Bayesian Recurrent Neural Network for Language Modeling

Jen-Tzung Chien, *Senior Member, IEEE*, and Yuan-Chu Ku

***Abstract*— A language model (LM) is calculated as the probability of a word sequence that provides the solution to word prediction for a variety of information systems. A recurrent neural network (RNN) is powerful to learn the large-span dynamics of a word sequence in the continuous space. However, the training of the RNN-LM is an ill-posed problem because of too many parameters from a large dictionary size and a high-dimensional hidden layer. This paper presents a Bayesian approach to regularize the RNN-LM and apply it for continuous speech recognition. We aim to penalize the too complicated RNN-LM by compensating for the uncertainty of the estimated model parameters, which is represented by a Gaussian prior. The objective function in a Bayesian classification network is formed as the regularized cross-entropy error function. The regularized model is constructed not only by calculating the regularized parameters according to the maximum *a posteriori* criterion but also by estimating the Gaussian hyperparameter by maximizing the marginal likelihood. A rapid approximation to a Hessian matrix is developed to implement the Bayesian RNN-LM (BRNN-LM) by selecting a small set of salient outer-products. The proposed BRNN-LM achieves a sparser model than the RNN-LM. Experiments on different corpora show the robustness of system performance by applying the rapid BRNN-LM under different conditions.**

***Index Terms*— Bayesian learning, Hessian matrix, language model, rapid approximation, recurrent neural network.**

I. Introduction

NATURAL language processing aims to analyze, understand, and generate languages that humans use naturally. Significant progress has been achieved in recent years, addressing important and practical real-world problems, enabling mass deployment of large-scale systems. New machine learning paradigms, such as deep neural networks (DNNs) and continuous space methods, have contributed to inferring language patterns from increasingly large real-world data and to making predictions about new data more accurately.

One important challenge is to represent the language in a form that can be processed effectively by learning systems. Traditionally, words in a sequence are treated as discrete symbols and represented by the n-gram language model (LM). LM plays an important role in many information systems, including machine translation, optical character recognition, writing correction, question answering, information retrieval, bioinformatics, and continuous speech recognition (CSR). According to the Bayes decision rule, LM $p(W)$ provides the prior probability for an automatic speech recognition (ASR) system to search for the most likely sequence $\hat{W}$ corresponding to a test speech utterance X via

$$\hat{W} = \arg\max_{W} p(W|X) = \arg\max_{W} p(X|W)p(W) \tag{1}$$

where $p(X|W)$ denotes the acoustic likelihood function. The statistical n-gram LM is calculated as the probability of a word sequence W that is written as a product of probabilities of individual words $w = w_t$ at each time t conditional on their preceding $n-1$ words $\mathbf{u} = \{w_{t-n+1}, \ldots, w_{t-1}\} \triangleq w_{t-n+1}^{t-1}$

$$p(W) = p(w_1, w_2, \ldots, w_T) \cong \prod_{t=1}^{T} p\left(w_t | w_{t-n+1}^{t-1}\right). \tag{2}$$

The N-gram model is efficient and effective in extracting the lexical regularities from sequential patterns. However, in the case of a high-order model and large vocabulary size, such an n-gram model suffers from the inadequate long-distance information due to the limitation of context window and the insufficient training data for computation of multinomial parameters for so many word combinations. To extract long-distance information outside the n-gram window, the association patterns [1] and the cache classes [2], [3] were identified and incorporated into the calculation of the n-gram LM. To cope with the issue of insufficient training data, the class-based n-grams [4] were calculated by considering the word generation given the concatenated classes rather than concatenated words. More effectively, the modified Kneser–Ney LM [5], [6] was proposed to estimate the higher order LM $p(w|\mathbf{u})$ by linearly interpolating with the lower order LM $p(w|\pi(\mathbf{u}))$ given by backoff context $\pi(\mathbf{u}) \triangleq w_{t-n+2}^{t-1}$ via

$$p(w|\mathbf{u}) = \frac{c_{\mathbf{u}w} - d(c_{\mathbf{u}w})}{c_{\mathbf{u}\cdot}} + \frac{d_1 N_1(\mathbf{u}\cdot) + d_2 N_2(\mathbf{u}\cdot) + d_{3+} N_{3+}(\mathbf{u}\cdot)}{c_{\mathbf{u}\cdot}} p(w|\pi(\mathbf{u})) \tag{3}$$

where $c_{\mathbf{u}w}$ denotes the count of a word sequence $\mathbf{u}w$, $c_{\mathbf{u}\cdot} = \sum_{w' \in \mathcal{V}} c_{\mathbf{u}w'}$ and $N_1(\mathbf{u}\cdot) = |\{w : c_{\mathbf{u}w} = 1\}|$ denotes the number of unique words appearing after $\mathbf{u}$ that have one count. The discount parameter $d(c_{\mathbf{u}w}) \geq 0$ is different for those

Manuscript received August 15, 2014; revised August 3, 2015 and November 2, 2015; accepted November 8, 2015. Date of publication November 23, 2015; date of current version January 18, 2016. This work was supported by the Ministry of Science and Technology, Taiwan, under Contract MOST 103-2221-E-009-078-MY3.

The authors are with the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: jtchien@nctu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2499302

n-grams with zero ($d(c_{\mathbf{u}w} = 0) = 0$), one ($d(c_{\mathbf{u}w} = 1) \triangleq d_1$), two ($d(c_{\mathbf{u}w} = 2) \triangleq d_2$), and three or more counts ($d(c_{\mathbf{u}w} \geq 3) \triangleq d_{3+}$).

A. Continuous Space Model

In recent years, the continuous space method provides a promising alternative that describes words and their semantic and syntactic relationships in a different way. In continuous space language modeling, we represent the words with real-valued vectors. The conditional probabilities of words are learned and expressed as the smoothing functions of these vectors; similar words are, therefore, described as the neighbors in a continuous space. The seen and unseen higher order LMs could be consistently estimated. The NN-LM is a typical example of such a continuous space method. Basically, the feedforward NN-LM (FNN-LM) [7], [8] is constructed by projecting the history words w_{t-n+1}^{t-1} into a continuous space and treating these continuous values as inputs for a multilayer perceptron (MLP). The FNN-LM was successfully applied for ASR in [9]. In [10], the deep FNN-LM was proposed to improve ASR performance by adopting the deep structure of the NN model. However, the FNN-LM could not represent the dependences from distant words, which are the outside n-gram context. Accordingly, the recurrent NN-LM (RNN-LM) [11] was proposed by incorporating the recurrent states into the calculation of LM. Long-distance history words are continuously aggregated through feedback mechanism and then combined with the current word for prediction of a new word. Instead of estimating multinomials in the n-gram LM, the RNN-LM estimates the synaptic weights for three groups of connections between layers, including input-hidden, recurrent-hidden, and hidden-output layers. RNN-LM deals with the data sparseness problem through the continuous space LM and simultaneously accommodates the long-distance information through dynamic feedback. The weaknesses of the n-gram LM are resolved. The RNN-LM has obtained the most competing performance among the advanced LMs [12], [13]. In general, the model complexity of the RNN-LM is controlled by the size of dictionary and the number of hidden units, which are prone to be too large. Although the RNN-LM achieved very good ASR performance, the regularization issue was not tackled, and the performance was constrained. The dropout could be applied to deal with model regularization due to the overtrained model parameters or the wrongly assumed model structure.

B. Main Contribution of This Work

Beyond the state-of-the-art LM using the RNN-LM [11], we apply a standard Bayesian learning approach [14]–[17] to build the regularized LM. This is the first study of implementing the Bayesian RNN-LM (BRNN-LM) for speech recognition. Such a BRNN-LM compensates for model uncertainties by minimizing the regularized cross-entropy error function for a classification network, where the regularization term is governed by a Gaussian hyperparameter. We comparably estimate the RNN-LM parameters through a maximum *a posteriori* (MAP) criterion. The optimal regularization parameter is calculated from the same training data by maximizing the marginal likelihood over the RNN-LM parameters. However, the solution to optimal regularization parameter depends on the calculation of the Hessian matrix, which is computationally expensive, because the summation of a huge amount of high-dimensional outer-product matrices should be implemented. This calculation is controlled by the size of dictionary and the numbers of model parameters and training words, which are all scaled up in real-world applications. We propose a rapid Bayesian learning for the RNN-LM by approximating the Hessian matrices based on a small set of salient outer-product matrices. A positive semidefinite Hessian matrix is produced to facilitate the optimization for the BRNN-LM. The BRNN-LM is demonstrated to be sparser than the RNN-LM. The BRNN-LM achieves robustness in system performance, even disregarding most of the rank-1 outer-product matrices from the computation of the Hessian matrix.

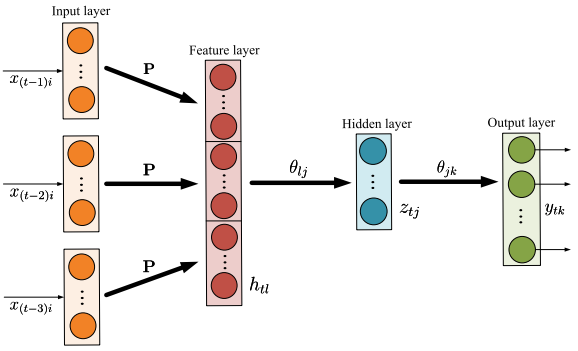

Fig. 1. FNN-LM for the fourth-grams.

The remainder of this paper is organized as follows. In Section II, we introduce the FNN-LM and the RNN-LM. Section III addresses the regularized error backpropagation (BP) through time for the BRNN-LM. Section IV describes the rapid approximation to the Hessian matrix. In Section V, a series of experiments is reported to investigate the proposed BRNN-LM under different tasks and conditions. Finally, the conclusions are drawn in Section VI.

II. Neural Network Language Model

This section addresses the construction of the FNN-LM [7], [9] and the RNN-LM [11].

