Classification-based Thai Word Tokenizer

Wittawat Jitkrittum† Thanaruk Theeramunkong∗

†Sugiyama Lab.
Department of Computer Science
Tokyo Institute of Technology
Japan

∗KINDML Lab.
School of Information, Computer and Communication Technology (ICT)
Sirindhorn International Institute of Technology
Thailand

12 April 2010

Outline

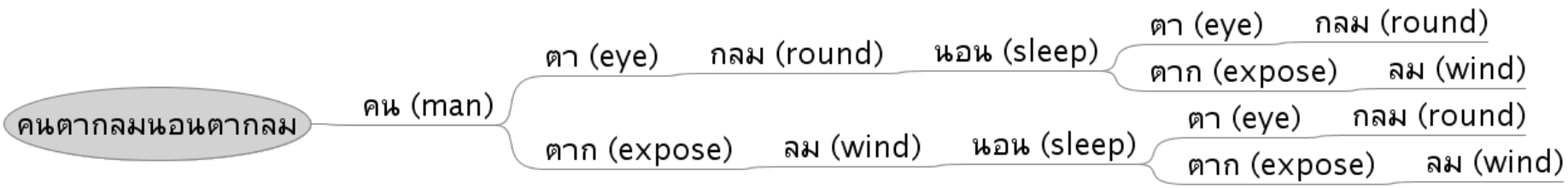
Introduction

- Generally, word tokenization is the first step in NLP.
- The errors from wrong tokenization will propagate to the subsequent processes.
- It is especially important for non-segmented language like Thai as there is no explicit word boundary written.
- In this work, we propose a tokenizing methodology using classification techniques.

Introduction

- Generally, word tokenization is the first step in NLP.
- The errors from wrong tokenization will propagate to the subsequent processes.
- It is especially important for non-segmented language like Thai as there is no explicit word boundary written.
- In this work, we propose a tokenizing methodology using classification techniques.

Example: Tokenization Possibilities



Characteristics of Thai Language

- **Tonal** – There are 5 tones (่ ้ ๊ ๋).
- **Non-segmented** – There are no spaces between words.
- **Sentences do not end with a period.**
- 44 consonants, 21 vowels, 4 tone markers
- **Interpreted from left to right** (no vertical writing).
- **No character cases** (i.e. no capital letters).
- **No inflection** (i.e. no plurals, no -ing, no past tense)

Thai Writing System

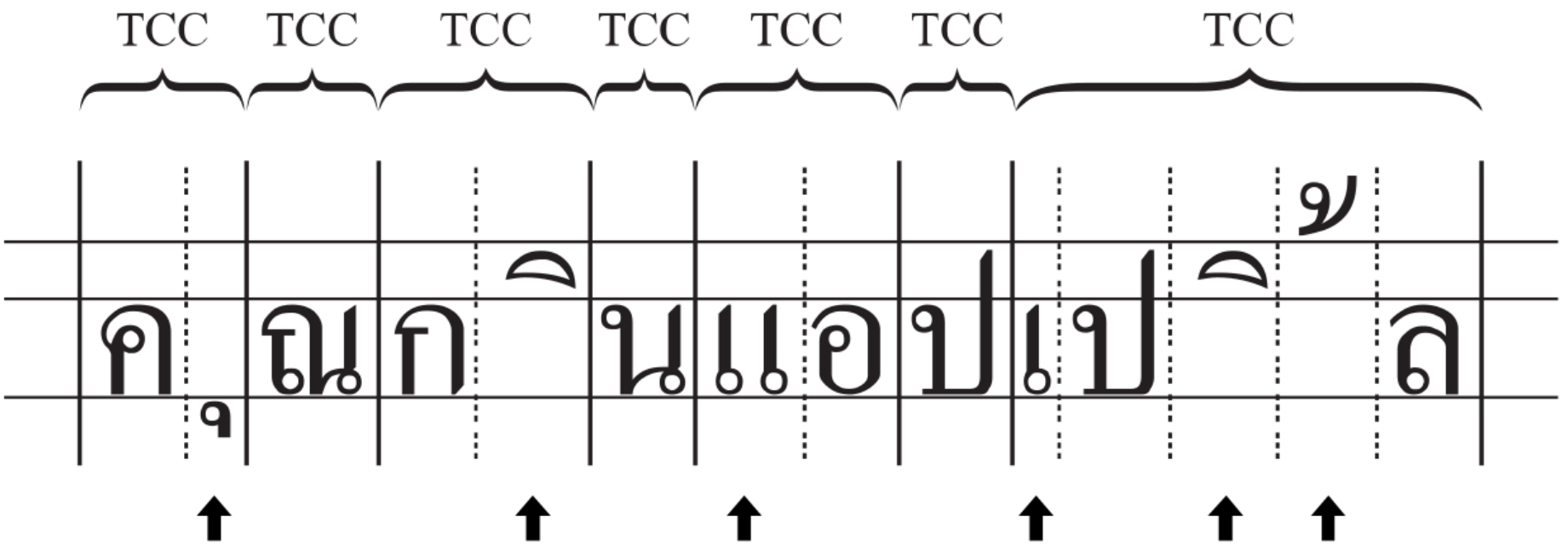


Figure: Thai writing system (image from Piya L., et al. 2009)

Sample of Thai Text

- Text from an encyclopedia about horses.

