CONTEXTUAL TEMPERATURE FOR LANGUAGE MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Temperature scaling has been widely used to improve performance for NLP tasks that utilize Softmax decision layer. Current practices in using temperature either assume a fixed value or a dynamically changing temperature but with a fixed schedule. Little has been known on an optimal trajectory of temperature that can change with the context. In this paper, we propose *contextual temperature*, a mechanism that allows temperatures to change over the context for each vocabulary, and to co-adopt with model parameters during training. Experimental results illustrate that contextual temperature improves over state-of-the-art language models significantly. Our model CT-MoS achieved a perplexity of 55.31 in the test set of Penn Treebank and a perplexity of 62.89 in the test set of WikiText-2. The in-depth analysis shows that the behavior of temperature schedule varies dramatically by vocabulary. The optimal temperature trajectory drops as the context becomes longer to suppress uncertainties in language modeling. These evidence further justifies the need for contextual temperature and explains its performance advantage over fixed temperature or scheduling.

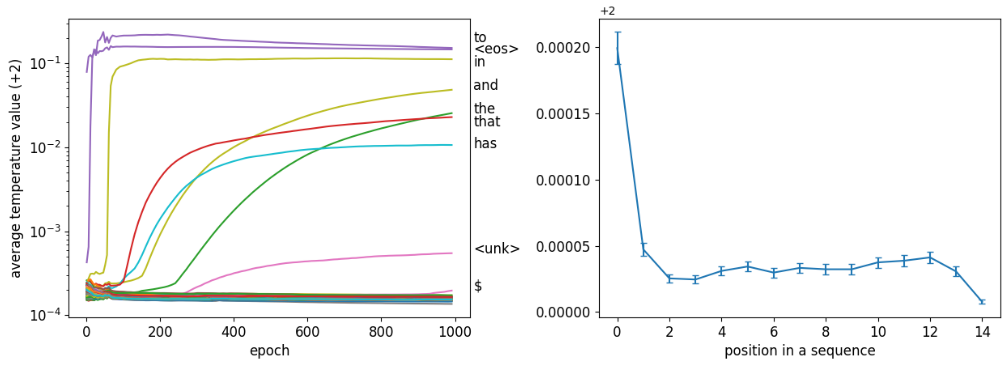
1 INTRODUCTION

The discrete nature of human language makes Softmax decision layer a must when converting learned representations into a sequence of linguistic tokens. A widely adopted technique in the natural language processing community is to apply a temperature as a denominator to the logits of the Softmax decision layer (Krizhevsky et al., 2012; Bahdanau et al., 2014; Hu et al., 2017; Caccia et al., 2018). The amount of temperature controls the smoothness of the Softmax distribution, with large temperature tends to make it uniform while small temperature makes the distribution spiky.

Although the temperature has been empirically justified to be useful, existing methods to use temperature are extremely limited. They either assume a constant temperature throughout training (Norouzi et al., 2016; Ma et al., 2017; Chen et al., 2019), or a fixed schedule that controls the temperature scaling (Hu et al., 2017). And most importantly, none of the exiting work explored the vocabulary differences when adjusting temperature. In reality, however, these temperatures can be dramatically different. Figure 1(a) showed us the optimized temperature during the course of training for our model. We can see from the figure that certain words have distinct temperature scaling that tends to be heating up while the majority of words have a temperature scaling that keeps cooling down. This makes existing methods that limited to fixed temperature schedule difficult to generalize. Another example can be seen from Figure 1(b), where the average temperature drops as the length of the history increases. This suggests that temperature mechanism helps to promote stochasticity early in the sentence while suppressing uncertainties when the context gets longer. All of these suggest a more generalized temperature mechanism that is capable of dealing with these phenomenons.

We propose *contextual temperature*, a method to generalize the use of temperature by making it a function of both the history of a sequence and the particular vocabulary. Contextual temperature optimizes the use of temperature that changes with the history. And is capable of generating a unique temperature for each vocabulary. As we parameterize contextual temperature using a deep neural network, its parameters co-evolve with the rest of model parameters, making the temperature schedule adapt to the training procedure. Our experiments on language modeling exhibited improved performance in both Penn Treebank and WikiText-2 datasets by a significant margin in both the standard fine-tune setting and the one without fine-tuning. The improvements are consistent for both validation and testing splits. We also conducted a comprehensive analysis and ablation studies to confirm the improvements of contextual temperature. We observed that our method is capable of controlling the uncertainties as the patterns of the history changes, allowing language modeling to achieve much better performance.

