การใช้การเรียนรู้ของเครื่องตรวจสอบข้อความสืบค้นของโปรแกรม GeoSearch

Using machine learning to validate search query in GeoSearch

นายชลทิตย์ เยาวลาภ

Chonlatit Yaovalap

นายธนบดี ชุมสาย ณ อยุธยา

Thanabordee Jumsai Na Ayudhya

นางสาวฝน วงศ์วรเชษฐ์

Fon Vongvorachate

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา พ.ศ. 2563

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ชื่อหัวข้อโครงงาน **การใช้การเรียนรู้ของเครื่องตรวจสอบข้อความสืบค้นของโปรแกรม**
Using machine learning to validate search query in Geosearch

นิสิต	นายชลทิตย์ เยาวลาภ	60102010331
	นายธนบดี ชุมสาย ณ อยุธยา	60102010334
	นางสาวฝน วงศ์วรเชษฐ์	60102010570

ปริญญา วิทยาศาสตร์บัณฑิต (วท.บ.)
ภาควิชา วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษาโครงงาน อ.ดร.ศิริสรรพ เหล่าหะเกียรติ

ลงชื่อ.....
(อ.ดร.ศิริสรรพ เหล่าหะเกียรติ)
อาจารย์ที่ปรึกษาโครงงาน

บทคัดย่อ

ในปัจจุบันเทคโนโลยีการค้นหาตำแหน่งของสถานที่ด้วยโปรแกรมคอมพิวเตอร์ ชื่อสถานที่หรือที่อยู่ได้มีความสำคัญอย่างมากในชีวิตประจำวัน ในระบบการค้นหานี้คำค้นหาจะมีความสำคัญเป็นอย่างมาก ถ้าผู้ใช้ป้อนคำค้นหาที่ผิดพลาด หรือเขียนผิด อาจทำให้โปรแกรมไม่สามารถหาตำแหน่งของสถานที่ที่ต้องการได้ การศึกษานี้มุ่งพัฒนาระบบการจัดการข้อความ เพื่อตรวจและแก้ไขข้อความสำหรับค้นหาสถานที่ โดยมีขอบเขตสถานที่ครอบคลุมทั้งประเทศไทย ใช้เทคนิค Name Entity Recognition ในการแยกแยะชื่อสถานที่ออกจากประโยค โดยอาศัยแบบจำลอง Conditional Random Field จากนั้นจะใช้แบบจำลอง Gated Recurrent Unit ในการแบ่งคำจากประโยค เพื่อนำคำแต่ละคำมาแก้ไขคำที่เขียนผิดพลาดให้ถูกต้องโดยใช้แบบจำลอง K Nearest Neighbor และนำคำเหล่านี้มาตรวจว่าเป็นคำที่เกี่ยวข้องกับที่อยู่หรือไม่ ด้วยแบบจำลอง Bi Long Short Term Memory เพื่อนำมาประกอบกันให้เป็นประโยคที่อยู่ที่ถูกต้อง ผลการทดลองพบว่าระบบที่พัฒนาขึ้นมีความแม่นยำถึง 78 เปอร์เซ็นต์

Abstract

Nowadays, location search using computer has become one of the important application in our life. However, if the user fails to input accurate searching phrase, the sever cannot provide correct location and navigation for that query. In this study, we aim at developing a machine learning system for processing and correcting location searching query. Our system is developed for searching query in Thai with the target locations in Thailand. We adopt name entity recognition technique to tokenize location name using conditional random field. Then, we use neural network to correct the erroneous phrases by using Bi-LSTM model to perform this task. The experimental results show that the accuracy of our system is as high as 78 percent.

กิตติกรรมประกาศ

การดำเนินโครงงาน การใช้การเรียนรู้ของเครื่องตรวจสอบข้อความสืบค้นของโปรแกรม (Using machine learning to validate search query in GeoSearch) สำเร็จลุล่วงได้ด้วยความกรุณาช่วยเหลือและให้คำปรึกษาอย่างดีเป็นอย่างยิ่งจาก อ.ดร.ศิริสรรพ เหล่าหะเกียรติ อาจารย์ที่ปรึกษาของโครงงาน ที่ได้กรุณาเสียสละเวลาเพื่อมาให้คำปรึกษาเสนอแนะแนวทางในการแก้ไขปัญหา และแก้ไขข้อบกพร่องที่เกิดขึ้นระหว่างการทำงาน คณะผู้จัดทำข้อขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิทยาการคอมพิวเตอร์ ที่สนับสนุนอุปกรณ์ สถานที่ในดำเนินโครงงานและขอขอบพระคุณ อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ทุก ๆ ท่าน ที่ได้กรุณาให้ความรู้ทำให้คณะผู้จัดทำสามารถนำความรู้ที่ได้มาประยุกต์ใช้ในการดำเนินโครงงานในครั้งนี้ รวมทั้งให้คำปรึกษาและคำแนะนำที่เป็นประโยชน์ทำให้คณะผู้จัดทำสามารถแก้ไขโครงงานให้ดีขึ้น และมีความสมบูรณ์มากยิ่งขึ้น

ขอขอบคุณมหาวิทยาลัยศรีนครินทรวิโรฒ ที่เป็นแหล่งศึกษาหาความรู้ที่เป็นประโยชน์ในการดำเนินโครงงาน และอำนวยความสะดวกสำหรับสถานที่ในการดำเนินโครงงานในครั้งนี้ รวมทั้งให้ประสบการณ์ในการทำกิจกรรมต่าง ๆ เพื่อที่จะนำไปประยุกต์ใช้ในการทำงานและการดำเนินชีวิตต่อไป

ขอขอบพระคุณ คุณพ่อ คุณแม่ ครอบครัวของคณะผู้จัดทำ ที่ให้การช่วยเหลือและสนับสนุนมาโดยตลอด รวมทั้งให้ความห่วงใยและเป็นกำลังใจสำคัญในการทำโครงงานเสมอมา สุดท้ายขอขอบพระคุณผู้มีส่วนเกี่ยวข้องทุก ๆ ท่าน ที่ทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำหวังว่าโครงงานการใช้การเรียนรู้ของเครื่องตรวจสอบข้อความสืบค้นของโปรแกรม (Using machine learning to validate search query in GeoSearch) จะเป็นประโยชน์ต่อผู้ที่สนใจ ศึกษา และสามารถเป็นที่ศึกษาสำหรับผู้ที่สนใจจะนำไปต่อยอดในบริบทต่าง ๆ ต่อไป

นายชลทิตย์ เยาวลาภ
นายธนบดี ชุมสาย ณ อยุธยา
นางสาวฝน วงศ์วรเชษฐ์

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1. ที่มาและความสำคัญของโครงงาน

ในอดีตผู้คนส่วนมากมักใช้คอมพิวเตอร์ในการค้นหาไฟล์หรือสิ่งต่าง ๆ มากมายซึ่งการค้นหาโดยวิธีการไล่ดูนั้นมักจะพบว่าใช้เวลานานและไม่มีประสิทธิภาพในการค้นหา แต่ในปัจจุบันนั้นตัวระบบปฏิบัติการก็ได้มีเครื่องมือที่ใช้สำหรับการ Search ซึ่งทำให้เพิ่มความสะดวกในการค้นหาสิ่งต่าง ๆ นอกจากนี้ยังมีโปรแกรมที่ออกแบบและพัฒนามาเพื่อช่วยผู้คนค้นหาข้อมูลได้อย่างรวดเร็วหรือที่เรียกกันว่า Search Engine

การใช้เครื่องมือในการค้นหาสิ่งต่าง ๆ นั้นเราจะต้องใส่คำต่าง ๆ หรือที่เรียกกันว่า keyword ที่ใช้สำหรับในการค้นหา ซึ่งส่วนใหญ่เมื่อเราทำการค้นหาผลลัพธ์ที่ได้กลับมาอาจจะไม่ใช้สิ่งที่เราต้องการ หรืออาจจะไม่ได้ผลลัพธ์เลย เช่น การใช้ Search Engine สำหรับการแปลงข้อมูลที่อยู่ให้กลายเป็นพิกัดจริงบนพื้นโลก ซึ่งถ้าเกิดผู้ใช้ใส่ข้อความที่ไม่จำเป็นมากเกินไปหรือใส่ข้อความที่ผิดอาจจะทำให้ไม่สามารถหาพิกัดเจอได้อย่างถูกต้อง

ดังนั้นทางกลุ่มนิสิตจึงจัดทำโครงงานที่จะ cleansing ข้อมูลคำค้นหา เพื่อช่วยลดปัญหาในการค้นหาสถานที่ให้มีความแม่นยำยิ่งขึ้นและลดโอกาสผิดพลาดให้น้อยที่สุดโดยนำ Machine Learning มาใช้ในการวิเคราะห์หาคำที่มีความคล้ายคลึงกัน เพื่อช่วยเพิ่มประสิทธิภาพในการค้นหา

1.2. วัตถุประสงค์ของโครงงาน

1. เพื่อให้แบบจำลองมีการทำงานที่มีประสิทธิภาพมากขึ้น
2. สามารถกำจัดคำขยะได้ในรูปแบบภาษาไทย
3. เป็นแนวทางให้ผู้ที่สนใจมาพัฒนาต่อยอดได้

1.3. ขอบเขตของโครงงาน

1. ผลลัพธ์ที่ได้เป็นไปตามรูปแบบคำค้นหาของบริษัท จีไอเอส จำกัด
2. ผลลัพธ์ที่ได้ใช้สำหรับค้นหาข้อมูลตำแหน่งของบริษัท จีไอเอส จำกัด เป็นหลัก
3. รองรับการค้นหาในรูปแบบภาษาไทย
4. พัฒนาและต่อยอดในรูปแบบของเว็บเซอร์วิส

1.4. สมมติฐานของโครงงาน

1. สามารถวิเคราะห์คำค้นหาให้ตรงกับความต้องการของผู้ใช้งาน
2. ผู้ใช้ป้อนข้อมูลที่อยู่ในรูปแบบภาษาไทย

1.5. ประโยชน์ที่คาดว่าจะได้รับ

1. ช่วยให้ผลลัพธ์ในการค้นหา มีประสิทธิภาพและตรงกับความต้องการของผู้ใช้งานมากยิ่งขึ้น
2. เพื่อเพิ่มพูนความเข้าใจในหลักการประมวลผลและจัดการกับข้อความ

บทที่ 2
องค์ความรู้ที่เกี่ยวข้อง

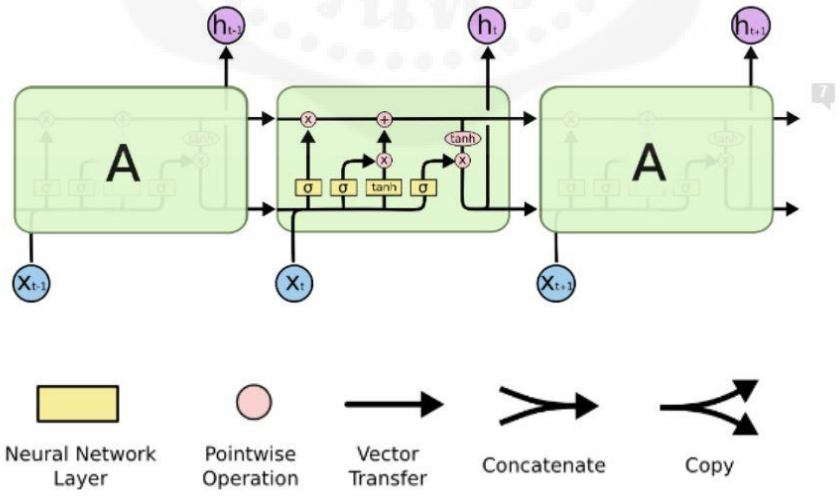
2.1 Recurrent Neural Network (RNN) [1]

Recurrent Neural Network (RNN) คือ Artificial Neural Network แบบหนึ่งที่ออกแบบมาแก้ปัญหาสำหรับงานที่ข้อมูลมีลำดับ Sequence โดยใช้หลักการ Feed สถานะภายในของแบบจำลองกลับมาเป็น Input ใหม่คู่กับ Input ปกติที่เรียกว่า Hidden State, Internal State, Memory ช่วยให้แบบจำลองรู้และจำ Pattern ของลำดับของ Input Sequence ได้

2.2 LSTM (Long Short-Term Memory) [2]

LSTM หรือชื่อเต็มคือ Long Short-Term Memory เป็นโครงข่ายประสาทเทียม Recurrent Neural Network รูปแบบหนึ่งที่มีประสิทธิภาพ มีความสามารถในการเรียนรู้ Long-Term Dependencies นั่นคือความสามารถในการจดจำระยะยาว โดยมี **Gate** ที่เปิดรับข้อมูลที่เข้ามาใหม่เพื่อบันทึกข้อมูลลงไปในแต่ละ Node และ Gate ที่ค่อยควบคุมว่าเมื่อใดจะนำข้อมูลจากในอดีตมาคำนวณแล้วส่งค่าไปยังการคำนวณครั้งต่อไป

จุดเด่นข้อง LSTM คือ สามาถเลือกได้ว่าข้อมูลไหนที่ควรจะจำ ข้อมูลไหนที่ควรจะกำจัดทิ้งไป ซึ่ง Gate ต่าง ๆ ที่กล่าวมานั้นเป็นตัวควบคุมทิศทางการไหลของข้อมูลในแต่ละ Node เป็นการทำงานที่เก็บสถานะของข้อมูลเอาไว้อยู่ตลอด โดยมี Notation และการทำงานดังรูป

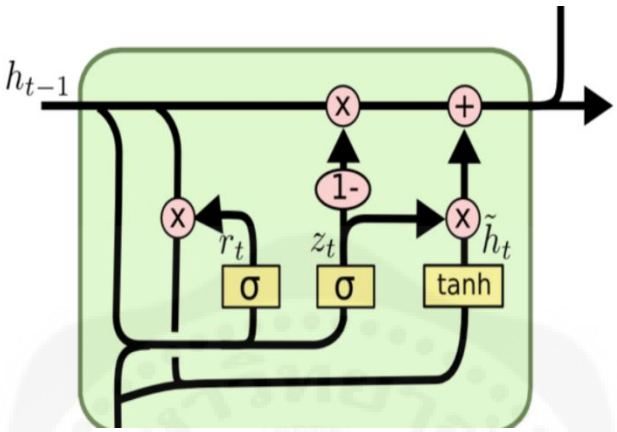


รูปที่ 1 แสดงโครงสร้างการทำงานของ LSTM

ที่มา : [ออนไลน์] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

2.3 Gated Recurrent Unit (GRU) [1]

Gated Recurrent Units (GRU) เป็นกลไกปิดเปิดการอัพเดทสถานะภายใน Recurrent Neural Network ที่คล้ายกับ Long Short-Term Memory (LSTM) ที่จะมี Forget Gate แต่มี Parameter น้อยกว่า LSTM เนื่องจากไม่มี Output Gate



รูปที่ 2 : แสดงโครงสร้างการทำงานของ GRU

ที่มา : [ออนไลน์] https://www.data-blogger.com/2017/08/27/gru-implementation-tensorflow/

2.4 Named-Entity Recognition (NER) [3]

Named-Entity Recognition เป็นวิธีหนึ่งในการประมวลผลภาษาธรรมชาติ (Natural Language Processing) หรือ NLP โดยเป็นกระบวนการทำความเข้าใจและรู้จำชื่อเฉพาะในข้อความหรือประโยคว่าหมายถึงสิ่งใด เช่น เป็นชื่อของบุคคล, สถานที่, องค์กร เป็นต้น ซึ่งการทำ Named-Entity Recognition นิยมทำ 2 วิธีคือ

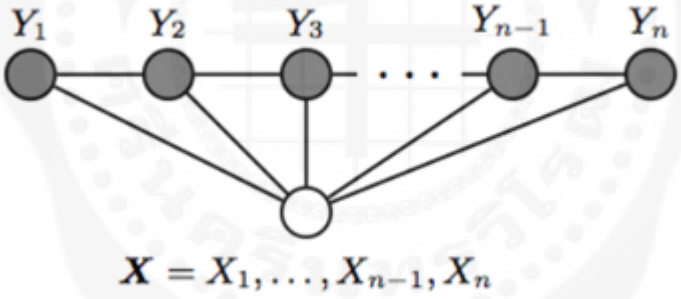
1. **Deep Neural Network Extractors** นำกระบวนการด้าน Deep Neural Network มาใช้ในการจัดจำแนกหมวดหมู่คำโดยอาศัยหลักทางสถิติ
2. **Pattern Matching Extractors** อาศัยรูปแบบเฉพาะของแต่ละกลุ่มคำในการจัดจำแนกหมวดหมู่ ตัวอย่างเช่น hi@example.com เป็นอีเมล โดยรูปแบบคือมี @ อยู่ในคำ และลงท้ายด้วย Top-Level Domain อย่าง .com เป็นต้น

2.5 Word Embedding [4]

Word Embedding คือการแปลงคำเป็นตัวเลข ถือเป็นหนึ่งในวิธีในการสร้าง Feature จากคำวิธีหนึ่ง และยังเป็นการลดขนาด Vector Space ลงด้วย โดยมีเครื่องมือที่ช่วยในการทำ Word Embedding อาทิ Word2Vec ซึ่งจะคำนวณค่าของคำนั้น ๆ จาก Context โดยรอบ ซึ่งสามารถแก้ปัญหา Speed ในการคำนวณที่ช้าและ Space ที่ใช้อย่างมากมาย รวมถึงปัญหา Coverage คือคำที่เกิดขึ้นร่วมกัน อาจจะไม่ได้เกิดจากคำที่อยู่ติดกันก็ได้ ซึ่งนอกจาก Word2Vec แล้วยังมีเครื่องมืออื่น ๆ ที่น่าสนใจอย่าง FastText หรือ Thai2Vec เป็นต้น

2.6 Condition Random Field (CRF) [5]

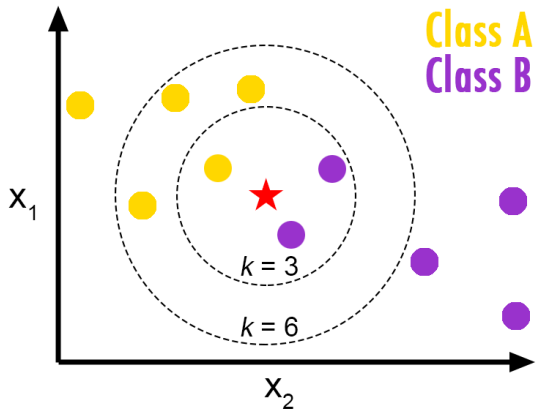
Condition Random Field เป็นแบบจำลองทางภาษาชนิดหนึ่งเหมาะกับข้อมูลที่เป็นลำดับ มีพื้นฐานมาจาก Markov Random Field ซึ่งอาศัยหลักความน่าจะเป็นทางสถิติในการจัดจำแนกประเภทของคำตามหมวดหมู่ แบบจำลองจะทำการสร้างกราฟความสัมพันธ์ของลำดับหมวดหมู่คำ เพื่อหาลำดับหมวดหมู่คำที่เหมาะสมที่สุด โดยผลลัพธ์ก่อนหน้าจะมีผลต่อผลลัพธ์ถัดไป



รูปที่ 3 : แสดงความสัมพันธ์ระหว่าง หมวดหมู่คำ (Y) และคำในประโยค (X) ของ CRF
ที่มา : [ออนไลน์] http://www.davidsbatista.net/blog/2017/11/13/Conditional_Random_Fields/

2.7 K-Nearest Neighbor (K-NN) [6]

K-Nearest Neighbor เป็นอัลกอริทึมใช้ในงานประเภท Classification เป็นส่วนมาก โดยอิงจากข้อมูลที่อยู่บริเวณใกล้เคียงด้วย Euclidean Distance เพื่อช่วยในการทำนาย และขึ้นอยู่กับการกำหนดค่า k ซึ่งหมายถึงจำนวนข้อมูลใกล้เคียงโดยรอบที่ใช้ในการพิจารณา ดังตัวอย่างรูปด้านล่าง จะเห็นว่าถ้ากำหนด k=3 จุดรูปดาวสีแดงจะถูกกำหนดให้เป็นคลาส B ทว่าหากกำหนด k=6 จุดรูปดาวสีแดงจะถูกกำหนดให้เป็นคลาส A แทน



รูปที่ 4 : แสดงหลักการของ K-Nearest Neighbor

ที่มา : [ออนไลน์] https://medium.com/@equipintelligence/k-nearest-neighbor-classifier-knn-machine-learning-algorithms-ed62feb86582

2.8 Levenshtein Distance [7]

Levenshtein Distance คือการหาค่าเปรียบเทียบความแตกต่างของสองคำในระดับอักขระ โดยวัดจากจำนวนครั้งในการเปลี่ยนแปลงอักษรในคำตั้งต้นจนกลายเป็นคำที่ต้องการ สามารถประยุกต์วิธีการนี้ในการหาความคล้ายคลึงระหว่างสองคำได้โดยวัดจากจำนวนครั้งในการเปลี่ยนแปลงที่น้อยที่สุด เรียกว่า Minimum Edit Distance ซึ่งการเปรียบเทียบความแตกต่างมีหลักพิจารณาดังนี้

- การแทรก หมายถึงการเพิ่มอักขระเข้าไปในคำ เพื่อให้คำทั้งสองมีความใกล้เคียงหรือเหมือนกันยิ่งขึ้น เช่น คำตั้งต้นคือ ‘ยศ’ และคำที่ต้องการคือ ‘พยศ’ สามารถเปลี่ยนแปลงได้โดยทำการเพิ่ม พ เข้าไป
- การตัด หมายถึงการตัดอักขระที่ไม่ต้องการออกไปจากคำ เพื่อให้คำทั้งสองมีความใกล้เคียงหรือเหมือนกันยิ่งขึ้น เช่น คำตั้งต้นคือ ‘แผ่น’ และคำที่ต้องการคือ ‘แผ่’ สามารถเปลี่ยนแปลงได้โดยทำการตัด น ออกไป
- การแทนที่ หมายถึงการแทนที่อักขระในคำด้วยอีกอักขระหนึ่ง เพื่อให้คำทั้งสองมีความใกล้เคียงหรือเหมือนกันยิ่งขึ้น เช่น คำตั้งต้นคือ ‘ราก’ และคำที่ต้องการคือ ‘รฦก’ สามารถเปลี่ยนแปลงได้โดยทำการแทนที่ า ด้วย ฦ

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

รูปที่ 5 : การหาค่า edit distance ของคำว่า 'RELEVANT' กับ 'ELEPHANT'

ที่มา : [ออนไลน์] http://ntz-develop.blogspot.com/2011/03/fuzzy-string-search.html

2.9 งานวิจัยที่เกี่ยวข้อง

1. **งานวิจัยเรื่อง "Siamese Recurrent Architectures for Learning Sentence Similarity"** [8]

Jonas Mueller นำเสนอการปรับตัวของเครือข่ายหน่วยความจำระยะสั้น (LSTM) สำหรับการติดฉลากข้อมูลซึ่งประกอบด้วยลำดับความยาวแปรผันคู่ซึ่งแบบจำลองของเราถูกนำไปใช้ในการประเมินความคล้ายคลึงกันทางความหมายระหว่างประโยคมีความทันสมัยมีประสิทธิภาพสูงกว่าสมบัติที่ออกออกแบบด้วยมืออีกทั้งระบบเครือข่ายประสาทที่ถูกเสนอล่าสุดนั้นมีความซับซ้อนมากขึ้นสำหรับแอปพลิเคชันเหล่านี้เพราะเราให้มีการฝังเวกเตอร์ที่เสริมด้วยข้อมูลคำพ้องความหมายได้ให้กับ LSTM ซึ่งใช้เวกเตอร์ขนาดคงที่เพื่อเข้ารหัสความหมายพื้นฐานที่แสดงในประโยคโดยการจำกัดการปฏิบัติงานตามมาเพื่อพึ่งพาตัวชี้วัดแบบแมนฮัตตันอย่างง่ายซึ่งเราบังคับใช้การแทนประโยคที่เรียนรู้จากแบบจำลองเพื่อสร้างพื้นที่ที่มีโครงสร้างสูงและใช้เรขาคณิตสะท้อนความสัมพันธ์เชิงความหายที่ซับซ้อนโดยผลลัพธ์เป็นผลการวิจัยล่าสุดที่แสดงให้เห็นว่า LSTM เป็นแบบจำลองภาษาที่ทรงพลังซึ่งสามารถทำงานที่ต้องการความเข้าใจที่ซับซ้อน

2. **งานวิจัยเรื่อง"ระบบจำแนกและค้นคืนข้อมูลเว็บกระทู้ข่าวด้วยโครงข่ายประสาทเทียมเปอร์เซ็ปตรอนแบบหลายชั้น"(A Web News Information Classification and Retrieval System using Multilayer Perceptron Neural Network)** [9]

งานวิจัยนี้มีวัตถุประสงค์เพื่อพัฒนาระบบจำแนกและการค้นคืนข้อมูลเว็บกระทู้ข่าวโดยใช้โครงข่ายประสาทเทียม เปอร์เซ็ปตรอนแบบหลายชั้น (Multilayer Perceptron) ซึ่งใช้ 3 เครื่องมือหลักในการจำแนกข้อมูลกระทู้ข่าวจากเว็บไซต์พันทิป ได้แก่ 1 Rapidminer ใช้ในการพัฒนาแบบจำลองของโครงข่ายประสาทเทียม 2 JavaScript และ jQuery ในการพัฒนาระบบเก็บรวบรวมข้อมูล (Crawler) จากเว็บไซต์พันทิปและ 3 คลาสไลบรารี่เล็กซ์โต (ThaiLexemeTo-kenzer:LexTo) ใช้ในการตัดคำภาษาไทยและคำนวรหาค่าน้ำหนัก (Weight) ของคำนั้น ๆ เพื่อใช้เป็นชุดข้อมูลสำหรับการเรียนรู้ของโครงข่ายประสาทเทียมในการจำแนกข้อมูลและใช้ 4 เครื่องมือหลักในการคืนคืนข้อมูล ได้แก่ 1 Vector Space Model (VSM**)** ใช้ในการค้นคืนข้อมูลเพื่อเปรียบเทียบความคล้ายของคำค้นกับเอกสาร 2 Apache Solr ใช้ในการสร้างดัชนีข้อมูล (Index) ของเอกสารเพื่อใช้ในการค้นคืนข้อมูลอย่างมีประสิทธิภาพ 3 N-Gram ใช้ในการแนะนำชุดคำถามที่ถูกต้องแบบอัตโนมัติและ 4 LexTo ใช้ในการตัดคำเพื่อขยายชุดคำถาม (Query Expansion) ให้ได้ผลลัพธ์ที่ตรงตามความต้องการมากที่สุด พร้อมทั้งตัดคำหยุดหรือคำที่ไม่มีความหมาย (Stop Word) เพื่อให้ได้ผลลัพธ์ที่ตรงกับความต้องการของผู้สืบค้นมากที่สุดจากการทดสอบประสิทธิภาพของการจำแนกข้อมูลและการค้นคืนข้อมูลได้ค่าความแม่นยำ (Precision) เท่ากับ 74.51% และ 86.30% และค่าความระลึก (Recall) เท่ากับ 75.36% และ 100% ตามลำดับซึ่งเป็นค่าที่น่าพอใจจึงสรุปได้ว่างานวิจัยนี้สามารถจำแนกและค้นคืนข้อมูลได้ในระดับที่ดีมาก

3. งานวิจัยเรื่อง "Thai Named Entity Recognition Using Bi-LSTM-CRF with Word and Character Representation" [10]

งานวิจัยนี้ได้นำเสนอการประยุกต์ใช้ Bi-LSTM-CRF ที่มีการแสดงระดับคำ/อักขระเพื่อทำ Named Entity Recognition สำหรับข้อมูลภาษาไทย เริ่มจากเตรียมข้อมูลโดยการแยกข้อความหรือประโยคออกเป็นกลุ่มคำ จากนั้นจึงทำการแปลงกลุ่มคำและอักขระด้วย Bi-LSTM และในท้ายที่สุดได้สร้างเครือข่ายประสาทเทียม Recurrent Neural Network (R-NN) รวมกับ Condition Random Field (CRF) เพื่อเรียนรู้ลำดับของคำและเรียนรู้เพื่อสร้างการจดจำ NER ซึ่งแบบจำลองของงานวิจัยนี้ได้รับการประเมินโดย NER opensource corpus จากกลุ่ม Facebook ThaiNLP ผลลัพธ์ของแบบจำลองให้ค่าความแม่นยำ (Precision) เท่ากับ 91.79% ค่าความระลึก (Recall) เท่ากับ 91.51% และค่า F1 เท่ากับ 91.65%

บทที่ 3
วิธีการดำเนินโครงงาน

3.1 วิธีการดำเนินโครงงาน

1. ศึกษาค้นคว้าเอกสารและงานวิจัยที่เกี่ยวข้อง
2. นำแบบจำลองที่ได้มาปรับปรุงแก้ไขเพื่อเพิ่มประสิทธิภาพให้เหมาะกับการใช้งานจริง
3. ทดสอบแบบจำลองที่นำมาปรับปรุงประสิทธิภาพและวัดประสิทธิภาพ
4. สร้างเว็บเซอร์วิสเพื่อดึงข้อมูลจากฐานข้อมูลมาทดลองกับแบบจำลองที่สร้างขึ้น
5. ตรวจสอบและแก้ไจให้โครงงานสำเร็จสมบูรณ์มากขึ้นตามวัตถุประสงค์
6. สรุปผลงานวิจัย

