

# Tokenization

## 2110572: NLP SYS

**Peerapon Vateekul & Ekapol Chuangsawanich**

Department of Computer Engineering,  
Faculty of Engineering, Chulalongkorn University

Based on Aj.Ekapol's slide in 2017

# Outlines

## LexTo เล็กซ์ โต Thai Lexeme Tokenizer

The need for segmentation

Thai Tokenization

1) Longest Matching

2) Maximal Matching

- Dynamic Programming
- LexTo

ตามหามาสี

 Clear

ตามหา | มาสี |

[คำที่ไม่รู้จัก](#) | [คำรู้จัก](#) | [คำจำกัดความหมาย](#) | [คำภาษาอังกฤษหรือตัวเลข](#) | [อักษรพิเศษ](#)

# The need for segmentation

- Text as a stream of characters



- We need a way to understand the meaning of text

- Break into words (assign meaning to word) ←
- Break into sentences (put word meanings back to sentence meaning)

Word Tokenization

# Tokenization - Thai

- Thai has no space between words
- Thai has no clear sentence boundaries

- เริ่มจากชั้นวนของสังคมรวมแล้ว สามพันธุ์การค้า ทำการปิดล้อม-รกรานดาวนานุ ส่งผลต่อมาขยายเป็นสังคมโคลนอันมีฝ่ายแบ่งแยกดินแดนเป็นผู้ชักใยสังคม หมายโคนล้มฝ่ายสาธารณรัฐ นานาไปกับเรื่องราวด้านการของของเด็กน้อยคนหนึ่งผู้มีพลังสกิดแรงมาก ชาวดาวทะเลขรายทางทวีปในนาม "อนาคต สภាឯวออลค์เกอร์" ผู้ถูกคาดการณ์ว่าคือผู้ถูกเลือกในต้นนาของเจ้าได หลังจากสังคมรวมยุทธการดาวนานุ อนาคต ก็ไดรับฝึกฝนในวิถีเจ้าได โดย อ.เจได "โอบีวัน" แต่ เมื่ออนาคตได้เป็นหุ่นก็ต้นแรกกฎเจ้าไดโดยแอบมีความสัมพันธ์ซึ่งสาวลับๆกับ ราชินี "อมีดาลา" แห่งดาวนานุ จนเรอตั้งครรภ์ลูกแฟด ... และอนาคตก็ค่อยๆถูกกลืนเข้าสู่ด้านมืดของพลังจนกลายเป็นชีวลอร์ด ได้ฉะยา "ดาร์ธ เวเดอร์" ภายใต้การโน้มน้าวขึ้นนำของ ชีวลอร์ดลีกกลับนาม "ดาร์ธ ชีเดียล" ซึ่งเผยแพร่ในตอนท้ายว่า ชีเดียล "ไม่ใช่ใครที่ไหน แต่คือท่าน "พลพาร์ทิน" สมุหนายกผู้นำสูงสุดของฝ่ายสาธารณรัฐเสียเอง และแท้จริงก็ยัง เป็นผู้นำลับชักใยฝ่ายแบ่งแยกดินแดนด้วยอีกด้วย หาก ... สังคมรวมลงที่ฝ่ายสาธารณรัฐพ่ายแพ้ล้มลาย อมีดาลา ก็ตายหลังคลอดลูกแฟด ทั้งเหล่า อัศวินเจ้าไดก็ถูกฆ่ากวาดล้างสิ้นแบบไม่ทันตั้งตัว เหลือแต่ อ.โอบีวัน กับ อ.โยดา ต้องลี้ภัยหลบหนีช่อนด้วย โดยลูกแฟดของอนาคตได้ถูกส่งไปสู่ที่หลบซ่อนลับเช่นกัน ... และแล้ว พลพาร์ทิน ก็กินรวบทั้งกระดาน เปเลี่ยนการปกครองจาก ระบบสาธารณรัฐเดิมไปเป็น ระบบเผด็จการจักรพรรดิชีวแทน ตั้งตนเป็นจักรพรรดิปกครองแก่แล้วซึ่งทั้งปวง โดยมี ดาร์ธ เวเดอร์ เป็นขุนพลชีวลอร์ดเคียงข้างนับแต่นั้นมา

# Tokenization - Thai

~ พง .

