Natural Language Processing

XLNet: Generalized Autoregressive Pretraining for Language Understanding

Sogang University
Dept. of Artificial Intelligence

Presented by
조유빈

Introduction of XLNet

- Autoregressive (AR) 모델의 문제점
  - Bidirectional context information을 고려하지 못함
- Denoising autoencoding 모델인 BERT의 문제점
  - 마스크 된 단어 간의 종속성을 무시
  - Pretraining 중에만 입력 값에 [MASK] noise를 사용하고, finetuning 시에는 [MASK] noise를 사용하지 않아 pretrain-finetune discrepancy 발생
- Proposed method (XLNet)
  - Permutation Language Modeling
    - 단어들의 모든 순열 조합의 가능성을 고려함으로써 bidirectional contexts 학습 가능
  - Generalized AR language model
    - Data corruption (i.e. [MASK] noise) 사용 안 함
      - Pretrain-finetune discrepancy 발생 안 함

Permutation Language Modeling (PLM)

- 길이가 T인 sequence $\mathbf{x}$일 때, $T!$개의 different orders에 대해 autoregressive 수행
  - 모델은 모든 위치에서 정보를 모으는 방법을 학습하게 됨
    - Bidirectional context information 추출 가능
- Proposed permutation language modeling objective:

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim Z_T}\left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right].$$

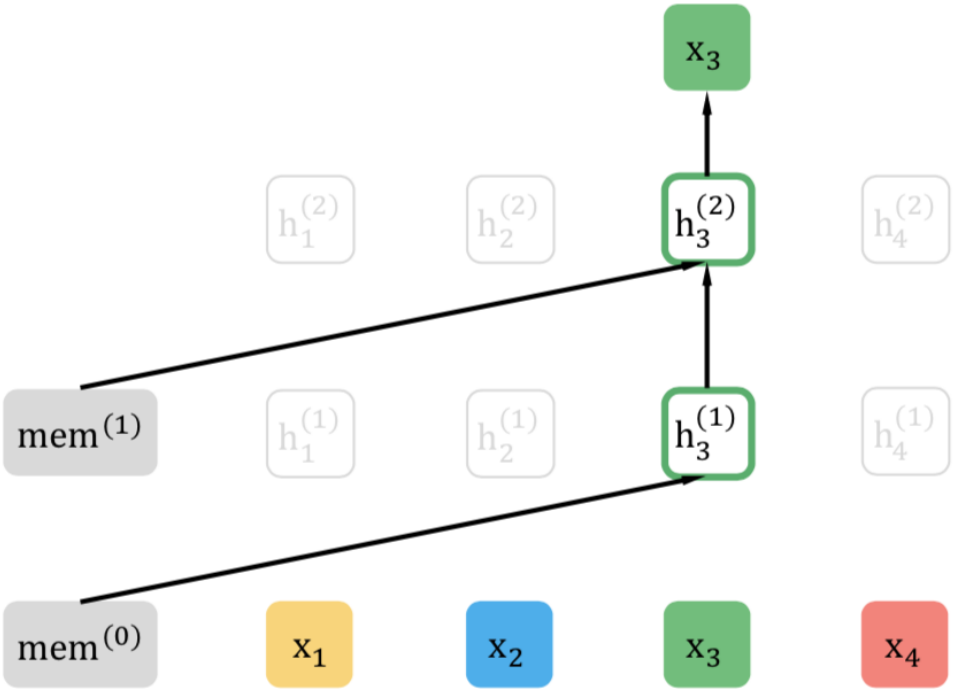
Z_T : sequence 길이가 T일 때 가능한 모든 permutations
z_t : t번째 element
$\mathbf{z}_{<t}$: t-1 elements of a permutation $\boldsymbol{z} \in Z_T$