To the best of our knowledge, our research is the first systematic work that studies the role of temperature on a wide range of aspects during the training of a language model. Our results suggested new ways of training sequential models with discrete outputs using a parameterized temperature, that can potentially be extended beyond language modeling. We have also established a link between the model uncertainty control and the use of temperature, paying the way for extensions on tasks that require such control over long term, such as summarization, translation and dialogue generation.



(a) (b)

Figure 1: (a) Temperature for each word using our method over training epochs. Each line in the figure represents a distinct word ranked by their frequencies. The vertical axis shows the actual temperature minus 2. (b) Mean and confidence interval for temperature over the position in the sentence. We see that the average temperature is high at the beginning of the sentence and gradually decreases over time.

2 RELATED WORK

2.1 LANGUAGE MODELING

Language modeling aims at predicting the next word x_t in a sentence given a sequence of history $x_{1..t-1}$. In other words, the goal is to model the probability of choosing the k^{th} word, $P(x_t = k|x_{1..t-1})$ by using an encoding function f parameterized with θ to compress history $\mathbf{z} = f(x_{1..t-1};\theta)^T \cdot W$ and a Softmax decision layer σ.

$$P(x_t = k|x_{1..t-1};\theta) = \sigma(\mathbf{z})_k \tag{1}$$

Here $W \in \mathbb{R}^{d\times K}$ is a matrix that converts d dimensional output of f into a K dimensional vector. Advanced language models have been adopting deep neural architectures as its encoding function f, evolving from the feed-forward (Bengio et al., 2003) based ones to the more recent recurrent (Mikolov et al., 2010) (e.g. LSTM Hochreiter & Schmidhuber (1997)) and attention based models (Devlin et al., 2018).

2.2 SOFTMAX DECISION LAYER

A Softmax decision layer $\sigma(\mathbf{z})$ normalizes a K dimension, real-valued vector $\mathbf{z}$ to make it sum to 1.

$$\sigma(\mathbf{z})_i = \frac{z_i}{\sum_j^K e^{z_j}} \tag{2}$$

Recent progress in language modeling has suggested that a Mixture of Softmaxes(MoS) (Yang et al., 2018) would significantly improve the performance by computing multiple Softmax distributions and use a mixture weight to sum them up as the final probability distribution. To achieve this, a set of M matrices W_m is applied to the output generated by f. That is, $\mathbf{z}_m = f(x_{1..t-1}; \theta)^T \cdot W_m$. Similarly, $W_m \in \mathbb{R}^{d \times K}$ and again d is the dimension of the embedding and K is the dimension of the vocabulary. The probability distribution of the next word x_t under the MoS model is thus a mixture of the M Softmaxes weighted by π. Here $\Theta = \cup_{m=1}^{M} W_m \cup \theta$.

$$P_{MoS}(x_t = k | x_{1..t-1}; \Theta) = \sum_{m}^{M} \pi_m \cdot \sigma(\mathbf{z}_m)_k \tag{3}$$

2.3 Temperature Scaling

Temperature scaling is an approach used to control the smoothness of the output distribution. Instead of applying the Softmax function as suggested in Equation 1, logits here are divided by the temperature τ before passing through the Softmax function.

$$P(x_t = k | x_{1..t-1}; \theta, \tau) = \sigma(\mathbf{z}/\tau)_k \tag{4}$$

Constant Temperature. Early work that adopts constant temperature can be traced back to Model Distillations (Hinton et al., 2014). Several other works have been taking advantage of a fixed temperature during training (Norouzi et al., 2016; Ma et al., 2017; Chen et al., 2019). For instance, (Norouzi et al., 2016; Ma et al., 2017) optimize log-likelihood on augmented outputs sampled proportionally to their exponential scaled rewards, where temperature controls the degree of output augmentation. Other works incorporate temperature during the inference time (Guo et al., 2017; Caccia et al., 2018).

Dynamic Temperature Over Training Iterations. A wide range of work exist that adopts a schedule to tune temperature during the course of training. Notably in (Hu et al., 2017), the authors introduced a new text generation architecture that combines VAE and a discriminator. Since text samples are discrete and non-differentiable, a continuous approximation based on Softmax with a decreasing temperature is used to enable gradient propagation from the discriminators to the generator. Similar techniques are adopted in the gumbel-softmax trick (Jang et al., 2017), which also allows gradients to pass through discrete sampling objectives. Research has also shown that temperature with a heating up schedule makes the embedding vectors more compact (Zhang et al., 2018).

