MULTI-TIMESCALE REPRESENTATION LEARNING IN LSTM LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models must capture statistical dependencies between words at timescales ranging from very short to very long. Earlier work has demonstrated that dependencies in natural language tend to decay with distance between words according to a power law. However, it is unclear how this knowledge can be used for analyzing or designing neural network language models. In this work, we derived a theory for how the memory gating mechanism in long short-term memory (LSTM) language models can capture power law decay. We found that unit timescales within an LSTM, which are determined by the forget gate bias, should follow an Inverse Gamma distribution. Experiments then showed that LSTM language models trained on natural English text learn to approximate this theoretical distribution. Further, we found that explicitly imposing the theoretical distribution upon the model during training yielded better language model perplexity overall, with particular improvements for predicting low-frequency (rare) words. Moreover, the explicit multi-timescale model selectively routes information about different types of words through units with different timescales, potentially improving model interpretability. These results demonstrate the importance of careful, theoretically-motivated analysis of memory and timescale in language models.

1 INTRODUCTION

Autoregressive language models are functions that estimate a probability distribution over the next word in a sequence from past words, $p(w_t|w_{t-1}, \ldots, w_1)$. This requires capturing statistical dependencies between words over short timescales, where syntactic information likely dominates (Adi et al., 2017; Linzen et al., 2016), as well as long timescales, where semantic and narrative information likely dominate (Zhu et al., 2018; Conneau et al., 2018; Gulordava et al., 2018). Because this probability distribution grows exponentially with sequence length, some approaches simplify the problem by ignoring long-range dependencies. Classical n-gram models, for example, assume word w_t is independent of all but the last $n-1$ words, with typical $n = 5$ (Heafield, 2011). Hidden Markov models (HMMs) assume that the influence of previous words decays exponentially with distance from the current word (Lin & Tegmark, 2016).

In contrast, neural network language models such as recurrent (Hochreiter & Schmidhuber, 1997; Merity et al., 2018; Melis et al., 2018) and transformer networks (Melis et al., 2019; Krause et al., 2019; Dai et al., 2019) include longer-range interactions, but simplify the problem by working in lower-dimensional representational spaces. Attention-based networks combine position and content-based information in a small number of attention heads to flexibly capture different types of dependencies within a sequence (Vaswani et al., 2017; Cordonnier et al., 2019). Gated recurrent neural networks (RNNs) compress information about past words into a fixed-length state vector (Hochreiter & Schmidhuber, 1997). The influence each word has on this state vector tends to decay exponentially over time. However, each element of the state vector can have a different exponential time constant, or "timescale" (Tallec & Ollivier, 2018), enabling gated RNNs like the long short-term memory (LSTM) network to flexibly learn many different types of temporal relationships (Hochreiter & Schmidhuber, 1997). Yet in all these networks the shape of the temporal dependencies must be learned directly from the data. This seems particularly problematic for very long-range dependencies, which are only sparsely informative (Lin & Tegmark, 2016).

This raises two related questions: what should the temporal dependencies in a language model look like? And how can that information be incorporated into a neural network language model?

To answer the first question, we look to empirical and theoretical work that has explored the dependency statistics of natural language. Lin & Tegmark (2016) quantified temporal dependencies in English and French language corpora by measuring the mutual information between tokens as a function of the distance between them. They observed that mutual information decays as a power law, i.e. $MI(w_k, w_{k+t}) \propto t^{-d}$ for constant d. This behavior is common to hierarchically structured natural languages (Lin & Tegmark, 2016; Sainburg et al., 2019) as well as sequences generated from probabilistic context-free grammars (PCFGs) (Lin & Tegmark, 2016).

Now to the second question: if temporal dependencies in natural language follow a power law, how can this information be incorporated into neural network language models? To our knowledge, little work has explored how to control the temporal dependencies learned in attention-based models. However, many approaches have been proposed for controlling gated RNNs, including updating different groups of units at different intervals (El Hihi & Bengio, 1996; Koutnik et al., 2014; Liu et al., 2015; Chung et al., 2017), gating units across layers (Chung et al., 2015), and explicitly controlling the input and forget gates that determine how information is stored and removed from memory (Xu et al., 2016; Shen et al., 2018; Tallec & Ollivier, 2018). Yet none of these proposals incorporate a specific shape of temporal dependencies based on the known statistics of natural language.

In this work, we build on the framework of Tallec & Ollivier (2018) to develop a theory for how the memory mechanism in LSTM language models can capture temporal dependencies that follow a power law. This relies on defining the *timescale* of an individual LSTM unit based on how the unit retains and forgets information. We show that this theory predicts the distribution of unit timescales for LSTM models trained on both natural English (Merity et al., 2018) and formal languages (Suzgun et al., 2019). Further, we show that forcing models to follow this theoretical distribution improves language modeling performance. These results highlight the importance of combining theoretical modeling with an understanding of how language models capture temporal dependencies over multiple scales.

2 Multi-timescale Language Models

2.1 Timescale of information

We are interested in understanding how LSTM language models capture dependencies across time. Tallec & Ollivier (2018) elegantly argued that memory in individual LSTM units tends to decay exponentially with a time constant determined by weights within the network. We refer to the time constant of that exponential decay as the unit's representational timescale.

Timescale is directly related to the LSTM memory mechanism (Hochreiter & Schmidhuber, 1997), which involves the LSTM cell state c_t, input gate i_t and forget gate f_t,

$$\begin{aligned} i_t &= \sigma(W_{ix}x_t + W_{ih}h_t + b_i) \\ f_t &= \sigma(W_{fx}x_t + W_{fh}h_t + b_f) \\ \tilde{c}_t &= \tanh(W_{cx}x_t + W_{ch}h_t + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \end{aligned}$$

where x_t is the input at time t, h_t is the hidden state, $W_{ih}, W_{ix}, W_{fh}, W_{fx}, W_{ch}, W_{cx}$ are the different weights and b_i, b_f, b_c the respective biases. $\sigma(\cdot)$ and $\tanh(\cdot)$ represent element-wise sigmoid and hyperbolic tangent functions. Input and forget gates control the flow of information in and out of memory. The forget gate f_t controls how much memory from the last time step c_{t-1} is carried forward to the current state c_t. The input gate i_t controls how much information from the input x_t and hidden state h_t at the current timestep is stored in memory for subsequent timesteps.

