Electronic Preprint for Journal of Information Processing Vol.27

Regular Paper

# Viterbi Approximation of Latent Words Language Models for Automatic Speech Recognition

Ryo Masumura[1,a)] Taichi Asami[1] Takanobu Oba[1]
Hirokazu Masataki[1] Sumitaka Sakauchi[1]

Received: January 29, 2018, Accepted: November 7, 2018

**Abstract:** This paper presents a Viterbi approximation of latent words language models (LWLMs) for automatic speech recognition (ASR). The LWLMs are effective against data sparseness because of their soft-decision clustering structure and Bayesian modeling, so LWLMs can perform robustly in multiple ASR tasks. Unfortunately, implementing an LWLM to ASR is difficult because of its computation complexity. In our previous work, we implemented an n-gram approximation of LWLM for ASR by sampling words according to a stochastic process and training word n-gram LMs. However, the previous approach cannot take into account a latent word sequence behind a recognition hypothesis. Our solution is the Viterbi approximation that simultaneously decodes both the recognition hypothesis and the latent word sequence. The Viterbi approximation is implemented as a two-pass ASR decoding in which the latent word sequence is estimated from a decoded recognition hypothesis using Gibbs sampling. Experiments show the effectiveness of the Viterbi approximation in an n-best rescoring framework. In addition, we investigate the relationship of the n-gram approximation and the Viterbi approximation.

**Keywords:** latent words language model, Viterbi approximation, Gibbs sampling, automatic speech recognition, n-best rescoring

## 1. Introduction

Language models (LMs) are necessary for various natural language processing tasks such as automatic speech recognition (ASR). One of the most common problems faced by LMs is data sparseness [1], [2], [3]. In ASR tasks, large amounts of domain-matched training data sets are not available because the data set must be obtained by manually transcribing speech. Therefore, LMs are often required to robustly predict the probability of unobserved linguistic phenomena even though the domain-matched training data is limited.

To mitigate the data sparseness problem, several techniques have been proposed. The most traditional technique is smoothing in n-gram modeling [4]. Various smoothing methods that improve LMs probability estimation have been studied for n-gram LMs [4], [5]. Another solution is based on dimensionality reduction. Class-based n-gram LMs [6] and decision tree LMs [7] are based on word classification, and neural network based LMs are based on learning the distributed representation of words [8], [9], [10], [11].

In order to achieve further domain robust language modeling, this paper focuses on latent words LMs (LWLMs) since they can flexibly perform both smoothing and dimensionality reduction [12]. LWLMs have latent variables which are called latent words. Remarkably, LWLM has a soft clustering structure in common with Bayesian hidden Markov models (HMMs) [13], [14] and the Bayesian class-based LMs [15], [16]. In contrast to these models, LWLM has a vast latent variable space whose size is equivalent to the vocabulary size of the training data. These flexible attributes help us to efficiently realize the smoothing and the dimensionality reduction. Therefore, it can be expected that LWLM robustly covers multiple domains in ASR. In addition, flexible mixture modeling in the latent variable space can be achieved by using multiple LWLMs [17], [18] while conventional class-based n-gram observed word space mixture modeling can only be performed by the Bayesian HMMs and the Bayesian class-based LMs. However, some approximation is inevitable for ASR implementation because these attributes seriously increase computation complexity.

One approximation method is an n-gram approximation, in which n-gram LM is trained from words sampled according to a stochastic process of the LWLM [19], [20]. The n-gram approximation can perform in a one-pass ASR decoder and shows effectiveness in multiple ASR tasks. However, the n-gram approximation cannot take into account latent words behind a recognition hypothesis because the approximation is based on a simple back-off n-gram structure. Actually, the n-gram approximation ignores an important concept which every observed word in a text has a latent word.

In order to directly take into account the latent words, this paper presents the Viterbi approximation of LWLMs. The Viterbi approximation simultaneously decodes a recognition hypothesis and its optimal latent word sequence using a joint probability between the two sequences. This technique is usually used in hidden Markov models (HMMs), and it is known that the perfor-

[1] NTT Media Intelligence Laboratories, NTT Corporation, Yokosuka, Kanagawa 239–0847, Japan
[a)] ryou.masumura.ba@hco.ntt.co.jp

mance is comparable to that obtained when taking account of all possible latent variable assignments [21]. It can be expected that the Viterbi approximation of LWLMs will provide further ASR improvements.

It is known that the Viterbi algorithm is a formal technique to compute the joint probability of an observed word sequence and its optimal latent variable sequence [22]. However, there are innumerable combinations of the recognition hypothesis and its latent word assignment in LWLMs. In fact, the computation complexity to determine the optimal latent assignment via the Viterbi algorithm is $O(|C|^n)$; where $|C|$ is the size of the latent word space and $n$ is the n-gram order for the latent word modeling. It is difficult to apply the Viterbi algorithm to the LWLMs because the size of the latent word space corresponds to the vocabulary size and the n-gram order is usually over two.

To overcome this problem, we implement the Viterbi approximation as a two-pass process using Gibbs sampling. In the process, several recognition hypotheses are preliminarily decoded in a first pass, and then each optimal latent word assignment is decoded in a second pass. This enables us to handle a joint probability between the recognition hypothesis and the optimal latent word assignment. The Gibbs sampling is used to approximately find the best latent word assignment [23], [24], [25]. The Gibbs sampling is a simple and widely used method for generating random samples from a joint distribution over several variables. Gibbs sampling can reduce the computation complexity from $O(|C|^n)$ to $O(|C|)$, so Viterbi approximation can be conducted more rapidly than is possible with the formal Viterbi algorithm.

This paper is an extended study of our previous work [26]. In this paper, we detail the definition of the Viterbi approximation more precisely and extend our evaluation to not only ASR evaluation presented in the previous work but also perplexity evaluation using Penn treebank corpus [27] so as to compare our proposed method with many previous studies in terms of perplexity. Furthermore, we compare the LWLM based ASR with state-of-the-art RNN language modeling [9], [10].

This paper is organized as follows. Section 2 details LWLMs and the n-gram approximation for implementing the LWLMs to ASR. Section 3 explains the definition of the Viterbi approximation and a method to compute the joint probability of a recognition hypothesis and its optimal latent word assignment. Sections 4 and 5 describe a perplexity evaluation and an ASR evaluation, respectively. Section 6 concludes this paper.

## 2. Latent Words Language Models

### 2.1 Definition

Latent words LMs (LWLMs) are generative models that set a latent variable for every observed word [12], [19], [20]. A graphic rendering of LWLM is shown in **Fig. 1**. The gray circles denote observed words and the white circles denote latent variables.

