Transformer-XL: Attentive Language Models

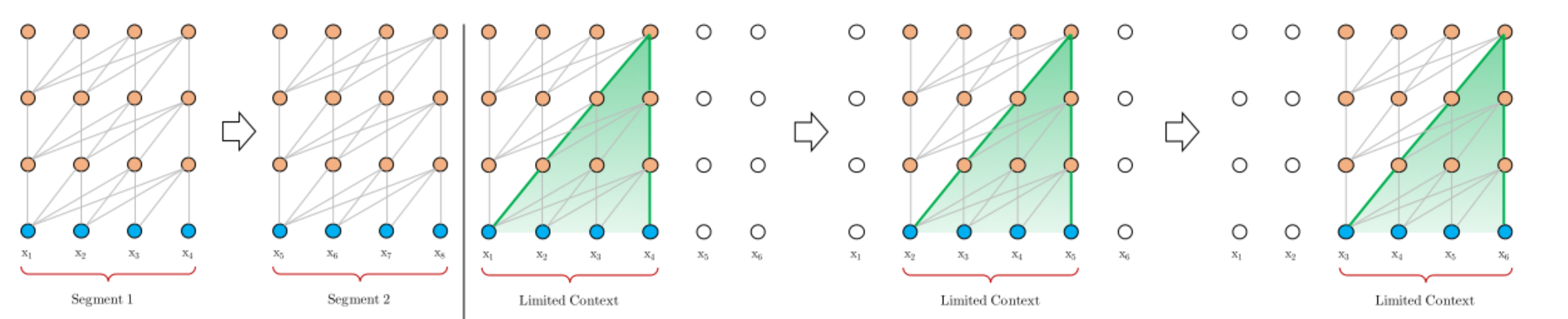
Beyond a Fixed-Length Context

최우진

2022/05/11

1. Introduction

- 초기 인공지능 언어 모델링: RNN, LSTM을 사용한 모델
- 이후, Transformer 기반의 언어 모델을 생성하여 LSTM 기반의 모델의 성능을 넘음
- 하지만, 학습과정에서 입력 받은 문단을 일정크기의 단어 시퀀스(segment)로 나눠서 학습을 진행하여, 해당 크기보다 큰 의존성을 학습할 수 없다는 문제 발생
- 또한, 단어 시퀀스는 의미에 맞게 나눈 것이 아니라 일정 크기에 맞게 나눈 문제 발생



(a) Train phase.

(b) Evaluation phase.

- 이러한 문제점을 해결하기 위해 Transformer-XL(eXtra Long) 모델 제시

1. Introduction

- 데이터셋 예시 (wikiText-103)