A. Feedforward Neural Network Model

The continuous space LM based on the FNN was first proposed to estimate the LM parameters for seen as well as unseen word combinations even with a higher order model [7]. Fig. 1 shows the system configuration of the FNN-LM. Instead of estimating the n-gram multinomials, the FNN-LM aims to estimate a set of weight parameters $\boldsymbol{\theta}$ from a sequence of training words $\mathcal{D} = \{w_t\}_{t=1}^{T}$. The FNN-LM encodes a history word w_{t-1} using a $|\mathcal{V}|$-dimensional vector $\mathbf{x}_{t-1} = \{x_{(t-1)i}\}_{i=1}^{|\mathcal{V}|}$ via a 1-of-$|\mathcal{V}|$ coding scheme

$$x_{(t-1)i} = \begin{cases} 1, & \text{if } w_{t-1} \text{ corresponds to word } i \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

For instance, when estimating the fourth gram based on the FNN, three encoded vectors $\{\mathbf{x}_{t-1}, \mathbf{x}_{t-2}$, and $\mathbf{x}_{t-3}\}$ serve as input vectors from the three history words in time steps $t-1$, $t-2$, and $t-3$, respectively. The FNN-LM is constructed to calculate n-grams $\mathbf{y}_t = \{y_{tk}\}_{k=1}^{|\mathcal{V}|}$, with the kth output unit given by

$$y_{tk} = p\left(w_t = k | w_{t-n+1}^{t-1}\right). \tag{5}$$

Here, we use t to denote the time index of a word sequence and i to represent the word index in dictionary $\mathcal{V}$. Each history word vector is simply projected onto a lower dimensional continuous space by using a shared transformation matrix $\mathbf{P}$. Such a projection is performed to avoid the curse of dimensionality in FNN-LM. The discrete words, which are semantically or syntactically related, shall be close to each other in this continuous space. The linearly projected vectors of $n-1$ history words are then concatenated to form a feature vector $\mathbf{h}_t = \{h_{tl}\}_{l=1}^{L}$ as inputs to an MLP with the hidden units $\mathbf{z}_t = \{z_{tj}\}_{j=1}^{K}$ and the output units $\mathbf{y}_t = \{y_{tk}\}_{k=1}^{|\mathcal{V}|}$ subject to the constraint of multinomial outputs $\sum_{k=1}^{|\mathcal{V}|} y_{tk} = 1$. The seen and unseen n-grams in the training data are consistently represented using this continuous space model.

More specifically, the inputs are forwarded to calculate the outputs of hidden neurons $\mathbf{z}_t = \{z_{tj}\}_{j=1}^{K}$ by

$$z_{tj} = \sigma(a_{tj}) = \sigma\left(\sum_{l=1}^{L} \theta_{lj} h_{tl}\right) \tag{6}$$

where the parameters in the feature-hidden layers $\{\theta_{lj}\}$ are used and the activation function $\sigma(\cdot)$ of the activation a_{tj} is given by a sigmoid function $\sigma(a) = (1/(1+\exp(-a)))$. The neurons in the output layer $\{y_{tk}\}$ are produced by combining the outputs of the hidden neurons $\{z_{tj}\}$ with the corresponding weights $\{\theta_{jk}\}$ via a softmax function $s(\cdot)$, which ensures that the values of the output layer are between zero and one, and the sum of these values is always one

$$y_{tk} = s(a_{tk}) = s\left(\sum_{j=1}^{K} \theta_{jk} z_{tj}\right) \tag{7}$$

where

$$s(a_{tk}) = \frac{\exp(a_{tk})}{\sum_{m=1}^{|\mathcal{V}|} \exp(a_{tm})}. \tag{8}$$

The MLP parameters $\boldsymbol{\theta}$ for the feature-hidden layers $\{\theta_{lj}\}$ and the hidden-output layers $\{\theta_{jk}\}$ in such a supervised classification network are trained by maximizing the log likelihood function or equivalently minimizing the cross-entropy error function

$$\begin{aligned} E(\boldsymbol{\theta}) &= -\log p(\mathcal{D}|\boldsymbol{\theta}) \\ &= -\sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|} r_{tk} \log y_{tk} \triangleq \sum_{t=1}^{T} E_t(\boldsymbol{\theta}) \end{aligned} \tag{9}$$

where r_{tk} denotes the target value of a predicted word w_t, which is also encoded by 1-of-$|\mathcal{V}|$ coding scheme for $|\mathcal{V}|$ words in a dictionary as given in (4). The stochastic gradient descent (SGD) algorithm is implemented to estimate new FNN-LM parameters $\boldsymbol{\theta}^{(\text{new})}$ by

$$\boldsymbol{\theta}^{(\text{new})} = \boldsymbol{\theta}^{(\text{old})} - \eta \nabla E_t(\boldsymbol{\theta}^{(\text{old})}) \tag{10}$$

where η is the learning rate. The derivatives $\nabla E_t(\boldsymbol{\theta}^{(\text{old})})$ with respect to current parameters $\boldsymbol{\theta}^{(\text{old})} = \{\theta_{lj}^{(\text{old})}, \theta_{jk}^{(\text{old})}\}$ at time t are calculated by using the error BP algorithm.

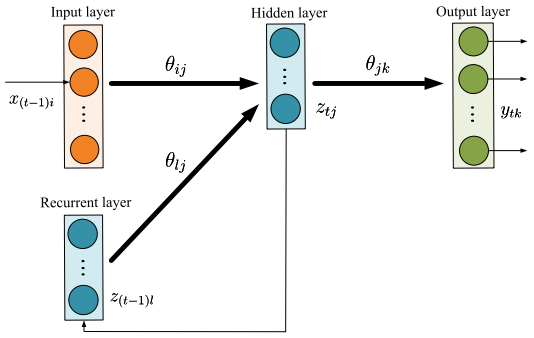

Fig. 2. RNN-LM.

B. Recurrent Neural Network Model

More recently, the FNN-LM is upgraded to the RNN-LM [11], which has been successfully developed for different ASR tasks. Fig. 2 displays the system configuration of the RNN-LM. The differences between the FNN-LM and the RNN-LM are twofold. First, the FNN-LM conducts a linear transformation to project the $n-1$ history words w_{t-n+1}^{t-1} onto a continuous space, while the RNN-LM does not perform any linear projection. For each time step t, the RNN-LM only considers an input of the previous word w_{t-1}. Second, the RNN-LM adopts a K-dimensional recurrent state vector $\mathbf{z}_{t-1}$ from the hidden layer of previous time step $t-1$, which is combined with the history word $\mathbf{x}_{t-1}$ as inputs for the MLP modeling. The recurrent context vector $\mathbf{z}_{t-1}$ comes from the previous outputs of the hidden neurons given the input word w_{t-2}. In this manner, the context vector or the state vector $\mathbf{z}_t$ is accumulated in all learning time steps t. Different from the FNN-LM, the RNN-LM effectively adopts this recurrent state vector to capture the contextual information of a word sequence starting from the first word w_1 to the current word w_{t-1}. The output neurons of the RNN-LM are equivalent to calculate the long-distance LM

$$y_{tk} = p\left(w_t = k | w_1^{t-1}\right) \tag{11}$$

for the prediction of individual words $\{w_t = k\}_{k=1}^{|\mathcal{V}|}$. The RNN-LM tackles the issues of n-grams due to sparse training data as well as insufficient long-distance information, while the FNN-LM only handles the issue of sparse training data. Notably, such a RNN-LM [11] is not the case of using the simple RNN in [18], which tends to forget information at a similar rate as the FNN, i.e., there is hardly any influence of distant words on the resulting probability estimation.

Using the RNN-LM, the history word w_{t-1} is encoded as $|\mathcal{V}|$-dimensional vector $\mathbf{x}_{t-1}$ based on (4). The groups of

synaptic parameters $\boldsymbol{\theta} = \{\theta_{ij}, \theta_{lj}, \theta_{jk}\}$ are used to characterize the connections for the input-hidden layers, recurrent-hidden layers, and hidden-output layers, respectively. Different from the FNN-LM, the outputs of the hidden neurons $\mathbf{z}_t = \{z_{tj}\}_{j=1}^{K}$ for each input word w_{t-1} are calculated by

$$z_{tj} = \sigma(a_{tj}) = \sigma\left(\sum_{i=1}^{|\mathcal{V}|} \theta_{ij} x_{(t-1)i} + \sum_{l=1}^{K} \theta_{lj} z_{(t-1)l}\right) \quad (12)$$

which combines the information from word $\mathbf{x}_{t-1}$ and the recurrent context $\mathbf{z}_{t-1}$ from the previous words w_1^{t-2} by using the synaptic weights θ_{ij} and θ_{lj}, respectively. The outputs of the RNN are calculated in the same way as those of the FNN in (7) and (8). Again, the error BP algorithm is applied to estimate the RNN-LM parameters $\boldsymbol{\theta}$ from a set of training words $\mathcal{D} = \{w_t\}_{t=1}^{T}$ by minimizing the cross-entropy error function in (9). For ease of expression, we ignore the bias parameters in formulation of different NN-LMs. The total number of weight parameters in the RNN-LM is determined by $N = 2 \cdot |\mathcal{V}| \cdot K + K^2$, where the first term is due to the parameters $\{\theta_{ij}\}$ and $\{\theta_{jk}\}$ and the second term is caused by the parameters $\{\theta_{lj}\}$.

III. Bayesian Language Model

This section addresses the motivation of Bayesian learning, surveys the regularized LMs, and formulates the proposed BRNN-LM.

A. Model Regularization

The overfitting problem or the ill-posed condition happens in many pattern recognition applications where the mismatch between training and test data and the risk of wrongly assumed model always exist [16], [17]. The collected data are usually noisy, mislabeled, ambiguous, misaligned, and nonstationary. The assumed model is prone to be overtrained in the presence of heterogeneous environments. Model generalization is not guaranteed for prediction or classification of future data. Such an adverse condition is obvious in the construction of LM from real-world data collection, where the domains between training and test corpora are different, the speech sentences may be wrongly transcribed, the spoken language may be too spontaneous, and the text data may be syntactically or semantically ill-formed with grammatical errors. LMs should be constructed by compensating for the variations in the ill-posed condition and the uncertainties due to the improper model structure or inevitable heterogeneous observations, so that the robustness for speech recognition could be assured [19].

B. Survey of the Regularized Language Models

There have been several approaches developed to handle model regularization for LM and apply it for the ASR system. A simple way to tackle the regularization issue was developed by adapting the n-gram parameters to a new test domain through the MAP adaptation [20]. The mismatch between training and test data was alleviated. In addition, the LM based on model M achieved a high-performance ASR in many tasks [21]. This model shrank the maximum entropy LM [22] to a middle-sized exponential model, which resulted in an improved prediction of test data. An empirical relationship between cross entropies of training and test data was investigated as a heuristic to perform model adaptation to improve ASR for test data. A middle-sized class-based LM was established. In [3], a continuous space LM called the cache Dirichlet class LM (cDC-LM) was proposed by integrating the real-valued topic or class information from the history words w_{t-n+1}^{t-1} into prediction of a new word w_t. The topic-based LM parameters were estimated by maximizing the marginal likelihood over different topics and topic proportionals, so that the uncertainties or topic variations were compensated for future prediction [23]. This model corresponds to a Bayesian regularized LM.

On the other hand, a nonparametric Bayesian approach was introduced to build the smoothed LM, which was used to interpret the modified Kneser–Ney LM in (3) from the Bayesian perspective [24]. The interpolation of a higher order LM with a lower order LM was seen as a hierarchical integration of a Pitman–Yor process [25] of a higher order model with a base measure drawn from a lower order Pitman–Yor process. These processes were controlled by the discount parameters that were used to resemble the power-law property in natural language. Such a nonparametric Bayesian LM was further extended to build a topic-based LM, where the size of topics was automatically grown up from the gradually increasing amount of training data by additionally incorporating a Dirichlet process [26]. Nevertheless, this paper focuses on model regularization for the construction of RNN-LM based on the Bayesian theory and investigates its effect on model perplexity and speech recognition. The merit of this paper is to introduce an optimal regularization term or prior distribution into training of RNN-LM, which improves the model generalization. The proposed BRNN-LM is different from [27], which was designed for the time-series modeling and different from [28] and [15], which was developed for the combination of the MLP and the hidden Markov model (HMM) for acoustic modeling.

