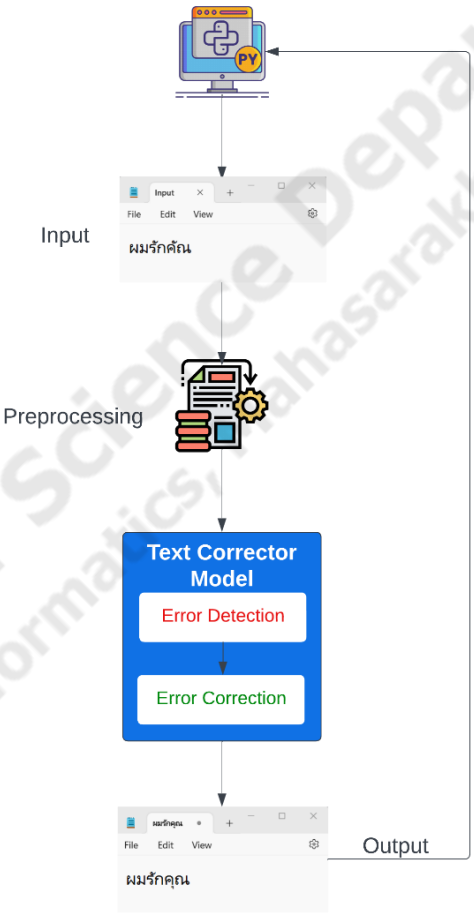
บทที่ 3

วิธีดำเนินงานวิจัย

ในบทนี้จะอธิบายถึงชุดข้อมูลเอกสารการตรวจสอบความถูกต้องข้อความภาษาไทยที่ใช้ในโครงงานนี้ และวิธีการดำเนินในการตรวจสอบความถูกต้องข้อความภาษาไทย ดังนี้

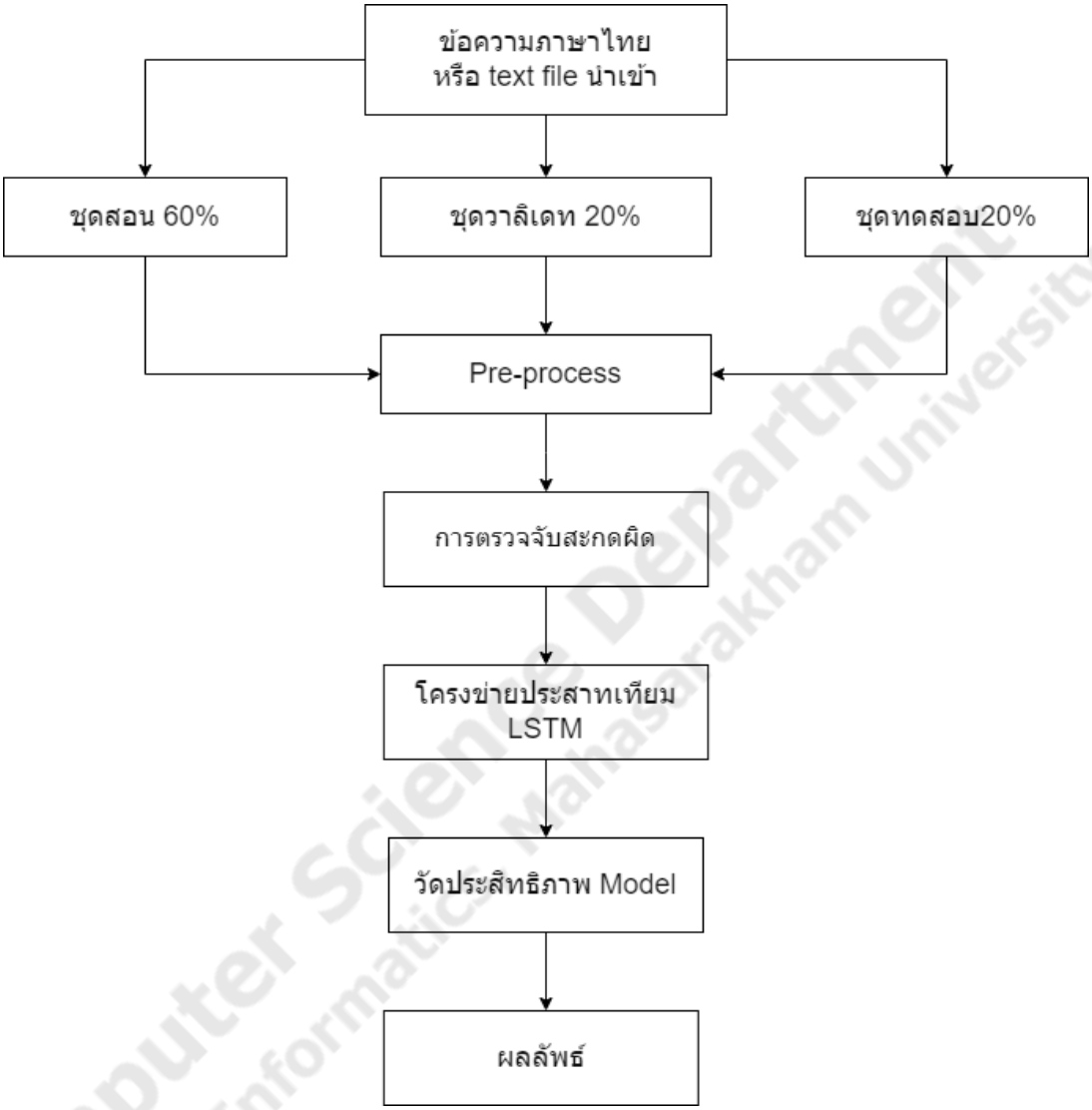
3.1 กรอบการดำเนินงาน



รูปภาพประกอบที่ 3.1 ภาพรวมระบบ

จากรูปภาพประกอบที่ 3.1 คือการทำงานของระบบ โดยที่ผู้ใช้เข้าใช้งานในโปรแกรม ผู้ใช้สามารถป้อนข้อมูลเป็นข้อความภาษาไทย หรือนำเข้าเป็น Text File ที่ต้องการประมวลผล จากนั้นระบบจะทำการ Pre-Processing ในส่วนของการตัดคำ (Word Segmentation) ก่อนนำเข้าสู่ Text Correction Model โดยมี 2 ขั้นตอน คือ Error Detection และ Error Correction โดยใช้ Deep Learning

3.2 การทำงานของส่วนของการสร้าง Model



รูปภาพประกอบที่ 3.2 ตัวอย่างการทำงานการสร้าง Model

จากภาพประกอบที่ 3.2 ชุดข้อมูลที่ถูกนำเข้าจะต้องผ่านการเตรียมข้อมูล เมื่อเสร็จแล้วจะแบ่งเป็นชุดทดสอบและชุดเรียนรู้ก่อนจะนำไปสู่การประมวลผลด้วยโครงข่ายประสาทเทียม LSTM จากนั้นจะทำการวัดประสิทธิภาพของโมเดล และจะคำนวณผลลัพธ์

3.2.1 การรวบรวมข้อมูล

ในส่วนนี้จะอธิบายถึงการรวบรวมข้อมูลและลักษณะข้อมูลที่ใช้ในการทดสอบการตรวจสอบความถูกต้องข้อความภาษาไทย ข้อมูลเอกสารที่ใช้ในโครงงานฉบับนี้ ได้นำข้อมูลมาจากโครงงานการประมวลผลลายมือเขียนเป็นตัวพิมพ์อัตโนมัติ จำนวน 1,504,138 อักขระ จากไฟล์ภาพ 61,184

ข้อความ โดยเอกสารจัดเก็บในรูปแบบไฟล์ นามสกุล .txt สำหรับตัวอย่างเอกสารที่ใช้ในการทดสอบตัวอย่าง Text File จะแสดงในภาพที่ 3.3

