RECURRENT NEURAL NETWORK LANGUAGE MODEL WITH STRUCTURED WORD EMBEDDINGS FOR SPEECH RECOGNITION

Tianxing He   *Xu Xiang*   *Yanmin Qian*   *Kai Yu*

Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering
SpeechLab, Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, China
{cloudygoose,chinoiserie,yanminqian,kai.yu}@sjtu.edu.cn

ABSTRACT

Due to effective word context encoding and long-term context preserving, recurrent neural network language model (RNNLM) has attracted great interest by showing better performance over back-off n-gram models and feed-forward neural network language models (FNNLM). However, it still has the difficulty of modelling words of very low frequency in training data. To address this issue, a new framework of structured word embedding is introduced to RNNLM, where both input and target word embeddings are factorized into weighted sum of the corresponding sub-word embeddings. The framework is instantiated for Chinese, where characters can be naturally used as the sub-word units. Experiments on a Chinese twitter LVCSR task showed that the proposed approach effectively outperformed the standard RNNLM, yielding a relative PPL improvement of 8.8% and an absolute 0.59% CER improvement in N-Best re-scoring.

***Index Terms*—** Recurrent Neural Network, Word Embeddings, Language Model, Speech Recognition

1. INTRODUCTION

Although the conventional back-off n-gram language model has been widely used in the automatic speech recognition (ASR) community for its simplicity and effectiveness, it has long suffered from the *curse-of-dimensionality* problem caused by huge number of possible word combinations in real text. Various smoothing techniques[1] are proposed to address this issue but the improvements have been limited. Recently, neural network based language models have attracted great interest due to its effective encoding of word context history [2, 3, 4, 5].In neural network based language models, both the word context and the target word are projected into a continuous space and the projection, represented by the transformation matrices in the neural network which are learned during training. The projected continuous word vectors are also referred to as *word embeddings*.With the word context representation, feed-forward neural network language models (FNNLM), including NNLM [2, 3, 4, 5], or log-bilinear LM [6], have achieved both better PPL and word error rate (WER) for ASR.

Despite the benefits of effective context modelling brought by word embeddings, FNNLM is still a short-span language model and not capable of utilizing long-term (e.g. context that is 5 or 6 words away) word history for the target word prediction. To address this issue, recurrent neural network language model (RNNLM), which introduces a recurrent connection in the hidden layer, is proposed to preserve long-term context. It has achieved the-state-of-art perplexity and word error rate (WER) performance on various data sets [7, 8, 9, 10, 11, 12], outperforming traditional back-off n-gram models and FNNLMs.

However, even with word context encoding and long term context preserving, the neural network model's prediction of rare words (i.e. OOV or words that only occur several times in the training data) is still poor due to data sparsity. To improve the performance of neural network based LM, a natural extension is to incorporate rich features, such as syntactic or morphological features, into the input layer of FNNLM or RNNLM [13, 14, 15]. The hope is that neural network can learn context representation via richer input information. Although overall performance can be improved with richer input features, rare word prediction improvement is still limited. The weights for the rare words on the output layer are still poorly trained since they always receive negative gradient during training. An alternative line of thought is to further work on embeddings of the target words on the output layer. In [16], factorised word embeddings are investigated within a log-bilinear LM framework, where both input and target word embeddings are represented as a summation of the surface word embedding and its corresponding morpheme embeddings. Although the structured word embedding idea in [16] provides a promising methodology, it has not been investigated within the state-of-the-art RNNLM framework and its performance on ASR has not been tested. Furthermore, using additive morpheme embeddings to form structured word embedding sometimes may lead to negative effects because linguistic correlation of a morpheme in different words can be ambiguous or even useless. Careful pre-processing and prior linguistic expert knowledge have to be used.

