A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A^* N-Best Search Algorithm

Masaaki NAGATA
NTT Network Information Systems Laboratories
1-2356 Take, Yokosuka-Shi, Kanagawa, 238-03 Japan
(tel) +81-468-59-2796
(fax) +81-468-59-3428
(e-mail) nagata@nttnly.ntt.jp

Abstract

We present a novel method for segmenting the input sentence into words and assigning parts of speech to the words. It consists of a statistical language model and an efficient two-pass N-best search algorithm. The algorithm does not require delimiters between words. Thus it is suitable for written Japanese. The proposed Japanese morphological analyzer achieved 95.1% recall and 94.6% precision for open text when it was trained and tested on the ATR Corpus.

1 Introduction

In recent years, we have seen a fair number of papers reporting accuracies of more than 95% for English part of speech tagging with statistical language modeling techniques [2–4, 10, 11]. On the other hand, there are few works on stochastic Japanese morphological analysis [9, 12, 14], and they don't seem to have convinced the Japanese NLP community that the statistically-based techniques are superior to conventional rule-based techniques such as [16, 17].

We show in this paper that we can build a stochastic Japanese morphological analyzer that offers approximately 95% accuracy on a statistical language modeling technique and an efficient two-pass N-best search strategy.

We used the simple tri-POS model as the tagging model for Japanese. Probability estimates were obtained after training on the ATR Dialogue Database [5], whose word segmentation and part of speech tag assignment were laboriously performed by hand.

We propose a novel search strategy for getting the N best morphological analysis hypotheses for the input sentence. It consists of the forward dynamic programming search and the backward A^* search. The proposed algorithm amalgamates and extends three well-known algorithms in different fields: the Minimum Connective-Cost Method [7] for Japanese morphological analysis, Extended Viterbi Algorithm for character recognition [6], and Tree-Trellis N-Best Search for speech recognition [15].

We also propose a novel method for handling unknown words uniformly within the statistical approach. Using character trigrams as the word model, it generates the N-best word hypotheses that match the leftmost substrings starting at a given position in the input sentence.

Moreover, we propose a novel method for evaluating the performance of morphological analyzers. Unlike English, Japanese does not place spaces between words. It is difficult, even for native Japanese, to place word boundaries consistently because of the agglutinative nature of the language. Thus, there were no standard performance metrics. We applied bracketing accuracy measures [1], which is originally used for English parsers, to Japanese morphological analyzers. We also slightly extended the original definition to describe the accuracy of the N-best candidates.

In the following sections, we first describe the techniques used in the proposed morphological analyzer, we then explain the evaluation metrics and show the system's performance by experimental results.

2 Tagging Model

2.1 Tri-POS Model and Relative Frequency Training

We used the tri-POS (or triclass, tri-tag, tri-Ggram etc.) model as the tagging model for Japanese. Consider a word segmentation of the input sentence $W = w_1 w_2 \ldots w_n$ and a sequence of tags $T = t_1 t_2 \ldots t_n$ of the same length. The morphological analysis task can be formally defined as finding a set of word segmentation and parts of speech assignment that maximize the joint probability of word sequence and tag sequence $P(W,T)$. In the tri-POS model, the joint probability is approximated by the product of parts of speech trigram probabilities $P(t_i|t_{i-2}, t_{i-1})$ and word output probabilities for given part of speech $P(w_i|t_i)$:

$$P(W,T) = \prod_{i=1}^{n} P(t_i|t_{i-2}, t_{i-1}) P(w_i|t_i) \qquad (1)$$

In practice, we consider sentence boundaries as special symbols as follows.

