A Re-estimation Method for Stochastic Language Modeling from Ambiguous Observations

Mikio Yamamoto
Institute of Information Sciences and Electronics
University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305, Japan
myama@is.tsukuba.ac.jp

Abstract

This paper describes a reestimation method for stochastic language models such as the N-gram model and the Hidden Markov Model(HMM) from ambiguous observations. It is applied to model estimation for a tagger from an untagged corpus. We make extensions to a previous algorithm that reestimates the N-gram model from an untagged segmented language (e.g., English) text as training data. The new method can estimate not only the N-gram model, but also the HMM from untagged, unsegmented language (e.g., Japanese) text. Credit factors for training data to improve the reliability of the estimated models are also introduced. In experiments, the extended algorithm could estimate the HMM as well as the N-gram model from an untagged, unsegmented Japanese corpus and the credit factor was effective in improving model accuracy. The use of credit factors is a useful approach to estimating a reliable stochastic language model from untagged corpora which are noisy by nature.

1 Introduction

Stochastic language models are useful for many language processing applications such as speech recognition, natural language processing and so on. However, in order to build an accurate stochastic language model, large amounts of tagged text are needed and a tagged corpus may not always match a target application because of, for example, differences between the tag systems. If the language model can be estimated from untagged corpora and the dictionary of a target application, then the above two problems would be resolved because large amounts of untagged corpora could be easily used and untagged corpora are neutral toward any applications.

Kupiec (1992) has proposed an estimation method for the N-gram language model using the Baum-Welch reestimation algorithm (Rabiner et al., 1994) from an untagged corpus and Cutting et al. (1992) have applied this method to an English tagging system. Takeuchi and Matsumoto (1995) also have developed an extended method for unsegmented languages (e.g., Japanese) and applied it to their Japanese tagger.

However, Merialdo (1994) and Elworthy (1994) have criticized methods of estimation from an untagged corpus based on the maximum likelihood principle. They pointed out limitation of such methods revealed by their experiments and said that the optimization of likelihood didn't necessarily improve tagging accuracy. In other words, the training data extracted from an untagged corpus using only a dictionary are, by nature, too noisy to build a reliable model.

I would like to know whether or not the noise problem occurs in other language models such as the HMM. Zhou and Nakagawa (1994) have shown, in the experiments of word prediction from the previous word sequence, that the HMM is more powerful than the bigram model and is nearly equivalent to the trigram model, though the number of parameters of the HMM is less than that in the N-gram model. In general, models with fewer parameters are more robust. Here, I investigate a method that can estimate HMM parameters from an untagged corpus and also a general technique for supressing noise in untagged training data. The goals of this paper are as follows.

- *Extension of Baum-Welch algorithm:* I formulate an algorithm that can be applied to untagged, unsegmented language corpora and estimate not only the N-gram model, but the HMM. Also, a scaling procedure is defined in the algorithm.
- *Credit factor:* In order to overcome the noise of untagged corpora, I introduce credit factors that are assigned to training data. The estimation algorithm can approximately maximize the modified likelihood that is weighted by the credit factors.

The problem of stochastic tagging is formulated in the next section(2) and the extended reestimation method in section 3. A way of determining the credit factor based on a rule-based tagger is described in section 4. Experiments which evaluate the proposed method are reported in section 5.

2 Stochastic Tagging Formulation

In general, the stochastic tagging problem can be formulated as a search problem in the stochastic space of sequences of tags and words. In this formulation, the tagger searches for the best sequence that maximizes the probability (Nagata, 1994):

$$(\hat{W},\hat{T}) = \arg\max_{W,T} p(W,T|S) = \arg\max_{W,T} p(W,T) \tag{1}$$

where W is a word sequence $(w_1, w_2, ..., w_n)$, T is a tag sequence $(t_1, t_2, ..., t_n)$ and S is an input sentence. Since Japanese sentences have no delimiters (e.g., spaces) between words, a morphological analyzer (tagger) must decide word segmentation in addition to part-of-speech assignment. The number of segmentation ambiguities of Japanese sentences is large and these ambiguities complicate the work of a Japanese tagger.

Although all possible $p(W,T)$s on combinations of W and T cannot be estimated, there are some particularly useful approximations such as the N-gram model and the HMM. The following formulae are straightforward formulations whose observed variables are pairs of words and tags:

$$p(W,T) \cong \prod_{i=1}^{n} p(w_i, t_i | w_{i-N+1}, ..., w_{i-1}, t_{i-N+1}, ..., t_{i-1}) \tag{2}$$

$$p(W,T) \cong \sum_{x} \prod_{i=0}^{n-1} a_{x(i),x(i+1)} b_{x(i+1)}(w_{i+1}, t_{i+1}) \tag{3}$$

Formula 2 is the N-gram model and formula 3 is the HMM. When N of formula 2 is two, the model is called the bigram, when N is three, it is the trigram. Symbol x of formula 3 denotes a possible path of states of the HMM and $x(i)$ denotes a state of the HMM that is visited at the i-th transition in the path x. $a_{x(i),x(i+1)}$ is the transition probability from $x(i)$ to $x(i+1)$. In particular, $a_{x(0),x(1)}$ represents the initial state probability ($\pi_{x(1)}$) of $x(1)$. $b_{x(i)}(w,t)$ is an output probability of a pair of word w and tag t on the state $x(i)$. A state of the HMM represents an abstract class of a part of the input symbol sequence. That is, we can regard the HMM as a mixed model of unigram, bigram, trigram, and so on.

