การพัฒนาระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ

Developing a Knowledge Search System with Natural Language Processing from Web Scraping

ธีรพงศ์ หารยงค์

โครงงานคอมพิวเตอร์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยราชภัฏสกลนคร
พ.ศ. 2565

การพัฒนาระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ

ธีรพงศ์ หารยงค์
รหัส 62102105125

โครงงานคอมพิวเตอร์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยราชภัฏสกลนคร
พ.ศ. 2565

DEVELOPING A KNOWLEDGE SEARCH SYSTEM WITH NATURAL LANGUAGE PROCESSING FROM WEB SCRAPING

TEERAPONG HANYONG

A COMPUTER PROJECT
SUBMITED IN PARTIAL FULFILLMENT OF
THE REQUIREMENT FOR THE DEGREE OF
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

SAKOON NAKHON RAJABHAT UNIVERSITY
2022

ใบรับรองโครงงานคอมพิวเตอร์
มหาวิทยาลัยราชภัฏสกลนคร
ปริญญาวิทยาศาสตรบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์

.....

ชื่อเรื่อง การพัฒนาระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ

ชื่อนักศึกษา นายธีรพงศ์ หารยงค์ **รหัส** 62102105125

อาจารย์ที่ปรึกษา ดร.ชัยนันท์ สงพงษ์

..... อาจารย์ที่ปรึกษา
(ดร.ชัยนันท์ สงพงษ์)

.....
(ดร.ชัยนันท์ สมพงษ์)
ประธานสาขาวิชาคอมพิวเตอร์

วันที่ 24 เดือน สิงหาคม พ.ศ. 2565
ลิขสิทธิ์ของสาขาวิชาคอมพิวเตอร์
คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสกลนคร

ชื่อเรื่อง การพัฒนาระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ

นักศึกษา นาย ธีรพงศ์ หารยงค์ **รหัสนักศึกษา** 62102105125

ปริญญา วิทยาศาสตรบัณฑิต

สาขาวิชา วิทยาการคอมพิวเตอร์

พ.ศ. 2565

อาจารย์ที่ปรึกษา ดร.ชัยนันท์ สมพงษ์

บทคัดย่อ

การพัฒนาระบบระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ มีวัตถุประสงค์ดังนี้ 1) เพื่อพัฒนาระบบสำหรับตอบคำถามภาษาไทย 2) เพื่อเปรียบเทียบประสิทธิภาพของโมเดล Transformers

โดยตัวของโมเดลที่เลือกมาชื่อว่า XLMRoBERTa ซึ่งใช้ชุดข้อมูลที่ชื่อว่า thai_squad ซึ่งมีข้อมูลในการฝึกฝนโมเดลมากกว่า 4000 ข้อความถามตอบ และข้อมูลทดสอบ 74 ข้อความถามตอบ โดยการฝึกฝนนั้น จะมีการเตรียมข้อมูลตั้งแต่ การตัด Tag html, ตัดช่องว่าง, และ ตัดเป็น Sequence โดยมีความยาวของ Sequence แตกต่างกันไปตั้งแต่ 100, 200, 400, 600, 1200 เพื่อที่จะทดสอบดูว่า เมื่อความยาวของ Sequence แตกต่างกันจะทำให้ประสิทธิภาพต่างกันอย่างไร โดยจะใช้การฝึกฝนทั้งหมดความยาวละ 10 รอบ

ผลการศึกษาพบว่า 1) ระบบสำหรับตอบคำถามภาษาไทย สามารถพัฒนาทั้งระบบเสิร์ช และระบบตอบคำถาม ได้อย่าง อัตโนมัติโดยที่ผู้ใช้งานแค่พูดคำถามที่ตนเองต้องการตรงๆได้เลย ระบบจะทำการค้นหาข้อมูลและหาคำตอบให้ได้ในทันที โดยจะมีการอ้างอิงข้อมูลที่ดึงมาคือจากที่ Thai wiki 2) ประสิทธิภาพของโมเดล Transformers จากผลลัพธ์การทดสอบจาก ชุดข้อมูลฝึกฝน พบว่ามีผลลัพธ์ที่ดีที่สุดคือความยาว 600 เนื่องจากมีอัตราความผิดพลาดจากการตอบคำถามผิดน้อยที่สุดเพียง 15.92% และมีอัตราความถูกต้องของการตอบคำถามแบบถูกต้องทุกตัวอักษรยู่ที่ 43.17% และตอบคำถามได้ใกล้เคียงคำตอบที่ถูกต้องได้ที่ 40.9%

คำสำคัญ: การเรียนรู้เชิงลึก (Deep Learning), การถามตอบอัตโนมัติ (Question-Answering), การเสิร์ชข้อมูล(Search), BERT, Transformers

กิตติกรรมประกาศ

ในการจัดทำโครงงานคอมพิวเตอร์ในครั้งนี้ สำเร็จลุล่วงไปได้ โดยได้รับความอนุเคราะห์จากอาจารย์ ดร.ชัยนันท์ สมพงษ์ ซึ่งเป็นอาจารย์ที่ปรึกษาที่คอยให้คำแนะนำ คำปรึกษาในข้อสงสัยต่างๆ ทางผู้จัดทำได้นำข้อเสนอแนะมาใช้ในการจัดทำโครงงานในครั้งนี้เป็นอย่างดีจนโครงงานสำเร็จตามวัตถุประสงค์

ขอกราบขอบพระคุณ บิดาและมารดา ที่ให้โอกาสทางการศึกษาแก่ลูก ที่ให้กำลังใจตลอดมาทำให้มีกำลังใจที่จะศึกษาจนสำเร็จ ซึ่งหากไม่มีครอบครัวที่อบอุ่นการที่จะกระทำการอย่างอื่นก็คงไม่ประสบผลสำเร็จเช่นนี้

สุดท้ายการจัดทำโครงงานคอมพิวเตอร์ในครั้งนี้จะประสบความสำเร็จไม่ได้ถ้าขาดบุคคลดังที่กล่าวมาแล้วนั้น คอยช่วยเหลือและให้กำลังใจ ความดีของเอกสารเล่มนี้ขอมอบให้แด่ พ่อ แม่ ผองเพื่อน และอาจารย์ทุกท่านที่มีส่วนช่วยในโอกาสในครั้งนี้

ธีรพงศ์ หารยงค์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

เนื่องจากในโลกยุคปัจจุบัน เป็นยุคแห่งเทคโนโลยีข้อมูลข่าวสาร มีข้อมูลมากมายหลากหลายบนโลกอินเทอร์เน็ต การเข้าถึงข้อมูลหรือข่าวสารต่างๆ นั้นสามารถทำได้อย่างรวดเร็ว ทำให้โลกยุคปัจจุบันเป็นยุคแห่งความเร็วของการสื่อสาร ข้อมูลข่าวสารรวมไปถึงองค์ความรู้ต่างๆ ไม่ว่าจะเกิดขึ้นจากมุมใดของโลก ก็สามารถที่จะแพร่กระจายมาถึงตัวเราได้ในเวลาไม่นาน แต่ตัวข้อมูลต่างๆ ที่แพร่กระจายอยู่ทั่วไปตามเว็บไซต์หรืออินเทอร์เน็ตนั้น บางทีมันก็อาจไม่ใช่ข้อมูลที่เราต้องการจริงๆเสียทีเดียว เราอาจจะอยากรู้แค่บางเรื่องที่เราอยากรู้จริงๆ ไม่ได้อยากจะรู้รายละเอียดที่ยืดยาวมากมาย อีกทั้งบางครั้ง การหาข้อมูลทางอินเทอร์เน็ตนั้นก็อาจใช้เวลานาน และยังต้องอ่านทำความเข้าใจเพื่อจะหาคำตอบกับสิ่งที่อยากรู้ซึ่งอาจเป็นแค่ส่วนเล็กๆ ในเนื้อหาเหล่านั้น

จึงได้มีแนวคิดในการดึงข้อมูลต่างๆ จากเว็บไซต์เหล่านั้นออกมา โดยใช้ความรู้ทางด้านการทำ Web Scraping ซึ่งก็คือวิธีการในการดึงข้อมูลจากหน้าเว็บเพจหรือเว็บไซต์ โดยใช้ภาษาโปรแกรมมิ่งเป็นเครื่องมือ (ในที่นี้คือ Python) ในการเขียนสคริปต์ในการดึงข้อมูลจากหน้าเว็บไซต์นั้นๆ ซึ่งเมื่อทำการดึงข้อมูลเสร็จแล้ว ก็จะมีขั้นตอนในการสกัด (Extract) เอาเฉพาะข้อมูลที่ต้องการเพื่อนำไปใช้งานต่อไปได้อย่างหลากหลาย แต่ตัวของข้อมูลเองก็ยังคงมีความยืดยาวจนเกินความจำเป็น จึงได้มีการพัฒนาโมเดล Machine learning มาใช้สำหรับการสกัดคำตอบจากข้อมูลต่างๆ ที่อาจจะมีมากเกินความจำเป็น ให้อยู่ในรูปแบบกระชับตรงตามคำถามที่ต้องการ ทำให้ไม่ต้องเสียเวลาอ่านข้อความที่ยืดยาวจนมากเกินไป สามารถใช้โมเดลที่เรียนรู้มาไว้ล่วงหน้าสำหรับการตอบคำถามภาษาไทย จากชุดข้อมูลหรือประโยค ได้อย่างมีความแม่นยำ

จากปัญหาดังกล่าว จึงได้มีการพัฒนาโมเดลในตระกูล Transformers ซึ่งเป็นโมเดลตระกูลที่ทำงานทางด้าน NLP ได้ดีที่สุดในปัจจุบัน ซึ่งโมเดลที่ได้คัดเลือกมาใช้งานนั้น คือ โมเดล XLM-RoBERTa ซึ่งมีความสามารถในการทำงานแบบ Sequence2Sequence ได้อย่างแม่นยำ โดยใช้หลักการ Encoder ที่ถูกออกแบบมาให้เหมาะแก่การนำมาทำโมเดลสำหรับ Question Answering เป็นอย่างมาก โดยนำมาพัฒนาควบคู่กับการใช้เทคโนโลยี SpeechRecognition กับ TextToSpeech เพื่อให้ผู้ใช้สามารถพูดสิ่งที่ต้องการถาม และรอฟังเสียงตอบกลับจากคำตอบที่สกัดมาโดยโมเดลได้เลย ทำให้สามารถนำไปประยุกต์สร้างเป็นแชทบอท หรือ โปรแกรมสนทนาอัตโนมัติ สำหรับการตอบคำถามที่เป็นภาษาไทย ได้อย่างรวบรัดและตรงประเด็น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาระบบสำหรับตอบคำถามภาษาไทย

1.2.2 เพื่อเปรียบเทียบประสิทธิภาพของ โมเดล Transformers

1.3 ขอบเขตโครงงาน

1.3.1 ด้านข้อมูล

ด้านข้อมูลในการฝึกฝนโมเดล จะใช้ Dataset “thaiqa_squad” ซึ่ง ประกอบไปด้วยคู่ประโยคภาษาไทย พร้อมคู่คำถามและคำตอบ ใน 1 แถว ซึ่งมีข้อมูลทั้งหมด 4,074 แถว โดยประกอบไปด้วย ข้อมูลในส่วนฝึกฝน 4,000 แถว และข้อมูลสำหรับการทดสอบ 74 แถว

ด้านข้อมูล สำหรับนำมาประยุกต์ใช้งานกับระบบ จะเป็นข้อมูลจาก Thai Wiki ซึ่งได้มาจากการดึงข้อมูลออกมาจากหน้าเว็บ ด้วยเทคนิค Web Scraping โดยใช้ BeautifulSoup และ Request ซึ่งเป็นไลบรารี่ ในภาษา Python ที่ช่วยให้สามารถสกัดข้อมูลที่ต้องการออกมาจากหน้าเว็บไซต์ได้ อีกทั้งยังมีการเรียกใช้งาน gtts และ SpeechRecognitin เพื่อแปลงคำพูดเป็นข้อความหรือข้อความกลับมาเป็นเสียงพูดได้แบบอัตโนมัติ

1.3.2 ด้านเทคนิค

โมเดลที่จะใช้ในการฝึกฝนเพื่อวัดประสิทธิภาพการทำงาน มีดังนี้

1.3.2.1 XLMRoBERTa

1.4 แผนการดำเนินงาน

1.4.1 กำหนดหัวข้อและนำเสนอหัวข้อ

1.4.2 ค้นหาปัญหา โอกาสและเป้าหมาย

1.4.3 ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง

1.4.4 เสนอเค้าโครงโครงงาน

1.4.5 ศึกษาและวิเคราะห์ข้อมูล

1.4.6 ทำความเข้าใจข้อมูลและเตรียมข้อมูล

1.4.7 ดำเนินการพัฒนาโมเดล

1.4.8 ประเมินประสิทธิภาพการพัฒนาโมเดล

1.4.9 จัดทำเอกสารประกอบโครงงาน

1.4.10 นำเสนอโครงงานจบ

1.4.11 รายงานด้วยเล่มสมบูรณ์

ตารางที่ 1.1 ระยะเวลาการดำเนินงาน

กิจกรรม	ระยะเวลาในการดำเนินงาน (พ.ศ 2565)								
	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.
1. กำหนดหัวข้อและนำเสนอหัวข้อ									
2. ค้นหาปัญหา โอกาสและเป้าหมาย									
3. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง									
4. เสนอเค้าโครงโครงงาน									
5. ศึกษาและวิเคราะห์ข้อมูล									
6. ทำความเข้าใจข้อมูลและเตรียมข้อมูล									
7. ดำเนินการพัฒนาโมเดล									
8. ประเมินประสิทธิภาพการพัฒนาโมเดล									
9. จัดทำเอกสารประกอบโครงงาน									
10. นำเสนอโครงงานจบ									
11. รายงานด้วยเล่มสมบูรณ์									

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้เปรียบเทียบ ประสิทธิภาพการทำงานของโมเดล Transformers XLMRoBERTa สำหรับการนำมาใช้งานกับภาษาไทย อีกทั้งยังสามารถนำโมเดลที่ได้ มาประยุกต์สร้างเป็น Prototype สำหรับการพัฒนาเพื่อใช้งานต่อไปในอนาคตได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้ผู้วิจัยได้นำเสนอเนื้อหาที่เน้นถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง รวมถึงเอกสารและงานเขียนอื่นๆ ที่เกี่ยวข้องกับงานวิจัยโดยในบทนี้จะแบ่งเนื้อหาหลักๆ ออกเป็น 8 หัวข้อ ประกอบด้วย

2.1 การเรียนรู้เชิงลึก (Deep Learning)

2.2 โมเดล Transformer

2.3 การคำนวณ Multihead attention

2.4 โมเดล BERT

2.5 โมเดล RoBERTa

2.6 โมเดล XLMRoBERTa

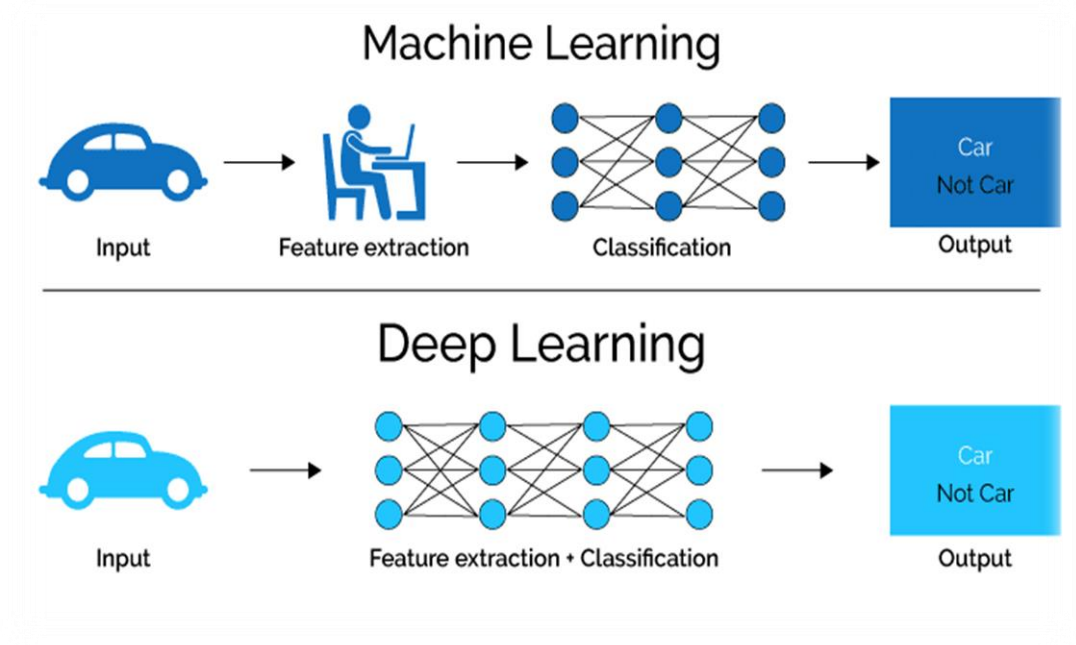
2.7 ภาษาและเครื่องมือที่ใช้

2.8 งานวิจัยที่เกี่ยวข้อง

2.1 การเรียนรู้เชิงลึก (Deep Learning)

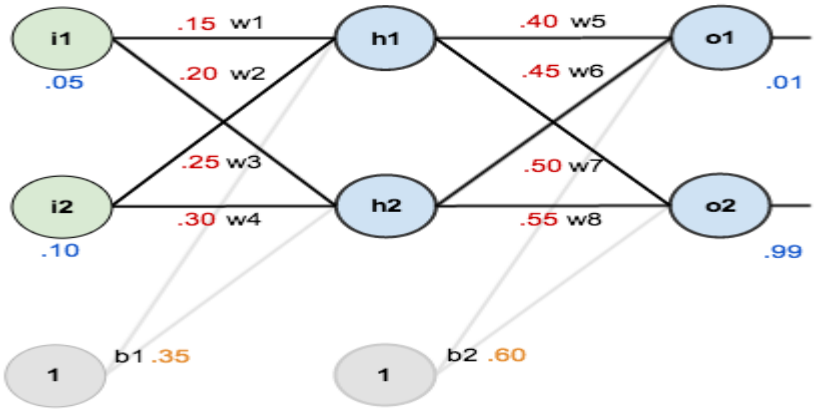
การเรียนรู้เชิงลึกเป็นส่วนหนึ่งของวิธีการการเรียนรู้ของเครื่องบนพื้นฐานของโครงข่ายประสาทเทียมและการเรียนเชิงคุณลักษณะ การเรียนรู้สามารถเป็นได้ทั้งแบบการเรียนรู้แบบมีผู้สอน การเรียนรู้แบบกึ่งมีผู้สอน และการเรียนรู้แบบไม่มีผู้สอน คำว่า "ลึก" ในความหมายมาจากการที่มีชั้นของโครงข่ายประสาทเทียมหลายชั้น โดยพื้นฐานของการเรียนรู้เชิงลึกคือ อัลกอริทึมที่พยายามจะสร้างแบบจำลองเพื่อแทนความหมายของข้อมูลในระดับสูงโดยการสร้างสถาปัตยกรรมข้อมูลขึ้นมาที่ประกอบไปด้วยโครงสร้างย่อยๆ หลายอัน และแต่ละอันนั้นได้มาจากการแปลงที่ไม่เป็นเชิงเส้น การเรียนรู้เชิงลึกอาจมองได้ว่าเป็นวิธีการหนึ่งของการเรียนรู้ของเครื่องที่พยายามเรียนรู้วิธีการแทนข้อมูลอย่างมีประสิทธิภาพ ตัวอย่างเช่น รูปภาพภาพหนึ่ง สามารถแทนได้เป็นเวกเตอร์ของความสว่างต่อจุด Pixel หรือมองในระดับสูงขึ้นเป็นเซตของขอบของวัตถุต่างๆ หรือมองว่าเป็นพื้นที่ของรูปร่างใดๆ ก็ได้ การแทนความหมายดังกล่าวจะทำให้การเรียนรู้ที่จะทำงานต่างๆ ทำได้ง่ายขึ้น ไม่ว่าจะเป็นการรู้จำใบหน้าหรือการรู้จำการแสดงออกทางสีหน้า การเรียนรู้เชิงลึกถือว่าเป็นวิธีการที่มีศักยภาพสูงในการจัดการกับฟีเจอร์สำหรับการเรียนรู้แบบไม่มีผู้สอนหรือการเรียนรู้แบบกึ่งมีผู้สอน ในปัจจุบันนักวิจัยในสาขานี้พยายามจะหาวิธีการที่ดีขึ้นในการแทนข้อมูลแล้วสร้างแบบจำลองเพื่อเรียนรู้จากตัวแทนของข้อมูลเหล่านี้ในระดับใหญ่ บางวิธีการก็ได้แรงบันดาลใจมาจากสาขาประสาทวิทยาขั้นสูง โดยเฉพาะเรื่องกระบวนการตีความหมายในกระบวนการประมวลผลข้อมูลในสมอง ตัวอย่างของกระบวนการที่การเรียนรู้เชิงลึกนำไปใช้ ได้แก่ การเข้ารหัสประสาท อันเป็นกระบวนการหาความสัมพันธ์ระหว่างตัวกระตุ้นกับการตอบสนองของเซลล์ประสาทในสมอง นักวิจัยด้านการเรียนรู้ของเครื่องได้เสนอสถาปัตยกรรมการเรียนรู้หลายแบบบนหลักการของการเรียนรู้เชิงลึกนี้ ได้แก่ โครงข่ายประสาทเทียมแบบลึก (Deep Artificial Neural Networks) โครงข่ายประสาทเทียมแบบสังวัตนาการ (Convolutional Neural Networks) โครงข่ายความเชื่อแบบลึก (Deep Belief Networks) และโครงข่ายประสาทเทียมแบบวนซ้ำ (Recurrent Neural Network) ซึ่งมีการนำมาใช้งานอย่างแพร่หลายในทางคอมพิวเตอร์วิทัศน์ การรู้จำเสียงพูด การประมวลผลภาษาธรรมชาติ การรู้จำเสียง และชีวสารสนเทศศาสตร์ (IBM Cloud Education, 2020: Online)

ซึ่งข้อดีของการทำ Deep Learning ก็คือการที่เราสามารถโยนข้อมูลจำนวนมหาศาลเข้าไปในโมเดลให้ โมเดลสามารถที่จะเรียนรู้ความสัมพันธ์ในเชิงลึกของข้อมูลเหล่านั้น ได้เลยในทันที ซึ่งต่างจากการทำ Machine Learning ในอดีต ที่ต้องมีการทำในส่วนของ Feature Extraction เสียก่อน



ภาพที่ 2.1 ความแตกต่างของ Deep Learning กับ Machine Learning
ที่มา: Nuchanat Rongroang (2020:Online)

