Interspeech 2018
2-6 September 2018, Hyderabad

Character-level Language Modeling with Gated Hierarchical Recurrent Neural Networks

Iksoo Choi, Jinhwan Park, Wonyong Sung

Department of Electrical and Computer Engineering
Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul, 08826 Korea
akacis@snu.ac.kr, bnoo@snu.ac.kr, wysung@snu.ac.kr

Abstract

Recurrent neural network (RNN)-based language models are widely used for speech recognition and translation applications. We propose a gated hierarchical recurrent neural network (GHRNN) and apply it to the character-level language modeling. GHRNN consists of multiple RNN units that operate with different time scales, and the frequency of operation at each unit is controlled by the learned gates from training data. In our model, GHRNN learns the hierarchical structure of character, sub-word, and word. Timing gates are included in the hierarchical connections to control the operating frequency of these units. The performance was measured for Penn Treebank and Wikitext-2 datasets. Experimental results showed lower bit per character (BPC) when compared to simply layered or skip-connected RNN models. Also, when a continuous cache model is added, the BPC of 1.192 is recorded, which is comparable to the state of the art result.

Index Terms: recurrent neural network, language model, hierarchical structure

1. Introduction

Language models (LMs) estimate the probability distribution over sequence of words or characters, and they are essential for speech recognition, machine translation, and text generation [1, 2, 3]. LMs can be classified into character-level LM (CLM) and word-level LM (WLM) according to the token set of the input and output. LM estimates the probability of the token x appearing at time t, $P(x)$, or $\prod P(x_t \mid x_{<t})$, which means the output depends on the previous input tokens. Recurrent neural networks (RNNs), which are known to be suitable for handling sequences, achieve very good results for LM benchmarks [4, 5, 6]. Among RNNs, long short-term memory (LSTM)[7] and gated recurrent unit (GRU) models [8] are widely used because of their overall performance and trainability [9].

An RNN-based LM consists of an embedding layer, recurrent layers, and a softmax layer. The embedding layer is a look-up table that converts an input token to a fixed size vectors. The RNN part uses the fixed size input vector and its own past information to predict the next token. The softmax layer is composed of a linear layer and a softmax function [10].

In the copy task experiments [7, 11], the RNN structure was shown to memorize the past information very well. However, it is not known whether the RNN is capable of preserving long-term information and estimating the probability distribution at the same time. Many studies have been conducted to maintain long-term information in RNN by employing the hierarchical structure. A hierarchical structure that operates in multiscale is proposed in [12]. The hierarchy is organized in such a way that the upper layer maintains the past information of the lower layer while the lower layer generates new information. Some studies have attempted to preserve information through a temporal hierarchy formed by linking distant inputs or states [13]. A CLM that uses the word boundary as a clock signal for better utilizing the word level information was proposed in [14]. These studies indicate that the information retention capacity of RNN is limited and also show that the performance of an RNN can be improved through structural modification.

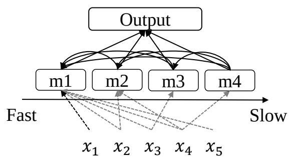
In this paper, we propose gated hierarchical recurrent neural networks (GHRNNs) for keeping long-term information. A GHRNN implementation for CLM consists of four LSTMs. Three of them are hierarchically connected for preserving long-term contexts, while the remaining one utilizes the output of these three LSTMs for prediction. We place a gate between hierarchically connected LSTMs, and the gate controls the change of information in the LSTMs. Depending on the gate value, the LSTM either preserves previous information, accepts new information, or mixes those two. The hierarchical structure and gate allow GHRNN maintain long-term information, and GHRNN shows performance gain utilizing that information. We evaluate our model on CLM task,where Penn Treebank dataset [15] and Wikitext-2 dataset [16] are used for evaluation.

This paper is organized as follows. We revisit previous works on hierarchical RNNs in Section 2. The proposed models are described in Section 3. The experimental results are shown in Section 4, and the concluding remarks are given in Section 5.

