# Context Sensitive Pattern Based Segmentation: A Thai Challenge

**Petr Sojka** and **David Antoš**
Faculty of Informatics, Masaryk University Brno, Czech Republic
sojka@informatics.muni.cz xantos@informatics.muni.cz

## Abstract

A Thai written text is a string of symbols *without* explicit word boundary markup. A method for a development of a segmentation tool from a corpus of already segmented text is described. The methodology is based on the technology of *competing patterns*, evolved from algorithm for English hyphenation. A new UNICODE pattern generation program, OPATGEN, is used for the learning phase. We have shown feasibility of our methodology by generating patterns for Thai segmentation from already segmented text of the Thai corpus ORCHID. The algorithm recognizes almost 100% of word boundaries in the corpus and performs well on unseen text, too. We discuss the results and compare them to the conventional methods of segmenting Thai text. Finally, we enumerate possible new applications based on pattern technique, and conclude with the suggestion of a general Pattern Translation Process. The technology is general and can be used for any other segmentation tasks as phonetic, morphologic segmentation, word hyphenation, sentence segmentation and text topic segmentation for any language.

## 1 Motivation and Problem Description

*From Latin* segmentum, *from* secare *'to cut' (as term in geometry).*
—Origin of word *segmentation*: (Hanks, 1998)

Many natural language processing applications need to cut strings of letters, words or sentences into segments: phonetic, morphologic segmentation, word hyphenation, word phrase and sentence segmentation may serve as examples of this *segmentation task*. In Thai, Japanese, Korean and Chinese languages, where there are no explicit word boundaries in written texts, performing character stream segmentation is a crucial first step in the natural language processing of written texts. An elegant way of solving of this task is to learn the segmentation from already segmented corpus by a supervised machine learning technique.

### 1.1 Thai Segmentation Problem

A Thai paragraph is a string of symbols (44 consonants, 28 vowels). There are neither explicit syllable, word and sentence boundaries, nor punctuation in Thai text streams. For lexical, semantic analysis or typesetting, crucial first step is to find syllable, word and sentence boundaries. The Thai typesetting engine has to be able to segment the text in order to break lines automatically, too. Similarly, tools are needed to insert the `<wbr>` HTML tag automatically for the web browser rendering engine. A good word segmentation is a prerequisite for any Thai text processing including Part-of-Speech (POS) tagging (Murata et al., 2002).

### 1.2 Existing Approaches to Thai Segmentation

There is a program SWATH (Smart Word Analysis for THai) with three implemented dictionary based algorithms (longest matching, maximal matching, bigram model). It is used by the Thai Word-break Insertion service `http://ntl.nectec.or.th/services/www/thaiwordbreak.html` at NECTEC, the Thai National Electronics and Computer Technology Center. These methods have limited performance because of problems with handling of unknown words. There are other approaches based on the probabilistic language modelling (Sornlertlamvanich, 1998; Sukhahuta and Smith, 2001) or logically combined neural networks (Ma et al., 1996).

Mamoru and Satoshi (2001) reported that their Thai syllable recognizer, in which knowledge rules based on heuristics derived from the analysis of unsuccessful cases were adapted, gave a ratio of segmentation

of 93.9% in terms of sentences for the input of Thai text. The Thai text used was *Kot Mai Tra Sarm Duang* (Law of Three Seals), and had 20,631 sentences (Jaruskulchai, 1998, Chapter 3).

Feature based approach using RIPPER and Winnow learning algorithms is described in (Meknavin et al., 1997). Aroonmanakun (2002) recently reported approach based on trigram model of syllables and syllable merging, with very high precision and recall. His Thai word segmentation online service on `http://www.arts.chula.ac.th/~ling/wordseg/` is performed using maximum collocation approach.

All these attempts show the need and importance of highly efficient and quality solution of Thai word segmentation problem.

## 2 Patterns

> *Middle English* patron *'something serving as a model', from Old French. The change in sense is from the idea of a patron giving an example to be copied. Metathesis in the second syllable occured in the 16th century. By 1700* patron *ceased to be used on things, and the two forms became differentiated in sense.*
> —Origin of word *pattern*: (Hanks, 1998)

Patterns are used to recognise "points of interest" (segment boundaries) in data. The pattern is a sub-word of a given word set and the information of the points of interest is written between its symbols.

