

NAIST-IS-DD0261013

Doctoral Dissertation

Automatic Model Generation for Speech Recognition

Takatoshi Jitsuhiro

March 24, 2005

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Takatoshi Jitsuhiro

Thesis Committee:

Professor Kiyohiro Shikano	(Supervisor)
Dr. Satoshi Nakamura	(Co-supervisor, ATR)
Professor Naokazu Yokoya	(Member)
Associate Professor Hiroshi Saruwatari	(Member)

Automatic Model Generation for Speech Recognition*

Takatoshi Jitsuhiro

Abstract

Recently, most techniques for speech recognition have been based on stochastic modeling, and parameters both of acoustic models and language models are estimated by using the large quantity of available training databases. The relation between the amount of training data and the number of parameters is very important: if the number of parameters is too small, it is difficult to obtain sufficient performance. On the other hand, if the number of parameters is too large, the obtained model is strongly dependent on the training data, and it cannot obtain enough performance. The latter problem is generally referred to as the over-fitting problem. Consequently, it is crucial to select an adequate size of models for training data.

These days, most speech recognizers employ phoneme context-dependent hidden Markov Models (HMMs). Shared-state structures of HMMs should be estimated from training data. To create structures, phonetic decision tree clustering is widely used and is based on the Maximum Likelihood (ML) criterion. However, when the number of parameters increases, likelihood also increases; therefore, the ML criterion cannot be used for a stop criterion because of the over-fitting problem. The primary objective of this thesis is to automatically estimate an adequate number of parameters for acoustic models.

To avoid the over-fitting problem, information criteria, especially the Minimum Description Length (MDL) criterion and the Bayesian Information Criterion (BIC), have been proposed for decision tree clustering. Information criteria can consider the number of both parameters and training samples, and they can be used as stop criteria. In this dissertation, we propose the Maximum Likelihood Successive State Splitting (ML-SSS) algorithm based on the MDL criterion. The ML-SSS algorithm can create

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0261013, March 24, 2005.

both contextual and temporal variations, whereas the decision tree clustering can create contextual variations only. Experiments show that the proposed method, MDL-SSS, can stop splitting automatically and obtain more appropriate models than those by the baseline method, the ML-SSS.

However, since information criteria were derived for simple models under large amounts of training data, theoretically speaking, it is difficult for information criteria to evaluate complicated models like HMMs precisely. Therefore, they do not work well for small amounts of data. To avoid this problem, we utilize the Variational Bayesian (VB) approach, which has recently been proposed in machine learning. The VB approach is applied to the ML-SSS to automatically create efficient models even for small amounts of data. In experiments, this proposed method, the VB-SSS, can stop splitting automatically for small amounts of training data that the MDL-SSS cannot work well for. Additionally, the VB-SSS can obtain almost the same performance for smaller models than those by the baseline method.

Furthermore, to improve the performance of language models, we propose a model that includes word patterns extracted from parse trees. Recently, Structured Language Models (SLMs) have been proposed, which can represent information on sentence structures, while word n-gram models only represent local relations of word concatenation. However, conversational speech includes relatively shorter sentences, and phrase structures are more important for short utterances. Our proposed method includes two processes. First, each phrase is modified to a new phrase with an easier form, and modified-word trigram models are created. Second, using relations in each parse tree, word patterns are extracted from parse trees, and they are used as models. Experiments demonstrate that modified-word trigram models obtained a strong improvement, and the performance was marginally improved by adding word pattern models. In additional experiments, the word pattern models were more effective for long sentences. Finally, combination of the acoustic model by the MDL-SSS and the word pattern models was evaluated. Experiments showed that our proposed methods are effective.

Keywords:

speech recognition, acoustic model, MDL criterion, variational Bayesian approach, language model, parse tree

音声認識のためのモデル自動生成法*

實廣 貴敏

内容梗概

現在、一般に用いられる音声認識技術は確率モデルに基づいており、音響モデル、言語モデルとも、大量な学習データベースから各パラメータを推定する。このとき問題になるのが、学習データ量とパラメータサイズの関係である。パラメータサイズが小さすぎると性能を十分得られず、学習データを生かしきることができない。また、パラメータサイズが大きすぎると、学習データに強く依存したモデルになり、逆に精度を落とすことになる。これは一般に過学習と呼ばれる。学習データに対し、適切なパラメータサイズを選択することは重要な問題である。

音響モデルとして現在の主流技術である音素環境依存型隠れマルコフモデル (hidden Markov Model, HMM) は、状態共有構造を学習データから推定し、作成することが一般的になっている。手法としては、音素カテゴリを利用した音素決定木クラスタリングが手法として広く用いられている。一般的にはゆう度最大化 (Maximum Likelihood, ML) 規準を用いてクラスタリングを行う。しかし、パラメータ数が増加するにつれ、ゆう度は一般に増加する。過学習を起こしやすく、ML 規準のみでは停止条件として使うことができない。本研究では、はじめに音響モデルの適切なパラメータサイズを自動推定することを目標とする。

過学習の問題を避けるために、情報量規準、中でも最小記述長規準 (Minimum Description Length, MDL) または Bayesian Information Criterion (BIC) を用いた音素決定木クラスタリングが提案されている。情報量規準を用いることで、パラメータ数および学習サンプル数を考慮することができ、分割規準だけでなく、停止規準として用いることができ、自動的にパラメータ数の決定に用いることができる。本研究では、MDL 規準をゆう度最大化規準逐次状態分割法 (Maximum Likelihood Successive State Splitting algorithm, ML-SSS) に適用することで、決定木クラスタリングでは扱うことのできない時間方向の状態長を各異音モデルごとに自動推定

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DD0261013, 2005年3月24日.

できるアルゴリズムを実現する。評価実験により、提案手法 MDL-SSS 法は自動的に分割停止可能で、従来法 ML-SSS 法に比べ、より適切な状態共有構造を得ることができることがわかった。

また、情報量規準は、簡単なモデルにおいて大量データに対し導出されたもので、HMM のような複雑なモデルを厳密に扱うことはできない。また、少量データではうまく働かないことが多い。そこで、近年、機械学習の分野で提案されている変分ベイズ法を利用する。ML-SSS 法に変分ベイズ法を適用し、少量データにおいても効率的なモデルを自動生成できる手法を提案する。評価実験では、MDL-SSS 法がうまく働かない少量データにおいて、自動分割停止が可能であり、従来法 (ML-SSS) よりパラメータ数の少ないモデルで同程度以上の性能が得られることがわかった。

次に、言語モデルの精度向上を目指して、構文木から単語パターンを抽出し、モデル化したモデルを提案する。単語 N-gram モデルが局所的な単語連鎖のみを考慮したものであるのに対して、近年、提案されている構文解析そのものを言語モデルに用い、文全体の構造情報を利用するモデルが提案されている。しかし、対話文のように、比較的短い文では、むしろ文節レベルでの構造が重要である。そこで、まず、前処理として単語連鎖などから、より扱いやすいパターンに変形し、trigram モデルとして用いる。さらに、構文木から各単語に木構造内で関連のある単語パターンを抽出してモデルとして用いる。評価実験から、変形単語 trigram モデルにより大きな改善が得られ、さらに単語パターンモデルで若干の改善が得られた。また、単語パターンモデルは長い文に対し、特に効果的であることが分かった。最後に、提案手法である MDL-SSS 法による音響モデル、単語パターン言語モデルでの組合せで評価を行い、効果が得られることを確認できた。

キーワード

音声認識, 音響モデル, 最小記述長規準, 変分ベイズ法, 言語モデル, 構文木

Acknowledgment

First, I would like to thank my boss, Dr. Satoshi Nakamura of ATR Spoken Language Laboratories, for so much advice, encouragement, and for admitting me to his group. I still consider this period at ATR is the last chance for me to perform research.

Next, let me express my gratitude to my supervisor, Prof. Kiyohiro Shikano of Nara Institute of Science and Technology. Without his suggestion to enter NAIST, I would never have written this thesis. I appreciate Prof. Naokazu Yokoya and Prof. Hiroshi Saruwatari for their advice.

I also thank Prof. Yoshinori Sagisaka, Waseda Univ., Dr. Genichiro Kikui, ATR-SLT, and Dr. Hirofumi Yamamoto, particularly for guiding me toward language modeling.

I am very grateful to Prof. Tomoko Matsui of ISM. She has been my first supervisor, for only two months, after I joined NTT Human Interface Labs in 1993. She also gave me advice after I joined Dr. Nakamura's group at ATR. It is still of great benefit for me to talk with her about NTT, other researchers, and so on. I wish to thank the members of our department for all their advice, especially Dr. Konstantin Markov and Dr. Frank K. Soong who gave me a lot of questions and comments at every presentation.

I also feel grateful to Prof. Shigeki Sagayama of Tokyo University, Prof. Kiyooki Aikawa, and Prof. Takeshi Kawabata, who were my bosses at NTT Labs. before I was transferred to ATR. They led me to become a researcher, even though our group was focused on creating real products.

Finally, I would like to express my deepest gratitude to my parents for their support.

Contents

List of Figures	xii
List of Tables	xiii
List of Notations	xv
1 Introduction	1
1.1. Current Status of Automatic Speech Recognition	1
1.2. Objective of Automatic Acoustic Modeling	3
1.2.1 Acoustic model generation	4
1.2.2 Criteria for selection of acoustic models	5
1.2.3 Objective of this thesis for automatic acoustic modeling	8
1.3. Objective of Language Modeling Using Parse Trees	9
1.3.1 Structured language modeling	9
1.3.2 Objective of this thesis for structured language modeling	11
1.4. Overview of This Thesis	12
2 Automatic Speech Recognition	13
2.1. Introduction	13
2.2. Acoustic Modeling	14
2.3. Topology Training of Acoustic Modeling	14
2.3.1 Decision Tree Clustering	15
2.3.2 Successive State Splitting algorithm	16
2.3.3 Gain function by ML-SSS algorithm	20
2.4. Information Criteria for Model Selection	21
2.4.1 Akaike Information Criterion	22

2.4.2	Bayesian Information Criterion	22
2.4.3	Minimum Description Length Criterion	23
2.5.	Variational Bayesian Approach for Model Selection	23
2.5.1	Bayesian Learning	24
2.5.2	Variational Bayesian approach	25
2.6.	Language Modeling	29
2.6.1	Word n-gram model	29
2.6.2	Trigger model	29
2.6.3	Structured Language Model (SLM)	30
2.7.	Summary	31
3	Successive State Splitting Algorithm Based on Minimum Description Length Criterion	33
3.1.	Introduction	33
3.2.	SSS Algorithm Using MDL Criterion	34
3.2.1	Flow of MDL-SSS algorithm	34
3.2.2	Gain function by MDL-SSS	35
3.3.	Experiments	39
3.3.1	Conditions	39
3.3.2	Comparison of gender-independent models with single Gaussian	40
3.3.3	Comparison of gender-dependent models with five Gaussians	42
3.3.4	Effectiveness for different amounts of training data	42
3.3.5	Evaluation using lecture speech	46
3.4.	Summary	49
4	Variational Bayesian Approach for the SSS Algorithm	51
4.1.	Introduction	51
4.2.	Variational Bayesian Approach for SSS Algorithm	52
4.2.1	Overview of VB-SSS	52
4.2.2	Contextual and temporal splitting	53
4.2.3	Priors	55
4.2.4	Posteriors	56
4.2.5	Objective function	57
4.3.	Increasing Mixture Components Based on the VB Approach	57

4.3.1	Splitting mixture method	57
4.3.2	VB approach for increasing mixture components	58
4.4.	Experiments	60
4.4.1	Experimental conditions	60
4.4.2	Evaluation for topology training	61
4.4.3	Evaluation of mixture splitting	66
4.5.	Summary	68
5	Language Modeling Using Patterns Extracted from Parse Trees	71
5.1.	Introduction	71
5.2.	Phrase Structure Representation	72
5.2.1	Phrase structure using constituent boundaries	72
5.2.2	Parsing procedure	74
5.3.	Proposed Language Models	75
5.3.1	Modified word trigram models	75
5.3.2	Word pattern models	78
5.3.3	Formulation	79
5.3.4	Training and recognition schemes	80
5.4.	Experiments	80
5.4.1	Experimental conditions	80
5.4.2	Evaluation by perplexity	82
5.4.3	Evaluation by word accuracy	83
5.5.	Evaluation of Pattern Models	84
5.5.1	Performance in the APP task	84
5.5.2	Evaluation for the length of sentences	86
5.6.	Evaluation of ASR Using MDL-SSS and Pattern Language Model	88
5.6.1	Evaluation	90
5.6.2	Experimental Conditions	90
5.6.3	Experimental Results	91
5.7.	Summary	94
6	Conclusion	95

A Definitions for the VB-SSS Algorithm **99**
 A.1. Objective Function of the VB-SSS Algorithm 99

Bibliography **106**

List of Publications **107**

List of Figures

1.1	Automatic Speech Recognition.	2
1.2	Hidden Markov Model.	3
1.3	A curve of BIC or MDL criterion.	6
1.4	Representation of unknown parameters in (a) ML estimation, and (b) Bayesian estimation. Each dot represents each training sample, and a “x” mark means a mean vector of these training samples. A ellipse with dashed lines represents the deviation of the mean vector.	7
1.5	Several types of language model.	10
2.1	Shared-state triphone models.	15
2.2	Example of phonetic decision tree clustering.	16
2.3	Contextual splitting and temporal splitting.	17
2.4	Flow chart of the ML-SSS algorithm. N_s is the total number of states and N_p is the maximum number of states in one allophone.	18
2.5	Contextual splitting by using Chou’s algorithm [7].	19
2.6	Principle of the Variational Bayesian approach.	26
3.1	Flow chart of MDL-SSS algorithm.	35
3.2	Overview of MDL-SSS algorithm.	38
3.3	Word accuracy for GI models with one Gaussian distribution per state trained by using TRA and BLA.	41
3.4	Word accuracy for GD models with five Gaussian mixtures trained by using TRA and BLA.	43
3.5	The number of states for each phoneme.	44
3.6	The maximum length of states for each phoneme.	44
3.7	Word accuracy for models trained by using TRA data.	45

3.8	Word accuracy for the CSJ corpus.	47
3.9	The number of states for each phoneme for the CSJ corpus.	48
3.10	The maximum number of states per phoneme for the CSJ corpus.	48
4.1	Flow of the Variational Bayesian SSS algorithm.	54
4.2	Splitting each distribution into two distributions.	58
4.3	Average phoneme recognition rates for vowels.	62
4.4	Average phoneme recognition rates for consonants.	62
4.5	Word accuracy rates by single Gaussian models.	63
4.6	Word accuracy rates by Gaussian mixture models.	65
4.7	Word accuracy rates by four types of combinations.	67
5.1	Structures of (a) “ <i>I go</i> ” and (b) “ <i>I go to Kyoto.</i> ” (a) The marker, “ <i><pronoun-verb></i> ,” is inserted between “ <i>I</i> ” and “ <i>go</i> ,” and the structure is a pattern “ <i>X <pronoun-verb> Y.</i> ” (b) An example structure is made by patterns, “ <i>X <pronoun-verb> Y</i> ” and “ <i>X to Y.</i> ” “Head” means a headword that is representative of a sub-tree.	74
5.2	Structures of example Japanese sentences. These structures are constructed by (a) patterns “ <i>X wa Y</i> ” and “ <i>X e Y</i> ” and (b) patterns “ <i>X <pronoun-> Y</i> ” and “ <i>X e Y,</i> ” respectively.	75
5.3	Modified word trigram models.	76
5.4	Extraction of pattern models from a parse tree.	77
5.5	Training and recognition schemes.	81
5.6	Perplexity improvement rates for the TRA task.	87
5.7	Perplexity improvement rates for the APP task.	87
5.8	Error reduction rates for the TRA task.	89
5.9	Error reduction rates for the APP task.	89
5.10	Performance by the ML-SSS and the MDL-SSS with the multi-class composite bigram models, or rescoring by the word pattern models.	92

List of Tables

3.1	Average speaking rate for each training data	49
4.1	Word accuracy rates [%] and # of states in parentheses for several hyperparameters	64
4.2	Word accuracy rates for thirty minutes of training data	64
4.3	The average number of mixture components per state, the total number of mixture components, and word accuracy rate	66
5.1	Text data	82
5.2	Total number of entries	83
5.3	Perplexity for the TRA task	84
5.4	Word accuracy for the TRA task	85
5.5	APP Test Set	85
5.6	Perplexity for the APP task	86
5.7	Word accuracy for the APP task	86
5.8	Perplexity for the TRA task by several language models used in this chapter	91

List of Notations

Notations

$P()$	Discrete probability
$p()$	Probability density function
$q()$	Variational posterior probability (probability density function)
$\langle f(x) \rangle_{p(x)}$	Expectation of $f(x)$ as $\langle f(x) \rangle_{p(x)} = \int f(x)p(x)dx$
$\{x_n\}_{n=1}^N$	Set of elements as $\{x_1, x_2, \dots, x_N\}$
S_i	State of HMMs
\mathbf{O}	Sequence of feature vectors (observation data), $\{\mathbf{o}_t\}_{t=1}^T$
\mathbf{W}	Sequence of words, $\{w_n\}_{n=1}^N$
Θ	Set of HMM parameters, $\{\theta_i\}_{i=1}^I$
Z	Set of latent variables
$\gamma_t(S_i)$	Probability of staying in state S_i at time t
$\Gamma(S_i)$	Expected frequency of transition from state S_i as $\Gamma(S_i) = \sum_{t=1}^T \gamma_t(S_i)$
$\xi_t(S_i, S_j)$	Probability of transition from S_i to S_j at time t
$\Xi(S_i, S_j)$	Expected frequency of transition from S_i to S_j as $\Xi(S_i, S_j) = \sum_{t=1}^T \xi_t(S_i, S_j)$
N_s	Number of states
N_a	Number of transition probabilities
\mathbf{a}	Set of transition probabilities as $\{a_{ij}\}_{i=1, j=1}^{N_s, N_a}$
$\boldsymbol{\mu}$	Set of mean vectors of mixture components as $\{\boldsymbol{\mu}_{ik}\}_{i=1, k=1}^{N_s, M_i}$
$\boldsymbol{\Sigma}$	Set of covariance matrices of mixture components as $\{\boldsymbol{\Sigma}_{ik}\}_{i=1, k=1}^{N_s, M_i}$
\mathbf{w}	Set of mixture weights as $\{w_{ik}\}_{i=1, k=1}^{N_s, M_i}$
$\mathcal{N}()$	<i>Gaussian</i> distribution
$\mathcal{G}()$	<i>Gamma</i> distribution
$\mathcal{T}()$	<i>Student-t</i> distribution

Abbreviations

AIC	Akaike Information Criterion
ASR	Automatic Speech Recognition
BIC	Bayesian Information Criterion
GD	Gender Dependent
GI	Gender Independent
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
MDL	Minimum Description Length
MFCC	Mel-Frequency Cepstrum Coefficient
ML	Maximum Likelihood
SLM	Structured Language Model
SSS	Successive State Splitting (algorithm)
VB	Variational Bayesian (approach)

Chapter 1

Introduction

1.1. Current Status of Automatic Speech Recognition

People have long been dreaming of Automatic Speech Recognition (ASR) systems, which are used for machines to talk with humans, evidenced by their presence in many science fiction novels and movies. ASR has been studied and developed over the last 50 years. In the last decade, some applications that include ASR systems have been brought to market, for example, dictation systems on PCs, voice-activated telephony portal sites, and car navigation systems with voice commands. Such growth has been made possible by not only drastic computational advancements, but also progress in stochastic modeling techniques, and an increasing amounts of training data.

However, ASR systems that have been expected for a long time are still not fully developed. Most current ASR systems can obtain high performance for tasks with a small vocabulary, or read speech like the reading of newspapers if ASR systems are used in quiet environments. On the other hand, it is still difficult to recognize natural speech, i.e., spontaneous speech, automatically. To innovate in ASR, further deep consideration from many basic points is still required.

Figure 1.1 shows a block diagram of automatic speech recognition. First, speech input is analyzed by a feature extraction part, with characteristics of speech mainly represented by frequency envelopes. Currently, noise robustness is the main problem in feature extraction. Many methods with robust feature extraction or enhancement methods have been investigated.

Second, for input feature parameters, a recognizer, i.e., a decoder, searches the best

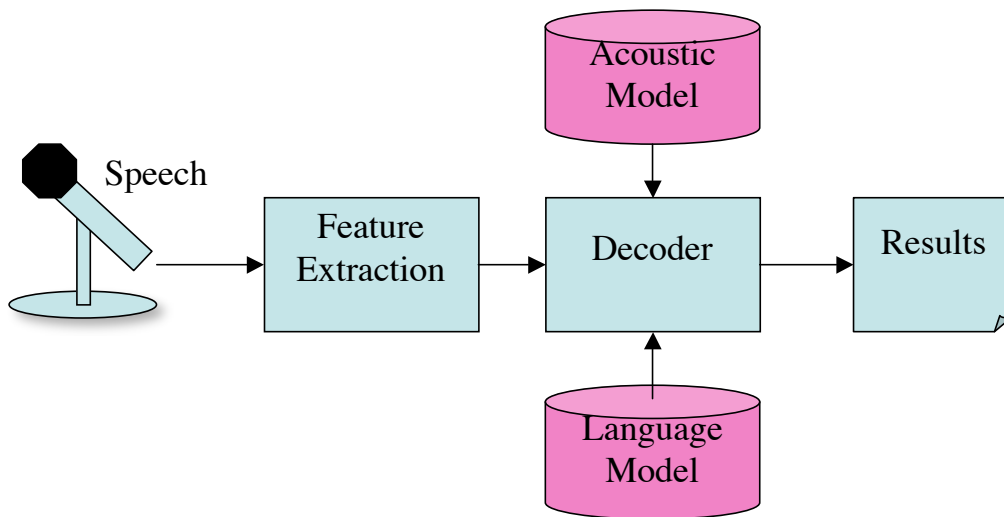


Figure 1.1. Automatic Speech Recognition.

recognized candidate that fits both acoustic models and language models. A recognizer gives isolated words or sentences as results. The multi-pass search strategy has become the most popular for decoding. For example, in the first pass, simpler and weak models are used, and then in the second pass, stronger models are employed for recognized candidates obtained during the first pass.

Before recognizing speech, both acoustic and language models must be created, and these days, stochastic models are used. Hidden Markov Models (HMMs) are used as acoustic models, and n-gram models are used as language models.

Figure 1.2 shows an example of an HMM. It includes plural states to represent feature sequences, for example, a phoneme. To represent speech characteristic and dynamics by state sequences, each state has transition probabilities and a number of distributions. Gaussian distribution is usually used for a mixture component. Therefore, this model is referred to as a Gaussian Mixture HMM. Additionally, each Gaussian distribution includes a mean vector and a covariance matrix. These parameters included in HMMs should be estimated by using training data. It is difficult to find the best set of parameters for HMMs because there are a lot of parameters and an HMM's performance is dependent on the parameter size and the amount of training data.

For language modeling, word n-gram models are widely used, especially word

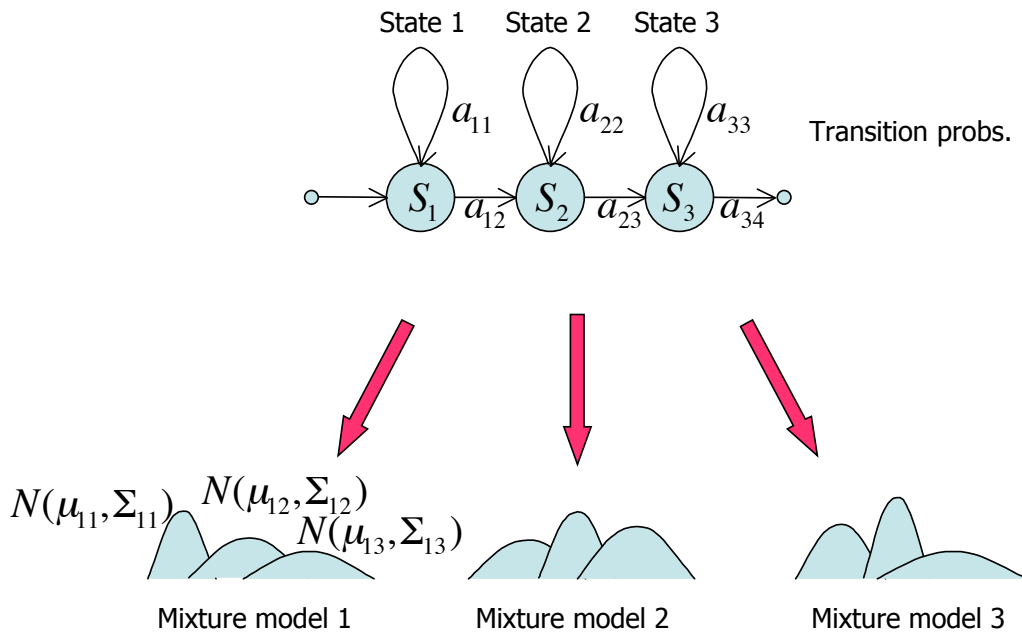


Figure 1.2. Hidden Markov Model.

trigram models in the following:

$$P(\mathbf{W}) \approx P(w_1)P(w_2|w_1)P(w_3|w_1w_2) \dots P(w_N|w_{N-2}, w_{N-1}), \quad (1.1)$$

where $P(\mathbf{W})$ is a probability of a word sequence $\mathbf{W} = \{w_1, w_2, \dots, w_N\}$. A trigram is a set of three successive words; therefore, word trigram models only represent local constraints.

In this paper, to improve performance of speech recognition, the aim is to develop a method to create adequate acoustic models automatically, and structured language modeling with more phrasal constraints. In the following sections, we will describe objectives both for acoustic modeling and language modeling.

1.2. Objective of Automatic Acoustic Modeling

One of the problems in ASR systems is how to obtain the optimal model from a given training database. ASR's performance is dependent on both the amount of training data

and the size of models for statistical models like HMMs and n-gram language models, which have become popular techniques in last decade.

Although there are many methods to obtain models from training data automatically the number of parameters is usually selected manually to obtain the best performance. Therefore, if the amount of training data or a recognition task is changed, it is necessary to find the optimal model manually. For example, in some telephone portal sites, the amount of speech data is increasing while services are provided if utterances can be collected. Using such collected data, it is possible to improve recognition performance by not only re-estimation of parameters but re-design of models, especially re-definition of the number of parameters. The re-created models can be used for updating ASR's models.

Additionally, if the number of parameters is too small, it is difficult to obtain enough performance; on the other hand, if the number of parameters is too large, the obtained model is strongly dependent on the training data, and it cannot obtain sufficient performance. The latter problem is generally referred to as the over-fitting problem. It is very important to select an adequate model size for training data, and some good methods are needed to find the optimal model without over-fitting even to the small data.

Therefore, there is a pressing need to automatically design methods for models to improve ASR's performance.

1.2.1 Acoustic model generation

In most current ASR systems, a model unit represented by HMMs is a sub-word unit, or a phoneme. Context-dependent HMMs are the most popular of these and depend on previous units and successive units. Such a model is referred to as an "allophone model," especially a "triphone model," which depends on both only one previous unit and only one successive unit. Each allophone is represented by one HMM.

These days, Gaussian mixture HMMs are widely used as acoustic models. Each Gaussian mixture HMM has plural states, and each state features a number of Gaussian mixture components and transition probabilities. Each mixture component possesses a mean vector and covariance matrix.

One of the greatest problems is how to decide the total number of states, the number of state per allophone, and the number of mixture components per state. Addi-

tionally, parameter tying should be considered to obtain robust models. If an HMM is created for each triphone, some triphone models would not have a sufficient amount of training data, and estimated parameters may over-fit the limited data. Therefore, context-dependent models are usually created by tying contextual information. Shared state models are the most popular for context-dependent models.

The methods of automatic generation for context-dependent models have been proposed as follows.

- Bottom-up clustering by agglomerated methods [1]
- Top-down clustering based on distortions of models [2][3]
- Decision Tree Clustering[4] [5]
- Successive State Splitting Algorithm [6][7]

1.2.2 Criteria for selection of acoustic models

In general, model generation can be considered as a model selection problem, and the type of model-selection criterion is important to performance. Three criteria have been used for acoustic modeling.

- Maximum Likelihood (ML) criterion
Decision Tree Clustering [5], or the SSS algorithms [6][7] use the ML criterion as a clustering or splitting criterion. The ML criterion has the over-fitting problem. If the number of parameters becomes larger, likelihood increase. Therefore, it is difficult to finish splitting and to select the best model.
- Information criterion
To overcome the over-fitting problem in the ML criterion, information criteria have been used as split and stop criteria. The Akaike Information Criterion (AIC) was used to create context-independent HMMs [8]. For context-dependent HMMs, decision tree clustering methods have been proposed, which are based on the Minimum Description Length (MDL) Criterion [9][10] and the Bayesian Information Criterion (BIC) [11][12]. Note that the MDL criterion

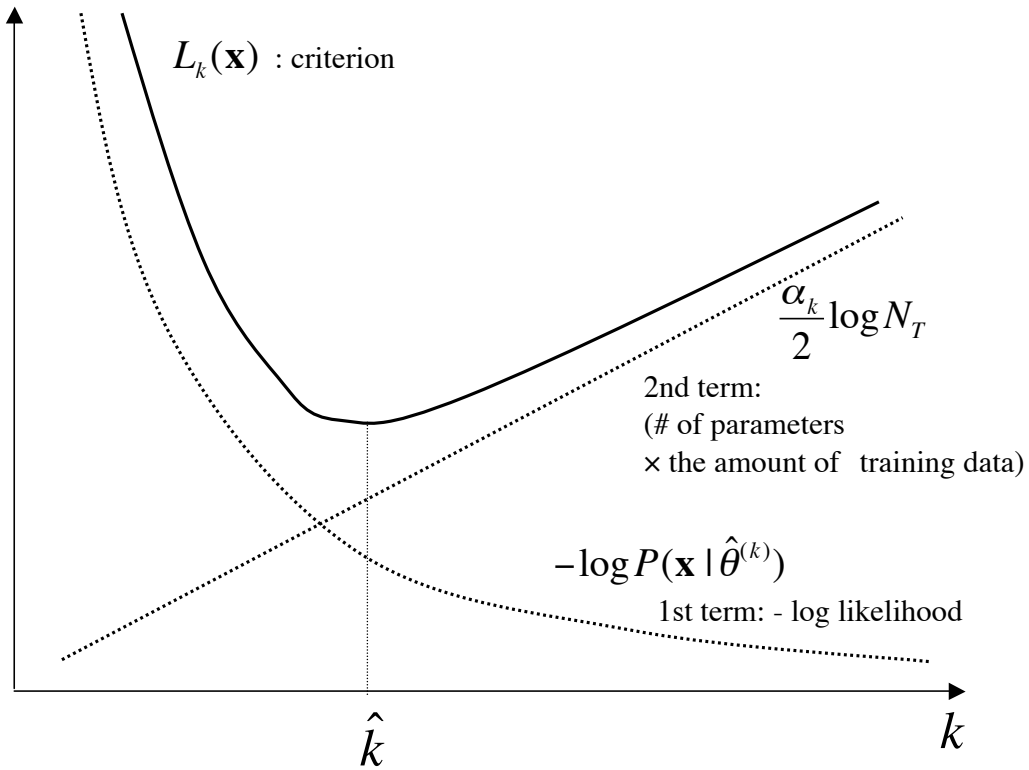


Figure 1.3. A curve of BIC or MDL criterion.

and the BIC provide the same criteria as the split and stop criterion. The MDL criterion for model k is defined in the following:

$$L_k^{(MDL)}(\mathbf{x}) = -\log P(\mathbf{x}|\hat{\theta}^{(k)}) + \frac{\alpha_k}{2} \log N_T + \log K, \quad (1.2)$$

where $\mathbf{x} = \{x_1, \dots, x_{N_T}\}$ is observation data, α_k is the number of free parameters, and $\hat{\theta}^{(k)}$ is the ML estimate of model k . The first term is the inverse sign of the log likelihood of model. The second term and the following term are related to the number of parameters, or the number of samples. The third term is usually constant. The BIC is the same as the MDL criterion if the third term of the MDL is not considered. Figure 1.3 shows how information criteria work to avoid over-fitting to training data when the number of parameters increases under the fixed training data. The likelihood value increases when the number of parameters grows. It is difficult to select the best model only from likelihood.