ม้าเป็นสัตว์เลี้ยงที่ใกล้ชิดกับคนมานับพันๆปี และเป็นสัตว์ที่สามารถใช้ประโยชน์ได้หลายอย่าง เช่น ใช้ในการขับขี่ บรรทุก ลากเข็น ขนส่ง หรือใช้ในการทำไร่ไถนา แต่ปัจจุบันการใช้แรงงานม้า สำหรับการทำไร่ไถนาลดน้อยลงไปมากเนื่องจากมีการพัฒนานำเครื่องจักรเครื่องมือทุ่นแรงมาใช้ทดแทนแรงงานสัตว์มากขึ้น นอกจากนี้ทางด้านการทหาร ม้าก็มีบทบาทสำคัญในการบรรทุกสัมภาระ และอุปกรณ์ต่างๆไปส่งยังแนวหน้าที่ยานพาหนะไปไม่ถึง และใช้เป็นยานพาหนะในราชการทหารม้าอีกด้วย ด้านการกีฬา ม้าก็มีส่วนสำคัญอย่างมากเช่นกัน ไม่ว่าจะเป็นกีฬาแข่งม้า การขี่ม้าข้ามเครื่องกีดขวาง หรือการขี่ม้าเล่นกีฬาโปโล

Outline

Thai Character Cluster (TCC)

- A group of characters which are inseparable according to Thai writing rules (Thanaruk T., et al. 2000). For example,
  - เพื่อ, จะ, กัน, ไฟ, ค่ำ, เอ็ง, เล่า
- Samples of TCC grammars:
  - tcc → อื
  - tcc → เ con ◌ิ TON con
  - tcc → แ con ◌็ con
  - tcc → con ◌ั TON con ะ

Overview of the Proposed Method

Input: $c_1, c_2, c_3, c_4, \ldots c_n$

1. Locate the Named Entities or long proper noun using a dictionary.
2. Tokenize the rest based on TCC rules.
3. For each TCC boundary b,
   - Use the trained classifier to classify b into either E or I (binary classification).

where . . .

- E = the end of a word
- I = a part of a word (not the end boundary)

Overview of the Proposed Method

Input: $c_1, c_2, c_3, c_4, \ldots c_n$

1. Locate the Named Entities or long proper noun using a dictionary.
2. Tokenize the rest based on TCC rules.
3. For each TCC boundary b,
   - Use the trained classifier to classify b into either E or I (binary classification).

where . . .

- E = the end of a word
- I = a part of a word (not the end boundary)

Overview of the Proposed Method

Input: $c_1, c_2, c_3, c_4, \ldots c_n$

1. Locate the Named Entities or long proper noun using a dictionary.
2. Tokenize the rest based on TCC rules.
3. For each TCC boundary b,
   - Use the trained classifier to classify b into either E or I (binary classification).

where . . .

- E = the end of a word
- I = a part of a word (not the end boundary)

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3.

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3.

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อ|ง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติด|ต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อ|ง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ขอ|ง|เข|ต|นั้น|ๆ| |เพื่อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. Model: เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ของ|เขต|นั้น|ๆ| |เพื่อ|ขอ|อนุญาต|ใช้|ไฟ|ฟ้า|

Example: Proposed Tokenization Method

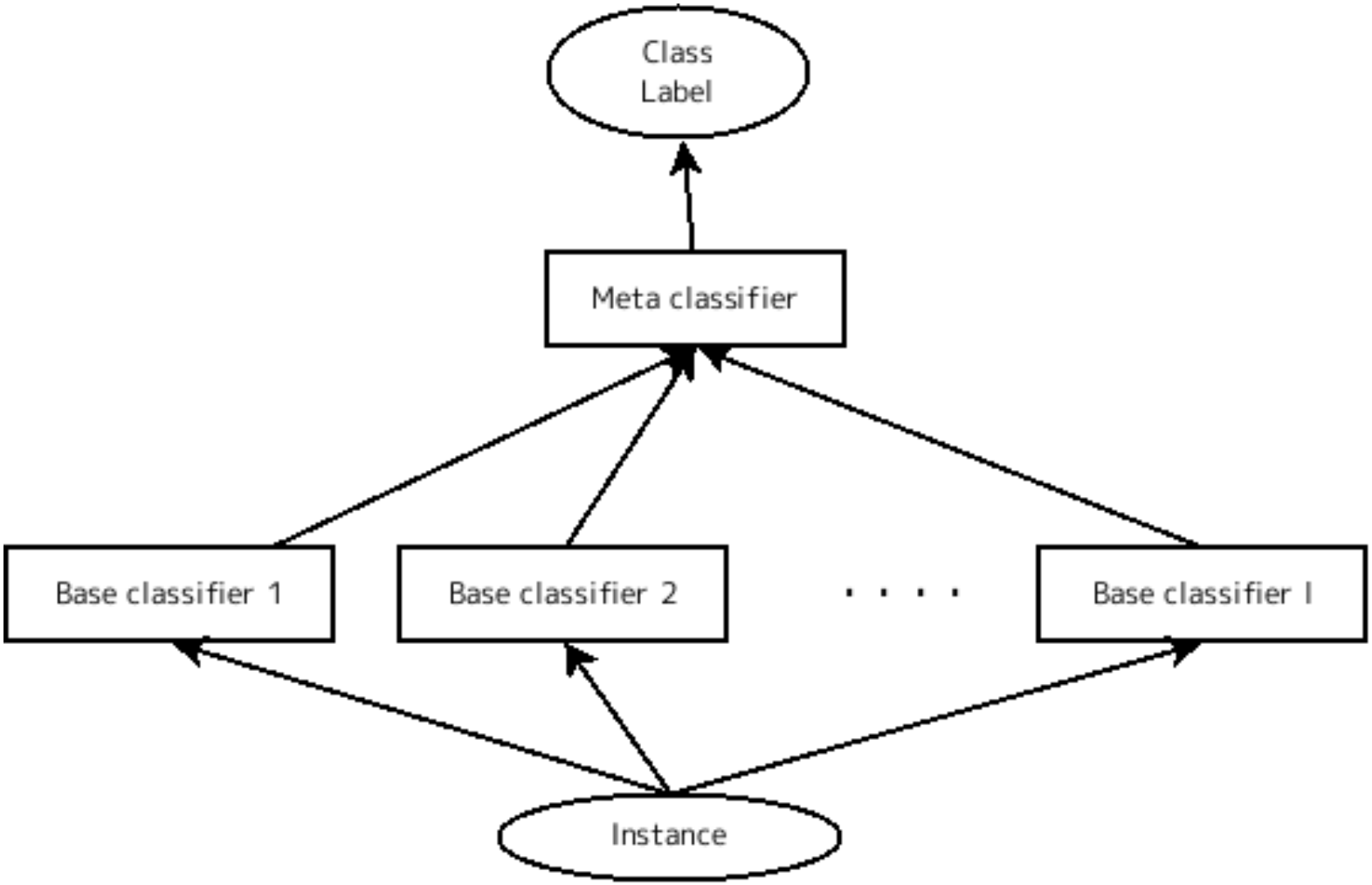
Input: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า

