PAPER

Latent Words Recurrent Neural Network Language Models for Automatic Speech Recognition

Ryo MASUMURA[†a)], **Taichi ASAMI**[†], **Takanobu OBA**[†], **Sumitaka SAKAUCHI**[†], ***and*** **Akinori ITO**[††], ***Members***

SUMMARY This paper demonstrates latent word recurrent neural network language models (LW-RNN-LMs) for enhancing automatic speech recognition (ASR). LW-RNN-LMs are constructed so as to pick up advantages in both recurrent neural network language models (RNN-LMs) and latent word language models (LW-LMs). The RNN-LMs can capture long-range context information and offer strong performance, and the LW-LMs are robust for out-of-domain tasks based on the latent word space modeling. However, the RNN-LMs cannot explicitly capture hidden relationships behind observed words since a concept of a latent variable space is not present. In addition, the LW-LMs cannot take into account long-range relationships between latent words. Our idea is to combine RNN-LM and LW-LM so as to compensate individual disadvantages. The LW-RNN-LMs can support both a latent variable space modeling as well as LW-LMs and a long-range relationship modeling as well as RNN-LMs at the same time. From the viewpoint of RNN-LMs, LW-RNN-LM can be considered as a soft class RNN-LM with a vast latent variable space. In contrast, from the viewpoint of LW-LMs, LW-RNN-LM can be considered as an LW-LM that uses the RNN structure for latent variable modeling instead of an n-gram structure. This paper also details a parameter inference method and two kinds of implementation methods, an n-gram approximation and a Viterbi approximation, for introducing the LW-LM to ASR. Our experiments show effectiveness of LW-RNN-LMs on a perplexity evaluation for the Penn Treebank corpus and an ASR evaluation for Japanese spontaneous speech tasks.

key words: *latent words recurrent neural network language models, n-gram approximation, Viterbi approximation, automatic speech recognition*

1. Overview

Language models (LMs) are essential for many natural language processing tasks including automatic speech recognition (ASR) and statistical machine translation. The most common problems faced by LMs are data sparseness and context constraints. In most ASR systems, target domain training data sets are often limited since the data sets must be obtained by manually transcribing speech samples. In addition, ASR systems consider only short context information when calculating the generative probabilities of words because they often use a traditional n-gram language modeling. In order to mitigate these problems, several techniques have been proposed [1]–[3]. In particular, approaches intended to improve LM structure have been aggressively pursued. LM structure can be split into two main types; discriminative models and generative models.

The former include intelligent models such as maximum entropy model [4], decision tree [5], random forest [6], and neural networks [7], [8] including deep neural networks [9]. Recurrent neural network LMs (RNN-LMs) have, in particular, demonstrated significant improvements in recent years [10], [11]. RNN-LMs can capture long-range context information via their recurrent structure and can efficiently represent context information as a continuous vector. Therefore, RNN-LMs are known to be one of the most powerful LMs. On the other hand, among the generative models, latent variable space modeling with Bayesian approach is attracting attention. Soft class-based LMs that have a latent variable space have demonstrated better performance than traditional class n-gram models [12], [13]. In addition, latent words LMs (LW-LMs) offer a more flexible form of the soft class-based LMs [14]. LW-LMs have a vast latent variable space whose size is equivalent to the vocabulary size of the training data. This flexible attributes helps to mitigate data sparseness efficiently. In fact, LW-LMs are robust for not only in-domain tasks but also out-of-domain tasks [15]–[17].

This paper focuses on two successful LMs, RNN-LMs and LW-LMs. Although both models have their own advantages, each also has problems at the same time. RNN-LMs can capture long-range context information and offer strong performance. However, RNN-LMs cannot explicitly capture the hidden relationship behind observed words since they ignore the concept of the latent variable space. While LW-LMs have a latent variable space based on Bayesian inference, the space is modeled as a simple n-gram structure. Therefore, LW-LMs cannot take into account the long-range relationships among latent variables.

To overcome the problems of both RNN-LMs and LW-LMs, this paper presents a novel modeling method called the latent word recurrent neural network language model (LW-RNN-LM). Our idea is to combine RNN-LM and LW-LM so as to ameliorate their individual disadvantages. Thus, LW-RNN-LM can support both latent variable space modeling and LW-LM while providing long-range relationship modeling as well as RNN-LMs at the same time. From the viewpoint of RNN-LMs, LW-RNN-LM can be considered as a soft class RNN-LM with a latent word space. In contrast, from the viewpoint of LW-LMs, LW-RNN-LM can be considered as an LW-LM whose latent word space modeling is based on an RNN structure instead of an n-gram structure. Consequently, it can be expected that LW-RNN-LM

Manuscript received July 9, 2018.
Manuscript revised April 10, 2019.
Manuscript publicized September 25, 2019.

[†]The authors are with NTT Media Intelligence Laboratories, NTT Corporation, Yokosuka-shi, 239–0847 Japan.
[††]The author is with the Graduate School of Engineering, Tohoku University, Sendai-shi, 980–8579 Japan.
a) E-mail: ryou.masumura.ba@hco.ntt.co.jp

DOI: 10.1587/transinf.2018EDP7242

has different attributes from standard RNN-LMs and standard LW-LMs and will yield further improvement through their combination.

Our idea parallels studies that expand RNN-LMs using other models. Latent Dirichlet allocation has been used to exploit the topic representation from complete speech input [18]. In addition, paraphrase LMs have been combined with RNN-LMs to create paraphrase RNN-LMs [19]. To the best of our knowledge, LW-RNN-LM is the first proposal to combine RNN-LM with latent variable space models.

There are two issues posed by LW-RNN-LM. The first is which training method should be used. It is impossible to estimate an LW-RNN-LM directly from a training data set since the RNN structure is not suitable for latent variable space modeling. To this end, an alternative training procedure is presented that uses a standard LW-LM. In the procedure, latent word sequences of the training data set are first decoded using the LW-LM, and then model parameters of the LW-RNN-LM are estimated using the latent word sequences. The second issue is ASR implementation. As is true for LW-LMs, LW-RNN-LM cannot be applied to ASR directly because its latent word space is vast. To this end, this paper presents two approximation methods, an n-gram approximation that converts a complex model structure into a back-off n-gram structure [15], [17], and a Viterbi approximation that uses a joint probability between an observed word sequence and an optimal latent word sequence [16]. While these two methods were originally applied to LW-LMs, this paper customizes them to suit LW-RNN-LM.

This paper is an extended study of our previous work [20]. It provides details of LW-RNN-LM, omitted from the previous work, that allow better understanding of its position relative to RNN-LM and LW-LM. Furthermore, we provide an additional evaluation that more fully reveals the properties of LW-RNN-LM.

This paper is organized as follows. In Sect. 2, we describe the model structures of RNN-LM and LW-LM. Section 3 provides a definition of LW-RNN-LM. In addition, a training method and two ASR implementation methods for LW-RNN-LM are introduced in detail. Sections 4 and 5 describe a perplexity evaluation and an ASR evaluation. Section 6 concludes this paper.

2. Previous Work

2.1 Recurrent Neural Network Language Models

Recurrent neural network LMs (RNN-LMs) have attracted significant attention in recent years [10]. RNN-LMs have two characteristics: one is that the word space can be represented as a continuous space vector based on neural networks, and the other is that long-range information can be flexibly taken into consideration based on its recurrent structure.

