Context-Based Spelling Correction for Japanese OCR

Masaaki NAGATA
NTT Information and Communication Systems Laboratories
1-2356 Take, Yokosuka-Shi, Kanagawa, 238-03 Japan
nagata@nttnly.isl.ntt.jp

Abstract

We present a novel spelling correction method for those languages that have no delimiter between words, such as Japanese, Chinese, and Thai. It consists of an approximate word matching method and an N-best word segmentation algorithm using a statistical language model. For OCR errors, the proposed word-based correction method outperforms the conventional character-based correction method. When the baseline character recognition accuracy is 90%, it achieves 96.0% character recognition accuracy and 96.3% word segmentation accuracy, while the character recognition accuracy of character-based correction is 93.3%.

1 Introduction

Automatic spelling correction research dates back in the 1960s. Today, there are some excellent academic and commercial spell checkers available for English (Kukich, 1992). However, for those languages that have a different morphology and writing system from English, spelling correction remains one of the significant unsolved research problems in computational linguistics.

The basic strategy for English spelling correction is simple: Word boundaries are defined by white space characters. If the tokenized string is not found in the dictionary, it is either a non-word or an unknown word. For a non-word, correction candidates are generated by approximately matching the string with the dictionary, using context-independent word distance measures such as edit distance (Wagner and Fischer, 1974; Kernighan et al., 1990).

It is impossible to apply these "isolated word error correction" techniques to Japanese in two reasons: First, in noisy texts, word tokenization is difficult because there are no delimiters between words. Second, context-independent word distance measures are useless because the average word length is very short (< 2), and the character set is huge (> 3000). There are a large number of one edit distance neighbors for a Japanese word.

In English spelling correction, "word boundary problem", such as splits (forgot → *for got*) and run-ons (in form → *inform*), and "short word problem"(*ot* → on, or, of, at, it, to, etc.) are also known to be very difficult. Context information, such as word N-gram, is used to supplement the underlying context-independent correction for these problematic examples (Gale and Church, 1990; Mays et al., 1991). To the contrary, Japanese spelling correction must be essentially context-dependent, because Japanese sentence is, as it were, a run-on sequence of short words, possibly including some typos, something like (*Iforgotoinformyou* → I forgot to inform you).

In this paper, we present a novel approach for spelling correction, which is suitable for those languages that have no delimiter between words, such as Japanese. It consists of two stages: First, all substrings in the input sentence are hypothesized as words, and those words that approximately matched with the substrings are retrieved from the dictionary as well as those that exactly matched. Based on the statistical language model, the N-best word sequences are then selected as correction candidates from all combinations of exactly and approximately matched words. Figure 1 illustrates this approach. Out of the list of character recognition candidates for the input sentence "申し込み用紙に必要事項を記入する。", which means "to fill out the necessary items in the application form.", the system searches the combination of exactly matched words (solid boxes) and approximately matched words (dashed boxes) [1].

The major contribution of this paper is its solutions of the word boundary problem and short word problem in Japanese spelling correction. By introducing a statistical model of word

[1] OCR output tends to be very noisy, especially for hand writing. To compensate for this behavior, OCRs usually output an ordered list of the best N characters. The list of the candidates for an input string is called character matrix.

