Span-Based Neural Buffer: Towards Efficient and Effective Utilization of Long-Distance Context for Neural Sequence Models

Yangming Li,[1,3] **Kaisheng Yao,**[1] **Libo Qin,**[3] **Shuang Peng,**[1] **Yijia Liu,**[2] **Xiaolong Li**[1]
[1]Ant Financial, [2]Alibaba,
[3]Research Center for Social Computing and Information Retrieval,
Harbin Institute of Technology, Harbin, China
{yangmingli, lbqin}@ir.hit.edu.cn, {kaisheng.yao, jianfeng.ps, xll}@antfin.com, yanshan.lyj@alibaba-inc.com

Abstract

Neural sequence model, though widely used for modeling sequential data such as the language model, has *sequential recency bias* (Kuncoro et al. 2018) to the local context, limiting its full potential to capture long-distance context. To address this problem, this paper proposes augmenting sequence models with a span-based neural buffer that efficiently represents long-distance context, allowing a gate policy network to make interpolated predictions from both the neural buffer and the underlying sequence model. Training this policy network to utilize long-distance context is however challenging due to the *simple sentence dominance* problem (Marvin and Linzen 2018). To alleviate this problem, we propose a novel training algorithm that combines an annealed maximum likelihood estimation with an intrinsic reward-driven reinforcement learning. Sequence models with the proposed span-based neural buffer significantly improve the state-of-the-art perplexities on the benchmark Penn Treebank and WikiText-2 datasets to 43.9 and 35.2 respectively. We conduct extensive analysis and confirm that the proposed architecture and the training algorithm both contribute to the improvements.

Introduction

The ability to make accurate prediction for sequence elements given their history is extremely important in analyzing, modeling, and utilizing time sequential data including speech and natural language. In language modeling, a word-level language model is trained to predict the next word given its previous words. It is an essential component in various natural language processing tasks.

Recent works of using neural sequence model (Mikolov et al. 2010; Merity, Keskar, and Socher 2017; Krause et al. 2017), in particular long short-term memory networks (LSTM) and other recurrent neural network (RNN) variants, have shown that they are very effective. However, sequential networks have structural *sequential recency bias* (Kuncoro et al. 2018) that causes difficulty of the models in utilizing long-distance context. Moreover, (Khandelwal et al. 2018) also confirms this issue by comprehensive experiments on LSTM-based language model.

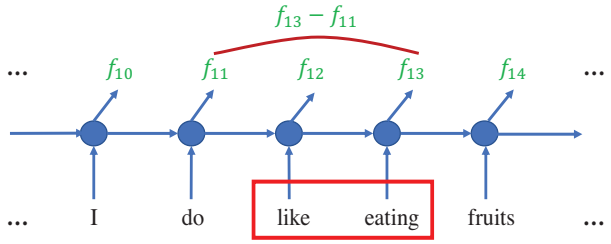

Figure 1: Illustration of a span. A context with consecutive words marked in red box is represented by the difference $f_{13} - f_{11}$ marked in red arc, where f_{ij}s are the hidden states from a sequential neural network.

To better capture the long-distance context for neural sequence model like language modeling, we have witnessed three approaches in recent years. The first focuses on using supervised syntactic parser (Wu et al. 2017), the second approach relies on latent tree learning (Shen et al. 2017; 2018), and the third approach augments sequential models with the memory of context (Tran, Bisazza, and Monz 2016; Daniluk et al. 2017). However, the first approach requires extra treebank data that can be domain-specific and expensive to annotate. The second approach of learning latent trees faces challenges of mitigating gaps of the structures to be discovered a posteriori from the data and those can be annotated and interpreted a priori, making it hard to train and unwieldy in practice (Williams, Drozdov, and Bowman 2018). Moreover, performance of the latent tree models is still behind the sequential network (Shen et al. 2018; Wang, Gong, and Liu 2019).

In the third approach, (Tran, Bisazza, and Monz 2016) introduces a memory block to help LSTM retain past hidden states. (Daniluk et al. 2017) uses attention mechanism on the extra memory. However, this type of memory is not memory efficient as it is word-level, requiring large memory to store. It is also not computationally efficient, as it needs much computation to retrieve information from it. Moreover, it is also observed in (Daniluk et al. 2017) that augmenting LSTM with this type of memory doesn't necessarily lead to effective utilization of long-distance context.

Extensive works (Nelson et al. 2017) have shown that language is processed in continuous chunks of words. This essential property can be deduced from tree-structure representation of languages. As resorting to annotation or induction of trees can lead to the aforementioned issues, we instead rely on this essential continuity property in building neural sequence models. The key insight in this paper is therefore to build contexts of a word with one or many chunks of consecutive words in adjacency to the word or in distance, without using trees.

