Mostly-Unsupervised Statistical Segmentation of Japanese: Applications to Kanji

Rie Kubota Ando and Lillian Lee
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501
{kubotar,llee}@cs.cornell.edu

Abstract

Given the lack of word delimiters in written Japanese, word segmentation is generally considered a crucial first step in processing Japanese texts. Typical Japanese segmentation algorithms rely either on a lexicon and grammar or on pre-segmented data. In contrast, we introduce a novel statistical method utilizing *unsegmented* training data, with performance on kanji sequences comparable to and sometimes surpassing that of morphological analyzers over a variety of error metrics.

1 Introduction

Because Japanese is written without delimiters between words,[1] accurate *word segmentation* to recover the lexical items is a key step in Japanese text processing. Proposed applications of segmentation technology include extracting new technical terms, indexing documents for information retrieval, and correcting optical character recognition (OCR) errors (Wu and Tseng, 1993; Nagao and Mori, 1994; Nagata, 1996a; Nagata, 1996b; Sproat et al., 1996; Fung, 1998).

Typically, Japanese word segmentation is performed by morphological analysis based on lexical and grammatical knowledge. This analysis is aided by the fact that there are three types of Japanese characters, *kanji*, *hiragana*, and *katakana*: changes in character type often indicate word boundaries, although using this heuristic alone achieves less than 60% accuracy (Nagata, 1997).

Character sequences consisting solely of kanji pose a challenge to morphologically-based segmenters for several reasons. First and most importantly, kanji sequences often contain domain terms and proper nouns: Fung (1998) notes that 50-85% of the terms in various technical dictionaries are composed at least partly of kanji. Such words tend to be missing from general-purpose lexicons, causing an *unknown word* problem for morphological analyzers; yet, these terms are quite important for information retrieval, information extraction, and text summarization, making correct segmentation of these terms critical. Second, kanji sequences often consist of compound nouns, so grammatical constraints are not applicable. For instance, the sequence *sha-choh|ken|gyoh-mu|bu-choh* (president|and|business|general manager = "a president as well as a general manager of business") could be incorrectly segmented as: *sha-choh|ken-gyoh|mu|bu-choh* (president|subsidiary business|Tsutomu [a name]|general manager); since both alternatives are four-noun sequences, they cannot be distinguished by part-of-speech information alone. Finally, heuristics based on changes in character type obviously do not apply to kanji-only sequences.

Sequence length	# of characters	% of corpus
1 - 3 kanji	20,405,486	25.6
4 - 6 kanji	12,743,177	16.1
more than 6 kanji	3,966,408	5.1
Total	37,115,071	46.8

Figure 1: Statistics from 1993 Japanese newswire (NIKKEI), 79,326,406 characters total.

Although kanji sequences are difficult to segment, they can comprise a significant portion of Japanese text, as shown in Figure 1. Since sequences of more than 3 kanji generally consist of more than one word, at least 21.2% of 1993 Nikkei newswire consists of kanji sequences requiring segmentation. Thus, accuracy on kanji sequences is an important aspect of the total segmentation process.

As an alternative to lexico-grammatical and supervised approaches, we propose a simple, effi-

[1]The analogous situation in English would be if words were written without spaces between them.

cient segmentation method which learns mostly from very large amounts of *unsegmented* training data, thus avoiding the costs of building a lexicon or grammar or hand-segmenting large amounts of training data. Some key advantages of this method are:

- No Japanese-specific rules are employed, enhancing portability to other languages.
- A very small number of pre-segmented training examples (as few as 5 in our experiments) are needed for good performance, as long as large amounts of unsegmented data are available.
- For long kanji strings, the method produces results rivalling those produced by Juman 3.61 (Kurohashi and Nagao, 1998) and Chasen 1.0 (Matsumoto et al., 1997), two morphological analyzers in widespread use. For instance, we achieve 5% higher *word precision* and 6% better *morpheme recall.*

2 Algorithm

Our algorithm employs counts of character n-grams in an unsegmented corpus to make segmentation decisions. We illustrate its use with an example (see Figure 2).