In this paper, a complete structured word embedding framework is investigated for RNNLM used in ASR. Here, both target word embeddings and input word embeddings in RNNLM are enhanced by sub-word embeddings using a linear combination . A word-dependent scaling factor is introduced for each word to reflect the importance of the additional sub-word embeddings. This offers more flexibility of the word embedding factorisation and allows the dependence of the sub-word embeddings to be learnt from data. The proposed RNNLM with structured word embedding is instantiated using a Chinese ASR task in this paper. This is because character is a natural and meaningful sub-word unit in Chinese, and the mapping between word and the corresponding characters is trivial. Note that the proposed approach be readily extended to alphabetical lan-

This work was supported by the Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning, the China NSFC project No.61222208, JiangSu NSF project No.201302060012 and the European Union Seventh Framework Program (No.247619)

guages.

The rest of the paper is organized as follows. In section 2, a detailed description of RNNLM with structured word embeddings is given. Experimental results and analysis are presented in section 3. Finally, section 4 concludes the paper and discusses the future work.

2. RNNLM WITH STRUCTURED WORD EMBEDDINGS

In this section, the architecture of standard RNNLM is reviewed, then the proposed RNNLM with structured word embeddings is described in detail.

2.1. Standard RNNLM

Language models aim to model the probability of the next word w_{t+1} based on word history (or context). It can be denoted as $P(w_{t+1}|w_1^t)$, where w_1^t represents the word sequence from time 1 to t. Short-span LMs, either n-gram or FNNLM, usually employ the Markov assumption that $P(w_{t+1}|w_1^t) \approx P(w_{t+1}|w_{t-n+1}^t)$. RNNLM, however, preserves long-term context by introducing "recurrent" connections which propagate the output of the hidden layer of the previous time instance to the current time instance, as depicted in figure 1.

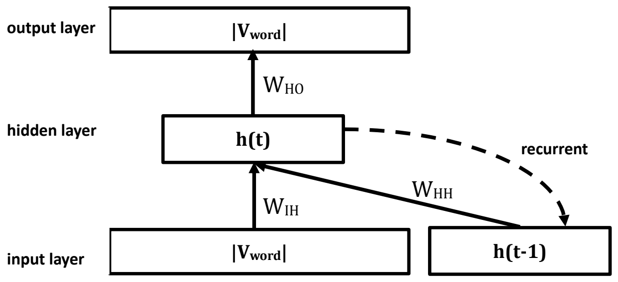

Fig. 1. Architecture of standard RNNLM

The input layer, denoted as $\mathbf{v}(t)$, encodes the current word w_t using 1-of-V coding. The activation of the hidden layer of time $t-1$, denoted as $\mathbf{h}(t-1)$, is propagated back to the input layer. The RNNLM output vector $\mathbf{y}(t)$ is the language model probability of all words in the vocabulary at time $t+1$ given the whole word sequence history up to time t, which is obtained using a *softmax* function. The propagation process of RNNLM can be formulated as

$$\mathbf{h}(t) = f(\mathbf{W}_{\mathtt{IH}}\mathbf{v}(t) + \mathbf{h}(t-1)) \quad (1)$$

$$\mathbf{u}(t) = \mathbf{W}_{\mathtt{HO}}\mathbf{h}(t) \quad (2)$$

$$P(w_{t+1} = k|w_1^t) = y_k(t) = \frac{e^{u_k(t)}}{\sum_{i \in V_{\mathtt{word}}} e^{u_i(t)}} \quad (3)$$

where activation function f is an element-wise *sigmoid* function, $\mathbf{u}(t)$ is a $|V_{\mathtt{word}}|$ dimensional vector denoting the input to the final softmax function. $\mathbf{W}_{\mathtt{IH}}$, $\mathbf{W}_{\mathtt{HH}}$, $\mathbf{W}_{\mathtt{HO}}$ denote the transformation matrices in the neural network, which form the parameter set of a standard RNNLM:

$$\Theta_{\mathtt{rnnlm}} = \{\mathbf{W}_{\mathtt{IH}}, \mathbf{W}_{\mathtt{HH}}, \mathbf{W}_{\mathtt{HO}}\}$$

Since each neuron in the input or the output layer represents a word, each column in $\mathbf{W}_{\mathtt{IH}}$ or each row in $\mathbf{W}_{\mathtt{HO}}$ can be viewed as a continuous vector representation of the corresponding word, i.e. the *word embedding*.