This paper proposes a span-based neural buffer to augment neural sequence model with directly accessible long-distance context. Different from the word-level memory in (Tran, Bisazza, and Monz 2016; Daniluk et al. 2017), the neural buffer is filled with spans representing chunks of consecutive words. Technically, we use RNN-minus feature to represent a span (Stern, Andreas, and Klein 2017; Wang and Chang 2016), as illustrated in Figure 1. This representation achieves memory efficiency in storing large amount of context with little memory. Moreover, additional cost to compute the span representation is very low. With the span-based representation, we are able to expand the model capacity without sacrificing run-time efficiency by simply increasing span size.

Prediction of the next word is an interpolation between the prediction from the neural buffer and that of the underlying neural sequence model. However, maximum likelihood estimation can lead the model to make inferences solely based on local context since most real-world sentences are grammatically simple and those words can be easily inferred by local context (Marvin and Linzen 2018), which we refer the phenomenon to *simple sentence dominance* problem. Moreover, the position of relevant distant context and occurrence of it are sparse from a text of finite size. This means that the maximum likelihood estimation method doesn't have reliable estimates of the interpolation for this type of rare but nevertheless possible long-distance context.

To solve this problem, we design a novel training algorithm that combines an annealed maximum likelihood estimation with reinforcement learning that utilizes an intrinsic reward. The reward encourages exploration of long-distance context. The proposed maximum likelihood estimation produces model parameters for both of the sequential networks and those in using span-based neural buffer. The annealing with high temperature during training allows gradients to be directed to both of the local context and distant context. This temperature is reduced during test so that prediction is not ambiguous. Moreover, this maximum likelihood estimation is a regularization for reinforcement learning, avoiding making trivial interpolation of using only long-distance context.

We have conducted extensive experiments on two standard benchmarks to evaluate the proposed model and its ability to utilize long-distance context. Experimental results demonstrate that it significantly outperforms all previous methods. Our contributions are as follows:

- a novel span-based neural buffer with policy network to address *sequential recency bias* problem, enabling efficient and effective utilization of long-distance context;
- a training algorithm to solve *simple sentence dominance* problem, using an annealed maximum likelihood estimation and a reinforcement learning with intrinsic reward;
- a language model using the proposed approach that significantly outperforms all previous methods and establishes new state-of-the-art results on PTB and WT2.

Preliminary

This section briefly introduces neural sequence model for language modeling. Given a prefix or history, a neural sequence language model predicts its next word. Let V be the vocabulary size of a corpus to train this model. Each word is represented by a one-hot encoding vector $x_t \in R^V$, corresponding to its index in the vocabulary. Using the chain rule, the probability of the sentence $S = [x_1, x_2, \cdots, x_T]$ is

$$p(S) = p([x_1 \cdots x_T]) = \prod_{t=1}^{T} p(x_t|x_{1:t-1}). \quad (1)$$

Neural sequence model utilizes a continuous-valued state vector $h_t \in R^d$ to encode the prefix $x_{1:t-1}$. With this representation, the conditional probability of the next word x_{t+1} can be parameterized as

$$p(x_{t+1}|x_{1:t}) \propto \exp(h_t^T O_{x_{t+1}}), \quad (2)$$

where O is a trainable matrix to project the d-dimension state vector h_t to a V-dimension vector for softmax operation. The model computes the state vector h_t recursively by the following equation at each position t

$$h_t = \Phi(x_t, h_{t-1}), \quad (3)$$

where Φ represents the state transition operation that is specific to neural network types such as LSTM.

Architecture

The proposed model uses an architecture illustrated in Fig. 2. The span-based neural buffer is used together with an underlying sequence model, for instance LSTM. Prediction of the next word is an interpolation of the predictions from the sequence model and the buffer. Weights for the interpolation are controlled by a policy network.

Span-based Neural Buffer

Instead of utilizing or inducing tree-like structure to endow a sequence model with access to long-distance context, the proposed model stores context with a neural buffer of size B. It uses continuity property of language to split context into several spans with equal length L. Because of the equal split, here the phrase structure in the sense of syntactics is approximated, without resorting to syntactic trees.

A span at position i is represented using the RNN-minus feature, i.e., $s_i = h_i - h_{i-L+1}$. The prefix C_t for word x_t at t is a concatenation of m spans, with $m = B/L$, i.e.,

$$C_t = [s_{t-B}, s_{t-B+L}, \cdots, s_{t-L+1}, s_{t-1}]. \quad (4)$$

Notice that the span index in prefix C_t is from $t - B$ to $t-1$ with a step size of L. The model uses attention to access long-distance context as follows