1. Dictionary: เจ้าของบ้านจะต้องไปติดต่อที่การไฟฟ้านครหลวง หรือการไฟฟ้าส่วนภูมิภาคของเขตนั้นๆ เพื่อขออนุญาตใช้ไฟฟ้า
2. TCC: เจ้าของบ้าน|จะ|ต้|อง|ไป|ติ|ด|ต่|อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ข|อ|ง|เข|ต|นั้น|ๆ| |เพื่|อ|ข|อ|อ|นุ|ญา|ต|ใช้|ไฟ|ฟ้า|
3. **Final Result:** เจ้าของบ้าน|จะ|ต้อง|ไป|ติดต่อ|ที่|การไฟฟ้านครหลวง| |หรือ|การไฟฟ้าส่วนภูมิภาค|ของ|เขต|นั้น|ๆ| |เพื่อ|ขอ|อนุญาต|ใช้|ไฟฟ้า|

Stacked Generalization

- Also known as stacking. Proposed by David H. Wolpert.
- Stacking is a method to combine many classifiers together. Classifiers are divided into 2 types:
  - **Base classifiers (Level-0 models)** – Classify an instance as usual. There can be any number of base classifiers.
  - **Meta classifier (Level-1 generalizer)** – Classify the instance formed by the results of the base classifiers.
- In short, stacking is a 2-level classification.
- In this work, stacking is applied in boundary classification after TCC step.

Stacked Generalization



Feature Overview

There are totally 32 features being used, which can be classified into 10 types.

1. **type(c_i)** – 13 character types (nominal attribute)

Value	Description	Samples
c	Final consonant	กขคฆงจชซญฐดตถทธนบปพฟภมยรลวศสอ
n	Non-final consonant	ฅ ฉ ผ ฝ ฌ ห ฮ ฤ ฦ
r	Unusual starting character	ฏ ฎ ฑ ฒ ฬ ฯ ษ ณ ฃ
v	Non-starting vowel	ะ ิ ี ึ ื ุ ู ั า ำ ๅ ็
w	Starting vowel	เ แ โ ใ ไ
t	Tone	่ ้ ๊ ๋
s	Thai Symbol	์ ๆ ฯ . ,
d	Digit	0-9, ๑-๙
q	Quote	' " “ ([ ])
p	Space	
f	Foreign alphabet	A-Z a-z
o	Other	
e	Non-existent character	e.g. non-existent c_{-5}

Feature Overview

2. **Next space** – Character distance from c_i to the next space.
3. **Previous space** – Character distance from c_i to the previous space.
4. **Suffix Proportion** – Proportion of gathered words ending with $..., c_{i-1}, c_i$
5. **Prefix Proportion** – Proportion of gathered words starting with $c_{i+1}, c_{i+2}, ...$
6. **Next person title** – Character distance from c_i to the next person title e.g. Dr., Professor.
7. **Previous person title** – Character distance from c_i to the previous person title.
8. **Left conditional probability** – $P(' \mid ' \mid ..., c_{i-2}, c_{i-1}, c_i)$
9. **Right conditional probability** – $P(' \mid ' \mid c_{i+1}, c_{i+2}, ...)$
10. **Separation Ratio** – $\mathrm{P}(\ ``c_i \mid '' \mid c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}\ )$ $= n(c_{i-2}, c_{i-1}, c_i, \mid, c_{i+1}, c_{i+2}) / n(c_{i-2}, c_{i-1}, c_i, *, c_{i+1}, c_{i+2})$

Full Set of Features

$$\ldots, c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}, c_{i+3}, \ldots$$

1. type(c_{i-6})
2. type(c_{i-5})
3. type(c_{i-4})
4. type(c_{i-3})
5. type(c_{i-2})
6. type(c_{i-1})
7. type(c_i)
8. type(c_{i+1})
9. type(c_{i+2})
10. type(c_{i+3})
11. type(c_{i+4})
12. type(c_{i+5})
13. type(c_{i+6})
14. next space
15. prev space
16. suffix(-3, 0)
17. suffix(-2, 0)
18. suffix(-1, 0)
19. prefix(1, 2)
20. prefix(1, 3)
21. prefix(1, 4)
22. next title
23. prev title
24. lcprob(-4, 0)
25. lcprob(-3, 0)
26. lcprob(-2, 0)
27. lcprob(-1, 0)
28. rcprob(1, 5)
29. rcprob(1, 4)
30. rcprob(1, 3)
31. rcprob(1, 2)
32. sepr(-2, 2)