Let "A B C D W X Y Z" represent an eight-kanji sequence. To decide whether there should be a word boundary between D and W, we check whether n-grams that are adjacent to the proposed boundary, such as the 4-grams s_1 ="A B C D" and s_2 ="W X Y Z", tend to be more frequent than n-grams that straddle it, such as the 4-gram t_1 = "B C D W". If so, we have evidence of a word boundary between D and W, since there seems to be relatively little cohesion between the characters on opposite sides of this gap.

The n-gram orders used as evidence in the segmentation decision are specified by the set N. For instance, if $N = \{4\}$ in our example, then we pose the six questions of the form, "Is $\#(s_i) > \#(t_j)$?", where $\#(x)$ denotes the number of occurrences of x in the (unsegmented) training corpus. If $N = \{2, 4\}$, then two more questions (Is "#(C D) > #(D W)?" and "Is #(W X) > #(D W)?") are added.

More formally, let s_1^n and s_2^n be the non-straddling n-grams just to the left and right of location k, respectively, and let t_j^n be the straddling n-gram with j characters to the right of location k.

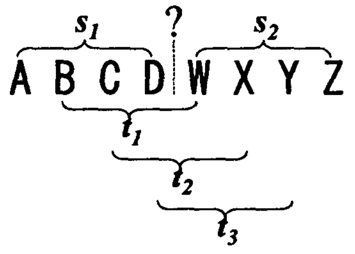

Figure 2: Collecting evidence for a word boundary – are the non-straddling n-grams s_1 and s_2 more frequent than the straddling n-grams t_1, t_2, and t_3?

Let $I_>(y, z)$ be an indicator function that is 1 when $y > z$, and 0 otherwise.[2] In order to compensate for the fact that there are more n-gram questions than $(n-1)$-gram questions, we calculate the fraction of affirmative answers separately for each n in N:

$$v_n(k) = \frac{1}{2(n-1)} \sum_{i=1}^{2} \sum_{j=1}^{n-1} I_>(\#(s_i^n), \#(t_j^n))$$

Then, we average the contributions of each n-gram order:

$$v_N(k) = \frac{1}{|N|} \sum_{n \in N} v_n(k)$$

After $v_N(k)$ is computed for every location, boundaries are placed at all locations ℓ such that either:

- $v_N(\ell) > v_N(\ell - 1)$ and $v_N(\ell) > v_N(\ell + 1)$ (that is, ℓ is a local maximum), or
- $v_N(\ell) \geq t$, a threshold parameter.

The second condition is necessary to allow for single-character words (see Figure 3). Note that it also controls the granularity of the segmentation: low thresholds encourage shorter segments.

Both the count acquisition and the testing phase are efficient. Computing n-gram statistics for all possible values of n simultaneously can be done in $O(m \log m)$ time using suffix arrays, where m is the training corpus size (Manber and Myers, 1993; Nagao and Mori, 1994). However, if the set N of n-gram orders is known in advance, conceptually simpler algorithms suffice. Memory allocation for

[2]Note that we do not take into account the magnitude of the difference between the two frequencies; see section 5 for discussion.

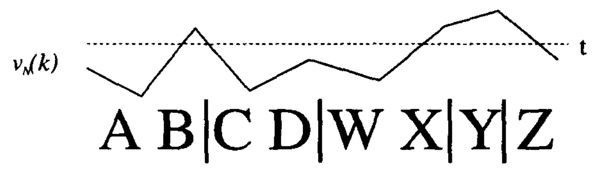

Figure 3: Determining word boundaries. The X- Y boundary is created by the threshold criterion, the other three by the local maximum condition.

count tables can be significantly reduced by omitting n-grams occurring only once and assuming the count of unseen n-grams to be one. In the application phase, the algorithm is clearly linear in the test corpus size if $|N|$ is treated as a constant.

