Unsupervised Statistical Segmentation of Japanese Kanji Strings

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

July 12, 1999

Abstract

Word segmentation is an important issue in Japanese language processing because Japanese is written without space delimiters between words. We propose a simple dictionary-less method to segment Japanese kanji sequences into words based solely on character n-gram counts from an unannotated corpus. The performance was often better than that of rule-based morphological analyzers over a variety of both standard and novel error metrics.

1 Introduction

This paper describes a simple, effective dictionary-less method to segment Japanese kanji sequences into words based solely on character n-gram counts from a completely unannotated corpus.

Word segmentation is an important issue in Japanese language processing because Japanese is written without space delimiters between words. Robust segmentation to recover the lexical items is thus a key preprocessing step to essentially any Japanese text analysis. Also, segmentation is often used to aid term extraction, either for automatically creating and augmenting dictionaries, or for indexing documents (Nagao and Mori, 1994; Nagata, 1996a; Fung, 1998). Wu and Tseng (1993) discuss the importance of segmentation to Chinese information retrieval (Chinese also lacks word delimiters). Yet another application that has been proposed in the literature is OCR spelling correction (Nagata, 1996b).

Typically, Japanese word segmentation is performed by morphological analysis based on lexical and grammatical knowledge. However, when *unknown* words – those not found in the analyzer's lexicon – are encountered, heuristics must be employed. These heuristics may take advantage of the fact that there are three types of Japanese characters: kanji (characters borrowed from Chinese), hiragana, and katakana. Changes in character type often indicate word boundaries, although Nagata (1997) notes that using this heuristic alone achieves less than 60% accuracy.

Fung (1998) attributes the difficulties in Chinese morphological analysis to a lack of morphological clues, and argues in favor of a statistical approach. Similar difficulties occur in the morphological analysis of long Japanese kanji sequences. These sequences are typically noun phrases or compound nouns, so grammatical constraints do not aid segmentation. For instance, *sha-choh-ken-gyoh-mu-bu-choh* ("a president as well as a general manager of business") should be

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

Figure 1: Statistics from 1993 Japanese newswire (NIKKEI), 79,326,406 characters total. The sequence length categories are disjoint, i.e., "theQUICKbrownFOX" has three lower-case characters in sequences of length 1-3 and five lower-case characters in sequences of length 4-6.

segmented as follows: *sha-choh/ken/gyoh-mu/bu-choh* (president/both/business/general manager). An example of an incorrect segmentation is: *sha-choh/ken-gyoh/mu/bu-choh* (president/subsidiary business/Tsutomu (personal name)/general manager). Since both alternatives are sequences of four nouns, they cannot be distinguished by part-of-speech information alone. Furthermore, heuristics based on character type clearly do not apply because all the characters are kanji. Thus, kanji sequences are problematic for morphological analysis.

Although potentially difficult to process, kanji sequences often contain domain terms and proper nouns, which are important in many applications such as information retrieval, information extraction, and text summarization. For example, Fung (1998) notes that 50-85% of the terms in various technical dictionaries are composed at least partly of kanji. Also, long kanji sequences can comprise a nontrivial portion of Japanese text, as shown in Figure 1. Since kanji sequences longer than 3 characters generally consist of more than one word, we see that 21.2% of 1993 Nikkei newswire consists of kanji sequences that need to be segmented, with 5.1% of the corpus made up of relatively long sequences. Thus, the accuracy of kanji sequence segmentation is an important aspect of the total segmentation process.

Many algorithms previously proposed for Japanese segmentation rely on lexical knowledge or pre-segmented training data. In contrast, the method we present in this paper for segmenting kanji sequences has the following four key features:

1. It uses neither a lexicon nor grammar;
2. It does not require annotated training data;
3. It does not rely on Japanese-specific heuristics (and therefore may be applicable to other languages as well), and
4. It is conceptually quite simple.

Our results show that in the segmentation of long kanji strings, our algorithm can provide error rate reductions of up to 41% with respect to Juman (Matsumoto et al., 1994) and Chasen (Matsumoto et al., 1997), two lexicon-based morphological analyzers in widespread use.

2 Algorithm

Our algorithm uses only the counts of character n-grams to make segmentation decisions. Before formally presenting it, we illustrate its use with an example.