Dynamic Temperature Over Word Position. Another work that is closely related to our paper is the adaptive temperature over an attention model (Lin et al., 2018). We note that contextual temperature further learns the temperature for each vocabulary in the output distribution.

3 Methods

3.1 Contextual Temperature

Contextual temperature is a mechanism that chooses the best temperature by considering the "context" of a word x_t. A context of a word includes not only the history $x_{1..t-1}$ but also the specific vocabulary k that we calculate probability on. Such a mechanism allows us to parameterize τ using a deep neural network and adapt the softness of the Softmax that optimizes the model performance.

Our temperature vector $\tau \in \mathbb{R}^K$ is generated from the same mapping function f as discussed before. Although this mapping can be any sequential models such as RNN or LSTM, we choose to parameterize f by using AWD-LSTM (Merity et al., 2018). We omit the details of its architecture due to the limit of space in this paper. The output of AWD-LSTM is a vector that has dimension D. We multiply this vector by two matrices $W_{\tau_1} \in \mathbb{R}^{D \times Q}$ and $W_{\tau_2} \in \mathbb{R}^{Q \times K}$. Please note that one can potentially use a single matrix to represent these two. However, doing so can significantly increase the number of parameters and thus in practice we factorize them into two. And finally, we scale the temperature using a Softmax function over the dimension of the vocabulary and scale it to the range of $[\frac{\alpha}{\beta}, \frac{1+\alpha}{\beta}]$.

$$\tau = \frac{\sigma(f(x_{1..t-1}; \theta)^T \cdot W_{\tau_1} \cdot W_{\tau_2}) + \alpha}{\beta} \tag{5}$$

3.2 Contextual Temperature MoS

We then use the Contextual Temperature Mixture of Softmaxes architecture for language modeling (CT-MoS). The CT-MoS model extends Equation 3 by adding the contextual temperature in Equation 5. Here $\oslash$ represents the element-wise division between the $\mathbf{z}_m$ and the temperature vector τ. The results it then sent to the mixture of Softmaxes model. Since new parameters are being added to the model, the CT-MoS now has more parameters $\Theta = \cup_{m=1}^{M} W_m \cup \theta \cup W_{\tau_1} \cup W_{\tau_2}$.

$$P_{CT-MoS}(x_t = k | x_{1..t-1}; \Theta) = \sum_{m}^{M} \pi_m \cdot \sigma(\mathbf{z}_m \oslash \tau)) \tag{6}$$

One thing that worth noticing is that compares to prior work, our contextual temperature model has the ability to adopt a different temperature for 1) different vocabulary in the prediction, 2) different position of the same vocabulary given history and 3) a tunable parameter that is capable of changing as the training process progresses. The detailed architecture is illustrated in Figure 2, which highlights the difference between our CT-MoS model and the MoS model.

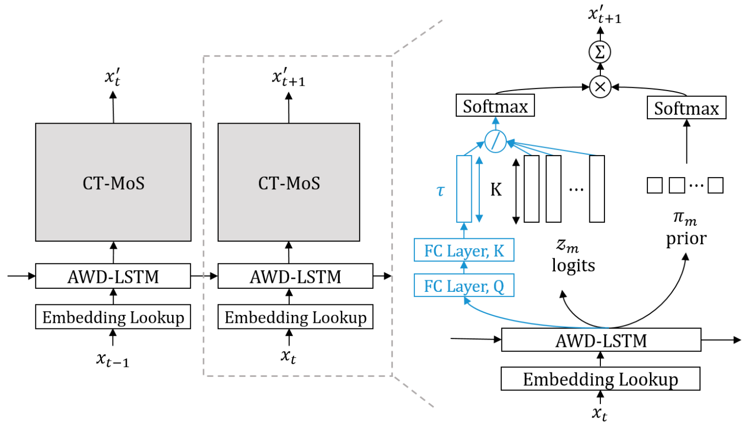


Figure 2: The architecture of our proposed CT-MoS model. Black components are the ones that are the same as the MoS model, while the blue ones are the newly added ones in our approach.

