XLNet: Generalized Autoregressive Pretraining for Language Understanding

김용
skystar23454566@gmail.com
NLP
2024/05/07

TRAIN AND TEST

Abstract

- Denoising autoencoding based의 BERT가 bidirectional contexts를 모델링 하는데 좋은 performance를 보여줬지만 한계가 있음.

- Input에 대한 mask를 하는 방식으로 corruption에 의존

➔Masking 한 토큰 간의 관계를 무시함 / pretrained와 finetune 간의 불일치

TRAIN AND TEST

Abstract

XLNet : autoencoding 방식이 아닌 autoregressive 방식을 이용함.

Autoencoding이란 ? -> input을 잠재 공간에 넣고 다시 원래 shape으로 돌아오게 하여 학습. -> bert의 encoder를 생각하면 됨.

Autoregressive란? -> 순서에 의미가 있는 데이터를 하나씩 확률을 계산하는 방식.

TRAIN AND TEST

introduction

- AR(autoregressive) modeling이란?

$$x = (x1, .., xT)$$

$$p(x) = \prod_{t=1}^{T} p(xt \mid x_{<T}) \; or \; p(x) = \prod_{t=T}^{1} p(xt \mid x_{>t})$$

ELMo에서는 concat하는 방식을 채택했는데 이전의 AR modeling은 bidirectional 한 context를 이해하는데 효율적이지 않다.

왜 bidirectional? -> downstream task에서는 bidirectional context information을 요구

TRAIN AND TEST

introduction

- AE(autoencoding) modeling이란?

$$x = (x1, .., xT)$$

Input sequence가 있을 때, 특정 tokens들을 mask를 하고, 복원하는 방식.

Bidirectional contexts for reconstruction

BERT-> 예측된 토큰들이 각각의 다른 unmasked tokens들과 independent라고 하는 단점이 있다. AR 모델링과는 다르게 조건부 확률 사용X

Ex) 나는 아침식사를 먹었다. -> 나는 다음의 단어를 예측할 때 각 토큰이 서로 독립적으로 예측되기 때문에 나는 이후에 올 수 있는 다음단어를 직접적으로 고려x

TRAIN AND TEST

Introduction

- XLNet

1. Bidirectional 하게 context를 학습할 수 있는 all possible permutations of the factorization order. -> 기존 AR 모델링의 한계 극복
2. Autoencoding 방식에서의 한계인 pretrain-finetuning discrepancy를 일으키지 않는다.

Related Work

- 모든 순열 경우의 수를 고려하는 것인데, 이는 ‘orderless’한가?

➔그렇지 않다.

XLNet은 two-stream attention을 이용하여 target position을 hidden state에 포함.

background

AR language modeling

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)},$$

$h_\theta(\mathbf{x}_{1:t-1})$ Context representation., Access only token to the left.

$e(x)$ Embedding of x

TRAIN AND TEST

background

BERT -> based on denoising auto-encoding.

$$\max_{\theta} \quad \log p_\theta(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t \mid \hat{\mathbf{x}}) = \sum_{t=1}^{T} m_t \log \frac{\exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x_t)\right)}{\sum_{x'} \exp\left(H_\theta(\hat{\mathbf{x}})_t^\top e(x')\right)},$$

$$x_{hat} = [x1, [mask], \ldots, xT]$$
$$x_{bar} = [x1, x2, \ldots, xT]$$

$$m_t = 1 \; indicates \; x_t \; is \; masked$$

$H_\theta(\mathbf{x})_t$ Access to the contextual information on both sides.

Ex) t = 2, t = 6 시점에서 [mask]토큰이 있다고 할 때,

$$\log p_\theta(x_2|x_{hat}) + \log p_\theta(x_6|x_{hat})$$

TRAIN AND TEST

Permutation Language Modeling

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

Z_T : set of all possible permutations of the length-T index sequence [1,2,…,T]

X_t 가 자기 자신을 제외한 모든 possible element in the sequence를 보기 때문에 bidirectional context를 이해할 수 있다.

TRAIN AND TEST

Remark on Permutation

- Only permute factorization order, not the sequence order.

➔Keep the original sequence order. Using positional encodings

기존 AR modeling식과는 달라질 필요가 있다.