C. Maximum a posteriori Estimation

There are two stages in Bayesian inference. The first stage is to estimate model parameters by incorporating the prior distribution of model parameters through the MAP estimation. The MAP estimate depends on a regularization parameter or a control hyperparameter α. In the second stage, we estimate the optimal hyperparameter by maximizing the marginal likelihood over the model parameters $\boldsymbol{\theta}$. No cross validation data are required. Only the training data $\mathcal{D}$ is applied to find the model parameters $\boldsymbol{\theta}$ as well as the hyperparameter α. In particular, we introduce an isotropic Gaussian prior $p(\boldsymbol{\theta}|\alpha) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \alpha^{-1}\mathbf{I})$, where α is the precision parameter and $\mathbf{I}$ is the N-dimensional identity matrix, to characterize the distribution of model parameters. This prior density is combined with the likelihood function in (9). The combination is performed at each time step t. The MAP parameter $\boldsymbol{\theta}_{\text{MAP}}$ is obtained by maximizing the logarithm of posterior distribution or minimizing the regularized cross-entropy

error function

$$\tilde{E}(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}|\mathcal{D}, \alpha) \propto -\sum_{t=1}^{T}\left(\sum_{k=1}^{|\mathcal{V}|} r_{tk}\log y_{tk} + \frac{\alpha}{2}\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{\theta}\right) \triangleq \sum_{t=1}^{T}\tilde{E}_t(\boldsymbol{\theta}) \quad (13)$$

where each output $y_{tk} \triangleq y_t(\mathbf{x}_{t-1}, \boldsymbol{\theta})$ is a function of the history word $\mathbf{x}_{t-1}$ and the model parameters $\boldsymbol{\theta}$. In (13), the first term $E(\boldsymbol{\theta})$ is known as the cross entropy between the target values $\{r_{tk}\}$ and the neural network outputs $\{y_{tk}\}$. The second term is obtained from the negative logarithm of a Gaussian prior, which is seen as a penalty function or a weight decay that penalizes the too complicated model. The hyperparameter α is treated as a regularization parameter in this regularized cross-entropy error function. A regularized and supervised classification network is built for language modeling.

D. Regularized Error Backpropagation

The SGD algorithm [18] is introduced to develop the regularized error BP algorithm for sequential learning of the BRNN-LM parameters. Each word w_{t-1} in text corpus $\mathcal{D} = \{w_t\}_{t=1}^{T}$ is used to predict the next word w_t and to update the model parameters according to (10) except that error function $E(\boldsymbol{\theta})$ is replaced by the regularized error function $\tilde{E}(\boldsymbol{\theta})$. During the implementation, we first calculate the local gradient of an output neuron k with activation a_{tk} at time t by

$$\frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}} = \sum_{m=1}^{|\mathcal{V}|}\frac{\partial E_t(\boldsymbol{\theta})}{\partial y_{tm}}\frac{\partial y_{tm}}{\partial a_{tk}} = y_{tk} - r_{tk} \triangleq \delta_{tk}. \quad (14)$$

The derivative of the regularized cross-entropy error function $\tilde{E}_t(\boldsymbol{\theta})$ with respect to the hidden-output parameter θ_{jk} is derived as

$$\frac{\partial \tilde{E}_t(\boldsymbol{\theta})}{\partial \theta_{jk}} = \frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}}\frac{\partial a_{tk}}{\partial \theta_{jk}} + \frac{\alpha}{2}\frac{\partial ||\boldsymbol{\theta}||^2}{\partial \theta_{jk}} = \delta_{tk}z_{tj} + \alpha\theta_{jk}. \quad (15)$$

Next, the local gradient of a hidden neuron j with activation a_{tj} at time t is calculated by

$$\frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tj}} = \sum_{k=1}^{|\mathcal{V}|}\frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}}\frac{\partial a_{tk}}{\partial z_{tj}}\frac{\partial z_{tj}}{\partial a_{tj}} = \sum_{k=1}^{|\mathcal{V}|}\delta_{tk}\theta_{jk}z_{tj}(1 - z_{tj}) \triangleq \delta_{tj} \quad (16)$$

where the summation is computed for all connections between output neuron k and hidden neuron j. In (16), the derivative of a sigmoid function $(\partial z_{tj}/\partial a_{tj}) = z_{tj}(1 - z_{tj})$ is used. Similarly, the derivatives of the regularized cross-entropy error function with respect to the input-hidden parameter θ_{ij} and the recurrent-hidden parameter θ_{lj} are obtained by

$$\frac{\partial \tilde{E}_t(\boldsymbol{\theta})}{\partial \theta_{ij}} = \frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tj}}\frac{\partial a_{tj}}{\partial \theta_{ij}} + \frac{\alpha}{2}\frac{\partial ||\boldsymbol{\theta}||^2}{\partial \theta_{ij}} = \delta_{tj}x_{(t-1)i} + \alpha\theta_{ij} \quad (17)$$

$$\frac{\partial \tilde{E}_t(\boldsymbol{\theta})}{\partial \theta_{lj}} = \frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tj}}\frac{\partial a_{tj}}{\partial \theta_{lj}} + \frac{\alpha}{2}\frac{\partial ||\boldsymbol{\theta}||^2}{\partial \theta_{lj}} = \delta_{tj}z_{(t-1)l} + \alpha\theta_{lj}. \quad (18)$$

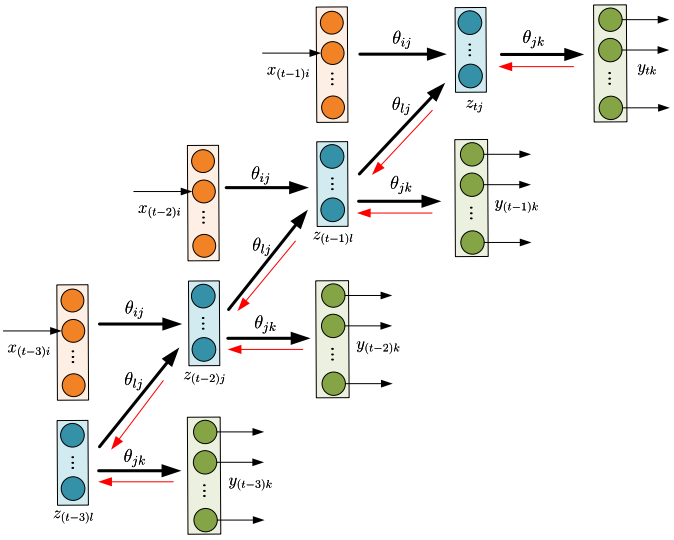

Fig. 3. RNN-LM unfolded as a deep feedforward network with three steps back in time. Red arrows indicate how the derivatives are propagated through the unfolded RNN.

The MAP estimates $\boldsymbol{\theta}_{\text{MAP}}$ for the groups of the hidden-output θ_{jk}, input-hidden θ_{ij}, and recurrent-hidden parameters θ_{lj} are obtained according to the regularized error BP procedure based on the derivatives in (15), (17), and (18), respectively. The derivatives of regularization term $(\alpha/2)||\boldsymbol{\theta}||^2$ with respect to three sets of parameters are included.

E. Backpropagation Through Time

BP through time (BPTT) [29] is known as an extended error BP algorithm for the RNN. In Section III-D, the regularized error BP is designed by considering only one previous hidden state $\mathbf{z}_{t-1}$ for concatenating with the input word $\mathbf{x}_{t-1}$. With the truncated BPTT, the BRNN-LM could be improved by propagating the regularized error function through recurrent connections back in time for a specific number of time steps (here referred to as τ). Such network learns to remember the contextual information of the history words for several time steps in the hidden layer. Fig. 3 shows the unfolded RNN-LM, where the truncated BPTT with $\tau = 3$ is applied. This model is seen as a deep feedforward network. The red arrows show how the derivatives propagated in the backward procedure. The BPTT with a large τ may lead to the issue of vanishing gradient [18].

According to the truncated BPTT with τ steps back in time, the regularized error function at current time t is accumulated from time index $t - \tau + 1$ to current time t in a form of

$$\tilde{E}_t^b(\boldsymbol{\theta}) = \sum_{m=t-\tau+1}^{t}\tilde{E}_m(\boldsymbol{\theta}). \quad (19)$$

SGD algorithm is then applied to estimate BRNN-LM parameters via

$$\boldsymbol{\theta}^{(\text{new})} = \boldsymbol{\theta}^{(\text{old})} - \eta\nabla\tilde{E}_t^b(\boldsymbol{\theta}^{(\text{old})}). \quad (20)$$

The updating rule of the hidden-output parameter θ_{jk} is not changed by the BPTT scheme, because the error

function $\tilde{E}_t^b(\boldsymbol{\theta})$ at each time step t due to the connections between hidden and output units is not affected by unfolding the RNN-LM. The derivative in (15) is applied to learn the hidden-output parameter θ_{jk}. However, the updating rules of the input-hidden parameter θ_{ij} and the recurrent-hidden parameter θ_{lj} are considerably affected by the BPTT with τ steps back in time. The learning algorithm for these two parameters is developed by using the derivatives

$$\frac{\partial \tilde{E}_t^b(\boldsymbol{\theta})}{\partial \theta_{ij}} = \sum_{m=t-\tau+1}^{t} \frac{\partial \tilde{E}_m(\boldsymbol{\theta})}{\partial \theta_{ij}} = \sum_{m=t-\tau+1}^{t} (\delta_{mj} x_{(m-1)i} + \alpha \theta_{ij}) \tag{21}$$

and

$$\frac{\partial \tilde{E}_t^b(\boldsymbol{\theta})}{\partial \theta_{lj}} = \sum_{m=t-\tau+1}^{t} \frac{\partial \tilde{E}_m(\boldsymbol{\theta})}{\partial \theta_{lj}} = \sum_{m=t-\tau+1}^{t} \left(\delta_{mj} z_{(m-1)l} + \alpha \theta_{lj}\right). \tag{22}$$

To implement the BRNN-LM based on the BPTT algorithm, we need to choose a proper regularization parameter α. How to determine the optimal regularization parameter is seen as a model selection problem that could be tackled through Bayesian learning.

