A Self-Organizing Japanese Word Segmenter using Heuristic Word Identification and Re-estimation

Masaaki NAGATA
NTT Information and Communication Systems Laboratories
1-1 Hikarinooka Yokosuka-Shi Kanagawa, 239 Japan
nagata@nttnly.isl.ntt.co.jp

Abstract

We present a self-organized method to build a stochastic Japanese word segmenter from a small number of basic words and a large amount of unsegmented training text. It consists of a word-based statistical language model, an initial estimation procedure, and a re-estimation procedure. Initial word frequencies are estimated by counting all possible longest match strings between the training text and the word list. The initial word list is augmented by identifying words in the training text using a heuristic rule based on character type. The word-based language model is then re-estimated to filter out inappropriate word hypotheses generated by the initial word identification. When the word segmenter is trained on 3.9M character texts and 1719 initial words, its word segmentation accuracy is 86.3% recall and 82.5% precision. We find that the combination of heuristic word identification and re-estimation is so effective that the initial word list need not be large.

1 Introduction

Word segmentation is an important problem for Japanese because word boundaries are not marked in its writing system. Other Asian languages such as Chinese and Thai have the same problem. Any Japanese NLP application requires word segmentation as the first stage because there are phonological and semantic units whose pronunciation and meaning is not trivially derivable from that of the individual characters. Once word segmentation is done, all established techniques can be exploited to build practically important applications such as spelling correction [Nagata, 1996] and text retrieval [Nie and Brisebois, 1996]

In a sense, Japanese word segmentation is a solved problem if (and only if) we have plenty of segmented training text. Around 95% word segmentation accuracy is reported by using a word-based language model and the Viterbi-like dynamic programing procedure [Nagata, 1994, Takeuchi and Matsumoto, 1995, Yamamoto, 1996]. However, manually segmented corpora are not always available in a particular target domain and manual segmentation is very expensive.

The goal of our research is unsupervised learning of Japanese word segmentation. That is, to build a Japanese word segmenter from a list of initial words and unsegmented training text. Today, it is easy to obtain a 10K-100K word list from either commercial or public domain on-line Japanese dictionaries. Gigabytes of Japanese text are readily available from newspapers, patents, HTML documents, etc..

Few works have examined unsupervised word segmentation in Japanese. Both [Yamamoto, 1996] and [Takeuchi and Matsumoto, 1995] built a word-based language model from unsegmented text

using a re-estimation procedure whose initial segmentation was obtained by a rule-based word se menter. The utility of this approach is limited because it presupposes the existence of a rule-base word segmenter like JUMAN [Matsumoto et al., 1994]. It is impossible to build a word segment for a new domain without human intervention.

For Chinese word segmentation, more self-organized approaches have been tried. [Sproat et al., built a word unigram model using the Viterbi re-estimation whose initial estimates were derive from the frequencies in the corpus of the strings of each word in the lexicon. [Chang et al., 199 combined a small seed segmented corpus and a large unsegmented corpus to build a word unigra model using the Viterbi re-estimation. [Luo and Roukos, 1996] proposed a re-estimation procedu which alternates word segmentation and word frequency re-estimation on each half of the trainin text divided into halves.

One of the major problems in unsupervised word segmentation is the treatment of unseen word [Sproat et al., 1996] wrote lexical rules for each productive morphological process, such as plur noun formation, Chinese personal names, and transliterations of foreign words. [Chang et al., 199 used a statistical method called "Two-Class Classifier", which decided whether the string is actuall a word based on the features derived from character N-gram.

In this paper, we present a self-organized method to build a Japanese word segmenter fro a small number of basic words and a large amount of unsegmented training text using a nov re-estimation procedure. The major contribution of this paper is its treatment of unseen word We devised a statistical word formation model for unseen words which can be re-estimated. W show that it is very effective to combine a heuristic initial word identification method with a r estimation procedure to filter out inappropriate word hypotheses. We also devised a new metho to estimate initial word frequencies.

Figure 1 shows the configuration of our Japanese word segmenter. In the following sections, w first describe the statistical language model and the word segmentation algorithm. We then describ the initial word frequency estimation method and the initial word identification method. Finall we describe the experiment results of unsupervised word segmentation under various conditions.

2 Language Model and Word Segmentation Algorithm

2.1 Word Segmentation Model

Let the input Japanese character sequence be $C = c_1 c_2 \ldots c_m$. Our goal is to segment it int word sequence $W = w_1 w_2 \ldots w_n$. The word segmentation task can be defined as finding a wor segmentation $\hat{W}$ that maximizes the joint probability of word sequence given character sequenc $P(W|C)$. Since the maximization is carried out with fixed character sequence C, the word segment only has to maximize the probability of the word sequence $P(W)$.

$$\hat{W} = \arg\max_W P(W|C) = \arg\max_W P(W) \quad (1$$

We approximate the joint probability $P(W)$ by the word unigram model, which is the product word unigram probabilities $P(w_i)$.

$$P(W) = \prod_{i=1}^{n} P(w_i) \quad (2$$

We used the word unigram model because of its computational efficiency.

