CONTEXT-AWARE TEMPERATURE FOR LANGUAGE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Current practices to apply temperature scaling assume either a fixed, or a manually-crafted dynamically changing schedule. However, our studies indicate that the individual optimal trajectory for each class can change with the context. To this end, we propose context-aware temperature, a generalized approach to provide an individual optimal temperature trajectory over the context for each vocabulary, while allowing the temperature to be learned along with the remaining model parameters during training. Experiment results confirm that the proposed method significantly improves state-of-the-art language models, achieving a perplexity of 19.90 on Penn Treebank, 33.88 on WikiText-2, and 4.7 on WikiText-103.[1]

1 INTRODUCTION

Temperature scaling has been widely used in various domains, such as natural language processing, model calibration, and knowledge distillation, to adjust the smoothness of a distribution, and achieve great performances or control the attribute of generated outputs (Norouzi et al., 2016; Hu et al., 2017; Caccia et al., 2018). Generally, a temperature scalar (or vector) τ is applied as a denominator to the logits, and then the divided logits pass through a Softmax layer to yield a probability distribution. If the temperature value $\tau \to \infty$, the distribution becomes more uniform, thus increasing the uncertainty. Contrarily, when $\tau \to 0$, the distribution collapses to a point mass. Although temperature scaling has been justified to achieve great success, existing implementations are limited. They assume either a constant value, or a manually-defined schedule for the temperature (Jang et al., 2017; Ma et al., 2017; Xie et al., 2019). Most importantly, none of them studies the effects on different word tokens when the temperature changes.

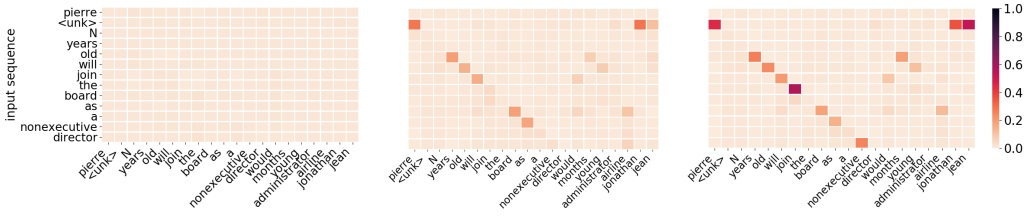
To this end, we propose *context-aware temperature*, which is capable of generating an unique temperature value for each token. Through taking the change of contexts into consideration, context-aware temperature serves as a generalized method to provide an optimal temperature for each word. Figure 1 illustrates the temperature trajectories of the tokens during the course of training. The word tokens along the Y-axis are from an input sentence, and the tokens along the X-axis are a selected subset of the vocabulary. At each row i, we show the contextual temperature values generated for the tokens along the X-axis, taking the context of the i-th input token in consideration. As shown in the figure, the temperatures of tokens gradually "heat up" or "cool down" as we sequentially process the input tokens. For instance, at row 5 (for the input token "old"), we see that "years" and "months" have obviously heated-up temperatures, while the temperature of "old" is slightly cooling down. These temperatures at row 5 are generated by our proposed model, considering the context of preceding tokens "pierre <unk> N years", with the goal of predicting the token "old".

Furthermore, the diagonal pattern indicates that our context-aware model assigns high temperature values to the token that just occurred, effectively suppressing the same token to appear repeatedly in a sentence. Such context-aware behavior of our model intuitively matches human intuition that words do not usually occur consecutively in a sentence. With these observations, we argue that existing methods limit themselves to some fixed schedules, and thus have great difficulty to generalize. The proposed context-aware temperature provides a more generalized temperature mechanism.

Our experimental results on language modeling datasets, including Penn Treebank, WikiText-2 and WikiText-103, demonstrate significant improvements. We also conduct comprehensive analyses

[1] We will release the code upon paper acceptance.

and ablation studies to confirm the improvements of the proposed method. The proposed method is capable of regulating the uncertainties as the context changes, allowing language models to achieve much better performances. To the best of our knowledge, this is the first systematic work that studies the role of temperature changing over contexts on a per-token basis.



(a) Epoch 11. (b) Epoch 351. (c) Epoch 1000.

Figure 1: The learned temperature trajectories of the tokens during the course of training.

2 Related Work

2.1 Language Modeling

Given an input sequence $x_{1:T} = x_1, ..., x_T$ of tokens $x_t \in V$, with the sequence length T and the vocabulary V, the objective of language modeling is to predict the next token x_{T+1}. A language model factorizes $P(x_{1:T+1})$, the joint probability of $x_{1:T+1}$, as the product of a series of conditional probabilities $P(x_i|x_{i-1})$, $i = 0, ..., T+1$. Specifically, $P(x_{1:T+1}) = \prod_{t=1}^{T+1} P(x_t|x_{1:t-1})$.