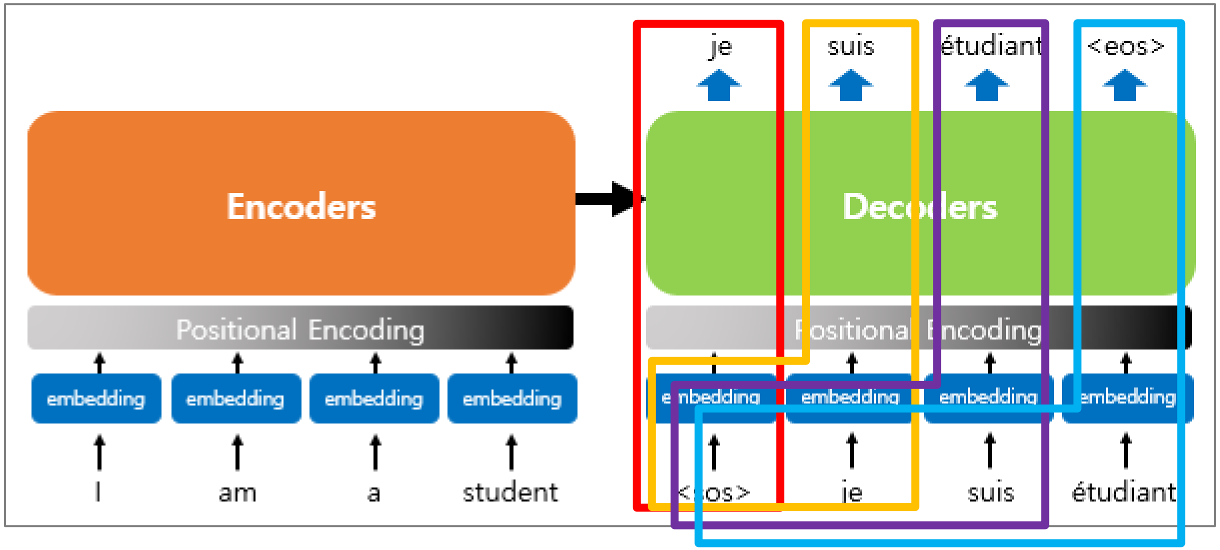
= Gold dollar =

The gold dollar or gold one @-@ dollar piece was a coin struck as a regular issue by the United States Bureau of the Mint from 1849 to 1889 . The coin had three types over its lifetime , all designed by Mint Chief Engraver James B. Longacre . The Type 1 issue had the smallest diameter of any United States coin ever minted . A gold dollar had been proposed several times in the 1830s and 1840s , but was not initially adopted . Congress was finally galvanized into action by the increased supply of bullion caused by the California gold rush , and in 1849 authorized a gold dollar . In its early years , silver coins were being hoarded or exported , and the gold dollar found a ready place in commerce . Silver again circulated after Congress in 1853 required that new coins of that metal be made lighter , and the gold dollar became a rarity in commerce even before federal coins vanished from circulation because of the economic disruption caused by the American Civil War .

2. Model

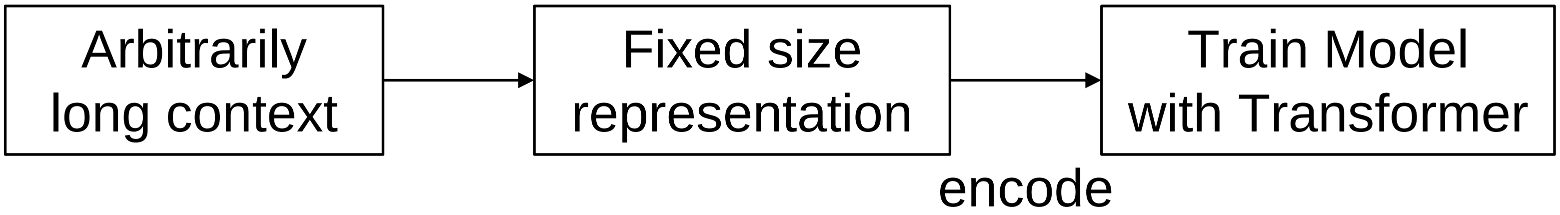
- 언어 모델링

$$x = (x_1, \dots, x_T) \quad \Rightarrow \quad P(x) = \prod_t P(x_t \mid x_{<t})$$