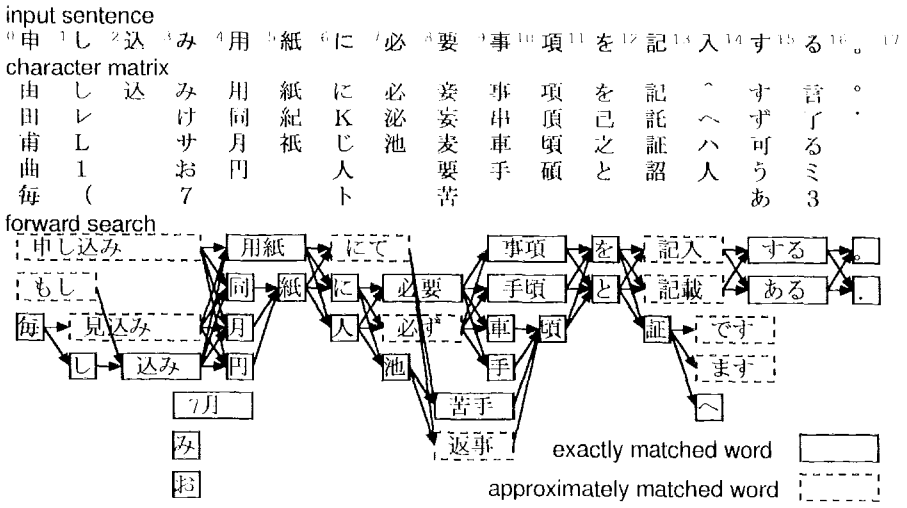

Figure 1: Possible Combinations of Exactly and Approximately Matched Words

length and spelling, the proposed system accurately places word boundaries in noisy texts that include non-words and unknown words. By using the character-based context model, it accurately selects correction candidates for short words from the large number of approximately matched words with the same edit distance.

The goal of our project is to implement an interactive word corrector for a handwritten FAX-OCR system. We are especially interested in texts that include addresses, names, and messages, such as order forms, questionnaires, and telegraph.

2 Noisy Channel Model for Character Recognition

First, we formulate the spelling correction of OCR errors in the noisy channel paradigm. Let C represent the input string and X represent the OCR output string. Finding the most probable string C given the OCR output X amounts to maximizing the function $P(X|C)P(C)$,

$$\hat{C} = \arg\max_C P(C|X) = \arg\max_C P(X|C)P(C) \quad (1)$$

Because Bayes' rule states that,

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)} \quad (2)$$

$P(C)$ is called the language model. It is computed from the training corpus. Let us call $P(X|C)$ the OCR model. It can be computed from the a priori likelihood estimates for individual characters,

$$P(X|C) = \prod_{i=1}^{n} P(x_i|c_i) \quad (3)$$

where n is the string length. $P(x_i|c_i)$ is called the confusion matrix of characters. It is trained using the input and output strings of the OCR.

The confusion matrix is highly dependent on the character recognition algorithm and the quality of the input document. It is a labor intensive task to prepare a confusion matrix for each character recognition system, since Japanese has more than 3,000 characters. Therefore, we used a simple OCR model where the confusion matrix is approximated by the correct character distribution over the rank of the candidates. We assume that the rank order distribution of the correct characters is a geometric distribution whose parameter is the accuracy of the first candidate.

Let c_i be the i-th character in the input string, x_{ij} be the j-th candidate for c_i, and p be the probability that the first candidate is correct. The confusion probability $P(x_{ij}|c_i)$ is approximated as,

$$P(x_{ij}|c_i) \approx P(x_{ij} \text{ is correct}) \approx p(1-p)^{j-1} \quad (4)$$

Equation (4) aims to approximate the accuracy of the first candidate and the tendency that the reliability of the candidate decreases abruptly as its rank increases. For example, if the recognition accuracy of the first candidate p is 0.75, we will assign the probability of the first, second, and third candidates to 0.75, 0.19, and 0.05, respectively, regardless of the input and output characters.

One of the benefits of using a simple OCR model is that the spelling correction system becomes highly independent of the underlying OCR characteristics. Obviously, a more sophisticated OCR model would improve error correction performance, but even this simple OCR model works fairly well in our experiments [2].

[2] One of the practical reasons for using the geometric distribution is that we used the confusion matrix for implementing the OCR simulator. We feel it is unfair to use the same confusion matrix both for error generation and error correction.

3 Word Segmentation Algorithm

3.1 Statistical Language Model

For the language model in Equation (1), we used the part of speech trigram model (POS trigram or 2nd-order HMM). It is used as tagging model in English (Church, 1988; Cutting et al., 1992) and morphological analysis model (word segmentation and tagging) in Japanese (Nagata, 1994).

Let the input character sequence be $C = c_1 c_2 \ldots c_m$. We approximate $P(C)$ by $P(W,T)$, the joint probability of word sequence $W = w_1 w_2 \ldots w_n$ and part of speech sequence $T = t_1 t_2 \ldots t_n$. $P(W,T)$ is then 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 $P(w_i|t_i)$,

$$P(C) \approx P(W,T) = \prod_{i=1}^{n} P(t_i|t_{i-2},t_{i-1})P(w_i|t_i) \quad (5)$$

$P(t_i|t_{i-2},t_{i-1})$ and $P(w_i|t_i)$ are estimated by computing the relative frequencies of the corresponding events in training corpus [3].

