Large-Scale Thai Statistical Machine Translation

Glenn Slayden, Mei-Yuh Hwang, Lee Schwartz
MSR-TR-2010-41

Abstract

Thai language text presents unique challenges for integration into large-scale multi-language statistical machine translation (SMT) systems, largely stemming from the nominal lack of punctuation and inter-word space. We review our independent solutions for Thai character sequence normalization, tokenization, typed-entity identification, sentence-breaking, and text re-spacing. We describe a general maximum entropy-based classifier for sentence breaking, whose algorithm can be easily extended to other languages such as Arabic. After integration of all components, we obtain a final translation BLEU score of 0.19 for English to Thai and 0.21 for Thai to English.

1 MT System Description

Our existing multilingual SMT system is based on hybrid generative/discriminative models, and we extend this approach for both English-to-Thai and Thai-to-English MT. Given a sequence of foreign words, $\boldsymbol{f}$, its best translation is the sequence of target words, $\boldsymbol{e}$, that maximizes

$$\begin{aligned}\boldsymbol{e}^* &= \text{argmax}_{\boldsymbol{e}}\, p(\boldsymbol{e}|\boldsymbol{f}) = \text{argmax}_{\boldsymbol{e}}\, p(\boldsymbol{f}|\boldsymbol{e})p(\boldsymbol{e}) \\ &= \text{argmax}_{\boldsymbol{e}}\, \{\log p(\boldsymbol{f}|\boldsymbol{e}) + \log p(\boldsymbol{e})\}\end{aligned}$$

where the translation model $p(\boldsymbol{f}|\boldsymbol{e})$ is computed on at least a dozen features, as briefly described in Section 1.1. The target language model (LM), $p(\boldsymbol{e})$, is represented by a smoothed n-grams (Chen 1996) and sometimes more than one LM is adopted in practice. To achieve the best performance, the log likelihoods evaluated by these features/models are linearly combined. After $p(\boldsymbol{f}|\boldsymbol{e})$ and $p(\boldsymbol{e})$ are trained, the combination weights λ_i are tuned on a held-out dataset to optimize an objective function, which we set to be the BLEU score (Papineni et al. 2002):

$$\{\lambda_i^*\} = \max_{\{\lambda i\}} \text{BLEU}(\{e^*\}, \{r\})$$

$$\boldsymbol{e}^* = \text{argmax}_{\boldsymbol{e}}\, \{\sum_i \lambda_i \log\, p_i(\boldsymbol{f}|\boldsymbol{e}) + \sum_j \lambda_j \log p_j(\boldsymbol{e})\}$$

where $\{r\}$ is the set of gold translations for the given input source sentences. To learn λ_i we use the algorithm described by Och (2003), where the decoder output at any point is approximated using n-best lists, allowing an optimal line search to be employed.

1.1 English-Thai Treelet Translation

Since we have a real-time rule-based English parser available, we based our English-Thai system on the "treelet" concept suggested in Menezes and Quirk (2008). This approach first parses the source language into a dependency tree which includes part-of-speech labels. A by-product of parsing is word breaking. For example, *John's* is parsed into *John* (noun) and *'s* (possessive).

During training we also break Thai sentences into tokens (Sections 2 and 3). Source and target sentences are lower-cased[1], to be able to share the training data between different casings. Next we apply word-dependent Hidden Markov Model (WDHMM) alignment (He 2007) to learn the mapping between the source and target token streams. Given a parallel corpus with these word alignments and dependency parses, we extract both ***treelet translation pairs*** and ***order templates***. From each aligned sub-graph of the source tree, we extract that sub-graph along with its translation, and collate these across the whole dataset to produce a set of treelets with their translations. Likewise from each source dependency tree node along with its full daughter set, we also extract an order template that specifies the relative order of those daughters in the target language, as well as any inserted or deleted words. Order templates are also collated across the whole dataset to produce a distribution of possible reordering and word insertion or deletion.

Taking a parsed source sentence, a ***derivation*** is a set of treelet translation pairs and order templates that completely cover the input dependency parse; the target side of this derivation may be read off to produce a target language string. To translate a given source sentence, we first parse the sentence, then find the best derivation for that sentence according to the log linear combination method as described above.

This system employs eleven features altogether. The first two features are the log probabilities of each source treelet given its target side treelet and each target treelet given its source side using relative frequency estimates. The next two features are again conditional log probabilities, estimated using Model 1 (word-based probabilities) instead (Brown et al. 1993).

Another feature is the log probability of the target side of each order template given its source side, again estimated using relative frequency. In the case that no template matches a given source configuration, we generate an order template that preserves the source order; we include a feature that counts the number of such source order templates. We also in-

[1] Although Thai does not use orthographic case, this operation affects Roman alphabet text which may appear in Thai input texts.

clude features to count the number of words inserted or deleted by order templates.

The log probability of the target translation according to an n-gram language model acts as another feature. Finally, we add a word count feature to counteract any language model preference for shorter outputs, and a phrase count feature to bias the search toward derivations with fewer and thus larger treelets.

After the treelet decoder outputs the translated Thai word sequences, we re-space the output to comply with Thai convention (see Section 5) based on linguistic rules and transfer of syntactic information between source and target dependency trees.