There are two possible values for this information. One value indicates the point of interest *is* here, the other indicates the point of interest *is not* here. Natural numbers are the typical representation of that knowledge; odd for yes, even for no. So we have *covering* and *inhibiting* patterns. Special symbols are often used, for example a dot for the word boundary. Patterns are as short as possible to store the information: context of variable length is modelled by this approach.

### 2.1 Competing Patterns

More formally, let us have an alphabet $\Sigma$. *Patterns* are words over the free monoid $\langle \Sigma, \cdot, \varepsilon \rangle$ where $\varepsilon$ is the empty word, and $\cdot$ (centered dot) is operation of *concatenation*. Let $\langle A, \leq \rangle$ be a partially ordered system, $\leq$ be a *lattice order* (every finite non-empty subset of $A$ has lower and upper bound). Let . be a distinguished symbol (dot) in $\Sigma' = \Sigma \cup \{.\}$ that denotes the beginning and the end of word: *begin of word marker* and *end of word marker*. *Classifying patterns* are the words over $\Sigma' \cup A$ such that the dot symbol is allowed at the beginning or end of patterns only.

Let $P$ be a set of patterns over $\Sigma' \cup A$ (*competing patterns*, *pattern base*). Let $w = w_1 w_2 \ldots w_n$ be a word to classify with $P$. Classification $\mathit{classify}(w, P) = a_0 w_1 a_1 w_1 \ldots w_n a_n$ of $w$ with respect to $P$ is computed from the pattern base $P$ by the following *competition*. All patterns whose projection to $\Sigma$ match substring of $w$ are collected. $a_i$ is the supremum of all values between characters $w_i$ and $w_{i+1}$ in matched patterns. $\mathit{classify}(w, P)$ is also called *winning pattern*. Winning pattern holds the definitive information (hyphenation, segmentation) about $w$ with respect to the pattern base $P$. There can be several pattern bases for the same $w$ that "compete" as well.

### 2.2 Example

An example of competing patterns used for hyphenation of English words is shown in Figure 1 on page 3. In this example the ordered system $\langle A, \leq \rangle$ used is for classification of candidates for hyphenation border is $\mathbb{N}$ (natural numbers). There are five pattern levels used in the example —Level `1`...Level `5`. There were eight patterns that matched the input (`1na`, `1tio`,...). Patterns in odd levels are *covering*, in an even levels *inhibiting*. Inhibiting patterns specify the exceptions with respect to the knowledge acquired in lower levels. Winner pattern is `.h0y3p0h0e2n5a4t2i0o0n.`: one sees that e.g. pattern `h e n5a t` *wins* over `n2a t`, thus possible segmentations are `hy-phen-ation`.

Competing pattern sets can be used on all levels of natural language processing—covering structures used in morphology, their exploration is seen on both syntax (parsing) and semantic (word sense discrimination) levels.

For the detailed definitions and more examples see (Liang, 1983; Sojka, 2000).

### 2.3 Comparison with Finite-State Approaches

Competing patterns technology can be viewed as one of finite-state approaches, with their pros and cons. Competing patterns extend the power of Finite-State Transducers (FST) similarly as adding the complement operator with respect to $\Sigma$. Ideally, instead of storing full FST, we make patterns that embody the same information in even more compact manner. Collecting patterns matching given word can be done in linear time, using trie data structure for pattern storage.

It has been shown that decomposition of the problem by using *local grammars* (Gross, 1997) or building cascades of Finite-State Machines (Hobbs et al., 1997) is a tractable, but very time-consuming task. Supervised learning methods for the induction of patterns from segmented text are needed.

### 2.4 Pattern Generation—PATGEN and OPATGEN Programs

Liang (1983) wrote a pattern generation program PATGEN for generation of hyphenation patterns from the list of already hyphenated words. The method for generation of patterns is not only independent of language

```
        h y p h e n a t i o n
Level 1           1n a
Level 1               1t i o
Level 2            n2a t
Level 2                 2i o
Level 2         h e2n
Level 3 .h y3p h
Level 4         h e n a4
Level 5         h e n5a t
        .h0y3p0h0e2n5a4t2i0o0n.
         h y-p h e n-a t i o n
```

Figure 1: Competing patterns and pattern levels for segmentation of English word `hyphenation`.

for which (hyphenation) patterns are generated, but of an application domain, too. PATGEN has been used for the preparation of quality hyphenation patterns for several dozens of languages (Sojka and Ševeček, 1995). A new enriched (UNICODE) version of PATGEN called OPATGEN, has been developed at Masaryk University Brno (Antoš and Sojka, 2001). The program opens new possibilities for pattern generation and new applications. The only thing that must be done to create patterns is to map the problem in mind to the alphabet used by OPATGEN (UNICODE). OPATGEN is based on separate PATLIB (Antoš, 2002) library, so even making a new special purpose frontend for a new application should be straightforward.