2.1. Vanilla Transformer Language Models [2]

- 언어 모델링에 Transformer 혹은 self-attention 적용 과정



- 해결방법

1. Long context sequence를 입력으로 받는 Model을 사용해서 모델링 진행

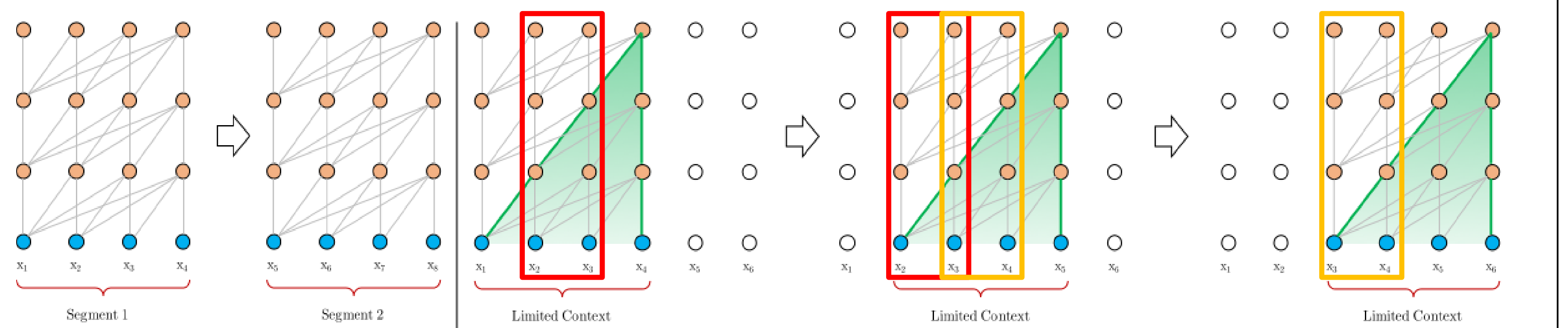
→ 많은 메모리와 계산이 사용되기 때문에 불가능

2. Long context sequence를 여러 segment로 쪼개서 Model에 입력

→ context sequence가 가지고 있던 문맥정보가 사라진다는 단점

2.1. Vanilla Transformer Language Models [2]

- 학습 및 평가 과정 그림 (Model using Transformer)



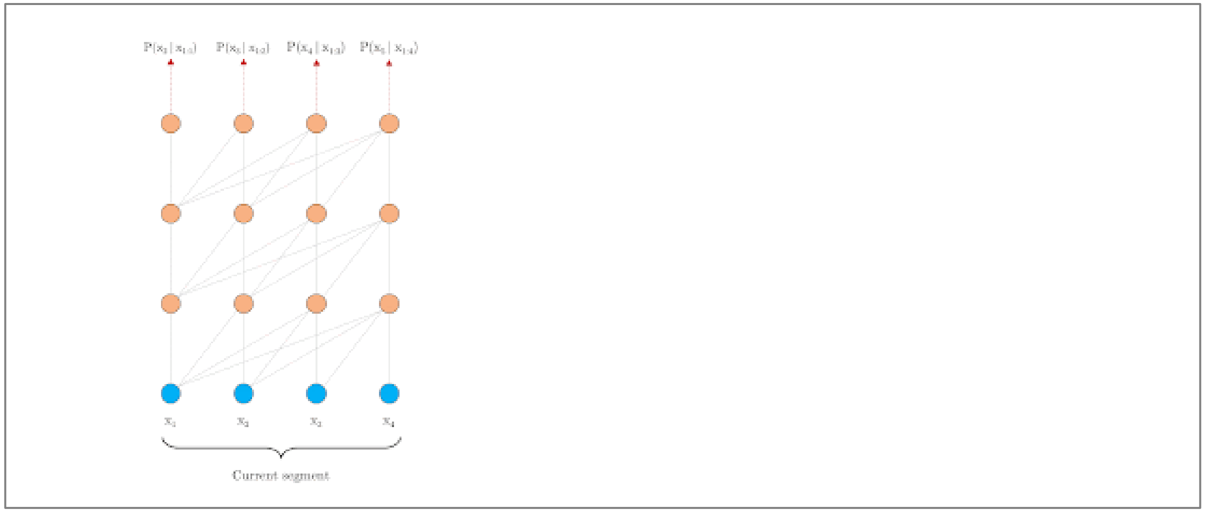
(a) Train phase.

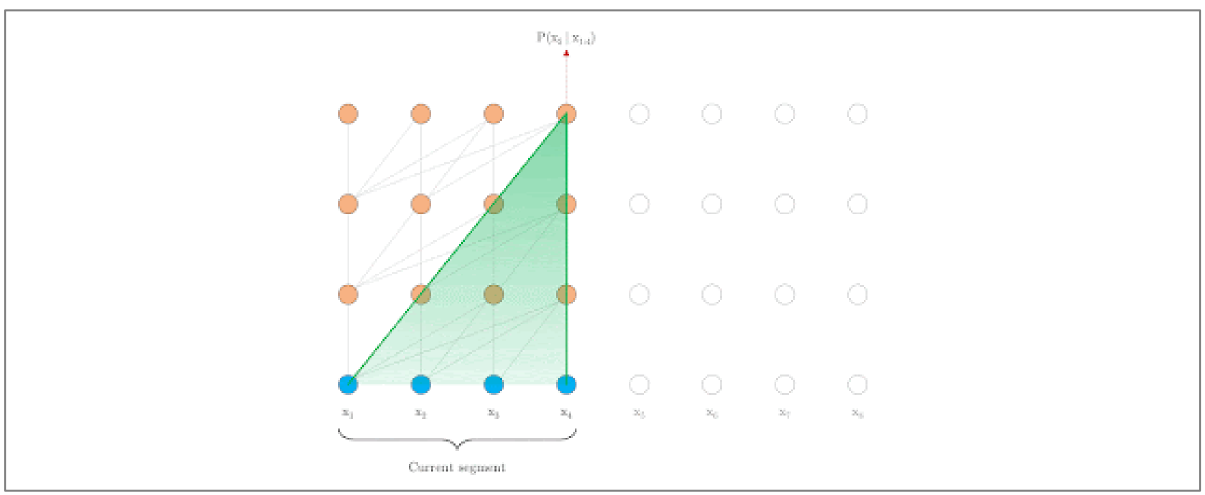
(b) Evaluation phase.

