DISCLAIMER:

This document does not meet the
current format guidelines of
the Graduate School at
The University of Texas at Austin.

It has been published for
informational use only.

The Thesis Committee for Shivangi Mahto
Certifies that this is the approved version of the following Thesis:

Multi-timescale representation learning in LSTM Language Models

APPROVED BY
SUPERVISING COMMITTEE:

Alexander Huth, Supervisor

Greg Durrett

Multi-timescale representation learning in LSTM Language Models

by

Shivangi Mahto

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Computer Science

The University of Texas at Austin

May 2020

Acknowledgements

I would like to thank my guide Prof. Alexander Huth, Shailee Jain, Javier Turek, Vy Vo, and members of Huth lab for their constant support and guidance.

Abstract

Multi-timescale representation learning in LSTM Language Models

Shivangi Mahto, M.S.COMP.SC

The University of Texas at Austin, 2020

Supervisor: Alexander Huth

Representations within Language Models (LMs) are difficult to interpret. For example, how different layers of an LSTM LM retain information over different periods of time is unclear. In this paper, we present methods to interpret and control the timescale of information routing through an LSTM unit. We found out that a standard LSTM LM favors representations of small timescale information (up to 20 tokens). We then introduce a prior based on statistical properties of natural language, which is applied on the distribution of timescale across LSTM units to achieve an effective multi-timescale LM. The proposed model learns representations of both short as well long timescale. It also achieves better prediction performance than a standard LSTM LM on Penn Treebank and WikiText-2 datasets, especially on rare words.

Table of Contents

Chapter 1: Introduction

Natural language contains information at different time-scales. At the time-scale of milliseconds, acoustic features are changing whereas at sub-seconds words are changing. At the longer time-scales of multiple seconds, semantics, emotions or narratives are varying in the language. Irrespective of temporal change in information, the statistical laws of natural language is always obeyed including Zipf's law (Zipf, 1949), heap's law (Heaps, 1978) (Herdan, 1964) (Guiraud, 1954) and the nature of mutual information decay between two symbols as the distance between the two symbols increases (Lin, 2016).

LSTM-based language models (LM) learn compact representations of prior linguistic context to predict the next word of a given text sequence. LSTM LMs learn syntax (short timescale) (Lakretz, 2019) and semantic (long timescale) (Khandelwal, 2018) information in their representations. They are also capable of learning the statistical laws of natural language (Takahashi, 2017). Long Short-Term Memory (LSTM) (Hochreiter, 1997) have been widely used to model temporal data for language processing tasks including language modelling (Merity, 2017) (Melis, 2017), contextualized word embeddings (Peters), and sequence-to-sequence modeling (Sutskever, 2014) (Jia, 2016). The gating mechanism (Hochreiter, 1997) of LSTM controls how information is propagated across time through each LSTM unit of an LM.

However, the LSTM layers of LM lack interpretability. We know little about what kind of time-scale information is encoded in each layer. To the best of our knowledge, there is no prior work in literature to quantify the time-scale of information routing through an LSTM unit. Such interpretability is helpful to understand what is lacking in the current architecture and towards designing a more effective language

model. For applications such as understanding language processing in the human brain, the structure, format and interpretability of the representations learned by the language model is more important than its performance.

Another issue is that we cannot control the time-scale of information flowing through an LSTM unit. Currently, the LSTM units learn useful properties of natural language from scratch over the training. The control on information-flow through LSTM units can help us to introduce a regularizer towards learning power-law decay of mutual information between symbols (Lin, 2016).

In this work, we are proposing a mechanism to interpret and control the timescale of information encoded in each LSTM unit of a LSTM LM. Such mechanism is helpful to design representations of natural language at different timescales. We introduce a prior to make combined LSTM units act as an approximation of power-law behavior of memory decay. In this way, we can empower the models to learn other information like longer time-range information with restricted resources. It also helps us to design more interpretable layers in the model.

Our contribution:

1. We analyze existing LSTM LMs and show how different layers retain information across time.
2. We add a prior based on statistical properties of language that controls the timescales of different units in the network, making it a (slightly) more effective language model
3. We show that this prior creates interpretable representations that separate long and short timescale information into different parts of the network.

Chapter 2: Related work

There are two related streams of literature on controlling the time-scale of information flowing through a network over the time. One focuses on explicitly controlling the forgetting and updating time of recurrent neural networks. Hierarchical timescale-based RNN (Chung, 2016) applies a binary boundary detector at each layer which turns on and off at certain intervals depending on the level of abstraction of the layer. The presented mechanism is similar to the gated mechanism of LSTM with forget gate forced to be 0 or 1 from outside. Our approach differs from this concept as we do not explicitly control forget gate values over the time.

