CHARACTER-LEVEL LANGUAGE MODELING WITH HIERARCHICAL RECURRENT NEURAL NETWORKS

Kyuyeon Hwang and Wonyong Sung

Department of Electrical and Computer Engineering
Seoul National University
1, Gwanak-ro, Gwanak-gu, Seoul, 08826 Korea
kyuyeon.hwang@gmail.com; wysung@snu.ac.kr

ABSTRACT

Recurrent neural network (RNN) based character-level language models (CLMs) are extremely useful for modeling out-of-vocabulary words by nature. However, their performance is generally much worse than the word-level language models (WLMs), since CLMs need to consider longer history of tokens to properly predict the next one. We address this problem by proposing hierarchical RNN architectures, which consist of multiple modules with different timescales. Despite the multi-timescale structures, the input and output layers operate with the character-level clock, which allows the existing RNN CLM training approaches to be directly applicable without any modifications. Our CLM models show better perplexity than Kneser-Ney (KN) 5-gram WLMs on the One Billion Word Benchmark with only 2% of parameters. Also, we present real-time character-level end-to-end speech recognition examples on the Wall Street Journal (WSJ) corpus, where replacing traditional mono-clock RNN CLMs with the proposed models results in better recognition accuracies even though the number of parameters are reduced to 30%.

***Index Terms*—** Character-level language model, hierarchical recurrent neural network, long short-term memory

1. INTRODUCTION

Language models (LMs) show the probability distribution over sequences of words or characters, and they are very important for many speech and document processing applications including speech recognition, text generation, and machine translation [1, 2, 3]. LMs can be classified into character-, word-, and context-levels according to the unit of the input and output. In the character-level LM (CLM) [2], the probability distribution of the next characters are generated based on the past character sequences. Since the number of alphabets is small in English, for example, the input and output of the CLM is quite simple. However, the word-level LM (WLM) is usually needed because the character-level modeling is disadvantaged in utilizing the long period of past sequences. However, the problem of the word-level model is the complexity of the input and output because the vocabulary size to be supported can be bigger than 1 million.

LMs have long been developed by analyzing a large amount of texts and storing the probability distribution of word sequences into the memory. The statistical language model demands a large memory space, often exceeding 1 GB, not only because the vocabulary size is large but also their combinations needs to be considered. In recent years, the language modeling based on recurrent neural networks (RNNs) have been actively investigated [4, 5]. However, the RNN based WLMs still demand billions of parameters because of the large vocabulary size.

In this work, we propose hierarchical RNN based LMs that combine the advantageous characteristics of both character- and word-level LMs. The proposed network consists of a low-level and a high-level RNNs. The low-level RNN employs the character-level input and output, and provides the short-term embedding to the high-level RNN that operates as the word-level RNN. The high-level RNN do not need complex input and output because it receives the character-embedding information from the low-level network, and sends the word-prediction information back to the low-level in a compressed form. Thus, when considering the input and output, the proposed network is a CLM, although it contains a word-level model inside. The low-level module operates with the character input clock, while the high-level one runs with the space (<w>) and sentence boundary tokens (<s>) that separates words. We expect this hierarchical LM can be extended for processing a longer period of information, such as sentences, topics, or other contexts.

2. RELATED WORK

2.1. Character-level language modeling with RNNs

CLMs need to consider longer sequence of history tokens to predict the next token than the WLMs, due to the smaller unit of tokens. Therefore, traditional N-gram models cannot be employed for CLMs. Thanks to the recent advances in RNNs, RNN-based CLMs has begun to show satisfactory performances [2, 6]. Especially, deep long short-term memory (LSTM) [7] based CLMs show excellent performance and successfully applied to end-to-end speech recognition system [8].

For training RNN CLMs, training data should be first converted to the sequence of one-hot encoded character vectors, $\mathbf{x}_t$, where the characters include word boundary symbols, <w> or space, and optionally sentence boundary symbols, <s>. Then, as shown in Figure 1, the RNN is trained to predict the next character $\mathbf{x}_{t+1}$ by minimizing the cross-entropy loss of the softmax output [9] that represents the probability distributions of the next character.

2.2. Character-aware word-level language modeling

There has been many attempts to make WLMs understand character-level inputs. One of the most successful approaches is to encode the arbitrary character sequence to fixed dimensional vector, which is called word embedding, and feed this vector to the word-level RNN

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).

