

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

- เริ่มจากชั้นวนของสังคมรากสูด สามพันธ์การค้า ทำการปิดล้อม-รกรานดาวนานุ ส่งผลต่อมาขยายเป็นสังคมโคลนอันมีฝ่ายแบ่งแยกดินแดนเป็นผู้ชักใยสังคม หมายโคนล้มฝ่ายสาธารณรัฐ นานาไปกับเรื่องราวพัฒนาการของของเด็กน้อยคนหนึ่งผู้มีพลังสกิดแรงมาก ชาวดาวทะเลขรายทางอินนาม "อนาคต สภាឯวอල์คเกอร์" ผู้ถูกคาดการณ์ว่าคือผู้ถูกเลือกในต้นนานของเจ้าได หลังจากสังคมรุกของการดาวนานุ อนาคต ก็ไดรับฝึกฝนในวิถีเจ้าได โดย อ.เจได "โอบีวัน" แต่ เมื่ออนาคตไดเป็นหุ่นก็ต้นแท้ก็ไม่สามารถมีความสัมพันธ์ซึ่งกันและกัน ราชินี "อมีดาลา" แห่งดาวนานุ จนเรอตั้งครรภ์ลูกแฟด ... และอนาคตก็ค่อยๆถูกกลืนเข้าสู่ด้านมืดของพลังจนกลายเป็นชีวโลร์ด ได้ฉะยา "ดาร์ธ เวเดอร์" ภายใต้การโน้มน้าวขึ้นนำของ ชีวโลร์ดลีกกลับนาม "ดาร์ธ ชีเดียล" ซึ่งเผยแพร่ในตอนท้ายว่า ชีเดียล "ไม่ใช่ใครที่ไหน แต่คือท่าน "พลพาร์ทิน" สมุหนายกผู้นำสูงสุดของฝ่ายสาธารณรัฐเสียเอง และแท้จริงก็ยัง เป็นผู้นำลับชักใยฝ่ายแบ่งแยกดินแดนด้วยอีกด้วย หาก ... สังคมจะลงที่ฝ่ายสาธารณรัฐพ่ายแพ้ล้มสถาบัน อมีดาลา ก็ตายหลังคลอดลูกแฟด ทั้งเหล่า อัศวินเจ้าไดก็ถูกฆ่ากวาดล้างสิ้นแบบไม่ทันตั้งตัว เหลือแต่ อ.โอบีวัน กับ อ.โยดา ต้องลี้ภัยหลบหนีช่อนด้า โดยลูกแฟดของอนาคตไดถูกส่งไปสู่ที่หลบซ่อนลับเช่นกัน ... และแล้ว พลพาร์ทิน ก็กินรวบทั้งกระดาน เปเลี่ยนการปกครองจาก ระบบสาธารณรัฐเดิมไปเป็น ระบบเผด็จการจักรพรรดิชีวแทน ตั้งตนเป็นจักรพรรดิปกครองแก่แล้วซึ่งทั้งปวง โดยมี ดาร์ธ เวเดอร์ เป็นขุนพลชีวโลร์ดเคียงข้างนับแต่นั้นมา

Tokenization - Thai

Social media text

#สตอรี่ของโน้ມ 😊🐙 #Days23ofMobile 🌈 ...23 วันแล้วนะเจ้าโน้ມ พีกกำลังคิดถึงหนูอยู่เลย แหนง เมื่อวานหายเลย
นะ 😊 พีกว่าหายไปไหนแอบหนีไปเล่นลงบอร์ดนี่เอง เล่นรัวๆนะพีเป็นห่วง เดวล้มແກນไม่มีตະหลາມคอยเล่น¹
เป็นเพื่อนอีก 😂 พีจะบอกว่า "หนูอย่าลืมลงชือเลือกตั้งนะ" เดียวพากพีเข้ากันไม่อOKEN 5555 ฝากไว้กันลืม ยิ่งเด้อ²
ๆอยู่ อีกอย่างๆหนูต้องกลับมานะพากเรารอหนูอยู่ 😊 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 ช่วยพีหน่อยยยยย 😂... #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
- 2) Machine-learning-based