Another stream focuses on making the network forget periodically through the gating mechanism of LSTMs. Cached LSTM (Xu, 2016) divides memory into various groups with different forgetting rates and thus enables the network to keep long time-scale information within a recurrent unit. Our work also tries to control memory at different time-scale. However, we are introducing a prior through forget and input gates instead of changing the LSTM update equation. Ordered neurons LSTM (Shen, 2018) introduces an inductive bias through forget and input gates to learn hierarchical layer representations. The proposed LSTM uses a new softmax function to make the units remember different time-scale information. Unlike this work, we do-not constrains neurons to update in order. Chrono initialization (Tallec, 2018) presented a relation between forgetting time of an LSTM unit and the initialization values of its input and forget gate biases. Those authors propose that by choosing appropriate bias initialization, they can control the memory time-scale of an LSTM unit. The control mechanism we use here builds upon this work in the sense that we are also using forget and input gate biases

to control the memory of the LSTM unit. However, firstly, instead of merely controlling initialization, we permanently fix the forget and input biases. Secondly, our function to extract forget and input gate bias values for a given time-scale and the distribution of time-scales across LSTM units is different. Fixation of biases is important for us because it gives us control over time-scale of a unit unlike initialization. Quantized LSTMs (Ardakani, 2018) (Xu C. a., 2018) uses binary/ternary weights resulting in fast recurrent computations while maintaining competitive results to that of a usual LSTM. Our focus is little different from such works as we want multiple time-scale representations in the model.

The most widely used method for interpreting information encoded in LSTM representations is to perform various probe tasks using those representations. Such task-based experiments can reveal whether LSTM units are capable of learning short time-scale information of word order (i.e. syntactic information) (Adi, 2016) (Linzen, 2016) as well as long time-scale semantic information (Zhu, 2018) (Gulordava, 2018).

Another method to interpret LSTM layer representations is through encoding models for predicting brain activity elicited by natural language (Jain, 2018) (Toneva, 2019). Encoding models were trained with word representations produced by different layers of the language models. Then these models were used to analyze which layer representation can better predict which kind of time-scale activity across different brain areas. These experiments have shown that the middle layers of language models tend to contain the longest time-scale information. However, these methods cannot quantify the time-scale of information routed through each LSTM unit.

Chapter 3: Multi-timescale Language Models

3.1 Defining time-scale of information flowing through an LSTM unit

How can we quantify the time-scale of information flowing through an LSTM unit? As we know, the forget and input gates control the flow of information. 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 time-step is stored in memory for subsequent time-steps. Mathematically,

$$i(t) = \sigma(W_{ix}x(t) + W_{ih}h(t) + b_i)$$
$$f(t) = \sigma\big(W_{fx}x(t) + W_{fh}h(t) + b_f\big)$$
$$\tilde{c}(t) = tanh(W_{cx}x(t) + W_{ch}h(t) + b_c)$$
$$c(t) = f(t) * c(t-1) + i(t) * \tilde{c}(t)$$

where $\sigma(.)$ and $tanh(.)$ represent element-wise sigmoid and hyperbolic tangent functions. One way to examine the time-scale of LSTM representations is to consider the ``free-input regime'': if there is no input to LSTM after a certain time-step t_0, i.e. input $x(t) = 0$ for $t > t_0$. On ignoring information leakage from current time-step i.e. $W_{ch} = 0, b_c = 0 \; and \; W_{fh} = 0$,

$$c(t) = f(t) * c(t-1)$$

which for $t > t_0$ can be simplified as:

$$c(t) = f_0^{(t-t_0)} * c_0 = e^{\log(f_0)(t-t_0)} * c_0$$

where c_0 is the cell state of the LSTM unit at t_0, and $f_0 = \sigma(b_f)$ is the value of forget gate, which depends on the forget gate bias b_f. This equation shows that LSTM memory exhibits exponential decay with characteristic *forgetting time* $\mathrm{T} = \frac{-1}{\log f_0}$. That is, values in the cell state tend to shrink by a factor of e every T time-steps. Ignoring contributions of the input and hidden state, the forgetting time T depends only on the forget gate bias b_f:

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

We use this forgetting time to represent the time-scale of information that is represented by each LSTM unit.

3.2 Estimation of time-scale of information flowing through LSTM units of an LM

Now that we have discussed a way to interpret the time-scale of information flowing through an LSTM unit, one thing we can do is to estimate the time-scale of information flowing through different units and layers in a trained standard LSTM LM (Merity, 2017). This LM consists of 3 stacked LSTM layers and trained on Penn Treebank (Mikolov, 2010). We have used this LM as the baseline model for our experiments and a detailed explanation of architecture and training process can be found in Section 4. We estimated the time-scale of each unit as $\mathrm{T_{est}} = \frac{-1}{\log \bar{f}}$, where $\bar{f}$ here is the mean value of the forget gate across different time steps and test sentences.

As we can observe in Figure *1*,the estimated time-scale of the LSTM units lies between 0 to 160 with most having timescale less than 20. This shows LSTM units prefer

smaller time-scale information for the prediction task. While controlling the time-scale of each LSTM unit, we want to make sure to have enough units within this range.