Finally, we note that some pre-segmented data is necessary in order to set the parameters N and t. However, as described below, very little such data was required to get good performance; we therefore deem our algorithm to be "mostly unsupervised".

3 Experimental Framework

Our experimental data was drawn from 150 megabytes of 1993 Nikkei newswire (see Figure 1). Five 500-sequence *held-out* subsets were obtained from this corpus, the rest of the data serving as the unsegmented corpus from which to derive character n-gram counts. Each held-out subset was hand-segmented and then split into a 50-sequence parameter-training set and a 450-sequence test set. Finally, any sequences occurring in both a test set and its corresponding parameter-training set were discarded from the parameter-training set, so that these sets were disjoint. (Typically no more than five sequences were removed.)

3.1 Held-out set annotation

Each held-out set contained 500 randomly-extracted kanji sequences at least ten characters long (about twelve on average), lengthy sequences being the most difficult to segment (Takeda and Fujisaki, 1987). To obtain the gold-standard annotations, we segmented the sequences by hand, using an observation of Takeda and Fujisaki (1987) that many kanji compound words consist of two-character *stem* words together with one-character prefixes and suffixes. Using this terminology, our two-level bracketing annotation may be summarized as follows.[3] At the *word level*, a stem and its affixes are bracketed together as a single unit. At the *morpheme level*, stems are divided from their affixes. For example, although both *naga-no* (Nagano) and *shi* (city) can appear as individual words, *naga-no-shi* (Nagano city) is bracketed as [[*naga-no*][*shi*]], since here *shi* serves as a suffix. Loosely speaking, word-level bracketing demarcates discourse entities, whereas morpheme-level brackets enclose strings that cannot be further segmented without loss of meaning.[4] For instance, if one segments *naga-no* in *naga-no-shi* into *naga* (long) and *no* (field), the intended meaning disappears. Here is an example sequence from our datasets:

[小学校][屋内][[運動][場]][建設]

Three native Japanese speakers participated in the annotation: one segmented all the held-out data based on the above rules, and the other two reviewed 350 sequences in total. The percentage of agreement with the first person's bracketing was 98.42%: only 62 out of 3927 locations were contested by a verifier. Interestingly, all disagreement was at the morpheme level.

3.2 Baseline algorithms

We evaluated our segmentation method by comparing its performance against Chasen 1.0[5] (Matsumoto et al., 1997) and Juman 3.61,[6] (Kurohashi and Nagao, 1998), two state-of-the-art, publically-available, user-extensible morphological analyzers. In both cases, the grammars were used as distributed without modification. The sizes of Chasen's and Juman's default lexicons are approximately 115,000 and 231,000 words, respectively.

Comparison issues An important question that arose in designing our experiments was how to enable morphological analyzers to make use of the parameter-training data, since they do not have parameters to tune. The only significant way that they can be updated is by changing their grammars or lexicons, which is quite tedious (for instance, we had to add part-of-speech information to new entries by hand). We took what we felt to be a reasonable, but not too time-consuming, course of creating new lexical entries for all the bracketed words in the parameter-training data. Evidence that this

[3]A complete description of the annotation policy, including the treatment of numeric expressions, may be found in a technical report (Ando and Lee, 1999).

[4]This level of segmentation is consistent with Wu's (1998) *Monotonicity Principle* for segmentation.

