Latent Words Recurrent Neural Network Language Models

Ryo Masumura[†‡], *Taichi Asami*[†], *Takanobu Oba*[†], *Hirokazu Masataki*[†],
Sumitaka Sakauchi[†], *Akinori Ito*[‡]

† NTT Media Intelligence Laboratories, NTT Corporation, Japan
‡ Graduate School of Engineering, Tohoku University, Japan

{masumura.ryo, asami.taichi, oba.takanobu, masataki.hirokazu, sakauchi.sumitaka}@lab.ntt.co.jp
aito@spcom.ecei.tohoku.ac.jp

10.21437/Interspeech.2015-515

Abstract

This paper proposes a novel language modeling approach called latent word recurrent neural network language model, which solves the problems present in both recurrent neural network language models (RNNLMs) and latent word language models (LWLMs). The proposed model has a soft class structure based on a latent variable space as well as LWLM, where the latent variable space is modeled using RNNLM. From the viewpoint of RNNLMs, the proposed model can be considered as a soft class RNNLM with a vast latent variable space. In contrast, from the viewpoint of LWLMs, the proposed model can be considered as an LWLM that uses the RNN structure for latent variable modeling instead of the n-gram structure. This paper also details the parameter inference method and two kinds of usages for natural language processing tasks. Our experiments show effectiveness of the proposed model on a perplexity evaluation for the Penn Treebank corpus and an automatic speech recognition evaluation for Japanese spontaneous speech tasks.

Index Terms: latent words recurrent neural network language models, n-gram approximation, Viterbi approximation

1. Introduction

Language models (LMs) are essential for natural language processing tasks including automatic speech recognition and statistical machine translation. Typical problems faced by LMs are data sparseness and locality, and several techniques have been proposed to mitigate these problems [1, 2]. In particular, approaches intended to improve LM structure have been aggressively pursued. The structure can be split into two main types; discriminative models and generative models.

Discriminative models include several intelligent models such as maximum entropy model [3], decision tree [4], random forest [5], and neural network [6, 7] including deep neural network [8]. Recurrent neural network LMs (RNNLMs) have, in particular, achieved significant improvements in recent years [9, 10]. RNNLMs can capture long-range context information via their recurrent structure and can efficiently represent context information as a continuous vector. Therefore, RNNLMs are known to be one of the most powerful LMs.

On the other hand, among the generative models, Bayesian modeling is attracting attention. Hierarchal Pitman-Yor LMs (HPYLMs), which are theoretically elegant Bayesian LMs based on the Pitman-Yor process, have demonstrated better performance than traditional smoothed n-gram models [11, 12]. Bayesian class-based LMs were also proposed as a further improvement [13, 14]. In addition, latent words LMs (LWLMs) were proposed as a more flexible form of Bayesian class-based LMs [15]. LWLMs have a vast latent variable space and soft class structure. These flexible attributes help us to mitigate data sparseness efficiently. In fact, LWLMs are robust for not only in-domain tasks but also out-of domain tasks [16, 17].

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

To tackle the problems of both RNNLMs and LWLMs, this paper proposes a novel modeling approach called the latent word recurrent neural network language model (LWRNNLM). Our idea is to combine RNNLM and LWLM to offset the faults of each. Thus, the proposed models can support both a latent variable space modeling as well as LWLMs and a long-range relationship modeling as well as RNNLMs at the same time. From the viewpoint of RNNLMs, the proposed model can be considered as a soft class RNNLM with a latent variable space. In contrast, from the viewpoint of LWLMs, the proposed model can be considered as an LWLM whose latent variable modeling is based on RNN structure instead of n-gram structure. Consequently, it can be expected that the proposed model has different attributes from conventional RNNLMs and LWLMs and yields further improvement through their combination.

There are two issues in LWRNNLMs. The first is which parameter inference method should be used. It is impossible to estimate LWRNNLMs directly from training data since the RNN structure is not suitable for latent variable space modeling. To this end, we present an alternative inference procedure using a trained LWLM. In the procedure, a latent variable assignment of training data are firstly decoded using the trained LWLM, and then a parameter of the proposed model is estimated using the sampled latent variable assignment. The second issue is usage. We note the impossibility of applying trained LWRNNLMs into natural language processing applications directly because their latent variable space is vast. To this end, this paper presents n-gram approximation and Viterbi approximation as well as those for LWLMs [16, 17].

