

# Tokenization

## 2110572: NLP SYS

Department of Computer Engineering,  
Faculty of Engineering, Chulalongkorn University  
**Based on Aj.Ekapol's slide in 2017**

# Outlines

The need for segmentation

Thai Tokenization

1) Longest Matching

2) Maximal Matching

- Dynamic Programming
- LexTo

**LexTo** เล็กซ์ โต  
Thai Lexeme Tokenizer

ตามหามเหสี

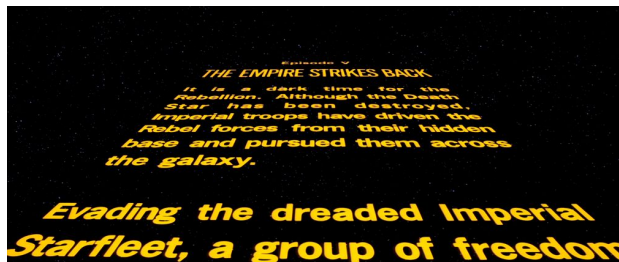
 Clear

ตามหา | มเหสี |

[คำที่ไม่รู้จัก](#) | [คำที่รู้จัก](#) | [คำกำกวม](#) | [คำภาษาอังกฤษหรือตัวเลข](#) | [อักขระพิเศษ](#)

# 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

- เริ่มจากชนวนของสงครามแรกสุด สมพันธ์การค้า ทำการปิดล้อม-ถูกรานดาวนานู ส่งผลต่อมาขยายเป็นสงครามโคลนอันมีฝ่ายแบ่งแยกดินแดนเป็นผู้ชักใยสงคราม หมายโค่นล้มฝ่ายสาธารณรัฐ ขนานไปกับเรื่องราวพัฒนาการของของเด็กน้อยคนหนึ่งผู้มีพลังศักดิ์แรงมาก ชาวดาวทะเลทรายหาทูอินนาม "อนาคิน สกายวอล์คเกอร์" ผู้ถูกคาดการณ์ว่าเป็นผู้ถูกเลือกในตำนานของเจได หลังจากสงครามยุทธการดาวนานู อนาคิน ก็ได้รับฝึกฝนในวิถีเจได โดย อ.เจได "โอปี้วัน" แต่เมื่ออนาคินโตเป็นหนุ่มก็ดันแหกกฎเจไดโดยแอบมีความสัมพันธ์ชู้สาวลับๆกับ ราชนินี "อามิดาลา" แห่งดาวนานู จนเธอตั้งครรภ์ลูกแฝด ... และอนาคินก็ค่อยๆถูกกลืนเข้าสู่ด้านมืดของพลังจนกลายเป็นซิทลอร์ด ได้ฉายา "ดาร์ธ เวเดอร์" ภายใต้การโน้มน้าวชี้นำของ ซิทลอร์ดลึกลับนาม "ดาร์ธ ซีเดียส" ซึ่งเผยในตอนท้ายว่า ซีเดียส ไม่ใช่ใครที่ไหน แต่คือท่าน "พัลพาทีน" สมุหนายกผู้มาสูงสุดของฝ่ายสาธารณรัฐเสียเอง และแท้จริงก็ยังเป็นผู้นำลับชักใยฝ่ายแบ่งแยกดินแดนด้วยอีกต่างหาก ... สงครามจบลงที่ฝ่ายสาธารณรัฐพ่ายแพ้ล่มสลาย อามิดาลาก็ตายหลังคลอดลูกแฝด ทั้งเหล่าอัศวินเจไดก็ถูกฆ่ากวาดล้างสิ้นแบบไม่หั่นดั่งตัว เหลือแต่ อ.โอปี้วัน กับ อ.โยดา ต้องลี้ภัยหลบหนีซ่อนตัว โดยลูกแฝดของอนาคินได้ถูกส่งไปสู่ที่หลบซ่อนลับเช่นกัน ... และแล้ว พัลพาทีน ก็กินรวบทั้งกระดาน เปลี่ยนการปกครองจาก ระบอบสาธารณรัฐเดิมไปเป็น ระบอบเผด็จการจักรวรรดิซิธแทน ตั้งตนเป็นจักรพรรดิปกครองแกแลคซีทั้งปวง โดยมี ดาร์ธ เวเดอร์ เป็นขุนพลซิทลอร์ดเคียงข้างนับแต่นั้นมา