ซึ่งอัลกอริทึ่มที่ทำงานอยู่เบื้องหลังการเรียนรู้เชิงลึกก็คือ Backpropagation ซึ่งเป็น Algorithm สำหรับการหาค่าที่เหมาะสมที่สุดสำหรับ Weight และ Bias ในชั้นของ DPNN ซึ่งต้องอาศัยค่าการเปลี่ยนแปลงของตัวแปรในการมาคำนวณ Backpropagation เริ่มจากเราต้องการหา ค่า Error ที่คำนวนได้จาก Output ของ Neural Network นำมาเปรียบเทียบกับ Target ที่เราคาดหวังไว้ เมื่อได้ค่า Error ก็จะ แพร่ค่า Error ที่ได้กลับไปยังผู้มีส่วนเกี่ยวข้อง ในที่นี้คือ ค่า Weight ต่างๆ Weight ใดให้ค่าน้ำหนักมา ก็จะได้รับผลกระทบของการปรับไปมาก Weight ใดให้ค่าน้อยก็จะได้รับผลกระทบ ไปน้อยๆ เช่นกัน เหมือนๆ หาคนรับผิดชอบที่ทำให้ค่า Output มันไม่เท่ากับ Target



ภาพที่ 2.2 ลักษณะของ Neural Network

ที่มา: Pisit Bee (2018 : Online)

ดังนี้จะเห็นว่า เรามี input อยู่ 2 ค่า คือ 0.5 และ 0.1 จาก i1 และ i2 กำหนดค่า Weight w1,w2,...w8 เป็นค่าดังรูปนะครับ ส่วน output เราตั้ง target ไว้ที่ output 1 เป็น 0.01 ส่วน output 2 เป็น 0.99 ทีนี้เราลองมาคำนวนค่า output จริงๆ ที่ได้จาก neural net กันครับ เริ่มจาก ค่า output ที่ได้จาก h1 ก่อน ค่าที่ได้จาก

h1 = i1*w1 + i2*w2 + b1 = 0.05*0.15 + 0.1*0.2 + 0.35 = 0.3775

netout_h1 = sigmoid(h1) = sigmoid(0.3775) = 1/(1+e^(-0.3775)) =0.59326

คำนวน h2

h2 = i1*w3 + i2*w4 + b1 = 0.05*0.25+0.1*0.3+0.35=0.3925

netout_h2 = sigmoid(0.3925) = 0.59688

ทำการคำนวน แบบเดิมครับ เพื่อหา O1 และ O2 ขอข้ามตัวอย่างการคำนวนไปนะครับ จาก netout_h1 จะกลายเป็นค่า input ของชั้นถัดไปแทน ทำลักษณะคล้ายๆ เดิมครับ อย่าไปเคลียดค่อยๆ อ่าน จาก netout ทั้ง 2

netout_h1= 0.59326

netout_h2= 0.59688

O1 = h1*w5 + h2*w6 + b2= 0.59326*0.4 + 0.59688*0.45 +0.6= 1.1059

netout_O1 = sigmoid(1.1059) = 1/(1+e^(-1.1059)) = 0.75136

O2 = h1*w7 + h2*w8 + b2 = 0.59326*0.5 + 0.59688*0.55 +0.6 = 1.2249

netout_O2 = sigmoid(1.2428) = 1/(1+e^(-1.2249)) = 0.7729

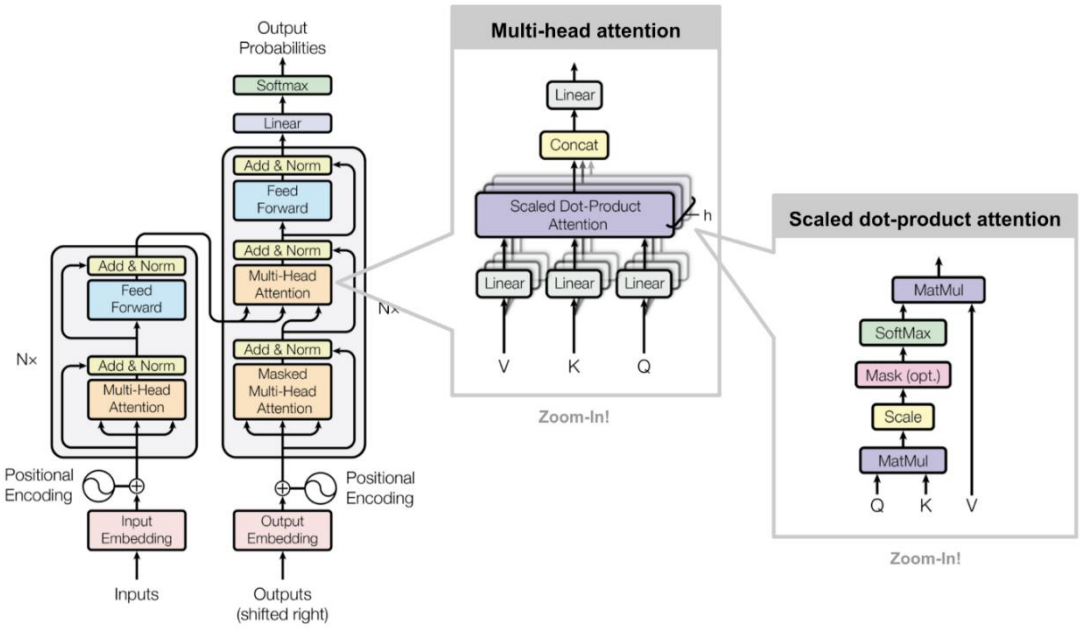
เมื่อได้ output ของแต่ละ กิ่งแล้วคือ o1 และ o2 เรามาหา Error กันครับ ซึ่ง ค่า Error หาได้จากที่ปลาย ของ O1 และ O2 ครับ สังเกตุว่า O1 เราได้ 0.75136 ซึ่งเราอยากได้ 0.01 ส่วน O2 เราอยากได้ 0.99 แต่คำนวนได้ 0.77292 ดังนั้น หา Error ของทั้งระบบ

$E\_total = ((0.75136 - 0.01)^2 + (0.77292–0.99)^2)/2$

เมื่อได้ค่า Error ทั้งหมดมาแล้ว ก็นำไปคำนวณตาม algorithm สำหรับหาค่าที่เหมาะสมที่สุดหรือ gradient descent ซึ่งจะใช้ค่า Error เหล่านั้นมาช่วยในการคำนวณเพื่อที่จะหาค่าที่เหมาะสมที่สุดในการปรับ weight และ bias ในชั้น Neural Network ต่อไป (Pisit Bee, 2018 : Online)

2.2 โมเดล Transformer

Transformer เป็นโมเดลประเภท Sequence to sequence ที่มีการทำงานแบบ Encoder-Decoder และใช้การคำนวณแบบ Multihead-Attention



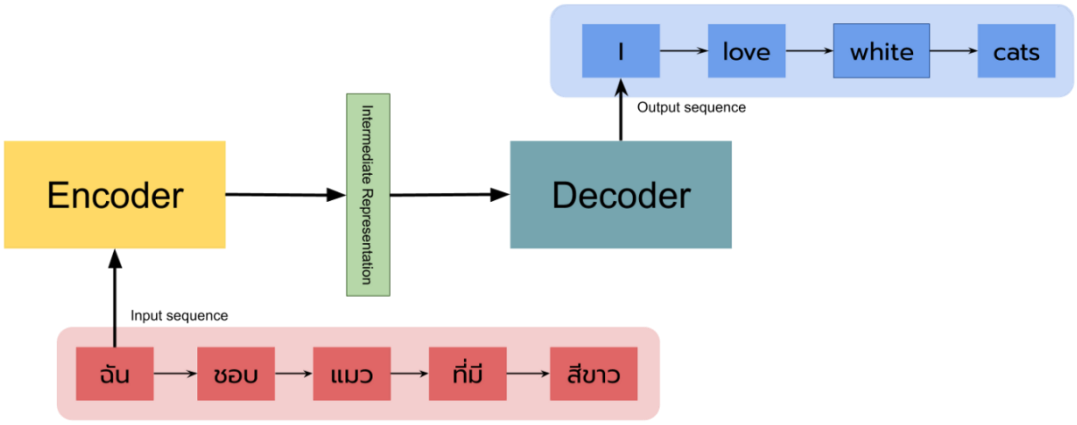
ภาพที่ 2.3 แผนภาพโครงสร้าง Transformers
ที่มา: Pakawat Nakwijit (2020:Online)

Transformer ออกแบบมาเพื่อแก้ปัญหาอันนึง ที่ชื่อว่า Machine Translation หรือ กระบวนการแปลภาษาโดยใช้คอมพิวเตอร์ เป็นปัญหาระดับ Classic ทางด้าน NLP ซึ่งสามารถสืบย้อนได้ไปถึงช่วงสงครามเย็น โดยใจความสำคัญของ Machine translation คือ การเปลี่ยนจากประโยคอะไรสักอย่างในภาษาหนึ่งไปเป็นอีกประโยคที่อยู่คนละภาษา แต่ยังคงความหมายเดิม ปัญหาสำคัญของการแปลภาษา คือ มันเป็นทั้งศาสตร์และศิลป์ การแปลที่ดีต้องมีองค์ประกอบทั้งความถูกต้อง (Correctness) และ ความครบถ้วน (Comprehensiveness) ทั้งในด้านอารมณ์และข้อมูล ซึ่งถือเป็นงานที่ยาก แม้ว่าจะใช้มนุษย์เป็นคนแปลก็ตาม นอกจากนี้ การแปลหลายๆอย่าง ยังจำเป็นจะต้องอาศัย Common sense หรือความเข้าใจธรรมชาติของโลกอีกด้วย (Pakawat Nakwijit, 2020:Online)

Machine Translation model ชิ้นแรกๆ เริ่มต้นจากการใช้ความน่าจะเป็น (Probabilistic model) เมื่อนำมาใช้จริงๆ แต่มักพบว่า มันไม่สามารถจับใจความสัมพันธ์ที่ซับซ้อนๆ ได้อย่างถูกต้อง เนื่องด้วยว่าโมเดลรูปแบบนี้ มักออกแบบให้จัดการคำในประโยค (Word) แยกออก

จากกัน หรือ อยู่ในรูปแบบ Bag of Words ทำให้ข้อมูลความสัมพันธ์ระหว่างคำหายไป หรือ อาจจะออกแบบโมเดลภายใต้เงื่อนไขบางอย่างซึ่งไม่สามารถใช้ได้จริงในเชิงปฏิบัติ

ต่อมา ความสนใจจึงโดนเบนมาที่ Neural Network ซึ่งสามารถจัดการความสัมพันธ์ที่ซับซ้อนได้ดีมากกว่า โดยหนึ่งในเทคนิคที่ใช้กันอย่างแพร่หลายในปัจจุบัน (2020) ไม่ว่าจะในเชิงวิชาการ หรือ เชิงธุรกิจ ในปัญหา Machine Translation คือ Sequence-to-sequence model หรือ Encoder-decoder model โดยมองว่า การแปลภาษาคือ การแปลง sequence หนึ่ง ไปเป็นอีกหนึ่ง Sequence (sequence-to-sequence tasks) โดยโมเดลประเภทนี้ประกอบด้วย 2 โมเดล คือ Encoder ทำหน้าที่ แปลงประโยคเริ่มต้น ไปเป็น สิ่งที่เรียกว่า intermediate Representation ซึ่งเข้ารหัส(Encode) ข้อมูลทั้งหมดในประโยค ทั้งในเชิงโครงสร้าง (Grammar) และเชิงความหมาย (Semantic) ลงในกลุ่มตัวเลขอันนึง Decoder ทำหน้าที่ แปลงIintermediate representation ที่ได้รับมาจากrแล้วแปลงกลับไปเป็นประโยค (Pakawat Nakwijit, 2020:Online)



ภาพที่ 2.4 แผนภาพโครงสร้าง Encoder Decoder

ที่มา: Pakawat Nakwijit (2020:Online)

2.3 การคำนวณ Multihead Attention

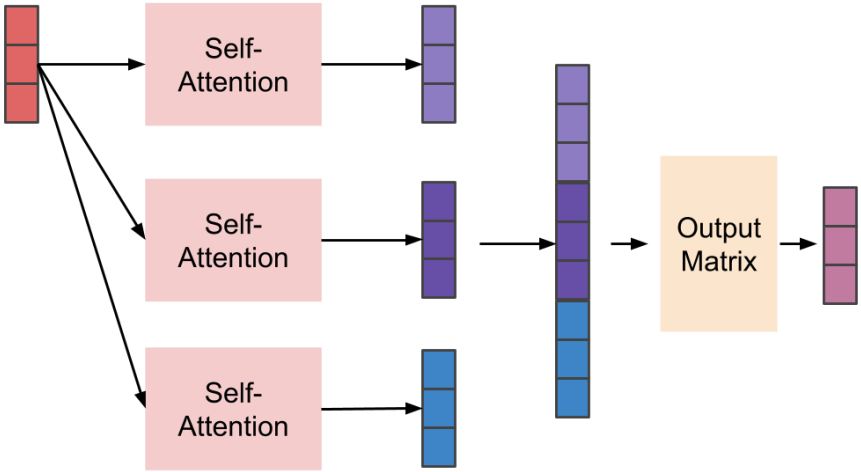
Transformer ได้เสนอเทคนิคที่เรียกว่า Self-Attention โดยไอเดีย คือ ในการแปลคำแต่ละคำ โมเดลจะเลือกสนใจเฉพาะคำที่เกี่ยวข้อง (Related context) และมีความสำคัญ (important information) ที่จำเป็นต้องใช้ในเพื่อแก้ปัญหา (ในที่นี้คือ Machine Translation) แทนที่จะต้องสนใจโครงสร้างทั้งประโยค ซึ่งประกอบไปด้วยคำที่ไม่เกี่ยวข้อง เช่น

- เธอชอบแมวเพราะว่าพวกมันนุ่มนิ่มและน่ารัก
- She loves cats because they are fluffy and cute.

สังเกตว่า การจะแปล คำว่า “ชอบ” เป็น คำว่า “loves” จำเป็นต้องรู้ว่า ชอบ ความหมายเหมือนกัน love ในภาษาอังกฤษ love ใช้คู่กับ she จะต้องเปลี่ยนเป็น loves (verb-noun agreement) แต่สิ่งที่ตามหลัง “loves” จะเป็นอะไรก็ได้ อาจจะเป็น หมา, กระต่าย หรือ แม้กระทั้ง สปาเก็ตตี้ ก็ไม่ได้ส่งผลต่อ คำว่า “loves” การทำงานแบบนี้ คล้ายๆกับการทำงานของสมอง เพื่อการแยกสิ่งของที่เรากำลังมองอยู่ สิ่งที่เกิดขึ้นคือ เราไม่จำเป็นต้องดูทุกองค์ประกอบในรูปเพื่อตัดสินใจว่า สิ่งนี้คือ แมว หรือไม่? เพียงแค่ข้อมูลบางจุดที่สำคัญก็เพียงพอในการตีความของเราแล้ว เพิ่มเติมจาก Self-Attention เนื่องจากว่า คำหนึ่งในประโยค ไม่ได้มีแค่ความสัมพันธ์เดียว เช่น

- Bobby takes his cats to the park because they love climbing trees.
- บ๊อบบี้พาแมวของเขาไปสวนสาธารณะ เพราะว่า พวกมันชอบปีนต้นไม้

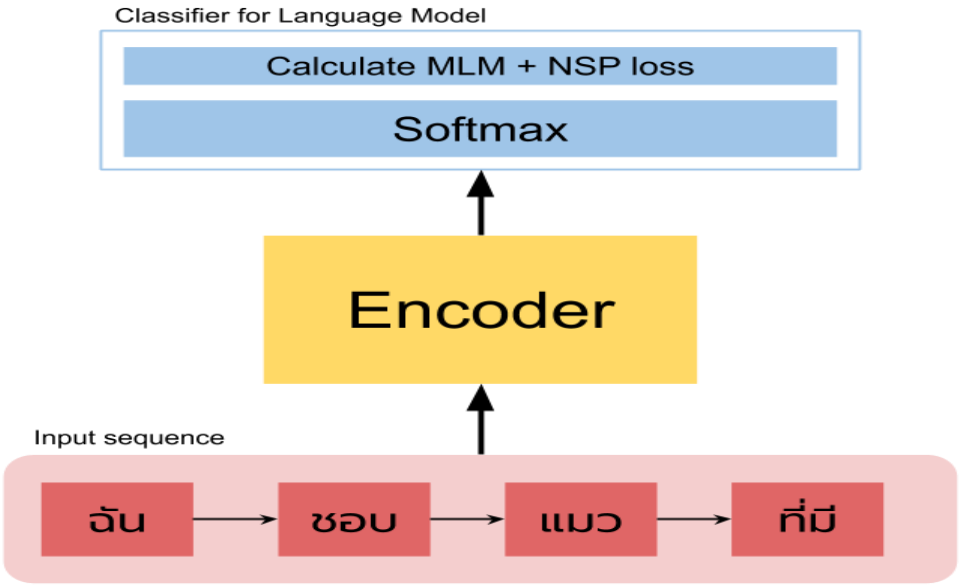
สังเกตุว่า cats มีความสัมพันธ์กับ Bobby + takes ในฐานะกรรมของประโยค และก็ยังมีความสัมพันธ์กับ love + climbing ในฐานะที่เป็นประธาน ในเชิงความหมาย Climbing trees มีความสัมพันธ์กับ cats เพราะว่าเป็นสิ่งที่แมวชอบ แต่ก็ยังมีความหมายโดยนัยว่า the park ต้องมี Trees เพื่อให้พวกแมวปีน ดังนั้นแค่ 1 Attention head ซึ่งสามารถเรียนรู้ได้แค่ความสัมพันธ์รูปแบบเดียว ไม่พอในการเรียนรู้ภาพรวมของทั้งประโยค ซึ่งแก้ด้วยวิธีง่ายๆ คือ ทำ Attention หลายๆรอบ โดยแต่ละรอบใช้ Wq, Wk และ Wv คนละตัวกัน แล้วค่อยนำมารวมกัน วิธีการนี้ จะทำให้โมเดลสามารถเรียนรู้ความสัมพันธ์ได้หลายๆมุมมองไปพร้อมๆกัน (Capture various different aspects of the input) สุดท้าย เพื่อรวมผลลัพธ์จากทุกๆ attention head เข้าด้วยกัน เราจะทำการเอาผลลัพธ์ทั้งหมดมาต่อกัน (Concatenation) เป็น Matrix ขนาดใหญ่ๆ แล้วคูณด้วยเมทริกซ์ Wo หรือ Output matrix ซึ่งจะมิกซ์ผลลัพธ์ที่ได้ แล้วย่อมันให้เหลือเท่ากับขนาด 512 ก่อนที่จะส่งต่อไปยังส่วนถัดไป (Pakawat Nakwijit, 2020:Online)



ภาพที่ 2.5 แผนภาพโครงสร้าง Transformers
ที่มา: Pakawat Nakwijit (2020:Online)

2.4 โมเดล BERT

ชื่อเต็มๆ ของโมเดล BERT นั่นก็คือ Bidirectional Encoder Representations from Transformers จะเห็นได้ว่า BERT เป็นอีกหนึ่งโมเดลที่ต่อยอดจาก Transformer คล้ายๆกับ GPT โดย BERT ต่างจาก GPT เนื่องจากมันถูกออกแบบให้เลือกใช้เฉพาะส่วนที่เป็น encoder ซึ่งทำหน้าที่แปลงคำในประโยคให้เปลี่ยนไปเป็นเวกเตอร์ (แต่ GPT เลือกใช้ decoder) เพื่อให้ Encoder สามารถทำหน้าที่เป็น Language Model ได้ BERT จึงเพิ่มโมเดลอีก 1 ตัว ต่อจาก Encoder ที่มีอยู่เดิม เพื่อทำหน้าที่เป็น Classifier นำเวกเตอร์ที่ได้จาก Encoder ไปคำนวนต่อให้ได้คำตอบในรูปแบบคล้ายๆ กับ Language Model ทั่วๆไป ซึ่งหากเป็นโมเดลในตระกูล Transformer ดั้งเดิมนั้น จะไม่มีความสามารถในการแก้ Task ได้หลากหลายเนื่องด้วยมี Input เข้าไปได้แค่ส่วนเดียว เพราะโดยจุดประสงค์ดั้งเดิมของโมเดล Transformer นั้นถูกออกแบบมาเพื่อที่จะทำงานด้าน Machine Translation แต่เมื่อเราต้องการที่จะปรับโครงสร้างการทำงานของโมเดลให้สามารถรองรับกับปัญหาต่างๆที่มากขึ้น เช่น การตอบคำถาม, การแบ่งกลุ่ม, การจัดแบ่งประเภท ก็จะต้องมีรูปแบบนส่วนของฝั่ง Encoder เข้ามาโดยสามารถที่จะมี Input ที่รับเข้ามาได้หลากหลายมากยิ่งขึ้น ทำให้เราสามารถนำ Transformer ไปปรับใช้ได้หลากหลายโดยตัวพัฒนาต่อยอดขึ้นมาในที่นี้นั้น ก็คือตัว BERT นั่นเอง (Pakawat Nakwijit, 2020:Online)



ภาพที่ 2.6 แผนภาพโครงสร้าง Encoder
ที่มา: Pakawat Nakwijit (2020:Online)

ดังนั้น ในภาพรวมแล้ว BERT ไม่ต่างจาก Transformer Encoder มากนัก ส่วนต่างที่ชัดที่สุด อาจจะเป็นเรื่องของขนาด โดย BERT ขยายขนาดให้มีจำนวน attention head มากขึ้น มีจำนวน Layer มากขึ้น และ เพิ่มขนาด Embedding vector ทั้งนี้ เพื่อให้มั่นใจว่าโมเดลสามารถเรียนรู้คุณสมบัติทางภาษาให้ได้มากที่สุด (ตามทฤษฎี ยิ่ง Neural Network มีขนาดใหญ่ๆ หลายๆ layer ยิ่งสามารถเรียนรู้ความสัมพันธ์ที่ซับซ้อนได้ดี) โดย Pre-trained models จาก Paper มี 2 ขนาด คือ

BERT-base: 12 layers, 768 hidden units, 12 heads, ~110M parameters in total

BERT-large: 24 layers, 1024 hidden units, 16 heads, ~340M parameters in total

โดย BERT-base ออกแบบมาให้มีขนาดใกล้เคียงกับ GPT เพื่อใช้ในการวัดผลและเปรียบเทียบ และ BERT-large ออกแบบมาเพื่อทดสอบขีดความสามารถของ BERT แต่นอกจากขนาดโมเดลที่ใหญ่กว่าเดิม อีกองค์ประกอบสำคัญที่ BERT แตกต่างจาก Transformer Encoder คือ activation function โดยมีการเปลี่ยนจาก ReLU ที่ใช้ใน Transformer เป็น GELU (Gaussian Error Linear Unit) เพื่อสร้าง Language Model ที่สามารถเรียนรู้จากบริบทที่มาจากคำที่อยู่ทางซ้ายและขวา ทีม Google AI ตัดสินใจเปลี่ยนโจทย์ Language Model แบบเดิมๆ ซึ่งเป็นการทำนายคำที่จะเกิดขึ้นต่อไปจากประโยคตั้งต้น มาเป็นการฝึกฝนด้วยโจทย์ใหม่ 2 โจทย์ คือ