3.3 Training Objectives and Loss Scaling

We adopt the same regularization techniques in the MoS (Yang et al., 2018) and AWD-LSTM (Merity et al., 2018) paper. Our loss function thus consists of four terms: Cross Entropy ($\mathcal{H}$), Activation Regularization (AR), Temporal Activation Regularization (TAR), and Weight Decay (WD). AR is used to penalize the high values of outputs, and TAR is used to prevent outputs from changing too much between timesteps. WD is used to prevent overfitting. Here $\gamma 1$, $\gamma 2$ and $\gamma 3$ are constants to scale the regularization terms. And m is the dropout mask (See in Merity et al. (2018)).

$$\mathcal{L}(\Theta) = \underbrace{\frac{\sum_i^K \tau_i}{K}}_{\text{LS}} \mathcal{H}(\hat{\mathbf{y}}, \mathbf{y}) + \gamma_1 \underbrace{L_2(m \odot f(x_{1..t-1}; \theta))}_{\text{AR}} + \gamma_2 \underbrace{L_2(f(x_{1..t-1}; \theta) - f(x_{1..t}; \theta))}_{\text{TAR}} + \gamma_3 \underbrace{L_2(\Theta)}_{\text{WD}} \tag{7}$$

The uniqueness in our setting is the Loss Scaling term (LS). The use of temperature makes gradients of $\mathcal{H}$ disproportional to that of the other three terms, so the scale of $\mathcal{H}$ needs to be adjusted accordingly. A similar phenomenon is reported in Hinton et al. (2014), where temperature is used for knowledge distillation. In our case, only the gradients of $\mathcal{H}$ will be influenced by temperature. As for the three regularization terms, their gradients are not related to temperature since they regularize on parameters before temperature scaling. This difference results in the unbalance between the four objectives we have. Therefore, we scale $\mathcal{H}$ for keeping the balance. We empirically found that scaling $\mathcal{H}$ with the average of temperatures works well in our setting.

4 Experiments

4.1 Datasets

We evaluated contextual temperature on two widely-used datasets for language modeling: Penn Treebank (PTB) (Marcus et al., 1993; Mikolov et al., 2011) and WikiText-2 (WT2) (Merity et al., 2017). The PTB dataset contains 929k training, 73k validation and 82k test tokens. The vocabulary size is capped at 10,000 most frequent unique words, and the rest of words are replaced with the <unk> token. We follow the common practice to pre-process the dataset (Mikolov et al., 2011): (a) words are lower-cased, (b) numbers are replaced with "N", (c) newlines are replaced with <eos> and (d) punctuation is removed.

WikiText-2 (WT2) is derived from Wikipedia articles and released as a popular option to train language models. WT2 contains 2M training tokens and a vocabulary of around 33k words. Compared to PTB, WT2 is approximately two times larger in example size and three times larger in vocabulary size.

4.2 Experimental Setups

We conduct experiments on PTB and WT2 with single and four 1080 Ti GPUs, respectively. The environment we use is PyTorch (Paszke et al., 2017). We follow the training configurations as reported in the MoS paper and their github[1]. For both PTB and WT2, we use the same number of parameters as the one found in the MoS paper. For PTB, we use three layers of LSTM with embedding size of 960-960-620 in PTB experiments. That is the say, the number of embedding for functional mapping, $D = 620$. The embedding size we use here is $Q = 280$. And for WT2, we use three layers of LSTM with embedding size of 1150-1150-650. In both experiments, the $\gamma_1, \gamma_2, \gamma_3$ for the regularization terms are 2.0, 1.0 and $1.2e^{-6}$, respectively. The number of Softmaxes to be mixed is $M = 15$. To ensure that the temperature values are positive, we do normalization as shown in Equation 5. We have tried several different settings for α and β, and find $(\alpha, \beta) = (1, \frac{1}{2})$ works best in all experiments.

4.3 Results

We first show experimental results on the PTB, showing in Table 1. The original MoS model (Yang et al., 2018) has a model size of 22M parameters. To make a fair comparison, we augmented the number of parameters of the MoS model and named it Mos$^+$ with 24M parameters. This is done by increasing its size of word embedding from 280 to 410. We show that our CT-MoS model outperforms both AWD-LSTM, MoS and MoS$^+$ model on both validation and test sets with consistently lower perplexity. The conclusion holds for the default setting with finetuning (Merity et al., 2018), the one without fine-tuning, and the one with dynamic evaluation (Merity et al., 2018). Our best model achieved 48.12 perplexity on the validation set and 47.42 on the test set, beating the best state-of-the-art model of MoS with a significant margin.