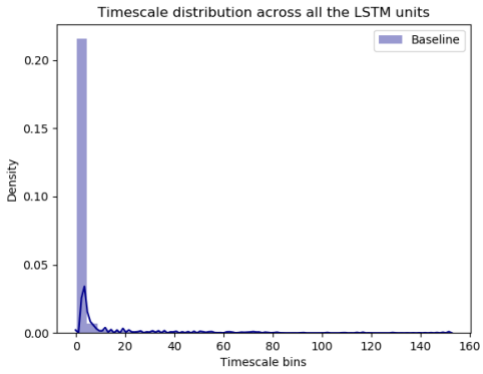


Figure 1: Distribution of estimated time-scale for the baseline model

3.3 Controlling time-scale of information

To get the desired forgetting time $T_{desired}$ and thus the desired time-scale for LSTM unit, we can set the bias of the forget gate as follows:

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

To keep a pace between forgetting an information from last time-step and adding new information from the current time-step in the cell state, we fixed bias values of input gate b_i to $-b_f$ such that the relation $i_t = 1 - f_t$ holds approximately true. These bias values for both gates are fixed and not learned during training.

3.4 How to assign time-scales to different LSTM units of the language model?

We want to assign time-scale to each LSTM unit of the language model such that they

help the model to perform better prediction, and also to create interpretable representations that contain information at different time-scales. We know that there are different types of information in natural language that evolve at different time-scales. All these different types of dependence can be summarized by looking at mutual information of tokens as a function of their separation (Lin, 2016). It has been observed that mutual information of tokens is a power-law decay function of the separation of tokens.

However, LSTM memory tends to decay exponentially. Hence, we need to get a power law decay out of exponentially decaying memory of LSTMs. One way to bridge this gap is to approximate the power law using a mixture of exponential functions. Practically this will be done by assigning different exponential time-scales to each unit in an LSTM. Let us assume that the time-scale T for each unit is drawn from a distribution $P(T)$. We want to define $P(T)$. 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:

$$t^{-d} = E_T\left[e^{-\frac{t}{T}}\right] = \int_0^\infty P(T)\, e^{-\frac{t}{T}}\, dT$$

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 $\alpha = d + 1$ and scale parameter $\beta = 1$.

We conducted experiments on designing LSTM LMs with different combinations of time-scales across the three LSTM layers. We found out that layer 1 i.e. the outermost layer (closest to input) always prefer a smaller timescale of range 1 to 5. This is

consistent with what has been observed in literature that first layer focuses more on syntactic information which is present in short time-scale (Peters) (Jain, 2018). We also observed that the layer 3 i.e. the innermost layer (closest to decoder) does not get affected by the assigned time-scale. Since we have tied encoder-decoder settings while training, it makes sense that layer 3 learns the global word representation and hence need a specific time-scale information irrespective of what we want. This left us with only middle LSTM layer (layer 2) where we can strategies time-scales such that we introduce a prior in LSTM units to learn the mutual information decay.

Thus, to achieve a Multi-time-scale Language Model, we have the outermost layer with units having bias values corresponding to small time-scales and the middle layers unit corresponding to inverse-gamma distributed time-scale. We do not control the time-scales of innermost layer and let it learn the required time-scales freely.

3.5 Visualizations to interpret the time-scale of each LSTM unit

In order to understand the effects of our manipulations, we used several techniques to interpret and visualize the time-scale of information passing through each LSTM unit.

1. Forget gate visualization: We visualized how forget gate values change over different time-steps for an input sentence. For each LSTM unit, we found out the mean forget gate value across different time-steps for a test sentence. We then sorted the units according to mean forget gate value and visualized forget gate values over different time-steps. Higher mean forget value implies that the unit will retain information from the past for longer time-steps and hence the stored information is long time-scale information. Such forget gate visualization gives a

sense of whether the assigned time-scale relates with the forget gate values of the unit or not.

2. Word ablation: The decay rate of the cell state of a layer indicates the time-scale of information passing through the layer. One way to visualize this decay rate is through the word ablation experiment during inference. In this experiment, we first ablate words which are k time-steps away w_{t-k} from the current time-step t in the input test sentence as shown in Eigure 2.We then observe the change in cell states of the three layers of the model with respect to the cell states without any ablation. Impact of ablating word w_{t-k} on cell state of layer i for predicting word $< w_{t+1} >$ is defined as Δc_i. $\Delta c_i(k)$ is the normalized L2 norm over the difference of cell states of layer i with and without ablation of word at k time-steps away from the current time-step *t* in the test sentence. It is defined as below:

$$\Delta\ \mathrm{c_i(k)} = \frac{|\mathrm{c_i'}\ (t)_\mathrm{k} - \mathrm{c_i(t)}|_2}{|c_i(t)|_2}$$

where $\mathrm{c_i'}\ (t)_\mathrm{k}$ and $\mathrm{c_i(t)}$ represents cell state vector of the layer i with and without any word ablation respectively.

In our experiments, we estimated $\Delta c_i(k)$ for $k$ranging from 0 to maximum length of the input test sentence. Lets understand the intuition behind this ablation experiment by an example. Suppose that we ablate a word w_{t-10} seen at timestamp $t-10$ while predicting the word w_{t+1} at timestamp $t+1$ during inference. If a layer of language model route long time-scale information, then it's cell state would significantly change as compared to the cell state with no ablation

of word w_{t-10}. This is because due to ablation the cell state would miss information contributed by w_{t-10}.