2.4.1 Masked Language Model

Masked Language Model หรือ MLM ถูกออกแบบมาเพื่อใช้แทนที่ Language Model แบบดั้งเดิม โดยในแต่ละรอบของการสอนโมเดล ประโยคจำนวนหนึ่งจะถูกป้อนเข้ามา โดยที่ประมาณ 15% ของคำทั้งหมดจะโดนลบออกไป (Masked words) และ สิ่งที่โมเดลต้องทำ คือ การเติมคำที่หายไปเหล่านั้นให้ถูกต้อง ซึ่งเพื่อเติมคำที่ถูกต้อง โมเดลต้องเข้าใจโครงสร้างของประโยคเช่นเดียวกับ Language Model แบบดั้งเดิม แต่ทั้งนี้โมเดลสามารถพิจารณาบริบทได้จากทุกคำที่อยู่รอบๆ จึงถือได้ว่า MLM เป็นคำตอบในอุดมคติที่ทีม Google AI ตามหา แต่ปัญหาหนึ่งที่มักเกิดขึ้น หลังจากโมเดลเรียนรู้ภายใต้ข้อมูลแบบ MLM ไปสักระยะหนึ่ง คือ โมเดลมักจะพยายามที่จะเดาคำที่หายไปเพียงอย่างเดียว จนไม่สนใจคำอื่นๆเลย (the model only tries to predict the [MASK] token) เวกเตอร์ของคำอื่นๆที่ได้จึงมีแนวโน้มว่า จะไม่มีข้อมูลที่มีประโยชน์ (might not be as rich as it could be) โดยเฉพาะอย่างยิ่ง เมื่อต้องการนำไปใช้การ Transfer learning นอกจากนี้แล้ว ในการเชิงปฏิบัติมักจะพบว่า การเรียนรู้เพื่อแก้ปัญหา MLM ต่างกับปัญหาที่มันต้องเรียนรู้ในขั้นตอน Fine-tuning ค่อนข้างมาก เนื่องจากมีการใช้ [MASK] token แค่เพียงเฉพาะในขั้นตอน Pre-training แต่ข้อมูลที่ใช้สอนในขั้นตอน Fine-tuning มักจะไม่มีการลบคำบางคำออกไป หรือ ก็คือไม่ได้ใช้ [MASK] token (the [MASK] token does not appear during fine-tuning/testing time) จึงมีแนวโน้มทำให้โมเดลไม่สามารถจัดการปัญหาอื่นๆได้ เพราะ ธรรมชาติของปัญหาต่างกันเกินไป ทางออกที่ถูกนำเสนอขึ้นมา คือ Mixed Mask Strategy โดย ภายใน 15% ของคำที่ทั้งหมดออกไป จะถูกแบ่งออกเป็น

80% ของคำที่หมด จะโดนแทนที่ด้วย [MASK] token

10% ของคำที่หมด จะโดนแทนที่ด้วยคำอื่นๆแบบสุ่ม

10% ของคำที่หมด จะไม่โดนเปลี่ยนแปลง

เนื่องจาก BERT จะคำนวน loss จากเฉพาะในส่วนที่เป็น 15% ที่เลือกมาในรอบแรก และบางส่วนของคำถาม ไม่ใช่ [MASK] token อีกต่อไป โมเดลจึงต้องสนใจทุกๆ Token โดยการคงให้คำบางส่วนอยู่เหมือนเดิม จะช่วยให้โมเดลสามารถสร้างเวกเตอร์ที่ตรงตามคำที่เจอจริงๆได้ง่ายขึ้น (to bias the representation towards the actual observed word) และการแทนที่คำด้วยคำอื่นแบบสุ่มนี้ เป็นการเพิ่ม noise ให้กับโมเดล คล้ายๆกับการทำ regularization เพื่อแกล้งโมเดล ไม่ให้มันมั่นใจเกินกับคำตอบตัวเองจนเกินไป โดย GELU เป็นการเพิ่มความสามารถให้กับ ReLU เพื่อแก้ปัญหา Dead neurons ซึ่งเกิดขึ้นเมื่อข้อมูลจาก neurons มีค่าน้อยกว่า 0 จำนวนมาก

โดยค่าที่เป็นลบเหล่านี้ จะส่งผลให้ ReLU ให้ผลลัพธ์เป็น 0 ไปด้วย มีผลต่อเนื่องทำให้ไม่เกิดการอัพเดตใน neurons กลุ่มนั้นไปตลอดการฝึกฝน GELU แก้ไขปัญหานี้ด้วยสมการใหม่ โดยยังคงผลลัพธ์แบบเดิม เมื่อในช่วงที่ข้อมูลนำเข้ามีค่ามากกว่า 0 {GELU(x) ~ x; x > 0} และแก้ไขผลลัพธ์ในช่วงข้อมูลที่มีค่าน้อยกว่า 0 เพื่อให้มีการเปลี่ยนแปลงเล็กน้อย ไม่ให้เกิด dead neurons (Pakawat Nakwijit, 2020:Online)

2.4.2 Next Sentence Prediction

เนื่องจากปัญหาในงาน NLP ค่อนข้างมีความหลากหลาย เช่น NER, POS tagging, Tokenizing ปัญหากลุ่มนี้ ถือได้ว่าเป็นปัญหาที่อยู่ในระดับคำ กล่าวคือ โมเดลต้องประมวลผลเพื่อทำนายคำตอบสำหรับแต่ละคำๆ ซึ่งแค่ MLM ก็เพียงพอในการจัดการปัญหากลุ่มนี้ได้เป็นอย่างดี แต่ยังมีปัญหากลุ่มใหญ่อีกกลุ่มหนึ่ง เช่น Sentiment Analysis, Question Answering, Summarization ซึ่งเป็นปัญหาในระดับประโยค โมเดลต้องอาศัยความเข้าใจในภาพรวมของทั้งประโยคเพื่อสรุปออกมาเป็นคำตอบที่ต้องการ แต่เนื่องจาก MLM ถูกออกแบบมาให้ประมวลผลในระดับคำ ดังนั้นมันจึงมีแนวโน้มว่า จะไม่สามารถทำความเข้าใจภาพรวมในลักษณะนี้ได้ดี BERT แก้ไขปัญหานี้ด้วยการเพิ่มโจทย์ Next Sentence Prediction หรือ NSP ไปพร้อมๆกับการทำ MLM เพื่อให้โมเดลเรียนรู้ความสัมพันธ์ระหว่างประโยค โดยการป้อน 2 ประโยค จากนั้นให้โมเดลตัดสินว่าประโยคทั้ง 2 นี้เป็นประโยคที่อยู่ติดกันหรือไม่? โดยแบ่งข้อมูลครึ่งนึง เป็นคู่ของประโยคที่อยู่ติดกันและอีกครึ่งนึงเป็นประโยคที่โดนจับคู่กันแบบสุ่ม ดังนั้นในภาพรวม ในแต่ละรอบของการสอน BERT จะรับคู่ของประโยค ซึ่งจะมีคำบางส่วนหายไป จากนั้น BERT ก็จะพยายามทายคำในช่องว่าง พร้อมกับทายว่าประโยคทั้ง 2 อันที่ได้รับมา เป็นประโยคที่อยู่ติดกันหรือไม่? แล้วนำคำตอบที่ได้จากทั้ง 2 คำถามไปคำนวนความถูกต้อง และอัพเดตโมเดล สิ่งนี้ทำให้ BERT สามารถเรียนรู้ความสัมพันธ์ทั้งในระดับคำ และระดับประโยคไปพร้อมๆกัน (Pakawat Nakwijit, 2020:Online)

2.5 โมเดล RoBERTa

เป็น BERT จากทีม Facebook ซึ่งได้ optimize การ pretrain MLM ของ BERT ดั้งเดิม รวมทั้งเพิ่ม Pretrained dataset ให้ใหญ่กว่าเดิมมาก ทำให้ได้ผลลัพธ์ดีกว่า BERT ซึ่งเป็นการพัฒนาโมเดลให้มีขนาดใหญ่มากขึ้น (Yinhan et al., 2019)

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

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

ภาพที่ 2.7 ตารางเปรียบเทียบขนาดและผลลัพธ์ของ RoBERTa กับ Model อื่นๆ

ที่มา: Yinhan et al. (2019)

2.6 โมเดล XLMRoBERTa

XLM-Roberta พัฒนาต่อจาก Roberta โดย pretrain จาก dataset ถึง 100 ภาษา พร้อมกันทำให้เข้าใจภาษาเกือบทุกภาษาทั่วโลก ผลลัพธ์คือสามารถช่วยให้ผลลัพธ์ในการแปลภาษาต่างๆดีขึ้น (Alexis et al., 2020)

Model	D	#M	#lg	en	fr	es	de	el	bg	ru	tr	ar	vi	th	zh	hi	sw	ur	Avg
Fine-tune multilingual model on English training set (Cross-lingual Transfer)																			
Lample and Conneau (2019)	Wiki+MT	N	15	85.0	78.7	78.9	77.8	76.6	77.4	75.3	72.5	73.1	76.1	73.2	76.5	69.6	68.4	67.3	75.1
Huang et al. (2019)	Wiki+MT	N	15	85.1	79.0	79.4	77.8	77.2	77.2	76.3	72.8	73.5	76.4	73.6	76.2	69.4	69.7	66.7	75.4
Devlin et al. (2018)	Wiki	N	102	82.1	73.8	74.3	71.1	66.4	68.9	69.0	61.6	64.9	69.5	55.8	69.3	60.0	50.4	58.0	66.3
Lample and Conneau (2019)	Wiki	N	100	83.7	76.2	76.6	73.7	72.4	73.0	72.1	68.1	68.4	72.0	68.2	71.5	64.5	58.0	62.4	71.3
Lample and Conneau (2019)	Wiki	1	100	83.2	76.7	77.7	74.0	72.7	74.1	72.7	68.7	68.6	72.9	68.9	72.5	65.6	58.2	62.4	70.7
$\text{XLM-R}_{\text{Base}}$	CC	1	100	85.8	79.7	80.7	78.7	77.5	79.6	78.1	74.2	73.8	76.5	74.6	76.7	72.4	66.5	68.3	76.2
XLM-R	CC	1	100	**89.1**	**84.1**	**85.1**	**83.9**	**82.9**	**84.0**	**81.2**	**79.6**	**79.8**	**80.8**	**78.1**	**80.2**	**76.9**	**73.9**	**73.8**	**80.9**
Translate everything to English and use English-only model (TRANSLATE-TEST)																			
BERT-en	Wiki	1	1	88.8	81.4	82.3	80.1	80.3	80.9	76.2	76.0	75.4	72.0	71.9	75.6	70.0	65.8	65.8	76.2
RoBERTa	Wiki+CC	1	1	**91.3**	82.9	84.3	81.2	81.7	83.1	78.3	76.8	76.6	74.2	74.1	77.5	70.9	66.7	66.8	77.8
Fine-tune multilingual model on each training set (TRANSLATE-TRAIN)																			
Lample and Conneau (2019)	Wiki	N	100	82.9	77.6	77.9	77.9	77.1	75.7	75.5	72.6	71.2	75.8	73.1	76.2	70.4	66.5	62.4	74.2
Fine-tune multilingual model on all training sets (TRANSLATE-TRAIN-ALL)																			
Lample and Conneau (2019)†	Wiki+MT	1	15	85.0	80.8	81.3	80.3	79.1	80.9	78.3	75.6	77.6	78.5	76.0	79.5	72.9	72.8	68.5	77.8
Huang et al. (2019)	Wiki+MT	1	15	85.6	81.1	82.3	80.9	79.5	81.4	79.7	76.8	78.2	77.9	77.1	80.5	73.4	73.8	69.6	78.5
Lample and Conneau (2019)	Wiki	1	100	84.5	80.1	81.3	79.3	78.6	79.4	77.5	75.2	75.6	78.3	75.7	78.3	72.1	69.2	67.7	76.9
$\text{XLM-R}_{\text{Base}}$	CC	1	100	85.4	81.4	82.2	80.3	80.4	81.3	79.7	78.6	77.3	79.7	77.9	80.2	76.1	73.1	73.0	79.1
XLM-R	CC	1	100	**89.1**	**85.1**	**86.6**	**85.7**	**85.3**	**85.9**	**83.5**	**83.2**	**83.1**	**83.7**	**81.5**	**83.7**	**81.6**	**78.0**	**78.1**	**83.6**

ภาพที่ 2.8 ตารางการเปรียบเทียบประสิทธิภาพการแปลภาษา

ที่มา: Alexis et al. (2020)

รวมไปถึงผลลัพธ์ในการทำงานทางด้าน QA ยังมีประสิทธิภาพที่สูงขึ้นอีกด้วย

Model	train	#lgs	en	es	de	ar	hi	vi	zh	Avg
BERT-Large†	en	1	80.2 / 67.4	-	-	-	-	-	-	-
mBERT†	en	102	77.7 / 65.2	64.3 / 46.6	57.9 / 44.3	45.7 / 29.8	43.8 / 29.7	57.1 / 38.6	57.5 / 37.3	57.7 / 41.6
XLM-15†	en	15	74.9 / 62.4	68.0 / 49.8	62.2 / 47.6	54.8 / 36.3	48.8 / 27.3	61.4 / 41.8	61.1 / 39.6	61.6 / 43.5
XLM-R_{Base}	en	100	77.1 / 64.6	67.4 / 49.6	60.9 / 46.7	54.9 / 36.6	59.4 / 42.9	64.5 / 44.7	61.8 / 39.3	63.7 / 46.3
XLM-R	en	100	**80.6 / 67.8**	**74.1 / 56.0**	**68.5 / 53.6**	**63.1 / 43.5**	**69.2 / 51.6**	**71.3 / 50.9**	**68.0 / 45.4**	**70.7 / 52.7**

Table 3: **Results on MLQA question answering** We report the F1 and EM (exact match) scores for zero-shot classification where models are fine-tuned on the English Squad dataset and evaluated on the 7 languages of MLQA. Results with † are taken from the original MLQA paper Lewis et al. (2019).

ภาพที่ 2.9 ตารางการเปรียบเทียบการตอบคำถาม

ที่มา: Alexis et al. (2020)

ด้วยผลลัพธ์ที่เหนือกว่าในหลายๆด้าน จากการฝึกฝนมาจากหลากหลายภาษาของ XLMRoBERTa ตัวผมจึงได้เลือกใช้ โมเดลนี้ เพื่อนำมา finetune เข้ากับ QA ภาษาไทย เพื่อให้ได้โมเดลที่มีประสิทธิภาพสูงที่สุด

2.7 ภาษาและเครื่องมือที่ใช้

2.7.1 ภาษา Python

ภาษาโปรแกรม Python คือภาษาโปรแกรมคอมพิวเตอร์ระดับสูง โดยถูกออกแบบมาให้เป็นภาษาสคริปต์ ที่อ่านง่าย โดยตัดความซับซ้อนของโครงสร้างและไวยากรณ์ของภาษาออกไป ในส่วนของการแปลงชุดคำสั่งที่เราเขียนให้เป็นภาษาเครื่อง Python มีการทำงานแบบ Interpreter คือเป็นการแปลชุดคำสั่งทีละบรรทัดเพื่อป้อนเข้าสู่หน่วยประมวลผลให้คอมพิวเตอร์ทำงานตามที่เราต้องการนอกจากนั้นภาษา Python นั้นมีคุณสมบัติเป็นภาษาเขียนโปรแกรมแบบไดนามิกส์และมีระบบการจัดการหน่วยความจำอัตโนมัติและสนับสนุนการเขียนโปรแกรมหลายรูปแบบ ที่ประกอบไปด้วย การเขียนโปรแกรมเชิงวัตถุ Imperative การเขียนโปรแกรมแบบฟังก์ชันและการเขียนโปรแกรมแบบขั้นตอน มันมีไลบรารี่ที่ครอบคลุมการทำงานอย่างหลากหลาย ตัวแปรภาษา (Interpreter) ของภาษา Python นั้นมีให้ใช้ในหลายระบบปฏิบัติการ ทำให้โค้ดของภาษา Python สามารถรันในระบบต่างๆ ได้อย่างกว้างขวาง CPython นั้นเป็นการพัฒนาในขั้นต้นของ Python ซึ่งเป็นโปรแกรมแบบ Open source และมีชุมชนสำหรับเป็นต้นแบบในการพัฒนาเนื่องจากมันได้มีการนำไปพัฒนากระจายไปอย่างหลากหลาย CPython นั้นจึงถูกจัดการโดยองค์กรไม่แสวงหาผลกำไรอย่าง Python Software Foundation ในปัจจุบันภาษาโปรแกรม Python มีเวอร์ชันให้เลือกใช้งานคือ Python 2.x และ Python 3.x ซึ่งเผยแพร่มาตั้งแต่ในปี 2000 และ 2008

ตามลำดับ โดยระหว่างที่ผู้เขียนกำลังเขียนบทความนี้ เวอร์ชันล่าสุดคือ Python 2.7.15 และ Python 3.7.2 สำหรับปัญหาทั่วไปของผู้เริ่มต้นศึกษาการเขียนโปรแกรมด้วยภาษา Python คือการตัดสินใจเลือกใช้งานระหว่างเวอร์ชัน Python 2.x หรือ Python 3.x แต่ก่อนที่จะตอบปัญหานี้ ผู้เขียนอยากจะขออธิบายพื้นฐานของหมายเลขเวอร์ชัน เพื่อให้เข้าใจความแตกต่างระหว่างสองเวอร์ชันนี้ก่อน โดยหลักการมาตราฐานการตั้งหมายเลขเวอร์ชันของซอฟต์แวร์ (Semantic Versioning) เป็นการกำหนดขอบเขตของการเปลี่ยนแปลงเวอร์ชันต่าง ๆ ซึ่งมีรูปแบบประกอบด้วยหมายเลข 3 หลัก คือ X.Y.Z ทำให้ผู้ใช้งานสามารถจัดการกับการเปลี่ยนแปลงเวอร์ชัน เพื่อให้ซอฟต์แวร์สามารถทำงานได้เป็นปกติ และรองรับฟีเจอร์ใหม่ๆ ได้ในอนาคต (9experttraining, 2564: ออนไลน์)

2.7.2 Google Colab

ภาพที่ 2.10 Google Colab
ที่มา: Michael J. Garbade (2021: Online)

Google Colab หรือชื่อ Google Colaboratory เป็นบริการ Software as a Service (Saas) โฮสต์โปรแกรม Jupyter Notebook บน Cloud ของ Google โดยสามารถใช้ Google Colab สร้าง Notebook เขียนโปรแกรมภาษา Python ได้ฟรีๆ และแถมยังมี GPU, TPU ให้เราได้ใช้ฟรีอีก ทีละ 12 ชั่วโมง โดยในขณะนี้ Google Colab มี GPU ให้เราใช้ ดังนี้ Nvidia Tesla K80, Nvidia Tesla T4 และ Nvidia Tesla P100 (ดีสุด) โดยเราสามารถเช็คสเปคของ GPU ได้ด้วยคำสั่ง !nvidia-smi ถ้าไม่พอใจสามารถ เลือกเมนู Runtime / Factory reset runtime เพื่อเปลี่ยนเครื่อง อาจจะได้ GPU ที่ดีขึ้น ถ้ามีเครื่องว่าง นอกจากนี้ยังมี PyTorch, TensorFlow, Keras และ OpenCV ที่ติดตั้งมาให้เรียบรอย (Michael J. Garbade, 2021: Online)

2.7.3 Hugging Face

ภาพที่ 2.11 Hugging Face
ที่มา: Hugging Face (2016: Online)

Hugging Face เป็นบริษัท startup ทางด้าน AI ซึ่งได้เปิดตัวบริการ ML ของตัวเองซึ่งเรียกว่า Inference API ซึ่งให้การเข้าถึงโมเดลที่ผ่านการฝึกอบรมมาแล้วนับพัน (ส่วนใหญ่เป็นTransformer) เมื่อเทียบกับตัวเลือกที่จำกัดของบริการอื่นๆ ลูกค้าสามารถเช่า Inference API ตามทรัพยากรที่ใช้ร่วมกันหรือให้ Hugging Face ตั้งค่าและดูแลโครงสร้างพื้นฐานสำหรับพวกเขา โมเดลที่โฮสต์ทำให้ ML สามารถเข้าถึงได้ในหลากหลายองค์กร และสามารถนำ pretrain model ต่างๆมาใช้งานได้ (Hugging Face, 2016: Online)

2.7.4 Library Tkinter

Tkinter นั้นย่อมาจาก TK Interface เป็น Library สำหรับการพัฒนา GUI ที่ติดมากับ Python ตอนคุณลง (standard library) มีการเรียนรู้ที่ค่อนข้างง่ายสำหรับมือใหม่ การใช้งานไม่ซับซ้อนตรงตัว ใน Python มี GUI Library หลายตัวแต่ Tkinter เป็นหนึ่งในตัวที่นิยมมากตัวหนึ่งด้วยความที่มันง่าย และ Cross platform สามารถใช้ได้ทั้ง MacOS, Linux, Windows โดยส่วนใหญ่แล้วจะเป็นตัวเลือกที่น่าสนใจสำหรับคนที่พึ่งเรียนพื้นฐานของ Python เสร็จแล้วอยากพัฒนาโปรแกรมขึ้นมาสักตัว (Isara, 2021: Online)

2.7.5 Library FastAPI

ภาพที่ 2.12 FasAPI
ที่มา: Tiangolo (2018: Online)

API คือกลไกที่ช่วยให้ส่วนประกอบซอฟต์แวร์สองส่วนสามารถสื่อสารกันได้โดยใช้ชุดคำจำกัดความและโปรโตคอล ตัวอย่างเช่น ระบบซอฟต์แวร์ของสำนักพยากรณ์อากาศประกอบด้วยข้อมูลสภาพอากาศรายวัน แอปสภาพอากาศในโทรศัพท์ของคุณจะ "สื่อสาร" กับระบบนี้ผ่าน API และแสดงการอัปเดตสภาพอากาศทุกวันบนโทรศัพท์ของคุณ API ย่อมาจาก “Application Program Interface” (ส่วนต่อประสานโปรแกรมประยุกต์) ในบริบทของ API คำว่า “Application” หมายถึงทุกซอฟต์แวร์ที่มีฟังก์ชันชัดเจน ส่วน “Interface” อาจถือเป็นสัญญาบริการระหว่างสองแอปพลิเคชัน ซึ่งสัญญานี้จะกำหนดวิธีที่ทั้งสองสื่อสารกันโดยใช้คำขอและการตอบกลับ เอกสารประกอบ API มีข้อมูลเกี่ยวกับวิธีที่นักพัฒนาจัดโครงสร้างคำขอและการ