Outline

BEST Competition

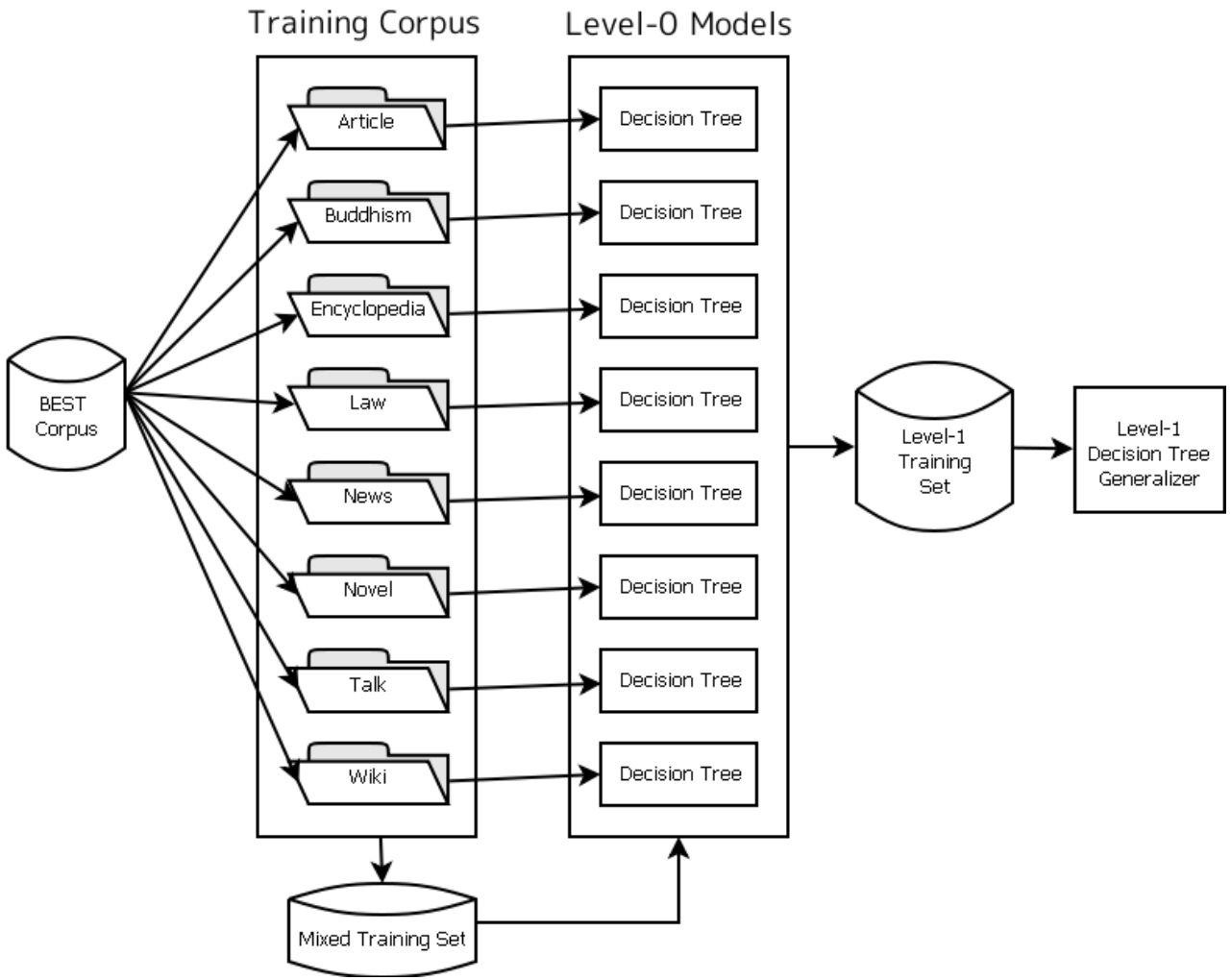
- **B**enchmark for **E**nhancing the **S**tandard for **T**hai language processing
- The first two years (2009, 2010) are Thai word tokenization competitions.
- Each competitor gets a manually tokenized corpus ($\approx$ 7 million words in 8 categories).
- The same test set (500K words) is tested on each developed program.
- The one with the highest F-measure wins.

Sample: BEST Corpus

- Sample text from news category

คำ|สั่ง|ให้|นำ|ตัว|ผู้|ต้อง|หา|ไป|คุมขัง|ที่|<NE>เรือนจำพิเศษกรุงเทพมหานคร</NE>| |เป็น|เวลา|อีก| |10| |วัน| |ตาม|คำ|ร้อง|ขอ|ออก|หมาย|ขัง|ครั้ง|ที่| |1| | โดย|<NE>นาย วีระ มุสิกพงศ์</NE>| |<NE>นายจตุพร พรหมพันธุ์</NE>| | <NE>นายจักรภพ เพ็ญแข</NE>| |<NE>นายณัฐวุฒิ ใสยเกื้อ</NE>| |<NE>นพ.เหวง โตจิราการ</NE>| |<NE>นายวิภูแถลง พัฒนภูมิไทย</NE>| |<NE>พ.อ. ดร. อภิวันท์ วิริยะชัย</NE>| |และ| <NE>นายมานิตย์ จิตต์จันทร์กลับ</NE>| |ผู้|ถูก|คุมขัง|ที่| |1|-|7| |และ| |9| |เห็น|ว่า|การ|ยื่น|คำ| ร้อง|ขอ|ออก|หมาย|ขัง|และ|การ|ที่|ศาล|มี|คำ|สั่ง|ให้|คุมขัง|ดัง|กล่าว| |เป็น|การ|คุมขัง|โดย|มิ|ชอบ|ด้วย|กฎหมาย|ตาม| |<AB>ป.</AB>|