1.2 Thai-English Phrasal Translation

Lacking a Thai parser, we use a purely statistical phrasal translator after Pharaoh (Koehn 2004) for Thai-to-English translation. As with the treelet system, Thai sentences are normalized, lower-cased, and tokenized. English words are broken mainly by whitespace, with a few rules to break contractions, and then lower-cased. Again WDHMM is used to align the parallel corpus to obtain phrase pair mappings. Similar 11 features are combined linearly in the log probability domain to evaluate a given translation.

2 Thai Character Sequence Normalization

The four Thai tone marks and some Thai vowel glyphs are super- and/or sub-scripted with respect to a base character. These combining marks are represented by individual Unicode code points. When two or more of these combining marks are present on the same base character, the ordering of these code points in memory should be consistent so that orthographically identical entities are recognized as equivalent by the MT system. However, many computer word processors do not enforce the correct sequence and/or do not properly indicate incorrect sequences to the user visually.[2] This often results in documents with invalid byte sequences.

Correcting these errors is desirable for SMT input. For this task, we developed a Finite State Transducer (FST) which detects and repairs a number of sequencing errors which render Thai text either linguistically invalid, or not in a correct Unicode sequence.

For example, a superscripted Thai tone mark should follow a super- or sub-scripted Thai vowel when they both apply to the same consonant. When the input has the tone mark and the vowel glyph swapped, the input can be fully repaired:

อ า ่ น ➜ อ ่ า น ➜ อ่าน
อ ั ิ น ➜ อ ิ ั น ➜ อิ้น

Figure 1. Two unambiguous repairs

Other cases are ambiguous. The occurrence of multiple adjacent vowel glyphs is an error where the original typist's intention is not clear. We retain the first-appearing glyph, unless it is a pre-posed vowel, in which case we retain the last-appearing instance. These two treatments are contrasted in Figure 2.

จะะา ➜ จะ
ใไป ➜ ไป

Figure 2. Two ambiguous repairs

Another common repair that our FST performs pertains to two compound vowel glyphs (อำ and แอ) which some Thai typists assemble from separate keys that appear on the Thai keyboard. Each of these compound vowels can be represented with a single Unicode code point, and this is the preferred form. Since the incorrect, two-code-point sequence displays as visually identical to the correct one, this error is persistent in Thai text processing. We normalize them to use the preferred, single-code-point sequence.

อ ◌ำ า ➜ อ ◌ำ ➜ อำ
เ เ อ ➜ แ อ ➜ แอ

Figure 3. Two common mis-codings

We measure the utility of our normalization filter by the proportion of lines for which we make at least one correction. We found that for newswire text we fix about 0.05% of lines, while for broad-range web scraped data, it is as high as 4.1%. This measure is expected to under-represent the utility of the filter to SMT, since Thai text streams, lacking intra-word spacing and permitting two unwritten vowels, have few re-alignment checkpoints and hence tokenization state machines can persist in misaligned states.

3 Thai Tokenization

Thai text does not normally use the space character to separate words, except in certain specific contexts. Although Unicode offers the Zero-Width Space (ZWSP) as one solution for indicating word breaks in Thai, it is hardly used. Programmatic tokenization has become a staple of Thai computational linguistics. The problem has been well studied, with precision and recall near 95%, and is now considered of primarily engineering interest. Lexical approaches (Poowarawan 1986) of two decades ago have given way to Support Vector Machine (SVM) (Haruechaiyasak et al. 2008), decision trees (DT) (Theeramunkong and and Usanavasin 2001) and Conditional Random Field (CRF) (Kruengkrai et al. 2006).

3.1 Uniscribe Thai Tokenization

For Thai tokenization (or "word breaking"), we apply manually-selected heuristics to the output of Uniscribe (Bishop et al. 2003), a commercially-available component. Post-processing heuristics fall into two categories: "re-gluing" words that Uniscribe broke too aggressively, and a smaller class of cases of manually breaking words that Uniscribe did not break.

Re-gluing is achieved by comparing Uniscribe's output against a Thai lexicon in which desired breaks within a word are tagged. Underbreaking by Unis-

[2] Windows Vista helps the situation by visually distinguishing certain incorrect character sequences.

cribe is less common and is restricted to a number of common patterns which are repaired explicitly.

Because most typed entities require re-gluing, it is best to identify them during tokenization. Many of these employ predictable spacing and punctuation patterns, and as we will see in Section 4, each space or punctuation character that can be eliminated at this stage improves sentence-breaker performance by allowing the model to concentrate on the classification of truly ambiguous cases. Accordingly, we now describe two of the typed entities our system attempts to discern.

