REDUCING MEMORY REQUIREMENTS AND COMPUTATIONAL COSTS FOR THE BAUM-WELCH ALGORITHM AND APPLICATION TO AUTOMATIC STOCHASTIC NETWORK GRAMMAR ACQUISITION.

Jin'ichi Murakami

ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
murakami@itl.atr.co.jp

ABSTRACT

This paper describes new techniques for language modeling in speech recognition based on the use of a discrete density Ergodic Hidden Markov Model (HMM).

A discrete-output Ergodic HMM has a structure similar to that of a stochastic network language model (SNLM), so it can automatically function as an SNLM from a large amount of text data through the Baum-Welch algorithm. However, when the number of states in this Ergodic HMM is large, a large amount of memory is required and the computational cost is high. Therefore, past studies have limited the number of states. Consequently, the resulting perplexity of the Ergodic HMM has been high, and results as good as those obtained for word bigram models have not been obtained.

This paper proposes new techniques to reduce the memory requirements and computational costs associated with the Baum-Welch algorithm. These techniques were evaluated for their ability to automatically give an SNLM for an international conference registration task. Based on both the perplexity obtained and the results of continuous speech recognition, this Ergodic HMM was found to outperform word bigram models or trigram models. This implies that the proposed techniques are effective.

1. INTRODUCTION

There are many ways of modeling natural language for speech recognition. Among these models, a network language model is often used because of its simplicity. However, this model normally has a high perplexity which decreases the speech recognition performance. Therefore, stochastic network language model (SNLMs) which can add probability to the network language model have been studied to resolve this problem.

On the other hand, Hidden Markov Models (HMMs) are popular for acoustic modeling in speech recognition[1]. One of the advantages of HMMs is that they can be automatically trained through the Baum-Welch maximum likelihood estimation procedure using training speech data. Among the various types of HMMs, the all-state-connected model is called an Ergodic HMM.

This Ergodic HMM has a structure similar to that of an SNLM [2]. Therefore, it can function as an SNLM automatically through the Baum-Welch algorithm. The resulting state transition probabilities of the Ergodic HMM are interpreted as transition probabilities in the SNLM and output probabilities of the HMM are interpreted as word output probabilities in the SNLM.

However, when the number of states in the Ergodic HMM is large, a large amount of memory is required and the computational cost is high. Therefore, past studies have limited the number of states. Because of this, the perplexity has been high and results as good as those for word bigram models have not been obtained [3],[4].

This paper proposes new techniques which reduce the memory requirements and computational costs significantly when the Baum-Welch algorithm is used. These techniques were evaluated for their ability to automatically give an SNLM for an international conference registration task. The Ergodic HMM was found to have a low perplexity compared to that of word bigram models. Furthermore, continuous speech recognition was performed. The results showed that the Ergodic HMM can outperform word bigram models for text-closed data and can outperform word trigram models for text-open data. These results indicate that the proposed techniques are efficient.

2. AUTOMATIC STOCHASTIC NETWORK LANGUAGE MODEL ACQUISITION USING AN ERGODIC HMM

A Mealy-type Ergodic discrete-density HMM is considered to be a stochastic network language model. This HMM has parameters $\lambda = (\Pi, A, B)$.

$\Pi = \pi_N(i); \quad i = 1, .., N$, is the initial state probability distribution.

$A = a_N(i, j); \quad i = 1, .., N; j = 1, .., N$, is the state transition probability distribution between state i and state j.

$B = b_N(i, j, w); \quad i = 1, .., N; j = 1, .., N; w = 1, .., V$, is the word w's output word probability belonging somewhere between state i and state j.

In these equations, N is the number of states and V is the vocabulary size.

2.1. Reducing memory requirements and computational costs

If a large-state Ergodic HMM is to be calculated, a large amount of memory is required. Therefore, we propose new techniques which reduce the memory requirements and computational costs significantly when the Baum-Welch algorithm is to be used. These techniques are divided into two main parts.

Technique 1 Set small probabilities to 0.

If $b_N(i,j,w)$ is less than a given small value, this probability is set to 0.0 and there are no calculation nor memory needs for the Baum-Welch algorithm.