A graphical rendering of RNN-LM is shown in Fig. 1. The gray circle denotes a word that can be observed as a linguistic phenomenon. The gray square denotes a continuous

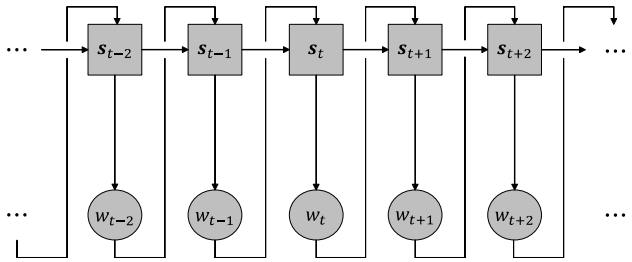

Fig. 1 Model structure of RNN-LMs.

representation that can be uniquely calculated. As shown in Fig. 1, previous word w_{t-1} is converted into a hidden representation $\boldsymbol{s}_t$ that depends on previous context representation $\boldsymbol{s}_{t-1}$, which includes long-range context information. Current word w_t is generated depending on context information $\boldsymbol{s}_t$.

In word-based RNN-LMs, the generative probability of word sequence $\boldsymbol{w} = \{w_1, \cdots, w_T\}$ is given as:

$$P(\boldsymbol{w}) = \prod_{t=1}^{T} P(w_t|w_{t-1}, \boldsymbol{s}_{t-1}, \boldsymbol{\Theta}_{\text{rnn}}), \tag{1}$$

$$= \prod_{t=1}^{T} P(w_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\text{rnn}}), \tag{2}$$

where $\boldsymbol{\Theta}_{\text{rnn}}$ is a model parameter of RNN-LM and $\boldsymbol{s}_t$ is the context information of the RNN structure. In the RNN structure, context information $\boldsymbol{s}_t$ and $P(w_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\text{rnn}})$ are calculated as:

$$\boldsymbol{s}_t = \boldsymbol{\sigma}(\boldsymbol{U}_{\text{rnn}}\boldsymbol{s}_{t-1} + \boldsymbol{V}_{\text{rnn}}\boldsymbol{\phi}(w_{t-1}) + \boldsymbol{b}_{\text{rnn}}), \tag{3}$$

$$\boldsymbol{p}_t = [p_{t1}, \cdots, p_{tk}, \cdots, p_{t|\mathcal{V}|}]^\top, \tag{4}$$

$$= \mathcal{G}(\boldsymbol{O}_{\text{rnn}}\boldsymbol{s}_t + \boldsymbol{o}_{\text{rnn}}), \tag{5}$$

$$P(w_t = k|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\text{rnn}}) = p_{tk}, \tag{6}$$

where $\boldsymbol{U}_{\text{rnn}}$, $\boldsymbol{V}_{\text{rnn}}$, and $\boldsymbol{b}_{\text{rnn}}$ are model parameters of a hidden layer of the RNN structure. $\boldsymbol{O}_{\text{rnn}}$ and $\boldsymbol{o}_{\text{rnn}}$ are model parameters of an output layer of the RNN structure. $\boldsymbol{p}_t$ is an output vector in the output layer of the RNN structure, p_{tk} is the k-th value in $\boldsymbol{p}_t$, and $|\mathcal{V}|$ is vocabulary size. $\boldsymbol{\phi}(w_{t-1})$ denotes 1-of-K coding of w_{t-1}. $\boldsymbol{\sigma}$ is a sigmoid function, and $\mathcal{G}$ is a softmax function.

Since the cost of computing the probability estimation is proportional to vocabulary size $|\mathcal{V}|$ in word-based modeling, class-based RNN-LMs are used most often [11]. The idea was proposed for maximum entropy models and feed forward neural networks [21], [22]. The resulting probability estimation is defined as:

$$P(w_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\text{rnn}}) = P(w_t|\boldsymbol{s}_t, c_t, \boldsymbol{\Theta}_{\text{rnn}})P(c_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\text{rnn}}), \tag{7}$$

where $\boldsymbol{s}_t$ is context information that includes the previous word and previous output in the hidden layer, and $c_t \in C$ is a word class where C represents the sets of classes and the class size is $|C|$.

Note that the mixing of several RNNs trained with different random initialization values is reported in [11]. The

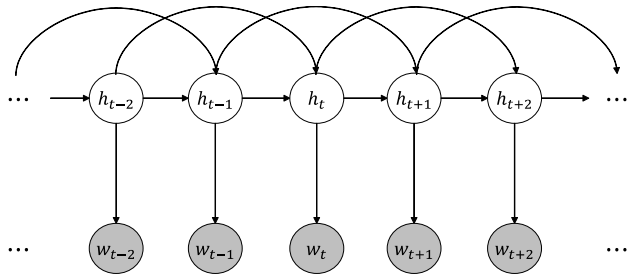

Fig. 2 Model structure of LW-LMs.

probability estimation of the ensemble of RNN-LMs is defined as:

$$P(\boldsymbol{w}) = \frac{1}{M}\sum_{m=1}^{M}\prod_{t=1}^{T} P(w_t|\boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathrm{rnn}}^m). \tag{8}$$

The generative probability can be calculated using M instances of RNN-LMs. $\boldsymbol{\Theta}_{\mathrm{rnn}}^m$ indicates the m-th model parameter and $\boldsymbol{s}_t^m$ is the context information of the m-th instance.

2.2 Latent Words Language Models

Latent words LMs (LW-LMs) are generative models that set a latent variable for each observed word. A graphic rendering of LW-LM is shown in Fig. 2. The gray circles denote observed words and the white circles denote latent variables.

In the generative process of LW-LM, 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 n-gram order. Next, observed word w_t is generated depending on the emission probability distribution given latent word h_t, i.e.,

$$h_t \sim P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}), \tag{9}$$
$$w_t \sim P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}), \tag{10}$$

where $\boldsymbol{\Theta}_{\mathtt{lw}}$ is a model parameter of LW-LM. 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 LW-LMs is that the latent word is expressed as a specific word that can be selected from complete vocabulary $\mathcal{V}$. Thus, the number of latent words is the same as vocabulary size $|\mathcal{V}|$. For this reason, the latent variable is called a latent word.

LW-LM is generally modeled using the Bayesian approach. LW-LM produces the following generative probability for observed words $\boldsymbol{w} = \{w_1, \cdots, w_T\}$:

$$P(\boldsymbol{w}) = \int \sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{\Theta}_{\mathtt{lw}}) d\boldsymbol{\Theta}_{\mathtt{lw}}, \tag{11}$$
$$\simeq \frac{1}{M}\sum_{m=1}^{M}\sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lw}}^m) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\mathtt{lw}}^m), \tag{12}$$
$$= \frac{1}{M}\sum_{m=1}^{M}\sum_{\boldsymbol{h}}\prod_{t=1}^{T} P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m) P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m), \tag{13}$$

where $\boldsymbol{h} = \{h_1, \cdots, h_T\}$ is a latent word assignment. $\boldsymbol{\Theta}_{\mathtt{lw}}^m$ means the m-th instance of the point estimated model parameter. Thus, the generative probability can be approximated using M instances of $\boldsymbol{\Theta}_{\mathtt{lw}}^m$.

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, $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m)$ and $P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m)$ can be calculated.