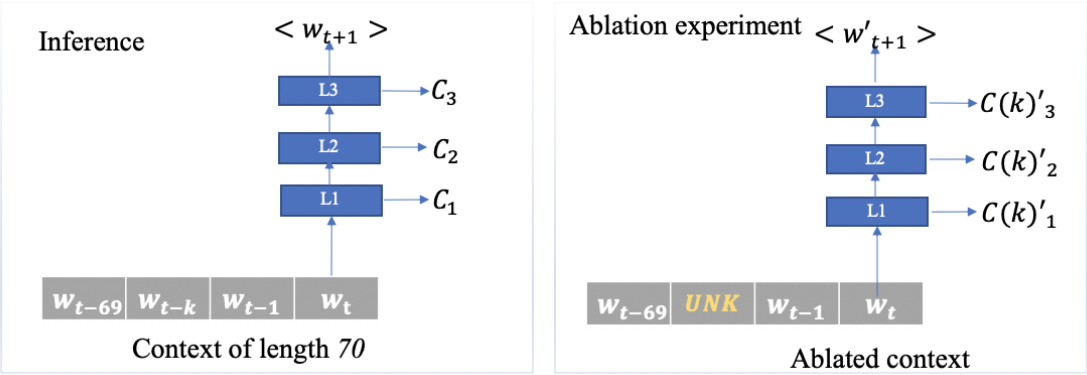


Figure 2 Word ablation experiment

Chapter 4: Evaluation

4.1 Experimental Setup

We experimentally evaluated the effectiveness of our proposed controlled time-scale mechanism for a language modeling task on Penn Treebank (PTB) (Mikolov, 2010) and WikiText-2 (WT2) (Bradbury, 2016) datasets. PTB contains a vocabulary of 10K unique words, with 0.93M tokens in the training, ~0.2M in validation, and ~82K in test data. WT2 is a larger dataset with a vocabulary size of 33k unique words, with almost double (~2M) tokens in the training, 0.22M in the validation, and 0.24M in the test dataset. In our experiments, for each of the two datasets, we have compared two language models: a standard stateful LSTM LM (Merity, 2017) as the baseline model and a multi-time-scale LM with time-scale controlled LSTM units.

The LM architectures consist of three LSTM layers with 1150 units in the first two layers and 400 units in the third layer, with an embedding size of 400. We tied the input and output embeddings. All models were trained using SGD followed by non-monotonically triggered ASGD for 1000 epochs. Training sentences had sequence length of 70 with a probability of 0.95 and of 35 sequence length with a probability of 0.05. During inference, all the test sentences are of 70 words long. All embedding weights were uniformly initialized in the interval $[-0.1, 0.1]$. All the weights and biases of hidden LSTM layers were initialized between $[-\frac{1}{H}, \frac{1}{H}]$ where H is the output size of the respective layer.

Multi-timescale LMs have the same architecture and training schedule as the baseline model except for the forget and input gate bias values for the first two LSTM layers as presented in Section 3. For layer 1, in order to get representations for short time-scale information, we assigned time-scale 3 to half of the units and time-scale 4 to the rest. Using equation 4, the corresponding forget and input gate bias values for the units with time-scale 3 were fixed to 0.92 and -0.92 whereas for the units with time-scale of 4, the forget and input gate bias values were fixed to 1.25 and -1.25.

For layer 2, we assigned time-scales to each unit by sampling from an Inverse Gamma distribution. These time-scales were then used to compute the forget and input biases for each unit. For selecting the best shape parameter for the inverse gamma distribution, we tested different values and found that 1.56 works best for our models. Hence, we carried out all our experiments with a shape parameter of 1.56 and a scale parameter of 1 of the inverse gamma distribution. This parameter allows 80% units of layer 2 to have time-scales between 1 to 20 and the rest the units with higher time-scale ranging up to the thousands. Note that we did not fix or set biases of any unit for the third LSTM layer as discussed in the previous section. For each dataset, we have one Multi-timescale model (MTS LM).

4.2 Experimental Results

4.2.1 Language modeling performance

We compared the performances of the baseline and Multi-timescale LMs for the language modeling task on PTB and WT2 datasets. Last column of Table *1* and Table *2*

shows the perplexity of both the LMs for the PTB and WT2 dataset respectively. We can see that the multi-timescale LMs outperformed the baseline models for both the datasets by an average margin of 1.75 perplexity.

Next, we observed how the language models' performance depends on the token's frequency in the training dataset. Information related to a frequent word lies in a short-timescale whereas for a rare word it is present in longer-timescale. This implies that a language model can predict a rare word only if it retains long time-scale information. This experiment shows how the language models retains information present at different time-scales.