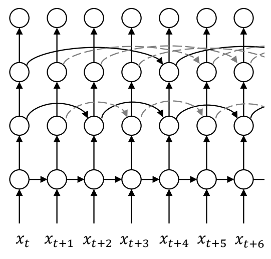
2. Hierarchical RNN based LMs

We describe seven hierarchical structure based LMs, which are clockwork RNN (CW-RNN), dilated RNN, hierarchical character-level LM (HCLM), hierarchical LSTM (HLSTM), hierarchical multiscale RNN (HM-RNN), bi-scale RNN (bi-RNN), and fast-slow RNN (FS-RNN) models. [17, 13, 18, 14, 12, 19, 6]. These hierarchical models can be classified into three groups according to their clock generation scheme such as fixed time steps, explicit steps, and automatic steps. Dilated RNN and CW-RNN form a hierarchy using fixed-rate clocks. HCLM and HLSTM use explicit boundary information, such as characters and words, to make each layer behave differently. On the other hand, HM-RNN and bi-RNN employ neither the fixed time step nor the specific boundary information. Instead, clocks are automatically generated according to the current input and internal state.

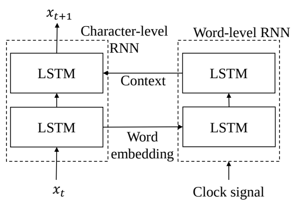
CW-RNN divides the hidden state of an RNN layer into several modules and assigns a different time scale to each module [17]. As shown in Fig. 1a, each module can receive an information from other modules only if its time scale is faster than the



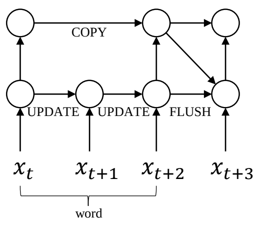
(a) *CW-RNN*


x_t x_{t+1} x_{t+2} x_{t+3} x_{t+4} x_{t+5} x_{t+6}

(b) *Dilated RNN*



(c) *HLSTM*



(d) *HM-RNN*

Figure 1: *The hierarchical structure of (a) CW-RNN, (b) Dilated RNN, (c) HLSTM, and (d) HM-RNN.*

other ones. The i-th module is updated at every 2^i-th time step.

Dilated RNN, shown in Fig. 1b, preserves long-term information through a temporal hierarchy [13]. Each layer of the dilated RNN is connected to a state that is temporally distant. The temporal distance of the higher layer is farther than that of the lower ones.

HLSTM, shown in Fig. 1c, consists of character-level and word-level RNNs, and the former operates with the character clock while the latter does with the explicit word clock [14]. The input and output are character tokens, thus this model is a CLM supplemented with WLM for long term context prediction. This model shows quite high performance when compared with a basic RNN based CLM, and has no out of vocabulary problem. Both character and word clocks are needed. HCLM works with word-level information, but employs an input CLM for word embedding generation from character input, and an output CLM for character output generation from word-level probability distribution [18].

HM-RNN, shown in Fig. 1d, introduces a 'boundary detector' that allows the model to automatically find the hierarchical structure of the sequence [12]. Each layer of HM-RNN conducts one of UPDATE, COPY, and FLUSH operations depending on the boundary detector value. The upper layer of the HM-RNN generates a sequence boundary whose period is longer than that of the lower layer. Especially, as a CLM, it shows character-, word- and n-gram- level hierarchy.

Bi-RNN has a hierarchical structure that operates according to the gate value [19]. The bi-RNN layer employs two gates that generate the hierarchical output. The gate values modify the hidden states of bi-RNN and create two different states connected to the next time step.

FS-RNN contains fast and slow layers, which forms two-level hierarchy [6]. The fast layer operates more than once per time step and the slow layer operates only once at each time step. Only the slow layer is connected to the past information, and the fast layer receives information from the slow layer and predicts the next character.