รูปภาพประกอบที่ 3.3 ตัวอย่างเอกสารที่ใช้ในการทดสอบ

3.2.2 ขั้นตอนการทำงานของ Pre Processing

ในส่วนนี้จะอธิบายการทำงานของการตัดคำ โดยการตัดคำจะเลือกใช้โมดูล Pythainlp ซึ่งเป็นโมดูลที่ถูกพัฒนาขึ้นเพื่องานวิจัยและการพัฒนาประมวลผลภาษาธรรมชาติของภาษาไทย [Natural Language Processing: NLP] ฟังก์ชั่นที่ใช้ในการตัดคำ คือ Word Token ซึ่งในโมดูลนี้มีหลายฟังก์ชั่นให้เลือกใช้ โดยงานวิจัยนี้เลือกใช้การตัดคำแบบ Newmm และ Deep Cut

(1) การตัดคำด้วย Newmm

การตัดคำด้วยอัลกอริทึม Newmm [15] เป็นการตัดคำโดยการหา Maximum Matching จาก ใน Dictionary (TCC) ซึ่งวิธีการตัดคำแบบ Maximal Matching (MM) นี้ จะนำเอาเฉพาะประโยคภาษาไทย โดยจะนำประโยคนั้นไปเปรียบเทียบกับคำศัพท์ที่อยู่ในพจนานุกรม หากเปรียบเทียบแล้วพบว่ามีในพจนานุกรมแสดงว่าเป็นคำที่มีความหมาย แต่หากไม่พบในพจนานุกรมจะทำการตัดอักขระโดยตัดอักขระทีละตัวจากข้างหลัง และนำไปเปรียบเทียบอีกครั้ง และทำซ้ำไปเรื่อย ๆ จนหมดประโยค และจะทำไปเรื่อย ๆ จนหมดบทความนั้น ๆ โดยข้อความที่ใช้ตัดอาจเป็นประโยคที่มีบรรทัดเดียวหรือหลายบรรทัดก็ได้ โดยมีอัลกอริทึมทำงาน Maximal Matching โดยเริ่มจากการหาทางเลือกของรูปแบบการตัดคำทั้งหมดที่เป็นไปได้ก่อน โดยทำการย้อนกลับ (Backtracking) ทีละคำ หลังจากได้คำตอบจากวิธีการเทียบคำที่ยาวที่สุดแล้ว แล้วจึงเลือกทางเลือกที่มีจำนวนคำน้อยที่สุด

ตัวอย่างเช่น ไปหามเหสี ขั้นตอนวิธีนี้จะหาทางเลือกทั้งหมดของรูปแบบการตัดคำที่เป็นไปได้ ได้แก่ ไปหามเหสี

ตารางที่ 3.1 แสดงผลลัพธ์การตัดคำ Newmm

ไป หาม เห สี	
ไป หา มเหสี	Maximal Matching

โดยในขั้นตอนที่หนึ่ง จะใช้วิธีการเทียบคำที่ยาวที่สุดจะได้คำตอบของการตัดคำเป็น “ไป-หาม-เห-สี” ก่อน หลังจากนั้นจึงเริ่มทำการย้อนกลับโดยเริ่มจากคำแรก พบว่า คำที่สอง “หาม” สามารถ แบ่งได้เป็น “หา-ม” ได้โดยเมื่อแบ่งเป็น “หา” แล้ว ทำให้สายอักขระที่เหลือ “มเหสี” สามารถตัดคำคือ “ไป-หา-มเหสี” เมื่อทำการย้อนกลับกับทุกคำสิ้นสุดแล้วจะไปสู่ขั้นตอนที่สอง โดยเลือกทางที่จำนวนคำน้อยที่สุดจากวิธีการเทียบคำที่ยาวที่สุด คือคำว่า “ไป หา มเหสี”
โดย Newmm ที่เราเลือกใช้ได้มีการพัฒนาพจนานุกรมใหม่สำหรับใช้เพื่อการตัดคำภาษาไทยซึ่งได้รับการเพิ่มคำ

(2) การตัดคำด้วย Deep Cut

หลักการของ Deep Cut [17] จะใช้ Deep Learning Model CNN หาความน่าจะเป็นของอักษรแต่ละตัวว่าเป็นตัวเริ่มต้นของคำหรือไม่ โดยการทำนายแบบ Binary Classification ซึ่งค่าที่ได้จะออกมาเป็น 0 หรือ 1 เท่านั้นโดยความหมายของ 0 คือไม่เป็นตัวเริ่มต้นของคำ และ 1 เป็นตัวเริ่มต้นของคำ โดยการสร้าง Model Deep Cut นี้จะนำข้อมูลในการเรียนรู้ Model มาจาก BEST I Corpus ซึ่งเป็นข้อความภาษาไทยที่มีการรวบรวมไว้เพื่อพัฒนามาตรฐานการประมวลภาษาไทย ในการเรียนรู้ของ Model Deep Cut นั้นจะใช้ข้อความที่เป็นประโยคเป็นข้อมูลนำเข้าและมีผลเฉลยเป็นประโยคที่ถูกตัดคำแล้วโดยจะเป็นค่าตัวเลขแทนตัวอักษรแต่ละตัวโดยมีค่า 0 และ 1 ซึ่งมี 0 นั้นมีความหมายว่าไม่เป็นอักษรเริ่มต้นของคำส่วน 1 นั้นมีความหมายว่าเป็นอักษรเริ่มต้นของคำ เมื่อได้ Model ที่ใช้ในการตัดคำแล้วจึงนำมาใช้ตัดคำที่เป็นผลเฉลยของงานนี้โดยมีขั้นตอนการตัดคำดังนี้

ตัวอย่างประโยค เช่น “อากาศร้อน” นำเข้า Model Deep Cut เพื่อตัดคำ จากประโยคนี้จะถูกแบ่งออกเป็นตัวอักษรแต่ละตัวก่อน จะได้ดังนี้

ตารางที่ 3.2 แสดงการแบ่งตัวอักษรจากประโยค

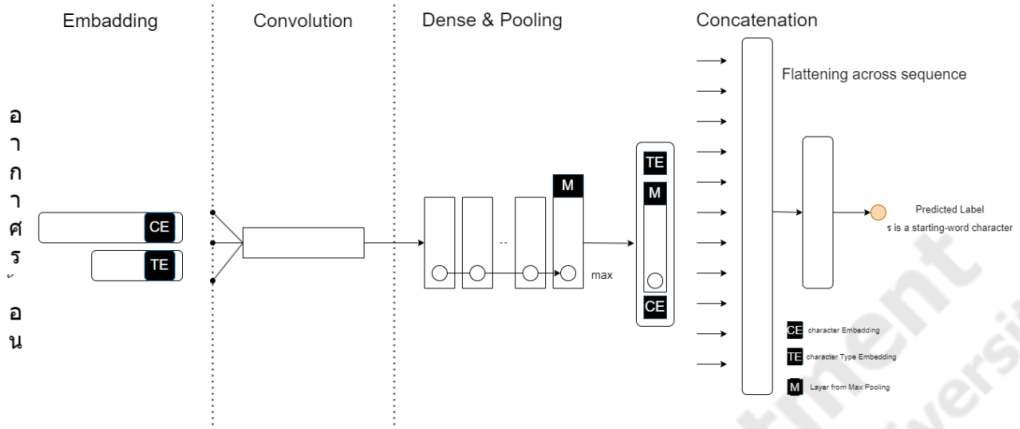
อ	า	ก	า	ศ	ร		อ	น

จากนั้นจะนำมาหาความน่าจะเป็นของแต่ละตัวอักษรโดยอ้างอิงจากในฐานข้อมูล BEST ของ NECTEC ว่าเป็นตัวเริ่มต้นของคำหรือไม่หากได้ความน่าจะเป็นตั้งแต่ 0.5 ขึ้นไปจะถูกตัดเป็น 1 และหากไม่ถึงจะถูกตัดเป็น 0 จะได้ดังนี้