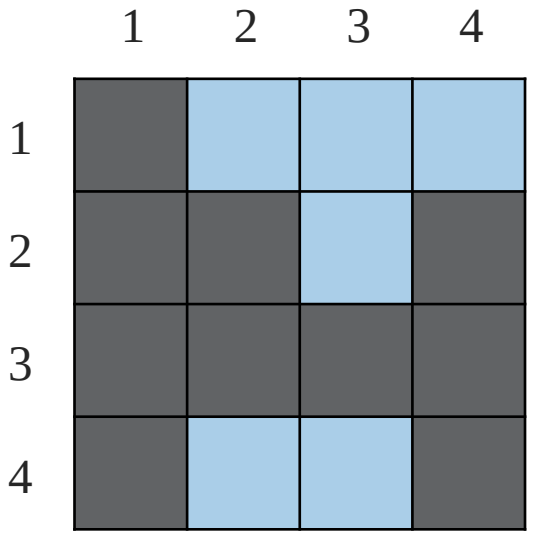
- Remark on Permutation
  - Original sequence 순서는 유지하고, attention mask를 통해 permutation 고려
    - Finetuning시에는 natural order의 sequence만 입력 받기 때문

Permutation Language Modeling (PLM)

- Factorization order : 3 → 2 → 4 → 1

$$p(x) = p(x_3|mem)p(x_2|mem,x_3)p(x_4|mem,x_3,x_2)p(x_1|mem,x_3,x_2,x_4)$$



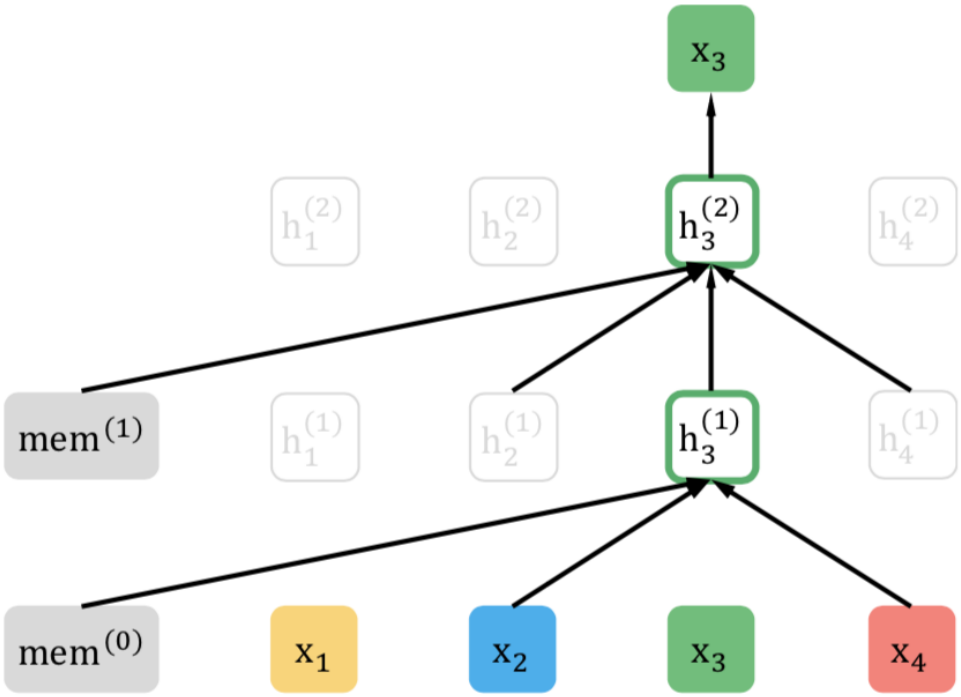


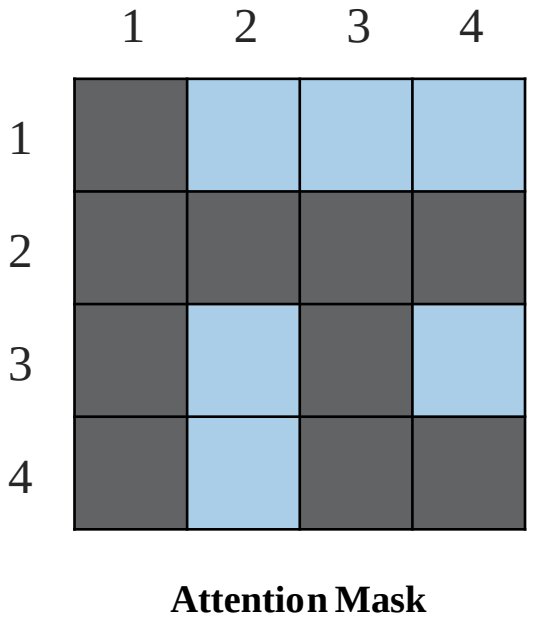