3.2 Person Names and Numbers

Person names in Thai adhere to a particular convention for the insertion of space characters. This allows Thai readers to identify the boundaries of multi-syllable surnames that they may not have seen before. The following grammar[3] summarizes the prescriptive conventions for names appearing in Thai text: [4]

```
<name-entity> ::= <honorific>  <name>
<full-name> ::= <first-name> [<last-name>]
<first-name> ::= <name-text> space
<last-name> ::= <name-text> space
<name-text> ::= <thai-alphabetic-char>+
<thai-alphabetic-char> ::= ก | ข | ฃ | ค | ...
```

Figure 4. Name entity recognition grammar

We use a hash lookup to determine if a token matches one of the following predefined categories: name-introducing honorific, Thai or foreign given name, token which is likely to form part of a surname (Figure 5), or token which aborts the gathering of a name (i.e. is unlikely to form part of a name).

likely name part	full surname example 1	example 2
ภรณ์	**ศิริ/ภัส/รา/ภรณ์**	**วิ/ทู/รา/ภรณ์**
จันทร์	**ผล/จันทร์**	**จันทร์/สว่าง**
พรรณ	**ศรี/สุ/พรรณ***	**วิไล/พรรณ**
ศรี	**ศรี/ประสาธน์**	**ศรี/สุข**

Figure 5. Examples of likely name-part tokens with example surnames. The name marked with an asterisk consists entirely of likely name-parts. Preliminary tokenization is shown by a slash.

...ว่านายจิระนุช วินิจจกุล ว่า...											
ว่า	**นาย**	**จิ**	**ระ**	**นุช**		**วิ**	**นิจ**	**จกุ**	**ล**		**ว่า**
	h	g0	g1	g2	sp0	s0	s1	s2	s3	sp1	
that	Mr.	<oov>	hit	beloved		<oov>	stable	<oov>	<oov>		said
...that	Mr.	Chiranut				Winichotkun					said...

Figure 6. Thai person-name entity recognition

Figure 6 shows a Thai name appearing within a text fragment. The second row indicates where Uniscribe detects tokens; these are the input tokens to our name identifier. In the third row we have identified four classes of tokens, described below. The fourth line shows the English translation gloss, or <oov> if none. The bottom row is the desired translation output.

Our tokenizer first notes the presence of an honorific {h} **นาย** followed by a pattern of tokens {g0-g*n*}, {s0-s*n*} and spaces {sp0, sp1} that is compatible with a person name and surname of sensible length.

Next, we determine which of those tokens in the ranges {g} and {s} following the honorific do not have a gloss translation (i.e., are not found in the lexicon). These tokens are indicated by <oov> in the gloss above. When the number of unknown tokens exceeds a threshold (discussed below), we hypothesize that these tokens form a name. The lack of lexical morphology in Thai facilitates this method because token (or syllable) lookup generally equates with the lookup of a stemmed lemma.

The algorithm refers to a manually-constructed set of stop tokens which abort either the surname gathering or the entire name recognition. More so than in English, where a common word like “and” can appear within a name such as “Sandy,” Thai morphemes are isolating, so a word such as **และ** “and” would not be expected to appear within a name.

Additionally, a set of complete names and likely name-parts is used to bias the probability of name detection in cases where the unseen-token ratio is judged inconclusive.

Continuing with the above example, neither a stop token nor a likely name-part is present so the last name qualifies on the basis that more than one-half of its syllables are unknown. Note that we complete our assessment of the last name first because it sandwiches the first name with the honorific. Its detection as a surname will positively bias the first name analysis if needed, whereas the reverse situation, where the putative surname is unbounded, is not predictive.

Finally, the first name, having no stop syllables or name-parts, is considered. In the example, having only one out of three unknown syllables is a threshold case, but the confirmed surname biases for positive detection. As a result of this analysis, the tokenizer emits the tokens shown in the bottom row of Figure 6.

The simple scoring described above characterizes our current system, but this infrastructure of stop tokens, likely name-part tokens, and unseen-token ratios could easily be adapted to a machine learning approach.

3.3 Calendar Date Recognition

We also attempt to identify and atomicize Thai calendar dates during the Thai tokenization process. As a pre-requisite to identifying dates, we also identify and package all numeric entities that occur in the input text during this phase. For Thai, this includes mapping Thai orthographic digits {๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙} to Arabic digits 0 through 9, respectively. In the

[3] In this modified-BNF form, square brackets are used to indicate optional items, '+' indicates an item repeated one or more times, and ellipses indicates a self-evident continuation.

[4] Unlike many East Asian languages, Thai names are written with first names first, followed by last names (family names). The honorific is still used with the first name when the last name is omitted, and this format is equally formal

following discussion, Thai and Arabic digits are taken to be interchangeable, as our system performs remapping as required. For example, our system would interpret the input text ๒๕๔๐ as equivalent to "2540."

...ในวันที่ 14 มีนาคม ๒๕๔๐ และ...									
ใน	วัน	ที่	sp	14	มีนาคม	sp	๒๕๔๐	sp	และ
on	day	which		14	March		2540		and
...on	March 14th, 1997								and...