Table 2 provides the results for WT2, a much larger language modeling dataset. We see a similar pattern to the previous dataset that CT-MoS outperforms the state-of-the-art model on the settings with and without finetune. On the dynamic evaluation, our model also achieved comparable results to MoS and AWD-LSTM.

4.4 Ablation Studies

Fixed Model Size Comparison. As we mentioned in the previous section, we increased the number of parameters of the MoS model and made it MoS$^+$ to conduct a fair comparison with our model using the same number of parameters in Table 1. To construct MoS$^+$ with more parameters, we increased the embedding size from 280 to 410. In Table 1, we notice that MoS$^+$ has higher perplexity compared to both MoS and CT-MoS, which shows increasing model parameters cannot improve the performance in this case. Similar observation and results are also reported by Yang et al. (2018). This ablation study shows the CT-MoS can outperform MoS in the same number of parameters.

[1]`https://github.com/zihangdai/mos`

Table 1: Performance Comparison on the Penn Treebank (PTB) dataset

Model	#Param	Validation	Test
AWD-LSTM w/o finetune	24M	60.7	58.8
AWD-LSTM	24M	60.0	57.3
AWD-LSTM + dynamic evaluation	24M	51.6	51.1
MoS w/o finetune	22M	58.08	55.97
MoS	22M	56.54	54.44
MoS + dynamic evaluation	22M	48.33	47.69
MoS$^+$ w/o finetune	24M	59.72	57.43
MoS$^+$	24M	58.54	56.36
MoS$^+$ + dynamic evaluation	24M	50.49	49.81
CT-MoS w/o finetune	24M	**56.95**	**54.69**
CT-MoS	24M	**55.31**	**53.2**
CT-MoS + dynamic evaluation	24M	**48.12**	**47.42**

Table 2: Perfomrance Comparison on the WikiText-2 (WT2) dataset

Model	#Param	Validation	Test
AWD-LSTM w/o finetune	33M	69.1	66.0
AWD-LSTM	33M	68.6	65.8
AWD-LSTM + dynamic evaluation	33M	46.4	44.3
MoS w/o finetune	35M	66.01	63.33
MoS	35M	63.88	61.45
MoS + dynamic evaluation	35M	**42.41**	**40.68**
CT-MoS w/o finetune	45M	**65.25**	**62.21**
CT-MoS	45M	**62.89**	**60.13**
CT-MoS + dynamic evaluation	45M	42.88	40.96

Temperature Normalization Methods. Instead of using the Softmax function in Equation 5 to normalize the temperature between $[\frac{\alpha}{\beta}, \frac{1+\alpha}{\beta}]$, we have considered several alternative temperature normalization methods: (a) $\lambda^{Tanh(\mu)}$, provided by Lin et al. (2018), makes the range of temperature to be $(\frac{1}{\lambda}, \lambda)$. And (b) Tanh($\mu$) + λ, which is capable of generating a range that is similar to the Softmax method used in Equation 5. To make our notations simple and clear, we define $\mu = f(x_{1..t-1};\theta)^T \cdot W_{\tau_1} \cdot W_{\tau_2}$. (c) $(\sigma(\mu) + \alpha)/\beta$ is our method. Results are illustrated in Table 3. All three methods are evaluated on PTB, and in this ablative study the experiments are conducted without either fine-tuning or dynamic evaluation for conciseness. The experimental results show that our proposed method outperforms other temperature normalization methods.

Table 3: Different methods for temperature normalization on the PTB dataset

Model	hyper-parameter	range	#Param	Validation	Test
$\lambda^{Tanh(\mu)}$ (Lin et al., 2018)	λ = 4	[1/4, 4]	24M	65.11	62.21
Tanh(μ) + λ	λ = 3	[2,4]	24M	61.35	58.89
$(\sigma(\mu) + \alpha)/\beta$	α = 1, β = 0.5	[2,4]	24M	**56.95**	**54.69**

Effects of Loss Scaling. As discussed in Section 3.3, the loss values may need to be scaled since using temperature may lead to smaller gradients. Here, we compare the results of whether applying loss scaling or not in Table 4. Note that with loss scaling leads to better results (lower perplexity).