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

- 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**

ไป | หาม | เห | สี 4 words

ไป | หา | มเหสี 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

คุณ | อกร | กช

> Language Modeling

คุณ | อ | กรกช

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^{th} character (row) and ends with j^{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 (1st 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^{th} character (row) and ends with j^{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^{th} character (row) and ends with j^{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^{th} character (row) and ends with j^{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^{th} character (row) and ends with j^{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 $d(1,1) = 1$.

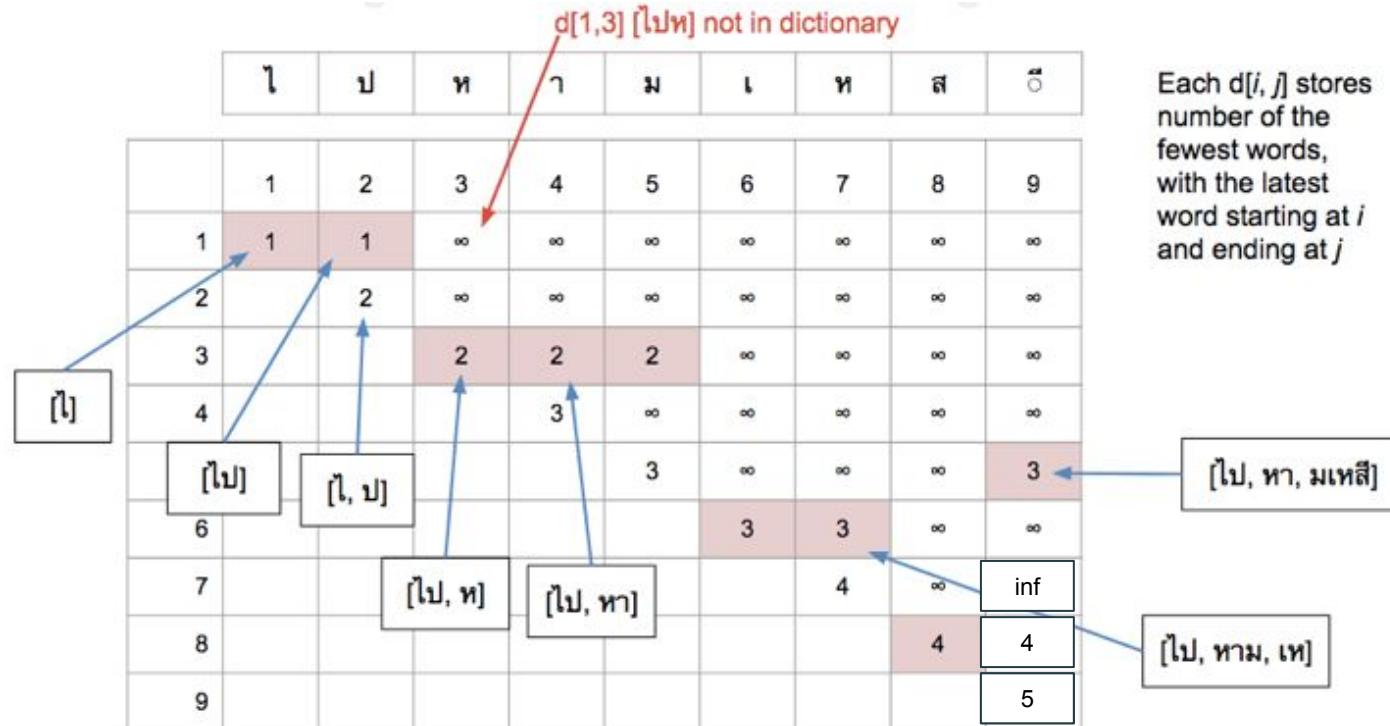
Dynamic programming example (1)