- 문제점

1. 종속성의 길이가 고정된 크기로 상한이 결정된다.
2. 패딩을 언어적 의미 관점에서 사용하기 보다는 효율적인 성능을 위해 사용
3. context fragmentation (문맥 조각화) 및 많은 계산량

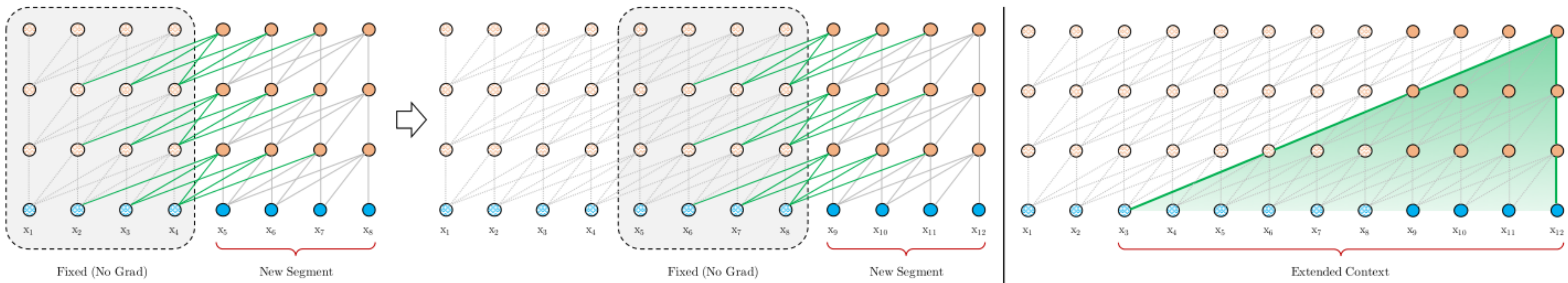
2.1. Vanilla Transformer Language Models [2]





2.2. Segment-Level Recurrence with State Reuse

- Vanilla Transformer의 문제를 해결하기 위해 recurrence 방법을 적용



(a) Training phase.

(b) Evaluation phase.

- 학습 동안, 각 세그먼트의 결과를 저장하여 다음 세그먼트에서 사용

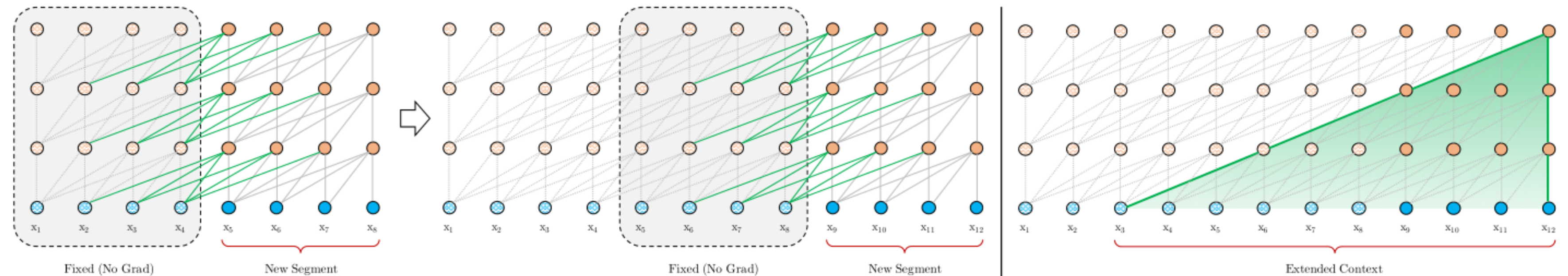
$$\widetilde{\mathbf{h}}_{\tau+1}^{n-1} = \left[\mathrm{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}\right],$$

