XLNet: Generalized Autoregressive Pretraining for Language Understanding

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

한양대학교 인공지능연구실
송지수

XLNet

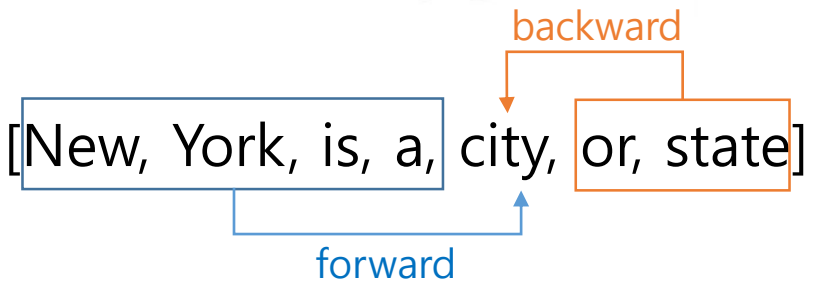
- Transformer-XL (Dai et al., 2019) 을 backbone 모델로 사용하는 Pretraining Language Model

- 20개의 NLP downstream task에서 BERT를 이기고 그 중 18개의 task에서 SOTA를 찍은 모델

Autoregressive vs Autoencoding

Autoregressive (AR)

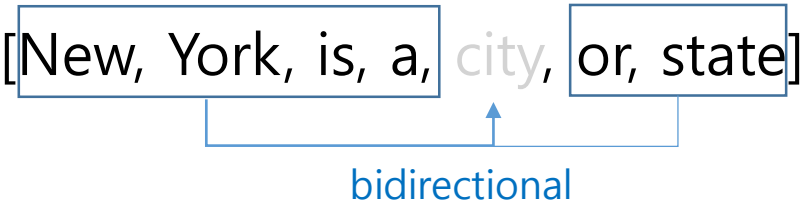
$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t})$$



ELMo, GPT, Word2Vec, …

Autoencoding (AE)

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}})$$



BERT

Weakness of AR & AE

- AR
  - 한 쪽 방향의 문맥 정보 (forward or backward) 만 활용 가능

- AE
  - Independent Assumption: 모든 masked token이 독립적으로 예측됨
  - Pretrain-Finetune discrepancy: 실제 corpora 에는 <MASK> 가 없음

Weakness of AR & AE

- Limitation of BERT

[<MASK>, <MASK>, is, a, city, …]

$$\mathcal{J}_{\text{BERT}} = \log p(\textbf{New} \mid \textbf{is a city}) + \log p(\textbf{York} \mid \textbf{is a city}),$$

  - ‘New’ 라는 masked token은 ‘York’ 를 예측하는 데 분명 도움이 될 만한 token 임에도, Independent Assumption에 따라 ‘New’ 는 활용되지 못함

$$\mathcal{J}_{XLNet} = \log p(New \mid is\ a\ city) + \log p(York \mid \boldsymbol{New}, is\ a\ city)$$

Proposed method: XLNet

AR과 AE의 단점을 버리고 장점을 취하기 위해,

Permutation Language Modeling

Target-aware Representation

Two-stream Self-Attention

세 가지의 방법을 제시

Permutation Language Modeling

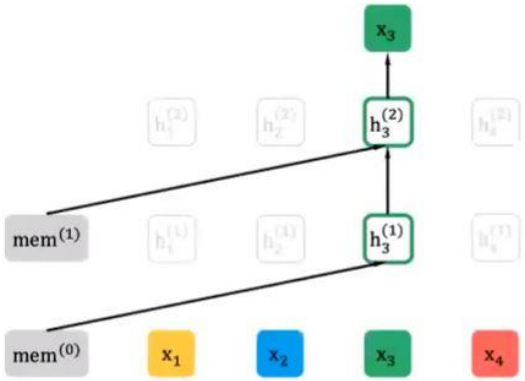
$$input\ sequence : x = (x_1, x_2, \ldots, x_T)$$

$$likelihood : \mathbb{E}_{z \backsim Z_T}[\Pi_{t=1}^{T} p(x_{z_t} \mid x_{z<t})]$$