$$P(W,T) = P(t_1|\#)P(w_1|t_1)P(t_2|\#,t_1)P(w_2|t_2) \prod_{i=3}^{n} P(t_i|t_{i-2},t_{i-1})P(w_i|t_i)P(\#|t_{n-1},t_n) \quad (2)$$

where "#" indicates the sentence boundary marker. If we have some tagged text available, we can estimate the probabilities $P(t_i|t_{i-2},t_{i-1})$ and $P(w_i|t_i)$ by computing the relative frequencies of the corresponding events on this data:

$$P(t_i|t_{i-2},t_{i-1}) = f(t_i|t_{i-2},t_{i-1}) = \frac{N(t_{i-2},t_{i-1},t_i)}{N(t_{i-2},t_{i-1})} \quad (3)$$

$$P(w_i|t_i) = f(w_i|t_i) = \frac{N(w,t)}{N(t)} \quad (4)$$

where f indicates the relative frequency, $N(w,t)$ is the number of times a given word w appears with tag t, and $N(t_{i-2},t_{i-1},t_i)$ is the number of times that sequence $t_{i-2}t_{i-1}t_i$ appears in the text. It is inevitable to suffer from sparse-data problem in the part of speech tag trigram probability[1]. To handle open text, trigram probability is smoothed by ***interpolated estimation***, which simply interpolates trigram, bigram, unigram, and zerogram relative frequencies[8],

$$P(t_i|t_{i-2},t_{i-1}) = q_3 f(t_i|t_{i-2},t_{i-1}) + q_2 f(t_i|t_{i-1}) + q_1 f(t_i) + q_0 V \quad (5)$$

where f indicates the relative frequency and V is a uniform probability that each tag will occur. The nonnegative weights q_i satisfy $q_3 + q_2 + q_1 + q_0 = 1$, and they are adjusted so as to make the observed data most probable after the adjustment by using EM algorithm[2].

2.2 Order Reduction and Recursive Tracing

In order to understand the search algorithm described in the next section, we will introduce the second order HMM and extended Viterbi algorithm [6]. Considering the combined state sequence $U = u_1u_2\ldots u_n$, where $u_1 = t_1$ and $u_i = t_{i-1}t_i$, we have

$$P(u_i|u_{i-1}) = P(t_i|t_{i-2},t_{i-1}) \quad (6)$$

Substituting Equation (6) into Equation (1), we have

$$P(W,T) = \prod_{i=1}^{n} P(u_i|u_{i-1})P(w_i|t_i) \quad (7)$$

Equation (7) have the same form as the first order model. Consider the partial word sequence $W_i = w_1 \ldots w_i$ and the partial tag sequence $T_i = t_1 \ldots t_i$, we have

$$P(W_i,T_i) = P(W_{i-1},T_{i-1})P(u_i|u_{i-1})P(w_i|t_i) \quad (8)$$

Equation (8) suggests that, to find the maximum $P(W_i,T_i)$ for each u_i, we need only to: remember the maximum $P(W_{i-1},T_{i-1})$, extend each of these probabilities to every u_i by computing Equation (8), and select the maximum $P(W_i,T_i)$ for each u_i. Thus, by increasing i by 1 to n, selecting the u_n that maximize $P(W_n,T_n)$, and backtracing the sequence leading to the maximum probability, we can get the optimal tag sequence.

3 Search Strategy

The search algorithm consists of a forward dynamic programming search and a backward A* search. First, a linear time dynamic programming is used for recording the scores of all partial paths in a table[3]. A backward A^* algorithm based tree search is then used to extend the partial paths. Partial paths extended in the backward tree search are ranked by their corresponding full path scores, which are computed by adding the scores of backward partial path scores to the corresponding best possible scores of the remaining paths which are prerecorded in the forward search. Since the score of the incomplete portion of a path is exactly known, the backward search is admissible. That is, the top-N candidates are exact.

3.1 The Forward DP Search

Table 1 shows the two data structures used in our algorithm. The structure **parse** stores the information of a word and the best partial path up to the word. **Parse.start** and **parse.end** are the indices of the start and end positions of the word in the sentence. **Parse.pos** is the part of speech tag, which is a list of part of speech, conjugation type, and conjugation form in our system for Japanese. **Parse.nth-order-state** is a list of the last two parts of speech tags including that of the current word. This slot corresponds to the combined state in the second order HMM. **Parse.prob-so-far** is the score of the best partial path from the beginning of the sentence to the word. **Parse.previous** is the pointer to the (best) previous **parse** structure as in conventional Viterbi decoding, which is not necessary if we use the backward N best search.