We divided the tokens of the test dataset into 4 bins depending on their frequencies in the training dataset: a) above10K for words with frequency greater than 10,000 b) 1K-10K for words with frequency in between 1000 to 10,000 c) 100-1K for words with frequency in between 100 to 1000 and d) below-100 for extremely rare words with frequency less than 100. Then we compared performance of the models for test tokens belonging to each frequency bins as shown in the first four columns of Table *1* and Table *2*. For each frequency bin, multi-time-scale models performed better than baseline, especially for rare words. It shows that multi-time-scale models can retain information present in longer time-scales as compared to the baseline models.

Model	above10K	1K-10K	100-1K	below100	All tokens
Baseline	6.82	27.77	184.19	2252.50	61.40
Multi-time-scale	6.84	27.14	176.11	2100.89	59.70

Table 1: Performance of the multi-time-scale and baseline models for tokens across different frequency bins in PTB

Model	above10K	1K-10K	100-1K	below100	All tokens
Baseline	7.49	49.70	320.59	4631.08	69.88
Multi-time-scale	7.46	48.52	308.43	4381.72	68.08

Table 2: Performance of the multi-timescale and baseline models for tokens across different frequency bins in WT2

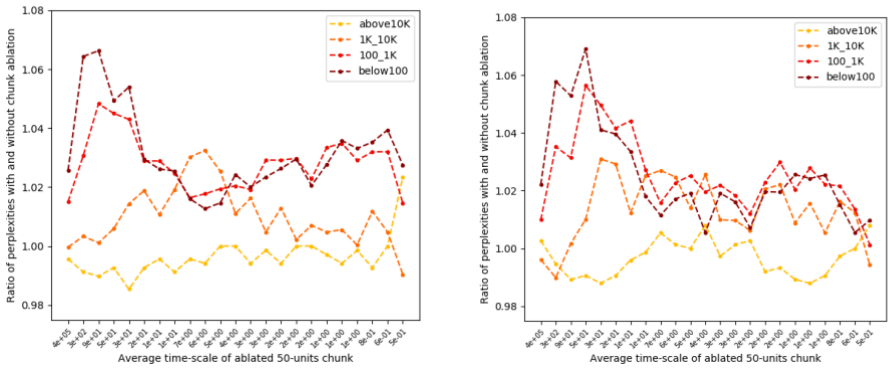
4.2.2 Routing of information through LSTM units with different time-scale

We observed earlier that multi-time-scale model is retaining information of the long time-scale. Next question is to find out which LSTM units are routing the longer-time-scale information? Is it related to the time-scales assigned to those units? For this, we analyzed how the information is routing through LSTM units having different time-scales. LSTM units across layer 2 of the multi-time-scale LMs have a wide variation of assigned timescales from almost 0 to thousands. Starting from the left-most end, we divided the layer 2 in 23 chunks of 50 consecutive LSTM units. We then looked into information routing through these chunks.

For each of the 23 chunks, we observed the performance of the multi-time-scale model after ablating that chunk in layer 2 while keeping the rest of the units active. To ablate a chunk of LSTM units, we explicitly make their output 0. To interpret the time-scale of blocked information because of ablation, we looked into model's performance for different frequency's bins. If the model's performance gets worse in a particular frequency bin then it implies that the ablated chunk route information corresponding to that time-scale. We then calculated the ratio of model's perplexity with and without ablation.

Figure 3 (a) and (b) shows this ratio across all the frequency bins for all the 23 chunks for PTB and WT2 datasets.

We can see that on ablating chunks with high average timescale, the model's performance degrades most for the low frequency bins (below100 and 1K-10K). By ablating units having medium range timescale, model's performance for tokens in medium frequency range (1k-10k) worsen. Similarly, for higher frequency words, ablating units with very small time-scale resulted in worse performance. This result shows that time-scale dependent information is routing through different units of a layer.



(a) PTB dataset (b) WT2 dataset

Figure 3: Ratio of perplexity of multi-timescale model with and without ablation of chunk having different average time-scales. Plots shows perplexity across different frequency bins.

4.2.3 Forget gate visualization

We visualized forget gate values of the LSTM units of all the three layers of both multi-time-scale and baseline LMs. Goal is to interpret time-scales of information routing through these units as discussed in Section 3.5. Firstly, we sorted LSTM units of each layer by their mean forget gate values over different time-steps of a test sentence. For

visualization, we calculated average forget gate values of every 10 consecutive sorted units for each time-step. We then plotted heat maps of these down-sampled units as shown in Figure *4* and Figure *5*. X-axis shows the time-steps across test sentence. Y-axis shows different down-sampled units, with increasing averaged forget gate values over time-steps from top to bottom.

Figure *4* shows that all the three layers of the baseline LM contains few units with almost one forget gate values over different time-steps. This shows that these time-steps retain information for longer duration and hence capture long time-scale information.