Overall Training Process



Training Process

Category	samples
article	4.48 M
buddhism	1.83 M
encyclopedia	4.35 M
law	2.78 M

Category	samples
news	6.48 M
novel	5.56 M
talk	1.43 M
wiki	2.99 M

- Train 1 decision tree for each category.
- Altogether, there are $8 + 1$ decision trees (One for Level-1 Generalizer).
- Full training set is used to train each Level-0 Model.
- Due to computational limitation, only 1.4M samples (stratified sampling) from each category are used to train the Level-1 Generalizer.
- Total training samples for Level-1 Generalizer: $8 \times 1.4 = 11.2M$

Training Process

Category	samples	Category	samples
article	4.48 M	news	6.48 M
buddhism	1.83 M	novel	5.56 M
encyclopedia	4.35 M	talk	1.43 M
law	2.78 M	wiki	2.99 M

- Train 1 decision tree for each category.
- Altogether, there are $8 + 1$ decision trees (One for Level-1 Generalizer).
- Full training set is used to train each Level-0 Model.
- Due to computational limitation, only 1.4M samples (stratified sampling) from each category are used to train the Level-1 Generalizer.
- Total training samples for Level-1 Generalizer: $8 \times 1.4 = 11.2M$

Outline

Why do we use decision trees ?

- 10-fold cross validation is performed on each category using different classifiers.
- 30K stratified-sampled data from each category are used.
- Shown below are classification accuracies of each category-classifier pair.
- Use 32 proposed features

Dataset	1-R	NB	7-NN	DT	Bayes	RBFNet	RT
article	84.1	83.9	87.5 ○	95.0 ○	92.4 ○	84.1	87.4 ○
buddhism	82.8	86.3 ○	89.7 ○	96.4 ○	93.9 ○	87.0 ○	89.2 ○
encyclopedia	84.4	84.9	87.9 ○	94.8 ○	92.4 ○	85.2	86.0 ○
law	84.8	86.3 ○	90.5 ○	96.5 ○	93.5 ○	86.5 ○	91.1 ○
news	83.7	82.2 ●	87.6 ○	93.6 ○	91.1 ○	82.4 ●	85.6 ○
talk	83.9	84.2	88.1 ○	95.5 ○	93.3 ○	85.2 ○	87.3 ○
novel	83.5	84.6 ○	87.0 ○	95.2 ○	93.7 ○	85.0 ○	86.5 ○
wiki	81.5	83.4 ○	87.2 ○	93.7 ○	90.6 ○	82.3	84.4 ○
Average	83.6	84.5	88.2	**95.1**	92.6	84.7	87.2

○ =Significant improvement (compared to 1-R), ● = Significantly lower

BEST Evaluation

- Precision: $P = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Output words}}$
- Recall: $R = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Reference words}}$
- F-measure $= \frac{2 \cdot P \cdot R}{P+R}$

Reference input: `A|B|<NE>C D</NE>|E|`

Actual input: `ABC DE`

- Perfect tokenization: $A|B|C\ D|E|$ $(R = 1, P = 1)$
- Sample 1: $AB|C\ D|E|$ $(R = \frac{2}{4}, P = \frac{2}{3})$
- Sample 2: $A|B|C|D|E|$ $(R = \frac{3}{4}, P = \frac{3}{5}$, Considered as a perfect tokenization in BEST 2009.) [1]

[1] BEST 2009 did not count errors caused by NEs.

BEST Evaluation

- Precision: $P = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Output words}}$
- Recall: $R = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Reference words}}$
- F-measure $= \frac{2 \cdot P \cdot R}{P+R}$

Reference input: `A|B|<NE>C D</NE>|E|`

Actual input: `ABC DE`

- Perfect tokenization: $A|B|C\ D|E|$ $(R = 1, P = 1)$
- Sample 1: $AB|C\ D|E|$ $(R = \frac{2}{4}, P = \frac{2}{3})$
- Sample 2: $A|B|C|D|E|$ $(R = \frac{3}{4}, P = \frac{3}{5}$, Considered as a perfect tokenization in BEST 2009.) [1]

[1] BEST 2009 did not count errors caused by NEs.

BEST Evaluation

- Precision: $P = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Output words}}$
- Recall: $R = \frac{\#\text{ Correctly tokenized words}}{\#\text{ Reference words}}$
- F-measure $= \frac{2 \cdot P \cdot R}{P+R}$

Reference input: `A|B|<NE>C D</NE>|E|`

Actual input: `ABC DE`

- Perfect tokenization: $A|B|C\ D|E|$ $(R = 1, P = 1)$
- Sample 1: $AB|C\ D|E|$ $(R = \frac{2}{4}, P = \frac{2}{3})$
- Sample 2: $A|B|C|D|E|$ $(R = \frac{3}{4}, P = \frac{3}{5}$, Considered as a perfect tokenization in BEST 2009.) [1]

[1] BEST 2009 did not count errors caused by NEs.