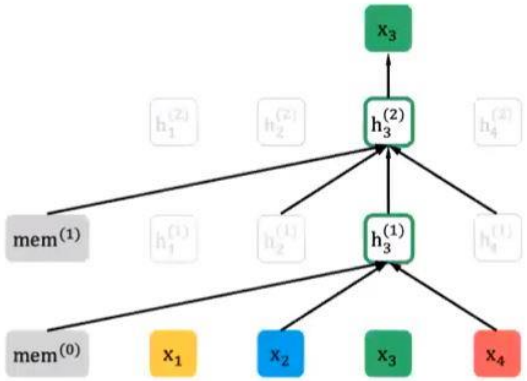
$$training\ objective : \max_{\theta}\ \mathbb{E}_{z \backsim Z_T}[\Sigma_{t=1}^{T} \log\ p_\theta(x_{z_t} \mid x_{z<t})]$$

- Input sequence index의 모든 permutation 을 고려한 AR
- $[x_1, x_2, x_3, x_4]$ 라는 sequence가 있을 때, index의 모든 permutation 의 가짓수는 4! = 24 가지
- 각 permutation 에 대해 AR 방식 적용

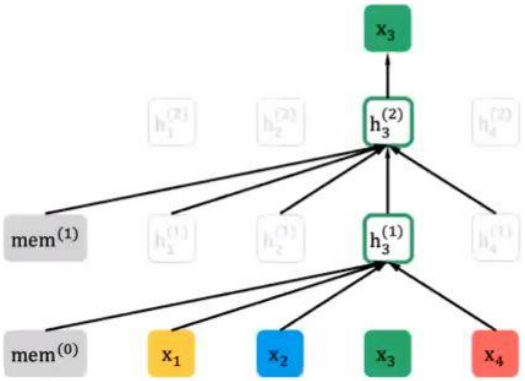
Permutation Language Modeling



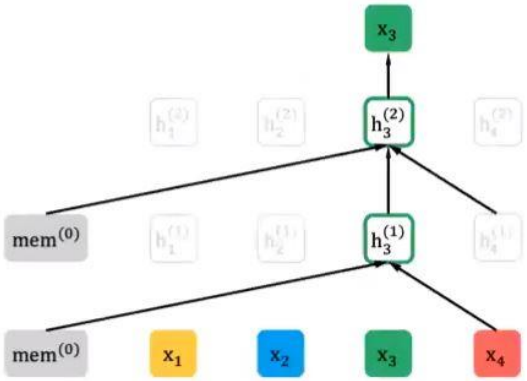
Factorization order: 3 → 2 → 4 → 1



Factorization order: 2 → 4 → 3 → 1



Factorization order: 1 → 4 → 2 → 3



Factorization order: 4 → 3 → 1 → 2

Permutation

[3, 2, 4, 1]

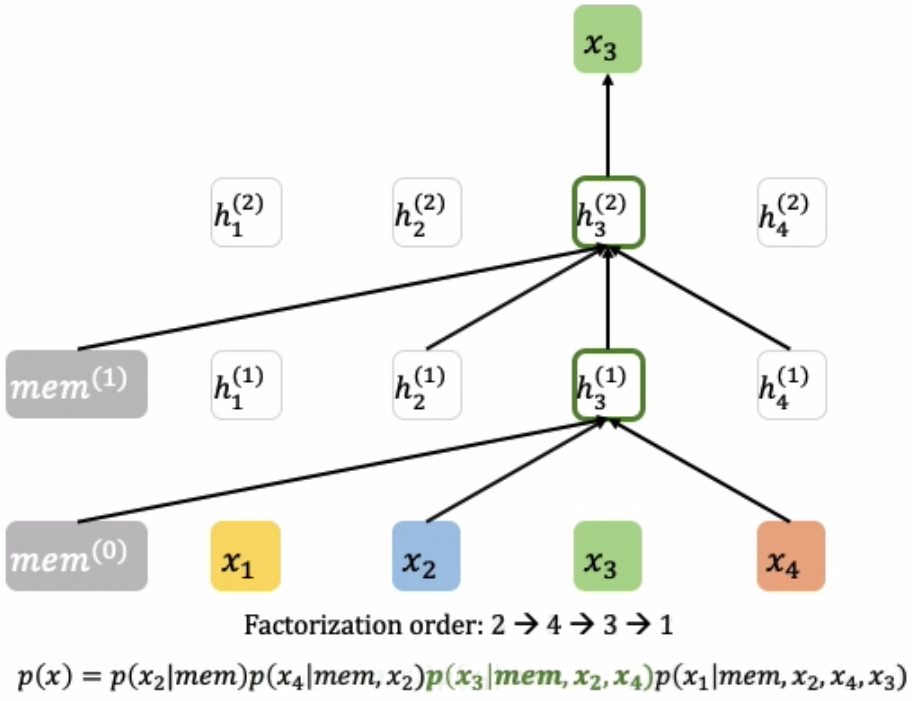
[2, 4, 3, 1]