We can also decrease the number of model parameters by separating the tag model from formulae 2 and 3. In the models, the N-gram and the HMM are used to model tag sequence and $p(w|t)$ is used for another part of the model.

$$p(W,T) \cong \prod_{i=1}^{n} p(t_i|t_{i-N}...t_{i-1})p(w_i|t_i) \tag{4}$$

$$p(W,T) \cong \sum_{x} \prod_{i=0}^{n-1} a_{x(i),x(i+1)} b'_{x(i+1)}(t_{i+1}) p(w_{i+1}|t_{i+1}) \tag{5}$$

The PAIR-HMM, TAG-bigram model, and TAG-HMM based on formulae 3, 4 (where $N = 2$) and 5, respectively, will be investigated in section 5. In the next section, I describe an extension to the forward-backward algorithm for determining HMM parameters from ambiguous observations.

3 Re-estimation Method from Ambiguous Observations

3.1 Ambiguous Observation Structure

Here, we define an ambiguous observation as a lattice structure with a credit factor for each branch. In unsegmented languages that have no delimiter between words, such as Japanese, candidates for alignment of tag and word have different segmentation. That is, they must be represented by a lattice. We can create a lattice structure from untagged Japanese sentences and a Japanese dictionary.

The following is the definition of the lattice of candidates representing ambiguous word and tag sequences called the morpheme network. All morphemes on the morpheme network are numbered.

w_s or $word(s)$: The spelling of the s-th morpheme.
t_s or $tag(s)$: The tag of the s-th morpheme.
$suc(s)$: The set of morpheme numbers that the s-th morpheme connects to.
$pre(s)$: The set of morpheme numbers that connect to the s-th morpheme.
$credit(r, s)$: The credit factor of the connection between the r-th and the s-th morphemes.

For example, a morpheme network can be derieved from the input sentence "船出しない" which means "not to sail" (Fig. 1). The real and dotted lines in Figure 1 represent the correct and incorrect paths of morphemes, respectively. Of course, any algorithm for estimation from untagged corpora cannot determine whether the connections are correct or not. The connections of dotted lines constitute noise for the estimation algorithm. The numbers on the lines show the credit factor of each connection that is assigned by the method described in section 4. The numbers at the right of colons are morpheme numbers. In Figure 1, $word(3)$ is '出し', $tag(3)$ is 'verb', $pre(3)$ is the set $\{1\}$, $suc(3)$ is the set $\{6, 7\}$ and the credit factor $credit(1, 3)$ is 0.8.

3.2 Re-estimation Algorithm

Given a morpheme network, we can formulate the reestimation algorithm for the HMM parameters. The original forward-backward algorithm calculates the probability of the partial observation sequence given the state of the HMM at the time (position of word in the input sentence). The original algorithm does this by a time synchronous procedure operating on unambiguous observation sequence. The extended algorithm calculates the probability of the

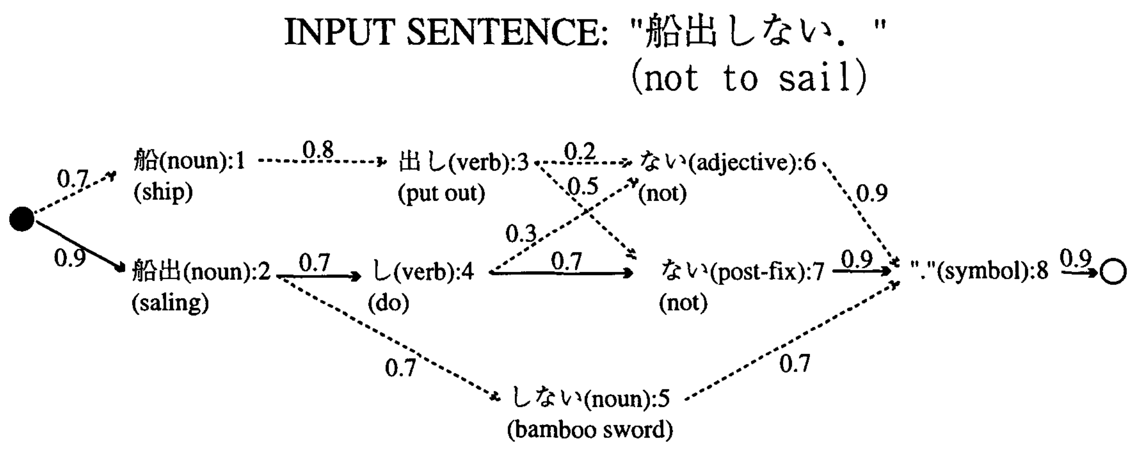

Figure 1: An example of the morpheme network.