This paper is organized as follows. Section 2 and Section 3 detail RNNLMs and LWLMs, respectively. Details of RNNLMs are given in Section 4. Section 5 describes our experiments on two corpora. Section 6 concludes this paper.

2. Recurrent Neural Network Language Models

RNNLMs are discriminative models which have attracted significant attention in recent years [9]. RNNLMs have two points: one is that 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 the recurrent structure.

In RNNLMs, the previous word w_{k-1} is fed to the input of the network using 1-of-K coding together with the context information $\boldsymbol{s}_{k-1}$ which includes the previous output of the hidden layer. In word-based RNNLMs, the generative probability of word sequence $\boldsymbol{w} = w_1, \cdots, w_K$ is defined as:

$$\begin{aligned} P(\boldsymbol{w}) &= \prod_{k=1}^{K} P(w_k | w_{k-1}, \boldsymbol{s}_{k-1}, \boldsymbol{\Theta}_{\mathtt{rnn}}), \\ &= \prod_{k=1}^{K} P(w_k | \boldsymbol{s}_k, \boldsymbol{\Theta}_{\mathtt{rnn}}), \end{aligned} \tag{1}$$

where $\boldsymbol{\Theta}_{\mathtt{rnn}}$ is a model parameter of RNNLM. By propagating the input layer, context information $\boldsymbol{s}_k$ is updated. Note that class-based RNNLMs are also used since the cost of computing the probability estimation is proportional to the lexical size of the output layer in word-based modeling [10].

Furthermore, the mixing of several RNNs trained with different random initialization values is reported in [10]. The probability estimation of mixture of word-based RNNLMs is defined as:

$$P(\boldsymbol{w}) = \frac{1}{T} \prod_{k=1}^{K} \sum_{t=1}^{T} P(w_k | \boldsymbol{s}_k^t, \boldsymbol{\Theta}_{\mathtt{rnn}}^t). \tag{2}$$

The generative probability can be calculated using T instances of RNNLMs. $\boldsymbol{\Theta}_{\mathtt{rnn}}^t$ indicates the t-th model parameter and $\boldsymbol{s}_k^t$ is the context information of the t-th instance.

The recurrent neural network architecture can be trained by the back-propagation through time algorithm. The cross entropy criterion is used to obtain an error vector in the output layer. In the training procedure, validation data is necessary to permit its early stopping.

RNNLMs can be used directly for calculating the probability of any word sequence. In automatic speech recognition, RNNLMs are usually used as a rescoring model. Moreover, another usage is based on n-gram approximation, where a lot of text data is generated using a trained RNNLM and simple n-gram model is constructed from the generated data [18].

3. Latent Words Language Models

LWLMs are generative models with a latent variable for every observed word [15]. LWLM has a soft clustering structure that differs from a simple hard clustering structure. In the soft clustering structure, one word belongs to multiple classes. In fact, each word belongs to all classes in LWLM. The latent variable is expressed as a specific word that can be selected from the entire vocabulary, so the latent variable is called the latent word. Thus, the number of latent words equals the number of observed words.

In LWLMs, a latent word h_k is generated from a transition probability distribution given its context $\boldsymbol{l}_k = h_{k-n+1}, ..., h_{k-1}$. An observed word w_k is generated from an emission probability distribution given its latent word h_k. The transition probability distribution $P(h_k | \boldsymbol{l}_k, \boldsymbol{\Theta}_{\mathtt{lw}})$ is expressed as an n-gram model for latent words, and the emission probability distribution $P(w_k | h_k, \boldsymbol{\Theta}_{\mathtt{lw}})$ models the dependency between the observed word and the latent word. $\boldsymbol{\Theta}_{\mathtt{lw}}$ indicates a model parameter of LWLM. Usually, a hierarchical Pitman-Yor prior is used for the transition probability distribution, and a Dirichlet prior is used for the emission probability distribution [16, 17].