สถาปัตยกรรม API มักจะถูกอธิบายในแง่ของไคลเอ็นต์และเซิร์ฟเวอร์ แอปพลิเคชันที่ส่งคำขอเรียกว่าไคลเอ็นต์ และแอปพลิเคชันที่ส่งการตอบกลับเรียกว่าเซิร์ฟเวอร์ ในตัวอย่างสภาพอากาศ ฐานข้อมูลสภาพอากาศของสำนักงานคือเซิร์ฟเวอร์ และแอปมือถือคือไคลเอ็นต์ API ทำงานใน 4 รูปแบบด้วยกัน โดยขึ้นอยู่กับเวลาและสาเหตุที่สร้าง API

- SOAP API

API เหล่านี้ใช้ Simple Object Access Protocol (โปรโตคอลการเข้าถึงอ็อบเจกต์อย่างง่าย) ไคลเอ็นต์และเซิร์ฟเวอร์จะแลกเปลี่ยนข้อความโดยใช้ XML ซึ่งเป็น API ที่มีความยืดหยุ่นน้อยซึ่งเคยได้รับความนิยมมากกว่านี้ในอดีต

- RPC API

API เหล่านี้เรียกว่า Remote Procedure Call (การเรียกใช้กระบวนการระยะไกล) ไคลเอ็นต์ดำเนินการฟังก์ชัน (หรือกระบวนการ) หนึ่งๆ บนเซิร์ฟเวอร์ และเซิร์ฟเวอร์ส่งผลลัพธ์กลับไปยังไคลเอ็นต์

- Websocket API

คืออีกหนึ่งการพัฒนา Web API สมัยใหม่ที่ใช้อ็อบเจกต์ JSON ในการส่งข้อมูล WebSocket API รองรับการสื่อสารสองทางระหว่างแอปไคลเอ็นต์และเซิร์ฟเวอร์ เซิร์ฟเวอร์สามารถส่งข้อความเรียกกลับไปยังไคลเอ็นต์ที่เชื่อมต่อ จึงทำให้มีประสิทธิภาพมากกว่า REST API

- REST API

API เหล่านี้เป็น API ที่ได้รับความนิยมและยืดหยุ่นที่สุดที่พบในเว็บไซต์ปัจจุบัน ไคลเอ็นต์ส่งคำขอไปยังเซิร์ฟเวอร์เป็นข้อมูล เซิร์ฟเวอร์ใช้ข้อมูลอินพุตจากไคลเอ็นต์นี้เพื่อเริ่มต้นฟังก์ชันภายในและส่งคืนข้อมูลเอาต์พุตกลับไปยังไคลเอ็นต์ REST ย่อมาจาก Representational State Transfer (การโอนสถานะแบบตัวแทน) REST ช่วยกำหนดชุดฟังก์ชันต่างๆ เช่น GET, PUT, DELETE ฯลฯ ที่ไคลเอ็นต์สามารถใช้เพื่อเข้าถึงข้อมูลเซิร์ฟเวอร์ได้ ไคลเอ็นต์และเซิร์ฟเวอร์แลกเปลี่ยนข้อมูลโดยใช้ HTTP คุณสมบัติหลักของ REST API คือ ความเป็นอิสระ ความเป็นอิสระ

หมายความว่าเซิร์ฟเวอร์จะไม่บันทึกข้อมูลไคลเอ็นต์ระหว่างคำขอ คำขอของไคลเอ็นต์ไปยังเซิร์ฟเวอร์นั้นคล้ายกับ URL ที่คุณพิมพ์ในเบราว์เซอร์ของคุณเพื่อเยี่ยมชมเว็บไซต์ การตอบสนองจากเซิร์ฟเวอร์เป็นข้อมูลธรรมดา โดยไม่มีการแสดงผลแบบกราฟิกทั่วไปของหน้าเว็บ (Aws. 2022: Online)

2.7.6 ภาษา PyTorch

ภาพที่ 2.13 PyTorch

ที่มา: Anurag Lahon (2020:Online)

PyTorch อิงจาก Python: PyTorch เป็น Python-centric หรือ "pythonic" ซึ่งออกแบบมาเพื่อการบูรณาการในโค้ด Python อย่างลึกซึ้ง แทนที่จะเป็นอินเทอร์เฟซไปยังไลบรารีที่เขียนในภาษาอื่น Python เป็นหนึ่งในภาษาที่นิยมใช้กันมากที่สุดโดยนักวิทยาศาสตร์ด้านข้อมูลและยังเป็นหนึ่งในภาษาที่นิยมใช้มากที่สุดสำหรับการสร้างแบบจำลองการเรียนรู้ของเครื่องและการวิจัย ML โดยมีข้อดีต่างๆดังนี้

- ง่ายต่อการเรียนรู้: เนื่องจากไวยากรณ์ของมันคล้ายกับภาษาโปรแกรมทั่วไปเช่น Python ดังนั้น PyTorch จึงเรียนรู้ได้ง่ายกว่า Deep Learning Framework อื่น ๆ
- Debugging: PyTorch สามารถดีบักได้โดยใช้ Debugging Tools ของ Python ที่มีอยู่มากมาย (เช่น เครื่องมือ pdb และ ipdb ของ Python)
- Dynamic computational graphs: PyTorch รองรับกราฟการคำนวณแบบไดนามิก ซึ่งหมายความว่าพฤติกรรมเครือข่ายสามารถเปลี่ยนแปลงได้โดยทางโปรแกรมขณะรันไทม์ สิ่งนี้ทำให้การปรับโมเดลให้เหมาะสมได้ง่ายขึ้นมาก และทำให้ PyTorch มีข้อได้เปรียบเหนือ machine learning frameworks อื่นๆ ซึ่งถือว่าโครงข่ายประสาทเทียมเป็นวัตถุคงที่
- Data parallelism: คุณลักษณะการขนานข้อมูลช่วยให้ PyTorch สามารถแจกจ่ายงานการคำนวณระหว่างแกนประมวลผล CPU หรือ GPU หลายตัว แม้ว่าความขนานนี้สามารถทำได้ในเครื่องมือการเรียนรู้ของเครื่องอื่น ๆ แต่ PyTorch นั้นง่ายกว่ามา

- Community: PyTorch มีชุมชนและฟอรัมที่กระตือรือร้นมาก (discuss.pytorch.org) เอกสารประกอบ (pytorch.org) มีการจัดระเบียบและเป็นประโยชน์สำหรับผู้เริ่มต้น มันอัพเดทอยู่เสมอด้วยการเปิดตัว PyTorch และเสนอชุดบทช่วยสอน PyTorch ใช้งานง่ายมาก ซึ่งหมายความว่าช่วงการเรียนรู้สำหรับนักพัฒนาค่อนข้างสั้น (Mindphp, 2022: Online)

2.7.7 Library Simple Transformers

ภาพที่ 2.14 Simple Transformers
ที่มา: ThilinaRajapakse (2022:Online)

โมเดล Simple Transformer สร้างขึ้นโดยคำนึงถึงงาน Natural Language Processing (NLP) โดยเฉพาะ โมเดลดังกล่าวแต่ละรุ่นมาพร้อมกับคุณสมบัติและฟังก์ชันการทำงานที่ออกแบบมาให้เหมาะสมกับงานที่เราจะทำโดยใช้ transformer โดยกระบวนการระดับสูงของการใช้โมเดล Simple Transformers เป็นไปตามรูปแบบเดียวกัน เริ่มต้นแบบจำลองเฉพาะงาน ฝึกโมเดลด้วย train_model() ประเมินแบบจำลองด้วย eval_model() ทำการคาดคะเนข้อมูล (ไม่มีป้ายกำกับ) ด้วย predict() อย่างไรก็ตาม มีความแตกต่างที่จำเป็นระหว่างรุ่นต่างๆ เพื่อให้แน่ใจว่าเหมาะสมกับงานที่ต้องการ ความแตกต่างที่สำคัญมักจะเป็นความแตกต่างในรูปแบบข้อมูลอินพุต/เอาต์พุต และคุณสมบัติเฉพาะ/ตัวเลือกการกำหนดค่างานใดๆ สิ่งเหล่านี้สามารถพบได้ในส่วนเอกสารสำหรับแต่ละงาน โมเดล Simple Transformer เฉพาะงานที่ใช้งานในปัจจุบันพร้อมกับงานต่างๆที่สามารถทำได้ มีทั้งหมดดังนี้

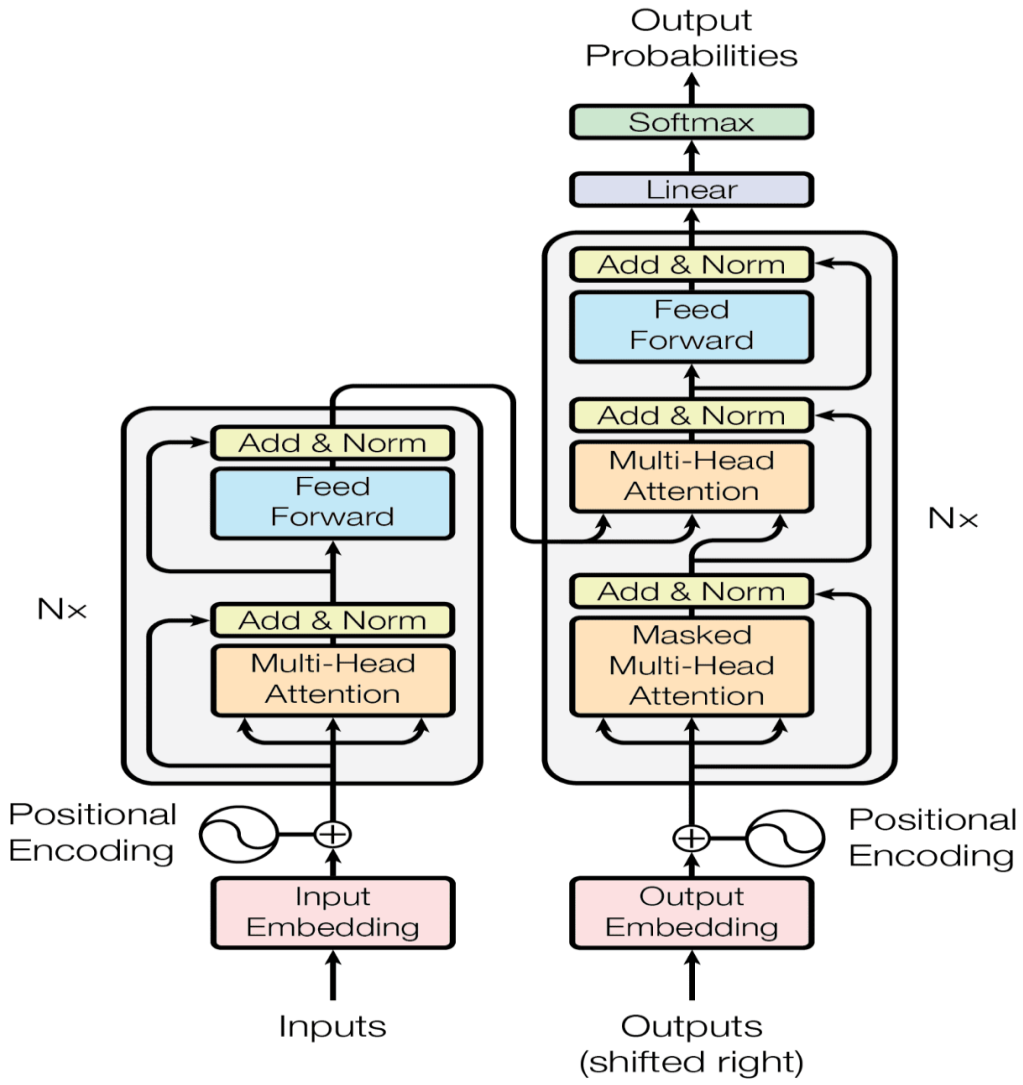
- การจัดประเภทข้อความไบนารีและหลายคลาส (ClassificationModel)
- AI สนทนา (ConvAIModel)
- การสร้างภาษา (LanguageGenerationModel)
- การฝึกโมเดลภาษา/การปรับจูนอย่างละเอียด (LanguageModelingModel)
- การจัดประเภทข้อความหลายป้ายกำกับ (MultiLabelClassificationModel)
- การจำแนกประเภทหลายรูปแบบ (MultiModalClassificationModel)
- การรับรู้ชื่อเฉพาะ (NERModel)
- ตอบคำถาม (QuestionAnsweringModel)
- การถดถอย (ClassificationModel)

- การจำแนกคู่ประโยค (ClassificationModel)
- การสร้างการแทนข้อความ (RepresentationModel)
- การแก้ไขเอกสาร (RetrievalModel)

(Simple Transformer, 2022: Online)

2.8 งานวิจัยที่เกี่ยวข้อง

Ashish et al. (2019) เป็นโมเดลประเภท Sequence to sequence โดยอาศัยกระบวนการคำนวณ Attention mechanism มาช่วยในการเสริมประสิทธิภาพการทำงานของโมเดลสำหรับการแก้ปัญหา machine translation ซึ่งจะมความแม่นยำน้อยลงเมื่อ Sequence มีความยาวมากขึ้นซึ่งการใช้ attention เข้ามาช่วยในการคำนวณ ทำให้โมเดลสามารถเรียนรู้ความสัมพันธ์ในเชิงลึกของข้อคววามภาษาได้มากยิ่งขึ้น อีกทั้งยังเข้าใจในหลายๆแง่มุมได้มากขึ้น โดยใช้ Multihead Attention ซึ่งส่งผลให้โมเดล สามารถเข้าใจโครงสร้างทางไวยากรณ์ที่ซับซ้อนของภาษาได้มากขึ้น อีกทั้งยังสามารถคำนึงถึงบริบทรอบข้างของข้อความได้อีกด้วย สำหรับปัญหา Machine translation นั้นก็จะใช้การทำ Encoder-Decoder สำหรับช่วยแปลงภาษาต้นทางไปยังภาษาปลายทาง สำหรับตัวโมเดลนี้ ถูกตั้งชื่อว่า Transformers หัวใจหลักของ Transformer คือกระบวนการที่เรียกว่า Self-Attention โดยกระบวนการนี้นอกจากจะเป็นสิ่งที่ทดแทน RNN และ CNN ได้แล้ว ยังแสดงถึงความข้องเกี่ยวกันของคำต่างๆ ในข้อความ ทำให้สามารถแก้ปัญหา Coreference Resolution ไปได้ ซึ่งการแก้ปัญหานี้มีความสำคัญอย่างมากต่องาน NLP หลายประเภท เช่น Machine translation โดยในส่วนของ Transformer Model นี้ จะยังคงแบ่งเป็นสองฝั่งอยู่ โดยฝั่งซ้ายจะเป็น encoder ที่รับ Input sequence เข้ามา ส่วนฝั่งขวาคือ Decoder ที่รับ Output sequence โดยในขั้นตอนการฝึกฝนนี้ Output sequence จะถูกเลื่อนไปทางขวาหนึ่งตำแหน่ง ซึ่งเป็นการฝึกฝนแบบ Teacher forcing



ภาพที่ 2.15 โครงสร้างโมเดล transformer

ที่มา: Ashish et al. (2019)

Jacob et al. (2019) โมเดล BERT หรือ Bidirectional Encoder Representations from Transformers จะเห็นได้ว่า BERT เป็นอีกหนึ่งโมเดลที่ต่อยอดจาก Transformer คล้ายๆกับ GPT โดย BERT ต่างจาก GPT เนื่องจากมันถูกออกแบบให้เลือกใช้เฉพาะส่วนที่เป็น encoder ซึ่งทำหน้าที่แปลงคำในประโยคให้เปลี่ยนไปเป็นเวกเตอร์ (แต่ GPT เลือกใช้ decoder) เพื่อให้ encoder สามารถทำหน้าที่เป็น Language Model ได้ BERT จึงเพิ่มโมเดลอีก 1 ตัว ต่อจาก encoder ที่มีอยู่เดิม เพื่อทำหน้าที่เป็น Classifier นำเวกเตอร์ที่ได้จาก encoder ไปคำนวนต่อให้ได้คำตอบในรูปแบบคล้ายๆกับ Language Model ทั่วๆไป เพื่อสร้าง Language Model ที่สามารถเรียนรู้จากบริบทที่มาจากคำที่อยู่ทางซ้ายและขวา ทีม Google AI ตัดสินใจเปลี่ยนโจทย์ Language Model แบบเดิมๆ ซึ่ง

เป็นการทำนายคำที่จะเกิดขึ้นต่อไปจากประโยคตั้งต้น มาเป็นการฝึกฝนด้วยโจทย์ใหม่ 2 โจทย์ คือ **Masked Language Model , Next Sentence Prediction** ซึ่งช่วยให้โมเดล BERT สามารถเรียนรู้บริบท หลักไวยากรณ์ทางาษานั้นๆรวมถึงความสัมพันธ์ในเชิงภาษาได้อย่างมีระสิทธิภาพมากขึ้น

Wicharn RueangkhajornWicharn and Jonathan H. Chan. (2022) เป็นการนำโมเดล BERT และ โมเดล RoBERTa มาฝึกฝนกับขอมูลชุด Dataset Wikipedia dataset ของบริษัท Iapp โดยมีข้อมูลฝึกฝนทั้งหมด 5,829 ข้อมูลเทสทั้งหมด 723 และข้อมูลทดสอบ 690 รวมทั้งหมดมี Dataset 7242 โดย BERT จะใช้ pretrain เป็น Bert-multi-cased-finetuned-xquadv1 และ RoBERTa จะใช้ Pretrain เป็น deepset/xlm-roberta-base-squad2 โดยมีผลลัพธ์เมื่อนำมาใช้งานกับ ภาษาไทย ดังนี้

TABLE VI. EVALUATION SCORE

Metrics	BERT		RoBERTa	
	Before fine-tuned	*After fine-tuned*	*Before fine-tuned*	*After fine-tuned*
F1-score	48.26	68.10	59.99	73.62
Exact match	31.67	57.39	44.26	62.51

ภาพที่ 2.16 ผลลัพธ์ประสิทธิภาพของทั้งสองโมเดล

ที่มา: Wicharn RueangkhajornWicharn and Jonathan H. Chan. (2022)

บทที่ 3
วิธีดำเนินงานวิจัย

สำหรับการดำเนินการพัฒนาโปรแกรมเสิร์ชความรู้อัตโนมัติด้วยการประมวลผลภาษาธรรมชาตินั้น สามารถแบ่งออกเป็น 8 ส่วน ดังนี้

3.1 การเตรียมข้อมูล

3.1.1 Pre-train XLMRoBERTa

3.1.2 ชุดข้อมูลฝึกฝน

3.2 การฝึกฝนโมเดล

3.3 การออกแบบระบบ

3.3.1 ออกแบบระบบแปลงเสียงเป็นข้อความ/ข้อความเป็นเสียง

3.3.2 ออกแบบระบบเสิร์ชความรู้

3.3.3 ระบบ Web scraping Thai wiki

3.3.4 ระบบถามตอบภาษาไทย

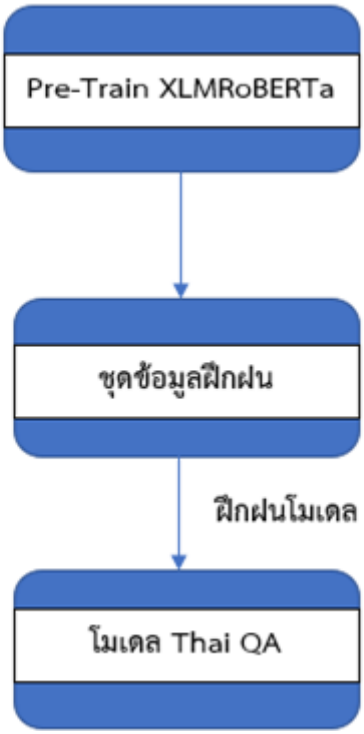
3.3.5 การออกแบบหน้าจอ

3.4 การออกแบบหน้าจอ

3.5 การวัดประสิทธิภาพ

3.1 การเตรียมข้อมูล

ในขั้นตอนของการเตรียมข้อมูลนั้น จะมีขั้นตอนตั้งแต่การเริ่มโหลด Pretrain Model มาจาก Hugging Face และการดาวน์โหลด ชุดข้อมูลฝึกฝน โดยมีภาพรวมการทำงานดังนี้



ภาพที่ 3.1 ขั้นตอนการเตรียมข้อมูล

3.1.1 ชุดข้อมูลฝึกฝน

ข้อมูลที่ใช้ในการศึกษาครั้งนี้ คือ ข้อมูลจากชุดข้อมูลฝึกฝน “thaiqa_squad” ซึ่ง ประกอบไปด้วยคู่ประโยคภาษาไทย พร้อมคู่คำถามและคำตอบ ใน 1 แถว ซึ่งมีข้อมูลทั้งหมด 4,074 แถว โดยประกอบไปด้วย ข้อมูลในส่วนฝึกฝน 4,000 แถว และข้อมูลสำหรับ ทดสอบ 74 โดยมีลักษณะของชุดข้อมูลดังนี้

ตารางที่ 3.1 องค์ประกอบของชุดข้อมูล

ชื่อคอลัมน์	ความหมาย
Answer	ข้อความคำตอบ
Question_id	Id ของคำถาม
Article_id	Id ของข้อความ
Context	ข้อความ
Question	คำถาม
Answers	องค์ประกอบของคำตอบ

ชุดข้อมูลจะประกอบไปด้วย Question_id, Article_id, Context, Question, Answer, Answers โดยในส่วนของ answers นั้นจะมีองค์ประกอบเพิ่มเติมคือ Answer_begin_position หมายถึงตำแหน่งแรกของอักษรคำตอบ, Answer_end_position หมายถึงตำแหน่งสุดท้ายของอักษรคำตอบ, Answer หมายถึงคำตอบที่เป็นลักษณะข้อความ

โดยในส่วนของ ชุดข้อมูลฝึกฝนนั้น ก็จะมีส่วนของข้อมูลฝึกฝนกับ ข้อมูลทดสอบ Context ทั้งหมดนั้น ได้มาจาก Thai Wiki โดยการดึงข้อมูลออกมาจากเว็บไซต์ เพื่อนำมาสร้างไว้เป็นชุดข้อมูลฝึกฝน ทั่วไปสำหรับผู้ที่ต้องการพัฒนาระบบ ถามตอบภาษาไทย สามารถที่จะโหลดไปทดสอบหรือฝึกฝนโมเดลได้อย่างทันที ซึ่งถูกพัฒนาขึ้นมาโดย NECTEC โดยในขั้นตอนก่อนการฝึกฝนนั้น จะมีการจัดการกับ ชุดข้อมูลฝึกฝน ก่อน เนื่องมาจากว่า ตัวของ ชุดข้อมูลฝึกฝน นั้น จะมีส่วนของ Context ถูกดึงออกมาจาก Thai Wiki นั้นจะมี tag html อยู่ในตัวของ Context ด้วย ดังนั้นก่อนที่จะนำไปฝึกฝนนั้น จึงต้องมีการตัดเอา Tag เหล่านั้นออกเสียก่อน โดยมีข้อความตัวอย่างยกมา 1 Context ดังนี้