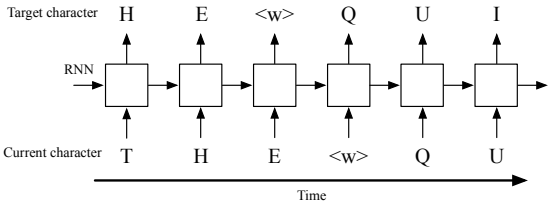


Fig. 1: Training an RNN-based CLM.

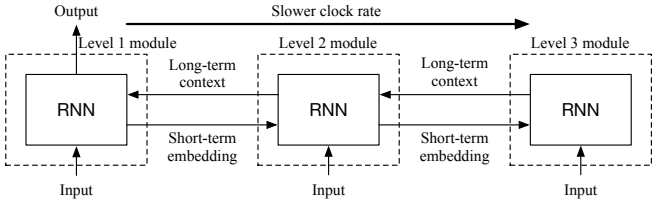


Fig. 2: Hierarchical RNN (HRNN).

LMs. In [10], convolutional neural networks (CNNs) are used to generate word embeddings, and achieved the state of the art results on English Penn Treebank corpus [11]. The similar CNN-based embedding approach is used by [5] with very large LSTM networks on the One Billion Word Benchmark [12], also achieving the state of the art perplexity. In [13, 14], bidirectional LSTMs are employed instead of CNNs for word embedding. However, in all of these approaches, LMs still generate the output probabilities at the word-level. Although the character-level modeling approach of the output word probability is introduced using CNN softmax in [5], the base LSTM network still runs with a word-level clock.

Our approach is different from the above ones in many ways. First, our base model is the character-level RNN LMs, instead of WLMs, and we extend this model to enhance the model to consider long-term contexts. Therefore, the output probabilities are generated with a character-level clocks. This property is extremely useful for character-level beam search for end-to-end speech recognition [8]. Also, the input and output of our model are the same as those of the traditional character-level RNNs, thus the same training algorithm and recipe can be used without any modifications. Furthermore, the proposed models have significantly less number of parameters compared to WLM-based ones, since the size of our model does not directly depend on the vocabulary size of the training set. Note that a similar hierarchical concept has been used for character-level machine translation [15]. However, we propose more general hierarchical unidirectional RNN architecture that can be applied for various applications.

3. RNNS WITH EXTERNAL CLOCK AND RESET SIGNALS

In this section, we generalize the existing RNN structures and extend them with external clocks and reset signals. The extended models become the basic building blocks of the hierarchical RNNs.

Most types of RNNs or recurrent layers can be generalized as

$$\mathbf{s}_t = f(\mathbf{x}_t, \mathbf{s}_{t-1}) \ , \quad \mathbf{y}_t = g(\mathbf{s}_t) \tag{1}$$

where $\mathbf{x}_t$ is the input, $\mathbf{s}_t$ is the state, $\mathbf{y}_t$ is the output at time step t, $f(\cdot)$ is the recurrence function, and $g(\cdot)$ is the output function. For example, a hidden layer of Elman networks [16] can be written as

$$\mathbf{y}_t = \mathbf{s}_t = \mathbf{h}_t = \sigma(W_{hx}\mathbf{x}_t + W_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \tag{2}$$

where $\mathbf{h}_t$ is the activation of the hidden layer, $\sigma(\cdot)$ is the activation function, W_{hx} and W_{hh} are the weight matrices and $\mathbf{b}_h$ is the bias vector.

LSTMs [7] with forget gates [17] and peephole connections [18] can also be converted to the generalized form. The forward equations of the LSTM layer are as follows:

$$\mathbf{i}_t = \sigma(W_{ix}\mathbf{x}_t + W_{ih}\mathbf{h}_{t-1} + W_{im}\mathbf{m}_{t-1} + \mathbf{b}_i) \tag{3}$$

$$\mathbf{f}_t = \sigma(W_{fx}\mathbf{x}_t + W_{fh}\mathbf{h}_{t-1} + W_{fm}\mathbf{m}_{t-1} + \mathbf{b}_f) \tag{4}$$

$$\mathbf{m}_t = \mathbf{f}_t \circ \mathbf{m}_{t-1} + \mathbf{i}_t \circ \tanh(W_{mx}\mathbf{x}_t + W_{mh}\mathbf{h}_{t-1} + \mathbf{b}_m) \tag{5}$$