[1, 4, 2, 3]

[4, 3, 1, 2]

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

Permutation Language Modeling



- x_3 를 예측하기 위해, x_3 제외 나머지 token $[x_1, x_2, x_4]$ 의 모든 부분집합에 conditional 한 x_3 의 probability 계산 가능

- 이를 통해 masking 없이도 AR 방식을 이용해 양방향의 context 를 이용한 모델링이 가능함

Target-aware Representation

- 그러나 Permutation Language Modeling 에서도 치명적인 단점 존재
  - $[x_3, x_1, x_4, x_2]$ 인 경우, x_4를 예측하기 위해 $h_\theta(x_3, x_1)$ 의 representation 이용
  - $[x_3, x_1, x_2, x_4]$ 인 경우, x_2를 예측하기 위해 $h_\theta(x_3, x_1)$ 의 representation 이용

- 즉, permutation 에 따라 같은 representation 을 이용해 다른 target 을 예측해야 하는 상황이 발생

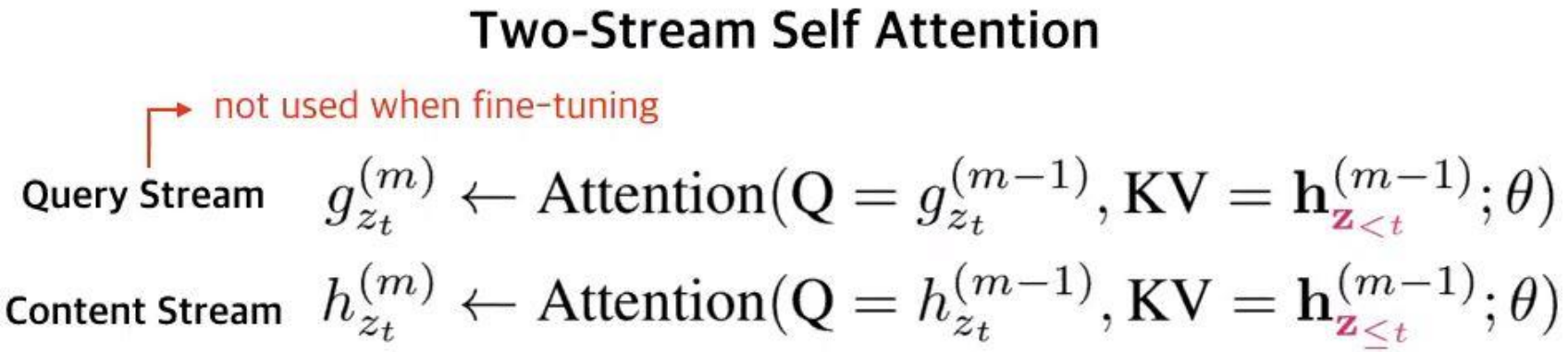
- 이를 피하기 위해 Target position을 encoding 하여 representation 에 함께 이용