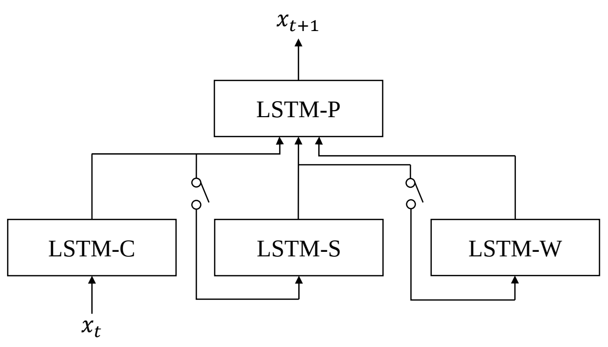


Figure 2: *GHRNN structure. This example shows gate values are 0. Therefore, LSTM-S and LSTM-W are inactivated*

3. Model

Our proposed model uses LSTM as a basic building block. The LSTM operates according to the following formula.

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\ f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\ o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\ \tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1} + b^c) \\ c_t &= i_t \odot \tilde{c}_t + f_t \odot c_{t-1} \\ h_t &= o_t \odot \tanh(c_t), \end{aligned} \tag{1}$$

where $W^{(\cdot)}$ and $U^{(\cdot)}$ are the weight matrices, $b^{(\cdot)}$ is the bias vector, i_t, f_t and o_t are the input gate, forget gate and output gate at time t, respectively. $\sigma(\cdot)$ is the sigmoid function and $\odot$ is the element-wise multiplication operator. c_t is the cell state and h_t is the hidden state. These two states are delivered to the next time step.

3.1. Gated hierarchical recurrent neural network (GHRNN)

In this section, we explain the basic structure of GHRNN and its operation. The GHRNN, shown in Fig. 2, consists of four LSTMs, which are LSTM-C, LSTM-S, LSTM-W, and LSTM-P. The LSTM-C receives the character embedding vector, and the output is connected to the LSTM-S. The output of LSTM-S is applied to LSTM-W. These three LSTM units can be interpreted as encoding character-level, sub-word-level, and word-level information. LSTM-P receives the concatenated vector of these three LSTM outputs and estimates the probability distribution of the next character. LSTM-C, LSTM-S, and LSTM-W generate hierarchical information by operating LSTM-S and LSTM-W at different clock rates. The clock rates are determined by the gates. Each LSTM of GHRNN operates according to Eq. 1. But the cell and hidden states of LSTM-S and LSTM-W are changed according to the gates as follows.

$$\begin{aligned} c_t &= g_t(i_t \odot \tilde{c}_t + f_t \odot c_{t-1}) + (1-g_t)c_{t-1} \\ h_t &= g_t(o_t \odot \tanh(c_t)) + (1-g_t)h_{t-1}, \end{aligned} \tag{2}$$

where g_t is the scalar gate value. If g_t is 0, the LSTM will retain the value of the previous time step, $t-1$. This deactivation or time skipping also gives a computational advantage. The gate value, g_t, is calculated as follows.

$$g_t = \text{Hard}\tanh_0^1(W^g v_t + b^g)$$
$$\text{Hard}\tanh_0^1(z) = \begin{cases} 0 : z \le 0 \\ z : 0 < z < 1 \\ 1 : z \ge 1 \end{cases} \quad (3)$$

Hard $\tanh_0^1$ is a hard tangent hyperbolic function that has minimum value of 0. v_t is a concatenated vector of two vectors; one is the output vector of lower hierarchy and the other is the hidden state of the previous time step. In the case of LSTM-S, v_t is a concatenated vector of h_t of LSTM-C and $h_{t-1}$of LSTM-S. For LSTM-W, v_t is the concatenation of h_t of LSTM-S and h_{t-1} of LSTM-W.