$$\xi_t = C_t^T \alpha_t. \quad (5)$$

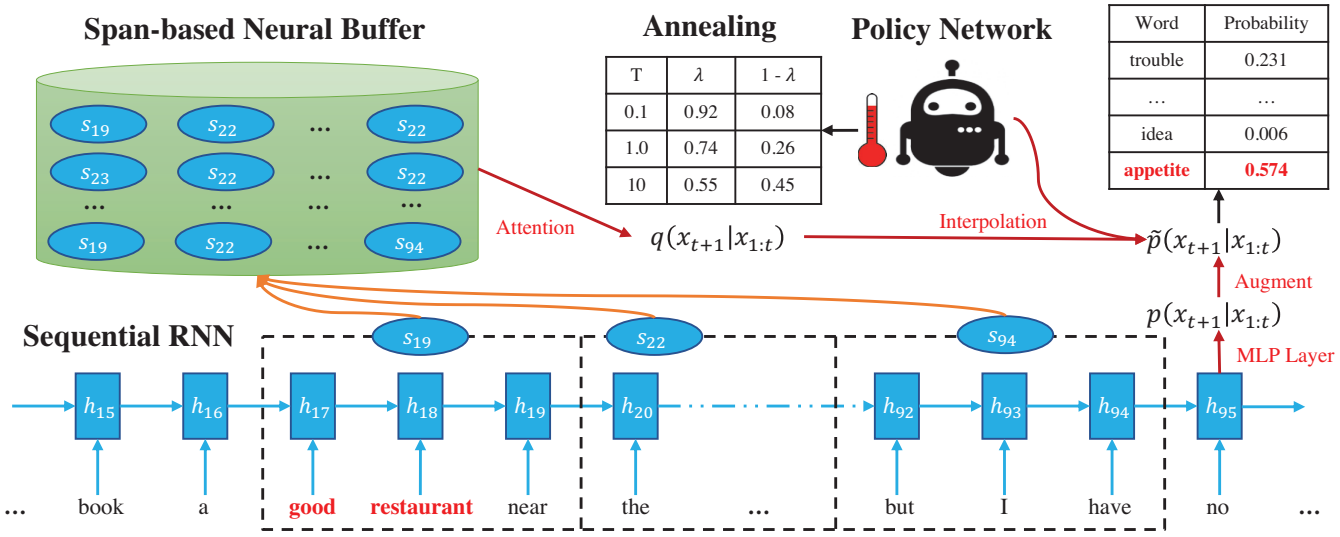

Figure 2: The model consists of three components, an underlying neural sequence model such as a sequential RNN, a span-based neural buffer and a policy network. The policy network interpolates probabilities of predicting the next word from the hidden state of the RNN and from the neural buffer, the latter is obtained using attention. Policy network has different temperature during training and test. P_{buffer} and P_{hidden} each refer to $p(x_{t+1}|x_{1:t})$ in Eq. (2) and $q(x_{t+1}|x_{1:t})$ in Eq. (8).

The attention weight $\alpha_t \in R^m$ has its i-th element $\alpha_{t,i}$ which is computed as follows:

$$\alpha_{t,i} = \frac{\exp(\gamma_{t,i})}{\sum_{j\in\{t-1,t-1-S,\cdots,t-B\}} \exp(\gamma_{t,j})}, \quad (6)$$

in which function $\gamma(\cdot)$ is certain similarity measure. Here we adopt the formula used in (Wu et al. 2017).

$$\gamma_{t,i} = v^T(W_h h_t + W_s s_i), \quad (7)$$

with v, W_h and W_s as learnable parameters.

The logit for the next word x_{t+1} is obtained by a linear projection $\xi_t^T O_{x_{t+1}}$ with the same matrix O tied with Eq. (2). The probability of x_{t+1} is obtained via a softmax output layer of this logit over all of logits, i.e.,

$$q(x_{t+1}|x_{1:t}) \propto \exp(\xi_t^T O_{x_{t+1}}). \quad (8)$$

Gated Interpolation

To predict the next word x_{t+1}, we interpolate the probabilities from the sequential RNNs $p(x_{t+1}|x_{1:t})$ in Eq. (2) and the neural buffer $q(x_{t+1}|x_{1:t})$ in Eq. (8) as follows

$$\begin{aligned}\tilde{p}(x_{t+1}|x_{1:t}) &= \lambda_t * q(x_{t+1}|x_{1:t}) \\ &+(1-\lambda_t) * p(x_{t+1}|x_{1:t}).\end{aligned} \quad (9)$$

The interpolation weight $\lambda_t \in [0.0, 1.0]$ to the neural buffer prediction is computed with a policy network. It uses hidden state of the sequence model as its observation and outputs a two dimensional weight vector. After normalization with softmax, one of the dimensions is used as λ_t; i.e.,

$$\lambda_t \propto \exp(W_g h_t), \quad (10)$$

where $W_g \in R^{2\times d}$ is the policy network parameter. The interpolation weight can be considered as a gate to accommodate predictions from the neural buffer. When it is zero, the prediction from the buffer is completely ignored.