# Tokenization - Thai

## Social media text

#สตอรี่ของโม 😊🐙 #Days23ofMobile 🌈 ...23 วันแล้วนะเจ้าโม พี่กำลังคิดถึงหนอยู่เลย แหะนะ เมื่อวานหายเลยนะ 🙈 พี่ก็ว่าหายไปไหนแอบหนีไปเล่นลองบอร์ดนี้เอง เล่นระว่างๆนะพี่เป็นห่วง เดวล์มเกมไม่มีทะเลามคอยเล่น เป็นเพื่อนอีก 😂 พี่จะบอกว่า "หนอย่าลืมลงชื่อเลือกตั้งนะ" เดียวพวกพี่ซำกันไม่ออกนะ 5555 ฝากไว้กันลืม ยิ่งเด้อๆอยู่ อีกอย่างๆหนูต้องกลับมานะพวกเรารอหนอยู่ 😊 พี่นี่เตรียมรอโหวตให้หนูเต็มทีเลย 😊 ไปเรียนเป็นใจบ้าง ยังเหงาอยู่มัย แต่พี่ว่าคงไม่แล้วหละมั้ง วันนี้ขึ้นสเตจอีกแล้วสินะ เหนื่อยมัยคะ แต่คงสนุกมากกว่าอยู่แล้วเนอะ 😊 แค่ได้เห็นรอยยิ้มของหนูพี่ก็สบายใจและ แต่เอ๊ะ เมื่อวานใครบ่นอยากกลับบ้านอีกแล้วนะ อดดู the toy เลยอะดี 😂 หว้ายๆๆ น่าสงสาร 555 โอกาสหนายังมีนะ ตั้งใจทำงานนะคะ เจ้าหลามแฟนหนูก็อด ไม่ได้อดคนเดียวซะหน่อยนะ 555 🙈 ดูมีความสุขจังน่าเจ้าตัวเล็ก 😊❤️ คิดถึงนะ เดียวก็ได้ 2shot กันแล้ว พี่โคตรตื่นเต้นเลย ตื่นเต้นกว่าไปจับมือเยอะ 🙈 คิดทำไมออกเลย มีท่าไหนแนะนำมัย ช่วยพี่หน่อยยยย 😂... #Mobilebnk48 #ตู้เพลงโมบิล #ชาวเหรียญหยอดตู้ #MOTA09

# Tokenization - Thai

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

**ตา กลม vs ตาก ลม**

**คณะกรรมการตรวจสอบถนน**

- 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

- Dictionary-based
  - Longest matching
  - Maximal matching
- 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
- **ป้าย**กลับรถ                      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**



# 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 | อากร | กช

> Language Modeling

คุณ | อา | 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  $i^{\text{th}}$  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:

$$d(i, j) = \begin{cases} 1 & \text{1<sup>st</sup> row and it is a word} \\ 1 + \min_{k=1 \dots i-1} d(k, i-1) & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ \infty, & \text{if } c[i..j] \text{ is in the dictionary.} \\ & \text{otherwise.} \end{cases}$$

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:

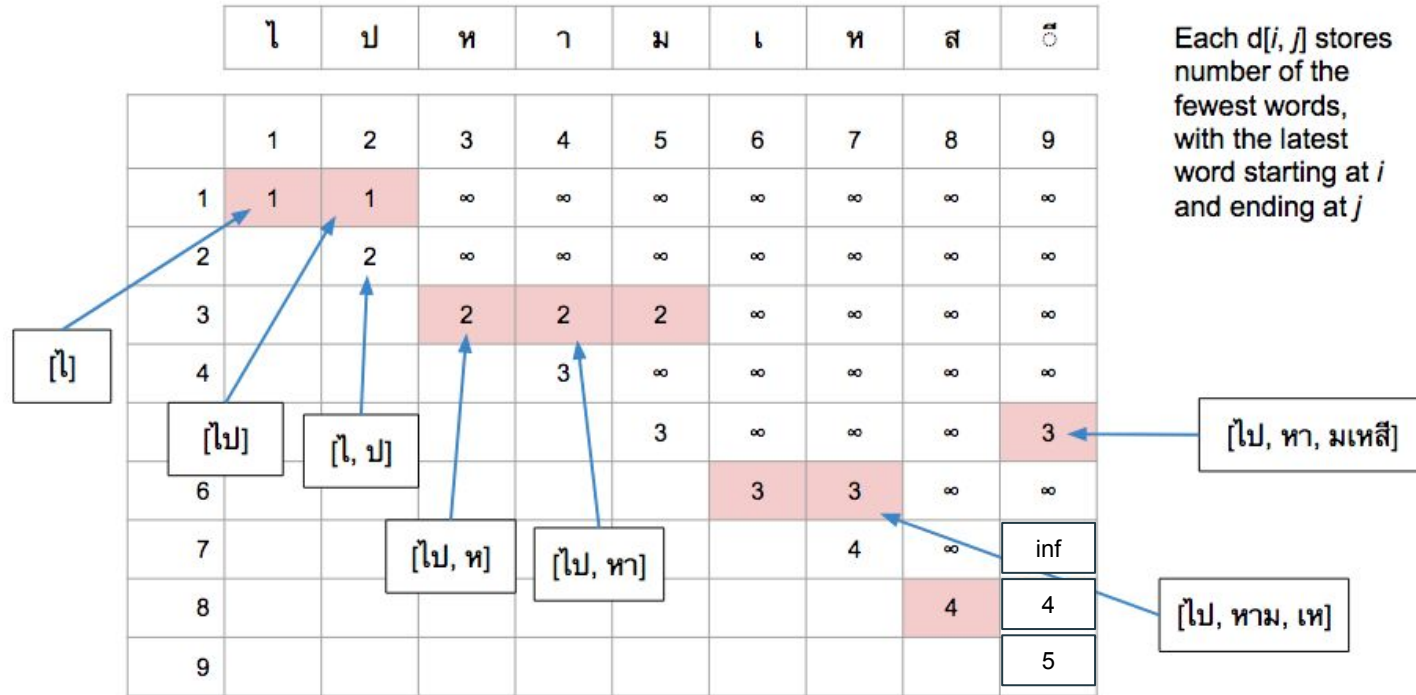
$$d(i, j) = \begin{cases} 1 & \text{if } i=1 \text{ and } c[1, j] \text{ is in the dictionary.} \\ \boxed{1} + \min_{k=1 \dots i-1} d(k, i-1) & \text{if } c[i..j] \text{ is in the dictionary.} \\ \infty, & \text{otherwise.} \end{cases}$$

The final 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  $d(1,1) = 1$ .