$$\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n} = \mathbf{h}_{\tau+1}^{n-1}\mathbf{W}_{q}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{k}^{\top}, \widetilde{\mathbf{h}}_{\tau+1}^{n-1}\mathbf{W}_{v}^{\top},$$

$$\mathbf{h}_{\tau+1}^{n} = \text{Transformer-Layer}\left(\mathbf{q}_{\tau+1}^{n}, \mathbf{k}_{\tau+1}^{n}, \mathbf{v}_{\tau+1}^{n}\right).$$

2.2. Segment-Level Recurrence with State Reuse

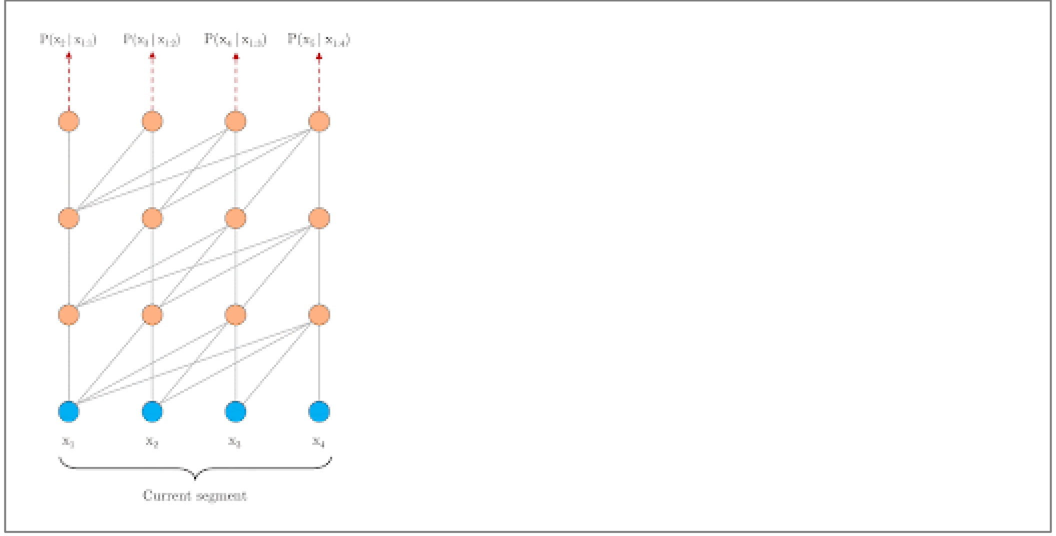
- Vanilla Transformer의 문제를 해결하기 위해 recurrence 방법을 적용