$$\mathbf{o}_t = \sigma(W_{ox}\mathbf{x}_t + W_{oh}\mathbf{h}_{t-1} + W_{om}\mathbf{m}_t + \mathbf{b}_o) \tag{6}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{m}_t) \tag{7}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$ are the input, forget, and output gate values, respectively, $\mathbf{m}_t$ is the memory cell state, $\mathbf{h}_t$ is the output activation of the layer, $\sigma(\cdot)$ is the logistic sigmoid function, and $\circ$ is the element-wise multiplication operator. These equations can be generalized by setting $\mathbf{s}_t = [\mathbf{m}_t, \mathbf{h}_t]$ and $\mathbf{y}_t = \mathbf{h}_t$.

Any generalized RNNs can be converted to the ones that incorporate an external clock signal, c_t, as

$$\mathbf{s}_t = (1 - c_t)\mathbf{s}_{t-1} + c_t f(\mathbf{x}_t, \mathbf{s}_{t-1}) \ , \quad \mathbf{y}_t = g(\mathbf{s}_t) \tag{8}$$

where c_t is 0 or 1. The RNN updates its state and output only when $c_t = 1$. Otherwise, when $c_t = 0$, the state and output values remain the same as those of the previous step.

The reset of RNNs is performed by setting $\mathbf{s}_{t-1}$ to 0. Specifically, (8) becomes

$$\mathbf{s}_t = (1 - c_t)(1 - r_t)\mathbf{s}_{t-1} + c_t f(\mathbf{x}_t, (1 - r_t)\mathbf{s}_{t-1}) \tag{9}$$

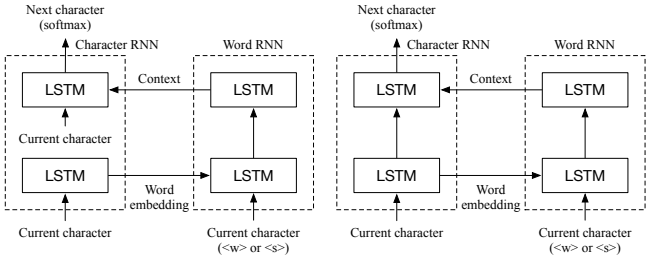
where the reset signal $r_t = 0$ or 1. When $r_t = 1$, the RNN forgets the previous contexts.

If the original RNN equations are differentiable, the extended equations with clock and reset signals are also differentiable. Therefore, the existing gradient-based training algorithms for RNNs, such as backpropagation through time (BPTT), can be employed for training the extended versions without any modifications.

4. CHARACTER-LEVEL LANGUAGE MODELING WITH A HIERARCHICAL RNN

The proposed hierarchical RNN (HRNN) architectures have several RNN modules with different clock rates as depicted in Figure 2. The higher level module employs a slower clock rate than the lower module, and the lower level module is reset at every clock of the higher level module. Specifically, if there are L hierarchy levels, then the RNN consists of L submodules. Each submodule l operates with an external clock $c_{l,t}$ and a reset signal $r_{l,t}$, where $l = 1, \cdots, L$. The lowest level module, $l = 1$, has the fastest clock rate, that is, $c_{1,t} = 1$ for all t. On the other hand, the higher level modules, $l > 1$, have slower clock rates and $c_{l,t}$ can be 1 only when $c_{l-1,t}$ is 1. Also, the lower level modules $l < L$ are reset by the higher level clock signals, that is, $r_{l,t} = c_{l+1,t}$.

The hidden activations of a module, $l < L$, are fed to the next higher level module, $l + 1$, *delayed by one time step* to avoid unwanted reset by $r_{l,t} = c_{l+1,t} = 1$. This hidden activation vector,



(a) HLSTM-A (b) HLSTM-B

Fig. 3: Two-level hierarchical LSTM (HLSTM) structures for CLMs.

or embedding vector, contains compressed short-term context information. The reset of the module by the higher level clock signals helps the module to concentrate on compressing only the short term information, rather than considering longer dependencies. The next higher level module, $l+1$, process this short-term information to generate the long-term context vector, which is fed back to the lower level module, l. There is no delay for this context propagation.