3.2 Forward-DP Backward-A^* Algorithm

Using the language model (5), Japanese morphological analysis can be defined as finding the set of word segmentation and parts of speech $(\hat{W},\hat{T})$ that maximizes the joint probability of word sequence and tag sequence $P(W,T)$.

$$(\hat{W},\hat{T}) = \arg\max_{W,T} P(W,T) \quad (6)$$

This maximization search can be efficiently implemented by using the forward-DP backward-A^* algorithm (Nagata, 1994). It is a natural extension of the Viterbi algorithm (Church, 1988; Cutting et al., 1992) for those languages that do not have delimiters between words, and it can generate N-best morphological analysis hypotheses, like tree-trellis search (Soong and Huang, 1991).

The algorithm consists of a forward dynamic programming search and a backward A^* search. The forward search 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 the point. If the connection between a partial parse and a word hypothesis is allowed by the language model, that is, the corresponding part of speech trigram probability is positive, a new continuation parse is made and registered in the best partial path table. For example, at point 4 in Figure 1, the final word of the partial parses ending at 4 are 申し込み ('application'), 見込み ('prospect'), and 込み ('inclusive'), while the word hypotheses starting at 4 are 用紙 ('form'), 同 ('same'), 月 ('moon'), and 円 ('circle').

In the backward A^* search, we consider a partial parse recorded in the best partial path table as a state in A^* search. The backward search starts at the end of the input sentence, and backtracks to the beginning of the sentence. Since the probabilities of the best possible remaining paths are exactly known by the forward search, the backward search is admissible.

We made two extensions to the original forward-DP backward-A^* algorithm to handle OCR outputs. First, it retrieves all words in the dictionary that match the strings which consist of a combination of the characters in the matrix. Second, the path probability is changed to the product of the language model probability and the OCR model probability, so as to get the most likely character sequence, according to Equation (1).

4 Word Model for Non-Words and Unknown Words

The identification of non-words and unknown words is a key to implement Japanese spelling corrector, because word identification error severely affects the segmentation of neighboring words.

We take the following approach for this word boundary problem. We first hypothesize all substrings in the input sentence as words, and assign a reasonable non-zero probability. For example, at point 7 in Figure 1, other than the exactly and approximately matched words starting at 7 such as 必要 ('necessary'), 必ず ('necessarily'), and 池 ('pond'), we hypothesize the substrings 必, 必妾, 必妾事, 必妾事項, ... as words. We then locate the most likely word boundaries using the forward-DP backward-A^* algorithm, taking into account the entire sentence.

We use a statistical word model to assign a probability to each substring (Nagata, 1996). It is defined as the joint probability of the character sequence if it is an unknown word. Without loss of generality, we can write,

$$\begin{aligned} P(w_i|\texttt{<UNK>}) &= P(c_1 \ldots c_k|\texttt{<UNK>}) \\ &= P(k)P(c_1 \ldots c_k|k) \end{aligned} \quad (7)$$

where $c_1 \ldots c_k$ is the character sequence of length k that constitutes word w_i. We call $P(k)$ the word length model, and $P(c_1 \ldots c_k|k)$ the spelling model.

We assume that word length probability $P(k)$ obeys a Poisson distribution whose parameter is the average word length λ,

$$P(k) = \frac{(\lambda-1)^{k-1}}{(k-1)!} e^{-(\lambda-1)} \quad (8)$$

This means that we think word length is the interval between hidden word boundary markers, which are randomly placed where the average interval equals the average word length. Although

[3] As a word segmentation model, the advantage of the POS trigram model is that it can be trained using a smaller corpus, than the word bigram model.

this word length model is very simple, it plays a key role in making the word segmentation algorithm robust.

We approximate the spelling probability given word length $P(c_1 \ldots c_k|k)$ by the word-based character trigram model, regardless of word length. Since there are more than 3,000 characters in Japanese, the amount of training data would be too small if we divided them by word length.

$$P(c_1 \ldots c_k) = P(c_1|\#,\#)P(c_2|\#,c_1) \prod_{i=3}^{k} P(c_i|c_{i-2}c_{i-1})P(\#|c_{k-1},c_k) \quad (9)$$

where "#" indicates the word boundary marker.