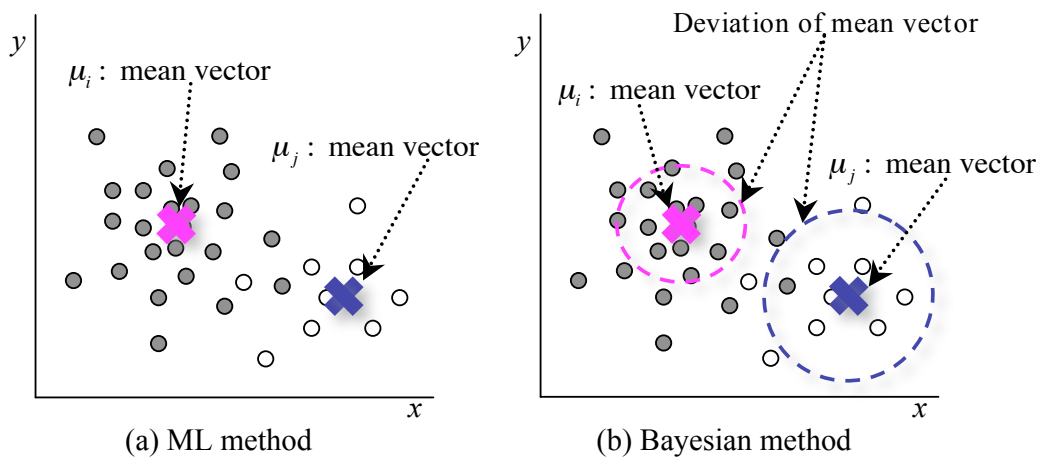


Figure 1.4. Representation of unknown parameters in (a) ML estimation, and (b) Bayesian estimation. Each dot represents each training sample, and a “x” mark means a mean vector of these training samples. A ellipse with dashed lines represents the deviation of the mean vector.

On the other hand, since the second term is also increasing, the criterion has the smallest value for a set of parameters. This model represents the best model in this information criterion. However, since information criteria are derived by using some assumptions, information criteria cannot exactly evaluate complicated models like neural networks and HMMs.

- Variational Bayesian approach

The Variational Bayesian (VB) approach was proposed as an approximated solution of Bayesian learning in the field of machine learning [13]. Additionally, a general VB framework was proposed, and the EM steps of the VB approach were defined in [14][15].

Bayesian learning considers that all unknown parameters are stochastic variables, while maximum likelihood estimation assumes that all unknown parameters are deterministic variables. Figure 1.4 shows the concept for the representation of unknown parameters in ML estimation, or Bayesian estimation. Bayesian learning also considers that each parameter itself has a distribution, for example, each mean vector in this figure has a mean vector and a covariance matrix, while

each mean vector only has one vector in ML estimation. Each distribution represents the reliability of each mean vector. In Fig. 1.4(b), the distribution of μ_j is broader than that of μ_i because the number of samples for μ_j is smaller than that of μ_i . Therefore, models in Bayesian learning can avoid the over-fitting problem because of model representations.

The Variational Bayesian approach is one approximation method for Bayesian learning. This approach can deal with complicated models by deriving both posterior probabilities and VB objective functions for given models. For model selection, the VB objective function can be used as a criterion to select models. Recently, this approach has been applied to many research areas. For example, in speech recognition, decision tree clustering based on the VB approach was proposed [16].

1.2.3 Objective of this thesis for automatic acoustic modeling

One of the objectives of this work is to create more elaborate and more suitable HMMs automatically. These days, the decision tree clustering is widely employed to make acoustic models, and many methods using different criteria instead of the ML. However, the decision tree clustering does NOT consider temporal variations, although the temporal length of states for each triphone should be dependent on each triphone data. Therefore, the proposed method is based on the SSS algorithm [6][7]. It can create both contextual and temporal variations.

However, as we describe, these methods are based on the ML criterion. The ML criterion has an over-fitting problem and cannot be used as a stop criterion; the total number of states and the maximum length of states are usually used as stop criteria. These parameters are empirical parameters and are dependent on speech data.

First, we introduce the MDL criterion to the SSS algorithm to determine the number of parameters automatically. Both the MDL and BIC are widely used because they can work well for large amounts of data and can be applied easily.

Second, we combine the VB approach with the SSS algorithm. The VB approach can evaluate complicated models more exactly than information criteria. We also apply this approach to increase the number of mixture components for topologies generated by the VB-based SSS algorithm.

1.3. Objective of Language Modeling Using Parse Trees

Word n-gram models are widely used as language models. They can just represent local constraints within a few successive words but lack the ability to capture the global structures of sentences. In general, a sentence has some structures extracted from relations of words. In natural language processing, many methods analyzing sentences have been proposed to extract some structures and relations among words such as syntactic parsing, analysis of modification relations, and so on. These methods are promising to improve language models for speech recognition.

1.3.1 Structured language modeling

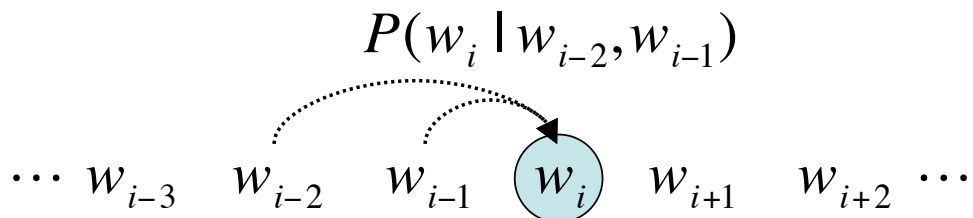
Mainly, there are two types of language model imposing correlation between words and sentence structures. Figure 1.5 shows concepts of several language models.

- Trigger model

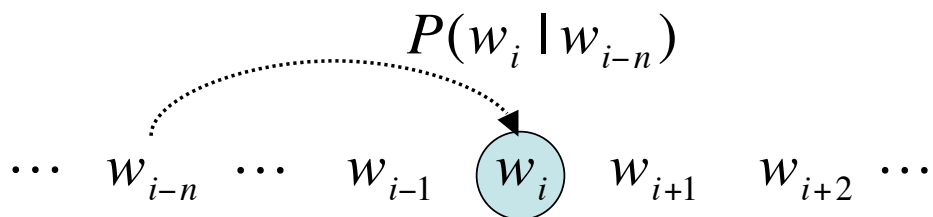
Trigger models have been proposed to represent word co-occurrence characteristics beyond 2-3 grams [17]. Figure 1.5(a) shows a word trigram model. Word n-gram models just represent relations among concatenated words. On the other hand, trigger models can represent distant relations between preceding words like in Figure 1.5(b). S. Zhang et al. also proposed a solution called Linkgram [18], a model that has word pairs extracted from parse trees and can represent syntactic relations between word pairs. Such constraints, however, are weak for the global structures of sentences.

- Structured Language Model

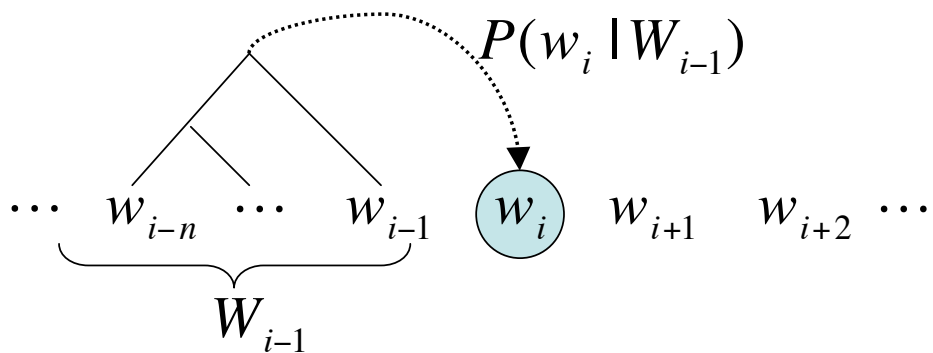
To represent more powerful constraints, a language model using a stochastic parser has been proposed [19][20]. This model is referred to as a Structured Language Model (SLM). Figure 1.5(c) shows the concept of this model, which it operates by three modules: (1) WORD-PREDICTOR predicts next word from previous word sequences and parse subtrees, (2) TAGGER predicts POS tags of next words from word sequences, parse subtrees and predicted next words, and (3) CONSTRUCTOR repeatedly generates a parse tree from subtrees. These modules use conditional probabilities and these probabilities are used for lattice rescoring. Furthermore, some researchers have recently proposed different



(a) Word trigram model



(b) Trigger model



(c) Structured Language Model

Figure 1.5. Several types of language model.

types of structured language model. Mori et al. proposed an SLM based on tree-structured history [21] while Akiba et al. used probabilistic generalized LR parsing as a language model [22]. These models require a large number of parse trees. They represent sentence structures very strictly.

1.3.2 Objective of this thesis for structured language modeling

It is important to introduce sentence structures into language models, especially, to consider long relations among words in one sentence.

We propose two new types of language model that use phrasal constraints extracted from parse trees produced by an example-based parser. For spontaneous speech, extremely strict constraints of sentence structures are not so important, but partial structures are more useful for constraints.

First, we propose n-gram models for sentences with phrase constituent boundary markers. Second, we propose word pattern models using partial structure patterns of parse trees. Both of these methods attach weight to constraints of partial phrase structures. These models are created from outputs of the example-based parser, called the Constituent Boundary Parser (CBP), which was developed by Furuse et al. at ATR[23], for example-based machine translation, called Transfer-Driven Machine Translation (TDMT). The parser analyzes sentences by example word patterns and rules produced manually.

These proposed models extracted by the parser can represent not only intra-phrase constraints, but also inter-phrase constraints, the latter of which appear as syntactic long-distance constraints between words. Therefore, the proposed models can represent local structures extracted from only surface word sequences after preprocessing, and larger structures extracted from subtrees of parse trees. Particularly in spontaneous speech, there are so many sentences with ungrammatical global structures but with grammatical local structures. It is more important for spontaneous speech recognition to represent the local constraints.

1.4. Overview of This Thesis

In the next chapter, we will present an overview of automatic speech recognition. Following that, there are two topics on automatic model generation of acoustic modeling. One is the MDL-based SSS algorithm in Chapter 3, the other is the SSS algorithm based on the VB approach in Chapter 4. In Chapter 5, we will describe word pattern language modeling, extracting from parse trees. Furthermore, the combination of the MDL-SSS and pattern language models will be evaluated to verify our proposed methods. Finally, this thesis will conclude in Chapter 6.

Chapter 2

Automatic Speech Recognition

2.1. Introduction

This chapter describes the standard techniques of automatic speech recognition. Speech recognition is to estimate a word sequence $\hat{\mathbf{W}}$ that generates a given acoustic observation sequence \mathbf{O} . Using the maximum a posteriori probability $P(\mathbf{W}|\mathbf{O})$, speech recognition can be formulated by the following equation:

$$P(\hat{\mathbf{W}}|\mathbf{O}) = \max_{\mathbf{W}} P(\mathbf{W}|\mathbf{O}). \quad (2.1)$$

Equation (2.1) can be rewritten by Bayes' Rule,

$$P(\mathbf{W}|\mathbf{O}) = \frac{P(\mathbf{O}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{O})}. \quad (2.2)$$

Since $P(\mathbf{O})$ is independent of \mathbf{W} , Eq. (2.1) becomes the following:

$$\hat{\mathbf{W}} = \operatorname{argmax}_{\mathbf{W}} P(\mathbf{O}|\mathbf{W})P(\mathbf{W}), \quad (2.3)$$

where $P(\mathbf{O}|\mathbf{W})$ is a probability of an acoustic model. In fact, it is the probability of an acoustic observation sequence conditioned on the given word sequence. These days, this probability is represented by Hidden Markov Models (HMMs). $P(\mathbf{W})$ is a probability of a language model: it is the probability of a word sequence including relations among words. The word n-gram models are generally used to estimate this probability.

The following section briefly describes modeling techniques to estimate these two probabilities. First, acoustic modeling using HMMs is described in Section 2.2. Second, as techniques related to this thesis, topology training of acoustic modeling is

explained in Section 2.3, and information criteria and Variational Bayesian approach are described in Section 2.4, and Section 2.5, respectively. In Section 2.6, language modeling is explained, including conventional methods and models related to this thesis.

2.2. Acoustic Modeling

Acoustic modeling needs feature extraction from speech, model units, and specific representation of each unit. As a feature of speech, spectral envelopes are important, especially, in speech recognition. After several decades of research, for feature extraction, the mel-frequency cepstrum coefficient (MFCC) has become widely used. It is a non-parametric representation that includes mel-filter banks and smoothing by cepstrum coefficients.

For model units, sub-words, especially phonemes, are widely used because it is necessary in large-vocabulary speech recognition to create pronunciations of words. Whole word models are also used for small vocabulary tasks, especially, digits.

These days, HMMs are generally employed as representations of acoustic models. There are several types of HMM, namely, discrete, continuous-density, and semi-continuous-density HMMs. Especially, continuous-density HMMs are widely used, and HMMs in this thesis refers to continuous-density HMMs. As shown in Figure 1.2, one HMM represents one unit, that is, a phoneme, and it has several states which include transition probabilities and distributions of speech features.

Figure 1.2 shows a Gaussian mixture HMM that is widely used for acoustic models as we described in Section 1.1. Each Gaussian mixture HMM has plural states, and each state has a number of Gaussian mixture components and transition probabilities. Each mixture component includes a mean vector and covariance matrix.

2.3. Topology Training of Acoustic Modeling

Context-dependent models are effective and are widely used, and a shared-state structure of a context-dependent model is needed to obtain robustness against insufficiency in the amount of training data. If the number of units, i.e., subwords, or phonemes, is 40, then 64,000 triphone models are needed. Unfortunately, it is difficult to ob-

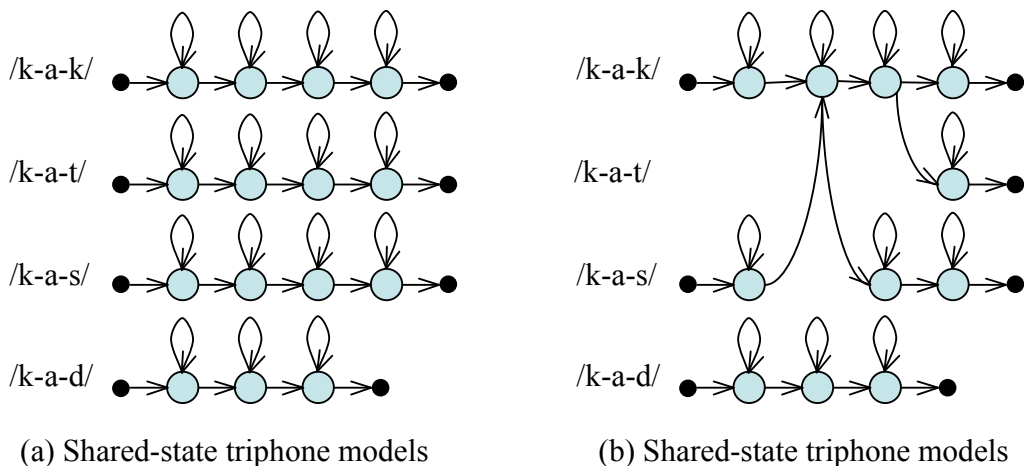


Figure 2.1. Shared-state triphone models.

tain enough data for each triphone model. Figure 2.1 shows examples of shared-state triphone models, with 2.1(a) illustrating unshared-state triphone models. Each triphone has its own state, though it is difficult to estimate parameters robustly because training data is small for some triphone models. In (b), a few states are shared by the other triphone models. Since each set of training data is shared in shared states, and parameters can be reduced, it is possible to estimate parameters robustly.

The training to obtain shared-state structures from training data is called “Topology Training.” In this section, two major methods to create shared-state structures are explained. One is the decision tree clustering, and the other is the Successive State Splitting (SSS) algorithm.

2.3.1 Decision Tree Clustering

Although many types of Phonetic Decision Tree Clustering have been proposed, the PDT clustering proposed by S. J. Young et al. [5] is widely used because it matches up-to-date acoustic models, that is, continuous-density HMMs.

Here, a set of phoneme categories is needed, and each category is used as a question to split one class into two. Before clustering, the number of states for each triphone should be fixed, and state alignments are fixed by an initial model. A single Gaussian is estimated for each state of each allophone, and all Gaussians are collected in one

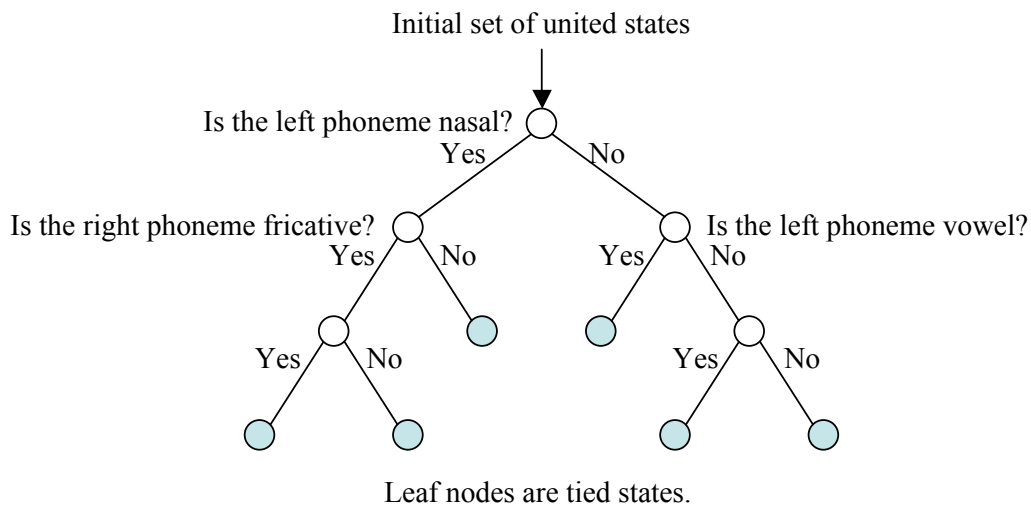


Figure 2.2. Example of phonetic decision tree clustering.

node, which is called “the root node.”

Figure 2.2 shows an example of phonetic decision tree clustering. Each question is applied to classify distributions into two classes at each clustering. Questions are usually related to phoneme categories, for example, “is the phone on the left of the current phone an /a/?” Each distribution is classified according to a question. After that, one distribution is estimated for each class, and a likelihood gain is calculated. Finally, the question with the maximum likelihood gain is selected, and each clustering is finished. This clustering is continued until no gain is obtained, or the total number of nodes (= states) is sufficient. The maximum number of states is usually set before clustering and it is used as a stop criterion.

2.3.2 Successive State Splitting algorithm

The Successive State Splitting (SSS) algorithm was originally proposed by J. Takami and S. Sagayama to create a network of HMM states of speaker-dependent models [6]. This method can create both contextual and temporal variations. The SSS algorithm iteratively constructs the appropriate context-dependent model topologies by finding a state that should be split in each iteration and then it re-estimates the parameters of HMMs based on the ML criterion. This algorithm supposes the two types of split-

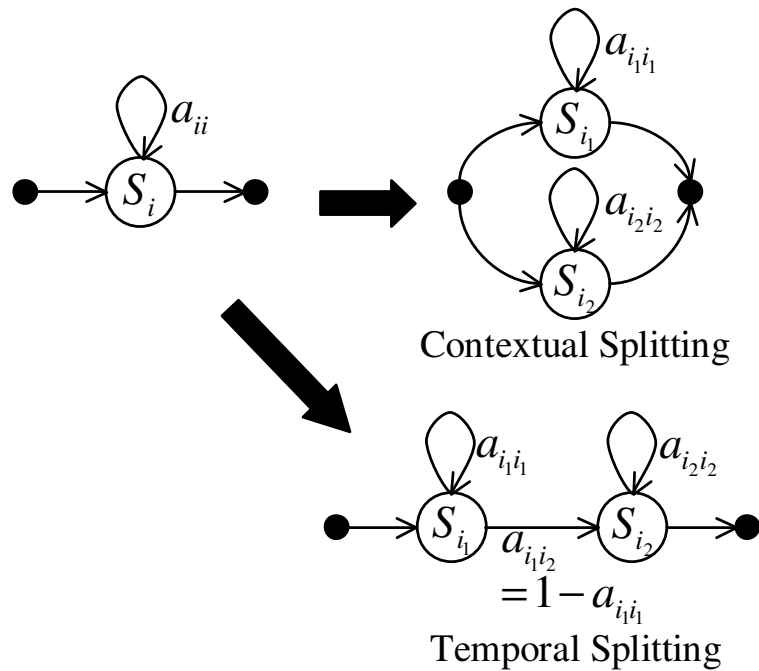


Figure 2.3. Contextual splitting and temporal splitting.

ting shown in Fig. 2.3. However, it cannot be applied to generate speaker-independent models because data-driven clustering without contextual information is used for contextual splitting. Therefore, it was subsequently expanded to the ML-SSS algorithm by M. Ostendorf and H. Singer to create speaker-independent models by data-driven clustering with contextual information [7]. The ML-SSS algorithm includes not only contextual splitting by P. A. Chou's algorithm [24], but also temporal splitting that can extend each triphone. Therefore, this algorithm does not need to decide beforehand the temporal length of states to split for each triphone. Figure 2.4 illustrates the basic procedure of the ML-SSS algorithm. Here, Chou's algorithm is used to assign phoneme contexts into two new states at the contextual splitting stage, as shown in Figure 2.5.

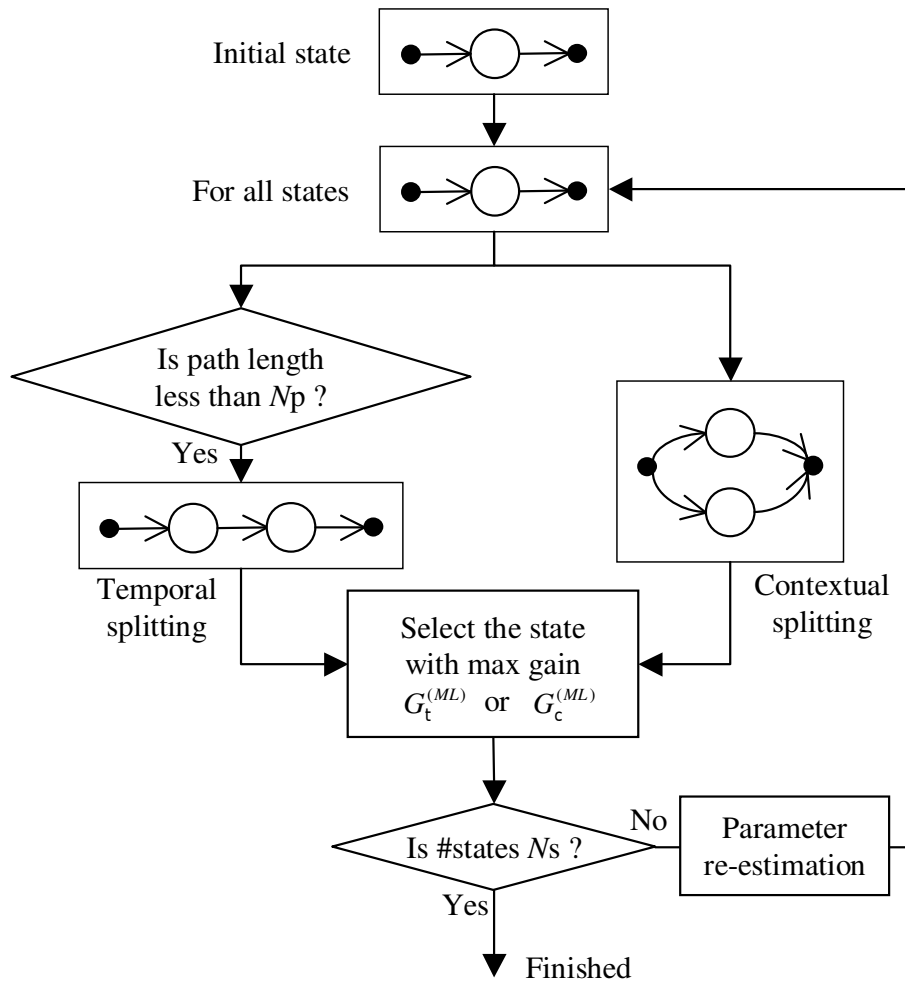


Figure 2.4. Flow chart of the ML-SSS algorithm. N_s is the total number of states and N_p is the maximum number of states in one allophone.

Design of ML contextual splitting algorithm:

1. **Initialization** (iteration number, $n = 0$): Initialize the distribution parameters of the two hypothesized states $S_{i_1}^{(c)}, S_{i_2}^{(c)}$ for each state S_i .

$$\theta^{(0)}(S_{i_1}) = \theta(S_i) = (\mu(S_i), \Sigma(S_i)),$$

$$\theta^{(0)}(S_{i_2}) = ((1 + \epsilon)\mu(S_i), \Sigma(S_i)),$$

where $\mu(S_i)$ is a mean vector of the original state, S_i , and $\Sigma(S_i)$ is a covariance matrix. ϵ is a constant.

2. **Splitting step** ($n = 1, 2, \dots$)

- (a) Calculate the likelihood value for each triphone and classify them into two classes considering phone contexts.
- (b) Estimate the parameters of the two distributions using the standard ML estimation.
- (c) Test for convergence: stop if the partition does not change or if

$$\frac{L^{(n)} - L^{(n-1)}}{L^{(n-1)}} < \eta,$$

$$L^{(n)} = -\Gamma^{(n)}(S_{i_1}) \log |\Sigma^{(n)}(S_{i_1})| - \Gamma^{(n)}(S_{i_2}) \log |\Sigma^{(n)}(S_{i_2})|,$$

where η is experimentally set as a convergence threshold. $\Gamma^{(n)}(S_i) = \sum_t \gamma_t^{(n)}(S_i)$ is the expected frequency of transition from state S_i and t is the time frame. $\Sigma^{(n)}(S_i)$ is the covariance matrix of state S_i at the iteration step n and is a diagonal covariance in this paper. Note that this algorithm requires that one state has one Gaussian distribution.

- (d) Set $n = n + 1$ and go to 2.

Figure 2.5. Contextual splitting by using Chou's algorithm [7].

After contextual splitting, the total expected gain, which is explained in the next sub-section, is calculated. For temporal splitting, the ML-SSS algorithm creates one more state and connects it to the original state. The parameters of the two distributions are estimated by the forward-backward algorithm, and the total expected gain of temporal splitting is also calculated for the temporal split states. Since it is computationally expensive to re-estimate all of the parameters of a network at every splitting, only the parameters of the two candidate states are re-estimated by using probabilities weighted by the statistics of the target state [7]. Usually, the larger the split, the greater the likelihood it will be obtained. It is difficult to use it as a stop criterion. Next, the gains of both contextual and temporal splitting are calculated for all states. Finally, these expected gains are compared with each other and the split with the best gain among all states is selected.

For each triphone model, N_s is the total number of states and N_p is the maximum temporal length of states. These parameters must be given before starting the splitting. Basically, contextual splitting is stopped if there are no more than two phoneme contexts, and temporal splitting is stopped if there are no data longer than state sequences. However, it is difficult for these criteria to stop splitting; therefore, the number of states and the maximum number of states per triphone have been used as stop criteria. Nonetheless, it is still difficult to find the optimal values of these parameters. Accordingly, a sequence of experiments needs to be done to maximize the performance by changing parameters heuristically.

2.3.3 Gain function by ML-SSS algorithm

Next, we describe the total expected gains of splitting states with the ML-SSS algorithm. The total expected gain of contextual splitting for state S_i into two new states S_{i_1} and S_{i_2} is

$$G(S_i) = G_{\text{output}}(S_i) + G_{\text{trans}}(S_i), \quad (2.4)$$

where $G_{\text{output}}(S_i)$ is the expected gain of output probabilities and $G_{\text{trans}}(S_i)$ is the expected gain of transition probabilities;

$$G_{\text{output}}(S_i) = -\frac{1}{2}\{\Gamma(S_{i_1}) \log |\Sigma(S_{i_1})| + \Gamma(S_{i_2}) \log |\Sigma(S_{i_2})| - \Gamma(S_i) \log |\Sigma(S_i)|\}, \quad (2.5)$$

$$\begin{aligned} G_{\text{trans}}(S_i) &= \Xi(S_{i_1}, S_{i_1}) \log a_{i_1 i_1} + \Xi(S_{i_1}, S_{i_2}) \log a_{i_1 i_2} \\ &\quad + \Xi(S_{i_2}, S_{i_2}) \log a_{i_2 i_2} - \Xi(S_i, S_i) \log a_{ii} \\ &= \Xi(S_{i_1}, S_{i_1}) \log a_{i_1 i_1} + \{\Gamma(S_{i_1}) - \Xi(S_{i_1}, S_{i_1})\} \log(1 - a_{i_1 i_1}) \\ &\quad + \Xi(S_{i_2}, S_{i_2}) \log a_{i_2 i_2} - \Xi(S_i, S_i) \log a_{ii}, \end{aligned} \quad (2.6)$$

where $\Sigma(S_i)$ is the covariance matrix of the state i and $\Gamma(S_i) = \sum_{t=1}^T \gamma_t(S_i)$ is the expected frequency of transition from state S_i . $\gamma_t(S_i)$ is the probability of staying in S_i at the time t , $\Xi(S_i, S_j) = \sum_{t=1}^T \xi_t(S_i, S_j)$ is the expected frequency of transition from S_i to S_j , $\xi_t(S_i, S_j)$ is the probability of transition from S_i to S_j at t , and a_{ii} is the self-loop probability. In Eq. 2.6, $a_{i_1 i_2} = 1 - a_{i_1 i_1}$ and $\Xi(S_{i_1}, S_{i_2}) = \Gamma(S_{i_1}) - \Xi(S_{i_1}, S_{i_1})$ are used.