ตารางที่ 3.3 แสดง output ของการตัดคำ

Input	อ	า	ก	า	ศ	ร		อ	น
Output	0	0	0	0	1	0	0	0	1

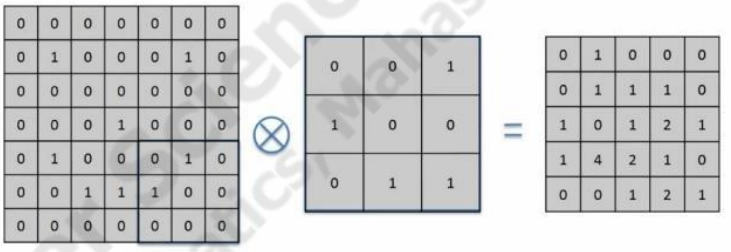
หลักการทำงานของ Deep Cut



รูปภาพประกอบที่ 3.4 ภาพรวมการทำงานของ Deep Cut

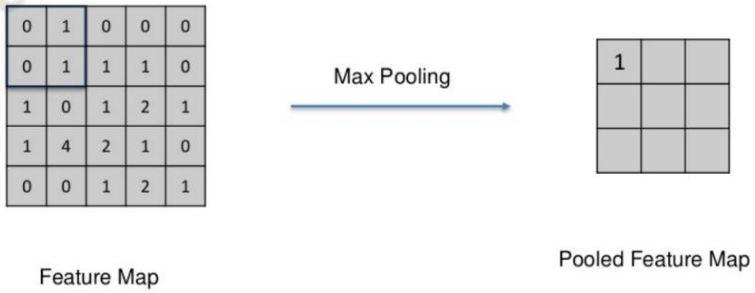
ขั้นที่ 1 สร้างตัวแปลงข้อมูลด้วยอัลกอริทึม Word Embedding จากข้อความเป็นเวกเตอร์

ขั้นที่ 2 การทำงานของในขั้นตอนสกัดคุณลักษณะของข้อความออกมา (Convolution) โดยการใช้ Filter โดยเอาค่าในภาพมาคูณกับค่าใน Filter ที่ตำแหน่งตรงกัน ทำจนครบขนาดของ Filter แล้วหาผลรวมของทุกตำแหน่ง จะได้ Feature Map จำนวนมากโดยจะเรียกทั้งหมดว่า Convolutional Layer

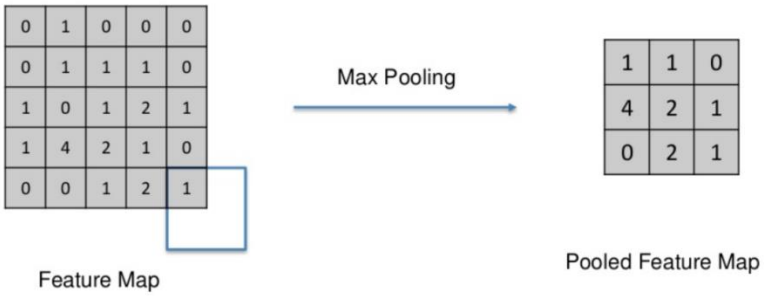


รูปภาพประกอบที่ 3.5 ตัวอย่างภาพรวมการทำ Convolution

ขั้นที่ 3 Dense & Pooling หลังจากที่ทำ Convolution เสร็จแล้ว จะทำการลดขนาดของข้อมูล ซึ่งจะใช้ Max Pooling ในการคำนวณ

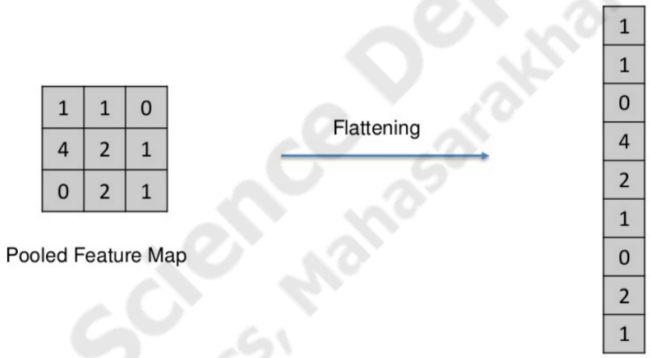


รูปภาพประกอบที่ 3.6 การทำงานของ Max Pooling



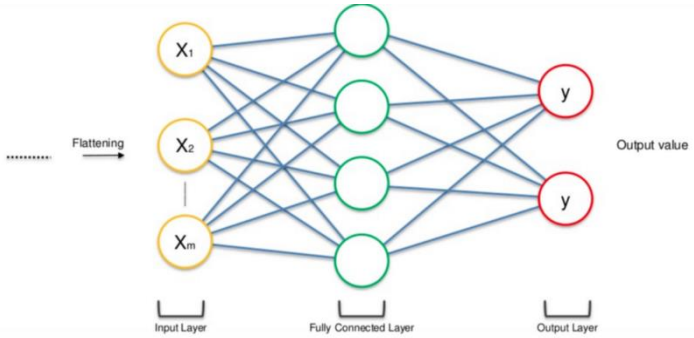
รูปภาพประกอบที่ 3.7 ผลลัพธ์ของ Feature Map

ขั้นที่ 4 Concatenation โดยการทำ Flattening ทำ Pooling Feature Map ที่ได้ทำเป็นคอลัมน์เดียวกัน ซึ่งจะถูกจัดให้อยู่ในรูปของอาเรย์ 1 มิติกระบวนการทั้งสามนี้เป็นกระบวนการที่ทำซ้ำได้หลายครั้ง



รูปภาพประกอบที่ 3.8 การทำงานของ Flattening

สุดท้ายมีการเชื่อมต่อกันของแต่ละชั้นอย่างสมบูรณ์ Fully Connected Layer ผลลัพธ์จากคอนโวลูชันและพูลลิ่ง นั้นให้ลักษณะเด่น (High-Level Features) ของข้อมูลที่รับเข้ามาและชั้นสุดท้ายเพื่อนำลักษณะเด่นไปทำการคัดกรองรับเข้าให้อยู่ในรูปของ Classes เพื่อนำค่าทำนายออกมาจะให้ค่าเป็น 0 หรือ 1 เท่านั้น



รูปภาพประกอบที่ 3.9 การเชื่อมโยงข้อมูล

3.2.3 หลักการทำงานของการแก้ไขข้อความ

3.2.3.1 การตรวจสอบคำผิด (Error Detection)

การตรวจจับการสะกดผิด เป็นขั้นตอนแรกของการตรวจแก้การสะกดคำมีหน้าที่ตรวจหาคำที่สะกดผิดในข้อความ การแก้การสะกดผิดเป็นงานที่มีนักวิจัยเสนอวิธีการต่าง ๆ แต่ยังไม่มีการระบุที่ชัดเจนว่าใช้วิธีใด โดยส่วนใหญ่วิธีการที่นำมาประยุกต์ใช้นั้นมีความแตกต่างกันตามวัตถุประสงค์และประยุกต์ใช้แก้ปัญหา รวมไปถึงการเลือกใช้อัลกอริทึมใดที่จัดการกับปัญหาได้ดีที่สุด

ซึ่งเนื้อหาในส่วนนี้จะ กล่าวถึงวิธีการที่นำมาใช้จัดการตรวจแก้ปัญหาการสะกดผิด โดยใช้ Spellchecker ใน Pythainpl ใช้อัลกอรึทึม Norvig Spell Checker algorithm [14] คือ อัลกอรึทึมสำหรับ Word based Spell Corrector ที่สร้างขึ้นโดย Peter Norvig ในการตรวจจับคำผิด ซึ่งหลักการทำงานของอัลกอรึทึม จะอาศัยคลังศัพท์พจนานุกรมจาก Thai National Corpus (TNC) [16] โดยมีสมการคำนวณดังนี้