Training Algorithm

Training the policy network parameter W_g and those in the sequential neural buffer reliably is however challenging. Most natural language sentences are grammatically simple and most words can be predicted from their local context (Khandelwal et al. 2018; Marvin and Linzen 2018), a phenomenon we name it as *simple sentence dominance* problem. An example of this problem is illustrated in Fig. 2. Conditioning on the prefix "I have no", words "idea" and "trouble" have higher probabilities than "appetite" to be used as the next word. Only with the information from long-distance context "good restaurant", a model can disambiguate the predictions and have the correct prediction of "appetite". However, because of the *simple sentence dominance* problem, long-distance context is rarely used and sparse. Maximum likelihood estimation for the policy network and the attention parameters for the neural buffer can be unreliable and insufficient. To address this problem, we define the following objective function:

$$Ł_t = -\log \tilde{p}(x_{t+1}|x_{1:t}) - \eta * r_t \log \lambda_t, \quad (11)$$

where the first term $\log \tilde{p}(x_{t+1}|x_{1:t})$ aims at increasing likelihood of predicting the correct next word. The second term $r_t \log \lambda_t$ is an objective with intrinsic rewards that encourages exploration of the neural buffer. The constant $\eta > 0$ in the second term is a trade-off coefficient between the likelihood objective and the intrinsic reward objective. We describe in more details below.

Training Policy Network using Intrinsic Rewards

The form $r_t \log \lambda_t$ resembles that used in policy gradient in REINFORCE algorithm (Williams 1992). Indeed, exploration of long-distance context is encouraged by having a

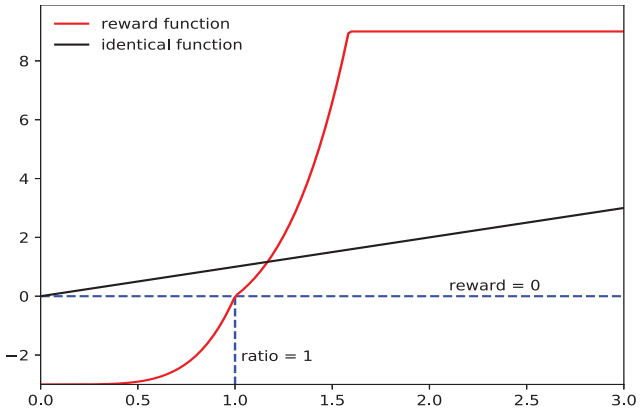

Figure 3: The reward function versus likelihood ratio. Also shown an identical function for comparison.

large λ_t. However, our heuristic rule is that, only with a well-trained sequential neural buffer that outputs higher likelihood than that from the underlying neural sequence model, should a large probability λ_t be encouraged. We design the intrinsic reward as follows:

$$r_t = f(\min((\frac{q(x_{t+1}|x_{1:t})}{p(x_{t+1}|x_{1:t}) + \epsilon})^{\kappa}, a) - b), \quad (12)$$

where $f(z)$ is a leaky RELU function defined as

$$f(z) = \begin{cases} z & \text{if } z \geq 0 \\ \beta z & \text{otherwise} \end{cases}. \quad (13)$$

The ratio $\frac{q(x_{t+1}|x_{1:t})}{p(x_{t+1}|x_{1:t})}$ in Eq. (12) is the division between the likelihoods from neural buffer $q(x_{t+1}|x_{1:t})$ and hidden state $p(x_{t+1}|x_{1:t})$. It acts as a measure of relative strength between neural buffer and RNN hidden state for predicting next word x_{t+1}. Since the exact next word x_{t+1} is observed during training, this ratio is exact and can be computed. The larger the ratio, the higher the reward is. a is the reward clipping threshold, preventing having an extremely large likelihood ratio. We use 10.0 for a in this paper. ϵ is a small constant, e.g. 1e-10 to avoid numerical issue.

To encourage the policy network outputting λ_t that makes clear preference between neural buffer and its underlying sequence model, we raise the likelihood ratio using power function. We also use a ratio larger than 1.0 for the leaky RELU. In this paper, we use 5 for κ and 3 for β.

A baseline is always subtracted from reward to reduce variance in reinforcement learning (Weaver and Tao 2001). Here the baseline b is 1. We illustrate the reward function versus likelihood ratio $\frac{q(x_{t+1}|x_{1:t})}{p(x_{t+1}|x_{1:t})}$ in Figure 3.

Training Neural Buffers using Annealed Maximum Likelihood Estimation

Because of *simple sentence* problem, not enough gradient may be directed to update neural buffer. Here we apply annealing during training. We set a temperature T high enough so that gradients from the output layer can be distributed evenly to the neural buffer and the underlying sequence model. We reformulate Eq. (10) as

$$\hat{\lambda}_t \propto \exp(\frac{W_g h_t}{T}). \quad (14)$$

This prevents a model from *explaining away* training sequences based on local context only; in fact, this encourages the neural buffer to also learn to predict the next word. Importantly, during training, $\hat{\lambda}_t$ shall be only applied to the first likelihood term $\log \tilde{p}(x_{t+1}|x_{1:t})$ in Eq. (11).

