Deterministic Word Segmentation Using Maximum Matching with Fully Lexicalized Rules

Manabu Sassano
Yahoo Japan Corporation
Midtown Tower, 9-7-1 Akasaka, Minato-ku, Tokyo 107-6211, Japan
msassano@yahoo-corp.jp

Abstract

We present a fast algorithm of word segmentation that scans an input sentence in a deterministic manner just one time. The algorithm is based on simple maximum matching which includes execution of fully lexicalized transformational rules. Since the process of rule matching is incorporated into dictionary lookup, fast segmentation is achieved. We evaluated the proposed method on word segmentation of Japanese. Experimental results show that our segmenter runs considerably faster than the state-of-the-art systems and yields a practical accuracy when a more accurate segmenter or an annotated corpus is available.

1 Introduction

The aim of this study is to improve the speed of word segmentation. Applications for many Asian languages including Chinese and Japanese require word segmentation. Such languages do not have explicit word delimiters such as white spaces. Word segmentation is often needed before every task of fundamental text processing such as counting words, searching for words, indexing documents, and extracting words. Therefore, the performance of word segmentation is crucial for these languages. Take for instance, information retrieval (IR) systems for documents in Japanese. It typically uses a morphological analyzer[1] to tokenize the content of the documents. One of the most time consuming tasks in IR systems is indexing, which uses morphological analysis intensively.

Major approaches to Japanese morphological analysis (MA) are based on methods of finding the best sequence of words along with their part-of-speech tags using a dictionary where they use the Viterbi search (e.g., (Nagata, 1994), (Kudo et al., 2004)). However, computation cost of modern MA systems is mainly attributed to the Viterbi search as Kaji et al. (2010) point out.

One of methods of improving the speed of MA or word segmentation will be to avoid or reduce the Viterbi search. We can avoid this by using maximum matching in the case of word segmentation. Since there are many applications such as IR and text classification, where part-of-speech tags are not mandatory, in this paper we focus on word segmentation and adopt maximum matching for it. However, maximum matching for Japanese word segmentation is rarely used these days because the segmentation accuracy is not good enough and the accuracy of MA is much higher. In this paper we investigate to improve the accuracy of maximum-matching based word segmentation while keeping speedy processing.

2 Segmentation Algorithm

Our algorithm is basically based on maximum matching, or longest matching (Nagata, 1997). Although maximum matching is very simple and easy to implement, a segmenter with this algorithm is not sufficiently accurate. For the purpose of improving the segmentation accuracy, several methods that can be combined with maximum matching have been examined. In previous studies (Palmer, 1997; Hockenmaier and Brew, 1998), the combination of maximum matching and character-based transformational rules has been investigated for Chinese. They have reported promising results in terms of accuracy and have not mentioned the running time of their methods, which might supposedly be very slow because we have to scan an input sentence many times to apply learned transformational rules.

In order to avoid such heavy post processing, we simplify the type of rules and incorporate the process of applying rules into a single process of maximum matching for dictionary lookup. We

[1] Japanese has a conjugation system in morphology and does not put white spaces between words. Therefore, we have to do morphological analysis in order to segment a given sentence into words and give an associated part-of-speech (POS) tag to each word. In the main stream of the research of Japanese language processing, morphological analysis has meant to be a joint task of segmentation and POS tagging.

```
Input: $c_i$: sentence which is represented as a character sequence. $N$: the number of characters in a given sentence. $t$: dictionary, the data structure of which should be a trie. $o_j$: map an ID $j$ to a single word or a sequence of words. $h_j$: total length of $o_j$.
Function: Lookup($t$, $c$, $i$, $N$): search the dictionary $t$ for the substring $c$ starting at the position $i$ up to $N$ by using maximum matching. This returns the ID of the entry in the dictionary $t$ when it matches and otherwise returns $-1$.
procedure Segment($c$, $N$, $t$)
var $i$: index of the character sequence $c$
var $j$: ID of the entry in the trie dictionary $t$
begin
  $i \leftarrow 1$
  while ($i \leq N$) do begin
    $j =$ Lookup($t$, $c$, $i$, $N$)
    if ($j = -1$) then
      { unknown word as a single character }
      print $c_i$ ; $i = i + 1$
    else
      print $o_j$ ; $i = i + h_j$
      { if $o_j$ is a sequence of words, print each word in the sequence with a delimiter. Otherwise print $o_j$ as a single token. }
    endif
    print delimiter
    { delimiter will be a space or something. }
  end
end
```

Figure 1: Algorithm of word segmentation with maximum matching that incorporates execution of transformational rules.

show in Figure 1 the pseudo code of the algorithm of word segmentation using maximum matching, where the combination of maximum matching and execution of simplified transformational rules is realized. If each of the data o_j in Figure 1 is a single token, the algorithm which is presented here is identical with segmentation by maximum matching.