Feature Set Testing

Without using a dictionary, 100K test set, Stacking (1 DT for each category)

Table: F-measure/Recall/Precision on different feature sets

Feature Set	BEST 2009	BEST 2010
32 features	95.38 / 95.17 / 95.59	**93.45** / 94.15 / 92.76
8 features	93.68 / 93.33 / 94.03	91.21 / 92.07 / 90.37
23 features	92.52 / 92.17 / 92.86	90.20 / 90.95 / 89.46
11 features	72.63 / 71.96 / 73.31	69.49 / 70.36 / 68.64

- 8 features = forward wrapper with 10-fold CV on mixed 30K samples
- 23 features = backward wrapper with 10-fold CV on mixed 30K samples
- 11 features = $type(c_i)$ where $-5 \leq i \leq 5$ (Language dependent feature set)

Performance vs. NE Dictionary Sizes

Use 32 features with Stack(DT, DT[])

Table: F-measure/Recall/Precision comparison on different sizes of NE dictionary

	BEST 2009	BEST 2010
NE $\geq$ 5 chars	95.13 / 94.80 / 95.46	**93.84** / 94.13 / 93.55
NE $\geq$ 7 chars	95.15 / 94.83 / 95.47	93.81 / 94.13 / 93.49
NE $\geq$ 9 chars	95.24 / 94.96 / 95.52	93.80 / 94.20 / 93.42
NE $\geq$ 11 chars	95.28 / 95.01 / 95.54	93.81 / 94.23 / 93.39
NE $\geq$ 15 chars	95.34 / 95.11 / 95.58	93.75 / 94.25 / 93.26
NE $\geq$ 19 chars	95.36/ 95.14 / 95.59	93.67 / 94.23 / 93.12
NE $\geq$ 25 chars	**95.37** / 95.16 / 95.59	93.56 / 94.18 / 92.95

Performance vs. Common-word Sizes

Use 32 features with Stack(DT, DT[])

Table: F-measure/Recall/Precision comparison on different sizes of common-word dictionary

	BEST 2009	BEST 2010
All common words	24.29 / 15.27 / 59.34	24.06 / 15.17 / 58.07
Common words $\geq$ 3 chars	42.32 / 30.69 / 68.14	41.76 / 30.44 / 66.52
Common words $\geq$ 5 chars	81.64 / 75.92 / 88.28	80.20 / 75.16 / 85.96
Common words $\geq$ 7 chars	90.26 / 87.69 / 92.99	88.49 / 86.74 / 90.31
Common words $\geq$ 10 chars	94.78 / 94.17 / 95.41	92.90 / 93.17 / 92.63
Common words $\geq$ 13 chars	95.37 / 95.07 / 95.67	**93.47** / 94.07 / 92.87
Common words $\geq$ 15 chars	**95.39** / 95.13 / 95.65	93.46 / 94.11 / 92.82
Common words $\geq$ 18 chars	95.38 / 95.15 / 95.61	93.44 / 94.12 / 92.78
Common words $\geq$ 20 chars	95.37 / 95.15 / 95.59	93.44 / 94.12 / 92.76

Overall Performance Comparison

Use 32 features with Stack(DT, DT[])

	BEST 2009	BEST 2010
NE ≥ 5 , Common words ≥ 13	95.12 / 94.70 / 95.54	**93.83** / 94.04 / 93.63
No dictionary	95.38 / 95.17 / 95.59	93.45 / 94.15 / 92.76
NE ≥ 5 chars	95.13 / 94.80 / 95.46	**93.84** / 94.13 / 93.55
Common words ≥ 13	95.37 / 95.07 / 95.67	93.47 / 94.07 / 92.87
No dict, 1 DT (1.4M / cat.)	95.10 / 94.93 / 95.27	93.27 / 93.95 /92.60
NE ≥ 5 chars, 1 DT (1.4M / cat.)	94.84 / 94.55 / 95.13	93.57 / 93.88 / 93.27

- 0.39% improvement in F-measure when dictionary of NE ≥ 5 is used.
- Running time is shorter when a dictionary is used.
- Precision increases when a dictionary is used.

Overall Performance Comparison

Use 32 features with Stack(DT, DT[])

	BEST 2009	BEST 2010
NE ≥ 5 , Common words ≥ 13	95.12 / 94.70 / 95.54	**93.83** / 94.04 / 93.63
No dictionary	95.38 / 95.17 / 95.59	93.45 / 94.15 / 92.76
NE ≥ 5 chars	95.13 / 94.80 / 95.46	**93.84** / 94.13 / 93.55
Common words ≥ 13	95.37 / 95.07 / 95.67	93.47 / 94.07 / 92.87
No dict, 1 DT (1.4M / cat.)	95.10 / 94.93 / 95.27	93.27 / 93.95 /92.60
NE ≥ 5 chars, 1 DT (1.4M / cat.)	94.84 / 94.55 / 95.13	93.57 / 93.88 / 93.27

- 0.39% improvement in F-measure when dictionary of NE ≥ 5 is used.
- Running time is shorter when a dictionary is used.
- Precision increases when a dictionary is used.