partial ambiguous sequence given the state of the HMM at the node (morpheme candidate) in the morpheme network by a node synchronous procedure. The algorithm formulation is as follows:

initial:

$$\alpha_u(j) = \pi_j b_j(w_u, t_u) credit(\#, u) \qquad where \quad u \in on(1)$$
$$\beta_v(i) = 1 \qquad where \quad v \in on(B)$$

recursion:

$$\alpha_r(j) = \sum_{s \in pre(r)} \sum_{i=1}^{N} \alpha_s(i) a_{ij} b_j(w_r, t_r) credit(s, r)$$
$$\beta_s(i) = \sum_{r \in suc(s)} \sum_{j=1}^{N} a_{ij} b_j(w_r, t_r) \beta_r(j) credit(s, r)$$

where $on(1)$ is the set of numbers of the left most morphemes in the morpheme network and $on(B)$ is the set of numbers of the right most morphemes. The '#' in $credit(\#, u)$ means the beginning-of-text indicator.

The trellis, that is often used to explain the original forward-backward algorithm, is extended into a network trellis. Figure 2 is an example of the network trellis that is generated from the morpheme network example given above (Fig. 1). In this example, $\alpha_7(1)$ means a forward probability of the 7th morpheme at the 1st state of the HMM.

Using the extended forward-backward probabilities we can formulate the reestimation algorithm from ambiguous observations:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{r \in pre(s)} \alpha_r^k(i) a_{ij} b_j(t_s) \beta_s^k(j) credit(r, s)}{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{s} \alpha_s^k(i) \beta_s^k(i)} \tag{6}$$

$$\bar{b}_i(w, t) = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{word(s)=w, tag(s)=t} \alpha_s^k(i) \beta_s^k(i)}{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{s} \alpha_s^k(i) \beta_s^k(i)} \tag{7}$$

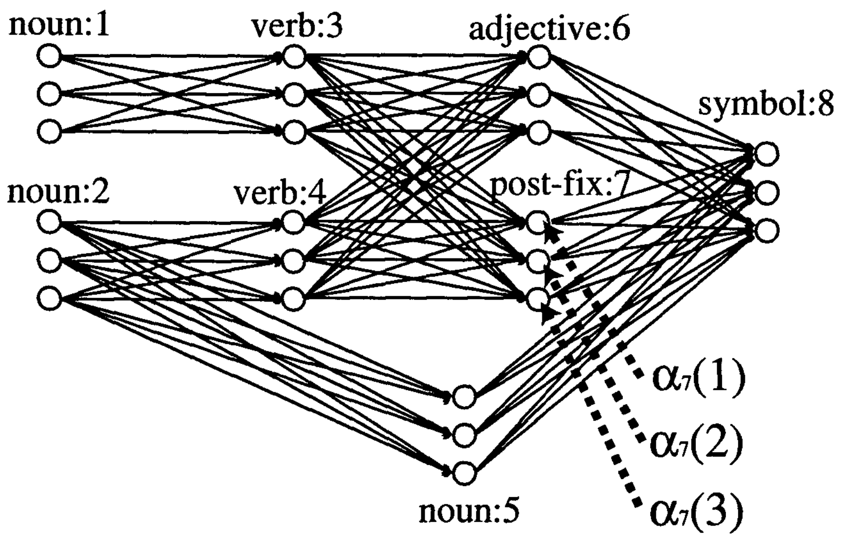

Figure 2: An example of the network trellis

$$\bar{\pi}_i = \frac{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{s \in on(1)} \alpha_s^k(i)\beta_s^k(i)}{\sum_{k=1}^{K} \frac{1}{p_k} \sum_{s \in on(1)} \sum_{j=1}^{N} \alpha_s^k(j)\beta_s^k(j)} \tag{8}$$

where k represents the k-th input sentence and p_k is sum of the probabilities of possible sequences in the k-th morpheme network weighted by the credit factors.

3.3 Scaling

In the calculation of forward-backward probabilities, under-flow sometimes occurs if the dictionary for making the morpheme network is large and/or the length of the input sentence is long, because the forward-backward algorithm multiplies many small transition and output probabilities together. This problem is native to speech modeling, but in general, the modeling of text is free from this problem. However, since Japanese sentences tend to be relatively long and the recent Japanese dictionary for research is large, under-flow is sometimes a problem. For example, the EDR Japanese corpus (EDR, 1994) includes sentences that consist of more than fifty words at a frequency of one percent. In fact, we experienced the underflow problem in preliminary experiments with the EDR corpus.