- ข้อความก่อนตัด Tag html

“ <doc id="115035" url=https://th.wikipedia.org/wki?curid=115035 title="เบนจี้">เบนจี้ เบนจี้ () เป็นชื่อตัวละครหมาพันทางแสนรู้ ที่ปรากฏอยู่ใน.... doc> “

- ข้อความหลังตัด Tag html

“ เบนจี้ เบนจี้ () เป็นชื่อตัวละครหมาพันทางแสนรู้ ที่ปรากฏอยู่ใน.... ”

เมื่อนำ Tag html ออกไปแล้ว จะต้องสร้าง List ใหม่ขึ้นมาเพื่อเก็บข้อมูลของ Context ที่ถูกลบ Tag เหล่านั้นออกหมดแล้วเอาไว้ เนื่องจาก Data type ของ ชุดข้อมูลฝึกฝน ที่ไม่สามารถ Assign ค่าใหม่ลงไปได้ ทำให้เราจำเป็นต้องนำข้อมูลเหล่านั้น ไปเก็บไว้ในตัวแปรใหม่ ซึ่งชนิด Data type ที่เลือกใช้ในการเก็บข้อมูลคือ List เพราะมีความสะดวกในการใช้งาน ในการทำ Iteration และยังรวมไปถึงการแก้ไขค่าต่างๆก็ทำได้อย่างง่ายดาย ซึ่งหลังจากที่เราได้ตัด html ออกจาก Context แล้วกลับพบว่ายังมีสิ่งที่ไม่น่าจะมีความเกี่ยวข้องกับ Context หรือเกี่ยวข้องกับใจความสำคัญของ Context มากนัก นั่นก็คือ ช่องว่าง ที่เว้นวรรค คำแต่ละคำออกจากกัน ซึ่งมีความเป็นไปได้ว่าอาจมีการเว้นว่างที่ไม่คงที่ และในหลายๆบทความ อาจเกิดการแบ่งช่องว่างที่แตกต่างกัน อีกทั้งยังอาจไม่มีนัยยะสำคัญอะไรกับส่วนของความหมายใน Context

ด้วยเหตุนี้ จึงได้ทำการ ตัดช่องว่างทั้งหมดออกเพื่อให้ข้อความทั้งหมดเรียงชิดติดกันเป็นชุดของข้อความยาวๆ ที่จะถูดตัดแบ่งออกเป็นส่วนๆในภายหลัง อีกทั้งการขยับช่องว่างทั้งหมดเข้ามานี้ ยังสามารถลดความยาวโดยรวมของข้อมูลลงได้อีกด้วย ทำให้โมเดล สามารถมองเห็นชุดความสัมพันธ์ของตัวอักษรที่เรียงกันได้อย่างตรงจุด ไม่มีช่องว่างมาคั่นไว้ เพราะแม้ว่าตัวอักษรทั้งหมดจะติดกันโดยไม่มีช่องไฟเลย แต่ตัวของ Tokenizer ของ Pretrain ที่เราจะโหลดมาเพื่อทำการจัดการกับข้อมูลนั้น ก็มีความสามารถในการตัดแบ่ง และแปลงข้อมูลเหล่านั้น ให้อยู่ในรูป Vector ที่สามารถ Represent ชุดของ Sequence ของ Context เหล่านั้น ได้อย่างถูกต้อง

- ข้อความหลังลบช่องว่างออก
“เบนจี้เบนจี้()เป็นชื่อตัวละครหมาพันทางแสนรู้ที่ปรากฏอยู่ใน...”

เมื่อข้อมูล Context ทั้งหมดใน ชุดข้อมูลฝึกฝน ของเราถูกนำมา ในขั้นตอนการเตรียมข้อมูล ในลักษณะเช่นนี้ทั้งหมดแล้ว จะเกิดข้อผิดพลาดลบางอย่างขึ้นกับตัวของ ชุดข้อมูลฝึกฝน ในส่วนของ Answers ซึ่ง มีสิ่งที่อยู่ใน Key answers นั่นก็คือ Answer_begin_position หรือก็คือ ตัวอักษรแรกของคำตอบที่ถูกต้องของคำถาม ที่ถามขึ้นมาเกี่ยวกับข้อมูลใน Context นั้นๆ เนื่องจากในตอนที่ ชุดข้อมูลฝึกฝน นี้ถูกสร้างขึ้นมา นั้นได้ถูกนับตัวอักษรของคำตอบ จาก Context ที่เป็นต้นฉบับ ที่มีทั้งช่องว่างและ Tag html ดังนั้น เมื่อผ่านการลบ tag html และ ลบช่องว่างทั้งหมดออกไป ทำให้ตัวเลข Index ของ string แรกของคำตอบใน Answer_begin_position ถูกขยับออกไปจากที่เดิม

ด้วยเหตุนี้ จึงต้องเขียน Algorithm มาแก้ปัญหานี้โดยการหา Index ของคำตอบมาใหม่ โดยในตัวของ Answers นั้นจะมีส่วนของ Answer อยู่ด้านใน ซึ่งก็คือ คำตอบของคำถาม ที่มีลักษณะเป็นข้อความคำตอบและเราจะใช้ข้อความเหล่านั้น ในการนำมาหา Answer_begin_position ซึ่งเป็น list ที่เก็บ index แรกของคำตอบเทียบกับ String ใน Context

นั้นๆของคำถาม จากนั้นเก็บข้อมูล Index แรกของคำตอบใหม่ไว้ใน Answer_start พร้อมกับนำ list เหล่านั้นมาทดสอบความถูกต้องของคำถามอีกครั้งเพื่อความแน่ใจว่า algorithm สามารถทำงานได้อย่างถูกต้อง

จากนั้นเมื่อผ่านขั้นตอนของการ เตรียมข้อมูล ทั้งหมดนี้แล้ว จึงจะสามารถนำข้อความทั้งหมดมาเข้าสู่ Tokenizer ของ Transformer โมเดล ที่เราต้องการ เพื่อที่จะแบ่งข้อความยาวๆขงเราออกเป็นช่วงๆผ่าน Max length และ Stride โดย Max length จะหมายถึงความยาวของ Sequence ที่เราจะตัด ออกมาจากข้อความ ส่วน Stride คือ จำนวนก้าวที่จะก้าวผ่าน Sequence แรกไปเพื่อที่จะเริ่มตัดข้อความตาม Max length อีกครั้ง หรือเปรียบเหมือนการทำ N-gram ซึ่งเมื่อทำ Input tokenizer ได้แล้ว จะมีผลลัพธ์ของมาเป็นชุดของ Vector ที่มีองค์ประกอบ 4 ส่วน ซึ่งเมื่อได้ Vector เหล่านี้มาแล้ว มันจะอยู่ในรูปแบบของ Encoder ของ Tokenizer ที่ถูก Pretrain มาก่อนแล้ว การจะอ่านข้อความ Sequence เหล่านี้ได้จะต้องทำการ Decoder ออกมาสียก่อน จึงจะกลับจาก Vector มาเป็นข้อความตัวอักษร ซึ่งสามารถอ่านเข้าใจได้ ในส่วนที่เป็น Represent ของข้อความจะอยู่ในส่วนของ Input_ids ซึ่ง เมื่อเราได้ทดสอบ Decode ออกมาแล้ว ก็จะเห็นเป็น Sequence ของข้อความที่มีความยาวตาม Max_length ที่เราได้กำหนดไว้ และเป็นภาษาที่เราสามารถอ่านเข้าใจได้ โดยจะมีองค์ประกอบของ tag คือ <s> หมายถึง เริ่มต้นประโยคและ </s> หมายถึง Tag ที่ใช้สำหรับการคั่นประโยค เพื่อแบ่งว่า Sequence ใหนเป็น Sequence ใหน เพื่อให้โมเดลสามารถที่จะรับข้อมูลเข้าไปเทนได้อย่างถูกต้อง ตามลำดับ โดยจะมีผลลัพธ์ดังนี้

<s> สุนัขตัวแรกรับบทเป็นเบนจี้ในภาพยนตร์เรื่อง Benji ที่ออกฉายในปี พ.ศ. 2517 มีชื่อว่าอะไร </s></s> เบนจี้เบนจี้()เป็นชื่อตัวละครหมาพันทางแสนรู้ที่ปรากฏอยู่ในภาพยนตร์หลายเรื่องที่เขียนบทและกำกับโดยโจแคมป์ในช่วงทศวรรษ1970ถึง1980ภาพยนตร์เรื่องแรกในชุดใช้ชื่อเรื่องว่าเบนจี้เช่นเดียวกับตัวละครถ่ายทำที่เมืองดัลลัสรัฐเทก</s>

จากนั้น หากเรานำ Inputs ที่ผ่าน Tokenizer มาตรวจสอบดูองค์ประกอบดูจะพบว่า มีโครงสร้างข้อมูลเป็น Dictionary ซึ่งมี Key ทั้งหมด 4 อัน ได้แก่

- Input_ids คือการแปลงคำและ Tag ต่างๆให้อยู่ในรูปของตัวเลขบางอย่างที่สามารถ Represent ข้อความที่ถูกตัดออกกมาเป็น Sequence ซึ่งตัวเลขเหล่านี้ คือส่วนที่ถูก Encoder เอาไว้ของข้อความ Context โดย Tokenizer คือสิ่งที่สามารถ ช่วยให้เราแปลงข้อความเหล่านี้ให้เป็น Vector ได้ โดยที่ มีการทำ Pretrain มาแล้วในก่อนหน้านี้ เมื่อเรียกใช้ Tokenizer ตัวเดิมกลับมา จึงสามารถที่จะ Encode หรือ Decode ข้อมูลได้อย่างถูกต้องตามเดิม ตามที่โมเดลนั้น

ถูกฝึกฝนมา โดยมีลักษณะเป็น Array ของ List ซึ่งแต่ละ List นั้นก็หมายถึงแต่ละประโยคที่ตัดมาในก่อนหน้านี้นั่นเอง โดยมีตัวอย่าง ดังนี้

([0, 6, 110860, 4214, 21573, 8448, 35266, 1805, 206040, 46924, 8181, 1201, 80043, 10650, 3419, 658, 5261, 9562, 136964, 71923, 11308, 5, 7322, 5, 714, 2489, 5451, 176511, 16697, 2, 2, 6, 206040, 46924, 8181, 6, 206040, 46924, 8181, 15, 16, 7526, 23075, 198441, 114215, 61328, 4436, 66156, 21268, 5261, 85830, 22530, 80043, 16322,]),

- Attention_mask คือการใส่ Tag ให้กับ Sequence ที่ถูกตัดออกมาเหล่านั้น โดยเลข 1 จะหมายถึงคำที่รู้จัก ส่วนอักขระพิเศษ หรือคำที่ไม่รู้จักจะถูกแทนด้วยเลข 0 เนื่องจาก Pretrain ที่ถูกฝึกฝนมาก่อนหน้า นั้นได้มีการฝึกฝนด้วยภาษาไทยมาแล้ว จึงทำให้สวนใหญ่ พบเป็นเลข 1 ทั้งหมด

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

- Offset_mapping คือตัวเลขของการเชื่อมแต่ละคำในข้อความนั้นๆ ที่ถูกตัดออกมาเป็น sequence โดย จะเป็นชุดตัวเลขในวงเล็บ ตัวเลขแรกนั้นหมายถึง ตัวอักษรแรกของคำนั้นๆ ส่วนตัวเลขหลังคือตัวอักษรสุดท้ายของคำนั้นๆ โดยในส่วนของ Tag แรกสุดที่เป็น <s> นั้นจะถูกแทนด้วยค่า (0,0) และค่าช่องว่างถัดมาจาก Tag แรกก็จะถูกแทนด้วยค่า (0,1) และหลังจากนั้นไปก็จะแทนตัวเลขตัวเลขในวงเล็บด้วย Index ของตัวอักษรแรก และตัวเลขตัวที่สองในวงเล็บ ก็จะถูกแทนด้วย Index ของตัวอักษรตัวสุดท้าย ของคำนั้นๆ

[(0, 0), (0, 1), (0, 5), (5, 8), (8, 11), (11, 14), (14, 16), (16, 20), (20, 23), (23, 25), (25, 26), (26, 28), (28, 36), (36, 42), (43, 46), (46, 48), (49, 52), (52, 55), (55, 58), (58, 62), (63, 64),]

- Overflow_to_sample_mapping หมายถึง จำนวนของ Sequence ที่ตัดออกมาได้ทั้งหมดตามจำนวนของ Max length และ stride โดยจะถูแทนแต่ละ Sequence ด้วยตัวเลข 0 โดยจำนวนของลข 0 ที่ปรากฏใน list นั้นก็คือจำนวนทั้งหมดของ Sequence

([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

เมื่อนำข้อมูลมาตัดเป็น Sequence เสร็จแล้ว ขั้นต่อไปคือการหาว่า ในแต่ละ Sequence ที่ถูกตัดออกมานั้น มี Sequence ใหนบ้าง ที่มีคำตอบของคำถามอยู่ในส่วนหนึ่งของข้อความ Sequence นั้นๆ โดยการเขียนฟังก์ชั่นมาเก็บค่าเหล่านั้นไว้ใน List ที่แสดงถึง Start Position และ End position ของแต่ละ Sequence

[0, 0, 0, 0, 88, 70, 52, 34, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 91, 73, 55, 37, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

จากตัวอย่างด้านบนจะเห็นว่า เมื่อแบ่งข้อความออกเป็น Sequence ทั้งหมด 18 Sequence จะมีคำตอบอยู่ใน Sequence ที 4,5,6 และ 7 (นับแบบ Index) ซึ่ง list ที่อยู่ด้านบนจะบ่งบอกถึง Start position ของ คำตอบซึ่งอยู่ใน Sequence นั้นๆ ส่วน List ด้านล่างนั้น จะหมายถึง End position ของคำตอบซึ่งอยู่ใน Sequence นั้นๆ เช่นกัน นั่นก็เป็นเพราะว่า Sequence ที่เราตัด sequence ออกมาด้วยวิธี n-gram นั้น อาจจะตัดโดนส่วนที่เป็นคำตอบได้หลายช่วงของ sequence ดังนั้น คำตอบที่สามารถพบได้ใน sequence เหล่านั้น จึงสามารถที่จะมีมากกว่า 1 sequence ได้ แม้คำตอบจะมีเพียงคำตอบเดียว นั่นเอง

ขั้นตอนถัดไป คือการนำ ชุดข้อมูลฝึกฝน ทั้งหมด เข้าไปเป็น input ผ่านตัวของ Tokenizer ของ Pretrain ที่เราเลือกมาสำหรับการฝึกฝน (ในที่นี้คือ XLMRoBERTa) โดยการที่นำชุดข้อมูลฝึกฝน ไปผ่าน Tokenizer เพื่อแปลงเป็น Input นี้จะใช้ฟังก์ชั่น Map ซึ่งจะเรียกใช้งานฟังก์ชั่น ที่เราเราเขียน Algorithm สำหรับการจัดการ ข้อมูล การจัดการ Start position และส่วนต่างๆของข้อมูลให้อยู่ในรูปแบบที่ถูกต้องตามแพทเทิร์นที่จะสามารถนำเข้าไปฝึกฝนใน Transformers ได้ ซึ่งเมื่อทำเช่นนั้นจะพบว่าความยาวของ ชุดข้อมูลฝึกฝน นั้นเพิ่มจากเดิม 4000 เป็น 79572 และในส่วน ชุดข้อมูลทดสอบนั้น มีความยาวจากเดิม 74 เป็น 14595

3.1.2 Pre-train XLMRoBERTa

ใช้ Pre-train จาก Hugging face โดยใช้ Pre-train ของ deepset/xlm-roberta-base-squad2 โดยเราจะทำการโหลดมาทั้ง weight และ tokenizer ของโมเดล เพื่อที่จะสามารถนำ Tokenizer ตัวเดียวกันกับตอนที่ถูกทำ Pretrain มาใช้งานได้เหมือนเดิม ในขั้นตอนของการ finetune เพราะ Tokenizer นั้น จะถูกฝึกฝนมาตามตัวต้นฉบับของโมเดล ซึ่งจะทำหน้าที่ในการตัดแบ่ง Sequence ขอเราออกเป็นส่วนๆ ซึ่งวิธีการตัดเหล่านี้ ถูกผ่านการฝึกฝนมาในช่วง Pretrain และเราต้องใช้ Tokenizer ตัวเดิมเท่านั้น โมเดลถึงจะสามารถนำมา ฝึกฝนต่อได้อย่างถูกต้อง

3.2 การฝึกฝนโมเดล

โดยในการฝึกฝน เราจะปรับความยาวของ Max length ของ Vector ก่อนนำเข้าโมเดล โดยมีความยาวดังนี้ 100,200,400,600,1200 ตามลำดับ ซึ่งนอกจาก พารามิเตอร์ Max length ยังมี Stride ซึ่งมีค่าเท่ากับ 128 (คงที่) ซึ่งมีหน้าที่ในการทำ N-gram ในประโยคที่ตัดมาตามความยาวของ Max length หากจะเปรียบง่ายๆก็คือ Max length คือความยาวของไม้บรรทัด ที่จะตัดเอามาจากประโยค Context ส่วน stride นั่นคือตัวที่จะบอกว่า จะค่อยๆขยับไม้บรรทัดนั้นไปที่ละเท่าใด ที่ต้องค่อยๆขยับไปคล้าย N-gram ก็เพื่อที่จะให้โมเดล สามารถเห็นคำตอบได้จากหลายๆ Sequence และสามารถเรียนรู้ความสัมพันธ์ได้หลากหลายมากยิ่งขึ้น เพราะหากเราตัดตามความยาวของ Max length ไปเลย มันอาจจะตัดโดนส่วนของคำตอบ โดยที่คำตอบนั้น ไม่มีความเกี่ยวข้องกับบริบทของ Context เหล่านั้นเลยซึ่งหากเป็นเช่นนั้น อาจทำให้การเรียนรู้ความสัมพันธ์ระหว่างคำถามและคำตอบของโมเดลอาจทำได้ยากและไม่มีประสิทธิภาพมากพอเนื่องจากโมเดลไม่ได้เรียนรู้ ประโยค, คำถาม, และ คำตอบ รวมไปถึงความสัมพันธ์ของส่วนต่างๆเหล่านั้นได้อย่างครอบคลุม ซึ่งสามารถยกตัวอย่างการทำ Max length และ Stride ได้ ดังนี้ เช่น

Max_lenght = 20

Stride = 10

Question = “การทำงานที่ดีคืออะไร”

Context = “การทำงานที่ดี คือการทำงานที่ดี”

<s>การทำงานที่ดีคืออะไร </s> การทำงานที่ดี คือการ</s>

<s>การทำงานที่ดีคืออะไร </s> ดี คือการทำงานที่ดี [PAD] [PAD] </s>

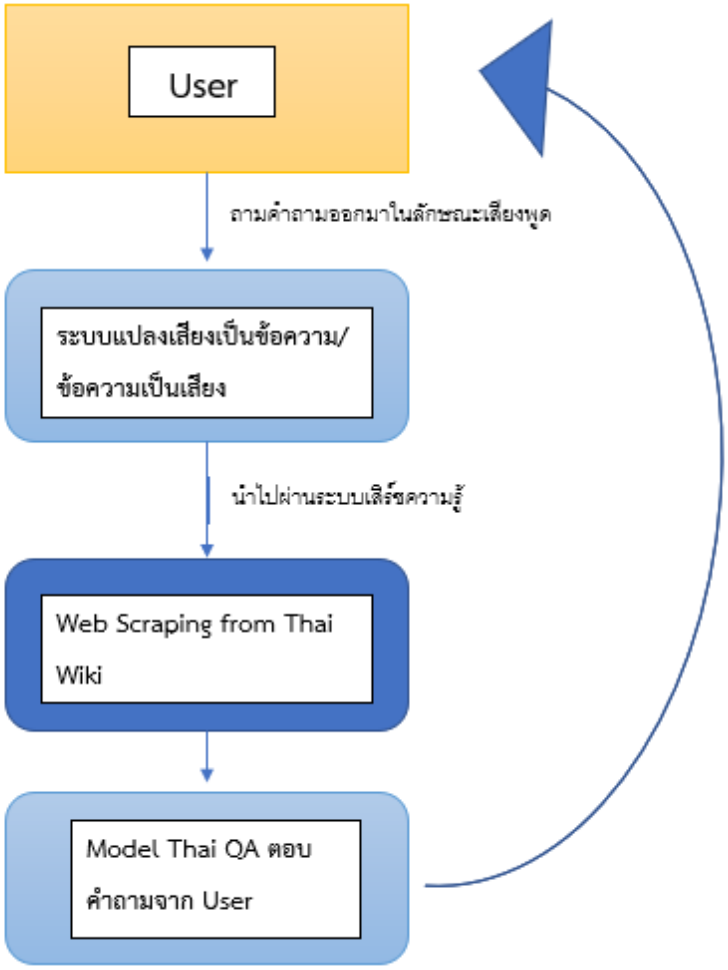
เมื่อตัดประโยคออกมา จะเห็นได้ว่า มีลักษณะเป็น Sequence ที่มีความยาวคงที่ โดยหากส่วนไหนที่มีค่าว่าง ก็จะมีการเพิ่มโทเคนพิเศษไป เรียกว่าการทำ padding เพื่อให้ได้ Vector ที่มีขนาดเท่ากันเป็น Input รวมไปถึงใส่ tag <s> และ </s> เพื่อระบุความสัมพันธ์ของสิ่งที่อยู่ใน Sequence ว่าส่วนไหนคือส่วนของคำถาม ส่วนไหนคือส่วนของคำตอบ

จากนั้นจึง ทำการฝึกฝนทั้งหมด 10 รอบ เพื่อที่จะทดสอบว่า ด้วยความยาวของ sequence ที่ถูกตัดมาด้วย max length ที่แตกต่างกัน จะทำให้ประสิทธิภาพของโมเดลนั้นมีความแตกต่างกันมากน้อยแค่ไหน ทั้งในแง่ของความถูกต้องของการตอบคำถาม และในแง่ของเวลาที่ใช้ในการฝึกฝน โดยการทดสอบ จะใช้ ข้อมูล ฝึกฝน ทั้งหมด 4000 ข้อความถาม-ตอบ รวมถึง ข้อมูล