Remark on Permutation

$$\max_{\theta} \quad \log p_\theta(\mathbf{x}) = \sum_{t=1}^{T} \log p_\theta(x_t \mid \mathbf{x}_{<t}) = \sum_{t=1}^{T} \log \frac{\exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x_t)\right)}{\sum_{x'} \exp\left(h_\theta(\mathbf{x}_{1:t-1})^\top e(x')\right)},$$

$h_\theta(\mathbf{x}_{\mathbf{z}<t})$ T 이하의 sequence에 대한 representation

➔ 어떤 위치의 position을 예측할 것인지 라는 변수에 depend하지 않는다.

Sequence [1,2,3,4]가 있을 때, [1,2 | 3,4]를 예측하는경우와 [1,2 | 4, 3]을 예측하는 경우가 있는데, 전자는 original sequence의 3번째 토큰을 예측하는 것이고 후자는 4번째 토큰을 예측하는 것이기 때문에 문제가 된다.

➔ XLNet은 sequence order가 아닌 factorization order를 permute함.

➔ Positional encoding, Z_t를 이용한다.

Remark on Permutation

$$p_\theta(X_{z_t} = x \mid \mathbf{x}_{z_{<t}}) = \frac{\exp\left(e(x)^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)}{\sum_{x'} \exp\left(e(x')^\top g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right)},$$

TRAIN AND TEST

Two-Stream Self-Attention

g_θ를 어떻게 구성할 것인가?

1. Sequence Z의 t시점 토큰인 X_Z_t에 대한 정보를 알기 위해서는 position 정보인 Z_t를 이용해야한다. Not X_Z_t
2. X_Z_j (j > t) 를 예측하기 위해서는 우리는 x_Z_t에 대한 정보를 알고 있어야 한다.

Standard transformer에서는 각 layer에서 한 token당 하나의 representation을 갖는 구조인데, 이 구조는 위의 두 조건을 동시에 만족시킬 수 없습니다.

TRAIN AND TEST

Two-Stream Self-Attention

$g_i^{(0)} = w$ Trainable vector.

$h_i^{(0)} = e(x_i)$ Embedding of x

m = 1, …., M (self-attention layer)

$h_\theta(\mathbf{x}_{\mathbf{z}_{\leq t}})$ h_{z_t}

$g_\theta(\mathbf{x}_{\mathbf{z}_{<t}}, z_t)$ g_{z_t}

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

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

TRAIN AND TEST

Query Representation

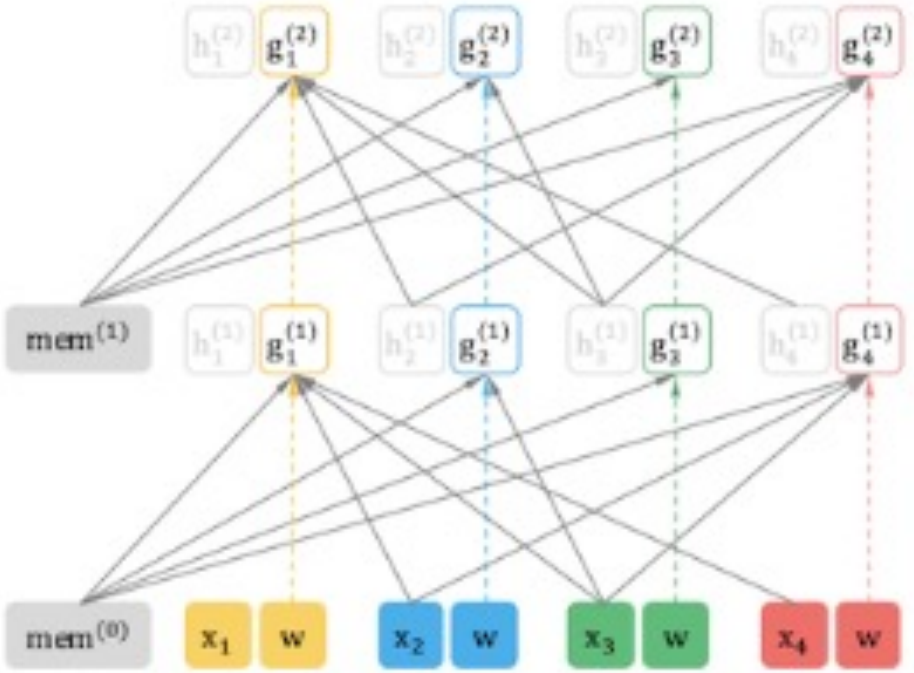
1. 현재 시점을 제외한 이전 시점 token들의 content와 현재 시점 위치정보를 이용하여 계산되는 representation입니다.