Let "A B C D W X Y Z" represent an eight-kanji sequence. To decide whether the fourth location (between D and W) should be a word boundary, we check whether n-grams that are adjacent to the proposed boundary, such as the trigrams "B C D" and "W X Y", tend to be more frequent than n-grams that straddle it, such as the trigram "C D W". If this is indeed the case, then we have evidence that there is a word boundary between D and W, since there seems to be relatively little cohesion between the characters occurring to the left and those occurring to the right of the fourth location.

We choose to consider only a fixed set $\mathcal{N}$ of n-gram orders to use as evidence in the segmentation decision. Continuing with our example, suppose that $\mathcal{N} = \{2\}$. Then, letting $\#(s)$ denote the number of times the character n-gram s occurs in the (unannotated) training corpus, we pose the following two questions:

$$\text{Is } \#(\text{C D}) > \#(\text{D W})?$$
$$\text{Is } \#(\text{W X}) > \#(\text{D W})?$$

Each "yes" constitutes evidence that there should be a boundary between D and W. If $\mathcal{N} = \{2, 4\}$, then six more questions are posed:

Is #(A B C D) > #(B C D W)?	Is #(W X Y Z) > #(B C D W)?
Is #(A B C D) > #(C D W X)?	Is #(W X Y Z) > #(C D W X)?
Is #(A B C D) > #(D W X Y)?	Is #(W X Y Z) > #(D W X Y)?

In order to compensate for the fact that there are more questions for $(n+1)$-grams than there are for n-grams, we calculate the fraction of affirmative answers separately for each n in $\mathcal{N}$. Our algorithm then simply creates a segment boundary if and only if the average fraction of affirmative answers is either (1) greater than that of both the locations to the immediate left and right (i.e., a local maximum), or (2) exceeds some threshold t. The second condition is necessary to allow for words consisting of a single character. Notice that it also gives us a degree of control over the granularity of the segmentation: if the threshold value is high, then longer words are created.

More formally, let $g_n(x)$ denote the n-gram starting at the xth character; for instance, in the example above, $g_4(2) =$ "B C D W". Let $I_>(y, z)$ be an indicator function that is 1 when $y > z$, and 0 otherwise. Then, for each location i (i.e., the location just after the ith character), we compute $v_n(i)$, the average evidence from n-character sequences for i being a word boundary, as follows:

$$v_n(i) = \frac{1}{2n-2} \sum_{k=i-n+2}^{i} \left(I_> \left(\#(g_n(i-n+1)), \#(g_n(k)) \right) + I_> \left(\#(g_n(i+1)), \#(g_n(k)) \right) \right)$$

The quantity $v_{\mathcal{N}}(i)$ is the average $v_n(i)$:

$$v_{\mathcal{N}}(i) = \sum_{n \in \mathcal{N}} v_n(i)/|\mathcal{N}|.$$

After $v_{\mathcal{N}}(i)$ is computed for every location i of the sequence, word boundaries are located at all locations ℓ such that either

- $v_{\mathcal{N}}(\ell) > v_{\mathcal{N}}(\ell - 1)$ and $v_{\mathcal{N}}(\ell) > v_{\mathcal{N}}(\ell + 1)$, or
- $v_{\mathcal{N}}(\ell) \geq t$, the threshold parameter.

The algorithm can be implemented to be efficient both in the training (count acquisition) and testing phase. Computing n-gram statistics for all possible values of n simultaneously can be done in a space-efficient manner using suffix arrays in $O(m \log m)$ time, where m is the training corpus size (Manber and Myers, 1993; Nagao and Mori, 1994). However, if the set of n-gram orders $\mathcal{N}$ is known ahead of time, conceptually simpler algorithms with the same or linear running times suffice; we implemented an $O(m \log m)$ procedure that explicitly stores the n-gram counts in a table. Memory allocation for this table can be significantly reduced by omitting n-grams occurring only once and assuming the count of unseen n-grams to be one. For instance, the kanji n-gram table for $\mathcal{N} = \{2, 3, 4, 5, 6\}$ extracted from our 150-megabyte corpus was 18 megabytes in size, which fits easily in memory. The size of the n-gram table grows very slowly with corpus size once the corpus is sufficiently large.

In the application phase, the algorithm is clearly linear in the size of the test data, assuming $|\mathcal{N}|$ is treated as a constant.

3 Experimental Framework

The data for our experiments was drawn from 150 megabytes of 1993 Japanese newspaper articles (details of this corpus are given in Figure 1 above). Five test sets were obtained from this corpus. Proposed segmentations were evaluated by comparison to a segmentation created by hand.

Section 3.1 describes the test sets and our procedure for annotating them. Section 3.2 briefly sketches the morphological analyzers and simple baseline algorithm against which we compared our segmentation method. In section 3.3, the error metrics we used to measure performance are presented.