Table 4: Perplexity on PTB w/o finetune

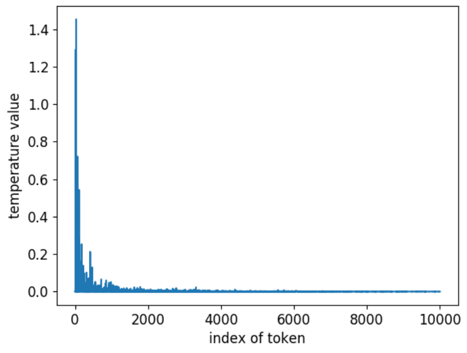
Model	#Param	Validation	Test
CT-MoS w/o loss scaling	24M	57.13	55.28
CT-MoS	24M	**56.95**	**54.69**

4.5 Analysis

Temperature Change During Training. Going back to Figure 1(a) that we covered in the Introduction section, we have shown that contextual temperature is capable of generating a changing temperature as the training proceeds on a per vocabulary basis. This allows each vocabulary to have a flexible temperature schedule that is optimal to the model. Vocabularies that have higher temperature are common words that don't convey much information. This suggests that models are more confident about these words. Our method has the advantage of determining the schedule automatically, which is not possible in the traditionally fixed temperature scheduling method.

Average Temperature Over Word Positions. We also want to dive deeper into Figure 1(b), which shows the mean and confidence interval of the temperatures over different positions of the words. We grouped sentences whose length ranges from 15 to 25. For sentences whose length are greater than 15, we pick the first 5 tokens, the middle 5 tokens, and the last 5 tokens to form a new sentence. At the beginning of a sentence, temperatures are usually high to smooth the probabilities of the Softmax. This is the region where the model has little confidence since there is just too little information in the history. Over time, as the history builds up and the model becomes more confident, temperature begins to cool down, which makes Softmax spiky and trust more on the learned model. The confidence intervals of the temperature also become tighter as the length of the history goes up.

Word Frequencies And Temperature Change. Evidence from Figure 1(a) suggests that only a small fraction of the vocabularies have a significantly larger temperature. However, even small changes in the temperature might deliver large impacts to the performance due to the facts that these vocabularies might be used more often than others. To further analyze we present an analysis on the weighted temperature which is equal to the temperature multiplied by the frequencies of the vocabularies in Figure 3(a). Here we can see that for the majority of the words, small changes in the temperature might still be significant as many of them occur fairly frequently.



(a)

Figure 3: (a) The weighted temperature over the token index ranked by frequency. Here the weighted temperature refers to the absolute temperature multiplied by the frequency of the word in the corpus.

Case Studies. To visualize the effects of temperature, we analyzed the performane of our model on several samples from the PTB dataset. Table 5 shows the comparison of CT-MoS v.s. MoS. Here we highlight two differences between our model and MoS, annotated using red and blue colors. In the first spot, we see that both CT-MoS and MoS failed to predict the correct answer. With the correct token being “single-a-1”, which refers to a rating for securities. The CT-MoS predicted “triple-a”, which is not the same as the reference but is much closer to the answer since “triple-a” also refers to the highest rating for securities. MoS, on the other hand, predicted <unk>, which deviates too much from the ground truth. Taking a closer look at the temperature, the word “triple-a” has a temperature of $2{+}8.34{\times}10^{-5}$, which is a bit smaller than that of the <unk> of $2{+}8.6{\times}10^{-5}$. This contributes to the factor that the model chooses “triple-a” over <unk>. Another example can be illustrated by the prediction of the word “standard” of the CT-MoS model. Here its temperature is smaller than that “s&p”, making the model more likely to predict the prior word.

Table 5: Analysis of Model Performance on a Sample from the PTB Dataset

Reference	rated **single-a-1** by moody ’s investors service inc. and single-a by **standard** & poor ’s corp. ...			
CT-MoS	rated **triple-a** by moody ’s and service inc. and <unk> by **standard** & poor ’s corp. ...			
MoS	rated **<unk>** by moody ’s and service inc. and <unk> by **s&p** & poor ’s corp. ...			
CT-MoS top-4	**triple-a 0.34**	single-a-2 0.2	single-a-1 0.15	single-a-3 0.11
MoS top-4	**<unk> 0.28**	**triple-a 0.27**	single-a-2 0.24	single-a-1 0.1
$\tau(1e^{-5}+2)$	**triple-a 8.34**		**<unk> 8.60**	
CT-MoS top-4	**standard 0.53**	**s&p 0.22**	moody 0.17	dow 0.02
MoS top-4	**s&p 0.4**	moody 0.23	**standard 0.19**	<unk> 0.03
$\tau(1e^{-5}+2)$	**standard 11.2**		**s&p 11.4**	

Another indicator of how contextual temperature work is to look at the temperature change across different positions in a sentence. In Table 6, we visualize the occurrence of the word “mortage” and its temperature. Here we see that as the position changes, contextual temperature chooses a different value for each of the position, adjusting its confidence of model’s belief. As we analyzed before, a general trend is that words appear early in the sentence got larger temperatures while the ones approach the end of the sentence got smaller temperatures.