	ໄ	ປ	ຫ	າ	ນ	ເ	ທ	ສ	ີ
	1	2	3	4	5	6	7	8	9
1	1	1	∞	∞	∞	∞	∞	∞	∞
2	∞	2	∞	∞	∞	∞	∞	∞	∞
3	2	2	2	∞	∞	∞	∞	∞	∞
4	3	∞	∞	∞	∞	∞	∞	∞	∞
6	[ໄປ]	[ໄ, ປ]	[ໄ, ປ, ຫ]	[ໄ, ປ, ຫ, ຄ]	[ໄ, ປ, ຫ, ຄ, ຏ]	[ໄ, ປ, ຫ, ຄ, ຏ, ສ]	[ໄ, ປ, ຫ, ຄ, ຏ, ສ, ຕ]	[ໄ, ປ, ຫ, ຄ, ຏ, ສ, ຕ, ສ]	[ໄ, ປ, ຫ, ຄ, ຏ, ສ, ຕ, ສ, ອ]
7									inf
8								4	4
9									5

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

Dict: ໄ, ປ, ຫ, ຄ, ນ, ເ, ສ, ຕ, ໄປ, ກາ, ມາ, ແ, ສີ, ມເລສີ

Dynamic programming example (2)



Dict: ໄ, ປ, ທ, ລ, ນ, ເ, ສ, ກ, ໄປ, ທາ, ທາມ, ແ, ສີ, ມເສດ

Dynamic programming example (3)

Thai characters above the grid:

ໄ	ປ	ຫ	ກ	ນ	ຕ	ຫ	ສ	ດ
---	---	---	---	---	---	---	---	---

Input character: ດ

Dictionary (Dict): ໄ, ປ, ຫ, ກ, ນ, ຕ, ສ, ດ, ໄປ, ຫາ, ມໍາ, ແ, ສີ, ມເຂີສ

DP Grid (d[i, j]):

	1	2	3	4	5	6	7	8	9
1	1	1	∞	∞					
2	2	2	∞	∞					
3	3	2	2	2	∞	∞	∞	∞	∞
4	4	3	3	3	∞	∞	∞	∞	∞
6	[ໄປ]	[ໄ, ປ]	[ໄປ, ຫ]	[ໄປ, ກ]	[ໄປ, ນ]	[ໄປ, ຫາ]	[ໄປ, ມໍາ]	[ໄປ, ແ]	[ໄປ, ສີ]
7			[ໄປ, ຫາ]	[ໄປ, ມໍາ]					[ໄປ, ມເຂີສ]
8									[ໄປ, ຫາມ, ແ]
9									[ໄປ, ຫາມ, ແ]

Annotations:

- Cell (3, 4) contains "c[3,4] ຫາ is in the dict".
- Text "1+ Min between d[1,2] and d[2,2]" is shown near the red arrow.
- Cells (4, 6), (4, 7), (4, 8), and (4, 9) are shaded pink.
- Cells (6, 7), (6, 8), and (6, 9) contain boxes with word sequences: [ໄປ, ຫາ], [ໄປ, ມໍາ], and [ໄປ, ມເຂີສ].
- Cells (7, 8) and (7, 9) contain boxes with numbers: 4 and 5 respectively.
- Cell (8, 9) contains a box with "inf".
- Blue arrows point from the input character "ດ" to the start of words in the dictionary, and from the sequence boxes back to the grid.
- Red arrows point from the sequence boxes to the grid cell (3, 4).