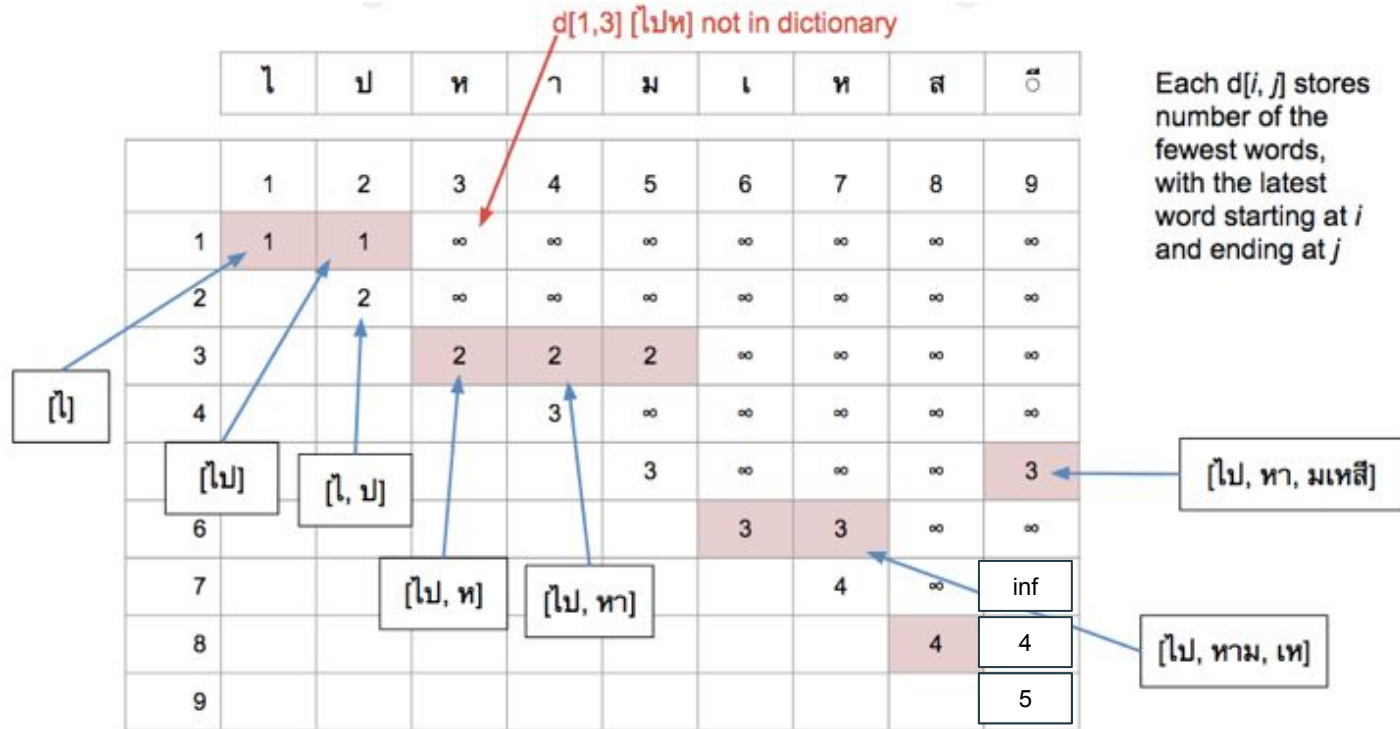


# Dynamic programming example (1)