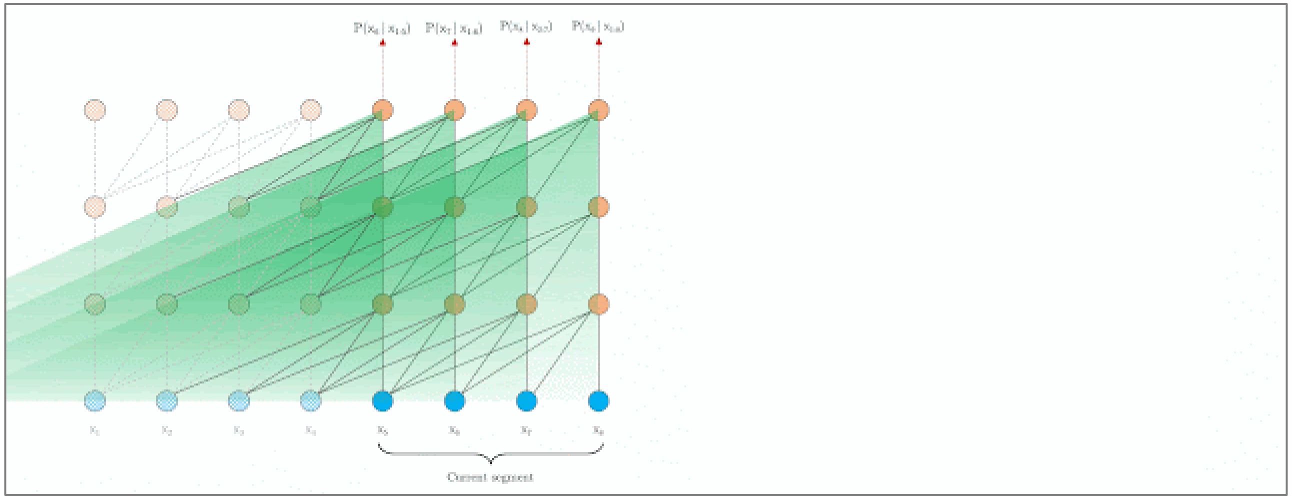


(a) Training phase. (b) Evaluation phase.

- 최대 의존관계: segment length x layer (예시: 4 x 3 = 12)
- 평가할 때 속도는, 이전 세그먼트의 state를 저장함으로써, sliding window 방식을 이용하지 않아도 되기 때문에 기존 방법보다 최대 약 1800배 빠른 연산이 가능하다.

2.2. Segment-Level Recurrence with State Reuse





2.3. Relative Positional Encodings

- 기존의 Absolute Positional Encoding을 transformer-xl에 적용

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

$$\boldsymbol{s}_{\tau} = [\boldsymbol{x}_{\tau,1}, \dots, \boldsymbol{x}_{\tau,L}]$$
$$\boldsymbol{s}_{\tau+1} = [\boldsymbol{x}_{\tau+1,1}, \dots, \boldsymbol{x}_{\tau+1,L}]$$

E: word embedding
U: Positional Encoding

⇒ 모델이 $x_{\tau,j}$와 $x_{\tau,+1\ j}$를 구분할 수 없다.

- Absolute Positional Encoding을 초기 임베딩에서 하지 않고
  각 layer의 attention score에 직접 포함하여 Relative Positional Encoding 적용

2.3. Relative Positional Encodings

- 기존 transformer의 attention 계산식

$$A_{i,j}^{abs} = (E_{x_i}^T + U_i^T)W_q^T\left(\left(E_{x_j}^T + U_j^T\right)W_k^T\right)^T$$

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

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Sinusoid encoding matrix

- Transformer-xl의 attention 계산식

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

Trainable Parameter

$$A_{i,j}^{rel} = (E_{x_i}^T W_q^T + u^T)W_{k,E}E_{x_j} + (E_{x_i}^T W_q^T + v^T)W_{k,R}R_{i-j}$$

2.3. Relative Positional Encodings

- Transformer-xl의 attention 부분 의미

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

(a): content-based addressing (컨텐츠 기반의 전달)

(b): content-dependent positional bias (컨텐츠 의존적인 위치 편향)

(c): global content bias (글로벌 컨텐츠에 대한 편향)

(d): global positional bias (글로벌 위치에 대한 편향)

2.4. Transformer-XL architecture

- Transformer-XL 구조

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

Experiment

- Word-level, character-level 언어 모델링 데이터셋으로 성능 측정
- 데이터셋: WikiText-103, enwiki8, text8, One Billion Word, Penn Treebank

- 평가지표(perplexity => '헷갈리는 정도')
- 작을수록 좋은 지표

$$PPL(W) = P(w_1, w_2, w_3, \ldots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, \ldots, w_N)}}$$

$$PPL(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, \ldots, w_N)}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i | w_1, w_2, \ldots, w_{i-1})}}$$

3.1. Main Results

- WikiText-103: 장기의존성을 가진 가장 큰 word-level의 언어모델링 벤치마크
- 28K의 기사들로부터 103M 학습 토큰들을 포함 (기사당 3.6K 토큰의 평균 길이)
- Attention 길이: train 384, test 1600