During test, we reduce the temperature so that the probability λ_t can be sharp. The conjecture is that, once the policy network are well trained, preference to either long-distance context or local context shall be clear.

Experiments

We use language modeling tasks to verify the effectiveness of our model. We first compare it with other state-of-the-art methods on two datasets. We then conduct experiments with oracle policy to validate the potential of long-distance context in neural buffer. We conduct ablation and case studies to understand the importance of each component in the proposed model. Finally, we also define a measure of long-distance context utilization and conduct experiments to visualize its dynamic change during training.

Datasets

We compare models on two benchmark datasets. The first is the Penn Treebank (PTB) with preprocessing (Mikolov et al. 2010) [1]. It consists of 923k training, 73k validation and 82k test words. Another word level corpus is WikiText-2 (WT2) [2]. It is about twice the size of Penn Treebank with a larger vocabulary and much lighter preprocessing. It contains about 2 million words.

Overall Results on Benchmark Datasets

We compare the proposed method (denoted as **SNB**) with baselines that have ever achieved state-of-the-art performances. We also implement dynamic evaluation (Krause et al. 2017) for all of the settings. For methods that don't have WT2 results, we obtain their performances by running their publicly-available implementations [3]. We made necessary tuning for these experiments. Following (Wang, Gong, and Liu 2019), we report language model performances using perplexity (PPL) on development and test sets. We also report the number of parameters. The results on PTB are summarized in Table 1. Table 2 has results for WT2.

Notably, SNB with the dynamic evaluation (denoted as **AWD-LSTM + MoS + PS + AT + SNB**), shown in the last row of each table, achieves new state-of-the-art results on PTB and WT2. It has the same number of parameters with the recently best performing model in (Wang, Gong, and

[1] http://www.fit.vutbr.cz/ imikolov/rnnlm/

[2] https://einstein.ai/research/the-wikitext-long-termdependency-language-modeling-dataset

[3] https://github.com/yikangshen/Ordered-Neurons, https://github.com/yikangshen/PRPN

Method	Params	Valid	Test
w/o dynamic evaluation			
AWD-LSTM (Merity, Keskar, and Socher 2017)	24M	60.00	57.30
AWD-LSTM + SNB	24M	**56.82**	**54.92**
AWD-LSTM + MoS (Yang et al. 2017)	22M	56.54	54.44
AWD-LSTM + MoS + SNB	22M	**54.21**	**52.62**
AWD-LSTM + MoS + PS (Press 2019)	22M	55.89	53.92
AWD-LSTM + MoS + PS + SNB	22M	**53.45**	**51.10**
AWD-LSTM + MoS + PS + AT (Wang, Gong, and Liu 2019)	22M	54.10	52.20
AWD-LSTM + MoS + PS + AT + SNB	22M	**52.37**	**50.18**
+ dynamic evaluation (Krause et al. 2017)			
AWD-LSTM (Merity, Keskar, and Socher 2017)	24M	51.60	51.10
AWD-LSTM + SNB	24M	**48.23**	**47.54**
AWD-LSTM + MoS (Yang et al. 2017)	22M	48.33	47.69
AWD-LSTM + MoS + SNB	22M	**46.53**	**45.41**
AWD-LSTM + MoS + PS (Press 2019)	22M	47.93	47.49
AWD-LSTM + MoS + PS + SNB	22M	**45.51**	**45.33**
AWD-LSTM + MoS + PS + AT (Wang, Gong, and Liu 2019)	22M	46.63	46.01
AWD-LSTM + MoS + PS + AT + SNB	22M	**44.37**	**43.87**

Table 1: Perplexity performance of each method on PTB dataset.

Liu 2019), but reduces PPLs to 43.87 on PTB and 35.16 on WT2, by an absolute reduction of 2.14 and 2.91, respectively. Similar reductions of PPLs are also observed when dynamic evaluation is not used.

Results show that span-based neural buffer is applicable to all of these settings; reductions from these baselines are consistently more than 2 points. The reductions from **AWD-LSTM + SNB** over **AWD-LSTM** setting are especially significant, reaching 5.53/5.68 and 4.37/3.71 in validation and test perplexity on the PTB and WT2 dataset.

Oracle Policy Experiments

The experiments here aim at showing the potential of the long-distance context in the neural buffer. We define an oracle policy as, between the neural buffer and its underlying sequence model, knowing which one to select to output the prediction of the next word. Since the oracle knows the next word x_{t+1}, its policy is simply picking the one with higher likelihood for the next word. The perplexity obtained is a lower bound of all possible policies.

We compare the oracle policy with the proposed model **AWD-LSTM+MoS+PS+AT+SNB** used in the overall experiments. Other comparisons include a model using only the underlying sequential RNN model and one model that uses prediction from the neural buffer only.

