



# Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Zihang Dai, Zhilin Yang, Yiming Yang , Jaime Carbonell , Quoc V. Le , Ruslan Salakhutdinov



## XLNet 논문 읽으려다가..

### Abstract

With the capability of modeling bidirectional contexts, denoising autoencoding based pretraining like BERT achieves better performance than pretraining approaches based on autoregressive language modeling. However, relying on corrupting the input with masks, BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy. In light of these pros and cons, we propose XLNet, a generalized autoregressive pretraining method that (1) enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and (2) overcomes the limitations of BERT thanks to its autoregressive formulation. Furthermore, XLNet integrates ideas from Transformer-XL, the state-of-the-art autoregressive model, into pretraining. Empirically, under comparable experiment settings, XLNet outperforms BERT on 20 tasks, often by a large margin, including question answering, natural language inference, sentiment analysis, and document ranking.<sup>1</sup>.

# 연구 배경

**문제:** 기존 Transformer는 Long term dependency를 고려하는 능력이 떨어진다.

**LSTM:** Long term dependency에 적합한 구조(길이를 계속 늘릴 수 있다)  
Backpropagation 과정에서 gradient vanish / exploding 문제가 있음  
(Gradient clipping 해결 방법 충분하지 않음)

**Standard Transformer:** Gradient 문제에서 벗어나지만  
입력 sequence 길이가 제한되기 때문에 Long term dependency 고려하는 능력이 떨어짐

# 연구 배경

Long term dependency 해결하려면 엄청 긴 input에 대해서도 attention을 계산할 수 있어야 한다.

**Long term dependency 문제: 길이를 알지 못하는 input을  
고정된 길이(hidden state dimension  $h$ )로 어떻게 표현?**

AI Rfou: Character-Level Language Modeling with Deeper Self-Attention 에서 고안한 방법

**입력 sequence의 길이와 상관없이 일정 길이로 쪼개서 하나씩 처리하기**

Dodo bird, a flightless species sadly became extinct due to human activities. (엄청나게 긴 문장)



Dodo bird, a flightless

**Representation 1**



species sadly became extinct

**Representation 2**



due to human activities.

**Representation 3**

# 연구 배경

Dodo bird, a flightless species sadly became extinct due to human activities. (엄청나게 긴 문장)



Dodo bird, a flightless

**Representation 1**



species sadly became extinct

**Representation 2**



due to human activities.

**Representation 3**

## 문제점

1. Representation을 계산할 때 이전 내용을 고려하지 못한다.

2. Representation간의 정보를 공유하지 못한다

**= Context Fragmentation**

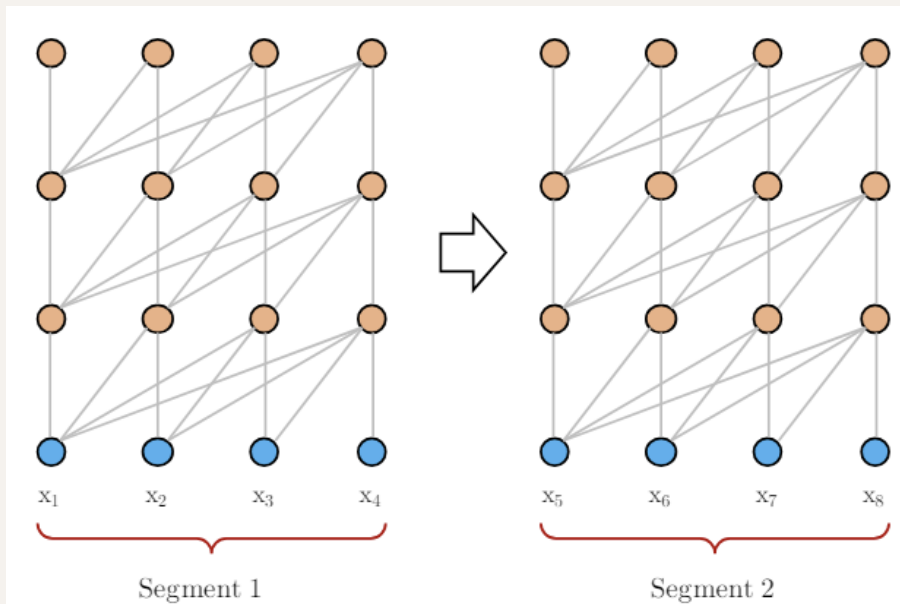
3. 내용을 고려하지 않고 길이를 기준으로 자르기 때문에 초반 단어들을 예측할 때 어려움을 겪음

→ 이런 문제 근본적인 원인: 고정된 길이의 input을 사용한다

# 연구 배경

고정된 길이의 input을 사용한다

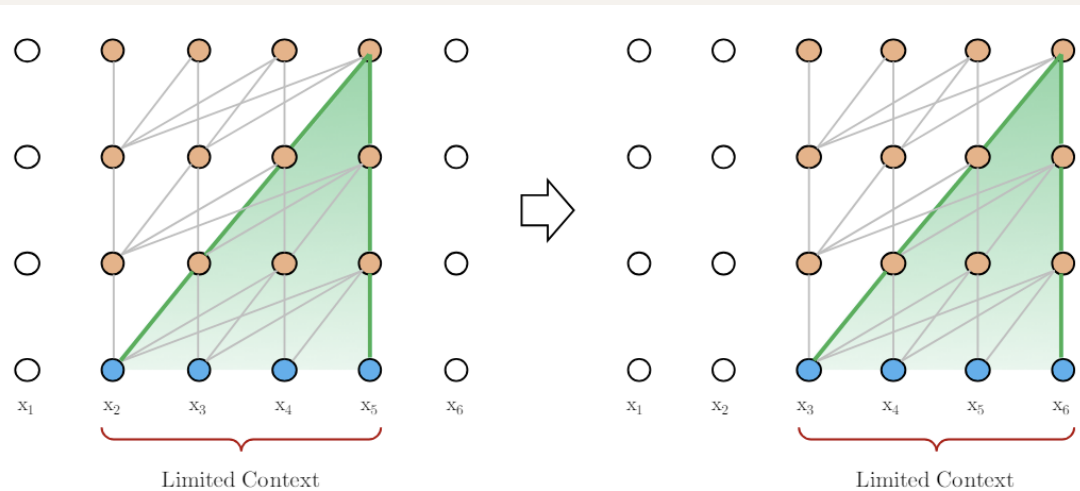
= Dependency 고려하는 범위가 고정된 길이로 제한된다. (Transformer의 문제)



(a) Train phase.