To put it concretely, the forward value $\alpha_t(j)$ must be calculated as follows[1] with the basic Baum-Welch algorithm.

$$\alpha_t(j) = [\sum_i \alpha_{t-1}(i) a_N(i,j)] b_N(i,j,O_t)$$

Instead of this equation, the following equations are used:

$$\alpha_t(j) = [\sum_i \alpha_{t-1}(i) a_N(i,j)] b_N(i,j,O_t) \qquad b_N(i,j,Ot) > C$$

$$\alpha_t(j) = 0.0 \qquad b_N(i,j,Ot) \leq C$$

In experiments, we set C to 10.0^{-300}.

The output probability $b_N(i,j,w)$ has the largest memory requirements among the parameters in the Ergodic HMM. So, in using this algorithm, the memory requirements are decreased for word output probabilities. The computational costs are also decreased at the same ratio.

Technique 2 Increase the number of states successively.

To calculate a large-state Ergodic HMM, a large amount of memory is required. So, we start with a 1-state Ergodic HMM and increase the number of states successively.

If the parameters of the N-state Ergodic HMM have already been estimated, the initial parameters of a $2N$-state Ergodic HMM are constructed as follows;

$\pi_{2N}(i) = 0.5 \times \pi_N(i/2); \quad i = 1,..,2N.$
$a_{2N}(i,j) = 0.5 \times a_N(i/2, j/2); \quad i = 1,..,2N; j = 1,..,2N.$
The symbol / indicated that the integer has been rounded-up.

The output probability B is a little different, as random weights are used.

$b_{2N}(i,j,w) = b_N(i/2, j/2, w) \times random(i,j,w)$
$i = 1,..,2N; j = 1,..,2N; w = 1,..,V.$
b_{2N} is normalized as $\sum_w b_{2N}(i,j,w) = 1.0.$

The training of this large-state Ergodic HMM is carried out as follows.

1. First, a 1-state Ergodic HMM is constructed. In this model, the initial state probability and the state transition probability are set to 1.0 and the word output probability is set to a random value.

2. Second, Ergodic HMM parameters are re-estimated using the proposed Baum-Welch algorithm (Technique 1).

3. Third, the number of states in the Ergodic HMM is increased using Technique 2.

4. Fourth, the Ergodic HMM parameters are re-estimated using the proposed Baum-Welch algorithm (Technique 1).

These sequence are repeated.

A similar technique where small probabilities are set to 0.0 is described in [5]. As such, this study's focus is mainly to reduce the computational costs and not to reduce the memory requirements. Also, as before, the number of states is kept small and there are no outstanding results.

3. STOCHASTIC NETWORK GRAMMAR ACQUISITION USING THE ERGODIC HMM

3.1. Experimental conditions for the Ergodic HMM

To test these new techniques, experiments on stochastic network grammar acquisition were performed. For training data, the ATR Dialog Database was used. The learning conditions of these experiments are shown in Table 1 .

Table 1: Learning conditions for Ergodic HMM

type of Hidden Markov model	Mealy type
number of training sentences	8,475
total number of words	57,354
vocabulary	6,418 words
end of HMM learning	40th iteration

3.2. The number of word output probabilities

Figure 1 shows the number of word output probabilities versus the number of states for the Ergodic HMM. In this figure, the thin line is the result for the basic Baum-Welch algorithm and the thick line is the result for the improved Baum-Welch algorithm. The figure shows that the latter method greatly reduces the number of word output probabilities. This means that the memory requirements and computational costs have been greatly reduced.

3.3. Entropy for the Ergodic HMM

To understand how the Ergodic HMM proved adequate, we studied the entropy of the Ergodic HMM. Figure 2 shows these results. In this figure, the vertical axis represents entropy and the horizontal axis represents the number of states. The line of dots is the result for the basic Baum-Welch algorithm and the thick line is that for the improved Baum-Welch algorithm. For comparison purposes, the calculated entropy of word bigram and trigram models are included.