Modern neural language models are comprised of two parts: a mapping function and a probability function. A mapping function $\theta^{emb} \in \mathbb{R}^{|V| \times d}$ maps a token x_t into a real-value vector $x_t^{emb} \in \mathbb{R}^d$, where d is the dimension of the vector and $|V|$ is the size of vocabulary. A probability function then converts x_t^{emb} into a $|V|$ dimensional vector using a weight matrix $\theta^{model} \in \mathbb{R}^{d \times |V|}$. Afterwards, a Softmax function σ is used to obtain a probability distribution. So, at timestep t, we have

$$P(x_t|x_{1:t-1}) = \sigma(f(x_{1:t-1}; \theta^{emb} \cup \theta^{model})) \tag{1}$$

where f is a nonlinear mapping parameterized by θ^{emb} and θ^{model}. Current language models adopt various architectures as θ^{model}, for instance, recurrent-based networks (Hochreiter & Schmidhuber, 1997), convolution-based ones (Dauphin et al., 2017), and attention-based ones (Dai et al., 2019).

2.2 Softmax Layer

A Softmax layer σ normalizes a $|V|$ dimensional, real-value vector $\mathbf{z}$ to make it sum to 1. Specifically, $\sigma(\mathbf{z})_i = z_i / \sum_j^{|V|} e^{z_j}$, where $\mathbf{z} = f(x_{1:t-1}; \theta^{emb} \cup \theta^{model})$ and z_i is the i-th element in $\mathbf{z}$.

Recent progress suggests that Mixture of Softmaxes (MoS) (Yang et al., 2018) significantly improves the performance by computing multiple Softmax distributions, and summing them up through a set of weights to provide the final probability distribution. Specifically, a set of M matrices W_m is applied to $\mathbf{z}$, that is, $\mathbf{z}_m = \mathbf{z}^T \cdot W_m$. The probability distribution under the MoS model is thus

$$P_{MoS}(x_t|x_{1:t-1}; \Theta) = \sum_{m=1}^{M} \pi_m \cdot \sigma(\mathbf{z}_m) \tag{2}$$

where $\Theta = \cup_{m=1}^{M} W_m \cup \theta^{emb} \cup \theta^{model}$.

2.3 Temperature Scaling

Temperature, denoted as $\boldsymbol{\tau}$, often serves as a hyper-parameter in the Softmax layer to control the smoothness of the distribution. Applying temperature scaling on logits $\mathbf{z}$ gives

$$P(x_t|x_{1:t-1}; \theta^{emb}, \theta^{model}, \boldsymbol{\tau}) = \sigma(\mathbf{z}/\boldsymbol{\tau}) \tag{3}$$

Here $\boldsymbol{\tau}$ is a vector with a same dimension $|V|$ as $\mathbf{z}$, and is normally set to 1. Below we divide related works into three categories: (a) constant temperature, where each element in $\boldsymbol{\tau}$ has the same value, (b) dynamic temperature over training iterations, where $\boldsymbol{\tau}$ is constant in one single iteration, but has different values in every iteration, and (c) dynamic temperature over word position, where $\boldsymbol{\tau}$ is dynamic in one iteration, and besides, $\boldsymbol{\tau}$ could change in every iteration.

Constant Temperature. Earlier works can be traced back to model distillation (Hinton et al., 2014), where $\boldsymbol{\tau}$ is constant and chosen empirically. Constant temperature is also used during training (Norouzi et al., 2016; Ma et al., 2017) to control the degree of augmentation. Other works that incorporate τ at inference time include model calibration (Guo et al., 2017), and controlling the trade-off between quality and diversity in text generation tasks (Caccia et al., 2018).

Dynamic Temperature Over Training Iterations. Most works adopt dynamic temperature through a manually-crafted schedule. Notably, Hu et al. (2017) uses an approximation based on Softmax with a decreasing temperature to enable gradient propagation. Similar techniques are adopted in gumbel-softmax (Jang et al., 2017). In addition, Zhang et al. (2018) shows that with a heating up temperature scaling, embedding vectors are more compact.

Dynamic Temperature Over Attention. Another work that is closely related to our work, is the adaptive temperature over an attention model (Lin et al., 2018). The model learns to output a dynamic temperature to control the softness of the attention. The temperature is learned based on the information of decoding at the current step and the attention in previous steps.

3 METHODS

3.1 CONTEXT-AWARE TEMPERATURE

Context-aware temperature is a mechanism that chooses the optimal temperature for each vocabulary token, by considering the "context" of a token x_t, which is the history $x_{1:t-1}$.

In this work, the temperature vector $\boldsymbol{\tau} \in \mathbb{R}^{|V|}$ is generated based on the hidden representations learned from a non-linear mapping function f (as described in Equation 1). The function f can be any sequential models. In this work, we explore two parameterizations of the function f, based on two different models, AWD-LSTM (Merity et al., 2018) and Transformer-XL (Dai et al., 2019). As an example, Figure 2 illustrates the architecture of our model when implemented on top of the MoS model (Yang et al., 2018). In this figure, the above-mentioned function f is the output of the AWD-LSTM layer, and the components colored in blue constitute the proposed mechanism of context-aware temperature, which we will describe next.

The output from f is then multiplied with W_τ to produce the logits of the temperatures, denoted as $\mathbf{z}_\tau$, which subsequently are scaled using a Softmax layer and two parameters (α,β):

$$\tau = \frac{\sigma(\mathbf{z}_\tau) + \alpha}{\beta}, \text{ where } \mathbf{z}_\tau = f(x_{1:t-1}; \theta^{emb}, \theta^{model})^T \cdot W_\tau. \tag{4}$$

Note that while the values of (α, β) could be assigned manually, they could be learnt by the model itself. We choose to implement them as learnable parameters in this work.