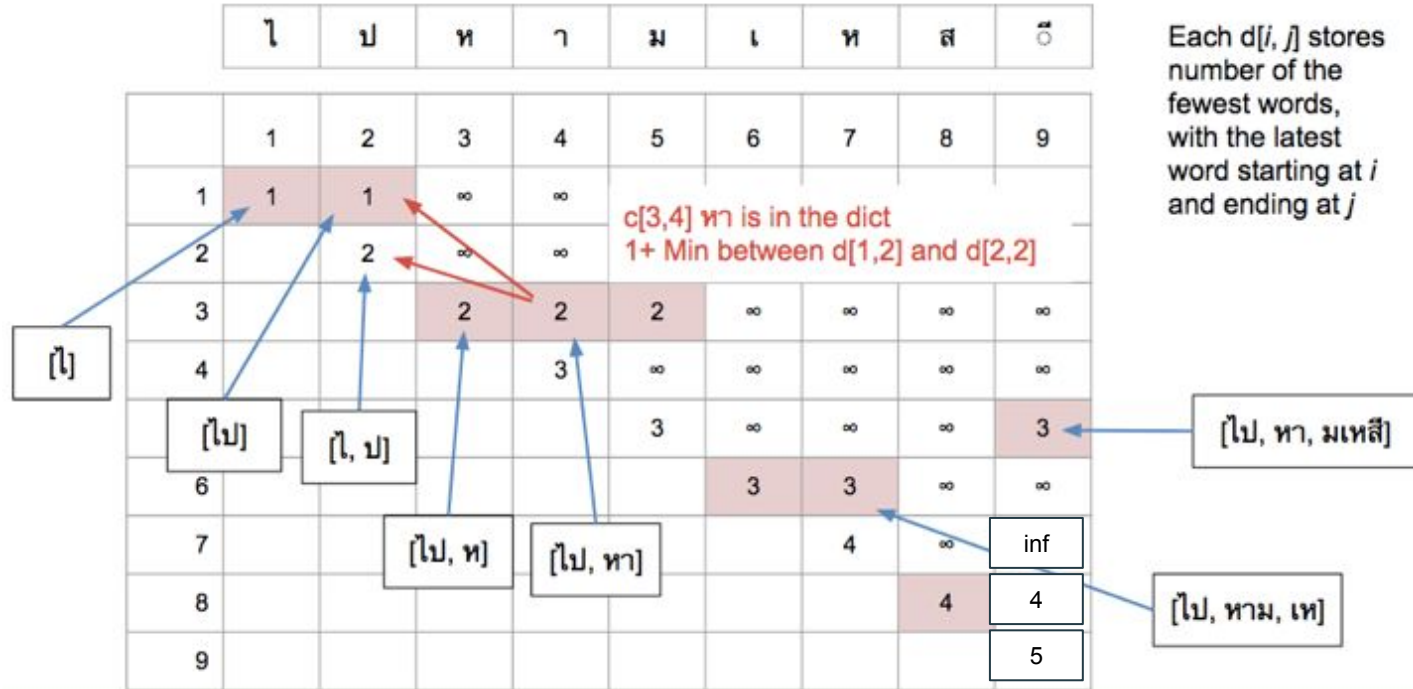
Dict: ไ, ป, ห, า, ม, เ, ห, ส, ี, ไป, หา, หาม, เท, สี้, มเหสี