Attention Mask

Permutation Language Modeling (PLM)

- Factorization order : 2 → 4 → 3 → 1

$$p(x) = p(x_2|mem)p(x_4|mem,x_2)p(x_3|mem,x_2,x_4)p(x_1|mem,x_2,x_4,x_3)$$





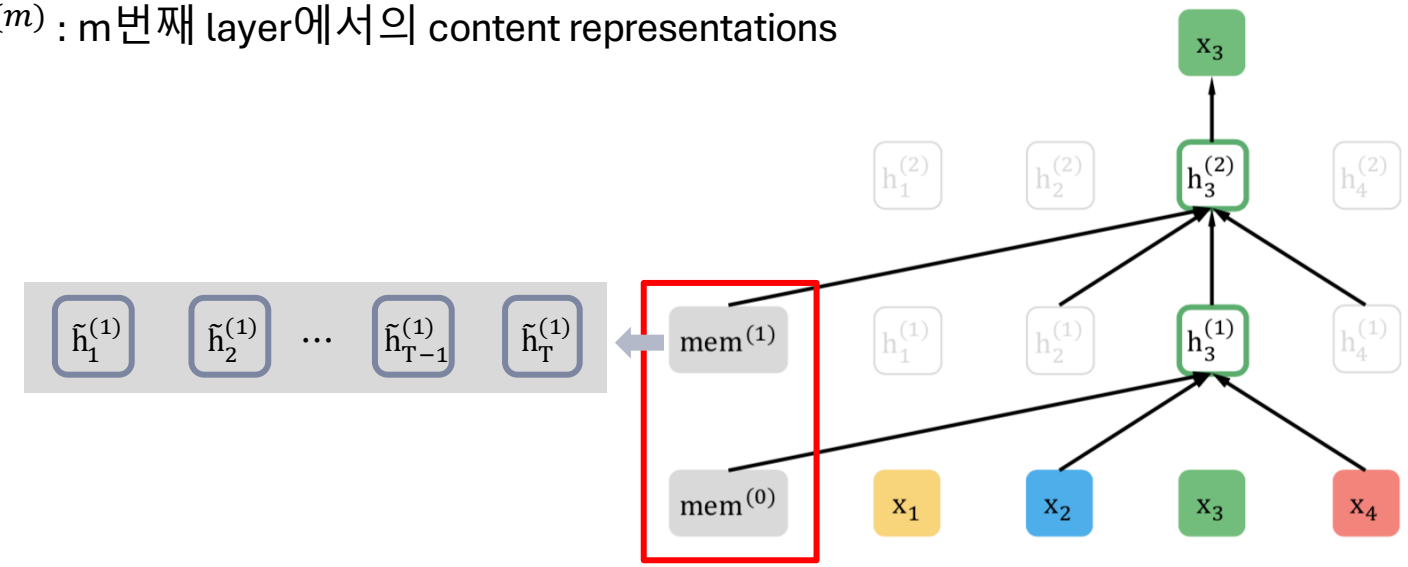
Attention Mask

Permutation Language Modeling (PLM)

- Segment Recurrence Mechanism (from Transformer-XL)
  - 긴 문장을 여러 segment로 분리하고 segment 정보를 저장해 놓은 후 다음 segment에서 이전 segment의 hidden states를 재사용