3.2. binary-gated hierarchical recurrent neural network (bGHRNN)

In the previous section, we define the gate value as a continuous value. If we constrain the gate value to be a binary number, the gates behaves like clock signals. We also propose a binary-GHRNN (bGHRNN) that can reduce computational cost more than GHRNN. GHRNN is deactivated only when the gate value is 0 but bGHRNN skips the operation when the gate value is less than 0.5. Therefore, the operating frequencies of LSTM-S and LSTM-W are further reduced. A binary-gate value, clock, is decided as follows.

$$\text{clock}_t = \text{binarize}(g_t)$$
$$\text{binarize}(a) = \begin{cases} 0 \text{ if } a < 0.5 \\ 1 \text{ if } a \ge 0.5 \end{cases} \quad (4)$$

Although the binary-gate can reduce the computational cost of LSTM, it can not mix past and current information because the gate value is a binary number. This reduces the ability to maintain a long-term context.

3.3. Training with boundary hints

When training GHRNN, we add the following boundary loss for the first ten epochs to help GHRNN to learn the sub-word and word boundaries.

$$\text{loss}_{\text{boundary}} = \frac{1}{T}\sum_{t=0}^{T}\{(g_{s,t} - B_{s,t})^2 + (g_{w,t} - B_{w,t})^2\}, \quad (5)$$

where $g_{s,t}$ and $g_{w,t}$ are the gate value of LSTM-S and LSTM-W at time t, respectively, and $B_{s,t}$ and $B_{w,t}$ are the boundary values of sub-word and word at time t. We employ byte-pair-encoding (BPE) as the sub-word unit. BPE is an encoding method that reduces the length of data by converting the most frequently appearing two adjacent code to one new code [20]. This new code works as a sub-word [21]. The sub-word boundary value is 1 at the last character of BPE code and 0 elsewhere. The word-boundary value is 1 at every <*space*> token and 0 for the remainder. With this boundary loss, GHLSTM learns the gate operation during the first ten epochs and fine-tunes it during the remaining training epochs. We add the boundary loss because the model rarely learns the boundaries in the rest of the training if it cannot find them in the initial training phase. The effect of adding boundary loss during the first a few epochs is maintained until the end of training.

4. Experiments

The proposed GHRNN based CLMs are evaluated with two text datasets: Penn Treebank dataset (PTB)[15] and Wikitext-2 dataset (Wiki2) [16]. We train the models using Adam with the default learning rate of 0.002 [25]. We use the batch size of 256 and the sequence length 128. We clip the norm of the gradient to 0.025 [26]. The models are regularized with the dropout probability of 0.5 [27].

We use bit-per-character (BPC) as an evaluation metric, which is defined as follows.

$$\text{BPC} = -\frac{1}{L}\sum_{t=1}^{L}\log_2 P(c_t \mid c_{<t}), \quad (6)$$

where L is the length of the test set and $c_{<t} = \{c_1, c_2, \ldots, c_{t-1}\}$

4.1. Penn Treebank dataset

We follow the common data preprocessing method suggested in [28]. The PTB dataset consists of 50 types of characters and the train, valid, and test sets contain 5.1M, 0.4M, and 0.4M characters, respectively. Table 1 shows BPCs of GHRNN, other hierarchical models, and notable models conducted on the test set. The BPCs of the models from CW-RNN to FS-RNN are based on the performance reported in each reference and the BPCs of the remaining models are measured in this experiment. '5-layer stacked LSTM' is the baseline model, which has a similar amount of parameter size with simply stacking LSTM layers. The 'No gate HRNN' has the same basic structure as GHRNN but does not have gates between hierarchical connections. Thus we can evaluate the performance effect of these gates by comparing GHRNN with this model. GHRNN consists of an embedding layer of size 512, four LSTMs with 512 units in the hidden layers, and a softmax layer of size 50. GHRNNs marked with '256' are the models whose sequence length is set to 256. 'BH' marks mean that the models are trained with the boundary hint as mentioned in Section 3.3. When we apply a continuous cache [29] on the best performing GHRNN, the model shows the BPC of 1.192 which is the comparable result with the state of the art FS-RNN model.