## Social media text

#สตอรี่ของโน้ມ 😊🐙 #Days23ofMobile 🌈 ...23 วันแล้วนะเจ้าโน้ມ พีกกำลังคิดถึงหนูอยู่เลย แหนนะ เมื่อวานหายเลย  
นะ 😳 พีกว่าหายไปไหนแอบหนีไปเล่นลงบอร์ดนี่เอง เล่นรัวๆนะพีเป็นห่วง เดวล้มແກນไม่มีตະหลາມคอยเล่น<sup>1</sup>  
เป็นเพื่อนอีก 😂 พีจะบอกว่า "หนูอย่าลืมลงชือเลือกตั้งนะ" เดียวพากพีเข้ากันไม่อOKEN 5555 ฝากไว้กันลืม ยิ่งเด้อ<sup>2</sup>  
ๆอยู่ อีกอย่างๆหนูต้องกลับมาแนะนำพากเรารอหนูอยู่ 😊 พีนีเตรียมรอโหวตให้หนูเต็มที่เลย 😊 ไปเรียนเป็นไปบ้าง  
ยังเหงาอยู่มั้ย แต่พีว่าคงไม่แล้วหละมั้ง วันนี้ขึ้นสเตจอีกแล้วสินะ เหนื่อยมั้ยค่ะ แต่คงสนุกมากกว่าอยู่แล้วเนอะ 😃  
แค่ได้เห็นรอยยิ้มของหนูพีกสบ้ายใจและ แต่เอ๊ะ เมื่อวานใครบ่นอยากกลับบ้านอีกแล้วนะ อดดู the toy เลยอะดี  
😂 ห่วยๆๆ น่าสงสาร 555 โอกาสหน้ายังมีนะ ตั้งใจทำงานนะค่ะ เจ้าหลามแฟนหนูก็อด ไม่ได้อดคนเดียว  
อะหน่อนะ 555 🥺 ดูมีความสุขจังน้าเจ้าตัวเล็ก 😊❤️ คิดถึงนะ เดี่ยวก็ได้ 2shot กันแล้ว พีโคตรตีนเต้นเลย  
ตีนเต้นกว่าไปลับมือเยอะ 😳 คิดทำไม่ออกเลย มีท่าไหนแนะนำมั้ย ช่วยพีหน่อยยียียย 😂... #Mobilebnk48 #ตู้  
เพลงโมบิล #ชาวหรីញុយอดตู้ #MOTA09

# Tokenization - Thai

- Many word boundaries depends on **the context (meaning)**

تا กลม vs ตาก ล�

คณะกรรมการตรวจสอบคณาน

- Even amongst Thais the definition of word boundary is **unclear** .

- Needs a consensus when designing a corpus
- Sometimes depends on the application
  - Linguist vs machine learning concerns

# Dictionary-based vs Machine-learning-based

- 1) Dictionary-based
  - Longest matching
  - Maximal matching *Popular*
- 2) Machine-learning-based

*greedy.*

# Dictionary-based word segmentation

- Perform by scanning a string and match each substring against words from a **dictionary**.  
(No dataset needed, just prepare a dictionary!)
- However, there is ambiguity in matching.(There are many many ways to match)

ป้ายกลับรถ

- So, **matching methods** are developed:
  1. Longest matching
  2. Maximal matching

# 1) Longest Matching

- Scan a sentence from left to right
- Keep finding a word from the starting point, until no word matched (longest), then move to the next point
  - Backtrack if current segmentation leads to an un-segmentable chunk

greedy

- ป้ายกลับรถ Start scanning with “ป” as the starting point
- ป้ายกลับรถ Keep scanning ...
- ป้าย/กลับรถ No more words start with “ป้าย”, move to the next point
- ...
- ป้าย/กลับ/รถ

## 2) Maximal Matching

- Generate all possible segmentations
- Select the segmentations with the fewest words

*Dynamic programming.*

ไป หาม เหลี่

ไป | หาม | เหลี่      4 words

ไป | หา | มเหลี่      3 words       maximal matched

# What if?

- What if there are more than one segmentation with the fewest words?
- Other heuristics are applied, for example
  - Language model score

> Longest Matching

> Language Modeling

คุณ | อกร | กช

คุณ | อา | กรกช

n-word min. prob.  
↓  
(Lang. model)

Homework.