[5]http://cactus.aist-nara.ac.jp/lab/nlt/chasen.html

[6]http://pine.kuee.kyoto-u.ac.jp/nl-resource/juman-e.html

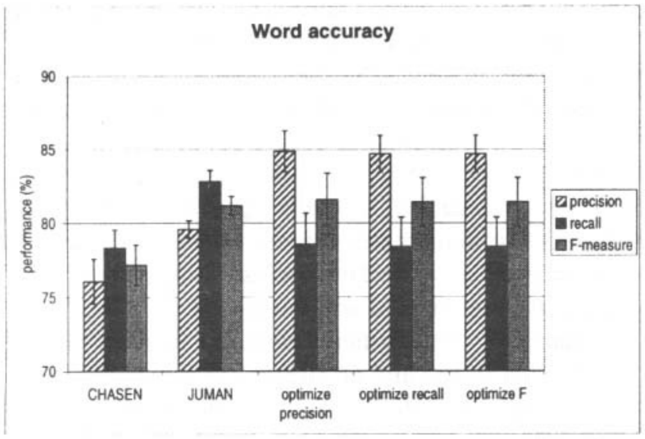

Figure 4: Word accuracy. The three rightmost groups represent our algorithm with parameters tuned for different optimization criteria.

was appropriate comes from the fact that these additions never degraded test set performance, and indeed improved it by one percent in some cases (only small improvements are to be expected because the parameter-training sets were fairly small).

It is important to note that in the end, we are comparing algorithms with access to different sources of knowledge. Juman and Chasen use lexicons and grammars developed by human experts. Our algorithm, not having access to such pre-compiled knowledge bases, must of necessity draw on other information sources (in this case, a very large unsegmented corpus and a few pre-segmented examples) to compensate for this lack. Since we are interested in whether using simple statistics can match the performance of labor-intensive methods, we do not view these information sources as conveying an unfair advantage, especially since the annotated training sets were small, available to the morphological analyzers, and disjoint from the test sets.

4 Results

We report the average results over the five test sets using the optimal parameter settings for the corresponding training sets (we tried all nonempty subsets of $\{2, 3, 4, 5, 6\}$ for the set of n-gram orders N and all values in $\{.05, .1, .15, \ldots, 1\}$ for the threshold t)[7]. In all performance graphs, the "error bars" represent one standard deviation. The results for Chasen and Juman reflect the lexicon additions described in section 3.2.

Word and morpheme accuracy The standard metrics in word segmentation are word precision and recall. Treating a proposed segmentation as a non-nested bracketing (e.g., "|AB|C|" corresponds to the bracketing "[AB][C]"), *word precision* (P) is defined as the percentage of proposed brackets that exactly match word-level brackets in the annotation; *word recall* (R) is the percentage of word-level annotation brackets that are proposed by the algorithm in question; and *word F* combines precision and recall: $F = 2PR/(P + R)$.

One problem with using word metrics is that morphological analyzers are designed to produce morpheme-level segments. To compensate, we altered the segmentations produced by Juman and Chasen by concatenating stems and affixes, as identified by the part-of-speech information the analyzers provided. (We also measured morpheme accuracy, as described below.)

Figures 4 and 8 show word accuracy for Chasen, Juman, and our algorithm for parameter settings optimizing word precision, recall, and F-measure rates. Our algorithm achieves 5.27% higher precision and 0.26% better F-measure accuracy than Juman, and does even better (8.8% and 4.22%, respectively) with respect to Chasen. The recall performance falls (barely) between that of Juman and that of Chasen.

As noted above, Juman and Chasen were designed to produce morpheme-level segmentations. We therefore also measured *morpheme precision, recall*, and *F measure*, all defined analogously to their word counterparts.

Figure 5 shows our morpheme accuracy results. We see that our algorithm can achieve better recall (by 6.51%) and F-measure (by 1.38%) than Juman, and does better than Chasen by an even wider margin (11.18% and 5.39%, respectively). Precision was generally worse than the morphological analyzers.

Compatible Brackets Although word-level accuracy is a standard performance metric, it is clearly very sensitive to the test annotation. Morpheme accuracy suffers the same problem. Indeed, the authors of Juman and Chasen may well have constructed their standard dictionaries using different notions of word and morpheme than the definitions we used in annotating the data. We therefore developed two new, more robust metrics to measure the number of proposed brackets that would be incor-

[7]For simplicity, ties were deterministically broken by preferring smaller sizes of N, shorter n-grams in N, and larger threshold values, in that order.