3.2 CONTEXT-AWARE TEMPERATURE MOS

The first model that we explored is the *Context-aware Temperature Mixture of Softmaxes (CT-MoS)*, which extends the MoS model for language modeling with our proposed context-aware temperature mechanism. Equation 5 shows the modified mechanism with the context-aware temperatures.

$$P_{\text{CT-MoS}}(x_t = k | x_{1:t-1}; \Theta) = \sum_{m}^{M} \pi_m \cdot \sigma(\mathbf{z}_m \oslash \boldsymbol{\tau})) \tag{5}$$

where $\oslash$ represents element-wise division. $\Theta = \cup_{m=1}^{M} W_m \cup \theta^{emb} \cup \theta^{model} \cup W_\tau$ is the set of all of the parameters of CT-MoS. And, π_m are the weights for the logits, which are defined by the

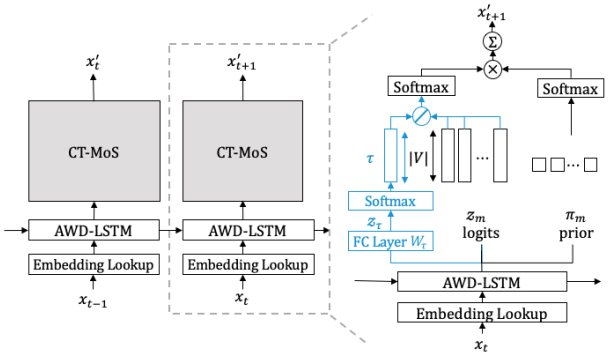


Figure 2: The architecture of the proposed CT-MoS model. Black components are those the same as the MoS model, while the blue ones are the newly added ones in the proposed approach.

MoS model. Please note that using a single matrix W_τ may increase the number of parameters significantly, and thus in practice we factorize it into two matrices W_{τ_1} and W_{τ_2}. Figure 2 highlights the difference between the proposed CT-MoS model and the MoS model.

Compared to prior works, the proposed context-aware temperature is not only dynamic over training iterations, but also dynamic over the word positions in a sentence (see Figure 1). That is, for the same token at different positions with different context, the proposed method would learn a different temperature vector dependent on the token's history context.

3.3 Context-Aware Temperature Transformer-XL

Besides experimenting with the MoS model, we apply the context-aware temperature mechanism on the Transformer-XL (Dai et al., 2019) model for language modeling. Compared to MoS, Transformer-XL is a purely self-attentive model. We apply the context-aware temperature on the logits $\mathbf{z}_{\text{TXL}}$ of the Transformer-XL model, and denote the resulting model as "CT-Transformer-XL".

$$P_{\text{CT-Transformer-XL}}(x_t = k | x_{1:t-1}; \Theta) = \sigma(\mathbf{z}_{\text{TXL}} \oslash \boldsymbol{\tau})) \tag{6}$$

where $\Theta = \theta_{emb} \cup \theta_{model} \cup W_\tau$.

3.4 Effects of Context-Aware Temperature

In this section, we discuss how context-aware temperature effects the logits $\mathbf{z}$ and the temperature itself $\boldsymbol{\tau}$, through illustrating their corresponding gradients. We consider a language modeling task with a small vocabulary of only two words, i.e., $|V| = 2$. In this setting, the dimensionality of logits $\mathbf{z}$ is 2 and so is that of the temperature vector $\boldsymbol{\tau}$. The range of $\boldsymbol{\tau}$ is set to $[0, 1]$, that is, $(\alpha, \beta) = (0, 1)$.

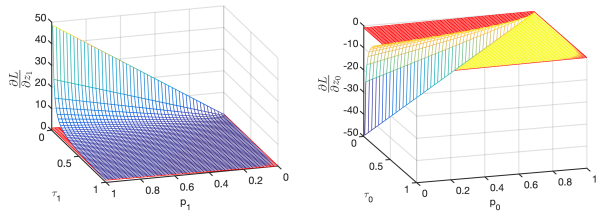
Gradient of the logit At a given input, assume that the ground-truth token is $i = 0$, thus the cross-entropy loss is $L = -\ln p_0$, where p_0 is the model's output probability for word token $i = 0$. In this case, the gradients of logits $\mathbf{z}$ are illustrated in Figure 3 (the derivation of the gradients is in Appendix A.1). In Figure 3(a), we consider the gradient of z_1 of token $i = 1$. When no temperature mechanism is applied, it can be seen that the magnitude of the gradient is bounded within [0, 1] (as shown by the red mesh), with the largest magnitude 1 (most aggressive update) happens when the probability p_1 is closer to 1 (note that the ideal value for p_1 should be close to 0, as the other token $i = 0$ is assumed to be the ground truth).

On the other hand, when the context-aware temperature is applied, the gradient can now be set to a much more substantial magnitude by setting a smaller temperature value τ_1 (as shown by the blue mesh). The additional flexibility of magnifying the gradient by temperature enables the model to be more efficient (or aggressive) in adjusting the parameters to reduce the training loss.