The results are listed in the top half of Table 3. In comparison with **AWD-LSTM+MoS+PS+AT+SNB**, **using only sequential RNN** has 3 point and 4 point higher PPL, and its results are close to (Wang, Gong, and Liu 2019), indicating that the improvement is mainly from the introduced sequential neural buffer. Furthermore, the oracle policy reaches a much lower perplexity. This shows great potential for utilizing the long-distance context.

PPLs from using only sequential neural buffer are higher than from using only sequential RNN. This validates the claim in (Marvin and Linzen 2018) that local context provides much information for language model. However, the combination of SNB and the underlying sequence model in **AWD-LSTM+MoS+PS+AT+SNB** is significantly better than using either of them. This indicates that the gains from sequential RNN and neural buffer are complementary.

Ablation Study

We investigate the impact of each methods in the proposed model, with results shown in the lower part of Table 3.

Annealing is verified with two setups. The first is without using annealing in training (denoted as **w/o annealing in training**) and the second is without annealing in test (denoted as **w/o annealing in test**). We observe that it is beneficial to raise temperature during training. Perplexity by **w/o annealing in training** is increased from 43.87 and 35.16 to 45.64 and 38.23, respectively, on the Penn Treebank and WikiText-2. Results from w/o annealing in test show that such degradation is less severe for test. These results indicate that, during training, it is important to distribute gradients to both sequential RNN and the neural buffer. However, during test, model should have clear preference of the context to use.

Intrinsic reward is verified by training without the second term $r_t \log \lambda_t$ in Eq. (11) (denoted as **w/o reward**) and making comparison against AWD-LSTM+MoS+PS+AT+SNB. Notice that both of them use sequential neural buffer. Results in Table 3 show that w/o reward performs much worse than (Wang, Gong, and Liu 2019)+SNB, therefore confirm the effectiveness of intrinsic reward-driven learning.

RNN-minus representation is verified by comparing it with word-level representation. That is, s_i in Eq. (7) is replaced with h_i. Similarly, Eq. (6) is normalized with respect to all of words in the context. This model is denoted

Method	Params	Valid	Test
w/o dynamic evaluation			
AWD-LSTM (Merity, Keskar, and Socher 2017)	33M	68.60	65.80
AWD-LSTM + SNB	33M	**63.07**	**60.43**
AWD-LSTM + MoS (Yang et al. 2017)	35M	63.88	61.45
AWD-LSTM + MoS + SNB	35M	**60.62**	**58.79**
AWD-LSTM + MoS + PS (Press 2019)	35M	62.38	59.98
AWD-LSTM + MoS + PS + SNB	35M	**60.11**	**57.87**
AWD-LSTM + MoS + PS + AT (Wang, Gong, and Liu 2019)	35M	61.10	58.95
AWD-LSTM + MoS + PS + AT + SNB	35M	**58.75**	**56.12**
+ dynamic evaluation (Krause et al. 2017)			
AWD-LSTM (Merity, Keskar, and Socher 2017)	24M	46.40	44.30
AWD-LSTM + SNB	24M	**40.72**	**38.59**
AWD-LSTM + MoS (Yang et al. 2017)	22M	42.41	40.68
AWD-LSTM + MoS + SNB	22M	**40.61**	**37.76**
AWD-LSTM + MoS + PS (Press 2019)	22M	40.75	39.03
AWD-LSTM + MoS + PS + SNB	22M	**39.98**	**37.01**
AWD-LSTM + MoS + PS + AT (Wang, Gong, and Liu 2019)	22M	39.58	38.07
AWD-LSTM + MoS + PS + AT + SNB	22M	**38.42**	**35.16**

Table 2: Perplexity performance of each method on WT2 dataset.

method	PTB	WT2
AWD-LSTM+MoS+PS+AT+SNB	43.87	35.16
using only sequential RNN	46.85	39.11
using only sequential neural buffer	52.54	45.57
oracle policy	**38.72**	**32.35**
w/o annealing in training	45.64	38.23
w/o annealing in test	44.12	37.45
w/o reward	45.92	38.76
w/o RNNs-minus feature	44.72	37.88

Table 3: Combined results for analysis and ablation study.

as **w/o RNN-minus feature**. Results in Table 3 show that PPL of this modification is higher, especially on WT2, than using span-based representation.

Dynamics of Long-distance Context Utilization

We define a measure for the utilization of long-distance context. The Percentage of Utilization (PoU) of long-distance context is the relative frequency of preferring long-distance context, computed as follows

$$PoU = \frac{\sum_t \delta(\lambda_t \geq 0.5)}{N}, \tag{15}$$

where $\delta(\cdot)$ is an indicator function that outputs one when its argument is true and zero otherwise. N is the number of samples. Figure 4 plots the change of PoU with training epochs. Initially, PoU is close to 0.5, meaning that the model is ambiguous of using long-distance context. It quickly drops to almost zero, from which PoU starts to increase and then converges to a certain percentage. In this example, it converges to approximately 10%. We quest the oracle policy and obtain the oracle PoU of 7%. This result indicates that the long-distance context is indeed utilized.