`[[data][base]][system]` **(annotation brackets)**

Proposed segmentation	word errors	morpheme errors	compatible-bracket errors: crossing	compatible-bracket errors: morpheme-dividing
`[data][base] [system]`	2	0	0	0
`[data][basesystem]`	2	1	1	0
`[database] [sys][tem]`	2	3	0	2

Figure 6: Examples of word, morpheme, and compatible-bracket errors. The sequence "data base" has been annotated as "[[data][base]]" because "data base" and "database" are interchangeable.

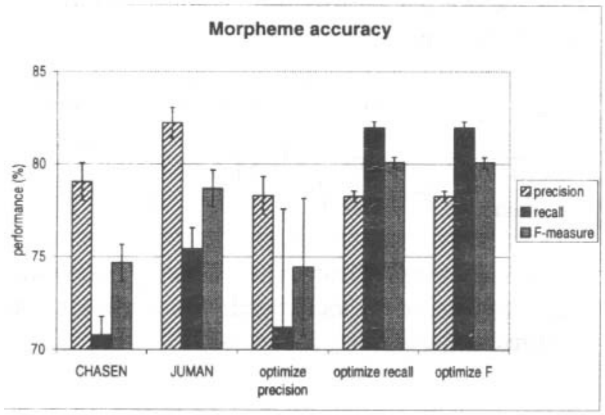

Figure 5: Morpheme accuracy.

rect with respect to *any* reasonable annotation.

Our novel metrics account for two types of errors. The first, a *crossing bracket*, is a proposed bracket that overlaps but is not contained within an annotation bracket (Grishman et al., 1992). Crossing brackets cannot coexist with annotation brackets, and it is unlikely that another human would create such brackets. The second type of error, a *morpheme-dividing bracket*, subdivides a morpheme-level annotation bracket; by definition, such a bracket results in a loss of meaning. See Figure 6 for some examples.

We define a *compatible bracket* as a proposed bracket that is neither crossing nor morpheme-dividing. The *compatible brackets rate* is simply the compatible brackets precision. Note that this metric accounts for different levels of segmentation simultaneously, which is beneficial because the granularity of Chasen and Juman's segmentation varies from morpheme level to compound word level (by our definition). For instance, well-known university names are treated as single segments by virtue of being in the default lexicon, whereas other university names are divided into the name and the word "university". Using the compatible brackets rate, both segmentations can be counted as correct.

We also use the *all-compatible brackets rate*, which is the fraction of sequences for which *all* the proposed brackets are compatible. Intuitively, this function measures the ease with which a human could correct the output of the segmentation algorithm: if the all-compatible brackets rate is high, then the errors are concentrated in relatively few sequences; if it is low, then a human doing post-processing would have to correct many sequences.

Figure 7 depicts the compatible brackets and all-compatible brackets rates. Our algorithm does better on both metrics (for instance, when F-measure is optimized, by 2.16% and 1.9%, respectively, in comparison to Chasen, and by 3.15% and 4.96%, respectively, in comparison to Juman), regardless of training optimization function (word precision, recall, or F — we cannot directly optimize the compatible brackets rate because "perfect" performance is possible simply by making the entire sequence a single segment).

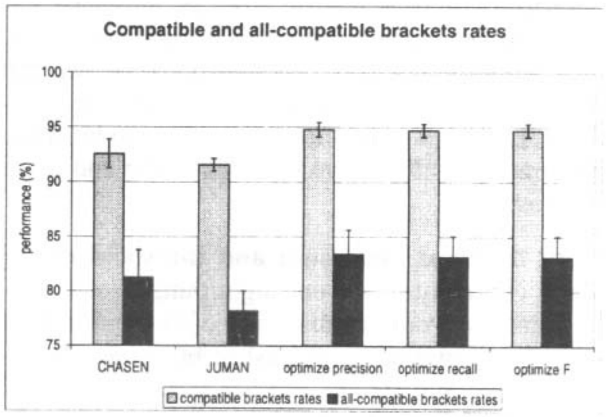

Figure 7: Compatible brackets and all-compatible bracket rates when word accuracy is optimized.