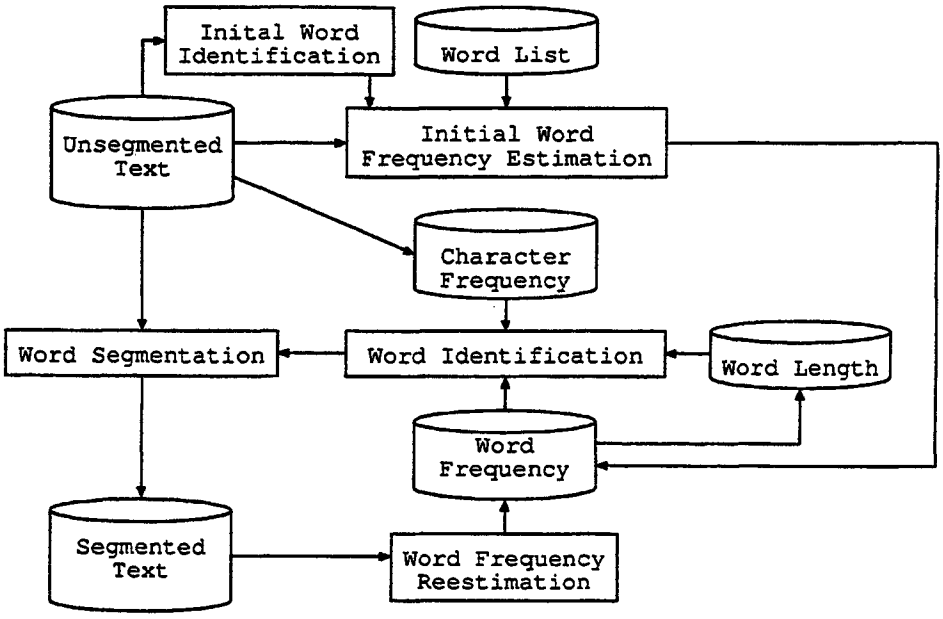


Figure 1: Block Diagram for the Self-Organizing Japanese Word Segmenter

2.2 Unknown Word Model

We defined a statistical word model to assign a reasonable word probability to an arbitrary substring in the input sentence. It is formally defined as the joint probability of the character sequence $c_1 \ldots c_k$ if w_i is an unknown word. We decompose it into the product of word length probability and word spelling probability,

$$P(w_i|\texttt{<UNK>}) = P(c_1 \ldots c_k|\texttt{<UNK>}) = P(k)P(c_1 \ldots c_k|k) \tag{3}$$

where k is the length of the character sequence and <UNK> represents unknown word.

We assume that word length probability $P(k)$ obeys a Poisson distribution whose parameter is the average word length λ in the training corpus. This means that we regard word length as the interval between hidden word boundary markers, which are randomly placed with an average interval equal to the average word length.

$$P(k) = \frac{(\lambda - 1)^{k-1}}{(k-1)!} e^{-(\lambda-1)} \tag{4}$$

We approximate the spelling probability given word length $P(c_1 \ldots c_k|k)$ by the product of character unigram probabilities regardless of word length.

$$P(c_1 \ldots c_k) = \prod_{i=1}^{k} P(c_i) \tag{5}$$

Character unigram probabilities can be estimated from unsegmented texts. The average word length λ can be computed, once the word frequencies in the texts are obtained.

$$\lambda = \frac{\sum |w_i| C(w_i)}{\sum C(w_i)} \tag{6}$$

where $|w_i|$ and $C(w_i)$ are the length and the frequency of word w_i, respectively. Therefore, t only parameters we have to (re)estimate in the language model are the word frequencies.