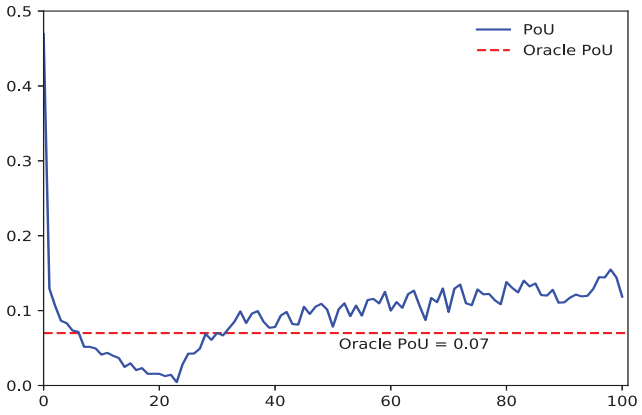

Figure 4: Changes of Percentage of Utilization (PoU) of the long-distance context with training epochs.

Notice that there is a gap between the PoU after convergence and the oracle PoU. Together with the performance gap in Table 3 between oracle policy and the proposed model, it indicates a direction for future performance improvements with better policy.

Case Study

We present a case study from PTB dataset in Figure 5. The heat map in the top half of the figure implies the relative strength of attention weights in SNB. Given the observation sequence "edlerly N N ... social security" in the bottom of the figure, the method using neural buffer attends to long distance context, "of poor <unk> families" and "edlerly N N in", implying that, when predicting the next word, it catches, to some extent, the global semantics from the observation and context in the buffer. In this example, its prediction of the next word "assist" is related to "of poor families" and

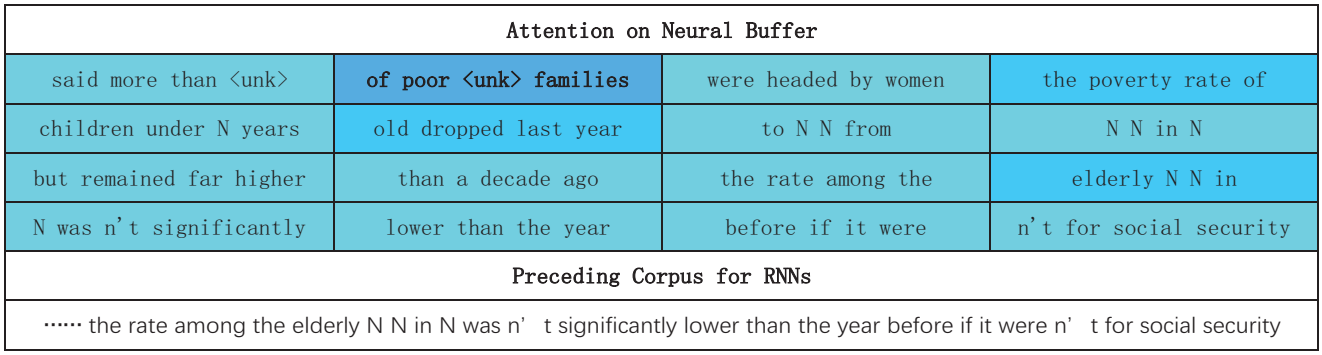

Figure 5: A snapshot of sequential neural buffer and the preceding words fed into its underlying sequence model. The special tokens such as <unk> (out of vocabulary) or N (digit number) are results from preprocessing in (Mikolov et al. 2010).

Method	Top 4 candidates for prediction			
SNB only	payment	assist	money	sponsor
Sequential RNN only	tax	office	number	payment
Sequental RNN + SNB	payment	assist	money	tax

Table 4: The predicted words from SNB only, its underlying sequential RNN only, and the proposed model with a combination of Sequential RNN and SNB. They are ranked in likelihood in descending order.

Hyper-parameter	PTB	WT2
training temperature	100	10
evaluation temperature	0.1	0.01
span length L	8	4
buffer size B	2048	512

Table 5: Table for hyper-parameters on Penn Treebank and WikiText-2 dataset.

"elderly NN in". In contrast, the words predicted from the underlying sequential RNN are closely related to the preceding 2-gram "social security", but are less relevant to the whole sentence.

More details

Table 5 lists the hyper-parameter settings.

Related Work

Recent best-performing neural sequence model for language modeling use a variety of regularization and optimization techniques (Press and Wolf 2017; Inan, Khosravi, and Socher 2016). In particular, AWD-LSTM (Merity, Keskar, and Socher 2017) improves drop-out for regularization and gradient descent for optimization. Adversarial training is applied in (Wang, Gong, and Liu 2019) to improve generalization. However, all of them are not directly addressing the structural *sequential recency bias* problem.