To find the representational timescale, we consider a "free input" regime with zero input to the LSTM after timestep t_0, i.e., $x_t = 0$ for $t > t_0$. Ignoring information leakage through the hidden state (i.e., assuming $W_{ch} = 0$, $b_c = 0$, and $W_{fh} = 0$) the cell state update becomes $c_t = f_t \odot c_{t-1}$. For $t > t_0$, it can be further simplified as

$$\begin{aligned} c_t &= f_0^{t-t_0} \odot c_0 \\ &= e^{(\log f_0)(t-t_0)} \odot c_0, \end{aligned} \tag{1}$$

where $c_0 = c_{t_0}$ is the cell state at t_0, and $f_0 = \sigma(b_f)$ is the value of the forget gate, which depends only on the forget gate bias b_f. Equation 1 shows that LSTM memory exhibits exponential decay with characteristic *forgetting time*

$$T = -\frac{1}{\log f_0} = \frac{1}{\log(1 + e^{-b_f})}. \tag{2}$$

That is, values in the cell state tend to shrink by a factor of e every T timesteps. We refer to the forgetting time in Equation 2 as the representational timescale of an LSTM unit.

Beyond the "free regime", we can estimate the timescale for a LSTM unit by measuring the average forget gate value over a set of test sequences,

$$T_{est} = -\frac{1}{\log \bar{f}}, \tag{3}$$

where $\bar{f} = \frac{1}{KN} \sum_{j=1}^{N} \sum_{t=1}^{K} f_t^j$, in which f_t^j is the forget gate value of the unit at t-th timestep for j-th test sequence, N is the number of test sequences, and K is the test sequence length.

2.2 Combining exponential timescales to yield a power law

From earlier work, we know that temporal dependencies in natural language tend to decay following a power law (Lin & Tegmark, 2016; Sainburg et al., 2019). Yet from Equation 1 we see that LSTM memory tends to decay exponentially. These two decay regimes are fundamentally different–the ratio of a power law divided by an exponential always tends towards infinity. However, LSTM language models contain many units, each of which can have a different timescale. Thus LSTM language models might approximate power law decay through a combination of exponential functions. Here we derive a theory for how timescales should be distributed within an LSTM language model in order to yield overall power law decay.

Let us assume that the timescale T for each LSTM unit is drawn from a distribution $P(T)$. We want to find a $P(T)$ such that the expected value over T of the function $e^{\frac{-t}{T}}$ approximates a power law decay t^{-d} for some constant d,

$$t^{-d} \propto \mathbb{E}_T[e^{-t/T}] = \int_0^\infty P(T) e^{-t/T} dT. \tag{4}$$

Noting the similarity between this equation and the Laplace transform, we can solve this problem to find that $P(T)$ is an *Inverse Gamma distribution* with shape parameter $\alpha = d$ and scale parameter $\beta = 1$. The probability density function of the Inverse Gamma distribution is given as $P(T; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} (1/T)^{\alpha+1} e^{(-\beta/T)}$.

This theoretical result suggests that in order to approximate the power law decay of information in natural language, unit timescales in LSTM language models should follow an Inverse Gamma distribution. We next perform experiments to test whether this prediction holds true for models trained on natural language and models trained on samples from a formal language with known temporal statistics. We then test whether enforcing an Inverse Gamma timescale distribution at training time improves model performance.

2.3 Controlling LSTM unit timescales

To enforce a specific distribution of timescales in an LSTM and thus create an explicit multi-timescale model, we drew again upon the methods developed in Tallec & Ollivier (2018). Following the analysis in Section 2.1, the desired timescale $T_{desired}$ for an LSTM unit can be controlled by setting the forget gate bias to the value

$$b_f = -\log(e^{\frac{1}{T_{desired}}} - 1). \tag{5}$$

The balance between forgetting information from the previous timestep and adding new information from the current timestep is controlled by the relationship between forget and input gates. To maintain this balance we set the input gate bias b_i to the opposite value of the forget gate, i.e., $b_i = -b_f$. This ensures that the relation $i_t \approx 1 - f_t$ holds true. Importantly, these bias values remain fixed (i.e. are *not* learned) during training, in order to keep the desired timescale distribution across the network.

3 EVALUATION

3.1 EXPERIMENTAL SETUP

Here we describe our experimental setup. We examined two regimes: natural language data, and synthetic data from a formal language. All models were implemented in pytorch (Paszke et al., 2019).

3.1.1 NATURAL LANGUAGE

We experimentally evaluated LSTM language models trained on the Penn Treebank (PTB) (Marcus et al., 1999; Mikolov et al., 2011) and WikiText-2 (WT2) (Merity et al., 2017) datasets. PTB contains a vocabulary of 10K unique words, with 930K tokens in the training, 200K in validation, and 82K in test data. WT2 is a larger dataset with a vocabulary size of 33K unique words, almost 2M tokens in the training set, 220K in the validation set, and 240K in the test set. As a control, we also generated a Markovian version of the PTB dataset. Using the empirical bigram probabilities from PTB, we sampled tokens sequentially until a new corpus of the same size had been generated.

We compared two language models: a standard stateful LSTM language model (Merity et al., 2018) as the baseline, and our multi-timescale language model. Both models comprise three LSTM layers with 1150 units in the first two layers and 400 units in the third layer, with an embedding size of 400. Input and output embeddings were tied. All models were trained using SGD followed by non-monotonically triggered ASGD for 1000 epochs. Training sequences were of length 70 with a probability of 0.95 and 35 with a probability of 0.05. During inference, all test sequences were length 70. For training, all embedding weights were uniformly initialized in the interval $[-0.1, 0.1]$. All weights and biases of the LSTM layers in the baseline language model were uniformly initialized between $\left[\frac{-1}{H}, \frac{1}{H}\right]$ where H is the output size of the respective layer.

