Transformer-XL

Attentive Language Models Beyond a Fixed-Length Context

Zihang Dai*[1], Zhilin Yang*[2], Yiming Yang[1], Jaime Carbonell[1],
Quoc V. Le[2], Ruslan Salakhutdinov[1]
[1]Carnegie Mellon University, [2]Google Brain
{dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, {zhiliny,qvl}@google.com

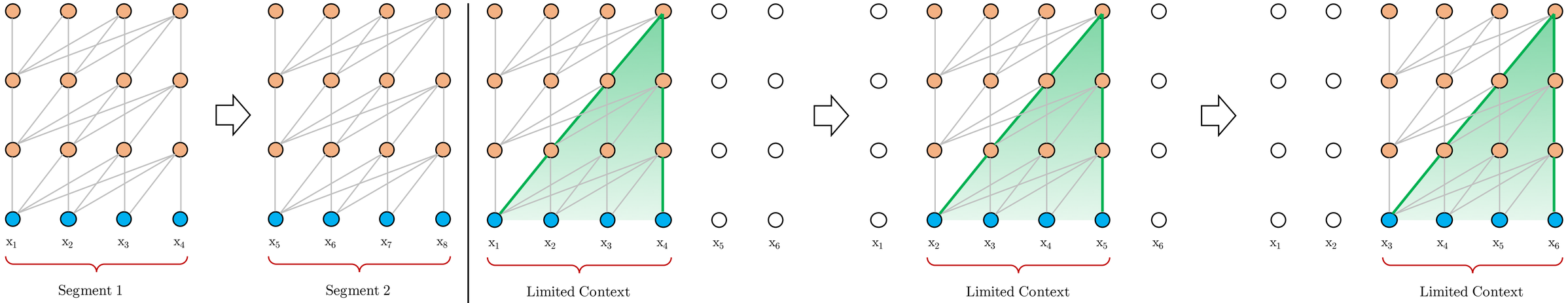
Introduction

- It is difficult to modeling long-term dependency for language models
  - Previous work: LSTM use 200 context words on average
- Attention mechanism enables the learning of long-term dependency
  - Transformer networks outperform LSTMs by a large margin for character-level language modeling. (Al-Rfou et al. 2018)
  - Despite the success, Transformer models face **context fragmentation** problem

Context Fragmentation

- LM training is performed on separated fixed-length segments
  - No information flow across segments
  - Model can’t capture dependency beyond the predefined context length
- Fixed-length segments are created by selecting a consecutive chunk of symbols without respecting the sentence or any other semantic boundary
- As a result, model lacks necessary contextual information for the prediction of first few symbols.

Vanilla Models



(a) Training phase.

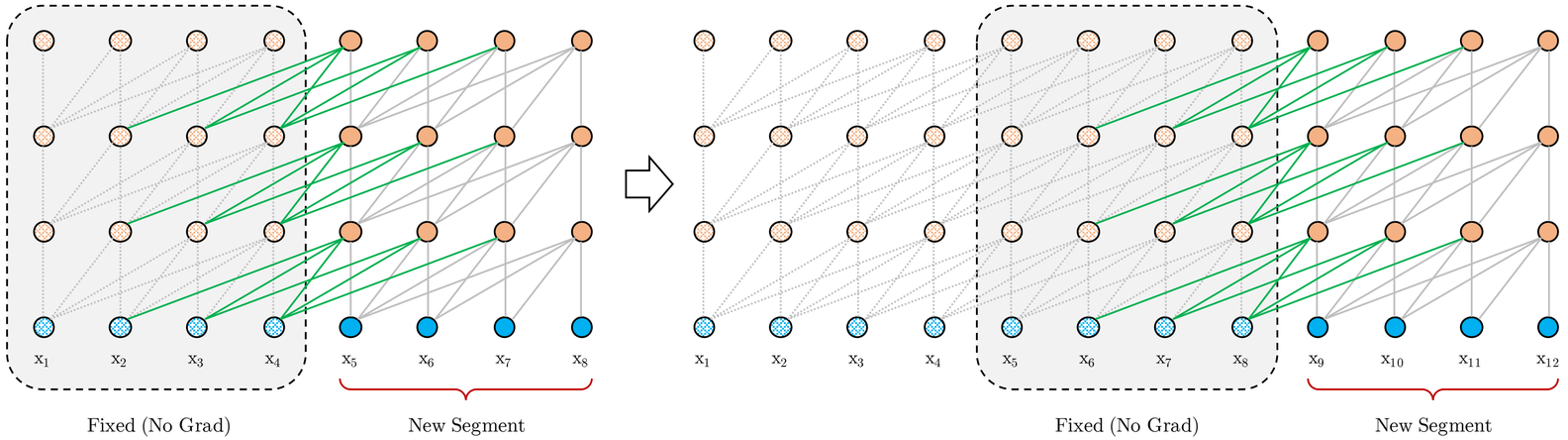
(b) Evaluation phase.