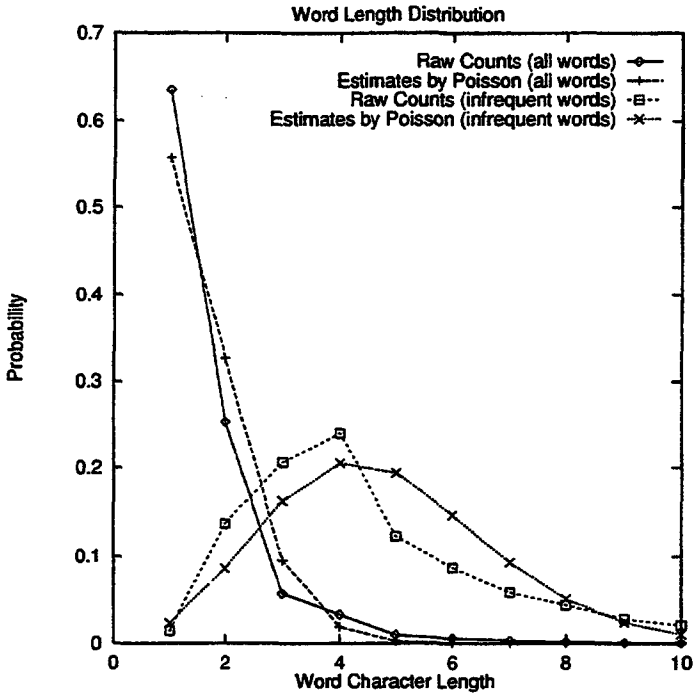


Figure 2: Word Length Distribution of the EDR corpus

Figure 2 shows the actual and estimated word length distributions in the corpus we used the experiment. It shows two pairs of distributions: word length of all words ($\lambda = 1.6$) and th of words appearing only once ($\lambda = 4.8$). The latter is expected to be close to the distribution unknown words. Although the estimates by Poisson distribution are not so accurate, they enab us to make a robust and computationally efficient word model.

2.3 Viterbi Re-estimation

We used the Viterbi-like dynamic programing procedure described in [Nagata, 1994] to get t most likely word segmentation. The generalized Viterbi algorithm starts from the beginning of t input sentence, and proceeds character by character. At each point in the sentence, it looks the combination of the best partial word segmentation hypothesis ending at the point and all wc hypotheses starting at the point.

We used the Viterbi re-estimation procedure to refine the word unigram model because its computational efficiency. It involves applying the above segmentation algorithm to a traini corpus, using a set of initial estimates of the word frequencies. The best analysis of the corpus taken to be the true analysis, the frequencies are re-estimated, and the algorithm is repeated un it converges.

3 Initial Word Frequency Estimation

3.1 Longest Match

We can get a set of initial estimates of the word frequencies by segmenting the training corpus using a heuristic (non-stochastic) dictionary-based word segmenter. In both Japanese and Chinese, one of the most popular non-stochastic dictionary-based approaches is the longest match method [1].

There are many variations of the longest match method, possibly augmented with further heuristics. We used a simple greedy algorithm described in [Sproat et al., 1996]. It starts at the beginning of the sentence, finds the longest word starting at that point, and then repeats the process starting at the next character until the end of the sentence is reached. We chose the greedy algorithm because it is easy to implement and guaranteed to produce only one segmentation.

3.2 String Frequency

[Sproat et al., 1996] also proposed another method to estimate a set of initial word frequencies without segmenting the corpus. It derives the initial estimates from the frequencies in the corpus of the strings of character making up each word in the dictionary *whether or not* each string is actually an instance of the word in question. The total number of words in the corpus is derived simply by summing the string frequency of each word in the dictionary. Finding (and counting) all instances of a string W in a large text T can be efficiently accomplished by making a data structure known as a suffix array, which is basically a sorted list of all the suffixes of T [Manber and Myers, 1993].

3.3 Longest Match String Frequency

The estimates of word frequencies by the above string frequency method tend to inflate a lot especially in short words, because of double counts. We devised a slightly improved version which we term the "longest match string frequency" method. It counts the instances of string W_1 in text T, unless the instance is also a substring of another string W_2 in dictionary D.

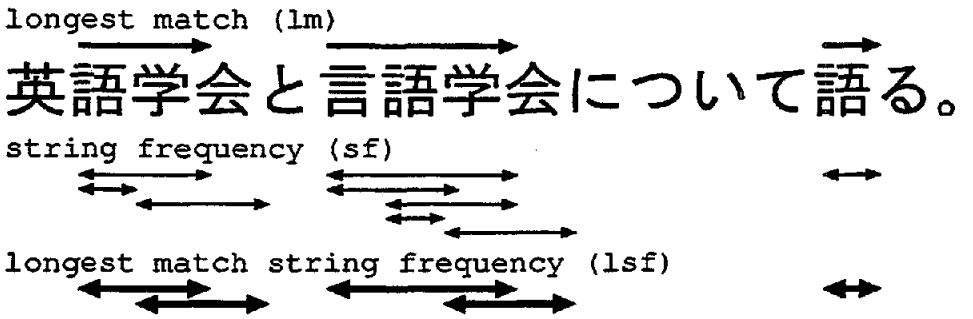
This method can be implemented by making two suffix arrays, S_T and S_D for text T and dictionary D. By using S_T, we first make list L_W of all occurrences of string W in the text. By using S_D, we then look up all strings $\bar{W}$ in the dictionary that include W as a substring, and make list $L_{\bar{W}}$ of all their occurrences in the text by using S_T. The longest match string frequency of word W in text T with respect to dictionary D is obtained by counting the number of elements in the set difference $L_W - L_{\bar{W}}$.