For contextual splitting, since the transition probabilities are not re-estimated to reduce the amount of calculation, the total expected gain related to only the observation distributions is calculated. Thus, $G_{\text{trans}}(S_i)$ is omitted for contextual splitting. For temporal splitting, the transition probabilities are considered because one transition probability is created after temporal splitting. Therefore, the splitting conditions $G_c^{(ML)}(S_i)$ for contextual splitting and $G_t^{(ML)}(S_i)$ for temporal splitting are

$$G_c^{(ML)}(S_i) = G_{\text{output}}(S_i), \quad (2.7)$$

$$G_t^{(ML)}(S_i) = G_{\text{output}}(S_i) + G_{\text{trans}}(S_i). \quad (2.8)$$

Equations (2.7) and (2.8) are calculated for each state, and the split with the maximum gain is selected.

2.4. Information Criteria for Model Selection

The topology training methods described in the previous section are based on the Maximum Likelihood criterion. Because of the nature of the ML estimation, the likelihood value for training data increases as the number of parameters increases. To overcome this problem, information criteria have been introduced for splitting and stop criteria [8, 9, 10, 11, 12].

S. Ikeda proposed a method using Akaike’s Information Criterion (AIC) to determine the topologies of context-independent models [8]. However, it is now clear that context-dependent models can improve performance more easily than other factors.

To create context-dependent models by using phonetic decision tree clustering, K. Shinoda and T. Watanabe introduced the Minimum Description Length (MDL) criterion [9][10]. After their first paper [9] was published, the method of the Bayesian Information Criterion (BIC) was proposed in [11][12], although these methods [9, 10, 11, 12] are essentially the same because the difference of two BIC values is the same as the difference of two MDL values.

The next few sections describe the three most popular information criteria in brief.

2.4.1 Akaike Information Criterion

The Akaike Information Criterion (AIC) [25] is the first criterion that was proposed to select stochastic models reasonably. The definition of this problem, “how to select models reasonably,” itself was first proposed by Hirotugu Akaike. Following his research, many criteria have been proposed, and have been applied to many research fields.

When a set of models $\{\theta^{(k)}|k = 1, \dots, K\}$ is given, the AIC for model k is

$$L_k^{(AIC)}(\mathbf{x}) = -2 \log P(\mathbf{x}|\hat{\theta}^{(k)}) + 2\alpha_k, \quad (2.9)$$

where $\mathbf{x} = \{x_1, \dots, x_{N_T}\}$ represents observation data, α_k is the number of free parameters, and $\hat{\theta}^{(k)}$ is the ML estimates of model k .

The AIC was derived from the fact that a maximum likelihood estimate of a regular model approaches a normal distribution when the number of samples is increasing. Therefore, the number of samples should be large enough for the number of parameters.

2.4.2 Bayesian Information Criterion

The Bayesian Information Criterion (BIC) [26] was defined by considering the asymptotic behavior of Bayes estimation.

$$L_k^{(BIC)}(\mathbf{x}) = -\log P(\mathbf{x}|\hat{\theta}^{(k)}) + \frac{\alpha_k}{2} \log N_T. \quad (2.10)$$

2.4.3 Minimum Description Length Criterion

The MDL criterion [27][28] was proposed to give the minimum coding length to describe models and data. Generally, the MDL criterion for model k is defined in the following:

$$L_k^{(MDL)}(\mathbf{x}) = -\log P(\mathbf{x}|\hat{\theta}^{(k)}) + \frac{\alpha_k}{2} \log N_T + \log K. \quad (2.11)$$

Figure 1.3 shows a conceptual curve of information criteria for model selection, especially BIC or MDL criterion. In these three popular criteria, the first term is the inverse sign of the log likelihood of the model. The second term and the following term are related to the number of parameters, or the number of samples, and work as a penalty to avoid over-fitting. Therefore, the model with the smallest information criterion is the best model.

2.5. Variational Bayesian Approach for Model Selection

The information criteria have been applied to many fields and usually work well. However, they require some assumptions, e.g., asymptotic normality, and theoretically it is difficult to evaluate complicated models like neural networks, or HMMs, that cannot satisfy such assumptions. The information criteria were derived by using asymptotic normality of maximum likelihood estimates. We assume that $p(x|w)$ is a model, x is training data, and w is a model parameter. We also assume that a maximum likelihood estimate exists and converges to w_0 . When n samples are given and $n \rightarrow \infty$, a random variable $\sqrt{n}(w - w_0)$ converges to a normal distribution, for which the mean is 0, and a covariance matrix is $I(w_0)^{-1}$, where $I(w_0)$ is the Fisher Information matrix. This is referred to as the asymptotic normality. It is necessary that a model is statistically regular. Moreover, this model should satisfy the following conditions: $\log p(x|w)$ is differentiable three times w.r.t. w . The Fisher Information matrix $I(w)$ can be defined and should be a positive definite matrix. Some complicated models, e.g., HMMs and neural networks, are not statistical regular models, and they cannot have asymptotic normality.

In the field of machine learning, the Variational Bayesian (VB) method was proposed by S. R. Waterhouse et al. [13] and H. Attias [14][15] to avoid over-fitting by ML estimation. This method is one of the approximate solutions for Bayesian learning.

Furthermore, a method to obtain the optimal model structure by the VB framework was proposed in order to avoid the local optimal problem for a mixture-of-experts model [29]. The VB approach can evaluate complicated models more exactly than information criteria and can be used for model selection.

Recently, some methods that include the VB approach have been proposed for speech recognition. Decision tree clustering with the VB method was proposed by S. Watanabe et al. [30], and Variational Bayesian GMMs were applied to speech recognition by F. Valente and C. Wellekens [31]. A VB general framework for speech recognition was proposed in [30]; however, their method of making HMM structures assumed the alignment is given and therefore did not use any latent variables. Furthermore, in both [30] and [31], their models did not consider any temporal structures.

2.5.1 Bayesian Learning

As described in Section 1.2.2, Bayesian learning considers that all unknown parameters are random variables. Each parameter can be represented by a distribution. The Bayesian approach tries to estimate *posterior predictive distribution* $p(\mathbf{x}|\mathbf{O}, m)$ for a new observation \mathbf{x} , that is, test data, defined in the following:

$$p(\mathbf{x}|\mathbf{O}, m) = \int p(\mathbf{x}|\Theta, m)p(\Theta|\mathbf{O}, m)d\Theta, \quad (2.12)$$

where \mathbf{O} is all of training samples, Θ is a set of parameters under a fixed model structure m where m is the complexity of a model (for example, the number of mixture components for a mixture model), $p(\mathbf{x}|\Theta, m)$ is a likelihood of \mathbf{x} , and $p(\Theta|\mathbf{O}, m)$ is a posterior distribution of Θ . Equation (2.12) represents an average of likelihood weighted by a posterior distribution. Therefore, the Bayesian approach can alleviate the over-fitting problem.

However, this approach requires estimation of posterior probabilities and calculations of integrals, and it is much too difficult to solve them directly. As approximated calculation methods, the Laplace approximation methods and Markov-chain Monte Carlo (MCMC) methods have been used. The Laplace approximation methods are based on the method that posterior probabilities are approximated by a Gaussian function. These methods assume infinite samples. The MCMC methods estimate posterior predictive distributions by sampling distributions under the Markov chain. These methods need a lot of calculations for sampling.

Recently, the Variational Bayesian approach has been proposed to solve these problems more efficiently [13][14][15]. This approach can estimate posterior distributions by analytical calculations using calculus of variations. This approximation is more efficient than Laplace approximation and produces non-trivial posteriors for any sample size.

2.5.2 Variational Bayesian approach

First, to estimate posterior probabilities, in the Variational Bayesian approach the log marginal likelihood with all random quantities marginalized is considered in the following:

$$\mathcal{L}(\mathbf{O}) = \log p(\mathbf{O}) = \log \sum_m \sum_Z \int p(\mathbf{O}, Z, \Theta, m) d\Theta, \quad (2.13)$$

where Z is a set of latent variables that are hidden variables. When a new distribution $q(Z, \Theta, m)$ is introduced, and Jensen's inequality is applied to $\mathcal{L}(\mathbf{O})$, and $\mathcal{L}(\mathbf{O})$ can be bounded by the following $\mathcal{F}[q]$:

$$\begin{aligned} \mathcal{L}(\mathbf{O}) &= \log \sum_m \sum_Z \int q(Z, \Theta, m) \frac{p(\mathbf{O}, Z, \Theta, m)}{q(Z, \Theta, m)} d\Theta \\ &= \log \left\langle \frac{p(\mathbf{O}, Z, \Theta, m)}{q(Z, \Theta, m)} \right\rangle_{q(Z, \Theta, m)} \\ &\geq \left\langle \log \frac{p(\mathbf{O}, Z, \Theta, m)}{q(Z, \Theta, m)} \right\rangle_{q(Z, \Theta, m)} \\ &= \sum_m \sum_Z \int q(\mathbf{O}, \Theta, m) \log \frac{p(\mathbf{O}, Z, \Theta, m)}{q(Z, \Theta, m)} d\Theta \\ &\equiv \mathcal{F}[q], \end{aligned} \quad (2.14)$$

where q is an approximation of the true posterior distribution and is referred to as the variational posterior distribution. Furthermore, $\langle f(x) \rangle_{p(x)}$ represents the expectation of $f(x)$ w.r.t. $p(x)$:

$$\langle f(x) \rangle_{p(x)} = \int f(x) p(x) dx. \quad (2.15)$$

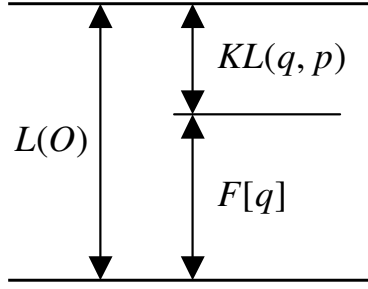


Figure 2.6. Principle of the Variational Bayesian approach.

The difference between $\mathcal{L}(\mathbf{O})$ and $\mathcal{F}[q]$ is represented by the Kullback-Leibler divergence in the following.

$$\begin{aligned}
\mathcal{L}(\mathbf{O}) - \mathcal{F}[q] &= \sum_m \sum_Z \int q(Z, \Theta, m) \log p(\mathbf{O}) d\Theta \\
&\quad - \sum_m \sum_Z \int q(Z, \Theta, m) \log \frac{p(\mathbf{O}, Z, \Theta, m)}{q(Z, \Theta, m)} d\Theta \\
&= \sum_m \sum_Z \int q(Z, \Theta, m) \log \frac{q(Z, \Theta, m)}{p(Z, \Theta, m|\mathbf{O})} d\Theta \\
&\equiv KL(q(Z, \Theta, m), p(Z, \Theta, m|\mathbf{O})). \tag{2.16}
\end{aligned}$$

Therefore, the following relation is obtained.

$$\mathcal{L}(\mathbf{O}) = \mathcal{F}[q] + KL(q(Z, \Theta, m|\mathbf{O}), p(Z, \Theta, m|\mathbf{O})). \tag{2.17}$$

As Figure 2.6 shows, since $\mathcal{L}(\mathbf{O})$ is a constant under a given \mathbf{O} , maximizing $\mathcal{F}[q]$ w.r.t. q is equivalent to minimizing the Kullback-Leibler divergence between q and the true posterior distribution. The q becomes the best approximation of the true posterior when the maximum of $\mathcal{F}[q]$ is obtained.

We assume that the joint distribution of all random quantities can be factorized in the following:

$$p(\mathbf{O}, Z, \Theta, m) = P(m)p(\mathbf{O}, Z|m) \prod_{i=1}^I p(\theta_i|m), \tag{2.18}$$

where $\Theta = \{\theta_i\}_{i=1}^I$. In the same manner, the variational posterior can be also factorized.

$$q(Z, \Theta, m) = q(m)q(Z|m) \prod_{i=1}^I q(\theta_i|m). \tag{2.19}$$

$\mathcal{F}[q]$ can be derived in the following.

$$\mathcal{F}[q] = \sum_m q(m) \left\{ \left\langle \log \frac{p(\mathbf{O}, Z|\theta, m)}{q(Z|m)} \right\rangle_{q(Z|m), q(\theta|m)} + \sum_{i=1}^I \left\langle \log \frac{p(\theta_i|m)}{q(\theta_i|m)} \right\rangle_{q(\theta_i|m)} + \log \frac{P(m)}{q(m)} \right\}. \quad (2.20)$$

The optimal posteriors of Z and θ_i can be derived by the Lagrange multiplier method. The following $J[q(Z|m)]$ is defined for $q(Z|m)$ in Eq. (2.19),

$$J[q(Z|m)] = \mathcal{F}[q(Z|m)] + \lambda \left\{ \sum_Z q(Z|m) - 1 \right\},$$

$$\mathcal{F}[q(Z|m)] = \sum_m q(m) \left\langle \frac{p(\mathbf{O}, Z|\Theta, m)}{q(Z|m)} \right\rangle_{q(Z|m), q(\Theta|m)}.$$

The following partial differentials are set to zero:

$$\frac{\partial J}{\partial q(Z|m)} = 0,$$

$$\frac{\partial J}{\partial \lambda} = 0.$$

The optimal $q(Z|m)$ can be derived in the following:

$$q(Z|m) = C \exp \langle \log p(\mathbf{O}, Z|\Theta, m) \rangle_{q(\Theta|m)}, \quad (2.21)$$

where C is a normalization constant. In the same manner, the optimal $q(\theta_i|m)$ in Eq. (2.19) is

$$q(\theta_i|m) = C_i \exp \langle \log p(\mathbf{O}, Z|\theta, m) \rangle_{q(Z|m), q(\theta_{-i}|m)}, \quad (2.22)$$

where C_i is a normalization constant and θ_{-i} is a set of θ_i except for θ_i .

Since Eqs. (2.21) and (2.22) are dependent on each other, they can be derived iteratively. Finally, the Variational Bayesian EM algorithm can be defined as follows:

[Variational Bayesian EM Algorithm]

1. Set initial distributions $q(\Theta|m) = \prod_{i=1}^I p(\theta_i|m)$, and $t = 0$.
2. Repeat the following process until they are convergent.

(a) **E-Step**

$$q(Z|m)^{(t+1)} = C \exp \langle \log p(\mathbf{O}, Z|\Theta, m) \rangle_{q(\Theta|m)^{(t)}}$$

(b) **M-Step**

For $i = 1, \dots, I$,

$$q(\theta_i|m)^{(t+1)} = C_i \exp \langle \log p(\mathbf{O}, Z|\theta, m) \rangle_{q(Z|m)^{(t+1)}, q(\theta_{-i}|m)^{(t)}},$$

(c) $t = t + 1$.

Using estimated posterior probabilities, $\mathcal{F}[q]$ can thus be calculated. For model selection, it is necessary to select the model that maximizes $\mathcal{F}[q]$.

2.6. Language Modeling

In Eq. (2.3), the language model can be simply represented by $P(\mathbf{W})$, although it is difficult to define the probability of a word string since many expressions of sentences usually exist to represent the same meaning. Spontaneous speech is so flexible that expressions proper to spoken language cannot be constrained by grammar.

2.6.1 Word n-gram model

The first experimental results in large-vocabulary speech recognition were obtained in 1976 [32]. The vocabulary size included 1,000 of the most frequent words. From this work, as a kind of simple language modeling, word n-gram models are widely used. If we assume that \mathbf{W} is a word sequence, $\{w_1, w_2, \dots, w_e\}$, $P(\mathbf{W})$ may be computed in the following:

$$P(\mathbf{W}) = P(w_1 w_2 \dots w_e) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_N|w_1 w_2 \dots w_e). \quad (2.23)$$

However, it is impossible to estimate $P(w_i|w_1 w_2 \dots w_{i-1})$ for all word sequences, and it is therefore easier to use word n-gram models. This probability is approximated by the probability of an N -word sequence.

$$P(w_i|w_1 w_2 \dots w_{i-1}) \approx P(w_i|w_{i-N+1} \dots w_{i-1}). \quad (2.24)$$

In practice, $N = 3$ is usually enough due to the robustness of parameter estimation. Here, $P(\mathbf{W})$ is approximated by trigram models in the following:

$$P(\mathbf{W}) \approx P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \dots P(w_N|w_{N-2}, w_{N-1}). \quad (2.25)$$

2.6.2 Trigger model

Although word n-gram models are convenient, they cannot represent all language constraints. One promising model capitalizes on the information present in the document history. To extract information from the document history, a “trigger pair” can be used. If a word, or a word string \mathbf{W}_A , is significantly correlated with another word string \mathbf{W}_B , $(\mathbf{W}_A \rightarrow \mathbf{W}_B)$ is considered a trigger pair. If \mathbf{W}_B is included in the document history \mathbf{W} ,

the probability $P(W_A)$ can be represented by using a trigger pair model is given in the following:

$$P(W_A) \approx P(W_A | W_B \in W). \quad (2.26)$$

In [17], R. Lau et al. proposed trigger modeling based on the Maximum Entropy principle. This model is combined with the n-gram model. The trigger model can only represent co-occurrences between trigger pairs; Some of such co-occurrences do not have any syntactic relations. To constrain trigger pairs, S. Zhang et al. used trigger pairs extracted from parse trees [18]. This model is referred to as the ‘‘Linkgram’’ model. However, these constraints represented by word pairs are still weak for global structures of sentences.

2.6.3 Structured Language Model (SLM)

To represent more powerful constraints, C. Chelba and F. Jelinek proposed a language model using a stochastic parser [19][20], which is called the Structured Language Model (SLM). Here, a stochastic model is developed for parsing, and this model itself is used as a language model for speech recognition, that is, parsing scores are considered as scores of a language model. This model parses by bottom-up and left-to-right operation and operates using three modules: (1) WORD-PREDICTOR predicts the next word from previous word sequences and parse subtrees, (2) TAGGER predicts POS tags of next words from word sequences, parse subtrees and predicted next words, and (3) CONSTRUCTOR repeatedly generates a parse tree from subtrees. These modules use conditional probabilities and those probabilities are used for lattice rescoring.

The probability $P(W, T)$ of a word sequence W and a complete parse T can be calculated as follows:

$$P(W, T) = \prod_{k=1}^{n+1} \left[P(w_k | W_{k-1} T_{k-1}) P(t_k | W_{k-1} T_{k-1}, w_k) P(T_{k-1}^k | W_{k-1} T_{k-1}, w_k, t_k) \right], \quad (2.27)$$

$$P(T_{k-1}^k | W_{k-1} T_{k-1}, w_k, t_k) = \prod_{i=1}^{N_k} P(p_i^k | W_{k-1} T_{k-1}, w_k, t_k, p_1^k \dots p_{i-1}^k), \quad (2.28)$$

where $W_{k-1} T_{k-1}$ is the word-parse $(k - 1)$ -prefix, w_k is the word predicted by WORD-PREDICTOR, t_k is the tag assigned to w_k by the TAGGER, and T_{k-1}^k is the incremental

parse structure that generates $T_k = T_{k-1} || T_{k-1}^k$ when attached to T_{k-1} . It is the parse structure built on top of T_{k-1} and the newly predicted word w_k . The “||” notation stands for concatenation, $N_k - 1$ is the number of operations the CONSTRUCTOR executes at position k of the input string before passing control to the WORD-PREDICTOR, and p_k denotes the i th CONSTRUCTOR action carried out at position k in the word string.

Furthermore, some researchers have recently proposed different types of structured language models. An SLM based on tree-structured history has been proposed in [21], and a probabilistic generalized LR parsing is also used as a language model [22].

2.7. Summary

This chapter described the outline of automatic speech recognition, acoustic modeling, and language modeling. Both topology training and criteria of model selection were explained, especially for acoustic modeling. We also described related works in brief for each topic. The next chapter onward, our proposed methods will be presented and discussed, which are the MDL-based SSS algorithm in Chapter 3, the VB-based SSS algorithm in Chapter 4, and language modeling using patterns extracted from parse trees in Chapter 5.

Chapter 3

Successive State Splitting Algorithm Based on Minimum Description Length Criterion

3.1. Introduction

As we described in Section 2.3, for topology training, the Decision Tree Clustering[5] is widely used. While this method can only create contextual variations, the Successive State Splitting (SSS) Algorithm can create both contextual and temporal variations. These method use the Maximum Likelihood (ML) criterion to choose a model. However, owing to the nature of the ML estimation, the likelihood value for training data increases as the number of parameters increases. Consequently, it is impossible to stop state splitting using only the ML criterion.

Therefore, many researchers have used information criteria to avoid this problem as we described in Section 2.4. The information criterion is used as both splitting and stop criteria. As a splitting criterion, the information criterion is calculated for split states, and the state that provides the best improvement of the information criterion in all states is selected. As a stop criterion, splitting is stopped when there is no state that can improve the information criterion.

In this chapter, we propose the SSS algorithm based on the MDL criterion as the splitting and stop criteria. This algorithm is hereinafter referred to as “the MDL-SSS algorithm.” The MDL criterion was successfully introduced in phonetic decision tree

clustering as a criterion of contextual clustering[10]. In this chapter, the MDL criterion is extensively used as the criterion for both contextual and temporal splitting in the ML-SSS algorithm. We define new gain functions based on the MDL criterion.

The AIC may be used as the criteria to make context-dependent models in the same manner as [8]. The second term of the MDL criterion is dependent on both the amount of data and the number of parameters. On the other hand, the AIC only considers the number of parameters. For acoustic models, it is well known that recognition performance is dependent on the amount of training data. For this reason, the MDL criterion is more suitable for speech recognition. Therefore, we focus only on the MDL criterion in this paper.

In Section 2.3.2, we have explained the ML-SSS algorithm and the stop-splitting problem. Also, the MDL criterion has been described in Section 2.4.3. We define the MDL-SSS algorithm in Section 3.2. In Section 3.3, we evaluate the performance of the MDL-SSS algorithm and describe the results for an ATR travel arrangement task and the results for lecture speech as an example of more spontaneous speech. We summarize the results in Section 3.4.

3.2. SSS Algorithm Using MDL Criterion

Next, we introduce the MDL criterion to the ML-SSS algorithm. In this section, we define the MDL-SSS algorithm, which uses the MDL criterion instead of the ML criterion as the splitting criterion for the ML-SSS algorithm.

3.2.1 Flow of MDL-SSS algorithm

Figure 3.1 shows the flow of the MDL-SSS algorithm. The differences in the MDL values for both contextual and temporal splitting are calculated for each state, and the split with the smallest difference value is chosen. Splitting is finished when there is no state that can be split and reduce the criterion by splitting. The total number of states and the maximum number of states per triphone are not required as stop criteria.

The MDL criterion is driven theoretically and guarantees the prevention of the over-fitting problem. On the other hand, to stop splitting in the ML-SSS algorithm, some thresholds which can evaluate convergence of likelihood values can be used as

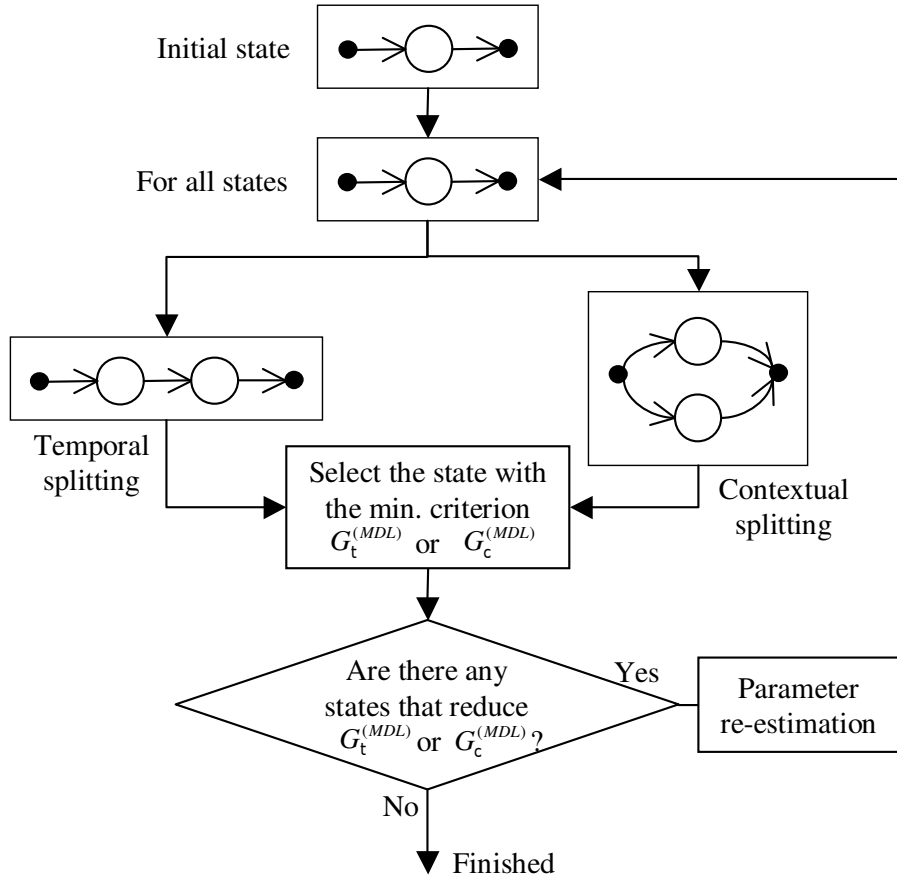


Figure 3.1. Flow chart of MDL-SSS algorithm.

one of the stop criteria instead of the number of states. However, such thresholds need to be set heuristically, and there is no guarantee that they will avoid the over-fitting problem.

3.2.2 Gain function by MDL-SSS

First, we redefine the MDL criterion for the model k here.

$$L_k^{(MDL)}(\mathbf{x}) = -\log P(\mathbf{x}|\hat{\theta}^{(k)}) + \frac{\alpha_k}{2} \log \Gamma(S) + \log K, \quad (3.1)$$

where α_k is the number of parameters for the model k , and $\Gamma(S) = \sum_{i=1}^{N_s} \Gamma(S_i)$ represents the expected frequency of the number of training samples for all states included in the model k . As the same manner as Section 2.3.3, we assume that a state S_i in the model

k is split into two states S_{i_1}, S_{i_2} . The MDL criterion is changed as follows.

$$L_{k'}^{(MDL)}(\mathbf{x}) = -\log P(\mathbf{x}|\hat{\theta}^{(k')}) + \frac{\alpha'_{k'}}{2} \log \Gamma(S) + \log K, \quad (3.2)$$

where $\alpha'_{k'}$ is the number of parameters after splitting. The first term of the MDL criterion is a negative value of likelihood. Therefore, the difference of the first term between Eqs. (3.1) and (3.2) is the negative value of the gain of likelihood, Eq. (2.7) or (2.8). Therefore, we can define the criteria for contextual splitting and temporal splitting, $G_c^{(MDL)}$ and $G_t^{(MDL)}$, respectively, as follows:

$$G_c^{(MDL)}(S_i) = -G_c^{(ML)}(S_i) + \frac{\alpha'_c - \alpha_c}{2} \log \Gamma(S), \quad (3.3)$$

$$G_t^{(MDL)}(S_i) = -G_t^{(ML)}(S_i) + \left\{ \frac{\alpha'_t}{2} \log \Gamma'(S) - \frac{\alpha_t}{2} \log \Gamma(S) \right\}, \quad (3.4)$$

where the suffixes, c and t, mean that the values are related to the contextual splitting, and the temporal splitting, respectively. α' represents the number of parameters after splitting. $\Gamma'(S)$ is the value after temporal splitting. Equation (3.4) compensates the total number of samples, $\Gamma(S)$, because segments that are shorter than the lengths of state sequences are discarded. $\Gamma(S)$ will be decreased to $\Gamma'(S)$ if a temporal split is selected.

In the previous works[9, 10, 11, 12], they introduced the scaling factors of the second terms to control the value of criterion for their contextual clustering. We introduce the scaling factors of the second terms, C_c and C_t , both for contextual and temporal splitting.

$$G_c^{(MDL)}(S_i) = -G_c^{(ML)}(S_i) + C_c \frac{\alpha'_c - \alpha_c}{2} \log \Gamma(S), \quad (3.5)$$

$$G_t^{(MDL)}(S_i) = -G_t^{(ML)}(S_i) + C_t \left\{ \frac{\alpha'_t}{2} \log \Gamma'(S) - \frac{\alpha_t}{2} \log \Gamma(S) \right\}. \quad (3.6)$$

The number of parameters before and after contextual splitting are $\alpha_c = 2KM$ and $\alpha'_c = 2K(M + 1)$, where the order of features is K , the total number of states is M , and each state has one Gaussian distribution with a diagonal covariance matrix. For temporal splitting, we suppose that transition probabilities do not depend on both mean vectors and covariances of Gaussian mixtures. Each state has one Gaussian distribution and one transition probability. Therefore, the number of parameters before and after temporal splitting are $\alpha_t = (2K + 1)M$ and $\alpha'_t = (2K + 1)(M + 1)$, respectively.

The scaling factors, C_c and C_t , are not derived from the original MDL criterion. We experimentally found that it is difficult to stop splitting without these factors. This problem can be considered to be caused by the approximation of the likelihood values of temporal split states as we described in Section 2.3.2. In [10] and [12], a scaling factor for contextual splitting was also introduced and experimentally found to be effective.

Accordingly, Eqs. (3.5) and (3.6) are rewritten as follows:

$$G_c^{(MDL)}(S_i) = -G_c^{(ML)}(S_i) + C_c K \log \Gamma(S), \quad (3.7)$$

$$G_t^{(MDL)}(S_i) = -G_t^{(ML)}(S_i) + C_t \frac{(2K+1)}{2} \{(M+1) \log \Gamma'(S) - M \log \Gamma(S)\}. \quad (3.8)$$

The MDL-SSS algorithm selects the state with the smallest $G_c^{(MDL)}$ or $G_t^{(MDL)}$, and stops splitting when $G_c^{(MDL)} > 0$ and $G_t^{(MDL)} > 0$ for all states.