3.1 Test datasets

The five disjoint test sets, summarized in Figure 2, consisted of randomly extracted kanji sequences at least ten characters in length: long sequences were selected as being the most difficult to segment (Takeda and Fujisaki, 1987).

	# of sequences (tokens)	average length	# of words	average word length
test1	500	12.27	2554	2.40
test2	500	12.40	2574	2.41
test3	500	12.05	2508	2.40
test4	500	12.12	2540	2.39
test5	500	12.22	2556	2.39

Figure 2: The statistics of test datasets. Lengths are in characters.

To obtain the gold-standard segmentations of the test datasets, we added bracket annotations to them by hand. Our annotation scheme was motivated by 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 hand segmentation may be described as follows. At the *word level*, both a stem word and its affixes are bracketed together as a single word. At the *morpheme level*, stem words are divided from their prefixes and suffixes. For example, *den-wa-ki* (phone device) is segmented as [[*den-wa*][*ki*]]: *den-wa* (phone) can be used independently, but *ki*, a suffix that adds the meaning "device", cannot. Following Takeda and Fujisaki (1987), we treat a one-character word that can be appear on its own as an affix when it serves such a role. For instance, 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 *shi* serves as a suffix in *naga-no-shi*.

Loosely speaking, word-level bracketing demarcates discourse entities; it appears to be a natural segmentation level for native Japanese speakers (as shown below). Morpheme-level brackets enclose strings that cannot be segmented into smaller pieces without loss of meaning.[1] For instance, if *naga-no* in *naga-no-shi* is segmented into the two units *naga* (long) and *no* (field), the intended meaning disappears.

Numerics are grouped with the unit (measure)[2] at the word level; the characters representing the actual number were bracketed together as a single morpheme. For instance, *ichi-kyuh-kyuh-san-nen* (the year 1993) is segmented as [[*ichi kyuh kyuh san*][*nen*]] ([[one nine nine three][year]]). Reasonable alternative bracketings of numeric expressions exists; however, in our tests we altered the output of all the algorithms tested so as to conform to this bracketing, so that our numerics segmentation policy does not affect the relative performance of the various methods.

Three native Japanese speakers participated in the annotation of the test data: one human segmented all the test 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%: at only 62 out of 3927 locations did one or both of the humans disagree with the annotation. It is interesting to note that all disagreement was at the morpheme level.

[1] This level of segmentation may be seen as consistent with the *Monotonicity Principle* for segmentation proposed by Wu (1998).

[2] In Japanese, *measure words* are used after numbers to indicated the unit of measure. That is, one must say *san satsu no hon* (three book-units of book); *san hon* is not permissible.

3.2 Baseline algorithms

In order to evaluate our segmentation method, we chose to compare its performance against that of three other algorithms.

Juman (Matsumoto et al., 1994), downloadable from `http://www-nagao.kuee.kyoto-u.ac.jp/-nl-resource/juman-e.html`, is a state-of-the-art user-extensible morphological analyzer. Chasen (Matsumoto et al., 1997), which can be found at the URL `http://cactus.aist-nara.ac.jp/-lab/nlt/chasen.html`, is a later version of Juman. In both cases, the default values were used for all parameters, and the dictionaries and grammar rules were used as distributed without modification.

We also tested a simple baseline algorithm that produces a boundary after every second character (the average test-set word length, according to Figure 2). However, the segmentation performance using this method was very poor; we omit the results from our discussion below.

3.3 Metrics

To evaluate a given proposed segmentation against the gold-standard test annotation, we employed the following evaluation metrics: word and morpheme precision and recall; a modified crossing brackets rate and zero crossing brackets rate; and location precision, location recall, and location total error rates.

3.3.1 Word and Morpheme Accuracy

The most standard performance metrics in word segmentation are, naturally, word precision and recall. Treating a segmentation as a flat (i.e., non-nested) bracketing[3], *word precision* is defined as the percentage of proposed brackets that exactly match word-level brackets in the annotation; *word recall* is the percentage of word-level annotation brackets that are proposed by the algorithm in question.