The multi-timescale language model has the same initialization, except for the forget and input gate bias values that are assigned following Equation 5 and fixed during training. For layer 1, we assigned timescale $T = 3$ to half the units and $T = 4$ to the other half. For layer 2, we assigned timescales to each unit by selecting values from an Inverse Gamma distribution. The shape parameter of the distribution was set to $\alpha = 0.56$ after testing different values (see A.4). The timescales in this layer had 80% of the units below 20, and the rest ranging up to the thousands. For layer 3, the biases were initialized like the baseline language model and trained (not fixed).

3.1.2 FORMAL LANGUAGE: THE DYCK-2 GRAMMAR

We also tested the LSTM language models on a formal language with known temporal statistics. The Dyck-2 grammar (Suzgun et al., 2019) defines all possible valid sequences using two types of parentheses. Because Dyck-2 is a probabilistic context-free grammar, its temporal dependencies will tend to decay following a power law (Lin & Tegmark, 2016). Further, timescales within particular sequences can be measured as the distance between matching opening and closing parenthesis pairs. We randomly sampled from the grammar with probabilities $p_1 = 0.25$, $p_2 = 0.25$, and $q = 0.25$ to generate training, validation, and test sets with 10K, 2K, and 5K sequences. Sequences contain a maximum of 200 elements.

To solve this task, we followed Suzgun et al. (2019). Models were trained predict which closing parentheses could appear next. Elements in the sequence were one-hot encoded and fed as input to the models. We trained a baseline model and a multi-timescale version. Each model consists of a 256-unit LSTM layer followed by a linear output layer. The output layer reduces the dimension to 2, the number of possible closing parentheses. These output values are each converted to a $(0, 1)$ interval with a sigmoid function. Parameters were initialized using a uniform distribution. The multi-timescale model has the forget and input gate bias values assigned using an Inverse Gamma distribution with $\alpha = 0.49$, and fixed during training. This parameter best matched the measured distribution of timescales. Training minimized the mean squared error (MSE) loss using the Adam optimizer (Kingma & Ba, 2015) with learning rate $1e{-}4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 1e{-}8$ for 2000 epochs.

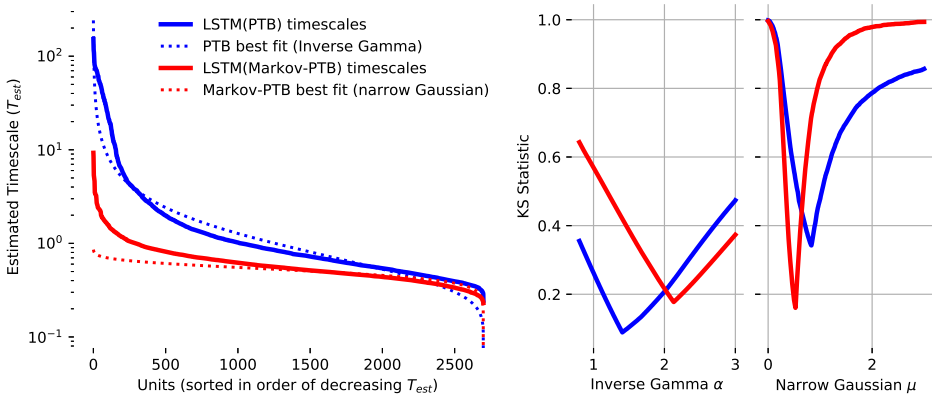


Figure 1: Empirical timescale distributions and best distribution fits for an LSTM language model trained on natural language (PTB; blue), and the same model trained on a Markovian corpus (Markov-PTB; red). Left: An Inverse Gamma distribution is the best fit for the PTB corpus (dotted blue), while a narrow Gaussian is the best fit for the Markovian corpus (dotted red). Right: We determined the best fits by varying distribution parameters (α for the Inverse Gamma, μ for the narrow Gaussian) and computing the Kolmogorov-Smirnov statistic to measure the difference between the empirical and theoretical distributions (lower is better). The LSTM trained on natural language (blue) is best fit by the Inverse Gamma distribution with $\alpha = 1.4$, while the LSTM trained on a Markovian corpus (red) is best fit by the narrow Gaussian distribution with $\mu = 0.5$.

3.2 Experimental Results

3.2.1 Do empirical LSTM unit timescales approximate language statistics?

In Section 2.2 we predicted that the distribution of unit timescales within an LSTM language model should follow an Inverse Gamma distribution. To test this prediction, we trained an LSTM language model on a natural language corpus (PTB), and then trained the same model on a Markovian dataset generated with the bigram statistics of PTB (Markov-PTB). We estimated the timescale of each unit using Equation 3. While representing PTB should require an Inverse Gamma distribution of timescales, representing Markov-PTB should only require a single timescale (i.e. a delta function, which can be approximated as a narrow Gaussian with σ=0.1). We tested this by generating a family of Inverse Gamma distributions or narrow Gaussian functions, and measuring the difference between the empirical distributions and these theoretical distributions. This was quantified with the Kolmogorov-Smirnov test statistic (Massey Jr, 1951).

The results in Figure 1 suggest that our theoretical prediction is correct. The Inverse Gamma distribution better fits the model trained on natural language, while the approximate delta function (narrow Gaussian distribution) better fits the model trained on the Markovian corpus.

3.2.2 Does imposing an Inverse Gamma distribution improve model performance?

The previous section showed that the empirical distribution of timescales in an LSTM LM trained on natural language is well-approximated by an Inverse Gamma distribution, as predicted. However, the model had to learn this distribution from a small and noisy dataset. If the Inverse Gamma distribution is indeed optimal for capturing power law dependencies, then it should be beneficial to simply enforce this distribution. To test this hypothesis, we constructed models with enforced Inverse Gamma timescales, which we call multi-timescale (MTS) language models (see Section 3.1.1 for details). Compared to the baseline LSTM model, the multi-timescale model incorporates a timescale prior that matches the statistical temporal dependencies of natural language. The multi-timescale model is particularly enriched in long timescale units (see A.4).