Finally, we give the flow of the MDL-SSS algorithm in Fig. 3.2.

MDL-SSS Algorithm:

1. Initialization.
2. For all states: $\{S_1, S_2, \dots, S_i, \dots\}$
 - (a) Contextual splitting:
 - i. Create two new states by the contextual splitting of S_i .
 - ii. Calculate the criterion, $G_c^{(MDL)}(S_i)$.
 - (b) Temporal splitting:
 - i. Create two new states by the temporal splitting of S_i .
 - ii. Calculate the criterion, $G_t^{(MDL)}(S_i)$.
 - (c) Select the contextual or temporal splitting with a smaller criterion.
3. Select the state which obtains the smallest criterion among all states.
4. If no state can obtain less than zero in the criterion, then stop. Otherwise, re-estimate parameters and go to 2.

Figure 3.2. Overview of MDL-SSS algorithm.

3.3. Experiments

3.3.1 Conditions

Experiments were carried out using the proposed acoustic models for Japanese travel dialogues in “The Travel Arrangement Task (TRA)” of the ATR spontaneous speech database [33]. This corpus consists of role-playing pseudo-dialogs between a hotel clerk and a customer about room reservations, cancellation, trouble-shooting, etc.

For the acoustic training set, we used the TRA database and 503 phonetically balanced sentences (BLA) read by the same speakers of the TRA. These data include 407 speakers. TRA dialog data include about 5 hours of speech and BLA data include about 25 hours of speech. For the test set, we used one side of 42 dialogues with 4,990 words that were not included in the training set and that included 42 speakers.

For analysis conditions, the frame length was 20 ms and the frame shift was 10 ms. 12 order MFCC, 12 order Δ MFCC, and Δ log power were used as feature parameters. The cepstrum mean subtraction was applied to each utterance. We used 26 kinds of phonemes and one silence. A silence model with three states was built separately from the phoneme models. Three states were used as the initial model for each phoneme. One Gaussian distribution for each state was used during topology training. For both the ML-SSS and MDL-SSS algorithms, phoneme alignments were not changed during topology training.

In Section 3.3.2, we used speaker-independent models with one Gaussian distribution per state. These models are unable to produce high performance. Therefore, after we obtained the topology, we increased the number of mixtures and re-estimated the parameters of the HMMs. The final models were gender-dependent models with five Gaussian mixtures for each state. These models were used in the experiments described in Section 3.3.3. In these models, each gender model had the same topology as the other gender model. We have compared them to GD models in which topologies were dependent on gender in the preliminary experiments. The GD models with gender-independent topologies obtained better performance than the GD models with gender-dependent topologies. Therefore, we used the GD models with the same topologies.

For the language training set, we used 7,195 one-side dialogues which included 1.6×10^6 words. Multi-class composite bigram models [34] with 700 classes and 5,216

composite words were used as the language models. The perplexity for the evaluation data was about 21. The vocabulary size in the set was 27,398.

The number of mixture distributions was increased after splitting. The topology training can obtain approximate optimal topologies represented by one Gaussian distribution for each state. Therefore, first, we evaluated speaker-independent and single Gaussian models by using a small lexicon including 5,100 words. Second, we used gender-dependent and five-Gaussian models by using the full lexicon including 27,398 words.

3.3.2 Comparison of gender-independent models with single Gaussian

We initially investigated the performance by gender-independent models with a single Gaussian distribution to confirm the adequacy of the model topologies obtained by our proposed method. In this section, the lexicon had 5,100 words including the words in the evaluation data. Figure 3.3 shows word accuracy rates comparing the performance of the ML-SSS and MDL-SSS algorithms. This figure includes results for the ML-SSS algorithm limited by the maximum number of states for each triphone model, $N_p = 3, 4, 5$. The models limited by $N_p = 3$ were produced by only contextual splitting because all initial models had three states in these experiments. The performance for models with $N_p = 3$ is worse than that for models with $N_p = 4$ but better than that for models with $N_p = 5$. This suggests that the temporal splitting is effective but should be controlled for each phoneme by some suitable criterion.

The MDL-SSS algorithm with $C_c = 1, C_t = 10, 40$, and $C_c = 2, C_t = 10, 20$ obtained almost the same performance as the ML-SSS algorithm. For the ML-SSS algorithm, $N_s = 2, 500$ and $N_p = 4$ showed the best performance.

To achieve a balance between contextual splitting and temporal splitting, the optimal temporal scaling factor, C_t , was much larger than the contextual scaling factor, C_c . This is because the gain by temporal splitting is evaluated as being smaller in the MDL criterion and the likelihood value is also an approximated one as described in [7].

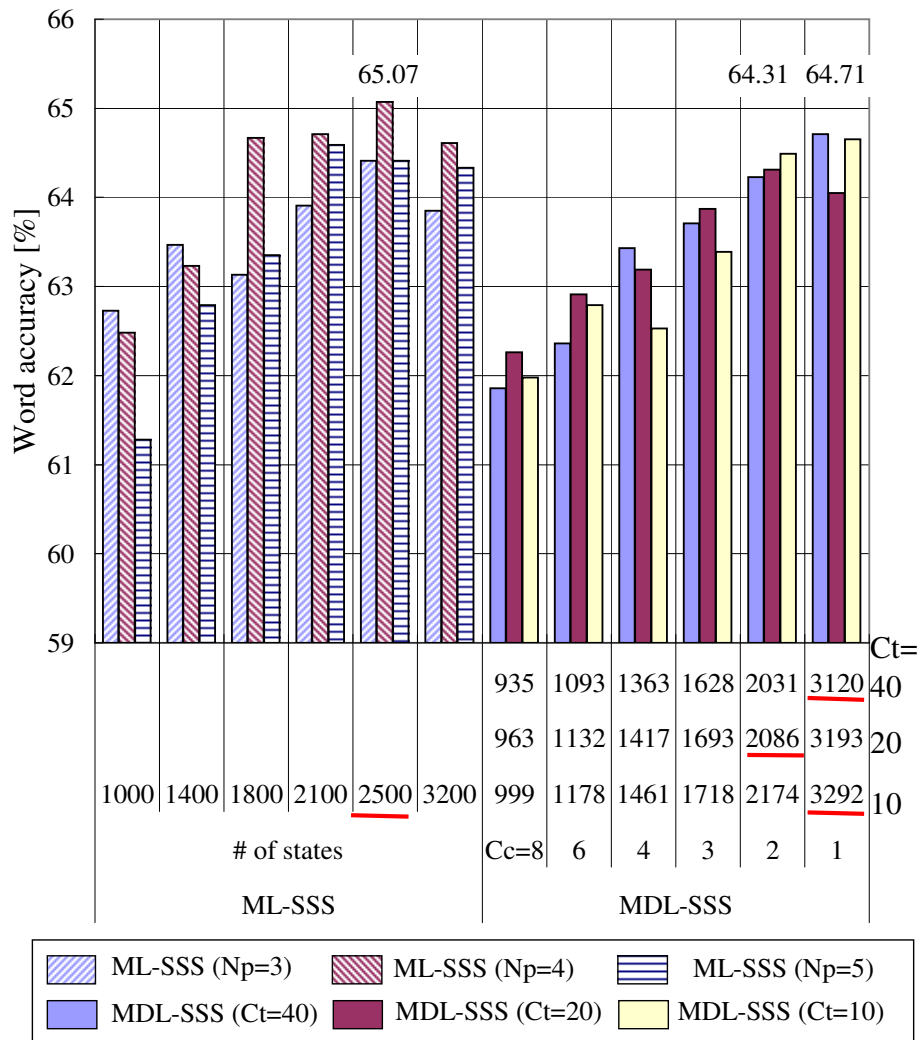


Figure 3.3. Word accuracy for GI models with one Gaussian distribution per state trained by using TRA and BLA.

3.3.3 Comparison of gender-dependent models with five Gaussians

In this section, we used gender-dependent models with five Gaussian mixtures. Figure 3.4 shows the word accuracy rates of these models.

For the MDL-SSS algorithm, $C_c = 2$ and $C_t = 20$ performed the best. Almost the best model was obtained by almost the same scaling factors as described in the previous section. On the other hand, for the ML-SSS algorithm, $N_s = 1,400$ and $N_p = 4$ performed the best. The total number of states of the best model was different from that of the best model with one Gaussian distribution per state. Therefore, for the ML-SSS algorithm, N_s needs to be carefully adjusted according to experiments to find the best model. In contrast to the ML-SSS algorithm, the MDL-SSS algorithm can automatically obtain the best performance with the same parameters.

We further compared the models of the MDL-SSS algorithm and the ML-SSS algorithm in detail. Figure 3.5 shows the number of states for each phoneme model. Most of the phoneme models by the MDL-SSS algorithm have larger numbers of states than the models by the ML-SSS algorithm. Figure 3.6 shows the maximum length of states for each phoneme extracted from both the “ML-SSS (1,400 states, $N_p=4$)” whose number of states per triphone were set to a limit of four states, and the “MDL-SSS ($C_c = 2$, $C_t = 20$, 2,086 states).” All phoneme models by the ML-SSS algorithm had the same maximum length of states as the limit number of states. On the other hand, each phoneme model by the MDL-SSS algorithm had a different maximum length of states. Although some phoneme models by the MDL-SSS algorithm have larger numbers of states than the models by the ML-SSS algorithm, they have shorter maximum lengths of states. These two figures suggest that more adequate path lengths are selected for each allophone by using the MDL-SSS algorithm. Although the ML-SSS algorithm can limit the number of states for each phoneme, it is difficult to find the optimal set of the maximum number of states.

3.3.4 Effectiveness for different amounts of training data

To confirm whether scaling factors in the MDL-SSS algorithm are dependent on the amount of training data, we examined the performance of acoustic models for a smaller amount of training data by using only the TRA data. The experimental conditions were the same as described in Section 3.3.3. Figure 3.7 shows word accuracy for

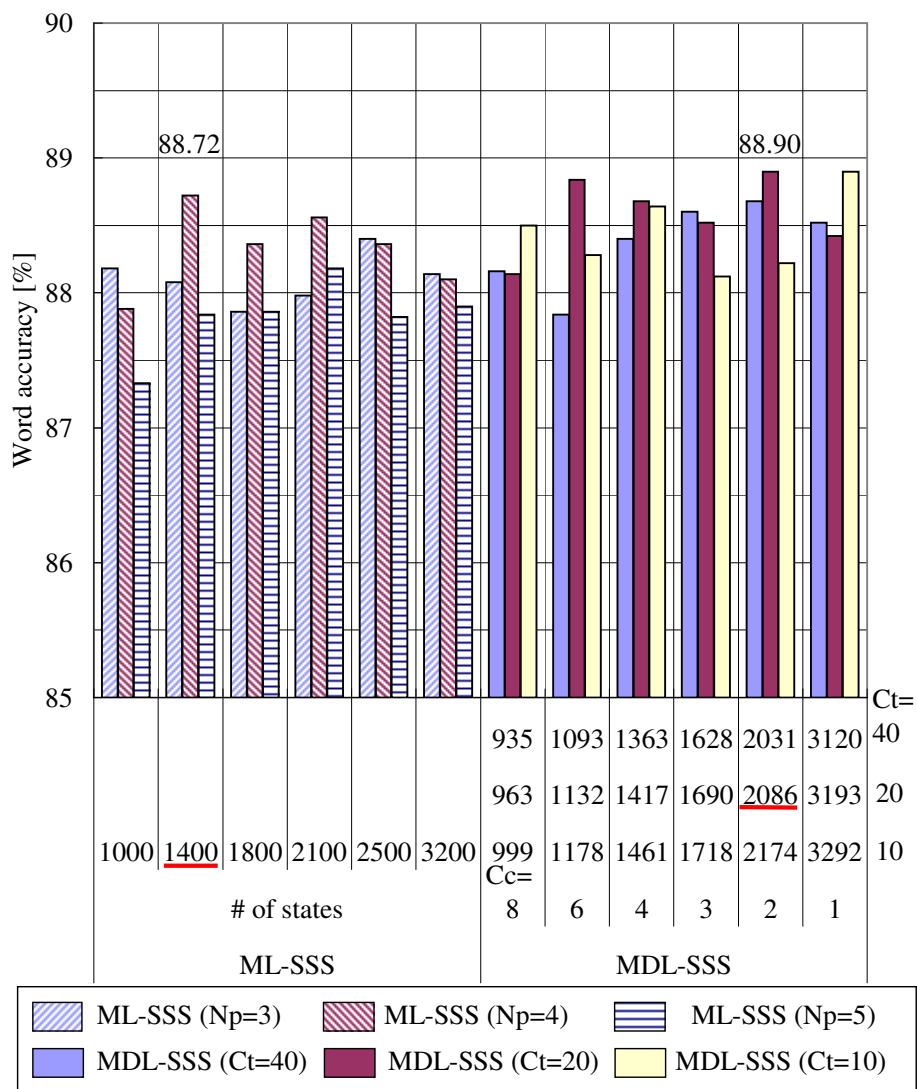


Figure 3.4. Word accuracy for GD models with five Gaussian mixtures trained by using TRA and BLA.

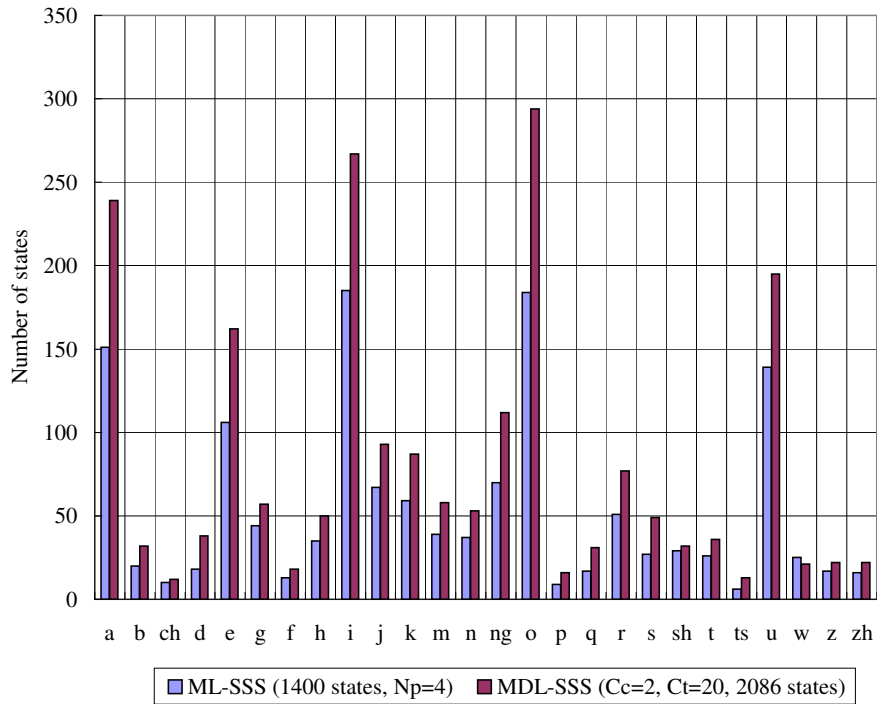


Figure 3.5. The number of states for each phoneme.

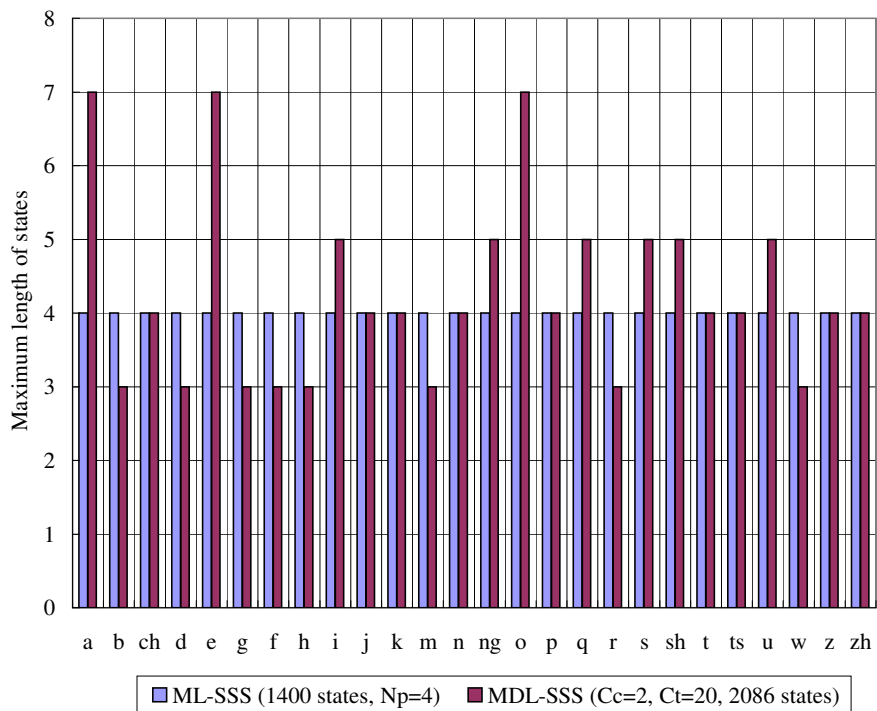


Figure 3.6. The maximum length of states for each phoneme.

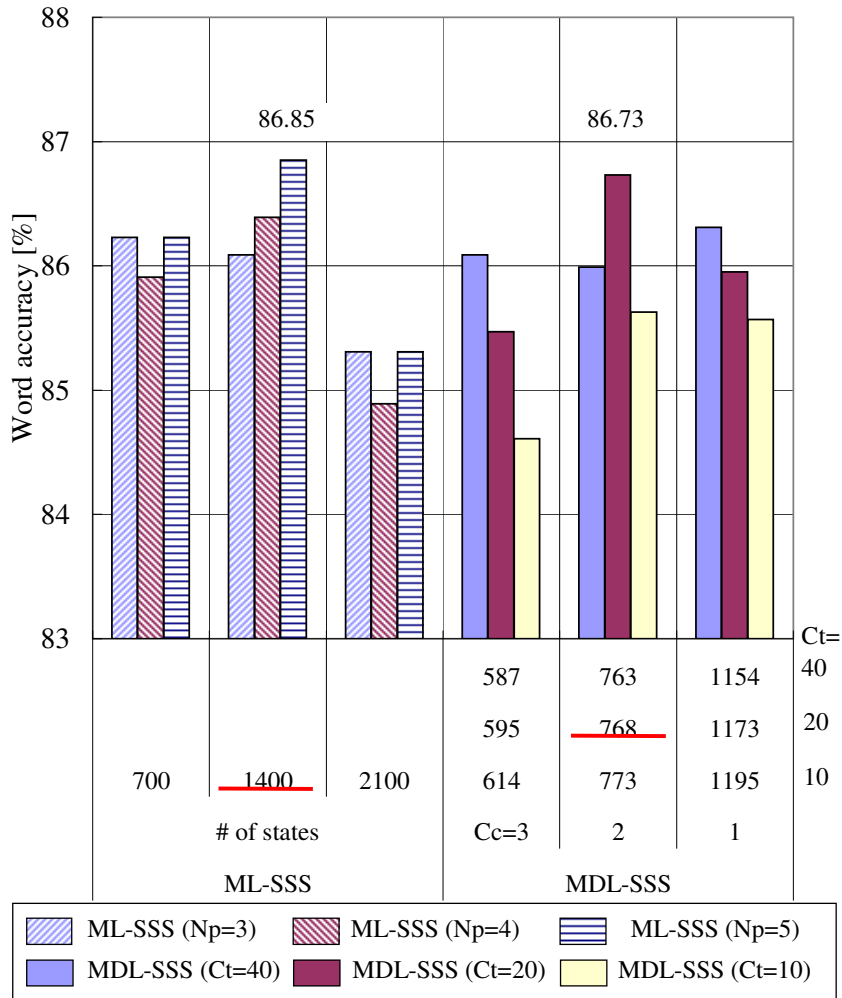


Figure 3.7. Word accuracy for models trained by using TRA data.

models generated by the ML-SSS algorithm or the MDL-SSS algorithm. The MDL-SSS algorithm obtained the best performance with $C_c = 2$ and $C_t = 20$. Therefore, scaling factors for both contextual and temporal splitting can be set robustly and thus the MDL-SSS algorithm can make appropriate HMM topologies more easily than the ML-SSS algorithm.

3.3.5 Evaluation using lecture speech

To confirm the optimal values of the scaling factors in our proposed method, we also evaluated our method by using the Japanese lecture speech corpus, “The Corpus of Spontaneous Japanese (CSJ)”[35]. This corpus mainly includes monologues, such as presentations at conferences. Therefore, the CSJ corpus is more spontaneous than the TRA corpus.

The training data for the acoustic models include 200 lectures (about 34 hours) by male speakers. The analysis conditions were the same as described in Section 3.3.1. The number of mixtures for each state was 10. The size of the lexicon distributed with the CSJ database was 19 K words. We used multi-class composite bigram models with 700 classes and 3,466 composite words trained by using these 200 lectures. Perplexity was about 137. For evaluation, we used four male speakers, “A01M0007”, “A01M0035”, “A01M0074”, and “A05M0031.” The average number of words included in each speaker data was 4,388.

Figure 3.8 shows the average word accuracy rates for the four speakers. For the MDL-SSS algorithm, performance is almost saturated by the model with $C_c = 2$ and $C_t = 20$. These factors are almost the same as those described in Sections 3.3.2, 3.3.3, and 3.3.4. Therefore, the trend of the results is similar to that of the TRA task. This shows that the MDL-SSS algorithm can automatically stop splitting and obtain almost the best performance by around $C_c = 2$ and $C_t = 20$ for other tasks.

Figure 3.9 shows the number of states for each phoneme model extracted from both “ML-SSS (2,178 states, $N_p=4$)” and “MDL-SSS ($C_c = 2, C_t = 20, 2,178$ states).” These models have almost the same number of states for each phoneme. Figure 3.10 shows the maximum number of states for each phoneme. Although each model by the ML-SSS algorithm had a different maximum number of states, the model by the MDL-SSS algorithm had more variety in the maximum number of states. Especially, the phoneme /e/ had a long triphone /silence - e - e/ with eight states. This triphone appears in the beginning of some Japanese filler words, for example, “eeto,” that are very often uttered in spontaneous speech.

Compared to the TRA and BLA corpora, the maximum number of states per phoneme in the CSJ corpus is smaller than that in the TRA corpus. Table 3.1 shows the average speaking rate for each training database. These speaking rates were calculated from the forced alignments of the training data. It shows that the distribution of speaking rates

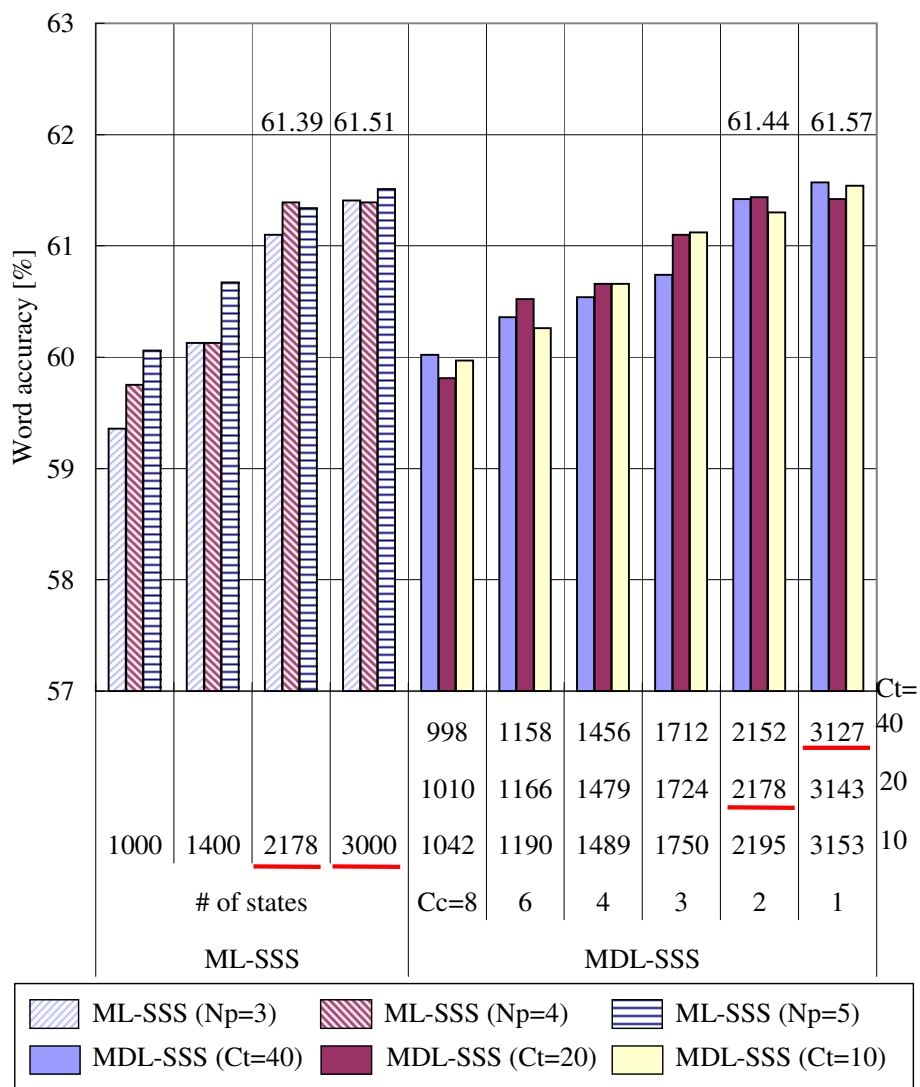


Figure 3.8. Word accuracy for the CSJ corpus.

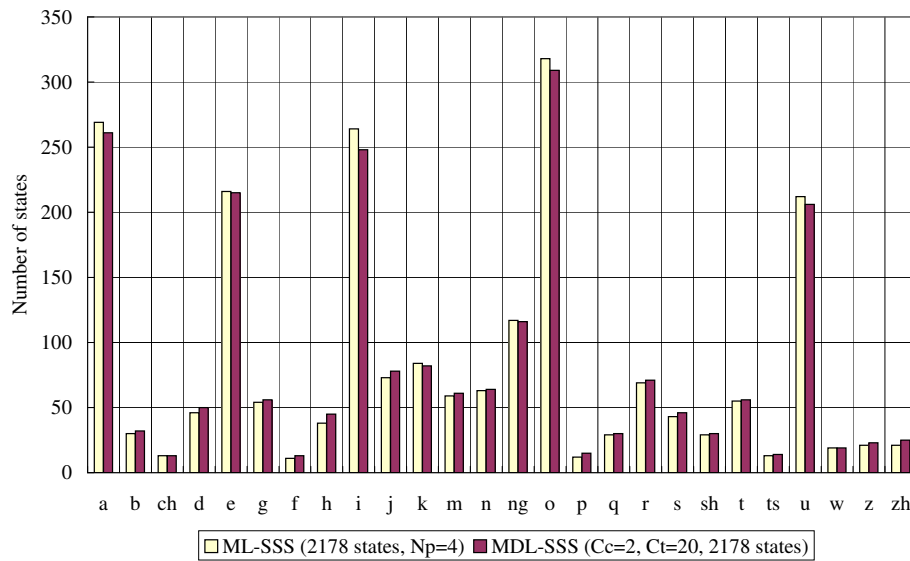


Figure 3.9. The number of states for each phoneme for the CSJ corpus.

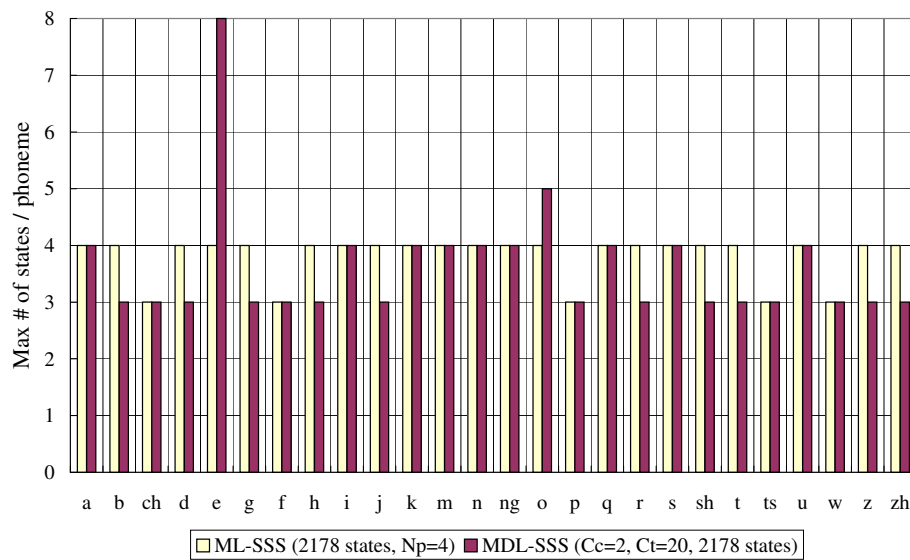


Figure 3.10. The maximum number of states per phoneme for the CSJ corpus.

Table 3.1. Average speaking rate for each training data

Database	Average [mora/sec]	Standard deviation
CSJ (male)	8.27	2.12
TRA+BLA (male)	8.22	1.03
TRA+BLA (female)	7.76	0.94
TRA (male)	8.46	1.47
TRA (female)	7.96	1.25
BLA (male)	8.15	0.83
BLA (female)	7.69	0.80

in the CSJ corpus is faster and broader than that of the TRA and BLA corpora. The standard deviation of the CSJ corpus is larger than that of the TRA and BLA corpora. Therefore, the CSJ corpus has much more variation in speaking rates. It shows that the CSJ corpus is a more spontaneous speech database than the TRA corpus. Furthermore, these results show that the MDL-SSS algorithm can reflect the properties of a database in HMM topologies.

3.4. Summary

We proposed a new method for automatically creating non-uniform, context-dependent HMM topologies using the Successive State Splitting algorithm based on the MDL criterion. The conventional methods of topology training are based on the ML criterion and require the total number of states as the stop criterion. The ML-SSS algorithm offers the advantage that it includes both contextual and temporal splitting, but it is also based on the ML criterion and requires stop criteria.

We introduced the MDL criterion to the ML-SSS algorithm in order to select suitable models automatically. Experimental results show that the MDL criterion can stop both contextual and temporal state splitting by the SSS algorithm. The best model can be obtained only by changing the number of states in the ML-SSS algorithm. However, the MDL-SSS algorithm yields almost the best performance by almost the same scaling factors in spite of the amount of training data and kinds of tasks without changing

the number of states. These scaling factors seem to be relatively consistent within the tasks.