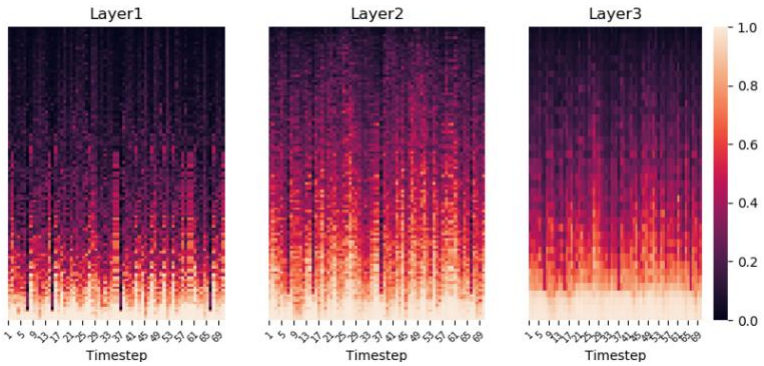


Figure 4: Visualization of forget gate values of LSTM units in Baseline LM for a test sentence of PTB dataset

On Y-axis of Figure *5*, we are also showing the allotted time-scale to the LSTM units of multi-time-scale LM. We can observe that on moving along y-axis from small to large average forget gate values, the assigned time-scales of the units are also increasing. This signifies that assigned time-scale and the time-scale of information retained by LSTM units are highly correlated.

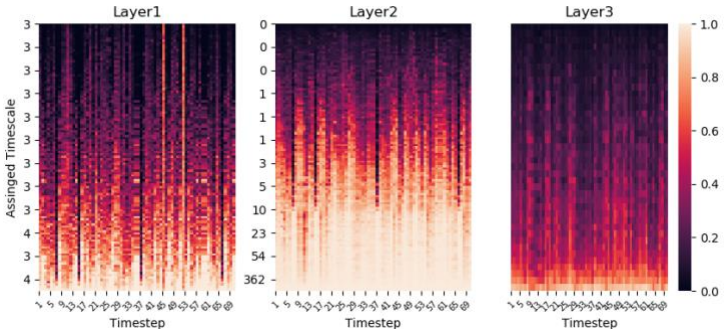


Figure 5: Visualization of forget gate values of LSTM units in Multi-time-scale LM for a test sentence of PTB dataset

4.2.4 Relationship between forget gate and time-scale

We mathematically derived in Section 3 that forget gate bias controls the time-scale of the unit. We checked this assertion by experimentally estimating the time-scale of different units in layer 1 and layer 2 of Multi-timescale LM. To estimate the time-scale we applied $\mathrm{T}_{\mathrm{est}} = \frac{-1}{\log \bar{f}}$ (discussed in Section 3.2) where f_0 is the mean forget gate value of the unit across different time-steps and test sentences as shown in Figure *6*. We can see a strong correlation between allotted and estimated time-scale of the units. This plot shows that the time-scale of a unit is effectively controlled by the forget gate bias.

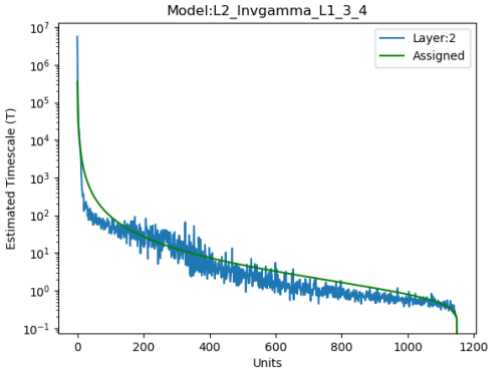


Figure 6: Estimated vs. assigned time-scale of units of Layer 2 of Multi-timescale LMs

4.2.5 Word ablation:

Another way to interpret time-scale of information retained by layers of a language model is to visualize cell state decay over time. We can visualize cell state decay is through word ablation experiments as explained in Section 3.5.

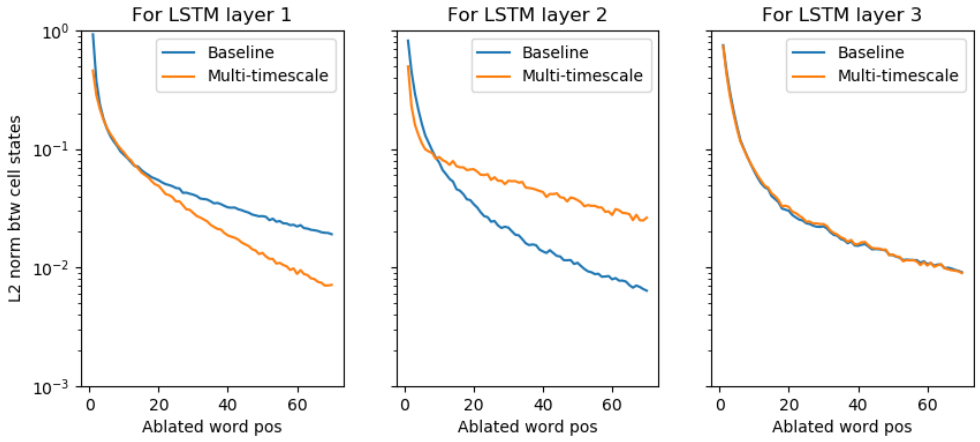


Figure 7: Change in cell state of all the three layers for both the baseline and Multi-timescale LMs in word ablation experiment for PTB datasets.