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \left[\tilde{\mathbf{h}}^{(m-1)}, \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}\right]; \theta)$$

$\tilde{\mathbf{x}} = \mathbf{s}_{1:T}$ and $\mathbf{x} = \mathbf{s}_{T+1:2T}$
$\tilde{\boldsymbol{z}}$: permutations of $[1, \ldots, T]$
$\mathbf{z}$: permutations of $[T+1, \ldots, 2T]$
$\tilde{\mathbf{h}}^{(m)}$: m번째 layer에서의 content representations



Problem of Standard Parameterization

- Proposed objective의 likelihood 부분에 softmax function을 적용하면 아래와 같음

$$p_\theta(X_{z_t} = x|\mathbf{x}_{\mathbf{z}<t}) = \frac{\exp(e(x)^{\mathrm{T}} h_\theta(\mathbf{x}_{\mathbf{z}<t}))}{\sum_{x'} \exp(e(x')^{\mathrm{T}} h_\theta(\mathbf{x}_{\mathbf{z}<t}))}$$

- Hidden representation $h_\theta(\mathbf{x}_{\mathbf{z}<t})$는 현재 target index 이전의 context tokens에만 의존
  - 이렇게 계산된 값은 예측하고자 하는 target index의 position에 관계없이 같은 값을 갖게 되는 문제 발생

Input sequence $[x_1, x_2, x_3, x_4]$*와 index의 permutation*

$Z_T = [[1,2,3,4],[1,3,2,4],\ldots[4,3,2,1]]$ *에 대해 학습을 진행한다고 가정해 보겠습니다.*

1. *[2,3,1,4]의 경우* $p(x_1 \mid x_2, x_3)$*을 계산하기 위해* $h_\theta(x_2, x_3)$ *과 같은 representation을 이용합니다.*
2. *[2,3,4,1]의 경우에도* $p(x_4 \mid x_2, x_3)$*을 계산하기 위해* $h_\theta(x_2, x_3)$ *과 같은 representation을 이용합니다.*
3. *결과적으로* ***같은 representation을 이용하여*** x_4***과*** x_1***을 예측해야 하는 문제****가 발생합니다.*

- 이를 해결하기 위해 이전의 context tokens($\mathbf{x}_{\mathbf{z}<t}$) 뿐만 아니라 target index의 position 정보(z_t)도 함께 이용하는 Target Position-Aware Representation 제안 : $h_\theta(\mathbf{x}_{\mathbf{z}<t}) \rightarrow g_\theta(\mathbf{x}_{\mathbf{z}<t}, z_t)$

Two-Stream Self-Attention

- g_θ 수식화 조건
  - Token x_{z_t}와 $g_\theta(\mathbf{x}_{\mathbf{z}<t}, z_t)$을 예측하기 위해서 position z_t와 이전의 content $\mathbf{x}_{\mathbf{z}<t}$만 사용해야 함. 현재 시점의 content x_{z_t}는 사용하면 안 됨.
  - $j > t$일 때, 다른 token x_{z_j}를 예측하려면 완전한 context information을 제공하기 위해 $g_\theta(\mathbf{x}_{\mathbf{z}<t}, z_t)$가 content x_{z_t}를 인코딩 해야함
- 위 조건을 만족하기 위해 두 hidden representations를 사용
  - Content representation $h_\theta(\mathbf{x}_{\mathbf{z}\leq t})$는 standard hidden state와 비슷한 역할
    - 이전 시점과 현재 시점 content 모두 이용
  - Query representation $g_\theta(\mathbf{x}_{\mathbf{z}<t}, z_t)$는 이전 content와 현재 시점의 position 정보만을 이용하여 계산됨

Two-Stream Self-Attention

- 각각의 layer $m = 1, \ldots, M$의 경우
  - $g_i^{(0)} = w$: trainable vector, $h_i^{(0)} = e(x_i)$: word embedding