For the ML-SSS algorithm, there is no guarantee that the best model can be obtained by using the number of states and the maximum number of temporal states per phoneme as a stop criterion. They need to be adjusted carefully according to experiments. On the other hand, almost the same scaling factors in the MDL-SSS can generate almost the best model. Lots of experiments to find a best model will no longer be necessary. Additionally, both the maximum number of states and the maximum number of temporal states per phoneme can be added to stop criteria in the MDL-SSS algorithm.

Chapter 4

Variational Bayesian Approach for the SSS Algorithm

4.1. Introduction

In the previous chapter, we have proposed the MDL-based SSS algorithm and showed that the MDL criterion works well for the large amount of training data. However, conventional information criteria require some assumptions, e.g., asymptotic normality. Theoretically speaking, It is difficult for information criteria to evaluate complicated models like neural networks, or HMMs exactly because these models cannot satisfy such assumptions.

As we described in Section 2.5, the Variational Bayesian (VB) method was proposed to avoid these problems[13][14][15]. Recently, the VB approach have been applied to speech recognition. Decision tree clustering with the VB method was proposed[30], and Variational Bayesian GMMs were applied to speech recognition[31]. These methods do not consider any temporal structures.

We propose an automatic topology creation method using the SSS algorithm with the Variational Bayesian method, which we call the VB-SSS algorithm, to estimate topologies more exactly. The SSS algorithm can create contextual and temporal variations. In contrast, decision tree clustering can only create contextual variations. In [30], they describe the general parameter estimation of HMMs based on the VB approach and the topology estimation by tree-clustering based on the VB approach. In the decision tree clustering, the number of states per triphone must be decided before

clustering, and it is never changed after clustering. That method divides utterances into state segments and never changes state boundaries to avoid considering temporal variations. Therefore, in the decision tree clustering, the number of states per triphone is considered as one of the problems with initial models. However, the SSS-based algorithm can consider both contextual and temporal variations. It can create a triphone with a different number of states from the other triphones according to training data. Therefore, our proposed method, the SSS algorithm based on the VB approach, has a higher number of degrees of freedom than that of the decision tree clustering. Furthermore, latent variables should be employed in the SSS algorithm because the alignments of phonemes are fixed but those of states are not. Therefore, the occupancy probabilities of training samples should be considered by using latent variables to introduce the VB method into the SSS algorithm.

We also evaluate a method for increasing the number of mixture components by using the VB approach, based on a topology obtained by the VB-SSS algorithm. In [30], Watanabe et al. evaluated two methods for constructing Gaussian mixture models. One sets the same number of Gaussians per state for all states, and selects an appropriate model by a VB objective function. The other determines the number of Gaussians for each state by splitting and merging Gaussians in each state with the objective function. In [31], they produced GMMs by decreasing the number of mixture components in each phoneme. Since the VB-SSS algorithm generates HMM structures with temporal structures, our proposed methods consider temporal structures to make mixture models by splitting Gaussians with the VB approach.

In Section 4.2, we present the VB-SSS algorithm, and in Section 4.3 explain a method for increasing the mixture components. In Section 4.4, we evaluate the performance of our proposed methods with experiments. Finally, we provide our summary about this topic in Section 4.5.

4.2. Variational Bayesian Approach for SSS Algorithm

4.2.1 Overview of VB-SSS

Our proposed method is based on the ML-SSS algorithm[7]. The ML-SSS algorithm assumes that each state has a single Gaussian distribution, and that each category can be

represented by one Gaussian distribution when splitting is performed. This algorithm also assumes that suboptimal models can be obtained by increasing the number of mixture components after this topology training even if such models are not optimal for the number of parameters. Therefore, our proposed method, the VB-SSS algorithm, also uses only a single Gaussian model, and after this algorithm, there is a need for a method to increase the number of mixture components.

Figure 4.1 shows the flow of the VB-SSS algorithm. This section briefly explains the VB-SSS algorithm. First, the topology of an initial model is set and its parameters are estimated. Second, the prior parameters for each state are set, after which the posterior parameters for each state are estimated, and the VB objective function, \mathcal{F}_m , (see [15] for details) is calculated as the baseline energy.

After that, each type of splitting is performed in the same manner as with the ML-SSS algorithm. For each splitting, after two new states are created, the posterior parameters are estimated, and the energy gains of both the contextual splitting and the temporal splitting are calculated. Next, the state splitting with the maximum energy gain is selected. If there is no state that can increase its energy, the splitting is stopped. Furthermore, when \mathcal{F}_m decreases or converges, the splitting is stopped. Otherwise, the parameters of HMMs are estimated, and these procedures are repeated. In this paper, all of the posterior parameters are estimated by using all of the data for each test splitting.

4.2.2 Contextual and temporal splitting

The probability density of the HMM Θ , which has N_s states with one Gaussian distribution and N_a transitions for each state for both contextual and temporal splitting, is

$$p(\mathbf{O}|\Theta) = \prod_{t=1}^T \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}) a_{s_t r_{t+1}}, \quad (4.1)$$

where $\mathbf{O} = \{\mathbf{o}_1, \dots, \mathbf{o}_t, \dots, \mathbf{o}_T\}$ is a set of training samples, s_t denotes the state number at time t , and r_t represents the transition arc number at time t . In addition, $\boldsymbol{\mu}_{s_t}$ is a mean vector at s_t , $\boldsymbol{\Sigma}_{s_t}$ denotes a covariance matrix at s_t , and $a_{s_t r_{t+1}}$ is a transition probability. We use a diagonal matrix as the covariance matrix. The maximum of N_a is N_s , and N_a

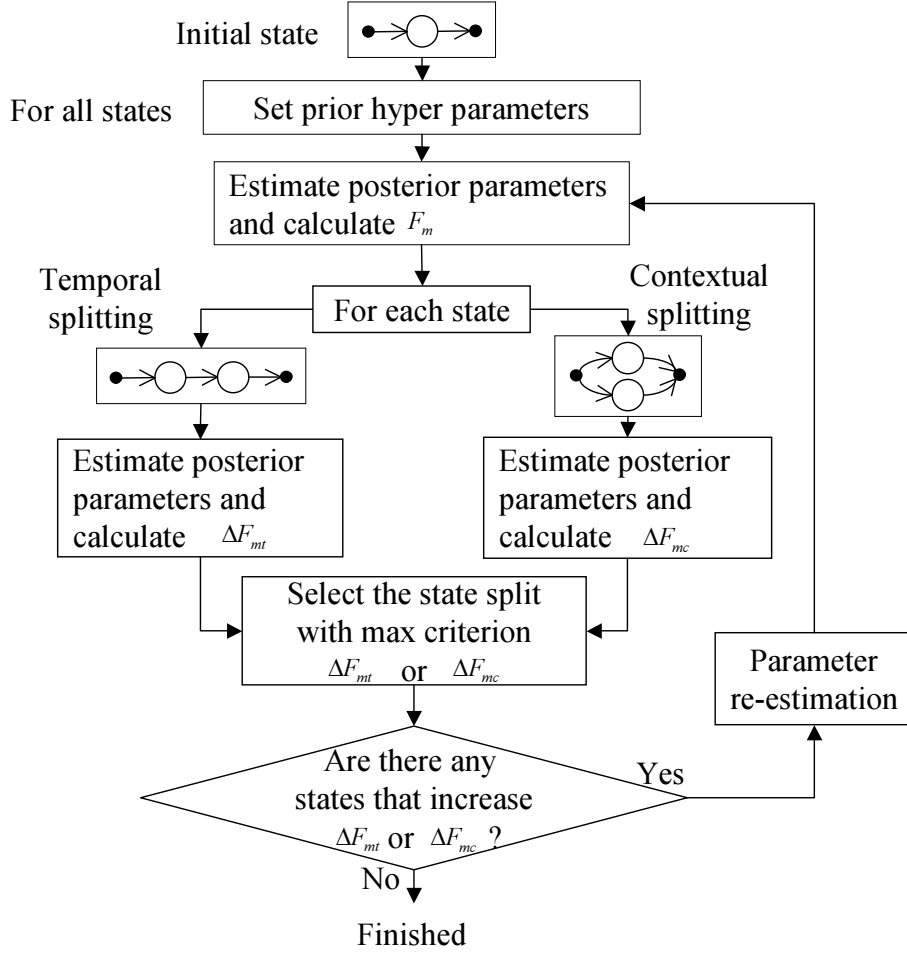


Figure 4.1. Flow of the Variational Bayesian SSS algorithm.

in this paper can be replaced by N_s . However, this splitting algorithm can use $N_a = 2$ only.

The probability for the complete data set to which the latent variables are introduced is

$$p(\mathbf{O}, Z|\Theta) = \prod_{t=1}^T \prod_{i=1}^{N_s} \prod_{j=1}^{N_a} \{\mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) a_{ij}\}^{z_{ij}^t}, \quad (4.2)$$

where $Z = \{z_{ij}^t\}_{i=1, j=1, t=1}^{N_s, N_a, T}$ is the set of latent variables.

The objective function \mathcal{F}_m is defined as a lower bound of a marginal likelihood over

all random quantities with a fixed structure m [15]:

$$\mathcal{F}_m = \int q(Z)q(\Theta) \ln \frac{p(\mathbf{O}, Z|\Theta)p(\Theta)}{q(Z)q(\Theta)} dZd\Theta, \quad (4.3)$$

where $q()$ stands for a variational posterior probability, which approximates a true posterior probability; $q()$ becomes the closest distribution to its true posterior probability when \mathcal{F}_m is maximized. An iterative procedure to find the optimal variational posteriors is defined by the partial derivative of \mathcal{F}_m w.r.t. each $q()$. It is referred to as the Variational Bayesian EM Steps.

When the i th state with the HMM parameter Θ_i is split into the i_1 th state and the i_2 th state, and the parameter $\hat{\Theta}_i$ is estimated for the current splitting, the splitting criterion can be represented by using the objective function \mathcal{F}_m as follows,

$$\Delta\mathcal{F}_m^{(n+1)} = \mathcal{F}_m^{(n+1)}(\hat{\Theta}_i) - \mathcal{F}_m^{(n)}(\Theta_i), \quad (4.4)$$

where n is the iteration number.

4.2.3 Priors

We assume that the probability of parameters can be factorized as follows.

$$p(\Theta) = p(N_s, N_a)p(\mathbf{a}|N_s, N_a)p(\Sigma|N_s)p(\mu|\Sigma, N_s). \quad (4.5)$$

We also assume that the prior of $\mathbf{a} = \{a_{ij}\}_{i=1, j=1}^{N_s, N_a}$, $a_{ij} \geq 0$, $\sum_{j=1}^{N_a} a_{ij} = 1$ is a *Dirichlet* distribution, and that the prior of $\{\mu, \Sigma\} = \{\{\mu_i\}_{i=1}^{N_s}, \{\Sigma_i\}_{i=1}^{N_s}\}$ is a *normal-Gamma* distribution,

$$p(\mathbf{a}|N_s, N_a) = \prod_{i=1}^{N_s} \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \phi_0) \propto \prod_{i=1}^{N_s} \prod_{j=1}^{N_a} a_{ij}^{\phi_0-1}$$

$$p(\mu, \Sigma|N_s) = \prod_{i=1}^{N_s} \prod_{k=1}^D \mathcal{N}(\mu_{ik}; \nu_{0k}, \xi_0^{-1} \sigma_{ik}) \mathcal{G}(\sigma_{ik}^{-1}; \eta_0/2, b_{0k}/2),$$

where D is the order of parameters, μ_{ik} and σ_{ik} are the k th elements of μ_i and Σ_i , respectively, $\mathcal{N}()$ denotes the Gaussian distribution, $\mathcal{G}()$ represents the Gamma distribution, and ϕ_0 , ν_{0k} , ξ_0 , η_0 , and b_{0k} are prior parameters. The definition of the Gamma distribution is $\mathcal{G}(s; \eta, \lambda) = \lambda^\eta / \Gamma(\eta) \cdot s^{\eta-1} \exp(-\lambda s)$, where $\Gamma()$ is the Gamma function.

4.2.4 Posteriors

We also assume that the posterior probability of parameters can be factorized as follows.

$$q(\Theta) = q(N_s, N_a)q(\mathbf{a}|N_s, N_a)q(\Sigma|N_s)q(\boldsymbol{\mu}|\Sigma, N_s). \quad (4.6)$$

The posterior probability can be derived from the Variational Bayesian EM algorithm[15].

$$q(\mathbf{a}|\mathbf{O}, N_s, N_a) = \prod_{i=1}^{N_s} \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \{\phi_{ij}\}_{j=1}^{N_a}), \quad (4.7)$$

$$\phi_{ij} = \phi_0 + \bar{N}_{ij}, \quad \bar{N}_{ij} = \sum_{t=1}^T \bar{z}_{ij}^t, \quad \bar{z}_{ij}^t = \langle z_{ij}^t \rangle_{q(Z)},$$

$$q(\boldsymbol{\mu}, \Sigma|\mathbf{O}, N_s) = \prod_{i=1}^{N_s} \prod_{k=1}^D \mathcal{N}(\mu_{ik}; \nu_{ik}, \xi_i^{-1} \sigma_{ik}) \mathcal{G}(\sigma_{ik}^{-1}; \eta_i/2, b_{ik}/2), \quad (4.8)$$

$$\bar{N}_i = \sum_{t=1}^T \bar{z}_i^t, \quad \bar{z}_i^t = \langle z_i^t \rangle_{q(Z)},$$

$$\nu_{ik} = \frac{\bar{N}_i \bar{o}_{ik} + \xi_0 \nu_{0k}}{\bar{N}_i + \xi_0}, \quad \xi_i = \xi_0 + \bar{N}_i, \quad \eta_i = \eta_0 + \bar{N}_i,$$

$$b_{ik} = b_{0k} + \bar{c}_{ik} + \frac{\bar{N}_i \xi_0}{\bar{N}_i + \xi_0} (\bar{o}_{ik} - \nu_{0k})^2,$$

$$\bar{o}_i = \frac{1}{\bar{N}_i} \sum_{t=1}^T \bar{z}_i^t \mathbf{o}_t, \quad \bar{c}_{ik} = \sum_{t=1}^T \bar{z}_i^t (o_{tk} - \bar{o}_{ik})^2.$$

Here, $\langle x \rangle_{f(x)} = \int x f(x) dx$ is the expectation of x for $f(x)$. The variational posterior probability of latent variables is also derived in the same manner as the unknown parameters; \mathcal{F}_m can be derived from these priors and posteriors.

The variational posterior probability of latent variables is

$$\bar{z}_{ij}^t = \exp(\gamma_{ij}^t) / \sum_{k=1}^{N_s} \sum_{l=1}^{N_a} \exp(\gamma_{kl}^t), \quad (4.9)$$

$$\begin{aligned} \gamma_{ij}^t \propto & \Psi(\phi_{ij}) - \Psi\left(\sum_{j=1}^{N_a} \phi_{ij}\right) + \frac{D}{2} \Psi\left(\frac{\eta_i}{2}\right) - \frac{1}{2} \sum_{k=1}^D \ln \frac{b_{ik}}{2} \\ & - \frac{D}{2\xi_i} - \frac{1}{2} \sum_{k=1}^D \frac{\eta_i}{b_{ik}} (o_{tk} - \nu_{ik})^2, \end{aligned} \quad (4.10)$$

where $\Psi(x) = \partial \ln \Gamma(x) / \partial x$ is the digamma function.

4.2.5 Objective function

Hereafter, $p(\cdot|N_s, N_a)$ is simplified to $p(\cdot)$, i.e., $p(\Theta|N_s, N_a) \rightarrow p(\Theta)$. The variational Bayesian objective function is

$$\begin{aligned}
\mathcal{F}_m &= \int q(Z)q(\Theta) \ln \frac{p(\mathbf{O}, Z|\Theta)p(\Theta)}{q(Z)q(\Theta)} dZd\Theta \\
&= \int q(Z)q(\Theta) \ln \prod_{t=1}^T p(\mathbf{o}_t, z_t|\Theta) dZd\Theta - \int q(Z) \ln q(Z) dZ \\
&\quad - \int q(\Theta) \ln q(\Theta) d\Theta + \int q(\Theta) \ln p(\Theta) d\Theta.
\end{aligned} \tag{4.11}$$

A brief derivation is provided in Appendix A.1.

4.3. Increasing Mixture Components Based on the VB Approach

4.3.1 Splitting mixture method

After topologies are obtained by the VB-SSS algorithm, the number of mixture components is increased by the following algorithm based on the VB approach. We define the splitting mixture method as follows.

[Splitting mixture method]

1. Set an initial model obtained by topology training. $M^{(0)} = 1, n = 0$.
2. Calculate the objective function $\mathcal{F}_m^{(n)}$.
3. Iterate the following steps for each phoneme.
 - (a) Split each distribution into two distributions in each state. $M^{(n+1)} = 2M^{(n)}$. (Fig. 4.2)
 - (b) Estimate posterior distributions, and calculate the objective function $\mathcal{F}_m^{(n+1)}$, repeatedly.
 - (c) Stop splitting when $\Delta\mathcal{F}_m^{(n+1)} = \mathcal{F}_m^{(n+1)} - \mathcal{F}_m^{(n)}$ is a negative number. Otherwise, $n = n + 1$, and go to 3(a).

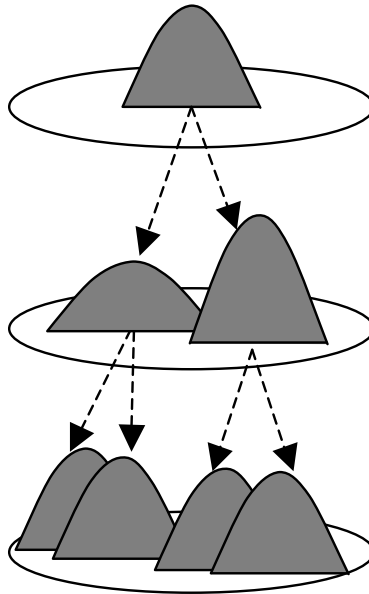


Figure 4.2. Splitting each distribution into two distributions.

This algorithm splits each mixture component to two distributions, as in Fig. 4.2. In this algorithm, the number of mixture components is estimated for each phoneme. It obtains more suitable models than models with the same number of mixture components for all phonemes.

We also evaluated the other algorithm that increased one distribution for each state at a time. However, this algorithm performed a slightly worse than the above algorithm in preliminary experiments.

4.3.2 VB approach for increasing mixture components

In [30] and [31], the authors estimated the number of mixture components for each state because their methods are the same as those used for GMMs. On the other hand, the VB-SSS algorithm estimates model structures by considering the transition probabilities using the forward-backward algorithm. Therefore, our proposed method estimates the number of mixture components with the forward-backward algorithm for phoneme periods.

Gaussian mixture HMMs can be represented as follows.

$$p(\mathbf{O}|\Theta) = \prod_{t=1}^T \left\{ \sum_{k=1}^{M_{s_t}^{(n)}} w_{s_t k} \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{s_t}, \boldsymbol{\Sigma}_{s_t}) \right\} a_{s_t r_{t+1}}, \quad (4.12)$$

where s_t denotes the state index at time t , $\{w_{ik}\}_{k=1}^{M_{s_t}^{(n)}}$ is a set of mixture weights for state i , $\boldsymbol{\mu}_{s_t}$ is a mean vector, and $\boldsymbol{\Sigma}_{s_t}$ is a covariance matrix. In addition, r_t is an arc index at time t , $\{a_{ij}\}_{j=1}^{N_a}$ is a set of transition probabilities.

The priors and posteriors for transition probabilities, mean vectors, and precision matrices can be defined just as those of the VB-SSS algorithm. For transition probabilities,

$$p(\mathbf{a}|N_s, N_a) = \prod_{i=1}^{N_s} \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \phi_0),$$

and for mean vectors and precision matrices,

$$p(\boldsymbol{\mu}, \mathbf{S}|N_s, \{M_i\}_{i=1}^{N_s}) = \prod_{i=1}^{N_s} \prod_{k=1}^{M_i} \prod_{l=1}^D \mathcal{N}(\mu_{ikl}; \nu_{0l}, \xi_0^{-1} \sigma_{ikl}) \mathcal{G}(\sigma_{ikl}^{-1}; \eta_0/2, b_{0l}/2).$$

For mixture weights, a *Dirichlet* distribution can be used.

$$p(\mathbf{w}|N_s, \{M_i\}_{i=1}^{N_s}) = \prod_{i=1}^{N_s} \mathcal{D}(\{w_{ik}\}_{k=1}^{M_i}; \rho_0),$$

where ρ_0 is a prior parameter. The posterior probabilities for these probabilities and the VB objective function, including mixture components, can be derived in the same manner as these in the VB-SSS algorithm.

For recognition, posterior predictive probability is used for the Bayesian approach.

$$p(\mathbf{x}|m, \mathbf{O}) = \prod_{t=1}^T \int p(\mathbf{x}_t | \Theta_{s_t s_{t+1}}, m, \mathbf{O}) p(\Theta_{s_t s_{t+1}} | m, \mathbf{O}) d\Theta_{s_t s_{t+1}}. \quad (4.13)$$

Here, $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ is a set of test data, and m represents a structure indicator, that is, the number of states, transitions, and mixture components in this work. The true posterior probability $p(\Theta_{ij}|m, \mathbf{O})$ is approximated by the variational posterior probability

$$\begin{aligned}
& q(\{a_{ij}\}_{j=1}^{N_a} | m) \prod_{k=1}^{M_i} q(\{w_{ik}\}_{k=1}^{M_i} | m) q(\boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik} | m). \\
p(\mathbf{x} | m, \mathbf{O}) & \simeq \prod_{t=1}^T \left\langle a_{ij} \right\rangle_{q(\{a_{ij}\}_{j=1}^{N_a} | m)} \sum_{k=1}^{M_i} \left\{ \left\langle w_{ik} \right\rangle_{q(\{w_{ik}\}_{k=1}^{M_i} | m)} \left\langle \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \right\rangle_{q(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | m)} \right\}, \\
& \left\langle a_{ij} \right\rangle_{q(\{a_{ij}\}_{j=1}^{N_a} | m)} = \phi_{ij} / \sum_{j'} \phi_{ij'}, \\
& \left\langle w_{ik} \right\rangle_{q(\{w_{ik}\}_{k=1}^{M_i} | m)} = \rho_{ik} / \sum_{k'} \rho_{ik'}, \\
& \left\langle \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik}) \right\rangle_{q(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i | m)} = \mathcal{T}(\mathbf{x}_t; \boldsymbol{\nu}_{ik}, \Phi_{ik}, f_{ik}), \\
& f_{ik} = \eta_{ik}, \Phi_{ik} = \mathbf{B}_{ik}(\xi_{ik} + 1) / (\xi_{ik} f_{ik}).
\end{aligned} \tag{4.14}$$

$\mathcal{T}(\cdot)$ is a *Student-t* distribution.

$$\begin{aligned}
\mathcal{T}(\mathbf{x}_t; \boldsymbol{\nu}_{ik}, \Phi_{ik}, f_{ik}) & = C_{ik} \{1 + (\mathbf{x}_t - \boldsymbol{\nu}_{ik})' (f_{ik} \Phi_{ik})^{-1} (\mathbf{x}_t - \boldsymbol{\nu}_{ik})\}^{-\frac{1}{2}(f_{ik}+1)}, \\
C_{ik} & = \frac{\Gamma((f_{ik} + D)/2)}{(f_{ik}\pi)^{D/2} \Gamma(f_{ik}/2) |\Phi_{ik}|^{\frac{1}{2}}}.
\end{aligned}$$

Here, “'” represents a transpose.

4.4. Experiments

4.4.1 Experimental conditions

In this section, we evaluated our proposed method by both segmented phoneme recognition and conventional continuous speech recognition. Segmented phoneme recognition is the classification test for segments that are divided into phonemes in order to evaluate each phoneme model’s performance. We compared our proposed method, the VB-SSS, to the ML-SSS and the MDL-SSS algorithms. For the ML-SSS, two models with different maximum state lengths, 3 or 4, were created. These two models are the baseline models.

For the acoustic training set, we used Japanese dialog speech from the ATR travel arrangement task (TRA) database[33] uttered by 166 males. The total length of speech was 2.1 hours. The MDL criterion is not suitable for such a small database, but the VB approach is applicable theoretically; the VB-SSS still requires more computation.

Therefore, in this work, we used this small amount of training data for the experiments, in this paper.

For testing, we used dialog speech that includes 213 sentences from the TRA database uttered by a different set of 17 males. For topology training, we employed the VB approach only for the splitting and stopping criteria. Multi-class composite bigram models [34] were used, and the vocabulary size was 5,000. The sampling frequency was 16 kHz, the frame length was 20 ms, and the frame shift was 10 ms. We used 12-order MFCC, Δ MFCC, and Δ log power as feature parameters. In addition, cepstrum mean subtraction was applied to each utterance. We used 26 kinds of phonemes and one silence. Three states were used as the initial model for each phoneme, and one Gaussian distribution for each state was used during topology training. A silence model with three states was built separately from the phoneme models, and to increase the number of mixture components with the VB approach, the number of Gaussians for the silence model was determined by employing the VB approach. In these experiments, we used $\phi_0 = 1.0$, $\xi_0 = 1.0$, $\eta_0 = 2.0$ for the prior parameters of the VB-SSS. ν_{0k} and b_{0k} were set from the element values of the mean vectors and the covariance matrices.

4.4.2 Evaluation for topology training

Figures 4.3 and 4.4 show the average phoneme recognition rates for vowels and consonants, respectively. The “phoneme recognition rate” means the rate of correctly classified segments in a phoneme. It includes the results by the ML-SSS with a maximum state length of 3 or 4, by the MDL-SSS, and by the VB-SSS. To compare topologies generated by these methods, we assigned a single Gaussian distribution for each state. The VB-SSS obtained better performance for vowels and a slightly worse one than the ML-SSS with a maximum state length = 3 for consonants. In addition, the topology created by the MDL-SSS is too small to obtain performance comparable with the other methods because the MDL criterion generally does not work for a small amount of training data. Our work on the MDL-SSS[36] shows that the MDL-SSS can automatically obtain almost the same performance as the ML-SSS; however, the total amount of training data in this paper is much smaller than that in the reference[36]. Checking the results in detail, the VB-SSS obtained better results for many phonemes than did the ML-SSS, though the VB-SSS obtained worse results for a few consonants, especially,

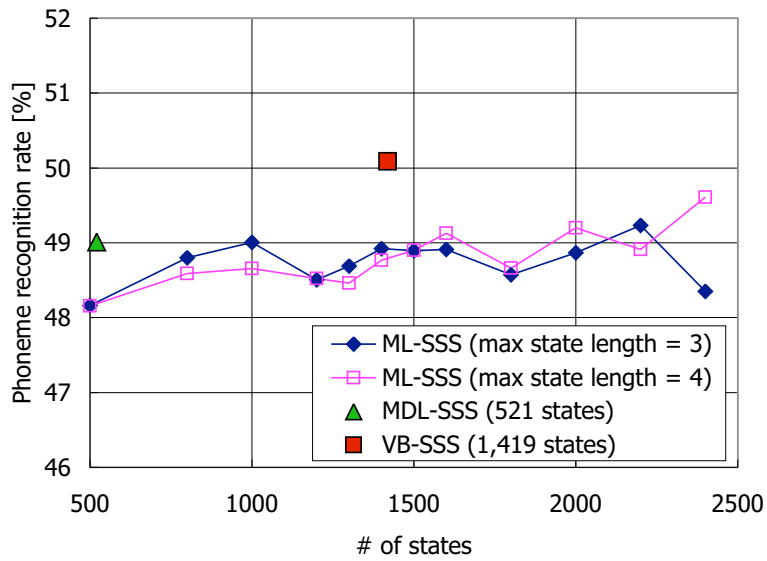


Figure 4.3. Average phoneme recognition rates for vowels.

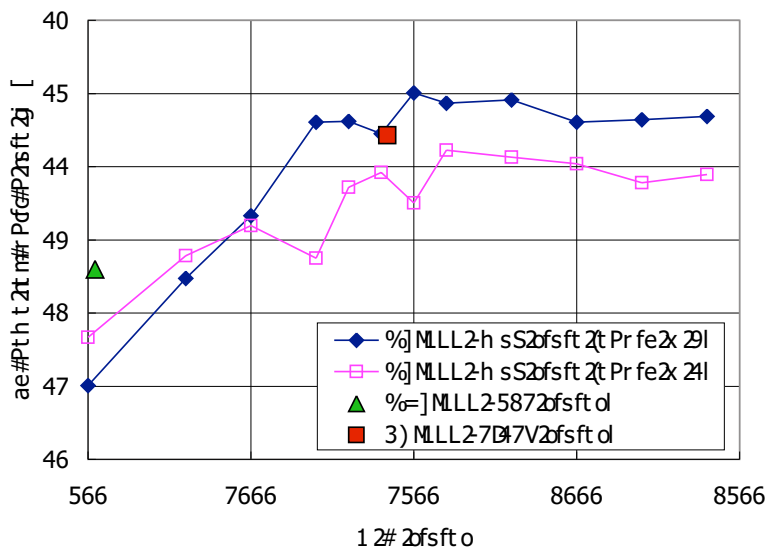


Figure 4.4. Average phoneme recognition rates for consonants.

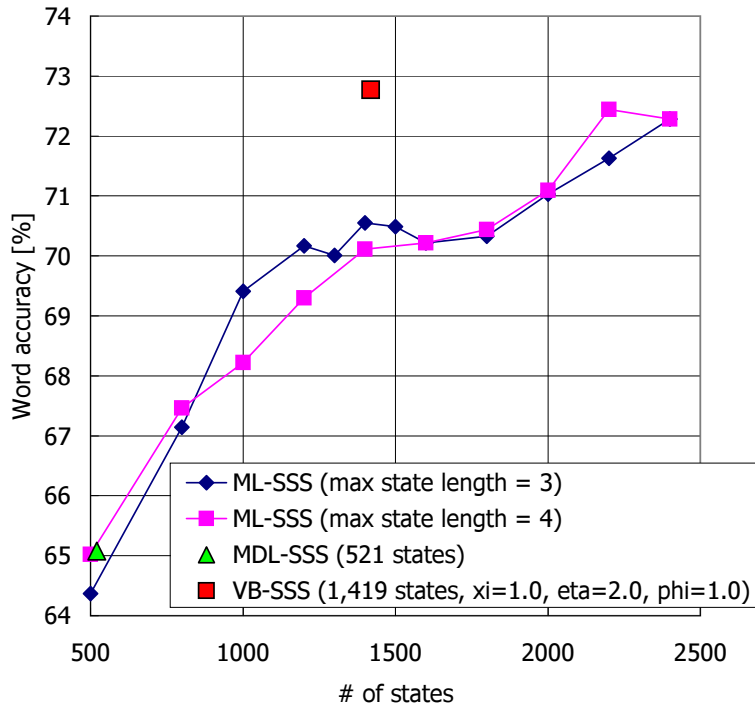


Figure 4.5. Word accuracy rates by single Gaussian models.