## Maximal matching

- Maximal matching can be done using **dynamic programming**.
- Let  $d(i,j)$  be the function which returns number of the fewest word possible with the **last word starts with  $i^{\text{th}}$  character (row) and ends with  $j^{\text{th}}$  character (column)**. It can be defined as:

$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ 1 + \min_{k=1 \dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise.} \end{cases}$$

when  $c[i..j]$  is a string of word in the sentence (assume it is started at index 1) and the base case is  $d(1,1) = 1$ .

# Maximal matching: Initialization (1<sup>st</sup> row)

- Maximal matching can be done using **dynamic programming**.
- Let  $d(i,j)$  be the function which returns number of the fewest word possible with the **last word starts with  $i^{\text{th}}$  character (row) and ends with  $j^{\text{th}}$  character (column)**. It can be defined as:

$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ 1 + \min_{k=1 \dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise.} \end{cases}$$

when  $c[i..j]$  is a string of word in the sentence (assume it is started at index 1) and the base case is  $d(1,1) = 1$ .

# Maximal matching: Find a word in dictionary

- Maximal matching can be done using **dynamic programming**.
- Let  $d(i,j)$  be the function which returns number of the fewest word possible with the **last word starts with  $i^{\text{th}}$  character (row) and ends with  $j^{\text{th}}$  character (column)**. It can be defined as:

$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ 1 + \min_{k=1\dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise.} \end{cases}$$

**If last word is a word**

when  $c[i..j]$  is a string of word in the sentence (assume it is started at index 1) and the base case is  $d(1,1) = 1$ .

# Maximal matching: Check all possible segmentations before the final word

- Maximal matching can be done using **dynamic programming**.
- Let  $d(i,j)$  be the function which returns number of the fewest word possible with the **last word starts with  $i^{\text{th}}$  character (row) and ends with  $j^{\text{th}}$  character (column)**. It can be defined as:

$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ 1 + \min_{k=1 \dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise} \end{cases}$$

Check all possible segmentations before the final word  
Check the whole column in the previous row

when  $c[i..j]$  is a string of word in the sentence (assume it is started at index 1) and the base case is  $d(1,1) = 1$ .

# Maximal matching: The final word

- Maximal matching can be done using **dynamic programming**.
- Let  $d(i,j)$  be the function which returns number of the fewest word possible with the **last word starts with  $i^{\text{th}}$  character (row) and ends with  $j^{\text{th}}$  character (column)**. It can be defined as:

$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ \boxed{1} + \min_{k=1\dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise. } \textcolor{red}{\text{The final word}} \end{cases}$$

when  $c[i..j]$  is a string of word in the sentence (assume it is started at index 1) and the base case is  $d(1,1) = 1$ .

# Dynamic programming example (1)

ຄູນຄະນະ.