Table 6: Analysis of Temperature for Same Words at Different Positions

CT-MoS	loan **mortgage(1)** corp freddie mac posted posted yields on 30-year **mortgage(2)** commitments for delivery within N days <eos> N N standard conventional fixed-rate **mortgages(3)** N N N rate rate capped one-year adjustable rate **mortgages(4)** <eos> source telerate systems inc <eos> federal national **mortgage(5)** association fannie mae posted posted yields on N year **mortgage(6)** commitments for delivery within N days priced at par N N N standard conventional fixed-rate **mortgages(7)** N ...						
$\tau(1e^{-5}+2)$	(1) 18.9	(2) 19.3	(3) 20.1	(4) 19.2	(5) 18.9	(6) 19.2	(7) 18.2

5 Conclusion and Future Work

In this paper, we propose contextual temperature that is capable of generating a changing temperature based on the history of the text. Our temperature model is parameterized using a deep neural network and will generate a unique schedule for each vocabulary. Experiments on the language modeling datasets suggest the advantages of this method on language generation and prediction problems. Our work opened up potential new research directions along the line of fully automated temperature mechanism to replace the existing practices of manually designed ones.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

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

Chaoqi Chen, Weiping Xie, Wenbing Huang, Yu Rong, Xinghao Ding, Yue Huang, Tingyang Xu, and Junzhou Huang. Progressive feature alignment for unsupervised domain adaptation. In *CVPR*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

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, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

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.

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 - More Samples from PTB

Table 7: More samples from PTB.

Reference	these rate indications are n't directly comparable lending practices vary widely by location <eos> treasury bills <eos> **results** of the tuesday ...			
CT-MoS	these rate indications are n't directly comparable lending practices vary widely by location <eos> treasury bills results **results** of the monday ...			
MoS	these rate indications are n't directly comparable lending practices vary widely by location <eos> treasury bills results **treasury** of the monday ...			
CT-MoS top-4	**results 0.81**	**treasury 0.07**	a 0.01	bonds 0.01
MoS top-4	**treasury 0.64**	**results 0.09**	the 0.04	N 0.02
$\tau(1e^{-6}+2)$	**results 8.11**		**treasury 8.58**	
Reference	corporate loans at large u.s. money center commercial banks <eos> federal funds <eos> N N **N** ...			
CT-MoS	corporate loans at large u.s. money center commercial banks <eos> federal funds N N N **N** ...			
MoS	corporate loans at large u.s. money center commercial banks< eos> federal funds N N N **high** ...			
CT-MoS top-4	**N 0.45**	**high 0.41**	<eos> 0.05	low 0.04
MoS top-4	**high 0.46**	**N 0.41**	<eos> 0.06	and 0.02
$\tau(1e^{-4}+2)$	**N 1.56**		**high 2.01**	
Reference	a share compared with a net loss of $ N million last year after a **loss** from discontinued operations ...			
CT-MoS	a share <eos> with $ loss loss of $ N million or year <eos> the **loss** of the operations ...			
MoS	a share <eos> with $ $ loss of $ N million or year <eos> the **$** of the operations ...			
CT-MoS top-4	**loss 0.24**	**$ 0.20**	N 0.06	<unk> 0.06
MoS top-4	**$ 0.11**	**loss 0.09**	one-time 0.07	N 0.07
$\tau(1e^{-4}+2)$	**loss 1.40**		**$ 1.62**	
Reference	in the **nine** months <unk> 's net rose N N to $ N million ...			
CT-MoS	the the **nine** months the said net income N N to $ N million ...			
MoS	the the **first** months the said net income N N to $ N million ...			
CT-MoS top-4	**nine 0.17**	third 0.10	year-ago 0.09	year-earlier 0.09
MoS top-4	**first 0.14**	**nine 0.12**	third 0.12	year-ago 0.09
$\tau(1e^{-5}+2)$	**nine 7.37**		**first 7.94**	