RNNLM can be trained using the "back propagation through time" (BPTT) algorithm[17]. Note that since 1-of-V coding is used in the input layer, most computational cost is induced by propagation on $\mathbf{W}_{\mathtt{HO}}$, which can be sped up by segmenting output layer in to word classes [18]:

$$P_{\mathtt{class}}(w_t|w_1^{t-1}) = P(w_t|c_t, w_1^{t-1})P(c_t|w_1^{t-1}) \quad (4)$$

Note that the output layer is first extended to incorporate neurons that represent the classes. During inference or training, only the fraction of $W_{\mathtt{HO}}$ related to the classes and words in the same class as the target word is propagated or updated, saving a lot of computation. Commonly used word classes are frequency-based classes. Class-based RNNLM is widely used because significant speed-up can be achieved with this technique at a cost of slight degradation of performance.

2.2. RNNLM with Structured Word Embeddings

The introduction of structured word embedding aims to give a better representation for rare words they can borrow well-trained shared sub-word embeddings from high-frequency words, with the assumption that the data sparsity issue for sub-word units is much less serious than words. In addition to input and target word embeddings, two set of sub-word embeddings are added to the model which are shared among the input and target words, respectively. The word embeddings are then enhanced by summing up the additional sub-word embeddings. As noted in section 1, not all words can be expressed well by its sub-word units, so a word-dependent scaling factor α is introduced for each word to endow the model more flexibility. The definition of *sub-word* may vary for different languages. In this paper, Chinese is used to instantiate the structured embedding framework and *character* is employed as the sub-word unit. It is worth noting that sub-word definition is independent of the framework proposed in this paper and hence, the RNNLM with structured word embedding can also be readily applied to other languages provided the mapping between word and sub-word can be well defined. Equation (5) shows both Chinese and English examples of structured word embeddings:

$$\begin{array}{llll} \textbf{Eng:} & \overrightarrow{decaf} & = \overrightarrow{decaf} & + \alpha_{decaf}(\overrightarrow{de} + \overrightarrow{caf}) \\ \textbf{Chn:} & \overrightarrow{你好} & = \underbrace{\overrightarrow{你好}}_{\mathtt{word-embd}} & + \alpha_{你好}\underbrace{(\overrightarrow{你} + \overrightarrow{好})}_{\mathtt{char-embd}} \end{array} \quad (5)$$

To incorporate character embeddings into the RNNLM model, two additional hidden layers are added into the neural network, as shown in figure 2. Apart from two sets of character embeddings, a sparse 0-1 mapping matrix($|V_{\mathtt{word}}|$ by $|V_{\mathtt{char}}|$) is constructed to map a word index into its corresponding character indexes. The matrix is a *constant* matrix as there is deterministic and trivial mapping between Chinese words and characters. When used for other languages, the construction of the mapping matrix may need linguistic knowledge and can be probabilistic. The input and the output weight matrices with structured word embeddings can be written as

$$\widehat{\mathbf{W}}_{\mathtt{HO}} = \mathbf{W}_{\mathtt{HO}} + \mathtt{diag}(\boldsymbol{\alpha}_{\mathtt{o}})\mathbf{W}_{\mathtt{map}}\mathbf{W}_{\mathtt{OC}}$$

$$\widehat{\mathbf{W}}_{\mathtt{IH}} = \mathbf{W}_{\mathtt{IH}} + \mathbf{W}_{\mathtt{IC}}\mathbf{W}_{\mathtt{map}}^{\top}\mathtt{diag}(\boldsymbol{\alpha}_{\mathtt{i}})$$