# 연구 배경

## 문제점 (다음 단어 예측하는 과정)



(b) Evaluation phase.

예측을 할 때 Input을 나눈 sequence length로 동일하게 평가 input을 나눈다

하나의 sequence마다 하나의 예측을 한다.

= 하나의 sequence를 처리할 때마다 계산을 처음부터 다 해야함

→ 계산 오래 걸린다

# 연구 배경

하나의 segment의 길이가 2라면...

X1, X2, X3, X4, X5, X6, X7, X8



Output 1



# 연구 배경

하나의 segment의 길이가 2라면...

X1, X2, X3, X4, X5, X6, X7, X8



Output 2

# 연구 배경

하나의 segment의 길이가 2라면...

X1, X2, X3, X4, X5, X6, X7, X8



Output 3

# 연구 배경

하나의 segment의 길이가 2라면...

X1, X2, X3, X4, X5, X6, X7, X8



Output 4

X2, X3, X4이 두번씩 계산에 사용되었다 → 계산이 expensive한 이유

# Transformer-XL

Input을 일정한 길이의 Segment로 나누는 것은 똑같다. 그럼 Context fragmentation 문제 해결 어떻게?

## 해결 방법의 첫 부분

**1. Recurrence(재발생) 개념 사용:** 이전 Segment의 hidden state 정보를 fix (no grad) + cache화해서 다음 segment representation 계산할 때 사용한다.

# Transformer-XL

Input을 일정한 길이의 Segment로 나누는 것은 똑같다

**Recurrence 개념 사용:** 이전 Segment 정보를 fix (no grad) + cache화를 해서 다음 segment representation 계산할 때 사용한다.

길이가 L인 두 이어진 segment (r, r+1):  $[x_{\tau,1}, \dots, x_{\tau,L}]$   $[x_{\tau+1,1}, \dots, x_{\tau+1,L}]$

r번째 segment의 n-th layer의 hidden state  $\mathbf{h}_{\tau}^n \in \mathbb{R}^{L \times d}$


$$\tilde{\mathbf{h}}_{\tau+1}^{n-1} = [\text{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}],$$

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

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

# Transformer-XL

미분 못하게 만든(Stop Gradient) 이전 segment n-1th layer hidden state와  
현재 segment의 n-1th layer hidden state를 연결

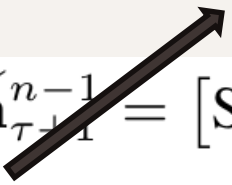

$$\tilde{\mathbf{h}}_{\tau+1}^{n-1} = [\text{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}] ,$$

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

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

# Transformer-XL

Q,K,V를 임의의 matrix로 정의하지 않고 지금까지의 context를 바탕으로 정의한다


$$\tilde{\mathbf{h}}_{\tau+1}^{n-1} = [\text{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}] ,$$

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

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

# Transformer-XL

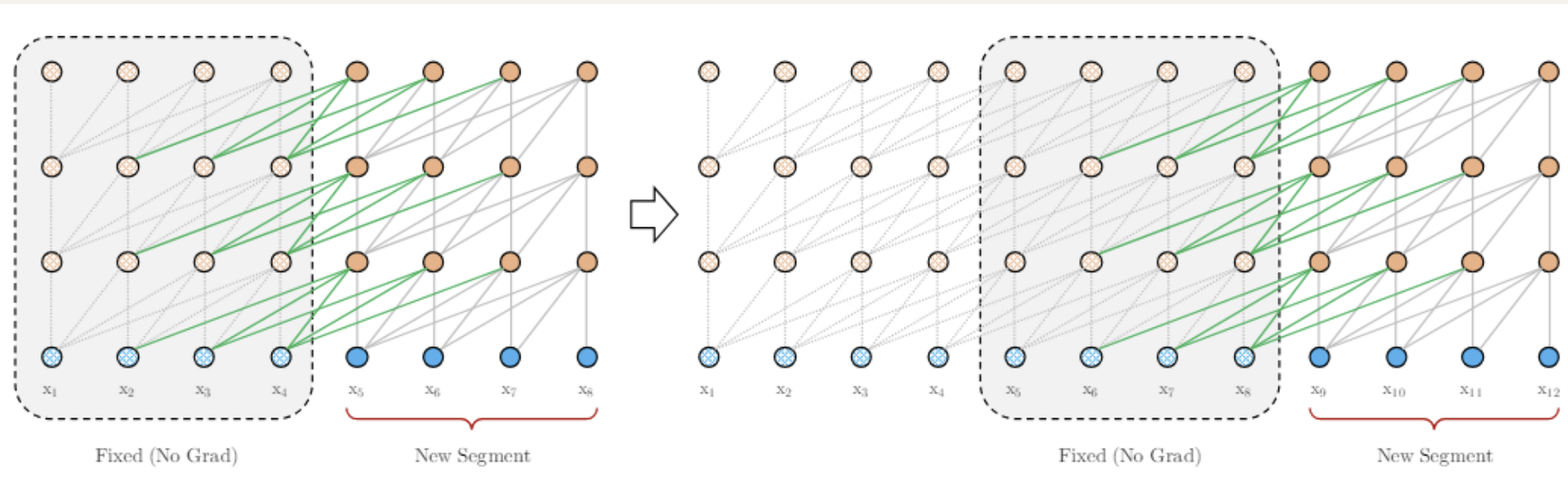
$$\begin{aligned}\tilde{\mathbf{h}}_{\tau+1}^{n-1} &= [\text{SG}(\mathbf{h}_{\tau}^{n-1}) \circ \mathbf{h}_{\tau+1}^{n-1}] , \\ \mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n &= \mathbf{h}_{\tau+1}^{n-1} \mathbf{W}_q^\top, \tilde{\mathbf{h}}_{\tau+1}^{n-1} \mathbf{W}_k^\top, \tilde{\mathbf{h}}_{\tau+1}^{n-1} \mathbf{W}_v^\top , \\ \mathbf{h}_{\tau+1}^n &= \text{Transformer-Layer}(\mathbf{q}_{\tau+1}^n, \mathbf{k}_{\tau+1}^n, \mathbf{v}_{\tau+1}^n) .\end{aligned}$$