Note that the word-based character trigram model is different from the sentence-based character trigram model. The former is estimated from the corpus which is segmented into words. It assigns large probabilities to character sequences that appear within a word, and small probabilities to those that appear across word boundaries.

5 Approximate Match for Correction Candidates

As described before, we hypothesize all substrings in the input sentence as words, and retrieve approximately matched words from the dictionary as correction candidates. For a word hypothesis, correction candidates are generated based on the minimum edit distance technique (Wagner and Fischer, 1974). Edit distance is defined as the minimum number of editing operations (insertions, deletions, and substitutions) required to transform one string into another. If the target is OCR output, we can restrict the type of errors to substitutions only. Thus, the similarity of two words can be computed as c/n, where c is the number of matched characters and n is the length of the misspelled (and dictionary) word.

For longer words ($\geq$ 3 characters), it is reasonable to generate correction candidates by retrieving all words in the dictionary with similarity above a certain threshold ($c/n \geq 0.5$). For example, at point 0 in Figure 1, 申し込み ('application') is retrieved by approximately matching the string 由し込み with the dictionary ($c/n = 3/4 = 0.75$).

However, for short words (1 or 2 character word), this strategy is unrealistic because there are a large number of words with one edit distance. Since the total number of one character words and two character words amounts to more than 80% of the total word tokens in Japanese, we cannot neglect these short words.

It is natural to resort to context-dependent word correction methods to overcome the short word problem. In English, (Gale and Church, 1990) achieved good spelling check performance using word bigrams. However, in Japanese, we cannot use word bigram to rank correction candidates, because we have to rank them before we perform word segmentation.

Therefore, we used character context instead of word context. For a short word, correction candidates with the same edit distance are ranked by the joint probability of the previous and the following two characters in the context. This probability is computed using the sentence-based character trigram model. For 2 character words, for example, we first retrieve a set of words in the dictionary that match exactly one character with the one in the input string. We then compute the 6-gram probability for all candidate words $s_i s_{i+1}$, and rank them according to the probability.

$$P(c_{i-2},c_{i-1},s_i,s_{i+1},c_{i+2},c_{i+3}) = P(s_i|c_{i-2},c_{i-1})P(s_{i+1}|c_{i-1},s_i) P(c_{i+2}|s_i,s_{i+1})P(c_{i+3}|s_{i+1},c_{i+2}) \quad (10)$$

For example, at point 12 in Figure 1, there are many two character words whose first character is 記, such as 記載 ('mention'), 記事 ('article'), 記者 ('journalist'), 記入 ('entry'), 記念 ('commemoration'), etc. By using character contexts, the system selects 記入 and 記載 as approximately matched word hypotheses.

6 Experiments

6.1 Language Data and OCR Simulator

We used the ATR Dialogue Database (Ehara et al., 1990) to train and test the spelling correction method. It is a corpus of approximately 800,000 words whose word segmentation and part of speech tagging were laboriously performed by hand. In this experiment, we used one fourth of the ATR Corpus, a portion of the keyboard dialogues in the conference registration domain. Table 1 shows the number of sentences, words, and characters for training and test data. The test data is not included in the training data. That is, open data were tested in the experiment.

Table 1: The Amount of Training and Test Data

	Training set	Test set
Sentences	10945	100
Words	150039	1134
Characters	268830	2097

For the spelling correction experiment, we used an OCR simulator because it is very difficult to obtain a large amount of test data with arbitrary recognition accuracies. The OCR simulator takes an input string and generates a character matrix using a confusion matrix for Japanese handwriting OCR, developed in our laboratory. The parameters of the OCR simulator are the recognition accuracy of the first candidate (first candidate correct rate), and the percentage of the correct char-

acters included in the character matrix (correct candidate included rate).

In general, the accuracy of current Japanese handwriting OCR is around 90%. It is lower than that of printed characters (around 98%) due to the wide variability in handwriting. When the input comes from FAX, it degrades another 10% to 15%, because the resolution of most FAX machines is 200dpi, while that of scanners is 400dpi. Therefore, we made four test sets of character matrices whose first candidate correct rates and correct candidate included rates were (70%, 90%), (80%, 95%), (90%, 98%), and (95%, 98%), respectively. The average number of candidates for a character was 8.9 in these character matrices [4].