If this is an effective and useful prior, the multi-timescale model should perform better than the baseline when very long timescales are needed to accomplish a task. We first compared total language

Table 1: Perplexity of the multi-timescale and baseline models for tokens across different frequency bins for the Penn TreeBank (PTB) and WikiText-2 (WT2) test datasets. We also report the mean difference in perplexity (baseline − multi-timescale) across 10,000 bootstrapped samples, along with the 95% confidence interval (CI).

Dataset: Penn TreeBank

Model	**above 10K**	**1K-10K**	**100-1K**	**below 100**	**All tokens**
Baseline	6.82	27.77	184.19	2252.50	61.40
Multi-timescale	6.84	27.14	176.11	2100.89	59.69
Mean diff.	-0.02	0.63	8.08	152.03	1.71
95% CI	[-0.06, 0.02]	**[0.38, 0.88]**	**[6.04, 10.2]**	**[119.1,186.0]**	**[1.41, 2.02]**

Dataset: WikiText-2

Baseline	7.49	49.70	320.59	4631.08	69.88
Multi-timescale	7.46	48.52	308.43	4318.72	68.08
Mean diff.	0.03	1.17	12.20	312.13	1.81
95% CI	**[0.01,0.06]**	**[0.83,1.49]**	**[9.96,14.4]**	**[267.9,356.3]**	**[1.61,2.01]**

modeling performance between the baseline and multi-timescale models. The overall perplexities on the test datasets are shown in the far right column of Table 1. The multi-timescale language model outperforms the baseline model for both datasets by an average margin of 1.75 perplexity. Bootstrapping over the test set showed that this improvement is statistically significant. Test data were divided into 100-word sequences and resampled with replacement 10,000 times. For each sample, we computed the difference in model perplexity (baseline − multi-timescale) and reported the 95% confidence intervals (CI) in Table 1. Differences are significant at $p < 0.05$ if the CI does not overlap with 0.

Previous work has demonstrated that common, typically closed-class, words rely mostly on short timescale information, whereas rare, typically open-class, words require longer timescale information (Khandelwal et al., 2018; Griffiths et al., 2005; Rosenfeld, 1994; Iyer & Ostendorf, 1996). To test whether the improved performance of the multi-timescale model could be attributed to the very long timescale units, we computed model perplexities across different word frequency bins. We hypothesized that the multi-timescale model should show the greatest improvements for infrequent words, while showing little to no improvement for common words.

We divided the words in the test dataset into 4 bins depending on their frequencies in the training corpus: more than 10,000 occurrences; 1000-10,000 occurrences; 100-1000 occurrences; and fewer than 100 occurrences. Then we compared performance of the models for words in each bin in Table 1. The multi-timescale model performed significantly better than baseline in both datasets for the 3 less frequent bins (bootstrap test), with increasing difference for less frequent words. This suggests that the performance advantage of the multi-timescale model is highest for infrequent words, which require very long timescale information.

These results strongly support our claim that the Inverse Gamma timescale distribution is a good prior for models trained on natural language that is known to have power law temporal dependencies.

3.2.3 Generalization to formal languages

Complex dependencies in natural language data may not necessarily follow any precise statistical law, so we also turned to formal languages to evaluate our hypotheses. In this experiment we used the Dyck-2 grammar, which enables us to precisely measure the distribution of timescales within the data (see Figure 2 right). We first used this capability to find the best α parameter value for the Inverse Gamma distribution, and then to measure model performance as a function of timescale.

Following Suzgun et al. (2019), baseline and multi-timescale language models were trained to predict closing parenthesis types. A sequence is considered correctly predicted when every output from the model is correct across the entire sequence. The baseline model was able to correctly predict the output of 91.66% of test sequences, while the multi-timescale model achieved 93.82% correct.

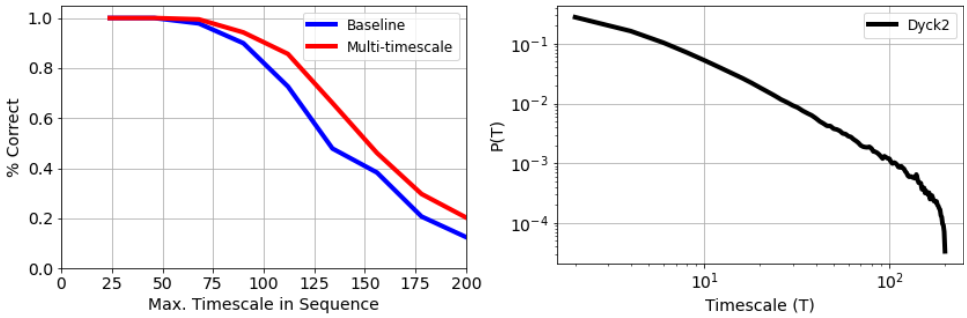


Figure 2: The left plot shows the performance of the baseline (blue) and multi-timescale (red) models on the Dyck-2 grammar as a function of the maximum timescale in a sequence. A correctly predicted sequence is considered to have all the outputs correct. The graph shows that the multi-timescale model is predicting better up to 10% more sequences in the longer timescales regime ($T > 75$). The right plot describe the distribution of all timescales as computed from the training data. The timescale distribution decays as a power law. The sharp decay near $T = 200$ is due to the limitation of the sequence lengths imposed to generate the training data.