# Dynamic programming example (2)



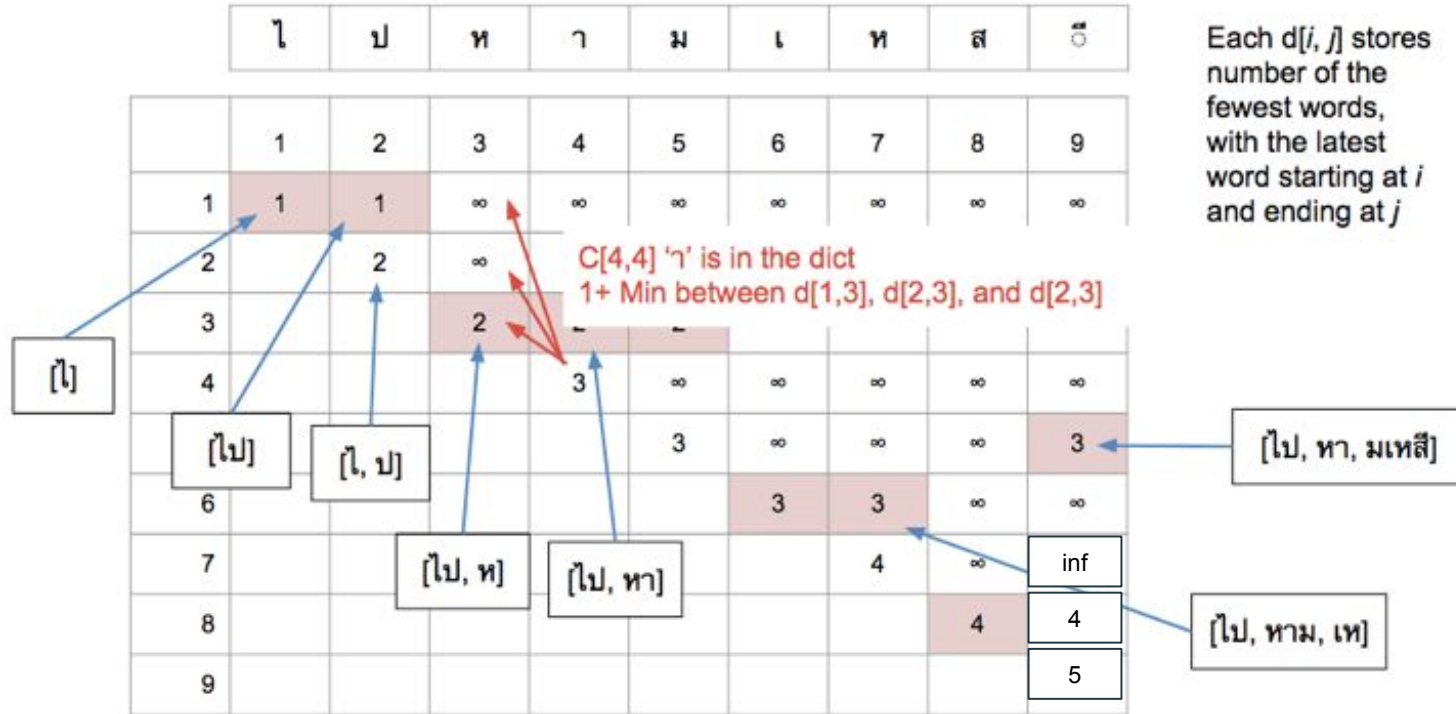
Dict: ไ, ป, ห, า, ม, เ, ส, ี, ไป, หา, ทาม, เท, สี่, มเหสี