## 3 Thai Texts in ORCHID Corpus

*Literally 'free'.*
—Origin of word *Thai*: (Hanks, 1998)

There is a freely available corpus of already segmented Thai texts called ORCHID (Sornlertlamvanich et al., 1997). Snapshot of the corpus loaded in Emacs editor can be seen in Figure 2 on page 4. Parts of speech are tagged, too, using bootstrapping technique, manual editing and proofreading. There are 9,967 paragraphs in the corpus (6 MB in TIS-620 encoding).

Even native Thai speakers do not agree on the definition of the main notion—Thai word (Jaruskulchai, 1998) (problems appear whether a "compound word" should be considered as single entity or not). We have based our machine learning experiments purely on the data available in the ORCHID corpus, showing the power of the machine learning technique. We cannot comment on the quality of the corpus tagging, as we are not Thai native speakers.

The corpus consists of articles. Every article has headers containing meta-information, usually in Thai and English, followed by the text, consisting of paragraphs. Paragraphs are numbered and tagged with `#Pn` marks. Paragraphs contain sentences. The sentences are tagged with `#n`. Each sentence appears twice, first untagged. The second occurrence is tagged with part-of-speech tags. Each word is followed by the tag, eg., `/VACT` for active verb.

### 3.1 Corpus Preprocessing

In order to create patterns recognizing Thai word boundaries we had to pre-process the corpus. We used a simple Perl script. The word boundaries are marked in the second occurrences of sentences in the corpus. Therefore we cut out only the marked parts. The "points of interest" should be denoted with '-' sign for OPATGEN. We substituted all the part-of-speech tags with the minus signs. There are also text entities marked with single angle bracket tags, e.g., `<space>`. All of them act as word separators in the corpus and we also substituted them with our word boundary mark. That is also what we did with numbers, we silently removed them as there is no reason to encounter them into patterns. When applying patterns, numbers are trivially word boundaries.

Finally, we joined the whole paragraphs up. The places between sentences are also word boundaries. We decided not to join larger portions of the text (like several paragraphs or even articles) as we did not want the words OPATGEN had to deal with to be longer than hundreds of symbols. It would slow the pattern generation down and we would add only a bit of information, only the word boundaries that appear between words finishing and starting a paragraph. The preprocessed eight paragraphs from ORCHID (input for OPATGEN) look like this:

-การ-ประชุม-ทาง-วิชาการ-ครั้ง-ที่-
โครงการวิจัยและพัฒนา-อิเล็กทรอนิกส์-และ-
คอมพิวเตอร์-ปีงบประมาณ-เล่ม-
-ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ-
กระทรวงวิทยาศาสตร์-เทคโนโลยีและการพลังงาน-
-วัน-ที่-สิงหาคม-ห้องประชุม-ชั้น-
-สาร-