Application of the scaling technique of the original backward-forward algorithm (Rabiner et al., 1994) to our reestimation method would solve the under-flow problem. The original technique is based on synchronous calculation with positions of words in the input sentence in left-to-right fashion. However, since word boundaries in the morpheme network may or may not cross on the input character sequence, we cannot directly apply this method to the extended algorithm.

Let us introduce synchronous points on an input characters sequence to facilitate synchronization of the calculation of forward-backward probabilities. All possible paths of a morpheme

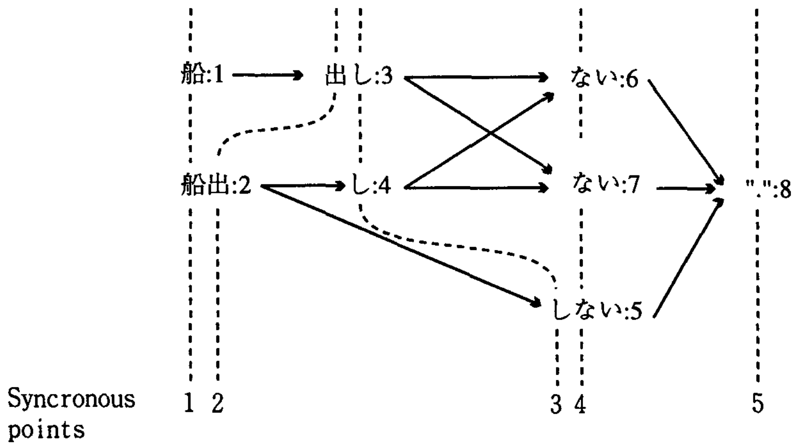

Figure 3: An example of syncronous points

network have one morpheme on each synchronous point. The synchronous points are defined as positions of the head character of all morphemes in a morpheme network and are numbered from left to right. The synchronous point number of the left most word is defined as 1. A morpheme is associated with the synchronous points which are located in the flow of characters of the morpheme.

The symbols and $on(q)$ function are defined as follows:

B: The maximum number of synchronous points in a morpheme network.

$on(q)$: The set of morpheme numbers that are associated with synchronous point q.

L_s: The left most synchronous point that is associated with the s-th morpheme.

R_s: The right most synchronous point that is associated with the s-th morpheme.

Figure 3 is an example of the syncronous points for the morpheme network example given above (Fig. 1). The values of the symbols and function defined above are as follows in this example; $B = 5$, $on(2) = \{2, 3\}$, $L_5 = 3$, $R_5 = 4$ and so on.

The scaled forward probabilities are defined with the above definitions. The notation $\alpha_{sl}(i)$ is used to denote the unscaled forward probabilities of the s-th morpheme on the syncronous point l, $\hat{\alpha}_{sl}(i)$ to denote the scaled forward probabilities, and $\hat{\hat{\alpha}}_{sl}(i)$ to denote the local version of α before scaling. c_l is the scaling factor of synchronous point l.

initial:

$$\hat{\hat{\alpha}}_{s1}(i) = \alpha_{s1}(i) = \pi_i b_i(w_s, t_s) credit(\#, s) \qquad where \quad s \in on(1)$$

$$c_1 = 1 / \sum_{s \in on(1)} \sum_{i=1}^{N} \hat{\hat{\alpha}}_{s1}(i)$$

$$\hat{\alpha}_{s1}(i) = c_1 \hat{\hat{\alpha}}_{s1}(i) \qquad where \quad s \in on(1)$$

cost	0	1-10	11-20	21-50	51-100	101-200	201-500	501-1000
precision	0.84	0.16	0.13	0.069	0.074	0.0083	0.0017	0

Table 1: The precision on each cost of Juman

recursion:

$$\hat{\hat{\alpha}}_{sl}(j) = \begin{cases} \hat{\alpha}_{s,l-1}(i) & if \;\; L_s \neq l \\ \sum_{r \in pre(s)} \sum_{i=1}^{N} \hat{\alpha}_{r,l-1}(i) a_{ij} b_{tj}(w_s, t_s) credit(r,s) & if \;\; L_s = l \end{cases}$$

$$c_l = 1 \sum_{s \in on(l)} \sum_{i=1}^{N} \hat{\hat{\alpha}}_{sl}(i)$$

$$\hat{\alpha}_{sl}(i) = c_l \hat{\hat{\alpha}}_{sl}(i)$$

The scaled forward probabilities can be calculated synchronizing with the synchronous points from left to right. The scaled backward probabilities are defined in the same way using the scaling factors obtained in the calculation of the forward probabilities.

The scaled forward-backward probabilities have the following property:

$$\hat{\alpha}_s(i) a_{ij} b_j(w_r, t_r) \hat{\beta}_r(j) = \frac{1}{p_k} \alpha_s(i) a_{ij} b_j(w_r, t_r) \beta_r(j) \tag{9}$$

where $\hat{\alpha}_s = \hat{\alpha}_{sR_s}$ and $\hat{\beta}_s = \hat{\beta}_{sL_s}$. Using this property, the reestimation formulae can be replaced with the scaled versions. The replaced formulae are free of the under-flow problem and their use also obviates the need to calculate the weighted sum of path probabilities of the k-th ambiguous observation, p_k.