[1]We used 120 part of speech tags. In the ATR Corpus, 26 parts of speech, 13 conjugation types, and 7 conjugation forms are defined. Out of 26, 5 parts of speech have conjugation. Since we used a list of part of speech, conjugation type, and conjugation form as a tag, there are 119 tags in the ATR Corpus. We added the sentence boundary marker to them.

[2]To handle open text, word output probability $P(w_i|t_i)$ must also be smoothed. This problem is discussed in a later section as the unknown word problem.

[3]In fact, we use two tables, **parse-list** and **path-map**. The reason is described later.

The structure **word** represents the word information in the dictionary including its lexical form, part of speech tag, and word output probability given the part of speech.

Table 1: Data structures for the N best algorithm

parse structure	
start	the beginning position of the word
end	the end position of the word
pos	part of speech tag of the word
nth-order-state	a list of the last two parts of speech
prob-so-far	the best partial path score from the start
previous	a pointer to previous parse structure

word structure	
form	lexical form of the word
pos	part of speech tag of the word
prob	word output probability

Before explaining the forward search, we will define some functions and tables used in the algorithm. In the forward search, we use a table called **parse-list**, whose key is the end position of the **parse** structure, and whose value is a list of **parse** structures that have the best partial path scores for each combined state at the end position. Function **register-to-parse-list** registers a **parse** structure against the **parse-list** and maintains the best partial parses. Function **get-parse-list** returns a list of **parse** structures at the specified position. We also use the function **leftmost-substrings** which returns a list of **word** structures in the dictionary whose lexical form matches the substrings starting at the specified position in the input sentence.

```
function forward-pass(string)
begin
initial-step(); # Pads special symbols at both ends.
for i=1 to length(string) do
  foreach parse in get-parse-list(i) do
    foreach word in leftmost-substrings(string,i) do
      pos-ngram := append(parse.nth-order-state,
                           list(word.pos))
      if (transprob(pos-ngram) > 0) then
        new-parse := make-parse();
        new-parse.start := i;
        new-parse.end := i + length(word.form);
        new-parse.pos := word.pos;
        new-parse.nth-order-state := rest(pos-ngram);
        new-parse.prob-so-far := parse.prob-so-far
                 * transprob(pos-ngram) * word.prob;
        new-parse.previous := parse;
        register-parse-to-parse-list(new-parse);
        register-parse-to-path-map(new-parse);
      endif
    end
  end
end
final-step(); # Handles transition to the end symbol.
end
```

Figure 1: The forward DP search algorithm

Figure 1 shows the central part of the forward dynamic programming search algorithm. It starts from the beginning of the input sentence, and proceeds character by character. At each point in the sentence, it looks up the combination of the best partial parses ending at the point and word hypotheses starting at that point. If the connection of a partial parse and a word hypothesis is allowed by the tagging model, a new continuation parse is made and registered in the **parse-list**. The partial path score for the new continuation parse is the product of the best partial path score up to the point, the trigram probability of the last three parts of speech tags and the word output probability for the part of speech[4].

3.2 The Backward A^* Search

The backward search uses a table called **path-map**, whose key is the end position of the **parse** structure, and whose value is a list of **parse** structures that have the best partial path scores for each distinct combination of the start position and the combined state. The difference between **parse-list** and **path-map** is that **path-map** is classified by the start position of the last word in addition to the combined state.

This distinction is crucial for the proposed N best algorithm. For the forward search to find a parse that maximizes Equation (1), it is the parts of speech sequence that matters. For the backward N-best search, however, we want N most likely word segmentation and part of speech sequence. **Parse-list** may shadow less probable candidates that have the same part of speech sequence for the best scoring candidate, but differ in the segmentation of the last word. As shown in Figure 1, **path-map** is made during the forward search by the function **register-parse-to-path-map**, which registers a **parse** structure to **path-map** and maintains the best partial parses in the table's criteria.