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

Dict: ໄ, ປ, ຫ, ກ, ນ, ຕ, ສ, ດ, ໄປ, ຫາ, ມໍາ, ແ, ສີ, ມເຂີສ

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	∞	∞	∞	∞	∞	∞	∞
2	2	1	∞	∞	∞	∞	∞	∞	∞
3	3	2	2	∞	∞	∞	∞	∞	∞
4	4	3	2	3	∞	∞	∞	∞	∞
6	[ໃປ]	[ໃ, ປ]			3	∞	∞	∞	∞
7			[ໄປ, ຫ]	[ໄປ, ມ]	3	∞	∞	∞	3
8				[ໄປ, ມເກສີ]	3	∞	∞	∞	inf
9					4	∞	4	4	5

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

Dict: ໄ, ປ, ຫ, ຢ, ມ, ແ, ສ, ຕ, ໄປ, ມເກສີ, ພາມ, ແສ, ສີ, ມເກສີ

Dynamic programming example (5): Backtracking

Dict: ໃ, ປ, ທ, ກ, ມ, ເ, ສ, ຊ, ໄປ, ພາ, ພາມເສີ, ໄປ, ພາ, ພາມ, ເຫ, ສີ, ມເລສີ

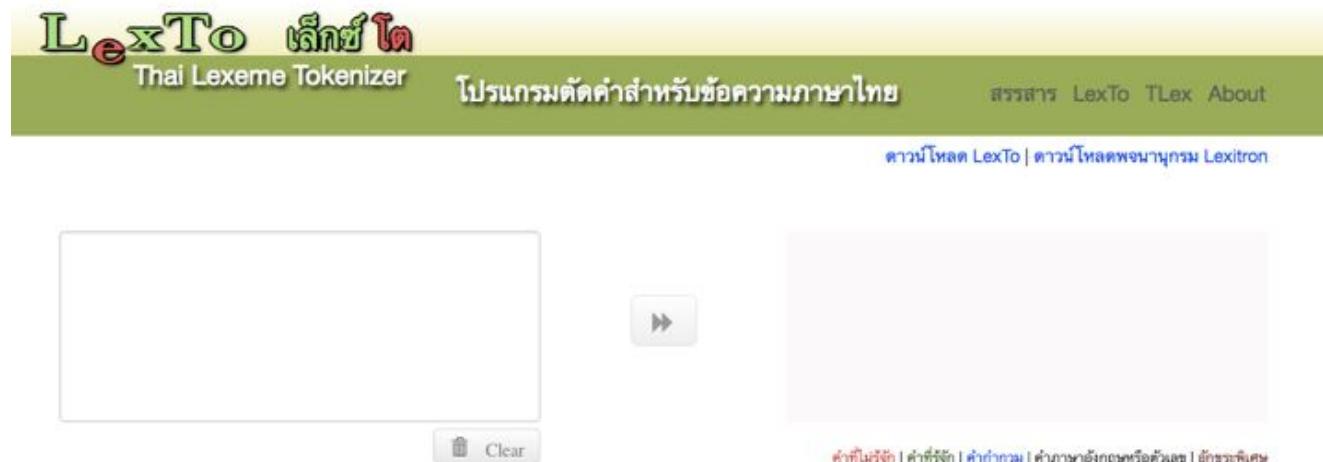
The grid shows the number of shortest word partitions for each position. The first few rows are initialized with 1 or infinity. Subsequent rows show the count increasing as more words are considered. The last row shows the final counts for the entire input string.

	ໃ	ປ	ທ	ກ	ມ	ເ	ສ	້
1	1	1	∞	∞	∞	∞	∞	∞
2	2	2	∞	∞	∞	∞	∞	∞
3	2	2	2	∞	∞	∞	∞	∞
4	3	3	3	∞	∞	∞	∞	∞
6	[ໄປ]	[ໃ, ປ]	[ໄປ, ທ]	[ໄປ, ທ, ກ]	[ໄປ, ທ, ກ, ມ]	[ໄປ, ທ, ກ, ມ, ເ]	[ໄປ, ທ, ກ, ມ, ເ, ສ]	[ໄປ, ທ, ກ, ມ, ເ, ສ, ເຫ]
7							3	inf
8							4	4
9								5