Error Analysis

- $|F_{2009} - F_{2010}| \approx 1.29\%$ error caused by NEs
  - The system can handle NEs quite well.
  - The rest of 6% results from common-word errors.
- In most cases, tokenizing person names yields . . .
  - นายสมชาย| ไกลชนะเลิศ| (Correct: นายสมชาย ไกลชนะเลิศ|)
  - นางสมหญิง |คงแก่เรียน| (Correct: นางสมหญิง คงแก่เรียน|)
  - The fact that we have นายสมชาย| ไกลชนะเลิศ| instead of นายสมชาย| |ไกลชนะเลิศ| shows that the features help in detecting person names.
- Many errors on common words result from inconsistency in the corpus. The followings are all acceptable.
  - เพราะ|ฉะนั้น| , เพราะฉะนั้น|
  - แต่|ว่า| , แต่ว่า|
  - ร้อย|ละ| , ร้อยละ|

Error Analysis

- $|F_{2009} - F_{2010}| \approx 1.29\%$ error caused by NEs
  - The system can handle NEs quite well.
  - The rest of 6% results from common-word errors.
- In most cases, tokenizing person names yields ...
  - นายสมชาย| ไกลชนะเลิศ| (Correct: นายสมชาย ไกลชนะเลิศ|)
  - นางสมหญิง |คงแก่เรียน| (Correct: นางสมหญิง คงแก่เรียน|)
  - The fact that we have นายสมชาย| ไกลชนะเลิศ| instead of นายสมชาย| |ไกลชนะเลิศ| shows that the features help in detecting person names.
- Many errors on common words result from inconsistency in the corpus. The followings are all acceptable.
  - เพราะ|ฉะนั้น| , เพราะฉะนั้น|
  - แต่|ว่า| , แต่ว่า|
  - ร้อย|ละ| , ร้อยละ|

Error Analysis

- $|F_{2009} - F_{2010}| \approx 1.29\%$ error caused by NEs
  - The system can handle NEs quite well.
  - The rest of 6% results from common-word errors.
- In most cases, tokenizing person names yields ...
  - นายสมชาย| ไกลชนะเลิศ| (Correct: นายสมชาย ไกลชนะเลิศ|)
  - นางสมหญิง |คงแก่เรียน| (Correct: นางสมหญิง คงแก่เรียน|)
  - The fact that we have นายสมชาย| ไกลชนะเลิศ| instead of นายสมชาย| |ไกลชนะเลิศ| shows that the features help in detecting person names.
- Many errors on common words result from inconsistency in the corpus. The followings are all acceptable.
  - เพราะ|ฉะนั้น| , เพราะฉะนั้น|
  - แต่|ว่า| , แต่ว่า|
  - ร้อย|ละ| , ร้อยละ|

Error Analysis

- $|F_{2009} - F_{2010}| \approx 1.29\%$ error caused by NEs
  - The system can handle NEs quite well.
  - The rest of 6% results from common-word errors.
- In most cases, tokenizing person names yields ...
  - นายสมชาย| ไกลชนะเลิศ| (Correct: นายสมชาย ไกลชนะเลิศ|)
  - นางสมหญิง |คงแก่เรียน| (Correct: นางสมหญิง คงแก่เรียน|)
  - The fact that we have นายสมชาย| ไกลชนะเลิศ| instead of นายสมชาย| |ไกลชนะเลิศ| shows that the features help in detecting person names.
- Many errors on common words result from inconsistency in the corpus. The followings are all acceptable.
  - เพราะ|ฉะนั้น| , เพราะฉะนั้น|
  - แต่|ว่า| , แต่ว่า|
  - ร้อย|ละ| , ร้อยละ|

Conclusions and Future Works

- We proposed a Thai word tokenizing methodology based on TCC, dictionary, and Stacking of DTs.
- The system can achieve 93.24 % on 100K test set.
- Future works:
  - To improve the runtime performance, convert DT models into code by representing each node as a function in the code.
  - Use word-based features to catch more semantic senses.
  - Incorporate Part-of-Speech informations (provided by *Orchid* corpus) into the feature set.

Conclusions and Future Works

- We proposed a Thai word tokenizing methodology based on TCC, dictionary, and Stacking of DTs.
- The system can achieve 93.24 % on 100K test set.
- Future works:
  - To improve the runtime performance, convert DT models into code by representing each node as a function in the code.
  - Use word-based features to catch more semantic senses.
  - Incorporate Part-of-Speech informations (provided by *Orchid* corpus) into the feature set.

Acknowledgement

- We thank National Electronics and Computer Technology Center (NECTEC) and Software Industry Promotion Agency SIPA for the supporting fund to participate in National Software Contest (NSC 2010).
- We thank Mr. Nattapong Tongthep (KINDML, SIIT) who allowed us to use his collection of person titles.
- We acknowledge Large Scale Simulation Research Lab. (LSR), NECTEC for providing computing resources that have contributed to the research results reported within this work.

  `http://www.lsr.nectec.or.th`

Thank you for your attention.

Do you have any question ?

Practicality of the System

- Compressed size: 17 MB
- Runtime size: 68 MB
- Runtime performance on 500K test set with Quad-core CPU:
  - Memory usage: $\approx$ 410 MB
  - Run time: 24 + 42 seconds = 66 seconds
    - 24 seconds for initial model loading
    - 42 seconds for tokenization process
  - 500K / 66 $\approx$ 7,575 words/second
- Written in Java (platform independent)
- Model pluggable at runtime (not limited to decision trees)
- Open-source (LGPL)