Table 1: *CLM results on Penn Treebank dataset*

Model	BPC	Parameter size
CW-RNN[17]	1.46	–
Zone out[22]	1.27	–
HyperLSTM[23]	1.27	4.9M
HCLM with cache[18]	1.247	–
LayerNorm HM-LSTM[12]	1.24	–
LayerNorm HyperLSTM[23]	1.23	5.1M
HyperRHN[24]	1.195	15.5M
FS-RNN[6]	**1.190**	7.2M
5-layer stacked LSTM	1.303	10.5M
No gate HRNN	1.247	10.5M
GHRNN	1.225	10.5M
GHRNN(seq256)	1.209	10.5M
GHRNN(seq256) with cache	**1.192**	10.5M
GHRNN-BH	1.226	10.5M
bGHRNN	1.248	10.5M
bGHRNN-BH	1.228	10.5M

_	n	o	_	i	t	_	w	a	s	_	n	'	t	_	b	l	a	c	k
0.8272	0.5619	0.446	1	0.4709	0.6184	0.9334	0.5729	0.4069	0.3725	1	0.6506	0.3243	0.3613	0.998	0.6222	1	0.4121	0.8959	0.6649
1	0	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0

_	m	o	n	d	a	y	_	<eos>	_	b	u	t	_	w	h	i	l	e	_
1	0.5304	0.5397	0.553	0.8189	0.6002	0.4919	1	0.5034	0.7724	0.5933	0.6085	0.4499	0.8022	0.5822	0.6198	0.3819	0.4224	0.3577	0.7102
1	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1

t	h	e	_	n	e	w	_	y	o	r	k	_	s	t	o	c	k	_	e
0.4962	0.5746	0.3119	1	0.6362	0.599	0.5696	0.7595	0.5188	0.2111	0.4709	0.3852	1	0.5366	0.595	0.3854	0.3588	0.3665	1	0.4423
0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0

x	c	h	a	n	g	e	_	d	i	d	_	n	'	t	_	f	a	l	l
0.369	0.3048	0.3119	0	0.077	0.3317	0.3383	1	0.5359	0.6405	0.5529	0.92	0.5894	0.2776	0.2552	1	0.6921	0.5955	0.7875	0.7345
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0

_	a	p	a	r	t	_	f	r	i	d	a	y	_	a	s	_	t	h	e
1	0.4365	1	0.3227	0.6527	0.69	0.9903	0.5225	0.4458	0.5172	0.8246	0.5911	0.4521	0.9441	0.4327	0.5905	0.9022	0.421	0.5079	0.2624
1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0

Figure 3: *Gate values of the first 100 characters of PTB test set. <eos> token indicates end of sentence marker. <space> token is replaced with '_' mark for readability.*

We can consider 'No gate GHRNN' as a 4 layered model with skip-connections. It shows a better result than the simply layered model but is worse than the other GHRNNs. This comparison shows that the gates improve the performance.

When we compare GHRNN and GHRNN-256, we can find that training with a longer sequence is advantageous. It also means that GHRNN can preserve the long-term context and make better prediction with it.

As observed in Table 1, using the binary gates lowers the performance. However, bGHRNN shows comparable performance to GHRNN when the boundary hint is given. We check how much bGHRNN can reduce LSTM computation on PTB test set. The bGHRNN reduces the amount of computation required to inference the test set by 21%.

GHRNN-BH shows similar performance to GHRNN, and reduces LSTM calculation by 41%. Most of the reduction in computation is due to LSTM-W. Fig. 3 shows gate values of GHRNN-BH model on the first 100 characters of PTB test set. Each row is listed in order of the input characters, the gate values of LSTM-S, and the gate values of LSTM-W. The gate value of LSTM-W is 1 for <*space*> token and 0 for other characters. We can consider that LSTM-W maintains the current word information until a new word comes in. The gate value of LSTM-S appears mostly 1 at the <*space*> token because many BPE codes end with a <*space*> token. In the PTB data set, of the BPE codes that contain a <*space*> token, the BPE code ending with a <*space*> token is 98%.

4.2. Wikitext-2 dataset

The Wiki2 dataset has been proposed to compensate for the problems of PTB dataset that is limited to small word vocabulary, lower case, and punctuation removal [16]. We use preprocessed Wiki2 dataset [18]. Train, valid and test sets employ 255, 128 and 138 kinds of tokens, and contain 10.9M, 1.1M, and 1.3M of characters, respectively.