For example, if the input sentence is "英語学会と言語学会について語る。" (talk about the Association of English and the Association of Linguistics) and the dictionary has 言語学 (linguistics), 言語 (language), 語学 (language study), 学会 (association), and 語 (talk). Figure 3 shows the difference of the three methods.

The longest match string frequency (lsf) method considers all possible longest matches in the text, while the greedy longest match (lm) algorithm considers only one possibility. It is obvious that the longest match string frequency method remedies the problem that the string frequency (sf) method consistently and inappropriately favors short words.

The problem of the longest match string frequency method is that if a word W_1 is a substring of other word W_2 and if W_1 always appears as a substring of W_2 in the training text, just like 言語

[1] Although [Sproat et al., 1996] calls it "maximum matching", we call this method "longest match" according to a review on Chinese word segmentation [Wu and Tseng, 1993] and the literal translation of the Japanese name of the method 最長一致.



	longest match	string frequency	lm string freq.
言語学	1	1	1
言語	0	1	0
語学	1	2	1
学会	0	2	2
語	1	3	1
total	3	9	5

Figure 3: Comparison of the initial word frequency estimation methods

and 言語学 in the above example, the frequency estimate of W_1 becomes 0. Although this rarely happens for a large training text, we have to smooth the word frequencies.

4 Initial Word Identification Method

To a first approximation, a point in the text where character type changes is likely to be a word boundary. This is a popular heuristics in Japanese word segmentation. To help readers understand the heuristics, we have to give a brief introduction to the Japanese writing system.

In contemporary Japanese, there are at least five different types of characters other than punctuation marks: *kanji*, *hiragana*, *katakana*, Roman alphabet, and Arabic numeral. *Kanji* which means 'Chinese character' is used for both Chinese origin words and Japanese words semantically equivalent to Chinese characters. There are two syllabaries *hiragana* and *katakana*. The former is used primarily for grammatical function words, such as particles and inflectional endings, while the latter is used primarily to transliterate Western origin words. Roman alphabet is also used for Western origin words and acronyms. Arabic numeral is used for numbers.

By using just this character type heuristics, a non-stochastic and non-dictionary word segmenter can be made. In fact, using the estimated word frequencies obtained by the heuristics results in poor segmentation accuracy [2]. We found, however, that it is very effective to use the character type based word segmenter as a lexical acquisition tool to augment the initial word list.

The initial word identification procedure is as follows. First, we segment the training corpus by the character type based word segmenter, and make a list of words with frequencies. We then filter out *hiragana* strings because they are likely to be function words. We add the extracted word

[2]The word segmentation accuracy of the character type based method was less than 60%, while other estimation methods achieves around 70-80% as we show in the next section.

list to the original dictionary with associated frequencies, whether or not each string is actually a word. Although there are a lot of erroneous words in the augmented word list, most of them are filtered out by the re-estimation. This method works surprisingly well, as shown in the experiment.

5 Experiment

5.1 Language Data

We used the EDR Japanese Corpus Version 1.0 [EDR, 1995] to train and test the word segmenter. It is a corpus of 5.1 million words (208 thousand sentences). It contains a variety of Japanese sentences taken from newspapers, magazines, dictionaries, encyclopedias, textbooks, etc. It has a variety of annotations including word segmentation, pronunciation, and part of speech tag.

In this experiment, we randomly selected two sets of training sentences, each consisting of 100 thousand sentences. The first training set (training-0) is used to make initial word lists of various sizes. The second training set (training-1) is used to train various word segmenters. From the remaining of 8 thousand sentences, we randomly selected 100 test sentences to evaluate the accuracy of the word segmenters. Table 1 shows the number of sentences, words, and characters in the training and test sets [3].

Table 1: The amount of training and test data

	training-0	training-1	test
Sentences	100000	100000	100
Word Tokens	2460188	2465441	2538
Word Types	85966	85967	919
Characters	3897718	3906260	3984

Based on the frequency in the manually segmented corpus training-0, we made 7 different initial word lists (D1-D200) whose frequency threshold were 1, 2, 5, 10, 50, 100, 200, respectively. The size of the resulting word lists and their out-of-vocabulary rate (OOV rate) in the test sentences are shown in the second and third columns of Table 2. For example, D200 consists of words appearing more than 200 times in training-0. Although D200 consists of only 826 words, it covers 76.6% (OOV rate 23.4%) of the test sentences. This is an example of the Zipf law.

5.2 Evaluation Measures

Word Segmentation accuracy is expressed in terms of recall and precision as is done for bracketing of partial parses [Nagata, 1994, Sproat et al., 1996]. Let the number of words in the manually segmented corpus be Std, the number of words in the output of the word segmenter be Sys, and the number of matched words be M. *Recall* is defined as M/Std, and *precision* is defined as M/Sys.