Target-aware Representation

- Target-aware representation 을 설계할 때는 다음의 두 가지 조건이 필요
  - t 시점의 token x_{z_t}를 예측하기 위해서, hidden representation $g(x_{z<t}, z_t)$ 는
    t 시점 이전의 context representation 인 $x_{z<t}$ 와 target position 정보 z_t 만을 사용 가능
  - t 시점 이후인 j 시점의 token x_{z_j}를 예측하기 위해서, hidden representation $g(x_{z<t}, z_t)$ 는
    t 시점의 content 인 x_{z_j} 정보를 포함해야 함

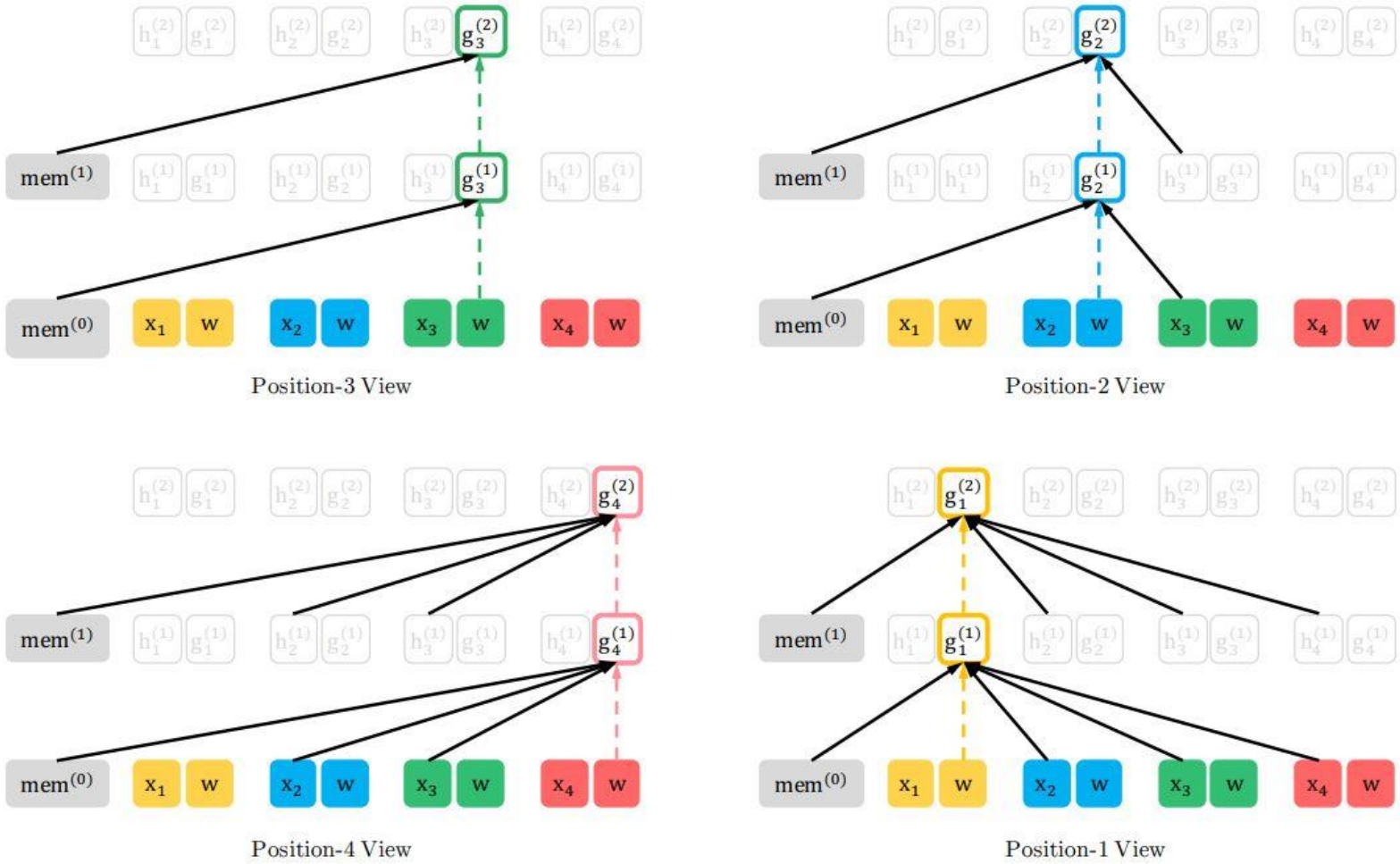
- 통상적인 Transformer: 한 layer 에서 하나의 token은 하나의 representation 가짐
- 저자들은 2개의 representation 을 가질 수 있는 변형된 transformer 구조를 제안