Bayesian inference is suitable for LWLM as it takes account of all possible model parameters and produces the generative probability of observed words $\boldsymbol{w} = w_1, \cdots, w_K$. Since the modeling is analytically intractable, a feasible approximation is given by:

$$\begin{aligned} P(\boldsymbol{w}) &= \int \sum_{\boldsymbol{h}} P(\boldsymbol{w} | \boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{h} | \boldsymbol{\Theta}_{\mathtt{lw}}) P(\boldsymbol{\Theta}_{\mathtt{lw}}) \mathrm{d}\boldsymbol{\Theta}_{\mathtt{lw}}, \\ &\simeq \frac{1}{T} \sum_{\boldsymbol{h}} \sum_{t=1}^{T} P(\boldsymbol{w} | \boldsymbol{h}, \boldsymbol{\Theta}_{\mathtt{lw}}^t) P(\boldsymbol{h} | \boldsymbol{\Theta}_{\mathtt{lw}}^t), \\ &= \frac{1}{T} \prod_{k=1}^{K} \sum_{t=1}^{T} \sum_{h_k} P(w_k | h_k, \boldsymbol{\Theta}_{\mathtt{lw}}^t) P(h_k | \boldsymbol{l}_k, \boldsymbol{\Theta}_{\mathtt{lw}}^t). \end{aligned} \tag{3}$$

The probability distribution can be approximated using T instances of point estimated parameter; $\boldsymbol{\Theta}_{\mathtt{lw}}^t$ indicates the t-th point estimated parameter.

For the inference, we have to estimate latent word assignment $\boldsymbol{h}$ that underlies training data $\boldsymbol{w}$. To this end, Gibbs sampling is suitable [19, 20, 21]. The conditional probability distribution of possible values for latent word h_k is given by:

$$P(h_k | \boldsymbol{w}, \boldsymbol{h}_{-k}, \boldsymbol{\Theta}_{\mathtt{lw}}) \propto P(w_k | h_k, \boldsymbol{\Theta}_{\mathtt{lw}}) \prod_{j=k}^{k+n-1} P(h_j | \boldsymbol{l}_j, \boldsymbol{\Theta}_{\mathtt{lw}}), \tag{4}$$

where $\boldsymbol{h}_{-k}$ represents all latent words except h_k, Gibbs sampling samples a new value for the latent word according to its distribution and places it at position k in $\boldsymbol{h}$. Along with $\boldsymbol{h}$, the model parameters are updated.

When introducing LWLMs to natural language processing tasks, it is impractical to rigorously compute the generative probability since the latent variable space in LWLM is vast. Therefore, two approximate implementation methods have been proposed. One is n-gram approximation, which randomly generates a lot of text data according to a stochastic process and a simple n-gram model is constructed from the generated data [16]. The other is Viterbi approximation, it uses the joint probability between the optimal latent variable sequence and the observed word sequence [17]. The joint probability can be used as can other generative probabilities given by n-gram models or RNNLMs.

4. Latent Words Recurrent Neural Network Language Models

4.1. definition

LWRNNLMs are novel models that combine RNNLMs with LWLMs. The models have a soft class structure based on a latent variable space as well as LWLM and the latent variable space is modeled using RNNLM. A transition probability distribution $P(h_k | \boldsymbol{s}_k, \boldsymbol{\Theta}_{\mathtt{lrn}})$ is expressed as a RNNLM for latent words and $\boldsymbol{s}_k$ is context information of the RNN structure. An emission probability distribution $P(w_k | h_k, \boldsymbol{\Theta}_{\mathtt{lrn}})$ models the dependency between the observed word and the latent word as well as LWLMs. $\boldsymbol{\Theta}_{\mathtt{lrn}}$ indicates a model parameter of

LWRNNLMs. A feasible approximation of the generative probability of word sequence $\boldsymbol{w} = w_1, \cdots, w_K$ is obtained by:

$$\begin{aligned} P(\boldsymbol{w}) &= \int \sum_{\boldsymbol{h}} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\texttt{lrn}}) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\texttt{lrn}}) P(\boldsymbol{\Theta}_{\texttt{lrn}}) \mathrm{d}\boldsymbol{\Theta}_{\texttt{lrn}}, \\ &\simeq \frac{1}{T} \sum_{\boldsymbol{h}} \sum_{t=1}^{T} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\texttt{lrn}}^t) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\texttt{lrn}}^t), \\ &= \frac{1}{T} \prod_{k=1}^{K} \sum_{t=1}^{T} \sum_{h_k} P(w_k|h_k, \boldsymbol{\Theta}_{\texttt{lrn}}^t) P(h_k|\boldsymbol{s}_k^t, \boldsymbol{\Theta}_{\texttt{lrn}}^t), \end{aligned} \tag{5}$$

where $\boldsymbol{\Theta}_{\texttt{lrn}}^t$ is the t-th model parameter and $\boldsymbol{s}_k^t$ is the context information of the t-th RNN structure. The proposed models are extremely related to conventional RNNLMs and LWLMs. Eq. (5) can be regarded as Eq. (2) with soft class structure based on latent variable space. In addition, Eq. (5) can be regarded as Eq. (3) where the transition probability distribution is based on RNN structure instead of n-gram structure.