Since it is inconvenient to use both recall and precision all the we also use the F-measure to indicate the overall performance. The F-measure was originally developed by the information

[3]Training-1 was used as plain texts that are taken from the same information source as training-0. Its word segmentation information was never used to ensure that training was unsupervised.

retrieval community. It is calculated by

$$F = \frac{(\beta^2 + 1.0) \times P \times R}{\beta^2 \times P + R} \tag{7}$$

where P is precision, R is recall, and β is the relative importance given to recall over precision. We set $\beta = 1.0$ throughout this experiment. That is, we put equal importance on recall and precision.

5.3 Comparison of Various Word Frequency Estimation Methods

We first compared the three frequency estimation methods described in the previous section: greedy longest match method (lm), string frequency method (sf), and longest match string frequency method (lsf). The sixth, seventh, and eighth columns of Table 2 show the word segmentation accuracy (F-measure) of each estimation method using different sets of initial words (D1-D200). For comparison, the word segmentation accuracy using real word frequency (wf), computed from the manual segmentation of training-1 (not training-0!), is shown in the fifth column of Table 2. The results are also diagramed in Figure 4.

Table 2: Word Segmentation Accuracies

	freq	vocab	oov	wf	lm	sf	lsf	lm+ct	sf+ct	lsf+ct
D1	≥1	85966	0.010	0.893	0.810	0.801	0.807	0.796	0.789	0.794
D2	≥2	39994	0.017	0.891	0.817	0.815	0.822	0.808	0.802	0.811
D5	≥5	18689	0.037	0.877	0.812	0.814	0.819	0.818	0.811	0.816
D10	≥10	10941	0.060	0.859	0.797	0.813	0.815	0.828	0.825	0.828
D50	≥50	3159	0.134	0.785	0.734	0.774	0.776	0.837	0.837	0.841
D100	≥100	1719	0.181	0.758	0.699	0.749	0.761	0.839	0.840	0.843
D200	≥200	826	0.234	0.729	0.644	0.643	0.731	0.828	0.830	0.832

First of all, word segmentation accuracy using real word frequencies (wf) significantly (5-10%) outperformed that of any frequency estimation methods. Among word frequency estimates, the longest match string frequency method (lsf) consistently outperformed the string frequency method (sf). The (longest match) string frequency method (sf and lsf) outperformed the greedy longest match method (lm) by about 2-5% when the initial word list size was under 20K (from D5 to D200). In all estimation methods, word segmentation accuracies of D1 are worse than D2, while D1 is slightly better than D2 in using real word frequencies.

5.4 Effect of Augmenting Initial Dictionary

We then compared the three frequency estimation methods (lm, sf, and lsf) with the initial dictionary augmented by the character type based word identification method (ct) described in the previous section. The word identification method collected a list of 108975 word hypotheses from training-1. The ninth, tenth, and eleventh columns of Table 2 show the word segmentation accuracies.

Augmenting the dictionary yields a significant improvement in word segmentation accuracy. Although the difference between the underlying word frequency estimation methods is small, the longest match string frequency method generally performs best. Surprisingly, the best word segmentation accuracy is achieved when the very small initial word list of 1719 words (D100) is augmented

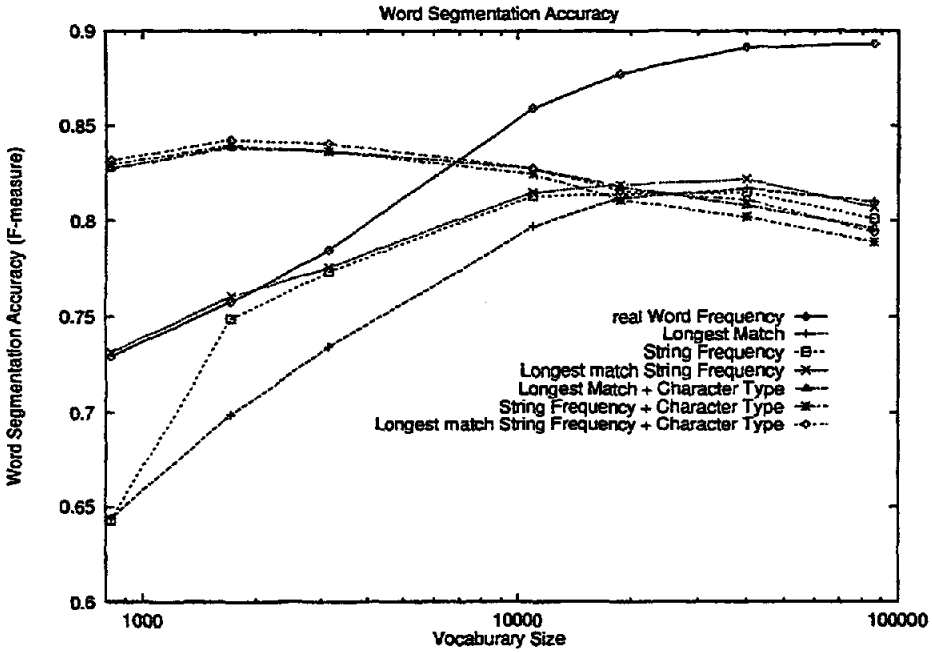


Figure 4: Initial word list size and word segmentation accuracies

by the heuristic word identification method, where the recall and precision are 86.3% and 82.5% (F-measure 0.843).