Now we describe the central part of the backward A^* search algorithm. But we assume that the readers know the A^* algorithm, and explain only the way we applied the algorithm to the problem.

We consider a **parse** structure as a state in A^* search. Two states are equal if their **parse** structures have the same start position, end position, and combined state. The backward search starts at the end of the input sentence, and backtracks to the beginning of the sentence using the **path-map**.

Initial states are obtained by looking up the entries of the sentence end position of the **path-map**. The successor states are obtained by first, looking up the entries of the **path-map** at the start position of the current **parse**, then checking whether they satisfy the constraint of the combined state transition in the second order HMM, and whether the transition is allowed by the tagging model. The combined state transition constraint means that the part of speech sequence in the **parse.nth-order-state** of the current **parse**, ignor-

[4]In Figure 1, function **transprob** returns the probability of given trigram. Functions **initial-step** and **final-step** treat the transitions at sentence boundaries.

ing the last element, equals that of the previous `parse`, ignoring the first element.

The state transition cost of the backward search is the product of the part of speech trigram probability and the word output probability. The score estimate of the remaining portion of a path is obtained from the **`parse.prob-so-far`** slot in the **`parse`** structure.

The backward search generates the N best hypotheses sequentially and there is no need to preset N. The complexity of the backward search is significantly less than that of the forward search.

4 Word Model

To handle open text, we have to cope with unknown words. Since Japanese do not put spaces between words, we have to identify unknown words at first. To do this, we can look at the spelling (character sequence) that may constitute a word, or look at the context to identify words that are acceptable in this context.

Once word hypotheses for unknown words are generated, the proposed N-best algorithm will find the most likely word segmentation and part of speech assignment taking into account the entire sentence. Therefore, we can formalize the unknown word problem as determining the span of an unknown word, assigning its part of speech, and estimating its probability given its part of speech.

Let us call a computational model that determines the probability of any word hypothesis given its lexical form and its part of speech the "word model". The word model must account for morphology and word formation to estimate the part of speech and the probability of a word hypothesis. For the first approximation, we used the character trigram of each part of speech as the word model.

Let $C = c_1c_2 \ldots c_n$ denote the sequence of n characters that constitute word w whose part of speech is t. We approximate the probability of the word given part of speech $P(w|t)$ by the trigram probabilities,

$$P(w|t) = P_t(C) = P_t(c_1|\#,\#)P_t(c_2|\#,c_1) \prod_{i=3}^{n} P_t(c_i|c_{i-2},c_{i-1})P_t(\#|c_{n-1},c_n) \qquad (9)$$

where special symbol "#" indicates the word boundary marker. Character trigram probabilities are estimated from the training corpus by computing relative frequency of character bigram and trigram that appeared in words tagged as t.

$$P_t(c_i|c_{i-2},c_{i-1}) = f_t(c_i|c_{i-2},c_{i-1}) = \frac{N_t(c_{i-2},c_{i-1},c_i)}{N_t(c_{i-2},c_{i-1})} \qquad (10)$$

where $N_t(c_{i-2},c_{i-1},c_i)$ is the total number of times character trigram $c_{i-2}c_{i-1}c_i$ appears in words tagged as t in the training corpus. Note that the character trigram probabilities reflect the frequency of word tokens in the training corpus. Since there are more than 3,000 characters in Japanese, trigram probabilities are smoothed by interpolated estimation to cope with the sparse-data problem.