In the generative process of LWLM, a latent variable, called latent word $h_t$, is generated depending on the transition probability distribution given context $\boldsymbol{l}_t = h_{t-n+1}, \ldots, h_{t-1}$, where $n$ is an n-gram order. Next, observed word $w_t$ is generated depending on the emission probability distribution given latent word $h_t$, i.e.,

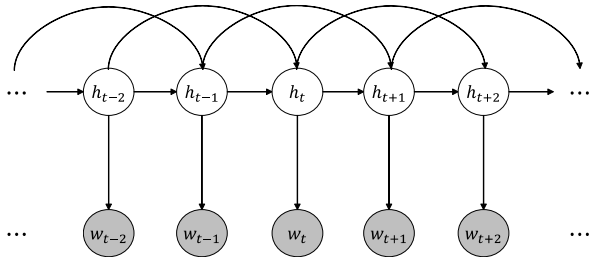


**Fig. 1** Model structure of LWLMs.

$$h_t \sim P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}), \tag{1}$$

$$w_t \sim P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}), \tag{2}$$

where $\boldsymbol{\Theta}_{\mathtt{lw}}$ is a model parameter of LWLM. Here, $P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}})$ is expressed as an n-gram model for latent words, and $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}})$ models the dependency between the observed word and the latent word.

An important property of LWLMs is that the latent word is expressed as a specific word that can be selected from an entire vocabulary $\mathcal{V}$. Thus, the number of latent words is the same as the vocabulary size $|\mathcal{V}|$. This is the reason the latent variable is called a latent word.

### 2.2 Bayesian LWLMs

LWLMs are often modeled by the Bayesian inference. The Bayesian LWLM produces the following generative probability for observed words $\boldsymbol{w} = w_1, \cdots, w_T$ as:

$$P(\boldsymbol{w}) = \int \sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{\Theta}_{\mathtt{lw}}) d\boldsymbol{\Theta}_{\mathtt{lw}}, \tag{3}$$

where $\boldsymbol{h} = h_1, \cdots, h_T$ is a latent word assignment. This equation can be detailed as:

$$P(\boldsymbol{w}) = \int \prod_{t=1}^{T} \sum_{\boldsymbol{h}} P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}) P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{\Theta}_{\mathtt{lw}}) d\boldsymbol{\Theta}_{\mathtt{lw}}. \tag{4}$$

The Bayesian approach takes account of all possible model parameters. As the integral with respect to $\boldsymbol{\Theta}_{\mathtt{lw}}$ is analytically intractable, a sampling technique is used as a feasible approximation. Equation (3) is approximated as:

$$P(\boldsymbol{w}) \simeq \frac{1}{M} \sum_{m=1}^{M} P(\boldsymbol{w}|\boldsymbol{\Theta}_{\mathtt{lw}}^m), \tag{5}$$

$$P(\boldsymbol{w}|\boldsymbol{\Theta}_{\mathtt{lw}}^m) = \prod_{t=1}^{T} \sum_{\boldsymbol{h}} P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m) P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m), \tag{6}$$

where $\boldsymbol{\Theta}_{\mathtt{lw}}^m$ means the $m$-th point estimated model parameter. The generative probability can be approximated using $M$ instances of $\boldsymbol{\Theta}_{\mathtt{lw}}^m$. In fact, the ensemble of several models ($M > 1$) is effective for LMs such as random class based LMs [28] and random forest LMs [29].

LWLM has a similar structure to the standard class based n-gram model. The latent word corresponds, approximately, to the class of the standard class based n-gram model [6]. LWLM has a soft word clustering structure that differs from a simple hard word clustering structure in the standard class based n-gram model. In

the hard word clustering structure, one word belongs to only one class. In the soft word clustering structure, on the other hand, one word belongs to multiple classes. Strictly speaking, each word belongs to all classes in LWLM. In addition, LWLM has a vast class space, about as large as the vocabulary, while the number of classes in the standard class based n-gram model is often defined as several hundreds or thousands.

### 2.3 Training

LWLMs are trained from a training data set $\mathcal{W}$. In LWLM training, the latent word assignment $\mathcal{H}$ behind $\mathcal{W}$ have to be inferred. In fact, multiple latent word assignments $\mathcal{H}_1, \cdots, \mathcal{H}_M$ are estimated for the Bayesian modeling. Once a latent word assignment $\mathcal{H}_m$ is defined, $P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})$ and $P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})$ can be calculated.

To estimate the latent word assignments, Gibbs sampling is suitable. Gibbs sampling samples a new value for the latent word in accordance with its distribution and places it at position $t$ in $\mathcal{H}$. The conditional probability distribution of possible values for latent word $h_t$ is given by:

$$P(h_t|\mathcal{W}, \mathcal{H}_{-t}) \propto P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw},-t}) \prod_{j=t}^{t+n-1} P(h_j|\boldsymbol{l}_j, \boldsymbol{\Theta}_{\mathtt{lw},-t}), \tag{7}$$

where $\mathcal{H}_{-t}$ represents all latent words except for $h_t$ and $n$ is the n-gram order for the latent word modeling. In the sampling procedure, $P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw},-t})$ and $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw},-t})$ can be calculated from $\mathcal{W}$ and $\mathcal{H}_{-t}$.

The transition probability distribution and the emission probability distribution are calculated on the basis of their prior distributions. For the transition probability distribution, this paper uses a prior hierarchical Pitman-Yor. $P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}})$ is given as:

$$P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}) = P(h_t|\boldsymbol{l}_t, \mathcal{H}), \tag{8}$$

$$P(h_t|\boldsymbol{l}_t, \mathcal{H}) = \frac{c(h_t, \boldsymbol{l}_t) - d_{|\boldsymbol{l}_t|}s(h_t, \boldsymbol{l}_t)}{\theta_{|\boldsymbol{l}_t|} + c(\boldsymbol{l}_t)} + \frac{\theta_{|\boldsymbol{l}_t|} + d_{|\boldsymbol{l}_t|}s(\boldsymbol{l}_t)}{\theta_{|\boldsymbol{l}_t|} + c(\boldsymbol{l}_t)} P(h_t|\pi(\boldsymbol{l}_t), \mathcal{H}), \tag{9}$$

where $\pi(\boldsymbol{l}_t)$ is the shortened context obtained by removing the earliest word from $\boldsymbol{l}_t$. $c(h_t, \boldsymbol{l}_t)$ and $c(\boldsymbol{l}_t)$ are counts calculated from a latent word assignment $\mathcal{H}$. $s(h_t, \boldsymbol{l}_t)$ and $s(\boldsymbol{l}_t)$ are calculated from a seating arrangement defined by the Chinese restaurant franchise representation of the Pitman-Yor process [30]. $d_{|\boldsymbol{l}_t|}$ and $\theta_{|\boldsymbol{l}_t|}$ are discount and strength parameters of the Pitman-Yor process, respectively. Moreover, a Dirichlet prior is used for the emission probability distribution [31]. $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}})$ is given as:

$$P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}) = P(w_t|h_t, \mathcal{W}, \mathcal{H}), \tag{10}$$

$$P(w_t|h_t, \mathcal{W}, \mathcal{H}) = \frac{c(w_t, h_t) + \alpha P(w_t)}{c(h_t) + \alpha}, \tag{11}$$

where $P(w_t)$ is the maximum likelihood estimation value of unigram probability in the training data set $\mathcal{W}$. $c(w_t, h_t)$ and $c(h_t)$ are counts calculated from $\mathcal{W}$ and latent word assignment $\mathcal{H}$. A hyper parameter $\alpha$ can be optimized via a validation data set.