Figure 7. Date entity recognition

Figure 7 shows a fragment of Thai text which contains a calendar date for which our system will emit a single token. The first row of the table shows the Thai input; the second row of the table shows the initial tokenization result; the third row shows linguistic glosses, and the bottom row shows the desired output. As shown in the example, our system detects and adjusts for the use of Thai Buddhist year dates when necessary, by adding 543 to the Christian year. This allows the SMT system to display dates in accordance with a calendar system that is prevalent for the culture of the target language, and hence one that the end user is likely to find meaningful. The gathering of disparate and optional parts of the Thai date is summarized by the grammar in Figure 8.

```
<date-entity> ::= [<cardinal-words>] [space] <date>
<cardinal-words> ::= วันที่ | ที่
<date> ::= month-date [space] year
<year> ::= <tha-digit> <tha-digit> <tha-digit> <tha-digit>
<year> ::= <ara-digit> <ara-digit> <ara-digit> <ara-digit>
<month-date> ::= <day> [space] <month>
<day> ::= <thai-digit>+
<day> ::= <ara-digit>+
<month> ::= <month-full> | <month-abbr>
<month-full> ::= มกราคม | กุมภาพันธ์ | มีนาคม | ...
<month-abbr> ::= ม.ค. | ก.พ. | มี.ค. | ...
<tha-digit> ::= ๐ | ๑ | ๒ | ๓ | ๔ | ๕ | ๖ | ๗ | ๘ | ๙
<ara-digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Figure 8. Date recognition grammar

If the constraints of this grammar cannot be met, we may still gather atomic tokens for stand-alone year references that appear in the text. Integers without proximal lexical evidence are presumed to be a year if they fall in a relatively narrow range corresponding to the years 1960 through 2025 (2503-2568). This range is widely expanded for integers followed by the token signifying "Buddhist Era" (พ.ศ.), which is similar to the English designator "A.D.," or preceded by the word ปี, "year."

4 Max-Entropy Based Sentence-Breaking

SMT relies on parallel text corpora to learn the mapping between two languages. Our training data includes both purchased and web-crawled content. For the latter, we identify paired documents and perform automatic sentence alignment. This requires first identifying sentence boundaries.

In general, Thai is written without sentence-end punctuation, but the space character is always present between sentences. Space may also appear within a sentence according to linguistic or prescriptive-orthographic motivation (see Section 5.1). The lack of sentence-final punctuation in Thai disallows many important methods such as Palmer and Hearst (1997). Thai sentence-breaking (SB) is therefore a task of classifying a space token as either sentence-breaking (**sb**) or non-sentence-breaking (**nsb**). We train a maximum entropy (ME) classifier (Ratnaparkhi 1996; Reynar and Ratnaparkhi, 1997) on features which describe the proximal environment of the space token under consideration and use this model at runtime to classify space tokens with possibly unseen contexts.

Under the ME framework, let B={**sb**, **nsb**} represent the set of possible classes we are interested in predicting for each space token in the input stream. Let C={linguistic contexts} represent the set of possible contexts that we can observe, which must be encoded by binary features, $f_j(b,c), 1 \le j \le k$, such as:

$$f_1(b,c) = \begin{cases} 1 \text{ if the previous word is English } \textit{and } b = \textbf{nsb}. \\ 0 \text{ otherwise.} \end{cases}$$

This feature helps us learn that the space after an English word is usually not a sentence boundary.

$$\boldsymbol{f_2(b,c)} = \begin{cases} \textbf{1 if the distance to the previous honorific} \\ \textbf{is less than 15 tokens } \boldsymbol{and\ b} = \textbf{nsb} \\ \textbf{0 otherwise} \end{cases}$$

This feature enables us to learn that spaces which follow an honorific are less likely to mark sentence boundaries. Assume the joint probability *p(b,c)* is modeled by

$$p(b,c) = Z \prod_{j=1}^{k} \alpha_j^{f_j(b,c)}$$

where we have *k* free parameters $\{\alpha_j\}$ to estimate and Z is a normalization factor to make $\sum_{b,c} p(b,c) = 1$. The ME learning algorithm finds a solution $\{\alpha_j\}$ representing the most uncertain commitment

$$\max\ H(p) = -\sum p(b,c) \log p(b,c)$$

that satisfies the observed distribution $\hat{p}(b,c)$ of the training data:

$$\sum p(b,c) f_j(b,c) = \sum \hat{p}(b,c) f_j(b,c),\ 1 \le j \le k$$

This is solved via the *Generalized Iterative Scaling* algorithm (Darroch and Ratcliff 1972). At run-time, a space token is considered an **sb**, if and only if p(**sb**|c) > 0.5, where

$$p(\boldsymbol{sb}|c) = \frac{p(\boldsymbol{sb},c)}{p(\boldsymbol{sb},c) + p(\boldsymbol{nsb},c)}$$

4.1 Sentence Breaker Training Corpus

Thai corpora which are marked with sentence breaks are required for training. We assembled a corpus of 361,802 probable sentences. This corpus contains 911,075 spaces, a figure which includes one inter-sentence space per sentence.

4.2 Out-of-context Sentences

For 90.7% of our training sentences, the paragraphs from which they originate are inaccessible. In feature

extraction for each of these sentences (see Section 4.3), we wrap the sentence's head around to its tail to obtain its **sb** context. For a sentence of tokens t_0-t_n, its **sb** context is given by

$$\{ w=t_{n-2}, x=t_{n-1}, y=t_0, z=t_1 \}$$