We use the following types of transformational rules: $c_0 c_1 ... c_{l-1} c_l \rightarrow w_0 ... w_m$ where c_i is a character and w_j is a word (or morpheme). Below are sample rules for Japanese word segmentation:

- はないか (ha-na-i-ka) $\rightarrow$ は (ha; topic-marker) ない (na-i; "does not exist") か (ka; "or") [2]
- 大工学部 (dai-ko-gaku-bu) $\rightarrow$ 大 (dai; "university") 工学部 (ko-gaku-bu; "the faculty of engineering")

Note that the form of the left hand side of the rule is the sequence of characters, not the sequence of words. Due to this simplification, we can combine dictionary lookup by maximum matching with execution of transformational rules and make them into a single process. In other words, if we find a sequence of characters of the left hand side of a certain rule, then we write out the right hand side of the rule immediately. This construction enables us to naturally incorporate execution (or application) of transformational rules into dictionary-lookup, i.e., maximum matching.

Although the algorithm in Figure 1 does not specify the algorithm or the implementation of function Lookup(), a trie is suitable for the structure of the dictionary. It is known that an efficient implementation of a trie is realized by using a double-array structure (Aoe, 1989), which enables us to look up a given key at the $O(n)$ cost, where n is the length of the key. In this case the computation cost of the algorithm of Figure 1 is $O(n)$.

We can see in Figure 1 that the Viterbi search is not executed and the average number of dictionary lookups is fewer than the number of characters of an input sentence because the average length of words is longer than one. This contrasts with Viterbi-based algorithms of word segmentation or morphological analysis that always require dictionary lookup at each character position in a sentence.

3 Learning Transformational Rules

3.1 Framework of Learning

The algorithm in Figure 1 can be combined with rules learned from a reference corpus as well as hand-crafted rules. We used here a modified version of Brill's error-driven transformation-based learning (TBL) (Brill, 1995) for rule learning.

In our system, an initial system is a word segmenter that uses maximum matching with a given

[2] If we use simple maximum matching, i.e., with no transformational rules, to segment the samples here, we will get wrong segmentations as follows: はないか $\rightarrow$ はな (ha-na; "flower") いか (i-ka; "squid"), 大工学部 $\rightarrow$ 大工 (dai-ku; "carpenter") 学部 (gaku-bu; "faculty").

$w'_0 w'_1 \cdots w'_n \rightarrow w_0 w_1 \cdots w_m$
$L\, w'_0 w'_1 \cdots w'_n \rightarrow L\, w_0 w_1 \cdots w_m$
$w'_0 w'_1 \cdots w'_n\, R \rightarrow w_0 w_1 \cdots w_m\, R$
$L\, w'_0 w'_1 \cdots w'_n\, R \rightarrow L\, w_0 w_1 \cdots w_m\, R$

Table 1: Rule templates for error-driven learning

word list and words which occur in a given reference corpus (a training corpus). Our segmenter treats an unknown word, which is not in the dictionary, as a one-character word as shown in Figure 1.

3.2 Generating Candidate Rules

In order to generate candidate rules, first we compare the output of the current system with the reference corpus and extract the differences (Tashiro et al., 1994) as rules that have the following form: $L\, w'_0 w'_1 \cdots w'_n\, R \rightarrow L\, w_0 w_1 \cdots w_m\, R$ where $w'_0 w'_1 \cdots w'_n$ is a word sequence in the system output and $w_0 w_1 \cdots w_m$ is a word sequence in the reference corpus and L is a word in the left context and R is a word in the right context. After this extraction process, we generate four lexicalized rules from each extracted rule by using the templates defined in Table 1.