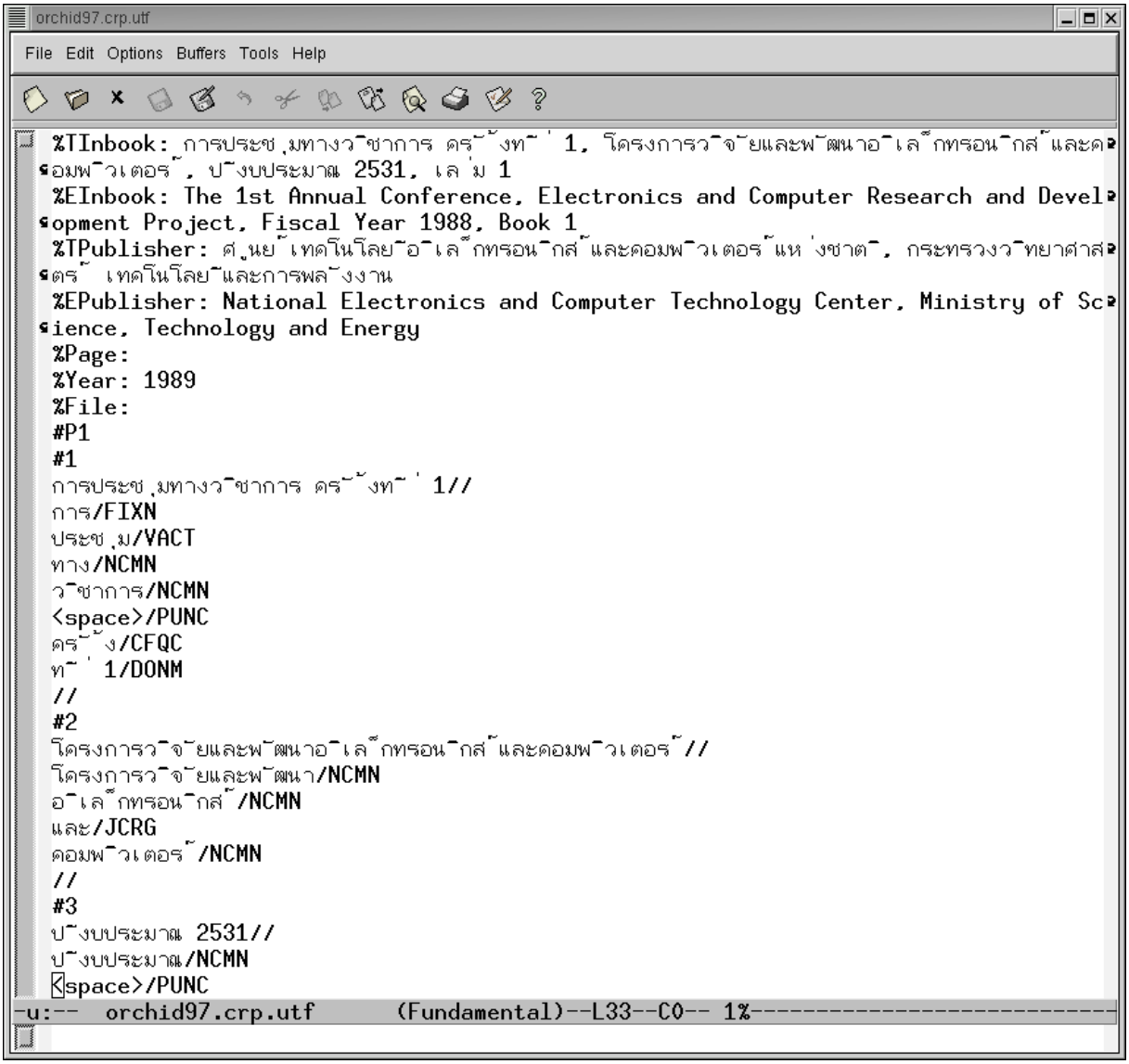


Figure 2: ORCHID loaded into Emacs.

-ฯพณฯ-รัฐมนตรีว่าการ-กระทรวงวิทยาศาสตร์-เทคโนโลยีและการพลังงาน-

-ประเทศไทย-ได้-มี-การ-ปรับเปลี่ยน-โครงสร้าง-ใน-การ-พัฒนา-เศรษฐกิจ-ของ-ประเทศ-จาก-ประเทศ-เกษตรกรรม-ไปสู่-ความ-เป็น-ประเทศอุตสาหกรรม-มาก-ยิ่งขึ้น-ใน-การ-ดำเนินการ-เพื่อให้-บรรลุ-วัตถุประสงค์-ดังกล่าว-จะ-ต้อง-อาศัย-ปัจจัยพื้นฐาน-หลาย-ประการ-ใน-การ-เป็น-ตัวเร่ง-และ-เป็น-ฐาน-เช่น-การ-พัฒนา-เทคโนโลยี-ที่-ใช้-ใน-การ-ผลิต-ของ-ภาคอุตสาหกรรม-กระทรวงวิทยาศาสตร์-เทคโนโลยีและการพลังงาน-จึง-ได้-ให้ความสำคัญ-เป็น-ลำดับ-สูง-ใน-การ-พัฒนา-อุตสาหกรรม-อิเล็กทรอนิกส์-และ-คอมพิวเตอร์-ซึ่ง-อุตสาหกรรม-นี้-จะ-มี-บทบาท-ที่-สำคัญ-มาก-ใน-ภาคอุตสาหกรรม-โดย-เป็น-ปัจจัยพื้นฐาน-หรือ-ส่วนประกอบ-ที่-สำคัญ-ของ-การ-ผลิต-ผลิตภัณฑ์อุตสาหกรรม-แทบ-ทุก-สาขา-