Although not an ideal substitute for sentences in context, this ensures that we extract at least one **sb** context per sentence. The number of **nsb** contexts extracted per sentence is equal to the number of interior space tokens in the original sentence.

Sentence wrapping is not needed when training with sentence-delimited paragraph sources. Contexts **sb** and **nsb** are extracted from the token stream of the entire paragraph and wrapping is used only to generate one additional **sb** for the entire paragraph.

4.3 Feature Selection

Input sentences are first trimmed of leading and trailing whitespace. Internal whitespace is consolidated, and the sentence is tokenized as described in Section 3. The resulting space tokens, if any, plus the wrapping tokens described above, are classified.

The core context of our model {*w*, *x*, *y*, *z*} is a window spanning two tokens to the left (positions *w* and *x*) and two tokens to the right (positions *y* and *z*) of a classification candidate space token.

For each of these window positions, the feature emitted, *c*, is either the Thai word itself at that position (for this it is convenient that Thai has no morphology), the space designator "sp," or one of several categorical designations for non-Thai tokens, a system which reflects the idea that, since foreign words are always surrounded by space, it is not necessary to dilute the feature space with foreign lemmas. If the second token of an input sentence is a space, "sp" is emitted for position *w* (and likewise for the second-to-last token, where "sp" is emitted for position *z*).

According to the first match among the following order, exactly one categorical, derived, or specific feature is emitted for each of {*w*, *x*, *y*, *z*}:

c	**token characteristic**
yk	Thai yamok (syllable reduplication) symbol
sp	all whitespace
๐๙	Thai numeric digits
num	Arabic numeric digits
ABC	ASCII with no lowercase
c*nn*	single character (derived from hex representation)
c*kkmmnn*	single character (derived from UTF8 and hex)
ascii	any amount of non-Thai text
(Thai text)	Thai word (feature name derived from actual text)

Table 1. Categorical and derived feature names

For Thai sentence-breaking, foreign-text tokens plus any intervening space are merged, so a single "ascii" feature may represent an arbitrary amount of non-Thai script with interior space.

Figure 9 shows an example sentence that has been tokenized. Token boundaries are indicated by slashes. Although there are three space tokens in the original input, we extract four contexts. The shaded line indicates the **sb** context that is synthesized by wrapping as described in Section 4.2.

/ลักษณะ/การ/อ้าง/อิง/แบบ/ /R1C1/ /ถูก/แปลง/ไป/เป็น/ลักษณะ/การ/อ้าง/อิง/แบบ/ /A1

b	c=*w*	*c=x*	*c=y*	*c=z*	*c=l*	*c=r*
nsb	อิง	แบบ	ABC	sp	5	1
nsb	sp	ABC	ถูก	แปลง	1	9
nsb	อิง	แบบ	ABC	sp	9	1
sb	sp	ABC	ลักษณะ	การ	1	5

Figure 9. A Thai sentence and the contexts extracted.

For each context, in addition to the {*w*, *x*, *y*, *z*} features, we extract two more features indicated by {*l* ,*r*} in Figure 9. They are the number of tokens between the previous space token (or beginning of text) and the current one, and the number of tokens between the current space token and the next space token (or end of text). These features do not distinguish whether the bounding space token is **sb** or **nsb**. This is because, processing left-to-right, it is permissible to use a feature such as "number of tokens since last **sb**," but not "number of tokens until next **sb**," which would be available during training but not at run-time.

In addition to the above core features, our model emits certain extra features only if they appear:

- An individual feature for each English punctuation mark, since these are sometimes used in Thai. For example, there is one feature *fs* for the sentence end period (i.e. full-stop) in English;
- The current nest depth for paired glyphs with directional variation, such as brackets, braces, and parentheses;
- The current parity value for paired glyphs without directional distinction such as "straight" quotation marks.

The following example illustrates paired directional glyphs (in this case, parentheses):

.../ยูนิลิเวอร์/ /(/ประเทศ/ /ไทย/)/ /จำกัด/ / เปิดเผย/ว่า/...
...Unilever (Thailand) Ltd. disclosed that...

b	c=*w*	*c=x*	*c=y*	*c=z*	*c=pn*
nsb	(	ประเทศ	ไทย	)	1

Figure 10. Text fragment illustrating paired directional glyphs

In Figure 10, the space between **ประเทศ** "country" and **ไทย** "Thai," generates an **nsb** context which includes the features shown, where "pn" is an extra feature which indicates the parenthesis nesting level. This feature helps the model learn that spaces which occur within parentheses are likely to be **nsb**.

Parity features for the non-directional paired glyphs, which do nest, are true binary features. Since these features have only two possible values (*inside* or *outside*), they are only emitted when their value is "inside," that is, when the space under consideration occurs between such a pair.

4.4 Sentence-Breaker Evaluation

Although evaluation against a single-domain corpus does not measure important design requirements of our system, namely resilience to broad-domain input texts, we evaluated against the Orchid corpus (Charoenporn et al. 1997) for the purpose of comparison with the existing literature. Following the methodology of the studies cited in Table 2, we test against 10% of ORCHID.