4.2. Parameter Inference

It is impossible to estimate an LWRNNLM directly. Therefore, we preliminarily train an LWLM and use it to decode the latent word assignment of the observed word sequence. The latent word assignment is used for estimating a model parameter of the proposed model.

The optimal latent word assignment of observed word sequence can be estimated for each instance in the LWLM. For the estimation, we sample the several candidates of optimal latent word assignment based on Gibbs sampling, Eq. (4), and find the best of the samples by re-evaluating. Several candidates of latent word assignment that are sampled using the t-th model parameter $\boldsymbol{\Theta}_{\texttt{lw}}^t$ are denoted as $\boldsymbol{H}^t = \{\boldsymbol{h}_1^t, \cdots, \boldsymbol{h}_M^t\}$ where M is a number of candidates. The t-th optimal latent variable assignment $\hat{\boldsymbol{h}}^t$ can be selected as:

$$\hat{\boldsymbol{h}}^t = \underset{\boldsymbol{h} \in \boldsymbol{H}^t}{\arg\max}\, P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\texttt{lw}}^t) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\texttt{lw}}^t). \tag{6}$$

This procedure is applied to not only training data but also validation data. The t-th model parameter $\boldsymbol{\Theta}_{\texttt{lrn}}^t$ can be determined from the t-th latent word assignments of them. The transition probability distribution $P(h_k|\boldsymbol{s}_k^t, \boldsymbol{\Theta}_{\texttt{lrn}}^t)$ can be estimated from the t-th latent variable assignment of training data as in usual RNNLM training. The t-th latent variable assignment of validation data is used for early stopping. The emission probability distribution $P(w_k|h_k, \boldsymbol{\Theta}_{\texttt{lrn}}^t)$ can be diverted from the trained LWLM directly.

4.3. Usages

Although it is impossible to directly apply the LWRNNLM to natural language processing tasks as well as LWLMs, it can be resolved in the same or similar methods as that of LWLMs, n-gram approximation and Viterbi approximation [16, 17].

N-gram approximation of LWRNNLMs follows the usages of RNNLMs and LWLMs. A lot of text data is generated by random sampling using trained LWRNNLMs, and n-gram models are constructed from the generated data.

Viterbi approximation of LWRNNLMs is not same as that of LWLMs. In fact, we have to consider all possible latent word assignments when determining the Viterbi path since Gibbs sampling [19] or Viterbi algorithm [22] cannot be introduced due to their use of the RNN structure. To avoid considering all possible latent word paths, our Viterbi approximation also uses the trained LWLM. We decode several Viterbi path candidates using a trained LWLM, and sampled candidates are re-evaluated using a trained LWRNNLM. Several candidates of Viterbi path $\boldsymbol{H} = \{\boldsymbol{h}_1, \cdots, \boldsymbol{h}_M\}$ can be extracted from a trained LWLM by Gibbs sampling, Eq. (4). The Viterbi path probability, i.e. joint probability $P(\boldsymbol{w}, \hat{\boldsymbol{h}})$, is defined as:

$$P(\boldsymbol{w}, \hat{\boldsymbol{h}}) = \max_{\boldsymbol{h} \in \boldsymbol{H}} \frac{1}{T} \sum_{t=1}^{T} P(\boldsymbol{w}|\boldsymbol{h}, \boldsymbol{\Theta}_{\texttt{lrn}}^t) P(\boldsymbol{h}|\boldsymbol{\Theta}_{\texttt{lrn}}^t), \tag{7}$$

where $\hat{\boldsymbol{h}}$ indicates the Viterbi path for observed word sequence $\boldsymbol{w}$. The joint probability can be used as can the generative probability given by n-gram models or RNNLMs.