Gradient of the temperature Next, we analyze how the model with context-aware temperature updates the value of the temperature. We recall that Equation 4 defines the formulation of the

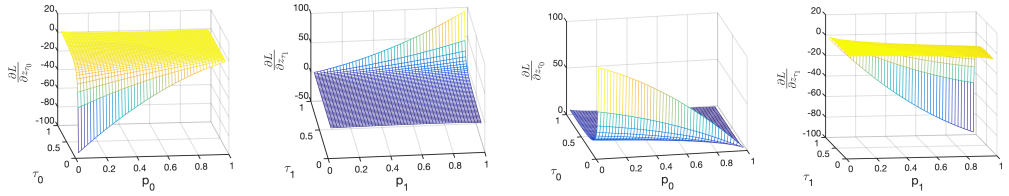
temperature. We leave the detailed derivation of the gradient of the temperature in Appendix A.1. Figure 4 visualizes the gradients $\frac{\partial L}{\partial z_{\tau_0}}$ and $\frac{\partial L}{\partial z_{\tau_1}}$ when z_0 is either positive or negative. We will use the case in Figure 4(a) (where $z_0 \in \mathbb{R}^-$) as an example to explain the effect of the gradient on the temperature. In this case, if p_0 is close to 0, then we expect that the model should be more aggressive to update the parameters (since we assume that the ground-truth class is $i = 0$, meaning that p_0 should ideally be close to 1). This aggressive response is indeed visible in the figure, showing values of larger magnitude when $p_0 \to 0$.

The amount of update on the temperature τ_0 also depends on its current value. In the same Figure 4(a), let's consider a fixed p_0, say, $p_0 = 0.1$. In this case, as described above, the model wants to increase the value of p_0 to closer to 1. To do that, the model will attempt to increase the term e^{z_0/τ_0}. Since τ_0 is a positive value in [0, 1] and $z_0 < 0$ in this case, to maximize z_0/τ_0, the optimal is to have the temperature value $\tau_0 \to 1$. When the model is updating the value of τ_0, if its current value is already close to 1, then the gradient will be small, since it is already close to the optimal value. On the other hand, if the temperature τ_0 is still far from 1, then the gradient will update it's value more aggressively. This behavior is exactly visualized in the figure.



(a) $i = 1$ (b) $i = 0$

Figure 3: Gradients of loss with respect to (a) logit z_0 and (b) logit z_1. In each figure, the x-axis is the probability p_i of class i, y-axis is the temperature value τ_i of class i, and z-axis is the gradient $\frac{\partial L}{\partial z_i}$. The colorful mesh represents the gradients when the context-aware temperature is applied, while the red mesh represents the case without temperature.



(a) $z_0 \in \mathbb{R}^-, i = 0$ (b) $z_0 \in \mathbb{R}^-, i = 1$ (c) $z_0 \in \mathbb{R}^+, i = 0$ (d) $z_0 \in \mathbb{R}^+, i = 1$

Figure 4: Gradients of loss with respect to z_{τ_i}. The x-axis is the probability p_i of class i, y-axis is the temperature value τ_i of class i, and z-axis is the gradient $\frac{\partial L}{\partial z_{\tau_i}}$.

4 Experiments

4.1 Experimental Setups

We evaluate context-aware temperature on three datasets for language modeling: Penn Treebank, WikiText-2 and WikiText-103 (Marcus et al., 1993; Mikolov et al., 2011; Merity et al., 2017).

CT-MoS. We conduct experiments on PTB and WT2 using one and four 1080 Ti GPUs, respectively. The environment is PyTorch (Paszke et al., 2017). We follow the training configurations as reported in the MoS paper and their github[2]. For both data sets, we use the same number of param-

[2] https://github.com/zihangdai/mos

eters as MoS. That is, we use three layers of LSTM with embedding sizes of 960-960-620 on PTB; and for WT2, three layers of LSTM with embedding sizes of 1150-1150-650. The only difference is on the choice of the optimizer, where we use the Adam optimizer with a learning rate of $1e^{-4}$.

CT Transformer-XL. The experiments on WikiText-103 is conducted with four 1080 Ti GPUs. The baseline model is Transformer-XL, and the environment we use is PyTorch. We follow the training configurations as reported in the official Transformer-XL github[3]. There are two training configurations: standard and large. We adopt the standard one, which has 16 layers, 10 attention heads and the hidden size of 410. Adaptive input representations (Baevski & Auli, 2018) and adaptive softmax (Grave et al., 2016) are used.

4.2 Main Results

Penn Treebank. We compare the proposed method with MoS and AWD-LSTM, including the performance with and without dynamic evaluation (Krause et al., 2017). Since the original MoS model has approximately 22M parameters, to make a fair comparison, we augment its number of parameters to have 24M parameters. We increase the size of each layer proportionally, giving the word embedding size $d = 300$, and making the sizes of the three LSTM layers be 1030, 1030 and 670. We denote this augmented model as MoS+.

In Table 1, experimental results show that our CT-MoS model outperforms AWD-LSTM, MoS, and MoS+ on both validation and test sets. Our model achieves 22.92 perplexity on the validation set and 19.90 on the test set without dynamic evaluation. When using dynamic evaluation, the proposed CT-MoS model also achieves significant better perplexities of 10.63 and 10.04.