3.2 แผนการดำเนินโครงงาน

การดำเนินการในการทำโครงงานได้แบ่งออกเป็น 4 ระยะ คือ

ระยะที่ 1 : วางแผนจัดทำโครงงาน ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง ศึกษาเครื่องมือที่ใช้ในงานวิจัย

ระยะที่ 2 : ออกแบบขั้นตอนการเตรียมข้อมูลและการฝึกฝนแบบจำลอง

ระยะที่ 3 : ฝึกฝนแบบจำลองและปรับปรุงประสิทธิภาพพร้อมกับประมวลการทำงานของแบบจำลอง

ระยะที่ 4 : นำส่วนของแบบจำลองแต่ละส่วนมารวมกันทดสอบการทำงานและปรับปรุงแก้ไขพร้อมทำ Web Service

ระยะเวลาในการดำเนินการตั้งแต่เดือนมิถุนายน 2563 ถึงเดือนเมษายน 2564 รวมเวลาทั้งสิ้น 11 เดือน โดยมีแผนการดำเนินโครงงาน ดังต่อไปนี้

3.3 แผนการดำเนินงานตลอดโครงงาน

แผนการดำเนินงาน	พ.ศ.2563							พ.ศ.2564			
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1) วางแผนโครงงาน											
เลือกหัวข้อโครงงาน											
ศึกษาแนวคิดและทฤษฎี											
ศึกษาเทคโนโลยีที่จะนำมาใช้											
2) รวบรวมข้อมูล											
ศึกษาและเตรียมข้อมูลให้พร้อมใช้งาน											
3)การสกัดที่อยู่จากประโยค											
ใช้งานแบบจำลองและปรับปรุงประสิทธิภาพ											
4)การตัดคำของประโยคที่อยู่											
ใช้งานแบบจำลองและปรับปรุงประสิทธิภาพแบบจำลอง											
5) การแท็กประเภทของคำ											
เตรียมข้อมูลให้พร้อมสำหรับฝึกฝน											
ฝึกฝนแบบจำลองและปรับปรุงประสิทธิภาพ											
6) การแก้ไขคำ											
เตรียมข้อมูลให้พร้อมสำหรับฝึกฝน											
ฝึกฝนแบบจำลองและปรับปรุงประสิทธิภาพ											
7) การเติม/แก้ไขข้อมูลที่อยู่ให้สมบูรณ์ยิ่งขึ้น											
สร้างและปรับปรุงประสิทธิภาพฟังก์ชัน											
8) สรุปและประเมินผลโครงงาน											
สรุปและนำเสนอโครงงาน											

ตารางที่ 1 แผนการดำเนินโครงงานวิจัย

3.4 อุปกรณ์และเครื่องมือที่ใช้ในการวิจัย

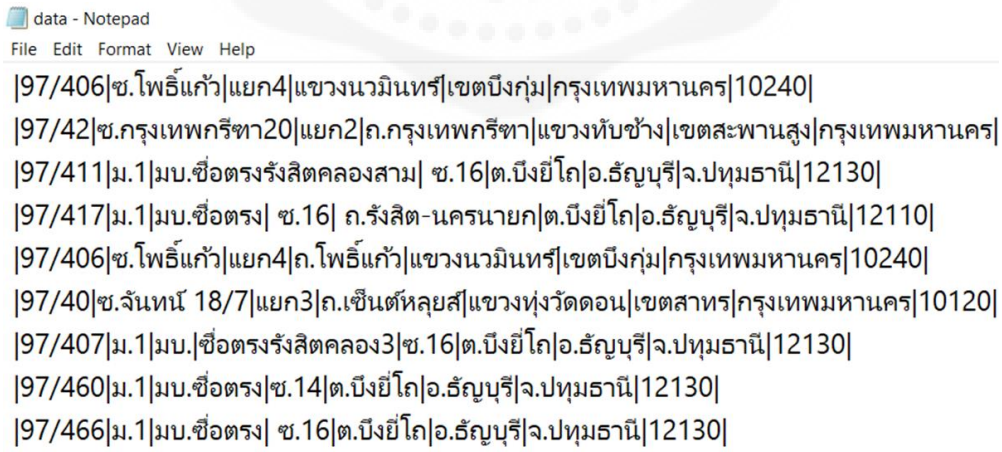
1. **ฮาร์ดแวร์ (Hardware):**
    1. คอมพิวเตอร์

2. **ซอฟต์แวร์ (Software):**
    1. Visual Studio Code ใช้ในด้านการพัฒนาซอฟต์แวร์ในส่วนของการเขียนและแก้ไขโค้ด
    2. GitLab ใช้เก็บซอร์สโค้ด (Repository) ของโครงงาน
    3. Jupyter Notebook/Google Colab เป็นเครื่องมือในการจัดการข้อมูล
    4. Django ใช้สำหรับทำให้รัน Python.py เป็น Web Service
    5. Ubuntu เพื่อรัน Django แบบ wsgi เพื่อให้ประมวลผลพร้อมกัน

3. **ภาษาที่ใช้ (Language):**
    1. JavaScript พัฒนาในส่วนของการเรียกใช้ Web Service และ ประมวลผลข้อมูลฝั่ง Python
    2. Python ใช้ในการนำข้อมูลมาวิเคราะห์ผลและใช้เป็นเครื่องมือในการสร้างแบบจำลองต่าง ๆ

3.5 ชุดข้อมูล (Dataset) ที่ใช้งาน

1. ข้อมูลที่อยู่ที่ในประเทศไทยจำนวนทั้งหมด 1,400,000 Rows



รูปที่ 6 : ข้อมูลที่อยู่ในประเทศไทย

2. Bi-LSTM_CRF Dataset เป็นชุดข้อมูลที่อยู่ 33 จังหวัด ที่ได้มีการสร้างแท็กคำระบุประเภทของข้อมูลในแต่ละส่วนของข้อมูล จำนวนทั้งสิ้น 332,496 ที่อยู่ ซึ่งใน Dataset ประกอบด้วยแท็กระบุประเภทของข้อมูล โดยมีความหมายและจำนวนแต่ละแท็กดังนี้

'H_ID' หมายถึง บ้านเลขที่ จำนวน 332,463 แท็ก

'H_NAME' หมายถึง ชื่อหมู่บ้าน/อาคาร จำนวน 152,121 แท็ก

'MOO' หมายถึง หมู่ จำนวน 77,050 แท็ก

'ALLEY' หมายถึง ซอย จำนวน 233,221 แท็ก

'ROAD' หมายถึง ถนน จำนวน 229,055 แท็ก

'DIST' หมายถึง แขวง/ตำบล จำนวน 332,496 แท็ก

'ZONE' หมายถึง เขต/อำเภอ จำนวน 332,496 แท็ก

'PROVINCE' หมายถึง จังหวัด จำนวน 332,496 แท็ก

'POSTCODE' หมายถึง รหัสไปรษณีย์ จำนวน 317,927 แท็ก

```
[('12-14', 'H_ID'),('ท่าดินแดง 2', 'ALLEY'),('คลองสาน', 'DIST'),('คลองสาน',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE')]
[('43', 'H_ID'),('ท่าดินแดง 5', 'ALLEY'),('ท่าดินแดง', 'ROAD'),('สมเด็จเจ้าพระยา', 'DIST'),('คลองสาน',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10600', 'POSTCODE')]
[('10', 'H_ID'),('ท่าเตียน', 'ALLEY'),('มหาราช', 'ROAD'),('พระบรมมหาราชวัง', 'DIST'),('พระนคร',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10200', 'POSTCODE')]
[('38', 'H_ID'),('ท่าพระจันทร์', 'ALLEY'),('มหาราช', 'ROAD'),('พระบรมมหาราชวัง', 'DIST'),('พระนคร',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10200', 'POSTCODE')]
[('1201/104', 'H_ID'),('ทาวน์อินทาวน์ 16', 'ALLEY'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10310', 'POSTCODE')]
[('1213/66', 'H_ID'),('ทาวน์อินทาวน์ 2', 'ALLEY'),('ลาดพร้าว', 'ROAD'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10310', 'POSTCODE')]
[('1467', 'H_ID'),('ทาวน์อินทาวน์ 3', 'ALLEY'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE')]
[('1375', 'H_ID'),('ทาวน์อินทาวน์ 3', 'ALLEY'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10310', 'POSTCODE')]
[('1493', 'H_ID'),('ทาวน์อินทาวน์ 3/2', 'ALLEY'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10310', 'POSTCODE')]
[('1213/383', 'H_ID'),('ทาวน์อินทาวน์ 3/3', 'ALLEY'),('พลับพลา', 'DIST'),('วังทองหลาง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10310', 'POSTCODE')]
[('51/99', 'H_ID'),('ทินกร', 'ALLEY'),('สุทธิสารวินิจฉัย', 'ROAD'),('ดินแดง', 'DIST'),('ดินแดง',  'ZONE'),('กรุงเทพมหานคร', 'PROVINCE'),('10400', 'POSTCODE')]
```

รูปที่ 7 *: Bi-LSTM_CRF Dataset*

3. Words Correcting Dataset เป็นชุดข้อมูลคำภาษาไทยสำหรับใช้ในการเปรียบเทียบและแทนคำเขียนผิดด้วยคำที่ถูกต้อง จำนวน 65,777 คำ

```
words[:50]

array(['กก', 'กกกอก', 'กกกาแล', 'กกกุง', 'กกขนาก', 'กกจาน', 'กกช้าง',
       'กกดู่', 'กกตาล', 'กกตูม', 'กกต้อง', 'กกทอง', 'กกธูป', 'กกบก',
       'กกบกเก่า', 'กกบกใหม่', 'กกบาก', 'กกปลาซิว', 'กกมะขามใหญ่',
       'กกสะทอน', 'กกสับ', 'กกหู', 'กกุธภัณฑ์', 'กกเกลือ', 'กกเสียว',
       'กกแก้ว', 'กกแก้วบูรพา', 'กกแดง', 'กกแต้ใหญ่', 'กกแรต', 'กกโก',
       'กกโท้ง', 'กกโพธิ์', 'กกไทร', 'กกไม้แดง', 'กง', 'กงกอน', 'กงกะยาง',
       'กงกาง', 'กงการ', 'กงค้าง', 'กงจักร', 'กงฉาก', 'กงซีลัง', 'กงดาก',
       'กงทอง', 'กงพะเนียง', 'กงพัด', 'กงพาน', 'กงรถ'], dtype=object)
```

รูปที่ 8 *: Words Correcting Dataset*

4. DZPP Dataset เป็นชุดข้อมูลความสัมพันธ์ระหว่าง ตำบล, อำเภอ, จังหวัด และรหัสไปรษณีย์ทั้งหมดในประเทศไทย จำนวน 8,496 ความสัมพันธ์

	DIST	ZONE	PROVINCE	POSTCODE
0	พระโขนง	คลองเตย	กรุงเทพมหานคร	10110
1	คลองตัน	คลองเตย	กรุงเทพมหานคร	10110
2	คลองเตย	คลองเตย	กรุงเทพมหานคร	10110
3	พระโขนง	คลองเตย	กรุงเทพมหานคร	10260
4	คลองเตย	คลองเตย	กรุงเทพมหานคร	10260
...	...	...	...	...
8491	สว่าง	สว่างวีระวงศ์	อุบลราชธานี	34190
8492	หนองบก	เหล่าเสือโก้ก	อุบลราชธานี	34000
8493	เหล่าเสือโก้ก	เหล่าเสือโก้ก	อุบลราชธานี	34000
8494	แพงใหญ่	เหล่าเสือโก้ก	อุบลราชธานี	34000
8495	โพนเมือง	เหล่าเสือโก้ก	อุบลราชธานี	34000

8496 rows × 4 columns

รูปที่ 9 *: DZPP Dataset*

5. Addrutf Dataset เป็นชุดข้อมูลที่มีการแบ่งแยกที่อยู่จากประโยค จำนวน 80,000 rows

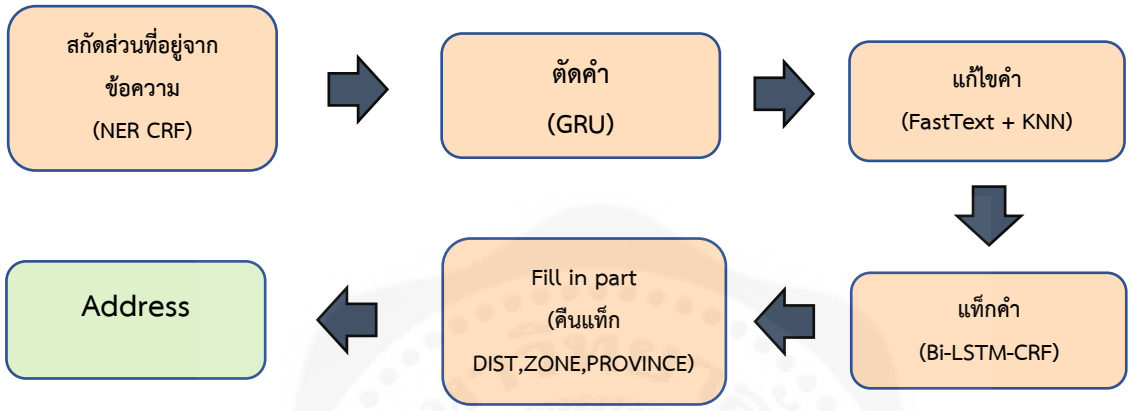
*addrutf - Notepad
File Edit Format View Help

สุชาดา เจริญสุข [ADDRESS]930 ม.5 เดอะวิณีหาดใหญ่ซิตี้รีสอร์ทแอนด์สปา ซ.ทรัพย์สมบูรณ์ ถ.สุริยะดำรงค์ ต.พระอาจารย์ อ.องครักษ์ จ.นครนายก 26
เพชรดา อมรพุฒิพงศ์ ที่อยู่จัดส่ง [ADDRESS]313 ม.11 บริษัท พัทยาบิวดิ้งแมนเนจเม้นท์ จำกัด ซ.ทรายทอง3ม.14 ทางหลวงแผ่นดินหมายเลข2 ต.หนอง
สัชมากร เมืองคุณ [ADDRESS]541/29 ม.5 บริษัท อาร์.เอส. เคมารีนแอนด์ไมนิ่ง จำกัด ซ.นครนายก-สาริกา1/1 ถ.แจ้งวัฒนะ ต.วังโบสถ์ อ.หนองไผ่ จ.เพ
โอนเรียบร้อยแล้วนะคะ ฝ่านธ.ไทยพาณิชย์ ยอดโอ [ADDRESS]567 ม.7 บจก.เฉลิมชัยดีเวลลอปเมนท์ ซ.ม.บ้านชเลรมย์ ถ.แสงหิรัญ ต.ดงหลวง อ.ดงหลวง
นส.รชรินทร์ ไชพณะ [ADDRESS]892/52 ม.11 บริษัท ฟิโลเอ็นจิเนียริ่งกรุ๊ป จำกัด ซ.ม.บ้านภัสสร์ไพรด์ศรีนครินทร์-เทพารักษ์ ถ.บ้านประจำรัง ต.แดงให
คุณเบญจมาศ เบ็ญจรูญ ม.พฤกษาวิลล์ [ADDRESS]415 ม.12 จิ้มจุ่มต้องมาจุ่ม ซ.ห้วยโจ้ใต้4 ทางหลวงแผ่นดินหมายเลข4060 ต.ห้วยแก อ.ชนบท จ.ขอนแ
เสาวณีย์ ทองรักษา [ADDRESS]149/71 ม.13 ธนวัฒน์แมนชั่น ซ.ม.บ้านไพร์มเพลสทาวน์ภูเก็ต-อนุสาวรีย์2 ถ.เทพรัตน ต.ทุ่งพระยา อ.สนามชัยเขต จ.ฉะ
ที่อยู่ในการจัดส่ง นางสาวจริยา หยูทองคำ ธนาคารกสิกรไทย [ADDRESS]734/95 ม.5 มิสซิกตี้เบสิกยินส์ ซ.รามคำแหง118แยก33-5 ถ.บรมธาตุ ต.ขุมโค
จิระนันท์ จันทร์มังกร [ADDRESS]204/43 ม.11 บริษัท เมกะเฮิร์ทซ์ จำกัด ซ.วงแหวนห้วเวียง ถ.เพิ่มสิน ต.ซอนสมบูรณ์ อ.หนองม่วง จ.ลพบุรี 15170[
นางสาวภัทรวดี ไกรจันทร์ มาหาวิทยาลัยราชภัฏกำแพงเพชร [ADDRESS]995 ม.10 เฟอ ซ.จันทรนิยม1 ถ.อบจ.บร.2028 ต.โคกสว่าง อ.ปลาปาก จ.นครพน
ชื่อ-ที่อยู่ กานต์ทิตา บุตรลบ [ADDRESS]959/21 ม.1 พัฒนาเด็กเนอรสรี่ ซ.ม.บ้านบ้านเอื้ออาทรลาดหลุมแก้ว2ซ.10 ทางหลวงชนบทขก.1023 ต.วังวน อ
ชื่อ เพ็ญผกา บุษบา [ADDRESS]4 ม.4 ที่ทำการผู้ใหญ่บ้าน ดอนมูลม.5 ซ.ม.บ้านบัวทองธานี8 ถ.โยธาธิการกส.3011 ต.เชิงคีรี อ.ศรีสาคร จ.นราธิวาส 962
วรางคณา ชิราวัธน์ [ADDRESS]980 ม.8 ไอเบอร์รี่ เซ็นทรัลพลาซา เวสต์เกต ซ.เจริญดิษฐ์อินทร์8 ถ.อุดมสุข(สุขุมวิท103) ต.ป่าต้น อ.แม่ทะ จ.ลำปาง 521
ที่อยู่ อนินท์พัฒ ไชยศิลป์ [ADDRESS]446/59 ม.13 นิโลบลนวดแผนโบราณ ซ.พาร์คอเวนิว5 ถ.ศรีประทีป ต.ดอนยาง อ.เมืองเพชรบุรี จ.เพชรบุรี 760

รูปที่ 10 *: Addrutf Dataset*

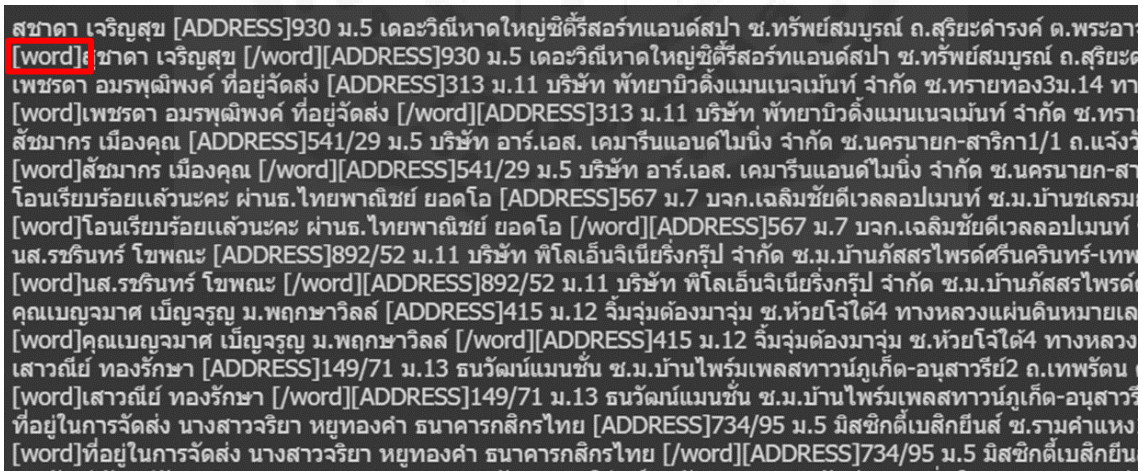
3.6 ขั้นตอนการดำเนินโครงงาน

ขั้นตอนการทำงานของระบบการใช้การเรียนรู้ของเครื่องตรวจสอบข้อความสืบค้นของโปรแกรม GeoSearch มี 5 กระบวนการดังนี้



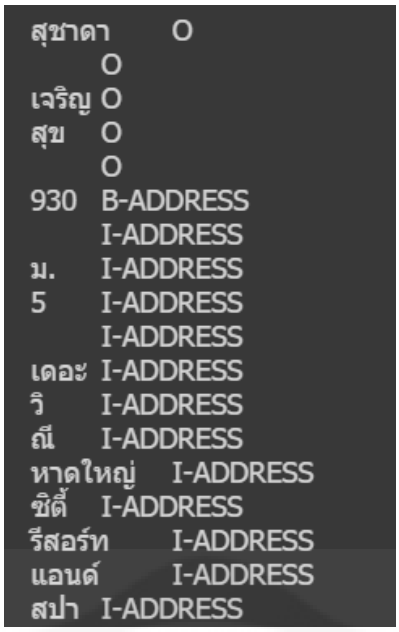
1. Named Entity Recognition

1. เตรียมชุดข้อมูลและทำการเติม Tag [word] เข้าไปในประโยคที่เป็นคำปกติ



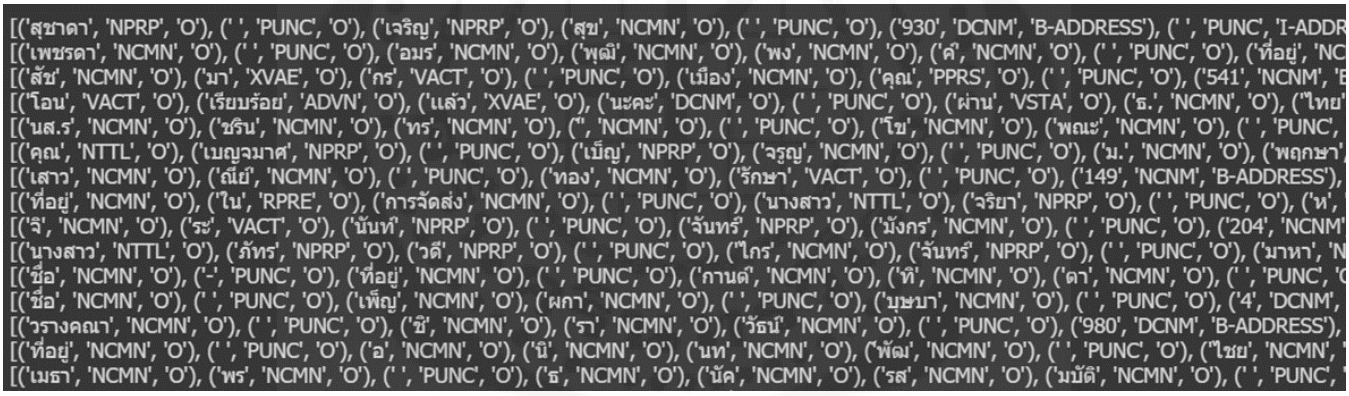
รูปที่ 11 : รูปภาพแสดง *Tag [word]*

2. ทำการแปลงคำแต่ละคำตามแท็กที่ให้ไว้แล้วบรรจุลงใน List โดยให้ Tag Word เป็น O, Tag Address เป็น I-Address เมื่อเป็นส่วนที่อยู่ และ B-Address เมื่อเป็นส่วนของเลขที่บ้าน



รูปที่ 12 *:* รูปภาพแสดง *Tag O* และ *I-Address*

3. นำแท็กที่ได้ไปผ่านการทำ Postag จะได้ข้อมูลที่พร้อมสำหรับการนำไปฝึกฝน CRF

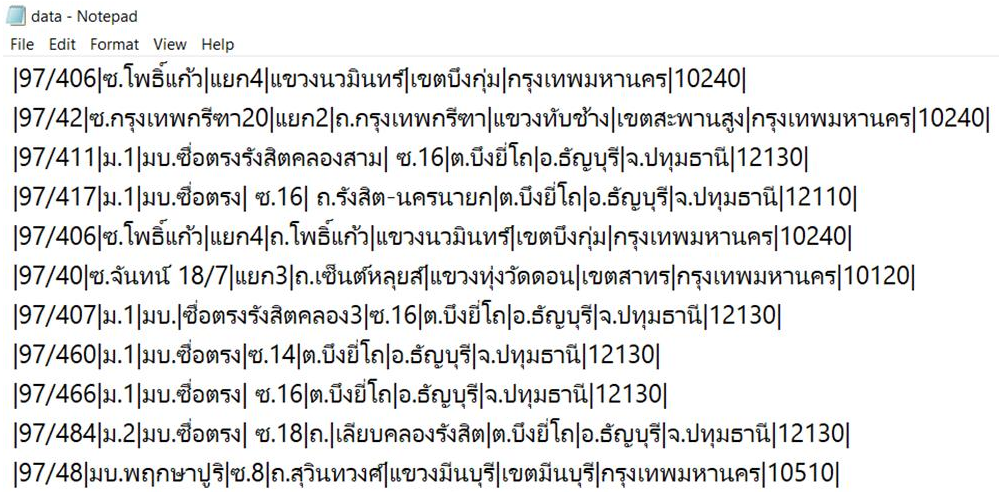


รูปที่ 13 *:* รูปภาพแสดงการทำ *Postag*

4. ทำการฝึกฝนโมเดล CRF โดยแบ่งข้อมูลสำหรับฝึกฝน 90 เปอร์เซ็นต์ ข้อมูลสำหรับทดสอบ 10 เปอร์เซ็นต์ โดยกำหนดพารามิเตอร์ดังนี้ Algorithm = ‘lbfgs’, c1 และ c2 = 0.1, max_iterations = 500

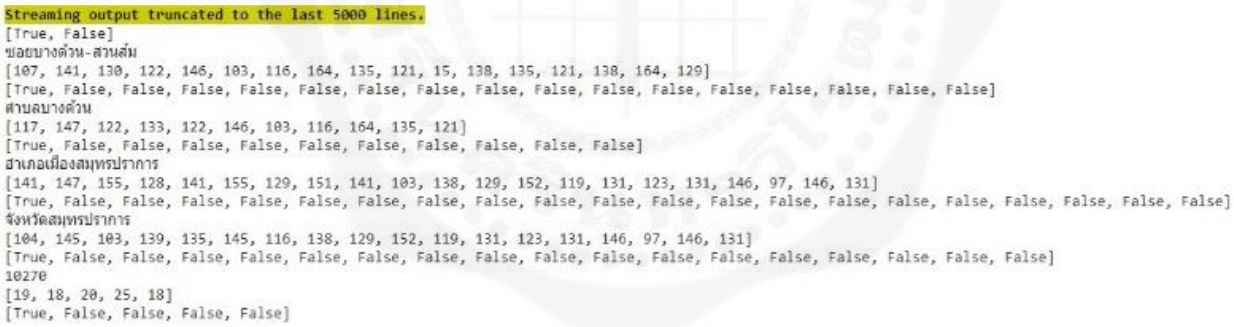
2. Recurrent Neural Network โดย GRU

1. นำชุดข้อมูลเริ่มต้นที่มีที่อยู่ที่มีคำแปลกปลอมอยู่ จากนั้นทำการใส่สัญลักษณ์ “ | ” คั่นหน้าหลังของในแต่ละคำและนำข้อมูลทั้งหมดบันทึกลงในไฟล์ข้อความ จะได้ดังรูปที่ 14



รูปที่ 14 *: Dataset* ที่ใส่สัญลักษณ์ / คั่นหน้าหลังของคำ

2. อ่านไฟล์ข้อความที่ละบรรทัดทำการตัดคำจากสัญลักษณ์ “|” เพื่อให้ได้คำแต่ละคำมาเก็บไว้ใน Array หลังจากนั้นทำการแปลงแต่ละคำให้เป็นตัวเลขโดยแปลงที่ละตัวอักษรเทียบกับตัวเลข ดังรูปที่ 15
3. จากนั้นทำการสร้าง Label โดยให้คำ 1 คำ มี label เป็น 1 ข้างหน้าหลังจากนั้นจะตามด้วย 0 ทั้งหมดซึ่ง 1 ก็คือส่วนที่บอกว่าต้องตัดคำ ดังรูป



รูปที่ 15

4. สร้าง SequenceExample โดยใน Feature ของ Sequence จะมี 3 อย่างคือ ความยาวของประโยค, Character Sentence ที่เป็นตัวเลข และ ประโยค Label ที่เป็น 1 และ 0 จากนั้นนำ Sequence ที่ได้ไปเขียนลงไฟล์ tfrecord โดยสุ่มเขียนลง Train 90 % และ Validation 10 %

รูปที่ 16 *:* รูปภาพแสดงตัวอย่าง *SequenceExample*

5. ทำการอ่านไฟล์ tfrecord เพื่อนำไปฝึกฝน ในแบบจำลอง RNN โดยใช้ GPU ในการตัดคำ โดยพารามิเตอร์ ดังนี้ Dropout = 0.5 , State Size = 128, Learning Rate = 0.001
6. ทำการฝึกฝนข้อมูลโดยการฝึกฝนข้อมูลไปเรื่อย ๆ แต่ละครั้งจะมีการ Optimum Learning Rate จากค่า Loss และเมื่อครบ 100 ครั้งจะทำการหาเฉลี่ยของการฝึกฝนมาและทำการ Save Model ไว้สำหรับใช้งาน สามารถหยุดการฝึกฝนได้เมื่อได้ผลลัพธ์ที่พอใจ