F. Estimation of Regularization Parameter

Instead of applying the cross validation[1] to find regularization parameter α from additional validation data, it is more attractive to directly estimate the optimal α from the same training data $\mathcal{D}$ based on the type 2 maximum likelihood (ML2) estimation [15] or the evidence framework [30]

$$\alpha_{\mathrm{ML2}} = \arg\max_{\alpha} \; p(\mathcal{D}|\alpha) \tag{23}$$

where the objective for optimization is formed by the marginal likelihood over the RNN-LM parameters $\boldsymbol{\theta}$. The integral over $\boldsymbol{\theta}$ is derived by Laplace approximation due to the nonlinearity in likelihood function $p(\mathcal{D}|\boldsymbol{\theta})$

$$p(\mathcal{D}|\alpha) = \int p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\alpha) d\boldsymbol{\theta} \simeq f(\boldsymbol{\theta}_{\mathrm{MAP}}) \times \int \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{MAP}})^{\mathsf{T}} \mathbf{A} (\boldsymbol{\theta} - \boldsymbol{\theta}_{\mathrm{MAP}})\right) d\boldsymbol{\theta} \tag{24}$$

where $f(\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\alpha)$ and $\mathbf{A} \triangleq -\nabla\nabla \log f(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\mathrm{MAP}}} = \mathbf{H} + \alpha \mathbf{I}$. The marginal likelihood is now approximated by

$$\log p(\mathcal{D}|\alpha) \simeq \log p(\mathcal{D}|\boldsymbol{\theta}_{\mathrm{MAP}}) + \log p(\boldsymbol{\theta}_{\mathrm{MAP}}|\alpha) - \frac{1}{2} \log |\mathbf{A}| \propto -\tilde{E}(\boldsymbol{\theta}_{\mathrm{MAP}}) + \frac{N}{2} \log \alpha - \frac{1}{2} \log |\mathbf{A}| \tag{25}$$

[1]In statistical theory, divide up the training set into many subsets. Using the one subset (training data) to train the model, the other subsets (validation data) will verify and confirm the model in practice.

where N is the number of parameters, and $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^N$ denotes the collection of three groups of parameters and

$$|\mathbf{A}| = \prod_{i=1}^{N} (\lambda_i + \alpha) \tag{26}$$

is calculated through finding the eigenvalues λ_i of the Hessian matrix

$$\mathbf{H} = -\nabla\nabla \log p(\mathcal{D}|\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\mathrm{MAP}}}. \tag{27}$$

We then take the derivative of marginal likelihood with respect to α

$$\frac{d}{d\alpha} \log p(\mathcal{D}|\alpha) = -\frac{\boldsymbol{\theta}_{\mathrm{MAP}}^T \boldsymbol{\theta}_{\mathrm{MAP}}}{2} + \frac{N}{2\alpha} - \frac{1}{2} \sum_{i=1}^{N} \frac{1}{\lambda_i + \alpha} \tag{28}$$

and set it to zero to find the optimal hyperparameter [15], [16]

$$\alpha_{\mathrm{ML2}} = \frac{\gamma}{\boldsymbol{\theta}_{\mathrm{MAP}}^T \boldsymbol{\theta}_{\mathrm{MAP}}} \tag{29}$$

where $\gamma = \sum_{i=1}^{N} (\lambda_i / (\lambda_i + \alpha))$ denotes the effective number of parameters. If α is relatively smaller than λ_i, the value of γ is closer to N where more parameters in $\boldsymbol{\theta}$ are effective. Contrarily, if the prior $p(\boldsymbol{\theta}|\alpha)$ is sparser with larger α, it is meaningful that the effective number of parameters in $\boldsymbol{\theta}$ become smaller.

Basically, (29) is an implicit solution to α, since its right-hand side (RHS) is also a function of α. In the implementation, we continue the recursive estimation of three variables

$$\boldsymbol{\theta}_{\mathrm{MAP}}^{(1)} \rightarrow \mathbf{H}^{(1)} \rightarrow \alpha_{\mathrm{ML2}}^{(1)} \rightarrow \boldsymbol{\theta}_{\mathrm{MAP}}^{(2)} \rightarrow \mathbf{H}^{(2)} \rightarrow \alpha_{\mathrm{ML2}}^{(2)} \cdots \tag{30}$$

until convergence. In (30), the superscripts imply the iteration index. Such an implicit solution converges rapidly. The BRNN-LM is accordingly constructed by an iterative three-step procedure.

1) Train the network and compute the MAP estimates of weights $\boldsymbol{\theta}_{\mathrm{MAP}}$.
2) Calculate the Hessian matrix $\mathbf{H}$.
3) Reestimate the regularization parameter α_{ML2}.

In general, the estimated hyperparameter or the regularization parameter α_{ML2} could compensate for the variations of parameters $\boldsymbol{\theta}$ or avoid the overfitting problem in the RNN-LM. The prediction for test data could be improved.

IV. Rapid Bayesian Learning

The BRNN-LM relies on the computation of the Hessian matrix $\mathbf{H} = \{H_{ij}\}_{(i,j)=1}^N$ that is formed by the second-order derivatives of the negative logarithm of marginal likelihood with respect to the high-dimensional parameters $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^N$.

A. Hessian Matrix

Each entry of the Hessian matrix H_{ik} is computed from the first-order derivative

$$\frac{\partial \left(-\log p(\mathcal{D}|\boldsymbol{\theta})\right)}{\partial \theta_i} = \sum_{t=1}^{T} \sum_{k=1}^{|\mathcal{V}|} \frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}} \frac{\partial a_{tk}}{\partial \theta_i} \tag{31}$$

to the second-order derivative

$$\begin{aligned} H_{ij} &= \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\frac{\partial}{\partial\theta_j}\left(\frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}}\frac{\partial a_{tk}}{\partial\theta_i}\right) \\ &= \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\left[\sum_{m=1}^{|\mathcal{V}|}\frac{\partial}{\partial a_{tm}}\left(\frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}}\right)\frac{\partial a_{tm}}{\partial\theta_j}\frac{\partial a_{tk}}{\partial\theta_i}\right. \\ &\qquad \left. + \frac{\partial E_t(\boldsymbol{\theta})}{\partial a_{tk}}\frac{\partial^2 a_{tk}}{\partial\theta_i\partial\theta_j}\right] \\ &= \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\left[\sum_{m=1}^{|\mathcal{V}|} y_{tk}(I_{km}-y_{tm})\frac{\partial a_{tm}}{\partial\theta_j}\frac{\partial a_{tk}}{\partial\theta_i}\right. \\ &\qquad \left. +(y_{tk}-r_{tk})\frac{\partial^2 a_{tk}}{\partial\theta_i\partial\theta_j}\right] \end{aligned} \tag{32}$$

where y_{tk} and y_{tm} denote the LMs of the next word $w_t = k$ and $w_t = m$ given all history words w_1^{t-1}, and $I_{km} = 1$ for $k = m$ and $I_{km} = 0$ for $k \neq m$. In (32), we apply (14) and use the derivatives of a softmax function with respect to different output activations

$$\frac{\partial y_{tk}}{\partial a_{tm}} = \begin{cases} y_{tk}(1-y_{tk}), & \text{if } m = k \\ -y_{tk}y_{tm}, & \text{if } m \neq k. \end{cases} \tag{33}$$

Assuming the RNN-LM is well trained with the output value y_{tk} very close to the target value r_{tk}, i.e., $y_{tk} \simeq r_{tk}$, the second term in the RHS of (32) can be ignored. Each entry of the Hessian matrix is then approximated by

$$H_{ij} \simeq \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\sum_{m=1}^{|\mathcal{V}|} y_{tk}(I_{km}-y_{tm})\frac{\partial a_{tm}}{\partial\theta_j}\frac{\partial a_{tk}}{\partial\theta_i}. \tag{34}$$

In this approximation, we need to calculate the derivative of output activation with respect to individual parameters $\boldsymbol{\theta}$ in the input-hidden, recurrent-hidden, and hidden-output layers

$$\nabla a_{tk} = \left[\frac{\partial a_{tk}}{\partial\theta_1},\ldots,\frac{\partial a_{tk}}{\partial\theta_N}\right]^{\mathsf{T}}. \tag{35}$$

As a result, the training procedure of the BRNN-LM is implemented by estimating $\boldsymbol{\theta}_{\text{MAP}}$ and α_{ML2} according to Algorithm 1. We can see that the computation overhead for finding a regularization parameter α_{ML2} is considerable, because the complexity of the computing Hessian matrix $\mathbf{H}$ using (34) is determined by $\mathcal{O}(T|\mathcal{V}|^2N^2)$ which is highly affected by large numbers of the training words T, the dictionary words $|\mathcal{V}|$, and the estimated parameters N.

B. Rapid Approximation

In order to tackle the curse of dimensionality in the BRNN-LM, we propose an approach to speed up the computation of the Hessian matrix for rapid Bayesian learning of the RNN. First of all, we arrange the computation of the Hessian matrix by

$$\begin{aligned} \mathbf{H} \simeq \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\sum_{m=1}^{|\mathcal{V}|}\underbrace{y_{tk}(I_{km}-y_{tm})\|\nabla a_{tm}\|\|\nabla a_{tk}\|}_{\triangleq q_{tkm}} \\ \times\frac{\nabla a_{tm}}{\|\nabla a_{tm}\|}\frac{(\nabla a_{tk})^{\mathsf{T}}}{\|\nabla a_{tk}\|}. \end{aligned} \tag{36}$$

Algorithm 1 Regularized Error BP

Input: Training words $\mathcal{D} = \{w_t\}_{t=1}^{T}$ and their encoded vectors $\{\mathbf{x}_t\}_{t=1}^{T}$
Output: BRNN-LM parameters $\boldsymbol{\theta}_{\text{MAP}} = \{\theta_{ij}, \theta_{lj}, \theta_{jk}\}$

```
for t = 1, ..., T do
    Feedforward the network through (12) and (7)
    Evaluate δ_tk for output neurons using (14)
    Backpropagate δ_tj for hidden neurons using (16)
    Update θ_MAP using α & derivatives (15)(21)(22)
end
for i = 1, ..., N do
    for j = 1, ..., N do
        Calculate the Hessian entry H_ij using (34)
    end
end
Calculate the eigenvalues of Hessian matrix H
Update the hyperparameter α_ML2 using (29)
Repeat the whole process until θ_MAP is converged
```

There are $T|\mathcal{V}|^2$ calculations of $N \times N$ the outer-product matrices of the normalized gradient vectors ∇a_{tm} and ∇a_{tk}. Each outer-product is a rank-1 matrix. The notation q_{tkm} is defined as a weight in the linear combination of these outer-products. For a practical consideration, we alleviate the computation by taking the summation of the outer-products with only $k = m$ in the BRNN-LM. The outer-products with $k \neq m$ are disregarded. The Hessian matrix is accordingly approximated as

$$\mathbf{H} \simeq \sum_{t=1}^{T}\sum_{k=1}^{|\mathcal{V}|}\underbrace{y_{tk}(1-y_{tk})\|\nabla a_{tk}\|^2}_{\triangleq q_{tk}}\frac{\nabla a_{tk}}{\|\nabla a_{tk}\|}\frac{(\nabla a_{tk})^{\mathsf{T}}}{\|\nabla a_{tk}\|}. \tag{37}$$

In addition to the computational concern, the approximated Hessian matrix in (37) is assured to be positive semidefinite, because $q_{tk} \geq 0$ and the self outer-product $\nabla a_{tk}(\nabla a_{tk})^{\mathsf{T}}$ is always positive semidefinite. This property implies that the estimated $\boldsymbol{\theta}_{\text{MAP}}$ can converge to a global optimum.