One problem that arises in using word precision and recall is that our main baseline algorithms, being morphological analyzers, are meant to produce morpheme-level segments. To compensate for this fact in our word-level experiments, we altered the segmentations produced by Juman and Chasen by concatenating together stem words with their affixes (identified by the part-of-speech information Juman and Chasen provide). Numeric strings were also problematic, because Juman and Chasen appear to implement different segmentation policies with respect to these types of character sequences. To factor out this difference, we removed boundaries between numeric kanji characters from the segmentations produced by every algorithm tested, including our own. Since in practice, it is easy to implement a post-processing step that concatenates together number characters, we do not feel that this change misrepresents the performance of any of the algorithms.

Since Juman and Chasen were designed to segment at the morphemic level, we also measured *morpheme precision* and *morpheme recall*, defined analogously to word precision and word

[3] For instance, the segmentation "| A B | C |" corresponds to the bracketing "[A B][C]".

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

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

Figure 3: 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.

recall. Numerical expressions were again resegmented as in the word accuracy tests.

3.3.2 Compatible Brackets and All-Compatible Brackets Rates

Although word-level accuracy is a standard performance metric, it is clearly very sensitive to the test annotation. The authors of Juman and Chasen may well have tuned their algorithms using different notions of word and morpheme than the definitions we used in annotating the data. We therefore also developed a new, more robust metric, the *compatible brackets rate*, intended to measure the number of proposed brackets that would be incorrect with respect to *any* reasonable annotation.

The compatible brackets rate takes into account two types of errors. The first type, 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 the annotation brackets; furthermore, it is unlikely that another human would create a segmentation with brackets that cross the annotation brackets[4]. The second type of error is a bracket that subdivides a morpheme-level annotation bracket. Recall that if morpheme-level brackets are divided, loss of meaning results; thus, we expect that any reasonable bracketing should not further segment morphemes. See Figure 3 for some examples.

We define a *compatible bracket* as a proposed bracket that is neither a crossing bracket nor a morpheme-dividing bracket. Then, we define the compatible brackets rate as the compatible brackets precision:

$$\text{Compatible brackets rate} = \frac{\#\text{ of compatible brackets}}{\#\text{ of proposed brackets}}$$

We observe that the compatible brackets rate metric accounts for different levels of segmentation simultaneously. The granularity of Chasen's segmentation varies from morpheme level to word level to compound word level (by our definition); for instance, the name of a well-known university is recognized as one segment, whereas a lesser-known university name is divided into the name and the word "university". Using the compatible brackets rate, both segmentations can be counted as correct.

[4]The situation is different for parsing, where attachment decisions are involved.

We use the *all-compatible brackets rate* (related to the zero crossing brackets rate) to measure the number of kanji sequence segmentations in which all brackets are compatible:

$$\text{all-compatible brackets rate} = \frac{\text{\# of sequences segmented without incompatible brackets}}{\text{\# of sequences}}$$

Intuitively, this function corresponds to the ease with which a human could correct the output of the segmentation algorithm. That is, 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 expend effort on a great many sequences.

3.3.3 Location precision, recall and total accuracy

Using the compatible brackets rate, we can robustly measure the fitness of a proposed segmentation with respect to multiple levels of a correctly structured segmentation. However, the compatible brackets rate, being a measure of precision, tends to favor minimal bracketings: indeed, if the whole sequence is deemed to be a single (large) segment, the crossing brackets rate is 0%. Furthermore, this measure evaluates the goodness of an output segmentation as a bracketing, rather than as a segmentation *per se*, which may not be appropriate for all applications.

To measure the tradeoff between precision and recall in the placement of segmentation boundaries, we use *location precision, recall*, and *total accuracy*, defined as follows. Let L_{all} be the set of all locations between characters in a given test set, and let B_M and B_W be the set of morpheme and word boundaries, respectively. Let T be the set of boundaries proposed by the algorithm being tested. To account for the two levels of bracketing in the gold standard segmentation, we allow the algorithm being tested to omit morpheme boundaries (recall that this was the level at which all the human disagreements occurred); however, both omitting a word boundary and positing a boundary where there is none count as mistakes:

$$\begin{aligned} \text{location precision} &= |T \cap (B_M \cup B_W))|/|T| \\ \text{location recall} &= |T \cap B_W|/|B_W| \\ \text{location total accuracy} &= (|T \cap (B_W \cup B_M)| + |(L_{all} - T) \cap (L_{all} - B_W - B_M)|) / |L_{all}| \end{aligned}$$

4 Experimental Results and Analysis

The five test datasets were segmented using our algorithm, Juman, Chasen, and the baseline algorithm. Section 4.1 describes the test results. In section 4.2, the error cases are analysed.