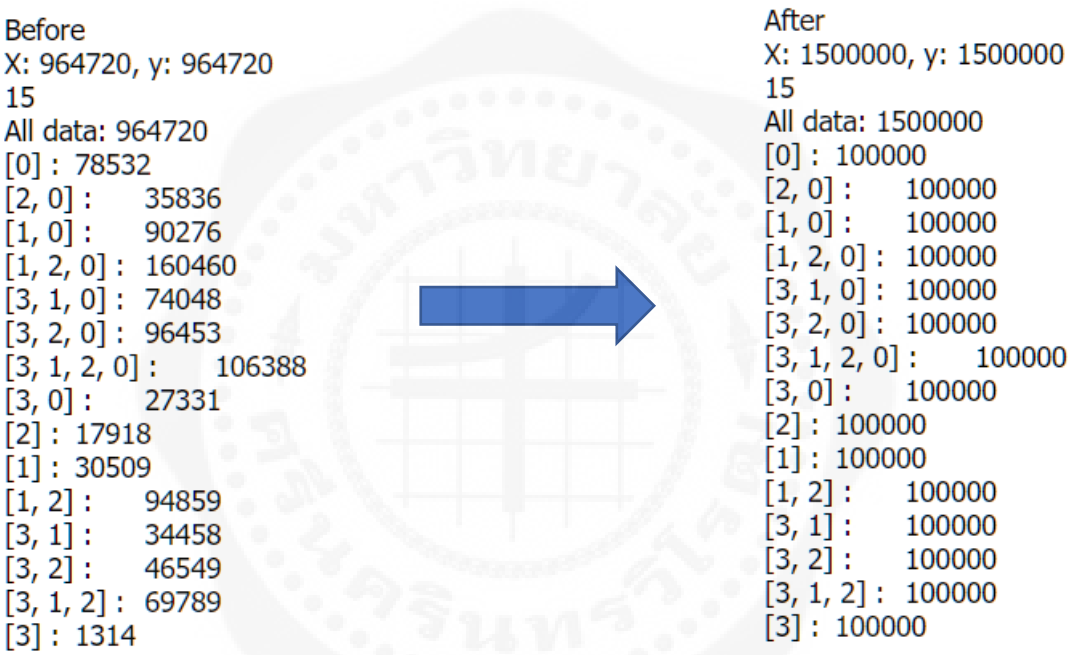
3. Bi-Directional Long Short Term Memory-Conditional Random Fields (Bi-LSTM-CRF)

1. เตรียมข้อมูลจาก Bi-LSTM_CRF Dataset โดยทำการเปลี่ยนแท็ก 'DIST', 'ZONE' และ 'PROVINCE' เป็นแท็ก 'dzp' เพื่อลดความจำนวนแท็ก และทำการแยกข้อมูลที่มีแท็ก 'H_ID' , 'MOO' และ 'POSTCODE' ออก เนื่องจากข้อมูลดังกล่าวมีรูปแบบที่แน่นอน จึงสามารถใช้ Regular expression ในการตรวจจับแทนแบบจำลองได้ อีกทั้งยังช่วยลดความซับซ้อนของแบบจำลอง ซึ่งหลังผ่านกระบวนการดังกล่าวจะเหลือแท็กสำหรับการฝึกฝนแบบจำลองจำนวน 4 แท็ก ได้แก่ 'H_NAME', 'ALLEY', 'ROAD' และ 'dzp'

2. ทำการแปลงข้อมูลที่อยู่เป็นตัวเลขโดยผ่านแบบจำลองการแก้ไขคำ ซึ่งจะได้ตัวเลขแทนของแต่ละข้อความนั้น และแทนข้อมูลแท็กด้วยตัวเลขดังนี้

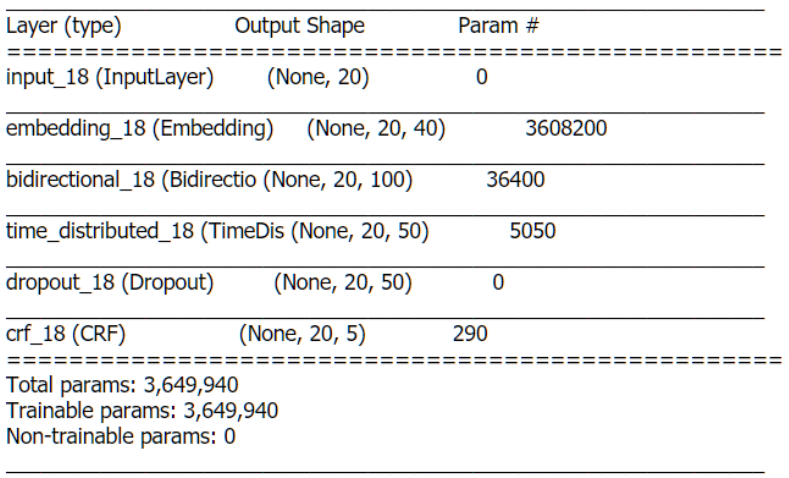
    - 'dzp': 0
    - 'ALLEY': 1
    - 'ROAD': 2
    - 'H_NAME': 3

3. ทำการ Oversampling ข้อมูลเพื่อให้แต่ละลำดับแท็กมีจำนวนเท่ากัน คือ ลำดับแท็กละ 100,000 ข้อมูล



รูปที่ *17 :* ภาพการทำ *oversampling*

4. ใช้ pad_sequences() ซึ่งอยู่ใน keras.preprocessing.sequence เพื่อทำให้ทุกข้อมูลมีขนาดเดียวกัน โดยกำหนดขนาดเท่ากับ 20 (maxlen=20) เมื่อเรียบร้อยจึงนำข้อมูลไปฝึกฝนแบบจำลอง โดยแต่ละชั้นของแบบจำลอง Bi-LSTM-CRF มีรายละเอียดดังนี้



Layer (type)	Output Shape	Param #
input_18 (InputLayer)	(None, 20)	0
embedding_18 (Embedding)	(None, 20, 40)	3608200
bidirectional_18 (Bidirectio	(None, 20, 100)	36400
time_distributed_18 (TimeDis	(None, 20, 50)	5050
dropout_18 (Dropout)	(None, 20, 50)	0
crf_18 (CRF)	(None, 20, 5)	290

Total params: 3,649,940
Trainable params: 3,649,940
Non-trainable params: 0

รูปที่ *18 :* ภาพสรุปการทำงานของ *Bi-LSTM-CRF*

4. K-Nearest Neighbours (K-NN)

1. เตรียมข้อมูลจาก Words Correcting Dataset โดยทำการแยกตัวอักษรในแต่ละคำด้วยช่องว่างและนำมาฝึกฝนกับ Fast Text เพื่อให้ได้ Model สำหรับแปลงคำเป็น Feature Vector โดยใช้คำสั่ง train_unsupervised() ของ Fast Text กำหนดค่า Parameter ดังนี้

   - epoch = 200
   - ws = 3

2. นำแบบจำลองที่ได้จากข้อ 1 มาแปลงคำศัพท์ทั้งหมดใน Words Correcting Dataset ให้เป็น Feature Vector เพื่อใช้สำหรับฝึกฝน K-NN
3. นำ Feature Vector ที่ได้จากข้อ 2 และ Words Correcting Dataset มาฝึกฝน K-NN โดยกำหนดให้ Feature Vector เป็นค่า Input ส่วน Words Correcting Dataset เป็นค่า Output
4. นำ K-NN Model ที่ฝึกฝนเรียบร้อยมาทดสอบเพื่อวัดประสิทธิภาพ

5. Fill in Part

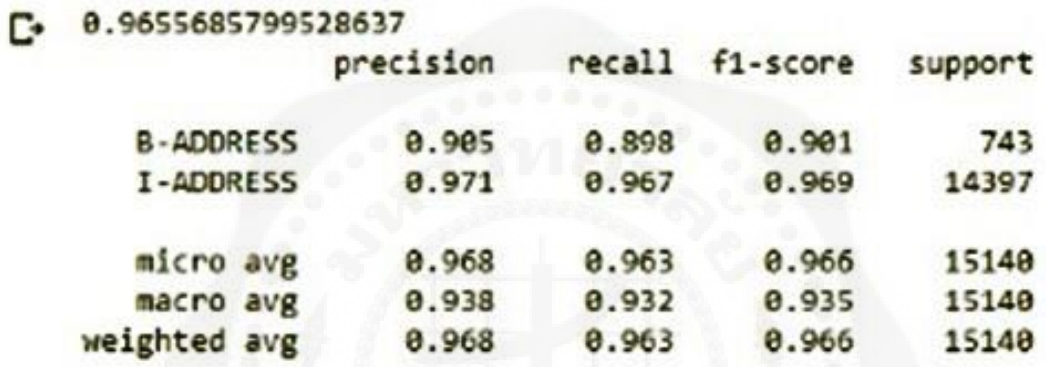
เป็นส่วนสุดท้ายของกระบวนการ ซึ่งเป็นขั้นตอนต่อจากการแก้ไขคำผิดด้วย K-NN โดยจะนำผลลัพธ์ที่ได้เทียบกับชุดข้อมูล DZPP Dataset ที่จัดอยู่ในรูปแบบตารางเพื่อแก้ไขหรือเพิ่มเติมข้อมูลที่อยู่ให้สมบูรณ์ยิ่งขึ้นก่อนส่งค่าออกเป็น Output

บทที่ 4
ผลการดำเนินโครงงาน

การดำเนินโครงงานในครั้งนี้ผู้จัดทำได้เสนอผลการดำเนินโครงงาน โดยแสดงผลลัพธ์ดังต่อไปนี้

4.1 ผลการดำเนินงาน Named Entity Recognition

ในการฝึกฝนข้อมูลด้วย CRF เรากำหนดพารามิเตอร์โดยใช้ อัลกอริทึม Limited Memory BFGS ใช้ c1 และ c2 เท่ากับ 0.1 และ Max Iterations เท่ากับ 500 ซึ่งได้ผลลัพธ์ค่า F1-Score อยู่ที่ประมาณ 0.96



0.9655685799528637

	precision	recall	f1-score	support
B-ADDRESS	0.905	0.898	0.901	743
I-ADDRESS	0.971	0.967	0.969	14397
micro avg	0.968	0.963	0.966	15140
macro avg	0.938	0.932	0.935	15140
weighted avg	0.968	0.963	0.966	15140

รูปที่ 19 *:* รูปภาพแสดงประสิทธิภาพของการทำ *Named Entity Recognition*

ทำการฝึกฝนแบบจำลองและปรับพารามิเตอร์ เมื่อนำแบบจำลองไปทดสอบกับไฟล์ Test Dataset และ CDG Dataset อีกครั้งได้ผลลัพธ์สรุปดังนี้

ชุดข้อมูล	ค่าความถูกต้องแบบจำลองเก่า	ค่าความถูกต้องแบบจำลองใหม่
Test Dataset 5000 rows	2800 (56%)	4213 (84.26%)
CDG Dataset 8069 rows	6676 (82.73%)	6676 (82.73%)

ตารางที่ 2 ตารางเปรียบเทียบประสิทธิภาพ Name Entity Recognition *ของแบบจำลองเก่าและใหม่*

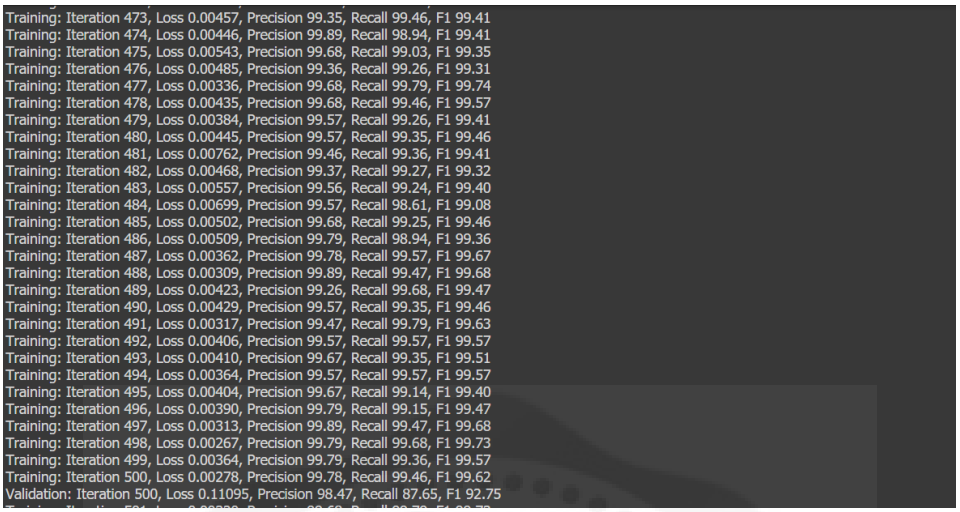
4.2 ผลการดำเนินงานโมเดลตัดคำ (Gate Recurrent Unit : GRU)

ผลลัพธ์ที่ได้จากการฝึกฝนแบบจำลองการตัดคำซึ่งฝึกฝนทั้งหมด 500 ครั้ง ได้ผลลัพธ์การประเมินผลดังต่อไปนี้

1.1 ประเมินผลโดยใช้ Precision ได้ผลลัพธ์ประมาณ 98.47

1.2 ประเมินผลโดยใช้ Recall ได้ผลลัพธ์ประมาณ 87.65

1.3 ประเมินผลโดยใช้ F1-Score ได้ผลลัพธ์ประมาณ 92.75



รูปที่ 20 *:* ผลลัพธ์ที่ได้จากการฝึกฝนข้อมูล 500 ครั้งของการตัดคำ

เมื่อนำแบบจำลองไปทดสอบกับไฟล์ Test Dataset ที่สร้างขึ้นมาด้วยวิธีการสุ่มคำพบว่ามีความแม่นยำค่อนข้างต่ำอยู่ที่ประมาณ ข้อมูลทั้งหมด 5000 rows สามารถตัดถูกต้องอยู่ที่ประมาณ 500 Rows คิดเป็น 10 เปอร์เซ็นต์ ดังนั้นเราจึงทำการปรับแบบจำลองและเพิ่มข้อมูลที่ใช้สำหรับฝึกฝน โดยฝึกฝนประมาณ 500 ครั้ง ได้ผลลัพธ์การประเมินผล ดังต่อไปนี้

1.1 ประเมินผลโดยใช้ Precision ได้ผลลัพธ์ประมาณ 99.80

1.2 ประเมินผลโดยใช้ Recall ได้ผลลัพธ์ประมาณ 92.61

1.3 ประเมินผลโดยใช้ F1-Score ได้ผลลัพธ์ประมาณ 96.07

เมื่อนำแบบจำลองไปทดสอบกับไฟล์ Test Dataset อีกครั้งจะได้ผลลัพธ์ดีกว่าครั้งก่อนคือ จากข้อมูล 5000 rows จะสามารถตัดถูกต้องประมาณ 4174 Rows คิดเป็น 82.94 เปอร์เซ็นต์และนำแบบจำลองไปทดสอบกับไฟล์ Dataset ที่ทาง CDG มีอยู่จะได้ผลลัพธ์สรุปดังนี้

ชุดข้อมูล	ค่าความแม่นยำแบบจำลองเก่า	ค่าความแม่นยำแบบจำลองใหม่
Test Dataset จำนวน 5000 rows	500 (10%)	4174 (83.48%)
CDG Dataset จำนวน 200 rows	138 (69%)	158 (79%)

ตารางที่ 3 ตารางเปรียบเทียบประสิทธิภาพผลลัพธ์จากการตัดคำของแบบจำลองเก่าและใหม่

4.3 ผลการดำเนินงาน Bi-Directional Long Short Term Memory-Conditional Random Fields (Bi-LSTM-CRF)

ทำการทดสอบและเปรียบเทียบประสิทธิภาพแบบจำลอง Bi-LSTM-CRF กับแบบจำลอง CRF ซึ่งเป็นแบบจำลองของครั้งก่อน โดยได้ผลลัพธ์ดังนี้

	precision	recall	f1-score	support
ALLEY	0.64	0.54	0.59	7503
H_NAME	0.21	0.69	0.32	1271
PAD	1.00	1.00	1.00	155257
ROAD	0.81	0.71	0.76	5978
dzp	1.00	0.96	0.98	29991
accuracy			0.97	200000
macro avg	0.73	0.78	0.73	200000
weighted avg	0.98	0.97	0.97	200000

ประสิทธิภาพของแบบจำลอง Bi-LSTM-CRF

	precision	recall	f1-score	support
ALLEY	0.95	0.87	0.91	7503
H_ID	0.00	0.00	0.00	0
H_NAME	1.00	0.98	0.99	1271
POSTCODE	0.00	0.00	0.00	0
ROAD	0.97	0.89	0.93	5978
ULESS	0.00	0.00	0.00	0
dzp	0.99	1.00	0.99	29991
accuracy			0.96	44743
macro avg	0.56	0.53	0.55	44743
weighted avg	0.98	0.96	0.97	44743

ประสิทธิภาพของแบบจำลอง CRF

รูปที่ 21 : รูปภาพแสดงการเปรียบเทียบประสิทธิภาพของแบบจำลอง *Bi-LSTM-CRF* และ *CRF*

จากผลลัพธ์พบว่าแบบจำลอง Bi-LSTM-CRF ได้ค่า accuracy อยู่ที่ 0.97 ส่วนแบบจำลอง CRF ได้ค่า accuracy อยู่ที่ 0.96 เมื่อดูค่า precision, recall และ f1-score ของแต่ละแท็กพบว่าแบบจำลอง CRF ได้ค่าสูงกว่า ซึ่งอาจเป็นเกิดจากข้อมูลที่ใช้ในการฝึกฝนต่างกัน เนื่องจาก Bi-LSTM-CRF ฝึกฝนกับข้อมูลเพียง 33 จังหวัดทำให้คลังศัพท์ยังไม่ครอบคลุม จึงทำให้ในขั้นตอนแปลงข้อมูลที่อยู่เป็นตัวเลขก่อนเข้าแบบจำลอง

นั้นอาจเกิดความคลาดเคลื่อนได้ ส่วนแบบจำลอง CRF แม้โดยรอบดูมีประสิทธิภาพสูงกว่าแต่จะพบว่าแบบจำลอง CRF ได้มีการแท็ก 'H_ID', 'PROVINCE', 'ULESS' ซึ่งไม่มีอยู่ในชุดข้อมูล test set แสดงให้เห็นถึงปัญหา overfitting และนั่นทำให้เมื่อคำนวณค่า Macro average แบบจำลอง CRF จึงได้ค่าน้อยกว่าแบบจำลอง Bi-LSTM-CRF

4.4 ผลการดำเนินงาน K-Nearest Neighbours (K-NN)

เนื่องจากผลลัพธ์ที่ได้จาก K-NN Model คือ List ของคำศัพท์ที่ใกล้เคียงกับคำตั้งต้นจำนวน 5 คำ เพื่อป้องกันโอกาสที่คำที่ถูกต้องอาจไม่ใช่คำลำกับแรกที่คัดเลือกโดย K-NN Model ดังนั้นจึงได้มีการใช้เครื่องมือช่วยในการคัดเลือกคำที่เหมาะสมที่สุดจาก List ของคำศัพท์ที่ใกล้เคียงอีกครั้ง ซึ่งเครื่องมือที่ใช้ในการคัดเลือกคำมี 2 แบบเพื่อเปรียบเทียบประสิทธิภาพและเลือกใช้เครื่องมือที่เหมาะสมที่สุดต่อไป โดยเครื่องมือดังกล่าวได้แก่

- Edit distance คือการเปรียบความเหมือนของ 2 คำ โดยใช้การนับจำนวนครั้งในการแก้ไขตัวอักษรและเลือกคำที่มีจำนวนครั้งในการแก้ไขน้อยที่สุด

- MaxMatchWord() คือฟังก์ชันที่สร้างขึ้นเพื่อเปรียบเทียบความเหมือนของ 2 คำ ผลลัพธ์ที่จะได้คือค่าระหว่าง 0-1 และเลือกคำที่มีค่า MaxMatchWord() มากที่สุด โดยวิธีการคำนวณเป็นดังนี้

$$\frac{\textit{จำนวนคู่อักษรที่ตรงกันในคำตั้งต้นกับคำเปรียบเทียบ}}{(\textit{ความยาวคำตั้งต้น} - 1) + |\textit{ความยาวคำตั้งต้น} - \textit{ความยาวคำเปรียบเทียบ}|}$$

การทดสอบประสิทธิภาพ ได้สุ่มคำคลังศัพท์จาก Words Correcting Dataset จำนวน 22,144 คำ จากนั้นทำการสร้างคำผิดโดยการสุ่มเพิ่ม/ลดตัวอักษรในแต่ละคำ และเก็บคำตั้งต้นไว้เป็น Label ในการวัดความถูกต้องในการแก้ไขคำ

	incorrect words	correct word
0	เซ่อลไนส์	เซเลไนด์
1	เจณิญบรจน์	เจริญโรจน์
2	รกบท	หกบท
3	บางฮสียบ	บางเสียบ
4	แพร่งขษหยังพัดนก	แพร่งขาหยั่งพัฒนา
...	...	...
22139	กระบหร่ง	กระโหร่ง
22140	ซลียบคลองหัผดนเพลง	เลียบคลองหัวดนเพลง
22141	สบายวัฟย	สบายน้อย
22142	สฝฆดฝ่อง	สองคลอง
22143	ตรีสบ่าง	ตรีสว่าง

22144 rows × 2 columns

รูปที่ *22 :* รูปภาพแสดงคอลัมน์คำที่ผิดและคำที่ถูกต้อง

ซึ่งผลลัพธ์ในการแก้คำผิดของ K-NN Model ทำงานคู่กับ Edit Distance และ MaxMatchWord() สามารถสรุปเป็นตารางได้ดังนี้

ผลลัพธ์ของ k-nn ในการเลือกคำ		
จำนวนคำทั้งหมด	จำนวนคำที่ k-nn เลือกถูกต้อง	ร้อยละ
22,144	14,977	67.63

ตารางที่ 4 ผลลัพธ์การเลือกคำ K-NN

เครื่องมือ	ผลลัพธ์ของเครื่องมือในการเลือกคำ		
	จำนวนคำที่ k-nn เลือกถูกต้อง	จำนวนคำที่เครื่องมือเลือกถูกต้อง	ร้อยละ
Edit distance	14,977	14,278	95.33
MaxMatchWord()	14,977	13,513	90.22

ตารางที่ 5 ผลลัพธ์การเลือกคำระหว่าง Edit distance และ MaxMatchWord()

เครื่องมือ	ผลลัพธ์		
	จำนวนคำผิดทั้งหมด	จำนวนคำผิดที่แก้ได้	ร้อยละ
Edit distance	22,144	14,278	64.48
MaxMatchWord()	22,144	13,513	61.02

ตารางที่ 6 ตารางแสดงผลลัพธ์โดยรวมของ K-NN *ในการเลือกคำ*

จากผลลัพธ์พบว่า Edit Distance มีประสิทธิภาพในการเลือกคำที่ถูกต้องมากกว่า MaxMatchWord() ดังนั้นเครื่องมือที่จะนำมาคัดเลือกคำที่เหมาะสมที่สุดจาก List ซึ่งได้จาก K-NN คือ Edit Distance ที่มีประสิทธิภาพร้อยละ 95.33

บทที่ 5
สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการค้นคว้า

แบบจำลองในการสกัดที่อยู่จากประโยค ได้นำหลักการ Named Entity Recognition มาประยุกต์ใช้สำหรับแยกประโยคทั่วไปกับประโยคที่เป็นที่อยู่ แต่อาจจะเกิดผิดพลาดได้กรณีที่เจอคำแปลกมากเกินไปที่แบบจำลองไม่เคยเรียนรู้ซึ่งจากการฝึกฝนของแบบจำลองจะมีความแม่นยำร้อยละ 96.07 เมื่อนำไปทดสอบกับชุดข้อมูลพบว่าความแม่นยำในการสกัดคำลดน้อยลงเหลือ 84.26 สำหรับ test dataset และความแม่นยำในการสกัดคำลดน้อยลงเหลือ 82.73 สำหรับ cdg dataset ซึ่งอยู่ในเกณฑ์ที่คาดหวังไว้คือมีความถูกต้องร้อยละ 70 ขึ้นไป

แบบจำลองการตัดคำที่อยู่จากประโยค ได้นำหลักการ Recurrent Neural Network โดยใช้ GRU มาประยุกต์ใช้สำหรับการตัดคำจากประโยคเช่น การตัดแยกคำของ อำเภอและตำบล เพื่อแบ่งข้อมูลเป็นคำ ๆ และนำไปใช้เพื่อระบุว่าคำแต่ละชนิด เป็นที่อยู่หรือไม่ ซึ่งอาจจะเกิดข้อผิดพลาดเนื่องจากข้อมูลคำศัพท์ต่าง ๆ ในภาษาไทยนั้นมีมากมายอาจทำให้การตัดคำมีประสิทธิภาพลดลงตามคำที่ใช้ เช่นการใช้คำแสลงหรือคำเฉพาะพื้นที่ เป็นต้น ซึ่งได้ความแม่นยำในการฝึกฝนร้อยละ 99.80 เมื่อนำไปทดสอบกับชุดข้อมูลพบว่าความแม่นยำในการสกัดคำลดน้อยลงเหลือ 83.48 สำหรับ test dataset และความแม่นยำในการสกัดคำลดน้อยลงเหลือ 79 สำหรับ cdg dataset ซึ่งอยู่ในเกณฑ์ที่คาดหวังไว้คือมีความถูกต้อง 70 เปอร์เซ็นต์ขึ้นไป

แบบจำลองการแก้คำผิด ได้นำหลักการ Natural Language Processing เกี่ยวกับ Word Embeddings มาประยุกต์ในการแก้ไขคำผิด โดยอาศัยคุณสมบัติของ Word Vector ที่หากคำมีความคล้ายคลึงกันจะทำให้เมื่อแปลงเวกเตอร์จะอยู่ใกล้เคียงกัน มาใช้ในการหาคำที่ถูกต้องจากในคลังศัพท์ ซึ่งหลักการนี้มีปัญหาคือ หากคำที่ต้องการแก้เป็นคำที่พิมพ์ผิดไปจาดคำจริงที่ถูกต้องค่อยข้างมาก จะมีผลให้คำดังกล่าวเมื่อถูกแปลงในรูปเวกเตอร์ อาจไปใกล้เคียงกับเวกเตอร์ของคำอื่นซึ่งไม่ใช้คำที่ต้องการแทน ด้วยเหตุนี้จึงนำหลักการ Edit Distance มาช่วยในการเลือกคำที่ถูกต้อง ซึ่งจากผลการดำเนินโครงงานพบว่า การหาเลือกคำที่ถูกด้วย Edit Distance จากกลุ่มคำที่คัดด้วยความใกล้เคียงโดยอิงจากเวกเตอร์ในขั้นแรก ซึ่งได้ความแม่นยำถึงร้อยละ 95.33 และจากวิธีการข้างต้นทำให้นอกจากสามารถแก้ไขคำผิดได้แล้วยังสามารถคัดแยกคำที่ไม่ต้องการซึ่งปะปนอยู่ในข้อมูลที่อยู่ได้อีกด้วย

แบบจำลองการแท็กส่วนประกอบของที่อยู่ ได้นำหลักการ bidirectional LSTM (Bi-LSTM) มาประยุกต์ร่วมกับ Conditional random field (CRF) ซึ่งเป็นแบบจำลองทางด้าน Natural Language Processing มีความสามารถการแยกส่วนประกอบของบทความ โดย Bi-LSTM เป็นส่วนแรกในการรับข้อมูลก่อนทำการแปลงและสกัดข้อมูลแล้วส่งต่อให้ CRF ทำนายว่าข้อมูลดังกล่าวเป็นส่วนประกอบใดของข้อมูลที่อยู่ ซึ่งได้ค่าความถูกต้องร้อยละ 97 แต่หลักการดังกล่าวอาจเกิดความผิดพลาดขึ้นได้ในกรณีที่ข้อมูลมีคำที่ไม่เกี่ยวข้องกับที่อยู่จริงปะปน ซึ่งหลุดรอดมาจากกระบวนการแก้ไขคำ โดยเป็นผลเนื่องมาจากข้อมูลที่ใช้สำหรับการฝึกฝนนั้นอาจไม่ครอบคลุมมากพอ

5.2 อภิปรายผล

จากการดำเนินโครงงานทดลองพัฒนาแบบจำลองเพื่อขจัดคำที่ไม่เกี่ยวข้องกับข้อมูลที่อยู่พบว่าแบบจำลองของงานวิจัยนี้ ค่อนข้างมีความยุ่งยากและซับซ้อนในการทำ เนื่องจากมีแบบจำลองที่ต้องทำงานสัมพันธ์กันถึง 5 แบบจำลองทำให้เมื่อแบบจำลองในขั้นแรกเริ่มเกิดความผิดพลาดจะส่งผลต่อการทำงานของแบบจำลองถัดไป ดังนั้นในส่วนท้ายสุดของกระบวนการจึงมีฟังก์ชันสำหรับการตรวจสอบผลลัพธ์เพื่อลดและป้องกันความผิดพลาดอีกทีหนึ่ง โดยเป็นการเทียบผลลัพธ์กับตารางความสัมพันธ์ของข้อมูลตำบล อำเภอ จังหวัด และรหัสไปรษณีย์ เนื่องจากข้อมูลส่วนดังกล่าวมีความสำคัญอย่างมากในกระบวนการค้นหาระบุตำแหน่ง ซึ่งกระบวนในขั้นสุดท้ายนี้นอกจากช่วยตรวจสอบผลลัพธ์แล้ว ยังสามารถเพิ่มเติมข้อมูลที่สำคัญของที่อยู่ซึ่งอาจขาดหายไป ส่งผลให้กระบวนการถัดจากงานวิจัยที่นำผลลัพธ์ของแบบจำลองงานวิจัยนี้ไปดำเนินการค้นหาในระบบฐานข้อมูลมีความถูกต้องแม่นยำมากขึ้น ซึ่งถือเป็นประโยชน์ที่ได้รับจากงานวิจัย และบรรลุตามวัตถุประสงค์หลักของงานวิจัยที่ต้องการขจัดข้อมูลที่ไม่เกี่ยวข้องให้เหลือมีเพียงข้อมูลที่สำคัญของที่อยู่

5.3 ปัญหาและอุปสรรค

ไม่สามารถรวบรวมคำศัพท์ภาษาไทยเกี่ยวกับที่อยู่อาศัยเพื่อใช้สำหรับเปรียบเทียบและแก้ไขคำผิดทั้งหมดได้ เนื่องจากมีจำนวนมากและอาจทำให้ใช้เวลาประมวลผลนานยิ่งขึ้น

ในการฝึกฝนแบบจำลองบางชนิดค่อนข้างใช้เวลานานเนื่องจากมีข้อมูลในการฝึกฝนเยอะทำให้เสียเวลาในการทำงานค่อนข้างมาก

ข้อมูลที่ใช้สำหรับฝึกฝนไม่ครอบคลุมและอาจส่งผลต่อประสิทธิภาพของแบบจำลองเมื่อต้องทำงานกับข้อมูลที่ไม่เคยพบในการฝึกฝน

5.4 ข้อเสนอแนะ

ถ้าเพิ่มฟีเจอร์การเคลียร์คำขยะอื่น ๆ เพิ่มเติมทดลองใช้แบบจำลองอื่น ที่นอกเหนือจาก GRU และมีการตรวจสอบเพิ่มข้อมูลภาษาที่มีการเพิ่มขึ้นตลอดเวลาอาจจะมีโอกาสที่ประสิทธิภาพการตรวจจับจะสูงขึ้นมากกว่างานวิจัยนี้

เนื่องจากแบบจำลองของโครงงานรองรับเฉพาะข้อมูลที่เป็นภาษาไทย อาจสามารถนำหลักการทำงานของโครงงานนี้ไปต่อยอดในการทำกับชุดข้อมูลในภาษาอื่น ๆ หรือต่อยอดงานในด้านการประมวลผลภาษาธรรมชาติได้

บรรณานุกรม

[1] K. Surapong, “Recurrent Neural Network (RNN) คืออะไร Gated Recurrent Unit (GRU) คืออะไร สอนสร้าง RNN ถึง GRU ด้วยภาษา Python – NLP ep.9,” 2019. https://www.bualabs.com/archives/3103/what-is-rnn-recurrent-neural-network-what-is-gru-gated-recurrent-unit-teach-how-to-build-rnn-gru-with-python-nlp-ep-9/ (accessed Oct. 10, 2020).