where $\mathbf{W}_{\mathtt{map}}$ is the $|V_{\mathtt{word}}| \times |V_{\mathtt{char}}|$ constant mapping matrix, $\mathbf{W}_{\mathtt{IH}}$ and $\mathbf{W}_{\mathtt{HO}}$ are the input and output word embedding matrices, $\mathbf{W}_{\mathtt{IC}}$ and $\mathbf{W}_{\mathtt{TC}}$ are the input and target character embedding matrices, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_o$, which are vectors of length $|V_{\mathtt{word}}|$, denote the word-dependent scaling factors for input word embeddings and target word embeddings, respectively, $\mathtt{diag}(\boldsymbol{\alpha})$ means a $|V_{\mathtt{word}}| \times |V_{\mathtt{word}}|$ matrix whose diagonal is $\boldsymbol{\alpha}$.

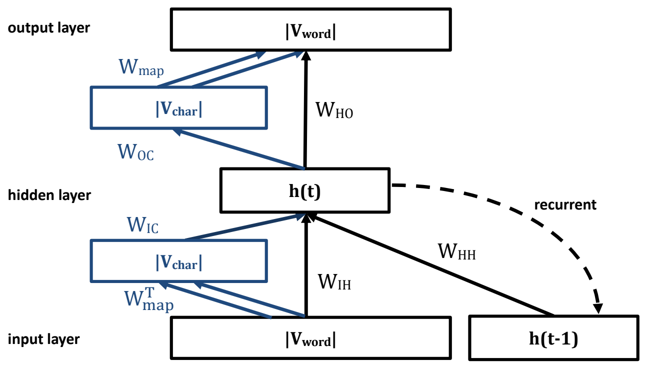

Fig. 2. Architecture RNNLM with structured word embeddings.

RNNLM with structured word embedding can be obtained by replacing $\mathbf{W}_{\mathtt{IH}}$ and $\mathbf{W}_{\mathtt{HO}}$ in equation (1) and (2) with $\widehat{\mathbf{W}}_{\mathtt{IH}}$ and $\widehat{\mathbf{W}}_{\mathtt{HO}}$. The proposed model, denoted as `rnnlm_se`, has free parameter set:

$$\Theta_{\mathtt{rnnlm\_se}} = \{\mathbf{W}_{\mathtt{IH}}, \mathbf{W}_{\mathtt{HH}}, \mathbf{W}_{\mathtt{HO}}, \mathbf{W}_{\mathtt{IC}}, \boldsymbol{\alpha}_i, \mathbf{W}_{\mathtt{OC}}, \boldsymbol{\alpha}_o\}$$

Note that structured input and target word embeddings can be either or both applied(e.g. Input word embeddings can be left unchanged by forcing $\mathbf{W}_{\mathtt{IC}} = \mathbf{0}$), thus bringing variations of `rnnlm_se`, which will be closely studied in section 3. The proposed model can be trained using standard BPTT algorithm, with slight modification that errors are back-propagated through $\mathbf{W}_{\mathtt{OC}}$ and $\mathbf{W}_{\mathtt{HO}}$ and aggregated on the hidden layer. In all our experiments, all weights are randomly initialized and $\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_o$ are initialized to 1.

3. EXPERIMENTS

RNNLM with structured word embeddings was evaluated on a Chinese twitter LVCSR task. Unlike alphabet based languages like English, Chinese is a character based language, so no word factorisation is needed. The training data consisted of 780k sentences and 4m tokens(tokenization is done by Chinese word segmentation). The cross-validation set for RNNLM training consisted of 25k sentences and 157k tokens. A word vocabulary of size 50k(including single-character words) whose OOV rate on test data is 1.8%, and a character set of size 6k which covered all words in the vocabulary are used. OOV words are neglected when calculating perplexity. The test dataset consists of 6-hour transcribed smart-phone audio data (6k utterances). It was used for both PPL and character error rate (CER) evaluation. For fair comparison, in all RNNLM training the network was unfolded 3 times(bptt=3).