### 2.4 N-gram Approximation

The n-gram approximation of LWLMs is to convert LWLMs into the back-off n-gram structure. A basic concept is to construct a smoothed n-gram LM that can generate similar words to those generated from LWLM. Thus, the approximated LWLM $P(\boldsymbol{w}|\boldsymbol{\Theta}_{\mathtt{lwng}})$ has the following properties:

$$\boldsymbol{w}_{\mathtt{lw}} \sim P(\boldsymbol{w}|\boldsymbol{\Theta}^1_{\mathtt{lw}}, \cdots, \boldsymbol{\Theta}^M_{\mathtt{lw}}), \tag{12}$$

$$\boldsymbol{w}_{\mathtt{lwng}} \sim P(\boldsymbol{w}|\boldsymbol{\Theta}_{\mathtt{lwng}}), \tag{13}$$

$$\boldsymbol{w}_{\mathtt{lw}} \simeq \boldsymbol{w}_{\mathtt{lwng}}, \tag{14}$$

where $\boldsymbol{w}_{\mathtt{lw}}$ is an observed word sequence generated from the LWLM, and $\boldsymbol{w}_{\mathtt{lwng}}$ is an observed word sequence generated from the approximated LWLM with back-off n-gram structure. The approximated LWLM can be constructed from words generated from the LWLM. In fact, any back-off n-gram structure, including hierarchical Pitman-Yor LMs (HPYLMs) [30], can be used for the approximation.

As LWLM is a generative model, it can generate latent words and observed words based on random sampling. Therefore, a lot of observed words can be easily sampled, and the smoothed n-gram LM can be constructed. Although the model size of n-gram approximated LWLM is large, its n-gram entries can be easily reduced by the entropy pruning technique [32].

## 3. Viterbi Approximation of LWLMs

### 3.1 Definition

The Viterbi approximation of LWLMs uses the joint probability of a word sequence $\boldsymbol{w} = w_1, \cdots, w_T$ and its optimal latent word assignment $\bar{\boldsymbol{h}} = \bar{h}_1, \cdots, \bar{h}_T$. The joint probability is called the Viterbi probability. The Viterbi probability $P(\boldsymbol{w}, \bar{\boldsymbol{h}})$ is defined as:

$$P(\boldsymbol{w}, \bar{\boldsymbol{h}}) = \max_{\boldsymbol{h}} P(\boldsymbol{w}, \boldsymbol{h}), \tag{15}$$

$$= \max_{\boldsymbol{h}} \frac{1}{M} \sum_{m=1}^{M} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}^m_{\mathtt{lw}}) P(\boldsymbol{h}|\boldsymbol{\Theta}^m_{\mathtt{lw}}). \tag{16}$$

The probability is also denoted as:

$$P(\boldsymbol{w}, \bar{\boldsymbol{h}}) = \max_{\boldsymbol{h}} \frac{1}{M} \prod_{t=1}^{T} \sum_{m=1}^{M} P(w_t|\bar{h}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}) P(\bar{h}_t|\bar{\boldsymbol{l}}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}), \tag{17}$$

where $\boldsymbol{l}_t = \bar{h}_{t-n+1}, \cdots, \bar{h}_{t-1}$. Note that a perplexity calculated from the Viterbi probability is called a Viterbi perplexity. The Viterbi perplexity which utilizes the joint probability of the word sequence and the optimal latent word sequence does not exactly correspond to word perplexity although the Viterbi perplexity can be used for evaluating availability to estimate word sequences.

In ASR, the Viterbi approximation can be defined as a problem of the Viterbi ASR decoding. The Viterbi ASR decoding simultaneously decodes the recognition hypothesis $\hat{\boldsymbol{w}}$ and its latent word assignment $\hat{\boldsymbol{h}}$ using a joint probability $P(\boldsymbol{w}, \boldsymbol{h})$. This probabilistic decision problem is defined as:

$$(\hat{\boldsymbol{w}}, \hat{\boldsymbol{h}}) = \arg\max_{(\boldsymbol{w}, \boldsymbol{h}) \in \mathcal{S}} P(\boldsymbol{x}|\boldsymbol{w}) P(\boldsymbol{w}, \boldsymbol{h}), \tag{18}$$

where $\mathcal{S}$ is the search space. $\boldsymbol{x}$ is an input speech signal, and

$P(\boldsymbol{x}|\boldsymbol{w})$ is an acoustic model.

The Viterbi ASR decoding is impractical to implement as a one-pass ASR process because innumerable combinations of the recognition hypothesis and its latent word assignment have to be taken into consideration. Therefore, this paper implements the Viterbi ASR decoding as a two-process. By using the Viterbi probability, the ASR decoding can be defined as:

$$\hat{\boldsymbol{w}}, \hat{\boldsymbol{h}} = \arg\max_{(\boldsymbol{w},\bar{\boldsymbol{h}})\in\mathcal{L}} P(\boldsymbol{x}|\boldsymbol{w})P(\boldsymbol{w},\bar{\boldsymbol{h}}), \tag{19}$$

$$= \arg\max_{(\boldsymbol{w},\bar{\boldsymbol{h}})\in\mathcal{L}} \{\log P(\boldsymbol{x}|\boldsymbol{w}) + \log P(\boldsymbol{w},\bar{\boldsymbol{h}})\}, \tag{20}$$

where $\mathcal{L}$ denotes the limited search space. The limited search space is produced by a first decoding pass in which several recognition hypotheses are preliminarily decoded using the standard n-gram LM. Thus $\mathcal{S}$ is regarded as a recognition hypotheses list generated in the first pass. Then, each optimal latent word assignment is estimated from the recognition hypothesis in a second pass. In the two pass process, only the Viterbi probability of each recognition hypothesis is calculated. This computation cost is much smaller than that of an intuitive one-pass process. In addition, in the two-pass process, the n-gram probability in the first decoding pass and the Viterbi probability calculated in the second decoding pass can be mixed. In this paper, $\mathcal{L}$ corresponds to n-best lists, so the two-pass process is achieved by n-best rescoring.