4.1 Experimental Results

In this section, we report the best average results over the five test sets from preliminary searches of the parameter space, namely, the values for the set $\mathcal{N}$ of n-gram orders to consider and the threshold parameter t. However, a formal search for parameter settings optimizing performance on a held-out parameter training set was not performed.

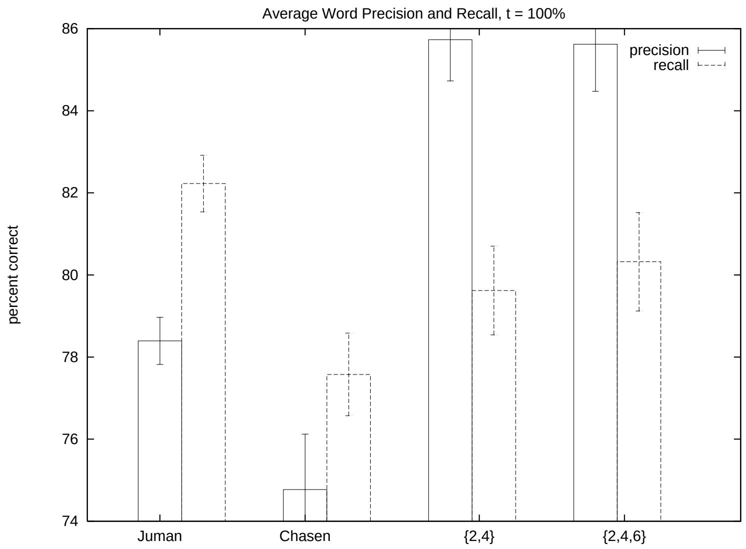

Figure 4: Word precision and recall.

Figure 4 depicts word precision and recall rates for Juman, Chasen, and our algorithm for parameter settings $(\mathcal{N}, t) = (\{2, 4\}, 100\%)$ and $(\mathcal{N}, t) = (\{2, 4, 6\}, 100\%)$ (in this and all performance graphs, the "error bars" represent one standard deviation). Our algorithm achieves much higher precision, with recall falling between that of Juman and that of Chasen. We note that the higher the value of t, the longer the segments produced by the algorithm tend to be, which accounts for the extremal value of t in this test of word-level accuracy; however, such a high value makes the algorithm more conservative, which causes its recall performance to suffer.

We conjecture that the relatively poor performance of Chasen with respect to the older version Juman may be due to the fact that our notion of a correct segmentation differs from that of the authors of these two programs. Thus, our comparative results should be taken with a grain of salt.

Figure 5 shows morpheme precision and recall rates. Since segmentation at the morpheme-level requires shorter brackets, a lower value of the threshold parameter ($t = 35\%$) gave better results. Here, the performance of our algorithm with setting $\mathcal{N} = \{2, 4\}$ was somewhere between Juman and Chasen, but with setting $\mathcal{N}\{2, 4, 6\}$, our algorithm has worse precision and recall than the morphological analyzers. Again, though, note that Chasen seems to perform more poorly than Juman, which may indicate a mismatch in the notion of morpheme we used and the notion employed by the creators of Juman and Chasen.

We now turn to our more robust metrics, meant to measure "egregious" segmentation mistakes that should be incorrect with respect to any human annotation. Figure 6 depicts the compatible brackets and all-compatible brackets rates. Our algorithm does noticeably better on both metrics. Observe that Chasen yields higher accuracy than its predecessor Juman, which provides added confidence in the fitness of our compatible-brackets measures.

Finally, Figure 7 shows location accuracy rates. As in the word and morpheme accuracy cases, we see that our algorithm yields better precision but inferior recall with respect to Juman and Chasen. The total location accuracy did not greatly differ from algorithm to algorithm.

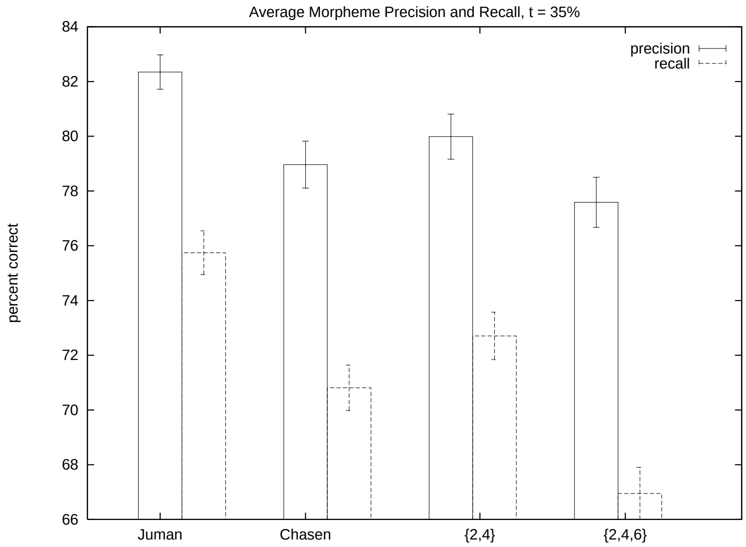

Figure 5: Morpheme precision and recall.