Table 1: Perplexity comparison on the Penn Treebank dataset. † indicates using dynamic evaluation.

Model	Paras	Validation	Test
AWD-LSTM	24M	60.7	58.8
AWD-LSTM †	24M	51.6	51.1
MoS	22M	58.08	55.97
MoS †	22M	48.33	47.69
MoS+	24M	57.96	55.78
MoS+ †	24M	48.89	48.28
CT-MoS	24M	**22.92**	**19.90**
CT-MoS †	24M	**10.63**	**10.04**

WikiText-2. Table 2 presents our experimental results for WT2. Similar to the results in Table 1, CT-MoS also outperforms the state-of-the-art models in this case. Compared to the MoS model, CT-MoS achieves great improvements, with perplexity 34.81 and 33.88 on the validation and test sets, respectively. With dynamic evaluation, CT-MoS obtains even better perplexity of 14.51 and 13.93. To conduct a fair comparison, we enlarge the number of parameters of MoS to have 45M parameters. The size of each layer is increased proportionally: the embedding size is $d = 360$ and sizes of the three LSTM layers are 1320-1320-760. We denote the enlarged model as WT2+.

WikiText-103. Table 3 shows the experimental results on the WikiText-103 data set. We report the performances of both standard and large Transformer-XL models. The better results achieved by CT-Transformer-XL not only indicates the effectiveness of context-aware temperature, but also demonstrates its applicability on modeling the long-term dependency in an article-based corpus.

4.3 Ablation Studies

To further illustrate the effects of the context-aware temperature, we compare it with conventional temperature scaling method, that is, using a constant temperature. We experiment with different

[3] https://github.com/kimiyoung/transformer-xl

Table 2: Perplexity comparison on the WikiText-2 dataset. † indicates using dynamic evaluation.

Model	Paras	Validation	Test
AWD-LSTM	33M	69.1	66.0
AWD-LSTM †	33M	46.4	44.3
MoS	35M	66.01	63.33
MoS †	35M	42.41	40.68
MoS+	45M	65.33	62.66
MoS+ †	45M	42.73	40.74
CT-MoS	45M	**34.81**	**33.88**
CT-MoS †	45M	**14.51**	**13.93**

Table 3: Perplexity comparison on the WikiText-103 dataset.

Model	Paras	Validation	Test
Transformer XL Standard	151M	24.0	23.1
Transformer-XL Large	257M	18.3	18.2
CT Transformer-XL	261M	**4.1**	**4.7**

constant values $\{0.5,1,2,4\}$ on the MoS model. Results in Table 4 show that employing context-aware temperature provides better performance than the conventional method. This indicates that the proposed method indeed has merits on adjusting the model parameters, by considering the context and providing a dynamic and optimized temperature value for each token.

Table 4: Effects of context-awareness. Experiments are conducted on the PTB dataset.

Model	Validation	Test
MoS ($\tau = 0.5$)	60.73	58.38
MoS ($\tau = 1.0$)	58.08	55.97
MoS ($\tau = 2.0$)	57.25	55.11
MoS ($\tau = 4.0$)	57.39	55.21
CT-MoS	**22.92**	**19.90**

Furthermore, we conduct two ablation studies regarding (a) the model size, and (b) the value range of the temperature vector. Details are left in Appendix A.2 and A.3, respectively.

4.4 Analysis

Language Uncertainties Another interesting observation of the proposed context-aware temperature is the correlation between a word's relative position in a sentence and its learned temperature. Figure 5 shows the means of temperature vectors at different positions in a sentence. We analyze sentences of three different lengths. For instance, the purple line corresponds to our analysis on sentences that are longer than 21 words and shorter than 35 words. For each of these sentences, we consider positions in three disjoint segments: the first 7, middle 7 and last 7 words. The three segments are concatenated to form a "normalized" sentence. This pre-processing ensures the positions of a token only have relative effects to the analysis, and are not effected by the sentence length.

In Figure 5, we observe that the temperature value is low at beginning positions of a sentence. As the position gets further away, the averaged temperature value first has a sharp increase and then decreases at positions near the end. Our intuition is as the following: at the beginning positions the model has little confidence, since there is limited information in the history. The model recognizes this fact and learns a uniform temperature vector over the tokens, to be less assertive and assigns relatively uniform probability over the tokens. As the history builds up at later positions, the model

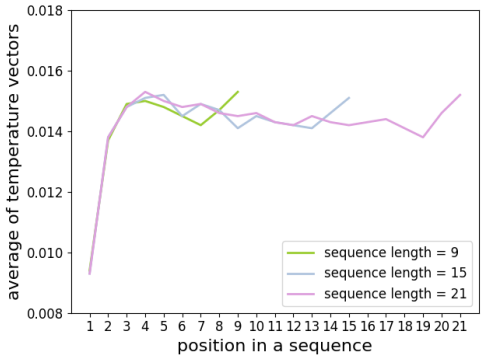


Figure 5: Means of the temperature vectors over positions in a sentence. The average temperature is low at the beginning of a sentence and gradually increases towards latter positions.