Furthermore, we examine the values of combination weights $\{q_{tk}\}$ from different vocabulary words $w_t = k \in \mathcal{V}$ and find that a large portion of weights has very small values. The corresponding outer-product $\nabla a_{tk}(\nabla a_{tk})^{\mathsf{T}}$ is redundant in the calculation of the Hessian matrix. Therefore, we are motivated to compute only for the outer-products with the significant values of q_{tk}. The sorting of $T|\mathcal{V}|$ elements of $\{q_{tk}\}$ in descending order is conducted to identify those salient terms in (37) within the top percentage $P\%$ of the values of $\{q_{tk}\}$. The approximation to the Hessian matrix is implemented in Algorithm 2.

C. Analysis of Computation Complexity

The computation complexity of (36) is given by $\mathcal{O}(T|\mathcal{V}|^2N^2)$. Using (37), the computation complexity is greatly reduced to $\mathcal{O}(T|\mathcal{V}|N) + \mathcal{O}(LN^2)$, where the first term is due to the calculation of q_{tk} and the second term is caused by the calculation of the L outer-products of the

Algorithm 2 Rapid Calculation of Hessian Matrix

Input: RNN outputs and derivatives $\{y_{tk}, \nabla a_{tk}\}_{t=1,k=1}^{T,|\mathcal{V}|}$
Output: Hessian matrix $\mathbf{H}$

for $t = 1, \ldots, T$ **do**
  for $k = 1, \ldots, |\mathcal{V}|$ **do**
    $q_{tk} = y_{tk}(1 - y_{tk})\|\nabla a_{tk}\|^2$
  end
end
Sort $\{q_{tk}\}$ in descending order
Select top $P\%$ of the sorted $\{q_{tk}\}$, say L elements
Calculate $\mathbf{H} \simeq \sum_{l=1}^{L} y_l(1 - y_l)\nabla a_l(\nabla a_l)^{\mathsf{T}}$

top P percentage of q_{tk}. The redundancy in calculating $\mathbf{H}$ is significantly reduced, because the number of selected elements L in the implementation is much smaller than the number of training words T and the size of dictionary $|\mathcal{V}|$. We obtain the property $L \ll T|\mathcal{V}|$. Therefore, the rapid Bayesian learning for the RNN-LM is achieved, since the computation complexity $\mathcal{O}(T|\mathcal{V}|N)+\mathcal{O}(LN^2)$ in (37) is much simpler than $\mathcal{O}(T|\mathcal{V}|^2N^2)$ in (36).

V. Experiments

A series of experiments is conducted to investigate the effects of the estimated parameters and the hyperparameter based on the BRNN-LM. The experimental conditions on the convergence property, the selection percentage P, the number of hidden units K, the number of time steps τ in the BPTT, the size of training data T, and the model interpolation are evaluated. We compare different LMs in terms of training time, perplexity, and word error rate (WER) (%). Perplexity of an LM using the test data $\mathcal{D} = \{w_t\}_{t=1}^{T}$ is calculated by

$$10^{H(\mathcal{D})} = 10^{-\frac{1}{T}\sum_{t=1}^{T}\log_{10} p(w_t|w_1,\ldots,w_{t-1})} \tag{38}$$

where $H(\mathcal{D})$ denotes the cross entropy of LM. Perplexity is interpreted as the average number of branches in the text. The LM with a lower perplexity implies less confusion and more likely achieves better speech recognition performance.

A. Experimental Setup

The proposed BRNN-LM was evaluated by using three corpora as described in what follows [31] and [32].

1) Penn Treebank Corpus: This corpus comes from a portion of Wall Street Journal (WSJ) corpus. We adopted the sections 0–20 as the training set with 930-k words, the sections 21 and 22 as the validation set with 74k words, and the sections 23 and 24 as test set with 82k words. The vocabulary size $|\mathcal{V}|$ of this corpus is 10k.

2) Benchmark Corpus: This corpus was sampled from the 1B-Word-Benchmark corpus[2] [13] that consisted of 0.8 billions words with 100 disjoint partitions (about 8M words per partition). The vocabulary size is about 191k words. One such partition (held-out set) was split again into 50 disjoints to be used as the validation data and test data. We followed the same validation data and the test data as described in [13]. We randomly sampled 20 partitions from the other 99 disjoint partitions of all data as the training data. Each partition was used to estimate an individual LM. Perplexity was measured by averaging over those from 20 models. The full 1B-Word-Benchmark corpus is much larger than the subset used in this paper.

3) Wall Street Journal Corpus: This corpus contains broadcast news data and is used to evaluate the performance of the CSR. The WSJ0 (SI-84) training set was adopted to estimate the acoustic models. Triphone models were built for 39 phones and one background silence. The training data consisted of 7138 utterances from 83 speakers. The 1987–1989 WSJ corpus with 38M words was adopted to train LMs. A total of 330 and 333 utterances were sampled from the November 1992 ARPA CSR benchmark test data with the vocabulary sizes of 5k and 20k, respectively. These utterances were used to examine the model perplexity and WER using different LMs. The development sets with 1951 and 4002 utterances were adopted for the vocabulary sizes of 5k and 20k, respectively [3].

SRI toolkit [33] was adopted to evaluate the perplexity of LMs. The FNN-LM [7], [9] and the cDC-LM [3] were carried out for comparison. The RNN-LM was implemented by using [34]. In implementation of the FNN-LM, the RNN-LM, and the BRNN-LM using the SGD algorithm, the initial learning rate η was set to be 0.1. If log likelihood of validation set was not improved, we kept training NN-LM in the next learning epoch. In the case of no improvement again, η was halved to let network learning more stable and accurate. The initial prior hyperparameter α was set to be 0.01. The error BP and the error BPTT were implemented for comparison. For comparative study, the proposed rapid BRNN-LM (denoted by R-BRNN-LM hereafter) was implemented under different selection percentages $P\%$ in calculation of the Hessian matrix. The R-BRNN with $P = 100\%$ turns out to be the standard BRNN.

We carried out the results of the individual models based on 5 g with KN smoothing (denoted by KN5), RNN-LM, BRNN-LM, and R-BRNN-LM, and also the hybrid models by interpolating RNN-LM, BRNN-LM, and R-BRNN-LM with KN5 through [12]

$$\begin{aligned} p(w_t|w_1^{t-1}) = {} & \beta p_{\text{rnn}}\left(w_t|w_1^{t-1}\right) \\ & +(1-\beta)p_{\text{kn}}\left(w_t|w_{t-n+1}^{t-1}\right) \end{aligned} \tag{39}$$

where $0 \le \beta \le 1$ is the interpolation coefficient. In this paper, the linear interpolation scheme was also applied to estimate the integrated LM, which combined LMs under different experimental conditions in a form of

$$p(w_t|w_1^{t-1}) = \sum_{m=1}^{M} \beta_m\, p_{\text{rnn}}\left(w_t|w_1^{t-1}, m\right) \tag{40}$$

where $0 \le \beta_m \le 1$ and $\sum_{m=1}^{M}\beta_m = 1$. Model regularization could be improved accordingly.

The training time in minutes was measured by using the personal computer equipped with a CPU of Intel Core i7-4930K at 3.4 GHz six cores and a memory of 64-GB RAM. This time

[2]https://code.google.com/p/1-billion-word-language-modeling-benchmark/

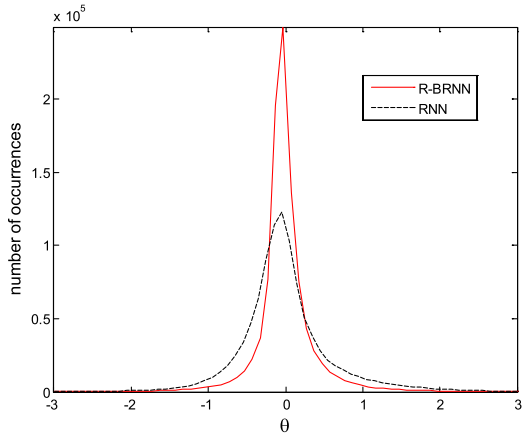

Fig. 4. Histograms of the estimated weights $\boldsymbol{\theta}$ by using the RNN-LM and the R-BRNN-LM.

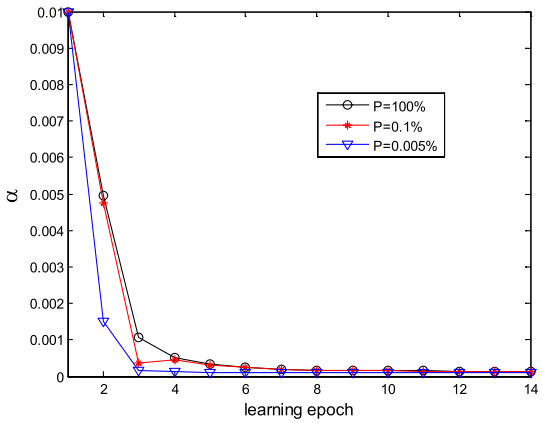

Fig. 5. Learning curves of the estimated hyperparameter α using the R-BRNN-LM under different selection percentages P.

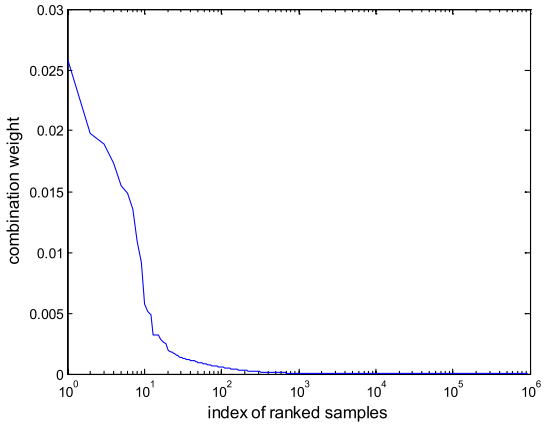

Fig. 6. Distribution of the sorted combination weights q_{tk}.

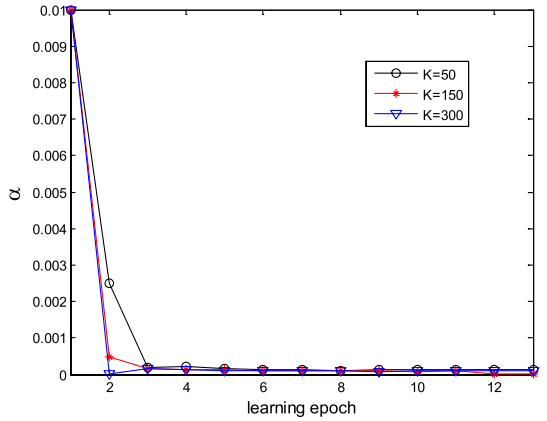

Fig. 7. Learning curves of the estimated hyperparameter α using the R-BRNN-LM under different numbers of the hidden units K.

included all of the calculations for estimation of the parameters and the hyperparameter that might be converged in different epochs.