We estimated the normalized cell state difference $\Delta c_i(k)$ for k ranging from 0 to 69 i.e. across all the time-steps of the input test sentence and averaged over all the 1177 test sentences for PTB datasets and 3508 test sentences for WT2 dataset. We then plotted the normalized cell state difference for all the three layers of both the multi-time-scale and baseline models as shown in Figure *7* and Figure *8*. For the baseline model, layer 1 shows slow decay in cell state as compared to layer 2. This shows that the layer 2 retain information for smaller time-scale as compared to layer 1. This trend is reversed, however, for the multi-timescale model. In its layer 2, we can see that cell state decay rate is low as compared to layer 1. This is according to our expectation because layer 1

has units with small time-scale range (3-4) whereas layer 2 also has units with very long timescale. Layer 3 has almost the same cell state decay rate across both models. Again this is expected because layer 3 in both the models is initialized randomly and majorly driven by the LM task. This experiment shows that we could control the time-scale of information retained in different layers of multi-time-scale model.

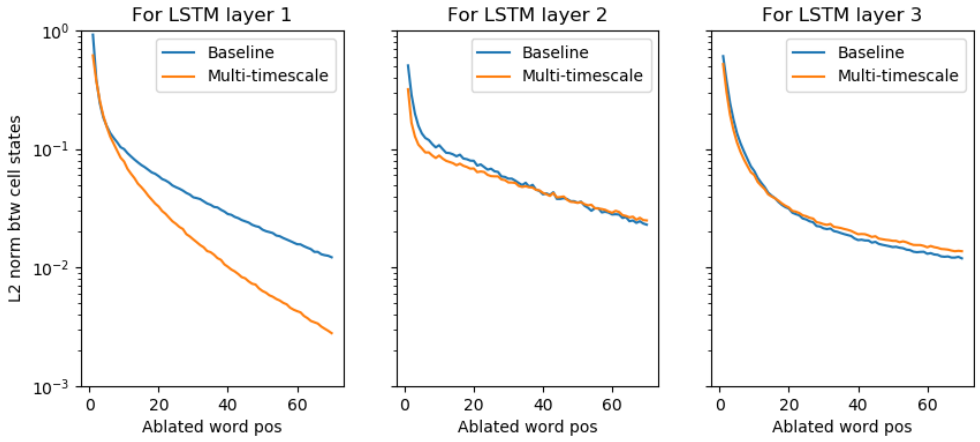
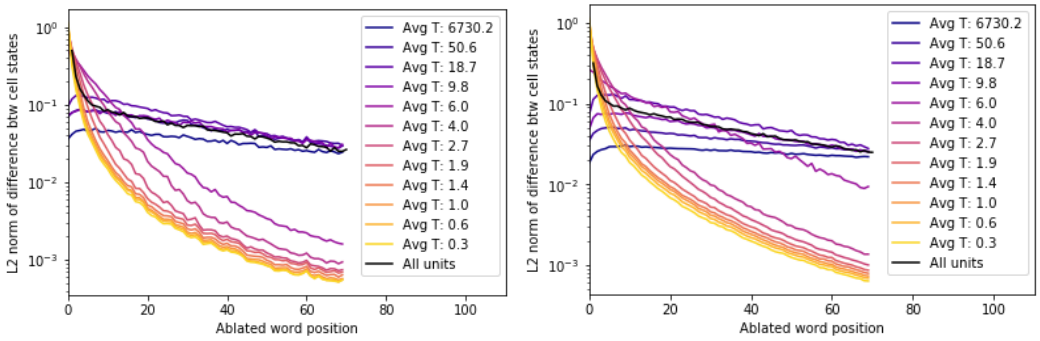


Figure 8: Change in cell state of all the three layers for both the baseline and Multi-timescale LMs in word ablation experiment for WT2 dataset.

Next, we explored the cell state decay rate across units with different time-scale of layer 2 of the Multi-timescale model. Since LSTM units across layer 2 has wide variation of time-scale ranging from almost 0 to 40000. It is important to analyze whether the decay rate of these units actually depends on the assigned time-scale or not?

We estimated the cell state decay for every consecutive 100-unit chunk of layer 2. Now our cell state vector is of 100-dimension for a given chunk instead of 1150 and the rest of calculation of normalized l2 norm of difference of cell state is the same as above.

As we can see in Figure 9 chunks with smaller average time-scale have faster decay whereas chunks with long averaged time-scale have slower memory decay. These results justify our claim that we effectively controlled the time-scales of information flowing through these units.