It is ideal to make this character trigram model for all open class categories. However, the amount of training data is too small for low frequency categories if we divide it by part of speech tags. Therefore, we made trigram models only for the 4 most frequent parts of speech that are open categories and have no conjugation. They are common noun, proper noun, sahen noun[5], and numeral.

```
> (estimate-part-of-speech '都ホテル) ; Miyako Hotel
((固有名詞 2.7621915641723623E-7)      ; proper noun
 (普通名詞 6.3406095003694205E-9)      ; common noun
 (サ変名詞 5.840424519473811E-19)      ; sahen noun
 (数詞 5.7364195413101E-29))            ; numeral
> (estimate-part-of-speech '１９９４)
((数詞 1.8053860295767367E-6)           ; numeral
 (固有名詞 6.512248681540477E-17)      ; proper noun
 (普通名詞 2.288684007246524E-17)      ; common noun
 (サ変名詞 7.50515322380211E-20))      ; sahen noun
```

Figure 2: N-best Tags for Unknown Words

Figure 2 show two examples of part of speech estimation for unknown words. Each trigram model returns a probability if the input string is a word belonging to the category. In both examples, the correct category has the largest probability.

```
> (get-leftmost-substrings-with-word-model
                        "転送してもらって下さい。")
((転 サ変名詞 2.519457597358691E-7)
 (転 固有名詞 2.3449215070189967E-8)
 (転 普通名詞 7.02439907471337745E-9)
 (転 数詞 2.375650975098567E-9)
 (転送 サ変名詞 5.706874990251415E-10)
 (転送 普通名詞 4.735628004876359E-13)
 (転送 固有名詞 8.928942348107183E-14)
 (転送し サ変名詞 7.266613344265452E-14)
 (転送 数詞 6.866499496132O7E-16)
 (転送し 普通名詞 2.4530239052513518E-17))
```

Figure 3: N-Best Word Hypotheses

Figure 3 shows the N-best word hypotheses generated by using the character trigram models. A word hypothesis is a list of word boundary, part of speech assignment, and word probability that matches the left-most substrings starting at a given position in the input sentence. In the forward search, to handle unknown words, word hypotheses are generated at every position in addition to the ones generated by the function `leftmost-substrings`, which are the words found in the dictionary. However, in our system, we limited the number of word hypotheses generated at each position to 10, for efficiency reasons.

[5] A noun that can be used as a verb when it is followed by a formal verb "*suru*".

5 Evaluation Measures

We applied the performance measures for English parsers [1] to Japanese morphological analyzers. The basic idea is that morphological analysis for a sentence can be thought of as a set of labeled brackets, where a bracket corresponds to word segmentation and its label corresponds to part of speech. We then compare the brackets contained in the system's output to the brackets contained in the standard analysis. For the N-best candidate, we will make the union of the brackets contained in each candidate, and compare them to the brackets in the standard.

For comparison, we count the number of brackets in the standard data (Std), the number of brackets in the system output (Sys), and the number of matching brackets (M). We then calculate the measures of *recall* (= M/Std) and *precision* (= M/Sys). We also count the number of *crossings*, which is the number of cases where a bracketed sequence from the standard data overlaps a bracketed sequence from the system output, but neither sequence is completely contained in the other.

We defined two equality criteria of brackets for counting the number of matching brackets. Two brackets are *unlabeled-bracket-equal* if the boundaries of the two brackets are the same. Two brackets are *labeled-bracket-equal* if the labels of the brackets are the same in addition to unlabeled-bracket-equal. In comparing the consistency of the word segmentations of two bracketings, which we call *structure-consistency*, we count the measures (recall, precision, crossings) by unlabeled-bracket-equal. In comparing the consistency of part of speech assignment in addition to word segmentation, which we call *label-consistency*, we count them by labeled-bracket-equal.

```
> (morph-n-best "それでは登録用紙をお送り致します。")
-31.90894138309038
それでは / 接続詞 登録 / 普通名詞 用紙 / 普通名詞
 を / 格助詞 お / 接頭語 送り / 本動詞・連用・五段
 致し / 補助動詞・連用・五段 ます / 助動詞・終止 。 / 記号
-38.59433836658236
それ / 代名詞 で / 格助詞 は / 係助詞 登録 / 普通名詞
 用紙 / 普通名詞 を / 格助詞 お / 接頭語 送り / 本動詞・連用・五段
 致し / 補助動詞・連用・五段 ます / 助動詞・終止 。 / 記号
-43.10567483646801
それでは / 接続詞 登録 / 普通名詞 用紙 / 普通名詞
 を / 格助詞 お / 接頭語 送り / 本動詞・連用・五段
 致し / 本動詞・連用・五段 ます / 助動詞・終止 。 / 記号
```