Vanilla Models

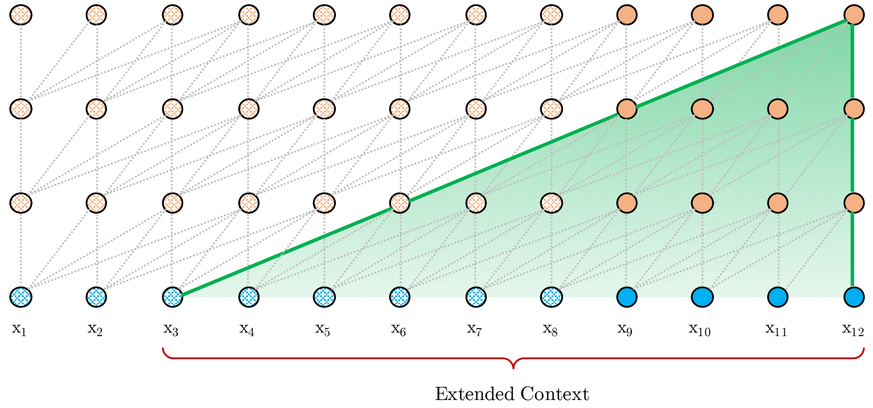
- Limitations:
  - Largest possible dependency length is upper bounded by the segment length
  - Simply chunking a sequence into fixed-length segments will lead to context fragmentation problem
  - Evaluation procedure is extremely expensive

Transformer-XL

- Segment-level recurrence with state reuse
- Hidden state of previous segment is fixed and cached to be reused



(a) Training phase.



(b) Evaluation phase.

Transformer-XL

- Formulation:
  - Two consecutive segments of length L :

$$\mathbf{s}_\tau = [x_{\tau,1}, \cdots, x_{\tau,L}] \text{ and } \mathbf{s}_{\tau+1} = [x_{\tau+1,1}, \cdots, x_{\tau+1,L}]$$

  - The n-th layer hidden state sequence produced for the τ-th segment $\mathbf{s}_\tau$:

$$\mathbf{h}_\tau^n \in \mathbb{R}^{L \times d}$$

  - The n-th layer hidden state for segment $\mathbf{s}_{\tau+1}$:

$$\begin{aligned}
\widetilde{\mathbf{h}}_{\tau+1}^{n-1} &= [\text{SG}(\mathbf{h}_\tau^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}], && \text{(extended context)} \\
\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n &= \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_q^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_k^\top, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_v^\top, && \text{(query, key, value vectors)} \\
\mathbf{h}_{\tau+1}^n &= \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n\right). && \text{(self-attention + feed-forward)}
\end{aligned}$$

Transformer-XL

- Benefits of recurrence mechanism:
  - The effective context being utilized can go way beyond just two segments. O(L+N)
  - Faster evaluation: the representations from the previous segments can be reused instead of being computed from scratch. Up to 1800+ faster than vanilla model
  - We can cache as many previous segments as the GPU memory allows

Positional Encoding

- Challenge: how to keep the positional information coherent when we reuse the states?

- Standard Transformer: $\mathbf{U} \in \mathbb{R}^{L_{\max} \times d}$

  the i-th row $\mathbf{U}_i$ corresponds to the i-th *absolute* position within a segment

$$\mathbf{h}_{\tau+1} = f(\mathbf{h}_{\tau}, \mathbf{E}_{\mathbf{s}_{\tau+1}} + \mathbf{U}_{1:L}) \quad \text{and} \quad \mathbf{h}_{\tau} = f(\mathbf{h}_{\tau-1}, \mathbf{E}_{\mathbf{s}_{\tau}} + \mathbf{U}_{1:L})$$

  - $\mathbf{E}_{\mathbf{s}_{\tau}}$ and $\mathbf{E}_{\mathbf{s}_{\tau+1}}$ are associated with the same positional encoding

  - The model has no information to distinguish the positional difference between $x_{\tau,j}$ and $x_{\tau+1,j}$

Relative Positional Encoding

- Idea: only encoding the relative positional information in the hidden states
- Instead of adding PE statically into embedding, we can inject the positional information dynamically into attention score of each layer

  $\mathbf{R} \in \mathbb{R}^{L_{\max} \times d}$, where the i-th row $\mathbf{R}_i$ indicates a relative distance of i between two positions

- $$\mathbf{A}^{\text{abs}}_{i,j} = q_i^\top k_j = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}$$