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

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

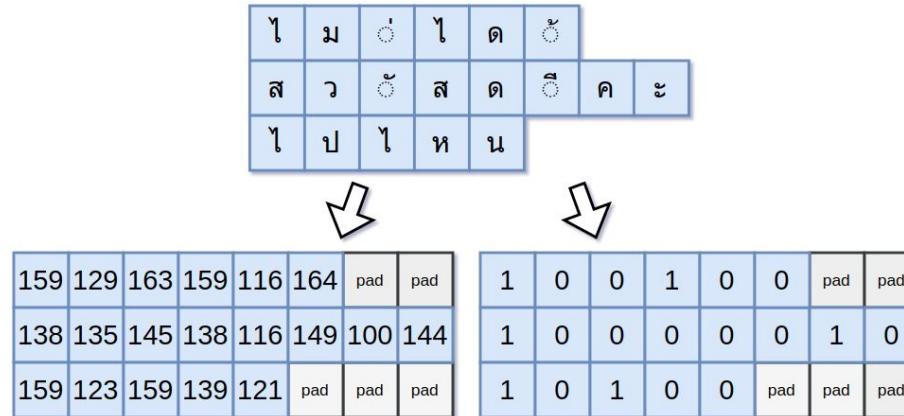
`list[str]`

Options for engine

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

Neural-based Tokenizer

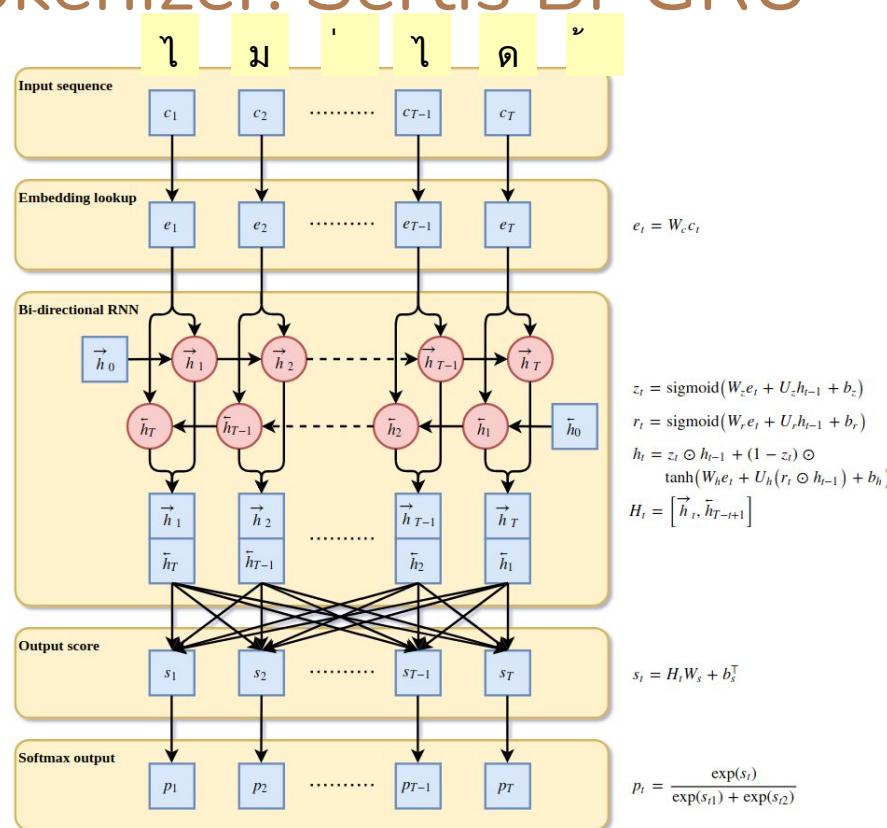
Step1: Preprocess



Reference: http://web.archive.org/web/20181005101442/https://sertiscorp.com/thai-word-segmentation-with-bi-directional_rnn/

Neural-based Tokenizer: Sertis Bi-GRU

Step2: Model



Reference: http://web.archive.org/web/20181005101442/https://corpuskit.org/doc/Tokenization-with-Recurent_RNN/

1 0 0 1 0 0