B. Evaluation for the Estimated Model Parameters

Fig. 4 shows the histograms of the weight parameters $\boldsymbol{\theta}$ that are estimated from the three groups of parameters $\{\theta_{ij}, \theta_{lj}, \theta_{jk}\}$ in the RNN-LM and R-BRNN-LM under $K = 50$ and $P = 0.005\%$ by using the training set of PT corpus. We can see that there are many parameters with values close to zero. The standard deviation of the distribution of $\boldsymbol{\theta}$ using the R-BRNN-LM is measured as 0.44, which is much smaller than that using the RNN-LM that is 0.64. The R-BRNN estimates the sparser weights for language modeling than the RNN. The property of the sparse parameters in the R-BRNN-LM is helpful to handle the overfitted parameters and accordingly improve the model regularization or system robustness for the RNN-LM [35]. Such a property is similarly obtained for individual groups of model parameters from our investigation.

C. Evaluation for the Estimated Regularization Parameter

Fig. 5 shows the learning curves of the estimated regularization parameter α in the R-BRNN-LM by using $K = 50$ and different percentages P in rapid approximation of the Hessian matrix. PT corpus is used. Typically, the R-BRNN-LM converges well under different percentages. In this comparison, the estimation of optimal α in the R-BRNN using a percentage $P = 0.005\%$ converges faster than that using the percentages $P = 0.1\%$ and $P = 100\%$ (also known as BRNN). Rapid approximation to the Hessian matrix is beneficial for the convergence of model training based on the BRNN-LM.

To investigate the effect of selection percentage P in the outer-product approximation to the Hessian matrix, we further examine the distribution the sorted values of the normalized combination weights $\{q_{tk}\}$, as shown in Fig. 6. Among the samples of combination weights in an order of 10^6, only thousand or even a few hundreds of samples have significant values of combination weights. Most of them are too small to contribute the calculation of the Hessian matrix. This is the reason why we select a small percentage of salient samples for rapid calculation of $\mathbf{H}$.

In addition, we fix $P = 0.005\%$ and evaluate the learning curves of hyperparameter α in the R-BRNN-LM by changing the number of hidden units (K). Here, the cases of $K = 50$, $K = 150$, and $K = 300$ are evaluated. As shown in Fig. 7, the convergence of the estimated hyperparameter goes well.

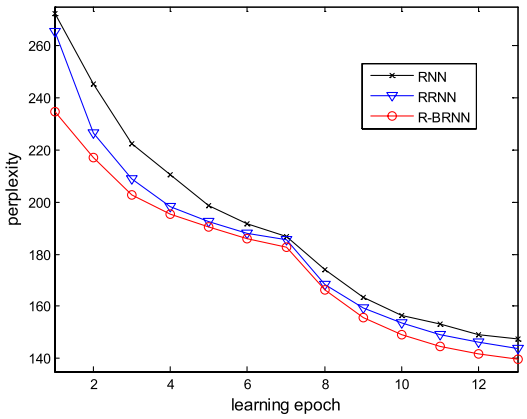

Fig. 8. Learning curves of perplexity using the RNN-LM, RRNN-LM, and R-BRNN-LM. BP algorithm is implemented.

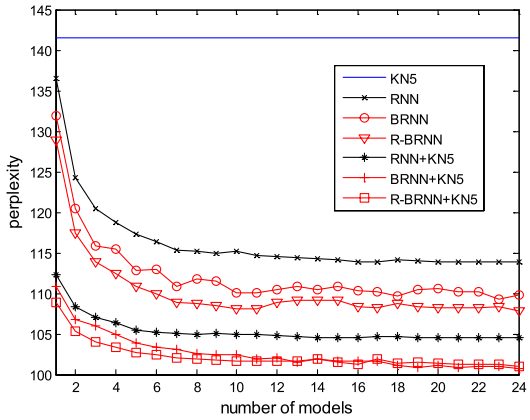

Fig. 9. Perplexity versus number of models in the linearly interpolated LM by using the randomly initialized weights in RNN, BRNN, and R-BRNN with $K = 300$. Individual model and hybrid model are compared. BP algorithm is implemented.

The more the hidden units are used, the faster the learning curve of α converges. For example, the hyperparameter for the case of $K = 300$ converges to the optimum value $\alpha_{\mathrm{ML2}} = 0.000106$ within very few learning epochs.

D. Comparison of Model Perplexity Using PT Corpus

Next, we compare the performance of different methods using the metric of perplexity. Fig. 8 shows the learning curve of perplexity measured from the test data of PT corpus. In this comparison, the number of hidden units $K = 300$ and the selection percentage $P = 0.005\%$ are fixed for the RNN-LM and the R-BRNN-LM. To access the effect of the estimated regularization parameter α_{ML2}, we further compare the perplexity of R-BRNN-LM with that of the regularized RNN (RRNN)-LM, where the regularization parameter $\hat{\alpha}$ in the RNN-LM is obtained from cross validation using the validation set based on the perplexity. We find that the R-BRNN-LM achieves lower perplexity than the RNN-LM and the RRNN-LM in different learning epochs even though a large number of hidden units $K = 300$ is adopted. The RRNN-LM performs better than the RNN-LM, while the R-BRNN-LM performs better than the RRNN-LM. It is because the R-BRNN-LM optimally estimates α_{ML2} and $\boldsymbol{\theta}_{\mathrm{MAP}}$ in each iteration through (30), while the RRNN-LM estimates the regularized maximum likelihood (RML) parameters $\boldsymbol{\theta}_{\mathrm{RML}}$ by using the fixed regularization parameter $\hat{\alpha}$ for different iterations. In this case, $\hat{\alpha} = 0.000550$ is selected. The R-BRNN-LM using $\boldsymbol{\theta}_{\mathrm{MAP}}$ could reduce the perplexity of test data when compared with the RRNN-LM using $\boldsymbol{\theta}_{\mathrm{RML}}$. Still, the additional validation data are required for the RRNN-LM.

Furthermore, we compare the perplexity and the training time (in minutes) of using different methods by changing the number of hidden units K. Table I reports these two metrics by using individual models and hybrid models with KN5. The BRNN with $P = 100\%$ and the R-BRNN with $P = 0.005\%$ are compared under the cases of $K = 50$, $K = 150$, and $K = 300$. We can see that the perplexity is improved by increasing K, but the computation cost is increased as well. Individual methods using RNN, RRNN, BRNN, and R-BRNN work better than KN5 in the case of $K = 300$ but worse than KN5 in the cases of $K = 50$ and $K = 150$. For the case of $K = 300$, the training time is significantly reduced from 140.9 minutes of using BRNN to 82.9 minutes of using R-BRNN. The speedup measure $1.70\times$ is achieved, and simultaneously the perplexity is reduced from 136.2 to 132.4. Rapid approximation results in reducing the redundancy in calculation of the Hessian matrix and achieving the global optimum in the estimated hyperparameter. A perplexity is reduced from 141.4 of using KN5 to 132.4 of using R-BRNN with $K = 300$. In this comparison, the RRNN is better than the RNN but worse than the R-BRNN in terms of perplexity. Computation overhead of using the RRNN is limited. Among these results, the lowest perplexity 110.7 is attained by applying the hybrid method R-BRNN + KN5 with $K = 300$. Linear interpolation of different LMs with KN5 performs much better than individual KN5, RNN, RRNN, BRNN, and R-BRNN. In what follows, two more evaluation tasks are performed.

1) Evaluation for Model Interpolation: To further improve system performance, we are motivated to train different LMs with random initialization of RNN parameters. We evaluate the perplexity of using different methods by conducting the model interpolation of LMs. Basically, the combination of multiple complimentary systems is beneficial to improve model regularization to boost the system performance. In the experiments, the combination of LMs is performed by using linear interpolation where the interpolation coefficients are selected from held-out data. Model interpolation is not only applied for RNN-LM but also for BRNN-LM and R-BRNN-LM. Individual models and hybrid models are implemented for comparison. In this comparison, $K = 300$ is considered, and the R-BRNN-LM with $P = 0.005\%$ is implemented. Fig. 9 shows the perplexity versus the number of randomly initialized LMs based on individual models using RNN, BRNN, and R-BRNN, and the hybrid models using RNN + KN5, BRNN + KN5, and R-BRNN + KN5. Perplexity is continuously decreased when the number of models in model interpolation is increased. In this comparison, we find the superiority of BRNN and R-BRNN to RNN and that of BRNN + KN5 and R-BRNN + KN5 to RNN + KN5

TABLE I

PERPLEXITIES AND TRAINING TIMES (IN MINUTES) UNDER DIFFERENT LMS AND NUMBERS OF HIDDEN UNITS K. INDIVIDUAL MODEL AND HYBRID MODEL ARE COMPARED. R-BRNN WITH $P = 0.005\%$ IS CONSIDERED. BP ALGORITHM IS IMPLEMENTED. PT CORPUS IS USED

Model	Time	Perplexity	
		individual	+ KN5
KN5	0.18	141.4	–
RNN ($K = 50$)	4.75	157.4	119.9
RRNN ($K = 50$)	4.97	155.4	119.2
BRNN ($K = 50$)	10.62	154.3	119.2
R-BRNN ($K = 50$)	6.90 (1.54x)	**150.8**	**118.7**
RNN ($K = 150$)	23.4	143.8	115.0
RRNN ($K = 150$)	24.8	141.1	114.0
BRNN ($K = 150$)	41.9	140.2	113.6
R-BRNN ($K = 150$)	27.7 (1.51x)	**136.8**	**112.7**
RNN ($K = 300$)	58.5	138.7	113.3
RRNN ($K = 300$)	65.6	137.1	112.8
BRNN ($K = 300$)	140.9	136.2	112.6
R-BRNN ($K = 300$)	82.9 (1.70x)	**132.4**	**110.7**

in terms of perplexity. The R-BRNN obtains lower perplexity than BRNN, while the perplexities of BRNN + KN5 and R-BRNN + KN5 are comparable. Compared with the lowest perplexity 110.7 of R-BRNN + KN5 with $M = 1$, model interpolation (i.e., R-BRNN + KN5 with $M > 1$) could further reduce model perplexity to 101.0. Model regularization using the Bayesian learning and model interpolation is complimentary and could be jointly performed to boost system performance. Basically, the experiments shown in Figs. 8 and 9 and Table I are conducted by using the BP algorithm. Hereafter, we investigate the effect of time steps back in time in the BPTT algorithm for RNN and BRNN models.