4 Credit Factor

In the estimation of a Japanese language model from an untagged corpus, the segmentation ambiguity of Japanese sentences severely degrades the model reliability. I will show that model estimations excluding the credit factors cannot overcome the noise problem in section 5. Credit factors play a very important role by supressing noise in the training data. However, a way of calculating the optimal value of credit is not yet available, so a preliminary method described in this section was used for the experiments.

The 'costs' of candidates outputted by a rule-based tagger were used as the source of information related to the credit. Juman (Matsumoto et al., 1994) was used in our experiments to generate the morpheme network. Juman is a rule-based Japanese tagging system which uses hand-coding cost values that represent the implausibility of morpheme connections, and word- and tag-occurences. Given a cost-width, Juman outputs the candidates of morpheme sequences pruned by this cost-width. A larger cost-width would result in a larger number of output candidates.

We evaluated the precision of a set of morpheme candidates that have a certain cost. The precision value was used as the credit factor of each branch in the morpheme network to be outputted by Juman (Table 1). In the experiments described in the next section, we approximated the results from this example (see Table 1) by the formula $1/(a * cost + b)$, where a was 0.5 and b 1.19.

5 Experiments

5.1 Implementation

The experimental system for model estimation was implemented using the extended reestimation method. A morpheme network of each input sentence was generated with Juman (Matsumoto et al., 1994) and the credit factor was attached to each branch as described above. The system can estimate three kinds of models; the PAIR-HMM (formula 3) with output symbols as pairs of words and tags, the TAG-bigram model (formula 4, where $N = 2$) and TAG-HMM (formula 5) with output symbols as tags and $p(w|t)$. The scaling technique was used with all estimations.

The numbers of parameters of the TAG-bigram model, the TAG-HMM and the PAIR-HMM are approximated by the equations $NT^2 + ND$, $NS^2 + NS * NT + ND$, and $NS^2 + NS * ND$, respectively, where NT is the number of tags, NS is the number of states of the HMM, and ND is the number of entries in the dictionary. In all experiments, NT, NS and ND were fixed at 104, 10, and 130,000, respectively. The numbers of parameters of the TAG-bigram model, TAG-HMM, PAIR-HMM were $10816 + ND$, $1140 + ND$, and $100 + 10ND$, respectively. Note that the number of parameters of the *tag* model of the TAG-HMM is one tenth that of the TAG-bigram model.

For the model evaluation, a stochastic tagger was implemented. Given a morpheme network generated by Juman with a cost-width, the implemented tagger selects the most probable path in the network using each stochastic model. The best path was calculated by the Viterbi-algorithm on the paths of the morpheme network.

5.2 Data and Evaluation

I used 26108 Japanese untagged sentences as training data and 100 hand-tagged sentences as test data, both from the Nikkei newspaper 1994 corpus (Nihon Keizai Shimbun, Inc., 1995). The test sentences include about 2500 Japanese morphemes. The tags were defined as the combination of part-of-speech, conjugation, and class of conjugation. The number of kinds of tags was 104.

In the precision evaluation, the correct morpheme was defined as that matching the segmentation, tag, and spelling of the base form of the hand-tagged morpheme. The precision was defined as the proportion of correct morphemes relative to the total number of morphemes in the sequence which the tagger outputted as the best alignment of tags and words.

5.3 Results

Three kinds of models were estimated using the untagged training data with the initial parameters set to the equivalent probabilities. Each model was estimated both with and without use of the credit factor. The reestimation algorithm was iterated for five to twenty times.

The precision of the most plausible segmentation and tag assignment was outputted by the tagger based on each stochastic model estimated either without (Figs. 4 and 5) or with (Fig. 6) the credit factor assignment function described in the previous section. Two versions of the morpheme network for the estimations were used; one limited by a cost-width of 500 (Fig. 4) and the other by a cost-width of 70 (Figs. 5 and 6). The cost-width of 500 required almost all of the morphemes to be used for the estimation. In other words, a morpheme network of cost-width 500 was equivalent to that extracted from the input sentence with a dictionary only. Although one experiment (Fig. 5) didn't use the credit factor assignment function, it is regarded as using a special function of the credit factor that returns 0 or 1, that

cost-width	0	10	20	50	100	200	500	1000
precision	84.4	79.8	79.3	71.0	66.3	43.6	36.5	36.3
recall	94.7	96.0	96.1	97.2	98.0	98.7	98.7	98.7

Table 2: The precision and recall of Juman on each cost-width.

is a step function, with a cost threshold of 70. However, this function doesn't differentiate among morphemes whose costs are 0 and 70.

