TRANSFORMER-XL:
ATTENTIVE LANGUAGE MODELS BEYOND A FIXED-LENGTH CONTEXT

서 상우

Transformer XL

- ACL 2019
- Google과 Carnegie Mellon University

- Language Model의 SOTA
- BERT를 이긴 XLNet의 기반이 되는 모델

Results on GLUE

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
BERT	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0
XLNet	**89.8**	**93.9**	**91.8**	**83.8**	**95.6**	**89.2**	**63.6**	**91.8**

기존 연구

- 기존 NLP의 gradient vanishing and explosion을 해결하기 위한 방법론
- LSTM, Attention Mechanism, Gradient Clipping
  - *고정된 고정 길이 세그먼트에 대해 수행*
  - *고정된 컨텍스트 길이의 결과*
  - *모델은 사전 정의된 컨텍스트 길이를 초과하는 장기 의존성을 캡처할 수 없음*

fixed-length segment

- 문장이나 다른 의미 경계를 고려하지 않고 연속적인 기호 덩어리를 선택하여 생성
- 처음 몇 개의 기호를 잘 예측하는 데 필요한 상황 정보가 부족
- 비효율적인 최적화 및 성능 저하를 초래
- context fragmentation

Transformer XL (eXtra Long)

- 각 세그먼트에 대해 숨겨진 상태를 처음부터 계산하는 방식은 어렵다.
- 이전 **세그먼트에서 얻은 숨겨진 상태를 재사용(반복적인 연결을 형성하는 현재 세그먼트의 메모리 역할)**하여 gradient vanishing and explosion를 해결
- 이전 세그먼트에서 정보를 전달하면 컨텍스트 단편화 문제도 해결할 수 있는데 시간적 혼란을 야기하지 않고 상태 재사용을 가능하게 하기 위해 **절대적 인코딩(absolute positional embedding)보다는 상대적 위치 인코딩(relative positional encodings)**을 사용

auto-regressively factorized

- **x** = (x1, . . . , xt)와 같이 주어졌을 때 Joint Probability P(x)를 추정(estimate)하는 Language Model

$$P(\mathbf{x}) = \prod_t P(x_t \mid \mathbf{x}_{<t})$$

Vanilla Transformer Language Models

- 언어 모델링에 Transformer나 Self-Attention을 적용
  - *중심적인 문제는 어떻게 하면 임의적으로 긴 맥락을 고정된 크기 표현으로 효과적으로 인코딩할 수 있도록 Transformer를 학습시키는가에 대한 것*
  - *무한대의 메모리와 계산을 고려할 때, 간단한 해결책은 전체 컨텍스트 시퀀스를 처리하는 것이지만, 이는 컴퓨터 리소스의 제한(메모리 부족, 등등)으로 한계점*

Vanilla Transformer Language Models

- Al-Rfou et al. (2018)에서 소개된 **전체 말뭉치를 사람이 알 수 있는 크기의 더 짧은 세그먼트로 나누고, 이전 세그먼트에서 나온 모든 상황 정보를 무시하고 각 세그먼트 내에서만 모델을 학습**
- **학습 시에는 (a)와 같이 전체 문장의 길이를 Segment로 나누고, 각 Segment에 대해서 이동을 하면서 학습**
- 평가(evaluation)시에는 (b)와 같이 t를 1씩 증가시키며 P(w5|w1,w2,w3,w4)와 같은 방식으로 진행