The acoustic model was a context-dependent DNN-HMM[19] model with 6021 tied triphone states, trained on 600 hours of audio data collected from smart-phones. The acoustic feature used was 13-dimensional PLP with its first and second derivatives. Altogether 11 frames, 5 on the right and 5 on the left, were used as the input to DNN. An RBM initialized DNN with 6 hidden layer and 2048 nodes per layer was then trained using stochastic gradient descent. Back-off N-gram model with Kneser-Ney smoothing is used as language model for generating lattice, from which N-best lists are generated. N-best re-scoring was used to evaluate the ASR performance. A LM scaling factor of 11 was used for all experiments and 100-best lists were used for re-scoring with different language models. The oracle CER of the 100-best lists is 23.57%.

3.1. LM and ASR Performance Evaluation

In the experiments, back-off n-gram models and standard RNNLM and class-based RNNLM were used as baselines, on which are structured word embeddings are then applied. As note in section 2.2, several variations of `rnnlm_se` exist and it is worth investigating the separate or combined effect of applying structured input embeddings or structured target embeddings. Evaluation results are shown in table 1 and table 2 for RNNLM and class-based RNNLM, respectively, where "SI", "SO", "$\boldsymbol{\alpha}$" stand for structured input word embeddings, structured target word embeddings and word-dependent scaling factors, respectively.

LM	Structured Embedding			#Hidden Units	PPL	CER(%)
	SI	SO	$\boldsymbol{\alpha}$			
3-gram	N/A			N/A	451.5	35.32
4-gram	N/A			N/A	448.5	35.30
RNN	—	—	—	200	410.5	33.63
	—	—	—	300	407.9	33.50
RNN + SE	√	—	—	200	408.7	33.56
	—	√	—		405.0	33.35
	√	√	—		387.1	33.08
	√	√	√		**374.2**	**33.04**

Table 1. Evaluation results of standard RNNLM with structured word embeddings

The hidden layer size of the baseline RNNLM is set to 200 because using larger hidden layer did not give further performance improvement. It is shown that RNNLM obtained significant improvement over the back-off n-gram models. RNNLM get a significant PPL(relative 8.4%) and CER (absolute 1.7%) improvement over the back-off n-gram LM. The proposed model, `rnnlm_se` has a relative PPL improvement of 8.8% and an absolute 0.59% CER improvement comparing to the baseline RNNLM. Also note that small performance improvement can be achieved by applying either structured input or target word embeddings.

It is also interesting to investigate the performance of structured word embeddings on class-based RNNLM in which target word embeddings are trained separately for each class but character embeddings are shared across classes. In the experiment, all words were clustered to 30 classes according to their frequency in the training data. The results are shown in table 2. Comparing to the baseline class-based RNNLM whose hidden layer size is 300, `rnnlm_se` has a relative PPL improvement of 9.1% and an absolute CER improvement 0.5%.

LM	Structured Embedding			#Hidden Units	PPL	CER(%)
	SI	SO	$\boldsymbol{\alpha}$			
Class RNN	—	—	—	300	425.1	33.80
	—	—	—	400	420.2	33.62
Class RNN +SE	√	—	—	300	409.4	33.58
	—	√	—		413.3	33.60
	√	√	—		392.5	33.32
	√	√	√		**386.1**	**33.30**

Table 2. Evaluation results of class-based RNNLM with structured word embedding(class number is 30)

Finally, evaluation results of interpolated LM are shown in table 3, where standard non-class RNNLM and the proposed `rnnlm_se` is interpolated with the 4-gram LM and interpolation weight for `rnnlm` and `rnnlm_se` is set to 0.6. It is shown that structured word embeddings give a relative PPL improvement of 5.4% and an absolute CER improvement of 0.45%.