Our results are consistent with recent work using the Winnow algorithm (Charoenpornsawat and Sornlertlamvanich 2001), which itself compared favorably with probabilistic POS trigram approach (Mittrapiyanuruk and Sornlertlamvanich 2000). Both of these studies use evaluation metrics, attributed to Black and Taylor (1997), which aim to more usefully measure sentence-breaker utility. Accordingly, the following definitions are used in Table 2:

$$\text{space-correct} = \frac{(\#\text{correct sb}+\#\text{correct nsb})}{\text{total \# of space tokens}}$$

$$\text{false break} = \frac{\#\text{sb false positives}}{\text{total \# of space tokens}}$$

It was generally possible to reconstruct full precision and recall results from these published results[5] and we present a comprehensive table of results. Reconstructed values are marked with a dagger and the optimal result in each category is marked in boldface.

	Mittrapiyanuruk et al.	Charoenpornsawat et al.	Our result
method	**POS Trigram**	**Winnow**	**MaxEnt**
#sb in reference	10528	1086†	2133
#space tokens	33141	3801	7227
nsb-precision	90.27†	91.48†	**93.18**
nsb-recall	87.18†	**97.56**†	94.41
sb-precision	74.35†	**92.69**†	86.21
sb-recall	79.82	77.27	**83.50**
"space-correct"	85.26	89.13	**91.19**
"false-break"	8.75	**1.74**	3.94
F_2	83.10	88.97	**89.20**

Table 2. Evaluation of Thai Sentence Breakers against ORCHID

Finally, we would be remiss in not acknowledging the general hazard of assigning sentence breaks in a language such as Thai, where authors may intentionally include or omit spaces in order to create syntactic or semantic ambiguity. We defer to Mittrapiyanuruk and Sornlertlamvanich (2000) and Charoenpornsawat and Sornlertlamvanich (2001) for informed commentary on this topic.

5 Thai Text Re-Spacing

Much as the lack of intra-sentence space complicates Thai text input, it poses specific challenges when rendering Thai textual output. In English to Thai translation, word tokens output by our treelet translator must be adjoined with or without space in accordance with Thai convention when forming sentences.

5.1 Rule Classes

Wathabunditkul's (2003) rules for **nsb** insertion fall into the following four general categories:

- Rules dependant solely on Thai glyphs;
- Rules dependant on a combination of Thai glyphs and lexical/syntactic information;
- Rules dependant on notions that can be captured by syntax;
- Rules that are beyond lexical semantics, morphology and syntax.

The easiest guidelines to implement are those based solely on glyph/character type. For these we reference the lexical node in the Thai transfer tree. As we read nodes from the tree, we insert spaces before and after non-Thai words, digits (Thai or non-Thai), specific Thai glyphs such as repetition (ๆ) and omission (ฯ) marks, and marks such as plus signs, equal signs, and ampersands.

Examining only the lexical nodes in the Thai tree, however, does not get us very far in implementing spacing guidelines. Wathabunditkul's first guideline, hereafter the *phrase-final spacing guideline*, is to "add one space when you finish a phrase, clause or sentence, and wish to start a new idea." With a syntactic parser at hand, we are able to find where a phrase or clause ends, as discussed below.

5.2 Lexico-Semantic Transfer

Our English-Thai MT system contains no linguistic information about Thai. However, the English input to the system undergoes a complete syntactic analysis following lexical lookup and morphological analysis. In the MT transfer phase, the English parse tree (Figure 11) is transformed into both English and Thai dependency trees (Figures 12 and 13). Each node in the Thai tree is linked to a node in the English tree according to correspondences which hold across all trees (Table 3). Thus, for any word in a translation, there is ready access to information about the type of phrase or clause to which the corresponding English source word belongs, as well as its lexical properties.

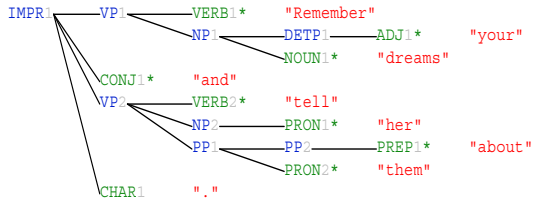


Figure 11. Parse tree result from the sentence "Remember your dreams and tell her about them."

[5] Full results for Charoenpornsawat et al. are reconstructed based on remarks in their text, including that "the ratio of the number of [nsb to sb] is about 5:2."

remember	จำ
dreams	ความฝัน
your	นของคุณ
and	และ
tell	บอก
her	เธอ
about	เกี่ยวกับ
them	เหล่านั้น

Table 3. English-Thai Correspondences.

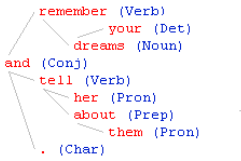
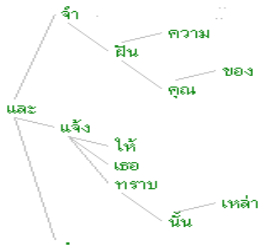


Figure 12. English dependency tree for "Remember your dreams and tell her about them."



จำความฝันของคุณ และบอกเธอเกี่ยวกับเหล่านั้น

Figure 13. Thai dependency tree for the translation of the sentence in Figure 12

As an initial exercise in implementing the phrase-final spacing guideline, we took a set of English sentences, analyzed them, produced Thai transfer trees, and read the Thai leaf nodes off the tree, adding a space after each word that ended the translation of an English source phrase/clause. The resulting set was evaluated by a native Thai reader who indicated that this produced many more spaces than were acceptable. The reason for this is that we had failed to implement what was perhaps the most important part of the guideline, namely, to add a space when wishing "to start a new idea." While translating "new idea" into code is problematic, we managed to capture this notion to an extent, as well as to implement many of the other spacing guidelines, by using the parser as described below.