To estimate the latent word assignments, Gibbs sampling can be utilized. 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:

$$P(h_t|\mathcal{W}, \mathcal{H}_{-t}) \propto P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw},-t}) \prod_{j=t}^{t+n-1} P(h_j|\boldsymbol{l}_j, \boldsymbol{\Theta}_{\mathtt{lw},-t}), \tag{14}$$

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:

$$P(h_t|\boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}) = P(h_t|\boldsymbol{l}_t, \mathcal{H}), \tag{15}$$
$$P(h_t|\boldsymbol{l}_t, \mathcal{H}) = \frac{c(h_t, \boldsymbol{l}_t) - d_{|\boldsymbol{l}_t|}s(h_t, \boldsymbol{l}_t)}{\theta_{|\boldsymbol{l}_t|} + c(\boldsymbol{l}_t)} + \frac{\theta_{|\boldsymbol{l}_t|} + d_{|\boldsymbol{l}_t|}s(\boldsymbol{l}_t)}{\theta_{|\boldsymbol{l}_t|} + c(\boldsymbol{l}_t)} P(h_t|\pi(\boldsymbol{l}_t), \mathcal{H}), \tag{16}$$

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 [23]. $d_{|\boldsymbol{l}_t|}$ 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 [24]. $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}})$ is given as:

$$P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}) = P(w_t|h_t, \mathcal{W}, \mathcal{H}), \tag{17}$$
$$P(w_t|h_t, \mathcal{W}, \mathcal{H}) = \frac{c(w_t, h_t) + \alpha P(w_t)}{c(h_t) + \alpha}, \tag{18}$$

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 α can be optimized via a validation data set.

3. Latent Words Recurrent Neural Network Language Models

3.1 Definition

Latent words recurrent neural network LMs (LW-RNN-LMs) are generative models that combine RNN-LM and LW-LM. The models have a soft class structure based on a latent word space as does LW-LM and the latent word space is modeled using an RNN-LM.

A graphic rendering of LW-RNN-LM is shown in Fig. 3. Gray circles denote words and white circles denote latent words. White squares denote a hidden representation of latent words. As shown in Fig. 3, previous latent word h_{t-1} is converted into hidden representation $\boldsymbol{s}_t$ depending on previous context representation $\boldsymbol{s}_{t-1}$, which includes long-range context information. Current latent word h_t is generated depending on context information $\boldsymbol{s}_t$. Finally, observed word w_t is generated depending on latent word h_t. The generative process is formulated as:

$$h_t \sim P(h_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\mathtt{lr}}), \tag{19}$$

$$w_t \sim P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lr}}), \tag{20}$$

where $\boldsymbol{\Theta}_{\mathtt{lr}}$ is a model parameter of LW-RNN-LM. The transition probability distribution $P(h_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\mathtt{lr}})$ is expressed as an RNN-LM for latent words. The emission probability distribution $P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lr}})$ models the dependency between the observed word and the latent word as does LW-LM.

In the Bayesian approach, LW-RNN-LM defines the following generative probability for observed words $\boldsymbol{w} = \{w_1, \cdots, w_T\}$:

$$P(\boldsymbol{w}) = \int \sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lr}}) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\mathtt{lr}}) P(\boldsymbol{\Theta}_{\mathtt{lr}}) \mathrm{d}\boldsymbol{\Theta}_{\mathtt{lr}}, \tag{21}$$

$$\simeq \frac{1}{M} \sum_{m=1}^{M} \sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lr}}^m) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\mathtt{lr}}^m), \tag{22}$$

$$= \frac{1}{M} \sum_{m=1}^{M} \sum_{\boldsymbol{h}} \prod_{t=1}^{T} P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lr}}^m) P(h_t|\boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathtt{lr}}^m), \tag{23}$$

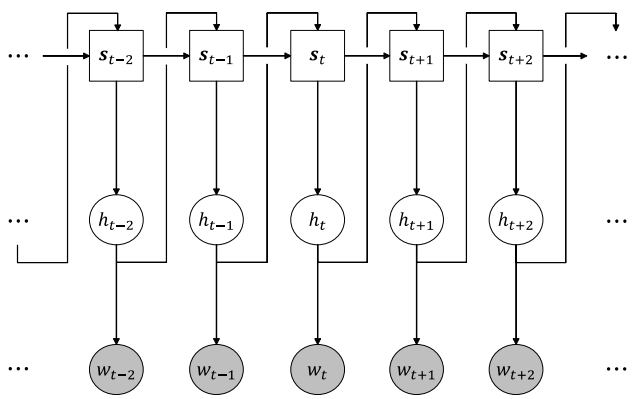

Fig. 3 Model structure of LW-RNN-LM.

where $\boldsymbol{\Theta}_{\mathtt{lr}}^m$ is the m-th model parameter and $\boldsymbol{s}_t^m$ is the context information of the m-th instance for the RNN structure. Thus, the generative probability can be approximated using M instances of $\boldsymbol{\Theta}_{\mathtt{lr}}^m$. In the m-th RNN structure, context information $\boldsymbol{s}_t^m$ and transition probability $P(h_t|\boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathtt{lr}}^m)$ are calculated as:

$$\boldsymbol{s}_t^m = \sigma(\boldsymbol{U}_{\mathtt{lr}}^m \boldsymbol{s}_{t-1}^m + \boldsymbol{V}_{\mathtt{lr}}^m \boldsymbol{\phi}(h_{t-1}) + \boldsymbol{b}_{\mathtt{lr}}^m), \tag{24}$$

$$\boldsymbol{p}_t^m = [p_{t1}^m, \cdots, p_{tk}^m, \cdots, p_{t|\mathcal{W}|}^m]^\top, \tag{25}$$

$$= \mathcal{G}(\boldsymbol{O}_{\mathtt{lr}}^m \boldsymbol{s}_t^m + \boldsymbol{o}_{\mathtt{lr}}^m), \tag{26}$$

$$P(h_t = k|\boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathtt{lr}}^m) = p_{tk}^m, \tag{27}$$

where $\boldsymbol{U}_{\mathtt{lr}}^m$, $\boldsymbol{V}_{\mathtt{lr}}^m$, and $\boldsymbol{b}_{\mathtt{lr}}^m$ are model parameters in a hidden layer of the m-th RNN structure. $\boldsymbol{O}_{\mathtt{lr}}^m$ and $\boldsymbol{o}_{\mathtt{lr}}^m$ are model parameters in an output layer of the m-th RNN structure. $\boldsymbol{p}_t^m$ is an output vector in the output layer of the m-th RNN structure. $\boldsymbol{\phi}(h_{t-1})$ denotes 1-of-K coding of h_{t-1}.

In addition, class-based RNN structure described in Sect. 2 is also utilized for LW-RNN-LM. In this case, latent words are additionally map into classes. When the class-based RNN structure is used in LW-RNN-LM, the generative probability of h_t is defined as:

$$P(h_t|\boldsymbol{s}_t, \boldsymbol{\Theta}_{\mathtt{lr}}^m) = P(h_t|\boldsymbol{s}_t^m, c_t, \boldsymbol{\Theta}_{\mathtt{lr}}^m) P(c_t|\boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathtt{lr}}^m), \tag{28}$$

where $c_t \in C$ represents the class where C represents the sets of classes and the class size is $|C|$.