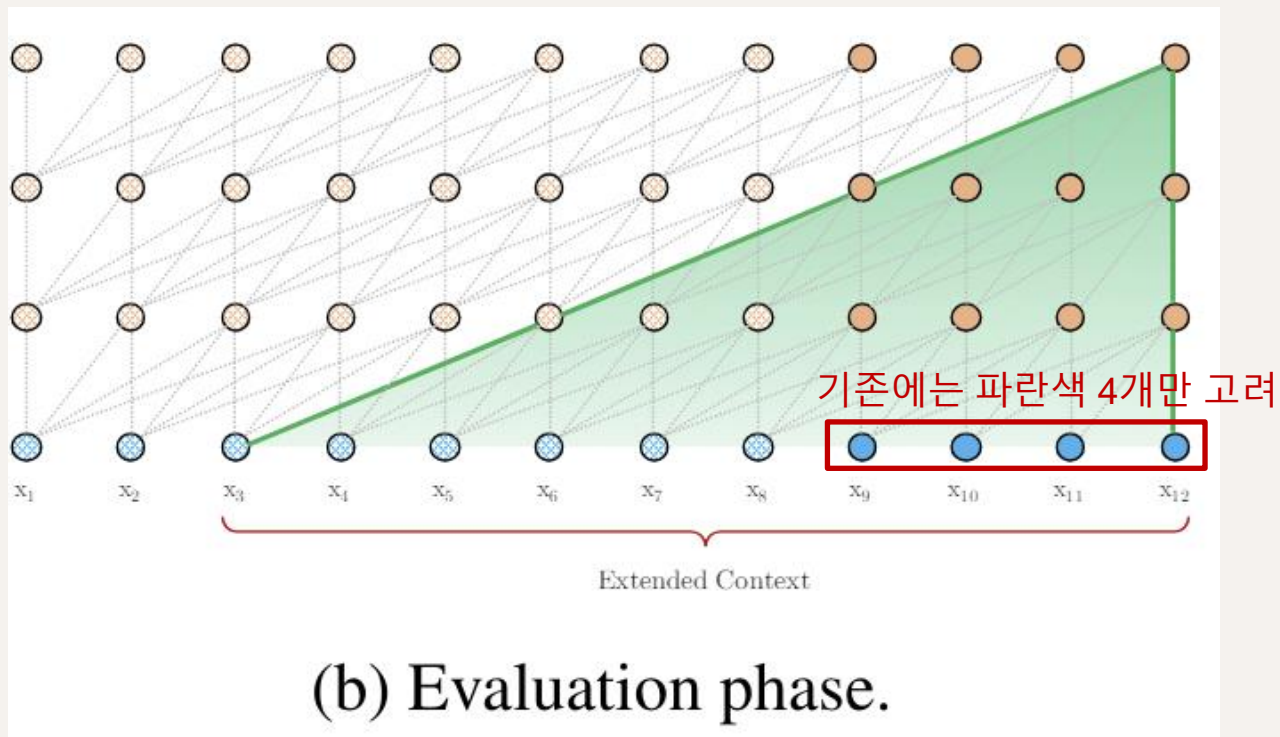
새로운 Q,K,V로 Transformer Decoder에 대입



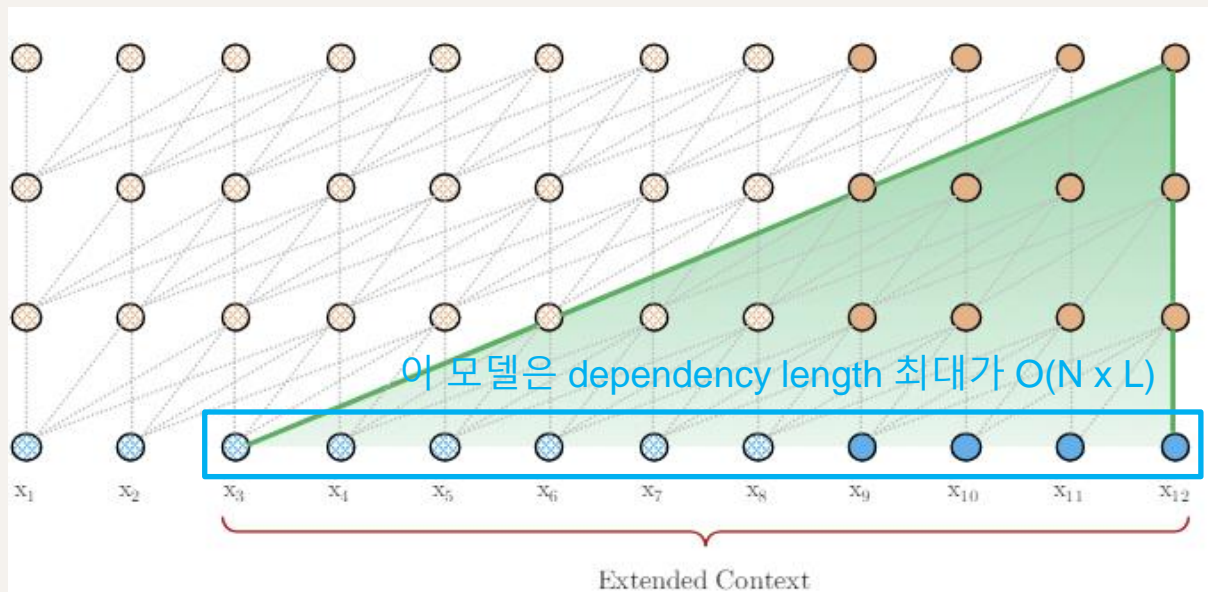
# Transformer-XL (Train)



# Transformer-XL (Evaluation)



# Transformer-XL (Evaluation)



$N$  = hidden layer 수  
 $L$  = segment length

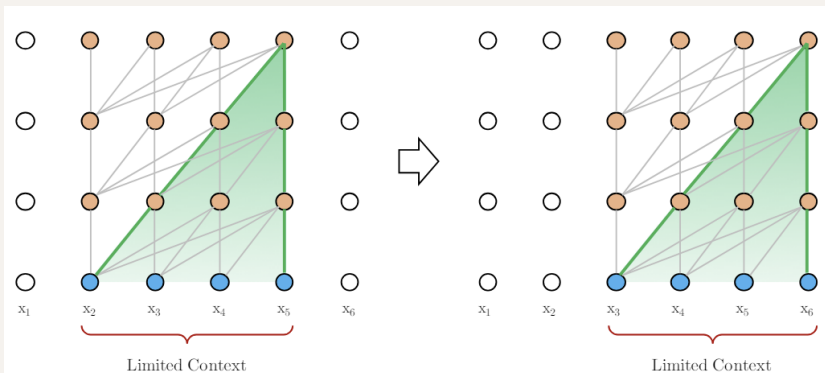
(b) Evaluation phase.