### 3.2 Viterbi Probability Computation

#### 3.2.1 Viterbi Algorithm

The Viterbi algorithm simultaneously solves the problems of finding the optimal latent word assignment $\bar{\boldsymbol{h}}$ of an observed word sequence $\boldsymbol{w}$ and computing the Viterbi probability based on dynamic programming methods. The Viterbi probability can, based on the Viterbi algorithm, be obtained as:

$$P(\boldsymbol{w},\bar{\boldsymbol{h}}) = \max_{\boldsymbol{l}_{L+1}} \sum_{m=1}^{M} \delta(\boldsymbol{l}_{L+1}, \boldsymbol{\Theta}^m_{\mathtt{lw}}). \tag{21}$$

where $\delta(\boldsymbol{l}_{t+1}, \boldsymbol{\Theta}^m_{\mathtt{lw}})$ is the joint probability of observing $w_1, \cdots, w_t$ together with sequence $\boldsymbol{l}_{t+1} = h_{t-n+2}, \cdots, h_t$. It is defined recursively as:

$$\delta(\boldsymbol{l}_{t+1}, \boldsymbol{\Theta}^m_{\mathtt{lw}}) = P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})\delta(\bar{\boldsymbol{l}}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})P(h_t|\bar{\boldsymbol{l}}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}), \tag{22}$$

where $\bar{\boldsymbol{l}}_t = \bar{h}_{t-n+1}, h_{t-n+2}, \cdots, h_{t-1}$. $\bar{h}_{t-n+1}$ is the optimal latent word through $\boldsymbol{l}_{t+1}$. $\bar{h}_{t-n+1}$ is determined as:

$$\bar{h}_{t-n+1} = \arg\max_{h_{t-n+1}} \sum_{m=1}^{M} P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})\delta(\boldsymbol{l}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}). \tag{23}$$

The computation complexity via the Viterbi algorithm is equal to that of the forward algorithm, that is $O(|\mathcal{V}|^n)$. As noted previously, it is difficult to use the Viterbi algorithm directly in LWLM.

#### 3.2.2 Gibbs Sampling Based Computation

This paper computes the Viterbi probability by finding the optimal latent word assignment $\bar{\boldsymbol{h}}$ from $\boldsymbol{w}$ using the Gibbs sampling. The optimal latent word assignment $\bar{\boldsymbol{h}}$ is, with regard to $\boldsymbol{w}$, obtained as:

$$\bar{\boldsymbol{h}} = \arg\max_{\boldsymbol{h}} P(\boldsymbol{h}|\boldsymbol{w}), \tag{24}$$

$$= \arg\max_{\boldsymbol{h}} \frac{1}{M} \sum_{m=1}^{M} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}^m_{\mathtt{lw}})P(\boldsymbol{h}|\boldsymbol{\Theta}^m_{\mathtt{lw}}), \tag{25}$$

$$= \arg\max_{\boldsymbol{h}} \frac{1}{M} \prod_{t=1}^{T} \sum_{m=1}^{M} P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}). \tag{26}$$

The Gibbs sampling technique can be used in order to find the approximately optimal latent word assignment more rapidly. First, several latent words assignments are sampled on the basis of Gibbs sampling. A conditional probability distribution of the possible values for latent word $h_t$ is defined as:

$$P(h_t|\boldsymbol{w}, \boldsymbol{h}_{-t}) \propto \sum_{m=1}^{M} \left\{ P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}) \prod_{j=t}^{t+n-1} P(h_j|\boldsymbol{l}_j, \boldsymbol{\Theta}^m_{\mathtt{lw}}) \right\}, \tag{27}$$

where $\boldsymbol{h}_{-t}$ is a latent word assignment except for $h_t$. By the sampling, $I$ samples of latent words assignments $\boldsymbol{h}_1, \cdots, \boldsymbol{h}_I$ are obtained. The Viterbi probability is calculated by finding the optimal one:

$$P(\boldsymbol{w},\bar{\boldsymbol{h}}) \simeq \max_{\boldsymbol{h}\in\{\boldsymbol{h}_1,\cdots,\boldsymbol{h}_I\}} \frac{1}{M} \prod_{t=1}^{T} \sum_{m=1}^{M} P(w_t|h_t, \boldsymbol{\Theta}^m_{\mathtt{lw}})P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}^m_{\mathtt{lw}}). \tag{28}$$

The computation complexity based on Gibbs sampling is $O(|\mathcal{V}|)$, so the approximately optimal latent words assignment can be found much more rapidly than is possible with the formal Viterbi algorithm.

### 3.3 Interpolation with N-gram Probability

In ASR implementation, both n-gram probability in a first decoding pass and Viterbi probability calculated in a second decoding pass can be used. Their combination is based on a linear interpolation of both probabilities. In this case, the interpolated probability $P(\boldsymbol{w},\bar{\boldsymbol{h}}|\boldsymbol{\Theta}_{\mathtt{mix}})$ is defined as:

$$P(\boldsymbol{w},\bar{\boldsymbol{h}}|\boldsymbol{\Theta}_{\mathtt{mix}}) = \prod_{t=1}^{T} P(w_t,\bar{h}_t|\boldsymbol{u}_t,\bar{\boldsymbol{l}}_t,\boldsymbol{\Theta}_{\mathtt{mix}}), \tag{29}$$

where $P(w_t,\bar{h}_t|\boldsymbol{u}_t,\bar{\boldsymbol{l}}_t,\boldsymbol{\Theta}_{\mathtt{mix}})$ is defined as:

$$P(w_t,\bar{h}_t|\boldsymbol{u}_t,\bar{\boldsymbol{l}}_t,\boldsymbol{\Theta}_{\mathtt{mix}}) = \lambda P(w_t|\boldsymbol{u}_t,\boldsymbol{\Theta}_{\mathtt{ng}}) + (1-\lambda)\frac{1}{M}\sum_{m=1}^{M} P(w_t|\bar{h}_t,\boldsymbol{\Theta}^m_{\mathtt{lw}})P(\bar{h}_t|\bar{\boldsymbol{l}}_t,\boldsymbol{\Theta}^m_{\mathtt{lw}}), \tag{30}$$

where $P(w_t|\boldsymbol{u}_t,\boldsymbol{\Theta}_{\mathtt{ng}})$ which represents the generative probability of $w_t$ given by context words $\boldsymbol{u}_t = w_{t-n+1}, \cdots, w_{t-1}$ is calculated by the n-gram LM in a first decoding pass, and $\lambda$ is an interpolation weight. Note that $\boldsymbol{\Theta}_{\mathtt{mix}}$ corresponds to $\{\boldsymbol{\Theta}_{\mathtt{ng}}, \boldsymbol{\Theta}^1_{\mathtt{lw}}, \cdots, \boldsymbol{\Theta}^M_{\mathtt{lw}}, \lambda\}$. Strict Viterbi decoding can be performed when the interpolation weight is 0.0. On the other hand, standard decoding can be performed when the interpolation weight is 1.0. The interpolation weight can be optimized by minimizing the joint probability using a validation data set via the expectation maximization algorithm. In this paper, the interpolated probabilities are also used to calculate perplexity.

## 4. Experiment 1: Perplexity Evaluation

### 4.1 Setups

The first experiments used the Penn Treebank corpus [27], in which sections 0–20 were used as a training data set (Train), sections 21 and 22 were used as a validation data set (Valid), and sections 23 and 24 were used as a test data set (Test A). These selections match those of many previous studies, so we can fairly compare our proposed method with various methods in terms of perplexity. In addition, human-human discussion text data set (Test B) were prepared for evaluations in a domain different from that of the training data set. Each vocabulary was limited to 10K words and there were no out-of-vocabulary (OOV) words. **Table 1** details number of words in each data set.