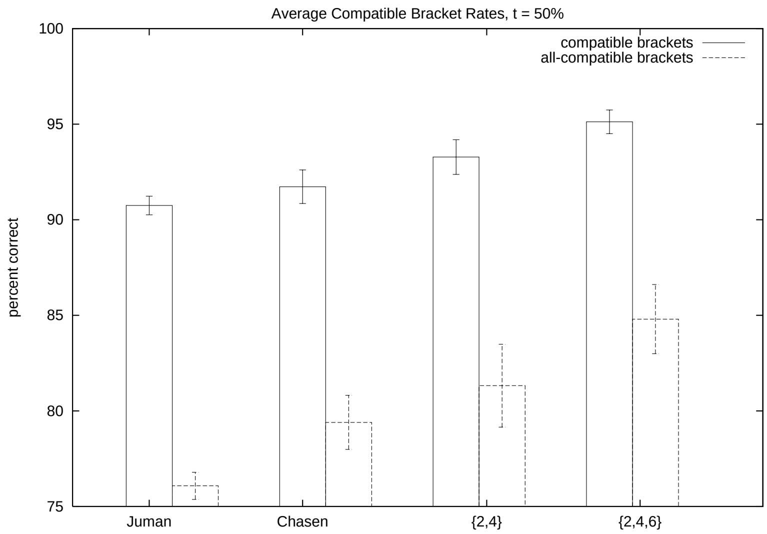

Figure 6: Compatible brackets and all-compatible bracket rates.

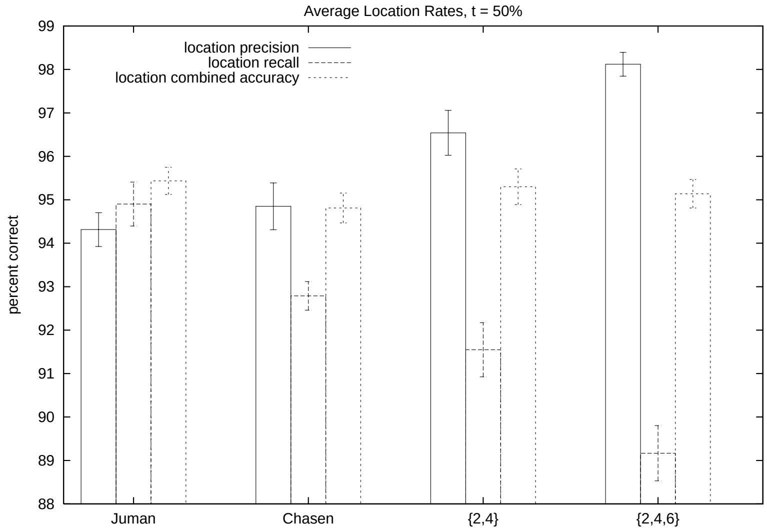

Figure 7: Location accuracy rates.

4.2 Analysis of Results

To study the influence of n, we evaluated performance on a single test set, `test1`, for parameter settings $\mathcal{N} = \{2\}, \{3\}, \{4\}, \{5\}$, and $\{6\}$. To avoid the influence of changing the threshold, $t = 100\%$ was used for this experiment.

The count column in Figure 8 shows the number of words and possible compound words of each length (possible compound words were counted based on the assumption that any consecutive words can make a compound word, as an approximation of the actual word length distribution). Figure 8 shows that the ranking of the n-gram orders by word count almost exactly matches the ranking of n-gram orders by performance on the compatible brackets rate and location precision measures. We also see that $\mathcal{N} = \{2\}$ and $\{4\}$ yield good performance with respect to the other error metrics as well. These ranking correlations are preserved for the averages over all the data sets. Thus, it appears that the effectiveness of a given n is correlated with word-length distribution in the test dataset.

n	count	rank	c.b.	rank	L-prec	rank	all-c.b.	rank	L-rec	rank	L-total	rank
2	1427	1	94.09	1	98.02	1	82.0	2	85.2	1	93.65	1
4	913	2	87.59	3	96.07	2	76.8	3	46.59	2	78.84	2
6	599	3	88.28	2	95.73	3	84.6	1	25.86	5	71.13	4
5	524	4	80.45	4	90.39	4	75.6	4	26.87	4	70.89	5
3	508	5	80.01	5	87.71	5	69.2	5	39.92	3	74.81	3

Figure 8: Performance on `test1` of n-grams for different n, $t = 100\%$, sorted by counts.