6.2 Character Recognition Accuracy

First, we compared the proposed word-based spelling corrector using the POS trigram model (POS3) with the conventional character-based spelling corrector using the character trigram model (Char3). Table 2 shows the character recognition accuracies after error correction for various baseline OCR accuracies. We also changed the condition of the approximate word match. In Table 2, Match1, Match2, and Match3 represent that the approximate match for substrings whose lengths were more than or equal to one, two, and three characters, respectively.

In general, the approximate match for short words improves character recognition accuracy by about one percent. When the first candidate correct rate is low (70% and 80%), the word-based corrector significantly outperforms the character-based corrector. This is because, by approximate word matching, the word-based corrector can correct words even if the correct characters are not present in the matrix. When the first candidate correct rate is high (90% and 95%), the word-based corrector still outperforms the character-based corrector, although the difference is small. This is because most correct characters are already included in the matrix.

Table 2: Comparison of Character Recognition Accuracy (Character Trigram vs. POS trigram)

OCR	Char3	POS3		
		Match1	Match2	Match3
70% (90%)	74.4%	84.6%	83.9%	83.1%
80% (95%)	86.0%	92.5%	92.0%	90.6%
90% (98%)	93.3%	96.0%	95.9%	95.6%
95% (98%)	95.0%	96.6%	96.7%	95.9%

[4]The parameters are selected considering the fact that the correct candidate included rate increases as the first candidate correct rate increases, and that some correct characters are never present in the matrix even if the first candidate correct rate is high.

6.3 Word Segmentation and Word Correction Accuracy

First, we define the performance measures of Japanese word segmentation and word correction. We will think of the output of the spelling corrector as a set of 2-tuples, word segmentation and orthography. We then compare the tuples contained in the system's output to the tuples contained in the standard analysis. For the N-best candidate, we will make the union of the tuples contained in each candidate, in other words, we will make a word lattice from N-best candidates, and compare them to the tuples in the standard. For comparison, we count the number of tuples in the standard (Std), the number of tuples in the system output (Sys), and the number of matching tuples (M). We then calculate *recall* (M/Std) and *precision* (M/Sys) as accuracy measures.

We define two degrees of equality among tuples for counting the number of matching tuples. For word segmentation accuracy, two tuples are equal if they have the same word segmentation regardless of orthography. For word correction accuracy, two tuples are equal if they have the same word segmentation and orthography.

Table 3 shows the words segmentation accuracy and word correction accuracy. The word segmentation accuracy of the spelling corrector is significantly high, even if the input is very noisy. For example, when the accuracy of the baseline OCR is 80%, since the average number of characters and words in the test sentences are 20.1 and 11.3, there are 4.0 (=20.1*(1-0.80)) character errors in the sentence, in average. However, 94.5% word segmentation recall means that there are only 0.62 (=11.3*(1-0.945)) word segmentations that are not found in the first candidate.

Moreover, we feel the word correction accuracy in Table 3 is satisfactory for an interactive spelling corrector. For example, when the accuracy of the baseline OCR is 90%, there are 2.0 (=20.1*(1-0.90)) character errors in the test sentence. However, 92.8% recall for the first candidate and 95.6% recall for the top-5 candidates means that there are only 0.81 (11.3*(1-0.928)) words that are not found in the first candidate, and if you examine the top-5 candidates, this value is reduced to 0.50 (11.3*(1-0.956)). That is, about half of the errors in the first candidate are corrected by simply selecting the alternatives in the word lattice.

7 Discussion

Previous works on Japanese OCR error correction are based on either the character trigram model or the part of speech bigram model. Their targets are printed characters, not handwritten characters. That is, they assume the underlying OCR's accuracy is over 90%. Moreover, their treatment of unknown words and short words is rather ad hoc.