2. 첫번째 layer의 query stream은 random trainable variable (w)로 초기화하고 최종 representation까지의 흐름(stream), 각 layer의 state는

$$Query\ Stream:\ g_{z_t}^{(m)} \leftarrow Attention(Q = g_{z_t}^{(m-1)}, KV = h_{z_{<t}}^{(m-1)}; \theta)$$

TRAIN AND TEST

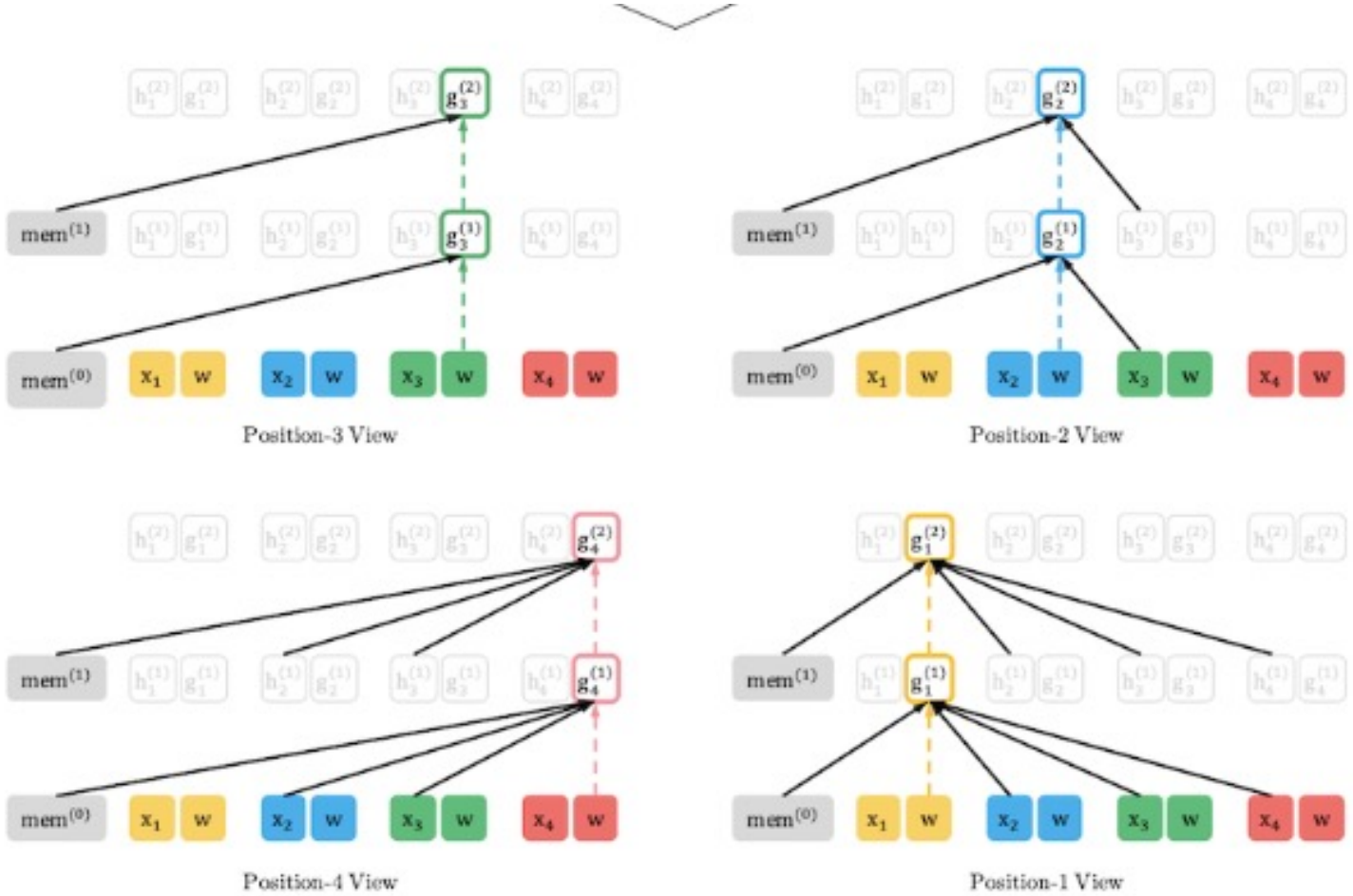
Query representation



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

TRAIN AND TEST

Query representation