ทดสอบ แบบที่โมเดลไม่เคยเห็นมาก่อน ทั้งหมด 74 ข้อความถาม-ตอบ โดยการฝึกฝนนั้น จะฝึกฝนอยู่บน google- colab เนื่องมาจากเป็น GPU ที่มีความเร็วและมีหน่วยความจำมากกว่า 128 GB ทำให้สามารถฝึกฝนโมเดลขนาดใหญ่อย่าง transformers ได้ และใช้เวลาที่ไม่นานมากจนเกินไป อันเนื่องมาจากโมเดลที่มีขนาดใหญ่มาก ทำให้หากฝึกฝนในคอมพิวเตอร์ทั่วไปอาจใช้เวลาที่นานเกินไป

3.3 การออกแบบระบบ

ภาพรวมของระบบการทำงานนั้น ก็จะมีขั้นตอนการทำงานดังนี้



ภาพที่ 3.2 ภาพรวมของระบบการทำงาน

โมเดล Thai QA รับคำถามมาจากผู้ใช้งาน (แบบเป็นเสียงพูด) จากนั้นจึงแปลงเสียงพูดเป็นข้อความ Text และนำข้อความที่ได้มาหา Keyword และนำ Keyword ไปเสิร์ชข้อมูลมาจาก

Thai wiki จากนั้นจึงนำข้อความที่ได้มาเข้าโมเดลพร้อมคำถาม และส่วนของโมเดลก็จะส่งคำตอบที่มีความน่าจะเป็นมากที่สุดออกมาเป็นข้อความ Text และถูกนำไปผ่านระบบแปลงข้อความเป็นเสียงพูด เพื่อแปลงข้อความนั้นตอบกลับผู้ใช้งานกลับไปในรูปแบบเสียงพูด หลังจากนำโมดูลการทำงานทั้งหมดมารวมกัน ก็จะสามารถสร้าง Chat bot ถามตอบคำถามภาษาไทยแบบอัตโนมัติขึ้นมาได้ เป็นต้นแบบ (Prototype) สำหรับการพัฒนาต่อยอดต่อไปในอนาคต

ซึ่งในการทำงานจะมีส่วนการใช้งานเป็น API ใน Google Colab โดยใช้ FastAPI ซึ่งการทำงานในส่วนของ API นั้น จะเป็นในส่วนที่มีการประมวลผลมาก เนื่องจากคอมพิวเตอร์ทั่วไปที่มีแค่ CPU นั้น ไม่สามารถทำงานกับ โมเดลขนาดใหญ่และมีการคำนวณที่ซับซ้อนได้ดี เพราะต้องใช้เวลาที่นานมาก แต่เนื่องจาก Google Colab นั้นมีการรันอยู่บน server ที่เป็น GPU ทำให้สามารถรันโมเดลขนาดใหญ่ได้อย่างรวดเร็ว อีกทั้งโมเดลที่เรานำมาใช้งานนั้น ยังมี base การทำงานอยู่บนพื้นฐานของ GPU ทำให้เหมาะแก่การนำมาทำเป็น API มากกว่าจะนำไปใช้งานบนเครื่องคอมพิวเตอร์แบบทั่วไป

3.3.1 ออกแบบระบบแปลงเสียงเป็นข้อความ/ข้อความเป็นเสียง

เทคโนโลยีรู้จำเสียงพูด (Automatic Speech Recognition: ASR) เป็นสาขาย่อยของ วิชาภาษาศาสตร์คอมพิวเตอร์ที่พัฒนาวิธีการและเทคโนโลยีที่ช่วยให้การรับรู้และการแปลภาษาพูดเป็นข้อความโดยคอมพิวเตอร์ ซอฟต์แวร์รู้จำเสียงพูดขั้นพื้นฐานมีคำศัพท์ที่จำกัด องคำและวลีและอาจระบุสิ่งที่พูดอย่างชัดเจน ซึ่งประเภทของระบบรู้จำเสียงพูดสามารถแบ่งได้ เป็น 3 ประเภท เช่น เทคโนโลยีรู้จำเสียงพูดแบบคำโดด (Isolated speech) คือระบบที่รู้จำคำสั้นๆเพียงไม่กี่คำสั่ง เพื่อให้ระบบบสามารถตอบโต้ได้อย่างรวดเร็ว, เทคโนโลยีรู้จำเสียงพูดแบบต่อเนื่อง (Continuous speech) คือระบบรู้จำคำจากเสียงอย่างต่อเนื่อง แล้วทำการพิจารณาตัดเสียงพูด, เทคโนโลยีรู้จำที่จำเสียงเพียงบ้างส่วน (Spontaneous speech) คือระบบที่จดจำเสียงที่ตรวจหาคำสำคัญเพียงคำเดียวในประโยคเพื่อหาใจความสำคัญ

Speech Recognition คือระบบโปรแกรมคอมพิวเตอร์ที่สามารถแปลงเสียงพูด (Audio File) เป็นข้อความตัวอักษร (Text) โดยสามารถแจกแจงคำพูดต่างๆ ที่มนุษย์สามารถพูดใส่ไมโครโฟน โทรศัพท์หรืออุปกรณ์อื่นๆ และเข้าใจคำศัพท์ทุกคำอย่างถูกต้องเกือบ 100% โดยเป็นอิสระจากขนาดของกลุ่มคำศัพท์ ความดังของเสียงและลักษณะการออกเสียงของผู้พูด โดยระบบจะรับฟังเสียงพูดและตัดสินใจว่าเสียงที่ได้ยินนั้นเป็นคำๆใด เทคโนโลยีที่เป็นส่วนสำคัญในการทำ ASR เรียกว่า Hidden Markov Model (HMM) เทคโนโลยีชนิดนี้สามารถที่จะเข้าใจคำพูดจากการจำแนกความแตกต่างและการประมาณการถึงความเป็นไปได้ของส่วนประกอบของหน่วยที่เป็นพื้นฐานของเสียงที่อยู่ติดๆกัน โดยอาศัยหลักการที่ว่าเสียงแต่ละเสียงจะมีขอบเขตของสัญญาณและลักษณะเฉพาะที่มีความแตกต่างกัน

บทบาทของเทคโนโลยีการรู้จำเสียงพูดที่สำคัญในปัจจุบัน คือ เป็นตัวเชื่อมประสานกับผู้ใช้งาน (User Interface) ซึ่งอำนวยความสะดวกในการติดต่อระหว่างมนุษย์กับคอมพิวเตอร์ ขณะที่มือไม่ว่าง ต้องการความคล่องตัว สายตาไม่ว่าง ไม่ต้องการใช้คีย์บอร์ด ทัศนวิสัยไม่ดี มีข้อจำกัดด้านร่างกาย ฯลฯ ตัวอย่างเช่น รถเข็นคนพิการควบคุมด้วยระบบรู้จำเสียงพูด ระบบรู้จำเสียงพูด (Speech Recognition) ใช้ในการควบคุมรถเข็นคนพิการให้เคลื่อนที่ไปในทิศทางต่างๆ โดยกำหนดด้วยคำสั่ง 9 คำสั่ง ประกอบด้วยคำว่า เดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา กึ่งซ้าย กึ่งขวา เร็วขึ้น ช้าลง และหยุด ซึ่งจะเป็นคำสั่งที่ใช้เป็นสัญญาณอินพุตเข้าสู่ระบบ และระบบก็จะประมวลผลตัดสินใจและส่งค่าเอาท์พุต ออกไปควบคุมมอเตอร์เพื่อเคลื่อนรถเข็นคนพิการในทิศทางที่สั่ง องค์ประกอบหลักๆ ของระบบรู้จำเสียงพูดแบ่งได้เป็น 3 ขั้นตอนดังนี้

- การเตรียมสัญญาณขั้นต้น (Preprocessing) เป็นขั้นตอนที่จะทำให้สัญญาณเสียงที่จะนำไปใช้ หรือรับเข้ามานั้น มีความสมบูรณ์มากที่สุด โดยจะทำการกำจัดสัญญาณรบกวน (Noise) และตัดส่วนที่ไม่ใช่สัญญาณเสียง (Unvoice) ออกซึ่งจะเหลือแต่เพียง ช่วงที่เป็นข้อมูลเสียง

- การหาลักษณะสำคัญของเสียง (Feature Extraction) เป็นขั้นตอนที่ใช้สำหรับหาองค์ประกอบสำคัญต่างๆ ของเสียงแต่ละเสียงที่รับเข้ามา ให้รู้ว่าคำแต่ละคำนั้นมีลักษณะเด่นอย่างไร

- การรู้จำเสียงพูด (Speech Recognition) เป็นขั้นตอนที่ให้ระบบทำการเรียนรู้โดยการนำสัญญาณเสียงเข้าสู่ระบบโครงข่ายประสาทเทียม (Neural Network System) เพื่อระบบจะทำการตัดสินใจ และให้ผลลัพธ์ตามสัญญาณเสียงที่แตกต่างกันได้ถูกต้อง ซึ่งในกรณีของการนำมาใช้งานในครั้งนี้ ผู้จัดทำได้เลือกใช้ library python สำหรับแปลงเสียงพูดให้กลายเป็นข้อความได้ โดยมีชื่อว่า SpeechRecognition ซึ่งเป็น library ที่สามารถเรียกใช้งาน speechrecognition ได้ฟรีและอีกทั้งยังมีความแม่นยำสูงอีกด้วย สามารถใช้งานได้หลายภาษาและหนึ่งในนั้นก็คือภาษาไทย ผู้จัดทำจึงได้นำมาต่อยอดโดยการ นำมาแปลงเสียงพูด ในการถามคำถามของผู้ใช้งาน มาเป็นข้อความเพื่อที่จะสามารถนำไปใช้งานกับระบบ เสิร์ชความรู้ต่อไปได้ และเมื่อกระบวนการทำงานทั้งหมดเสร็จสิ้นแล้ว โมเดลจะได้ผลลัพธ์ออกมาเป็น ข้อความจากนั้นจะใช้เทคโนโลยี TextToSpeech library อีกตัวก็คือ gtts ในการช่วยแปลงข้อความเป็นเสียงพูด ซึ่งเป็น API ฟรีจาก google สามารถทำงานได้เร็วอีกทั้งยังมีความแม่นยำสูงมาก แม้จะใช้งานกับภาษาไทย ผู้จัดทำจึงได้เลือก gtts มาใช้สำหรับแปลงข้อความตอบกลับจากโมเดลมาแปลงเป็นเสียงพูด เพื่อตอบกลับคำถามของผู้ใช้งานกลับด้วยคำตอบที่เป็นเสียงพูดได้

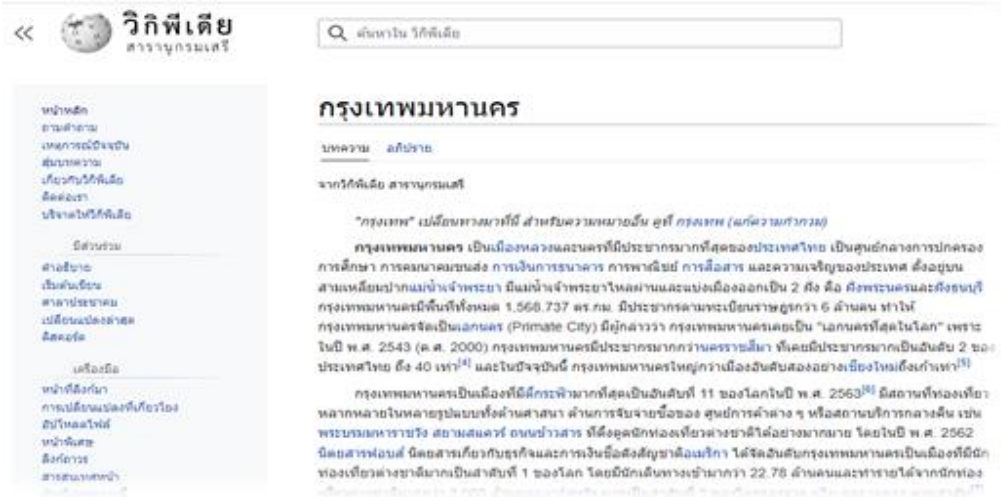
3.3.2 ระบบ Web Scraping Thai Wiki

web Scraping เป็นเทคนิคการรวบรวมข้อมูลบน Internet แบบอัตโนมัติ สามารถแบ่งออกได้เป็น 2 รูปแบบ คือ API Scraping และ web Scraping เทคนิคการรวบรวมข้อมูลมักจะอยู่ในรูปแบบที่มีการตกลงกันระหว่างผู้ส่งกับผู้รับข้อมูล หรือ Server ที่ให้บริการ web service นั้นๆ ที่สามารถติดต่อกับฐานข้อมูลได้ โดยส่วนใหญ่แล้ว จะใช้ protocol เช่น SOAP, XML, JSON และ HTML โดยส่วนใหญ่ในการรวบรวมข้อมูลจะขึ้นกับการอนุญาตการให้บริการข้อมูลที่สามารถเข้าถึงได้ ทั้งวิธีการลงทะเบียน การเข้าสู่ระบบ จนไปถึง Public Data หรือ OpenData โดยในรูปแบบนี้ จะใช้เทคนิค API Scraping เพื่อดึงข้อมูลมาทำอะไรบ้างนั้น ขึ้นอยู่กับผู้พัฒนา อาจจะนำมาเก็บเอาไว้ เพื่อนำไปวิเคราะห์ในภายหลัง หรือ จะแสดงผลบนแอปพลิเคชัน ซึ่งหากไม่สามารถเข้าถึงได้ผ่าน A อาจจะด้วยเหตุผล ไม่มีผู้พัฒนา Service นี้ให้ได้ เพราะจะต้องใช้งบประมาณในการพัฒนา มีแต่หน้าเว็บที่มีข้อมูลอยู่แล้วก็จะไปใช้เทคนิคการทำด้วย web Scraping ผ่านหน้าเว็บแทน ซึ่งความรู้อย่างหนึ่งที่สำคัญมากสำหรับการทำ web scraping ก็คือ โครงสร้างของ HTML หลายๆ คนที่เคยทำ เว็บไซต์หรือเว็บเพจก็จะสามารถมองภาพออกได้ว่า HTML เปรียบได้กับเหมือนกระดาษหนึ่งแผ่นที่ต้องมีข้อมูลบ่งบอกว่า นี้คือ เว็บไซต์ และต้องมีโครงสร้างตามด้านล่างนี้

```
<html>
   <head>
   <title> พื้นฐานในการทำ Web Scraping </title> </head>
   <body>
      <p id="author"> Com Science  </p>
      <p id="subject">พื้นฐานในการทำ Web Scraping </p>
   </body>
</html>
```

ซึ่งการจะดึงข้อมูลออกมาได้ จำเป็นจะต้องมความเข้าใจในโครงสร้าง HTML เหล่านี้เสียก่อน เพื่อที่จะสามารถเข้าถึง Tag ที่มีข้อมูลที่เราต้องการอยู่ และสามารถที่จะดึงออกมาได้อย่างถูกต้องและครบถ้วน โดยไม่มีข้อมูลอื่นๆที่เราไม่ต้องการมารวมอยู่ด้วย ในขั้นตอนนี้ เราจะใช้ความรู้ในการดึงข้อมูลจากเว็บไซต์ หรือการทำ Web scraping ในการช่วยดึงข้อมูลออกมาจาก Thai wiki โดยการใช้งาน Library BeautifulSoup ซึ่งข้อมูลที่ได้มานั้น โดยปกติจะเยอะมาก ซึ่งมันมากเกินความจำเป็น เกินขอบข่ายเนื้อหาไปมากโดยส่วนใหญ่ จึงได้มีการตัดเอาเฉพาะ 4000 ตัวอักษรแรกจากบทความมาเท่านั้น เพราะหากบทความที่นำเข้าโมเดลมีความยาวมากเกินไป ก็มีผลให้คำตอบที่ได้ออกมาแย่เช่นกัน และการทำงานของระบบ Web scraping นั้น จะอาศัย Keyword ในการเสิร์ช

ไปที่ Thai wiki ซึ่งยังคงมีปัญหาอยู่หลายอย่างเช่น คำที่ค้นหานั้น เป็นคำที่มีความเฉพาะตัวมาก ซึ่งส่วนมากนั้นจะเป็นภาษาเขียนมากกว่าภาษาพูด ดังนั้นการนำคำที่เป็นภาษาพูดเหล่านี้ไปเป็น Key word เพื่อที่จะค้นหาเนื้อหาที่เกี่ยวข้องกับหัวเรื่องนั้นๆอาจจะทำได้ไม่ดี และไม่มีความแม่นยำที่มากพอ อีกทั้งยังมีปัญหาเกี่ยวกับในเรื่องของการค้นหาชื่อบุคคล ซึ่งต้องมีการเว้นวรรค รวมไปถึงใส _ คั่นไว้ระหว่างชื่อและนามสกุลอีกด้วย ซึ่งนั่นทำให้การค้นหาชื่อบุคคลนั้นทำได้ยาก เนื่องด้วยยังไม่มีโมเดลที่สามารถแยก ชื่อ-นามสกุล ให้ได้ เมื่อเราต้องการจะนำชื่อที่ถูกพูดออกมาจากคำถามของผู้ใช้งาน ซึ่งยังเป็นปัญหาที่ยังหาทางแก้ไม่ได้ จึงต้องรอพัฒนาในส่วนการทำงานนี้ต่อไปในอนาคต



ภาพที่ 3.3 URL สำหรับเสิร์ชหาหน้าข้อมูลที่จะ Scraping

จากภาพด้านบนจะเห็นได้ว่า หากเราต้องการข้อมูลกรุงเทพมหานคร เราจำเป็นที่จะต้องมีคีย์เวิร์ด คือคำว่า ‘กรุงเทพมหานคร’ เสียก่อน เพื่อที่จะนำมาเสิร์ชหาหน้าข้อมูลของกรุงเทพมหานคร ที่เราต้องการจะดึงข้อมูลไปได้ ซึ่งหากเราต้องการถามคำถามเกี่ยวกับอะไร เราจำเป็นที่จะต้องบอกกับระบบให้ทราบก่อนว่า ต้องการจะรู้เรื่องอะไร เพื่อใช้คีย์เวิร์ดเรื่องนั้นๆไปเสิร์ชข้อมูลมาได้ แต่การทำแบบนั้นอาจทำให้การใช้งานไม่ค่อยสะดวกสบาย จึงได้มีการพัฒนาต่อยอดระบบเสิร์ชความรู้แบบอัตโนมัติขึ้นมา

3.3.3 ออกแบบระบบเสิร์ชความรู้

จากปัญหาของการพยายามหา Keyword ข้างต้น ทำให้เกิดไอเดียในการพัฒนาระบบการเสิร์ชความรู้ขึ้นมา โดยใสตอนแรกสุดนั้น ผู้ใช้งานจะถามคำถามบางอย่างมากับระบบ เช่นถามว่า “กรุงเทพมีประชากรเท่าไหร่?” หากสังเกตให้ดีก็จะพบว่า มี Keyword ที่จำเป็นต้องใช้ในการเสิร์ช อยู่ในตัวคำถามเองอยู่แล้ว จึงได้เกิดแนวคิดที่จะแยก Keyword นั้นออกมาจากข้อความ

ปกติ วิธีการคือการใช้ Pythainlp ในการตัดคำและช่วย Tag ประเภทของคำในประโยคคำถามนั้นๆ ให้ ออกมาเป็นประเภทของคำต่างๆ จากนั้นจึงเลือกเอาเฉพาะส่วนที่เป็น คำนาม (Noun) และ สรรพนาม (Pronoun) มาใช้ในการเสิร์ชข้อมูลต่อไป ซึ่งหากเป็นกรณีของคำถามที่ว่า “กรุงเทพมีประชากรเท่าไหร่?” ก็จะได้คำและประเภทของคำดังนี้

[('กรุงเทพฯ', 'PROPN', 'B-LOCATION'), ('มี', 'VERB', 'O'), ('พื้นที่', 'NOUN', 'O'), ('เท่าไหร่', 'NOUN', 'O')]

เมื่อนำมาหา Keyword ก็จะพบว่ามีสองคำคือคำว่า ‘กรุงเทพ’ และคำว่า ’พื้นที่’ เมื่อนำไปเสิร์ชเราจะได้ข้อมูลของกรุงเทพ คือข้อมูลที่ถูกต้องจริงๆ กับข้อมูลของพื้นที่ ซึ่งเป็นอีกเรื่องที่ไม่เกี่ยวข้องกัน

Algorithm การหา keyword และ เสิร์ชข้อมูล

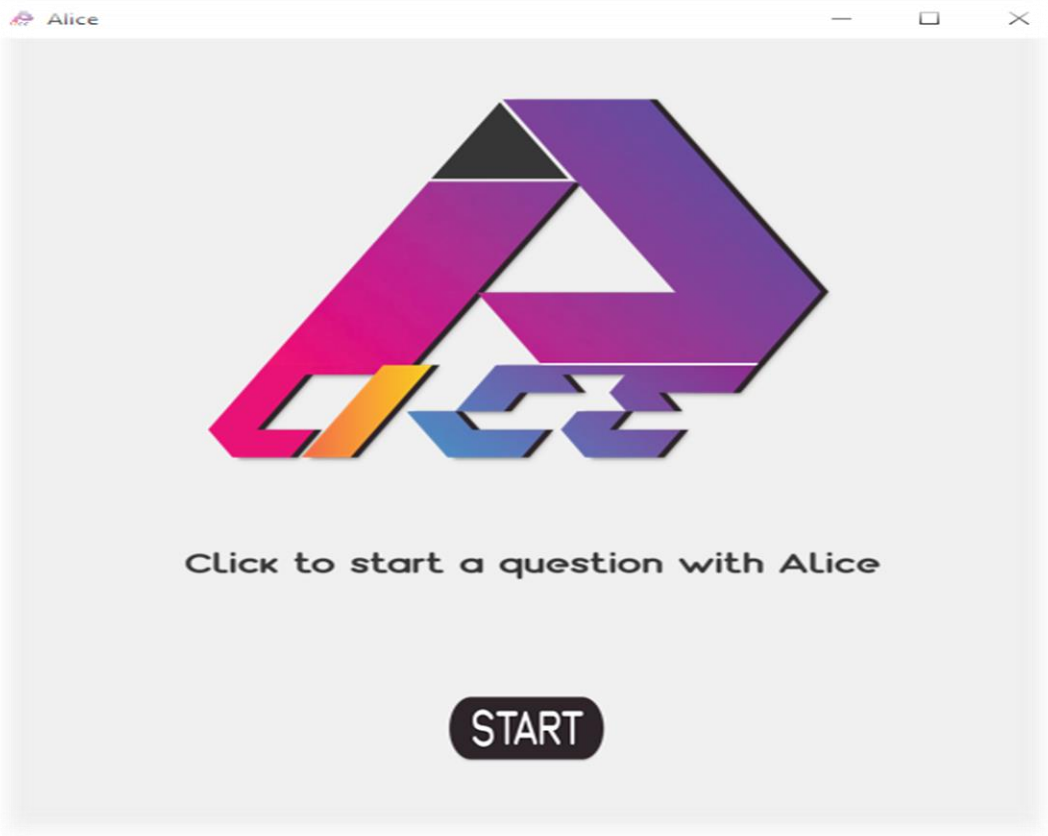
```
Input: question sentence
list  x =  tokenizer(input)
list result = []
function search(word) {
        x = scraping word
        for all <p> in x
                out = out + <p>
        return out
}
for all t from x do
        if t == NOUN or PRON
                result.append(search(t))
        return result
end
```