LM	PPL	CER%
`rnnlm` + 4-gram	342.3	33.47
`rnnlm_se` + 4-gram	**323.7**	**33.02**

Table 3. Evaluation results of interpolated LM

In this section, structured word embedding is applied on both standard and class-based RNNLM and both PPL and CER improvement are observed. It is also shown that either structured input or target word vectors can improve RNNLM's performance and their effects are complementary.

3.2. Discussion and analysis

To investigate on what kind of tokens RNNLM can get most gain from structured word embeddings, we divided the test text tokens in to three categories : *low-frequency, medium-frequency* and *high-frequency* tokens. Here *frequency* means the number of occurrences of the token in the training data, and the thresholds are carefully tuned so that the three categories' ratio in the test text is $1:1:1$. We then calculated PPL for the three categories separately, the results are shown in figure 3. It is shown that structured word embeddings deliver a very significant gain for low-frequency tokens, but doesn't help the model predict high-frequency tokens. Further, on the case of rare tokens, factoring target word embeddings gives much better gain than factoring input word embeddings. The reason could be that having better context representation doesn't *directly* help predict a rare word since the target word's representation is still poorly trained.

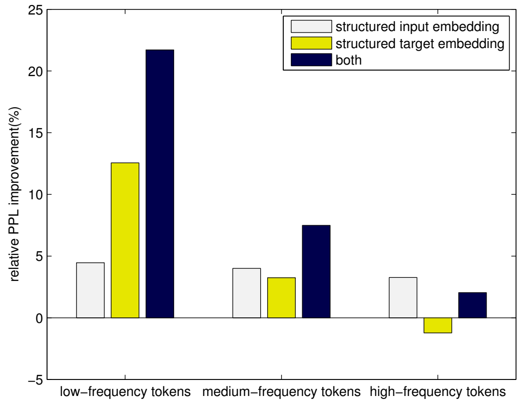

Fig. 3. PPL improvement of variations of `rnnlm_se` compared to the baseline class-based RNNLM on test data tokens of different frequencies.

Although adding word-dependent scaling factor did not give further significant performance improvement, it reflects whether character embeddings actually help modelling a specific word. In figure 4 we show α for target words of different frequencies. All word are sorted by their frequencies first, then we average their α's for each bin. It is observed that high-frequency words tend to have a small α since they are covered by sufficient data, and low-frequency words show more dependence on character embeddings. It would also be interesting to check words whose α are relatively high or low , in table 4 some examples are shown, most examples are chosen to have moderate frequency(between 20 and 100 counts in training data). Some phenomenon can be observed : high-frequency words, proper nouns, terminologies, words contain ambiguous characters and some compositionally complicated words have a low α, and words with high α usually consist of unambiguous characters or characters that have similar meanings. We conclude that the proposed word-dependent scaling factor gives the model more flexibility, but still, *additive* word representation has its limitations and sometimes fails to ”compose” a proper representation out of character embeddings.

$\alpha_o > 1.5$	国门,瞎话,徘徊,坠毁,身躯,搏斗,富强,尘土
$\alpha_o < 0.5$	一个,一言不发,好色,首长,王力宏,省略号

Table 4. Examples of words who have a low or high α in `rnnlm_se`

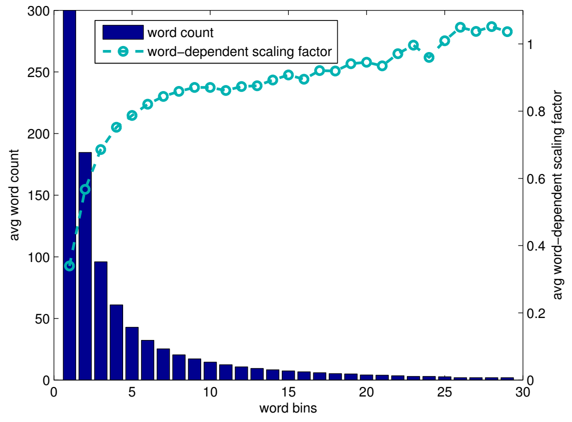

Fig. 4. Word-dependent scaling factor for words of different frequencies in `rnnlm_se`.