# Transformer-XL (Evaluation)

**Recurrence 개념 사용:** 이전 Segment 정보를 fix (no grad) + cache화를 해서 다음 segment representation 계산할 때 사용한다.

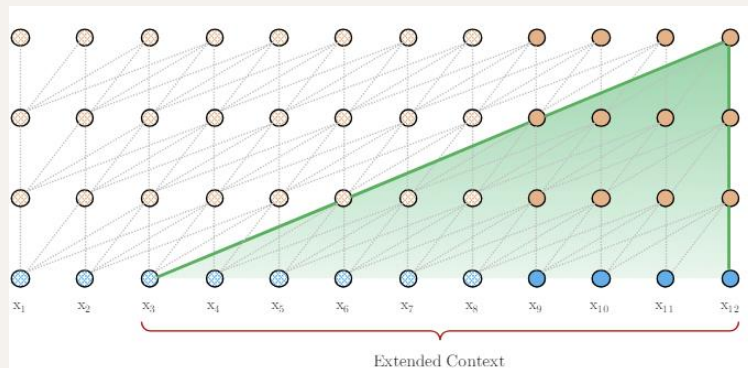
또 다른 장점: 이전 hidden state 계산 결과를 가져오기 때문에 속도도 빠르다

AI Rfou: 이전 hidden state들이 또 계산됨



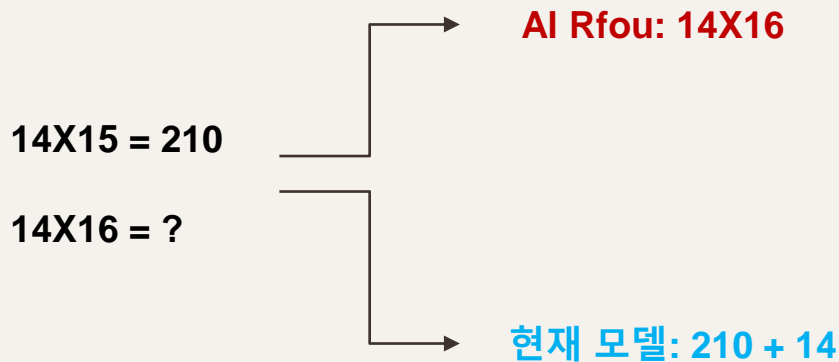
(b) Evaluation phase.

현재 모델



(b) Evaluation phase.

# Transformer-XL (Evaluation)



결론적으로 Transformer-XL이 기존 Transformer보다 1800+배 더 빠르다

# Transformer-XL

## 해결 방법의 두 번째 부분

### 2. 새로운 Positional Encoding 사용

왜 사용해야 돼?

Recurrence만 고려했을 때의 식

$$\begin{aligned} \mathbf{h}_{\tau+1} &= f(\mathbf{h}_{\tau}, \mathbf{E}_{s_{\tau+1}} + \mathbf{U}_{1:L}) \\ \mathbf{h}_{\tau} &= f(\mathbf{h}_{\tau-1}, \mathbf{E}_{s_{\tau}} + \mathbf{U}_{1:L}), \end{aligned}$$

여기서 문제 발생: 두 segment가 동일한 positional encoding 값 지닌다

$f$  = transformation function

$E$  = word embedding

$U$  = absolute positional encoding

# Transformer-XL

이 논문에서의 Vanilla Transformer는 **AI Rfou**의 Transformer 모델을 의미

이 Transformer는 **Absolute positional encoding**을 사용

→ 두 segment을 둘 다 고려하는 상황에서 이런 encoding을 사용한다면

<b>Corpus =</b>	준혁이가 예진에게 고백했다.			+	그녀는 처참하게 거절했다		
<b>Absolute positional encoding =</b>	0	1	2		0	1	2

**두 문장 순서 구분하지 못한다**

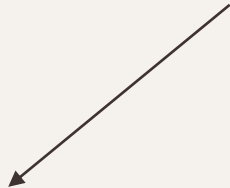
**Query(i)가 “그녀는”일 때 Attention score를 계산하기 위해 key(j) ( $j \leq i$ )들과의 연산이 필요한데 이전 segment position이 0,1,2이기 때문에 이들과 연산을 하지 못함**

# Transformer-XL

**Positional Encoding**은 입력 sequence를 어떻게 사용할지, 어떤 부분을 볼 지 알려주는 힌트

LSTM과 다르게 **Attention**은 모든 정보를 한번에 **병렬로** 연산하기 때문에 **Position** 정보 매우 중요

입력 sequence의 길이에 상관없이 보편적으로 position 정보를 encoding하기 위해서는 **상대적 위치**가 적합



Query와 Key의 index 차이 사용



# Transformer-XL

## 해결 방법의 두 번째 부분

### 2. 새로운 Positional Encoding 사용

$$\mathbf{R} \in \mathbb{R}^{L_{\max} \times d} \quad L_{\max} = \text{segment의 최대 길이}$$

i번째 행은 i-th token과 나머지 token들과의 상대적 거리 정보를 담는다

→ 이 정보를 attention score 계산 과정에 주입하여 segment의 순서를 알게 해준다

→ Sinusoid(Vaswani 2017) encoding matrix

# Transformer-XL

i=query

W=weight matrix

j=key

U = 절대적 위치 정보

E=word embedding

## 해결 방법의 두 번째 부분

### 2. 새로운 Positional Encoding 사용

$$\mathbf{U}_j \rightarrow \mathbf{R}_{i-j}$$

= 문장을 읽어 attention score를 계산할 때  
어떤 정보를 봐야하는지 알려주는 것은  
Relative 위치 정보에만 들어있다는 내포