LW-RNN-LM is closely related to conventional RNN-LMs and LW-LMs. LW-RNN-LM can be regarded as an RNN-LM with a soft class structure based on a latent word space. In addition, LW-RNN-LM can be regarded as an LW-LM whose transition probability distribution is based on an RNN structure instead of an n-gram structure.

3.2 Training

LW-RNN-LM is trained using an observed word sequence of a training data set $\mathcal{W} = \{w_1, \cdots, w_T\}$. It is impossible to estimate an LW-RNN-LM directly because it is founded on latent word space and RNN structure. Therefore, this paper presents a method that preliminarily trains an LW-LM to decode the latent word assignment of the observed word sequence. The latent word assignment is used in estimating model parameters of an LW-RNN-LM.

In fact, the optimal latent word assignment of an observed word sequence can be estimated for each LW-LM instance. The optimal latent word assignment for the m-th LW-LM instance, $\mathcal{H}^m = \{h_1^m, \cdots, h_T^m\}$, is given by:

$$\mathcal{H}^m = \arg\max_{\mathcal{H}} P(\mathcal{W}|\mathcal{H}, \boldsymbol{\Theta}_{\mathtt{lw}}^m) P(\mathcal{H}|\boldsymbol{\Theta}_{\mathtt{lw}}^m). \tag{29}$$

For the estimation required, Gibbs sampling is suitable. The conditional probability distribution of possible values for latent word h_t is given by:

$$P(h_t|\mathcal{W}, \mathcal{H}_{-t}) \propto P(w_t|h_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m) \prod_{j=t}^{t+n-1} P(h_j|\boldsymbol{l}_j, \boldsymbol{\Theta}_{\mathtt{lw}}^m), \tag{30}$$

where $\mathcal{H}_{-t}$ is a latent word assignment except for h_t. This procedure is applied to not only the training data set but also the validation data set.

The m-th model parameter $\mathbf{\Theta}^m_{\mathtt{lr}}$ can be determined from the m-th optimal latent word assignments $\mathcal{H}^m$. The transition probability distribution $P(h_t|\boldsymbol{s}^m_t, \mathbf{\Theta}^m_{\mathtt{lr}})$, i.e., $\boldsymbol{U}^m_{\mathtt{lr}}$, $\boldsymbol{V}^m_{\mathtt{lr}}$, $\boldsymbol{b}^m_{\mathtt{lr}}$, $\boldsymbol{O}^m_{\mathtt{lr}}$ and $\boldsymbol{o}^m_{\mathtt{lr}}$, is estimated from the m-th latent variable assignment of the training data set as in usual RNN-LM training. The m-th model parameter $\hat{\mathbf{\Theta}}^m_{\mathtt{lr}}$ is optimized by:

$$\hat{\mathbf{\Theta}}^m_{\mathtt{lr}} = \arg\max_{\mathbf{\Theta}_{\mathtt{lr}}} \prod_{t=1}^{T} P(h^m_t|\boldsymbol{s}^m_t, \mathbf{\Theta}_{\mathtt{lr}}). \tag{31}$$

Note that the m-th latent variable assignment of the validation data set is used for early stopping.

Emission probability distribution $P(w_t|h_t, \mathbf{\Theta}^m_{\mathtt{lr}})$ can be determined from the LW-LM:

$$P(w_t|h_t, \mathbf{\Theta}^m_{\mathtt{lr}}) = P(w_t|h_t, \mathbf{\Theta}^m_{\mathtt{lw}}), \tag{32}$$

$$= P(w_t|h_t, \mathcal{W}, \mathcal{H}^m), \tag{33}$$

$$P(w_t|h_t, \mathcal{W}, \mathcal{H}^m) = \frac{c(w_t, h^m_t) + \alpha P(w_t)}{c(h^m_t) + \alpha}, \tag{34}$$

where $P(w_t)$ is the estimated maximum likelihood value of unigram probability in training data set $\mathcal{W}$. $c(w_t, h^m_t)$ and $c(h^m_t)$ are counts calculated from $\mathcal{W}$ and latent word assignment $\mathcal{H}^m$. Hyper parameter α can be optimized via the validation data set.

3.3 N-Gram Approximation

An n-gram approximation can be used to apply LW-RNN-LM to ASR. The n-gram approximation is a technique that can convert an LM with complex model structure into a back-off n-gram structure. The n-gram approximation of LW-RNN-LM is the same as that for the LW-LM [15], [17]; an LM with a back-off n-gram structure is trained from observed words randomly sampled on LW-RNN-LM. The approximated LW-RNN-LM $P(\boldsymbol{w}|\mathbf{\Theta}_{\mathtt{lrng}})$ has the following properties:

$$\boldsymbol{w}_{\mathtt{lr}} \sim P(\boldsymbol{w}|\mathbf{\Theta}^1_{\mathtt{lr}}, \cdots, \mathbf{\Theta}^M_{\mathtt{lr}}), \tag{35}$$

$$\boldsymbol{w}_{\mathtt{lrng}} \sim P(\boldsymbol{w}|\mathbf{\Theta}_{\mathtt{lrng}}), \tag{36}$$

$$\boldsymbol{w}_{\mathtt{lr}} \simeq \boldsymbol{w}_{\mathtt{lrng}}, \tag{37}$$

where $\boldsymbol{w}_{\mathtt{lr}}$ is an observed word sequence generated from the LW-RNN-LM, and $\boldsymbol{w}_{\mathtt{lrng}}$ is an observed word sequence generated from the approximated LW-RNN-LM with back-off n-gram structure. The approximated LW-LM can be constructed from words generated from LW-RNN-LM.

The random sampling is based on Algorithm 1. As shown in the algorithm, context information of each RNN structure is updated in advance. After that, instance index $m_t \in \{1, \cdots, M\}$ for model parameters, a latent word $h_t \in \mathcal{V}$, and an observed word $w_t \in \mathcal{V}$ are recursively generated.

This iterative approach yields a large number of word sequences. T iterations can generate T latent words, and T observed words. The T observed words are used only for back-off n-gram model estimation.

Algorithm 1 Random sampling based on LW-RNN-LM.

Input: Model parameters $\mathbf{\Theta}^1_{\mathtt{lr}}, \cdots, \mathbf{\Theta}^M_{\mathtt{lr}}$, number of sampled words T
Output: Sampled words $\boldsymbol{w}$

$\boldsymbol{l}_1$ = <s>
for $t = 1$ to T **do**
  for $m = 1$ to M **do**
    $\boldsymbol{s}^m_t = \boldsymbol{\sigma}(\boldsymbol{s}^m_{t-1}, h_{t-1}, \mathbf{\Theta}^m_{\mathtt{lr}})$
  end for
  $m_t \sim P(m_t) = \frac{1}{\mathrm{M}}$
  $h_t \sim P(h_t|\boldsymbol{s}^{m_t}_t, \mathbf{\Theta}^{m_t}_{\mathtt{lr}})$
  $w_t \sim P(w_t|h_t, \mathbf{\Theta}^{m_t}_{\mathtt{lr}})$
end for
return $\boldsymbol{w} = w_1, \cdots, w_T$

3.4 Viterbi Approximation

The other applicable method is the Viterbi approximation that simultaneously decodes a recognition hypothesis and its latent word sequence using the joint probability between the two sequences. The joint probability is called the Viterbi probability. Viterbi probability $P(\boldsymbol{w}, \bar{\boldsymbol{h}})$ is defined as:

$$P(\boldsymbol{w}, \bar{\boldsymbol{h}}) = \max_{\boldsymbol{h}} P(\boldsymbol{w}, \boldsymbol{h}), \tag{38}$$

$$= \max_{\boldsymbol{h}} \frac{1}{M} \sum_{m=1}^{M} P(\boldsymbol{w}|\boldsymbol{h}, \mathbf{\Theta}^m_{\mathtt{lr}}) P(\boldsymbol{h}|\mathbf{\Theta}^m_{\mathtt{lr}}). \tag{39}$$

The Viterbi approximation of LW-RNN-LM is not the same as is true for LW-LMs. LW-RNN-LM makes it impossible to efficiently compute a Viterbi probability because neither the Viterbi algorithm nor Gibbs sampling can be introduced directly due to the RNN structure. In order to tackle this issue, a standard LW-LM is used for the Viterbi approximation. First, several latent word assignments are decoded using an LW-LM, and the candidates are re-evaluated using a LW-RNN-LM.