In this evaluation, the following LMs were constructed.

- `MKN5`: Word-based 5-gram LM with modified Kneser-Ney smoothing constructed from training data set [4].
- `HPY5`: Word-based 5-gram HPYLM constructed from the training data set [5]. For the training, 200 iterations were used for burn-in, and 10 sets of samples were collected.
- `RNN`: Word-based RNNLM constructed from training data set [9]. The hidden layer size was set as 200 by referring to a preliminary experiment.
- `LW-NA`: Word-based 5-gram HPYLM constructed from data generated on the basis of 5-gram LWLM (`LW`) constructed from the training data set [20]. `LW` was constructed from the training data set. For training, 500 iterations were used for burn-in, and 10 samples were collected. The generated data size was one billion words which was determined in consideration of previous work [20]. We pruned n-gram entries as to be comparable computation complexity to `HPY5` using entropy based pruning [32].
- `LW-VA`: Viterbi approximation of the LWLM. The LWLM corresponds to that in `LW-NA`. To calculate the Viterbi probability, 100 samples of latent words assignments were obtained using Gibbs sampling.

In addition, several mixed models constructed by linearly interpolating the above LMs were employed. The mixture weights and other hyper parameters were optimized using a validation data set.

### 4.2 Results

The results based on perplexity (PPL) are shown in **Table 2**. In lines (1)–(5), each LM was evaluated. In lines (6)–(11), mixed LMs were evaluated. In addition, **Fig. 2** shows PPL results where `HPY5` and `LW-VA` were combined using the linear interpolation. The PPL result is the same as that obtained by `HPY5` when the mixture weight is set to 1.0 while the PPL result is the same as that obtained by `LW-VA` when the mixture weight is set to 0.0.

Table 2 shows that `LW-VA` was relatively weaker than `LW-NA`.

**Table 1** Data sets in experiment 1.

| | Domain | Number of words |
|---|---|---|
| Train | Penn Treebank | 929,589 |
| Valid | Penn Treebank | 70,390 |
| Test A | Penn Treebank | 78,669 |
| Test B | Human-Human Discussion | 50,507 |

This is because the Viterbi probability which takes not only an observed word sequence but also an optimal latent word assignment behind the observed word sequence into account was used for computing PPL (Viterbi perplexity) in `LW-VA`. In other words, all possible latent word assignments have to be considered to compute a strict perplexity of the observed word sequence using LWLMs. In Valid and Test A, `LW-VA` is weaker than `MKN5`, `HPY5`, and `RNN`. On the other hand, in Test B, `LW-VA` outperformed `MKN5` and `HPY5`. The results suggested that the Viterbi approximation of LWLM robustly performs in out-of-domain tasks as well as the n-gram approximation of LWLMs. In Fig. 2, the combination of `HPY5` and `LW-VA` based on the linear interpolation improved the PPL in all data sets. This suggests that `LW-VA`, in which optimal latent word assignment is taken into consideration, has properties different from those in `HPY5`. In addition, `LW-NA+LW-VA` outperformed `LW-NA` and `LW-VA`. This result indicates that both the n-gram approximation and the Viterbi approximation should be introduced for utilizing the LWLM for ASR. In each data set, `HPY5+LW-NA+LW-VA` outperformed state-of-the-art `HPY5+RNN`. The best results were achieved by `HPY5+LW-NA+LW-VA+RNN` in each test set. This indicates that both n-gram approximation and Viterbi approximation of LWLM was complement with RNNLM.

Here are examples of an observed word sequence and an optimal latent word sequence extracted from the validation data set.

- Observed word sequence:
  consumers may want to move their telephones a little closer to the TV set
- Optimal latent word sequence:
  investors may want to make their set a lot better than the entire set

**Table 2** Perplexity results on each data set in experiment 1.

| | | Valid | Test A | Test B |
|---|---|---|---|---|
| (1). | `MKN5` | 148.0 | 141.2 | 238.6 |
| (2). | `HPY5` | 145.1 | 139.3 | 232.7 |
| (3). | `RNN` | 134.4 | 128.9 | 212.9 |
| (4). | `LW-NA` | 138.7 | 131.7 | 205.5 |
| (5). | `LW-VA` | 148.4 | 142.9 | 224.7 |
| (6). | `HPY5+RNN` | 111.4 | 107.9 | 180.6 |
| (7). | `HPY5+LW-VA` | 109.2 | 107.0 | 173.1 |
| (8). | `LW-NA+LW-VA` | 115.7 | 112.0 | 174.3 |
| (9). | `HPY5+LW-NA` | 128.6 | 123.1 | 200.8 |
| (10). | `HPY5+LW-NA+LW-VA` | 107.1 | 105.4 | 170.8 |
| (11). | `HPY5+LW-NA+LW-VA+RNN` | **101.5** | **98.9** | **158.5** |

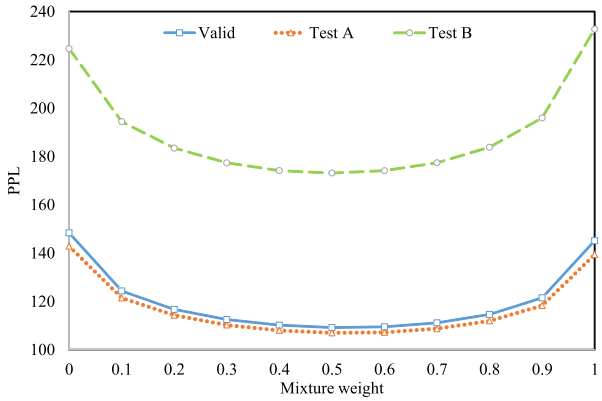


**Fig. 2** Perplexity results of combining `HPY5` and `LW-VA` in experiment 1.

As described above, the optimal latent word sequence is related to the observed word sequence. This suggests that taking into account a latent word sequence behind a recognition hypothesis is effective for ASR.

## 5. Experiment 2: ASR Evaluation

### 5.1 Setups

The second set of experiments used the Corpus of Spontaneous Japanese (CSJ) [33]. CSJ was divided into a training data set (Train), a small validation data set (Valid), and a test data set (Test A). In addition, for evaluation in out-of-domain environments, a contact center dialog task (Test B) and a voice mail task (Test C) were prepared. In order to perform same tokenization between CSJ and the out-of-domain data sets, we used JTAG based morpheme analyzer [34] to split sentences into words for both CSJ and the out-of-domain data sets. The vocabulary size of the training data set was 83,536. For each data set, the number of words and OOV rate are detailed in **Table 3**.

For speech recognition evaluation, an acoustic model (AM) based on hidden Markov models with deep neural networks (DNN-HMM) was prepared [35]. The DNN-HMM had 8 hidden layers with 2048 nodes. The speech recognizer was a weighted finite state transducer (WFST) decoder [36], [37].

In this evaluation, the following LMs were prepared.