Table 5: Analysis of model performance on a sample from the PTB dataset. τ denotes the temperature of a certain token. More examples can be found in Appendix A.4.

Reference	$ N million of general obligation **veterans** ' **tax** notes series N via competitive bid		
CT-MoS	$ N million of general obligation **veterans** ' **tax** notes series N via a bid		
MoS	$ N million of general obligation **bonds** bonds **bonds** bonds series N via a bid		
CT-MoS top-3	**veterans 0.874**	tax 0.082	<unk> 0.034
MoS top-3	**bonds 0.495**	revenue 0.062	**veterans 0.031**
Temperature		**veterans** $\tau = 0.0011$	**bonds** $\tau = 0.0099$
CT-MoS top-3	**tax 0.943**	N 0.056	<unk> 0.001
MoS top-3	**bonds 0.135**	revenue 0.101	**tax 0.064**
Temperature		**tax** $\tau = 0.0063$	**bonds** $\tau = 0.0096$

becomes more confident about the next token and outputs a more spiky probability distribution on plausible tokens. The formation of the spiky distribution is done by having high temperature values for *implausible* tokens. These higher temperature values increases the average temperature value as shown in Figure 5. The uncertainties drop sharply as we reach the end of the sentence.

Case Study: Effectiveness. In Table 5, we present a sentence from the PTB dataset to illustrate the differences between MoS and CT-MoS, against the Reference. We highlight two differences in red and blue colors. At the location highlighted in red, we see that CT-MoS successfully predicts the answer "veterans" with a high probability of 0.874. On the other hand, MoS predicts "bonds", which deviates from the ground truth. The word "veterans" has a temperature of 0.0011, which is much smaller than that of the word "bonds" (0.0099). This contributes to CT-MoS chooses "veterans" over "bonds". Similar observations can be found at the blue location. We also observe that MoS tends to output a same word repeatedly, such as the four "bonds" tokens predicted for this example. The context-aware temperature mechanism does not have this issue. As illustrated in Figure 1, the context-aware temperature inclines to suppress the token appearing at previous time step by raising the temperature value, which effectively discouraging the same token to be predicted consecutively.

5 Conclusion and Future Work

We propose a fully automated temperature mechanism, which learns an optimal temperature for each individual token based on the history of the context. The proposed *context-aware temperature* obtains strong results on various widely-adopted language modeling datasets. Our work opens up the research directions along the line of fully automated temperature mechanism in various NLP tasks, such as summarization, machine translation, and dialogue generation.

REFERENCES

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language gans falling short. In *NIPS Workshop*, 2018.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, 2019.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 933–941. JMLR, 2017.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for gpus. *arXiv perprint arXiv:1609.04309*, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural netowrks. In *ICML*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS*, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward controlled generation of text. In *ICML*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*, 2017.

Junyang Lin, Xu Sun, Xuancheng Ren, Muyu Li, and Qi Su. Learning when to concentrate or divert attention: Self-adaptive attention temperature for neural machine translation. In *EMNLP*, 2018.

Xuezhe Ma, Pengcheng Yin, Jingzhou Liu, Graham Neubig, and Eduard Hovy. Softmax q-distribution estimation for structured prediction: A theoretical interpretation for raml. *arXiv preprint arXiv:1705.07136*, 2017.

Mitchell P Marcus, Mary Ann Marcinkiewicz, , and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. In *Computational linguistics*, 1993.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. In *ICLR*, 2018.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černockỳ. Empirical evaluation and combination of advanced language modeling techniques. In *INTERSPEECH*, 2011.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. Reward augmented maximum likelihood for neural structured prediction. In *NIPS*, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. In *ICLR*, 2019.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. Breaking the softmax bottleneck: A high-rank RNN language model. In *ICLR*, 2018.

Xu Zhang, Felix Xinnan Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*, 2018.

A Appendix

A.1 Partial Derivatives of Loss to Logits of Temperature

Take the case of two classes as example, assume that the ground-truth class is $i = 0$. In this case, the loss L is $-\ln p_0$, where p_0 is the probability of class 0 output by the model. The probabilities of the two classes are $\mathbf{p} = \sigma(\mathbf{z} \oslash \boldsymbol{\tau}) = [p_0, p_1]$. Let $\mathbf{u} = \mathbf{z} \oslash \boldsymbol{\tau} = [u_0, u_1]$, and $u_i = z_i/\tau_i$. Then, we have

$$p_i = \frac{e^{u_i}}{e^{u_0} + e^{u_1}}. \tag{7}$$

$\boldsymbol{\tau}$ is defined in Equation 4, and is essentially $\sigma(\mathbf{z}_\tau)$. Therefore,

$$\tau_i = \frac{e^{z_{\tau_i}}}{e^{z_{\tau_0}} + e^{z_{\tau_1}}}. \tag{8}$$

The gradients of the loss with respect to logits z_0 and z_1 are

$$\frac{\partial L}{\partial z_i} = \frac{\partial(-\ln p_0)}{\partial z_i} = \begin{cases} (p_i - 1)\frac{1}{\tau_i} & i = 0 \\ p_i \frac{1}{\tau_i} & i \neq 0 \end{cases} \tag{9}$$