For character-level language modeling, we use a two-level ($L = 2$) HRNN with letting $l = 1$ be a character-level module and $l = 2$ be a word-level module. The word-level module is clocked at the word boundary input, <w>, which is usually a whitespace character. The input and softmax output layer is connected to the character-level module, and the current word boundary token (e.g. <w> or <s>) information is given to the word-level module. Since this HRNNs have a scalable architecture, we expect this HRNN CLM can be extended for modeling sentence-level contexts by adding an additional sentence-level module, $l = 3$. In this case, the sentence-level clock, $c_{3,t}$ becomes 1 when the input character is a sentence boundary token <s>. Also, the word-level module should be clocked at both the word boundary input, <w>, and the sentence boundary input, <s>. In this paper, the experiments are performed only with the two-level HRNN CLMs.

We propose two types of two-level HRNN CLM architectures. As shown in Figure 3, both models have two LSTM layers per submodule. Note that each connection has a weight matrix. In the HLSTM-A architecture, both LSTM layers in the character-level module receives one-hot encoded character input. Therefore, the second layer of the character-level module is a generative model conditioned by the context vector. On the other hand, in HLSTM-B, the second LSTM layer of the character-level module does not have direct connection from the character inputs. Instead, a word embedding from the first LSTM layer is fed to the second LSTM layer, which makes the first and second layers of the character-level module work together to estimate the next character probabilities when the context vector is given. The experimental results show that HLSTM-B is more efficient for CLM applications.

Since the character-level modules are reset by the word-boundary token (i.e. <w> or whitespace), the context vector from the word-level module is the only source for the inter-word context information. Therefore, the model is trained to generate the context vector that contains useful information about the probability distribution of the next word. From this perspective, the word-level module in both HRNN CLM architectures can be considered as a word-level RNN LM, where the input is a word embedding vector and the output is a compressed descriptor of the next word probabilities. Although the proposed model consists of several RNN modules with different timescales, these can be jointly trained by BPTT as described in Section 3.

Table 1: Perplexities of CLMs on the WSJ corpus

Model	Size	# Params	BPC	Word PPL
Deep LSTM	2x512	3.23 M	1.148	99.5
Deep LSTM	4x512	7.43 M	1.132	93.3
Deep LSTM	4x1024	29.54 M	1.101	82.4
HLSTM-A	4x512	7.50 M	1.089	78.5
HLSTM-B (no reset)	4x512	8.48 M	1.080	75.7
HLSTM-B	4x512	8.48 M	1.073	73.6
HLSTM-B	4x1024	33.74 M	1.058	69.2

Table 2: Perplexities of WLMs on the WSJ corpus in the literature

Model	# Params	PPL
KN 5-gram (no count cutoffs) [26]	-	80
RNN-640 + ME 4-gram feature [26]	2 G	59

5. EXPERIMENTS

The proposed HRNN based CLMs are evaluated with two text datasets: the Wall Street Journal (WSJ) corpus [19] and One Billion Word Benchmark [12]. Also, we present an end-to-end speech recognition example, where HLSTM CLMs are employed for prefix tree-based beam search decoding.

The RNNs are trained with truncated backpropagation through time (BPTT) [20, 21]. Also, ADADELTA [22] and Nesterov momentum [23] is applied for weight update. No regularization method, such as dropout [24], is employed. The training is accelerated using GPUs by training multiple sequences in parallel [25].

5.1. Perplexity

In this section, our models are compared with other WLMs in the literature in terms of word-level perplexity (PPL). The word-level PPL of our models is directly converted from bits-per-character (BPC), which is the standard performance measure for CLMs, as follows:

$$PPL = 2^{BPC \times \frac{N_c}{N_w}} \tag{10}$$

where N_c and N_w are the number of characters and words in a test set, respectively. Note that sentence boundary symbols (<s>) are also regarded as characters and words.

5.1.1. Wall Street Journal (WSJ) corpus

The Wall Street Journal (WSJ) corpus [19] is designed for training and benchmarking automatic speech recognition systems. For the perplexity experiments, we used the non-verbalized punctuation (NVP) version of the LM training data inside the corpus. The dataset consists of about 37 million words, where one percent of the total data is held out for the final evaluation and does not participate in training. All alphabets are converted to the uppercases.