[2] S. Tangruamsub, “Long Short-Term Memory (LSTM),” Jul. 04, 2017. https://medium.com/@sinart.t/long-short-term-memory-lstm-e6cb23b494c6 (accessed Jun. 11, 2020).

[3] “What is Named Entity Recognition (NER)?” https://www.techslang.com/definition/what-is-named-entity-recognition-ner/ (accessed Oct. 10, 2020).

[4] Lukkiddd, “Word Embedding และ Word2Vec คืออะไร,” 2018. https://lukkiddd.com/word-embedding-และ-word2vec-คืออะไร-e60bdf6d78d3 (accessed Jun. 11, 2020).

[5] David S. Batista, “Conditional Random Fields for Sequence Prediction.” http://www.davidsbatista.net/blog/2017/11/13/Conditional_Random_Fields/ (accessed Oct. 10, 2020).

[6] S. Panchal, “k Nearest Neighbor Classifier (kNN)-Machine Learning Algorithms | by Shubham Panchal | Medium,” 2018. https://equipintelligence.medium.com/k-nearest-neighbor-classifier-knn-machine-learning-algorithms-ed62feb86582 (accessed Oct. 10, 2020).

[7] N. SMETANIN, “Fuzzy string search,” Mar. 24, 2011. http://ntz-develop.blogspot.com/2011/03/fuzzy-string-search.html (accessed Jun. 11, 2020).

[8] J. Mueller and A. Thyagarajan, “Siamese recurrent architectures for learning sentence similarity,” *30th AAAI Conf. Artif. Intell. AAAI 2016*, no. 2014, pp. 2786–2792, 2016.

[9] ส. จันทา and น. ปรวัฒน์ปรียกร, “ระบบจำแนกและค้นคืนข้อมูลเว็บกระทู้ข่าว ด้วยโครงข่ายประสาทเทียมเปอร์เซ็ปตรอนแบบหลายชั้น,” 2013. Accessed: Jan. 18, 2021. [Online]. Available: www.pantip.com.

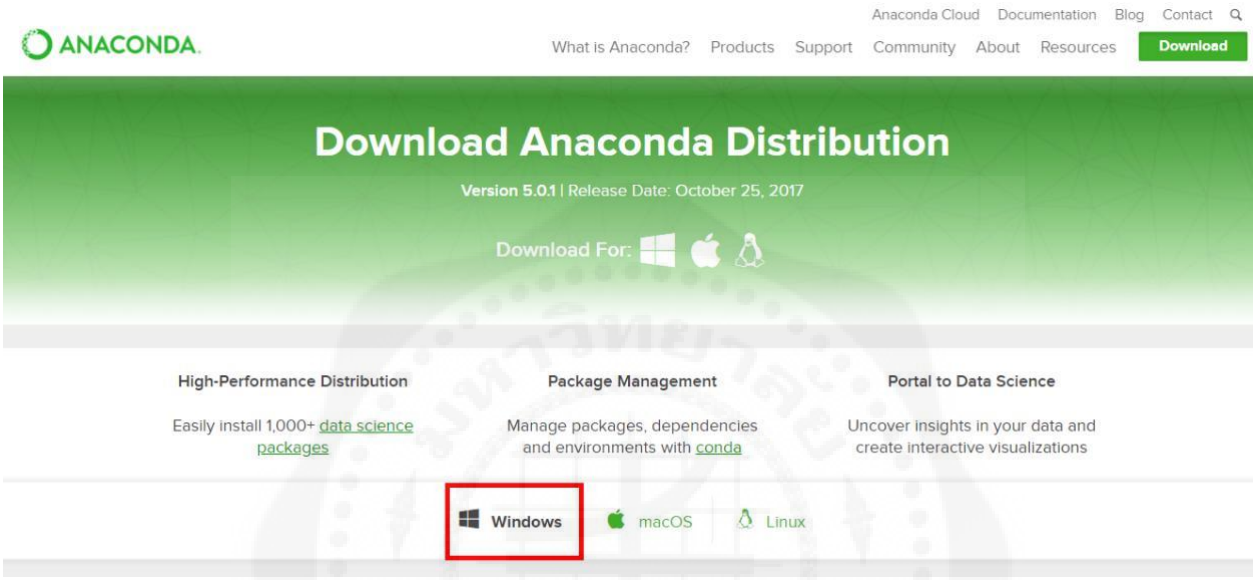
[10] S. Thattinaphanich and S. Prom-On, “Thai Named Entity Recognition Using Bi-LSTM-CRF with Word and Character Representation,” in *Proceedings of 2019 4th International Conference on Information Technology: Encompassing Intelligent Technology and Innovation Towards the New Era of Human Life, InCIT 2019*, Oct. 2019, pp. 149–154, doi: 10.1109/INCIT.2019.8912091.

ภาคผนวก

ภาคผนวก ก

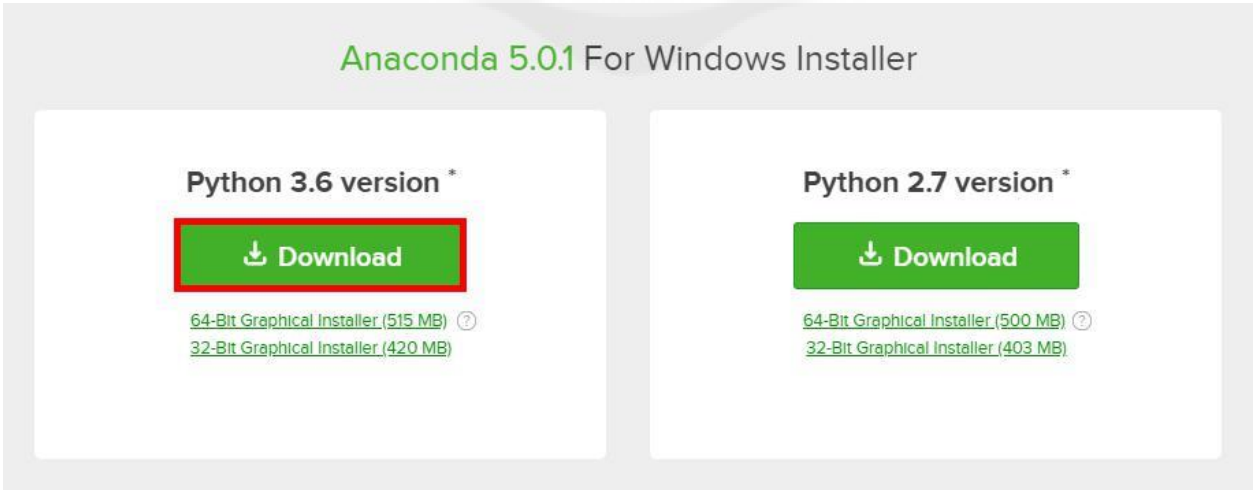
การติดตั้ง Anaconda

1. เข้าไปที่เว็บไซต์ https://www.anaconda.com/download/
2. เลือกดาวน์โหลดเวอร์ชันสำหรับระบบปฏิบัติการ Windows



รูปที่ *23* : รูปภาพแสดง*หน้าแรกของเว็บไซต์ตัวโปรแกรม*

3. คลิกที่ปุ่ม Download (Python 3.6 version)

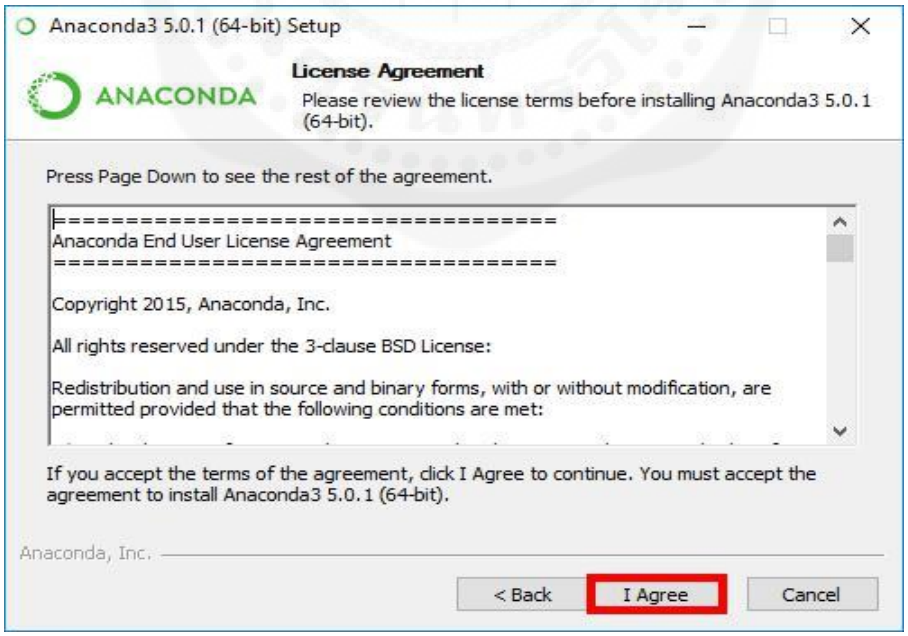


รูปที่ *24* : รูปภาพแสดง*หน้าของตัวดาวโหลดน์โปรแกรม*

4. ดับเบิลคลิกไฟล์ที่ดาวน์โหลดมาเพื่อติดตั้ง Anaconda
5. คลิกที่ปุ่ม Next

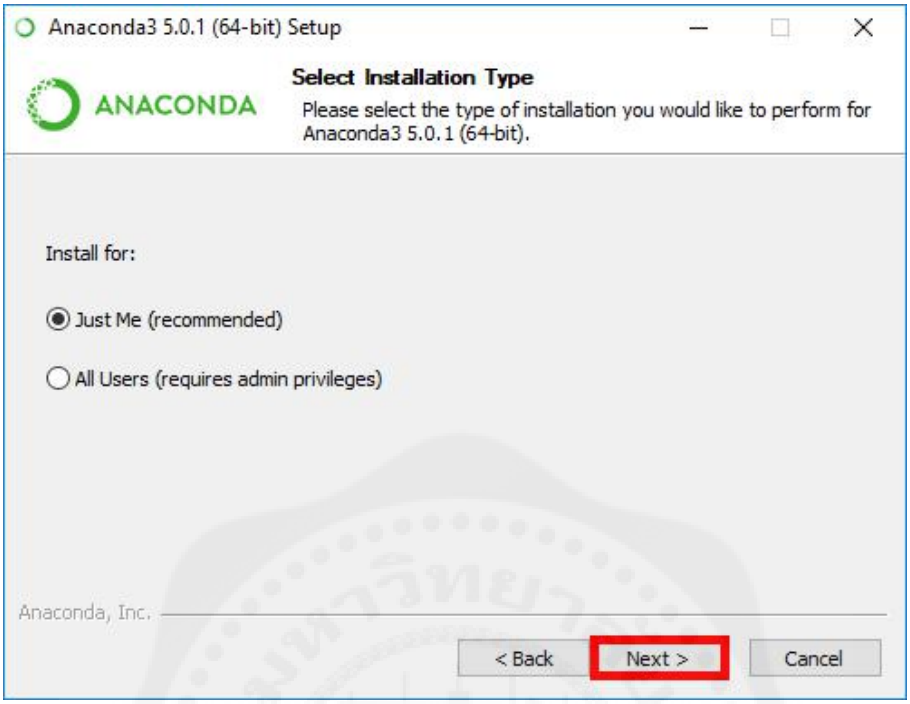
รูปที่ *25* : รูปภาพแสดง*หน้าต่างเมื่อเปิดตัวsetup*

6. คลิกที่ปุ่ม I Agree



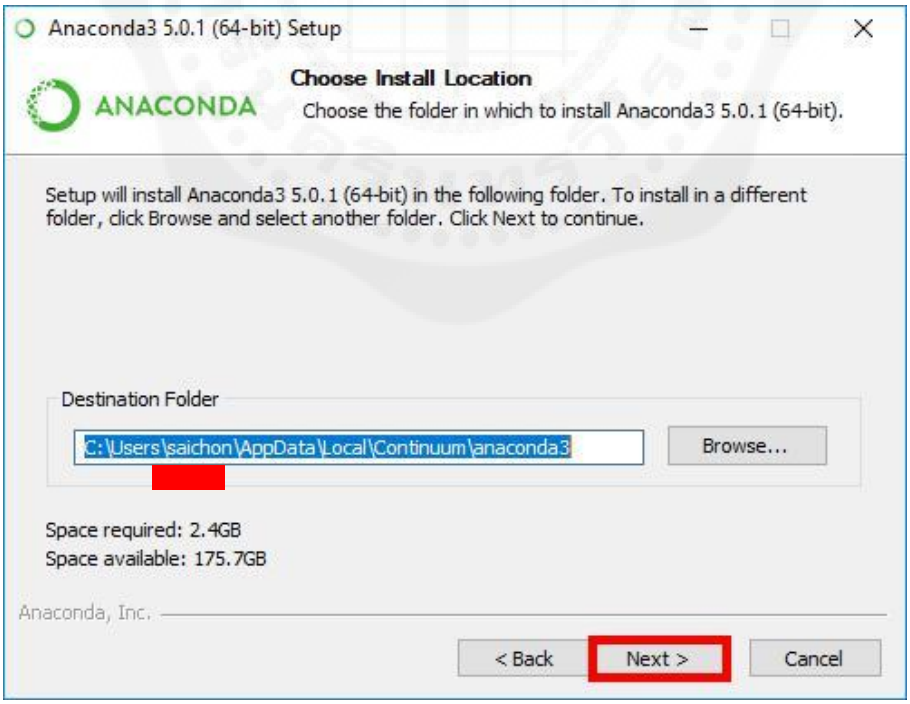
รูปที่ *26* : รูปภาพแสดง*หน้าต่างแสดงLicense*

7. เลือก Just Me (recommended) แล้วคลิกปุ่ม Next



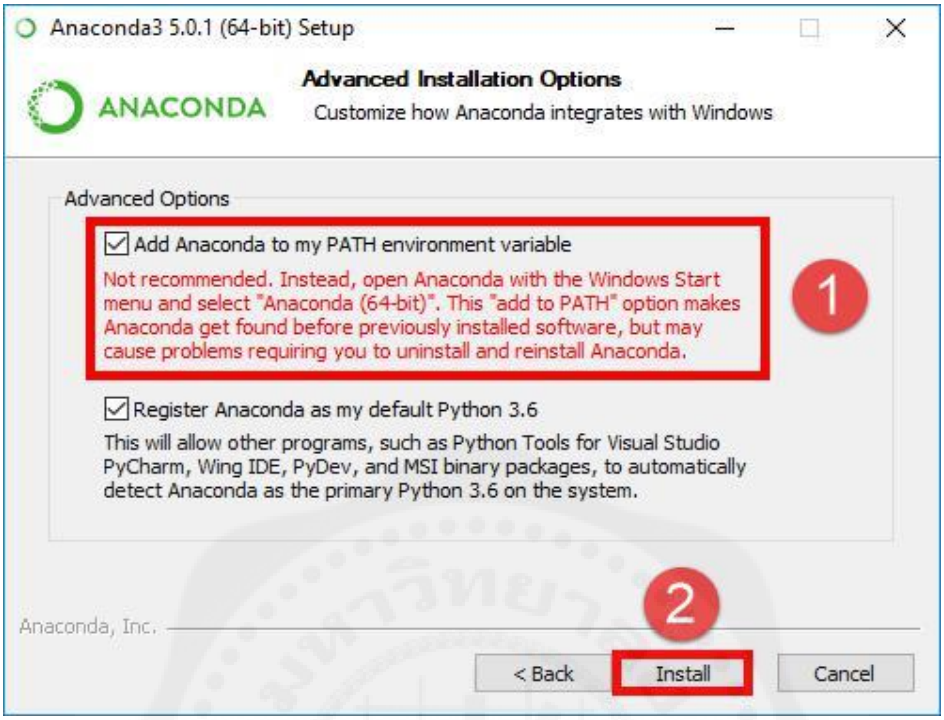
รูปที่ *27* : รูปภาพแสดง*หน้าต่างเลือกประเภทการลงโปรแกรม*

8. คลิกที่ปุ่ม Next



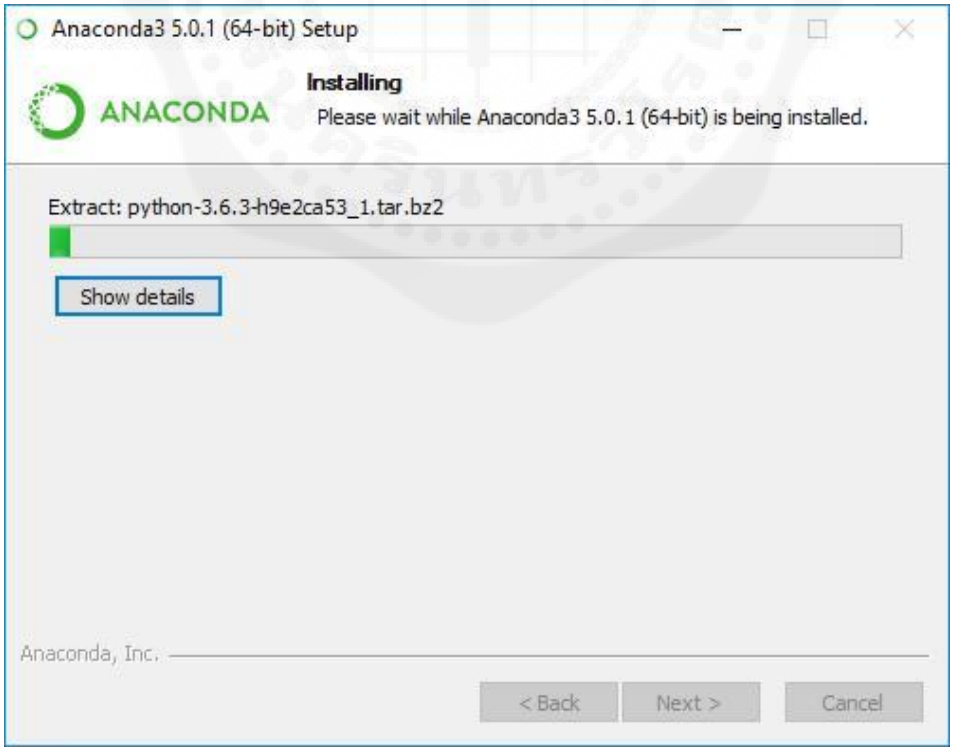
รูปที่ *28* : รูปภาพแสดง*หน้าต่างเลือกจุดที่จะลงโปรแกรม*

9. เลือก Add Anaconda to my PATH environment variable แล้วคลิกปุ่ม Install



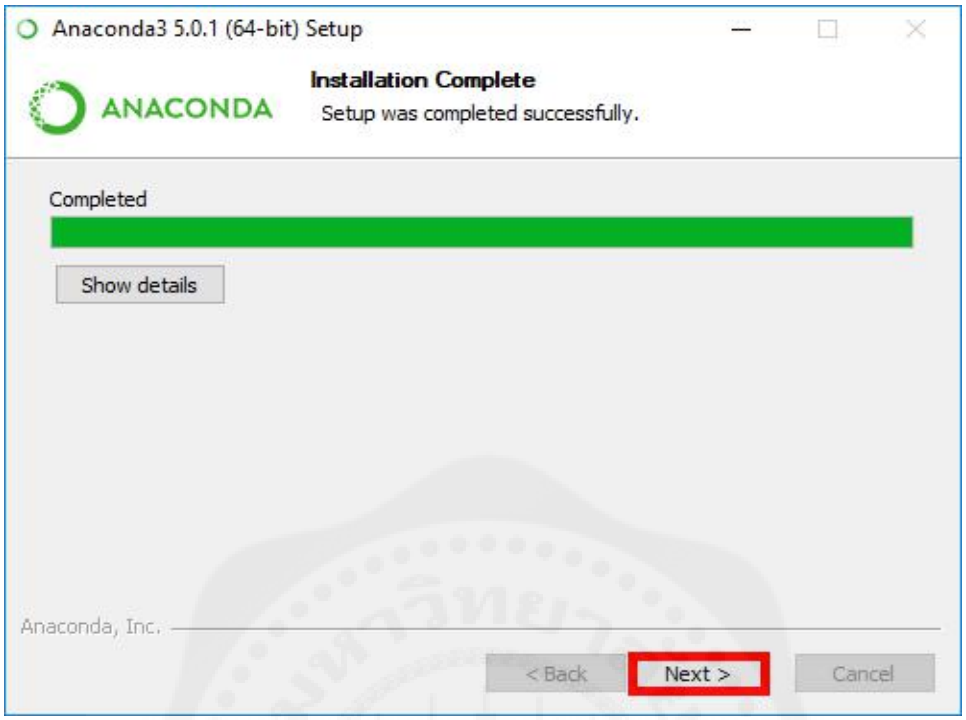
รูปที่ *29* : รูปภาพแสดง*หน้าต่างเลือก Path Environment*

10. รอจนกว่าการติดตั้งจะเสร็จสมบูรณ์



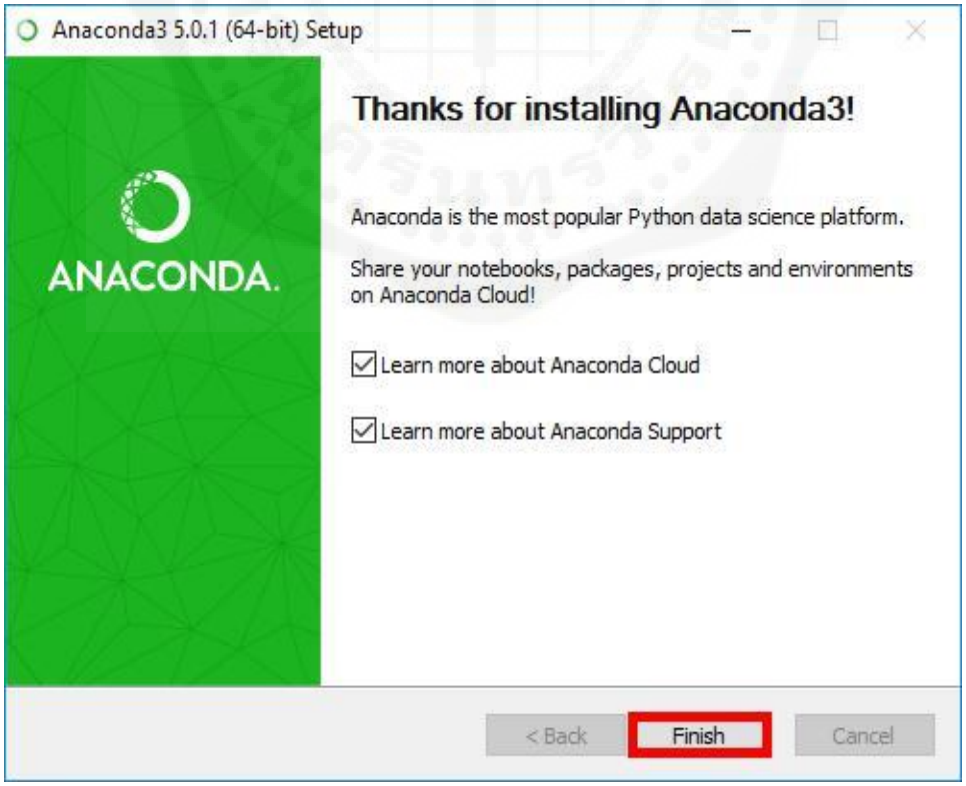
รูปที่ *30* : รูปภาพแสดง*หน้าต่างแสดงขณะกำลังติดตั้งโปรแกรม*