In this figure, the difference between the basic Baum-Welch algorithm and the improved Baum-Welch algorithm is small. This means that the proposed algorithm can perform training like the basic Baum-Welch algorithm with lower the memory requirements and computational costs. And as the number of

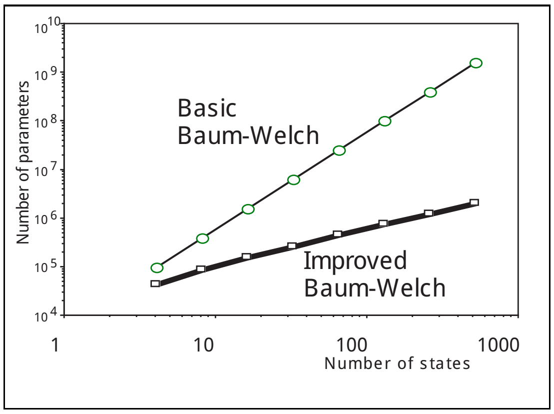

Figure 1: Number of word output probabilities versus number of states

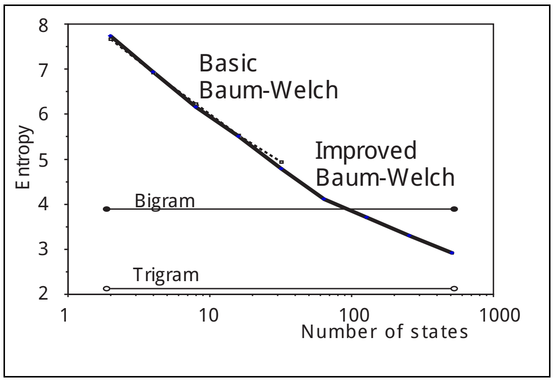

Figure 2: Entropy versus number of states

states increases, the entropy decreases. The entropy of the 128-state Ergodic HMM is lower than that of the word bigram models. However, the perplexity of the 512-state Ergodic HMM is higher than that of the word trigram models. It is possible though that if the number of states were increased, the entropy of the Ergodic HMM would have lowered that of the word trigram models.

4. EXPERIMENTS FOR CONTINUOUS SPEECH RECOGNITION

To show the effectiveness of the Ergodic HMM as a language model, speaker dependent continuous speech recognition experiments were carried out. In these experiments, a standard speech recognition algorithm employed a continuous mixture HMM and beam search, with the Ergodic HMM providing the language information[6].

4.1. Experimental conditions for continuous speech recognition

For the training of the continuous mixture HMM, 2,635 word utterances were used. For the training of the Ergodic HMM, 8,475 sentences of the ATR Dialog Database were used in text-open experiments, whereas the same 8,475 sentences plus 38 test sentences were used in text-closed experiments. In this way, it was possible to perform both the text-closed and text-open experiments using the same test data. The experimental conditions are summarized in Table 2

Table 2: Experimental conditions for continuous speech recognition using Ergodic HMM

algorithm	continuous mixture HMM + beam search + Ergodic HMM
mixture count	max 14 (valid for each syllable)
state number	3-state 4-loop left-to-right model
acoustic parameter	16th order LPC cepstrum + power + Δ power + 16th order Δ cepstrum
frame window	20 ms
frame period	5 ms
training voice	word speech (2,635 words)
phone category	52 syllables
vocabulary	435
beam width	4,096
duration control	no
test sentence count	261 sentences; same speaker
speaking style	read speech
speech content	international conference task

4.2. Experimental results for the text-closed data

The sentence recognition rate was evaluated for the text-closed experiments; the results are shown in Figure 3 . For comparison purposes, results for both the word bigram models and the word trigram models are also shown. In these experiments, the trigram and bigram values were smoothed by deleted-interpolation technique[8].

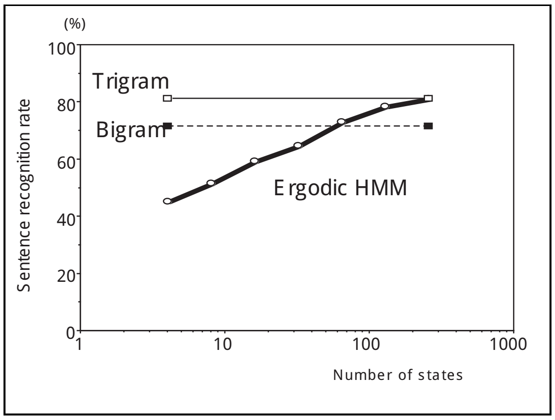

Figure 3: Recognition rate for text-closed data

In Figure 3 , the vertical axis represents the sentence recognition rate while the horizontal axis represents the number of states. It shows the recognition rate for the 128-state Ergodic HMM is to be higher than that for the word bigram models. It also shows the recognition rate for the 256-state Ergodic HMM to be equal to that for the word trigram models. It is possible though that if the number of states were increased, the recognition rate would have exceeded that for the word trigram models.