จากนั้นจึงนำข้อความทั้งสองนั้นไปเข้าในโมเดล และเลือกคำตอบที่มีความน่าจะเป็น สูงที่สุด กลับออกมาเป็นคำตอบที่จะถูกตอบกลับผู้ใช้งานกลับไป โดยในส่วนของ output ของโมเดจะประกอบไปด้วยสองส่วนคือ ส่วนที่เป็น answer คือ ส่วนที่เป็นคำตอบของคำถามที่โมเดลคิดว่าถูกต้อง ซึ่งจะอยู่ในรูปของข้อความ และอีกส่วนคือ probability คือ ความน่าจะเป็นที่

ข้อความเหล่านั้นจะถูก โดยทั้งสองส่วนนี้จะถูกเก็บไว้ใน list คนละ list โดยจะอิง ความน่าจะเป็นเชื่อมโยงหับตัวของข้อความคำตอบ

3.4 การออกแบบหน้าจอ

ลักษณะของหน้าจะ จะมีสัญลักษณ์ของไอคอน ชื่อว่า “Alice” ซึ่งไอคอนนี้มีความพิเศษตรงที่สามารถอ่านเป็น “Alice” หรือ “AI” ก็ได้ ขึ้นอยู่กับวิธีมอง จากนั้นทางด้านล่างไอคอน ก็จะมีปุ่มกดที่สามารถกดเพื่อเริ่มการทำงานได้ ซึ่งการกด 1 ก็จะสามารถามคำถามอะไรก็ได้กับ “Alice” ได้ 1 คำถาม โดยตัวโปรแกรมจะทำการเปิดไมค์ และส่งเสียง สวัสดีผู้ใช้งาน จากนั้นจึงแจ้งให้ผู้ใช้งานทราบว่า สามารถถามคำถามอะไรก็ได้ เมื่อผู้ใช้พูดคำถามกลับมา ก็จะมีการบันทึกเสียงไว้ จากนั้นจึงแปลงเสียงเหล่านั้นเป็น Text แล้วส่งไปยัง Request ของ API ที่เราเขียนไว้ใน Google colab ซึ่งจะรอรับ Text เหล่านั้น สำหรับไปประมวลผลต่อไป ในการแยกประเภทของคำ จัดการหา keyword และทำการ เสิร์ชข้อมูลออกมา เพื่อที่จะนำข้อมูลเหล่านั้นไปผ่านเข้าโมเดล XLMRoBERTa ที่ผ่านการฝึกฝนมาด้วยความยาว Max length = 400, stride = 128 จำนวนทั้งหมด 10 รอบ โดยโมเดลนี้จะได้รับข้อความเหล่านั้นที่เสิร์ชมาจากทุกๆ Keyword เข้าไปในโมเดลพร้อมด้วยคำถามที่ถูกถามมา จากนั้น จึงรันเอาผลลัพธ์จากการถามข้อความเหล่านั้นออกมาอยู่ในรูปแบบ List ของคำตอบและความน่าจะเป็น ซึ่งเราจะนำมาเลือกตัวโมเดลที่มีความน่าจะเป็นมากที่สุดมาใช้สำหรับเป็นคำตอบตอบกลับไปที่ผู้ใช้งาน โดยกระบวนการทำงานนั้น เริ่มแรกจะมีปุ่ม start อยู่ที่ด้านหน้าของหน้าจอโปรแกรม เมื่อผู้ใช้งานกด Alice จะทักทายผู้ใช้งานก่อนว่า “หนูคือ Alice หากมีอะไรที่สงสัยสามารถถาม Alice ได้เลยค่ะ” จากนั้นเมื่อ Alice พูดจบ ก็จะสามารถเริ่มถามคำถามกับ Alice ได้ในทันที ซึ่งหาก Alice สามารถหาคำตอบมาได้โดยไม่มีข้อผิดพลาด Alice ก็จะตอบคำถามเหล่านั้นกลับมาเป็นเสียงพูด แต่หาก Alice ติดปัญหา หรือไม่สามารถหาคำตอบเกี่ยวกับคำถามนั้นๆได้ Alice ก็จะตอบกลับผู้ใช้งานกลับมาว่า “ขออภัยค่ะ Alice ไม่สามารถหาคำตอบได้ค่ะ”



ภาพที่ 3.4 หน้าจอของโปรแกรม

3.5 การวัดประสิทธิภาพ

การวัดประสิทธิภาพของโมเดล จะใช้ด้วยกันทั้งหมดสองส่วนได้แก่

1) ผลลัพธ์การถามตอบคำถามภาษาไทย คือหาอัตราความถูกต้องของการตอบคำถามว่าสามารถตอบถูกต้องได้ทั้งหมดกี่เปอร์เซ็นต์ โดยคิดจาก คำตอบที่ถูกต้องทุกตัวอักษร (correct) คำตอบที่มีส่วนหนึ่งมากกว่า 80% อยู่ในคำตอบที่ถูกต้อง (similarly) และคำตอบที่ผิด (incorrect) โดยมีวิธีการคำนวณคือ

$$\text{อัตราความถูกต้อง\%} = \frac{\text{คำตอบที่ถูกต้องทั้งหมด} + \text{คำตอบที่คล้ายคำตอบที่ถูกต้อง}}{\text{จำนวนคำตอบทั้งหมด}}$$

โดยจะมีการทดสอบกับ ชุดข้อมูลที่ใช้ในการฝึกฝนโมเดลทั้งหมด 4000 ข้อมูลคำถาม และชุดข้อมูลสำหรับทดสอบที่โมเดลไม่เคยได้เรียนรู้มาก่อนทั้งหมด 74 ข้อความคำถาม

2) ผลลัพธ์จากการตรวจสอบคำตอบทั้ง 3 แบบ ได้แก่ correct, similar, incorrect

บทที่ 4
ผลการดำเนินการ

สำหรับผลการดำเนินงานการพัฒนาระบบสำหรับเสิร์ชความรู้อัตโนมัติด้วยการประมวลภาษาธรรมชาติ สำหรับการตอบคำถามภาษาไทย แบ่งออกได้เป็น

4.1 ผลลัพธ์การเตรียมข้อมูล

4.1.1 การจัดการข้อความ

4.1.2 การแปลงข้อความเป็น Vector

4.2 ผลการวัดประสิทธิภาพโมเดล

4.3 ผลการพัฒนา Application สำหรับการถามตอบอัตโนมัติ

4.4 ผลการทดสอบใช้งานระบบเสิร์ชความรู้

4.1 ผลลัพธ์การเตรียมข้อมูล

ชุดข้อมูลจะประกอบไปด้วย question_id, article_id, context, question, answer, answers โดยในส่วนของ answers นั้นจะมีองค์ประกอบคือ Answer_begin_position หมายถึง ตำแหน่งแรกของอักษร, Answer_end_position หมายถึงตำแหน่งสุดท้ายของอักษร, Answer หมายถึง คำตอบ ที่เป็นลักษณะข้อความ

โดยในส่วนของ ชุดข้อมูลฝึกฝนนั้น ก็จะมีส่วนของข้อมูลฝึกฝนกับ ข้อมูลทดสอบโดย ชุดข้อมูลฝึกฝนนั้นมีทั้งหมด 4000 แถว ส่วน ชุดข้อมูลทดสอบนั้นมีทั้งหมด 74 แถว ซึ่งข้อมูล Context ทั้งหมดนั้น ได้มาจาก Thai Wiki โดยการดึงข้อมูลออกมาจากเว็บไซต์ เพื่อนำมาสร้างไว้เป็น ชุดข้อมูลฝึกฝน ทั่วไปสำหรับผู้ที่ต้องการพัฒนาระบบ ถามตอบภาษาไทย สามารถที่จะโหลดไปทดสอบหรือฝึกฝนโมเดลได้อย่างทันที ซึ่งถูกพัฒนาขึ้นมาโดย NECTEC โดยในขั้นตอนก่อนการฝึกฝนนั้น จะมีการจัดการกับ ชุดข้อมูลฝึกฝน ก่อน เนื่องมาจากว่า ตัวของ ชุดข้อมูลฝึกฝน นั้น จะมีส่วนของ Context ถูกดึงออกมาจาก Thai Wiki นั้นจะมี Tag html อยู่ในตัวของ Context ด้วย ดังนั้นก่อนที่จะนำไปฝึกฝนนั้น จึงต้องมีการตัดเอา Tag เหล่านั้นออกเสียก่อน

4.1.1 การจัดการข้อความ

- ข้อความก่อนตัด Tag html

“ <doc id="115035" url=https://th.wikipedia.org/wiki?curid=115035 title="เบนจี้">เบนจี้ เบนจี้ () เป็นชื่อตัวละครหมาพันทางแสนรู้ ที่ปรากฏอยู่ในภาพยนตร์หลายเรื่องที่เขียนบท... </doc>“

- ข้อความหลังตัด Tag html

“ เบนจี้ เบนจี้ () เป็นชื่อตัวละครหมาพันทางแสนรู้ ที่ปรากฏอยู่ในภาพยนตร์หลายเรื่องที่เขียนบท...”

- ข้อความหลังลบช่องว่าง

“เบนจี้เบนจี้()เป็นชื่อตัวละครหมาพันทางแสนรู้ที่ปรากฏอยู่ในภาพยนตร์หลายเรื่องที่เขียนบท”

4.1.2 การแปลงข้อความเป็น Vector

ในส่วนนี้จะเรียกว่าการทำ Input tokenizer โดยใช้ Pretain-XLMRoBERTa ที่โหลดมาในก่อนหน้ามาช่วยในการแปลงข้อความเหล่านี้ให้เป็นเวกเตอร์ จะมีผลลัพธ์ออกมาเป็นชุดของ Vector ที่มีองค์ประกอบ 4 ส่วน ได้แก่ Iput_ids, Attention_mask, Offset_mapping, Overflow_sample_to_mapping เมื่อได้ Vector เหล่านี้มาแล้ว มันจะอยู่ในรูปแบบของ Encoder ของ Tokenizer ที่ถูก Pretrain มาก่อนแล้ว การจะอ่านข้อความ Sequence เหล่านี้ได้จะต้องทำการ Decoder ออกมาสียก่อน จึงจะกลับจาก Vector มาเป็นข้อความตัวอักษร ซึ่งสามารถอ่านเข้าใจได้ ในส่วนที่เป็น Represent ของข้อความจะอยู่ในส่วนของ Input_ids ซึ่ง เมื่อเราได้ทดสอบ Decode ออกมาแล้ว ก็จะเห็นเป็น Sequence ของข้อความที่มีความยาวตาม Max_length ที่เราได้กำหนดไว้ และเป็นภาษาที่เราสามารถอ่านเข้าใจได้ โดยจะมีองค์ประกอบของ Tag คือ <s> หมายถึง เริ่มต้นประโยคและ </s> หมายถึง Tag ที่ใช้สำหรับการคั่นประโยค เพื่อแบ่งว่า Sequence ใหนเป็น Sequence ใหน เพื่อให้โมเดลสามารถที่จะรับข้อมูลเข้าไปฝึกฝนได้อย่างถูกต้อง ตามลำดับ โดยจะมีผลลัพธ์ดังนี้

<s> สุนัขตัวแรกรับบทเป็นเบนจี้ในภาพยนตร์เรื่อง Benji ที่ออกฉายในปี พ.ศ. 2517 มีชื่อว่าอะไร </s></s> เบนจี้เบนจี้()เป็นชื่อตัวละครหมาพันทางแสนรู้ที่ปรากฏอยู่ในภาพยนตร์หลายเรื่องที่เขียนบทและกำกับโดยโจแคมป์ในช่วงทศวรรษ1970ถึง1980ภาพยนตร์เรื่องแรกในชุดใช้ชื่อเรื่องว่าเบนจี้ เช่นเดียวกับตัวละครถ่ายทำที่เมืองดัลลัสรัฐเทก</s>

จากนั้น หากเรานำ Inputs ที่ผ่าน Tokenizer มาตรวจสอบดูองค์ประกอบดู จะพบว่า มีโครงสร้างข้อมูลเป็น Dictionary ซึ่งมี Key ทั้งหมด 4 อัน ได้แก่

Input_ids คือการแปลงคำและ Tag ต่างๆให้อยู่ในรูปของตัวเลขบางอย่าง ที่สามารถ Represent ข้อความที่ถูกตัดออกกมาเป็น Sequence ซึ่งตัวเลขเหล่านี้ คือส่วนที่ถูก Encoder เอาไว้ของข้อความ Context โดย Tokenizer คือสิ่งที่สามารถ ช่วยให้เราแปลงข้อความเหล่านี้ให้เป็น Vector ได้ โดยที่ มีการทำ Pretrain มาแล้วในก่อนหน้านี้ เมื่อเรียกใช้ Tokenizer ตัวเดิมกลับมา จึงสามารถที่จะ Encode หรือ Decode ข้อมูลได้อย่างถูกต้องตามเดิม ตามที่โมเดลนั้นถูกฝึกฝนมา โดยมีลักษณะเป็น Array ของ List ซึ่งแต่ละ List นั้นก็หมายถึงแต่ละประโยคที่ตัดมาในก่อนหน้านี้นั่นเอง โดยมีตัวอย่าง ดังนี้

([0, 6, 110860, 4214, 21573, 8448, 35266, 1805, 206040, 46924, 8181, 1201, 80043, 10650, 3419, 658, 5261, 9562, 136964, 71923, 11308, 5, 7322, 5, 714, 2489, 5451, 176511, 16697, 2, 2, 6, 206040, 46924, 8181, 6, 206040, 46924, 8181, 15, 16, 7526,]),

Attention_mask คือการใส่ Tag ให้กับ Sequence ที่ถูกตัดออกมาเหล่านั้น โดยเลข 1 จะหมายถึงคำที่รู้จัก ส่วนอักขระพิเศษ หรือคำที่ไม่รู้จักจะถูกแทนด้วยเลข 0 เนื่องจาก Pretrain ที่ถูกฝึกฝนมาก่อนหน้า นั้นได้มีการฝึกฝนด้วยภาษาไทยมาแล้ว จึงทำให้สวนใหญ่ พบเป็นเลข 1 ทั้งหมด

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Offset_mapping คือตัวเลขของการเชื่อมแต่ละคำในข้อความนั้นๆ ที่ถูกตัดออกมาเป็น Sequence โดย จะเป็นชุดตัวเลขในวงเล็บ ตัวเลขแรกนั้นหมายถึง ตัวอักษรแรกของคำนั้นๆ ส่วนตัวเลขหลังคือตัวอักษรสุดท้ายของคำนั้นๆ โดยในส่วนของ Tag แรกสุดที่เป็น <s> นั้นจะถูกแทนด้วยค่า (0,0) และค่าช่องว่างถัดมาจาก Tag แรกก็จะถูกแทนด้วยค่า (0,1) และหลังจากนั้นไปก็จะแทนตัวเลขตัวเลขในวงเล็บด้วย Index ของตัวอักษรแรก และตัวเลขตัวที่สองในวงเล็บ ก็จะถูกแทนด้วย Index ของตัวอักษรตัวสุดท้าย ของคำนั้นๆ

[(0, 0), (0, 1), (0, 5), (5, 8), (8, 11), (11, 14), (14, 16), (16, 20), (20, 23), (23, 25), (25, 26), (26, 28), (28, 36), (36, 42), (43, 46), (46, 48), (49, 52), (52, 55), (55, 58), (58, 62), (63, 64), (64, 65), (65, 66), (66, 67), (68, 70), (70, 72), (73, 75), (75, 82), (82, 86), (0, 0), (0, 0), (0,)]

Overflow_to_sample_mapping หมายถึง จำนวนของ Sequence ที่ตัดออกมาได้ทั้งหมดตามจำนวนของ Max length และ Stride โดยจะถูกแทนแต่ละ Sequence ด้วยตัวเลข 0 โดยจำนวนของลข 0 ที่ปรากฏใน list นั้นก็คือจำนวนทั้งหมดของ Sequence

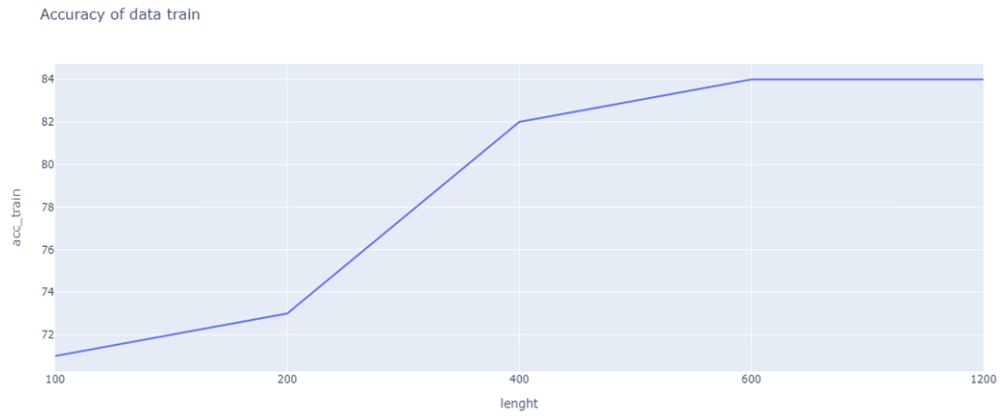
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])

เมื่อนำข้อมูลมาตัดเป็น Sequence เสร็จแล้ว ขั้นต่อไปคือการหาว่า ในแต่ละ Sequence ที่ถูกตัดออกมานั้น มี Sequence ใหนบ้าง ที่มีคำตอบของคำถามอยู่ในส่วนหนึ่งของข้อความ Sequence นั้นๆ โดยการเขียนฟังก์ชั่นมาเก็บค่าเหล่านั้นไว้ใน List ที่แสดงถึง Start Position และ End position ของแต่ละ Sequence

[0, 0, 0, 0, 88, 70, 52, 34, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 91, 73, 55, 37, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

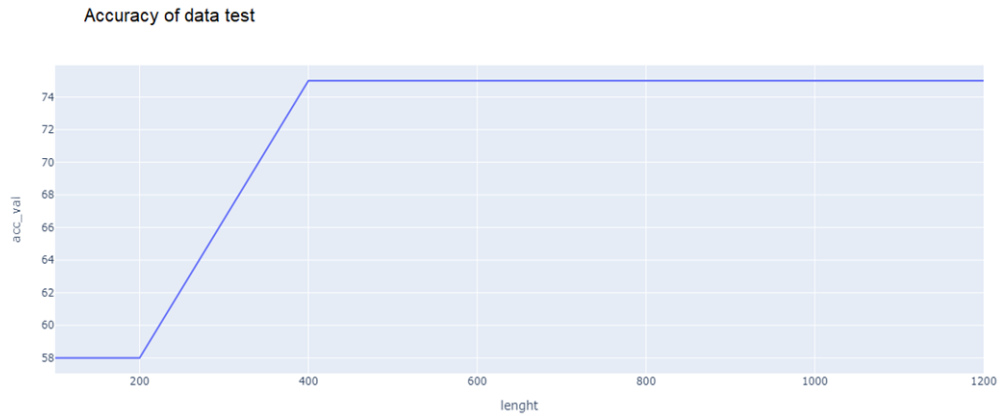
4.2 ผลการวัดประสิทธิภาพโมเดล

การฝึกฝนโมเดลนั้นจะทดสอบโดยการแบ่งช่วงข้อมูลที่จะฝึกฝนโมเดลออกเป็น sequence โดยมีความยาวเท่ากับ 100, 200, 400, 600, 1200 ตามลำดับ เพื่อที่จะทดสอบดูความสามารถในการเรียนรู้ของโมเดลว่า มีความสามารถในการเรียนรู้การตอบคำถาม มากหรือน้อยขึ้นเพียงใดหากเราได้ทำการเปลี่ยนแปลงความยาวของ sequence ที่จะส่งเข้าไปในส่วนของการฝึกฝน โดย Sequence เหล่านี้นั้น จะผ่าน Tokenizer ของ XLMRoBERTa เสียก่อน ซึ่งจะถูกโหลดผ่านมาทาง pretrain ที่เรากำหนด ในกรณีนี้คือ deepset/xlm-roberta-base-squad2 โดยจำนวนก้าวในการแบ่งจะถูกเรียกว่า Stride ซึ่งถูกกำหนดให้มีค่าที่คงที่คือ 128 จากนั้น ก็จะทำการ ทดสอบดูผลลัพธ์การตอบคำถามของแต่ละ Max length ดูว่า จะมีความแตกต่างกันมากน้อยแค่ใน ทั้งในส่วนของ Accuracy และ การตอบคำถาม 3 แบบ ได้แก่ Correct, Incorrect , Similarly โดยเราจะตรวสอบดูโดยใช้ทั้งข้อมูล ฝึกฝน 4000 ชุดคำถามตอบ ที่ใช้ในการฝึกฝนโมเดล และข้อมูล ทดสอบทั้งหมด 74 ข้อความถามตอบ เป็นการทดสอบดูว่าโมเดลมีการเรียนรู้จากข้อมูลมากเกินไปหรือเปล่า เพราะหากโมเดลมีการเลียนแบบข้อมูลมากเกินไปจะทำให้การทดสอบด้วย Dataset ส่วนที่ใช้ฝึกฝนทั้งหมด 4000 ชุดคำถามตอบนั้น มีความถูกต้องที่สูงมาก แต่ถ้าเจอกับข้อมูลที่ไม่เคยพบมาก่อนอย่าง dataset ชุดทดสอบทั้งหมด 74 ชุดคำถามตอบ ก็จะทำนายผลลัพธ์ได้ถูกต้องน้อยลง



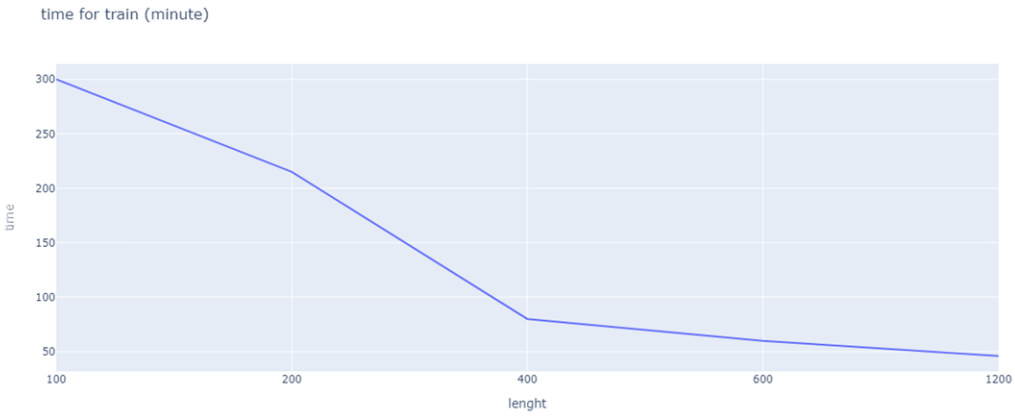
ภาพที่ 4.1 Accuracy dataset ชุดฝึกฝน เทียบกับ Max length