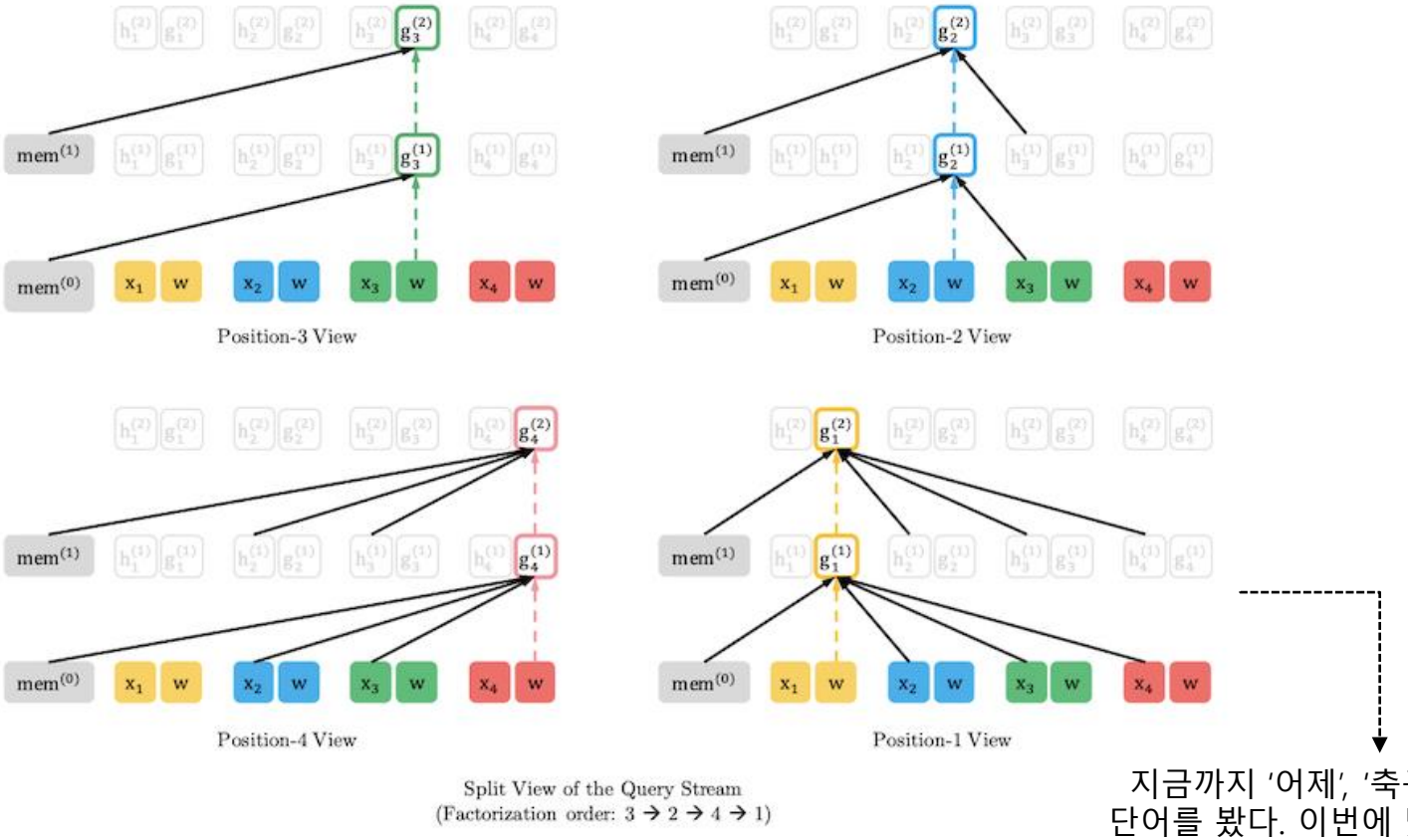
$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t})$$
$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}).$$