4. CONCLUSION AND FUTURE WORK

In this work, both target word embeddings and input word embeddings in RNNLM are enhanced by additive character embeddings, and word-dependent scaling factor is introduced to endow the model more flexibility. The character embeddings suffer less data sparsity problem and are shared among the words. The evaluation is done on a Chinese twitter data set and the proposed model got a relative PPL improvement of 8.8% and an absolute 0.59% CER improvement comparing to the baseline RNNLM. Further gain can be achieved when interpolated with 4-gram LM. It has also been shown that structured word embeddings greatly help the model give a better representation for rare words.

For future work, adopting additive sub-word embeddings has its limitations : it can not deal with ambiguous sub-word units and sometimes fail to compose a proper representation of a word out of its sub-word embeddings. Therefore, a more sophisticated way of factoring word embeddings is worth investigating.

5. REFERENCES

[1] Stanley F. Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," in *Proc. ACL*. 1996, pp. 310–318, Association for Computational Linguistics.

[2] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin, "A neural probabilistic language model," *Journal OF Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[3] Holger Schwenk, "Continuous space language models," *Computer Speech Language*, vol. 21, no. 3, pp. 492–518, 2007.

[4] Frederic Morin and Yoshua Bengio, "Hierarchical probabilistic neural network language model," in *AISTATS*, 2005, pp. 246–252.

[5] J. Park, X. Liu, M. J. F. Gales, and P. C. Woodland, "Improved neural network based language modelling andadaptation," in *Proc. InterSpeech*, 2010.

[6] Andriy Mnih and Geoffrey Hinton, "Three new graphical models for statistical language modelling," in *Proc. ICML*, 2007, pp. 641–648.

[7] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proc. InterSpeech*, 2010.

[8] Martin Sundermeyer, Ilya Oparin, Ben Freiberg, Ralf Schlüter, and Hermann Ney, "Comparison of feedforward and recurrent neural network language models," in *Proc. ICASSP*, 2013.

[9] Martin Sundermeyer, Ralf Schluter, and Hermann Ney, "Lstm neural networks for language modeling," in *Proc. InterSpeech*, 2012.

[10] Zhiheng Huang, Geoffrey Zweig, and Benoit Dumoulin, "Cache based recurrent neural network language model inference for first pass speech recognition," in *Proc. ICASSP*, 2014.

[11] X. Liu, Y. Wang, X. Chen, M. J. F. Gales, and P. C. Woodland, "Efficient lattice rescoring using recurrent neural network language models," in *Proc. ICASSP*, 2014.

[12] X. Chen, Y. Wang, X. Liu, M.J.F. Gales, and P. C. Woodland, "Efficient gpu-based training of recurrent neural network language models using spliced sentence bunch," in *Proc. InterSpeech*, 2014.

[13] Andrei Alexandrescu and Katrin Kirchhoff, "Factored neural language models," in *Proc. ACL*, 2006.

[14] Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau, "Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic," in *Proc. ICASSP*, 2013.

[15] Yangyang Shi, Pascal Wiggers, Catholijn, and M. Jonker, "Towards recurrent neural networks language models with linguistic and contextual features," in *Proc. InterSpeech*, 2012.

[16] Jan Botha and Phil Blunsom, "Compositional morphology for word representations and language modelling," in *Proc. ICML*, 2014.

[17] D.E. Rumelhart, G.E. Hintont, and R.J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[18] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur, "Extensions of recurrent neural network language model.," in *Proc. ICASSP*, 2011.

[19] George E. Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large vocabulary speech recognition," in *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.