/f/. The reason is under investigation.

Figure 4.5 shows the results by using the single Gaussian models. The performance of the MDL-SSS was again worse than the baseline, ML-SSS, due to the small amount of training data. On the other hand, with about 60% of the ML-SSS states, the VB-SSS achieved a comparable recognition rate.

Next, we analyzed the dependencies of the prior hyperparameters. Table 4.1 shows word accuracy rates and the number of states of several prior parameters for the 5k-CSR task, with the trend of results for segmented phoneme recognition being almost the same. The fluctuation of performance is small when ϕ_0 is changed under almost the optimal values, $\xi_0 = 1.0$ and $\eta_0 = 2.0$. Also, ϕ_0 is a hyperparameter of transition probabilities. Because transition probabilities do not have much effect on recognition performance, the influence of ϕ_0 is smaller than the other parameters. Although we also evaluated models with $\eta_0 = 20$, the parameters of posteriors could not be obtained in some phonemes because the parameters diverged and no model could be obtained. Consequently, nearly optimal values of ξ_0 and η_0 are limited to a certain range of values.

Table 4.1. Word accuracy rates [%] and # of states in parentheses for several hyperparameters

$\xi_0 = 0.1$	$\phi_0 = 1.0$	$\phi_0 = 10$	$\phi_0 = 100$
$\eta_0 = 0.2$	68.87 (766)	69.14 (764)	68.76 (775)
$\eta_0 = 2.0$	72.12 (1,361)	71.30 (1,252)	67.73 (1,246)
$\xi_0 = 1.0$	$\phi_0 = 1.0$	$\phi_0 = 10$	$\phi_0 = 100$
$\eta_0 = 0.2$	68.87 (760)	68.92 (761)	68.87 (749)
$\eta_0 = 2.0$	<u>72.77</u> (1,419)	<u>72.55</u> (1,425)	72.17 (1,433)
$\xi_0 = 10$	$\phi_0 = 1.0$	$\phi_0 = 10$	$\phi_0 = 100$
$\eta_0 = 0.2$	70.01 (771)	68.60 (757)	69.09 (780)
$\eta_0 = 2.0$	71.30 (1,315)	72.06 (1,358)	66.00 (1,211)

Table 4.2. Word accuracy rates for thirty minutes of training data

	#states	WA [%]
ML-SSS	300	57.44
	564	61.61
	812	61.78
MDL-SSS	199	52.14
VB-SSS	564	61.99

The posteriors of these prior parameters, ξ_0, η_0, ϕ_0 , are updated as they are shown after Eqs. (4.7), or (4.8), like $\xi_i = \xi_0 + \bar{N}_i$. Each of these posteriors is dependent on the number of samples belonging to each class, and the larger the number of samples, the smaller the influence of these prior parameters.

Additionally, we also evaluated our method by using the smaller amount of training data. Thirty minutes of utterances were extracted from the same training data used in the previous experiments. In these experiments, we used $C_c = 2, C_t = 20$ for the MDL-SSS, and $\phi_0 = 1.0, \xi_0 = 1.0, \eta_0 = 2.0$ for the prior parameters of the VB-SSS. Table 4.2 shows word accuracy rates for thirty minutes of training data. These results were obtained by single Gaussian models. The VB-SSS can automatically obtain almost the same performance with smaller parameters than that of the ML-SSS, indicating that the VB approach works well even for smaller databases.

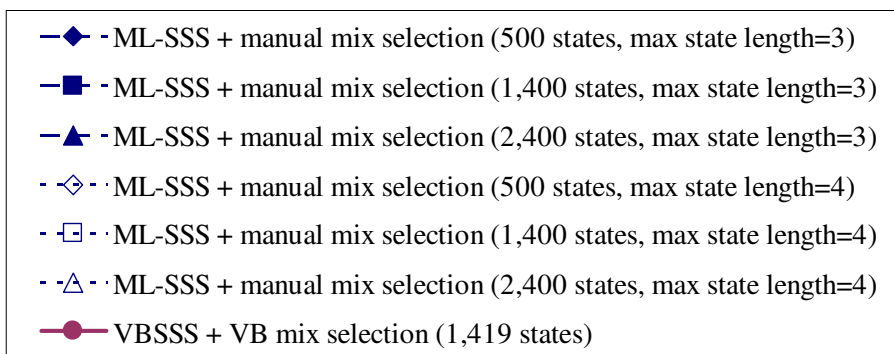
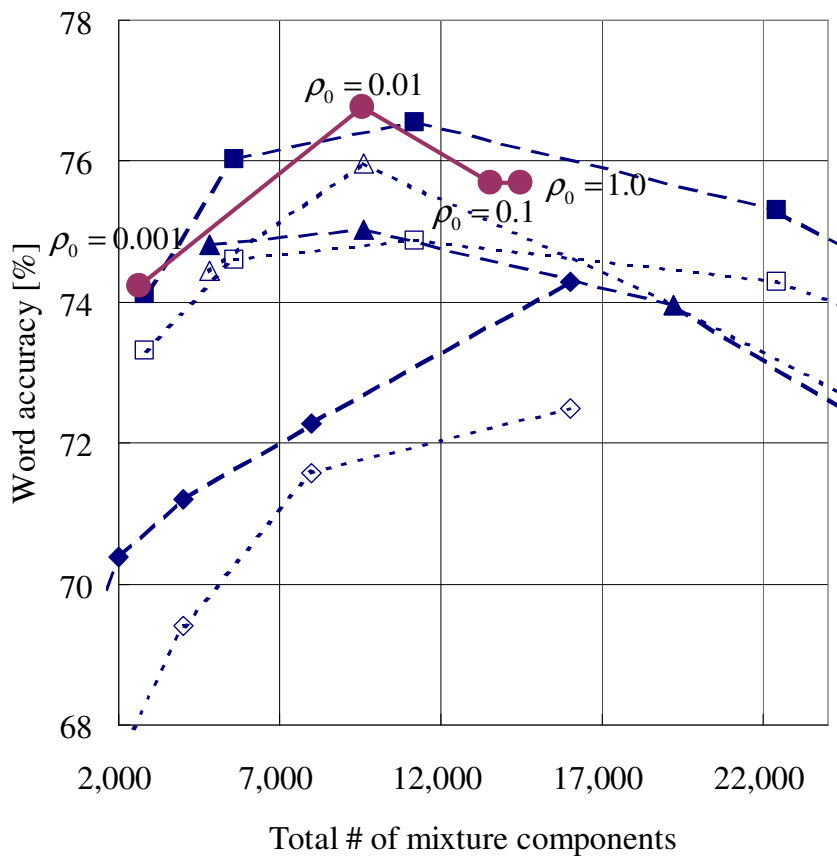


Figure 4.6. Word accuracy rates by Gaussian mixture models.

Table 4.3. The average number of mixture components per state, the total number of mixture components, and word accuracy rate

	ρ_0	#mixtures /state	#mixtures	WA[%]
ML-SSS + manual mix selection (1,400 states)	–	8	11,200	76.56
VB-SSS + VB mix selection (1,419 states)	0.001	1.87	2,652	74.23
	0.01	<u>6.74</u>	<u>9,564</u>	<u>76.77</u>
	0.1	9.53	13,520	75.69
	1.0	10.19	14,460	75.69

4.4.3 Evaluation of mixture splitting

Figure 4.6 shows the results by using the splitting mixture method. Furthermore, Table 4.3 shows the average number of mixture components, the total number of mixture components, and word accuracy rate for the best model of the baseline and the models by using the VB approach with several values of the prior parameter, ρ_0 . Posterior predictive probabilities defined by Eq. (4.13) are used for decoding by Bayesian approach, showing that the VB approach obtained almost the same performance with a 15%-smaller number of Gaussians than that obtained by using the ML based method. These results indicate that recognition performance is dependent on ρ_0 . This posterior parameter is updated by $\rho_{ik} = \rho_0 + \bar{N}_{ik}$. As we explained about the other prior parameters like ξ_0 , the effectiveness of ρ_0 is dependent on the number of samples, \bar{N}_{ik} ; the larger the number of samples, the smaller the effect. Furthermore, the amount of training data in these experiments is too small for use as conventional training data.

In addition, we evaluated four combinations of topology training methods and decoding methods to examine combinations of the VB-based topology training, the VB mixture splitting, and the decoding by the Bayesian approach. This experiment can show that criteria both for topology training and mixture selection should be consistent.

For topology training, we can select either the ML-SSS or the VB-SSS, while for

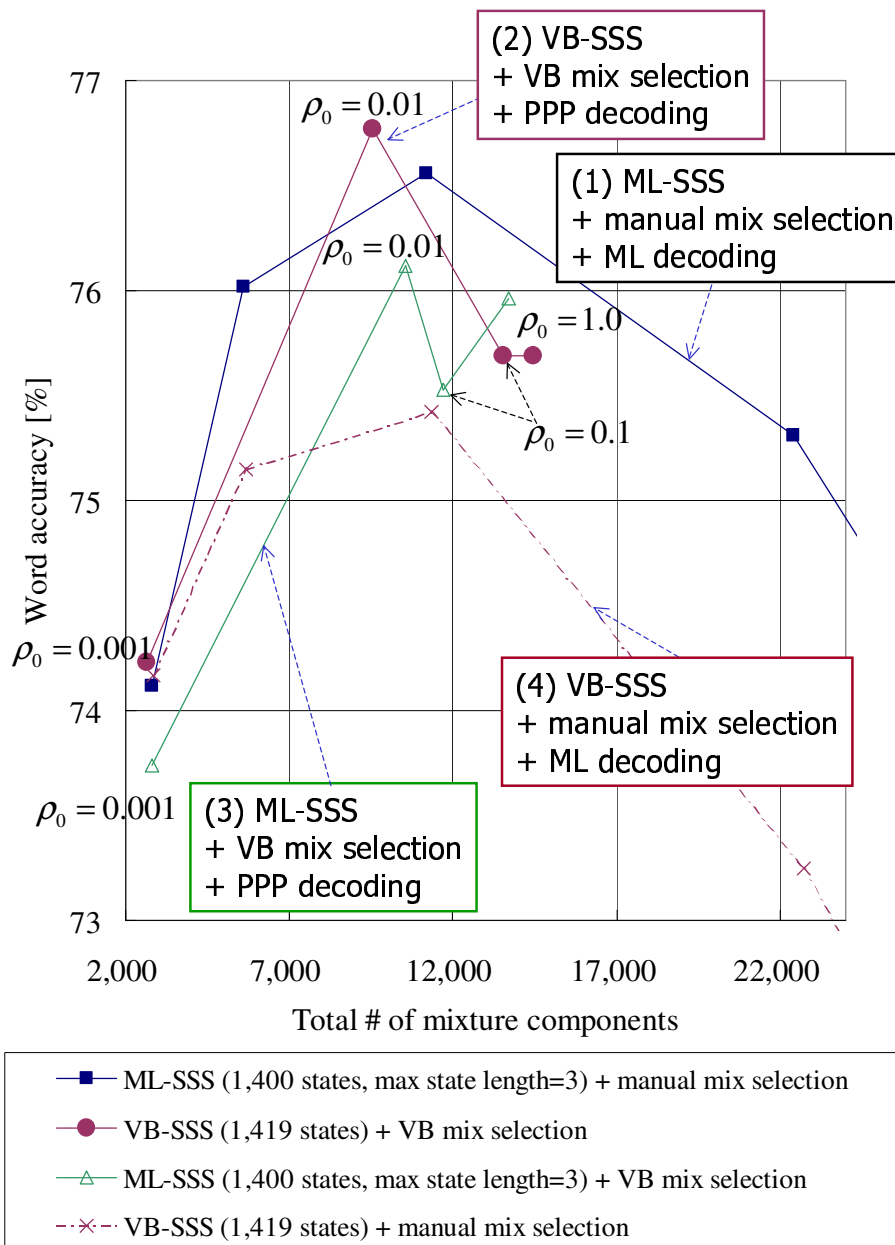


Figure 4.7. Word accuracy rates by four types of combinations.

mixture selection, we can use the ML-based manual selection or the VB-based method. The decoding method is dependent on the parameter estimation method, and for ML-based manual selection, it is the usual ML-based decoding (ML decoding) method. For models trained by VB-based mixture selection, posterior predictive probabilities are used for decoding. This is called “PPP decoding” for short in this section. Therefore, there are four combinations as listed below.

1. ML-SSS + manual mixture selection + ML decoding
2. VB-SSS + VB mixture selection + PPP decoding
3. ML-SSS + VB mixture selection + PPP decoding
4. VB-SSS + manual mixture selection + ML decoding

In both methods 1. and 2., the criteria for both topology training and mixture selection are the same, and their results are the same as those in Fig. 4.6 and Table 4.3.

Figure 4.7 shows word accuracy rates achieved by these four combinations. The VB approach both for topology training and mixture selection gave the best result among these combined methods.

4.5. Summary

We proposed using the Variational Bayesian approach to automatically create non-uniform, context-dependent HMM topologies. We introduced the VB approach to the SSS algorithm to create contextual and temporal variations for HMMs and then defined posterior probability densities and the VB free energy as split and stop criteria. We evaluated the proposed method for word-based continuous speech recognition. The VB-SSS automatically achieved comparable performance with about 60% of states generated by the ML-SSS. Furthermore, we evaluated a method for increasing the number of mixture components, employing the VB approach. Experimental results indicated that the VB approach could obtain almost the same performance with a 15%-smaller number of Gaussians than that obtained by using the ML-based method.

We evaluated performance for combinations of several values of prior parameters and found the almost optimal value or range for each parameter. Theoretically, their effectiveness is dependent on the number of samples, and the obtained suboptimal

values can be applied to other tasks. However, there are still some rooms for improving prior parameters.

The ML-SSS[7] cannot use Gaussian mixture HMMs, and is difficult to extend the production of mixture models. In [37], Kato et al. proposed a decision tree clustering for Gaussian mixture HMMs. It can deal with Gaussian mixture HMMs during topology training, but the number of mixture components and the number of states for each allophone model are fixed. For future work, we would like to develop a method to optimize the number of states for a whole model, the number of states for each allophone model, and the number of mixture components for each state, simultaneously.

Chapter 5

Language Modeling Using Patterns Extracted from Parse Trees

5.1. Introduction

In both Chapter 3 and Chapter 4, we proposed the two types of automatic generation of acoustic modeling. In this chapter, we will present our new method for language modeling.

In large vocabulary continuous speech recognition, word n-gram models are widely and effectively used as language models. However, they can represent only local constraints within a few successive words and lack the ability to capture the global structures of sentences. Trigger models have been proposed to cope with these weaknesses[17]. They can model word co-occurrence characteristics beyond 2-3 grams. S. Zhang et al. also proposed a solution called Linkgram[18]. This model has word pairs extracted from parse trees and can represent syntactic relations between word pairs. Such constraints, however, are weak for the global structures of sentences. Chelba and Jelinek proposed a method using a stochastic parser as a language model[19][20], which called as a Structured Language Model (SLM).

We propose two new types of language models using phrasal constraints extracted from parse trees produced by an example-based parser. For spontaneous speech, exact sentence structures are not so important, but partial structures are more useful for constraints. First, we propose n-gram models for sentences with phrase constituent boundary markers, and second we propose word pattern models using partial structure

patterns of parse trees. Both of these methods attach weight to constraints of partial phrase structures. These models are created from outputs of the example-based parser, called the Constituent Boundary Parser (CBP), which was developed by Furuse et al., ATR[23], for example-based machine translation, called Transfer-Driven Machine Translation (TDMT). The parser analyzes sentences by example word patterns and rules made by hand.

The proposed models extracted by the parser can represent not only intra-phrase constraints, but also inter-phrase constraints, the latter of which appear as syntactic long-distant constraints between words. Our models can represent local structures extracted from only surface word sequences after preprocessing, and bigger structures extracted from subtrees of parse trees. These models do not take account of representing global structures but they include local constraints, while the SLMs try to represent global structures. In spontaneous speech, there are so many sentences whose global structures are ungrammatical but local structures are grammatical. It is suitable for spontaneous speech to represent local constraints.

In this paper, first, we describe phrase structure representation by our parser in Section 5.2. Second, our proposed language models are introduced in Section 5.3. In Section 5.4, we apply these new models to the task of speech recognition of ATR travel dialogues in Japanese, and show the performance of these models. Furthermore, we investigate the effectiveness of word pattern models by additional experiments in Section 5.5.

5.2. Phrase Structure Representation

The language models proposed in this paper incorporate linguistic constraints depending on the phrase structures of sentences. This section gives a brief explanation about the phrase structure representation used in this research.

5.2.1 Phrase structure using constituent boundaries

As a parser of spoken-language translation, ATR proposed a method called Constituent Boundary Parsing (CBP) that uses pattern matching on the surface form [23]. CBP considers that every content word in a sentence is separated (or *bounded*) from another

content word by either a function word or an artificial *Part-of-Speech-bigram marker*. For example, “*I go to Kyoto*” can be thought of as follows:

“ <i>I</i> ”	“< <i>pronoun – verb</i> >”	“ <i>go</i> ”
content	marker	content
“ <i>to</i> ”	“ <i>Kyoto</i> ”	
function	content	

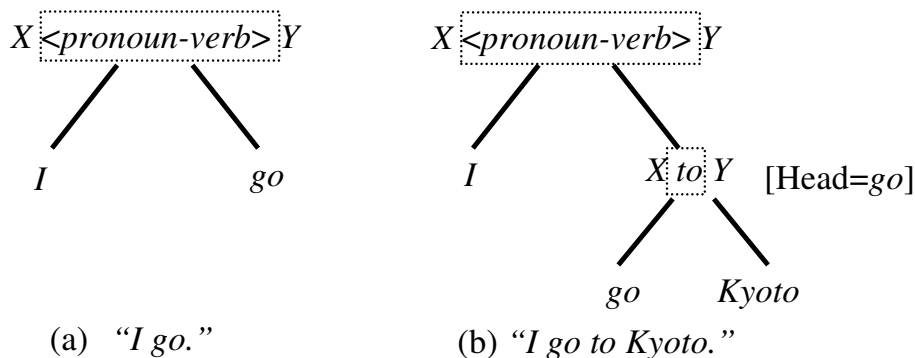
In this example, the original sentence has three content words, “*I*,” “*go*,” and “*Kyoto*.” Since there is no function word between “*I*” and “*go*,” a POS-bigram marker “< *pronoun – verb* >” is inserted to demarcate the boundary.

From the above representation, CBP builds up a tree by using “constituent patterns.” A constituent pattern, *X-b-Y*, represents a partial structure consisting of one boundary marker (i.e., a function word or a POS-bigram marker), *b*, and two content words, *X* and *Y*, surrounding *b*. For example, when *b* is a function word, “*to*,” *X-“to”-Y* is a constituent pattern showing that two content words adjacent to “*to*” (e.g., “*go to Kyoto*”) forms a partial phrase structure. Constituent patterns can be used recursively in such a way that a partial structure identified as a constituent pattern can be treated as one content word in another constituent pattern. Each pattern has one of the constituents as the headword or “head.” The head is regarded as “representative” of the pattern. As shown in Fig. 5.1, the phrase “*go to Kyoto*” is an internal element of another pattern “*X <pron-verb> Y*.” Note that *X* and *Y* have some constraints that preclude them from matching with arbitrary words and phrases.

It may appear that a constituent pattern is similar to the right-hand side of rewriting rules in context-free grammars. Actually, there is a difference in that a constituent pattern always has three elements and that the second element should be a boundary marker.

This procedure is language independent and can be applied to Japanese. In Japanese, function words are often omitted, especially in the case of spontaneous speech. To deal with such cases, boundary markers are inserted into places where function words are omitted. Therefore, the parser can make structures by analyzing only surface patterns. Accordingly, this parsing method is effective for spoken languages.

For example, Figure 5.2(a) shows a parse tree of “*watashi wa Kyoto e iku*,” which means “*I go to Kyoto*” in English. In Japanese spontaneous speech, this sentence may



□ ... Function words or boundary markers.

[Head=] ... Headwords.

Figure 5.1. Structures of (a) "I go" and (b) "I go to Kyoto." (a) The marker, " $\langle \text{pronoun-verb} \rangle$," is inserted between "I" and "go," and the structure is a pattern " $X \langle \text{pronoun-verb} \rangle Y$." (b) An example structure is made by patterns, " $X \langle \text{pronoun-verb} \rangle Y$ " and " $X \text{ to } Y$." "Head" means a headword that is representative of a sub-tree.

become "watashi, Kyoto e iku." The function word "wa" is removed in this case. To cope with such problems, first, the parser inserts a boundary marker between "watashi" and "Kyoto," like "watashi $\langle \text{pronoun} \rangle$ Kyoto e iku." Second, it extracts a structure like that in Fig. 5.2(b). Therefore, the same structure as (a) is obtained even if some function words are omitted.

5.2.2 Parsing procedure

The parsing procedure receives a sentence that is segmented into words with part-of-speech tags and outputs a phrase structure tree. The procedure consists of two major steps: preprocessing and main processing.

The preprocessing step is responsible for the following two tasks: the first task is to convert words in the semantic point of view. This includes combining compound words, dividing a word with complex meaning into a sequence of more primitive words, and normalizing morphological or lexical variants into a 'standard' form. These are done by using rules and dictionaries originally designed for machine translation.

The second task is to insert boundary markers by referring to a set of rules currently

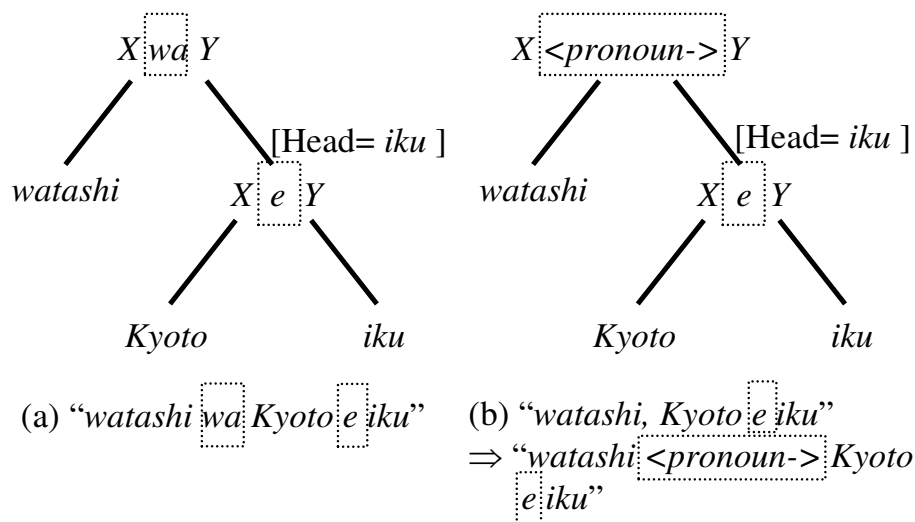


Figure 5.2. Structures of example Japanese sentences. These structures are constructed by (a) patterns “ $X wa Y$ ” and “ $X e Y$ ” and (b) patterns “ $X <pronoun-> Y$ ” and “ $X e Y$,” respectively.

manually written. They are dependent on POS tags for both preceding and succeeding words.

The resulting word sequences are called “modified word sequences.”

The main processing step is responsible for building a tree from analyzed sentences including boundary markers. TDMT employs a bottom-up chart parser for this purpose. Parsing ambiguity is resolved by using semantic scores of constituent patterns [38].


5.3. Proposed Language Models

5.3.1 Modified word trigram models

A modified word trigram model is a word trigram model of a preprocessed sentence, as shown in Figure 5.3. Since boundary markers in preprocessed sentences represent phrase boundaries, we can expect the modified trigram models to contain better structural information than the standard word n-gram models. In addition, the modified trigram models can be considered as variable-length-unit n-gram models because units

Preprocessed sentence with boundary markers

I *<pronoun-verb>* *go* *to* *Kyoto.*

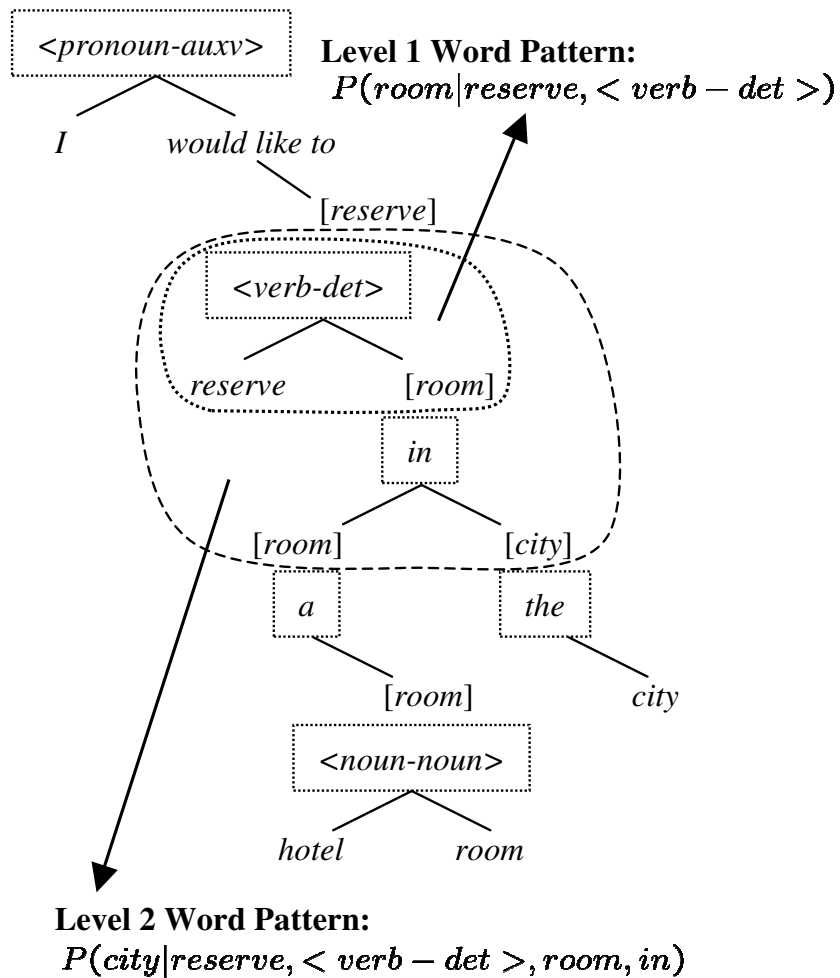
 Extract trigram models.

Modified word trigram models

{
<s> *I* *<pronoun-verb>*
I *<pronoun-verb>* *go*
<pronoun-verb> *go* *to*
go *to* *Kyoto*
to *Kyoto* *</s>*
}

- Function words or boundary markers
- <s>* Sentence start
- </s>* Sentence end

Figure 5.3. Modified word trigram models.



Input sentence with markers:

"I, <pronoun-auxv>, would like to, reserve, <verb-det>, a, hotel, <noun-noun>, room, in, the, city."

Original sentence:

"I, would, like, to, reserve, a, hotel, room, in, the, city."

Figure 5.4. Extraction of pattern models from a parse tree.

of words are modified by the rules included in the preprocessing of this parser.

5.3.2 Word pattern models

Each sub-tree in these parse trees usually includes two content words and either a function word, or a POS-bigram marker. Sub-trees that are not leaf sub-trees include headwords instead of content words. We define these words in a sub-tree as a set of word pattern features. If these patterns are used as models, they can represent not only phrasal constraints, but also neighboring phrasal constraints. Because headwords are keywords among lower sub-trees, patterns extracted from sub-trees connect headwords to other words in the same sub-trees even if these words are not contiguous with each other. Therefore, these pattern models can be considered as trigger models constrained by parse trees.

Figure 5.4 shows how word patterns are extracted from an English sentence. A level 1 word pattern is defined as a set of words included in one sub-tree. The smaller circled sub-tree in the center of the figure includes three words, “*reserve*,” “< *verb – det* >,” and “*room*.” The probability of this word set is defined as $P(\textit{room}|\textit{reserve}, < \textit{verb} - \textit{det} >)$. We call it a level 1 word pattern model. Since word pattern models include headwords, which in this example are “*reserve*,” and “*room*,” they can represent relations among neighboring phrases, that is, long-distance relations among words in original sentences. Word orders are kept in pattern models.

A level 2 word pattern is defined as a set of words included in two contiguous sub-trees. $P(\textit{city}|\textit{reserve}, < \textit{verb} - \textit{det} >, \textit{room}, \textit{in})$ can be extracted from the larger circled sub-trees as a level 2 word pattern. Since level 2 word patterns have longer word sequences than level 1 word patterns, pattern models at level 2 have more constraints than those at level 1.

Deeper level pattern models can be defined in the same manner. These pattern models can represent relations among more words. Therefore, these pattern models can be considered as variable-length n-gram models including long-distance dependency. Furthermore, we define level N pattern models as including level $N - 1$ pattern models.

5.3.3 Formulation

The probability of a modified trigram model is defined as the same as that of a conventional trigram model.

The probability of a word pattern model is defined as an n-gram probability,

$$P(h_i|h_{i-N+1}, \dots, h_{i-1}) = \frac{C(h_{i-N+1}, \dots, h_{i-1}, h_i)}{\sum_{h_i} C(h_{i-N+1}, \dots, h_{i-1}, h_i)}, \quad (5.1)$$

where $C(h_{i-N+1}, \dots, h_i)$ is the frequency of word pattern $\{h_{i-N+1}, \dots, h_i\}$ with N words in the training data. These words are not necessarily contiguous. The word h_i means any kind of word, e.g., a content word, a headword, a function word, or a boundary marker. The numerator is the frequency of the target pattern and the denominator is the frequency of patterns having the same left context as the target pattern.

One word in a sentence usually has some word patterns included in the word pattern models extracted from the training data. We define the pattern probability for the current word w_i as follows:

$$P_{pattern}(w_i|\mathbf{W}_1^{i-1}) = \frac{1}{N_p} \sum_{\mathbf{w}_p} P_{pattern}(w_i|\mathbf{w}_p), \quad (5.2)$$

where \mathbf{w}_p denotes a word pattern included in the word history and $\mathbf{W}_1^{i-1} = \{w_1, w_2, \dots, w_{i-1}\}$. N_p is the number of word patterns for the current word. This equation means an average of probabilities.