0, 9

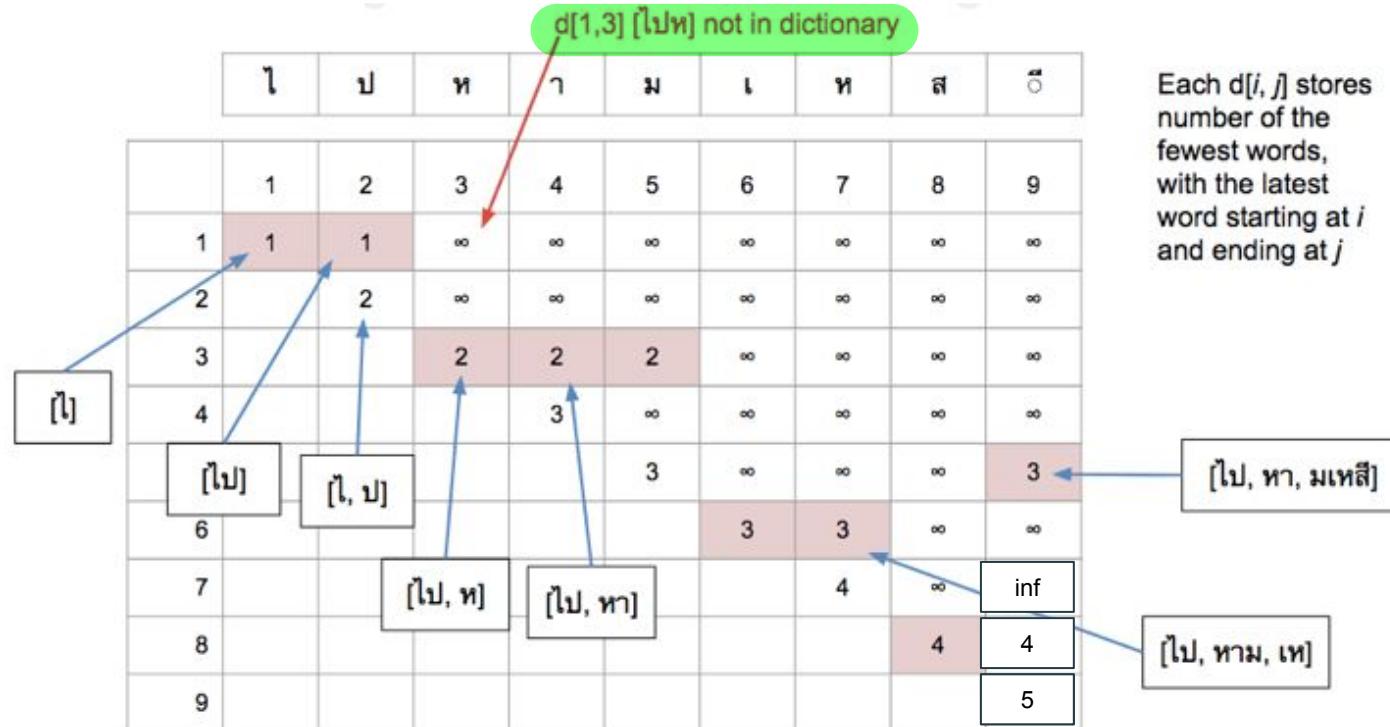
0 =

Each  $d[i, j]$  stores  
number of the  
fewest words,  
with the latest  
word starting at  $i$   
and ending at  $j$

	ໄ	ປ	ໜ	ກ	ມ	ເ	ຫ	ສ	ຣ
1	1	2	3	∞	∞	∞	∞	∞	∞
2	1	1	∞	∞	∞	∞	∞	∞	∞
3	2	2	2	2	∞	∞	∞	∞	∞
4	3	3	3	3	∞	∞	∞	∞	∞
5	[ໄປ] [ໄປ, ປ]	[ໄປ, ຜ]	[ໄປ, ຢ]	[ໄປ, ກ]	[ໄປ, ມ]	[ໄປ, ແ]	[ໄປ, ຫ]	[ໄປ, ສ]	[ໄປ, ລ]
6									
7									
8									
9									