Table 1: *Perplexity evaluation results on Penn Treebank corpus.*

Setup	Valid		Test	
	Self	+ALL5g	Self	+ALL5g
MKN5g	148.0	-	141.2	-
HPY5g	145.1	-	139.3	-
RNN5g	159.1	-	148.3	-
LW5g	141.6	-	134.3	-
LRN5g	153.0	-	144.2	-
RNN5g+LRN5g	147.3	-	138.5	-
LW5g+LRN5g	139.7	-	132.5	-
ALL5g	**126.3**	-	**120.5**	-
RNN	118.3	105.2	112.5	101.2
LW	468.4	118.6	444.7	114.0
LRN	531.6	117.1	506.5	113.1
RNN+LRN	112.6	102.3	107.6	98.6
LW+LRN	419.6	116.6	403.8	112.7
RNN+LW+LRN	**109.3**	**101.0**	**104.6**	**97.7**

5. Experiments

5.1. Perplexity Evaluation on Penn Treebank Corpus

The first experiments used the Penn Treebank corpus, one of the most widely used sources for evaluating LMs [23]. Sections 0-20 were used as a training set whose size was 930K, sections 21 and 22 were used as a validation set (Valid) whose size was 74K, and sections 23 and 24 were used as a test set (Test) whose size was 82K. Each vocabulary was limited to 10K words and there were no out-of-vocabulary words. This setup matches those of many previous works.

As a baseline, 5-gram LM with interpolated Kneser-Ney smoothing (MKN5g) [24] and 5-gram HPYLM (HPY5g) were trained from training data. We also trained three models, a mixture of word-based RNNLMs with 200 hidden nodes and 10 instances (RNN), 5-gram LWLM with 10 instances (LW), and LWRNNLM whose latent words were modeled by word-based RNNLMs with 200 hidden nodes (LRN). Moreover, we individually generated 200M words from each model and trained 5-gram HPYLM for n-gram approximation. The approximated n-gram models are denoted as RNN5g, LW5g, and LRN5g. ALL5g indicates a mixed model of all of 5-gram LMs (MKN5g, HPY5g, RNN5g, LW5g, LRN5g). Several hyper parameters including mixture weights were optimized using validation set.

The results based on perplexity (PPL) are shown in Table 1. First, we examine the n-gram approximation results. LRN5g

Table 2: *Speech recognition evaluation results on CSJ.*

Setup	Valid (In-Domain)		Test 1 (In-Domain)		Test 2 (Out-Of-Domain)		Test 3 (Out-Of-Domain)	
	PPL	WER(%)	PPL	WER(%)	PPL	WER(%)	PPL	WER(%)
MKN3g	80.84	19.98	68.08	24.79	157.52	38.67	188.56	32.31
HPY3g	78.33	19.74	66.24	24.67	148.43	38.29	174.30	32.00
RNN3g	104.12	21.63	83.48	26.24	148.21	39.32	161.84	31.96
LW3g	80.21	19.61	67.22	24.54	134.57	36.93	151.44	30.42
LRN3g	85.71	19.86	71.55	24.66	136.68	37.40	148.03	29.95
RNN3g+LRN3g	84.01	19.75	70.33	24.74	132.65	37.20	142.06	29.78
LW3+LRN3g	78.56	19.47	65.18	24.27	132.05	36.72	144.34	29.68
ALL3g	**73.38**	**18.80**	**62.02**	**23.42**	**126.05**	**36.34**	**139.32**	**29.32**
ALL3g+RNN	62.86	18.50	54.56	23.20	111.04	36.02	131.12	29.05
ALL3g+LW+LRN	66.40	18.72	57.86	23.24	105.14	35.78	121.87	28.86
ALL3g+RNN+LW+LRN	**59.05**	**18.38**	**52.27**	**23.02**	**99.52**	**35.65**	**114.71**	**28.60**

was superior to RNN5g and slightly weaker than LW5g. On the other hand, RNN5g+LRN5g and LW5g+LRN5g were superior to RNN5g and LW5g. These results show that LRN5g possessed properties different from RNN5g and LW5g and their combination was effective for language modeling. Under the restriction of the n-gram structure, the best PPL on the test data, 120.5, was obtained by ALL5g.