From B yes' Theorem;

$$\text{argmax}_{c \in \text{candidates}} \frac{P(c)|P(c|w)}{P(w)} \tag{6}$$

$$\text{argmag}_{c \in \text{candidates}} P(c)P(w|c) \tag{7}$$

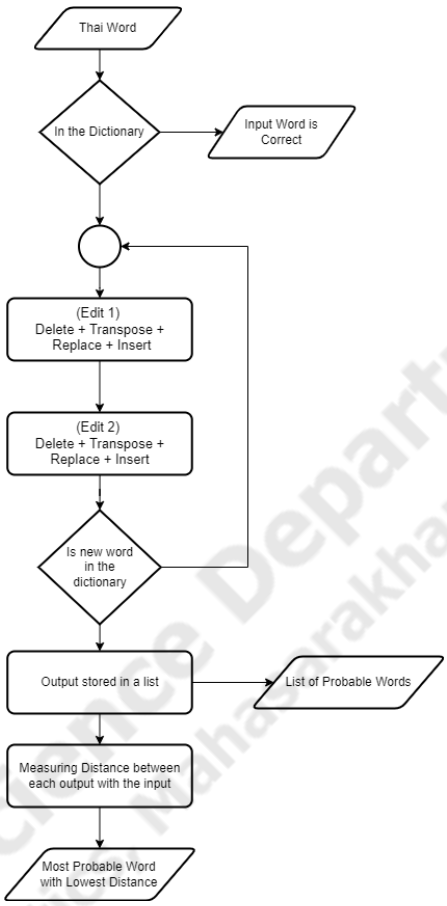
การทำงานของสมการข้างต้น คือ จากคำที่อยู่ w ที่เราต้องการที่จะตรวจสอบ เราต้องการหาคำที่อยู่ใน c ที่มีความน่าจะเป็นที่มากที่สุด เมื่อ c เป็นคำ Candidate ที่อาจจะมีการสะกดที่ใกล้เคียง w แต่ว่าเป็นคำที่เรานับว่า "สะกดได้ถูกต้อง"

P(c) คือ Language Model (เป็น N-gram) บอกค่าความน่าจะเป็นที่คำ c ปรากฏอยู่ในฐานข้อมูลภาษา เช่น P(the) มีค่าสูงกว่า p(thaw) เพราะปรากฏบ่อยกว่าในการใช้ภาษา

P(w|c) คือ Error Model บอกค่าความน่าจะเป็นที่ผู้เขียนจะพิมพ์ผิดว่า w เมื่อจริง ๆ แล้วคำที่ถูกต้องคือ c

P(c|w) คือ ความน่าจะเป็นที่จริง ๆ แล้วเป็นคำ c เมื่อคำที่เราพิมพ์คือ w

โดยมีรูปแบบผังการทำงานดังนี้



รูปภาพประกอบที่ 3.10 ผังงานของวิธีการแก้ไขคำภาษาไทย

กระบวนการในรูปภาพประกอบที่ 3.10 โดยขั้นแรก ระบบจะต้องใช้คำภาษาไทยเป็นอินพุต คำที่รับเข้าอาจจะถูกต้องหรือสะกดผิด แล้วจะทำการค้นหาในพจนานุกรมเพื่อค้นหาคำ หากพบคำในคลังพจนานุกรม ระบบจะประกาศคำที่ป้อนเป็นคำที่สะกดถูกต้องและจะหยุดขั้นตอนการทำงานสำหรับคำนี้ แต่ถ้าไม่พบคำนั้นในพจนานุกรมที่มีอยู่ ระบบจะพยายามสร้างรายการคำสำหรับคำที่ถูกต้อง ระบบจะไปแก้ไขอักขระ 1 ตัวก่อน จากนั้นจึงแก้ไขอักขระ 2 ตัว ในทั้งสองกรณี คำจะถูกแบ่งด้วยอักขระหนึ่งตัวและตามการแยก และจะมีการลบอักขระ N การแทรก N-character การเปลี่ยนตำแหน่งภายใน N-Character และแทนที่ N-Character (s) ด้วยอักขระ N ใหม่จากชุดข้อมูลของอักขระแต่ละตัวตามผลลัพธ์ของชุดแยก โดยที่ n สามารถเป็น 1 หรือ 2 ได้ จากนั้นจะสร้างรายการแบบยาวที่มีผลลัพธ์ทั้งหมดของขั้นตอนทั้งหมดที่กล่าวมา

รายการคำศัพท์จะถูกจับคู่กับคำศัพท์ในพจนานุกรมและคำที่ถูกต้องจะแสดงเป็นคำแนะนำสำหรับคำที่สะกดผิดนั้น ในบรรดาข้อเสนอแนะเหล่านั้น ข้อที่น่าจะเป็นไปได้มากที่สุดเลือกตามหมายเลขดัชนี หมายเลขดัชนีถูกสร้างขึ้นโดยระยะทางของอัลกอริทึมความคล้ายคลึงกันของสตริง พบ

ผลลัพธ์ที่ดีที่สุดโดยการใช้อัลกอริทึมระยะทาง Jaro-Wrinkler ยิ่งระยะทางต่ำ ความน่าจะเป็นของดัชนียิ่งสูงขึ้น ดัชนีสูงสุดที่แสดงถึงคำจากรายการสุดท้ายจะกลายเป็นคำที่น่าจะเป็นไปได้มากที่สุด

ตัวอย่างคำที่สะกดผิด เมื่อนำประโยคไปตรวจสอบการสะกดผิด โดยการตัดคำของ Deep Cut และนำคำที่ได้ไปตรวจสอบกับไลบรารี่ Spell() ใน Pythainlp โดยจะแสดงคำที่มีความน่าจะเป็นสูงสุดใน Dictionary มาแสดงได้ผลลัพธ์ดังนี้

```
1 from pythainlp import spell
2 spell( "ความี")
```

```
['ความ', 'ความดี', 'สวามี', 'คามี']
```

รูปภาพประกอบที่ 3.11 ตัวอย่างคำสั่งโค้ด ที่ใช้ Spell()

```
1 spell( "ร้อ4")
```

```
['ร้อน', 'ร้อง', 'ร้อย', 'ร้อก', 'ร้อ', 'ร้อบ', 'ร้อด', 'ร้อม']
```

รูปภาพประกอบที่ 3.12 ตัวอย่างผลลัพธ์ของคำที่ตรวจสอบ

ตัวอย่างประโยค “ความีโสดไม่ใช่เนื้อคู่” นำประโยคมาปรับใช้กับไลบรารี่ Spell() ใน Pythainlp จะต้องนำประโยคไปตรวจสอบการสะกดผิด และผ่านการตัดคำของ Deepcut โดยจะแสดงผลลัพธ์ดังนี้

```
True :  ความโสดไม่ใช่เนื้อคู่
Input :  ความีโสดไม่ใช่เนื้อคู่
spellCheck : ความีโสดไม่ใช่เนื้อคู่
nearby :  ['ความ', 'ความดี', 'สวามี', 'คามี']
```

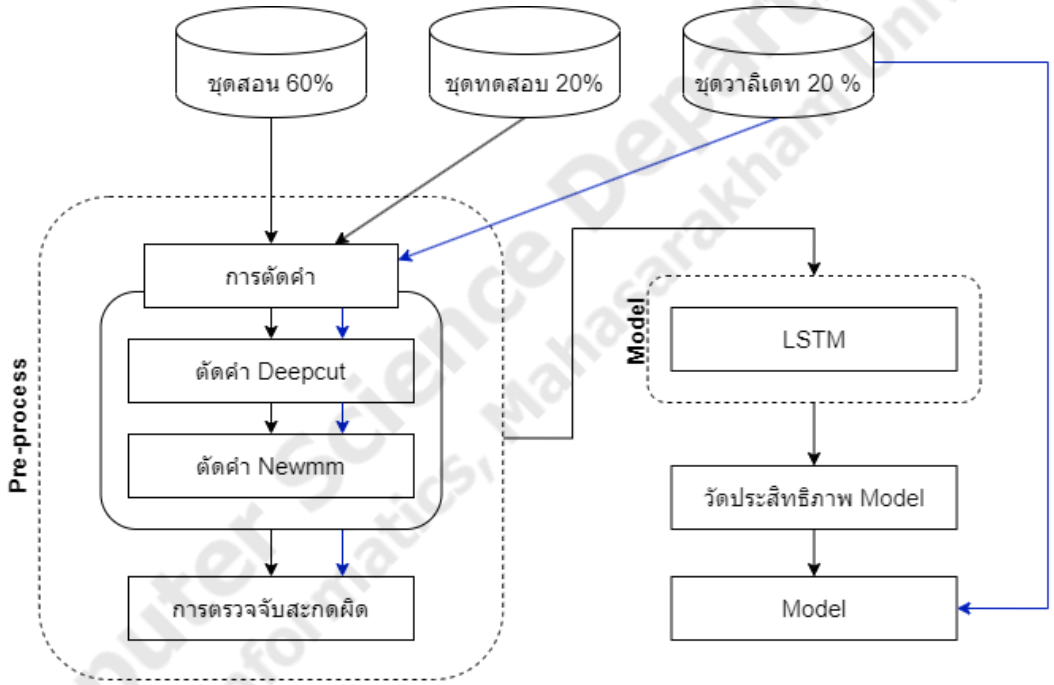
รูปภาพประกอบที่ 3.13 ตัวอย่างผลลัพธ์ของประโยคมาปรับใช้กับไลบรารี่