-ใน-ปลายปี-กระทรวงวิทยาศาสตร์-เทคโนโลยีและการพลังงาน-โดย-มติ-คณะ-รัฐมนตรี-ได้-จัดตั้ง-ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ-ขึ้น-เพื่อ-พัฒนา-เทคโนโลยี-อิเล็กทรอนิกส์-ใน-ประเทศ-และ-เทคโนโลยี-ด้าน-คอมพิวเตอร์-ทั้ง-ซอฟต์แวร์-และ-ฮาร์ดแวร์-โดย-การ-สนับสนุน-ให้-มี-การ-วิจัยและพัฒนา-ใน-ด้าน-นี้-จนถึง-ขั้น-พัฒนา-ผล-

ของ-การ-วิจัย-และ-พัฒนา-ไปสู่-การ-ผลิต-ใน-เชิง-อุตสาหกรรม-และ-พาณิชย์-ตลอดจน-สามารถ-แข่งขัน-ได้-ใน-ตลาด-ภายใน-และ-ต่างประเทศ-
. . .

## 4 Methodology

*Problems worthy of attack prove their worth by hitting back.*
—Piet Hein: Grooks

An important question is what kind of evaluation measures is most appropriate to compare the segmentation proposed by automated tools with the correct segmentations in the test set. A widely used evaluation scheme is the *PARSEVAL scheme*, based on the notions of *precision* and *recall*.

### 4.1 Evaluation Measures

Definition of the measures for our application is as follows:

$$\text{Precision} = \frac{\#\text{ found well}}{\#\text{ found well} + \#\text{ bad}} \quad (1)$$

$$\text{Recall} = \frac{\#\text{ found well}}{\#\text{ found well} + \#\text{ missed}} \quad (2)$$

Segment is correct if both the start and the end of the segment is correctly predicted.

The precision and recall scores are combined into a single measure, known as the *F-score* (Manning and Schütze, 1999):

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Another possibilities of an evaluation metric for segmentation are $P_k$ metric (Beeferman et al., 1997; Beeferman et al., 1999) or WindowDiff (Pevzner and Hearst, 2002). We do not use them, as they are appropriate for topic text segmentation, where small errors in positions of segment cuts are acceptable.

### 4.2 Experiments

We have divided the corpus into training set (3/5) and test set (2/5) and used the training set (6,000 paragraphs) for pattern generation. Ideally, we strive for smallest patterns solving the task with the highest F-score as possible. As general procedure how to achieve this goal is not known, parameters for the generation have been chosen after some trial and error (one has to find good-working thresholds for adding new patterns). Using the knowledge about threshold parameters used for the generation of hyphenation patterns we have quickly reached 100% precision (patterns were able to cover segmentation in training set without errors).

Parameters used for pattern generation are shown in Table 1 on page 6. In the second column, there are lengths of pattern candidates. A generation process can be parametrised by several parameters whose tuning strategies are beyond the scope of this paper; see (Sojka and Ševeček, 1995; Sojka, 1995) for details. Setting of the thresholds could be tuned so that virtually all hyphenation points are covered.

As there are quite long words in Thai (10 to 20-syllable word is not an exception), to achieve 100% precision, we may possibly need patterns as long as 20 characters to model long distance dependencies. This increases the time of pattern generation, but not above achievable level (it took half a day on Pentium 4 class PC).

The 'param' column contains the pattern choosing the rule weights. The percentages show the behaviour of the patterns on the corpus during generation. Finally, there is the number of patterns added in particular level and pattern size in kilobytes (coded in UTF-8 encoding). It is seen that most of the work is done by short patterns.

Next, we increased the training set to 8,000 paragraphs. Results are shown in Table 2 on page 6. Both precision and recall slightly increase with bigger training sets.

The behaviour of the patterns on data they were generated from does not show how they act on previously unseen data (generalization abilities). Therefore we tested performance on the test set (3,967 paragraphs). The obtained recall is above 90%. With the bigger training corpus we do get better performace measures as shown in Table 3 on page 7. From the main results given in this table follows that the the ORCHID Corpus is quite small for our task: given the bigger training corpus one would have even better performance.

Resulting 19424 patterns look like this:
.o1m .p1me .pre1p .s1f .s2mo .s1mp
.st1in .x1p .x1y .ก1ก .ก1ม .ก1ห
.การจ่าย3 .การ5พัฒนา .การพัฒนาระบบ5
.การพัฒนาโปรแกรม5ส .การรับ4 .การ1ว
.การ5ศึกษา .การออกแบบ5 .การออกแบบและ5
.การออกแบบและพัฒนา1 .การ5เริ่ม .ก่อนที่3
.คณะ5กรรมการนโยบาย. .คณะกรรมการ5บริหาร.
.คณะกรรมการอำนวยการ6 .คณะผู้5ทำ1 .คณะ3วิศ
.คณะ5อนุกรรมการท .คำ3ก .คำ1ภ .ง1ก .งาน1ค
.งาน1ฐ . . .