2) Evaluation for Backpropagation Through Time: In this set of experiment, we compare the perplexity of KN5 with that of individual models RNN, BRNN, and R-BRNN, where model interpolation is performed by linearly combing LMs under four conditions of the hidden units $K = 50$, $K = 150$, $K = 300$, and $K = 350$. Interpolation coefficients in this model interpolation with $M = 4$ are selected from the validation data. In particular, we evaluate the effect of time steps when applying BPTT for SGD learning of error BP in RNN-LM, BRNN-LM, and R-BRNN-LM with $P = 0.005\%$. Fig. 10 shows the perplexities of KN5, RNN, BRNN, and R-BRNN with different numbers of steps τ back in time in the BPTT algorithm. We find that the perplexities of RNN, BRNN, and R-BRNN are lower than that of KN5. BRNN-LM and R-BRNN-LM consistently achieve lower perplexity than RNN-LM for different values of τ. In this evaluation, the BPTT algorithm with five steps back in time attains the desirable performance. More time steps occasionally increase the model perplexity. The computation complexity is considerably increased by allowing too many steps back in time. Too many steps may cause too large error in BP procedure. For the case of $\tau = 5$, the perplexity of R-BRNN is obtained by 114.8, which is lower than 119.6 by using RNN and is comparable with 114.6 by using BRNN. These results are significantly better than 141.4 using KN5. When comparing these results with those in Table I in the case of

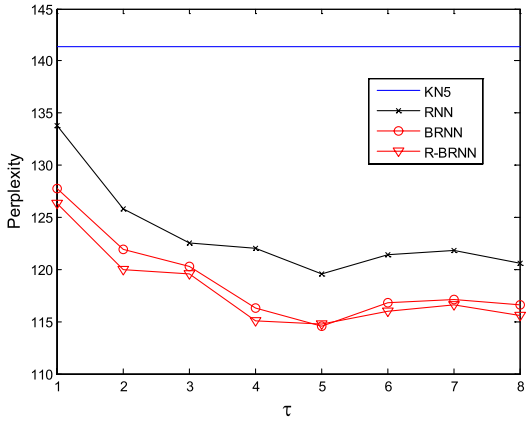

Fig. 10. Comparison of perplexity using KN5-LM, RNN-LM, BRNN-LM, and R-BRNN-LM, where the BPTT algorithm given different steps τ back in time is compared.

$K = 300$, it is found that the perplexity of RNN is reduced from 138.7 to 119.6, and that of R-BRNN is reduced from 132.4 to 114.8 by performing the BPTT algorithm and conducting the model interpolation of multiple LMs with different numbers of hidden units K. This confirms the superiority of the BPTT algorithm to the BP algorithm (or BPTT with $\tau = 1$) in optimization procedure and the robustness of RNN, BRNN, and R-BRNN with model interpolation.

In addition, we compare the perplexities of RNN and R-BRNN by using model interpolation based on randomly initialized weights $\boldsymbol{\theta}$ in Fig. 9 and different numbers of hidden units K in Fig. 10. Under the same case of $M = 4$ and BP or BPTT with $\tau = 1$, the perplexities of model interpolation using randomly initialized $\boldsymbol{\theta}$ (118.8 for RNN and 112.5 for R-BRNN) are lower than those using different numbers of hidden units K (133.8 for RNN and 126.3 for R-BRNN).

Notably, using the same RNN-LM, the perplexity 138.7 in Table I is obtained by using the BP algorithm, while the state-of-the-art perplexity 124.7 in [12] was obtained by applying the BPTT algorithm. Nevertheless, as shown in Fig. 10, the perplexity in our implementation 119.6, which is obtained by using RNN with BPTT ($\tau = 5$) and $M = 4$, is comparable and even lower than 124.7 in [12]. In the following experiments, we fix the BPTT algorithm [36] with five steps back in time for implementation of different individual and hybrid LMs using the other corpora.

E. Comparison of Model Perplexity Using Benchmark Corpus

In this set of experiment, we compare the perplexities of KN5, RNN, RRNN, BRNN, and R-BRNN under different selection percentages P by using the Benchmark corpus [13], which is known as one of the most challenging tasks in the evaluation of LMs. Model interpolation and hybrid modeling are not considered in this evaluation. Table II reports the computation time, the speedup, and the perplexity of using different LMs under different values of P, where the number of hidden units $K = 150$ is fixed. The speedup is defined as the ratio of training times of baseline BRNN and rapid

TABLE II

PERPLEXITIES AND TRAINING TIMES (IN MINUTES) UNDER DIFFERENT LMS AND SELECTION PERCENTAGES P. $K = 150$ IS FIXED. BENCHMARK CORPUS IS USED

Model	Time	Speed-up	Perplexity
KN5	1.87	–	223.6
RNN	1375.3	–	182.5
RRNN	1601.3	–	181.5
BRNN	3666.0	–	180.6
R-BRNN ($P = 80\%$)	3080.7	1.19x	181.1
R-BRNN ($P = 60\%$)	3055.0	1.20x	181.8
R-BRNN ($P = 30\%$)	2932.8	1.25x	180.2
R-BRNN ($P = 10\%$)	2820.0	1.30x	177.7
R-BRNN ($P = 1\%$)	2637.4	1.39x	177.4
R-BRNN ($P = 0.1\%$)	2618.6	1.40x	178.1
R-BRNN ($P = 0.01\%$)	2439.0	1.50x	**175.4**
R-BRNN ($P = 0.005\%$)	2437.5	1.50x	182.2
R-BRNN ($P = 0.0001\%$)	2436.4	1.50x	182.4

TABLE III

PERPLEXITIES AND TRAINING TIMES (IN MINUTES) UNDER DIFFERENT LMS AND SIZES OF TRAINING DATA. INDIVIDUAL MODEL AND HYBRID MODEL ARE COMPARED. $K = 150$ IS FIXED. R-BRNN WITH $P = 0.01\%$ IS CONSIDERED. BENCHMARK CORPUS IS USED

Model	Data Size	Time	Perplexity	
			individual	+ KN5
KN5	263K	0.10	326.9	–
RNN	263K	2.07	317.1	269.5
RRNN	263K	2.98	313.1	267.4
BRNN	263K	5.61	312.3	267.4
R-BRNN	263K	4.77 (1.18x)	**308.0**	**266.1**
KN5	1.3M	0.33	296.0	-
RNN	1.3M	22.2	269.9	224.1
RRNN	1.3M	24.4	266.1	223.8
BRNN	1.3M	36.4	264.0	223.6
R-BRNN	1.3M	28.9 (1.26x)	**257.0**	**222.8**
KN5	8M	1.87	223.6	-
RNN	8M	1375.3	182.5	159.6
RRNN	8M	1601.3	181.5	158.9
BRNN	8M	3666.5	180.6	158.2
R-BRNN	8M	2439.7 (1.50x)	**175.4**	**156.8**

BRNN under different values of P. In this comparison, RNN, RRNN, BRNN, and R-BRNN significantly improve the perplexity compared with KN5, while the training time of RNN methods is increased greatly compared with KN5. Also, BRNN with a rapid approximation to the Hessian matrix under different values of P is consistently better than RNN. The lowest perplexity 175.4 is achieved by using R-BRNN with $P = 0.01\%$. This result is lower than 180.6 by using BRNN. The compactness of the Hessian matrix with low P is helpful for Bayesian learning for RNN-LM. For the case of $P = 0.01\%$, the speedup $1.50\times$ is achieved by the R-BRNN compared with the BRNN. However, using too small percentage P in the R-BRNN may cause too limited information to reconstruct the compact Hessian matrix. BRNN and R-BRNN outperform RNN in terms of perplexity by using this Benchmark corpus. Such improvement is obtained for a wide range of P.

To examine the effect of the size of training data in RNN-LM and BRNN-LM, we further divide one partition of training set in Benchmark corpus with 8M words into two subsets that contain 263k words and 1.3M words. Table III compares the computation times and perplexities of individual LMs using KN5, RNN, RRNN, BRNN, and R-BRNN, and the hybrid LMs using RNN + KN5, RRNN + KN5, BRNN + KN5, and R-BRNN + KN5, where different sizes of training data T (263k, 1.3M, and 8M words) are investigated. The number of hidden units $K = 150$ is fixed. R-BRNN with $P = 0.01\%$ is considered. The speedup of R-BRNN relative to BRNN is shown. Training time is measured for individual LMs. In this comparison, RRNN, BRNN, and R-BRNN consistently perform better than RNN under different amounts of training words by using individual LMs as well as hybrid LMs. The computation time is increased a lot when the size of training data is increased from 1.3M to 8M. It is because the resulting number of vocabulary words is increased as well. We confirm the effectiveness of BRNN-LMs for the sparse training data as well as the abundant training data.

F. Comparison of Word Error Rate Using WSJ Corpus

FNN, cDC, RNN, RRNN, BRNN, and R-BRNN are carried out for the evaluation of the CSR by using WSJ corpus. In the experiments, different LMs are applied to rescore the 100-best candidates from speech recognition system which was implemented by using the Kaldi toolkit [37], which was written in C++ and licensed under the Apache License v2.0. The DNN acoustic modeling was implemented [38]. The baseline Gaussian mixture model and HMM (GMM-HMM) system was first trained to find the state alignments from all training utterances. Based on these alignments, DNN was trained by using the 40-D log Mel frequency bank features of training utterances. The GMM-HMM was trained by using 13-D Mel-frequency cepstral coefficients from each frame. Cepstral mean normalization was performed per speaker. The feature vector at each frame was spliced by those features from a context window of seven frames. We used the linear discriminant analysis (LDA) [39] to project from 91 (7×13) dimensions to 40-D and then applied the maximum likelihood linear transform [40] to make the features more accurately modeled by the diagonal covariance Gaussian. The GMM-HMM system was used to create the triphone-state labels by means of the forced-alignment during the initial training procedure of the DNN-HMM system. The leaves of a phonetic decision tree were inherited from the GMM-HMM system and corresponded to the DNN output targets. The DNN topology was formed by seven hidden layers, while each layer had 2048 neurons. The softmax output layer with 2035 units was used. In the DNN cross-entropy training, the stopping criterion for learning was based on the early stopping using development set. The minibatches with a size of 256 frames were used in the SGD training algorithm.