Two-stream Self-Attention



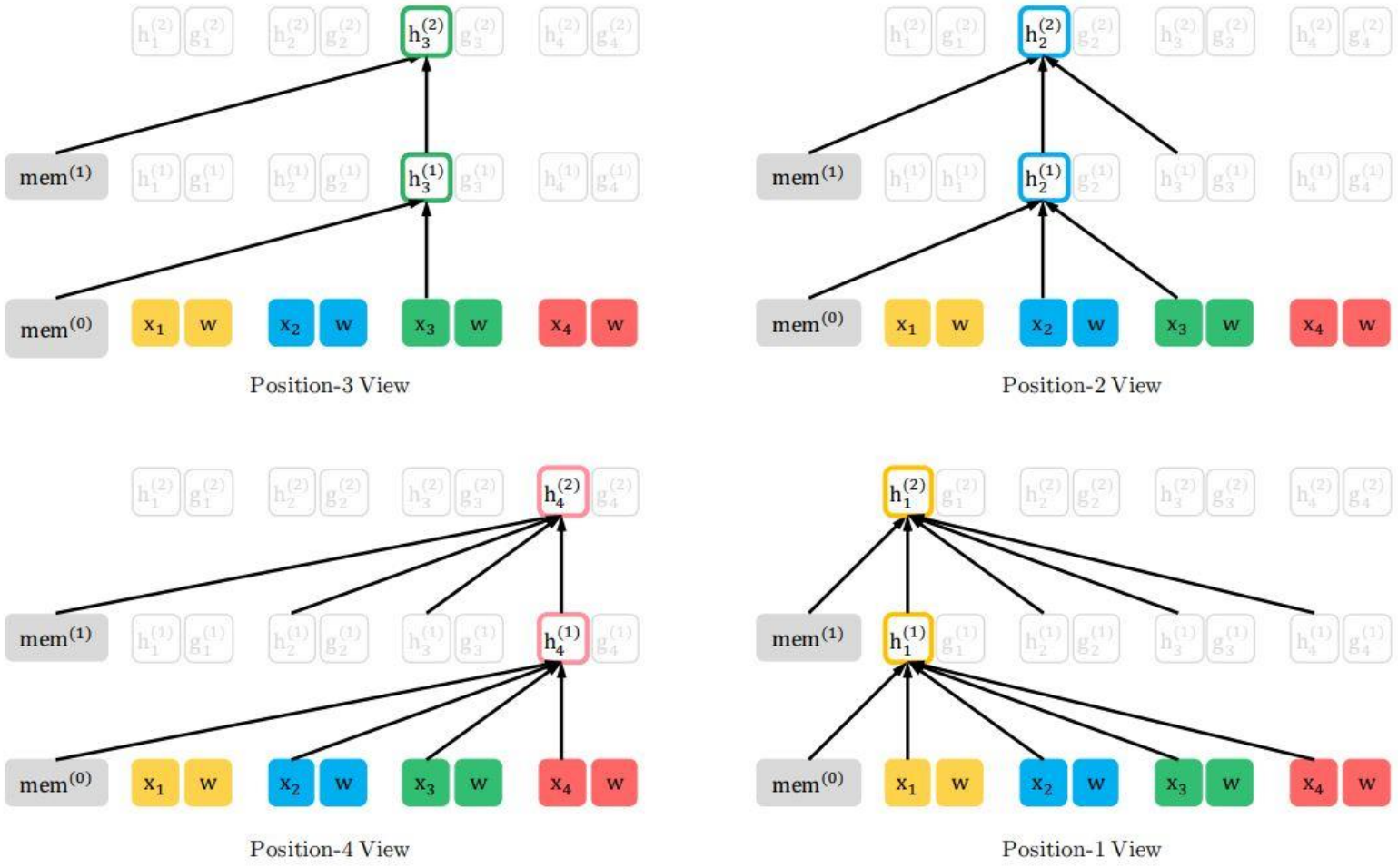
- Query Stream 의 마지막 Layer representation 을 이용하여 특정 position 의 token을 예측
- Content Stream 의 hidden state 를 이용하면 z > t 인 시점의 g를 계산할 때 t의 context 를 이용할 수 있음