(a) PTB dataset (b) WT2 dataset

Figure 9: Change in cell states of 100-unit chunks having different average time-scale of layer 2 in Multi-time-scale model in word ablation experiment

4.2.6 Performance robustness

We trained both models for both datasets 5 different times with different random seeds to numpy, torch, and cuda. Performance of the models are shown first column of Table *3* and Table *4*. We observed a small variance across different random seeds. We also performed bootstrapping on the test samples. We divided the test document into bins of 100 word-long sentences and obtained around ~820 such bins for the PTB test set and ~ for the WIKI dataset. We created 10K datasets by bootstrapping samples from these 820 bins and calculated the mean and variance of both the models' performances on these datasets as shown second column of Table *3* and Table *4*. We can observe similar variance for both the models with smaller mean for Multi-timescale models. We also

plotted the distribution of perplexity difference between Multi-timescale and the baseline model performance for all the bootstrapped 10K datasets and observed that for each of them, Multi-timescale performed better than the baseline.

Model	Different training instances	Bootstrapped Datasets
Baseline	61.64+/- 0.28	61.42 +/- 1.52
Multi-time-scale	59.94 +/- 0.15	59.71 +/- 1.48

Table 3: Averaged performance of the Multi-timescale and baseline models over five different training instances and over 10K bootstrapped datasets for PTB dataset.

Model	Different training instances	Bootstrapped Datasets
Baseline	70.04 +/- 0.24	69.88 +/- 0.86
Multi-time-scale	68.33+/- 0.12	68.07 +/- 0.84

Table 4: Averaged performance of the Multi-timescale and baseline models over five different training instances and over 10K bootstrapped datasets for WT2 dataset

Chapter 5: Conclusion

In this work, we presented a mechanism to interpret and control the time-scale of information routing through an LSTM unit via input and forget gate biases. We first interpret the time-scale of information flowing through a standard LSTM LM and found out that most of the LSTM units prefer small time-scale information. We then designed a multi-timescale LM with controlled forgetting of the LSTM units in first two layers. First layer was maintained to route smaller time-scale information of last 3-4 tokens. Middle layer of multi-timescale LM was assigned time-scale sampled from an inverse gamma distribution in order to introduce a prior based on statistical property of nature language. Our model was successful to learn representation of various time-scales including longer time-scales than a standard LSTM LM. Our proposed model also outperformed the standard model for prediction task on both Penn Treebank and WikiText-2 datasets. We also presented forget gate visualization, unit and word ablation experiments to interpret time-scale of LSTM units and layers in the language models.

BIBLIOGRAPHY

Adi, Y. a. (2016). Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

Ardakani, A. a. (2018). Learning recurrent binary/ternary weights. *arXiv preprint arXiv:1809.11086*.

Bradbury, J. a. (2016). Quasi-recurrent neural network. *arXiv preprint arXiv:1611.01576*.

Chung, J. a. (2016). Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Guiraud, P. (1954). Les caract{\`e}res statistiques du vocabulaire. Presses universitaires de France.

Gulordava, K. a. (2018). Colorless green recurrent networks dream hierarchically. *arXiv preprint arXiv:1803.11138*.

Heaps, H. S. (1978). Information retrieval, computational and theoretical aspects. Academic Press.

Herdan, G. (1964). Quantitative linguistics. Butterworth.

Hochreiter, S. &. (1997). Long short-term memory. *Neural computation, 9(8)*, 1735--1780.

Jain, S. a. (2018). Incorporating context into language encoding models for fmri. *Advances in neural information processing systems*, 6628--6637.

Jia, R. a. (2016). Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Khandelwal, U. a. (2018). Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*.

Lakretz, Y. a. (2019). The emergence of number and syntax units in LSTM language models. *arXiv preprint arXiv:1903.07435*.

Lin, H. W. (2016). Critical behavior from deep dynamics: a hidden dimension in natural language. *arXiv preprint arXiv:1606.06737*.

Linzen, T. a. (2016). Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics, 4*, 521--535.

Melis, G. a. (2017). On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Merity, S. a. (2017). Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*.

Mikolov, T. a. (2010). Recurrent neural network based language mode. *Eleventh annual conference of the international speech communication association*.

Peters, M. E. (n.d.). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Shen, Y. a. (2018). Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv preprint arXiv:1810.09536*.

Sutskever, I. a. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104--3112).

Takahashi, S. a.-I. (2017). Do neural nets learn statistical laws behind natural language. Public Library of Science.

Tallec, C. a. (2018). Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*.

Toneva, M. a. (2019). Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain). *Advances in Neural Information Processing Systems*, 14928--14938.

Xu, C. a. (2018). Alternating multi-bit quantization for recurrent neural networks. *arXiv preprint arXiv:1802.00150*.

Xu, J. a. (2016). Cached long short-term memory neural networks for document-level sentiment classification. *arXiv preprint arXiv:1610.04989*.

Zhu, X. a. (2018). Exploring semantic properties of sentence embeddings. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics , 2*, pp. 632--637.

Zipf, G. K. (1949). Human behavior and the principle of least effort. addison-wesley press.

Vita

Shivangi Mahto was born in a town of Madhya Pradesh, India. She has completed her Bachelor and Master of Technology in Electrical Engineering from IIT Bombay in 2015. After completing her graduation, she moved to Japan for working as a researcher at NEC Corporation. In September 2018, she entered the Graduate School at the University of Texas at Austin.

Email: shivi1701@gmail.com

This dissertation was typed by the author.