The cost-widths (see horizontal axes in Figs. 4, 5 and 6) were provided to Juman to generate the morpheme network used in the stochastic tagger for model evaluation. The tagger chose the best morpheme sequence from the network by each stochastic model. A larger cost-width would result in a larger network, lower precision, and higher recall (Table 2). Note that the precision of any model will never exceed the recall of Juman (see Table 2). If a model is correctly estimated, then a larger cost-width will improve precision. Therefore, we can estimate model accuracy from the precision at cost-width 500 or 1000.

When estimated without the credit factor (Fig. 4), neither the HMM nor the TAG-bigram model was robust against noisy training data. It was also observed in the experiments that the accuracy of tagging was degraded by excessive iterations of reestimation. I conclude that it is hard to estimate the Japanese model from only an untagged corpus and a dictionary.

Precision was improved by the step credit factor function whose threshold is 70 (Fig. 5). The precision of the HMMs are better than the precision of the TAG-bigram model, despite the number of parameters of the TAG-HMM being smaller than that for the TAG-bigram model. The HMM is very capable of modeling language, if the training data is reliable.

Including the variable credit factor in these models is an effective way to improve precision (Fig. 6). In particular, the results of the TAG-bigram model were dramatically improved by using the variable credit factor. Although incorporating the credit factor into the HMM improved the results, they remained at a level similar to that of the TAG-bigram model with the credit factor. Although it is not clear exactly why the HMM did not improved more, there are at least three possible explanations: (1) theoretical limitation of estimation using an untagged corpus, (2) using an untagged corpus, estimation of the HMM is harder than estimation of the bigram model, therefore more corpora are needed to train the HMM or (3) the credit factor in this experiment matched to the bigram model but not to the HMM. Investigation of these possibilities in the future is needed.

6 Discussion and Future Work

Merialdo (1994) and Elworthy (1994) have insisted, based on their experimental results, that the maximum likelihood training using an untagged corpus does not necessarily improve tagging accuracy. However, their likelihood was the probability with all paths weighted equivalently. Since more than half of the symbols in the observations may be noise, models estimated in this way are not reliable. The credit factor was introduced to redefine the likelihood of training data. The new likelihood was based on the probability with each possible path weighted by the credit factor. The extended reestimation algorithm can approximately maximize the modified likelihood and improve the model accuracy.

The Baum-Welch reestimation algorithm was also extended in two ways. The algorithm can be applied to an unsegmented language (e.g., Japanese), because of the extension for coping with lattice-based observations as training data. The other extension is that the algorithm can