Figure 4: N-Best Morphological Analysis hypotheses

For example, Figure 4 shows a sample of N-best analysis hypotheses, where the first candidate is the correct analysis[6]. For the second candidate, since there are 9 brackets in the correct data (Std=9), 11 brackets in the second candidate (Sys=11), and 8 matching brackets (M=8), the recall and precision with respect to label consistency are 8/9 and 8/11, respectively. For the top two candidates, since there are 12 distinct brackets in the systems output and 9 matching brackets, the recall and precision with respect to label consistency are 9/9 and 9/12, respectively. For the third candidate, since the correct data and the third candidate differ in just one part of speech tag, the recall and precision with respect to structure consistency are 9/9 and 9/9, respectively.

[6]Probabilities are in natural log base e.

6 Experiment

Table 2: The amount of training and test data

	training texts	closed test	open test
Sentences	10945	1000	1000
Words	149059	13176	13899
Characters	267422	94221	98997

We used the ATR Dialogue Database[5] to train and test the proposed morphological analysis method. It is a corpus of approximately 800,000 words whose word segmentation and part of speech tag assignment were laboriously performed by hand. In this experiment, we only used one fourth of the ATR Corpus, a portion of the keyboard dialogues in the conference registration domain. First, we selected 1,000 test sentences for an open test, and used the others for training. The corpus was divided into 90% for training and 10% for testing. We then selected 1,000 sentences from the training set and used them for a closed test. The number of sentences, words, and characters for each test set and training texts are shown in Table 2.

The training texts contained 6580 word types and 6945 tag trigram types. There were 247 unknown word types and 213 unknown tag trigram types in the open test sentences. Thus, both part of speech trigram probabilities and word output probabilities must be smoothed to handle open texts.

Table 3: Percentage of words correctly segmented and tagged: raw part of speech bigram and trigram

	bigram (closed text)			trigram (closed text)		
	recall	prec.	cross.	recall	prec.	cross.
1	96.2%	96.6%	0.001	97.5%	97.8%	0.001
2	98.0%	89.7%	0.004	99.0%	90.7%	0.007
3	98.9%	83.5%	0.010	99.5%	84.3%	0.012
4	99.2%	78.5%	0.013	99.7%	79.6%	0.015
5	99.4%	74.2%	0.017	99.8%	76.0%	0.015

First, as a preliminary experiment, we compared the performances of part of speech bigram and trigram. Table 3 shows the percentages of words correctly segmented and tagged, tested on the closed test sentences. The trigram model achieved 97.5% recall and 97.8% precision for the top candidate, while the bigram model achieved 96.2% recall and 96.6% precision. Although both tagging models show very high performance, the trigram model outperformed the bigram model in every metric.

We then tested the proposed system, which uses smoothed part of speech trigram with word model, on the open test sentences. Table 4 shows the percentages of words correctly segmented and tagged. In Table 4, label consistency 2 represents the accuracy of segmentation and tagging ignoring the difference in conjugation form.

For open texts, the morphological analyzer achieved 95.1% recall and 94.6% precision for the top candidate, and 97.8% recall and 73.2% precision for the 5 best candidates. This performance is very encouraging, and is comparable to the state-of-the-art stochastic tagger for English [2–4, 10, 11].

Since the segmentation accuracy of the proposed system is relatively high (97.7% recall and 97.2% precision for the top candidate) compared to the morphological analysis accuracy, it is likely that we can improve the part of speech assignment accuracy by refining the statistically-based tagging model. We find a fair number of tagging errors happened in conjugation forms. We assume that this is caused by the fact that the Japanese tag set used in the ATR Corpus is not detailed enough to capture the complicated Japanese verb morphology.