The latent word assignments can be decoded via Gibbs sampling. A conditional probability distribution of the possible values for latent word h_t is defined as:

$$P(h_t|\boldsymbol{w}, \boldsymbol{h}_{-t}) \propto \sum_{m=1}^{M} \{P(w_t|h_t, \mathbf{\Theta}^m_{\mathtt{lw}}) \prod_{j=t}^{t+n-1} P(h_j|\boldsymbol{l}_j, \mathbf{\Theta}^m_{\mathtt{lw}})\}, \tag{40}$$

where $\boldsymbol{h}_{-t}$ represents latent word assignment except for h_t. This sampling yields I samples of latent words assignments $\{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_I\}$. Viterbi probability $P(\boldsymbol{w}, \bar{\boldsymbol{h}})$ is based on LW-RNN-LM and is calculated as:

$$P(\boldsymbol{w}, \bar{\boldsymbol{h}}) = \max_{\boldsymbol{h} \in \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_I\}} \frac{1}{M} \sum_{m=1}^{M} \prod_{t=1}^{T} P(w_t|h_t, \mathbf{\Theta}^m_{\mathtt{lr}}) P(h_t|\boldsymbol{s}^m_t, \mathbf{\Theta}^m_{\mathtt{lr}}). \tag{41}$$

In addition, the Viterbi probability can be computed by combining LW-RNN-LM with LW-LM. That is, LW-RNN-LM and LW-LM are interpolated in the latent word space. The Viterbi probability is defined as:

$$P(\boldsymbol{w}, \bar{\boldsymbol{h}}) = \max_{\boldsymbol{h} \in \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_I\}} \frac{1}{M} \sum_{m=1}^{M} \prod_{t=1}^{T} P(w_t | h_t, \boldsymbol{\Theta}_{\mathtt{lr}}^m) \{\lambda P(h_t | \boldsymbol{s}_t^m, \boldsymbol{\Theta}_{\mathtt{lr}}^m) + (1-\lambda) P(h_t | \boldsymbol{l}_t, \boldsymbol{\Theta}_{\mathtt{lw}}^m)\}, \quad (42)$$

where λ is a mixture weight for latent word space modeling. As described above, the emission probabilities of both LW-RNN-LM and LW-LM are shared, so the transition probabilities are simply interpolated.

4. Experiment 1: Perplexity Evaluation

4.1 Setups

The first experiments used the Penn Treebank corpus in [25]. 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). In addition, a human-human discussion text data set (Test B) was prepared for evaluations in a domain different from the training data set. Each vocabulary was limited to 10K words and there were no out-of-vocabulary (OOV) words. Actually, the OOV words were map into a specific word, i.e, "UNK". The setups match those of many previous studies and allow us to evaluate perplexity of all words in a unified manner. Table 1 shows details.

In this evaluation, the following LMs were prepared.

- `MKN5`: A word-based 5-gram LM with modified Kneser-Ney smoothing constructed from the training data set [26].
- `HPY5`: A word-based 5-gram hierarchical Pitman-Yor LM (HPYLM) constructed from the training data set. For the training, 200 iterations were used for burn-in, and 10 instances were collected [27].
- `RNN`: Word-based RNN-LM [11]. The hidden layer size was set to 200 by referring to a preliminary experiment. The number of instances (M) was set to 1 since RNN-LMs with one instance were often used in most previous studies.
- `RNN-NA`: A word-based 5-gram HPYLM constructed from data generated on the basis of `RNN`. The generated data size was one billion words, which was determined in consideration of previous work [17]. We used entropy based pruning to n-gram entries that match the computation complexity of `HPY5` [28].

Table 1 Data sets for perplexity evaluation.

	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

- `LW-NA`: A word-based 5-gram HPYLM constructed from data generated on the basis of 5-gram LW-LM (`LW`) constructed from the training data set. For training `LW`, 500 iterations were used for burn-in, and 10 instances were collected ($M = 10$). The generated data size, one billion words, was determined in consideration of previous work [17]. We pruned n-gram entries as to be comparable computation complexity to `HPY5` using entropy based pruning.
- `LW-VA`: Viterbi approximation of `LW` [16]. To calculate the Viterbi probability, 100 samples of latent words assignments were obtained via Gibbs sampling ($I = 100$).
- `LR-NA`: A word-based 5-gram HPYLM constructed from data generated on the basis of LW-RNN-LM (`LR`) constructed from the training data set. Latent word space was modeled by an RNN structure. The hidden unit size and the class size $|C|$ in the RNN structure were varied in the evaluation. The generated data size was one billion words, which was determined in consideration of previous work [17].
- `LR-VA`: Viterbi approximation of `LR`. To calculate the Viterbi probability, 100 samples of latent words assignments were sampled using `LW` ($I = 100$).

In addition, several mixed models constructed by linearly interpolating the above LMs were employed. Note that M and I were constant in our experiments. Hyper parameters including α in Eq. (28) and λ in Eq. (36) and the interpolation weights were optimized using a validation data set.

4.2 Results

First, perplexity (PPL) results of `LR-NA` and `LR-VA` were investigated; hidden unit size and class size $|C|$ were varied. When the class size was set to 1,000, LW-RNN-LM is a class-based model. On the other hand, when the class size was set to 10,000, LW-RNN-LM is exactly a word-based model. We compared `LR-NA` and `LR-VA` with `LW-NA` and `LW-VA`. The results are shown in Table 2. The results show that PPL was improved when hidden unit size and class size were increased. The results indicate that rich parameters are necessary to construct precise LW-RNN-LMs. In the n-gram approximation results, `LR-NA` was inferior

Table 2 PPL results of `LW-NA`, `LW-VA`, `LR-NA` and `LR-VA`; hidden unit size and class size $|C|$ of RNN structure were varied for LW-RNN-LMs.

	Unit size	Class size	Valid	Test A	Test B
`LW-NA`	-	-	**138.7**	**131.7**	**205.5**
`LR-NA`	200	1,000	153.7	146.1	221.8
`LR-NA`	400	1,000	151.6	143.6	217.4
`LR-NA`	200	10,000	149.5	141.2	214.1
`LR-NA`	400	10,000	148.6	140.6	212.4
`LW-VA`	-	-	148.4	142.9	224.7
`LR-VA`	200	1,000	155.1	147.5	226.4
`LR-VA`	400	1,000	151.2	144.5	221.9
`LR-VA`	200	10,000	149.5	142.2	221.1
`LR-VA`	400	10,000	**147.1**	**139.89**	**216.6**

to LW-NA. On the other hand, in the Viterbi approximation results, LR-VA outperformed LW-VA in both the in-domain tasks and the out-of-domain tasks. It can be considered that the Viterbi approximation is an implementation method suitable for LW-RNN-LMs because it can directly utilize the RNN structure for computing the generative probabilities of words.