	Juman5 vs. Juman50	Our50 vs Juman50	Our5 vs. Juman5	Our5 vs. Juman50
precision	-1.04	+5.27	+6.18	+5.14
recall	-0.63	-4.39	-3.73	-4.36
F-measure	-0.84	+0.26	+1.14	+0.30

Figure 8: Relative word accuracy as a function of training set size. "5" and "50" denote training set size *before* discarding overlaps with the test sets.

4.1 Discussion

Minimal human effort is needed. In contrast to our mostly-unsupervised method, morphological analyzers need a lexicon and grammar rules built using human expertise. The workload in creating dictionaries on the order of hundreds of thousands of words (the size of Chasen's and Juman's default lexicons) is clearly much larger than annotating the small parameter-training sets for our algorithm. We also avoid the need to segment a large amount of parameter-training data because our algorithm draws almost all its information from an unsegmented corpus. Indeed, the only human effort involved in our algorithm is pre-segmenting the five 50-sequence parameter training sets, which took only 42 minutes. In contrast, previously proposed supervised approaches have used segmented training sets ranging from 1000-5000 sentences (Kashioka et al., 1998) to 190,000 sentences (Nagata, 1996a).

To test how much annotated training data is actually necessary, we experimented with using miniscule parameter-training sets: five sets of only *five* strings each (from which any sequences repeated in the test data were discarded). It took only 4 minutes to perform the hand segmentation in this case. As shown in Figure 8, relative word performance was *not degraded* and sometimes even slightly better. In fact, from the last column of Figure 8 we see that even if our algorithm has access to only five annotated sequences when Juman has access to ten times as many, we still achieve better precision and better F measure.

Both the local maximum and threshold conditions contribute. In our algorithm, a location k is deemed a word boundary if $v_N(k)$ is either (1) a local maximum or (2) at least as big as the threshold t. It is natural to ask whether we really need two conditions, or whether just one would suffice.

We therefore studied whether optimal performance could be achieved using only one of the conditions. Figure 9 shows that in fact both contribute to producing good segmentations. Indeed, in some cases, both are needed to achieve the best performance; also, each condition when used in isolation yields suboptimal performance with respect to some performance metrics.

accuracy	optimize precision	optimize recall	optimize F-measure
word	M	M & T	M
morpheme	M & T	T	T

Figure 9: Entries indicate whether best performance is achieved using the local maximum condition (M), the threshold condition (T), or both.

5 Related Work

Japanese Many previously proposed segmentation methods for Japanese text make use of either a pre-existing lexicon (Yamron et al., 1993; Matsumoto and Nagao, 1994; Takeuchi and Matsumoto, 1995; Nagata, 1997; Fuchi and Takagi, 1998) or pre-segmented training data (Nagata, 1994; Papageorgiou, 1994; Nagata, 1996a; Kashioka et al., 1998; Mori and Nagao, 1998). Other approaches bootstrap from an initial segmentation provided by a baseline algorithm such as Juman (Matsukawa et al., 1993; Yamamoto, 1996).

Unsupervised, non-lexicon-based methods for Japanese segmentation do exist, but they often have limited applicability. Both Tomokiyo and Ries (1997) and Teller and Batchelder (1994) explicitly avoid working with kanji charactes. Takeda and Fujisaki (1987) propose the *short unit model*, a type of Hidden Markov Model with linguistically-determined topology, to segment kanji compound words. However, their method does not handle three-character stem words or single-character stem words with affixes, both of which often occur in proper nouns. In our five test datasets, we found that 13.56% of the kanji sequences contain words that cannot be handled by the short unit model.