3.3 Learning Rules

In order to reduce huge computation when learning a rule at each iteration of TBL, we use some heuristic strategy. The heuristic score h is defined as: $h = f * (n + m)$ where f is a frequency of the rule in question and n is the number of words in $w'_0 w'_1 \cdots w'_n$ and m is the number of words in $w_0 w_1 \cdots w_m$. After sorting the generated rules associated with the score h, we apply each candidate rule in decreasing order of h and compute the error reduction. If we get positive reduction, we obtain this rule and incorporate it into the current dictionary and then proceed to the next iteration. If we do not find any rules that reduce errors, we terminate the learning process.

4 Experiments and Discussion

4.1 Corpora and an Initial Word List

In our experiments for Japanese we used the Kyoto University Text Corpus Version 4 (we call it KC4) (Kurohashi and Nagao, 2003), which includes newspaper articles, and 470M Japanese sentences (Kawahara and Kurohashi, 2006), which is compiled from the Web. For training, we used two sets of the corpus. The first set is the articles on January 1st through 8th (7,635 sentences) of KC4. The second one is 320,000 sentences that are selected from the 470M Web corpus. Note that the Web corpus is not annotated and we use it after word segmentation is given by JUMAN 6.0 (Kurohashi and Kawahara, 2007). The test data is a set of sentences in the articles on January 9th (1,220 sentences). The articles on January 10th were used for development.

We used all the words in the dictionary of JUMAN 6.0 as an initial word list. The number of the words in the dictionary is 542,061. They are generated by removing the grammatical information such as part-of-speech tags from the entries in the original dictionary of JUMAN 6.0.

4.2 Results and Discussion

Segmentation Performance We used word based F-measure and character-wise accuracy to evaluate the segmentation performance.

Table 2 shows comparison of various systems including ours. It is natural that since our system uses only fully lexicalized rules and does not use any generalized rules, it achieves a moderate performance. However, by using the Web corpus that contains 320,000 sentences, it yields an F-measure of near 0.96, which is at the same level as the F-measure of HMMs (baseline) in (Kudo et al., 2004, Table 3). We will discuss how we can improve it in a later section.

Segmentation Speed Table 3 shows comparison of the segmentation speed of various systems for 320,000 sentences of the Web corpus. Since, in general, such comparison is heavily dependent on the implementation of the systems, we have to be careful for drawing any conclusion. However, we can see that our system, which does not use the Viterbi search, achieved considerably higher processing speed than other systems.

Further Improvement The method that we have presented so far is based on lexicalized rules. That is, we do not have any generalized rules. The system does not recognize an unknown English word as a single token because most of such words are not in the dictionary and then are split into single letters. Similarly, a number that does not appear in the training corpus is split into digits.

It is possible to improve the presented method by incorporating relatively simple post-processing that concatenates Arabic numerals, numerals in

System	# of Sent.	F-measure	Char. Acc.	# of Rules
JUMAN 6.0	NA	0.9821	0.9920	NA
MeCab 0.98 w/ jumandic	7,958	0.9861	0.9939	NA
Ours w/o training corpus	0	0.8474	0.9123	0
Ours w/ KC4	7,635	0.9470	0.9693	2228
w/ Web320K	320,000	0.9555	0.9769	24267

Table 2: Performance summary of various systems and configurations. Jumandic for MeCab (Kudo et al., 2004) is stemmed from the dictionary of JUMAN.

System (Charset Encoding)	Model/Algorithm	Time (sec.)
JUMAN 6.0 (EUC-JP)	Markov model w/ hand-tuned costs	161.09
MeCab 0.98 (UTF-8) w/ jumandic	CRFs	13.71
KyTea 0.3.3 (UTF-8) w/ jumandic	Pointwise prediction w/ SVM	188.01
Ours (UTF-8)	Maximum matching w/ rules	**3.22**

Table 3: Running time on the Web320K corpus. We used a PC (Intel Xeon 2.33 GHz with 8GB memory on FreeBSD 6.3). The model for segmentation of KyTea (Neubig et al., 2011) in our experiments is trained with the word list of JUMAN on KC4 (see in Section 4.1).

System	F-measure
JUMAN 6.0	0.9821
MeCab 0.98 w/ jumandic	**0.9861**
KyTea 0.3.3 w/ jumandic	0.9789
MEMMs (Uchimoto et al., 2001)	0.9644
HMMs (Kudo et al., 2004, Table 3)	0.9622
Ours w/ KC4	0.9470
Ours w/ KC4 + post-proc.	0.9680
Ours w/ Web320K	0.9555
Ours w/ Web320K + post-proc.	**0.9719**

Table 4: Performance comparison to other systems.

kanji[3], Latin characters, and *katakana*[4] ones. This type of post processing is commonly used in Japanese morphological analysis. JUMAN and MeCab have a similar mechanism and use it.