The gradient of z_{τ_0} is calculated as below

$$\begin{aligned} \frac{\partial L}{\partial z_{\tau_0}} &= \frac{\partial L}{\partial p_0}\frac{\partial p_0}{\partial u_0}\frac{\partial u_0}{\partial \tau_0}\frac{\partial \tau_0}{\partial z_{\tau_0}} + \frac{\partial L}{\partial p_0}\frac{\partial p_0}{\partial u_1}\frac{\partial u_1}{\partial \tau_1}\frac{\partial \tau_1}{\partial z_{\tau_0}} \\ &= -p_1\frac{-z_0}{\tau_0^2}\frac{e^{z_{\tau_0}+z_{\tau_1}}}{(e^{z_{\tau_0}} + e^{z_{\tau_1}})^2} + p_1\frac{-z_1}{\tau_1^2}\frac{-e^{z_{\tau_0}+z_{\tau_1}}}{(e^{z_{\tau_0}} + e^{z_{\tau_1}})^2} \\ &= \frac{1}{\tau_0}p_1 z_0 \tau_1 + \frac{1}{\tau_1}p_1 z_1 \tau_0 \end{aligned} \tag{10}$$

Similarly, the gradient of z_{τ_1} is calculated as

$$\begin{aligned} \frac{\partial L}{\partial z_{\tau_1}} &= \frac{\partial L}{\partial p_0}\frac{\partial p_0}{\partial u_0}\frac{\partial u_0}{\partial \tau_0}\frac{\partial \tau_0}{\partial z_{\tau_1}} + \frac{\partial L}{\partial p_0}\frac{\partial p_1}{\partial u_1}\frac{\partial u_1}{\partial \tau_1}\frac{\partial \tau_1}{\partial z_{\tau_1}} \\ &= -p_1\frac{-z_0}{\tau_0^2}\frac{-e^{z_{\tau_0}+z_{\tau_1}}}{(e^{z_{\tau_0}} + e^{z_{\tau_1}})^2} + p_1\frac{-z_1}{\tau_1^2}\frac{e^{z_{\tau_0}+z_{\tau_1}}}{(e^{z_{\tau_0}} + e^{z_{\tau_1}})^2} \\ &= -\frac{1}{\tau_0}p_1 z_0 \tau_1 - \frac{1}{\tau_1}p_1 z_1 \tau_0 \end{aligned} \tag{11}$$

A.2 Model Size

We have demonstrated the results of MoS+ in both Table 1 and 2. Here we notice that MoS+ has a similar perplexity compared to MoS, indicating that directly increasing model parameters cannot improve the performance. Similar observation and results are also reported by Yang et al. (2018). This shows that the improvements brought by CT-MoS are more than the mere growth of parameters.

As for Transformer-XL, two results achieved by different sizes of the model are reported. We compare with the large one (257M parameters), which has a comparable parameter size to our model (261M parameters). Results in Table 6 demonstrate the significant improvements brought by the proposed method. Again, this shows that the mere growth of parameters do not account for the improvements. In addition, we report the performance of several state-of-the-art models, including Megatron-LM (Shoeybi et al., 2019) and the widely-known GPT-2 (Radford et al., 2019) models. Please note that these models use extra training data during the training procedure. However, the proposed model outperforms these baseline models without using any extra data.

Table 6: Perplexity comparison on the WikiText-103 dataset.

Model	Paras	Validation	Test	extra training data
Transformer XL Standard	151M	24.0	23.1	✗
Transformer-XL Large	257M	18.3	18.2	✗
CT Transformer-XL	261M	**4.1**	**4.7**	✗
Megatron-LM	8300M	-	10.81	✓
GPT-2 Full	1542M	-	17.48	✓
GPT-2 Medium	355M	-	26.37	✓

A.3 Temperature Normalization Range

The value range of the temperature vector is bounded by a Softmax layer and two parameters (α, β). Throughout all the experiments, we let (α, β) be learnable parameters and find their values to be (0,1). To further examine the learned value range, we conducted experiments with manually-defined temperature value ranges. The experiments are conducted on the Penn Treebank dataset.

Table 7 shows that the learned value ranges bring better performance than others. Even if we use a manually defined range that is same as the learned range $(0, 1)$, using the learned (α, β) slightly outperforms. Furthermore, the proposed method achieves great improvements when the temperature range is either wide or limited. For instance, using $(\alpha, \beta) = (1, 2)$ gives the range of $(2, 4)$ and the test perplexity of 41.09; using $(\alpha, \beta) = (0.2, 5)$ gives the range of $(1, 6)$ and the test perplexity of 22.12. Both results are obviously better than the test perplexity (47.69) of baseline MoS model.

Table 7: Performance of different value ranges of the temperature vector.

(α, β)	Validation	Test
$(0, 1)$	23.04	20.08
$(0.01, 2)$	23.12	20.09
$(0.2, 5)$	24.47	22.12
$(1, 3)$	43.97	41.09
$(1, 2)$	44.71	41.42
learned	**22.92**	**19.90**

A.4 Case Studies: Effectiveness

We present two other cases on the Penn Treebank dataset to visualize the effect of context-aware temperature, in Table 8 and 9. We highlight the five differences in red, brown, purple, blue and teal colors, respectively. Let's first see Table 8. At the colored locations, we see that the proposed model successfully predicts the correct words, while MoS fails to do so. When taking a look at the temperature value for the specified word, we see that CT-MoS gives the ground-truth tokens smaller temperature values, whereas giving the non ground-truth tokens (such as "the", "debt", "its" and "in") high temperature values. This contributes to the results that our model chooses the correct tokens over the wrong ones. Similar observations can be found in Table 9.