Key의 Absolute 위치 정보를 Query에 대한 Key의  
Relative 위치 정보로 바꾼다

### 기존 Attention score 분해

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

### Transformer-XL Attention score 분해

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

# Transformer-XL

## 해결 방법의 두 번째 부분

### 2. 새로운 Positional Encoding 사용

$$\mathbf{U}_i^\top \mathbf{W}_q^\top \rightarrow \mathbf{u}^\top \mathbf{v}^\top$$

$\mathbf{u}, \mathbf{v}$  = 학습이 가능한 parameter

Query의 절대적 위치 정보는 필요없음

Query의 위치와 상관없이 Query는 모든 정보를 중요하게 생각해야 한다는 의미를 내포

### 기존 Attention score 분해

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

### Transformer-XL Attention score 분해

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

# Transformer-XL

## 해결 방법의 두 번째 부분

### 2. 새로운 Positional Encoding 사용

$$\mathbf{W}_k \longrightarrow \begin{matrix} \text{위치} \\ \mathbf{W}_{k,R} \end{matrix} \quad \begin{matrix} \text{의미} \\ \mathbf{W}_{k,E} \end{matrix}$$

Key의 Weight matrix를  
의미 중심적, 위치 중심적 matrix로 나눈다

### 기존 Attention score 분해

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

### Transformer-XL Attention score 분해

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

# Transformer-XL

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

(a): Content Based Addressing

기본적인 어텐션 메커니즘

(b): Content dependent positional bias

쿼리 벡터와 키 벡터 간의 위치 차이에 따라 어텐션 스코어에 추가적인 bias.

위치 정보가 쿼리와 키의 내용에 따라 동적으로 조정 → 정교한 위치 정보를 반영

(c): Global Content Bias

특정 키 벡터가 전역적으로 중요한지를 반영

(d): Global Positional Bias Encoding

쿼리-키 쌍의 위치 차이 정보를 제공

모델이 위치 정보의 중요성을 반영

# Transformer-XL

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

# Transformer-XL (결과)

긴 시퀀스와 짧은 시퀀스 모두에서 문자 수준 및 단어 수준의 Task에서 SoTA

1. Transformer-XL은 RNN 대비 약 80% 더 긴 의존성을 학습
2. Transformer-XL은 언어 모델링 작업 평가 중에 바닐라 Transformer 대비 최대 1,800배 이상 빠름
3. Transformer-XL은 긴 시퀀스에서의 perplexity(샘플 예측 정확도)에서 더 나은 성능  
Long term dependency, Context fragment 문제 다 해결해서 짧은 시퀀스에서도 더 나은 성능

# Transformer-XL (결과)

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	<b>27.02</b>	<b>26.77</b>	<b>500</b>
-	✓	Shaw et al. (2018)	Full	27.94	27.94	256
-	✓	Ours	Half	28.69	28.33	460
-	✗ 상대적 위치	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) <sup>†</sup>	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
Transformer-XL (151M)	✓	절대적 위치			<b>23.09</b>	<b>640</b>
		Ours	Full	23.43	23.16	450
					23.35	300



# Transformer-XL (결과) 이 모델의 Recurrence, Encoding이 Context Length 범위 증가에 기여

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) <sup>†</sup>	✗	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

# Transformer-XL (결과) 이 모델의 Recurrence, Encoding이 Context Length 범위 증가에 기여

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	<b>27.02</b>	<b>26.77</b>	<b>500</b>
-	✓	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) <sup>†</sup>	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
					<b>23.09</b>	<b>640</b>
Transformer-XL (151M)	✓	Ours	Full	23.43	23.16	450
					23.35	300

# Transformer-XL (결과) Parameter 수 증가하면 Attention Length 더더욱 증가

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	<b>27.02</b>	<b>26.77</b>	<b>500</b>
-	✓	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) <sup>†</sup>	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
					<b>23.09</b>	<b>640</b>
Transformer-XL (151M)	✓	Ours	Full	23.43	23.16	450
					23.35	300

Transformer-XL (결과)

Attention Context Length 증가할수록 Perplexity도 감소

Remark	Recurrence	Encoding	Loss	PPL init	PPL best	Attn Len
Transformer-XL (128M)	✓	Ours	Full	<b>27.02</b>	<b>26.77</b>	<b>500</b>
-	✓	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) <sup>†</sup>	✗	Al-Rfou et al. (2018)	Half	31.16	31.16	120
Transformer-XL (151M)	✓	Ours	Full	23.43	<b>23.09</b>	<b>640</b>
					23.16	450
					23.35	300

# Transformer-XL (결과)

이전 hidden state를 이용해서 계산하려면 memory에 저장해야 하는데 memory 관련 문제는 없나?

Backprop Len	Recurrence	Encoding	Loss	pplx best	pplx init	Attn Len
128	✓	Ours	Full	<b>26.77</b>	<b>27.02</b>	<b>500</b>
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.

제한이 있어도 잘 작동한다

# Transformer-XL (결과)

## RECL: 고려하는 Context의 길이를 얼마까지 늘릴 수 있는지 Test

최고의 짧은 문맥 결과와 비교한 긴 문맥 성능

### ECL

상위  $r$ 개의 어려운 예제에 대한 비교

문맥 범위를 확장함으로써 얻는 이득이  
임계값보다 더 커지는 가장 긴 길이

그러나 ECL은 이미 더 짧은 문맥을 사용하여  
낮은 PPL를 달성하는 모델에서 문맥 길이를  
늘려 개선을 얻는 것이 더 어려운 점을 무시

→ 공정한 기준이 아니다

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

# Transformer-XL (결과)

기존 Transformer 모델들은 계산 복잡도가  $O(L^2)$

Transformer-XL은 Attention score를 계산하는 방법을 바꾸어서 계산 복잡도를  $O(L^2)$ 에서  $O(L)$ 로 바꾼다

As we discussed in section 3.3, the naive way of computing the  $\mathbf{W}_{k,R}\mathbf{R}_{i-j}$  for all pairs  $(i, j)$  is subject to a quadratic cost. Here, we present a simple method with only a linear cost. Firstly, notice that the relative distance  $i - j$  can only be integer from 0 to  $M + L - 1$ , where  $M$  and  $L$  are the memory length and segment length respectively. Hence, the rows of the matrix

$$\mathbf{Q} := \begin{bmatrix} \mathbf{R}_{M+L-1}^\top \\ \mathbf{R}_{M+L-2}^\top \\ \vdots \\ \mathbf{R}_1^\top \\ \mathbf{R}_0^\top \end{bmatrix} \quad \mathbf{W}_{k,R}^\top = \begin{bmatrix} [\mathbf{W}_{k,R}\mathbf{R}_{M+L-1}]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_{M+L-2}]^\top \\ \vdots \\ [\mathbf{W}_{k,R}\mathbf{R}_1]^\top \\ [\mathbf{W}_{k,R}\mathbf{R}_0]^\top \end{bmatrix} \in \mathbb{R}^{(M+L) \times d}$$

consist of all possible vector outputs of  $\mathbf{W}_{k,R}\mathbf{R}_{i-j}$  for any  $(i, j)$ . Note that we have defined  $\mathbf{Q}$  in a reversed order, i.e.,  $\mathbf{Q}_k = \mathbf{W}_{k,R}\mathbf{R}_{M+L-1-k}$ , to make further discussion easier.