Table 1 shows the perplexities of traditional mono-clock deep LSTM and HLSTM based CLMs on the held-out set. Note that the size NxM means that the network consists of N LSTM layers, where each layer contains M memory cells. The HLSTM models

Table 3: Perplexities of the HRNN CLMs on the One Billion Word Benchmark

Model	Size	# Params	BPC	Word PPL
HLSTM-B	4x512	9.06 M	1.228	83.3
HLSTM-B	4x1024	34.90 M	1.140	60.7

Table 4: Perplexities of WLMs on the One Billion Word Benchmark in the literature

Model	# Params	PPL
Sigmoid RNN-2048 [29]	4.1 G	68.3
Interpolated KN-5, 1.1B n-grams [12]	1.76 G	67.6
LightRNN [30]	**41 M**	**66**
Sparse non-negative matrix LM [31]	33 G	52.9
RNN-1024 + ME 9-gram feature [12]	20 G	51.3
CNN input + 2xLSTM-8192-1024 [5]	1.04 G	30.0

show better perplexity performanes even when the number of LSTM cells or parameters is much smaller than that of the deep LSTM networks. Especially, HLSTM-B network with the size of 4x512 has about 9% lower perplexity than deep LSTM (4x1024) model, even with only 29% of parameters.

It is important to reset the character-level modules at the word-level clocks for helping the character-level modules to better concentrate on the short-term information. As observed in Table 1, removing the reset functionality of the character-level module of the HLSTM-B model results in degraded performance.

The non-ensemble perplexities of WLMs in the literature are presented in Table 2. The Kneser-Ney (KN) smoothed 5-gram model (KN-5) [27] is a strong non-neural WLM baseline. With the standard deep RNN based CLMs, it is very hard to beat KN-5 in terms of perplexity. However, it is surprising that all HLSTM models in Table 1 shows better perplexities than KN-5 does. The RNN based WLM model combined with the maximum entropy 4-gram feature [28, 26] shows much better results than the proposed HLSTM based CLM models. However, like most of the WLMs, it also needs a very large number (2 G) of parameters and cannot handle out-of-vocabulary (OOV) words.

5.1.2. One Billion Word Benchmark

The One Billion Word Benchmark [12] dataset contains about 0.8 billion words and roughly 800 thousand words of vocabulary. We followed the standard way of splitting the training and test data as in [12]. Each byte of UTF-8 encoded text is regarded as a character. Therefore, the size of the character set is 256.

Due to the large amount of training data and weeks of training time, only two HLSTM-B experiments are conducted with the size of 4x512 and 4x1024. As shown in Table 3, there are large gap (22.5) in word-level perplexity between the two models. Therefore, further improvement in perplexity can be expected with bigger networks.

The perplexities of other WLMs are summarized in Table 4. The proposed HLSTM-B model (4x1024) shows better perplexities than the interpolated KN-5 model with 1.1 billion n-grams [12] even though the number of parameters of our model is only 2% of that of the KN-5 model. Also, our model performs better than LightRNN [30], which is a word-level RNN LM that has about 17% more parameters than ours. However, much lower perplexities are reported with sparse non-negative matrix LM and the maximum entropy feature based RNN model [12], where the number of parameters are 33 G and 20 G, respectively. Recently, the state of the art perplexity of 30.0 was reported in [5] with a single model that has 1 G parameters. The model is basically a very large LSTM LM. However, the convolutional neural network (CNN) is used to generate word embedding of arbitrary character sequences as the input of the LSTM LM. Therefore, this model can handle OOV word inputs, however, still the model runs with a word-level clock.

Table 5: End-to-end ASR results on the WSJ Nov'92 20K evaluation set (`eval92`)

Model	Size	# Params	Word PPL	WER
Deep LSTM	4x512	7.43 M	93.3	8.36%
Deep LSTM	4x1024	29.54 M	82.4	7.85%
HLSTM-B	4x512	8.48 M	73.6	7.79%
HLSTM-B	4x1024	33.74 M	69.2	7.78%

5.2. End-to-end automatic speech recognition (ASR)

In this section, we apply the proposed CLMs to the end-to-end automatic speech recognition (ASR) system to evaluate the models in more practical situation than just measuring perplexities. The CLMs are trained with WSJ LM training data as in Section 5.1.1. Unlike WLMs, the proposed CLMs have very small number of parameters, so they can be employed for real-time character-level beam search.