Table 8: Analysis of model performance on a sample from the PTB dataset.

Reference	**in august** resorts international inc. which sold more than $ N million of **junk** bonds suspended **interest payments**		
CT-MoS	**in august** the international inc. filed filed $ than $ N million of **junk** bonds under **interest payments**		
MoS	**the the** the had inc. said has N than N N billion of **debt** bonds to **its in**		
CT-MoS top-3	**in 0.413**	**the 0.163**	more 0.094
MoS top-3	**the 0.322**	**in 0.076**	a 0.044
Temperature		**in** $\tau = 0.0094$	**the** $\tau = 0.0122$
CT-MoS top-3	**august 0.418**	september 0.246	**the 0.153**
MoS top-3	**the 0.299**	addition 0.087	a 0.083
Temperature		**august** $\tau = 0.0043$	**the** $\tau = 0.0111$
CT-MoS top-3	**junk 0.677**	u.s. 0.107	the 0.066
MoS top-3	**debt 0.150**	assets 0.147	high-yield 0.099
Temperature		**junk** $\tau = 0.0056$	**debt** $\tau = 0.0096$
CT-MoS top-3	**interest 0.323**	**its 0.212**	from 0.099
MoS top-3	**its 0.273**	the 0.166	a 0.096
Temperature		**interest** $\tau = 0.0046$	**its** $\tau = 0.0068$
CT-MoS top-3	**payments 0.800**	**in 0.053**	rates 0.042
MoS top-3	**in 0.270**	**payments 0.240**	rates 0.072
Temperature		**payments** $\tau = 0.0038$	**in** $\tau = 0.0095$

One thing worth noticing is that, as mentioned in Section 4.4, the MoS model tends to predict a same word repeatedly. In Table 8, the MoS model consecutively predicts "the the the". This is again in accordance with our observations that the proposed CT-MoS model tends to give the previous token a high temperature value, so as to suppress the corresponding probability.

A.5 Case Study: Temperature and Token Position.

Another aspect to examine how context-aware temperature works is to look at the change of the temperature from a specific token across different positions in a sentence. In Table 10, we present a sample from the PTB dataset, and highlight the occurrences of the word "mortgage" in red. As the position changes, the proposed method chooses a different temperature value, adjusting its confidence of the model's belief. In this case, the temperature at the third occurrence of "mortgage" is 0.0023, and gradually decreases at subsequent occurrences. Such a decrease indicates that the model gains more confidence in making the prediction, most likely due to richer information from the longer history context.

Table 9: Analysis of model performance on a sample from the PTB dataset.

Reference	imperial **corp.** **based** in **san** diego is the **parent** of **imperial** savings & loan		
CT-MoS	imperial **corp.** **based** in **san** diego said attempting **parent** of **imperial** corp. & loan		
MoS	the **said** **said** in **<unk>** francisco said a **largest** of **the** bank & loan		
CT-MoS top-3	**corp. 0.443**	is 0.272	savings 0.104
MoS top-3	**said 0.189**	's 0.071	savings 0.067
Temperature		**corp.** $\tau = 0.0060$	**said** $\tau = 0.0073$
CT-MoS top-3	**based 0.370**	is 0.219	**said 0.207**
MoS top-3	**said 0.333**	a 0.071	's 0.070
Temperature		**based** $\tau = 0.0046$	**said** $\tau = 0.0089$
CT-MoS top-3	**san 0.718**	**<unk> 0.171**	imperial 0.029
MoS top-3	**<unk> 0.270**	new 0.097	los 0.070
Temperature		**san** $\tau = 0.0065$	**<unk>** $\tau = 0.103$
CT-MoS top-3	**parent 0.517**	first 0.364	<unk> 0.077
MoS top-3	**largest 0.161**	<unk> 0.097	parent 0.072
Temperature		**parent** $\tau = 0.0040$	**largest** $\tau = 0.0107$
CT-MoS top-3	**imperial 0.750**	**the 0.124**	<unk> 0.093
MoS top-3	**the 0.150**	<unk> 0.132	american 0.075
Temperature		**imperial** $\tau = 0.0012$	**the** $\tau = 0.0113$

Table 10: Analysis of the temperature for one specific word but at different positions.

CT-MoS	N N standard conventional fixed-rate **mortgages(1)** N N N N rate capped one-year adjustable rate **mortgages(2)** <eos> federal national **mortgage(3)** association fannie mae <eos> posted yields on N year **mortgage(4)** commitments for delivery within N days priced at par <eos> N N standard conventional fixed-rate **mortgages(5)** N N N rate capped one-year adjustable rate **mortgages(6)** <eos>					
Temperature τ	(1) 0.0020	(2) 0.0009	(3) 0.0023	(4) 0.0019	(5) 0.0010	(6) 0.0010