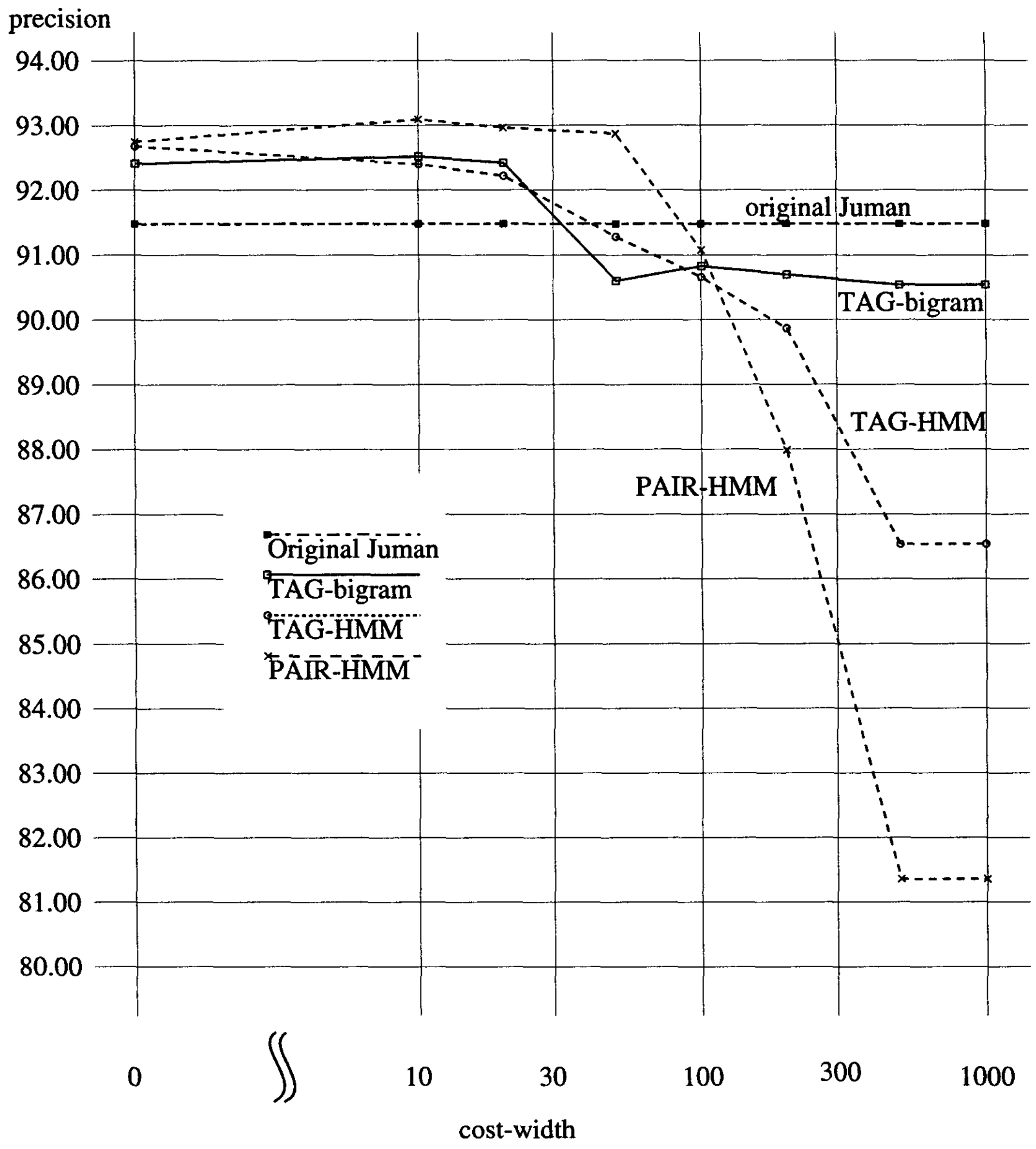

Figure 4: The precision of the models estimated without the credit factor

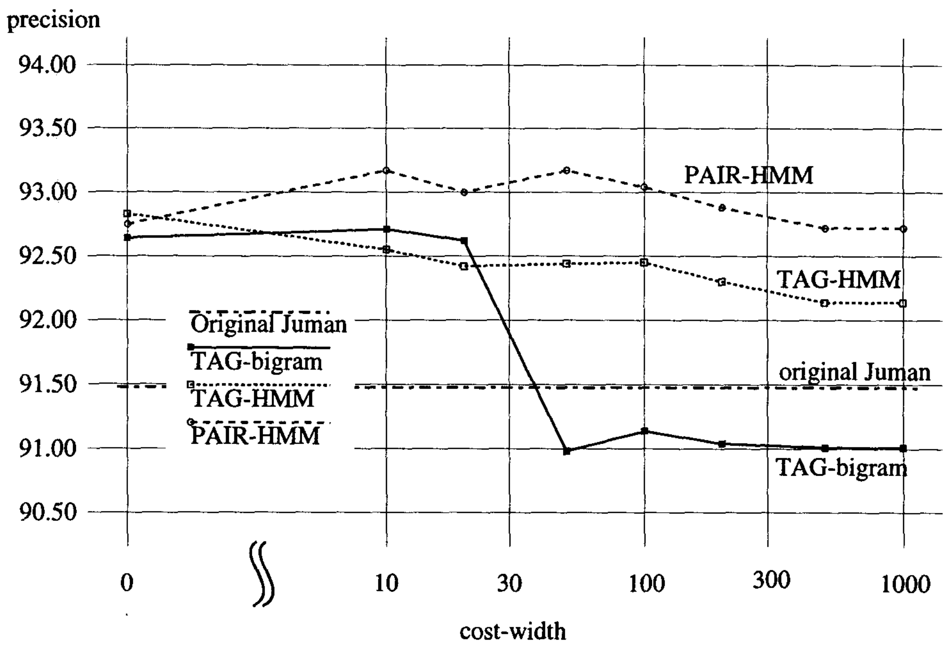

Figure 5: The precision of the models estimated with the step credit factor.

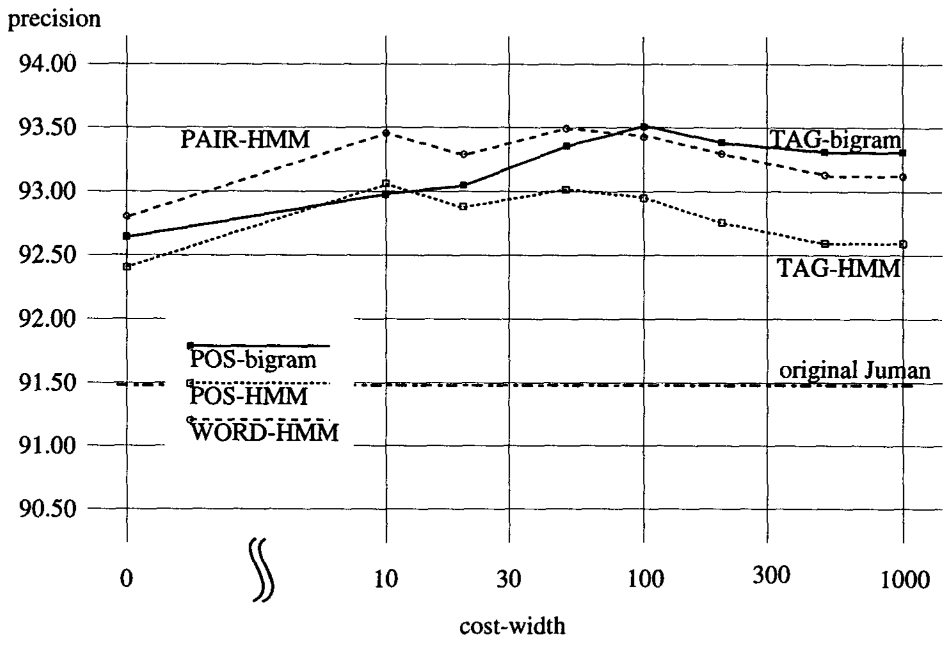

Figure 6: The precision of the models estimated with the variable credit factor.