Figure 9 ranks various values of $\mathcal{N}$ for the compatible-brackets and location measures on `test1`. When compatible brackets are used for the evaluation, $\mathcal{N} = \{2, 3, 4, 5, 6\}$ yields the best performance, whereas $\mathcal{N} = \{2, 4, 6\}$ is the best setting for location precision and recall.

In terms of the types of mistakes our method makes, mis-associated affixes and unseparated one-character words are the major errors. An example of a mis-associated prefix is that

$\mathcal{N}$	c.b.	rank	all-c.b.	rank	L-prec	rank	L-rec	rank	L-total	rank
$\{2,4,6\}$	95.61	2	85.6	3	98.59	1	87.83	1	94.84	1
$\{2,4\}$	95.60	3	85.8	2	98.42	2	87.05	2	94.49	2
$\{2,3,4\}$	95.31	4	85.2	4	98.24	4	86.47	3	94.21	3
$\{2,3,4,5,6\}$	95.87	1	87.2	1	98.42	2	85.54	4	93.93	4
$\{2\}$	94.42	6	83.4	6	98.02	6	85.2	5	93.65	5
$\{2,3\}$	94.83	5	84.2	5	98.21	5	84.71	6	93.54	6

Figure 9: Performance on `test1` for different values for $\mathcal{N}$, $t = 100\%$, sorted by location total accuracy.

[*ei-gyoh*][[*fuku*][*bu-choh*]] ([business][[vice][manager]]) is segmented as [*ei-gyoh-fuku*][*bu-choh*] – the character *fuku* has been mistakenly attached to *ei-gyoh* instead of *bu-choh*. An example of an unseparated 1-character word occurs when [*bu-choh*][*ken*][*sha-choh*] ([manager][as well as][president]) is segmented as [*bu-choh-ken*][*sha-choh*]. Figure 10 classifies the errors observed when using $(\mathcal{N}, t) = (\{2,4,6\}, 50\%)$ for `test1`. For the purpose of comparison, the same error types are counted for Chasen.

	$(\{2,4,6\}, 50\%)$	Chasen
Mis-associated prefix or suffix	33	0
Unseparated 1-character words	46	0
Wrongly segmented proper nouns:		
Location name	6	10
Person name	6	35
Organization name	1	13
Last and first name concatenated	39	0

Figure 10: Error types on `test1`.

Our algorithm does not model one-character words or morphemes directly: single-character segments appear as a result of the words around them being recognized correctly. Thus, it is not surprising that one-character words and affixes are the main causes of errors. Further work should be done to improve this point.

Since proper nouns tend to be unknown words, Chasen's error rates in this category are high. While Chasen tends to segment an uncommon name into one-character words found in the dictionary, our algorithm tends to recognize person names comparatively well. It often concatenates first and last names, but this type of error is better than excessive segmentation in practical applications. It is also worthwhile to note that in general, Chasen performs more poorly at names, which are objects of prime importance in information extraction and related tasks.

5 Related Work

Many previously proposed segmentation methods for Japanese text make use of either a pre-existing lexicon (Yamron et al., 1993; Matsumoto and Nagao, 1994; Nagao and Mori, 1994;

Takeuchi and Matsumoto, 1995; Nagata, 1997; Fuchi and Takagi, 1998) or pre-segmented training data (Nagata, 1994; Nagata, 1996a; Papageorgiou, 1994; Kashioka et al., 1998; Mori and Nagao, 1998). Other approaches first apply some baseline method, such as Juman, to create an initial segmentation of the test data and then attempt to improve the segmentation via a re-estimation method (Matsukawa et al., 1993; Yamamoto, 1996).