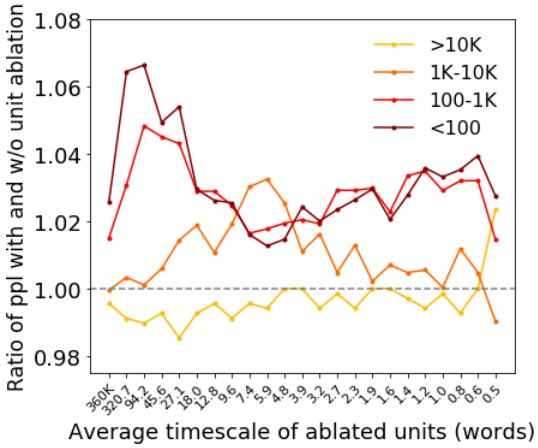


Figure 3: Information routing across different units of the multi-timescale LSTM for PTB dataset. Each line shows results for words in a different frequency bin. The ordinate axis shows the ratio of model perplexity with and without ablation of a group of 50 LSTM units, sorted and grouped by assigned timescale. Ratios above 1 indicate a decrease in performance following ablation, suggesting that the ablated units are important for predicting words in the given frequency bin. Abscissa shows the average timescale of each group.

Figure 2 (left) shows percent correct as a function of the maximum timescale in a sequence. Both models succeeded at predicting most sequences shorter than 75 steps. However, for longer timescales, the multi-timescale model correctly predicted 5-10% more sequences than the baseline model. This result supplements the idea that enforcing Inverse Gamma timescales improves the model's ability to capture longer-range dependencies. Nevertheless, both models are far from perfect on longer timescales, raising the question of how to further improve these longer timescales.

3.2.4 Routing of information through LSTM units with different timescales

Our previous results demonstrated successful control the timescale distribution in our multi-timescale model, and that this improved model performance. Breaking out performance results by word frequency also showed that the multi-timescale model was better able to capture some words–specifically, low-frequency words–than the baseline LSTM. This suggests that the multi-timescale model, which is enriched in long-timescale units relative to the baseline model, may be selectively routing information about infrequent words through specific long-timescale units. More broadly, the

multi-timescale model could be selectively routing information to each unit based on its timescale. This could make the multi-timescale model more interpretable, since at least one property of each unit–its timescale–would be known ahead of time. To test this hypothesis, we again divided the test data into word frequency bins. If long timescale information is particularly important for low frequency words, then we would expect information about those words to be selectively routed through long timescale units. We tested the importance of each LSTM unit for words in each frequency bin by selectively ablating the units during inference, and then measuring the effect on prediction performance.

We divided the LSTM units from layer 2 of the multi-timescale model into 23 groups of 50 consecutive units, sorted by assigned timescale. We ablated one group of units at a time by explicitly setting their output to 0, while keeping the rest of the units active. We then computed the model perplexity for different word frequency bins, and plotted the ratio of the perplexity with and without ablation. If performance gets worse for a particular word frequency bin when ablating a particular group of units, it implies that the ablated units are important for predicting those words.

Figure 3 shows this ratio across all frequency bins and groups for the PTB dataset (similar results for WikiText-2 are shown in the supplement). Ablating units with long timescales (20-300 words) causes performance to degrade the most for low frequency words (below 100 and 1K-10K); ablating units with medium timescales (5-10 words) worsens performance for medium frequency words (1k-10k); and ablating units with the shortest timescales ($<$1 word) resulted in worse performance on the highest frequency words. These results demonstrate that timescale-dependent information is routed through different units in this model, suggesting that the representations that are learned for different timescales are interpretable.

4 Conclusion

In this paper we developed a theory for how LSTM language models can capture power law temporal dependencies. We showed that this theory predicts the distribution of timescales in LSTM language models trained on both natural and formal languages. We also found that explicit multi-timescale models that are forced to follow this theoretical distribution give better performance, particularly over very long timescales. These results highlight the importance of theoretical modeling and understanding of how language models capture dependencies over multiple timescales.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=BJh6Ztuxl&noteId=BJh6Ztuxl.

Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *ICML*, pp. 2067–2075, 2015. URL http://proceedings.mlr.press/v37/chung15.html.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1di0sfgl.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2126–2136, 2018. URL https://www.aclweb.org/anthology/P18-1198/.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2019.

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

57th Annual Meeting of the Association for Computational Linguistics, pp. 2978–2988, 2019. URL https://www.aclweb.org/anthology/P19-1285/.

Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems*, pp. 493–499, 1996. URL http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies.pdf.

Thomas L Griffiths, Mark Steyvers, David M Blei, and Joshua B Tenenbaum. Integrating topics and syntax. In *Advances in neural information processing systems*, pp. 537–544, 2005.

Kristina Gulordava, Piotr Bojanowski, Édouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1195–1205, 2018. URL https://www.aclweb.org/anthology/N18-1108/.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pp. 187–197, 2011.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Rukmini Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*, volume 1, pp. 236–239. IEEE, 1996.

Shailee Jain and Alexander Huth. Incorporating context into language encoding models for fmri. In *Advances in neural information processing systems*, pp. 6628–6637, 2018. URL http://papers.nips.cc/paper/7897-incorporating-context-into-language-encoding-models-for-fmri.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 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)*, pp. 284–294, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1027. URL https://www.aclweb.org/anthology/P18-1027.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, pp. 1863–1871, 2014. URL http://proceedings.mlr.press/v32/koutnik14.html.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of transformer language models. *arXiv preprint arXiv:1904.08378*, 2019. URL https://arxiv.org/abs/1904.08378.

Henry W Lin and Max Tegmark. Critical behavior from deep dynamics: a hidden dimension in natural language. *arXiv preprint arXiv:1606.06737*, 2016. URL https://cbmm.mit.edu/sites/default/files/publications/1606.06737.pdf.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535, 2016. URL https://www.aclweb.org/anthology/Q16-1037/.

Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuan-Jing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2326–2335, 2015. URL https://www.aclweb.org/anthology/D15-1280/.

Mitchell P Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank-3. *Linguistic Data Consortium, Philadelphia*, 14, 1999. URL https://catalog.ldc.upenn.edu/LDC99T42.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

Gábor Melis, Tomáš Kočiskỳ, and Phil Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJe5P6EYvS&utm_campaign=NLP%20News&utm_medium=email&utm_source=Revue%20newsletter.

Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ByJHuTgA-.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyyGPP0TZ.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černockỳ. Empirical evaluation and combination of advanced language modeling techniques. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011. URL https://www.isca-speech.org/archive/interspeech_2011/i11_0605.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Ronald Rosenfeld. A hybrid approach to adaptive statistical language modeling. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1994.

Tim Sainburg, Brad Theilman, Marvin Thielk, and Timothy Q Gentner. Parallels in the sequential organization of birdsong and human speech. *Nature communications*, 10(1):1–11, 2019. URL https://www.nature.com/articles/s41467-019-11605-y.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l6qiR5F7&source=post_page----------.

Mirac Suzgun, Sebastian Gehrmann, Yonatan Belinkov12, and Stuart M Shieber. Lstm networks can perform dynamic counting. *ACL 2019*, pp. 44, 2019.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJcKhk-Ab.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuan-Jing Huang. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1660–1669, 2016. URL https://www.aclweb.org/anthology/D16-1172/.

Xunjie Zhu, Tingfeng Li, and Gerard De Melo. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 632–637, 2018. URL https://www.aclweb.org/anthology/P18-2100/.

A Supplementary Material

A.1 Relationship between forget gate and timescale

In Section 2.3, we showed that the forget gate bias controls the timescale of the unit, and derived a distribution of assigned timescales for the multi-timescale language model. After training this model, we tested whether this control was successful by estimating the empirical timescale of each unit based on their mean forget gate values using Equation 2. Figure 4 shows that the assigned and estimated timescales in layer 2 are strongly correlated. This demonstrates that the timescale of an LSTM unit can be effectively controlled by the forget gate bias.

Figure 1 shows estimated timescales in the LSTM language model from Merity et al. (2018) trained on Penn Treebank (Marcus et al., 1999; Mikolov et al., 2011). These timescales lie between 0 and 150 timesteps, with more than 90% of timescales being less than 10 timesteps, indicating that this network skews its forget gates to process shorter timescales during training. This resembles the findings by Khandelwal et al. (2018), which showed that the model's sensitivity is reduced for information farther than 20 timesteps in the past. Ideally, we would like to control the timescale of each unit to counter this training effect and select globally a distribution that follows from natural language data.

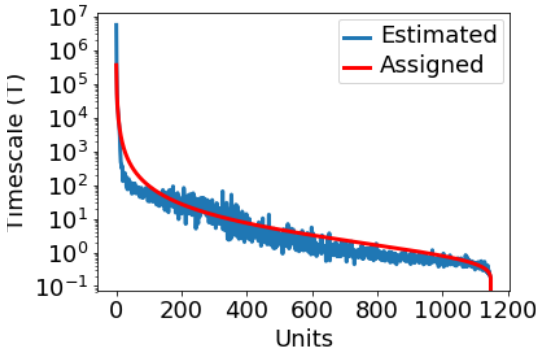
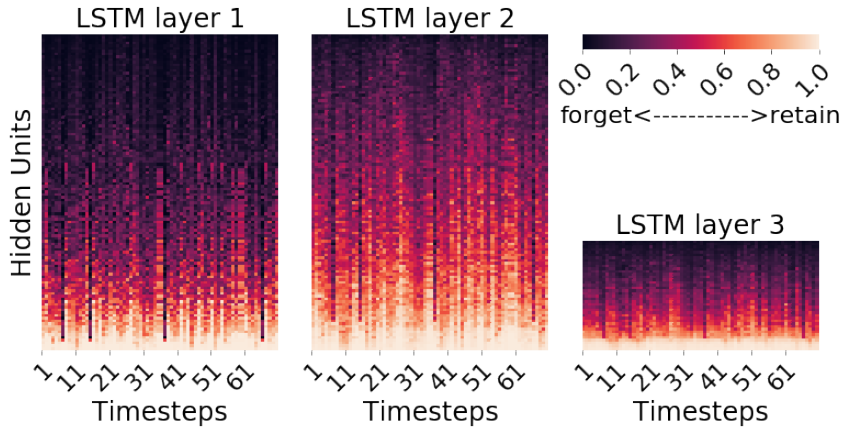
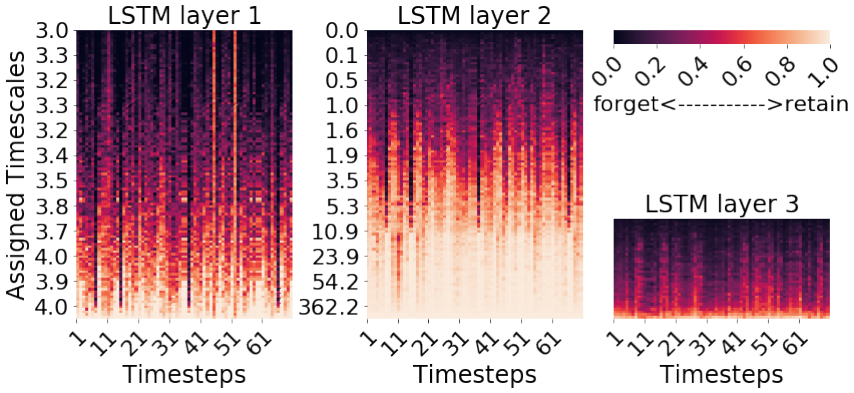


Figure 4: Estimated timescale is highly correlated with assigned timescale, shown for all 1150 units in LSTM layer 2 of the multi-timescale language model.