In English text, ideas are often separated by punctuation. While one does not need a parser to identify, for example, an opening parenthesis, a parser certainly helps in distinguishing non-directional paired glyphs, such as "straight" quotation marks. A parser also helps us discriminate between the various functions of marks such as commas.

In addition to making it possible to implement the phrase-final spacing guideline, the parser helps us generalize. So, for example, instead of copying long lists of Thai words from the spacing guidelines into our rules, we access the lexico-semantic information for the English word corresponding to a Thai word in the translation and proceed with the fallible assumption that the Thai word has similar properties.

Below are four of Wathabunditkul's spacing guidelines (G) and our implementations (I) using information in the parse tree for the English input:

- G1: *One space between a series or set of words or phrases.*
  I1: One space before a Thai word that corresponds to a source coordinate conjunction joining more than two non-verbal conjuncts; one space before a Thai word that corresponds to a conjunction joining one or more verbal conjuncts; and one space to replace conjoining commas in these coordinate constructions. (Note that simply inserting a space before any word corresponding to a source coordinate conjunction breaks up far too many coordinate constructions that, in actuality, express a single idea and should not be interrupted by spaces in Thai.)

- G2: *One space after* ว่า *that is used in combination with such verbs of speech as* กล่าว *(including* กล่าวไว้ *and* ได้กล่าว*),* พูด, เห็น, รายงาน, แถลง, ยืนยัน, *etc.:*
  I2: One space after ว่า "that" when it introduces the translation of a complement clause of a source verb marked in the English dictionary as a speech act verb. (Note that by generalizing we don't need the list of Thai words expressed in the guideline, nor do we need to figure out how to implement "etc.")

- G3: *One space after the verbs of definition, for example:* หมายถึง, หมายความถึง, แปลว่า, คือ *and* กล่าวคือ.
  I3: One space before the beginning of the translation of a non-pronominal predicate nominative, providing that the translation of the linking verb "be" is คือ.

- G4: *One space after adverbial phrases* ทันใดนั้น, อย่างไรก็ตาม, อย่างไรก็ดี, กระนั้น, ทว่า, โดยส่วนตัวแล้ว, จะว่าไป, ในการนี้, ทั้งนี้, อนึ่ง, *etc.*
  I4: If a connective or time adverbial phrase (including prepositional phrases and subordinate clauses) ends in a comma, replace the comma with a space; otherwise insert a space at the end of the translation of the phrase.

5.3 Qualitative Evaluation

To evaluate our implementation of the guidelines, we translated a set of 5000 sentences, and separated out those that had foreign words or numbers in the translation. Among the remaining 1400 pure Thai translations, a native Thai linguist reviewed 150. Allowing that the translations themselves were not perfect, the linguist judged the spacing acceptable. Thus syntactic notions, such as adverbial phrase and coordinate construction, and lexico-semantic notions, such as speech

act verb and time adverb, enabled us to approximate Thai spacing conventions.

Further improvement appears to require analysis beyond lexical semantics. Guidelines such as "one space before and after the official name of a building" or one space after all clauses "when writing a long sentence," appeal to world knowledge and we leave these for future implementation.

6 Final Integration for Translation

For our English (ENG) to Thai (THA) translation system, we assembled three resources: a parallel training corpus, a development bitext (also called the lambda set) for training the feature combination weights λ_i, and a test corpus for BLEU and human evaluation.

Although it is well known that language translation pairs are not symmetric, due to the lack of additional corpora, we use these same resources to build our Thai-English translation system. We enhance our target language models (LM) with additional monolingual Thai and monolingual English data.

Our parallel MT corpus consists of approximately 725,000 English-Thai sentence pairs from various sources. Additionally we have 9.6 million Thai sentences, which are used to train a Thai 4-gram model for ENG-THA translation, together with the Thai sentences in the parallel corpus. All together, there are 125 million word tokens in training the Thai 4-gram LM. Trigrams and 4-grams that occur only once are pruned, and n-gram backoff weights are re-normalized after pruning, with the surviving KN smoothed probabilities intact (Kneser and Ney 1995).

Data Set	#Sentences
(THA,ENG) training	725K
(THA,ENG) lambda	2K
(THA,ENG) smoke	5K
THA LM text	10.3M
ENG LM text	45.6M

Table 4. Corpus size of parallel and monolingual data

For THA-ENG translation, the English 4-gram LM is trained on 982 million words. Our data sizes, in terms of the number of sentences, are indicated in Table 4. The lambda data set and the test set both have one gold translation per sentence. We evaluate our system based on 4-gram case-insensitive BLEU scores, after the final translation output is tokenized by our Thai tokenizer for ENG-THA output, or by our simple English word breaker for Thai-English output.

	Thai LM	English LM
#training words	125M	982M