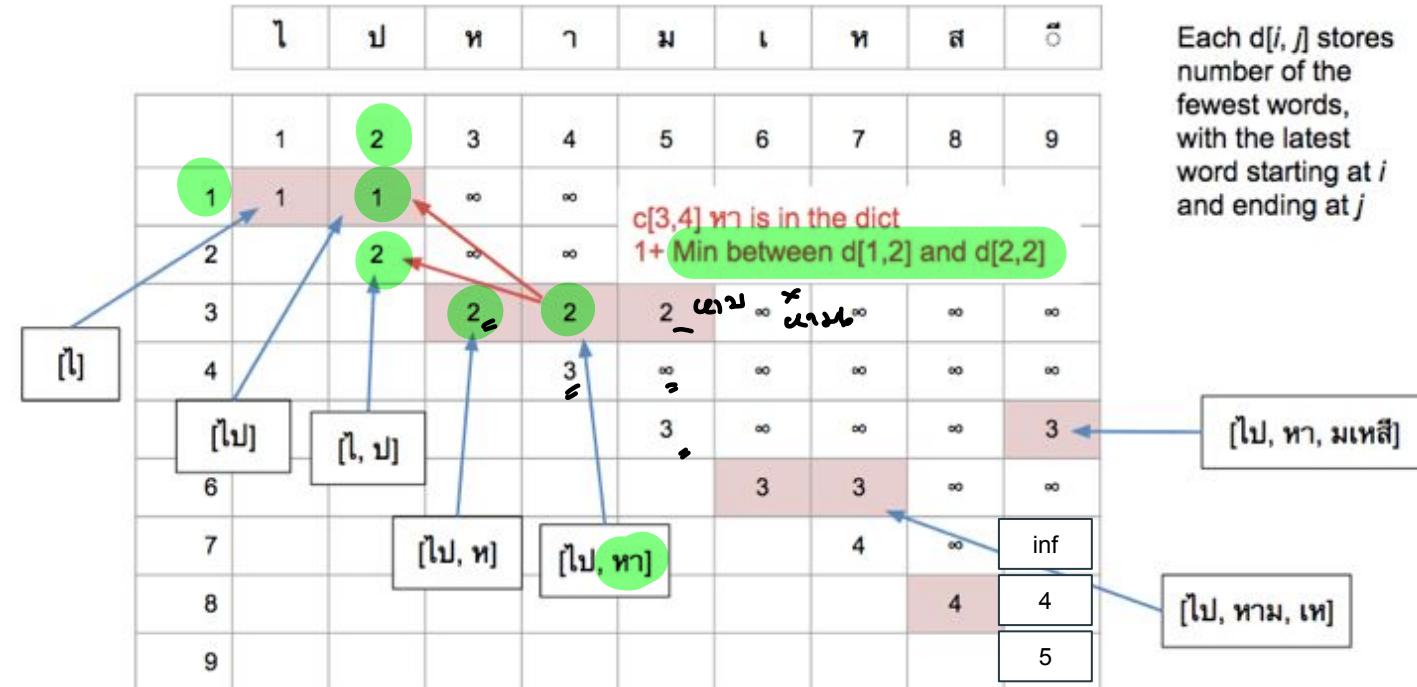
Dict: ໄ, ປ, ຜ, ກ, ມ, ແ, ຫ, ສ, ລ, ໄປ, ທາ, ມເຂີນ, ແທ, ຢາ, ສີ, ມເຂີນ

# Dynamic programming example (2)



Dict: ໄ, ປ, ຫ, ຢ, ມ, ເ, ສ, ລ, ໄປ, ຫາ, ໝາມ, ເຫ, ສື, ມເທສີ

# Dynamic programming example (3)



Dict: ໄ, ປ, ທ, ມ, ຢ, ສ, ດ, ໃປ, ພ, ທ, ມ, ຢ, ສ, ມເຫັດ

# Dynamic programming example (4)

Each  $d[i, j]$  stores number of the fewest words, with the latest word starting at  $i$  and ending at  $j$

ໃ	ປ	ຫ	າ	ນ	ເ	ໜ	ສ	ີ	
1	1	2	3	4	5	6	7	8	9
1	1	1	$\infty$						
2		2	$\infty$						
3			2	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
4				3	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
6	[ໃປ]	[ໃ, ປ]			3	$\infty$	$\infty$	$\infty$	3
7			[ໄປ, ຫ]	[ໄປ, ມ]		3	3	$\infty$	inf
8					4	$\infty$	4	4	5
9									

C[4,4] 'ກ' is in the dict  
1+ Min between d[1,3], d[2,3], and d[2,3]

Arrows show the path from the start index (1) to the end index (4). The path starts at index 1, goes to index 2, then to index 3, and finally to index 4. The value at d[4,4] is 3, which is highlighted in red.

Labels below the table indicate the sequences found:

- [ໃປ]
- [ໃ, ປ]
- [ໄປ, ຫ]
- [ໄປ, ມ]
- [ໄປ, ກ]
- [ໄປ, ມເກສີ]
- [ໄປ, ມາມ, ເກ]

Dict: ໄ, ປ, ຫ, ກ, ມ, ແ, ສ, ຕ, ໄປ, ມາມ, ເກ, ສີ, ມເກສີ

# Dynamic programming example (5): Backtracking



Each  $d[i, j]$  stores number of the fewest words, with the latest word starting at  $i$  and ending at  $j$