Table IV lists the results of perplexity, training time, and WER (%) on WSJ corpus by using individual LMs, including KN5, FNN, cDC, RNN, RRNN, BRNN, and R-BRNN, and the hybrid LMs, including FNN + KN5, cDC + KN5, RNN + KN5, RRNN + KN5, BRNN + KN5, and R-BRNN + KN5, where $K = 200$ and $P = 0.005\%$ were considered. The fourth gram-based FNN was trained with 100 neurons and 200 neurons specified in the first

TABLE IV
PERPLEXITIES, TRAINING TIMES (IN MINUTES), WERS (%), AND ERROR RATE REDUCTIONS (%) OVER BASELINE LM (GIVEN IN THE PARENTHESIS) UNDER DIFFERENT LMS AND VOCABULARY SIZES $|\mathcal{V}|$. INDIVIDUAL MODEL AND HYBRID MODEL ARE COMPARED. $K = 200$ IS FIXED. R-BRNN WITH $P = 0.005\%$ IS CONSIDERED. WSJ CORPUS IS USED

Model	Time	Perplexity	WER (%)	
		$\|\mathcal{V}\|$ = 20K	$\|\mathcal{V}\|$ = 5K	$\|\mathcal{V}\|$ = 20K
KN5	7.05	121.4	4.33 (–)	12.5 (–)
FNN	–	117.9	4.13	12.0
FNN + KN5	2325.9	102.1	3.58 (17.3)	11.3 (9.6)
cDC	–	115.3	3.92	11.7
cDC + KN5	2740.3	98.3	3.39 (21.7)	10.4 (16.8)
RNN	–	113.9	3.85	11.5
RNN + KN5	3150.3	95.9	3.36 (22.4)	10.2 (18.4)
RRNN	–	112.1	3.71	11.3
RRNN + KN5	3489.4	93.2	3.05 (29.6)	9.4 (24.8)
BRNN	–	103.2	3.57	11.1
BRNN + KN5	6220.2	88.7	2.98 (31.2)	9.0 (28.0)
R-BRNN	–	100.1	3.55	10.9
R-BRNN + KN5	4513.6	**87.2**	**2.96 (31.6)**	**8.9 (28.8)**

and the second hidden layers of an MLP, respectively. The cDC-LM with 200 Dirichlet classes was trained [3]. The perplexities under $|\mathcal{V}| = 20K$ and the WERs under $|\mathcal{V}| = 5K$ and $|\mathcal{V}| = 20K$ are evaluated for different LMs. The error rate reductions (%) of different hybrid LMs with respect to baseline KN5 are reported.

In this set of experiment, the perplexity and WER using KN5 are comparable with those in [41]. The individual LMs using FNN, cDC, RNN, RRNN, BRNN, and R-BRNN performs better than KN5 and could be further improved by combining with KN5. cDC obtains the perplexity and WER, which are lower than those of FNN but higher than those of RNN and RRNN. BRNN and R-BRNN achieve comparable performance in terms of perplexity as well as WER. For example, the perplexity is reduced from 121.4 using KN5 to 102.1 using FNN + KN5, 98.3 using cDC + KN5, 95.9 using RNN + KN5, 93.2 using RRNN + KN5, and 87.2 using R-BRNN + KN5, while the WER is reduced from 12.5% using KN5 to 11.3% using FNN + KN5, 10.4% using cDC + KN5, 10.2% using RNN + KN5, 9.4% using RRNN + KN5, and 8.9% using R-BRNN + KN5 in the case of 20k vocabulary words. The error rate reduction is improved from 18.4% using RNN + KN5 to 28.8% using R-BRNN + KN5. A similar improvement of using R-BRNN over RNN and RRNN is also obtained for the case of 5k vocabulary words. The results of R-BRNN + KN5 are slightly better than BRNN + KN5 in this comparison. Such results of BRNN and R-BRNN are competitive when compared with the state-of-the-art methods using FNN, cDC and RNN under the same experimental conditions. Again, the computation time is similarly increased by applying the Bayesian learning and the hybrid modeling.

VI. CONCLUSION

We have presented a Bayesian learning method for RNN-LM where a regularization term was introduced to conduct the penalized model training. A rapid approximation algorithm was proposed to derive a solution to BRNN-LM with a positive semidefinite Hessian matrix. BRNN-LM was established by alternatively estimating the RNN parameters based on the MAP criterion and calculating the regularization parameter by maximizing the marginal likelihood over uncertainty of RNN parameters. The computation of a high-dimensional Hessian matrix was reduced by selecting salient elements of the outer-products through a precalculation of combination weights. Removing the redundancy from those minor outer-products did improve the convergence and the system performance of BRNN-LM. Experimental results over three corpora showed the consistent improvement of BRNN-LM and R-BRNN-LM over RNN-LM in terms of perplexity and WER. The benefits of compensating for the model uncertainty from Bayesian treatment, reducing the redundancy from the outer-products and representing the variations of LMs from model interpolation were illustrated.

REFERENCES

[1] J.-T. Chien, "Association pattern language modeling," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 5, pp. 1719–1728, Sep. 2006.

[2] R. Kuhn and R. de Mori, "A cache-based natural language model for speech recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 6, pp. 570–583, Jun. 1990.

[3] J.-T. Chien and C.-H. Chueh, "Dirichlet class language models for speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 19, no. 3, pp. 482–495, Mar. 2011.

[4] P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer, "Class-based n-gram models of natural language," *Comput. Linguistics*, vol. 18, no. 4, pp. 467–479, Dec. 1992.

[5] R. Kneser and H. Ney, "Improved backing-off for M-gram language modeling," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Detroit, MI, USA, May 1995, pp. 181–184.

[6] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," *Comput. Speech Lang.*, vol. 13, no. 4, pp. 359–394, Oct. 1999.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Feb. 2003.

[8] A. Emami and F. Jelinek, "Exact training of a neural syntactic language model," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 1. Montreal, QC, Canada, May 2004, pp. I-245–I-248.

[9] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, Jul. 2007.

[10] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Deep neural network language models," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics-Human Lang. Technol. Workshop (NAACL-HLT)*, Montreal, QC, Canada, Jun. 2012, pp. 20–28.

[11] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Chiba, Japan, Sep. 2010, pp. 1045–1048.

[12] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. Černocký, "Empirical evaluation and combination of advanced language modeling techniques," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Florence, Italy, Aug. 2011, pp. 605–608.

[13] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," in *Proc. Annu. Conf. Int. Speech Commun. Assoc. (INTERSPEECH)*, Singapore, Sep. 2014, pp. 2635–2639.

[14] R. M. Neal, "Bayesian learning for neural networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 1995.

[15] D. J. C. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, May 1992.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, Feb. 2006.

[17] S. Watanabe and J.-T. Chien, *Bayesian Speech and Language Processing*. Cambridge, U.K.: Cambridge Univ. Press, Jul. 2015.

[18] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[19] G. Saon and J.-T. Chien, "Large-vocabulary continuous speech recognition systems: A look at some recent advances," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 18–33, Nov. 2012.
[20] H. Masataki, Y. Sagisaka, K. Hisaki, and T. Kawahara, "Task adaptation using MAP estimation in N-gram language modeling," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Munich, Germany, Apr. 1997, pp. 783–786.
[21] S. F. Chen, "Shrinking exponential language models," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics-Human Lang. Technol. Workshop (NAACL-HLT)*, Boulder, CO, USA, Jun. 2009, pp. 468–476.
[22] R. Rosenfeld, "A maximum entropy approach to adaptive statistical language modelling," *Comput. Speech Lang.*, vol. 10, no. 3, pp. 187–228, Jul. 1996.
[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.
[24] Y. W. Teh, "A hierarchical Bayesian language model based on Pitman–Yor processes," in *Proc. 21st Int. Conf. Comput. Linguistics, 44th Annu. Meeting Assoc. Comput. Linguistics*, Sydney, NSW, Australia, Jul. 2006, pp. 985–992.
[25] J. Pitman and M. Yor, "The two-parameter Poisson–Dirichlet distribution derived from a stable subordinator," *Ann. Probab.*, vol. 25, no. 2, pp. 855–900, Apr. 1997.
[26] J.-T. Chien, "Hierarchical Pitman–Yor–Dirichlet language model," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 8, pp. 1259–1272, Aug. 2015.
[27] D. T. Mirikitani and N. Nikolaev, "Recursive Bayesian recurrent neural networks for time-series modeling," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 262–274, Feb. 2010.
[28] S. Renals and D. J. C. MacKay, "Bayesian regularisation methods in a hybrid MLP–HMM system," in *Proc. 3rd Eur. Conf. Speech Commun. Technol. (EUROSPEECH)*, Berlin, Germany, Sep. 1993, pp. 1719–1722.
[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
[30] D. J. C. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 720–736, Sep. 1992.
[31] J.-T. Chien, Y.-C. Ku, and M.-Y. Huang, "Rapid Bayesian learning for recurrent neural network language model," in *Proc. 9th Int. Symp. Chin. Spoken Lang. Process. (ISCSLP)*, Singapore, Sep. 2014, pp. 34–38.
[32] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network language model," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, South Lake Tahoe, NV, USA, Dec. 2014, pp. 206–211.
[33] A. Stolcke, "SRILM—An extensible language modeling toolkit," in *Proc. Int. Conf. Spoken Lang. Process. (ICSLP)*, Denver, CO, USA, Sep. 2002, pp. 901–904.
[34] T. Mikolov, A. Deoras, S. Kombrink, L. Burget, and J. H. Černocký, "RNNLM—Recurrent neural network language modeling toolkit," in *Proc. IEEE Autom. Speech Recognit. Understand. Workshop (ASRU)*, Waikoloa, HI, USA, Dec. 2011, pp. 196–201.
[35] J.-T. Chien, "Laplace group sensing for acoustic models," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 5, pp. 909–922, May 2015.
[36] T. Mikolov, S. Kombrink, L. Burget, J. H. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5528–5531.
[37] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Autom. Speech Recognit. Understand. Workshop (ASRU)*, Waikoloa, HI, USA, Dec. 2011.
[38] J.-T. Chien and T.-W. Lu, "Tikhonov regularization for deep neural network acoustic modeling," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, South Lake Tahoe, NV, USA, Dec. 2014, pp. 147–152.
[39] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 2. Istanbul, Turkey, Jun. 2000, pp. 1129–1132.
[40] M. J. F. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 75–98, Apr. 1998.
[41] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Miami, FL, USA, Dec. 2012, pp. 234–239.

Jen-Tzung Chien (M'97–SM'04) received the Ph.D. degree in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 1997.

He was with National Cheng Kung University, Tainan, Taiwan, from 1997 to 2012. He was a Visiting Researcher with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, in 2010. Since 2012, he has been with the Department of Electrical and Computer Engineering and the Department of Computer Science, National Chiao Tung University, Hsinchu, where he is currently a University Chair Professor. He has authored extensively, including the book entitled *Bayesian Speech and Language Processing* (Cambridge University Press, 2015). His current research interests include machine learning, speech recognition, information retrieval, blind source separation, and face recognition.

Prof. Chien received the best paper award of the IEEE Automatic Speech Recognition and Understanding Workshop in 2011 and the Distinguished Research Award from the Ministry of Science and Technology, Taiwan, in 2006, 2010, and 2014. He served as the Associate Editor of the IEEE SIGNAL PROCESSING LETTERS from 2008 to 2011, the Guest Editor of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING in 2012, and the Tutorial Speaker of Interspeech in 2013 and the International Conference on Acoustics, Speech and Signal Processing in 2012 and 2015. He serves as an Elected Member of the IEEE Machine Learning for Signal Processing Technical Committee.

Yuan-Chu Ku received the B.S. degree in electrical and computer engineering from the National Taipei University of Technology, Taipei, Taiwan, in 2012, and the M.S. degree in electrical and computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2014.

His current research interests include machine learning, language modeling, Bayesian learning, and speech recognition.