3.2.3.2 การแก้ไขการสะกดผิด (Error Correction)

การแก้ไขการสะกดผิดนั้น จะทำการแก้ไขได้นั้นต้องผ่านขั้นตอนแรกคือการตรวจจับการสะกดำผิดมาก่อน ซึ่งในการแก้ไขการสะกดำผิดนั้น จะทำการแก้ไขได้จำเป็นต้องมีข้อมูลของบริบท ดังนั้นบริบทที่เกิดขึ้นมีหลายรูปแบบ ซึ่ง Model ที่ใช้และสามารถตอบสนองได้ดีในตอนนี้คือ Recurrent

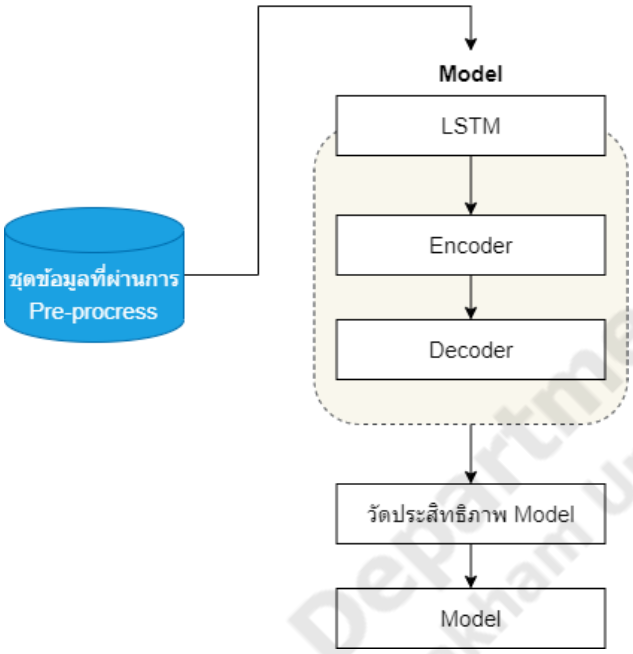
Neural Network ที่เป็น Long Short-Term Memory (LSTM) ดังนั้นในการแก้ไขคำผิดนั้น เราจะใช้หลักการ Long Short-Term Memory โดยจะทำการ Input ข้อมูลเข้าไปเพื่อฝึกฝนโดยขั้นตอนการฝึกฝนมีขั้นตอนดังนี้

ขั้นตอนแรก ชุดข้อมูลที่ใช้ในการเรียนรู้และการทดสอบ เริ่มต้นด้วยการแบ่งข้อมูลออกเป็น 3 ส่วน ชุดข้อมูลที่หนึ่งไว้สำหรับเรียนรู้ (Training Set) 60% ชุดข้อมูลที่สองใช้สำหรับการทดสอบ (Test Set) 20% และชุดข้อมูลที่สามใช้สำหรับการทดสอบโมเดล (Validation Set) 20% จากนั้นข้อมูลในชุดเรียนรู้จะถูกนำไปเข้าสู่กระบวนการ Pre-processing เพื่อทำความสะอาดข้อมูลที่นำเข้า และใช้ชุดทดสอบในการทดสอบประสิทธิภาพของ Model



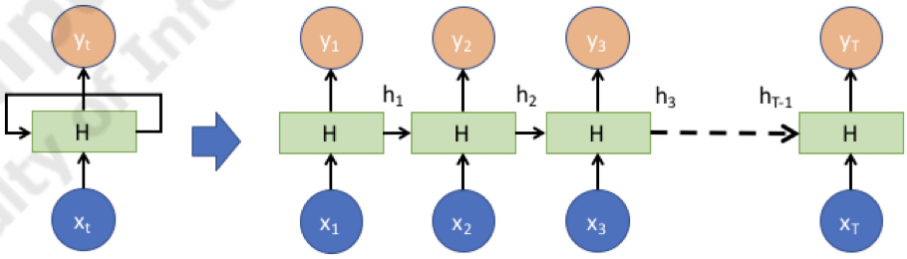
รูปภาพประกอบที่ 3.14 ชุดข้อมูลที่ใช้ในการเรียนรู้และทดสอบ

ขั้นตอนการทำงานในส่วนของ Model



รูปภาพประกอบที่ 3.15 การทำงานของ Model LSTM

ส่วนของ Recurrent Neural Network ที่เป็น Long Short Term Memory (LSTM) Cell งานประมวลผลภาษาธรรมชาติกับการเติมประโยคอัตโนมัติ สมมติว่าเรากำลังพยายามคาดเดาคำสุดท้าย เราจำเป็นต้องดูข้อมูลล่าสุดในเครือข่ายสำหรับงานทำนายเท่านั้น หากบริบทขึ้นอยู่กับการคาดการณ์ เครือข่ายที่เกิดซ้ำจำเป็นต้องมีเซลล์หน่วยความจำระยะสั้นระยะยาวดังรูปภาพประกอบที่ 3.16



รูปภาพประกอบที่ 3.16 การทำงานของ Long Short Term Memory (LSTM)

จากภาพประกอบที่ 3.16 จะเห็นว่ามี loop ที่วนกลับเข้ามาที่ Hidden Layer ของ Neural Network สิ่งที่สำคัญของ RNN คือ Hidden State ก่อนหน้าและ Input โดยประโยชน์หลักๆของ Loop แต่ละรอบคือการนำเอา Hidden State ก่อนหน้าเข้ามาเป็น Input Data ในรอบต่อ ๆ ไป เหมือนอย่างรูปทางด้านขวาที่คลี่ออกมาจากรูปทางด้านซ้ายเพื่อแสดงการทำงานทีละขั้นตอน

กำหนดให้

Input Data คือ x1, x2, ..., x_t

Hidden State ที่เวลา t จะใช้ตัวแปรว่า h_t โดยที่

H = Hidden Layer

y_t= Output จาก RNN ที่เวลา t

x_t= Input Data ที่เวลา t

h_t= Hidden State ที่เวลา t

f_h คือ Activation Function ของ Hidden Layer

f_y คือ Activation Function ของ Output Layer

W_h คือ Weight Matrix ของ Hidden Layer

U_h คือ Hidden-State-to-Hidden-State Matrix

สมการในการคำนวณ Hidden State และ Output

$$h_t=f_h(U_h h_{t-1}+W_h x_t+b_h) \quad (8)$$

$$y_t=f_y(W_y h_t+b_y)$$

การที่จะคำนวณ Hidden State ที่เวลา t ออกมาได้นั้น (h_t) ก็จะต้องใช้ 2 ตัวแปรสำคัญ หรือ Hidden State ก่อนหน้า (h_t-1) และ Input Data ณ ตอนนั้น (x_t)

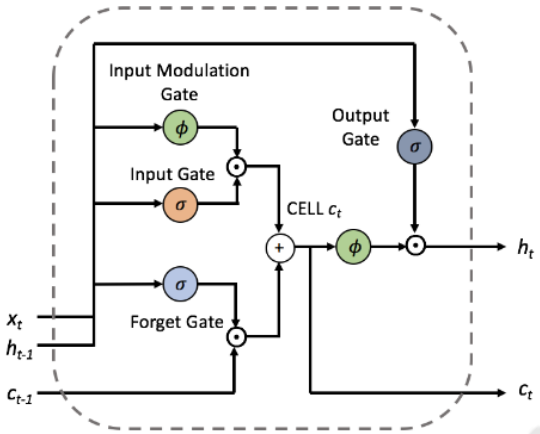
ในส่วนของ LSTM ส่วนที่สำคัญมีหน้าที่ดังนี้

Input Gate เป็นหน่วยย่อยในการกำหนดข้อมูลที่จะนำเข้ามาวิเคราะห์ใน Cell โดยรับข้อมูลเข้ามาเพื่อทำการเขียนค่าลงไปในแต่ละ Cell