Next, we collect the term (b) for all possible  $i, j$  into the following  $L \times (M + L)$  matrix,

$$\mathbf{B} = \begin{bmatrix} q_0^\top \mathbf{W}_{k,R}\mathbf{R}_M & \cdots & q_0^\top \mathbf{W}_{k,R}\mathbf{R}_0 & 0 & \cdots & 0 \\ q_1^\top \mathbf{W}_{k,R}\mathbf{R}_{M+1} & \cdots & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_1 & q_1^\top \mathbf{W}_{k,R}\mathbf{R}_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{M+L-1} & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_{L-1} & \cdots & q_{L-1}^\top \mathbf{W}_{k,R}\mathbf{R}_0 \end{bmatrix}$$

$$= \begin{bmatrix} q_0^\top \mathbf{Q}_{L-1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ q_1^\top \mathbf{Q}_{L-2} & \cdots & q_1^\top \mathbf{Q}_{M+L-2} & q_1^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix}$$

Then, we further define

$$\tilde{\mathbf{B}} = \mathbf{q}\mathbf{Q}^\top = \begin{bmatrix} q_0^\top \mathbf{Q}_0 & \cdots & q_0^\top \mathbf{Q}_M & q_0^\top \mathbf{Q}_{M+1} & \cdots & q_0^\top \mathbf{Q}_{M+L-1} \\ q_1^\top \mathbf{Q}_0 & \cdots & q_1^\top \mathbf{Q}_M & q_1^\top \mathbf{Q}_{M+1} & \cdots & q_1^\top \mathbf{Q}_{M+L-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ q_{L-1}^\top \mathbf{Q}_0 & \cdots & q_{L-1}^\top \mathbf{Q}_M & q_{L-1}^\top \mathbf{Q}_{M+1} & \cdots & q_{L-1}^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

Now, it is easy to see an immediate relationship between  $\mathbf{B}$  and  $\tilde{\mathbf{B}}$ , where the  $i$ -th row of  $\mathbf{B}$  is simply a left-shifted version of  $i$ -th row of  $\tilde{\mathbf{B}}$ . Hence, the computation of  $\mathbf{B}$  only requires a matrix multiplication  $\mathbf{q}\mathbf{Q}^\top$  to compute  $\tilde{\mathbf{B}}$  and then a set of left-shifts.

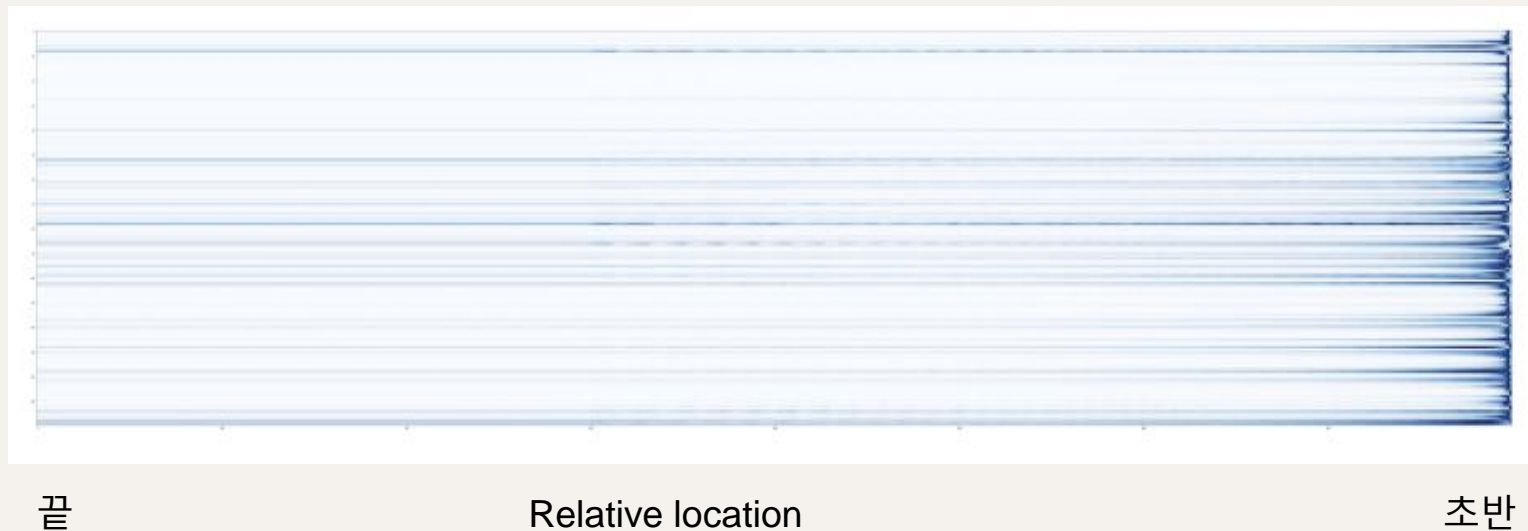
Similarly, we can collect all term (d) for all possible  $i, j$  into another  $L \times (M + L)$  matrix  $\mathbf{D}$ ,

$$\mathbf{D} = \begin{bmatrix} v^\top \mathbf{Q}_{L-1} & \cdots & v^\top \mathbf{Q}_{M+L-1} & 0 & \cdots & 0 \\ v^\top \mathbf{Q}_{L-2} & \cdots & v^\top \mathbf{Q}_{M+L-2} & v^\top \mathbf{Q}_{M+L-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ v^\top \mathbf{Q}_0 & \cdots & v^\top \mathbf{Q}_M & v^\top \mathbf{Q}_{M+1} & \cdots & v^\top \mathbf{Q}_{M+L-1} \end{bmatrix}.$$