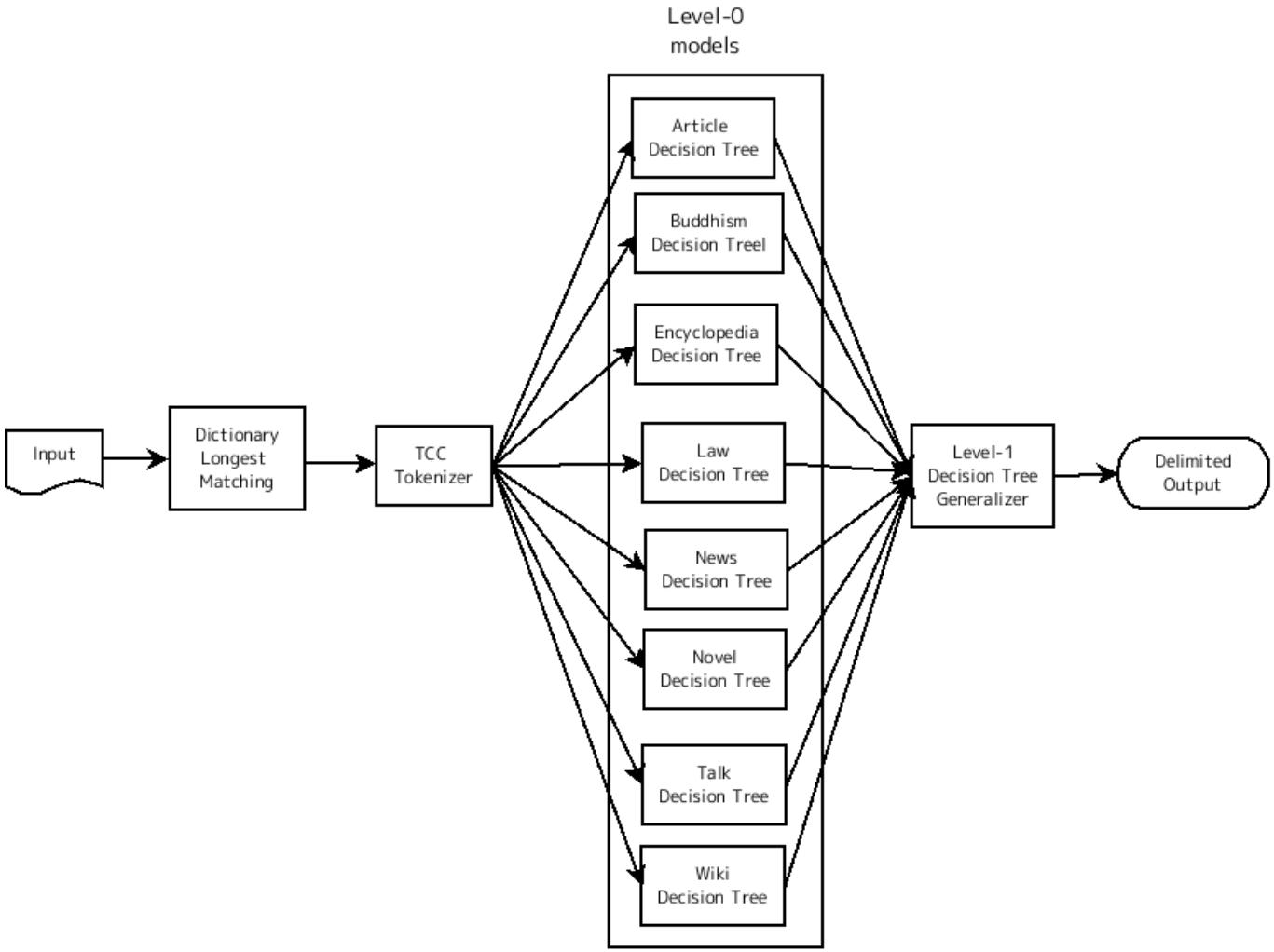
Practicality of the System

- Compressed size: 17 MB
- Runtime size: 68 MB
- Runtime performance on 500K test set with Quad-core CPU:
  - Memory usage: $\approx$ 410 MB
  - Run time: 24 + 42 seconds = 66 seconds
    - 24 seconds for initial model loading
    - 42 seconds for tokenization process
  - 500K / 66 $\approx$ 7,575 words/second
- Written in Java (platform independent)
- Model pluggable at runtime (not limited to decision trees)
- Open-source (LGPL)

Practicality of the System

- Compressed size: 17 MB
- Runtime size: 68 MB
- Runtime performance on 500K test set with Quad-core CPU:
  - Memory usage: $\approx$ 410 MB
  - Run time: 24 + 42 seconds = 66 seconds
    - 24 seconds for initial model loading
    - 42 seconds for tokenization process
  - 500K / 66 $\approx$ 7,575 words/second
- Written in Java (platform independent)
- Model pluggable at runtime (not limited to decision trees)
- Open-source (LGPL)

Overall Tokenization Process



Forward wrapper on DT, 30K Samples, 10-fold CV

$$\ldots, c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, \boxed{c_i}, c_{i+1}, c_{i+2}, c_{i+3}, \ldots$$

1. $type(c_{i-1})$
2. $type(c_i)$
3. $type(c_{i+1})$
4. $suffix(-3, 0)$
5. $lcprob(-2, 0)$
6. $rcprob(1, 5)$
7. $rcprob(1, 3)$
8. $sepr(-2, 2)$

Backward wrapper on DT, 30K Samples, 10-fold CV

$$\ldots, c_{i-4}, c_{i-3}, c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}, c_{i+3}, \ldots$$

1. type(c_{i-6})
2. type(c_{i-5})
3. type(c_{i-3})
4. type(c_{i-2})
5. type(c_{i-1})
6. type(c_i)
7. type(c_{i+1})
8. type(c_{i+3})
9. type(c_{i+4})
10. type(c_{i+6})
11. prev space
12. *suffix*$(-3, 0)$
13. *suffix*$(-2, 0)$
14. *prefix*$(1, 3)$
15. *prefix*$(1, 4)$
16. next title
17. prev title
18. *lcprob*$(-3, 0)$
19. *lcprob*$(-2, 0)$
20. *rcprob*$(1, 5)$
21. *rcprob*$(1, 4)$
22. *rcprob*$(1, 3)$
23. *sepr*$(-2, 2)$