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

TRAIN AND TEST

context representation

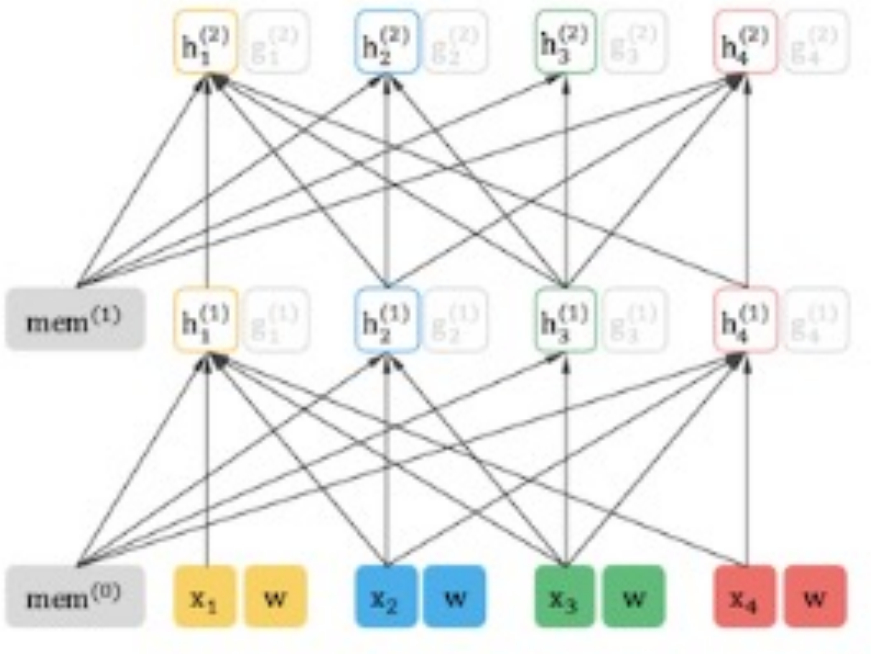
1. 현재 시점 및 이전 시점 token 들의 content를 이용하여 계산되는 representation입니다.

2. 첫번째 layer의 content stream은 해당 위치 token의 word embedidng으로 초기화 하고 최종 representation까지의 흐름(stream), 각 layer의 state는 다음과 같이 계산할 수 있습니다.

$$Content\ Stream:\ h_{z_t}^{(m)} \leftarrow Attention(Q = h_{z_t}^{(m-1)}, KV = h_{z_{<t}}^{(m-1)}; \theta)$$