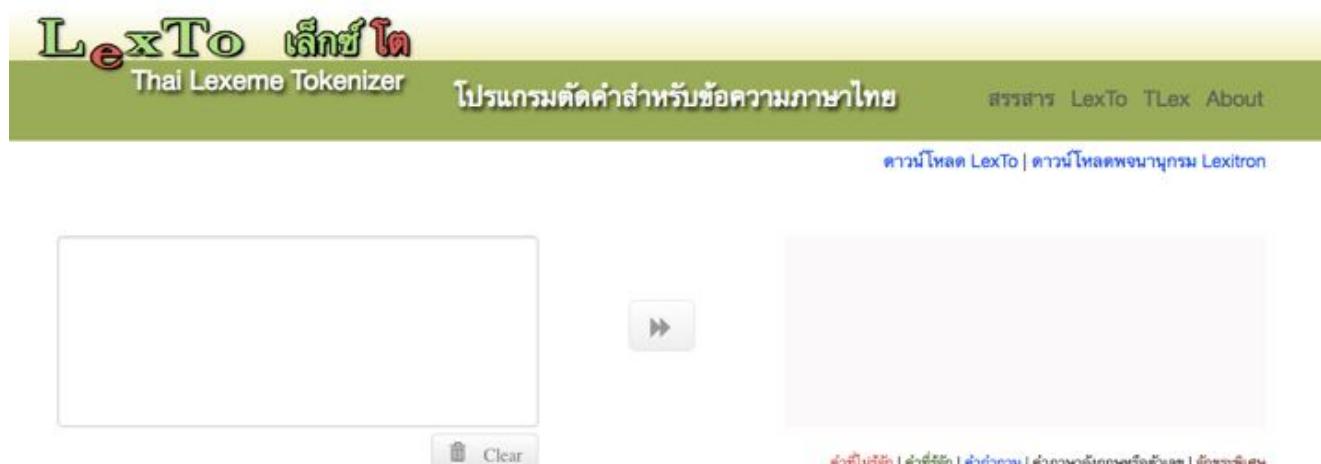
$\min \text{ word.}$

backtracking

Dict: ໄ, ປ, ທ, ກ, ມ, ເ, ສ, ຕ, ໄປ, ທາ, ທາມ, ໜ, ສີ, ມເຫສີ

# LexTo

- <http://www.sansarn.com/lext0/>
- Dictionary-based longest matching



Search docs

## NOTES

Command Line

Getting Started

Installation

From PyThaiNLP 1.7 to PyThaiNLP 2.0

## PACKAGE REFERENCE:

pythainlp.corpus

pythainlp.soundex

pythainlp.spell

pythainlp.summarize

pythainlp.tag

## pythainlp.tokenize

Modules

## Tokenization Engines

pythainlp.tools

pythainlp.transliterate

pythainlp.ulmfit

pythainlp.util

`pythainlp.tokenize.word_tokenize(text: str, custom_dict: marisa_trie.Trie = None, engine: str = 'newmm', keep_whitespace: bool = True) → List[str]` [source]

This function tokenizes running text into words.

## Parameters

- `text (str)` – text to be tokenized
- `engine (str)` – name of the tokenizer to be used
- `custom_dict (marisa_trie.Trie)` – marisa dictionary trie
- `keep_whitespace (bool)` – True to keep whitespaces, a common mark for end of phrase in Thai. Otherwise, whitespaces are omitted.

## Returns

list of words

## Return type

`list[str]`

## Options for engine

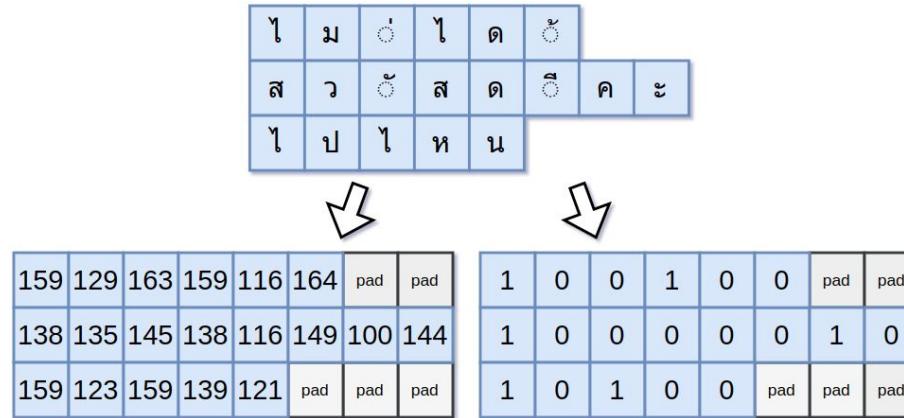
- `newmm` (default) - dictionary-based, Maximum Matching + Thai Character Cluster
- `longest` - dictionary-based, Longest Matching
- `deepcut` - wrapper for deepcut, language-model-based DL
- `icu` - wrapper for ICU (International Components for Unicode, using PyICU), dictionary-based
- `ulmfit` - for thai2fit

ສັນນິມາດ  
as 1 word  
ນອກຫຼຸດ.