Next, combinations of LR-VA (400 hidden units and 10,000 classes) and LW-VA were examined. The PPL results in which the mixture weight was varied are shown in Fig. 4. When the mixture weight was set to 0, the PPL result corresponds to LW-VA. When the mixture weight is set to 1, the PPL result corresponds to LR-VA. The results show that the combination of LR-VA and LW-VA based on linear interpolation can improve the PPL for all data sets. This suggests that a combination of n-gram structure and RNN structure in a latent word space is effective as well as in an observed word space.

PPL results including those for other LMs and their combinations are summarized in Table 3. First, the model combination was examined under the restriction of the back-off n-gram structure. Lines 1–9 show the results for LMs with the back-off n-gram structure. LR-NA was superior to RNN-NA and slightly weaker than LW-NA in in-domain tasks and out-of-domain tasks. By combining LR-NA with RNN-NA or LW-NA, the PPL was improved compared to their individual use. These results show that LR-NA possesses properties different from RNN-NA and LW-NA and that their combination is effective. In each data set, HPY5+LW-NA+LR-NA outperformed HPY5+LW-NA. These results show the n-gram approximation of LW-RNN-LM is beneficial for constructing an LM with a back-off n-gram structure.

Lines 10–17 show the results for RNN, LW-VA, LR-VA and their combination with ALL5 (HPY5+LW-NA+LR-NA). We only used ALL5 as an n-gram LM when combining with LW-VA, LR-VA and RNN. RNN performed strongly compared to LW-VA and LR-VA. Combining LR-VA with ALL5 yielded improved PPL results. This indicates that the Viterbi approximation of LW-RNN-LM is useful for improving back-off n-gram language modeling. The highest performance was attained by ALL5+LW-VA+LR-VA+RNN for all data sets. It seems that RNN, LW-VA and LR-VA complement each other, and each model yields characteristics different from their n-gram approximation methods. These results confirm that LW-RNN-LM is beneficial for improving word prediction performance.

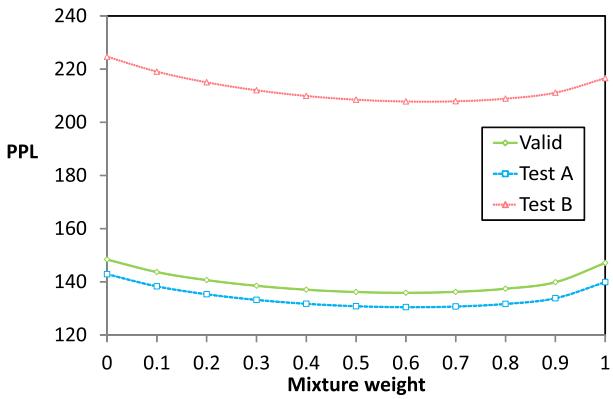

Fig. 4 PPL results of combining LR-VA and LW-VA.

Table 3 PPL results including those for other LMs and combined LMs.

		Valid	Test A	Test B
1.	MKN5	148.0	141.2	238.6
2.	HPY5	145.1	139.3	232.7
3.	RNN-NA	160.4	150.4	286.4
4.	LW-NA	138.7	131.7	205.5
5.	LR-NA	148.6	140.6	212.4
6.	LW-NA+LR-NA	136.5	129.3	197.3
7.	RNN-NA+LR-NA	136.5	129.3	197.3
8.	HPY5+LW-NA	144.3	137.2	205.7
9.	HPY5+LW-NA+LR-NA (ALL5)	**125.4**	**120.2**	**196.5**
10	LW-VA	148.4	142.9	224.7
11	LR-VA	147.1	139.8	216.6
12	LW-VA+LR-VA	138.0	132.3	211.3
13	RNN	134.4	128.9	212.9
14.	ALL5+LR-VA	105.0	102.4	165.6
15.	ALL5+LW-VA+LR-VA	103.9	101.6	164.2
16.	ALL5+RNN	108.7	103.2	172.7
17.	ALL5+RNN+LW-VA+LR-VA	**97.0**	**94.7**	**152.4**

5. Experiment 2: ASR Evaluation

5.1 Setups

The second experiment used the Corpus of Spontaneous Japanese (CSJ) [29]. CSJ was divided into a training data set (Train), a small validation data set (Valid), and a test data set (Test A). For evaluation in out-of-domain environments, a contact center dialog task (Test B) and a voice mail task (Test C) were prepared. 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 4.

For speech recognition evaluation, an acoustic model based on hidden Markov models with deep neural networks (DNN-HMM) was prepared [30]. The DNN-HMM had 8 hidden layers with 2048 nodes.

In this evaluation, the following LMs were prepared.

- MKN3: A word-based 3-gram LM with modified Kneser-Ney smoothing constructed from training data set [26].
- HPY3: A word-based 3-gram HPYLM constructed from the training data set [27].
- RNN: A class-based RNN-LM with 500 hidden nodes

Table 4 Data sets for ASR evaluation.

	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 5 PPL results of LW-NA, LW-VA, LR-NA and LR-VA; hidden layer size and class size $|C|$ were varied for LW-RNN-LMs.

	Unit size	Class size	Valid	Test A	Test B	Test C
LW-NA	-	-	**79.64**	**66.93**	141.34	147.87
LR-NA	200	200	90.69	75.85	142.40	146.55
LR-NA	200	500	89.95	75.38	140.81	145.88
LR-NA	200	1,000	92.22	76.51	141.47	146.37
LR-NA	500	500	90.17	75.17	**140.72**	**145.09**
LW-VA	-	-	86.84	74.50	**142.49**	**133.97**
LR-VA	200	200	87.95	74.41	156.49	163.55
LR-VA	200	500	86.49	73.53	154.15	162.16
LR-VA	200	1,000	86.49	73.23	152.31	162.54
LR-VA	500	500	**81.55**	**69.64**	146.35	158.38

and 500 classes by referring to a preliminary experiment [11]. Actually, we did not construct word-based RNN-LM because it is difficult to construct the word-based model from training data sets due to the computation complexity. The number of instances (M) was set to 1 since RNN-LMs with one instance were often used in most previous studies.

- RNN-NA: A word-based 3-gram HPYLM constructed from data generated on the basis of RNN. The generated data size was one billion words, which was determined in consideration of previous work [17]. We applied entropy based pruning to the n-gram entries to match the computation complexity of HPY3 [28].
- LW-NA: A word-based 3-gram HPYLM constructed from data generated on the basis of 3-gram LW-LM (LW) constructed from the training data set. For training LW, 500 iterations were used for burn-in, and 10 instances were collected ($M = 10$). The generated data size, one billion words, was determined in consideration of previous work [17]. We applied entropy based pruning as well as RNN-NA.
- LW-VA: Viterbi approximation of LW. To calculate the Viterbi probability, 100 samples of latent words assignments were obtained using Gibbs sampling ($I = 100$).
- LR-NA: A word-based 3-gram HPYLM constructed from data generated on the basis of LW-RNN-LM (LR) constructed from the training data set. Its latent word space was modeled by an RNN structure. The hidden unit size and the class size $|C|$ in the RNN structure were varied in the evaluation. The generated data size, one billion words, was determined in consideration of previous work [17]. We applied entropy based pruning as well as RNN-NA.
- LW-VA: Viterbi approximation of LR. To calculate the Viterbi probability, 100 samples of latent words assignments were sampled using LW ($I = 100$).