As shown in Table 2, the GHRNN-BH model shows a better result than the GHRNN model on Wiki2 dataset. It also shows better performance than HCLM without cache. However, when a cache is applied, both shows comparable performance. The caching mechanism used in HCLM is a version of [29] for open-vocabulary models. We consider the difference in performance improvement is caused by different cache mechanisms employed..

Table 2: *CLM results on Wikitext2 dataset*

Model	BPC	Parameter size
HCLM[18]	1.670	–
HCLM with cache[18]	**1.500**	–
GHRNN	1.600	10.5M
GHRNN with cache	1.519	10.5M
GHRNN-BH	1.590	10.5M
GHRNN-BH with cache	**1.506**	10.5M

5. Concluding remarks

In this paper, we propose the GHRNN (gated hierarchical recurrent neural networks) based CLMs. GHRNN has a hierarchical structure and gates for maintaining long-term contexts. GHRNN consists of four LSTMs; three of which encode character-level, sub-word level, and word-level contexts, while the remaining LSTM uses the output of the other LSTMs to estimate the probability distribution of the next character. Two gates are added to GHRNN and they coordinate the maintenance of existing information and the introduction of new one. In particular, if the gate is deactivated, the LSTMs associated with it need not be computed. We evaluate the GHRNN performance on PTB and Wiki2 dataset. When we apply a continuous cache to GHRNN, the model shows BPC of 1.192, which is comparable to the state of the art performance on PTB dataset.

6. Acknowledgements

This work was supported in part by the Brain Korea 21 Plus Project and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2015R1A2A1A10056051). This work was also supported by Samsung Advanced Institute of Technology through Neural Processing Research Center (NPRC) in Seoul National University.

7. References

[1] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[2] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th Interna-*

tional Conference on Machine Learning (ICML-11), 2011, pp. 1017–1024.

[3] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Comput. Linguist.*, vol. 16, no. 2, pp. 79–85, Jun. 1990. [Online]. Available: http://dl.acm.org/citation.cfm?id=92858.92860

[4] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, "Breaking the softmax bottleneck: a high-rank rnn language model," *arXiv preprint arXiv:1711.03953*, 2017.

[5] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent highway networks," *arXiv preprint arXiv:1607.03474*, 2016.

[6] A. Mujika, F. Meier, and A. Steger, "Fast-slow recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 5917–5926.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[9] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," *arXiv preprint arXiv:1611.09913*, 2016.

[10] C. M. Bishop, "Pattern recognition and machine learning," 2006.

[11] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *International Conference on Machine Learning*, 2016, pp. 1120–1128.

[12] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[13] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. A. Hasegawa-Johnson, and T. S. Huang, "Dilated recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 76–86.

[14] K. Hwang and W. Sung, "Character-level language modeling with hierarchical recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5720–5724.

[15] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[16] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[17] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," in *International Conference on Machine Learning*, 2014, pp. 1863–1871.

[18] K. Kawakami, C. Dyer, and P. Blunsom, "Learning to create and reuse words in open-vocabulary neural language modeling," *arXiv preprint arXiv:1704.06986*, 2017.

[19] J. Chung, K. Cho, and Y. Bengio, "A character-level decoder without explicit segmentation for neural machine translation," *arXiv preprint arXiv:1603.06147*, 2016.

[20] P. Gage, "A new algorithm for data compression," *The C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[21] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[22] D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, and C. Pal, "Zoneout: Regularizing rnns by randomly preserving hidden activations," *arXiv preprint arXiv:1606.01305*, 2016.

[23] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[24] J. Suarez, "Language modeling with recurrent highway hypernetworks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3269–3278.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[28] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[29] E. Grave, A. Joulin, and N. Usunier, "Improving neural language models with a continuous cache," *arXiv preprint arXiv:1612.04426*, 2016.