Forget Gate เป็นหน่วยย่อยที่ใช้ในการกำหนดข้อมูลที่จะนำเข้ามาวิเคราะห์ใน Cell โดยทำการกำหนดว่าข้อมูลควรที่จะถูกบันทึกหรือถูกลืม

Memory Cell State Gate เป็นหน่วยย่อยในการกำหนดข้อมูลที่นำเข้ามาวิเคราะห์ใน Cell และทำการคำนวณค่าสถานะ เพื่อใช้ในการคำนวณในครั้งถัดไป

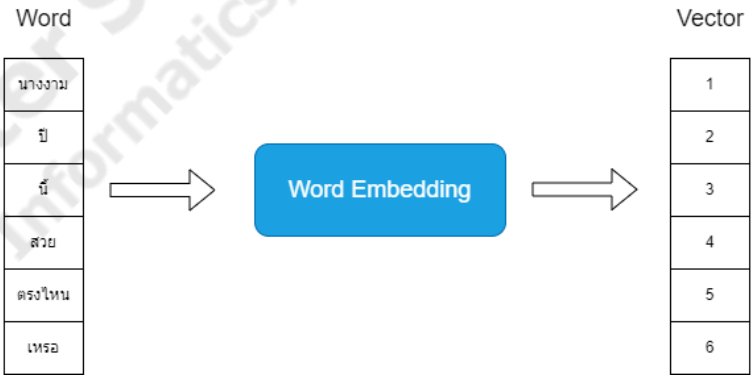
Output Gate เป็นหน่วยย่อยสำหรับการคำนวณ Output ของ Cell ซึ่งผลลัพธ์ที่ได้จาก Cell นั้นจะมีอยู่ 2 อย่าง ได้แก่ Output และ Hidden State สำหรับใช้ในการคำนวณครั้งถัดไป



รูปภาพประกอบที่ 3.17 ลำดับขั้นตอนของ LSTM

ขั้นตอนในส่วนตัวแก้ไขข้อความ Sequence-to-Sequence Model (Seq2Seq) เป็นโมเดลสร้างเวกเตอร์ของคำ โดยนำเข้าข้อมูลเป็น Sequence ของหน่วยภาษา และผลลัพธ์เป็น Sequence ของหน่วยภาษา ภายในโมเดลประกอบด้วย Encoder ทำหน้าที่รับข้อความที่อาจจะมีข้อผิดมา แล้วแปลงให้อยู่ในรูปของเวกเตอร์โดยเทคนิค “Word Embedding” และ Decoder ทำหน้าที่รับเวกเตอร์โมเดลแบบ LSTM เช่นกัน ทำหน้าที่รับเวกเตอร์ไปสร้างเป็นข้อความที่ไม่มีข้อผิดที่ต้องการ

ขั้นตอนในส่วนของ Word Embedding คือ Model จะแทนคำหนึ่งคำด้วยชุดของตัวเลข (หรือที่เรียกว่า vector)



รูปภาพประกอบที่ 3.18 ตัวอย่างการทำงานของ Word Embedding

ขั้นตอนการสร้าง Sequences ของคำหรือลำดับของคำ การสร้าง Sequences ของคำเพื่อนำไปใช้ในการ Train Model โดยขั้นตอนการสร้าง Sequences ของคำจะทำโดยนำเข้าชุดข้อมูลเข้าโดยจะสร้าง Sequences ของคำ ทีละประโยค และเก็บ Sequences ของคำทั้งหมดที่ได้ไว้ใน Array เพื่อนำไป Train Model ในแต่ละรอบที่สร้าง Sequences ของคำจะถูกเก็บลงใน Array และจะถูกนำเอา Array ไปใช้ในการ Train Model

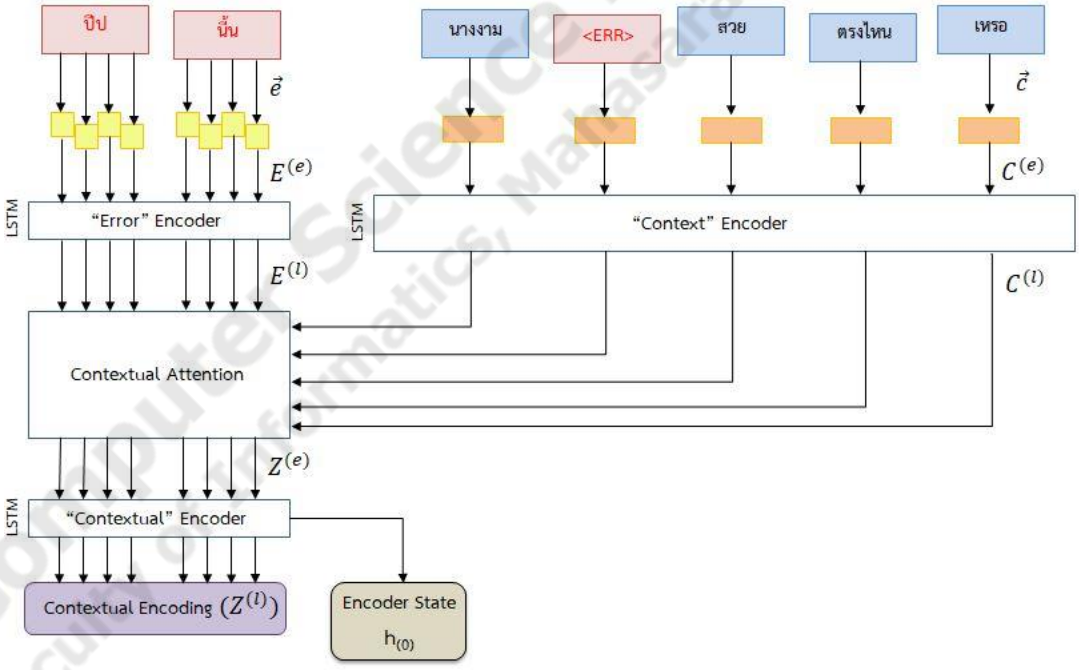
ตารางที่ 3.4 ตัวอย่างคำภาษาไทยที่นำเข้าและผลลัพธ์การเข้ารหัส

Input Sentences	Vector
นางงาม ปี นี้ สวย ตรงไหน เหรอ	1 2 3 4 5 6
วัน นี้ ท้อง ฟ้า สวย มาก	7 3 8 9 4 10
วัน นี้ ท้อง ฟ้า ไม สว่าง เลย	7 3 8 9 0 11 12

ตารางที่ 3.5 ตัวอย่างคลังคำที่นำไปเข้ารหัสจะได้ Index ดังนี้

คำศัพท์	UNK	นางงาม	ปี	นี้	สวย	ตรงไหน	เหรอ	วัน	ท้อง	ฟ้า	มาก	สว่าง	เลย
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

ขั้นตอนการทำงานในส่วนของ Encoder



รูปภาพประกอบที่ 3.19 การทำงานเข้ารหัส Encoder

จากรูปภาพประกอบที่ 3.19 ขั้นตอนการเข้ารหัส (Encoder) ตัวอย่างประโยคที่รับเข้ามาอาจจะมีคำผิด เช่น “นางงาม ปีป นี้น สวย ตรงไหน เหรอ”