จากกราฟด้านบนจะพบว่า ยิ่ง Max length มีความยาวมากขึ้น Accuracy ก็จะยิ่งเพิ่มตามด้วย โดยเริ่มต้นนั้น หาก Max length มีค่าน้อย เช่น เริ่มต้นที่ 100 Accuracy จะน้อยกว่า Max length อื่นๆที่มากขึ้น ซึ่งอาจเป็นผลสืบเนื่องมาจากการที่โมเดลไม่สามารถเรียนรู้ความสัมพันธ์ของขอมูลคำถามและคำตอบในระยะยาวได้ เพราะ Sequence ที่เราตัดมา มีความสั้นมากเกินไป การเรียนรู้จึงถูกจำกัดอยู่แค่ในช่วงสั้นๆของ Sequence จนอาจลเลยใจความสำคัญบางอย่างที่มีความเกี่ยวข้องกันกับคำตอบ



ภาพที่ 4.2 Accuracy dataset ชุดทดสอบ เทียบกับ Max length

เมื่อมาดูการทดลองกับชุดข้อมูลมทดสอบจะพบว่า Accuracy นั้นมีความคงที่ตั้งแต่ Max length 400 ขึ้นไป นั่นแสดงให้เห็นว่า การที่โมเดลสามารภเห็นความยาวของ Sequence ได้มากขึ้นนั้น ก็มีส่วนช่วยในการตอบคำถามที่ไม่เคยพบมาก่อนได้ถูกต้องมากยิ่งขึ้น



ภาพที่ 4.3 เวลาที่ใช้ในการฝึกฝนเทียบกับ Max length

หากทดสอบดูเวลาในการฝึกฝนที่ทำการฝึกฝนไปทั้งหมดกว่า 10 รอบนั้น ก็พบได้ว่า ยิ่ง Max length เรามีความยาวมากขึ้น เวลาที่ใช้ในการฝึกฝนนั้นก็จะยิ่งน้อยลง ซึ่งเป็นผลเนื่องมาจากการที่ Max length เรายาวขึ้นนั้น ตอนที่ตัด Sequence ออกมานั้น ก็จะได้จำนวน Sequence ที่น้อยลง อีกทั้ง การที่เราทำให้ Sequence มีความยาวที่มากขึ้นนั้น ยังส่งผลให้ผลลัพธ์การทำงานสำหรับตอบคำถามภาษาไทยของโมเดลนั้น มีประสิทธิภาพที่สูงขึ้นอีกด้วยในแง่ของ Accuracy ดังนั้นแต่ก็ยังมีอีกหนึ่งปัจจัยที่ต้องลองทดสอบดูก่อนว่า ดีกว่าจริงหรือไม่ ในแง่ไหน โดยจะใช้การทดสอบดูคำตอบที่ได้จากโมเดลว่า มีคำตอบประเภทใหนบ้าง โดยการทดสอบนี้จะทำให้ทราบได้ว่า มุมมองการทำนายคำตอบของโมเดลนั้น มีความเปลี่ยนแปลงไปอย่างไรเมื่อเทียบกับความยาวของ Sequence ที่ส่งเข้าไป เมื่อเทียบกับคำตอบทั้ง 3 แบบ (Correct, Incorrect, Similarly)

ตารางที่ 4.1 คำตอบ 3 แบบ เทียบกับ Max length

Max length	Correct	Similarly	Incorrect
100	68.52	0.967	28.55
200	48.45	25.2	26.35
400	44.55	38.3	17.15
600	43.17	40.9	15.92
1200	42.25	41.5	16.25

จากผลลัพธ์จะพบได้ว่า อัตราการตอบคำถามได้ถูกต้องทั้งหมดนั้น จะมีจำนวนมากขึ้นเมื่อมีความยาวของ Max length ที่น้อยลง ซึ่งนั่นหมายความว่า การที่โมเดลที่เรียนรู้มาจาก Sequence ที่สั้นกว่านั้น อาจเกิด Bias ขึ้นกับตัวโมเดล และทำให้สามารถตอบคำถามถูกได้มากขึ้นจาก Bias เหล่านั้น แต่ในขณะเดียวกัน ก็ทำให้คำตอบที่ผิดนั้น มีมากขึ้นเช่นเดียวกัน แต่เมื่อลองมองไปที่ Sequence ที่ยาวขึ้น จะพบได้ว่าคำตอบที่ถูกนั้นพอๆกัน แต่คำตอบที่ผิดนั้นกลับน้อยลงเรื่อยๆ ส่วนที่เพิ่มขึ้นมาอย่างมากได้แก่ คำตอบที่คล้ายคำตอบที่ถูกต้อง นั่นอาจเป็นเพราะว่าโมเดลเรียนรู้จาก Sequence ที่ยาวมากขึ้น จึงเกิด Bias ขึ้นมาน้อย แต่ก็ด้วยความยาวของ Sequence นี้เอง ก็ทำให้โมเดลมีความสับสนและเรียนรู้ได้ยากขึ้น จึงทำให้คำตอบที่ถูกต้องทั้งหมด หรือ Correct นั้น ก็ลดลงเองเช่นกัน

4.3 ผลการพัฒนา Application สำหรับการถามตอบอัตโนมัติ

ในส่วนของการพัฒนา Application สำหรับโปรแกรมถามตอบอัตโนมัติด้วยภาษาไทยนั้น ถึงจะเป็นเพียงโปรแกรมต้นแบบ แต่เพื่อความสะดวกในการทดลองใช้งาน จึงได้สร้างหน้าตาโปรแกรมมาให้สะดวก โดยการใช้งานนั้น จะมีปุ่ม Button ให้กดเพื่อเริ่ม Start การทำงาน ซึ่งการทำงานของส่วนโปรแกรมนั้นจะรันและทำงานอยู่บนเครื่องคอมพิวเตอร์ของราเอง โดยในตอนแรกสุดนั้น ตัวโปรแกรมที่ถูกรันขึ้นมาจะยังไม่มีการตอบสนองใดๆต่อผู้ใช้งาน จนกว่าจะเกิดเหตุการณ์ที่ผู้ใช้งาน Click ที่ปุ่ม start ตัวโปรแกรมถึงจะเริ่มทำงาน เมื่อโปรแกรมเริ่มทำงานขึ้นมาแล้ว จะเริ่มพูดกับผู้ใช้งานขึ้นมาว่า “หนูคือ Alice ค่ะ หากมีอะไรที่สงสัย สามารถถาม Alice ได้เลยค่ะ” ซึ่งหลังจาก Alice พูดจบ ผู้ใช้งานก็จะสามารถพพูดคำถามของตนเองออกมาได้เลย จากนั้นตัวรับเสียงของโปรแกรม ก็จะใช้งาน SpeechRecognition เพื่อแปลงเสียงพูดของผู้ใช้งานให้กลายเป็น Text เพื่อที่จะนำไปใช้งานต่อไป ในการเสิร์ชข้อมูล และนำข้อมูลเข้ามาให้โมเดล

สำหรับการคำนวณและพิจารณาหาคำตอบจากข้อมูลเหล่านั้น เนื่องจากโมเดลมีขนาดใหญ่และทำงานได้ช้า ในส่วนการคำนวณเหล่านี้ของโปรแกรม ผู้จัดทำจึงได้เลือกสร้างไว้เป็น API ซึ่งรันอยู่บน google colab ซึ่งมี GPU ที่เร็วและมีหน่วยความจำมากกว่า 120 GB ให้ใช้ ซึ่งสามารถรันโมเดลของเราได้โดยใช้เวลาไม่นาน แต่ก็แบกมากับการทำงานตรงส่วนนี้ จะถูกโยนเข้าไปสู่ API แทนจะไม่ได้ประมวลผลในตัวของโปรแกรมโดยตรง เพราะเมื่อนำโมเดลที่มีขนาดใหญ่และมีกระบวนการทำงานที่ซับซ้อนมารันอยู่บนเครื่องที่มีแค่ CPU นั้นจะใช้เวลาในการประมวลนานมาก ดังนั้น การพัฒนาระบบ API มาเชื่อมโยงกับ การทำงานในส่วนที่ต้องใช้ความสามารถในการประมวลผลและหน่วยความจำที่ค่อนมาก จึงทำให้โปรแกรมสามารถที่จะทำงานต่อไปได้ โดยไม่ใช้เวลามากเกินไป เมื่อผ่านช่วงการทำงานของ API มาได้ ก็จะได้รับ ข้อความที่มีลักษณะเป็น text กลับมา ซึ่งเป็นผลลัพธ์ที่ได้ออกมาจากโมเดที่เราฝึกฝนมา และทงานอยู่เบื้องหลังผ่าน API หลังจากที่รับ Text นั้นมาผ่าน request ของ API แล้ว จึงจะเรียกใช้งาน gtts ซึ่งเป็น API สำหรับทำงานด้าน text to speech

ภาพที่ 4.4 หน้าตาของโปรแกรม

4.4 ผลการทดสอบใช้งานระบบเสิร์ชความรู้

เริ่มแรกหากทดสอบป้อนข้อมูลเข้าไปเป็น ที่ว่า “กรุงเทพมีประชากรเท่าไหร่?” ก็จะได้คำและประเภทของคำดังนี้

[('กรุงเทพฯ', 'PROPN', 'B-LOCATION'), ('มี', 'VERB', 'O'), ('พื้นที่', 'NOUN', 'O'), ('เท่าไหร่', 'NOUN', 'O')]

เมื่อนำมาหา keyword ก็จะพบว่ามีสองคำคือคำว่า ‘กรุงเทพ’ และคำว่า ’พื้นที่’ และเมื่อนำไปเสิร์ชเราจะได้ข้อมูลของกรุงเทพ คือข้อมูลที่ถูกต้องจริงๆ กับข้อมูลของพื้นที่ ซึ่งเป็นอีกเรื่องที่ไม่เกี่ยวข้องกันเลย โดยมีผลลัพธ์จากการ Scraping ข้อมูลมาจาก Thai wiki ดังนี้

- ข้อมูลของกรุงเทพ

[“ซ่อนกรุงเทพมหานคร เป็นเมืองหลวงและนครที่มีประชากรมากที่สุดของประเทศไทย เป็นศูนย์กลางการปกครอง การศึกษา การคมนาคมขนส่ง การเงินการธนาคาร การพาณิชย์ การสื่อสาร และความเจริญของประเทศ ตั้งอยู่บนสามเหลี่ยมปากแม่น้ำเจ้าพระยา มีแม่น้ำเจ้าพระยาไหลผ่านและแบ่งเมืองออกเป็น 2 ฝั่ง คือ ฝั่งพระนครและฝั่งธนบุรี กรุงเทพมหานครมีพื้นที่ทั้งหมด 1,568.737 ตร. กม. มีประชากรตามทะเบียนราษฎรกว่า 6 ล้านคน...”]

- ข้อมูลของพื้นที่

['หรือรูปร่างสองมิติ ที่แสดงถึงขอบเขตเนื้อที่ในแนวแผ่นระนาบ พื้นที่สามารถเข้าใจได้ว่าเป็นจำนวนวัสดุที่หนาขนาดหนึ่งเท่าที่จำเป็นที่จะประกอบขึ้นเป็นรูปร่าง...’]

บทที่ 5
สรุปผลการดำเนินงาน

ในการพัฒนาระบบระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ สามารถสรุปผลการดำเนินโครงงานและข้อเสนอแนะได้ดังนี้.

5.1 สรุปผลการวิจัย

การพัฒนาระบบระบบเสิร์ชความรู้ด้วยการประมวลผลภาษาธรรมชาติจากการสกัดข้อมูลบนเว็บ มีวัตถุประสงค์ดังนี้ 1) พัฒนาระบบสำหรับตอบคำถามภาษาไทย ซึ่งในปัจจุบันนั้น มีF, โมเดลที่สามารถตอบคำถามได้อย่างแม่นยำ แต่ส่วนมากเป็นโมเดลขนาดใหญ่ และส่วนมากถูกฝึกฝนมาด้วยภาษาต่างประเทศ ดังนั้น หากนำโมเดลเหล่านั้นมาฝึกฝนด้วยภาษาไทยและพัฒนาการใช้งานขึ้นมา ก็จะสามารถนำไปต่อยอดได้มากมายในอนาคต โดยมีการทำงานร่วมกันจากหลายส่วนเพื่อให้เกิดโปรแกรมที่ทำงานได้อย่างอัตโนมัติและสะดวกที่สุดในการถามตอบ ซึ่งมีตั้งแต่ระบบการเสิร์ช การสกัด Key word ไปจนถึงกรแปลงเสียงเป็นข้อความและแปลงข้อความเป็นเสียงพูด จนสามารถเกิดเป็นแอปพลิเคชันต้นแบบสำหรับถามตอบคำถามที่เป็นภาษาไทยได้อย่างมีประสิทธิภาพ 2) วิธีการที่ใช้ เปรียบเทียบประสิทธิภาพของ Model transformers ซึ่งตัวของโมเดลที่เลือกมาชื่อว่า XLMRoBERTa โดยใช้ Dataset ที่ชื่อว่า thai_squad ซึ่งมีข้อมูลในการฝึกฝนมากกว่า 4000 ข้อความถามตอบ และข้อมูลทดสอบ 74 ข้อความถามตอบ โดยการฝึกฝนนั้น จะมีการ Preprocess Dataset ตั้งแต่ การตัด Tag html ต่างเป็น Sequence โดยมีความยาวของ Sequence แตกต่างกันไปตั้งแต่ 100, 200, 400, 600, 1200 เพื่อที่จะทดสอบดูว่า เมื่อความยาวของ Sequence ที่แตกต่างกันออกไปถูกนากับ Dataset ภาษาไทยแล้วนั้น จะมีประสิทธิภาพที่แตกต่างกันอย่างไร โดยการฝึกฝนนั้นจะทำในทุกความยาวทั้งหมด 10 รอบ และวัดผลประสิทธิภาพด้วยการดูจาก Accuracy และ ผลลัพธ์ของคำตอบทั้งสามแบบได้แก่ Correct (ตอบคำถามได้ถูกต้อง), Incorrect (ตอบคำถามได้ไม่ถูกต้อง), Similarly (ตอบได้คล้ายคำตอบที่ถูกต้อง) ซึ่งวิธีการประเมินนั้น ก็จะดูจากทั้ง Dataset ฝึกฝนทั้งหมด 4000 ข้อความถามตอบ และ Dataset ทดสอบ ที่โมเดลไม่เคยเห็นมาก่อนเลยทั้งหมด 74 ข้อความถามตอบ

ผลการศึกษาพบว่า ประสิทธิภาพของ Model transformers จากผลลัพธ์การทดสอบจาก dataset ชุดฝึกฝน พบว่าแต่ละ sequence 100, 200, 400, 600, 1200 มีผลลัพธ์ Accuracy ตามลำดับดังนี้ 71%, 73%, 82%, 84%, 84% ส่วนการวัดประสิทธิภาพในชุดข้อมูลทดสอบนั้น ได้ผลลัพธ์ตามลำดับดังนี้ 58%, 58%, 75%, 75%, 75% ส่วนเวลาที่ใช้ในการฝึกฝนทั้งหมด 10 รอบนั้น

มีผลลัพธ์ตามลำดับดังนี้ 180m, 155m, 80m, 60m, 45m นอกจากนั้น เมื่อนำคำตอบทั้งสามแบบมาทดสอบกับการวัดประสิทธิภาพในชุดข้อมูลฝึกฝน ตามลำดับความยาว Sequence พบว่ามีผลลัพธ์ที่ดีที่สุดคือความยาว 600 เนื่องจากมีอัตราความผิดพลาดจากการตอบคำถามผิดกน้อยที่สุดเพียง 15.92% และมีอัตราความถูกต้องของการตอบคำถามแบบถูกต้องทุกตัวอักษรยู่ที่ 43.17% และตอบคำถามได้ใกล้เคียงคำตอบที่ถูกต้องได้ที่ 40.9%

5.2 ปัญหาและข้อเสนอแนะ

ข้อเสนอแนะสำหรับการนำไปใช้งาน และพัฒนาระบบในขั้นต่อไป

5.2.1 ปัญหาระบบการเสิร์ชยังไม่มีความแม่นยำมากพอ อีกทั้งยังไม่สามารถ ค้นหาคำที่มีความเฉพาะเจาะจงหรือมีหลายชื่อเรียกได้ ข้อเสนอแนะคือควรจะพัฒนาระบบให้มีการเสิร์ชข้อมูลได้แม่นยำ และควรพัฒนาส่วนของการจดจำ Name entity และชื่อเฉพาะ

5.2.2 ปัญหาเรื่องข้อมูลของการเสิร์จนั้นยังแคบเกินไป เนื่องจากได้มาจากแหล่งเดียว คือ Thai wiki ข้อเสนอแนะคือควรพัฒนาการเสิร์ชให้สามารถเสิร์ชได้หลากหลายมากขึ้น เช่น ดึงข้อมูลมาจาก Twitter, Pantip, Facebook เป็นต้น

5.2.3 ระบบยังไม่มีความสามารถในการตอบคำถามในเชิงเหตุผลได้ สวนมากจะตอบได้เฉพาะคำถามที่เป็นในเชิง Fact มีค่าแน่นอนตายตัว ทำให้คำตอบดูไม่เป็นธรรมชาติ ไม่สวยงาม ไม่เหมือนมนุษย์ตอบ

บรรณานุกรม

บรรณานุกรม

Anurag Lahon. (2020). **5 Statistical Functions in PyTorch.** สืบค้น 31 กรกฎาคม 2565 https://towardsdatascience.com/statistical-functions-in-pytorch

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. (2017). **Attention is all you need**. arXiv preprint arXiv:1706.03762.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, Veselin Stoyanov. (2020). **Unsupervised Cross-lingual Representation Learning at Scale**. arXiv preprint arXiv:1911.02116.

Anurag Lahon. (2020). **5 Statistical Functions in PyTorch.** สืบค้น 15 กรกฎาคม 2565 จาก https://towardsdatascience.com/5-statistical-functions-in-pytorch-2d75e3dcc1fd

FastAPI. (2021). **FastAPI framework.** สืบค้น 31 กรกฎาคม 2565 จาก https://fastapi.tiangolo.com/

Aws. (2022). **API คืออะไร?.** สืบค้น 11 กรกฎาคม 2565 จาก https://aws.amazon.com/th/what-is/api/

Hugging Face. (2021). **การแบ่งปันโมเดลที่ผ่านการเทรนมาแล้ว (pretrained models).** สืบค้น 30 กรกฎาคม 2565 จาก https://huggingface.co/course/th/chapter4/3?fw=tf

Isara. (2021). **เขียน Python GUI ด้วย Tkinter ep1 – Introduction.** สืบค้น 20 กรกฎาคม 2565 จาก https://stackpython.co/tutorial/python-gui-tkinter-ep1-introduction

IBM Cloud Education. (2016). **What is deep learning?**. สืบค้น 31 กรกฎาคม 2565 จาก https://www.ibm.com/cloud/learn/deep-learning

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. (2019). **BERT:**

Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv.org:1810.04805, 2019.

Michael J. Garbade. (2021). **What is Google Colab?**. Retrieved 30 July 2022 from https://bit.ly/3z9HsSB

Mindphp. (2022). **TensorFlow vs. PyTorch ข้อดีและข้อเสีย.** สืบค้น 22 กรกฎาคม 2565 จาก https://www.mindphp.com/python-tensorflow/8522-tensorflow-vs-pytorch-pros-cons.html

Nuchanat Rongroang. (2020). **ความแตกต่างระหว่าง Machine Learning กับ Deep Learning** สืบค้น 27 กรกฎาคม 2565 จาก https://rdbi.co.th/2020/01/data-scientist-7/

Pisit Bee. (2018). **Backpropagation.** Retrieved 31 July 2022 from https://medium.com/boobeejung/backpropagation-a0c8c6363192

Pakawat Nakwijit. (2020). **ทำความเข้าใจ BERT** สืบค้น 28 กรกฎาคม 2565 จาก https://medium.com/@chameleontk/

Simple Transformers. (2022). **General Usage.** Retrieved 13 July 2022 from https://simpletransformers.ai/docs/usage/

Sigrid Ferreira Rodrigues. (2020). **Google Collabor – คู่มือสำหรับผู้เริ่มต้น.** สืบค้น 29 กรกฎาคม 2565 จาก https://ichi.pro/th/google-collabor-khumux-sahrab-phu-reim-tn-56407695742257/.

ThilinaRajapakse. (2022). **Simpletransformers.** สืบค้น 4 กรกฎาคม 2565 จาก https://github.com/ThilinaRajapakse/simpletransformers

Tiangolo. (2018). **FastAPI.** สืบค้น 2 กรกฎาคม 2565 จาก https://fastapi.tiangolo.com/

Thilina Rajapakse. (2020). **Simple Transformers.** สืบค้น 31 กรกฎาคม 2565 จาก https://github.com/ThilinaRajapakse/simpletransformers/.

thaiqa_squad dataset. **Thai Wikipedia QA dataset made by NECTEC [Online]**. Available: thaiqa_squad · Datasets at Hugging Face/. [Accessed: 8-Dec-2020].

Wicharn RueangkhajornWicharn and Jonathan H. Chan. (2022). **Question Answering Model in Thai by using Squad Thai Wikipedia dataset**. TechRxiv. preprint 15.12.2021

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. (2019). **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. arXiv. preprint arXiv:1907.11692.

9experttraining. (2562). **ภาษาโปรแกรม Python คืออะไร ?**. สืบค้น 31 กรกฎาคม 2565 จาก https://rb.gy/tgvdkk

ประวัติผู้จัดทำ

ผู้จัดทำ	นาย ธีรพงศ์ หารยงค์
ชื่อเล่น	หน่อง
รหัสนักศึกษา	62102105125
ที่อยู่	125/5 บ.ปุ่งใหญ่ ต.นาตงวัฒนา อ.โพนนาแก้ว จ.สกลนคร 47230
การศึกษา	ระดับประถมศึกษา โรงเรียนบ้านปุ่งสหราษฎร์บำรุง
	ระดับมัธยมศึกษาตอนต้น โรงเรียนโพนพิทยาคม
	ระดับมัธยมตอนปลาย โรงเรียนโดนพิทยาคม
	ปริญญาตรี (วท.บ. วิทยาการคอมพิวเตอร์) มหาวิทยาลัยราชภัฏสกลนคร
คติประจำใจ	สิ่งเดียวที่เรารู้ได้ คือสิ่งที่เรารู้
การติดต่อ	Facebook : Teerapong Hanyong
	E-mail : 0999362779a@gmail.com
	Tell : 0985878048