11. เมื่อติดตั้งเสร็จสมบูรณ์แล้วให้คลิกปุ่ม Next



รูปที่ *31 :* รูปภาพแสดง*หน้าต่างแสดงเมื่อติดตั้งโปรแกรมเสร็จสิ้น*

12. คลิกที่ปุ่ม Finish

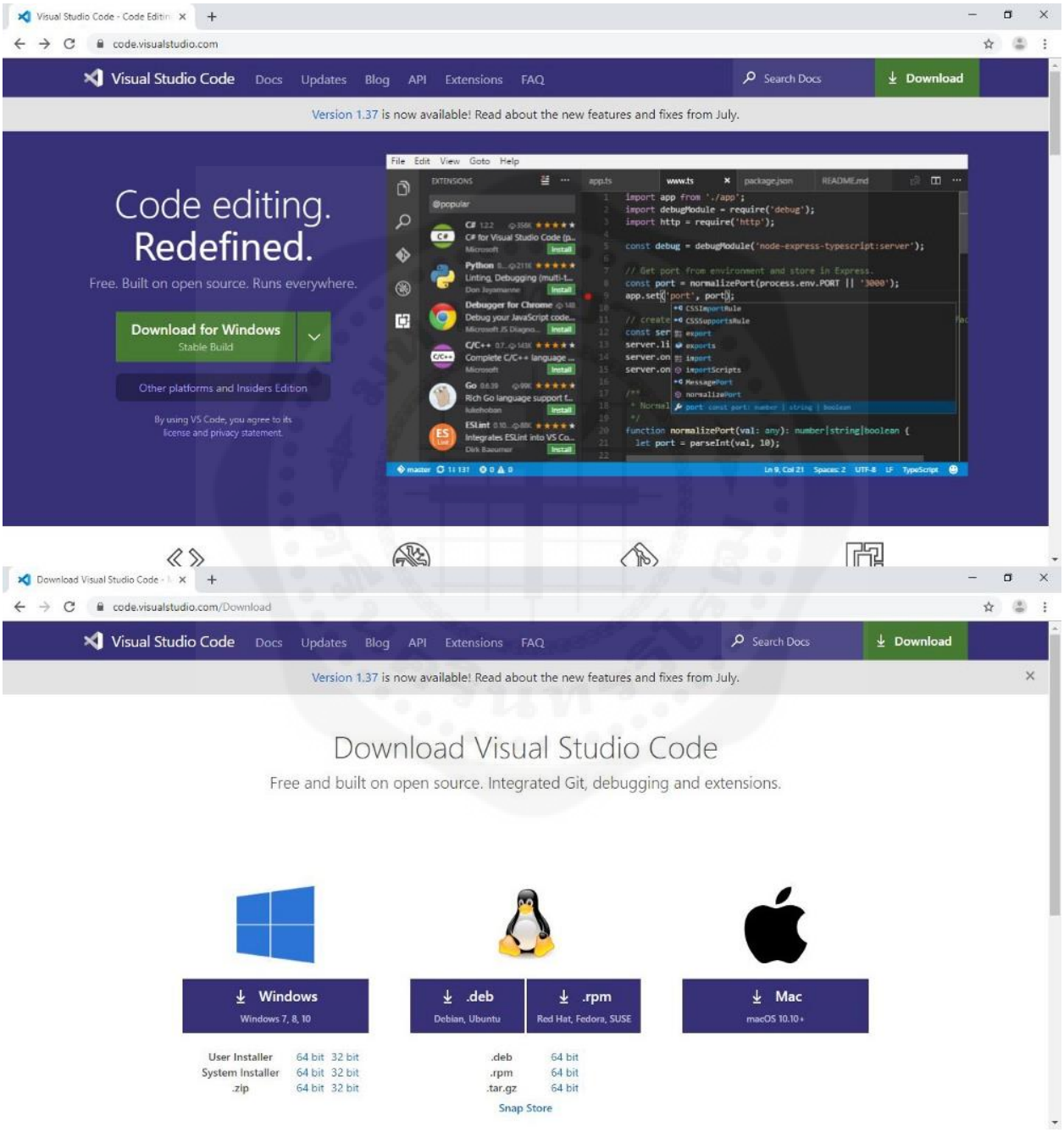


รูปที่ *32 :* รูปภาพแสดง*หน้าต่างแสดงเมื่อติดตั้งเสร็จเรียบร้อย*

ภาคผนวก ข

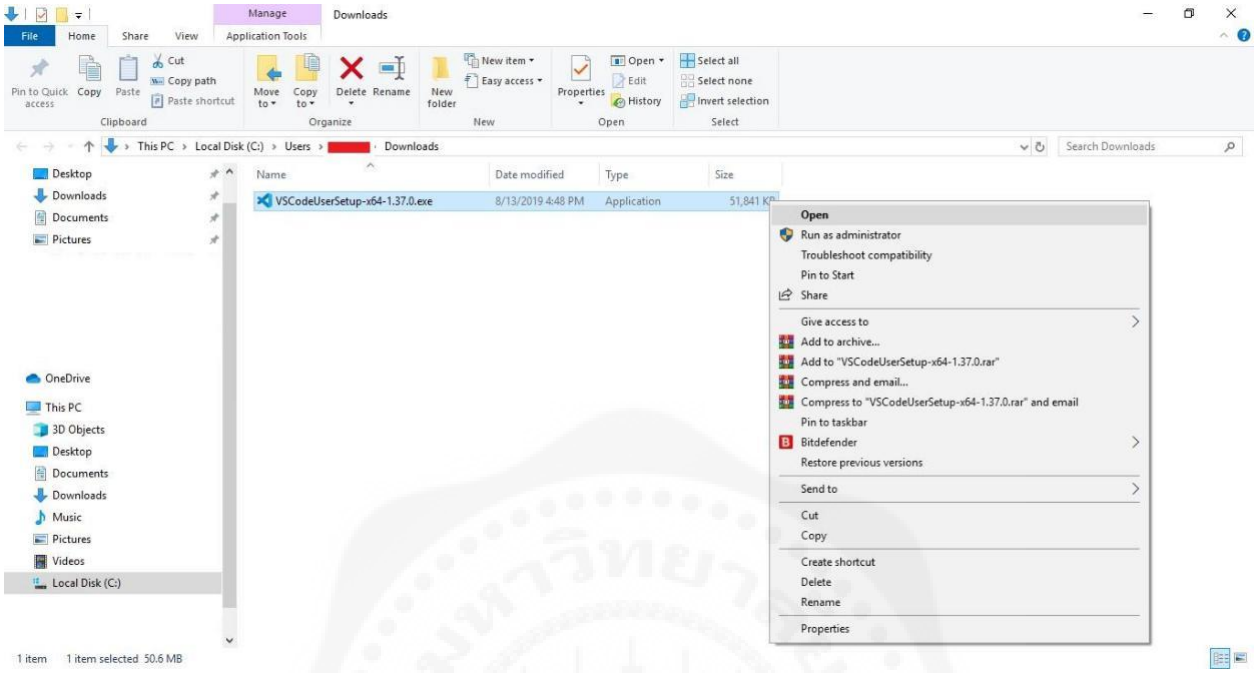
การติดตั้ง Visual Studio Code

1. เข้าไปที่เว็บไซต์ https://code.visualstudio.com/ และ Download โปรแกรม VS Code โดยเลือกให้ตรงกับ OS ของเครื่องคอมพิวเตอร์



รูปที่ *33 :* รูปภาพแสดง*ดาวน์โหลดตัวติดตั้ง Visual Studio Code*

2. ดับเบิ้ลคลิก หรือคลิกขวาและกด “Open” โปรแกรมที่ดาวน์โหลดมา

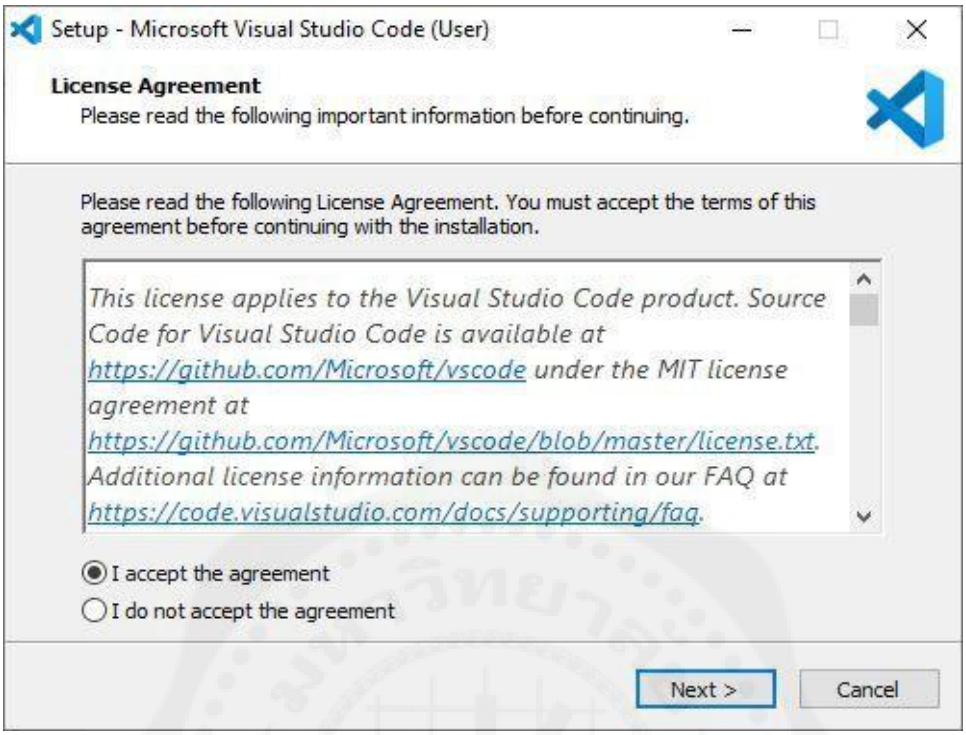


รูปที่ 34 : รูปภาพแสดง*โปรแกรมที่ดาวน์โหลดมาเพื่อติดตั้ง*

3. คลิกปุ่ม “Run”

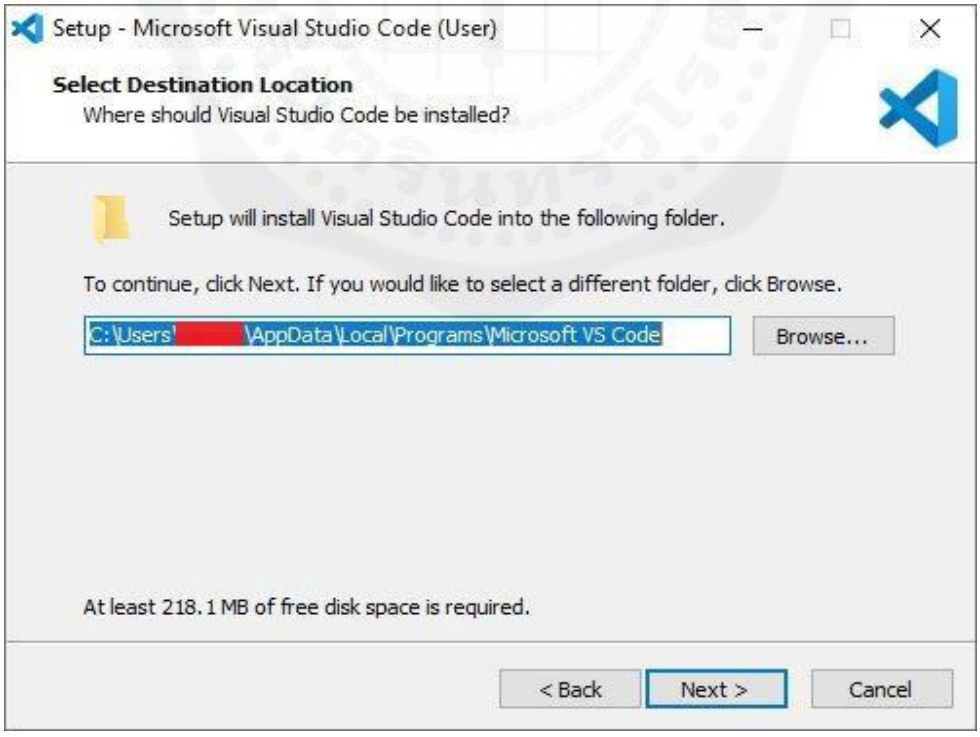
รูปที่ 35 : รูปภาพแสดง*หน้าต่างการ Run เพื่อลงโปรแกรม*

4. เลือก “I accept the agreement” และคลิกปุ่ม “Next >”



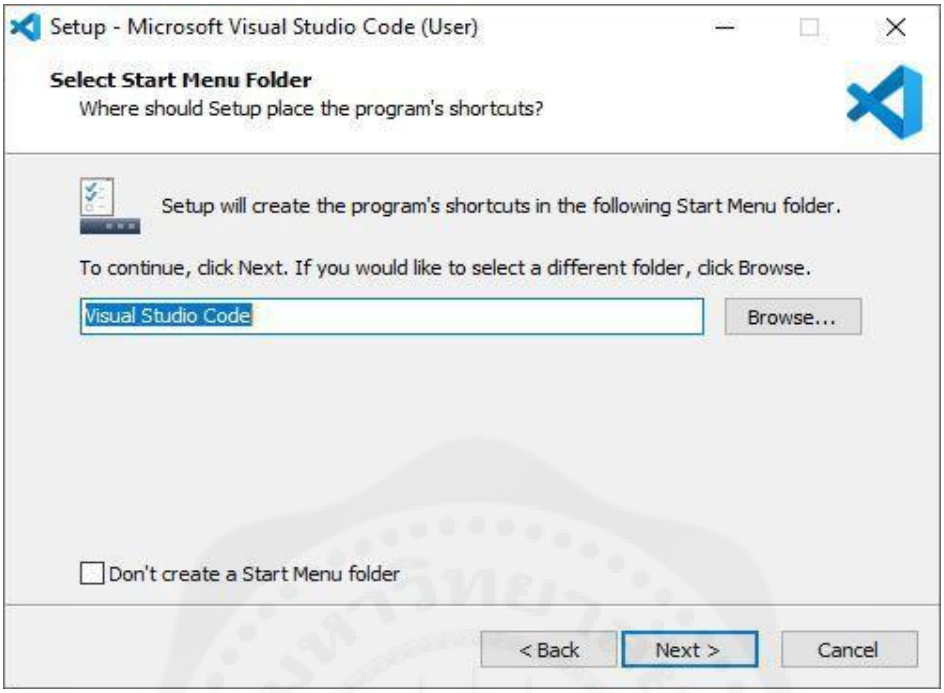
รูปที่ *36 :* รูปภาพแสดง*หน้าต่างแสดง License*

5. เลือกพื้นที่ในการจัดเก็บโปรแกรม (แนะนำให้ใช้ Default ที่ให้มา) และคลิกปุ่ม “Next >”



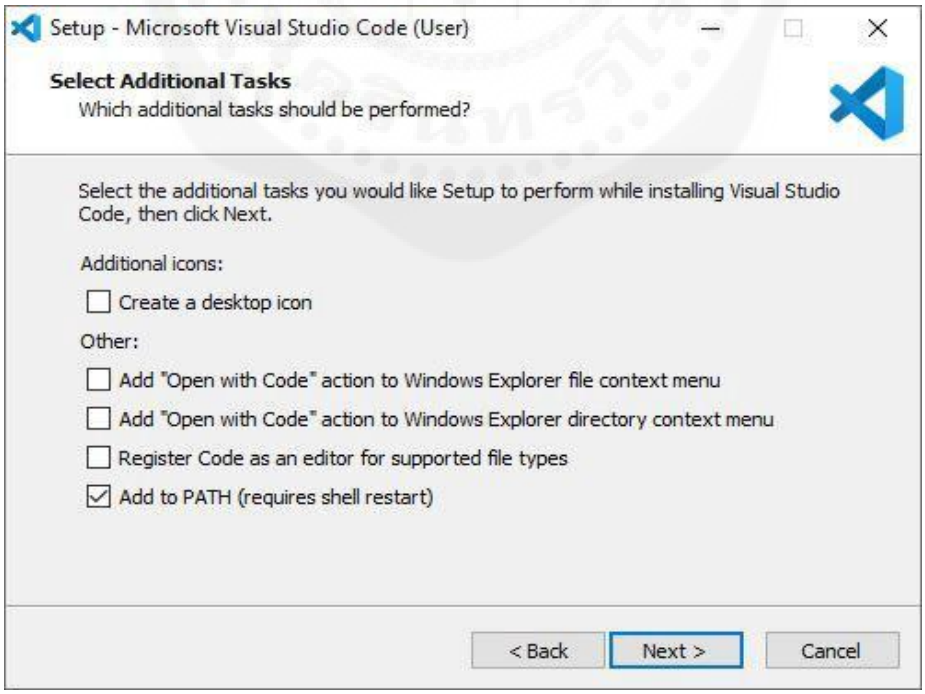
รูปที่ *37 :* รูปภาพแสดง*หน้าต่างเลือกจุดที่จะลงโปรแกรม*

6. คลิกปุ่ม “Next >”



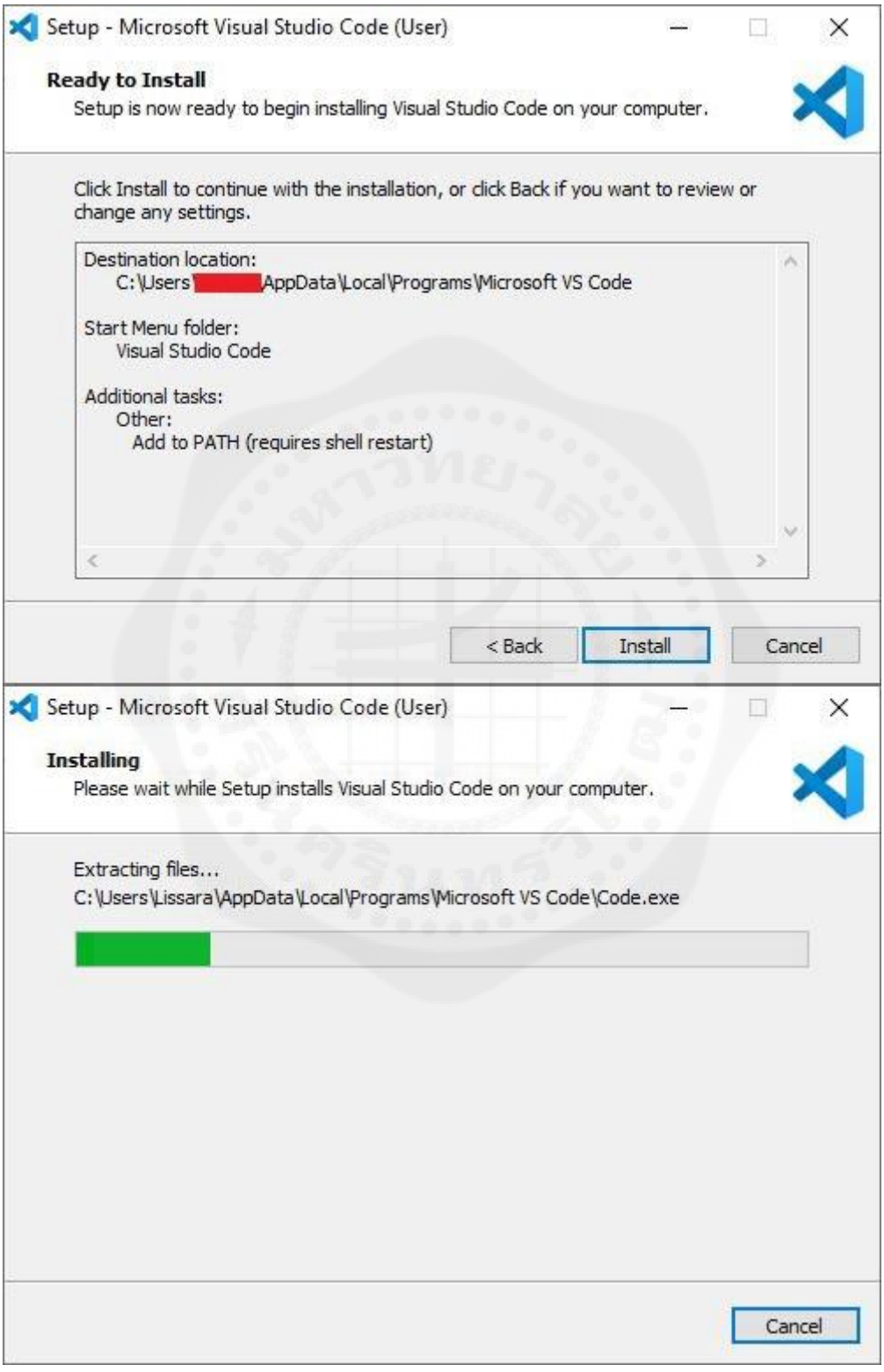
รูปที่ *38 :* รูปภาพแสดง*หน้าต่างแสดงโฟลเดอร์เริ่มต้น*

7. เลือกส่วนเพิ่มงานให้เลือกCreate a desktop icon และ Add to PATH (requires shell restart) จากนั้นให้ คลิกปุ่ม“Next >”



รูปที่ *39 :* รูปภาพแสดง*หน้าต่างเลือก Path Environment*

8. คลิกปุ่ม “Install” เพื่อติดตั้งโปรแกรม

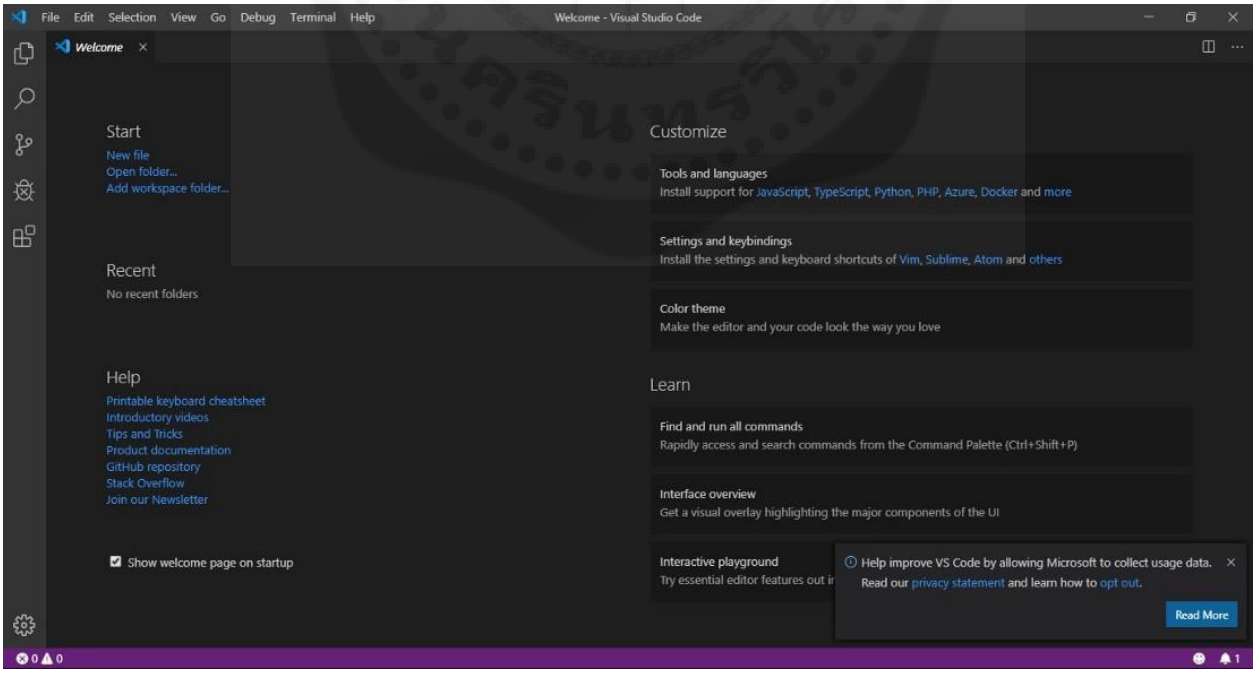


รูปที่ *40 :* รูปภาพแสดง*หน้าต่างแสดงขณะกำลังติดตั้งโปรแกรม*

9. คลิกปุ่ม “Finish” เสร็จสิ้นการติดตั้งโปรแกรม VSCode

รูปที่ *41 :* รูปภาพแสดง*หน้าต่างแสดงเมื่อติดตั้งโปรแกรมเสร็จสิ้น*

10. หน้าตาของโปรแกรมเมื่อติดตั้งเรียบร้อยแล้ว

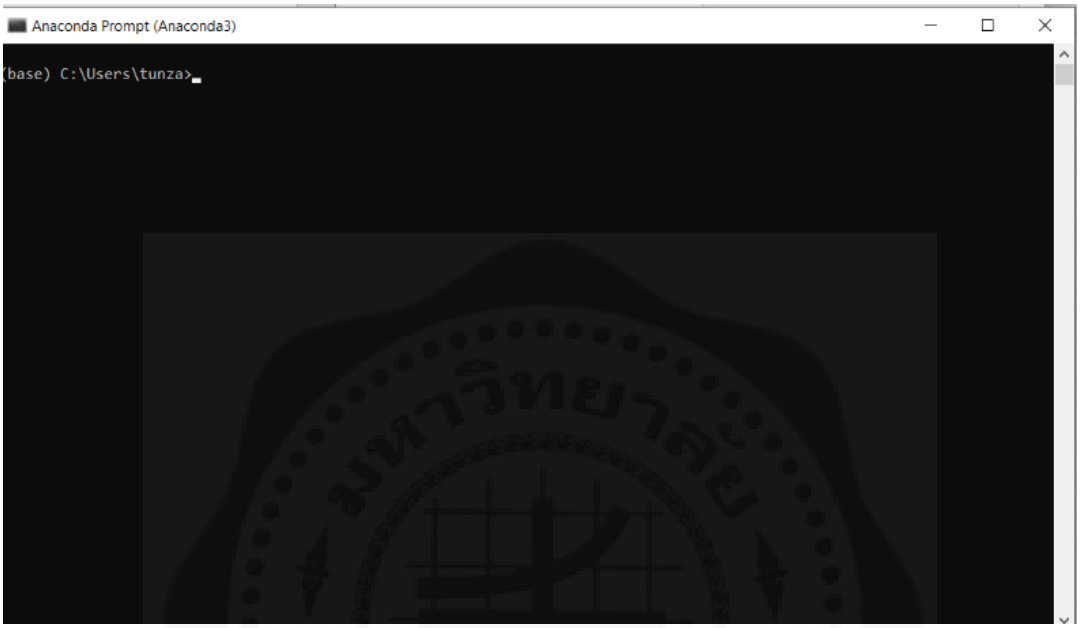


รูปที่ *42 :* รูปภาพแสดง*หน้าต่างของโปรแกรม Visual Studio Code*

ภาคผนวก ค

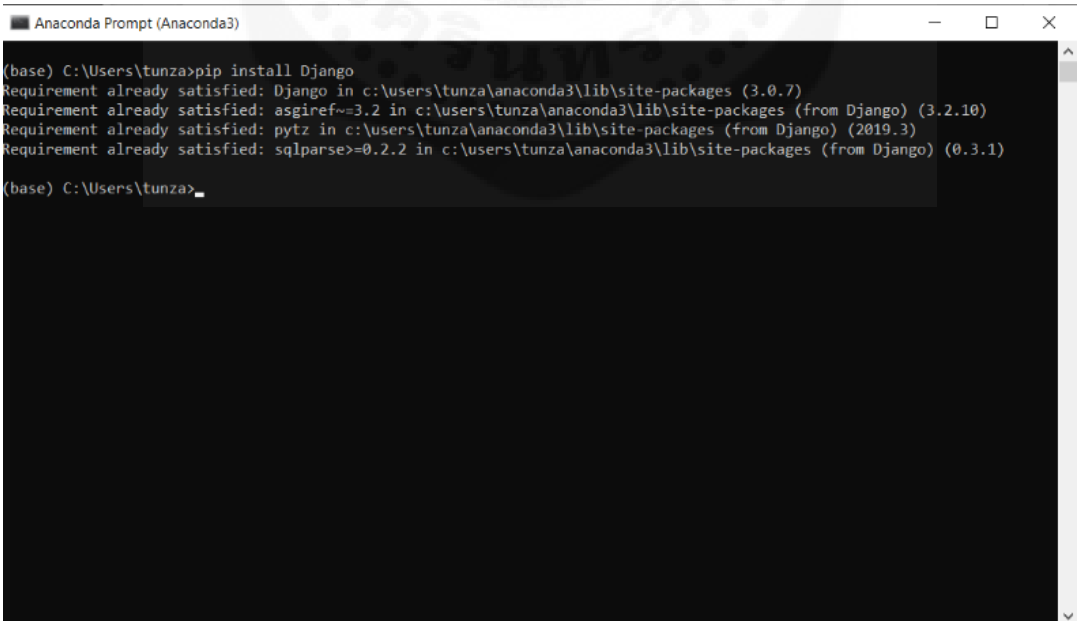
การติดตั้ง Django

1. เปิด Andconda Prompt ขึ้นมา



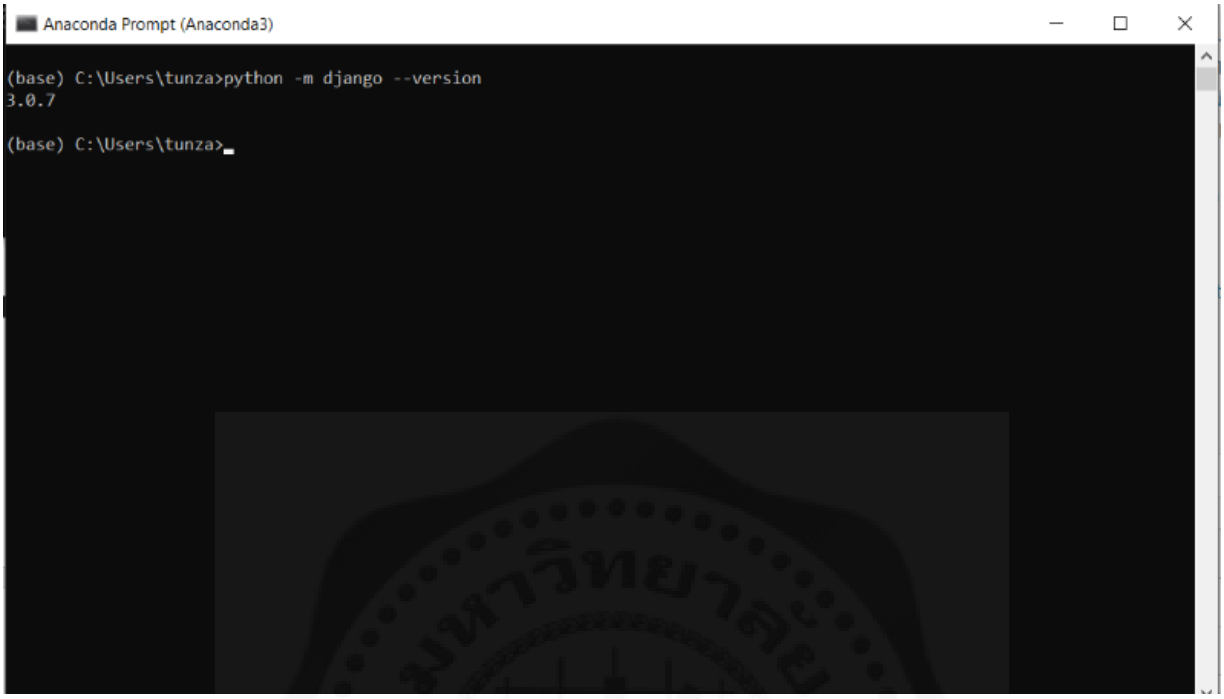
รูปที่ *43 :* รูปภาพแสดง*หน้าต่างของโปรแกรม Andconda Prompt*