One direction of using long distance context is through grammar induction. It aims at extracting underlying grammar from corpus without treebank annotation (Shen et al. 2017; Yogatama et al. 2018; Shen et al. 2018). Integrating the tree shall benefits utilization of long-distance context. However, its performance is still far behind sequence model for sequence prediction tasks.

Neural cache model in (Grave, Joulin, and Usunier 2016) is a non-parametric method to increase probability to reoccurring patterns during test. We observe that it performs worse than dynamic evaluation (Krause et al. 2017), which is also a method applied during test. Since parameters in our model are obtained during training, our method is orthogonal to these methods.

A recent work of transformer-xl (Dai et al. 2019) extends transformer architecture (Vaswani et al. 2017) to model dependency beyond fixed lengths, using a recurrence mechanism on adjacent segments. However, the transformer-xl relies on information from long-distance context to be kept in the recurrence mechanism. In contrast, the proposed sequential neural buffer directly accesses long-distance context without using any time-recurrent mechanism. Nevertheless, the proposed SNB can be used together with transformer, in which transformer is used similarly as the LSTM to encode local information.

Conclusion

This paper presents a neural architecture to enable neural sequence model to efficiently and effectively utilize long-distance context. It achieves the best perplexity results ever reported in standard evaluation settings. A key aspect of the proposed architecture is that it addresses the *sequential recency bias* problem by using a span-based representation of long-distance context, and a policy network to selectively use local and long-distance context. A novel training algorithm is also crucially important for success: we use both annealed maximum likelihood estimation and an intrinsic reward-driven learning to deal with *simple sentence dominance* problem during training.

Acknowledgments

This work was done while the first author did internship at Ant Financial. We also thank anonymous reviewers for valu-

able suggestions.

References

Dai, Z.; Yang, Z.; Yang, Y.; Le, J. C. Q. V.; and Salakhutdinov1, R. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*.

Daniluk, M.; Rocktäschel, T.; Welbl, J.; and Riedel, S. 2017. Frustratingly short attention spans in neural language modeling. *arXiv preprint arXiv:1702.04521*.

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

Inan, H.; Khosravi, K.; and Socher, R. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Khandelwal, U.; He, H.; Qi, P.; and Jurafsky, D. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 284–294. Melbourne, Australia: Association for Computational Linguistics.

Krause, B.; Kahembwe, E.; Murray, I.; and Renals, S. 2017. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*.

Kuncoro, A.; Dyer, C.; Hale, J.; Yogatama, D.; Clark, S.; and Blunsom, P. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1426–1436.

Marvin, R., and Linzen, T. 2018. Targeted syntactic evaluation of language models. *arXiv preprint arXiv:1808.09031*.

Merity, S.; Keskar, N. S.; and Socher, R. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Nelson, M. J.; Karoui, I. E.; Giber, K.; Yang, X.; Cohen, L.; Koopman, H.; Cash, S. S.; Naccache, L.; Hale, J. T.; Pallier, C.; and Dehaene, S. 2017. Neurophysiological dynamics of phrase-structure building during sentence processing. *Proceedings of the National Academy of Sciences of the United States of America* 114.

Press, O., and Wolf, L. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 157–163. Valencia, Spain: Association for Computational Linguistics.

Press, O. 2019. Partially shuffling the training data to improve language models. *arXiv preprint arXiv:1903.04167*.

Shen, Y.; Lin, Z.; Huang, C.-W.; and Courville, A. 2017. Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

Shen, Y.; Tan, S.; Sordoni, A.; and Courville, A. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Stern, M.; Andreas, J.; and Klein, D. 2017. A minimal span-based neural constituency parser. *arXiv preprint arXiv:1705.03919*.

Tran, K.; Bisazza, A.; and Monz, C. 2016. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 321–331. San Diego, California: Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Wang, W., and Chang, B. 2016. Graph-based dependency parsing with bidirectional LSTM. In *ACL*.

Wang, D.; Gong, C.; and Liu, Q. 2019. Improving neural language modeling via adversarial training. *arXiv preprint arXiv:1906.03805*.

Weaver, L., and Tao, N. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 538–545. Morgan Kaufmann Publishers Inc.

Williams, A.; Drozdov, A.; and Bowman, S. R. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics* 6:253–267.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Wu, S.; Zhang, D.; Yang, N.; Li, M.; and Zhou, M. 2017. Sequence-to-dependency neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 698–707.

Yang, Z.; Dai, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

Yogatama, D.; Miao, Y.; Melis, G.; Ling, W.; Kuncoro, A.; Dyer, C.; and Blunsom, P. 2018. Memory architectures in recurrent neural network language models.