Next, we added RNN, LW, and LRN to ALL5g. LW and LRN were used as Viterbi approximation. RNN performed strongly compared to ALL5g. LW and LRN were relatively weaker than with the other condition. It is because the joint probability was used for computing PPL. RNN+LRN and LW+LRN were superior to RNN and LW while a single use of LRN could not improve the performance compared to RNN and LW. These tendencies are the same as those seen with n-gram approximation. ALL5g+RNN+LW+LRN provided the highest PPL on the test data, 97.7. It seems that RNN, LW and LRN complement each other and each model yields characteristics different from their n-gram approximations.

5.2. Automatic Speech Recognition Evaluation on CSJ

The second experiments used the Corpus of Spontaneous Japanese (CSJ) for an automatic speech recognition evaluation [25]. We divided CSJ into a training set, validation set (Valid), and test set (Test 1). In addition, a contact center task (Test 2) and a voice mail task (Test 3) were also used for evaluation of out-of-domain tasks. The training data size was about 7M words, the vocabulary size was about 80K. The validation data size and each test data size were about 20K words.

We used an acoustic model based on hidden Markov models with deep neural networks (DNN-HMM) [26, 27]. The trained DNN-HMM had 8 hidden layers with 2048 nodes and 3072 outputs. The speech recognition decoder was VoiceRex, a WFST-based decoder [28, 29]. JTAG was used as the morpheme analyzer to split sentences into words [30].

As a baseline, 3-gram LM with interpolated Kneser-Ney smoothing (MKN3g) and 3-gram HPYLM (HPY3g) were constructed from the training data. We also trained three models, a class-based RNNLM with 500 hidden nodes and 500 classes (RNN), a 3-gram LWLM constructed from 10 instances (LW), and an LWRNNLM whose latent words were modeled by a class-based RNNLM with 500 hidden nodes and 500 classes (LRN). We also individually generated 200M words from each model and trained 3-gram HPYLM for n-gram approximation. The approximated n-gram models are denoted as RNN3g, LW3g, and LRN3g. ALL3g indicates a mixture of all of 3-gram models (MKN3g, HPY3g, RNN3g, LW3g, LRN3g). Several hyper parameters including mixture weights were optimized using a validation set. We compared each model and their combinations. The n-gram mixture model, i.e. ALL3g can be approximated as a single back-off n-gram model [31], so it can be used in a first decoding pass. On the other hand, RNN, LW and LRN were employed in 1000-best rescoring whose candidates were decoded by ALL3g.

Table 2 shows the perplexity (PPL) and word error rate (WER) results for each condition. First, we compared first decoding pass results gained from the n-gram models. RNN3g was weaker than the other models. LW3g consistently matched HPY3g in the in-domain tasks and far superior to HYP3g in the out-of-domain tasks. These tendencies match previous works [16]. The proposed LRN3g was comparable to LW3g. In addition, RNN3g+LRN3g and LW3g+LRN3g attained improvement compared with RNN3g or LW3g. These results show an effectiveness of the proposed model in automatic speech recognition tasks. The highest results were obtained by ALL3g in each condition. Especially, in out-of domain tasks, ALL3g strongly performed compared to MKN3g and HPY3g.

Next, we compared n-best rescoring results; the bottom 3 rows in Table 2. ALL3g+RNN achieved remarkably high performance in the in-domain tasks. In contrast, ALL3g+LW+LRN was consistently superior to ALL3g+RNN in the out-of-domain tasks. It seems that the Viterbi approximation which uses joint probability of optimal latent word assignment and observed word sequence offers robust performance in several tasks. The highest performance was attained by ALL3g+RNN+LW+LRN regardless of the condition.

6. Conclusions

In this paper, we proposed LWRNNLMs by combining RNNLMs and LWLMs to offset the faults of each. The proposed model has a latent variable space as well as LWLMs and the latent variable space is modeled using RNNLMs, so the proposed model is more flexible than the conventional RNNLMs or LWLMs. We also presented an inference algorithm and two methods, n-gram approximation and Viterbi approximation, for implementing the proposed model. Our experiments showed that LWRNNLM, RNNLM and LWLM complement each other and their combinations achieve performance improvement in both n-gram approximation and Viterbi approximation even though a single use of the proposed model was not powerful. Moreover, their combinations performed robustly in multiple domains.