ເພື່ອຕົວຢ່າງ  
Dynamic ດຳວັດ.

# Neural-based Tokenizer

## Step1: Preprocess

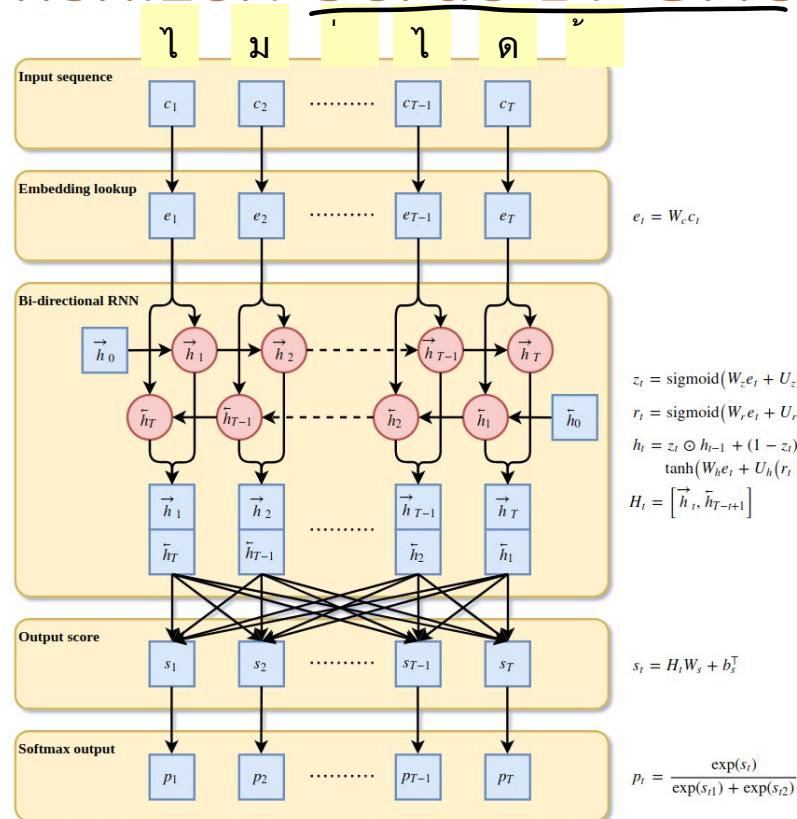


Reference: [http://web.archive.org/web/20181005101442/https://sertiscorp.com/thai-word-segmentation-with-bi-directional\\_rnn/](http://web.archive.org/web/20181005101442/https://sertiscorp.com/thai-word-segmentation-with-bi-directional_rnn/)

# Neural-based Tokenizer: Sertis Bi-GRU

## Step2: Model

next week.



Reference: [http://web.archive.org/web/20181005101442/https://corpus.thierrybenoain.com-with-sectionnal\\_rnn/](http://web.archive.org/web/20181005101442/https://corpus.thierrybenoain.com-with-sectionnal_rnn/)

1 0 0 1 0 0

0	1	2	3	4	5	6	7	8	9
0	1	1	x	x	x	x	x	x	x
1	-	2	x	x	x	x	x	x	x
2	-	-	2	(2)	2	x	x	x	x
3	-	-	-	3	x	x	x	x	x
4	-	-	-	-	3	x	x	x	(3) x
5	-	-	-	-	-	3	3	x	x
6	-	-	-	-	-	-	4	x	x
7	-	-	-	-	-	-	-	4	4 x
8	-	-	-	-	-	-	-	-	5 x
9	-	-	-	-	-	-	-	-	(4)

$r=9, C=9 / \text{old}=9$

$C=8; - + (8, 9)$   
 $r=4$

$r=4, C=8 / \text{old}=8$

$C=7, x$

$C=6, x$

$C=5, x$

$C=4, 3$

$C=3, -$

$+ (3, 8)$   
 $r=2$

$r=2,$