Error Encoding เป็นประโยคที่ตัดคำที่ผ่านการฝังคำมาแล้ว โดยกำหนดให้

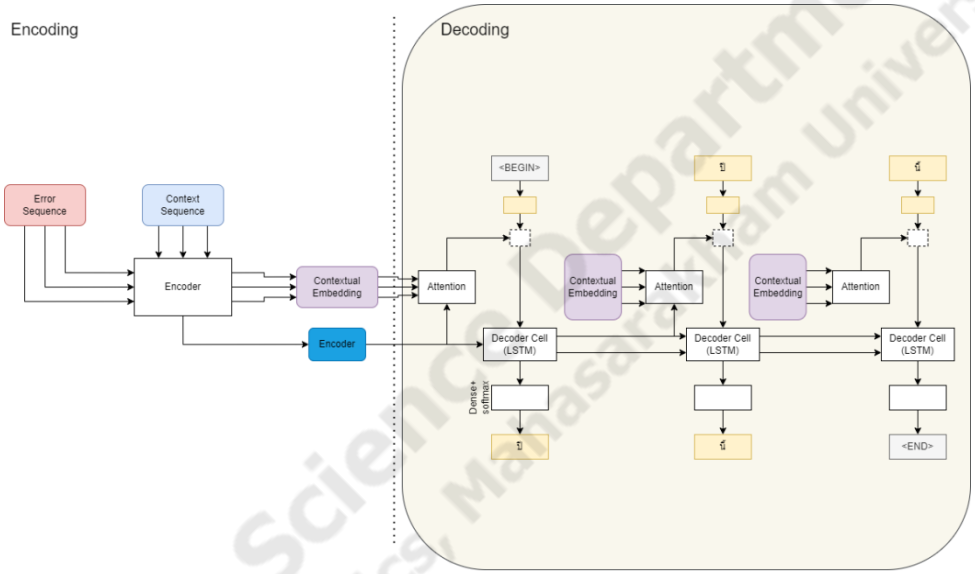
$\underset{e}{\rightarrow}$ = $\{e_1, e_2, \ldots, e_j\}$ คำที่ถูกฝังคำของคำแต่ละคำเป็น 2m Vector ในการสร้างเมทริกซ์ $E^{(e)}$

การฝังคำจะถูกเข้ารหัสโดย Error Encoder ส่งไปยังการเข้ารหัสผิดพลาด $E^{(l)}$

Context Encoding ประโยคตัดคำตามบริบท โดยกำหนดให้ $\overrightarrow{c}=\{c_1,c_2,...,c_k\}$ ที่ผ่านการทำฝังคำของคำแต่ละคำเป็น 2m vector ในการสร้างเมทริกซ์ $C^{(e)}$ เป็นการเข้ารหัสตามบริบท Context Encoder $C^{(l)}$

Contextual Encoding ในขั้นตอนนี้จะส่ง Error Encoder $E^{(l)}$ และ Context Encoder $C^{(l)}$ เป็นการรับเข้าใน Context Attention ไปคำนวณในชั้น Contextual Encoder $Z^{(e)}$ เป็นการเข้ารหัสตามบริบท Contextual Embedding $Z^{(l)}$

ขั้นตอนการทำงานในส่วนของ Decoder



รูปภาพประกอบที่ 3.20 การทำงานการถอดรหัส Decoder

จากรูปภาพประกอบที่ 3.20 เป็นส่วนที่เราจะนำ Contextual Embedding $Z^{(l)}$ ที่ได้จาก Encoder มาถอดความออกมาเป็นประโยคที่ถูกต้อง โดยเราจะใช้โครงสร้าง LSTM โดยเทคนิค Attention Mechanism

Decoder จะทำงานโดยการทำนายการตัดคำ $W_t^{(c)}$ ที่ได้รับการตัดคำคาดการณ์ก่อนหน้า $W_{t-1}^{(c)}$ ดังนั้นการตัดคำก่อนและหลังคำแก้ไข จะแสดงในสมการดังนี้ โดยให้ L คือจำนวนคำในการแก้ไข

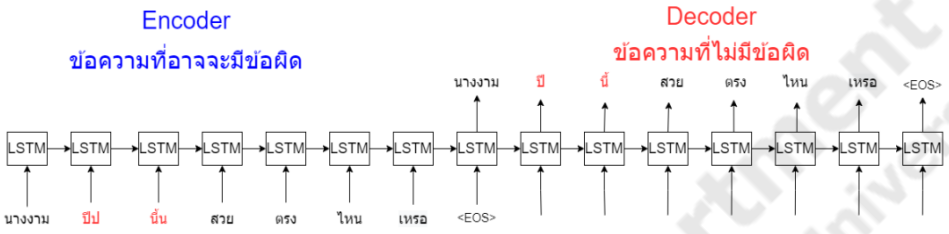
$$\overrightarrow{w}=\{BEGIN,w_1^{(c)},w_2^{(c)},...,w_L^{(c)},END\} \quad (9)$$

Decoder จะทำนายคำเรื่อย ๆ จนถึงขั้นตอน Time Step จุดสิ้นสุดของลำดับการแก้ไข t=L+1 โดยที่ <END> คือจบการทำงาน

Encoder state h_0 จะเริ่มต้นด้วยขั้นตอนสุดท้ายของ Contextual Encoder เป็นการรับข้อมูลไปยัง Decoder Cell (LSTM) และการตัดคำที่ทำจากขั้นตอน Time Step $W_{t-1}^{(c)}$ เป็น Embedding $e_t^{(c)}$เป็น Context Vector จะคำนวณด้วย Dot-Product attention จาก Hidden State

ไปยัง $Z^{(l)}$ Embedding e_t เป็นการต่อกันระหว่าง Word Embedding และ Context Vector ดังสมการ

$$e_t=[e_t^{(e)};e_t^{(c)}] \quad (10)$$
$$h_t,C_t=LSTM_{Decoder}(h_{t-1},c_{t-1},e_t)$$



รูปภาพประกอบที่ 3.21 ผลลัพธ์การทำ Sequence-to-Sequence Model (Seq2Seq)

3.3 การประเมินประสิทธิภาพ

3.3.1 การประเมินประสิทธิภาพในการตัดคำ จะใช้ในการตรวจสอบการตัดคำที่ถูกต้องที่สุด ซึ่งจะเปรียบเทียบประสิทธิภาพการตัดคำของ 2 อัลกอริทึมด้วยว่าเหมาะสมที่จะนำมาใช้หรือไม่ โดยการนำผลลัพธ์การตัดคำของ Deep Cut และ Newmm มาเปรียบเทียบกับผลเฉลยที่ได้จากการให้คนตรวจสอบมาทำการประเมินผล จากนั้นจะทำการประเมินประสิทธิภาพในการตัดคำโดยใช้ Benchmarks ใน Pythinlp [15] โดยเลือกใช้วิธีเทียบจาก Word Level โดย Benchmarks วัดประสิทธิภาพจาก Precision, Recall และ f1 มีหลักการดังนี้

Precision (ค่าความแม่นยำ) เป็นการเปรียบเทียบ การทำนายที่ถูกต้องว่า จริง และก็เกิดขึ้นจริง (TP) กับการทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง (FP)

$$\text{Precision} = TPs / (TPs + FPs) \quad (11)$$

Recall (ความถูกต้องของการทำนายว่าจะเป็น “จริง” เทียบกับ จำนวนครั้งของเหตุการณ์ทั้งทำนาย และ เกิดขึ้นว่า “เป็นจริง”)

$$\text{Recall} = TPs/(TPs + FNs) \quad (12)$$

F1-Score เป็นค่าเฉลี่ยแบบ Harmonic Mean ระหว่าง Precision และ Recall จุดประสงค์ของการสร้าง F1 ขึ้นมา คือ เพื่อเป็น Single Metric ที่วัดความสามารถของโมเดล

$$F1 = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall}) \quad (13)$$

คำสั่งที่ใช้ในวิธีเทียบจาก Word Level โดย Benchmarks

pythainlp.benchmarks.word_tokenization.benchmark(ref_samples: List[str], samples: List[str])

Parameters: ref_samples (list[str]) คือ ข้อมูลที่ถูกต้อง
samples (list[str]) คือ ข้อมูลที่ต้องการประเมิน