- Finetuning 중에는 query stream을 제거

Two-Stream Self-Attention-Query Stream

$$g_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = g_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{<t}}^{(m-1)}; \theta), \quad \text{(query stream: use } z_t \text{ but cannot see } x_{z_t}\text{)}$$

예) 나는 어제 축구를 봤다

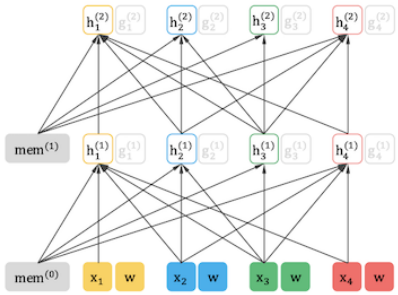


Split View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)

지금까지 '어제', '축구를', '봤다'라는 단어를 봤다. 이번에 맞춰야 할 단어는 원래 문장에서 첫번째 위치에 있다. 그렇다면 이 단어는 무엇일까?

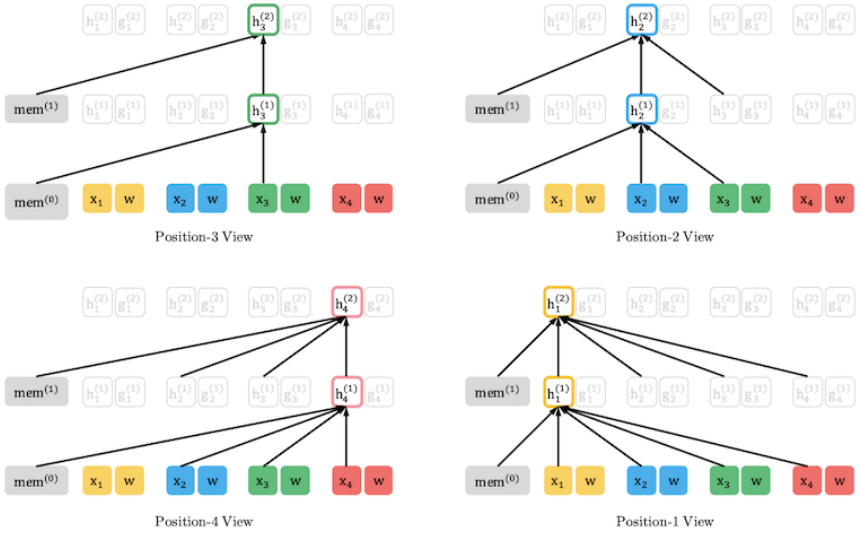
Two-Stream Self-Attention-Content Stream

$$h_{z_t}^{(m)} \leftarrow \text{Attention}(\text{Q} = h_{z_t}^{(m-1)}, \text{KV} = \mathbf{h}_{\mathbf{z}_{\leq t}}^{(m-1)}; \theta), \quad \text{(content stream: use both } z_t \text{ and } x_{z_t}\text{)}.$$



Joint View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)

Split View



Position-3 View | Position-2 View

Position-4 View | Position-1 View

Split View of the Content Stream
(Factorization order: 3 → 2 → 4 → 1)