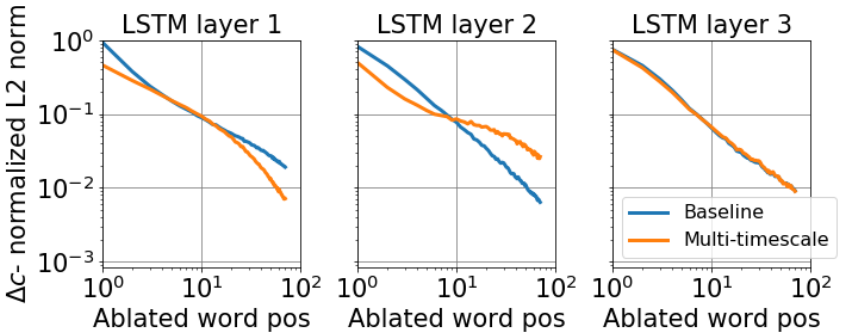


(a) Baseline Language Model

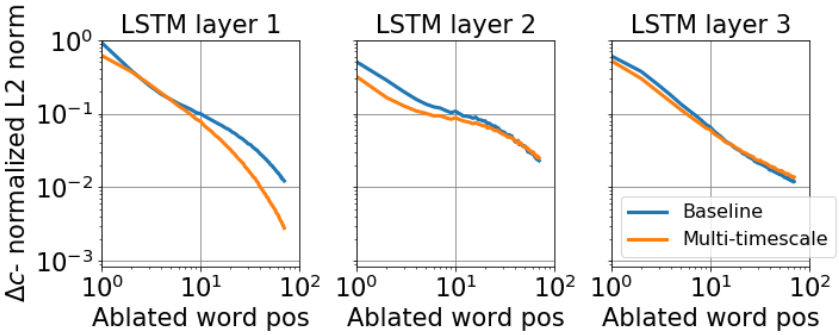


(b) Multi-timescale Language Model

Figure 5: Forget gate values of LSTM units for a test sentence from the PTB dataset. Units are sorted top to bottom by increasing mean forget gate value, and averaged in groups of 10 units to enable visualization. Figure 5b also shows average assigned timescale (rounded off) of the units.



(a) PTB dataset



(b) WikiText-2 dataset

Figure 6: Change in cell state of all the three layers for both the baseline and Multi-timescale language models in word ablation experiment. A curve with a steep slope indicates that cell state difference decays quickly over time, suggesting that the LSTM layer retains information of shorter timescales.

A.2 FORGET GATE VISUALISATION

To further examine representational timescales, we next visualized forget gate values of units from all three layers of both the multi-timescale and baseline language models as described in Section A.2. The goal is to compare the distribution of these forget gate values across the two language models, and to assess how these values change over time for a given input.

First, we sorted the LSTM units of each layer according to their mean forget gate values over a test sequence. For visualization purposes, we then downsampled these values by calculating the average forget gate value of every 10 consecutive sorted units for each timestep. Heat maps of these sorted and down-sampled forget gate values are shown in Figure 5. The horizontal axis shows timesteps (words) across a sample test sequence, and the vertical axis shows different units. Units with average forget gate values close to 1.0 (bottom) are retaining information across many timesteps, i.e. they are capturing long timescale information. Figure 5 shows that the baseline language model contains fewer long timescale units than the multi-timescale language model. They are also more evenly distributed across the layers than the multi-timescale language model. Figure 5b also shows the (approximate) assigned timescales for units in the multi-timescale language model. As expected, layer 1 contains short timescales and layer 2 contains a range of both short and long timescales. Layer 1 units with short (assigned) timescales have smaller forget gate values across different timesteps. In layer 2, we observe that units with large assigned timescale have higher mean forget gate values across different timesteps, for example the units with assigned timescale of 362 in 5b have forget gate values of almost 1.0 across all timesteps. Similar to the previous analysis, this demonstrates that our method is effective at controlling the timescale of each unit, and assigns a different distribution of timescales than the baseline model.

A.3 WORD ABLATION

Another way to interpret timescale of information retained by the layers is to visualize the decay of information in the cell state over time. We estimated this decay with word ablation experiments as described in Section A.2.

In Figure 6, we show the normalized cell state difference averaged across test sentences for all three layers of both baseline (blue) and multi-timescale (orange) models. In the PTB dataset, information in layer 1 of the baseline model decays more slowly than in layer 2. In this case, layer 2 of the baseline model retains shorter timescale information than layer 1. In the WikiText-2 dataset, the difference between layer 1 and layer 2 of the baseline model is inverted, with layer 2 retaining longer timescale information. However, in the multi-timescale model the trend is nearly identical for both datasets, with information in layer 2 decaying more slowly than layer 1. This is expected for our multi-timescale model, which we designed to have short timescale dependencies in layer 1 and longer timescale dependencies in layer 2. Furthermore, the decay curves are very similar across datasets for the multi-timescale model, but not for the baseline model, demonstrating that controlling the timescales gives rise to predictable behavior across layers and datasets. Layer 3 has similar cell state decay rate across both models. In both models, layer 3 is initialized randomly, and we expect its behavior to be largely driven by the language modeling task.

Next, we explored the rate of cell state decay across different groups of units in layer 2 of the multi-timescale language model. We first sorted the layer 2 units according to their assigned timescale and then partitioned these units into groups of 100 before estimating the cell state decay curve for each group. As can be seen in Figure 7, units with a shorter average timescale have faster decay rates, whereas units with longer average timescale have slower information decay. While the previous section demonstrated that our manipulation could control the forget gate values, this result demonstrates that we can directly influence how information is retained in the LSTM cell states.

A.4 SHAPE PARAMETER FOR INVERSE GAMMA DISTRIBUTION

We compared the performance of multi-timescale language model for different shape parameters in inverse gamma distribution. Figure 8 shows timescales assigned to LSTM units of layer 2 corresponding to different shape parameters. These shape parameters cover a wide range of possible timescale distribution to the units. Figure 9 shows that multi-timescale models performs best for $\alpha = 0.56$.