Model	#Param	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) - QRNN	151M	33.0
Rae et al. (2018) - Hebbian + Cache	-	29.9
Ours - Transformer-XL Standard	151M	**24.0**
Baevski and Auli (2018) - Adaptive Input$^{\diamond}$	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

3.1. Main Results

- enwik8: 처리되지 않은 Wikipedia text를 100M bytes 포함
- Attention 길이: train 784, test 3800
- 1) 동일한 layers 사용 2) layers를 증가시켜서 성능 향상

Model	#Param	bpc
Ha et al. (2016) - LN HyperNetworks	27M	1.34
Chung et al. (2016) - LN HM-LSTM	35M	1.32
Zilly et al. (2016) - RHN	46M	1.27
Mujika et al. (2017) - 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) - 12L Transformer	44M	1.11
Ours - 12L Transformer-XL	41M	**1.06**
Al-Rfou et al. (2018) - 64L Transformer	235M	1.06
Ours - 18L Transformer-XL	88M	1.03
Ours - 24L Transformer-XL	277M	**0.99**

3.1. Main Results

- text8: enwik8와 유사하지만 모든 문자를 소문자로 변경하고 a ~ z, ' '(space)를 제외한 문자를 제거한 100만개의 데이터셋이다.

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

3.1. Main Results

- One Billion Word: 문장이 뒤섞여 있어서 장기의존성을 보존하지 않는다.
- 단기의존성을 평가하는 데이터셋

Model	#Param	PPL
Shazeer et al. (2014) - Sparse Non-Negative	33B	52.9
Chelba et al. (2013) - RNN-1024 + 9 Gram	20B	51.3
Kuchaiev and Ginsburg (2017) - G-LSTM-2	-	36.0
Dauphin et al. (2016) - GCNN-14 bottleneck	-	31.9
Jozefowicz et al. (2016) - LSTM	1.8B	30.6
Jozefowicz et al. (2016) - LSTM + 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 and Auli (2018) - Adaptive Input$^{\diamond}$	0.46B	24.1
Baevski and Auli (2018) - Adaptive Input$^{\diamond}$	1.0B	23.7
Ours - Transformer-XL Base	0.46B	23.5
Ours - Transformer-XL Large	0.8B	**21.8**

3.1. Main Results

- Penn Treebank: 1M 학습 토큰들으로 이뤄져 있다. (작은 데이터셋)

Model	#Param	PPL
Inan et al. (2016) - Tied Variational LSTM	24M	73.2
Zilly et al. (2016) - Variational RHN	23M	65.4
Zoph and Le (2016) - NAS Cell	25M	64.0
Merity et al. (2017) - AWD-LSTM	24M	58.8
Pham et al. (2018) - Efficient NAS	24M	58.6
Liu et al. (2018) - Differentiable NAS	23M	56.1
Yang et al. (2017) - AWD-LSTM-MoS	22M	55.97
Melis et al. (2018) - Dropout tuning	24M	55.3
Ours - Transformer-XL	24M	**54.52**
Merity et al. (2017) - AWD-LSTM+Finetune†	24M	57.3
Yang et al. (2017) - MoS+Finetune†	22M	**54.44**

3.2. Ablation Study

- 데이터셋: WikiText-103
- Half Loss(segment의 절반만 cross-entropy loss 적용, attention 적은 부분 제외)
- Relative Positional Encoding: Ours, Shaw et al. (2018)
- Attention Length (training): 128

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
Transformer-XL (151M)	✓	Ours	Full	23.43	**23.09** 23.16 23.35	**640** 450 300

Reference

- [1] Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." arXiv preprint arXiv:1901.02860 (2019).
- [2] Al-Rfou, Rami, et al. "Character-level language modeling with deeper self-attention." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.
- https://baekyeongmin.github.io/paper-review/transformer-xl-review/
- https://wikidocs.net/21697
- http://mlgalaxy.blogspot.com/2019/07/transformer-xl.html
- https://wikidocs.net/31379
- https://ai.googleblog.com/2019/01/transformer-xl-unleashing-potential-of.html
- https://www.youtube.com/watch?v=lSTljZy8ag4&t=17s

감사합니다