Two-stream Self-Attention



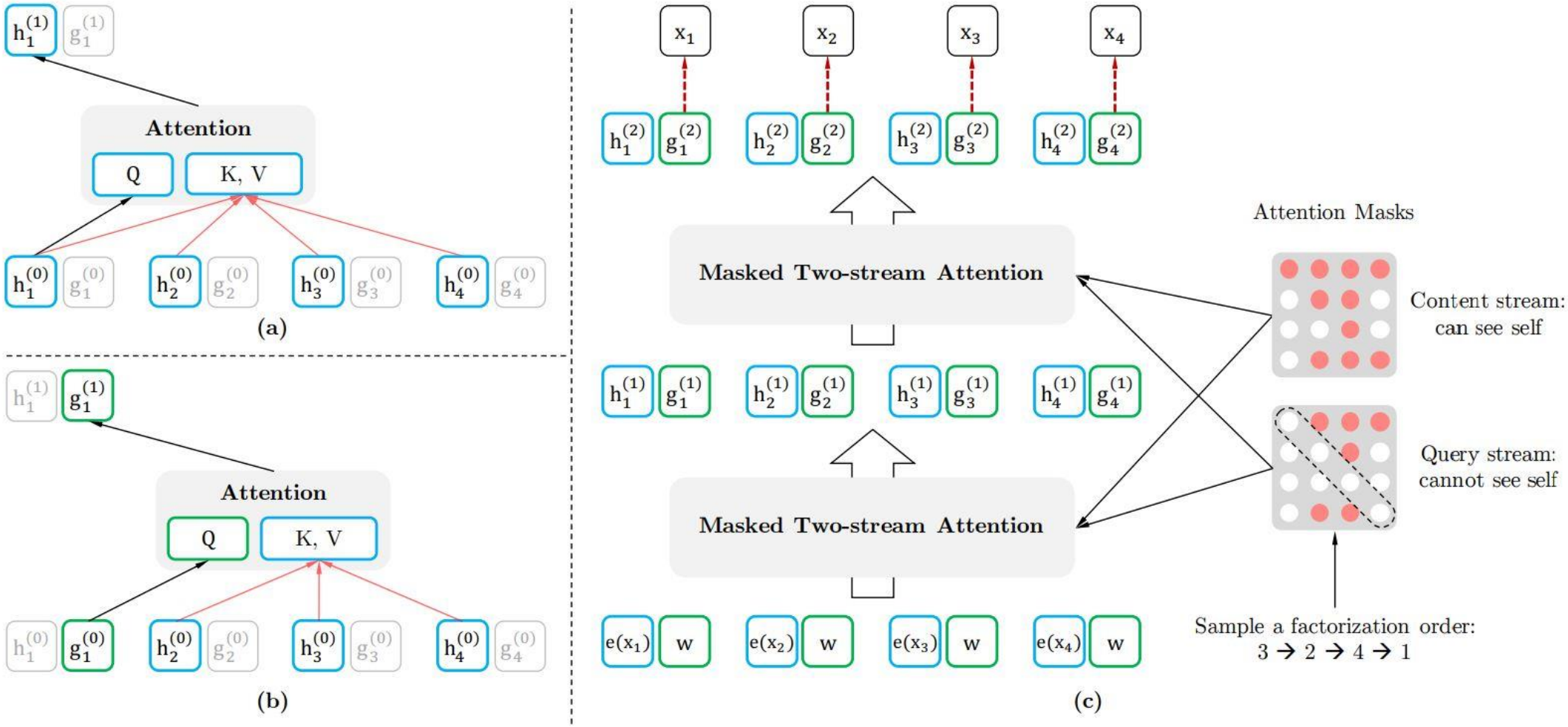
Split View of the Query Stream
(Factorization order: 3 → 2 → 4 → 1)