2. ใช้คำสั่ง pip install django เพื่อติดตั้ง Django



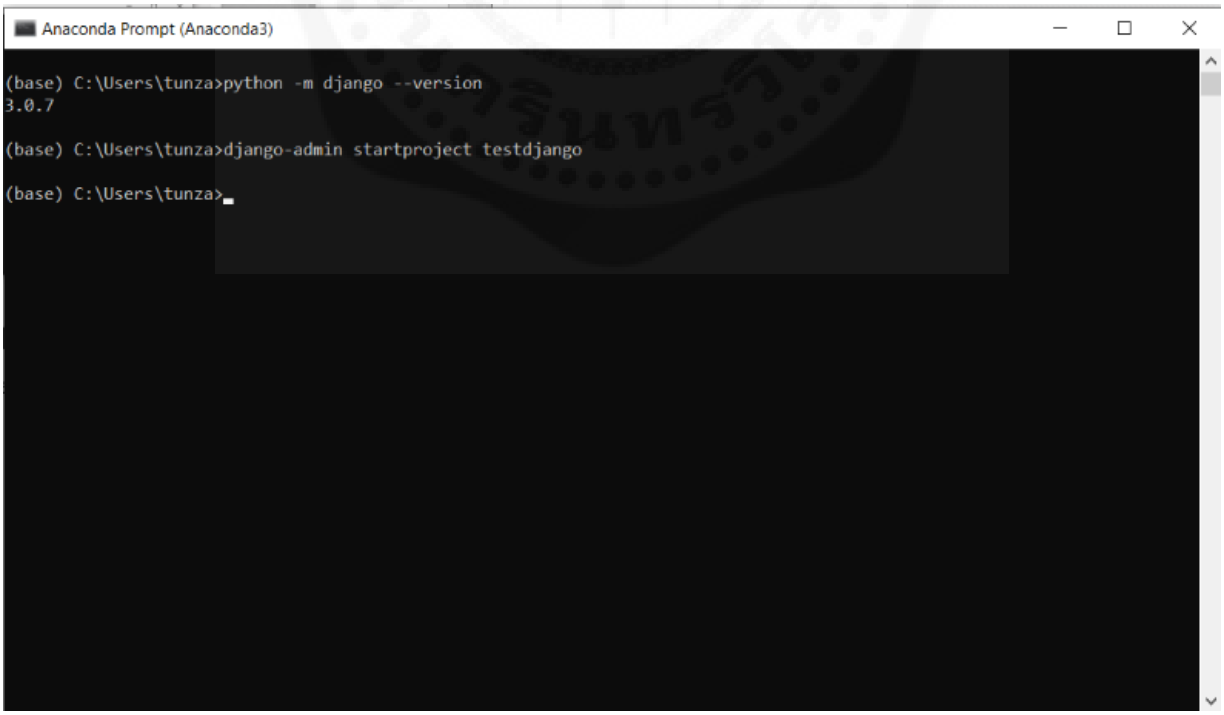
รูปที่ *44 :* รูปภาพแสดง*หน้าต่าง*การติดตั้ง *Django*

3. ทดสอบเวอร์ชั่นที่ติดตั้งด้วยคำสั่ง python -m django –version



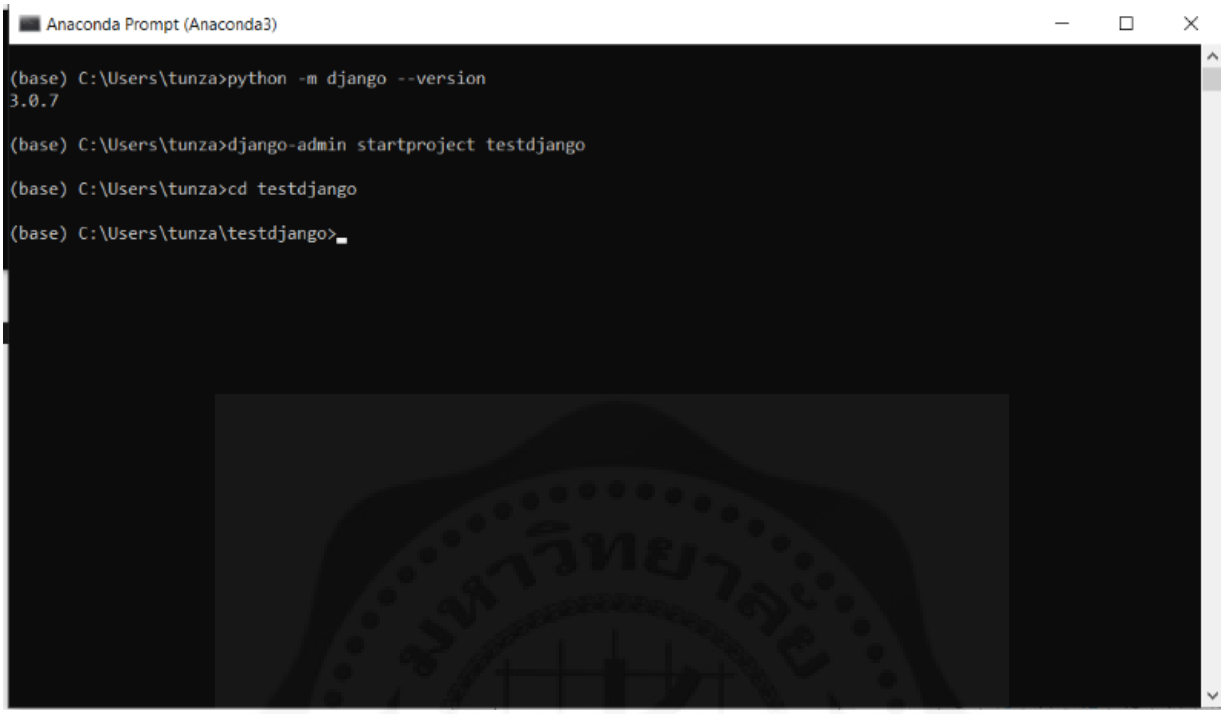
รูปที่ *45* **:** รูปภาพแสดง*หน้าต่าง*เรียกดูเวอร์ชั่นที่ติดตั้ง

4. สร้าง Project ใหม่เพื่อทดสอบด้วยคำสั่ง django-admin startproject [name]



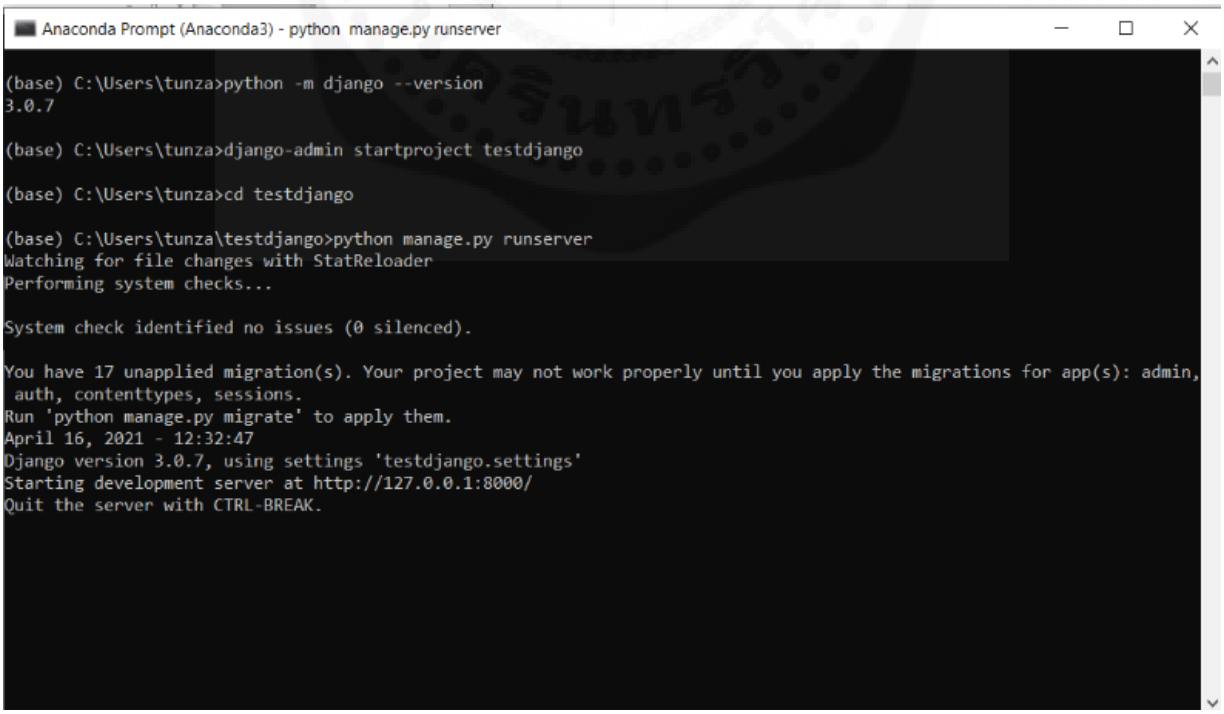
รูปที่ *46* **:** รูปภาพแสดงการสร้าง *Project* ใหม่

5. เข้าไปใน Project [name] Directory ด้วยคำสั่ง cd [name]



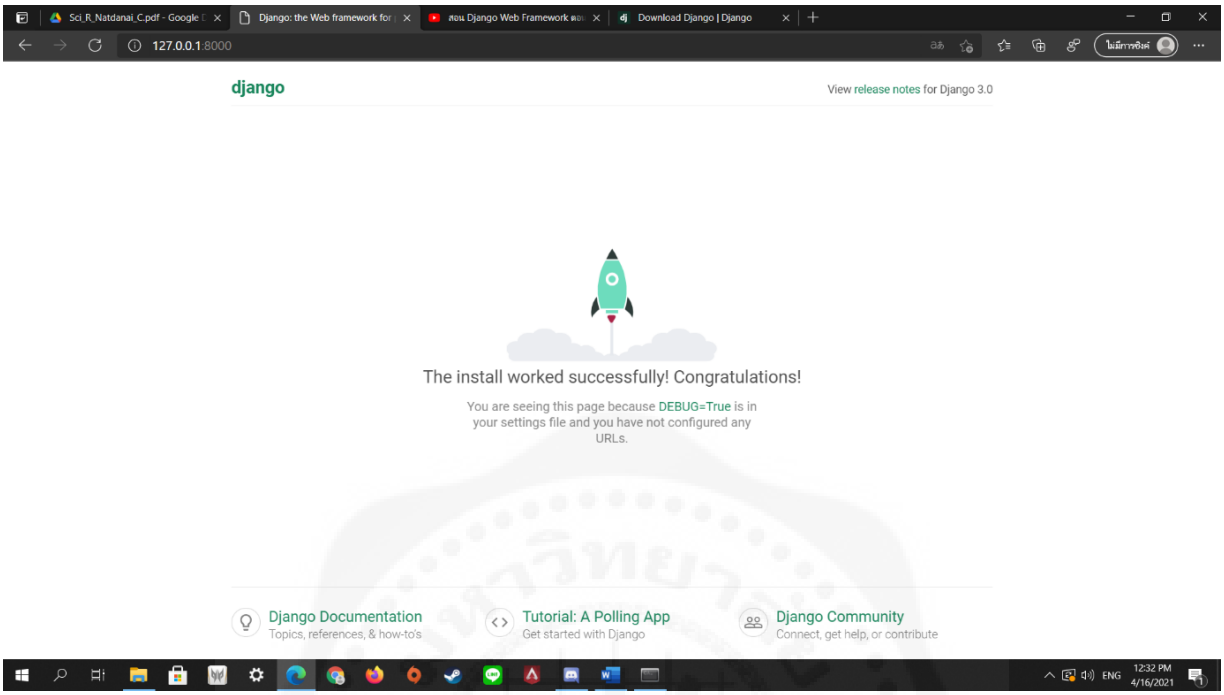
รูปที่ *47 :* รูปภาพแสดงการเข้าถึง *Directory*

6. ทดสอบการทำงานเซิร์ฟเวอร์ด้วยคำสั่ง python manage.py runserver



รูปที่ *48 :* รูปภาพแสดงการทดสอบการทำงานของเซิร์ฟเวอร์

7. เปิด Website เพื่อทดสอบโดยใช้ URL http://127.0.0.1:8000/

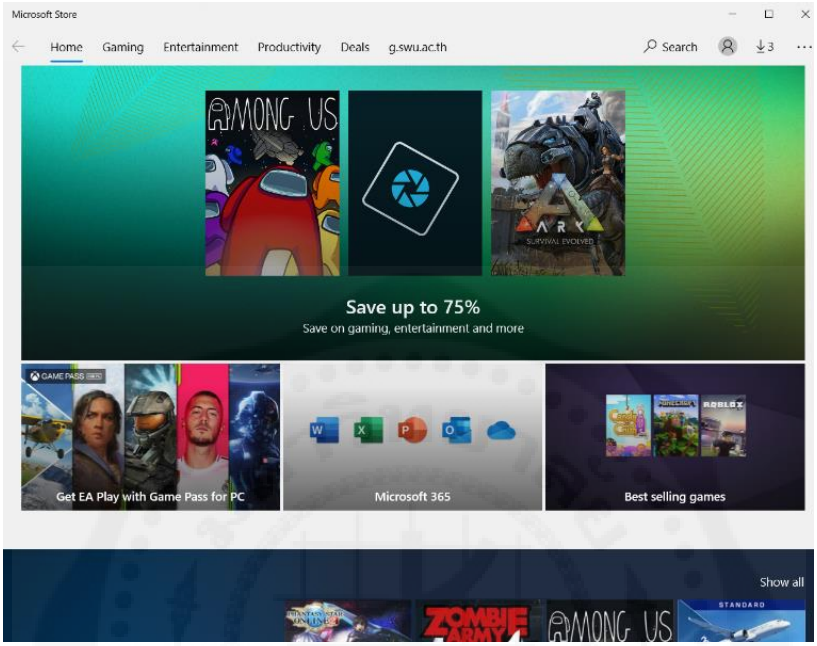


รูปที่ *49* : รูปภาพแสดงทดสดสอบการใช้งานของ *Django*

ภาคผนวก ง

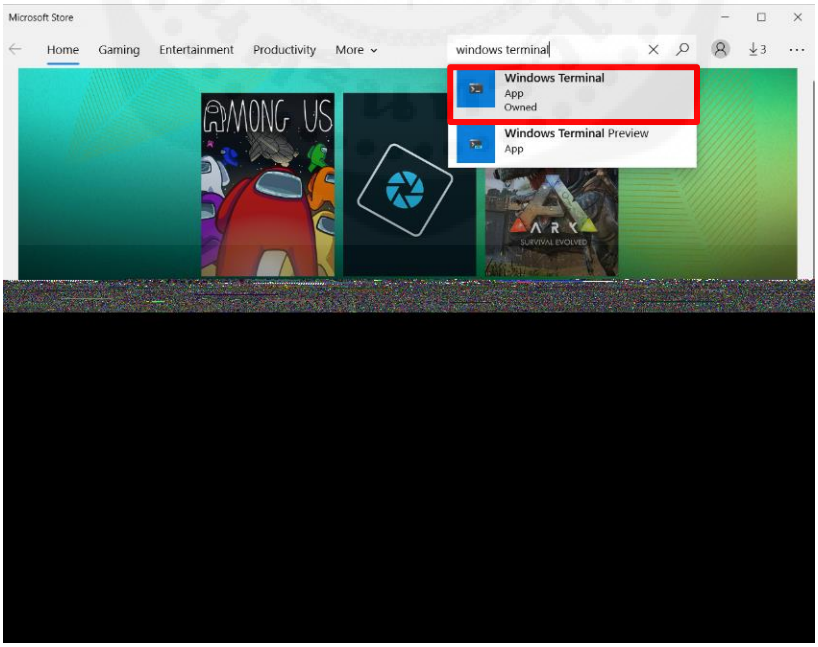
การติดตั้ง Terminal และ Ubuntu

1. เข้า Microsoft Store



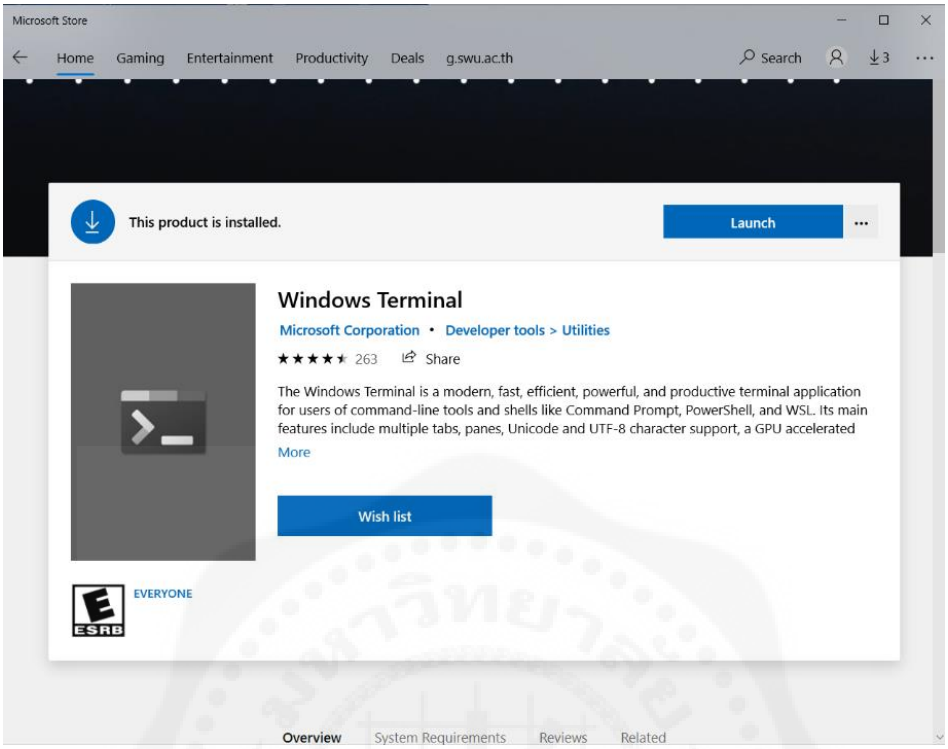
รูปที่ *50 :* รูปภาพแสดงหน้าต่าง *Microsoft Store*

2. ค้นหา Windows Terminal



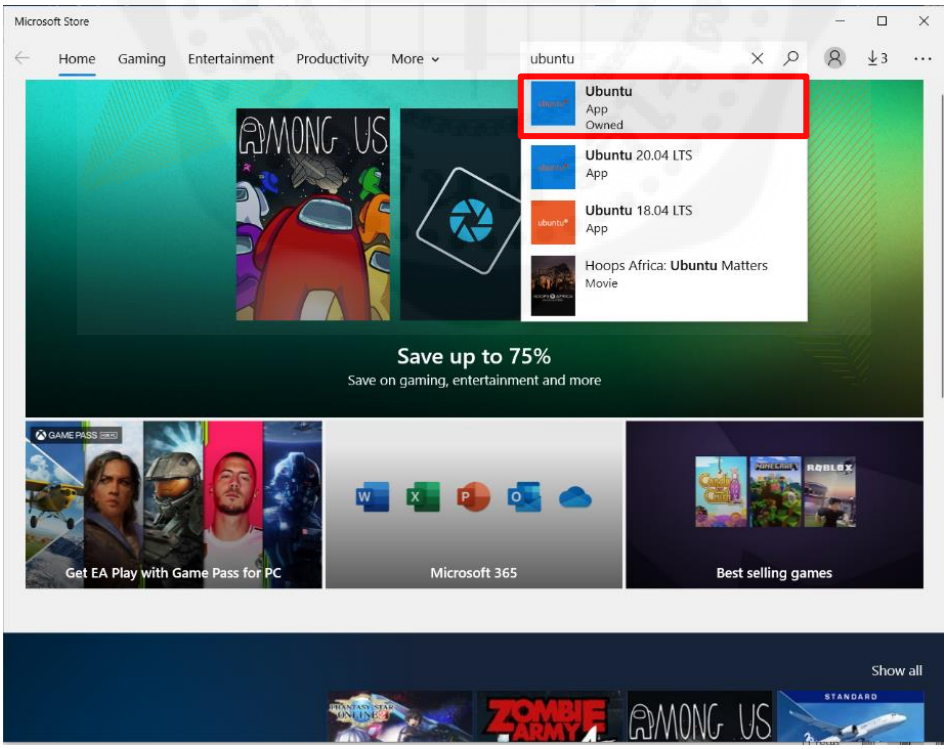
รูปที่ *51 :* รูปภาพแสดงการค้นหา *Windows Terminal*

3. กด Install Windows Terminal เพื่อติดตั้ง



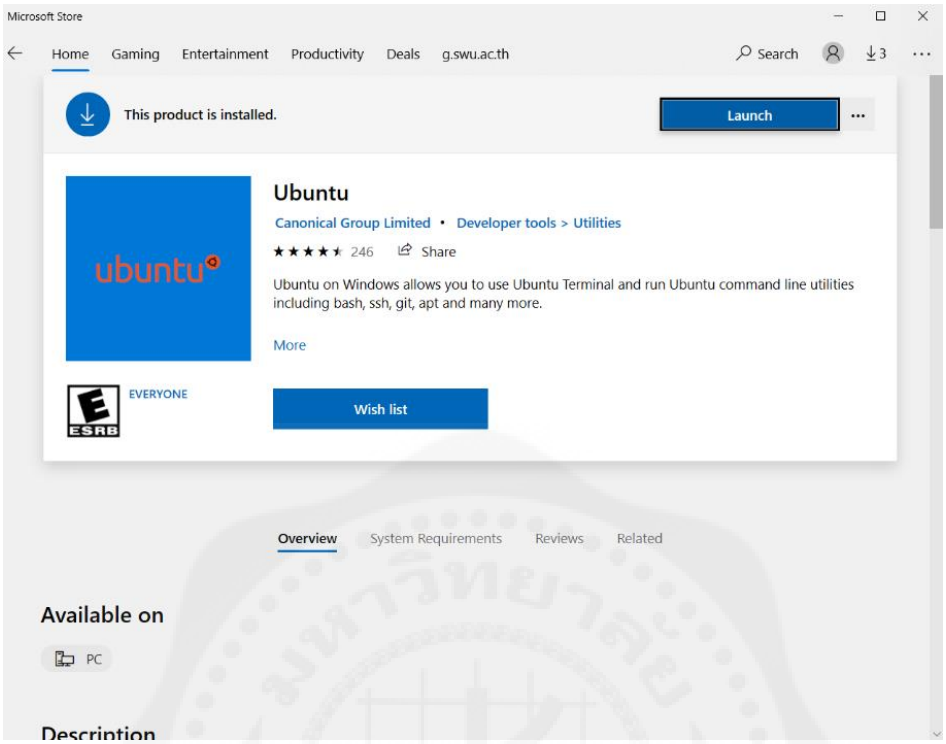
รูปที่ *52 :* รูปภาพแสดงการติดตั้ง *Windows Terminal*

4. กลับไปยังหน้าค้นหาเพื่อค้นหา Ubuntu



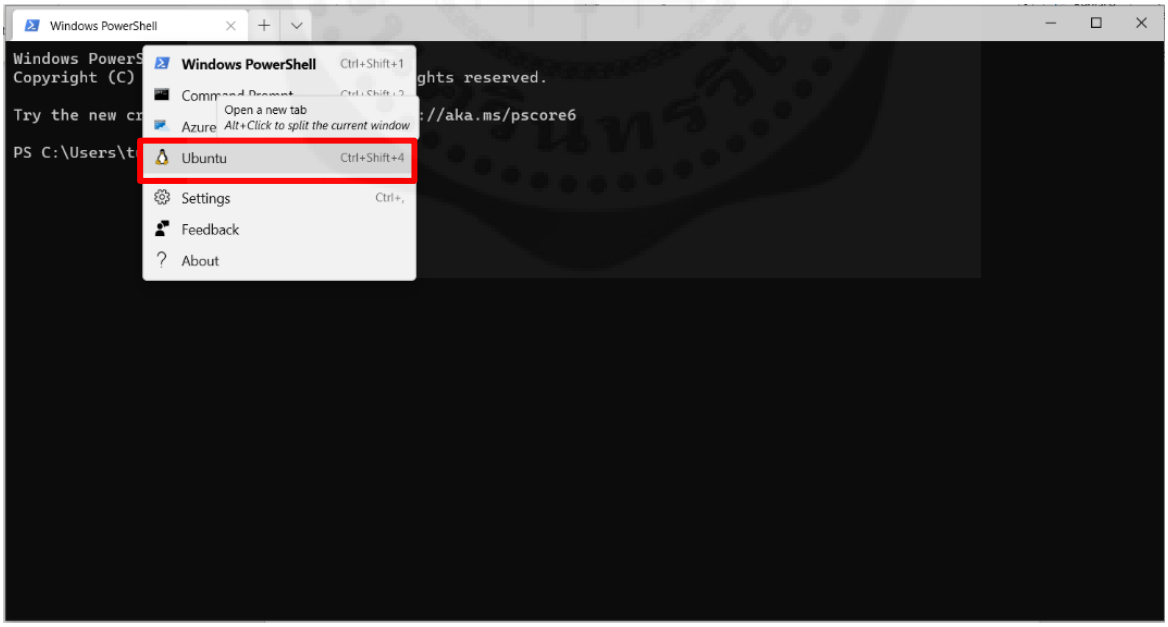
รูปที่ *53 :* รูปภาพแสดงการค้นหา *Ubuntu*

5. กด Install Ubuntu เพื่อติดตั้ง Ubuntu



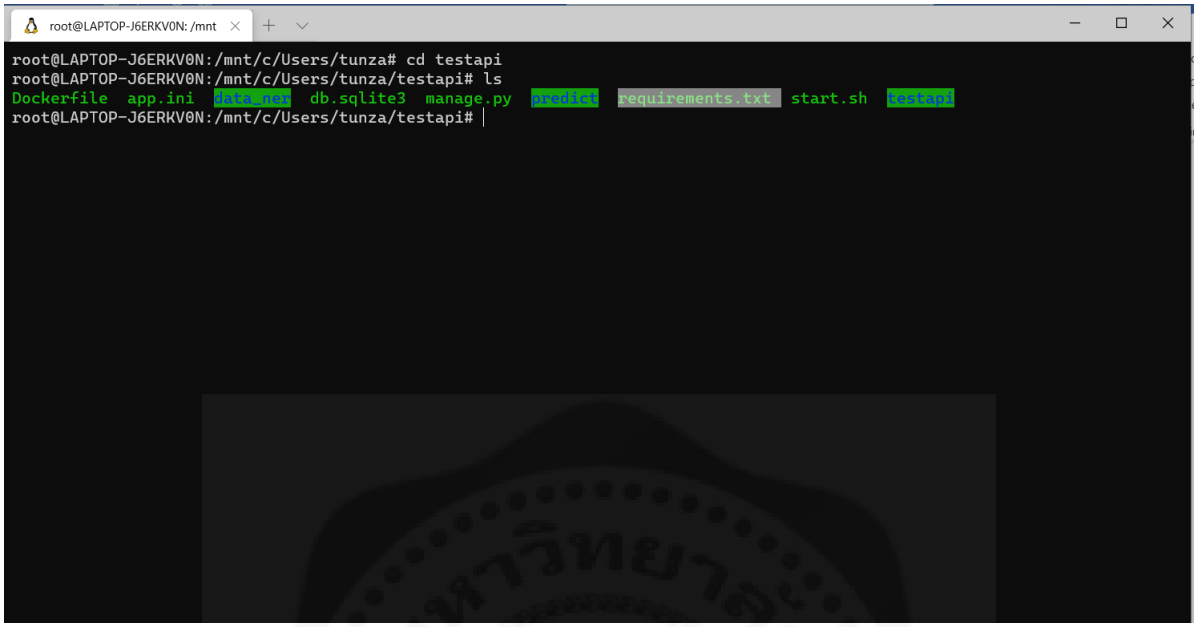
รูปที่ *54* : รูปภาพแสดงการติดตั้ง *Ubuntu*