To sum up the properties of pattern technique, even with small data like ORCHID Corpus we have got 1:20 compression of the information stored and hidden there. The patterns can be trained to 100% precision on the training data and making essentially no error (one can always add the pattern for the whole paragraph). One can balance tradeoff between recall and precision measured on testing data. Moreover, the application of patterns is very efficient. Speed of the segmentation is

Table 1: Results of Thai segmentation patterns generation (6,000 paragraphs from ORCHID).

| level | length | param | % correct | % wrong | # patterns | UTF-8 size (kB) |
|---|---|---|---|---|---|---|
| 1 | 1–5 | 1 6 1 | 97.98 | 4.87 | +12907 | 130 |
| 2 | 2–6 | 1 1 1 | 96.83 | 0.69 | + 2091 | 156 |
| 3 | 3–11 | 1 3 1 | 99.58 | 0.82 | + 2578 | 204 |
| 4 | 4–12 | 4 1 1 | 97.83 | 0.03 | + 685 | 217 |
| 5 | 9–19 | 1 3 1 | 99.58 | 0.15 | + 1689 | 270 |
| 6 | 10–20 | 1 1 1 | 99.56 | 0.04 | + 119 | 274 |

Table 2: Results of Thai segmentation patterns generation (8,000 paragraphs from ORCHID).

| level | length | param | % correct | % wrong | # patterns | UTF-8 size (kB) |
|---|---|---|---|---|---|---|
| 1 | 1–5 | 1 6 1 | 97.92 | 4.86 | +15443 | 161 |
| 2 | 2–6 | 1 1 1 | 96.53 | 0.65 | + 2596 | 196 |
| 3 | 3–11 | 1 3 1 | 99.57 | 0.79 | + 3448 | 267 |
| 4 | 4–12 | 4 1 1 | 97.87 | 0.03 | + 953 | 286 |
| 5 | 9–19 | 1 3 1 | 99.68 | 0.12 | + 2468 | 364 |
| 6 | 10–20 | 1 1 1 | 99.67 | 0.04 | + 129 | 368 |

*linear* wrt. the length of the word we apply them on: in our case with length of the paragraph. It makes them one of the first choices in cases where processing speed is important. The speed of the segmentation using developed Thai patterns is at the range of 10,000 wps (words per second) on a Pentium 4 class PC. Memory consumption using compact digital trie implementation used in TEX for this performance is much below 0.3 MB.

The generation process may be optimized with respect to the resulting pattern size; the tradeoff among size, covering ratio, and error is adjustable. Nevertheless good patterns may be small in size and therefore applicable for handhelds, mobile phones and other small equipment. There is no reason for SMS's to have awfully broken words on the display of a cellular phone.

Creating patterns is possible due to availability of large tagged corpora. Technique of competing pattern generation might be useful for corpora builders as well. Thresholds set for pattern generation can be tuned up in such a way that highly improbable (bad?) segmentation points are not learned. This way, pattern generation process may serve as filter selecting possible errors in input corpus tagging. These errors are a traditional nightmare for anybody who deals with large experimental data. Creating patterns for a phenomenon appearing in the corpus thus may help to clean the errors when the error list reported by the generator is checked manually. The size of the error list may be tuned by the number of levels and by the setting the thresholds appropriately.

# 5 Data-Driven Approach Based on Competing Patterns

*If all you have is a hammer, everything looks like a nail.*
—Abraham Maslow

Let us comment on the technology of competing patterns from different points of view. The application of the techniques of bootstrapping and stratification (Sojka, 1995; Sojka, 1999) made it even more attractive.

## 5.1 Pattern Translation Processes

A process based on competing patterns that adds markup to the string of symbols is called *Pattern Translation Process (PTP)* (Antoš and Sojka, 2001). In the terminology of automata theory, it is special type of finite state transducer. With this finite state approach (Roche and Schabes, 1997), quite powerfull engines could be designed, with exceptional speed: time complexity of the PTP implementation based on digital tries is *linear* with respect to the input length (length of input sentence). Putting PTP's in a cascade, we still stay in linear time. In addition to PATLIB, there are quite efficient digital trie publicly available implementations as JUDY (Silverstein, 2002). Such PTP implementations are very memory efficient.