Nagao and Mori (1994) propose using the heuristic that high-frequency character n-grams may represent (portions of) new collocations and terms, but the results are not experimentally evaluated, nor is a general segmentation algorithm proposed. The work of Ito and Kohda (1995) similarly relies on high-frequency character n-grams, but again, is more concerned with using these frequent n-grams as pseudo-lexicon entries; a standard segmentation algorithm is then used on the basis of the induced lexicon. Our algorithm, on the hand, is fundamentally different in that it incorporates no explicit notion of word, but only "sees" locations between characters.

Chinese According to Sproat et al. (1996), most prior work in Chinese segmentation has exploited lexical knowledge bases; indeed, the authors assert that they were aware of only one previously published instance (the mutual-information method of Sproat and Shih (1990)) of a purely statistical approach. In a later paper, Palmer (1997) presents a transformation-based algorithm, which requires pre-segmented training data.

To our knowledge, the Chinese segmenter most similar to ours is that of Sun et al. (1998). They also avoid using a lexicon, determining whether a given location constitutes a word boundary in part by deciding whether the two characters on either side tend to occur together; also, they use thresholds and several types of local minima and maxima to make segmentation decisions. However, the statistics they use (mutual information and t-score) are more complex than the simple n-gram counts that we employ.

Our preliminary reimplementation of their method shows that it does not perform as well as the morphological analyzers on our datasets, although we do not want to draw definite conclusions because some aspects of Sun et al's method seem incomparable to ours. We do note, however, that their method incorporates numerical differences between statistics, whereas we only use indicator functions; for example, once we know that one trigram is more common than another, we do not take into account the difference between the two frequencies. We conjecture that using absolute differences may have an adverse effect on rare sequences.

6 Conclusion

In this paper, we have presented a simple, mostly-unsupervised algorithm that segments Japanese sequences into words based on statistics drawn from a large unsegmented corpus. We evaluated performance on kanji with respect to several metrics, including the novel compatible brackets and all-compatible brackets rates, and found that our algorithm could yield performances rivaling that of lexicon-based morphological analyzers.

In future work, we plan to experiment on Japanese sentences with mixtures of character types, possibly in combination with morphological analyzers in order to balance the strengths and weaknesses of the two types of methods. Since our method does not use any Japanese-dependent heuristics, we also hope to test it on Chinese or other languages as well.

Acknowledgments

We thank Minoru Shindoh and Takashi Ando for reviewing the annotations, and the anonymous reviewers for their comments. This material was supported in part by a grant from the GE Foundation.

References

Rie Ando and Lillian Lee. 1999. Unsupervised statistical segmentation of Japanese kanji strings. Technical Report TR99-1756, Cornell University.

Takeshi Fuchi and Shinichiro Takagi. 1998. Japanese morphological analyzer using word co-occurrence - JTAG. In *Proc. of COLING-ACL '98*, pages 409–413.

Pascale Fung. 1998. Extracting key terms from Chinese and Japanese texts. *Computer Processing of Oriental Languages*, 12(1).

Ralph Grishman, Catherine Macleod, and John Sterling. 1992. Evaluating parsing strategies using standardized parse files. In *Proc. of the 3rd ANLP*, pages 156–161.

Akinori Ito and Kasaki Kohda. 1995. Language modeling by string pattern N-gram for Japanese speech recognition. In *Proc. of ICASSP*.

Hideki Kashioka, Yasuhiro Kawata, Yumiko Kinjo, Andrew Finch, and Ezra W. Black. 1998. Use of mutual information based character clusters in dictionary-less morphological analysis of Japanese. In *Proc. of COLING-ACL '98*, pages 658–662.

Sadao Kurohashi and Makoto Nagao. 1998. Japanese morphological analysis system JUMAN version 3.6 manual. In Japanese.