6. ทำการลงไลบรารี่ต่าง ๆ ใน Ubuntu โดยเปิด Terminal และเข้าไปที่ Ubuntu

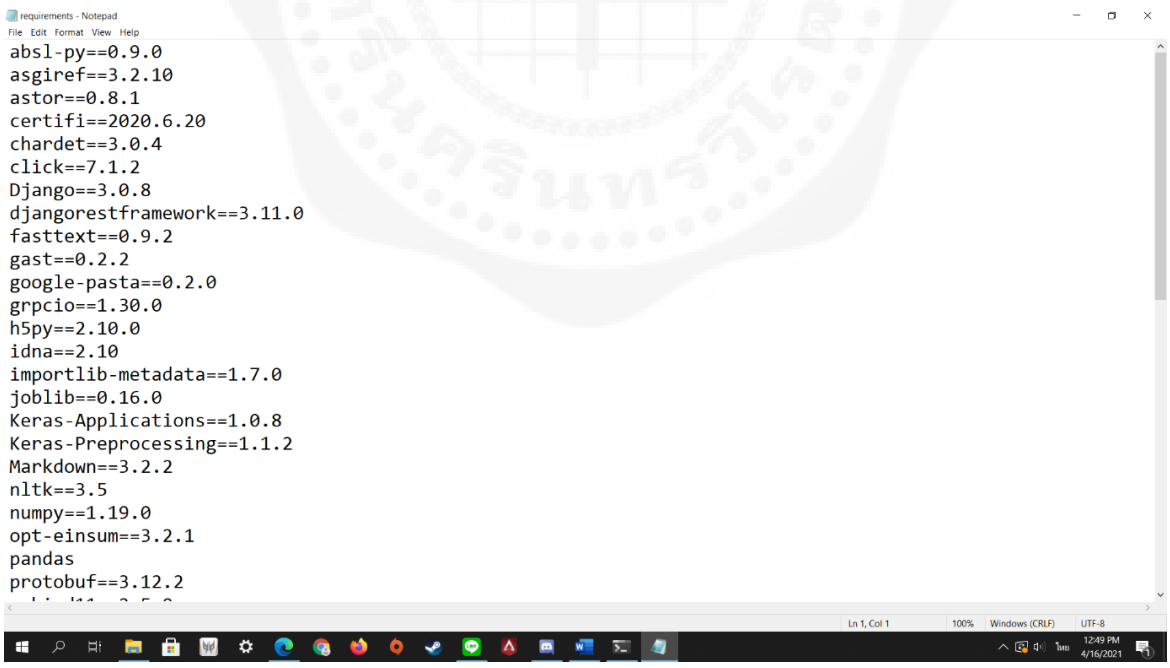


รูปที่ *55* : รูปภาพแสดงการเข้า *Ubuntu*

7. เข้าไปที่ Directory ที่มีไฟล์ Requirements.txt ที่เก็บข้อมูลไลบรารี่ไว้

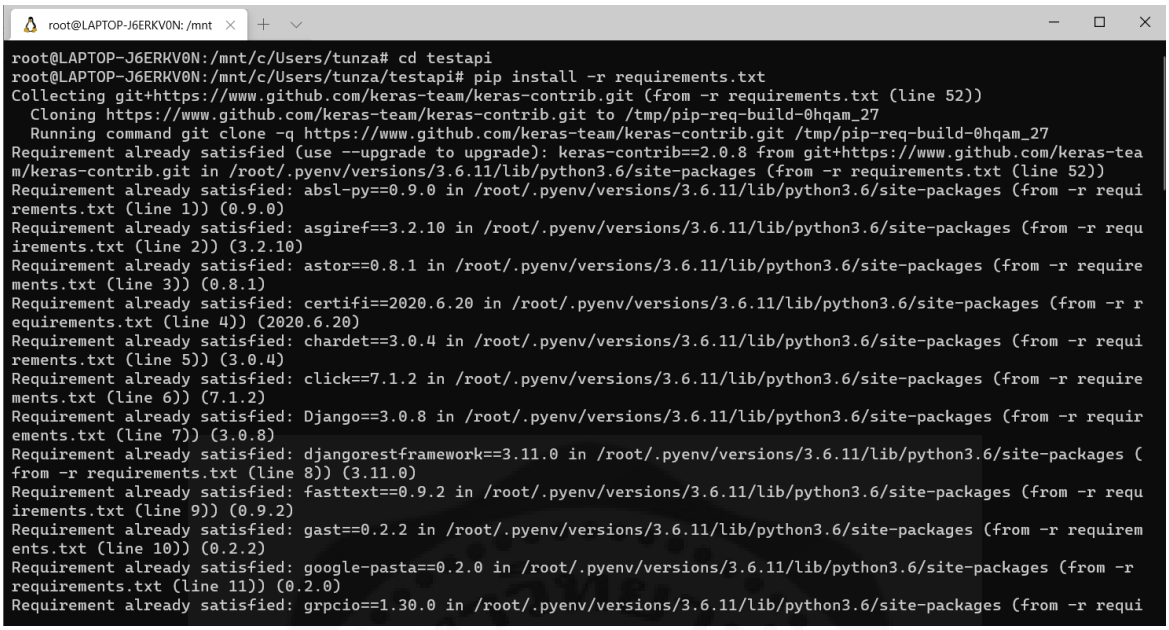


รูปที่ *56* : รูปภาพแสดงการเข้าถึง *Directory* ที่มีไฟล์ *Requirements.txt*



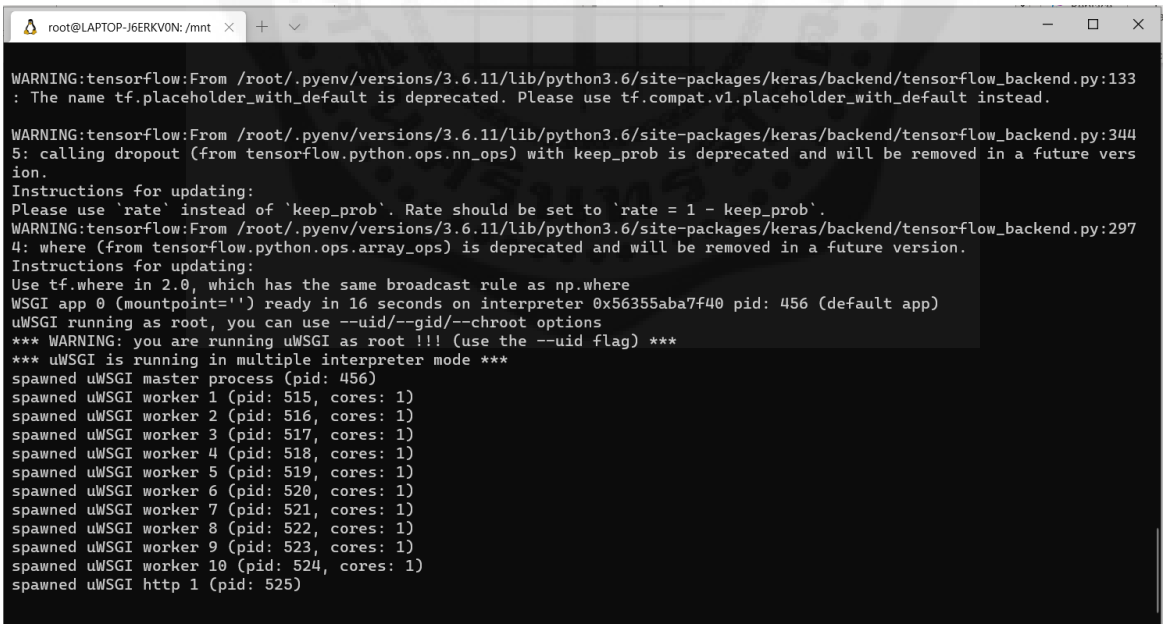
รูปที่ *57* : รูปภาพแสดงรายละเอียดไลบรารี่ของไฟล์ *Requirements.txt*

8. ทำการลงไลบรารี่ด้วยคำสั่ง pip install -r requirements.txt



รูปที่ *58 :* รูปภาพแสดงการลงไลบรารี่

9. ทดสอบการทำงานเซิร์ฟเวอร์รูปแบบ wsgi ด้วยคำสั่ง uwsgi --thunder-lock --ini app.ini



รูปที่ *59 :* รูปภาพแสดงการทดสอบการทำงานเซิร์ฟเวอร์รูปแบบ *wsgi*

ภาคผนวก จ

โค้ดของโปรแกรม

ชื่อไฟล์ views.py

ภาษาที่ใช้ python

```
from os import listdir
from os.path import isfile, join
from pythainlp.tag import pos_tag
from predict import apps
from django.apps import apps
from django.shortcuts import render
from collections import OrderedDict
from django.http import HttpResponse
from sklearn_crfsuite import metrics
from sklearn.neighbors import NearestNeighbors
from django.views.decorators.csrf import csrf_exempt
from sklearn.model_selection import train_test_split
from predict.thainlplib import ThaiWordSegmentLabeller
from sklearn.metrics import precision_score, recall_score, f1_score
from pythainlp.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Model, Input
from keras.layers import LSTM, Embedding, Dense, TimeDistributed, Dropout, Bidirectional
from keras_contrib.layers import CRF
from keras_contrib.utils import save_load_utils as sam

import copy
import editdistance
import random as ran
import multiprocessing as muti
```

```
import codecs
import glob
import nltk
import sys
import re
import os
import time
import timeit
import random
import joblib
import fasttext
import numpy as np
import pandas as pd
import tensorflow as tf
import sklearn_crfsuite
# Create your views here.

ner_model = joblib.load('predict/saved_model/ner_model.sav')
words_np= joblib.load('predict/add_tag_correct_word/bagOfWords_List_G2021.bin')
knn= joblib.load('predict/add_tag_correct_word/nbrsG2021.bin')
fastmodel= fasttext.load_model('predict/add_tag_correct_word/fastmodelG2021.bin')
pzdp = pd.read_csv('predict/add_tag_correct_word/autocompModify_df.csv')


idx2tag = {0: 'dzp', 1: 'ALLEY', 2: 'ROAD', 3: 'H_NAME', 4: 'PAD'}
keyChange = ('ALLEY', 'ROAD', 'DIST', 'ZONE', 'PROVINCE')
keyForAuto = ('DIST','ZONE' ,'PROVINCE' ,'POSTCODE')
prefixWord = {'MOO': ["ม."], 'ALLEY': ["ซ."], 'N-SUB_ALLEY': ["แยก"], 'ROAD': ["ถ."], 'DIST': ["ต.","
แขวง"], 'ZONE': ["อ.","เขต"], 'PROVINCE': ["จ."]}
BATCH_SIZE = 900  # Number of examples used in each iteration
EPOCHS = 10  # Number of passes through entire dataset
MAX_LEN = 20  # Max length of review (in words)
EMBEDDING = 40  # Dimension of word embedding vector
```

```
n_words = len(words_np)
n_tags = len(idx2tag)

saved_model_path='predict/saved_model'
iterator_get_next_1 = 'IteratorGetNext:1'
iterator_get_next_0 = 'IteratorGetNext:0'
placeholder_1 = 'Placeholder_1:0'
boolean_mask = 'boolean_mask_1/GatherV2:0'

model_check_bilstm = True
model_check = True
if (len(sys.argv) >= 2 and sys.argv[1] == 'runserver'):
    session = tf.Session()
    model = tf.saved_model.loader.load(session, [tf.saved_model.tag_constants.SERVING], sav
ed_model_path)
    graph = tf.get_default_graph()
    g_inputs = graph.get_tensor_by_name(iterator_get_next_1)
    g_lengths = graph.get_tensor_by_name(iterator_get_next_0)
    g_training = graph.get_tensor_by_name(placeholder_1)
    g_outputs = graph.get_tensor_by_name(boolean_mask)
    runmodel_uswgi = False
    print("Runserver Complete Create Model")
else:
    runmodel_uswgi = True

def createBiLstmCRF():
    input = Input(shape=(MAX_LEN,))
    model_bilstm = Embedding(input_dim=n_words+1, output_dim=EMBEDDING, # n_words
+ 1 (PAD)
                             input_length=MAX_LEN, mask_zero=True)(input)  # default: 20-
dim embedding
    model_bilstm = Bidirectional(LSTM(units=50, return_sequences=True,
```

```
                    recurrent_dropout=0.2))(model_bilstm)  # variational biLSTM
    model_bilstm = TimeDistributed(Dense(50, activation="relu"))(model_bilstm)  # a dense l
ayer as suggested by neuralNer
    model_bilstm= Dropout(0.2)(model_bilstm)
    crf = CRF(n_tags)  # CRF layer, n_tags+1(PAD)
    out = crf(model_bilstm)  # output
    model_bilstm = Model(input, out)

    return model_bilstm

model_bilstm = createBiLstmCRF()

@csrf_exempt
def Cleantext(request):

    def Change_Word(text):

        # Pretrained model weights location
        text = text.replace('เทศบาลตำบล', 'ทต.')
        text = text.replace('องค์การบริหารส่วนตำบล', 'อบต.')
        text = text.replace('องค์การบริหารส่วนจังหวัด', 'อบจ.')
        text = text.replace('สำนักงานสาธารณสุขอำเภอ', 'สนงอ')
        text = text.replace('สำนักงานเกษตรอำเภอ', 'สนงกอ')
        text = text.replace('หมู่บ้าน', 'มบ.')

        text = text.replace('รพสต', 'รพ.สต.')
        text = text.replace('รพสต.', 'รพ.สต.')
        text = text.replace('กทม.', 'จ.กรุงเทพ')
        text = text.replace('กทม', 'จ.กรุงเทพ')
        text = text.replace('กรุงเทพมหานคร', 'จ.กรุงเทพ')
        text = text.replace('เขต.', 'อ.')
        text = text.replace('อำเภอ.', 'อ.')
```

```
text = text.replace('ตำบล.', 'ต.')
text = text.replace('แขวง.', 'ต.')
text = text.replace('หมู่ที่', 'ม.')
text = text.replace('หมู่', 'ม.')
text = text.replace('เขต', 'อ.')
text = text.replace('อำเภอ', 'อ.')
text = text.replace('ตำบล', 'ต.')
text = text.replace('แขวง', 'ต.')
count_road = 0
while True:
    result = text.find('ถนน',count_road)
    if result == -1 :break
    count_road = result + 3
    sequence = 'ถ.'
    indicies = (result+1,result+3)
    if text[result-1] == " ":text=sequence.join([text[:indicies[0]-1], text[indicies[1]:]])
count_substreet = 0
while True:
    result = text.find('ซอย',count_substreet)
    if result == -1 :break
    count_substreet = result + 3
    sequence = 'ซ.'
    indicies = (result+1,result+3)
    if text[result-1] == " ":text=sequence.join([text[:indicies[0]-1], text[indicies[1]:]])

text = text.replace('มบ.', 'หมู่บ้าน')
text = text.replace('สพป.', 'สำนักงานคณะกรรมการเขตพื้นที่การศึกษา')
text = text.replace('สพม.', 'สำนักงานคณะกรรมการเขตพื้นที่การศึกษา')
text = text.replace('สพฐ.', 'สำนักงานคณะกรรมการศึกษาขั้นพื้นฐาน')
text = text.replace('สกอ.', 'สำนักงานคณะกรรมการการอุดมศึกษา')
text = text.replace('สกศ.', 'สำนักงานเลขาธิการสภาการศึกษา')
text = text.replace('ทต.', 'เทศบาลตำบล')
```

```
    text = text.replace('อบต.','องค์การบริหารส่วนตำบล')
    text = text.replace('อบจ.','องค์การบริหารส่วนจังหวัด')
    text = text.replace('สนงอ','สำนักงานสาธารณสุขอำเภอ')
    text = text.replace('สนงกอ','สำนักงานเกษตรอำเภอ')
    text = text.replace('รร.','โรงเรียน')
    text = text.replace('รพ.สต.','โรงพยาบาลส่งเสริมสุขภาพตำบล')
    text = text.replace('รพ.','โรงพยาบาล')
    text = text.replace('ร.พ.','โรงพยาบาล')

    find_right = text.find('(')
    find_left = text.find(')')
    text = text.replace(text[find_right:find_left+1], '')

    return text

def doc2features(doc, i):
    word_ner = doc[i][0]
    postag_ner = doc[i][1]
    # Features from current word
    features_ner={
        'word.word': word_ner,
        'word.isspace':word_ner.isspace(),
        'postag':postag_ner,
        'word.isdigit()': word_ner.isdigit()
    }
    if i > 0:
        prevword_ner = doc[i-1][0]
        postag1_ner = doc[i-1][1]
        features_ner['word.prevword'] = prevword_ner
        features_ner['word.previsspace']=prevword_ner.isspace()
        features_ner['word.prepostag'] = postag1_ner
        features_ner['word.prevwordisdigit'] = prevword_ner.isdigit()
```

```
    else:
        features_ner['BOS'] = True # Special "Beginning of Sequence" tag
    # Features from next word
    if i < len(doc)-1:
        nextword_ner = doc[i+1][0]
        postag1_ner = doc[i+1][1]
        features_ner['word.nextword'] = nextword_ner
        features_ner['word.nextisspace']=nextword_ner.isspace()
        features_ner['word.nextpostag'] = postag1_ner
        features_ner['word.nextwordisdigit'] = nextword_ner.isdigit()
    else:
        features_ner['EOS'] = True # Special "End of Sequence" tag
    return features_ner

def extract_features(doc):
    return [doc2features(doc, i) for i in range(len(doc))]

def get_addr(text):
    word_cut=word_tokenize(text,engine="newmm")
    list_word=pos_tag(word_cut,engine='perceptron')
    X_test = extract_features([(data,list_word[i][1]) for i,data in enumerate(word_cut)])
    y_ = ner_model.predict_single(X_test)
    a = [[word_cut[i],data] for i,data in enumerate(y_)]
    rr = np.array(a)
    I = (rr[:,1]=='I-ADDRESS')
    B = (rr[:,1]=='B-ADDRESS')
    A = B | I
    addr = rr[A]
    res = str.join('',(addr[:,0])).replace('<slash>','/')
    return res.replace('<space>',' ').replace('<full_stop>','.')

def nonzero(a):
```

```
    return [i for i, e in enumerate(a) if e != 0]


def split(s, indices):
    return [s[i:j] for i,j in zip(indices, indices[1:]+[None])]


def sepNum(t,r=''): # separated number and text, be used in AddressDictAndList()
    text=''
    num=[]
    if len(t)>0:
        t =re.sub(r, "", t).strip()
        if len(t)==0: return text, num # stop error
        wcut =re.findall("[\d\s]+|[^\d]+", t)
        for i in wcut:
            if re.search("\d", i):
                n =i.split()
                num.extend(n)
            else:
                text +=i
        if re.search("\d",wcut[0]):
            num.append(num.pop(0))


    return text, num


def AddressDictAndList(t):
    adlist=[]
    addict = {
    'H_ID' : "",
    'MOO' : "",
    'H_NAME' : "",
    'ALLEY' : "",
    'N-ALLEY' : "",
    'N-SUB_ALLEY' : "",
```

```
        'ROAD' : "",
        'N-ROAD' : "",
        'DIST' : "",
        'ZONE' : "",
        'PROVINCE' : "",
        'POSTCODE' : ""
        }
        t=  t.split("|")
        t.pop(0); t.pop() # if address text start and end with |
        for text in t: # no handle with H_NAME
            text =text.strip()
            if  re.search("^[หมู่ที่\.\s]+\d?\d$", text) and not re.search("^[
ห\.\s]+\d$", text) : # for unexpected error such as ห7, ห.2
                addict['MOO'] = sepNum(text)[1] # store only number
            elif  re.search( "ซ\.|ซอย", text) or re.search( "แยก\s*\d", text):
                if re.search( "แยก\s*\d", text):
                    sp = re.search( "แยก\s*\d",text).span()[0] # at index split
                    addict['N-SUB_ALLEY'] = sepNum(text[sp:])[1]
                    text = text[:sp]
                addict['ALLEY'], addict['N-ALLEY'] = sepNum(text, r="ซ\.|ซอย")
            elif  re.search( "^ถ\.|^ถนน", text):
                addict['ROAD'], addict['N-ROAD'] = sepNum(text, r="^ถ\.|^ถนน")
            elif  re.search( "^แขวง|^ตำบล|^ต\.", text):
                addict['DIST'] = sepNum(text, r="^แขวง|^ตำบล|^
ต\.")[0] #have only text, no number
            elif  re.search( "^เขต|^อำเภอ|^อ\.", text):
                addict['ZONE'] = sepNum(text, r="^เขต|^อำเภอ|^
อ\.")[0] #have only text, no number
            elif  re.search( "^จังหวัด|^จ\.|^กรุงเทพ|^ก\.*ท\.*ม\.*$", text):
                if re.search( "กรุงเทพ|^ก\.*ท\.*ม\.*$", text):
                    addict['PROVINCE'] = "กรุงเทพมหานคร"
                else:
```

```
                addict['PROVINCE'] = sepNum(text, r="^จังหวัด|^
จ\.")[0] #have only text, no number
            elif len(text) ==5 and text.isdigit():
                addict['POSTCODE'] = text
            elif re.search("^\d.+\d$|^\d+",text) and len(text) <10:
                addict['H_ID'] = text
            else:
                t,n = sepNum(text) # separated text and number
                adlist.append((t,n))

        return addict, adlist


    # cut subtext from word
    def indexCut (c, text):
        c = c.replace("(", "\(")
        c = c.replace(")", "\)")
        c = c.replace(".", "\.")
        if re.search(c, text):
            return text[re.search(c, text).span()[1]-1:]
        else:
            return ''


    # calculated similar word score
    def simWord(word1, word2):
            word1_old = word1
            sc=0
            total = 1 if len(word1)==1 else len(word1)-1
            for j in range(len(word2)-1):
                subtext = word2[j]+word2[j+1]
                if subtext in word1:
                    sc +=1
```

```
                word1 = indexCut(subtext, word1) # cut text that already match
                if word1 ==": break

            return  sc/(total+abs(len(word1_old)-len(word2)))

    def MaxMatchWord(word, clist): # select the most word similarly in list
        edw=[] # compare edistance
        for i in clist:
            edw.append(editdistance.eval(word,i)) # compare edistance


            index= edw.index(min(edw))
            bestWord= clist[index]
            acc= simWord(word, bestWord)
            if acc<0.2: # not match
                return -1, ''
            elif acc>0.5: # match
                return index, bestWord
            else: # no suggest to change word
                return index, ''

    def embAndCorWords(words_list):
        if type(words_list) == dict:
            words_input= [words_list[i] for i in keyChange]
        else:
            words_input= [i[0] for i in words_list]

        if len(words_input) ==0: return [], [] # stop error
        words_embspace = list(map(lambda v: re.sub(" ", '$', v), words_input))
        word_input_vec = [fastmodel.get_sentence_vector(' '.join(list(word))) for word in words
_embspace]
```

```
        indices = knn.kneighbors(word_input_vec, 5, False)  # n_neighbors is 5 (number of corr
ect word rank)
        suggestion = words_np[indices]


        lstm_input= []
        changeWordList=[]
        for i in range(len(words_input)):    # get index from bag of words
            w= words_input[i]
            index, newWord = MaxMatchWord(w, suggestion[i])
            if type(words_list) == dict:
                changeWordList.append(newWord) # keep new word for update in dict
            elif index == -1 and type(words_list) == list:
                dropWord= [x for x in words_list if w in x][0]
                words_list.remove(dropWord) #drop uless word
            else:
                lstm_input.append(indices[i][index]) # index n_words for uless word
                changeWordList.append(newWord)



        return lstm_input, changeWordList


    def createQueryText (dzplist, drop=-1): # new update 12/3/2564-21/3/2564
        txt = ''
        if drop== -1:
            for i in dzplist:
                sq = f"pzdp.isin(['{i}']).any(axis=1) & "
                txt +=sq
        else:
            for i in range(len(dzplist)):
                if i != drop:
                    sq = f"pzdp.isin([{dzplist[i]}]).any(axis=1) & " if  dzplist[i].isdigit() else  f"pzdp.isin([
'{dzplist[i]}']).any(axis=1) & "
```

```
            txt +=sq
        if txt == '': return []
        fullTextq = 'np.array(pzdp[' + txt[:-2] + '])'
        return eval(fullTextq) #  result maybe [[]]


    def autocomp(dzplist, dictAddr): # use with autocomp dataframe
        loop = True
        dropIndex= -1 # drop 1 index if no result in query
        result=[]
        while loop:
            result = createQueryText(dzplist, dropIndex)
            if len(result)==0:
                dropIndex+=1
            if len(result)>0 or dropIndex== len(dzplist):
                loop= False

        if len(result)>0: # have result to add in output dict
                qlist = [list(map(lambda x,y: (x,y),  a, keyForAuto)) for a in result]
                interResult = list(set(qlist[0]).intersection(*qlist)) # find intersect matching
                for v,k in interResult:
                    dictAddr[k] = v


    def updateDict(dict_addr, list_addr, changeWordDict, changeWordList, tags, get_dzp=Fals
e): # new update 21/3/2564
        changeWordList = list(zip(list_addr,changeWordList))
        for i in range(len(changeWordDict)):
            if len(changeWordDict[i])>0:
                dict_addr[keyChange[i]] = changeWordDict[i]

        baseDict={}
        for k,v in dict_addr.items(): baseDict[k]= True if v=='' else False
```

```
        dzpList=[]
        for i in range(len(tags)):
            k= tags[i]
            word = changeWordList[i][1] if len(changeWordList[i][1]) >0 and k != 'H_NAME' else ch
angeWordList[i][0][0] # no correct H_NAME
            word = word+' '.join(changeWordList[i][0][1]) if len(changeWordList[i][0][1])>0 else wo
rd
            if k not in ['dzp']:
                if baseDict[k]:
                    dict_addr[k] = word
                    if 'N-'+k in dict_addr:
                        dict_addr['N-'+k]= changeWordList[i][0][1]

            if get_dzp and k == 'dzp' and len(word)>0:
                dzpList.append(word)

        if get_dzp:
            if len(dzpList) >0:
                if len(dict_addr['POSTCODE']) >0 : dzpList.append(dict_addr['POSTCODE'])
            else:
                for i in keyForAuto:
                    if len(dict_addr[i]) >0: dzpList.append(dict_addr[i])
            return dzpList


    def printOutputText(dictAddr):
        ans=''
        bkk = -1 if dictAddr['PROVINCE'] == 'กรุงเทพมหานคร' else 0 # choose prefix
        for i in dictAddr:
            if dictAddr[i] == '': continue
            ans+= '' if i[0] =='N' else ' '
            if i in prefixWord and dictAddr[i] != 'กรุงเทพมหานคร' : ans += prefixWord[i][bkk]
            if type(dictAddr[i]) is list:
```

```
                num2text= ''
                for n in dictAddr[i]:
                    num2text+= str(n) if num2text=='' else ' '+str(n)
                ans+= num2text
            else:
                ans+= str(dictAddr[i])

        ans= ans.replace('$', ' ')
        return ans.strip()



    # ถ้าrun แบบ uwsgi จะเข้า if ตรงนี้
    if runmodel_uswgi :
        global model_check
        global session
        global model
        global graph
        global g_inputs
        global g_lengths
        global g_training
        global g_outputs
        global model_bilstm
        if model_check == True:
            model_check = False
            session = tf.Session()
            model = tf.saved_model.loader.load(session, [tf.saved_model.tag_constants.SERVIN
G], saved_model_path)
            graph = tf.get_default_graph()
            g_inputs = graph.get_tensor_by_name(iterator_get_next_1)
            g_lengths = graph.get_tensor_by_name(iterator_get_next_0)
            g_training = graph.get_tensor_by_name(placeholder_1)
            g_outputs = graph.get_tensor_by_name(boolean_mask)
```

```
            sam.load_all_weights(model_bilstm, 'predict/add_tag_correct_word/Bi-
LSTM_CRF_ModelV2_33provG2021.bin')
    else:
        global model_check_bilstm
        if model_check_bilstm == True:
            model_check_bilstm = False
            sam.load_all_weights(model_bilstm, 'predict/add_tag_correct_word/Bi-
LSTM_CRF_ModelV2_33provG2021.bin')

    if request.path == "/UpdateStatus":
        return HttpResponse("Success")
    else:
        #กำหนดpatternของตัวอักษร
        regex = re.compile(pattern="[ก-๙]")
        phone_check = re.compile(pattern = r'\d\d-?\d?-?\d\d\d-?\d\d\d\d?')
        id_store = []
        count_id = 0
        id_seg = ""
        text_seg = ""

        text = request.POST.get('input')

        phone_index = re.search(phone_check, text)
        if phone_index is not None:
            phone_index = phone_index.start()
            text = text[:phone_index] + 'โทร ' + text[phone_index:]
            print(text)
        phone = phone_check.findall(text)
        for i in phone: text = text.replace(i, "")

        tag_number = 4
        tag_end = 1
```

```
alley_seq = "ห้อง"
result = text.find('ห้อง')
if result != -1:
    while True:
        if text[result+tag_number] == " ":
            while True:
                if text[result+tag_number+tag_end].isnumeric():
                    alley_seq = " "
                    tag_end += 1
                else:
                    break
            break
        else:
            while True:
                print(text)
                print(result+tag_number+tag_end)
                if text[result+tag_number+tag_end].isnumeric():
                    alley_seq = " "
                    tag_end += 1
                else:
                    break
            break
        tag_number += 1
    if tag_number == 4:
        indicies = (result,result+tag_number+tag_end)
    else:
        indicies = (result,result+tag_number)
    if alley_seq != "ห้อง":
        if text[result-1] == " ":text=alley_seq.join([text[:indicies[0]-1], text[indicies[1]:]])


# test
# clean_text = text
```

```
        #ner
        text = re.sub(' +',' ',text)
        text = get_addr(text)
        text = text.replace('<ampersand>', '&')
        text = text.replace('<minus>', '-')
        text = text.replace('<left_parenthesis>', '(')
        text = text.replace('<right_parenthesis>', ')')
        text = text.replace('<equal>', '=')
        text = text.replace('<comma>', ',')
        text = text.replace('<at_mark>', '@')
        text = text.replace('<colon>', ':')

        print(text)
        clean_text = text
        text = Change_Word(text)
        # implement choose model condition from text
        try:
            if ("อ." in text and "ต." in text and "จ."in text) or ("
อ." in text and "ต." in text and "กรุงเทพ"in text) or ("
อ." in text and "ต." in text and "กรุงเทพมหานคร"in text) or ("
อ." in text and "ต." in text and "กรุงเทพฯ"in text):

                phone = phone_check.findall(text)
                for i in phone: text = text.replace(i, "")
                text = text.replace("()", "")
                text = text.replace("", '')
                text = re.sub(' +',' ',text)
                text_no = text.split(' ')

                for i in text_no:
                    weights = regex.findall(i)
```

```
            if not weights:
                count_id += 1
            else:
                break


        if count_id > 1:
            for i in range(count_id):
                id_store.append(text_no[i])
            for i in range(count_id):
                text_no.pop(0)
            for i in range(len( id_store)):
                id_seg += "|" + id_store.pop(0)
            for i in text_no:
                text_seg += i
            # Convert text to labels
            text = text_seg.replace(" ", "")
            inputs = [ThaiWordSegmentLabeller.get_input_labels(text)]
            lengths = [len(text)]
            #run predict word segmentation
            predict_word_cut = session.run(g_outputs, feed_dict = {g_inputs: inputs, g_len
gths: lengths, g_training: False})
            # Mark word boundaries with pipe character
            result = ""
            if predict_word_cut[0] == 0: predict_word_cut[0] = 1
            for w in split(text, nonzero(predict_word_cut)):  result +='|'+w
            # for w in split(text, nonzero(predict_word_cut)):  result += w + " "
            result = result.replace(" ", "")
            # if (result[-1] != "|"): result += '|'
            result = id_seg + result
        else:
            # Convert text to labels
            text = text.replace(" ", "")
```

```
                inputs = [ThaiWordSegmentLabeller.get_input_labels(text)]
                lengths = [len(text)]
                #run predict word segmentation
                predict_word_cut = session.run(g_outputs, feed_dict = {g_inputs: inputs, g_len
gths: lengths, g_training: False})
                # Mark word boundaries with pipe character
                result = ""
                if predict_word_cut[0] == 0: predict_word_cut[0] = 1
                for w in split(text, nonzero(predict_word_cut)):  result +='|'+w
                # for w in split(text, nonzero(predict_word_cut)):  result += w + " "
                result = result.replace(" ", "")
                # result += '|'
            print(predict_word_cut)
        else:
            text = re.sub(' +',' ',text)
            result = text.find('ม.')
            if result != -1:
                sequence = None
                if text[result+2] == " " :
                    if text[result+3] != " ":
                        if text[result+4] == " ":
                            sequence = "ม." + text[result+3]
                        else:
                            sequence = "ม." + text[result+3:result+4]
                indicies = (result+1,result+4)
                if sequence is not None:
                    if text[result-1] == " ":text=sequence.join([text[:indicies[0]-
1], text[indicies[1]:]])
                tag_number = 2
                tag_end = 1
                alley_seq = ""
                result = text.find('ซ.')
```

```
            if result != -1:
                while True:
                    if text[result+tag_number] == " ":
                        while True:
                            if text[result+tag_number+tag_end] != " ":
                                alley_seq = alley_seq + text[result+tag_number+tag_end]
                                tag_end += 1
                            else:
                                break
                        break
                    tag_number += 1
                indicies = (result+tag_number+1,result+tag_number+tag_end+1)
                if alley_seq != "":
                    if text[result-1] == " ":text=alley_seq.join([text[:indicies[0]-
1], text[indicies[1]:]])
                result = text
                result = result.replace(' ', '|')
                result = "|" + result
                result = result.replace('||', '|')

            # result = result + "|"
            print(" Word Segmentation result : "+result)
            clean_text = result
            dict_addr, list_addr =AddressDictAndList(result)

            _, changeWordDict =embAndCorWords(dict_addr)
            lstm_input, changeWordList =embAndCorWords(list_addr)
            lstm_input = pad_sequences(maxlen=MAX_LEN, sequences=[lstm_input], padding="
post", value=n_words) # Padding each sentence to have the same lenght

            sqtag= model_bilstm.predict(lstm_input)[0].tolist()
            outputTags= [idx2tag[i.index(1)] for i in sqtag]
```

```
            outputTags= outputTags[: len(list_addr)]

            dzpList =updateDict(dict_addr, list_addr, changeWordDict, changeWordList, outputTa
gs, get_dzp=True)

            autocomp(dzpList, dict_addr)

            clean_text = printOutputText(dict_addr)

            print(" No error " + clean_text)
        except:
            result = re.sub(' +',' ',text)
            # result = result.replace(' ', '|')
            # result = "|" + result + "|"
            # result = result.replace('||', '|')
            # # print(" error : " + sys.exc_info()[0])
            clean_text = "error"

        return HttpResponse(clean_text)
```