Next, we define a probability combined with an n-gram probability with boundary markers and a word pattern probability:

$$P(w_i|\mathbf{W}_1^{i-1}) = \lambda \cdot P_{pattern}(w_i|\mathbf{W}_1^{i-1}) + (1 - \lambda) \cdot P_{mod-3gram}(w_i|w_{i-2}, w_{i-1}), \quad (5.3)$$

where $P_{pattern}$ is a probability of word patterns defined in Equation (5.2), $P_{mod-3gram}$ is a modified word trigram probability, and λ is a constant. The word pattern models are used only when some pattern models can be extracted from sub-trees including a target word in a parse tree of each candidate. Although pattern models can be applied to patterns extracted from word histories without constraints of parse trees, some pattern models might be irrelevant to target contexts. Therefore, we use constraints of parse trees. Furthermore, in this paper, λ for each model was set to the optimal value obtained from experimental results.

5.3.4 Training and recognition schemes

Figure 5.5 shows training and recognition schemes. For training, first, word bigram models are extracted from the training database. Second, modified word trigram models are extracted from the modified training database analyzed by the preprocessing of the parser. Finally, word pattern models are extracted from the parse tree database created by the parser.

For recognition, word bigram models are used in the first pass search. The N-best candidates of the first pass are analyzed and parsed before the second pass search. The analyzed candidates are used for modified word trigram model, and the parse trees of candidates are used for word pattern models. These language models in the second pass search rescore the original candidate sentences. The final candidates are the rescored original sentences. The modified sentences are used in only internal calculations of the second pass search. Therefore, the rescored original sentences were evaluated in the next experiments.

5.4. Experiments

5.4.1 Experimental conditions

Experiments were carried out using the proposed language models for the Japanese travel dialogues in the ATR spontaneous speech database[33]. First, we used “The Travel Arrangement Task (TRA)” for this evaluation. For analysis conditions, the sampling frequency was 16 kHz, the frame length was 20 ms, and the frame shift was 10 ms. 12 order MFCC, 12 order Δ MFCC, log power, and Δ log power were used as feature parameters. Cepstrum Mean Subtraction was also used. 407 dialogues including about 5 hours of speech were used for training acoustic models. For acoustic models, gender dependent HMMs with 1,403 states were made by the ML-SSS algorithm[7]. The male model had five Gaussian mixtures per state and the female model had 15 mixtures. Table 5.1 shows the amount of training data for language models and the amount of test TRA data. It also includes the number of sentences before and after preprocessing. There were some sentences that could not be parsed. Therefore, the number of sentences were decreased. To compare our proposed models, traditional word trigram models were applied to the second pass search. Table 5.2 shows the total

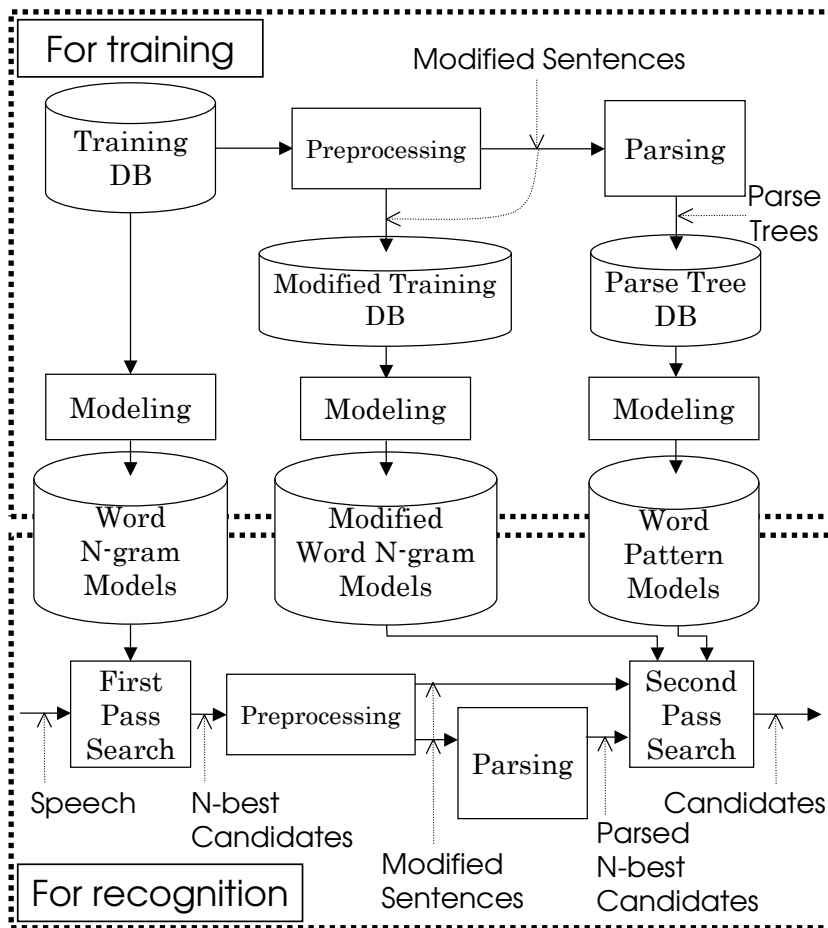


Figure 5.5. Training and recognition schemes.

Table 5.1. Text data

	Training	Test (TRA)
#dialogues	7,195	42
#sentences before processing	1.20×10^5 (1.35×10^6)	551 (5,221)
#sentences after processing	0.88×10^5 (1.13×10^6 [0.16×10^5])	551 (4,796 [674])
Vocabulary size	16,355	
#kinds of markers	98	

The values in () are the number of words included in the sentences. The values in [] are only the number of boundary markers.

number of entries for each model. These cutoff values of pattern models were decided by the results of experiments. In the same way, almost optimal λ s were used in the following results. Since models deeper than level 7 needed much more memory, we could only use level 1 to 7 word pattern models in the experiments.

5.4.2 Evaluation by perplexity

Table 5.3 shows the word and sentence perplexity for the TRA task for each language model. The word perplexity is the perplexity per word, and the sentence perplexity is the perplexity per sentence. Since word units are changed for the model (3) and model (4)–(11), the word perplexity values of the models (3)–(11) cannot be compared to those of both bigram models (1) and trigram models (2). Additionally, the sentence perplexity values between models (1)–(3) before the marker insertion and models (4)–(11) after that cannot be compared. However, they give some indications of improvements. Modified trigram models without markers (3) had large perplexity. Since the preprocessing of this parser modifies words by knowledge, analyzed sentences include more kinds of trigrams with lower frequencies. However, after the insertion of boundary markers, modified trigram models with markers (4) can improve perplexity because markers work like classifiers and frequencies of trigram models are increased. Furthermore, pattern models improved perplexity slightly.

Table 5.2. Total number of entries

	#entries	Cutoff
(1) Word bigram models	3.3×10^4	2
(2) Word trigram models	1.1×10^5	1
(3) Modified trigram models without markers	9.8×10^4	1
(4) Modified trigram models with markers	1.1×10^5	1
(5) (4) + Pattern (level 1)	1.6×10^4	4
(6) (4) + Pattern (level 2)	1.8×10^4	7
(7) (4) + Pattern (level 3)	6.4×10^3	24
(8) (4) + Pattern (level 4)	8.2×10^3	24
(9) (4) + Pattern (level 5)	1.4×10^4	20
(10) (4) + Pattern (level 6)	1.4×10^4	21
(11) (4) + Pattern (level 7)	1.3×10^4	26

5.4.3 Evaluation by word accuracy

Table 5.4 shows the word accuracy for each language model. The 100-best sentences of results by bigram models were rescored by another models. The 100-best accumulated word accuracy was 92.37%. Modified trigram models both with and without markers improved performance because these models use longer word units, where multiple words are treated as one word. Modified word trigram models obtained significant differences with less than 1% risk rate for word trigram models by the Wilcoxon signed rank test. The improvements by word pattern models were slight against modified trigram models. Our pattern models obtained significant differences with 12.5% risk rate by the signed test for each word. However, the deeper level models obtained higher accuracy. This shows that pattern models can improve performance if target sentences have the same patterns as those included in the pattern models. These models may obtain significant differences with lower risk rate if the large amount of data can be used for evaluation. This task does not have particularly long utterances. The average length of utterances in the evaluation data is 9.6 words per sentence. Pattern models may be suitable for longer sentences. We will investigate the characteristics of pattern

Table 5.3. Perplexity for the TRA task

Language models	Word perplexity	Sentence perplexity
(1) Word bigram models	27.19	290.2
(2) Word trigram models	19.73	205.9
(3) Modified trigram models without markers	27.73	235.2
(4) Modified trigram models with markers	17.49	168.8
(5) (4) + Pattern (level 1)	17.41	168.1
(6) (4) + Pattern (level 2)	17.43	168.2
(7) (4) + Pattern (level 3)	17.44	168.3
(8) (4) + Pattern (level 4)	17.44	168.4
(9) (4) + Pattern (level 5)	17.44	168.3
(10) (4) + Pattern (level 6)	17.44	168.3
(11) (4) + Pattern (level 7)	17.44	168.4

models in the next section.

5.5. Evaluation of Pattern Models

In the previous experiments, pattern models could obtain only slight improvements. Word pattern models are similar to trigram models when applied to short sentences because they have only small structures with a few sub-trees. Therefore, pattern models can be expected to be suitable for long sentences that have a lot of sub-trees. For evaluation of pattern models, we used another task data set whose utterances were much longer than those of the TRA task.

5.5.1 Performance in the APP task

For the evaluation, we used “The Appointment Scheduling Task (APP),” which contains dialogues to schedule a meeting and to show the way to meeting places. Table 5.5

Table 5.4. Word accuracy for the TRA task

Language models	Word accuracy [%]
(1) Word bigram models	83.64
(2) Word trigram models	85.49
(3) Modified trigram models without markers	86.56
(4) Modified trigram models with markers	86.64
(5) (4) + Pattern (level 1)	86.64
(6) (4) + Pattern (level 2)	86.68
(7) (4) + Pattern (level 3)	86.70
(8) (4) + Pattern (level 4)	86.70
(9) (4) + Pattern (level 5)	86.72
(10) (4) + Pattern (level 6)	86.74
(11) (4) + Pattern (level 7)	86.74

Table 5.5. APP Test Set

#dialogues	20
#sentences before processing	185 (3,511)
#sentences after processing	185 (3,426 [474])

lists text information of the test set. The average length of sentences for the APP task was 13.9 words. The acoustic models were trained by using the data of the APP task including 3,631 dialogues, about 51 hours of speech. The same language models were used as those in the previous section. The 6.7% of sentences included in the training data belonged to the APP task. The parser that we used for extracting models was not configured for this task.

Table 5.6 shows perplexity for the APP task and Table 5.7 shows word accuracy. The 300-best sentences by bigram models were used for rescoring by another models and the 300-best accumulated word accuracy was 85.37%. Since our parser was designed for the TRA task, our language models which employs parse trees are not well suited for the APP task compared with trigram models. Therefore, the absolute

Table 5.6. Perplexity for the APP task

Language models	Perplexity
(1) Word bigram models	30.96
(2) Word trigram models	20.94
(4) Modified trigram models with markers	21.78
(11) (4) + Pattern (level 7)	21.73

Table 5.7. Word accuracy for the APP task

Language models	Word accuracy [%]
(1) Word bigram models	74.96
(2) Word trigram models	78.57
(4) Modified trigram models with markers	76.92
(11) (4) + Pattern (level 7)	77.07

performances were not so good and conventional word trigram models obtained better performances than our proposed method. However, the relative performance between modified trigram models and pattern models for the APP task is similar to that for the TRA task. Therefore, it is enough to evaluate the relative performance between these models by using the APP task.

5.5.2 Evaluation for the length of sentences

We evaluated the performance of pattern models by comparing it to that of the modified word trigram models with markers for the different lengths of sentences.

Perplexity improvement rates

Figures 5.6 and 5.7 show perplexity improvement rates for the TRA task and the APP task, respectively. The horizontal axes show the number of words in one sentence. The left vertical axes show the perplexity improvement rates from modified trigram models with markers to level 7 word pattern models. The right vertical axes show the sentence

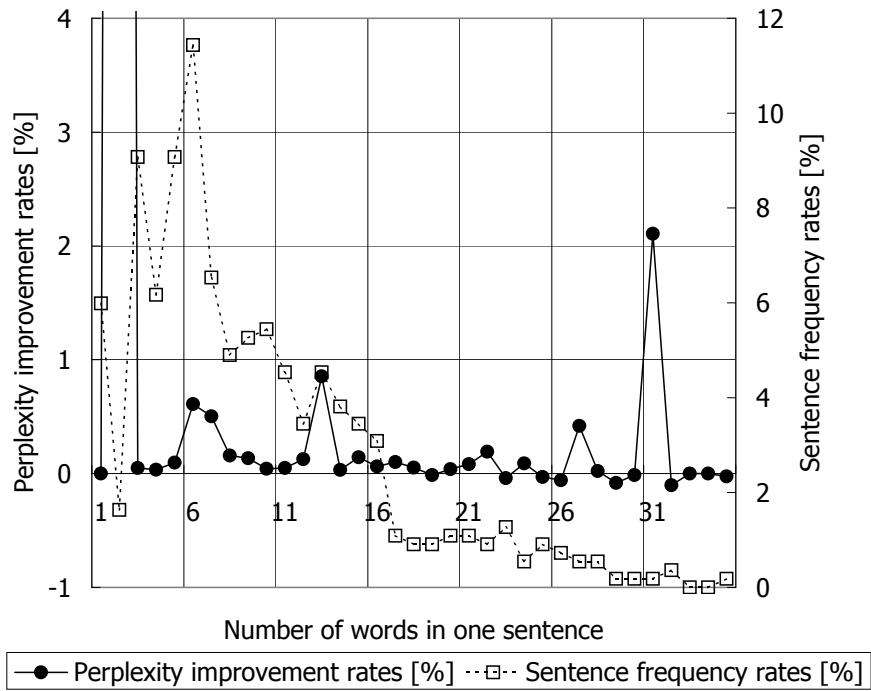


Figure 5.6. Perplexity improvement rates for the TRA task.

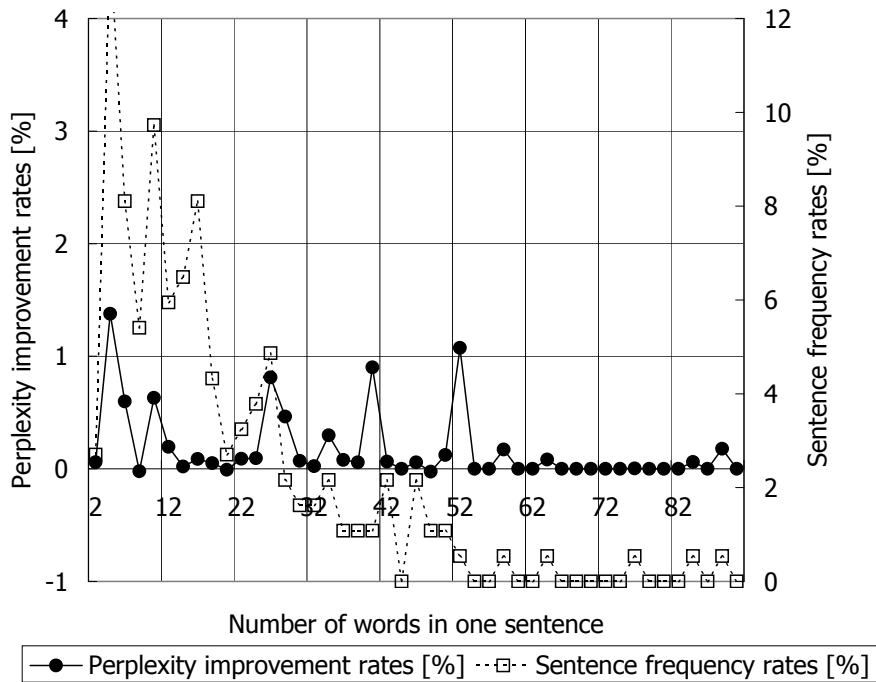


Figure 5.7. Perplexity improvement rates for the APP task.

frequency rates. Some large improvements were obtained for two- to four-word-length sentences for example, two-word-length sentences in the TRA task obtained a 43.5% perplexity improvement rate. Since a bigram marker is dependent on POS tags of both its preceding and succeeding words, probabilities of contiguous words around markers usually become higher after markers are inserted. Additionally, pattern models can give much higher probabilities if some patterns in evaluated sentences are included in pattern models. Apart from this, small improvements were obtained for five- to 52-word-length sentences by the pattern models.

Error reduction rates

Figures 5.8 and 5.9 show error reduction rates for the TRA task and the APP task, respectively. The left vertical axes show the error reduction rates and the other axes are the same as for the perplexity improvement rates. Some improvements were obtained for 10- to 34-word-length sentences by the pattern models. However, some decline in performance was observed for over 34-word-length sentences. Therefore, these results exactly show that the pattern models are more effective for long sentences, but their models have some upper limits related to sentence length. For these results, we performed the signed test for each word, and obtained significant differences with 6% risk rate for 10- to 34-word-length sentences of both the TRA task and the APP task. Using both of these data, we further obtained the significant difference with 1% risk rate. Therefore, it can be considered that longer patterns can improve performance and improvement rates are dependent on sentence lengths.

5.6. Evaluation of ASR Using MDL-SSS and Pattern Language Model

In this dissertation, we propose two types of topology training for acoustic modeling and the word pattern models extracted from parse trees for language modeling. In this section, we will evaluate performance by combination the proposed methods, the MDL-SSS algorithm and the word pattern language models.

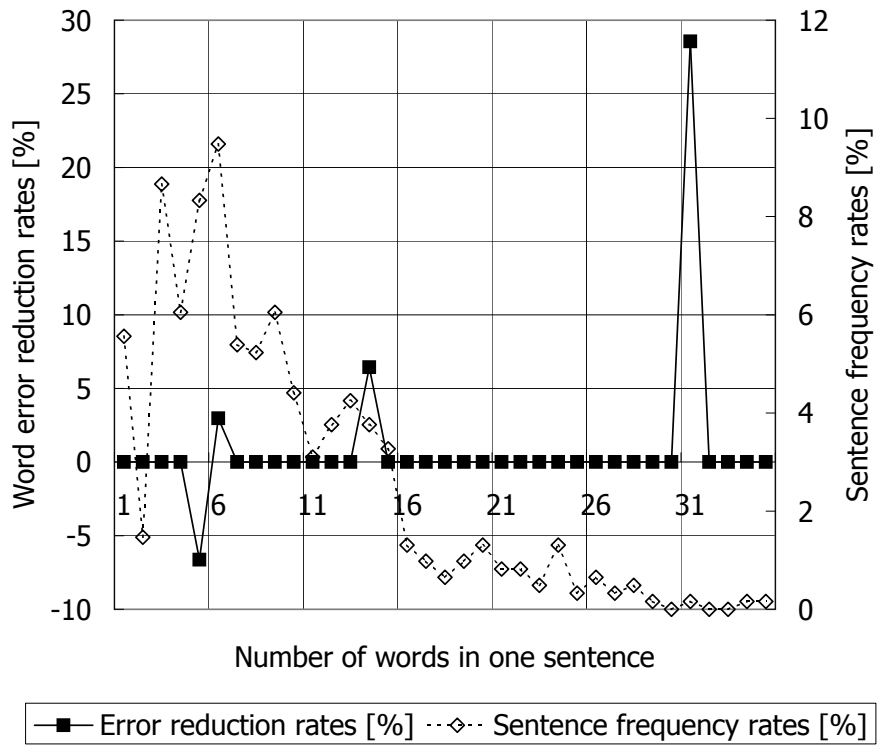


Figure 5.8. Error reduction rates for the TRA task.

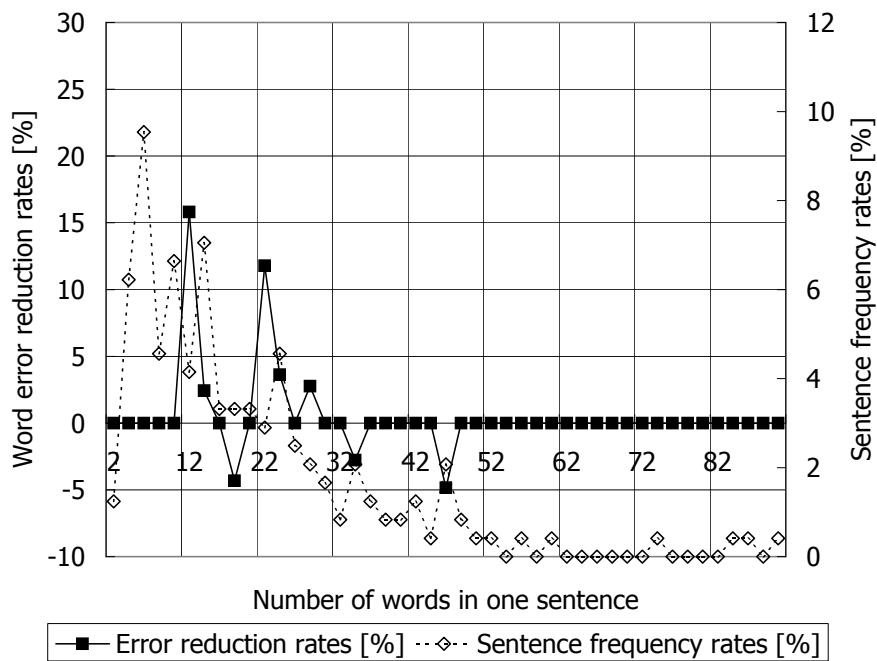


Figure 5.9. Error reduction rates for the APP task.

5.6.1 Evaluation

To evaluate our proposed method, we tested the following four combinations:

1. Acoustic models by ML-SSS and multi-class composite bigram models.
 - (a) 1 and rescoring by word trigram models.
 - (b) 1 and rescoring by word pattern models.
2. Acoustic models by MDL-SSS and multi-class composite bigram models.
 - (a) 2 and rescoring by word trigram models.
 - (b) 2 and rescoring by word pattern models.

These combinations were evaluated in the same manner as in Chapter 3.

For acoustic models, we only evaluated the MDL-SSS algorithm as a proposed method, and the ML-SSS algorithm as a baseline method. Although the VB-SSS algorithm is more effective than the MDL-SSS algorithm for small amounts of training data, our current program and algorithm of the VB-SSS have difficulty with large amounts of training data. On the other hand, the MDL-SSS algorithm can also work well for the training data used in this evaluation, which includes 30-hour speech data. Additionally, we expect that both the VB-SSS and the MDL-SSS can obtain comparable performance for sufficient amounts of training data.

The multi-class composite bigram model has been proposed as the language model that can obtain almost the same performance as that by a word trigram model [34]; apparently, this language model can obtain even better performance than that by a word bigram model. For this reason, we used the multi-class composite bigram models as the baseline language models. These word pattern models include the modified word trigram models and the word pattern models extracted from parse trees.

5.6.2 Experimental Conditions

These experimental conditions are almost the same as those of Section 3.3.1. For the test set, we used 42 one-side dialogs extracted from the Japanese TRA database. This evaluation set includes about 5,000 words.

Table 5.8. Perplexity for the TRA task by several language models used in this chapter

Language models	Word perplexity
(1) Multi-class composite bigram models	21.05
(2) Word trigram models	19.73
(3) Modified trigram models with markers	17.49
(4) (3) + Pattern (level 7)	17.44

First, we will describe the acoustic models. We used 26 kinds of phoneme and one silence. A silence model with three states was built separately from the phoneme models. The phoneme models were created in the same manner as in Section 3.3.1. In this chapter, acoustic models were generated by the ML-SSS, or MDL-SSS, and compared. For the ML-SSS, we constructed three kinds of model with 1,000, 1,400, or 2,100 states. For the MDL-SSS, we made the model generated by using $C_c = 2$, $C_t = 20$. For the acoustic training data, both the Japanese TRA database and the BLA database were used. The analysis conditions were a frame length of 20 ms and a frame shift of 10 ms. Twelve-order MFCC, 12-order Δ MFCC, and Δ log power were used as feature parameters. The cepstrum mean subtraction was applied to each utterance.

Next, we will describe language models used in these experiments. For the language training set, we employed 7,195 one-side dialogues which included 1.6×10^6 words. Multi-class composite bigram models [34] with 700 classes and 5,216 composite words were used as the language models in the first pass instead of conventional word bigram models. In the second pass, our proposed method, the word pattern models extracted from parse trees were used for rescoring. The word pattern models (level 7) were the same as those in Section 5.4. The vocabulary size in the set was 27,398, and 5,216 composite words were added to the vocabulary.

5.6.3 Experimental Results

Table 5.8 shows perplexity for each language model used in this chapter, although these values have already been presented in the preceding chapters. The multi-class composite bigram models obtained slightly greater perplexity than the word trigram

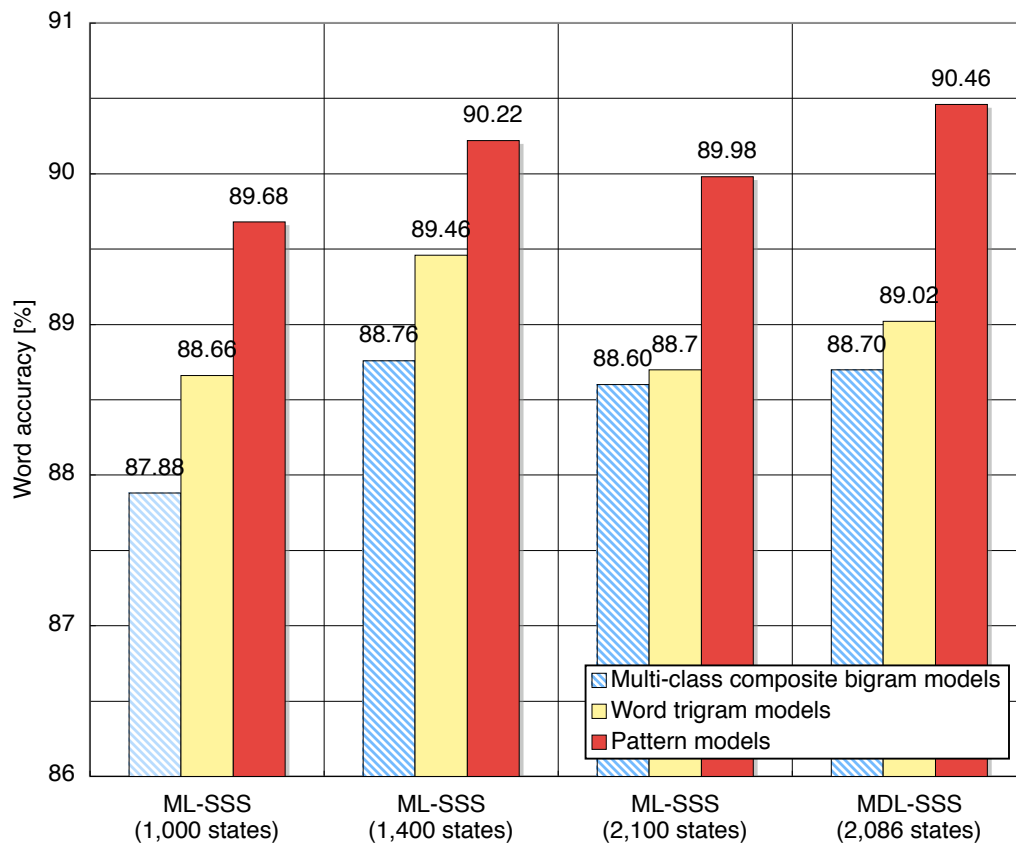


Figure 5.10. Performance by the ML-SSS and the MDL-SSS with the multi-class composite bigram models, or rescoring by the word pattern models.

models, although the multi-class composite models have some advantages for recognition. The multi-class means “multi-dimensional,” and the multi-class models represent the left- and the right-context dependency. Therefore, this method can obtain better performance than conventional single-class models. Furthermore, like the composite models, this model introduces higher order word n-gram models by regarding frequent-variable length word sequences. These word sequences are added to a lexicon as new entries, and those probabilities are defined by using multi-class models. Since the increase of parameters only corresponds to unigrams of word succession, this composite n-gram models can maintain a compact model size.

The perplexity by the modified-word trigram models was better than that for the original word trigram models because the modified-word trigram models include word normalization, exact segmentations, and other knowledge produced manually. The word pattern models obtained almost the same perplexity as the modified-word trigram models. The average frequency of applying word pattern models was 2.6 per utterance. Since the average length of utterances was 9.6 words per utterance for this task, the word pattern models were not so effective.

Figure 5.10 shows experimental results. For acoustic modeling, the ML-SSS with 1,400 states and the MDL-SSS with $C_c = 2, C_t = 20$ obtained almost the same performance when the multi-class composite bigram model was used as a language model in the first pass. For rescoring by word trigram models, the ML-SSS with 1,400 states obtained slightly better performance than that by the MDL-SSS. However, the MDL-SSS using $C_c = 2, C_t = 20$ obtained slightly better performance without any other experiments by using the word pattern models. Although the ML-SSS can obtain almost the same performance, the number of states should be selected carefully as we described in Chapter 3. For language modeling, the word pattern models including modified-word trigram models obtained much better performance than that by the word trigram rescoring. The absolute improvement rate was 1.7% from the baseline, that is, the system using the ML-SSS with 1,400 states and the multi-class composite bigram models. The relative error reduction rate was 15%. The score of 90.46% obtained by the MDL-SSS and the word pattern models is the best word accuracy in several evaluations of this task from ATR-ITL for more than five years.

Additionally, we constructed an another acoustic model by the MDL-SSS algorithm from the larger training database. This training data includes TRA, BLA, and

APP databases, giving a total length of utterances of about 85 hours, which is 2.8 times the amount of training data used in this section. The APP database includes utterances by about 4,000 persons from all over in Japan. Many years and a much money were needed to collect the utterances and to make transcriptions of them.

Using both this acoustic model and multi-class composite bigram models, we obtained a 90.34% word accuracy rate for the same evaluation data. This performance is almost the same as the best performance in this section, 90.46%, indicating that it is much more important to find better ways to extract good features from existing databases and to create models, although increasing the amount of training data is still important to improve performance.

5.7. Summary

We proposed two new types of language model to represent phrasal structures by patterns extracted from parse trees. First, modified word trigram models were proposed that used words modified by the knowledge of the preprocessing for parsing. These models can represent structures of phrases by using a few words including function words, or bigram markers that denote word boundaries. Second, we proposed word pattern models extracted from parse trees. These models can represent phrasal structures and much longer word dependency than trigram models. Experimental results demonstrated that modified trigram models were effective and that pattern models could improve performances slightly. Furthermore, additional results showed that pattern models were more effective than trigram models for long sentences.

Finally, our proposed method, which includes both the MDL-SSS and the word pattern model, was compared with the conventional method. The word pattern language model obtained much better performance than the multi-class composite bigram models. The model by the MDL-SSS was automatically obtained by using optimal parameters, and this model performed best in this evaluation. On the other hand, for the ML-SSS, several models should be evaluated to find the best model. Thus, the MDL-SSS algorithm with the word pattern models can automatically obtain better performance than the conventional method since these proposed methods can extract more efficient models.