5.5 Effect of Re-estimation

To investigate the effect of re-estimation, we tested the combination of three initial word lists: D1, D2, D100, and two initial word frequency estimation methods: string frequency method (sf) and longest match string frequency method augmented with the word identification method (lsf+ct).

We applied the Viterbi re-estimation procedure three times. It seems further re-estimation brings no significant change. At each stage of re-estimation, we measured the word segmentation accuracy on the test sentences (not the training texts!). Figure 5 shows the word segmentation accuracy, the number of word tokens in the training texts, and the number of word types in the dictionary at each stage of re-estimation.

In general, re-estimation has little impact on word segmentation accuracy. It gradually improves the accuracy when the initial word list is relatively large (D1 and D2), while it worsen the accuracy a little when the initial word list is relatively small (D100). This might correspond with the results on unsupervised learning performed by an English part of speech tagger. Although [Kupiec, 1992] presented a very sophisticated method of unsupervised learning, [Elworthy, 1994] reported that re-estimation is not always helpful. We think, however, our results are because we used a word unigram model; it is too early to conclude that re-estimation is useless for word segmentation, as discussed in the next section.

It seems the virtue of re-estimation lies in its ability to adjust word frequencies and removing unreliable word hypotheses that are added by heuristic word identification. The abrupt drop in the number of word tokens at the first re-estimation step indicates that the inflated initial estimates of

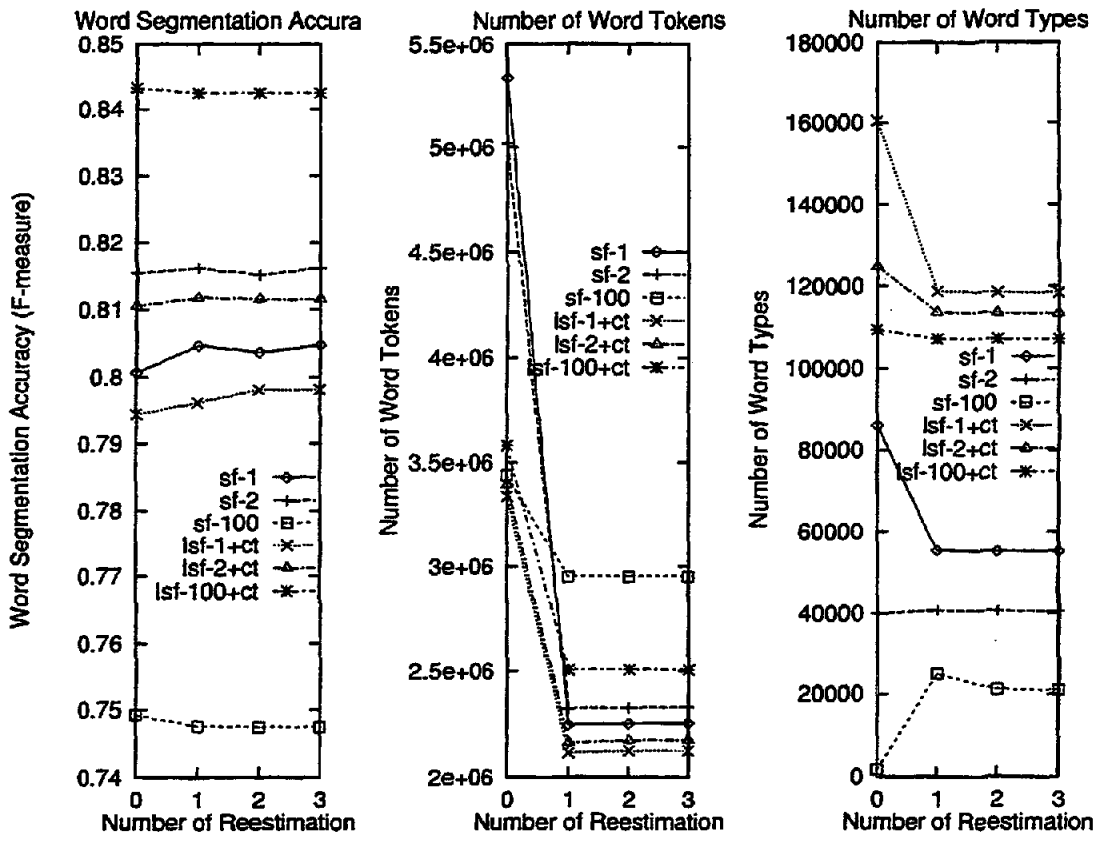


Figure 5: Word segmentation accuracy, the number of word tokens and word types at each re-estimation stage

word frequencies are adjusted to more reasonable values. The drop in the number of word types indicates the removal of infrequent words and unreliable word hypotheses from the dictionary.