ชื่อไฟล์ trainmodel.ipynb

ภาษาที่ใช้ python

```
%tensorflow_version 1.x
import pandas as pd
import tensorflow as tf
import numpy as np
import os
import time
from os import listdir
from os.path import isfile, join
import random
from collections import OrderedDict
from sklearn.metrics import precision_score, recall_score, f1_score
import re
import joblib
import fasttext
from pythainlp.tag import pos_tag
from sklearn.model_selection import train_test_split
import sklearn_crfsuite
from sklearn.neighbors import NearestNeighbors
from sklearn_crfsuite import metrics
from thainlplib import Predict
from thainlplib import ThaiWordSegmentLabeller, ThaiWordSegmentationModel
# Split tokens
def process_line(line):
    inputs = []
    outputs = []
    for token in line.split('|'):
        if len(token) == 0: continue
        inputs += ThaiWordSegmentLabeller.get_input_labels(token)
        outputs += ThaiWordSegmentLabeller.get_output_labels(token)
```

```
    return inputs, outputs

# Create a record for a sentence
def make_sequence_example(sequence, labels):
    token_features = [tf.train.Feature(int64_list=tf.train.Int64List(value=[x])) for x in sequence]
    label_features = [tf.train.Feature(int64_list=tf.train.Int64List(value=[x])) for x in labels]
    example = tf.train.SequenceExample(
        context = tf.train.Features(feature={
            'length': tf.train.Feature(int64_list=tf.train.Int64List(value=[len(sequence)]))
        }),
        feature_lists = tf.train.FeatureLists(feature_list={
            'tokens': tf.train.FeatureList(feature=token_features),
            'labels': tf.train.FeatureList(feature=label_features)
        })
    )
    return example

# Read input data line by line and split to training and validation TFRecord files
def preprocess_files(input_files, training_output_file, validation_output_file, training_proporti
on):
    options = tf.python_io.TFRecordOptions(compression_type=tf.python_io.TFRecordCompre
ssionType.ZLIB)
    training_writer = tf.python_io.TFRecordWriter(training_output_file, options=options)
    validation_writer = tf.python_io.TFRecordWriter(validation_output_file, options=options)
    file_number = 1
    for input_filename in input_files:
        print('Processing file {}/{}...'.format(file_number, len(input_files)))
        file_number += 1
        input_file = open(input_filename,encoding='utf-8')
        for line in input_file.readlines():
            x, y = process_line(line)
            p = random.random()
```

```
            example = make_sequence_example(x, y)
            if p <= training_proportion:
                training_writer.write(example.SerializeToString())
            else:
                validation_writer.write(example.SerializeToString())
        input_file.close()
    training_writer.close()
    validation_writer.close()

def flatten_list(l):
    return [item for sublist in l for item in sublist]

def list_files(paths):
    return flatten_list([[path + '/' + file for file in listdir(path) if isfile(join(path, file))] for path in
paths])

# Download and store the BEST corpus into data directory first
# Read and shuffle input files
input_files = list_files(['data'])
random.shuffle(input_files)

# input_files = open("data/test.txt",encoding='utf-8')

training_data_file = 'training.tf_record'
validation_data_file = 'validation.tf_record'

# Preprocess and split each sentence to training and validation files
preprocess_files(input_files, training_data_file, validation_data_file, .9)

from thainlplib import ThaiWordSegmentLabeller, ThaiWordSegmentationModel

# Training and validation data configuration
```

```
training_data_file = 'training.tf_record'
validation_data_file = 'validation.tf_record'
vocabulary_size = ThaiWordSegmentLabeller.get_input_vocabulary_size()
num_output_labels = ThaiWordSegmentLabeller.get_output_vocabulary_size()

# Model hyperparameters
dropout = 0.50
state_size = 128
learning_rate = 0.001

# Other configuration
buffer_size = 150000 # Read all data to CPU memory
batch_size = 112 # Lower/increase this depending on your GPU memory size
validate_every_n_iterations = 100
checkpoint_path = 'checkpoints'

model = ThaiWordSegmentationModel(training_data_file, validation_data_file, buffer_size, ba
tch_size,
                                  vocabulary_size, num_output_labels, state_size, dropout)
model.train(learning_rate, validate_every_n_iterations, checkpoint_path, restore_checkpoint=
False)

  # # Save Model
from thainlplib import ThaiWordSegmentLabeller, ThaiWordSegmentationModel
checkpoint_path = 'checkpoints'
saved_model_path = 'saved_model'
model.save_model(checkpoint_path,saved_model_path)


#test NER
import codecs
from pythainlp.tokenize import word_tokenize
from pythainlp.tag import pos_tag
```

```
from nltk.tokenize import RegexpTokenizer
import glob
import nltk
import re

def Unique(p):
 text=re.sub("<[^>]*>","",p)
 text=re.sub("\[(.*?)\]","",text)
 text=re.sub("\[\/(.*?)\]","",text)
 if text not in data_not:
  data_not.append(text)
  return True
 else:
  return False

def toolner_to_tag(text):
 text=text.strip()
 text=re.sub("<[^>]*>","",text)
 text=re.sub("(\[\/(.*?)\])","\\1***",text)#.replace('(\[(.*?)\])','***\\1')#  ตัดการกับพวกไม่มี tag word
 text=re.sub("(\[\w+\])","***\\1",text)
 text2=[]
 for i in text.split('***'):
  if re.search(pattern,i):
   text2.append(i)
  else:
   text2.append("[word]"+i+"[/word]")
 text="".join(text2)#re.sub("[word][/word]","","".join(text2))
 return text.replace("[word][/word]","")
# แปลง text ให้เป็น conll2002
def text2conll2002(text,pos=True):
    """
    ใช้แปลงข้อความให้กลายเป็น conll2002
```

```
    """
    text=toolner_to_tag(text) # นำไปใส่ tag [word]
    #print(text)
    text=text.replace("""","")
    text=text.replace("’","").replace("‘","")#.replace("'","")
    tag=tokenizer.tokenize(text) # แยก tag ออกมาจากข้อความ
    j=0
    conll2002="" # ประกาศตัวแปรเก็บ conll2002
    for tagopen,text,tagclose in tag: # ลูปใน tag โดยเป็น (tagopen,text,tagclose)
        word_cut=word_tokenize(text,engine="newmm") # ใช้ตัวตัดคำ newmm ของ PyThaiNLP
        i=0
        txt5=""
        while i<len(word_cut): #ลูปตามจำนวน token ที่ตัดในtag
            if word_cut[i]=="""" or word_cut[i]=="":pass
            elif i==0 and tagopen!='word': # ไม่เป็น tag [word] และเป็น i หรือตัวเริ่มต้น tag
                txt5+=word_cut[i]
                txt5+='\t'+'B-'+tagopen
            elif tagopen!='word':
                txt5+=word_cut[i]
                txt5+='\t'+'I-'+tagopen
            else: # เป็น [word]
                txt5+=word_cut[i]
                txt5+='\t'+'O'
            txt5+='\n'
            #j+=1
            i+=1
        conll2002+=txt5
    if pos==False:
        return conll2002
    return postag(conll2002) # เพิ่ม postag ใส่
# ใช้สำหรับกำกับ pos tag เพื่อใช้กับ NER
# print(text2conll2002(t,pos=False))
```

```
def postag(text):
    listtxt=[i for i in text.split('\n') if i!='']
    list_word=[]
    for data in listtxt:
        list_word.append(data.split('\t')[0])
    #print(text)
    list_word=pos_tag(list_word,engine="perceptron")
    text=""
    i=0
    for data in listtxt:
        text+=data.split('\t')[0]+'\t'+list_word[i][1]+'\t'+data.split('\t')[1]+'\n'
        i+=1
    return text
# อ่านข้อมูลจากไฟล์
def get_data(fileopen):
  """
    สำหรับใช้อ่านทั้งหมดทั้งในไฟล์ทีละรรทัดออกมาเป็น list
    """
  with codecs.open(fileopen, 'r',encoding='utf-8-sig') as f:
    lines = f.read().splitlines()
  return [a for a in lines if Unique(a)] # เอาไม่ซ้ำกัน

def alldata(lists):
    text=""
    for data in lists:
        text+=text2conll2002(data)
        text+='\n'
    return text

def alldata_list(lists):
    data_all=[]
    for data in lists:
```

```
        data_num=[]
        try:
            txt=text2conll2002(data,pos=True).split('\n') # นำไปแปลงเป็น conll2002
            #print(txt)
            for d in txt:
                tt=d.split('\t')
                if d!="":
                    if len(tt)==3:
                        data_num.append((tt[0],tt[1],tt[2]))
                    else:
                        data_num.append((tt[0],tt[1]))
            #print(data_num)
            data_all.append(data_num)
        except:
            print(data)
    #print(data_all)
    return data_all

def alldata_list_str(lists):
  string=""
  for data in lists:
    string1=""
    for j in data:
      string1+=j[0]+" "+j[1]+"  "+j[2]+"\n"
    string1+="\n"
    string+=string1
  return string

def get_data_tag(listd):
  list_all=[]
  c=[]
  for i in listd:
```

```
    if i !='':
      c.append((i.split("\t")[0],i.split("\t")[1],i.split("\t")[2]))
    else:
      list_all.append(c)
      c=[]
  return list_all
def getall(lista):
    ll=[]
    for i in lista:
        o=True
        for j in ll:
            if re.sub("\[(.*?)\]","",i)==re.sub("\[(.*?)\]","",j):
                o=False
                break
        if o==True:
            ll.append(i)
    return ll

from sklearn_crfsuite import scorers,metrics
from sklearn.metrics import make_scorer
from sklearn.model_selection import cross_validate,train_test_split
import sklearn_crfsuite
def doc2features(doc, i):
    word = doc[i][0]
    postag = doc[i][1]
    # Features from current word
    features={
        'word.word': word,
        'word.isspace':word.isspace(),
        'postag':postag,
        'word.isdigit()': word.isdigit()
    }
```

```
    if i > 0:
        prevword = doc[i-1][0]
        postag1 = doc[i-1][1]
        features['word.prevword'] = prevword
        features['word.previsspace']=prevword.isspace()
        features['word.prepostag'] = postag1
        features['word.prevwordisdigit'] = prevword.isdigit()
    else:
        features['BOS'] = True # Special "Beginning of Sequence" tag
    # Features from next word
    if i < len(doc)-1:
        nextword = doc[i+1][0]
        postag1 = doc[i+1][1]
        features['word.nextword'] = nextword
        features['word.nextisspace']=nextword.isspace()
        features['word.nextpostag'] = postag1
        features['word.nextwordisdigit'] = nextword.isdigit()
    else:
        features['EOS'] = True # Special "End of Sequence" tag
    return features

def extract_features(doc):
    return [doc2features(doc, i) for i in range(len(doc))]

def get_labels(doc):
    return [tag for (token,postag,tag) in doc]

def get_ner(text):
    word_cut=word_tokenize(text,engine="newmm")
    list_word=pos_tag(word_cut,engine='perceptron')
    X_test = extract_features([(data,list_word[i][1]) for i,data in enumerate(word_cut)])
    y_=crf.predict_single(X_test)
```

```
    return [(word_cut[i],list_word[i][1],data) for i,data in enumerate(y_)]

# Apache License 2.0
file_name= 'data_ner/addrutf.txt' # ชื่อไฟล์คลังข้อมูล

#จัดการประโยคซ้ำ
data_not=[]

# เตรียมตัวตัด tag ด้วย re
pattern = r'\[(.*?)\](.*?)\[\/(.*?)\]'
tokenizer = RegexpTokenizer(pattern) # ใช้ nltk.tokenize.RegexpTokenizer เพื่อ
ตัด [TIME]8.00[/TIME] ให้เป็น ('TIME','ไง','TIME')
# จัดการกับ tag ที่ไม่ได้ tag

data1=getall(get_data(file_name)) # นำคลังเข้าไป แยกออกเป็น list ละบรรทัด
datatofile=alldata_list(data1) # นำไปผ่านขั้นตอน 1 2 3 4



tt=[]
with open(file_name+"-pos.conll","w") as f:
    i=0
    while i<len(datatofile):
        for j in datatofile[i]:
            #print(j)
            try:
                f.write(j[0]+"\t"+j[1]+"\t"+j[2]+"\n")
                #tt.append(j)
            except:
                continue
        if i+1<len(datatofile):
            f.write("\n")
        i+=1
```

```
#dd=[]
with open(file_name+".conll","w") as f:
    i=0
    while i<len(datatofile):
        for j in datatofile[i]:
            try:
                f.write(j[0]+"\t"+j[2]+"\n")
                #dd.append(j)
            except:
                continue
        if i+1<len(tt):
            f.write("\n")
        i+=1

X_data = [extract_features(doc) for doc in datatofile] # เอา คำ แยกออกมา
y_data = [get_labels(doc) for doc in datatofile] # เอา tag แยกออกมา



X, X_test, y, y_test = train_test_split(X_data, y_data, test_size=0.1) # แบ่ง 0.1 หรือ 10%
crf = sklearn_crfsuite.CRF(
    algorithm='lbfgs',
    c1=0.1,
    c2=0.1,
    max_iterations=500,
    all_possible_transitions=True,
    model_filename=file_name+"-pos.model0", # ตั้งชื่อโมเดล
    verbose=False
)
crf.fit(X, y); # train

labels = list(crf.classes_)
labels.remove('O')
```

```
y_pred = crf.predict(X_test)
e=metrics.flat_f1_score(y_test, y_pred,
                        average='weighted', labels=labels)
print(e) # โชว์ประสิทธิภาพ
sorted_labels = sorted(
    labels,
    key=lambda name: (name[1:], name[0])
)
print(metrics.flat_classification_report(
    y_test, y_pred, labels=sorted_labels, digits=3
))


filename = 'ner_model.sav'
joblib.dump(crf, filename)
```

ชื่อไฟล์ model.py

ภาษาที่ใช้ python

```
import tensorflow as tf
import numpy as np
from sklearn.metrics import precision_score, recall_score, f1_score
import os

class ThaiWordSegmentationModel:
    # Parse each sentence of input and output labels and length of the sentence from
TFRecord file
    #4
    @staticmethod
    def _parse_record(example_proto):
        context_features = {
            "length": tf.FixedLenFeature([], dtype=tf.int64)
        }
        sequence_features = {
            "tokens": tf.FixedLenSequenceFeature([], dtype=tf.int64),
            "labels": tf.FixedLenSequenceFeature([], dtype=tf.int64)
        }

        context_parsed, sequence_parsed =
tf.parse_single_sequence_example(serialized=example_proto,
            context_features=context_features, sequence_features=sequence_features)

        return context_parsed['length'], sequence_parsed['tokens'], sequence_parsed['labels']

    # Read training data from TFRecord file, shuffle, loop over data infinitely and
    # pad to the longest sentence
    #2
    @staticmethod
```

```
def _read_training_dataset(data_file, batch_size, buffer_size=10000):
    return tf.data.TFRecordDataset([data_file], compression_type="ZLIB") \
        .map(ThaiWordSegmentationModel._parse_record) \
        .shuffle(buffer_size) \
        .repeat() \
        .padded_batch(batch_size, padded_shapes=([], [None], [None]))

# Read validation data from TFRecord file and pad to the longest sentence
#3
@staticmethod
def _read_validation_dataset(data_file, batch_size):
    return tf.data.TFRecordDataset([data_file], compression_type="ZLIB") \
        .map(ThaiWordSegmentationModel._parse_record) \
        .padded_batch(batch_size, padded_shapes=([], [None], [None]))

# Get iterators for training and validation data, a handle variable for alternating
# between the iterators and a data batch
#5
@staticmethod
def _init_iterators(training_dataset, validation_dataset):
    handle = tf.placeholder(tf.string, shape=[])
    iterator = tf.data.Iterator.from_string_handle(handle,
                                                   training_dataset.output_types,
                                                   training_dataset.output_shapes)

    batch = iterator.get_next()

    training_iterator = training_dataset.make_one_shot_iterator()
    validation_iterator = validation_dataset.make_initializable_iterator()

    return training_iterator, validation_iterator, handle, batch
```

```
    # Build layers for character embeddings and bi-directional RNN with GRU cells
    #7
    @staticmethod
    def _build_embedding_rnn(tokens, lengths, vocabulary_size, state_size, dropout):
        embedding_weights = tf.Variable(tf.random_uniform([vocabulary_size, state_size], -1.0,
1.0))
        embedding_vectors = tf.nn.embedding_lookup(embedding_weights, tokens)

        cell = tf.nn.rnn_cell.GRUCell(state_size)
        cell = tf.nn.rnn_cell.DropoutWrapper(cell, output_keep_prob=1-dropout)

        (forward_output, backward_output), _ = \
            tf.nn.bidirectional_dynamic_rnn(cell, cell, inputs=embedding_vectors,
                                            sequence_length=lengths, dtype=tf.float32)
        outputs = tf.concat([forward_output, backward_output], axis=2)

        return embedding_vectors, outputs

    # Build fully connected output layer and postprocess output data
    #8
    @staticmethod
    def _build_classifier(inputs, labels, lengths, num_output_labels):
        logits = tf.layers.dense(inputs=inputs, units=num_output_labels, activation=None)
        losses = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=labels, logits=logits)

        mask = tf.sequence_mask(lengths)
        loss = tf.reduce_mean(tf.boolean_mask(losses, mask))
        masked_prediction = tf.boolean_mask(tf.argmax(logits, axis=2), mask)
        masked_labels = tf.boolean_mask(labels, mask)

        return loss, mask, masked_prediction, masked_labels
```

```
    # Define training optimizer to minimize prediction loss
    #9
    @staticmethod
    def _build_optimizer(loss):
        global_step = tf.Variable(0, trainable=False)
        learning_rate = tf.placeholder(tf.float32, shape=[])
        optimizer = tf.contrib.layers.optimize_loss(loss=loss, global_step=global_step,
            learning_rate=learning_rate, optimizer='Adam')
        return optimizer, global_step, learning_rate

    #6
    @staticmethod
    def _build_graph(tokens, labels, lengths, vocabulary_size, num_output_labels, dropout,
state_size):
        training = tf.placeholder(tf.bool, shape=[])

        # Embedding and bidirectional RNN layer
        _, rnn_outputs = ThaiWordSegmentationModel._build_embedding_rnn(lengths=lengths,
tokens=tokens,
            vocabulary_size=vocabulary_size, state_size=state_size,
            dropout=tf.cast(training, tf.float32) * dropout)

        # Output layer
        loss, mask, masked_prediction, masked_labels = \
            ThaiWordSegmentationModel._build_classifier(inputs=rnn_outputs, labels=labels,
            lengths=lengths, num_output_labels=num_output_labels)

        # Optimizer
        optimizer, global_step, learning_rate =
ThaiWordSegmentationModel._build_optimizer(loss=loss)
```

```
        return loss, masked_prediction, masked_labels, optimizer, global_step, learning_rate,
training


    #1
    def __init__(self, training_data_file, validation_data_file, buffer_size, batch_size,
vocabulary_size,
                 num_output_labels, state_size, dropout):
        tf.reset_default_graph()

        # Load training and validation data sets
        training_dataset =
ThaiWordSegmentationModel._read_training_dataset(training_data_file, batch_size,
buffer_size)
        validation_dataset =
ThaiWordSegmentationModel._read_validation_dataset(validation_data_file, batch_size)

        # Initialize training and validation data iterators
        self.tf_training_iterator, self.tf_validation_iterator, self.tf_iterator_handle, tf_batch = \
            ThaiWordSegmentationModel._init_iterators(training_dataset=training_dataset,
                                                      validation_dataset=validation_dataset)
        # Break the batch into a batch of each variable
        self.tf_lengths_batch, self.tf_tokens_batch, self.tf_labels_batch = tf_batch

        # Build the neural graph
        self.tf_loss, self.tf_masked_prediction, self.tf_masked_labels, self.tf_optimizer, \
            self.tf_global_step, self.tf_learning_rate, self.tf_training = \
            ThaiWordSegmentationModel._build_graph(tokens=self.tf_tokens_batch,
labels=self.tf_labels_batch,
            lengths=self.tf_lengths_batch, vocabulary_size=vocabulary_size,
num_output_labels=num_output_labels,
            dropout=dropout, state_size=state_size)
```

```
    # Save the trained model to a file in case you want to stop and continue training
    # (with different hyperparameters)
    #12
    def _restore_checkpoint(self, session, saver, checkpoint_path):
        saver.restore(session,
tf.train.get_checkpoint_state(checkpoint_path).model_checkpoint_path)
        return session.run(self.tf_global_step)

    # Measure accuracy of the word boundary prediction using precision, recall and F1 score
    #11
    @staticmethod
    def _evaluate(tag, iteration, loss, observed, predicted):
        precision = precision_score(observed, predicted) * 100
        recall = recall_score(observed, predicted) * 100
        f1 = f1_score(observed, predicted) * 100
        print('{}: Iteration {}, Loss {:.5f}, Precision {:2.2f}, Recall {:2.2f}, F1 {:2.2f}' \
              .format(tag, iteration, loss, precision, recall, f1))
        return precision, recall, f1

    #10
    def train(self, learning_rate, validate_every_n_iterations, checkpoint_path,
restore_checkpoint=False):
        # Start session and initialize variables
        global_init_op = tf.global_variables_initializer()
        session = tf.Session()
        session.run(global_init_op)

        # Get handles of training and validation iterators
        training_handle = session.run(self.tf_training_iterator.string_handle())
        validation_handle = session.run(self.tf_validation_iterator.string_handle())

        last_global_step = 0
```

```
        saver = tf.train.Saver(pad_step_number=True)

        # Restore checkpoint if requested
        if restore_checkpoint == True:
            last_global_step = self._restore_checkpoint(session=session, saver=saver,
                                                        checkpoint_path=checkpoint_path)

        # Train forever, hence stop the training when you are happy with the result
        iteration = last_global_step
        while True:
            for _ in range(validate_every_n_iterations):
                iteration += 1
                labels_batch, loss, masked_labels, masked_prediction, optimizer = \
                    session.run([self.tf_labels_batch, self.tf_loss, self.tf_masked_labels,
                                 self.tf_masked_prediction, self.tf_optimizer],
                    feed_dict = {self.tf_iterator_handle: training_handle, self.tf_training: True,
                                 self.tf_learning_rate: learning_rate})
                ThaiWordSegmentationModel._evaluate('Training', iteration, loss, masked_labels,
masked_prediction)

            # Validate and save checkpoint
            masked_labels_all = np.empty([0], dtype=int)
            masked_prediction_all = np.empty([0], dtype=int)
            session.run(self.tf_validation_iterator.initializer)
            try:
                while True:
                    labels_batch, loss, masked_labels, masked_prediction = \
                        session.run([self.tf_labels_batch, self.tf_loss, self.tf_masked_labels,
                                     self.tf_masked_prediction],
                        feed_dict = {self.tf_iterator_handle: validation_handle, self.tf_training:
False})
                    masked_labels_all = np.append(masked_labels_all, masked_labels)
```

```
                masked_prediction_all = np.append(masked_prediction_all,
masked_prediction)
            except tf.errors.OutOfRangeError:
                _, _, f1 = ThaiWordSegmentationModel._evaluate('Validation', iteration, loss,
masked_labels_all,
                                                               masked_prediction_all)
                saver.save(session, os.path.join(checkpoint_path, 'model_{:2.2f}'.format(f1)),
global_step=iteration)

        session.close()

    # Read checkpoint and save model to file for predictions
    #13
    def save_model(self, checkpoint_path, saved_model_path,
                   signature_name = tf.saved_model.DEFAULT_SERVING_SIGNATURE_DEF_KEY):
        with tf.Session() as session:
            global_init_op = tf.global_variables_initializer()
            session.run(global_init_op)

            saver = tf.train.Saver()
            saver.restore(session, tf.train.latest_checkpoint(checkpoint_path))

            inputs = {'inputs': tf.saved_model.utils.build_tensor_info(self.tf_tokens_batch),
                      'lengths': tf.saved_model.utils.build_tensor_info(self.tf_lengths_batch),
                      'training': tf.saved_model.utils.build_tensor_info(self.tf_training)}
            outputs = {'outputs':
tf.saved_model.utils.build_tensor_info(self.tf_masked_prediction)}
            builder = tf.saved_model.builder.SavedModelBuilder(saved_model_path)
            prediction_signature = (tf.saved_model.signature_def_utils.build_signature_def(
                inputs = inputs,
                outputs = outputs,
                method_name = tf.saved_model.signature_constants.PREDICT_METHOD_NAME
```

```
))
builder.add_meta_graph_and_variables(
    session,
    [tf.saved_model.tag_constants.SERVING],
    signature_def_map={
      signature_name: prediction_signature
    }
)

builder.save()
```