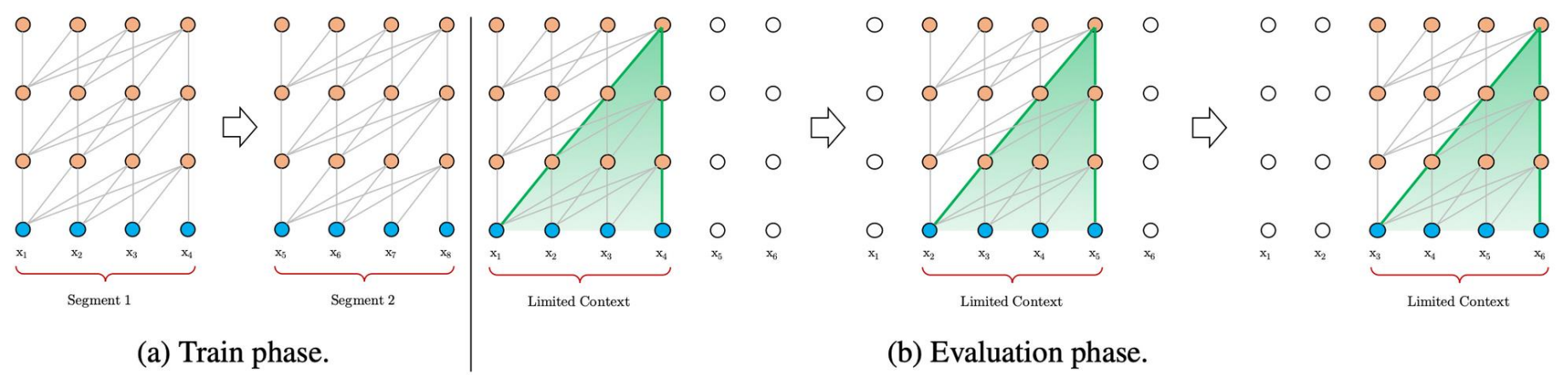


Figure 1: Illustration of the vanilla model with a segment length 4.

Vanilla Transformer Language Models

- 가능한 의존성 길이는 segment length의 상한(Upper Bounded)을 따르는데 문자 레벨의 Language Model에서는 매우 많음. 따라서 이는 RNN와 같은 vanilla 모델에서 충분히 사용할 수 없음
- 문장 또는 semantic boundaries를 구분하기 위해서 패딩(Padding)을 사용할 수 있지만, 실제로는 효율성 향상으로 인해 긴 텍스트를 고정 길이 세그먼트로 단순하게 채우는 것이 표준 관행.
- 그러나 단순히 Fixed Length로 처리하게 되면 앞에서 말한 단편화(fragmentation issue)에 빠지게 된다.

Segment-Level Recurrence with State Reuse

- Transformer-XL에서는 fixed-length context 문제를 해결하기 위해 위 그림과 같은 방식의 Segment-Level Recurrence를 사용
- 그림 (a)의 학습 시의 과정을 보면, Fixed와 New Segment 부분을 나눠서 학습을 하고, Fixed 부분은 feed-forward 방식이지만 No Grad로 학습 Weight가 업데이트 X
- 이 추가 입력은 네트워크가 이 기록의 정보를 이용할 수 있도록 하여 장기적인 의존성을 모델링하고 context fragmentation를 해결