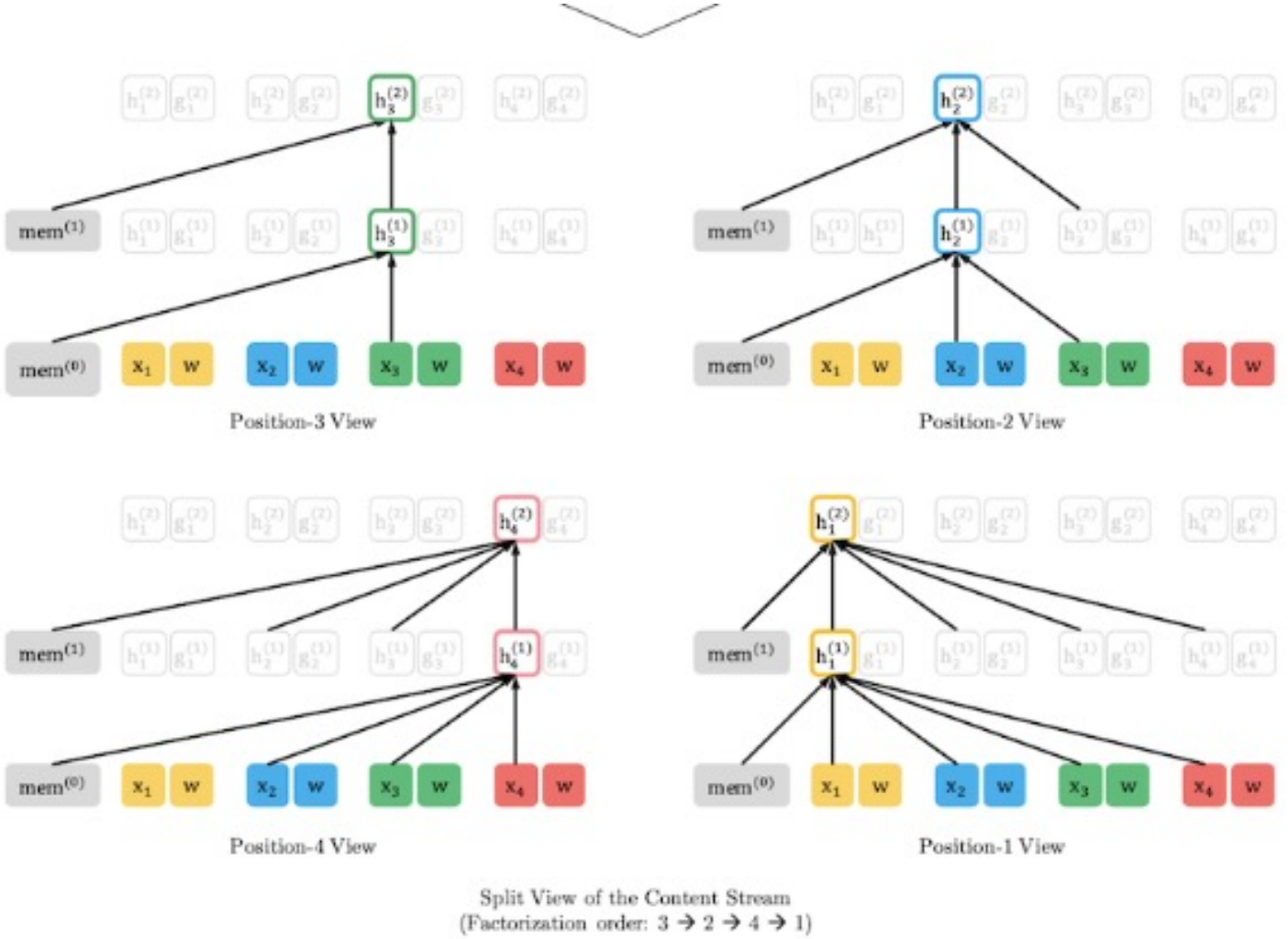
TRAIN AND TEST

context representation



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

context representation



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

TRAIN AND TEST

Partial prediction

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

- Sequence 내의 모든 단어들을 예측하기는 힘들다.
- -> cutting point를 정하고 그 이후의 단어들을 예측한다.

TRAIN AND TEST

Incorporating Ideas from Transformer-XL

Relative Positional Encoding

E : word embedding
U : absolute positional encoding

- Attention score in standard Transformer

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

- Attention score with Relative Positional Encoding

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

1. Term (b)와 (d)에서 기존 absoloute positional embedding U를 relative positional embedding R_i-j로 대체합니다. R은 learnable parameters가 아닌 sinusoid encoding matrix입니다.
2. Term (c)와 (d)에서 $U_i^T U_q^T$를 각각 $u^T, v^T \in R^d$로 대체합니다. Query vector가 모든 query position에 대해 같기 때문에, 다른 단어들에 대한 attention bias가 query position에 상관없이 동일하게 유지되어야 합니다.
3. $W_k \Rightarrow W_{K,E}$와 $W_{k,R}$로 분리합니다. 이는 content기반의 key vector와 location기반의 key vector를 각각 만들어 내기 위함입니다.

TRAIN AND TEST

Incorporating Ideas from Transformer-XL

Segment Recurrence Mechanism

XLNet은 Transformer-XL과 마찬가지로 긴 문장에 대해서 여러 segment로 분리하고 이에 대해서 recurrent하게 모델링을 합니다.

1. 어떻게 permutation setting에 recurrence mechanism을 적용할 것
2. 모델이 이전 segment로부터 얻어진 hidden state를 재사용할 수 있게 하는 것

Ex) 어떤 긴 문장이 $x^{\sim} = s_{1:T}\ and\ x = s_{T+1:2T}$ 두 세그먼트로 나뉩니다.

$z^{\sim} \Rightarrow [1 \ldots T]$의 permutation, z => [T+1…2T]의 permutation.

첫번째 segment($z^{\sim}$)에 대한 처리를 완료하고 $h^{\sim(m)}$을 caching합니다.

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

TRAIN AND TEST

Discussion and Analysis

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

TRAIN AND TEST

Results

RACE Dataset

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.

100K 개의 중국 중/고등학생을 위한 질문과 전문가들이 단 정답으로 이루어져 있습니다.

TRAIN AND TEST

Results

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.

TRAIN AND TEST

Results

Text Classification

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [14]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [14]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [30, 20]	4.32	-	-	0.70	4.95	-	-
ULMFiT [13]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.79**	**1.55**	**27.80**	**0.62**	**4.49**	**2.40**	**32.26**

Table 3: Comparison with state-of-the-art error rates on the test sets of several text classification datasets. All BERT and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

TRAIN AND TEST

Results

GLUE Dataset

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.

TRAIN AND TEST

TRAIN AND TEST