- `MKN3`: Word-based 3-gram LM with modified Kneser-Ney smoothing constructed from training data set [4].
- `HPY3`: Word-based 3-gram HPYLM constructed from the training data set [5]. For the training, 200 iterations were used for burn-in, and collected 10 samples.
- `RNN`: Class-based RNNLM with 500 hidden nodes and 500 classes constructed from the training data set [10].
- `LW-NA`: Word-based 3-gram HPYLM constructed from data generated on the basis of 3-gram LWLM (`LW`) constructed from the training data set [20]. For training `LW`, 500 iterations were used for burn-in, and 10 samples were collected. The generated data size was one billion words which was determined in consideration of our previous work [20]. We pruned n-gram entries as to be comparable computation complexity to HPY3 using entropy based pruning [32].
- `LW-VA`: Viterbi approximation of `LW`. To calculate the Viterbi probability, 100 samples of latent words assignments were obtained using Gibbs sampling.

For two-pass decoding process using `LW-VA` or `RNN`, 1000-best hypotheses were generated in the first pass. Several parameters such as hyper parameters, the mixture weight and LM score factors were optimized for the validation data set.

### 5.2 Results

**Table 4** shows the PPL results for each condition. In lines (1)–(5), each LM was evaluated. In lines (6)–(9), mixed LMs were evaluated. In addition, combinations of `HPY3` and `LW-VA` (`HPY3+LW-VA`) were examined in terms of the PPL. The PPL results in which the mixture weight was varied are shown in **Fig. 3**. The PPL result is the same as that obtained by `HPY3` when the mixture weight is set to 1.0 while the PPL result is the same as that obtained by `LW-VA` when the mixture weight is set to 0.0.

In Table 4, `LW-VA` was relatively not so good compared with the other condition in in-domain tasks. On the other hand, `LW-NA` and `LW-VA` were superior to `MKN3`, `HPY3` and `RNN` in out-of-domain tasks. This shows that both the n-gram approximation and the Viterbi approximation of LWLM robustly handle speech domains different from that of the training data set. Moreover, in Test C, `LW-VA` outperformed `LW-NA` in spite of using the Viterbi probability. Figure 3 shows that the combination of `HPY3` and `LW-VA` based on the linear interpolation can improve the PPL in all data sets.

In particular, the improvements in out-of-domain tasks were remarkable. The best PPL results were achieved by `HPY3+RNN` in in-domain tasks and `HPY3+LW-NA+LW-VA` in out-of-domain tasks. This indicates that LWLM based ASR approach is effective for out-of-domain tasks while RNNLM is powerful modeling in in-domain tasks.

Next, **Table 5** shows word error rate (WER) results for the validation data set and each of the test sets. In the first pass, four n-gram LMs were examined. In the second pass, `RNN` and `LW-VA` were mixed with the first pass by n-best rescoring. Note that LM in the first pass was not utilized for the n-best rescoring in line (4).

Line (1) where `MKN3` was employed were our baseline results. WER results in in-domain data sets (CSJ) were relatively lower than the state-of-the-art ASR systems for the CSJ. The re-

**Table 3** Data sets in experiment 2.

| | Domain | Number of words | OOV rate (%) |
|---|---|---|---|
| Train | Lecture | 7,317,392 | - |
| Valid | Lecture | 28,046 | 0.72 |
| Test A | Lecture | 27,907 | 0.51 |
| Test B | Contact center | 24,665 | 3.66 |
| Test C | Voice mail | 21,044 | 4.41 |

**Table 4** Perplexity results on each data set in experiment 2.

| | Setup | Valid (In-domain) | Test A (In-domain) | Test B (Out-of-domain) | Test C (Out-of-domain) |
|---|---|---|---|---|---|
| (1). | `MKN3` | 81.38 | 69.36 | 167.61 | 189.93 |
| (2). | `HPY3` | 79.32 | 67.50 | 158.13 | 175.63 |
| (3). | `RNN` | **69.49** | **60.78** | 145.05 | 158.57 |
| (4). | `LW-NA` | 79.64 | 66.93 | **141.34** | 147.87 |
| (5). | `LW-VA` | 86.84 | 74.50 | 142.49 | **133.97** |
| (6). | `HPY3+RNN` | **64.01** | **55.84** | 122.52 | 142.62 |
| (7). | `HPY3+LW-NA` | 72.86 | 62.05 | 134.65 | 141.23 |
| (8). | `HPY3+LW-VA` | 66.95 | 57.78 | 106.73 | 102.34 |
| (9). | `HPY3+LW-NA+LW-VA` | 65.72 | 56.05 | **102.21** | **100.36** |

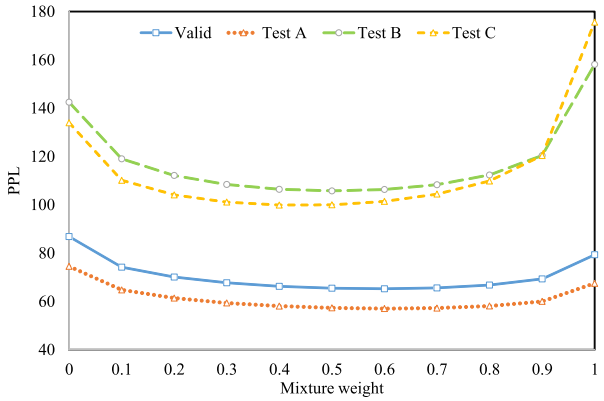


**Fig. 3** Perplexity results of combining `HPY3` and `LW-VA` in experiment 2.

**Table 5** Word error rate [%] results on each data sets in experiment 2.

| | 1-pass (WFST based decoding) | 2-pass (1000-best rescoring) | Valid (In-domain) | Test A (In-domain) | Test B (Out-of-domain) | Test C (Out-of-domain) |
|---|---|---|---|---|---|---|
| (1). | MKN3 | - | 19.98 | 24.79 | 38.67 | 32.00 |
| (2). | HPY3 | - | 19.74 | 24.67 | 38.29 | 31.69 |
| (3). | HPY3 | +RNN | 18.53 | 23.45 | 37.45 | 30.89 |
| (4). | HPY3 | LW-VA | 22.46 | 27.55 | 41.03 | 33.45 |
| (5). | HPY3 | +LW-VA | 19.23 | 23.69 | 37.04 | 30.44 |
| (6). | LW-NA | - | 19.61 | 24.54 | 36.93 | 30.42 |
| (7). | LW-NA | +LW-VA | 19.14 | 23.82 | 36.28 | 29.73 |
| (8). | HPY3+LW-NA | - | 18.65 | 23.58 | 35.99 | 28.74 |
| (9). | HPY3+LW-NA | +RNN | 17.85 | 22.68 | 35.36 | 28.06 |
| (10). | HPY3+LW-NA | +LW-VA | 18.45 | 22.84 | 35.36 | 28.15 |
| (11). | HPY3+LW-NA | +LW-VA+RNN | **17.75** | **22.52** | **35.26** | **28.00** |

sults were caused by degradation of tokenizing sentences compared with default ones in CSJ. In fact, we had to conduct re-tokenization of sentences to fairly evaluate WERs of out-of-domain data sets using LMs trained from the CSJ. In addition, WER results in out-of-domain data sets were relatively higher than those in in-domain data sets. This indicates that the contact center task and the voice mail task were much different from the CSJ. Our evaluation aims to clarify that the domain mismatch can be mitigated by introducing LWLMs. Therefore, we did not apply any adaptation methods to the LMs for improving the baseline of the out-of-domain data sets.