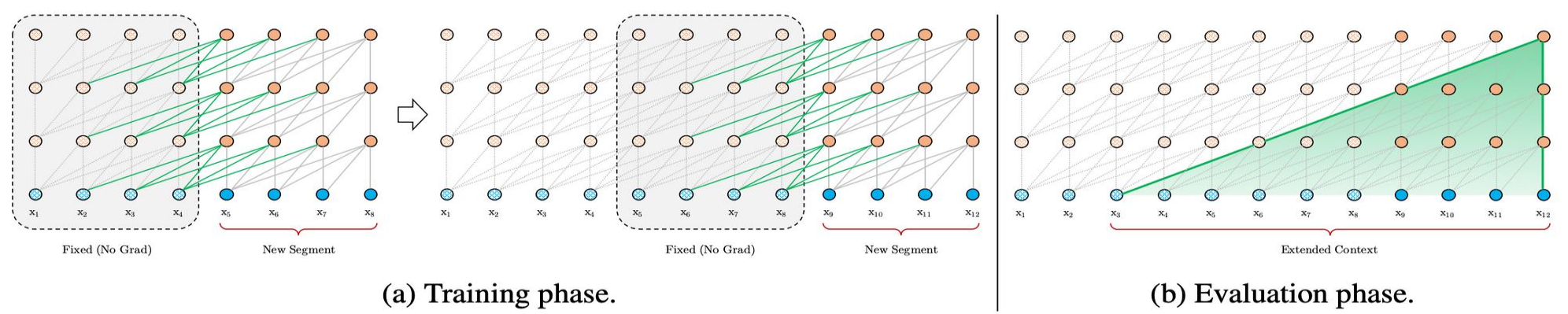
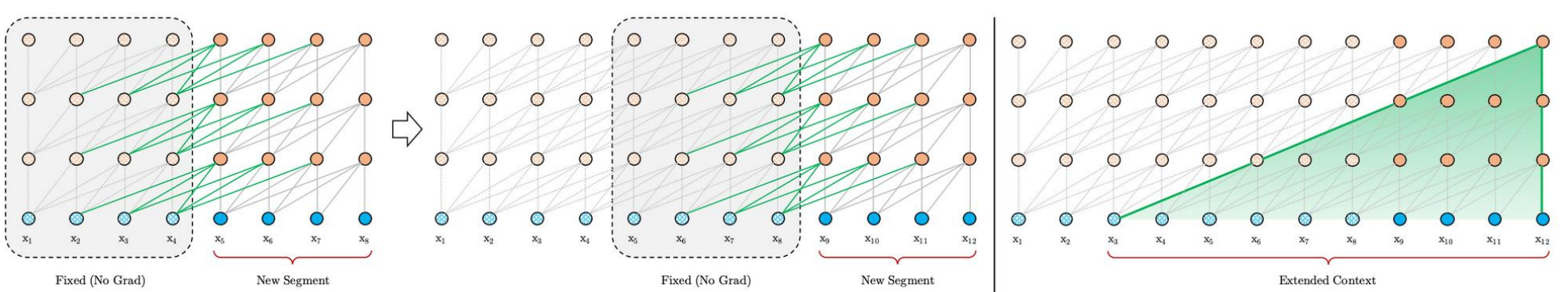


Figure 2: Illustration of the Transformer-XL model with a segment length 4.

Segment-Level Recurrence with State Reuse



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

Figure 2: Illustration of the Transformer-XL model with a segment length 4.

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

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

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

Relative Positional Encoding

- 이렇게 Segment 단위로 섹션을 분리하고(subsection) 재사용하는 것이 매우 효율적
- 하지만 State를 재사용(reuse)할 때 positional information coherent를 어떻게 유지하는지에 대한 문제
  - *Orignal Transformer의 Positional Encoding 어떤 세그먼트든 동일한 Positional Encoding을 가지며 서로 다른 세그먼트라고 구분 할 수 있는 정보를 잃게 된다.*
  - *Relative Positional Encoding*

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

where $\mathbf{E}_{\mathbf{s}_\tau} \in \mathbb{R}^{L \times d}$ is the word embedding sequence of $\mathbf{s}_\tau$, and f represents a transformation function. Notice that, both $\mathbf{E}_{\mathbf{s}_\tau}$ and $\mathbf{E}_{\mathbf{s}_{\tau+1}}$ are associated with the same positional encoding $\mathbf{U}_{1:L}$.

Relative Positional Encoding

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

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

- Positional Encoding은 모델에게 어떻게 정보를 모을 것인가에 대한 바이어스를 제공
- 초기 Embedding에 bias를 정적으로 통합하는 대신, 각 레이어의 Attention Score에 동일한 정보를 줄 수 있다.
- 더 중요한 것은 상대적인 방식으로 the temporal bias를 정의하는 것이 더욱 직관적

Relative Positional Encoding

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

Main Results

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◇	247M	20.5
Ours - Transformer-XL Large	257M	**18.3**

Table 1: Comparison with state-of-the-art results on WikiText-103. ◇ indicates contemporary work.

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**

Table 2: Comparison with state-of-the-art results on enwik8.

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**

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

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◇	0.46B	24.1
Baevski and Auli (2018) - Adaptive Input◇	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. ◇ indicates contemporary work.

Ablation Study

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

Backprop Len	Recurrence	Encoding	Loss	pplx best	pplx init	Attn Len
128	✓	Ours	Full	**26.77**	**27.02**	**500**
128	✓	Ours	Partial	28.33	28.69	460
176	✗	Ours	Full	27.98	28.43	400
172	✗	Ours	Partial	28.83	28.83	120

Table 10: Ablation study on WikiText-103 with the same GPU memory constraints.

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.