For implementing RNN, LW-VA, and LR-VA to ASR, 1,000-best hypotheses were generated in the first pass. Note that M and I were constant in our experiments. Hyper parameters including α in Eq. (28) and λ in Eq. (36) and the interpolation weights were optimized using a validation data set.

5.2 Results

First, PPL results of LR-NA and LR-VA were investigated; hidden unit size and class size $|C|$ were varied. The results, including LW-NA and LW-VA, are shown in Table 5.

Among the LW-RNN-LMs, the best results were attained with 500 hidden units and 500 classes for both the n-gram approximation and the Viterbi approximation. In in-domain tasks, LR-NA was inferior to LW-NA, and LR-VA was superior to LW-VA. On the other hand, in out-of-domain tasks, LR-NA was superior to LW-NA, and LR-VA was inferior to LW-VA. The results indicate that LW-RNN-LM is compatible with the Viterbi approximation while LW-LM is compatible with the n-gram approximation. This is because the Viterbi approximation directly utilizes the RNN structure for computing generative probabilities of words.

Table 6 shows the PPL and word error rate (WER) results for each condition. The results show that in-domain data sets were showed lower PPL and WER than out-of-domain data sets. This is because OOV rate in the in-domain data sets were lower than the in the out-of-domain data sets. First, one-pass decoding results gained from LMs with the back-off n-gram structure are examined. Lines 1–9 show the results for the back-off n-gram structure. The results show LR-NA outperformed LW-NA in out-of-domain tasks although the WER differences between LW-NA and LW-NA in out-of-domain tasks were not statistically significant ($p > 0.05$). On the other hand, LR-NA was weaker than LW-NA in in-domain tasks. LR-NA also outperformed RNN-NA in both in-domain tasks and out-of-domain tasks. The WER differences between LR-NA and RNN-NA in each test sets were statistically significant ($p < 0.01$). These results indicate that the n-gram approximation of LW-RNN-LM can yield ASR performance improvements although it is comparable to LW-LM. In addition, RNN-NA+LR-NA and LW-NA+LR-NA attained better ASR performance than RNN-NA or LW-NA. These results show the effectiveness of LW-RNN-LM compared with the conventional methods. It is thought that these improvements were attained because LW-RNN-LM has different attributes from standard RNN-LMs and standard LW-LMs. The highest ASR performance was attained by HPY3+LW-NA+LR-NA in each condition. In particular, in out-of-domain tasks, HPY3+LW-NA+LR-NA strongly out-

Table 6 PPL results and WER results [%] for in-domain tasks and out-of-domain tasks.

	Setup	Valid (In-domain)		Test A (In-domain)		Test B (Out-of-domain)		Test C (Out-of-domain)	
		PPL	WER	PPL	WER	PPL	WER	PPL	WER
1.	MKN3	81.38	19.98	69.36	24.79	167.61	38.67	189.93	32.00
2.	HPY3	79.32	19.74	67.50	24.67	158.13	38.29	175.63	31.69
3.	RNN-NA	98.65	21.63	82.23	26.24	153.89	39.32	163.99	31.96
4.	LW-NA	79.57	19.61	66.93	24.54	141.34	36.93	147.87	30.42
5.	LR-NA	90.17	19.89	75.17	25.30	140.72	36.64	145.09	29.75
6.	LW-NA+LR-NA	83.66	19.50	70.32	24.46	138.04	36.31	143.83	29.63
7.	RNN-NA+LR-NA	89.24	19.98	74.12	25.04	139.03	36.56	142.31	28.99
8.	HPY3+LW-NA	**72.86**	18.65	**62.05**	23.58	**134.65**	35.99	141.23	28.74
9.	HPY3+LW-NA+LR-NA (ALL3)	73.56	**18.60**	62.97	**23.42**	**132.87**	**35.76**	**139.51**	**28.62**
10	LW-VA	86.84	-	74.50	-	142.49	-	133.97	-
11	LR-VA	81.55	-	69.64	-	146.35	-	158.38	-
12	LW-VA+LR-VA	77.56	-	67.36	-	134.21	-	132.59	-
13	RNN	69.49	-	60.78	-	145.05	-	158.57	-
14.	ALL3+LR-VA	63.40	18.52	55.76	23.28	100.11	35.48	101.75	28.44
15.	ALL3+LW-VA+LR-VA	63.24	18.54	55.24	23.24	99.27	35.42	98.56	28.32
16.	ALL3+RNN	64.23	18.42	56.06	23.20	115.04	35.22	130.27	28.02
17.	ALL3+RNN+LW-VA+LR-VA	**57.58**	**18.32**	**51.20**	**23.02**	**93.25**	**35.07**	**96.39**	**27.92**

performed MKN3 and HPY3. In terms of WER, statistically significant performance improvements ($p < 0.01$) were achieved by HPY3+LW-NA+LR-NA compared to MKN3 and HPY3 in each test set.

Next, n-best rescoring results are investigated; they are shown in lines 10–17 in Table 6. We only used ALL3 as a first-pass decoding pass when combining an n-gram LM with rescoring LMs, i.e., LW-VA, LR-VA and RNN. Note that only PPL was evaluated for RNN, LW-VA, LR-VA, LW-VA+LR-VA since they cannot be applied to ASR directly. ALL3+RNN outperformed ALL3 in both the in-domain tasks and the out-of-domain tasks. This indicates that n-best rescoring approach can yield ASR performance improvements. Only slight ASR performance improvements were attained by adding LR-VA to ALL3 although PPL was remarkably improved. It is thought that the PPL improvements attained by the Viterbi approximation are not related to the ASR performance improvement. The highest performance was attained by ALL3+RNN+LW-VA+LR-VA in each test set although the WER differences between ALL3+RNN were not statistically significant ($p > 0.05$).

6. Conclusions

This paper presented the latent word recurrent neural network language model (LW-RNN-LM); it employs a latent word space as well as LW-LMs in which the latent word space is modeled using RNN-LMs. LW-RNN-LM can capture long range relationships in the latent word space unlike standard LW-LMs which can take only very little context information into consideration. For LW-RNN-LM, we introduced a training method and two implementation methods, n-gram approximation and Viterbi approximation, for ASR. Experiments showed that LW-RNN-LM, RNN-LM and LW-LM complement each other and their combinations attain performance improvements in the both n-gram approximation and Viterbi approximation forms. In future work, we will introduce long short-term memory structures [31], [32] instead of using RNN structure for improving LW-RNN-LM performance. Furthermore, we will examine domain adaptation for LW-RNN-LM by using latent word space mixture modeling [33], [34].