Chapter 6

Conclusion

This dissertation presented two new methods for automatic topology training of acoustic modeling and one new method for language modeling.

First, this dissertation addressed automatic generation of acoustic models, especially criteria for splitting and stopping. To create shared-state HMMs, the Maximum Likelihood (ML) criterion is widely used, for example, in the Decision Tree Clustering [5], or in the Successive State Splitting (SSS) algorithm [6][7]. Unfortunately, the ML criterion suffers the over-fitting problem: likelihood usually increases, when the parameters increase. Therefore, it is difficult to stop splitting and to find the best model automatically. Information criteria have been used for this problem, which is referred to as “model selection.” Although the Minimum Description Length (MDL) Criterion has already been applied to the Decision Tree Clustering to create context-dependent HMMs [9][10], this clustering method cannot deal with the number of states for temporal direction. It can only create models with the same number of state lengths.

To create non-uniform, context-dependent HMM topologies automatically, we proposed the SSS algorithm based on the MDL criterion in Chapter 3. The ML-based SSS algorithm requires both the total number of states and the maximum length of states in each triphone as stop criteria. Although our proposed method, the MDL-SSS algorithm, also needs two scaling factors, we confirmed that there are almost optimal values and that this method can create suitable models by using these values.

Second, in Chapter 4, we proposed an SSS algorithm based on the Variational Bayesian approach. Although the MDL-SSS algorithm works well for large amounts of training data, the MDL criterion cannot evaluate HMM structures precisely, espe-

cially for small amounts of training data. The Variational Bayesian approach, on the other hand, can evaluate complicated models like HMMs more precisely than usual information criteria. We applied the VB approach to the SSS algorithm for automatic generation of acoustic models. We defined prior and posterior probabilities for the SSS algorithm, and derived the criterion as splitting and stopping, and the VB-SSS algorithm obtained more efficient models than those by the ML-SSS algorithm. We also applied the VB approach to create Gaussian mixture HMMs for topologies obtained by the VB-SSS algorithm. This method automatically obtained smaller models than those by the method based on ML estimation.

Changing the subject to language modeling, although word n-gram models are widely used, they can only represent local constraints; sentences usually have partial relations among words. To utilize such sentence structures, the Structured Language Model (SLM) has been proposed in [19][20]. This model uses a stochastic parser as one of the language models, and it requires a large amount of parse trees and is very strict on sentence structure. To improve language models at the point of model structures by employing sentence structure, in Chapter 5, we proposed word pattern models extracted from parse trees. The proposed method includes modified word trigram models that used words modified by the knowledge of the preprocessing for parsing, and word patterns extracted from parse trees. The modified word trigram models can represent structures of phrases by using a few words including function words, or bigram markers that denote word boundaries. The word pattern models extracted from parse trees can represent phrasal structures and much longer word dependency than trigram models. Experimental results demonstrated that modified trigram models were effective and that pattern models could marginally improve performances. Furthermore, the word pattern models were more effective for long sentences than trigram models.

Additionally, in Section 5.6, we evaluated a combinations of our proposed methods, the MDL-SSS algorithm and the word pattern models. The word pattern language model obtained much better performance than the multi-class composite bigram models as one example of a conventional method. The model by the MDL-SSS was automatically obtained by using optimal parameters, and this model performed best in this evaluation. In fact, we obtained better performance by using both the MDL-SSS algorithm and the word pattern models.

Here, we will provide suggestions for future research related to issues discussed in this dissertation.

For automatic generation method of acoustic models, in this thesis we only proposed topology training methods to create single Gaussian HMMs and a method to increase mixture components after topology training. Methods that can automatically create Gaussian mixture HMMs are required, since conventional topology training methods can only create single Gaussian HMMs. It is difficult to find the best combination of both the number of states and the number of mixture components automatically. This may well prove to be a challenging issue.

We only focused on standard HMMs with phoneme-contextual dependency and temporal duration dependency, and some novel methods are still needed to create models automatically if one makes a new type of model that includes some new features. These days, many types of Bayesian network are applied to acoustic models, and such models also need automatic generation methods.

There is a large variety of speech corpora, for example, phonetically balanced sentences, speech read from newspapers, conversational speech, and lecture speech, though acoustic models are usually created for specific tasks from the same database. To utilize many kinds of database sufficiently, automatic generation methods of acoustic models are needed. Such methods should be able to extract a lot of acoustical features from databases automatically.

Our proposed methods only aim at acoustic modeling, but similar methods may be developed for language modeling. A method to find the best combination both of acoustic models and language models may be possible in future.

Additionally, we applied the VB approach to topology training. The VB approach was used for the split criterion, the stop criterion, and the parameter estimation method. Models derived by the VB approach are those based on the Bayesian learning, and such models should be investigated in greater depth. Also, the VB approach is more effective for small amounts of data than the ML estimation. Therefore, this approach can be used for speaker adaptation like the method proposed by Watanabe et al. [39]. The VB approach's value is that it can be applied to many kinds of adaptation method.

For language modeling, we proposed the modified word trigram models and the word pattern models extracted from parse trees. They can represent both phrasal structures and long-distance dependencies. However, SLMs require many parse trees, while

our proposed method also requires knowledge constructed manually. In our labs., that knowledge has been produced for machine translation for many years, but it is strongly dependent on the ATR travel-arrangement task. SLMs need some common databases that everyone can use, or some method that can extract such information automatically, and such databases require a lot of human power and time. For this reason, many groups should collaborate to produce some common databases that include a lot of parse trees for large amounts of text, and should open them for everyone to share, especially since computer processing power continually increasing, as is the amount of storage available. Therefore, much larger databases can be used and are certainly needed.

The more advanced language models should deal with local relations among words, distance relations, sentence structures, semantic relations, and so on. Additionally, some methods are needed to combine many kinds of language model. Since computer resources are constantly improving, models should use many language features as possible.

Appendix A

Definitions for the VB-SSS Algorithm

A.1. Objective Function of the VB-SSS Algorithm

The objective function can be written from Eq. (4.11) as follows.

$$\mathcal{F}_m = \mathcal{F}_{12} + \sum_{i=1}^{N_s} \left\{ \mathcal{F}_{341}(\{a_{ij}\}_{j=1}^{N_a}) + \mathcal{F}_{342}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) + \mathcal{F}_{343}(\boldsymbol{\Sigma}_i) \right\}, \quad (\text{A.1})$$

where \mathcal{F}_{12} is the combination of the first term and the second term of Eq. (4.11). \mathcal{F}_{341} , \mathcal{F}_{342} , and \mathcal{F}_{343} are derived from the third term and the fourth term. For the state i , $\mathcal{F}_{341}(\{a_{ij}\}_{j=1}^{N_a})$, $\mathcal{F}_{342}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, and $\mathcal{F}_{343}(\boldsymbol{\Sigma}_i)$ are the terms related to the transition probabilities, both the mean vector and the covariance matrix, and the covariance matrix, respectively.

The first term and the second term can be calculated as follows.

$$\begin{aligned} \mathcal{F}_{12} &= \int q(\mathbf{Z})q(\Theta) \ln \prod_{t=1}^T p(\mathbf{o}, z_t | \Theta) d\Theta - \int q(\mathbf{Z}) \ln q(\mathbf{Z}) d\mathbf{Z} \\ &= \sum_{t=1}^T \int q(\mathbf{Z}) d\mathbf{Z} \{ \langle \ln p(\mathbf{o}, z_t | \Theta) \rangle_{q(\Theta)} - \ln q(z_t) \} \\ &= \sum_{t=1}^T \ln \sum_{i=1}^{N_s} \sum_{j=1}^{N_a} \exp(\gamma'_{ij}). \end{aligned} \quad (\text{A.2})$$

The distribution of parameters is factorized as

$p(\theta) = \prod_{i=1}^{N_s} p(\{a_{ij}\}_{j=1}^{N_a})p(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)p(\boldsymbol{\Sigma}_i)$. Therefore, the third term is

$$\begin{aligned}\mathcal{F}_3 &= \int q(\Theta) \ln q(\Theta) d\Theta \\ &= \sum_{i=1}^{N_s} \left\{ \int q(\{a_{ij}\}_{j=1}^{N_a}) \ln q(\{a_{ij}\}_{j=1}^{N_a}) d\{a_{ij}\}_{j=1}^{N_a} \right. \\ &\quad \left. + \int q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)q(\boldsymbol{\Sigma}_i) \ln q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i) d\boldsymbol{\mu}_i d\boldsymbol{\Sigma}_i + \int q(\boldsymbol{\Sigma}_i) \ln q(\boldsymbol{\Sigma}_i) d\boldsymbol{\Sigma}_i \right\},\end{aligned}\quad (\text{A.3})$$

and the fourth term is

$$\begin{aligned}\mathcal{F}_4 &= \int q(\Theta) \ln p(\Theta) d\Theta \\ &= \sum_{i=1}^{N_s} \left\{ \int q(\{a_{ij}\}_{j=1}^{N_a}) \ln p(\{a_{ij}\}_{j=1}^{N_a}) d\{a_{ij}\}_{j=1}^{N_a} \right. \\ &\quad \left. + \int q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)q(\boldsymbol{\Sigma}_i) \ln p(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i) d\boldsymbol{\mu}_i d\boldsymbol{\Sigma}_i + \int q(\boldsymbol{\Sigma}_i) \ln p(\boldsymbol{\Sigma}_i) d\boldsymbol{\Sigma}_i \right\}.\end{aligned}\quad (\text{A.4})$$

The first term with the state i related to transition probabilities is as follows.

$$\begin{aligned}\mathcal{F}_{31}(\{a_{ij}\}_{j=1}^{N_a}) &= \int \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \{\phi_{ij}\}_{j=1}^{N_a}) \ln \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \{\phi_{ij}\}_{j=1}^{N_a}) da_{ij} \\ &= \ln \Gamma \left(\sum_{j=1}^{N_a} \phi_{ij} \right) - \sum_{j=1}^{N_a} \ln \Gamma(\phi_{ij}) + \sum_{j=1}^{N_a} (\phi_{ij} - 1) \left\{ \Psi(\phi_{ij}) - \Psi \left(\sum_{j=1}^{N_a} \phi_{ij} \right) \right\},\end{aligned}\quad (\text{A.5})$$

$$\begin{aligned}\mathcal{F}_{41}(\{a_{ij}\}_{j=1}^{N_a}) &= \int \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \{\phi_{ij}\}_{j=1}^{N_a}) \ln \mathcal{D}(\{a_{ij}\}_{j=1}^{N_a}; \phi_0) da_{ij} \\ &= \ln \Gamma(N_a \phi_0) - N_a \ln \Gamma(\phi_0) + (\phi_0 - 1) \sum_{j=1}^{N_a} \left\{ \Psi(\phi_{ij}) - \Psi \left(\sum_{j=1}^{N_a} \phi_{ij} \right) \right\}.\end{aligned}\quad (\text{A.6})$$

The free energy related to the transition probabilities is

$$\begin{aligned}\mathcal{F}_{341}(\{a_{ij}\}_{j=1}^{N_a}) &= -\mathcal{F}_{31}(\{a_{ij}\}_{j=1}^{N_a}) + \mathcal{F}_{41}(\{a_{ij}\}_{j=1}^{N_a}) \\ &= \ln \Gamma(N_a \phi_0) - \ln \Gamma \left(\sum_{j=1}^{N_a} \phi_{ij} \right) + \sum_{j=1}^{N_a} \ln \Gamma(\phi_{ij}) - N_a \ln \Gamma(\phi_0) \\ &\quad + \sum_{j=1}^{N_a} (\phi_0 - \phi_{ij}) \left\{ \Psi(\phi_{ij}) - \Psi \left(\sum_{j=1}^{N_a} \phi_{ij} \right) \right\}.\end{aligned}\quad (\text{A.7})$$

The second term in the left side of Eq. (A.3) is

$$\begin{aligned}\mathcal{F}_{32}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) &= \int q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)q(\boldsymbol{\Sigma}_i) \ln q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)d\boldsymbol{\mu}_i d\boldsymbol{\Sigma}_i \\ &= -\frac{D}{2} \ln 2\pi + \frac{1}{2} \ln \xi_i + \frac{D}{2} \Psi\left(\frac{\eta_i}{2}\right) - \frac{1}{2} \sum_{k=1}^D \ln\left(\frac{b_{ik}}{2}\right) - \frac{\eta_i}{2(\eta_i - D - 1)},\end{aligned}\quad (\text{A.8})$$

and the second term in the left side of Eq. (A.4) is

$$\begin{aligned}\mathcal{F}_{42}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) &= \int q(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)q(\boldsymbol{\Sigma}_i) \ln p(\boldsymbol{\mu}_i|\boldsymbol{\Sigma}_i)d\boldsymbol{\mu}_i d\boldsymbol{\Sigma}_i \\ &= -\frac{D}{2} \ln 2\pi + \frac{1}{2} \ln \xi_0 + \frac{D}{2} \Psi\left(\frac{\eta_0}{2}\right) - \frac{1}{2} \sum_{k=1}^D \ln\left(\frac{b_{0k}}{2}\right) - \frac{\eta_0}{2(\eta_0 - D - 1)}.\end{aligned}\quad (\text{A.9})$$

The energy related to both mean vectors and covariance matrices is

$$\begin{aligned}\mathcal{F}_{342}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) &= \frac{1}{2} \ln \frac{\xi_0}{\xi_i} + \frac{D}{2} \left\{ \Psi\left(\frac{\eta_0}{2}\right) - \Psi\left(\frac{\eta_i}{2}\right) \right\} \\ &\quad - \frac{1}{2} \sum_{k=1}^D \ln\left(\frac{b_{0k}}{b_{ik}}\right) - \frac{\eta_0}{2(\eta_0 - D - 1)} + \frac{\eta_i}{2(\eta_i - D - 1)}.\end{aligned}\quad (\text{A.10})$$

The third term in the left side of Eq. (A.3) is

$$\begin{aligned}\mathcal{F}_{33}(\boldsymbol{\Sigma}_i) &= \int q(\boldsymbol{\Sigma}_i) \ln q(\boldsymbol{\Sigma}_i)d\boldsymbol{\Sigma}_i \\ &= \sum_{k=1}^D \int \mathcal{G}(\sigma_{ik}^{-1}; \frac{\eta_i}{2}, \frac{b_{ik}}{2}) \ln \mathcal{G}(\sigma_{ik}^{-1}; \frac{\eta_i}{2}, \frac{b_{ik}}{2}) d\sigma_{ik} \\ &= D \left\{ \left(\frac{\eta_i}{2} - 1 \right) \Psi\left(\frac{\eta_i}{2}\right) - \ln \Gamma\left(\frac{\eta_i}{2}\right) - \frac{\eta_i}{2} \right\} + \sum_{k=1}^D \ln\left(\frac{b_{ik}}{2}\right),\end{aligned}\quad (\text{A.11})$$

and the third term in the left side of Eq. (A.4) is

$$\begin{aligned}\mathcal{F}_{43}(\boldsymbol{\Sigma}_i) &= \int q(\boldsymbol{\Sigma}_i) \ln p(\boldsymbol{\Sigma}_i)d\boldsymbol{\Sigma}_i \\ &= \sum_{k=1}^D \int \mathcal{G}(\sigma_{ik}^{-1}; \frac{\eta_i}{2}, \frac{b_{ik}}{2}) \ln \mathcal{G}(\sigma_{ik}^{-1}; \frac{\eta_0}{2}, \frac{b_{0k}}{2}) d\sigma_{ik} \\ &= D \left\{ \left(\frac{\eta_0}{2} - 1 \right) \Psi\left(\frac{\eta_i}{2}\right) - \ln \Gamma\left(\frac{\eta_0}{2}\right) \right\} \\ &\quad + \sum_{k=1}^D \left\{ \frac{\eta_0}{2} \ln\left(\frac{b_{0k}}{b_{ik}}\right) + \ln\left(\frac{b_{ik}}{2}\right) - \frac{\eta_i b_{0k}}{2b_{ik}} \right\}.\end{aligned}\quad (\text{A.12})$$

The energy related to only covariance matrices is

$$\begin{aligned} \mathcal{F}_{343}(\mathbf{\Sigma}_i) = & D \left\{ \frac{\eta_0 - \eta_i}{2} \Psi \left(\frac{\eta_i}{2} \right) - \ln \Gamma \left(\frac{\eta_0}{2} \right) + \ln \Gamma \left(\frac{\eta_i}{2} \right) + \frac{\eta_i}{2} \right\} \\ & + \frac{1}{2} \sum_{k=1}^D \left\{ \eta_0 \ln \left(\frac{b_{0k}}{b_{ik}} \right) - \frac{\eta_i b_{0k}}{b_{ik}} \right\}. \end{aligned} \quad (\text{A.13})$$

Bibliography

- [1] K. F. Lee, “Context-dependent phonetic Hidden Markov Models for speaker independent continuous speech recognition,” *IEEE Trans. ASSP*, vol. 38, no. 4, pp. 599–609, 1990.
- [2] S. Sagayama, “Phoneme environment clustering,” in *Proc. of ASJ Conf.*, 1987, vol. 1-5-15, pp. 29–30, (In Japanese).
- [3] S. Sagayama, “Phoneme environment clustering for speech recognition,” in *Proc. of ICASSP89*, 1989, pp. 397–400.
- [4] K. F. Lee, S. Hayamizu, H. W. Hon, C. Huang, J. Swartz, and R. Weide, “Allophone clustering for continuous speech recognition,” in *Proc. of ICASSP90*, 1990, pp. 749–752.
- [5] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based state tying for high accuracy acoustic modeling,” in *Proc. of the ARPA Workshop on Human Language Technology*, 1994, pp. 307–312.
- [6] J. Takami and S. Sagayama, “A successive state splitting algorithm for efficient allophone modeling,” in *Proc. ICASSP’92*, 1992, vol. 1, pp. 573–576.
- [7] M. Ostendorf and H. Singer, “HMM topology design using maximum likelihood successive state splitting,” *Computer Speech and Language*, vol. 11, pp. 17–41, 1997.
- [8] S. Ikeda, “Construction of phone HMM using model search method,” *IEICE*, vol. J78-D-II, no. 1, pp. 10–18, 1995, (In Japanese).

- [9] K. Shinoda and T. Watanabe, “Acoustic modeling based on the MDL principle for speech recognition,” in *Proc. of EUROSPEECH’97*, 1997, vol. 1, pp. 99–102.
- [10] K. Shinoda and T. Watanabe, “MDL-based context-dependent subword modeling for speech recognition,” *The Journal of the Acoustical Society of Japan (E)*, vol. 21, no. 2, pp. 79–86, 2000.
- [11] S. S. Chen and R. A. Gopinath, “Model selection in acoustic modeling,” in *Proc. of EUROSPEECH’99*, 1999, vol. 3, pp. 1087–1090.
- [12] W. Chou and W. Reichl, “Decision tree state tying based on penalized Bayesian information criterion,” in *Proc. of ICASSP’99*, 1999, vol. I, pp. 345–348.
- [13] S. R. Waterhouse, D. MacKay, and A. J. Robinson, “Bayesian methods for mixture of experts,” in *Advances in Neural Information Processing Systems*, M. Haselmo D. Touretzky, M. Mozer, Ed. 1996, vol. 8, pp. 351–357, MIT Press.
- [14] H. Attias, “Inferring parameters and structure of latent variable models by variational Bayes,” in *Proc. of Uncertainty in Artificial Intelligence*, 1999.
- [15] Hagai Attias, “A variational Bayesian framework for graphical models,” in *Advances in Neural Information Processing Systems*, K. R. Muller S. Solla, T. Leen, Ed. 2000, vol. 12, pp. 49–52, MIT Press.
- [16] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda, “Application of the variational bayesian approach to speech recognition,” in *NIPS2002*, 2002.
- [17] R. Lau, R. Rosenfeld, and S. Roukos, “Trigger-based language models: A maximum entropy approach,” in *Proc. of ICASSP’93*, 1993, vol. II, pp. 45–48.
- [18] S. Zhang, H. Yamamoto, H. Singer, D. Wu, and Y. Sagisaka, “To optimize language model by hierarchical modeling,” in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU’99)*, 1999.
- [19] C. Chelba and F. Jelinek, “Exploiting syntactic structure for language modeling,” in *Proc. of COLING-ACL’98*, 1998, pp. 225–284.
- [20] C. Chelba and F. Jelinek, “Structured language modeling,” *Computer Speech and Language*, vol. 14, no. 4, pp. 283–331, 2000.

- [21] S. Mori, M. Nishimura, and N. Itoh, "Improvement of a structured language model: Arbori-context tree," in *Proc. of EUROSPEECH2001*, 2001, pp. 713–716.
- [22] T. Akiba and K. Itou, "A structured statistical language model conditioned by arbitrarily abstracted grammatical categories based on GLR parsing," in *Proc. of EUROSPEECH2001*, 2001, pp. 705–708.
- [23] O. Furuse and H. Iida, "Constituent boundary parsing for example-based machine translation," in *Proc. of COLING-94*, 1994, pp. 105–111.
- [24] P. A. Chou, "Optimal partitioning for classification and regression trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 340–354, 1991.
- [25] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control*, vol. 19, pp. 716–723, 1974.
- [26] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.
- [27] J. Rissanen, "A universal prior for integers and estimation by minimum description length," *Annals of Statistics*, vol. 11, pp. 416–431, 1983.
- [28] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. on IT*, vol. IT-30, no. 4, pp. 629–636, 1984.
- [29] N. Ueda and Z. Ghahramani, "Optimal model inference for Bayesian mixture of experts," in *Proc. of IEEE Neural Networks for Signal Processing (NNSP)*, 2000, pp. 145–154.
- [30] S. Watanabe, Y. Minami, A. Nakamura, and N. Ueda, "Variational Bayesian estimation and clustering for speech recognition," *IEEE Trans. Speech and Audio Processing*, vol. 12, no. 4, pp. 365–381, 2004.
- [31] F. Valente and C. Wellekens, "Variational Bayesian GMM for speech recognition," in *Proc. of EUROSPEECH2003*, 2003, vol. 4, pp. 441–444.

- [32] L. R. Bahl, J. K. Baker, P. S. Cohen, F. Jelinek, B. L. Lewis, and R. L. Mercer, "Recognition of a continuously read natural corpus," in *Proc. of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, 1978.
- [33] T. Takezawa, T. Morimoto, and Y. Sagisaka, "Speech and language databases for speech translation research in ATR," in *Proc. of the 1st International Workshop on East-Asian Language Resources and Evaluation (EALREW'98)*, 1998, pp. 148–155.
- [34] H. Yamamoto and Y. Sagisaka, "Multi-class composite n-gram based on connection direction," in *Proc. of ICASSP'99*, 1999, vol. 1, pp. 533–536.
- [35] S. Furui, K. Maekawa, and H. Isaharaw, "Toward the realization of spontaneous speech recognition – introduction of a japanese priority program and preliminary results –," in *Proc. of ICSLP2000*, 2000, vol. 3, pp. 518–521.
- [36] T. Jitsuhiro, T. Matsui, and S. Nakamura, "Automatic generation of non-uniform context-dependent HMM topologies based on the MDL criterion," in *Proc. of EUROSPEECH'03*, 2003, vol. 4, pp. 2721–2724.
- [37] T. Kato, S. Kuroiwa, T. Shimizu, and N. Higuchi, "Tree-based clustering for Gaussian mixture HMMs," *IEICE Trans. D-II*, vol. J83–D–II, no. 11, pp. 2128–2136, 2000, (in Japanese).
- [38] E. Sumita and H. Iida, "Example-based transfer of Japanese adnominal particles into English," *IEICE Trans. Inf. & Syst.*, vol. E75-D, no. 4, pp. 585–594, 1992.
- [39] S. Watanabe and A. Nakamura, "Acoustic model adaptation based on coarse/fine training of transfer vectors and its application to a speaker adaptation task," in *Proc. of INTERSPEECH–ICSLP2004*, 2004, vol. IV, pp. 2933–2936.

List of Publications

Journal Papers

Takatoshi Jitsuhiro, Satoshi Takahashi, Kiyooki Aikawa, “Rejection of Unknown Words Using Phoneme Confidence Likelihood for Isolated-Word Speech Recognition” *IEICE Trans. D-II*, vol. J83-D-II, no. 2, pp. 478–485, 2000 (in Japanese).

Takatoshi Jitsuhiro, Hirofumi Yamamoto, Setsuo Yamada, Genichiro Kikui, Yoshinori Sagisaka, “Language Modeling Using Patterns Extracted from Parse Trees for Speech Recognition,” *IEICE Trans. Inf. & Syst.*, vol. E86-D, no. 3, pp. 446–453, 2003. (Chapter 5)

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, “Automatic Generation of Non-uniform HMM Topologies Based on the MDL Criterion,” *IEICE Trans. Inf. & Syst.*, vol. E87-D, no. 8, pp. 2121–2129, 2004. (Chapter 3)

Takatoshi Jitsuhiro, Satoshi Nakamura, “Automatic Generation of Non-uniform and Context-Dependent HMMs Based on the Variational Bayesian Approach,” *IEICE Trans. Inf. & Syst.*, vol. E88-D, no. 3, 2005. (Chapter 4)

International Conference

Takatoshi Jitsuhiro, Hirofumi Yamamoto, Setsuo Yamada, Yoshinori Sagisaka, “New Language Models Using Phrase Structures Extracted From Parse Trees,” *Proc. EU-ROSPEECH2001*, vol. 1, pp. 697–700, 2001.

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, “Automatic Generation of

Non-Uniform Context-Dependent HMM Topologies Based on The MDL Criterion,” *Proc. of EUROSPEECH2003*, vol. 4, pp. 2721–2724, 2003.

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, “A Successive State Splitting Algorithm Based on The MDL Criterion by Data-driven and Decision Tree Clustering,” *Proc. of ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition (SSPR2003)*, pp. 43–46, 2003.

Satoshi Nakamura, Konstantin Markov, Takatoshi Jitsuhiro, Jin-Song Zhang, Hirofumi Yamamoto, Genichiro Kikui, “Multi-Lingual Speech Recognition System for Speech-To-Speech Translation,” *Proc. of International Workshop on Spoken Language Translation (IWSLT)*, pp. 147–154, 2004.

Takatoshi Jitsuhiro, Satoshi Nakamura, “Variational Bayesian Approach for Automatic Generation of HMM Topologies,” *Proc. of 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU2003)*, pp. 77–82, 2003.

Takatoshi Jitsuhiro, Satoshi Nakamura, “Automatic Generation of Non-Uniform HMM Structures Based on Variational Bayesian Approach,” *Proc. of ICASSP2004*, vol. I, pp. 805–808, 2004.

Takatoshi Jitsuhiro, Satoshi Nakamura, “Increasing the Mixture Components of Non-Uniform HMM Structures Based on a Variational Bayesian Approach,” *Proc. of INTERSPEECH2004 – ICSLP*, vol. I, pp. 697–700, 2004.

Technical Reports

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, “Automatic Construction of HMM Structures by the MDL Criterion on the SSS Algorithm,” *IEICE Technical Report*, vol. NLC2002-50, SP2002-127, pp. 37–42, 2002 (in Japanese).

Takatoshi Jitsuhiro, Satoshi Nakamura, “The Successive State Splitting Algorithm based on the Variational Bayesian Approach,” *IEICE Technical Report*, vol. NLC2003-57, SP2003-120, pp. 43–48, 2003 (in Japanese).

Meeting Papers

Takatoshi Jitsuhiro, Hirofumi Yamamoto, Setsuo Yamada, Yoshinori Sagisaka, “A language model reflected by the relation between phrase structures,” *Proc. of the 2001 Spring Meeting of the Acoustical Society of Japan*, vol. 1, pp. 195–196, 2001 (in Japanese).

Takatoshi Jitsuhiro, Hirofumi Yamamoto, Genichiro Kikui, “A language model using phrase structures extracted by example-based parse,” *Proc. of the 2002 Spring Meeting of the Acoustical Society of Japan*, vol. 1, pp. 159–160, 2001 (in Japanese).

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, “State Splitting Algorithm Using MDL Criterion” *Proc. of the 2002 Autumn Meeting of the Acoustical Society of Japan*, vol. 1, pp. 41–42, 2002 (in Japanese).

Takatoshi Jitsuhiro, Satoshi Nakamura, “A Method of Generating Acoustic Models Including Temporal Splitting by Variational Bayesian Approach” *Proc. of the 2004 Spring Meeting of the Acoustical Society of Japan*, vol. 1, pp. 17–18, 2004 (in Japanese).

Patents

Takatoshi Jitsuhiro, Takeshi Kawabata, Akihiro Imamura, “A method and its system for speech recognition,” Patent no. 3378547.

(實廣 貴敏, 川端 豪, 今村 明弘, “音声認識方法及び装置,” 特許第 3378547)

Takatoshi Jitsuhiro, Satoshi Takahashi, Kiyooki Aikawa, “A method and its program-stored media for speech recognition,” Patent no. 3496706.

(實廣 貴敏, 高橋 敏, 相川 清明, “音声認識方法及びそのプログラム記録媒体,” 特許 3496706)

Takatoshi Jitsuhiro, Hirofumi Yamamoto, Setsuo Yamada, Yoshinori Sagisaka, “A generation system of language models and a speech recognition system,” Patent application no. 2001-63482.

(實廣 貴敏, 山本 博史, 山田 節夫, 匂坂 芳典, “言語モデル生成装置及び音声認識装

置,” 特願 2001-63482)

Takatoshi Jitsuhiro, Tomoko Matsui, Satoshi Nakamura, Yutaka Ashikari, Gen Ito, “A generation method of acoustic models for speech recognition,” Patent application no. 2002-275888, Patent publication no. 2004-109907.

(實廣 貴敏, 松井 知子, 中村 哲, 葦 莉 豊, 伊藤 玄 “音声認識用音響モデル構造生成方法,” 特願 2002-275888, 特開 2004-109907)

Takatoshi Jitsuhiro, Satoshi Nakamura, “A generation system of model structures, a speech recognition system, and a program for a generation of model structures,” Patent application no. 2003-363900.

(實廣 貴敏, 中村 哲, “モデル構造作成装置, 音声認識装置, 及びモデル構造作成プログラム,” 特願 2003-363900)

Takatoshi Jitsuhiro, Satoshi Nakamura, “A generation system of mixture distribution models, a speech recognition system, and a program for a generation of mixture distribution models,” Patent application no. 2004-280807.

(實廣 貴敏, 中村 哲, “混合分布モデル作成装置, 音声認識装置, 及び混合分布モデル作成プログラム,” 特願 2004-280807)