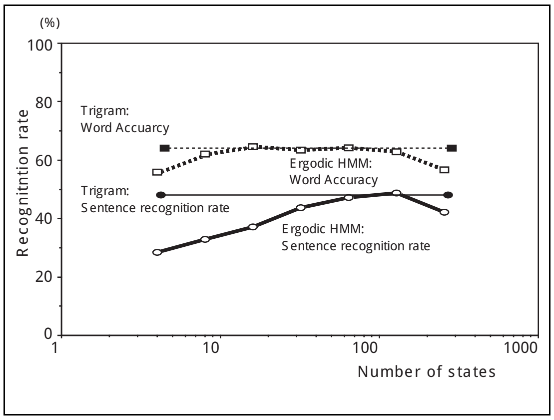

Figure 4: Recognition rate for text-open data

4.3. Experimental results for the text-open data

The results of the text-open continuous speech recognition experiments are shown in Figure 4 . For comparison purposes, results for the word trigram models are also shown. In these experiments, both the sentence recognition rate and word accuracy rate[7] were evaluated.

In the figure, a 48.6% sentence recognition rate is obtained for the 128-state Ergodic HMM. On the other hand, the recognition rate for the word trigram models is 47.1%. And the word accuracy rate of the 16-state Ergodic HMM which is equal to that of the word trigram models. These results show that the Ergodic HMM can outperform word trigram models for text-open data. On the other hand, the 256-state Ergodic HMM has lower performance compared to the word trigram models. This means that this model does overturning for text data.

5. DISCUSSION

5.1. Random Value

In this paper, word output probabilities are given random weights, when the number of states is increased (Technique 2). Otherwise, we use parts of speeches or categories using word dictionaries. For example, words belonging to the same category are given the same weight. This is a good way of trying to obtain better language models.

5.2. Second-order Ergodic HMM

The experiments studied a basic first-order Ergodic HMM. However, it would be very easy to extend the experiments to a second-order Ergodic HMM, which would probably produce a better performance. This will be examined in future research.

6. CONCLUSION

This paper discussed a large-state discrete Ergodic Hidden Markov Model as a stochastic network language model for language modeling. It proposed new techniques to reduce the memory requirements and computational costs associated with the Baum-Welch algorithm.

These techniques were evaluated using an international conference registration task by computing the perplexity. In addition, an Ergodic HMM was investigated for use in language modeling for continuous speech recognition. These results obtained compared favorably to those of word bigram models. This may mean that the proposed techniques are effective.

Acknowledgments

We would like to thank Dr. Yamazaki, President, and Dr. Sagisaka, Professor, Head of Department 1 , ATR Interpreting Telecommunications Research Laboratories, for their continuous support of this work. I am also grateful to Isotani of the members of Department 1 for the suggestion of the entropy of Ergodic HMM.

REFERENCES

[1] X.D.Huang, Y.Ariki, and M.A.Jack, “Hidden Markov Models for Speech Recognition,” Edinburgh University Press, p. 148, (1990).

[2] F. Jelinek, R. L. Mercer, and S. Roukos, “Principle of Lexical Language Modeling for Speech Recognition,” In S. Furui and M. M. Sondhi, editors, Advances in Speech Signal Processing, pp. 651-699, Marcel Dekker, Inc., New York, New York (1992).

[3] Y. Ueda, and S. Nakagawa, “Prediction for phoneme/syllable/word-category and identification of language using HMM”, ICSLP90, 27.6.1, pp.1209,1212 (1990).

[4] M.Woszczyna, and A.Waibel, “Inferring Linguistic Structure in Spoken Language” , ICSLP94, S16-18.1, pp.847,850 (1994).

[5] T. Kuhn, et al., “Ergodic Hidden Markov Models and Polygrams for Language Modeling,” ICASSP94, Vol. 1, pp. 357-360 (1994).

[6] J.Murakami, and S. Matsunaga, “ A Spontaneous Speech Recognition Algorithm Using Word Trigram Models and Filled-Pause Procedure”, ICSLP94, S16.11, pp.819-822 (1994).

[7] S. J. Young, et al., “HTK:Hidden Markov Model Toolkit V1.5 User Manual”, p.107 (Sep. 1993).

[8] F.Jelinek, “Self-Organized Language Modeling for Speech Recognition”, Readings in Speech Recognition, Morgan Kaufman Publishers, Inc. San Mateo, California pp. 450-506 (1990)