ตารางที่ 3.6 ตัวอย่างประโยคการตัดคำ

ข้อความ	ตัดคำโดยใช้ Newmm	ตัดคำโดยใช้ Deep Cut	ผลเฉลยที่ได้จากการให้คนตรวจสอบ
ความโสดไม่ใช่เนื้อคู่	ความ|โสด|ไม่|ใช่|เนื้อคู่	ความ|โสด|ไม่|ใช่|เนื้อคู่	ความ|โสด|ไม่|ใช่|เนื้อคู่
อย่าถามได้ไหมรู้สึกยังไง	อย่า|ถาม|ได้|ไหม| |รู้สึก|ยังไง	อย่า|ถาม|ได้|ไหม| |รู้สึก|ยัง|ไง	อย่า|ถาม|ได้|ไหม| |รู้สึก|ยังไง
สวัสดีครับ โดนตามตัวมาเล่น ลายมือไม่สวยเลย พยายามเขียนให้อ่านออกแล้วนะเนี้ย	สวัสดี|ครับ| |โดน|ตามตัว|มา|เล่น| |ลายมือ|ไม่|สวย|เลย| |พยายาม|เขียน|ให้|อ่าน|ออก|แล้ว|นะ|เนี้ย	สวัสดี|ครับ| |โดน|ตาม|ตัว|มา|เล่น| |ลายมือ|ไม่|สวย|เลย| |พยายาม|เขียน|ให้|อ่าน|ออก|แล้ว|นะ|เนี้ย	สวัสดี|ครับ| |โดน|ตาม|ตัว|มา|เล่น| |ลายมือ|ไม่|สวย|เลย| |พยายาม|เขียน|ให้|อ่าน|ออก|แล้ว|นะ|เนี้ย
การที่คิดถึงแล้วต้องรอเวลาที่จะได้เจอกัน ยังทรมานไม่เท่ากับการที่คิดถึงมากแค่ไหน ก็ไม่มีทางได้เจออีกแล้ว	การ|ที่|คิดถึง|แล้ว|ต้อง|รอเวลา|ที่จะ|ได้|เจอกัน| |ยัง|ทรมาน|ไม่|เท่ากับ|การ|ที่|คิดถึง|มาก|แค่|ไหน| |ก็|ไม่|มี|ทางได้|เจอ|อีกแล้ว	การ|ที่|คิด|ถึง|แล้ว|ต้อง|รอ|เวลา|ที่จะ|ได้|เจอ|กัน| |ยัง|ทรมาน|ไม่|เท่า|กับ|การ|ที่|คิด|ถึง|มาก|แค่|ไหน| |ก็|ไม่|มี|ทาง|ได้|เจอ|อีก|แล้ว	การ|ที่|คิดถึง|แล้ว|ต้อง|รอ|เวลา|ที่|จะ|ได้|เจอ|กัน| |ยัง|ทรมาน|ไม่|เท่า|กับ|การ|ที่|คิดถึง|มาก|แค่|ไหน| |ก็|ไม่|มี|ทาง|ได้|เจอ|อีก|แล้ว
ที่ถ่ายทอดมาจากบรรพบุรุษเดียวกันตามทฤษฎีวิวัฒนาการของสัตว์	ที่|ถ่ายทอด|มาจาก|บรรพบุรุษ|เดียวกัน| |ตาม|ทฤษฎี|วิวัฒนาการ|ของ|สัตว์	ที่|ถ่ายทอด|มาจาก|บรรพบุรุษ|เดียว|กัน| |ตาม|ทฤษฎี|วิวัฒนาการ|ของ|สัตว์	ที่|ถ่ายทอด|มา|จาก|บรรพบุรุษ|เดียว|กัน| |ตาม|ทฤษฎี|วิวัฒนาการ|ของ|สัตว์

จากตัวอย่างประโยคที่มีการตัดคำข้างต้น โดยแบ่งการทดสอบหาประสิทธิภาพของโปรแกรมจากการคำนวณหาค่าความถูกต้องดังตารางที่ 3.7

ตารางที่ 3.7 ตัวอย่างการคิดคำนวณการตัดคำจากตารางที่ 3.6

วิธีการตัดคำ	จำนวนคำที่โปรแกรมตัด	ผลการทดลอง				
		TP	TN	FP	FN	ความถูกต้อง
ตัดคำโดย Newmm	71	62	0	9	0	$\frac{62+9}{62+0+9+0} \times 100 = 87.32\%$
ตัดคำโดย Deep Cut	81	73	0	8	0	$\frac{73+0}{73+0+8+0} \times 100 = 90.12\%$
ผลเฉลยจากการให้คนตรวจสอบ	80	80	0	0	0	$\frac{80+0}{80+0+0+0} \times 100 = 100\%$

ตารางที่ 3.8 ผลลัพธ์การวัดประสิทธิภาพ

วิธีการตัดคำ	Precision	Recall	F1
Newmm	0.87	1	0.932
Deep Cut	0.9	1	0.946

3.3.2 การประเมินประสิทธิภาพในการแก้ไขข้อความ

3.3.2.1 ประสิทธิภาพในการตรวจสอบคำผิด

ในการการประเมินประสิทธิภาพในการตรวจสอบคำผิด จะใช้ประโยคที่มีคำผิดปะปนอยู่ และนำผลลัพธ์ของการตัดคำ มาเปรียบเทียบกับผลเฉลยที่ได้จากการให้คนตรวจสอบ จากนั้นจะประเมินประสิทธิภาพ [18] ตามเงื่อนไขต่อไปนี้

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

รูปภาพประกอบที่ 3.22 Confusion Matrix

โดยกำหนดให้

True Positive (TP) = สิ่งที่ทำนาย ตรงกับสิ่งที่เกิดขึ้นจริง ในกรณี ทำนายว่าจริง และสิ่งที่เกิดขึ้น ก็คือ จริง

True Negative (TN) = สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณี ทำนายว่า ไม่จริง และสิ่งที่เกิดขึ้น ก็คือ ไม่จริง

False Positive (FP) = สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง

False Negative (FN) = สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้น คือ จริง

เราจะใช้ Confusion Matrix มาคำนวณ การประเมินประสิทธิภาพของการทำนายด้วย Model ดังนี้ โดยเลือกใช้วิธี Accuracy คือ ความถูกต้องที่เราทำนายได้ตรงกับสิ่งที่เกิดขึ้นจริง ซึ่งสามารถแสดงตัวอย่างการวัดประสิทธิภาพได้ดังนี้

$$\text{Accuracy} = (TPs + TNs) / (TPs + TNs + FPs + FNs) \quad (14)$$

3.3.3 ประสิทธิภาพในการแก้ไขคำผิด [19] โดยผู้จัดทำได้ประเมินประสิทธิภาพของผลลัพธ์ที่ได้จากการตรวจสอบความถูกต้องของข้อความ ด้วย Word Error Rate (WER) [27] คือ อัตราความผิดพลาดของคำจากการทดสอบ การคำนวณ WER ต้องมีข้อความที่ถูกต้อง และผลลัพธ์จากกระบวนการทำนาย ซึ่งจะเปรียบเทียบระหว่างสองตัวอักษรหรือคำแต่ละตัว จำนวนคำที่ถูกต้องและคำที่ไม่ถูกต้องจะนับได้จากจำนวนของการแก้ไข การลบ และการเพิ่ม ที่เกิดขึ้น การคำนวณ WER จะใช้สูตรดังนี้:

$$WER = (S + D + I) / N \quad (15)$$

โดยที่

S คือ จำนวนของการแก้ไข (Substitutions) ที่เกิดขึ้น

D คือ จำนวนของการลบ (Deletions) ที่เกิดขึ้น

I คือ จำนวนของการเพิ่ม (Insertions) ที่เกิดขึ้น

N คือ จำนวนของคำทั้งหมดในข้อความที่ถูกต้อง (Reference)

ค่า WER จะอยู่ในรูปของเปอร์เซ็นต์ ซึ่งค่าที่ต่ำกว่ายิ่งดีและแสดงให้เห็นถึงความคลาดเคลื่อนของกระบวนการแก้ไขคำ ผู้จัดทำได้ใช้ Library ชื่อ Word Error Rate โดยจะใช้ภาษา Python ในการพัฒนา ซึ่งตัวอย่างนี้ทำในโปรแกรม Colaboratory

```
[1] !pip install jiwer

from jiwer import wer

if __name__ == "__main__":
    reference = "ไข่ เจียว หมู สับ" #gt
    hypothesis = "ไข่ เจียว หมู สุบ" #trans
    print(wer(reference, hypothesis))

0.25
```

รูปภาพประกอบที่ 3.23 ตัวอย่างโค้ดการตรวจสอบกับข้อความ

โดยวิธีการใช้งานมีดังต่อไปนี้

1.ติดตั้งไลบรารี่ด้วยคำสั่ง

คำสั่ง : pip install jiwer

2.เรียกใช้ from jiwer import wer

3.สามารถเขียนโค้ดได้ดังภาพประกอบที่ 3.23