In lines (2)–(5), HPY3 was employed in the first pass. Line (4) where HPY3 was used to generate ASR hypotheses and LW-VA was only used as the LM score in the second pass, show the ASR performance was substantially degraded in all data sets compared to line (2) although their PPL performance was comparable. This suggested that the ASR performance of LW-VA was hardly related to its PPL performance. Actually, for improving ASR performance, it is important to increase a score difference between a correct word sequence and a misrecognized word sequence. Therefore, we can conclude that the Viterbi probability of the correct word sequence was comparable to that of the misrecognized word sequence.

On the other hand, in line (5), combining HPY3 and LW-VA provided a higher performance than using them singly. The WER difference between HPY3 and HPY3+LW-VA in each test set was statistically significant ($p < 0.01$). It seems that taking account of the latent word assignment of the recognition hypothesis yields characteristics different from only using HPY3. In out-of-domain tasks, HPY3+LW-VA outperformed that of HPY3+RNN while HPY3+RNN was superior to HPY3+LW-VA. These results verify that the Viterbi approximation of LWLM can robustly perform in out-of-domain data sets compared with RNNLM.

In lines (6)–(7), LW-NA was used in the first pass. The WER results show combining LW-VA and LW-NA improved ASR performance compared with using them singly. In terms of WER, statistically significant performance improvements ($p < 0.05$) were achieved by LW-NA+LW-VA compared to LW-NA in Test A. This performance might be attributed to the efficiency of the proposed Viterbi approximation that directly takes account of the latent words. It indicates that the Viterbi approximation possesses properties different from those of the n-gram approximation. This confirms that both implementations should be simultaneously used to achieve high ASR performance.

In lines (8)–(11), HPY3+LW-NA was used in the first pass. The WER results show HPY3+LW-NA clearly outperformed MKN3 and HPY3 in all data sets. The best LWLM-based performance was achieved when HPY3+LW-NA was used in the first pass and LW-VA+RNN was used in the second pass. Actually, RNNLM is known to be good at capturing long-range context information while the Viterbi approximation of LWLM is suitable for taking into account the validity of a latent word sequence behind a recognition hypothesis. The results verify that the combination of these two properties is effective for improving ASR performance of both in-domain tasks and out-of-domain tasks.

## 6. Conclusions

In this paper, the Viterbi approximation of LWLMs was proposed. The Viterbi approximation is implemented as a two-pass process in which several recognition hypotheses based on standard decoding using a smoothed n-gram LM are initially created. These hypotheses are then rescored using the joint probability between the recognition hypothesis and the optimal latent word assignment. The optimal latent word assignment is determined using Gibbs sampling which runs more rapidly than the Viterbi algorithm. Experiments showed that the Viterbi approximation was effective when it was combined with the first pass results. Moreover, the combination of the n-gram approximation method and Viterbi approximation method improved ASR performance.

### References


[1] Goodman, J.T.: A bit of progress in language modeling, *Computer Speech & Language*, Vol.15, pp.403–434 (2001).

[2] Masumura, R., Asami, T., Oba, T., Masataki, H., Sakauchi, S. and Ito, A.: Combinations of Various Language Model Technologies including Data Expansion and Adaptation in Spointaneous Speech Recognition, *Proc. INTERSPEECH*, pp.463–467 (2015).

[3] Masumura, R., Asami, T., Oba, T., Masataki, H., Sakauchi, S. and Ito, A.: Investigation of Combining Various Major Language Model Technologies including Data Expansion and Adaptation, *IEICE Trans. Information and Systems*, Vol.E99-D, No.10, pp.2452–2461 (2016).

[4] Chen, S.F. and Goodman, J.: An Empirical Study of Smoothing techniques for language modeling, *Computer Speech & Language*, Vol.13, pp.359–383 (1999).

[5] Teh, Y.W.: A Hierarchical Bayesian Language Model based on Pitman-Yor Processes, *Proc. Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics* (*COLING/ACL*), pp.985–992 (2006).

[6] Brown, P.F., deSouza, P.V., Mercer, R.L., Pietra, V.J.D. and Lai, J.C.: Class-based n-gram models of natural language, *Computational Linguistics*, Vol.18, pp.467–479 (1992).