Table 3: Word Segmentation Accuracy and Word Correction Accuracy for Noisy Texts

OCR		Word Segmentation				Word Correction			
		Recall (Best-5)		Precision (Best-5)		Recall (Best-5)		Precision(Best-5)	
70%	(90%)	89.0%	(92.1%)	82.3%	(75.2%)	77.1%	(82.4%)	71.3%	(58.2%)
80%	(95%)	94.5%	(97.4%)	90.5%	(81.7%)	87.9%	(92.6%)	84.2%	(67.2%)
90%	(98%)	96.3%	(97.9%)	93.6%	(85.8%)	92.8%	(95.6%)	90.1%	(72.1%)
95%	(98%)	97.3%	(98.6%)	94.8%	(86.8%)	94.3%	(97.0%)	91.8%	(74.0%)

(Takao and Nishino, 1989) used part of speech bigram and best first search for OCR correction. They used heuristic templates for unknown words. (Ito and Maruyama, 1992) used part of speech bigram and beam search in order to get multiple candidates in their interactive OCR corrector [5]

The proposed Japanese spelling correction method uses part of speech trigram and N-best search. This combination is theoretically and practically more accurate than previous methods. In addition, by using statistical word model, and context-based approximate word match, it becomes robust enough to handle very noisy texts, such as the output of FAX-OCR systems.

To improve the word correction accuracy, more powerful language models, such as word bigram, are required. (Jelinek, 1985) pointed out that "POS (part of speech) classification is too crude and not necessarily suited to language modeling". However, it is too expensive to prepare a large manually segmented corpus of each target domain to compute the word bigram. Therefore, we are thinking of making a self-organized word segmentation method by generalizing the Forward-Backward algorithm for those languages that have no delimiter between words (Nagata, 1996).

8 Conclusion

We have presented a spelling correction method for noisy Japanese texts. We are currently building an interactive Japanese spelling corrector *jspell*, where words are the basic object manipulated by the user in operations such as replace, accept, and edit. It is something like the Japanese counterpart of Unix's spelling corrector *ispell*, with a user interface similar to *kana-to-kanji* converter, a popular Japanese input method for the ASCII keyboard.

[5] According to Fig. 6 in (Takao and Nishino, 1989), they achieved about 95% character recognition accuracy when the baseline accuracy is 91% for magazines and introductory textbooks of science and technology domain. According to Table. 1 in (Ito and Maruyama, 1992), they achieved 94.61% character recognition accuracy when the baseline accuracy is 87.46% for patents in electric engineering domain. We achieved 96.0% character recognition accuracy, when the baseline accuracy is 90% in the conference registration domain. It is very difficult to compare our results with the previous results because the experiment conditions are completely different.

References

Kenneth W. Church. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text, In *Proceedings of ANLP-88*, pages 136-143.

Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A Practical Part-of-Speech Tagger, In *Proceedings of ANLP-92*, pages 133-140.

Terumasa Ehara, Kentaro Ogura, Tsuyoshi Morimoto. 1990. ATR Dialogue Database. In *Proceedings of ICSLP*, pages 1093-1096.

William A. Gale and Kenneth W. Church. 1990. Poor Estimates of Context are Worse than None. In *Proceedings of DARPA Natural Language and Speech Workshop*, pages 283-287.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A Spelling Correction Program Based on a Noisy Channel Model. In *Proceedings of COLING-90*, pages 205-210.

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, Vol.24, No.4, pages 377-439.

Nobuyasu Ito and Hiroshi Maruyama. 1992. A Method of Detecting and Correcting Errors in the Results of Japanese OCR. In *Transaction of Information Processing Society of Japan*, Vol.33, No.5, pages 664-670 (In Japanese).

Frederick Jelinek. 1985. Self-organized Language Modeling for Speech Recognition. IBM Report.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context Based Spelling Correction. *Information Processing & Management*, Vol. 27, No.5, pages 517-522.

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.

Masaaki Nagata. 1996. Automatic Extraction of New Words from Japanese Texts using Generalized Forward-Backward Search. To appear in *Proceedings of EMNLP*.

Frank K. Soong and Eng-Fong Huang. 1991. A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition. In *Proceedings of ICASSP-91*, pages705-708.

Tetsuyasu Takao and Fumihito Nishino. 1989. Implementation and Evaluation of Post-processing for Japanese Document Readers. In *Transaction of Information Processing Society of Japan*, Vol.30, No.11, pages 1394-1401 (In Japanese).

Robert A. Wagner and Michael J. Fischer. 1974. The String-to-String Correction Problem. In *Journal of the ACM*, Vol.21, No.1, pages 168-173.