Udi Manber and Gene Myers. 1993. Suffix arrays:

A new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

T. Matsukawa, Scott Miller, and Ralph Weischedel. 1993. Example-based correction of word segmentation and part of speech labelling. In *Proc. of the HLT Workshop*, pages 227–32.

Yuji Matsumoto and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proc. of the International Workshop on Sharable Natural Language Resources*, pages 22–28.

Yuji Matsumoto, Akira Kitauchi, Tatsuo Yamashita, Yoshitaka Hirano, Osamu Imaichi, and Tomoaki Imamura. 1997. Japanese morphological analysis system ChaSen manual. Technical Report NAIST-IS-TR97007, Nara Institute of Science and Technology. In Japanese.

Shinsuke Mori and Makoto Nagao. 1998. Unknown word extraction from corpora using n-gram statistics. *Journal of the Information Processing Society of Japan*, 39(7):2093–2100. In Japanese.

Makoto Nagao and Shinsuke Mori. 1994. A new method of N-gram statistics for large number of n and automatic extraction of words and phrases from large text data of Japanese. In *Proc. of the 15th COLING*, pages 611–615.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proc. of the 15th COLING*, pages 201–207.

Masaaki Nagata. 1996a. Automatic extraction of new words from Japanese texts using generalized forward-backward search. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 48–59.

Masaaki Nagata. 1996b. Context-based spelling correction for Japanese OCR. In *Proc. of the 16th COLING*, pages 806–811.

Masaaki Nagata. 1997. A self-organizing Japanese word segmenter using heuristic word identification and re-estimation. In *Proc. of the 5th Workshop on Very Large Corpora*, pages 203–215.

David Palmer. 1997. A trainable rule-based algorithm for word segmentation. In *Proc. of the 35th ACL/8th EACL*, pages 321–328.

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

Richard Sproat and Chilin Shih. 1990. A statistical method for finding word boundaries in Chinese text. *Computer Processing of Chinese and Oriental Languages*, 4:336–351.

Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A stochastic finite-sate word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3).

Maosong Sun, Dayang Shen, and Benjamin K. Tsou. 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proc. of COLING-ACL '98*, pages 1265–1271.

Koichi Takeda and Tetsunosuke Fujisaki. 1987. Automatic decomposition of kanji compound words using stochastic estimation. *Journal of the Information Processing Society of Japan*, 28(9):952–961. In Japanese.

Kouichi Takeuchi and Yuji Matsumoto. 1995. HMM parameter learning for Japanese morphological analyzer. In *Proc. of the 10th Pacific Asia Conference on Language, Information and Computation (PACLING)*, pages 163–172.

Virginia Teller and Eleanor Olds Batchelder. 1994. A probabilistic algorithm for segmenting non-kanji Japanese strings. In *Proc. of the 12th AAAI*, pages 742–747.

Laura Mayfield Tomokiyo and Klaus Ries. 1997. What makes a word: learning base units in Japanese for speech recognition. In *Proc. of the ACL Special Interest Group in Natural Language Learning (CoNLL97)*, pages 60–69.

Zimin Wu and Gwyneth Tseng. 1993. Chinese text segmentation for text retrieval: Achievements and problems. *Journal of the American Society for Information Science*, 44(9):532–542.

Dekai Wu. 1998. A position statement on Chinese segmentation. http://www.cs.ust.hk/~dekai/-papers/segmentation.html. Presented at the Chinese Language Processing Workshop, University of Pennsylvania.

Mikio Yamamoto. 1996. A re-estimation method for stochastic language modeling from ambiguous observations. In *Proc. of the 4th Workshop on Very Large Corpora*, pages 155–167.

J. Yamron, J. Baker, P. Bamberg, H. Chevalier, T. Dietzel, J. Elder, F. Kampmann, M. Mandel, L. Manganaro, T. Margolis, and E. Steele. 1993. LINGSTAT: An interactive, machine-aided translation system. In *Proc. of the HLT Workshop*, pages 191–195.