# Dynamic programming example (3)

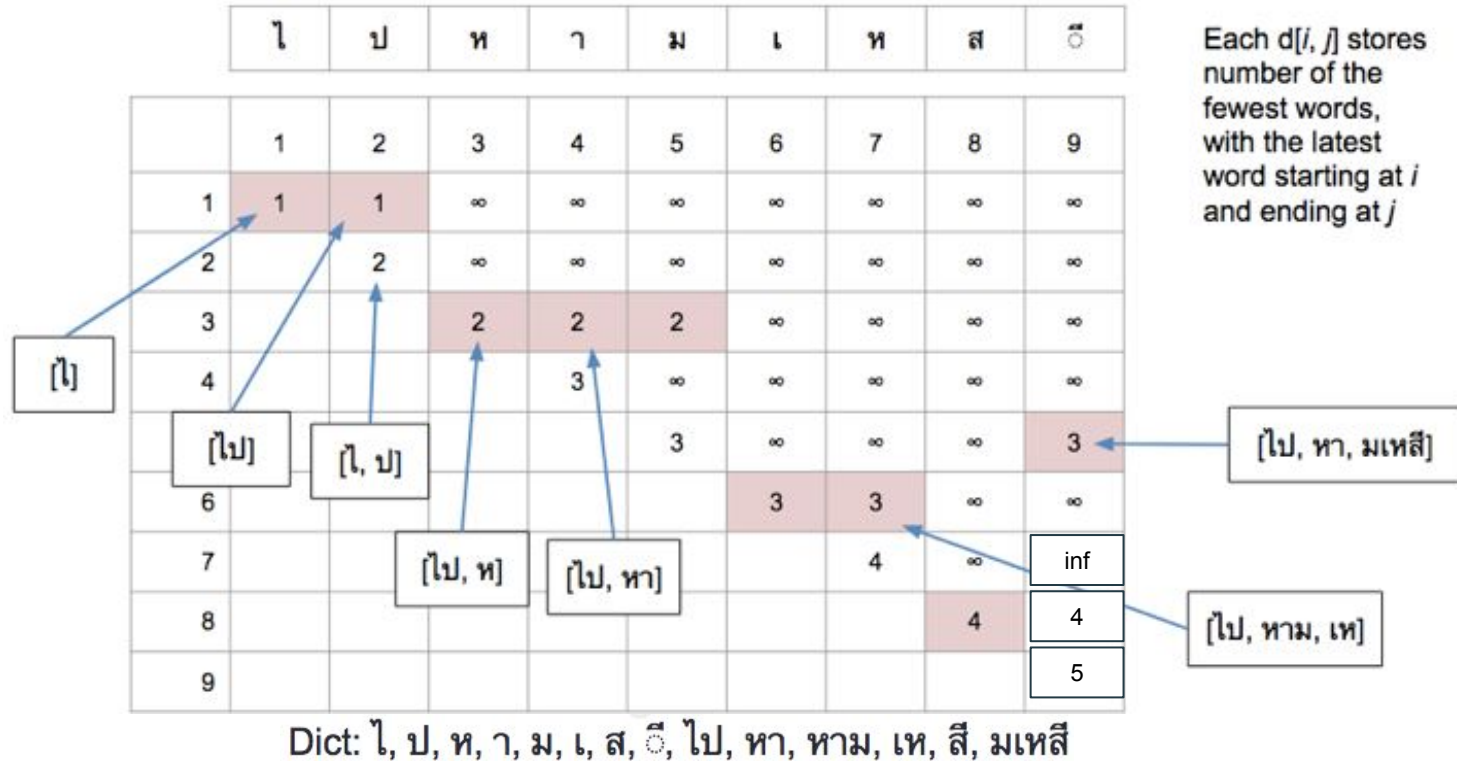


Dict: ไ, ป, ห, า, ม, เ, ส, ี, ไป, หา, หาม, เห, สี้, มเหสี

# Dynamic programming example (4)

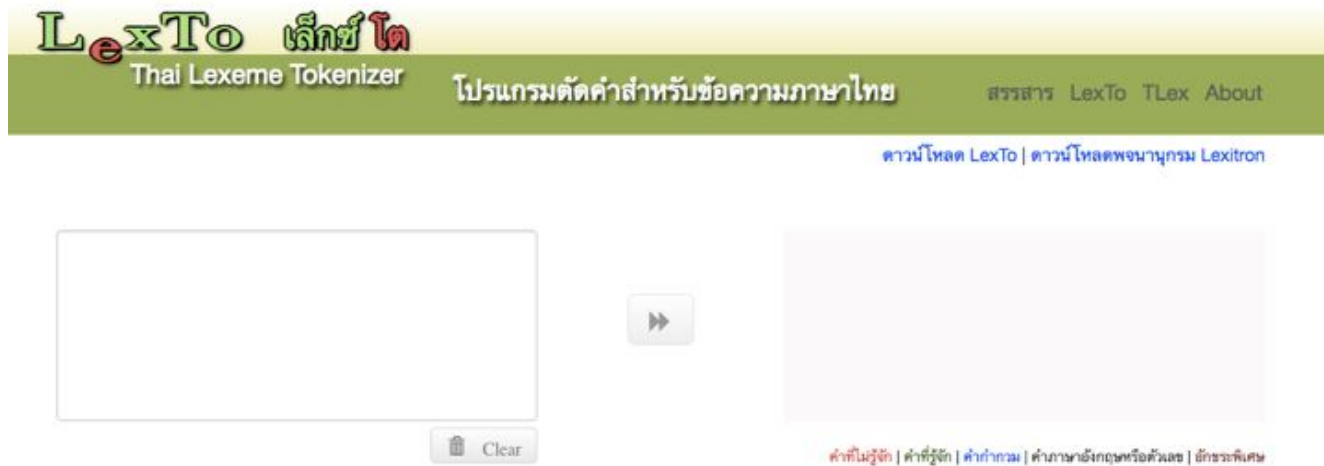


# Dynamic programming example (5): Backtracking



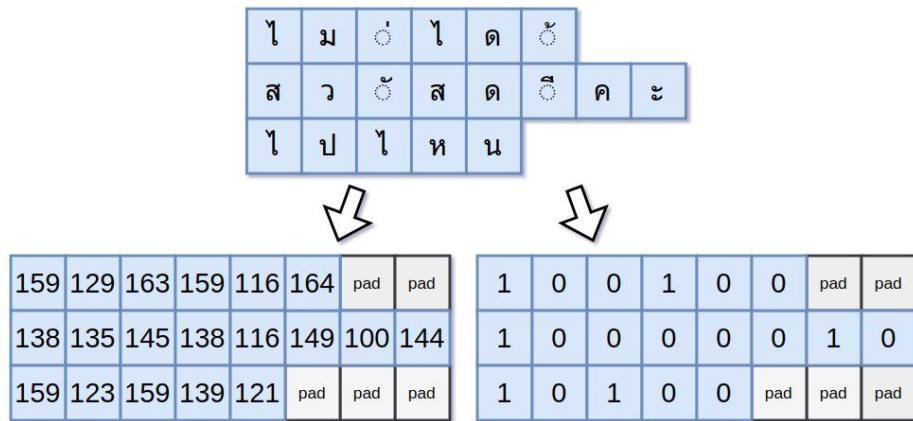