7. References

[1] R. Rosenfeld, "Two decades of statistical language modeling: Where do we go from here?" *Proc. IEEE*, vol. 88, 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. Rosenfeld, "A maximum entropy approach to adaptive statistical language modeling," *Computer Speech & Language*, vol. 10, pp. 187–228, 1996.

[4] 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.

[5] P. Xu and F. Jelinek, "Random forests in language modeling," *In Proc. EMNLP 2004*, pp. 325–332, 2004.

[6] 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.

[7] H. Schwenk, "Continuous space language models," *Computer Speech & Language*, vol. 21, pp. 492–518, 2007.

[8] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," *In Proc. NAACL-HLT 2012*, pp. 20–28, 2012.

[9] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," *In Proc. Interspeech 2010*, pp. 1045–1048, 2010.

[10] T. Mikolov, S. K. Stefan, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," *In Proc. ICASSP 2011*, pp. 5528–5531, 2011.

[11] Y. W. Teh, "A hierarchical bayesian language model based on pitman-yor processes," *In Proc. COLING/ACL 2006*, pp. 985–992, 2006.

[12] S. Huang and M. Yor, "Hierarchical pitman-yor language models for asr in meetings," *In Proc ASRU 2007*, pp. 124–129, 2007.

[13] Y. Su, "Bayesian class-based language models," *In Proc. ICASSP 2011*, pp. 5564–5567, 2011.

[14] J.-T. Chien and C.-H. Chueh, "Dirichlet class language models for speech recognition," *IEEE transactions on Audio, Speech and Language Processing*, vol. 19, pp. 1352–1365, 2011.

[15] K. Deschacht, J. D. Belder, and M.-F. Moens, "The latent words language model," *Computer Speech & Language*, vol. 26, pp. 384–409, 2012.

[16] R. Masumura, H. Masataki, T. Oba, O. Yoshioka, and S. Takahashi, "Use of latent words language models in asr: a sampling-based implementation," *In Proc. ICASSP 2013*, pp. 8445–8449, 2013.

[17] R. Masumura, T. Oba, H. Masataki, O. Yoshioka, and S. Takahashi, "Viterbi decoding for latent words language models using gibbs sampling," *In Proc. Interspeech 2013*, pp. 3429–3433, 2013.

[18] A. Deoras, T. Mikolov, S. Kombrink, M. Karafiat, and S. Khudanpur, "Variational approximation of long-span language models in lvcsr," *In Proc. ICASSP 2011*, pp. 5532–5535, 2011.

[19] G. Casella and E. I. George, "Explaining the gibbs sampler," *The American Statistician*, vol. 46, pp. 167–174, 1992.

[20] C. P. Robert, G. Celeux, and J. Diebolt, "Bayesian estimation of hidden markov chains: A stochastic implementation," *Statistics & Probability Letters*, vol. 16, pp. 77–83, 1993.

[21] S. L. Scott, "Bayesian methods for hidden markov models: Recursive computing in the 21st century," *Journal of the American Statistical Association*, vol. 97, pp. 337–351, 2002.

[22] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, pp. 268–278, 1973.

[23] 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.

[24] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," *In Proc. ICASSP*, vol. 1, pp. 181–184, 1995.

[25] K. Maekawa, H. Koiso, S. Furui, and H. Isahara, "Spontaneous speech corpus of japanese," *In proc. LREC*, pp. 947–952, 2000.

[26] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman 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.

[27] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," *In Proc. Interspeech 2011*, pp. 437–440, 2011.

[28] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *IEEE transactions on Audio, Speech and Language Processing*, vol. 15, no. 4, pp. 1352–1365, 2007.

[29] H. Masataki, D. Shibata, Y. Nakazawa, S. Kobashikawa, A. Ogawa, and K. Ohtsuki, "Voicerex spontaneous speech recognition technology for contact-center conversations," *NTT Tech. Rev.*, vol. 5, no. 1, pp. 22–27, 2007.

[30] T. Fuchi and S. Takagi, "Japanese morphological analyzer using word co-occurence-jtag," *In Proc. COLING-ACL*, pp. 409–413, 1998.

[31] A. Stolcke, J. Zheng, W. Wang, and V. Abrash, "Srilm at sixteen: Update and outlook," *In Proc. ASRU 2011, Demonstrations and Tool Kit*, 2011.