References

[1] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?," Proc. IEEE, vol.88, no.8, pp.1270–1278, 2000.
[2] J.T. Goodman, "A bit of progress in language modeling," Computer Speech & Language, vol.15, pp.403–434, 2001.
[3] R. Masumura, T. Asami, T. Oba, H. Masataki, S. Sakauchi, and A. Ito, "Investigation of combining various major language model technologies including data expansion and adaptation," IEICE Trans. Inf. & Syst., vol.E99-D, no.10, pp.2452–2461, 2016.
[4] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," Computer Speech & Language, vol.10, no.3, pp.187–228, 1996.
[5] G. Potamianos and F. Jelinek, "A study of n-gram and decision tree letter language modeling methods," Speech Communication, vol.24, no.3, pp.171–192, 1998.
[6] P. Xu and F. Jelinek, "Random forests in language modeling," Proc. EMNLP 2004, pp.325–332, 2004.
[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," Journal of Machine Learning Research, vol.3, pp.1137–1155, 2003.
[8] H. Schwenk, "Continuous space language models," Computer Speech & Language, vol.21, no.3, pp.492–518, 2007.
[9] E. Arisoy, T.N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," Proc. NAACL-HLT 2012, pp.20–28, 2012.
[10] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," Proc. Interspeech, pp.1045–1048, 2010.
[11] T. Mikolov, S.K. Stefan, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," Proc. ICASSP, pp.5528–5531, 2011.
[12] Y. Su, "Bayesian class-based language models," Proc. ICASSP 2011, pp.5564–5567, 2011.
[13] J.-T. Chien and C.-H. Chueh, "Dirichlet class language models for speech recognition," IEEE Transactions on Audio, Speech and Lan-

guage Processing, vol.19, no.3, pp.1352–1365, 2011.
[14] K. Deschacht, J.D. Belder, and M.-F. Moens, "The latent words language model," Computer Speech & Language, vol.26, no.5, pp.384–409, 2012.
[15] R. Masumura, H. Masataki, T. Oba, O. Yoshioka, and S. Takahashi, "Use of latent words language models in ASR: a sampling-based implementation," Proc. ICASSP, pp.8445–8449, 2013.
[16] R. Masumura, T. Oba, H. Masataki, O. Yoshioka, and S. Takahashi, "Viterbi decoding for latent words language models using Gibbs sampling," Proc. INTERSPEECH, pp.3429–3433, 2013.
[17] R. Masumura, T. Adami, T. Oba, H. Masataki, S. Sakauchi, and S. Takahashi, "N-gram approximation of latent words language models for domain robust automatic speech recognition," IEICE Trans. Inf. & Syst., vol.E99-D, no.10, pp.2462–2470, 2016.
[18] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," Proc. SLT, pp.234–239, 2012.
[19] X. Liu, X. Chen, M.J.F. Gales, and P.C. Woodland, "Paraphrastic recurrent neural network language models," Proc. ICASSP, pp.5406–5410, 2015.
[20] R. Masumura, T. Asami, T. Oba, H. Masataki, S. Sakauchi, and A. Ito, "Latent words recurrent neural network language models," Proc. INTERSPEECH, pp.2380–2384, 2015.
[21] J.T. Goodman, "Classes for fast maximum entropy training," Proc. ICASSP, pp.561–564, 2001.
[22] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," Proc. AISTATS, pp.246–252, 2005.
[23] Y.W. Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," Proc. ACL, pp.985–992, 2006.
[24] D.J.C. MacKay and L.C. Peto, "A hierarchical dirichlet language model," Natural language engineering, vol.1, pp.289–308, 1995.
[25] M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of English: The penn treebank," Computational Linguistics, vol.19, pp.313–330, 1993.
[26] S.F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," Computer Speech & Language, vol.13, no.4, pp.359–383, 1999.
[27] Y.W. Teh, "A hierarchical Bayesian language model based on Pitman-Yor processes," Proc. COLING-ACL, pp.985–992, 2006.
[28] A. Stolcke, "Entropy-based pruning of backoff language models," Proc. DARPA Broadcast News Transcription and Understanding Workshop, pp.270–274, 1998.
[29] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of Japanese," Proc. LREC, pp.947–952, 2000.
[30] G. Hinton, L. Deng, D. Yu, G. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," Signal Processing Magazine, pp.1–27, 2012.
[31] M. Sundermeyer, R. Schluter, and H. Ney, "LSTM neural networks for language modeling," Proc. INTERSPEECH, pp.194–197, 2012.
[32] M. Sundermeyer, H. Ney, and R. Schluter, "From feedforward to recurrent LSTM neural networks for language models," IEEE/ACM Transactions of Audio, Speech and Language processing, vol.23, no.3, pp.517–529, 2015.
[33] R. Masumura, T. Asami, T. Oba, H. Masataki, and S. Sakauchi, "Mixture of latent words language models for domain adaptation," Porc. INTERPSEECH, pp.1425–1429, 2014.
[34] R. Masumura, T. Asami, T. Oba, H. Masataki, S. Sakauchi, and A. Ito, "Domain adaptation based on mixture of latent words language models for automatic speech recognition," IEICE Trans. Inf. & Syst., vol.E101-D, no.6, pp.1581–1590, 2018.

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 the Acoustic Society of Japan (ASJ) in 2011 and 2013, respectively, the Sendai Section Student Awards The Best Paper Prize from the Institute of Electrical and Electronics Engineers (IEEE) in 2011, the Yamashita SIG Research Award from the Information Processing Society of Japan (IPSJ) in 2014, the Young Researcher Award from the Association for Natural Language Processing (NLP) in 2015, and the ISS Young Researcher's Award in Speech Field from the Institute of Electronic, Information and Communication Engineers (IEICE) in 2015. He is a member of the ASJ, the IPSJ, the NLP, the IEEE, and the International Speech Communication Association (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. He received the Awaya Kiyoshi Science Promotion Award and the Sato Prize Paper Award from the Acoustic Society of Japan (ASJ) in 2012 and 2014, respectively. He is a member of the ASJ, the Institute of Electronics, Information and Communication Engineers (IEICE), Institute of Electrical and Electronics Engineers (IEEE), and the International Speech Communication Association (ISCA).

Takanobu Oba received B.E. and M.E. degrees from Tohoku University, Sendai, Japan, in 2002 and 2004, respectively. 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. Since 2015, he has been engaged in development of spoken dialogue services at the NTT Docomo Corporation, Yokosuka, Japan. He received the Awaya Kiyoshi Science Promotion Award from the Acoustical Society of Japan (ASJ) in 2007. He received Ph.D. (Eng.) degree from Tohoku University in 2011. He is a member of the Institute of Electrical and Electronics Engineers (IEEE), the Institute of Electronics, Information, and Communication Engineers (IEICE) and the ASJ.

Sumitaka Sakauchi received M.S. degree from Tohoku University in 1995 and 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 Senior Manager in the Research and Development Planning Department of NTT. He received the Paper Award from the Institute of Electronics, Information and Communication Engineers (IEICE) in 2001, and Awaya Kiyoshi Science Promotion Award from the Acoustic Society of Japan (ASJ) in 2003. He is a member of the IEICE and the ASJ.

Akinori Ito received B.E., M.E., and Ph.D. degrees from Tohoku University, Sendai, Japan. Since 1992, he has worked with Research Center for Information Sciences and Education Center for Information Processing, Tohoku University. He was with the Faculty of Engineering, Yamagata University, from 1995 to 2002. From 1998 to 1999, he worked with the College of Engineering, Boston University, MA, USA, as a Visiting Scholar. He is now a Professor of the Graduate School of Engineering, Tohoku University. He is engaged in spoken language processing, statistical text processing, and audio signal processing. He is a member of the Acoustic Society of Japan, the Information Processing Society of Japan, and the IEEE.