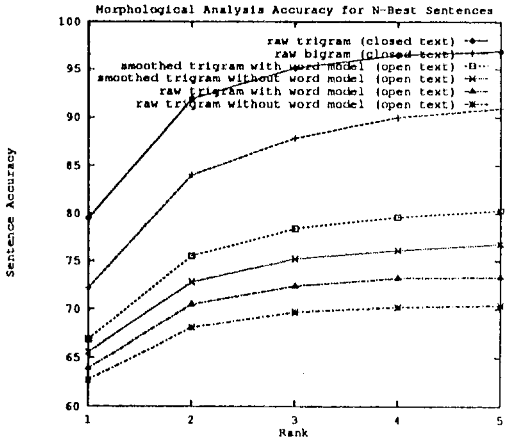

Figure 5: The percentage of sentences correctly segmented and tagged.

Figure 5 shows the percentage of sentences (not words) correctly segmented and tagged. For open texts, the sentence accuracy of the raw part of speech trigram without word model is 62.7% for the top candidate and 70.4% for the top-5, while that of smoothed trigram with word model is 66.9% for the top and 80.3% for the top-5. We can see that, by smoothing the part of speech trigram and by adding word model to handle unknown words, the accuracy and robustness of the morphological analyzer is significantly improved. However, the sentence accuracy for closed texts is still significantly better that that for open texts. It is clear that more research has to be done on the smoothing problem.

7 Discussion

Morphological analysis is an important practical problem with potential application in many areas including *kana-to-kanji* conversion[7], speech recognition, character recognition, speech synthesis, text revision support, information retrieval, and machine translation.

Most conventional Japanese morphological analyzers use rule-based heuristic searches. They usually use a connectivity matrix (part-of-speech-pair grammar) as the language model. To rank the morphological analysis hypotheses, they usually use heuristics such as Longest Match Method or Least *Bunsetsu*'s Number Method [16].

There are some statistically-based approaches to Japanese morphological analysis. The tagging models previously used are either part of speech bigram [9, 14] or Character-based HMM [12].

Both heuristic-based and statistically-based approaches use the Minimum Connective-Cost Method [7], which is a linear time dynamic programming algorithm that finds the morphological hypothesis that has the minimal connective cost (i.e. bigram-based cost) as derived by certain criteria.

To handle unknown words, most Japanese morphological analyzers use character type heuristics [17], which is "a string of the same character type is likely to constitute a word". There is one stochastic approach that uses bigram of word formation unit [13]. However, it does not learn probabilities from training texts, but learns them from machine readable dictionaries, and the model is not incorporated in working morphological analyzers, as far as the author knows.

The unique features of the proposed Japanese morphological analyzer is that it can find the exact N most likely hypotheses using part of speech trigram, and it can handle unknown words using character trigram. The algorithm can naturally be extended to handle any higher order Markov models. Moreover, it can naturally be extended to handle lattice-style input that is often used as the output of speech recognition and character recognition systems, by extending the function (`leftmost-substrings`) so as to return a list of words in the dictionary that matches the substrings in the input lattice starting at the specified position.

For future work, we have to study the most effective way of generating word hypotheses that can handle unknown words. Currently, we are limiting the number of word hypotheses to reduce ambiguity at the cost of accuracy. We have also to study the word model for open categories that have conjugation, because the training

[7] *Kana-to-kanji* conversion is a popular Japanese input method on computer using ASCII keyboard. Phonetic transcription by Roman (ASCII) characters are input and converted first to the Japanese syllabary *hiragana* which is then converted to orthographic transcription including Chinese character *kanji*.

Table 4: The percentage of words correctly segmented and tagged: smoothed trigram with word model

	smoothed trigram with word model (open text)								
	label consistency			label consistency 2			structure consistency		
	recall	precision	crossings	recall	precision	crossings	recall	precision	crossings
1	95.1%	94.6%	0.013	95.9%	95.4%	0.013	97.7%	97.2%	0.013
2	96.5%	88.0%	0.023	97.0%	90.3%	0.023	98.2%	94.4%	0.022
3	97.3%	82.1%	0.031	97.6%	85.1%	0.031	98.5%	91.7%	0.029
4	97.6%	77.4%	0.046	97.9%	80.7%	0.046	98.7%	89.6%	0.044
5	97.8%	73.2%	0.061	98.1%	77.1%	0.060	98.8%	87.9%	0.056

data gets too small to make trigrams if we divide it by tags. We will probably have to tie some parameters to solve the insufficient data problem.