Although many natural language special purpose tools are being developed, their implementation using competing patterns technology with bootstrapping, stratification and pattern generation techniques (Sojka and Ševeček, 1995; Sojka, 1995; Sojka, 1999) is possible. We believe that in addition to the one of hardest problems—Thai segmentation—many other NLP

Table 3: Precision, recall, and F-score on unseen text.

| trained on # paragraphs | good | bad | missed | precision | recall | F-score |
|---|---|---|---|---|---|---|
| 4,000 | 139788 | 11231 | 15529 | 92.56% | 90.00% | 91.26 |
| 6,000 | 98243 | 7951 | 9432 | 92.51% | 91.24% | 91.87 |
| 8,000 | 46361 | 3358 | 3703 | 93.25% | 92.60% | 92.92 |

problems can be solved by our competing pattern data-driven approach. Let us add couple of notes about applications in the Computer Typesetting area.

### 5.2 Applications in Computer Typesetting

A good list of tough problems in the area of the computer typesetting, most of which are tractable by OPATGEN, is presented in (Haralambous and Plaice, 2001). A new typesetting system Ω (Haralambous and Plaice, 1997), gradually developed from the well known TEX typesetting system, is designed to be able to typeset text in all languages of the world. To solve typesetting problems that are not supported by the Ω engine itself external special purpose programs outside of Ω are invoked as so called external OTP's (Ω Translation Processes).

When we analyze most of the problems and application of computer typesetting described in (Haralambous and Plaice, 2001), we see that most of them could be formulated as string rewriting of regular languages with varying context length. They can be seen as translation processes that typically add information to a token (character or word) stream.

In the typesetting engine application, the main idea is the usage of pattern recognition in the middle of the digestive process of a typesetting engine. A cascade of PTP's is able to efficiently solve the hardest problems known sofar, in linear time, given the sets of competing patterns.

## 6 Conclusion and Future Work

*We are all apprentices in a craft*
*where no-one ever becomes a master.*
— Ernest Hemingway

We have shown the feasibility of technology of competing patterns to tackle the Thai word segmentation problem.

To evaluate next steps of the technology—bootstrapping and stratification techniques—we are looking for (native Thai) partners to pursue further research. Improving consistency of tagging of the corpus will even improve the system performance. Application for Thai sentence segmentation problem (Charoenpronsawat and Sornlertlamvanich, 2001) is straighforward, too, but a bigger corpus is needed for learning.

Having a tagged corpus freely available, one may try easier task of not only word segmentation, but syllable segmentation. This may be needed for typesetting engine to use, due to long Thai words. Most promising approach thus seems to be using competing patterns for syllable segmentation, and then parse the text upwards, merging syllables into words and words into sentences.

Another general questions remain open. How to set the OPATGEN parameters to get space-minimal, level-minimal, highest-precision, highest recall patterns given the data? We are still looking for rigorous theory for setting the parameters of the patern generation process. We also think of an automated pattern generation, performing the parameter setting using an expert system or statistical methods.

We also spend some effort on developing better generation strategies. Implicit strategy used by OPATGEN is basically brute-force testing of all reasonable pattern candidates. It is not straightforward how to optimize the process, but using bigram or trigram statistics of wordlist is an idea worth trying.

Choice of best fitting data structure for patterns needs further investigation, even though keeping the set of patterns is a general dictionary problem, studied for years by computer scientists. There are other approaches than those used in TEX, PATGEN and OPATGEN, namely packed dynamic tries LC-tries (Nilsson and Karlsson, 1999) and new digital trie library implementations like JUDY (Silverstein, 2002).

**Acknowledgement**

Support of the grant CEZ:J07/98:143300003 is acknowledged.

## References

David Antoš and Petr Sojka. 2001. Pattern Generation Revisited. In Simon Pepping, editor, *Proceedings of the 16th European TEX Conference, Kerkrade, 2001*, pages 7–17, Kerkrade, The Netherlands, Sep. NTG.

David Antoš. 2002. PATLIB, Pattern Manipulation Library. `http://www.fi.muni.cz/~xantos/patlib/`.

Wirote Aroonmanakun. 2002. Collocation and Thai Word Segmentation. In *Proceedings of SNLP-Oriental COCOSDA 2002*, pages 68–75.

Douglas Beeferman, Adam Berger, and John Lafferty. 1997. Text segmentation using exponential models.

