1):
      for j in range(m + 1):
        if i == 0 or j == 0:
          table[i, j] = 0
        elif x[i - 1] == y[j - 1]:
          table[i, j] = table[i - 1, j - 1] + 1
        else:
          table[i, j] = max(table[i - 1, j], table[i, j - 1])
    return table

def _recon_lcs(x, y):
    """
    Returns the Longest Subsequence between x and y.
    Source: http://www.algorithmist.com/index.php/Longest_Common_Subsequence
    Args:
      x: sequence of words
      y: sequence of words
    Returns:
      sequence: LCS of x and y
    """
    i, j = len(x), len(y)
    table = _lcs(x, y)

    def _recon(i, j):
      """private recon calculation"""
      if i == 0 or j == 0:
        return []
      elif x[i - 1] == y[j - 1]:
        return _recon(i - 1, j - 1) + [(x[i - 1], i)]
      elif table[i - 1, j] > table[i, j - 1]:
        return _recon(i - 1, j)
      else:
        return _recon(i, j - 1)

    recon_tuple = tuple(map(lambda x: x[0], _recon(i, j)))
    return recon_tuple
```

```
def rouge_n(evaluated_sentences, reference_sentences, n=2):
  """
  Computes ROUGE-N of two text collections of sentences.
  Sourece: http://research.microsoft.com/en-us/um/people/cyl/download/
  papers/rouge-working-note-v1.3.1.pdf
  Args:
    evaluated_sentences: The sentences that have been picked by the summarizer
    reference_sentences: The sentences from the referene set
    n: Size of ngram.  Defaults to 2.
  Returns:
    A tuple (f1, precision, recall) for ROUGE-N
  Raises:
    ValueError: raises exception if a param has len <= 0
  """
  if len(evaluated_sentences) <= 0 or len(reference_sentences) <= 0:
    raise ValueError("Collections must contain at least 1 sentence.")

  evaluated_ngrams = _get_word_ngrams(n, evaluated_sentences)
  reference_ngrams = _get_word_ngrams(n, reference_sentences)
  reference_count = len(reference_ngrams)
  evaluated_count = len(evaluated_ngrams)

  # Gets the overlapping ngrams between evaluated and reference
  overlapping_ngrams = evaluated_ngrams.intersection(reference_ngrams)
  overlapping_count = len(overlapping_ngrams)

  # Handle edge case. This isn't mathematically correct, but it's good enough
  if evaluated_count == 0:
    precision = 0.0
  else:
    precision = overlapping_count / evaluated_count

  if reference_count == 0:
    recall = 0.0
  else:
    recall = overlapping_count / reference_count

  f1_score = 2.0 * ((precision * recall) / (precision + recall + 1e-8))

  # return overlapping_count / reference_count
  return f1_score, precision, recall
def _f_p_r_lcs(llcs, m, n):
  """
  Computes the LCS-based F-measure score
  Source: http://research.microsoft.com/en-us/um/people/cyl/download/papers/
  rouge-working-note-v1.3.1.pdf
  Args:
    llcs: Length of LCS
    m: number of words in reference summary
    n: number of words in candidate summary
  Returns:
    Float. LCS-based F-measure score
  """
  r_lcs = llcs / m
  p_lcs = llcs / n
  beta = p_lcs / (r_lcs + 1e-12)
  num = (1 + (beta**2)) * r_lcs * p_lcs
  denom = r_lcs + ((beta**2) * p_lcs)
  f_lcs = num / (denom + 1e-12)
  return f_lcs, p_lcs, r_lcs
```

```
def rouge_l_sentence_level(evaluated_sentences, reference_sentences):
  """
  Computes ROUGE-L (sentence level) of two text collections of sentences.
  http://research.microsoft.com/en-us/um/people/cyl/download/papers/
  rouge-working-note-v1.3.1.pdf
  Calculated according to:
  R_lcs = LCS(X,Y)/m
  P_lcs = LCS(X,Y)/n
  F_lcs = ((1 + beta^2)*R_lcs*P_lcs) / (R_lcs + (beta^2) * P_lcs)
  where:
  X = reference summary
  Y = Candidate summary
  m = length of reference summary
  n = length of candidate summary
  Args:
    evaluated_sentences: The sentences that have been picked by the summarizer
    reference_sentences: The sentences from the referene set
  Returns:
    A float: F_lcs
  Raises:
    ValueError: raises exception if a param has len <= 0
  """
  if len(evaluated_sentences) <= 0 or len(reference_sentences) <= 0:
    raise ValueError("Collections must contain at least 1 sentence.")
  reference_words = _split_into_words(reference_sentences)
  evaluated_words = _split_into_words(evaluated_sentences)
  m = len(reference_words)
  n = len(evaluated_words)
  lcs = _len_lcs(evaluated_words, reference_words)
  return _f_p_r_lcs(lcs, m, n)