Two-stream Self-Attention



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

Two-stream Self-Attention



Concept from Transformer-XL

- XLNet 은 긴 문장에 대한 효과적인 이해와 처리를 위하여 Transformer-XL 의 다음 두 가지 개념을 차용함
  - Segment Recurrence Mechanism
  - Relative Positional Encoding

Concept from Transformer-XL

- Segment Recurrence Mechanism
  - 문장이 $\tilde{x}=s_{1:T}$ 와 $x=s_{T+1:2T}$ 의 두 Segment 로 나누어졌을 때, $\tilde{x}$의 permutation $\tilde{z}$ 에 대한 처리를 완료하고 각 마지막 layer 로부터 얻어진 context representation을 $\tilde{h}^{(m)}$ 이라고 한다면 다음 segment x 에 대한 연산은 :

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

  - **서상우 연구원의 발표 자료 참고**

Concept from Transformer-XL

- Relative Positional Encoding
  - 일반적인 Attention Mechanism 의 Attention Score

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

  - Relative Position Encoding 을 적용한 Attention Store

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

  - **서상우 연구원의 발표 자료 참고**

Experiments

RACE	Accuracy	Middle	High
GPT [25]	59.0	62.9	57.4
BERT [22]	72.0	76.6	70.1
BERT+OCN* [28]	73.5	78.4	71.5
BERT+DCMN* [39]	74.1	79.5	71.8
XLNet	**81.75**	**85.45**	**80.21**

Table 1: Comparison with state-of-the-art results on the test set of RACE, a reading comprehension task. * indicates using ensembles. "Middle" and "High" in RACE are two subsets representing middle and high school difficulty levels. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large). Our single model outperforms the best ensemble by 7.6 points in accuracy.

SQuAD1.1	EM	F1	SQuAD2.0	EM	F1
Dev set results without data augmentation					
BERT [10]	84.1	90.9	BERT† [10]	78.98	81.77
XLNet	**88.95**	**94.52**	XLNet	**86.12**	**88.79**
Test set results on leaderboard, with data augmentation (as of June 19, 2019)					
Human [27]	82.30	91.22	BERT+N-Gram+Self-Training [10]	85.15	87.72
ATB	86.94	92.64	SG-Net	85.23	87.93
BERT* [10]	87.43	93.16	BERT+DAE+AoA	85.88	88.62
XLNet	**89.90**	**95.08**	XLNet	**86.35**	**89.13**

Table 2: A single model XLNet outperforms human and the best ensemble by 7.6 EM and 2.5 EM on SQuAD1.1. * means ensembles, † marks our runs with the official code.

Experiments

Model	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	WNLI
Single-task single models on dev									
BERT [2]	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**	-
Single-task single models on test									
BERT [10]	86.7/85.9	91.1	89.3	70.1	94.9	89.3	60.5	87.6	65.1
Multi-task ensembles on test (from leaderboard as of June 19, 2019)									
Snorkel* [29]	87.6/87.2	93.9	89.9	80.9	96.2	91.5	63.8	90.1	65.1
ALICE*	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8
MT-DNN* [18]	87.9/87.4	96.0	89.9	**86.3**	96.5	92.7	68.4	91.1	89.0
XLNet*	**90.2/89.7**†	**98.6**†	90.3†	**86.3**	**96.8**†	**93.0**	67.8	**91.6**	**90.4**

Table 4: Results on GLUE. ∗ indicates using ensembles, and † denotes single-task results in a multi-task row. All results are based on a 24-layer architecture with similar model sizes (aka BERT-Large). See the upper-most rows for direct comparison with BERT and the lower-most rows for comparison with state-of-the-art results on the public leaderboard.

- 비교적 긴 문장을 사용하는 task에서 더욱 성능 향상이 돋보임 (RACE, SQuAD)

Thank you!