As an additional experiment, we incorporated this post processing into our segmenter and measured the performance. The result is shown in Table 4. The segmenter with the post processing yields an F-measure of 0.9719 when it is trained on the 320k Web corpus. We observed that the performance gap between state-of-the-art systems such as JUMAN and MeCab and ours becomes smaller. Additional computation time was +10% for the post processing and this means the segmenter with the post processing is still much faster than other sophisticated MA systems. Many applications which have to process a huge amount of documents would gain the benefits from our proposed methods.

[3] *Kanji* in Japanese, or *hanzi* in Chinese, is a ideographic script. *Kanji* means Chinese characters.

[4] *Katakana* is one of the phonetic scripts used in Japanese. It is mainly used to denote loan words and onomatopoeias. Such type of words are very productive and are often unknown words in Japanese language processing.

5 Related Work

The use of transformational rules for improving word segmentation as well as morphological analysis is not new. It is found in previous work (Papageorgiou, 1994; Palmer, 1997; Hockenmaier and Brew, 1998; Gao et al., 2004). However, their approaches require the Viterbi search and/or a heavy post process such as cascaded transformation in order to rewrite the output of the base segmenter. This leads to slow execution and systems that incorporate such approaches have much higher cost of computation than ours.

6 Conclusion

We have proposed a new combination of maximum matching and fully lexicalized transformational rules. The proposed method allows us to carry out considerably faster word segmentation with a practically reasonable accuracy. We have evaluated the effectiveness of our method on corpora in Japanese. The experimental results show that we can combine our methods with either an existing morphological analyzer or a human-edited training corpus.

References

Jun-Ichi Aoe. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on Software Engineering*, 15(9):1066–1077.

Eric Brill. 1995. Transformation-based error driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Jianfeng Gao, Andi Wu, Cheng-Ning Huang, Hong qiao Li, Xinsong Xia, and Hauwei Qin. 2004. Adaptive Chinese word segmentation. In *Proc. of ACL-2004*, pages 462–469.

Julia Hockenmaier and Chris Brew. 1998. Error-driven learning of Chinese word segmentation. In *Proc. of PACLIC 12*, pages 218–229.

Nobuhiro Kaji, Yasuhiro Fujiwara, Naoki Yoshinaga, and Masaru Kitsuregawa. 2010. Efficient staggered decoding for sequence labeling. In *Proc. of ACL 2010*, pages 485–494.

Daisuke Kawahara and Sadao Kurohashi. 2006. Case frame compilation from the web using high-performance computing. In *Proc. of LREC 2006*, pages 1344–1347.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Appliying conditional random fields to Japanese morphological analysis. In *Proc. of EMNLP 2004*, pages 230–237.

Sadao Kurohashi and Daisuke Kawahara. 2007. JUMAN (a User-Extensible Morphological Analyzer for Japanese). `http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN`, `http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN`.

Sadao Kurohashi and Makoto Nagao. 2003. Building a Japanese parsed corpus. In Anne Abeille, editor, *Treebanks: Building and Using Parsed Corpora*, pages 249–260. Kluwer Academic Publishers.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proc. of COLING-94*, pages 201–207.

Masaaki Nagata. 1997. A self-organizing Japanese word segmenter using heuristic word identification and re-estimation. In *Proc. of WVLC-5*, pages 203–215.

Graham Neubig, Yosuke Nagata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proc. of ACL-2011*.

David D. Palmer. 1997. A trainable rule-based algorithm for word segmentation. In *Proc. of ACL-1997*, pages 321–328.

Constantine P. Papageorgiou. 1994. Japanese word segmentation by hidden Markov model. In *Proc. of HLT-1994*, pages 283–288.

Toshihisa Tashiro, Noriyoshi Uratani, and Tsuyoshi Morimoto. 1994. Restructuring tagged corpora with morpheme adjustment rules. In *Proc. of COLING-1994*, pages 569–573.

Kiyotaka Uchimoto, Satoshi Sekine, and Hitoshi Isahara. 2001. The unknown word problem: a morphological analysis of Japanese using maximum entropy aided by a dictionary. In *Proc. of EMNLP 2001*, pages 91–99.