[7] Potamianos, G. and Jelinek, F.: A study of n-gram and decision tree

letter language modeling methods, *Speech Communication*, Vol.24, No.3, pp.171–192 (1998).
[8] Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C.: A neural probabilistic language model, *Journal of Machine Learning Research*, Vol.3, pp.1137–1155 (2003).
[9] Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. and Khudanpur, S.: Recurrent Neural Network based Language Model, *Proc. Annual Conference of the International Speech Communication Association* (*INTERSPEECH*), pp.1045–1048 (2010).
[10] Mikolov, T., Stefan, S.K., Burget, L., Cernocky, J. and Khudanpur, S.: Extensions of Recurrent Neural Network Language Model, *Proc. International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp.5528–5531 (2011).
[11] Sundermeyer, M., Ney, H. and Schluter, R.: From Feedforward to Recurrent LSTM Neural Networks for language models, *IEEE/ACM Trans. Audio, Speech and Language Processing*, Vol.23, No.3, pp.517–529 (2015).
[12] Deschacht, K., Belder, J.D. and Moens, M.-F.: The latent words language model, *Computer Speech & Language*, Vol.26, pp.384–409 (2012).
[13] Goldwater, S. and Griffiths, T.: A fully Bayesian approach to unsupervised part-of-speech tagging, *Proc. Annual Meeting of the Association for Computational Linguistics* (*ACL*), pp.744–751 (2007).
[14] Blunsom, P. and Cohn, T.: A Hierarchical Pitman-Yor Process HMM for Unsupervised Part of Speech Induction, *Proc. Annual Meeting of the Association for Computational Linguistics* (*ACL*), pp.865–874 (2011).
[15] Su, Y.: Bayesian Class-Based Language Models, *Proc. International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), pp.5564–5567 (2011).
[16] Chien, J.-T. and Chueh, C.-H.: Dirichlet class language models for speech recognition, *IEEE Trans. Audio, Speech and Language Processing*, Vol.19, No.3, pp.1352–1365 (2011).
[17] Masumura, R., Asami, T., Oba, T., Masataki, H. and Sakauchi, S.: Mixture of Latent Words Language Models for Domain Adaptation, *Proc. INTERPSEECH*, pp.1425–1429 (2014).
[18] Masumura, R., Asami, T., Oba, T., Masataki, H., Sakauchi, S. and Ito, A.: Domain Adaptation based on Mixture of Latent Words Language Models for Automatic Speech Recognition, *IEICE Trans. Information and Systems*, Vol.E101-D, No.6, pp.1581–1590 (2018).
[19] Masumura, R., Masataki, H., Oba, T., Yoshioka, O. and Takahashi, S.: Use of latent words language models in ASR: A sampling-based implementation, *Proc. International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), pp.8445–8449 (2013).
[20] Masumura, R., Adami, T., Oba, T., Masataki, H., Sakauchi, S. and Takahashi, S.: N-gram Approximation of Latent Words Language Models for Domain Robust Automatic Speech Recognition, *IEICE Trans. Information and Systems*, Vol.E99-D, No.10, pp.2452–2461 (2016).
[21] Rabiner, L.R.: A tutorial on hidden Markov models and selected application in speech recognition, *Proc. IEEE*, Vol.77, No.2, pp.257–286 (1989).
[22] Forney, G.D.: The Viterbi Algorithm, *Proc. IEEE*, Vol.61, No.3, pp.268–278 (1973).
[23] Casella, G. and George, E.I.: Explaining the Gibbs sampler, *The American Statistician*, Vol.46, pp.167–174 (1992).
[24] Robert, C.P., Celeux, G. and Diebolt, J.: Bayesian Estimation of Hidden Markov Chains: A Stochastic Implementation, *Statistics and Probability Letters*, Vol.16, pp.77–83 (1993).
[25] Scott, S.L.: Bayesian methods for hidden Markov models: Recursive computing in the 21st century, *Journal of the American Statistical Association*, Vol.97, pp.337–351 (2002).
[26] Masumura, R., Oba, T., Masataki, H., Yoshioka, O. and Takahashi, S.: Viterbi Decoding for Latent Words Language Models Using Gibbs Sampling, *Proc. Annual Conference of the International Speech Communication Association* (*INTERSPEECH*), pp.3429–3433 (2013).
[27] Marcus, M.P., Marcinkiewicz, M.A. and Santorini, B.: Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, Vol.19, pp.313–330 (1993).
[28] Emami, A. and Jelinek, F.: Random clusterings for language modeling, *Proc. International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP*), Vol.1, pp.581–584 (2005).
[29] Xu, P. and Jelinek, F.: Random forests in language modeling, *Proc. Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pp.325–332 (2004).
[30] Teh, Y.W., Jordan, M.I., Beal, M.J. and Blei, D.M.: Hierarchical Dirichlet processes, *Journal of the American Statistical Association*, Vol.101, pp.1566–1581 (2006).
[31] MacKay, D.J.C. and Peto, L.C.: A hierarchical Dirichlet language model, *Natural Language Engineering*, Vol.1, pp.289–308 (1994).
[32] Stolcke, A.: Entropy-based pruning of backoff language models, *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, pp.270–274 (1998).
[33] Maekawa, K., Koiso, H., Furui, S. and Isahara, H.: Spontaneous speech corpus of Japanese, *Proc. Language Resources and Evaluation Conference* (*LREC*), pp.947–952 (2000).
[34] Fuchi, T. and Takagi, S.: Japanese morphological analyzer using word co-occurrence: JTAG, *Proc. COLING/ACL*, pp.409–413 (1998).
[35] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. and Kingsbury, B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition, *Signal Processing Magazine*, pp.1–27 (2012).
[36] Mohri, M., Pereira, F. and Riley, M.: Weighted finite-state transducers in speech recognition, *Computer Speech & Language*, Vol.16, pp.69–88 (2002).
[37] Hori, T., Hori, C., Minami, Y. and Nakamura, A.: Efficient WFST-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition, *IEEE Trans. Audio, Speech and Language Processing*, Vol.15, No.4, pp.1352–1365 (2007).

**Ryo Masumura** received B.E., M.E., and Ph.D. degrees in engineering from Tohoku University, Sendai, Japan, in 2009, 2011, 2016, respectively. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2011, he has been engaged in research on speech recognition, spoken language processing, and natural language processing. He received the Student Award and the Awaya Kiyoshi Science Promotion Award from ASJ in 2011 and 2013, respectively, the Sendai Section Student Awards The Best Paper Prize from IEEE in 2011, the Yamashita SIG Research Award from IPSJ in 2014, the Young Researcher Award from NLP in 2015, and the ISS Young Researcher's Award in Speech Field from IEICE in 2015. He is a member of ASJ, IPSJ, NLP, IEEE, and ISCA.

**Taichi Asami** received B.E. and M.E. degrees in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2004 and 2006, respectively. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 2006, he has been engaged in research on speech recognition and spoken language processing. In 2017, he started development of spoken dialogue services at NTT Docomo Corporation. He received the Awaya Kiyoshi Science Award and the Sato Prize Paper Award from ASJ in 2012 and 2014, respectively, the ISS Young Researcher's Award in Speech Field from IEICE in 2015, and the Industrial Achievement Award from IPSJ in 2017. He is a member of IPSJ, IEICE, ASJ, and IEEE.

**Takanobu Oba** received B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 2002 and 2004, respectively. He received his Ph.D. (Eng.) degree from Tohoku University in 2011. In 2004, he joined Nippon Telegraph and Telephone Corporation (NTT), where he was engaged in the research and development of spoken language processing technologies including speech recognition at the NTT Communication Science Laboratories, Kyoto, Japan. In 2012, he started the research and development of spoken applications at the NTT Media Intelligence Laboratories, Yokosuka, Japan. From 2015 to 2018, he was engaged in the development of spoken dialogue services at the NTT Docomo Corporation. Since 2018, he has been started the research and development at the NTT Corporation again. He received the Awaya Kiyoshi Science Promotion Award from ASJ in 2007. He is a member of IEICE and ASJ.

**Hirokazu Masataki** received B.E., M.E., and Ph.D. degrees from Kyoto University in 1989, 1991, and 1999, respectively. From 1995 to 1998, he worked with ATR Interpreted Telecommunications Research Laboratories, where he specialized in statistical language modeling for large vocabulary continuous speech recognition. He joined Nippon Telegraph and Telephone Corporation (NTT) in 2004 and has been engaged in the practical use of speech recognition. Since 2018, he has been engaged in development of speech recognition services at NTT TechnoCross Corporation. He received the Maejima Hisoka Award from the Tsushinbunko Association in 2013, and the 54-th Sato Prize Paper Award from ASJ in 2014. He is a member of IEICE and ASJ.

**Sumitaka Sakauchi** received an M.S. degree from Tohoku University in 1995 and a Ph.D. degree from Tsukuba University in 2005. Since joining Nippon Telegraph and Telephone Corporation (NTT) in 1995, he has been engaged in research on acoustics, speech and signal processing. He is now project manager of the Audio, Speech and Language Media Project, NTT Media Intelligence Laboratories. He received the Paper Award from IEICE in 2001, and Awaya Kiyoshi Science Promotion Award from ASJ in 2003. He is a member of IEICE and ASJ.