6 Discussion

6.1 The Nature of the Word Unigram Model

First, we will clarify the nature of the word unigram model. Roughly speaking, word unigram based word segmenters maximize the product of the word frequencies under the fewest word principle which subsumes the longest match principle.

If two word segmentation hypotheses differs in the number of words, the one with fewer words is almost always selected. For example, the input string is c_1c_2 and the dictionary includes three words c_1c_2, c_1, c_2. To prefer segmentation hypothesis $c_1|c_2$ over c_1c_2, the following relation must hold.

$$\frac{C(c_1c_2)}{N} < \frac{C(c_1)}{N}\frac{C(c_2)}{N} \tag{8}$$

where $C(\cdot)$ represents the word frequency and N is the number of word tokens in the training text.

Suppose N is one million. Even if $C(c_1c_2) = 1$, c_1c_2 is preferred unless c_1 and c_2 are highly frequent, say $C(c_1) \approx C(c_2) > 1000$. It is obvious that the segmentation with fewer words are preferred.

If two word segmentation hypotheses have the same number of words, the one with larger product of word frequencies is selected. For example, the input string is $c_1c_2c_3$ and the dictionary includes four words c_1c_2, c_3, c_1, c_2c_3. To prefer segmentation hypothesis $c_1c_2|c_3$ over $c_1|c_2c_3$, the following relation must hold.

$$\frac{C(c_1c_2)}{N}\frac{C(c_3)}{N} < \frac{C(c_1)}{N}\frac{C(c_2c_3)}{N} \tag{9}$$

Since the denominator N is cancelled, it is obvious that the segmentation with larger product of frequencies is preferred.

6.2 Classification of Segmentation Errors

There are three major types of segmentation errors. The first type is not an error but the ambiguity resulting from inconsistent manual segmentation, or the intrinsic indeterminacy of Japanese word segmentation. For example, in the manually segmented corpus, we found the string 外国人労働者 (foreign laborer) is identified as one word in some places while in others it is divided into two words 外国人 (foreigner) and 労働者 (laborer). However, the word unigram based segmenter consistently identifies it as a single word. We assume 3-5 % of the segmentation "errors" belong to this type.

The second type is breakdown of unknown words. For example, the word 珍妙 (funny) is segmented into two word hypotheses 珍 (rare) and 妙 (strange). This is because 妙 is included in the dictionary. When a substring of an unknown word coincides with other word in the dictionary, it is very likely to be broken down into the dictionary word and the remaining substring. This is a major flaw of our word model using character unigram. It assigns too little probability to longer word hypotheses, especially more than three characters.

The third type is erroneous longest match. This happens frequently at the sequence of grammatical function words written in *hiragana*. For example, the phrase 集ま (gather) | っ (INFL) | て (and) | き (come) | た (past-AUXV), which means "came and gathered", is segmented into 集ま | って (TOPIC) | きた (north), because the number of words is fewer. The larger the initial word list is, the more often a *hiragana* word happens to coincide with a sequence of other *hiragana* words, because the number of character types in *hiragana* is small (< 100). This is the major reason why word segmentation accuracy levels off or decreases at a certain point, as the size of the initial word list increases.

6.3 Classification of the Effects of Re-estimation

There are two types of major changes in segmentation with re-estimation: word boundary adjustment and subdivision. The former moves a word boundary keeping the number of words unchanged. The latter break down a word into two or more words.

Re-estimation usually improves a sequence of grammatical function words written in *hiragana* at the sentence final predicate phrase if the initial segmentation and the correct segmentation have the same number of words. For example, the incorrect initial segmentation 連れ去 (take away) | られ (INFL + passive-AUXV) | たま (ball) | まだ (not yet) is correctly adjusted to 連れ去 (take away) | られ (INFL + passive-AUXV) | た (past-AUXV) | まま (still) | だ (COPULA), which means "still be taken away".

Re-estimation subdivides an erroneous longest match if the frequencies of the shorter words are significantly large. For example, the incorrect initial segmentation 控え (restrain) | たい (sea

bream) is correctly subdivided into 控え (restrain) | た (want-AUXV) | い (INFL), which means "want to restrain".

One of the most frequent undesirable effects of re-estimation is subdividing an infrequent word into highly frequent words, or a frequent word and an unknown word. For example, the correct infrequent word 使節 (ambassador) is subdivided into two frequent words, 使 (use-ROOT) and 節 (node).