# LexTo

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



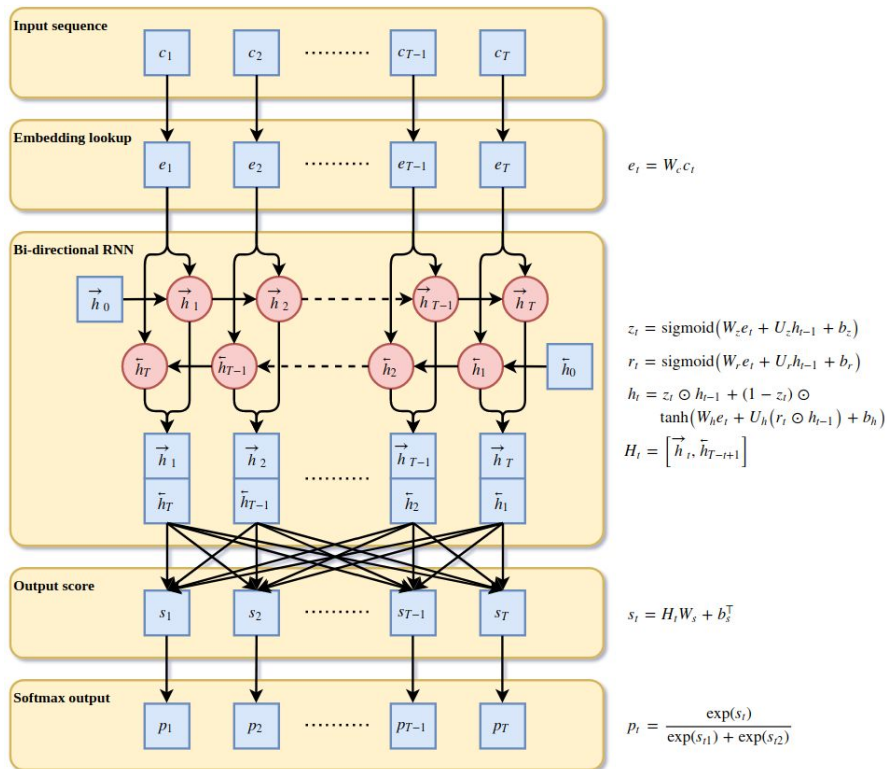
# Neural-based Tokenizer

## Step1: Preprocess



# Neural-based Tokenizer: Sertis Bi-GRU

## Step2: Model



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/)