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}$$

Relative Positional Encoding

- Each term has an intuitive meaning:
  - (a) content-based addressing
  - (b) content-dependent positional bias
  - (c) global content bias
  - (d) global positional bias

$$\mathbf{A}^{\text{abs}}_{i,j} = q_i^\top k_j = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(b)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{E}_{x_j}}_{(c)} + \underbrace{\mathbf{U}_i^\top \mathbf{W}_q^\top \mathbf{W}_k \mathbf{U}_j}_{(d)}$$

$$\mathbf{A}^{\text{rel}}_{i,j} = \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(a)} + \underbrace{\mathbf{E}_{x_i}^\top \mathbf{W}_q^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(b)} + \underbrace{u^\top \mathbf{W}_{k,E} \mathbf{E}_{x_j}}_{(c)} + \underbrace{v^\top \mathbf{W}_{k,R} \mathbf{R}_{i-j}}_{(d)}$$

Transformer-XL

- Summary

For $n = 1, \ldots, N$:

$$
\begin{aligned}
\widetilde{\mathbf{h}}_{\tau}^{n-1} &= \left[\text{SG}(\mathbf{m}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau}^{n-1}\right] \\
\mathbf{q}_{\tau}^{n}, \mathbf{k}_{\tau}^{n}, \mathbf{v}_{\tau}^{n} &= \mathbf{h}_{\tau}^{n-1}\mathbf{W}_{q}^{n\top}, \widetilde{\mathbf{h}}_{\tau}^{n-1}\mathbf{W}_{k,E}^{n}{}^{\top}, \widetilde{\mathbf{h}}_{\tau}^{n-1}\mathbf{W}_{v}^{n\top} \\
\mathbf{A}_{\tau,i,j}^{n} &= \mathbf{q}_{\tau,i}^{n}{}^{\top}\mathbf{k}_{\tau,j}^{n} + \mathbf{q}_{\tau,i}^{n}{}^{\top}\mathbf{W}_{k,R}^{n}\mathbf{R}_{i-j} + u^{\top}\mathbf{k}_{\tau,j} + v^{\top}\mathbf{W}_{k,R}^{n}\mathbf{R}_{i-j} \\
\mathbf{a}_{\tau}^{n} &= \text{Masked-Softmax}(\mathbf{A}_{\tau}^{n})\mathbf{v}_{\tau}^{n} \\
\mathbf{o}_{\tau}^{n} &= \text{LayerNorm}(\text{Linear}(\mathbf{a}_{\tau}^{n}) + \mathbf{h}_{\tau}^{n-1}) \\
\mathbf{h}_{\tau}^{n} &= \text{Positionwise-Feed-Forward}(\mathbf{o}_{\tau}^{n})
\end{aligned}
$$

Experiments

- Word-level language modeling:

Model	#Params	Validation PPL	Test PPL
Grave et al. (2016b) – LSTM	-	-	48.7
Bai et al. (2018) – TCN	-	-	45.2
Dauphin et al. (2016) – GCNN-8	-	-	44.9
Grave et al. (2016b) – LSTM + Neural cache	-	-	40.8
Dauphin et al. (2016) – GCNN-14	-	-	37.2
Merity et al. (2018) – 4-layer QRNN	151M	32.0	33.0
Rae et al. (2018) – LSTM + Hebbian + Cache	-	29.7	29.9
Ours – Transformer-XL Standard	151M	**23.1**	**24.0**
Baevski & Auli (2018) – adaptive input$^{\diamond}$	247M	19.8	20.5
Ours – Transformer-XL Large	257M	**17.7**	**18.3**

Table 1: Comparison with state-of-the-art results on WikiText-103. $^{\diamond}$ indicates contemporary work.

Experiments

- Unprocessed character-level language modeling:

Model	#Params	Test bpc
Ha et al. (2016) – LN HyperNetworks	27M	1.34
Chung et al. (2016) – LN HM-LSTM	35M	1.32
Zilly et al. (2016) – Recurrent highway networks	46M	1.27
Mujika et al. (2017) – Large FS-LSTM-4	47M	1.25
Krause et al. (2016) – Large mLSTM	46M	1.24
Knol (2017) – cmix v13	-	1.23
Al-Rfou et al. (2018) – 12-layer Transformer	44M	1.11
Ours – 12-layer Transformer-XL	41M	**1.06**
Al-Rfou et al. (2018) – 64-layer Transformer	235M	1.06
Ours – 18-layer Transformer-XL	88M	1.03
Ours – 24-layer Transformer-XL	277M	**0.99**

Table 2: Comparison with state-of-the-art results on enwiki8.

Experiments

- Processed character-level language modeling:

Model	#Params	Test bpc
Cooijmans et al. (2016) – BN-LSTM	-	1.36
Chung et al. (2016) – LN HM-LSTM	35M	1.29
Zilly et al. (2016) – Recurrent highway networks	45M	1.27
Krause et al. (2016) – Large mLSTM	45M	1.27
Al-Rfou et al. (2018) – 12-layer Transformer	44M	1.18
Al-Rfou et al. (2018) – 64-layer Transformer	235M	1.13
Ours – 24-layer Transformer-XL	277M	**1.08**

Table 3: Comparison with state-of-the-art results on text8.