Unsupervised methods for Japanese segmentation which do not explicitly rely on the existence of a lexicon do exist, but they are limited in their applicability. Tomokiyo and Ries (1997) study the problem of learning segmentations of (transcriptions of) Japanese speech without recourse to any data other than the (unsegmented) test data itself, applying a minimum-perplexity induction algorithm. However, they explicitly avoid working with kanji-based text, leaving this for future work. Teller and Batchelder (1994) report an experiment on segmenting Japanese hiragana sequences using the probability of non-kanji bigram occurrences. Although no lexicon or annotated training corpus is used, they employ specific morphologically-based heuristics; furthermore, they also explicitly do not handle kanji sequences. In contrast, there are no limits in principle to using our method on strings containing any type of character.

Although Sun et al. (1998) focus on segmentation of Chinese rather than Japanese text, their method bears some resemblance to ours. Like us, they avoid using a lexicon or pre-segmented training data, but rather determine whether a given location constitutes a word boundary in part by deciding whether the two characters on either side tend to "bind" (occur) together; and they use thresholds, local maxima, and secondary local maxima to make the segmentation decisions. However, the statistics they use (mutual information and t-score) are more complex than the simple n-gram counts that we employ.

Takeda and Fujisaki (1987) proposed the *short unit model*, a type of Hidden Markov Model with linguistically-determined topology, to segment kanji compound words into 2-character stem words and 1-character prefixes and suffixes. Kanji sequences are segmented based on state transition probabilities acquired in the training phase. However, the 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.

6 Conclusion

In this paper, we have presented a simple unsupervised algorithm that segments Japanese kanji sequences into words based solely on the counts of character n-grams in an unannotated corpus. We evaluated performance 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; for example, the average compatible-brackets error was slightly less than half that for Chasen.

In future work, we plan to experiment on Japanese sentences with mixtures of character types. Since our method does not use any Japanese-dependent heuristics, we hope to test it on Chinese or other languages as well.

7 Acknowledgments

We thank Minoru Shindoh and Takashi Ando for reviewing the annotations. This material is based in part on work supported by a grant from the GE Foundation.

References

Takeshi Fuchi and Shinichiro Takagi. 1998. Japanese morphological analyzer using word co-occurrence - JTAG. In *Proceedings of the 35th ACL/8th EACL*, pages 409–413.

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

Ralph Grishman, Catherine Macleod, and John Sterling. 1992. Evaluating parsing strategies using standardized parse files. In *Third Conference on Applied Natural Language Processing*, pages 156–161. Association for Computational Linguistics.

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 *Proceedings of the 35th ACL/8th EACL*, pages 658–662.

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 *Proceedings of the Human Language Technology Workshop*, pages 227–32.

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

Yuji Matsumoto, Sadao Kurohashi, Takehito Utsuro, Yutaka Myoki, and Makoto Nagao. 1994. Japanese morphological analysis system JUMAN manual. Technical Report NAIST-IS-TR 94025, Nara Institute of Technology and Science, Nara, Japan. In Japanese.

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.

Shunsuke 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, July. 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 *Fifteenth International Conference on Computational Linguistics (COLING '94)*, pages 611–615. International Conference on Computational Linguistics.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Fifteenth International Conference on Computational Linguistics (COLING '94)*, pages 201–7.

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

Masaaki Nagata. 1996b. Context-based spelling correction for Japanese OCR. In *Sixteenth International Conference on Computational Linguistics (COLING '96)*, pages 806–811.

Masaaki Nagata. 1997. A self-organizing Japanese word segmenter using heuristic word identification and re-estimation. In Joe Zhou and Kenneth Church, editors, *Proceedings of the Fifth Workshop on Very Large Corpora*, pages 203–215.

Constantine P. Papageorgiou. 1994. Japanese word segmentation by hidden Markov model. In *Proceedings of the Human Language Technology Workshop*, pages 283–288. Advanced Research Projects Agency, Software and Intelligent Systems Technology Office, Morgan Kaufmann.

Maosong Sun, Dayang Shen, and Benjamin K. Tsou. 1998. Chinese word segmentation without using lexicon and hand-crafted training data. In *Proceedings of the 35th ACL/8th EACL*, 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, September. In Japanese.

Kouichi Takeuchi and Yuji Matsumoto. 1995. HMM parameter learning for Japanese morphological analyzer. In *Proceedings 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 *Proceedings of the 12th National Conference on Artificial Intelligence*, volume 2, pages 742–747, Seattle, Washington.

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

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, 30 June to 2 July 1998.

Mikio Yamamoto. 1996. A re-estimation method for stochastic language modeling from ambiguous observations. In Eva Ejerhed and Ido Dagan, editors, *Proceedings of the Fourth 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 *Proceedings of the Human Language Technology Workshop*, pages 191–95.