Moreover, we have to study the method to adapt the system to a new domain. To develop an unsupervised learning method, like the forward-backward algorithm for HMM, is an urgent goal, since we can't always expect the availability of manually segmented and tagged data. We can think of an EM algorithm by replacing maximization with summation in the extended Viterbi algorithm, but we don't know how to handle unknown words in this algorithm.

8 Conclusion

We have developed a stochastic Japanese morphological analyzer. It uses a statistical tagging model and an efficient two-pass search algorithm to find the N best morphological analysis hypotheses for the input sentence. Its word segmentation and tagging accuracy is approximately 95%, which is comparable to the state-of-the-art stochastic tagger for English.

References

[1] Black, E. et al.: "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars", DARPA Speech and Natural Language Workshop, pp.306-311, Morgan Kaufmann, 1991.

[2] Charniak, E., Hendrickson, C., Jacobson, N., and Perkowitz, M.: "Equations for Part-of-Speech Tagging", AAAI-93, pp.784-789, 1993.

[3] Church, K.: "A Stochastic Part of Speech Tagger and Noun Phrase Parser for English", ANLP-88, pp.136-143, 1988.

[4] Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P.: "A Practical Part-of-Speech Tagger", ANLP-92, pp.133-140, 1992.

[5] Ehara, T., Ogura, K. and Morimoto, T.: "ATR Dialogue Database," ICSLP-90, pp.1093-1096, 1990.

[6] He, Y.: "Extended Viterbi Algorithm for Second Order Hidden Markov Process", ICPR-88, pp.718-720, 1988.

[7] Hisamitsu, T. and Nitta, Y.: "Morphological Analysis by Minimum Connective-Cost Method", Technical Report SIGNLC 90-8, IEICE, pp.17-24, 1990 (in Japanese).

[8] Jelinek, F.: "Self-organized language modeling for speech recognition", IBM Report, 1985 (Reprinted in *Readings in Speech Recognition*, pp.450-506).

[9] Matsunobu, E., Hitaka, T., and Yoshida, S.: "Syntactic Analysis by Stochastic BUNSETSU Grammar", Technical Report SIGNL 56-3, IPSJ, 1986 (in Japanese).

[10] Merialdo, B.: "Tagging Text with a Probabilistic Model", ICASSP-91, pp.809-812, 1991.

[11] Meteer, M. W., Schwartz, R. and Weischedel, R.: "POST: Using Probabilities in Language Processing", IJCAI-91, pp.960-965, 1991.

[12] Murakami, J. and Sagayama, S.: "Hidden Markov Model applied to Morphological Analysis", 45th National Meeting of the IPSJ, Vol.3, pp.161-162, 1992 (in Japanese).

[13] Nagai, H. and Hitaka, T.: "Japanese Word Formation Model and Its Evaluation", Trans IPSJ, Vol.34, No.9, pp.1944-1955, 1993 (in Japanese).

[14] Sakai, S.: "Morphological Category Bigram: A Single Language Model for both Spoken Language and Text", ISSD-93, pp.87-90, 1993.

[15] Soong, F. K. and Huang E.: "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", ICASSP-91, pp.705-708, 1991.

[16] Yoshimura, K, Hitaka, T., and Yoshida, S.: "Morphological Analysis of Non-marked-off Japanese Sentences by the Least BUNSETSU's Number Method", Trans. IPSJ, Vol.24, No.1, pp.40-46, 1983 (in Japanese).

[17] Yoshimura, K., Takeuchi, M., Tsuda, K. and Shudo, K.: "Morphological Analysis of Japanese Sentences Containing Unknown Words", Trans. IPSJ, Vol.30, No.3, pp.294-301, 1989 (in Japanese).