Experiments

- Short-term dependency only:

Model	#Params	PPL
Shazeer et al. (2014) – Sparse Non-Negative	33B	52.9
Chelba et al. (2013) – RNN-1024 + 9 Gram	20B	51.3
Jozefowicz et al. (2016) – LSTM-2048-512	0.83B	43.7
Kuchaiev & Ginsburg (2017) – BIG G-LSTM-2	-	36.0
Dauphin et al. (2016) – GCNN-14 bottleneck	-	31.9
Jozefowicz et al. (2016) – LSTM-8192-1024	1.8B	30.6
Jozefowicz et al. (2016) – LSTM-8192-1024 + CNN Input	1.04B	30.0
Shazeer et al. (2017) – Low-Budget MoE	∼5B	34.1
Shazeer et al. (2017) – High-Budget MoE	∼5B	28.0
Shazeer et al. (2018) – Mesh Tensorflow	4.9B	24.0
Baevski & Auli (2018) – Adaptive Input Large$^{\diamond}$	0.46B	24.1
Baevski & Auli (2018) – Adaptive Input Very Large$^{\diamond}$	1.0B	23.7
Ours – Transformer-XL Base	0.46B	23.5
Ours – Transformer-XL Large	0.8B	**21.8**

Table 4: Comparison with state-of-the-art results on One Billion Word. $^{\diamond}$ indicates contemporary work.

Experiments

- Small dataset:

Model	#Params	Dev PPL	Test PPL
Inan et al. (2016) – Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2016) – Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	25M	-	64.0
Merity et al. (2017) – AWD-LSTM	24M	60.7	58.8
Pham et al. (2018) – Efficient NAS	24M	60.8	58.6
Liu et al. (2018) – Differentiable NAS	23M	58.3	56.1
Yang et al. (2017) – AWD-LSTM-MoS	22M	58.08	55.97
Melis et al. (2018) – 2-layer skip-LSTM + dropout tuning	24M	57.1	55.3
Ours – Transformer-XL	24M	**56.72**	**54.52**
Merity et al. (2017) – AWD-LSTM + finetuning†	24M	60.0	57.3
Yang et al. (2017) – AWD-LSTM-MoS + finetuning†	22M	**56.54**	**54.44**

Table 5: Comparison with state-of-the-art results on Penn Treebank. † indicates using two-step finetuning.

Ablation Study I

- Ablation study on WikiText-103:

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	**27.02**	**26.77**	**500**
-	✓	Shaw et al. (2018)	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗	Ours	Full	29.59	29.02	260
-	✗	Ours	Half	30.10	30.10	120
-	✗	Shaw et al. (2018)	Full	29.75	29.75	120
-	✗	Shaw et al. (2018)	Half	30.50	30.50	120
-	✗	Vaswani et al. (2017)	Half	30.97	30.97	120
Transformer (128M)†	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
					23.09	**640**
Transformer-XL (151M)	✓	Ours	Full	23.43	23.16	450
					23.35	300

Ablation Study II

- Isolating the effects of resolving the context fragmentation problem from the benefit of capturing longer context length.

- Deliberately choose a dataset that does not require long-term dependency, so that any improvement from establishing the recurrence can be attributed to solving the context fragmentation.

Method	PPL
Ours	**25.2**
With Shaw et al. (2018) encodings	25.7
Without recurrence	27.1

Table 7: Ablation study on One Billion Word, a dataset without long-term dependency.

Relative Effective Context Length

- **Effective Context Length**: the longest length to which increasing the context span would lead to a gain more than a threshold.

- ECL ignores that it is harder to get improvement when a model already achieves a lower perplexity using only a shorter context

- **RECL**: defined on a group of models, gain of a long context is measure by the relative improvement over the *best* short context model

Relative Effective Context Length

Model	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	**900**	**800**	**700**
QRNN	500	400	300
LSTM	400	300	200
Transformer-XL 128M	**700**	**600**	**500**
- use Shaw et al. (2018) encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

Table 8: Relative effective context length (RECL) comparison. See text for the definition of RECL and r. The first three models and the last four models are compared as two *model groups* when we calculate RECL (RECL is computed on a model group rather than a single model). Each group has the same parameter budget.

Parameter r means constraining the comparison on top-r hard examples.

Evaluation Speed

Attn Len	How much Al-Rfou et al. (2018) is slower than ours
3,800	1,874x
2,800	1,409x
1,800	773x
800	363x

Table 9: Slowdown in terms of computational time during evaluation. Evaluation is based on per-token time on one GPU.