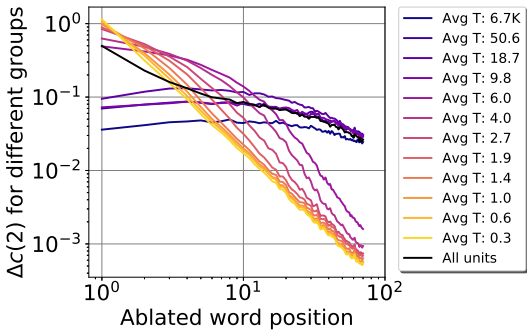


Figure 7: Change in cell states of 100-unit groups having different average timescale of layer 2 in Multi-timescale model in word ablation experiment for PTB dataset. As the assigned timescale to the group decreases the slope of the curve decreases indicating retained information of smaller timescale.

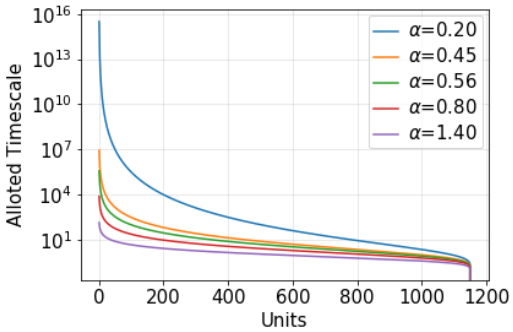


Figure 8: Assigned timescale to LSTM units of layer2 of multi-timescale language model corresponding to different shape parameter α.

A.5 Selecting the timescales for each layer

With the purpose to select proper timescales to each layer in Section 3.1, we conducted experiments on designing LSTM language models with different combinations of timescales across the three layers. We found that layer 1 (the closest layer to input) always prefers smaller timescales within the range from 1 to 5. This is consistent with what has been observed in literature: the first layer focuses more on syntactic information present in short timescales Peters et al. (2018); Jain & Huth (2018). We also observed that the layer 3, i.e., the layer closest to the output, does not get affected by the assigned timescale. Since we have tied encoder-decoder settings while training, layer 3 seems to learn global word representation with a specific timescale of information control by the training task (language modeling). The middle LSTM layer (layer 2) was more flexible, which allowed us to select specific distributions of timescales. Therefore, we achieve the Multi-timescale Language Model in Section 3.1 by setting layer 1 biases to small timescales, layer 2 to satisfy the inverse gamma distribution and thus aim to achieve the power-law decay of the mutual information, and layer 3 with freedom to learn the timescales required for the current task.

A.6 Robustness of model performance

We quantified the variability in model performance due to stochastic differences in training with different random seeds. Table 2 shows the mean perplexity and standard error across 5 different training instances. The variance due to training is similar across the two models.

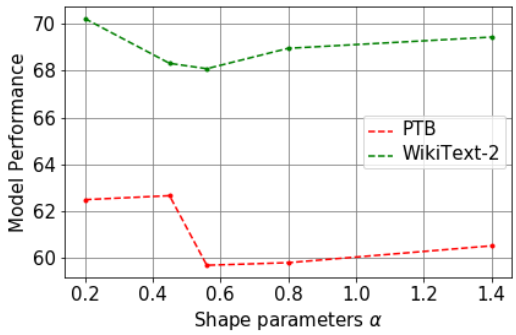


Figure 9: Performance of multi-timescale models for different shape parameters α on both PTB and WikiText-2 dataset.

Table 2: Perplexity of the baseline and multi-timescale models over 5 different training instances. Values are the mean and standard error over the training instances.

Dataset	**Model**	**Performance**
PTB	Baseline	61.64 ± 0.28
	Multi-timescale	59.63 ± 0.18
WikiText-2	Baseline	70.23 ± 0.24
	Multi-timescale	68.33 ± 0.12

A.7 Experiments on WikiText-2 dataset

We performed word ablation and information routing experiments for multi-timescale model trained on WikiText-2 dataset. Figure 10 shows timescale-dependent routing in the model, same as what we observed for PTB dataset in Section 3.2.5. We also explored the rate of cell state decay across units in layer 2 of the model with different assigned timescales. We can observe that assigned timescales to the units control information routing through them, similar to PTB dataset in Section 3.2.3.

A.8 The Dyck-2 grammar

The Dyck-2 grammar (Suzgun et al., 2019) is given as follows:

$$S \rightarrow \begin{cases} (S) & \text{with probability } p_1 \\ [S] & \text{with probability } p_2 \\ SS & \text{with probability } q \\ \varepsilon & \text{with probability } 1-(p_1+p_2+q) \end{cases}$$

where $0 < p_1, p_2, q < 1$ and $p_1 + p_2 + q < 1$.

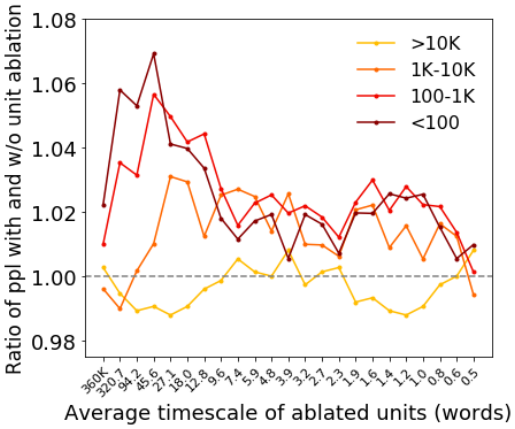


Figure 10: Information routing across different units of the multi-timescale LSTM for WikiText-2 dataset.

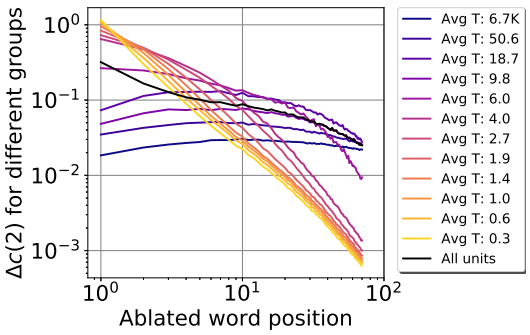


Figure 11: Change in cell states of 100-unit groups having different average timescale of layer 2 in Multi-timescale model in word ablation experiment for WikiText-2 dataset.