The incremental speech recognition system proposed in [8] is used for the evaluation. The acoustic model is 4x512 unidirectional LSTM and end-to-end trained with connectionist temporal classification (CTC) loss [32] using the non-verbalized punctuation (NVP) portion of WSJ SI-284 training set. The acoustic features are 40-dimensional log-mel filterbank coefficients, energy and their delta and double-delta values, which are extracted every 10 ms with 25 ms Hamming window. The beam-search decoding is performed on a prefix-tree with depth-pruning and width-pruning [8]. The insertion bonus is 1.6, the LM weight is 2.0, and the beam width is 512.

The results are summarized in Table 5. It is observed that the perplexity of LM and the word error rate (WER) have strong correlation. As shown in the table, we can achieve a better WER by replacing the traditional deep LSTM (4x1024) CLM with the proposed HLSTM-B (4x512) CLM, while reducing the number of LM parameters to 30%.

6. CONCLUDING REMARKS

In this paper, hierarchical RNN (HRNN) based CLMs are proposed. The HRNN consists of several submodules with different clock rates. Therefore, it is capable of learning long-term dependencies as well as short-term details. The experimental results on One Billion Benchmark show that HLSTM-B networks significantly outperform Kneser-Ney 5-gram LMs with only 2% of parameters. Although other RNN-based WLMs show better performance than our models, they have impractically many parameters. On the other hand, as shown in the WSJ speech recognition example, the proposed model can be employed for the real-time speech recognition with less than 10 million parameters. Also, CLMs can handle OOV words by nature, which is a great advantage for the end-to-end speech recognition and many NLP tasks. One of the interesting future work is to train the clock signals, instead of using manually designed ones.

7. REFERENCES

[1] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of speech recognition*, Prentice Hall, 1993.

[2] Ilya Sutskever, James Martens, and Geoffrey E Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[3] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[4] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Proc. Interspeech*, 2010, vol. 2, p. 3.

[5] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[6] Michiel Hermans and Benjamin Schrauwen, "Training and analysing deep recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2013, pp. 190–198.

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

[8] Kyuyeon Hwang and Wonyong Sung, "Character-level incremental speech recognition with recurrent neural networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[9] John S Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*, pp. 227–236. Springer, 1990.

[10] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush, "Character-aware neural language models," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[11] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[12] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005*, 2013.

[13] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso, "Finding function in form: Compositional character models for open vocabulary word representation," in *2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1520–1530.

[14] Yasumasa Miyamoto and Kyunghyun Cho, "Gated word-character recurrent language model," in *2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1992–1997.

[15] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black, "Character-based neural machine translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, vol. 357–361.

[16] Jeffrey L Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[17] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[18] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber, "Learning precise timing with LSTM recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[19] Douglas B Paul and Janet M Baker, "The design for the Wall Street Journal-based CSR corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[20] Paul J Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[21] Ronald J Williams and Jing Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories," *Neural Computation*, vol. 2, no. 4, pp. 490–501, 1990.

[22] Matthew D Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[23] Yurii Nesterov, "A method of solving a convex programming problem with convergence rate O (1/k2)," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[24] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[25] Kyuyeon Hwang and Wonyong Sung, "Single stream parallelization of generalized LSTM-like RNNs on a GPU," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 1047–1051.

[26] Tomáš Mikolov, *Statistical language models based on neural networks*, Ph.D. thesis, Brno University of Technology, 2012.

[27] Reinhard Kneser and Hermann Ney, "Improved backing-off for m-gram language modeling," in *1995 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 1995, vol. 1, pp. 181–184.

[28] Tomas Mikolov and Geoffrey Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop*, 2012, pp. 234–239.

[29] Shihao Ji, SVN Vishwanathan, Nadathur Satish, Michael J Anderson, and Pradeep Dubey, "BlackOut: Speeding up recurrent neural network language models with very large vocabularies," in *4th International Conference on Learning Representations*, 2016.

[30] Xiang Li, Tao Qin, Jian Yang, Xiaolin Hu, and Tieyan Liu, "LightRNN: Memory and computation-efficient recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4385–4393.

[31] Noam Shazeer, Joris Pelemans, and Ciprian Chelba, "Sparse non-negative matrix language modeling for skip-grams," in *Proc. Interspeech*, 2015, pp. 1428–1432.

[32] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 369–376.