def _union_lcs(evaluated_sentences, reference_sentence):
  """
  Returns LCS_u(r_i, C) which is the LCS score of the union longest common
  subsequence between reference sentence ri and candidate summary C. For example
  if r_i= w1 w2 w3 w4 w5, and C contains two sentences: c1 = w1 w2 w6 w7 w8 and
  c2 = w1 w3 w8 w9 w5, then the longest common subsequence of r_i and c1 is
  “w1 w2” and the longest common subsequence of r_i and c2 is “w1 w3 w5”. The
  union longest common subsequence of r_i, c1, and c2 is “w1 w2 w3 w5” and
  LCS_u(r_i, C) = 4/5.
  Args:
    evaluated_sentences: The sentences that have been picked by the summarizer
    reference_sentence: One of the sentences in the reference summaries
  Returns:
    float: LCS_u(r_i, C)
  ValueError:
    Raises exception if a param has len <= 0
  """
  if len(evaluated_sentences) <= 0:
    raise ValueError("Collections must contain at least 1 sentence.")

  lcs_union = set()
  reference_words = _split_into_words([reference_sentence])
  combined_lcs_length = 0
  for eval_s in evaluated_sentences:
    evaluated_words = _split_into_words([eval_s])
    lcs = set(_recon_lcs(reference_words, evaluated_words))
    combined_lcs_length += len(lcs)
    lcs_union = lcs_union.union(lcs)

  union_lcs_count = len(lcs_union)
  union_lcs_value = union_lcs_count / combined_lcs_length
  return union_lcs_value
```

```
def rouge_l_summary_level(evaluated_sentences, reference_sentences):
  """
  Computes ROUGE-L (summary level) of two text collections of sentences.
  http://research.microsoft.com/en-us/um/people/cyl/download/papers/
  rouge-working-note-v1.3.1.pdf
  Calculated according to:
  R_lcs = SUM(1, u)[LCS<union>(r_i,C)]/m
  P_lcs = SUM(1, u)[LCS<union>(r_i,C)]/n
  F_lcs = ((1 + beta^2)*R_lcs*P_lcs) / (R_lcs + (beta^2) * P_lcs)
  where:
  SUM(i,u) = SUM from i through u
  u = number of sentences in reference summary
  C = Candidate summary made up of v sentences
  m = number of words in reference summary
  n = number of words in candidate summary
  Args:
    evaluated_sentences: The sentences that have been picked by the summarizer
    reference_sentence: One of the sentences in the reference summaries
  Returns:
    A float: F_lcs
  Raises:
    ValueError: raises exception if a param has len <= 0
  """
  if len(evaluated_sentences) <= 0 or len(reference_sentences) <= 0:
    raise ValueError("Collections must contain at least 1 sentence.")

  # total number of words in reference sentences
  m = len(_split_into_words(reference_sentences))

  # total number of words in evaluated sentences
  n = len(_split_into_words(evaluated_sentences))

  union_lcs_sum_across_all_references = 0
  for ref_s in reference_sentences:
    union_lcs_sum_across_all_references += _union_lcs(evaluated_sentences,
                                                      ref_s)
  return _f_p_r_lcs(union_lcs_sum_across_all_references, m, n)
```

```python
def rouge(hypotheses, references):
    """Calculates average rouge scores for a list of hypotheses and
    references"""

    # Filter out hyps that are of 0 length
    # hyps_and_refs = zip(hypotheses, references)
    # hyps_and_refs = [_ for _ in hyps_and_refs if len(_[0]) > 0]
    # hypotheses, references = zip(*hyps_and_refs)

    # Calculate ROUGE-1 F1, precision, recall scores
    rouge_1 = [
        rouge_n([hyp], [ref], 1) for hyp, ref in zip(hypotheses, references)
    ]
    rouge_1_f, rouge_1_p, rouge_1_r = map(np.mean, zip(*rouge_1))

    # Calculate ROUGE-2 F1, precision, recall scores
    rouge_2 = [
        rouge_n([hyp], [ref], 2) for hyp, ref in zip(hypotheses, references)
    ]
    rouge_2_f, rouge_2_p, rouge_2_r = map(np.mean, zip(*rouge_2))

    # Calculate ROUGE-L F1, precision, recall scores
    rouge_l = [
        rouge_l_sentence_level([hyp], [ref])
        for hyp, ref in zip(hypotheses, references)
    ]
    rouge_l_f, rouge_l_p, rouge_l_r = map(np.mean, zip(*rouge_l))

    return {
        "rouge_1/f_score": rouge_1_f,
        "rouge_1/r_score": rouge_1_r,
        "rouge_1/p_score": rouge_1_p,
        "rouge_2/f_score": rouge_2_f,
        "rouge_2/r_score": rouge_2_r,
        "rouge_2/p_score": rouge_2_p,
        "rouge_l/f_score": rouge_l_f,
        "rouge_l/r_score": rouge_l_r,
        "rouge_l/p_score": rouge_l_p,
    }
```