Lexicon size	450K	3,127K
Lambda ppl	211	294
Test ppl	215	310
Lambda OOV	1.6%	0.7%
Test OOV	1.6%	0.7%

Table 5. Lexicon size and perplexity of language models

Table 5 shows the lambda-set and test-set perplexity of the Thai vs. the English LM, along with the unigram size. Words in the training data—but not in the unigram lexicon—are mapped to the <UNK> token, representing all out-of-vocabulary (OOV) words. Thai incurs higher OOV rates due to its smaller training set and thus smaller lexicon. Our systems defined the maximum phrase length to be 4 and the maximum re-ordering jump to be 4 as well. The BLEU score on the smoke set is shown in Table 6.

	BLEU
THA-ENG	0.213
ENG-THA	0.189

Table 6. 4-gram case-insensitive BLEU score

7 Future Work

Our adoption of the maximum entropy method for Thai sentence-breaking allowed us to achieve results which are consistent with contemporary state-of-the-art in this task. This general approach can be easily applied to a similar problem such as Arabic sentence-breaking. In Arabic writing, commas are often used to separate sentences until the end of a paragraph when a period is finally used. In this case, the comma character is similar to the space token in Thai where its usage is ambiguous. We can use the same approach (perhaps with different linguistic features) to identify which commas are sentence-breaking and which are not.

We have reviewed a range of independent solutions to problems in Thai text processing, including tokenization, sentence-breaking, and text re-spacing. We successfully integrated each solution into an existing large-scale SMT framework. There remains much room for improvement. We need to find or create true Thai-English directional corpora to train the lambdas and to test our models. The size of our parallel corpus for Thai should increase by at least an order of magnitude, without loss of bitext quality. Currently, sparsity concerns necessitate relaxing our training parameters; with a larger corpus, we can consider longer phrase length, higher-order n-grams, and longer re-ordering distance.

References

F. Avery Bishop, David C. Brown and David M. Meltzer. 2003. Supporting Multilanguage Text Layout and Complex Scripts with Windows 2000. *http://www.microsoft.com/typography/developers/uniscribe/intro.htm*

A. W. Black and P. Taylor. 1997. Assigning Phrase Breaks from Part-of-Speech Sequences. *Computer Speech and Language*, 12:99-117.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Dell Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19:263-311. Cambridge, MA: MIT Press.

Thatsanee Charoenporn, Virach Sornlertlamvanich, and Hitoshi Isahara. 1997. *Building A Thai Part-Of-Speech Tagged Corpus (ORCHID)*.

Paisarn Charoenpornsawat and Virach Sornlertlamvanich. 2001. Automatic sentence break disambiguation for Thai. In *International Conference on Computer Processing of Oriental Languages (ICCPOL)*, 231-235.

S. F. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, 310-318. Morristown, NJ: ACL.

J. N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470-1480.

Choochart Haruechaiyasak, Sarawoot Kongyoung, and Matthew N. Dailey. 2008. A Comparative Study on Thai Word Segmentation Approaches. In *Proceedings of ECTI-CON 2008*. Pathumthani, Thailand: ECTI.

Xiaodong He. 2007. Using Word Dependent Transition Models in HMM based Word Alignment for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, 80–87. Stroudsburg, PA: ACL.

Reinhard Kneser and Hermann Ney. 1995. Improved Backing-Off for M-Gram Language Modeling. In *Proceedings of International Conference on Acoustics, Speech and Signal Procesing (ICASSP)*, 1:181-184.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of the Association of Machine Translation in the Americas (AMTA-2004)*.

Canasai Kruengkrai, Virach Sornlertlamvanich, and Hitoshi Isahara. 2006. A Conditional Random Field Framework for Thai Morphological Analysis. *In Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*.

Arul Menezes, and Chris Quirk. 2008. Syntactic Models for Structural Word Insertion and Deletion during Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*.

P. Mittrapiyanuruk and V. Sornlertlamvanich. 2000. The Automatic Thai Sentence Extraction. In *Proceedings of the Fourth Symposium on Natural Language Processing*, 23-28.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg, PA: ACL.

David D. Palmer and Marti A. Hearst. 1997. Adaptive Multilingual Sentence Boundary Disambiguation. *Computational Linguistics*, 23:241-267.

Kishore Papineni, Salim Roukos , Todd Ward, and Wei-jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual meeting of the Association for Computational Linguistics*, 311–318. Stroudsburg, PA: ACL.

Yuen Pooworawan. 1986. Dictionary-based Thai Syllable Separation. *Proceedings of the Ninth Electronics Engineering Conference*, 409-418. Bangkok.

Adwait Ratnaparkhi, 1996. A Maximum Entropy Model for Part-of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 133-142.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries, In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 16-19.

Thanaruk Theeramunkong and Sasiporn Usanavasin. 2001. Non-Dictionary-Based Thai Word Segmentation Using Decision Trees. In *Proceedings of the First International Conference on Human Language Technology Research*.

Suphawut Wathabunditkul. 2003. Spacing in the Thai Language. *http://www.thailanguage.com/ref/spacing*