# Transformer-XL (결과)

Model이 입력을 처리할 때 어떤 부분을 많이 참고하는지에 대한 시각화

Attention head



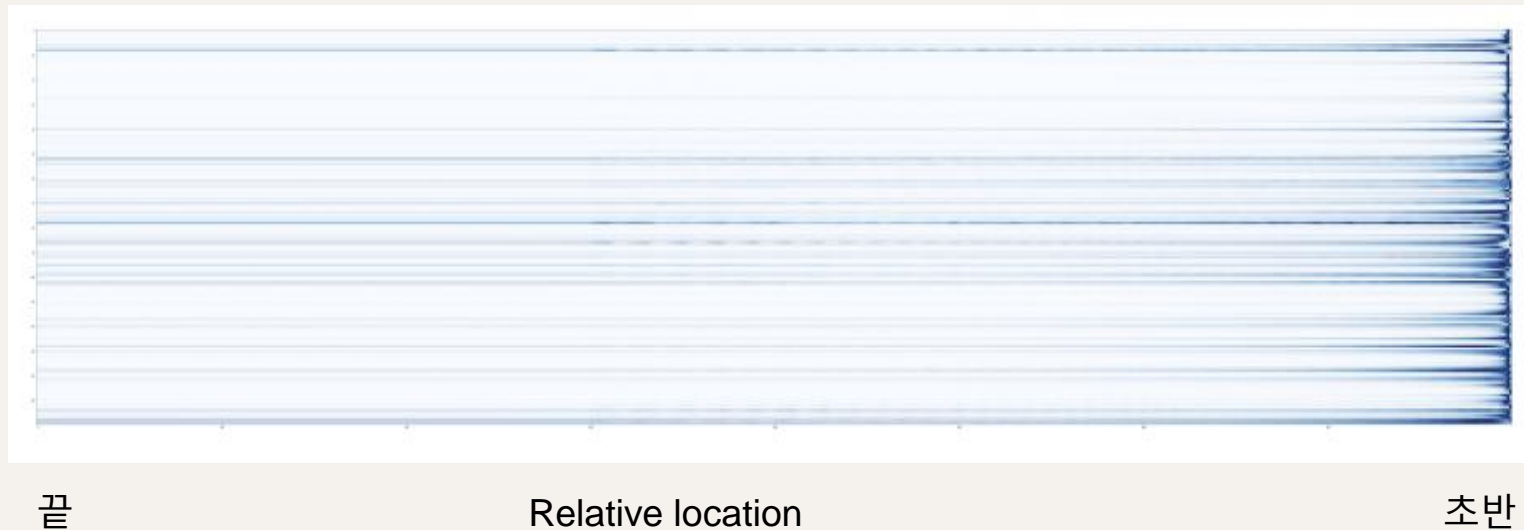
초반 부분을 많이 참고한다



# Transformer-XL (결과)

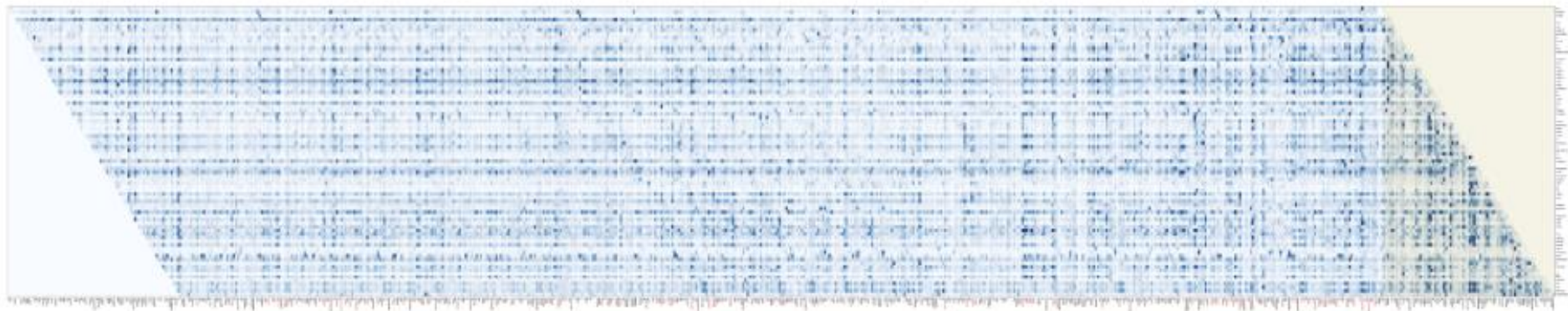
Model이 입력을 처리할 때 어떤 부분을 많이 참고하는지에 대한 시각화

Attention head



그럼 저 넓은 범위를 고려하는 head는 무엇

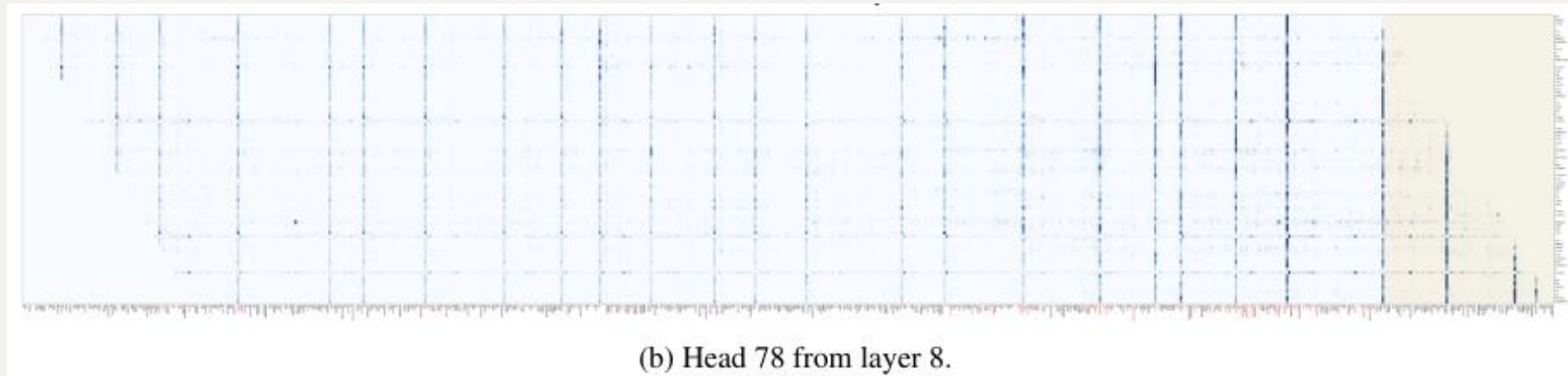
# Transformer-XL (결과)



(a) Head 8 from layer 1.