In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pages 35–46, Providence, RI.

Douglas Beeferman, Adam Berger, and John Lafferty. 1999. Statistical Models of Text Segmentation. *Machine Learning*, 34(1–3):177–210.

Paisarn Charoenpronsawat and Virach Sornlertlamvanich. 2001. Automatic Sentence Break Disambiguation for Thai. In *Proceedings of ICCPOL 2001*, pages 231–235, May.

Maurice Gross. 1997. The Construction of Local Grammars. (Roche and Schabes, 1997), pages 329–354.

Patrick Hanks, editor. 1998. *The New Oxford Dictionary of English*. Oxford University Press, Oxford.

Yannis Haralambous and John Plaice. 1997. Methods for Processing Languages with Omega. In *Proceedings of the Second International Symposium on Multilingual Information Processing, Tsukuba, Japan*. available as `http://genepi.louis-jean.com/omega/tsukuba-methods97.pdf`.

Yannis Haralambous and John Plaice. 2001. Traitement automatique des langues et composition sous Omega. *Cahiers GUTenberg*, (39–40):139–166, May.

Jerry R. Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. 1997. FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. (Roche and Schabes, 1997), pages 383–406.

Chuleerat Jaruskulchai. 1998. *Automatic Indexing for Thai Text Retrieval*. Ph.D. thesis, School of Engineering and Applied Science, George Washington University, August.

Franklin M. Liang. 1983. *Word Hy-phen-a-tion by Com-put-er*. Ph.D. thesis, Department of Computer Science, Stanford University, August.

Qing Ma, Hitoshi Isahara, and Hiromi Ozaku. 1996. Automatic part-of-speech tagging of thai corpus neural networks. In *Lecture Notes in Computer Science 1112*, pages 275–280. Springer-Verlag.

Shibayama Mamoru and Hoshino Satoshi. 2001. Thai Morphological Analyses Based on the Syllable Formation Rules. *Journal of Information Procesing*, 15(04–007).

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Surapant Meknavin, Paisarn Charoenpornsawat, and Boonserm Kijsirikul. 1997. Feature-based Thai Word Segmentation. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS 1997)*, pages 41–46.

Masaki Murata, Qing Ma, and Hitoshi Isahara. 2002. Comparision of Three Machine-Learning Methods for Thai Part-of-Speech Tagging. *ACM Transactions on Asian Language Information Processing*, 1(2):145–158.

Stefan Nilsson and Gunnar Karlsson. 1999. IP-Address Lookup Using LC-Tries. *IEEE Journal on Selected Areas in Communications*, 17(6):1083–1092.

Lev Pevzner and Marti A. Hearst. 2002. A Critique and Improvement of an Evaluation Metric for Text Segmentation. *Computational Linguistics*, 28(1):19–36.

Emmanuel Roche and Yves Schabes. 1997. *Finite-State Language Processing*. MIT Press.

Alan Silverstein. 2002. Judy IV Shop Manual. `http://judy.sourceforge.net/application/shop_interm.pdf`.

Petr Sojka and Pavel Ševeček. 1995. Hyphenation in TEX—Quo Vadis? *TUGboat*, 16(3):280–289.

Petr Sojka. 1995. Notes on Compound Word Hyphenation in TEX. *TUGboat*, 16(3):290–297.

Petr Sojka. 1999. Hyphenation on Demand. *TUGboat*, 20(3):241–247.

Petr Sojka. 2000. Competing Patterns for Language Engineering. In Petr Sojka, Ivan Kopeček, and Karel Pala, editors, *Proceedings of the Third International Workshop on Text, Speech and Dialogue—TSD 2000*, Lecture Notes in Artificial Intelligence LNCS/LNAI 1902, pages 157–162, Brno, Czech Republic, Sep. Springer-Verlag.

Virach Sornlertlamvanich, Thatsanee Charoenporn, and Hitoshi Isahara. 1997. ORCHID: Thai Part-Of-Speech Tagged Corpus. Technical Report TR-NECTEC-1997-001, Thai National Electronics and Computer Technology Center, December. `http://www.links.nectec.or.th/`.

Virach Sornlertlamvanich. 1998. *Probabilistic Language Modeling for Generalized LR Parsing*. Ph.D. thesis, Department of Computer Science, Tokyo Institute of Technology, September.

Rattasit Sukhahuta and Dan Smith. 2001. Information Extraction Strategies for Thai Documents. *International Journal of Computer Processing of Oriental Languages (IJCPOL)*, 14(2):153–172.