train the HMM in addition to the N-gram model. Takeuchi and Matsumoto (1995) proposed the bigram estimation method from an untagged Japanese corpus. Their algorithm divides a morpheme network into possible sequences that are then used for the normal Baum-Welch algorithm. This algorithm cannot take advantage of the scaling procedure, because it requires the synchronous calculation of all possible sequences in the morpheme network. Nagata (1996) recently proposed a generalized forward-backward algorithm that is a character synchronous method for unsegmented languages. He applied this algorithm to bigram model training from untagged Japanese text for new word extraction. However, he did not apply this algorithm to the estimation of HMM parameters.

Two additional experiments have been planned. One is related to the limitations of estimation using untagged corpora. The other is related to assignment of the credit factor without a rule-based tagger.

The credit factor improved the upper bound of the estimation accuracy from an untagged corpus. However, at higher levels of tagging accuracy, the reestimation method based on the Baum-Welch algorithm is limited by the noise of untagged corpora. On this point, I agree with Merialdo (1994) and Elworthy (1994). One promising direction for future work would be an integration of models estimated from tagged and untagged corpora. Although the total model estimated from an untagged corpus is worse than that from a model using a tagged corpus, a part of the model using the untagged corpus may be better, because estimations from untagged corpora can use very extensive training material. In the bigram model, we can weight each probability of a pair of tags in both models estimated from tagged or untagged corpora. A smoothing method, such as deleted interpolation (Jelinek, 1985), can be used for weighting.

Another promising avenue for research is the development of improved methods to assign the credit factor without using rule-based taggers. Any chosen rule-based tagger will impart its own characteristic errors to credit factors it has been used to assign. Such errors can be misleading in the modeling of language. In order to assign more neutral values to the credit factor, we can use the estimated model itself. In the initial estimation of a model, an equivalent credit factor is used for estimation. After several iterations of reestimation, development data tagged by hand is used to evaluate the estimated model. The credit factors can be assigned from this evaluation process and be used in the second phase of estimation. Following the second phase of estimation, new credit factors would be decided by evaluation of the new model. Such a global iteration is a special version of error correcting learning.

7 Conclusion

We have proposed an estimation method from ambiguous observations and a credit factor. This estimation method can use untagged, unsegmented language corpora as training data and build not only the N-gram model, but also the HMM. A credit factor can improve the reliability of the model estimated from an untagged corpus.

This method can be further improved and integrated with other language models. In particular, it is important to formulate a dynamic method to assign the credit factor based on small sets of tagged data for development.

Acknowledgement

Thanks are due to the members of both the Itahashi laboratory at the University of Tsukuba and the Nakagawa laboratory at the Toyohashi University of Technology for their help and criticism at various stages of this research.

References

Cutting, D., J. Kupiec, J. Pedersen and P. Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Second Conference on Applied Natural Language Processing*, pages 133–140. Association for Computational Linguistics, Morristown, New Jersey.

Elworthy, David. 1994. Does Baum-Welch re-estimation help taggers? In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 53–58. Association for Computational Linguistics, Morristown, New Jersey.

Japan Electronic Dictionary Research Institute. 1995. *EDR Electronic Dictionary Version 2 Technical Guide.* http://www.iijnet.or.jp/edr.

Jelinek, Frederick. 1985. Self-organized language modeling for speech recognition. IBM Report. (Reprinted in *Readings in Speech Recognition*, pages 450–506, Morgan Kaufmann).

Kupiec, Julian. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, 6, pages 225–242.

Matsumoto, Y., S. Kurohashi, T. Utsuro, Y. Nyoki and M. Nagao 1994. *Japanese morphological analysis system JUMAN manual* (in Japanese).

Merialdo, Bernard. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2), pages 155–171.

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

Nihon Keizai Shimbun, Inc. 1995. Nikkei newspaper database 1994, CD-ROM version.

Rabiner, Lawrence and Biing-Hwang Juang. 1994. *Fundamentals of Speech Recognition.* PTR Prentice-Hall, Inc.

Takeuchi, Kouichi 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).

Zhou, Min and Seiichi Nakagawa. 1994. A study of stochastic language models for Japanese and English. In *Proceedings of Symposium on Learning in Natural Language Processing*, pages 57–64 (in Japanese).