As we said before, one of the major virtues of re-estimation is its ability to remove inappropriate word hypotheses generated by the initial word identification procedure. For example, from the phrase ソ連 (Soviet Union) | 製 (made-SUFFIX) | 戦車 (tank), which means "Soviet Union-made tank", the initial word identifier extracts two word hypotheses ソ and 連製戦車, where the former is written in *katakana* and the latter is written in *kanji*. If ソ連 and 製 is in the dictionary, the two erroneous word hypotheses ソ and 連製戦車 are removed and the correct word 戦車 is added to the dictionary after re-estimation.

7 Conclusion and Future Work

We have presented a self-organized method that builds a stochastic Japanese word segmenter from a small word list and a large unsegmented text. We found that it is very effective to augment the initial word list with automatically extracted words using character type heuristics. Re-estimation helps in adjusting word frequencies and removing inappropriate word hypotheses, although it has little impact on word segmentation accuracy if the word unigram model is used.

The major drawbacks of the current word segmenter is its breakdown of unknown words whose substrings coincide with other words in the dictionary, and the erroneous longest match at the sequence of functional words written in *hiragana*. The first drawback results from the character unigram based word model that prefers short words, while the second drawback results from the nature of the word unigram model which prefers fewest words segmentation.

One may argue that we could use the word bigram model. However, we don't know how we can estimate the initial word bigram frequencies from scratch. One may also argue that we could use the character bigram in the word model. However, the character bigram for the word model must be computed from segmented texts. Both of these suggest that we need a word segmenter to build a more sophisticated word segmenter. Therefore, as a next step of our research, we are thinking of using the proposed unigram based word segmenter to obtain the initial estimates of the word bigrams and the word-based character bigrams which will then be refined by a re-estimation procedure.

References

[Chang et al., 1995] Jing-Shin Chang, Yi-Chung Lin, and Keh-Yih Su. 1995. Automatic Construction of a Chinese Electronic Dictionary, In *Proceedings of WVLC-95*, pages 107-120.

[EDR, 1995] Japan Electronic Dictionary Research Institute. 1995. *EDR Electronic Dictionary Version 1 Technical Guide*, EDR TR2-003. Also available as *The Structure of the EDR Electronic Dictionary*, `http:///www.iijnet.or.jp/edr/`.

[Elworthy, 1994] David Elworthy. 1994. Does Baum-Welch Re-estimation Help Taggers? In *Proceedings of ANLP-94*, pages 53-58.

[Kupiec, 1992] Julian Kupiec. 1992. Robust Part-of-Speech Tagging using a Hidden Markov Model. *Computer Speech and Language*, 6, pages 225-242.

[Luo and Roukos, 1996] Xiaoqiang Luo and Salim Roukos. 1996. An Iterative Algorithm to Build Chinese Language Models, In *Proceedings of ACL-96*, pages 139-143.

[Matsumoto et al., 1994] Yuji Matsumoto, S. Kurohashi, T. Utsuro, and Makoto Nagao. 1994. *Japanese morphological analysis system JUMAN manual* (in Japanese).

[Manber and Myers, 1993] Udi Manber and Gene Myers. 1993. Suffix Arrays: A New Method for On-Line String Searches, *SIAM J. Comput.*, Vol.22, No.5, pp.935-948.

[Nagata, 1994] Masaaki Nagata. 1994. A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A^* N-Best Search Algorithm. In *Proceedings of COLING-94*, pages 201-207.

[Nagata, 1996] Masaaki Nagata. 1996. Context-Based Spelling Correction for Japanese OCR. In *Proceedings of COLING-96*, pages 806-811.

[Nie and Brisebois, 1996] Jian-Yun Nie and Martin Brisebois. 1996. On Chinese Text Retrieval. In *Proceedings of SIGIR-96*, pages 225-233.

[Sproat et al., 1996] Richard Sproat, Chilin Shih, William Gale, and Nancy Chang. 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, Vol.22, No.3, pages 377-404.

[Takeuchi and Matsumoto, 1995] Kouichi Takeuchi and Yuji Matsumoto. 1995. Learning parameters of Japanese morphological analyzer based-on hidden Markov model. *IPSJ Technical Report SIG-NL*, 108-3, pages 13-19 (in Japanese).

[Wu and Tseng, 1993] Zimin Wu and Gwyneth Tseng. 1993. Chinese Text Segmentation for Text Retrieval: Achievements and Problems, *Journal of ASIS*, Vol.44, No.9, pages 532-544.

[Yamamoto, 1996] Mikio Yamamoto. 1996. A Re-estimation Method for Stochastic Language Modeling from Ambiguous Observations, in *Proceedings of WVLC-96*, pages 155-167.