초반 layer이기에 모든 부분을 중요하게 보는 것을 유추할 수 있다

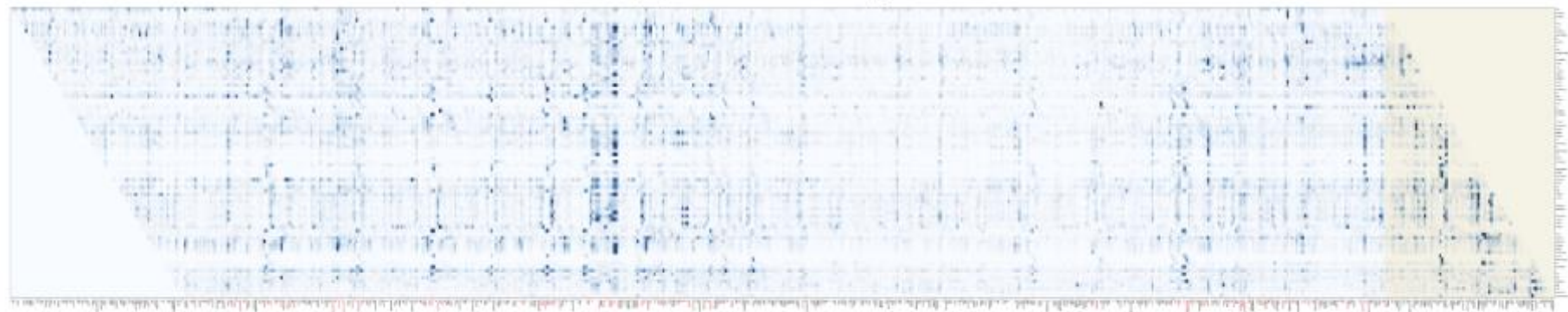
# Transformer-XL (결과)



특정 부분에만 쏠려있다. → 정보가 많아질수록 Neural Network는 특정 부분만 보게 된다

== 각각의 head는 학습될수록 특정 관점에 집중하여 정보를 본다

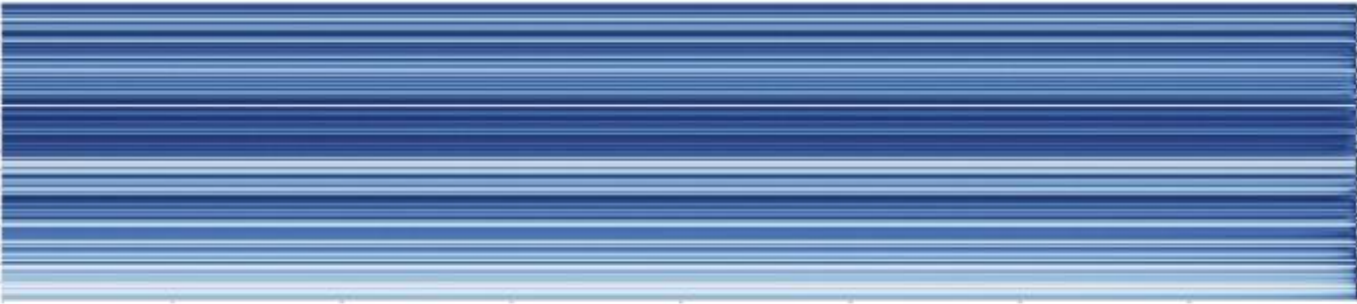
# Transformer-XL (결과)




(c) Head 158 from layer 16.

아까의 head와는 주의깊게 보는 지점들이 다르지만 특정 position에 몰려있는 것을 또 관찰할 수 있다.

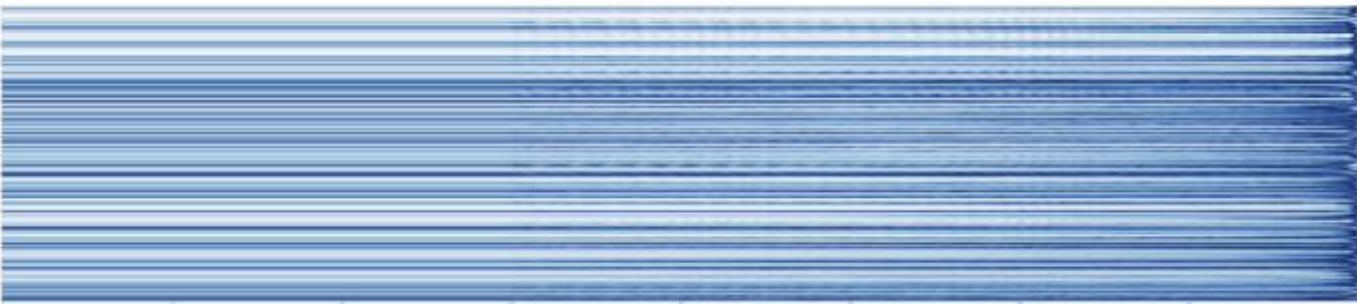
Multihead-attention 시각적으로 보여주는 그림



(a) Term (a).



(b) Term (b).



(c) Term (d).

(a): Content Based Addressing  
기본적인 어텐션 메커니즘  
 $Q, k$ 에 따른 값이기에 일정

(b): Content dependent  
positional bias  
쿼리 벡터와 키 벡터 간의 위치  
차이에 따라 어텐션 스코어에  
추가적인 bias.

가까운 애들을 주로 본다

(c): Global Content Bias  
특정 키 벡터가 전역적으로  
중요한지를 반영

(d): Global Positional Bias  
Encoding  
모델이 위치 정보의 중요성을  
반영  
가까운 애들만 한정하지 않고  
넓게 본다