Two-Stream Self-Attention

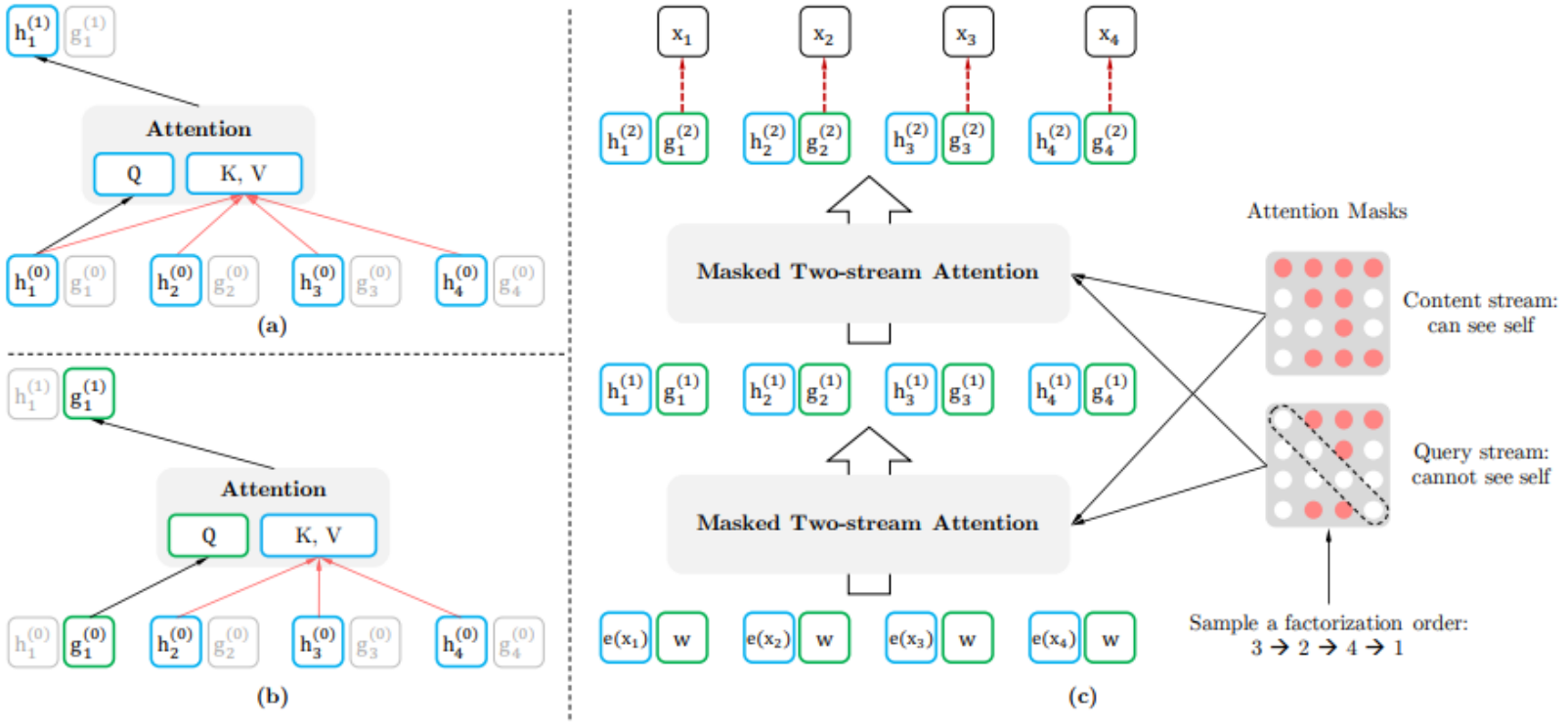


Figure 1: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t}. (c): Overview of the permutation language modeling training with two-stream attention.

Partial Prediction

- Permutation language modeling은 순열로 인해 최적화가 어렵고 수렴이 오래 걸림
  - 이를 해결하기 위해 특정 순서에서 마지막 몇 개만 예측하는 방법 사용

$$\max_{\theta} \quad \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log p_\theta(\mathbf{x}_{\mathbf{z}_{>c}} \mid \mathbf{x}_{\mathbf{z}_{\le c}})\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=c+1}^{|\mathbf{z}|} \log p_\theta(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}})\right].$$

$z_{\le c}$: non-target subsequence
$z_{>c}$: target subsequence
c : cutting point

$$p(x) = p(x_3|mem)p(x_2|mem,x_3)p(x_4|mem,x_3,x_2)p(x_1|mem,x_3,x_2,x_4)$$
$$\rightarrow p(x_4|mem,x_3,x_2)p(x_1|mem,x_3,x_2,x_4)$$

  - Hyperparameter K는 예측을 위해 1/K개의 token을 선택하기 위해 사용됨

    -즉, $|z|/(|z|-c) \approx K$