XLNET

학번/이름 : V2022117 / 황민규

전공 : 메타버스 테크놀로지

XLNet은 최근에 대부분의 NLP 테스크들에서 state-of-the-art 성능을 달성하고 있던 BERT를 큰 차이로 outperform 하면서 파장을 일으켰습니다.

XLNet은 GPT로 대표되는 auto-regressive(AR) 모델과 BERT로 대표되는 auto-encoder(AE) 모델의 장점만을 합한 generalized AR pretraining model입니다.

이를 위해 permutation language modeling objective과 two-stream attention mechanism을 제안합니다.

해당 모델은 다양한 NLP 테스크에서 기존 대비 상당한 향상을 보이며 state-of-the-art 성능을 보였습니다.

1. AR, AE의 단점 :

AR(Auto Regressive)의 단점 : AR은 방향성(forward, backward)이 정해져야 하므로, 한쪽 방향의 정보만을 이용할 수 있습니다. 따라서 양방향 문맥을 활용해 문장에 대해 깊이 이해하기 어렵습니다. ELMO의 경우 양방향을 이용하지만, 각각의 방향에 대해 독립적으로 학습된 모델을 이용하므로 얕은 이해만 가능합니다.

AE(Auto Encoding)의 단점 : independant assumption을 통하여 [mask] token들이 각각 독립적으로 존재하며, 이들간의 관계를 알아낼 수 없다는 단점이 있다. 또한 [maske] token 자체가 fine-tuning 과정에서 등장하지 않기에, pre-training과 fine-tuning 사이에 불일치가 발생한다.

2. XLNET :

기본적인 개념으로는 {X1, X2, X3, X4}라는 4개의 인풋을 이용해 가능한 모든 조합을 생성한다. 바뀐 순서에 따라 앞, 뒤에 오는 인풋이 바뀌게 되고, 이를 통해 AR기법으로 양방향 학습이 가능해진다.

더 나아가 AR을 사용하기에, AE방법에서의 [mask] token이 필요없으며, 모든 단어간의 관계도 학습이 가능하다.

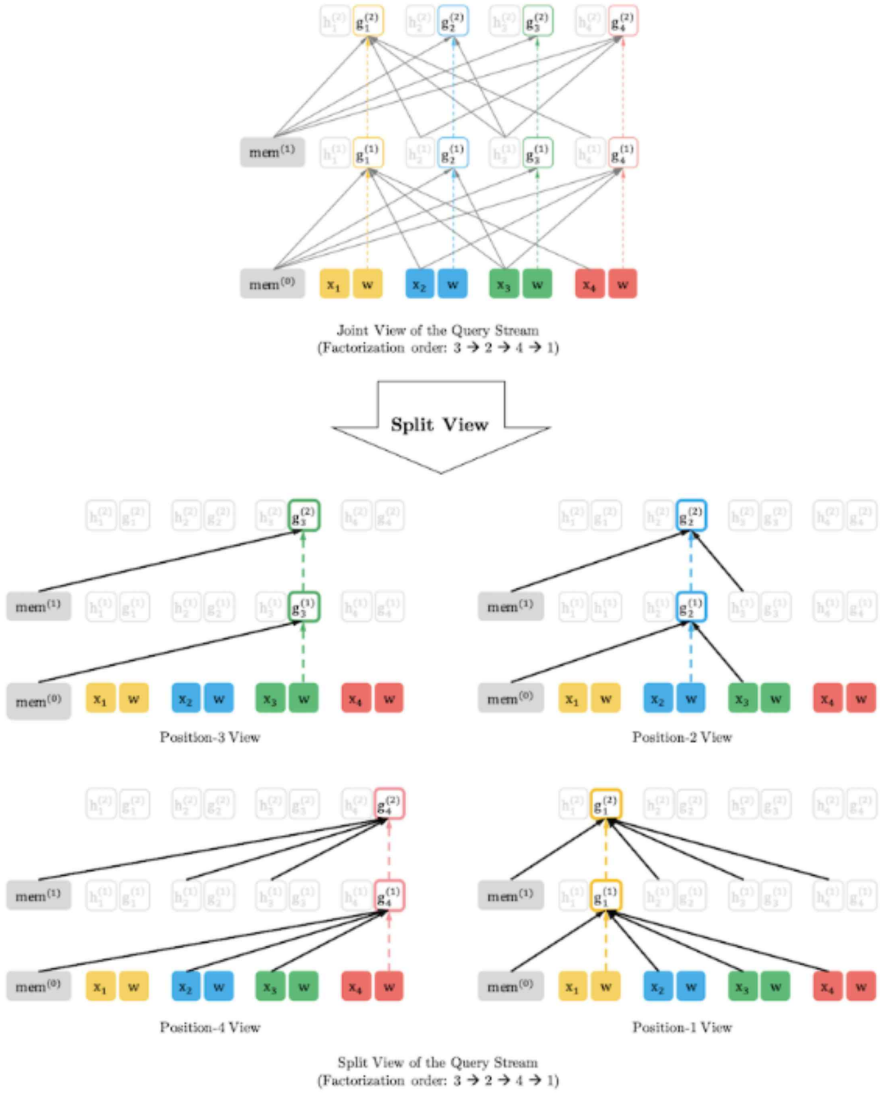
2.1 Permutaion Language Modeling Objective

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

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

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

input sequence index의 모든 permutation을 고려한 AR 방식을 이용. ex) [x1, x2, x3, x4]에 대한 permutation은 4!=24개(Zt=[[1,2,3,4], [1,2,4,3], ... , [4,3,2,1]]). 이 Zt에 대해 AR LM의 objective를 적용하면 아래와 같음.(mem(m)은 Transformer-XL의 memory state)



2.2 Target-Aware Representation for Transformer

> *Input sequence* $[x_1, x_2, x_3, x_4]$*와 index의 permutation*
> $Z_T = [[1,2,3,4],[1,3,2,4],\dots[4,3,2,1]]$ *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.*

이 문제점을 해결하기 위해서 이전의 context token들의 정보 (xz<t) 뿐만 아니라 target index의 position 정보 (zt)도 함께 이용하는 새로운 Target Position-Aware Representation을 제안합니다: hθ(xz<t)→gθ(xz<t,zt).

2.3 Architecture: Two-Stream Self-Attention for Target Aware Representation

이제 target position 정보를 추가적으로 이용하는 gθ를 어떻게 구성할지의 문제가 남아 있습니다. 그에 앞서, gθ의 조건은 다음과 같습니다.

1. 특정 시점 t에서 target position z_t 의 token x_{z_t}을 예측하기 위해, hidden representation $g(x_{z<t}, z_t)$는 t 시점 이전의 context 정보 $x_{z<t}$ 와 target position 정보 z_t 만을 이용해야 합니다.
2. 특정 시점 t 이후인 $j\ (>t)$ 에 해당하는 x_{z_j} 를 예측하기 위해, hidden representation $g(x_{z<t}, z_t)$ 가 t 시점의 content인 x_{z_t} 를 인코딩해야 합니다.

Standard transformer는 각 layer에서 한 token당 하나의 representation을 갖는 구조입니다. 이 구조는 위의 두 조건을 동시에 만족시킬 수 없습니다. 이와 같은 문제를 해결하기 위해서, 저자들은 2개의 hidden representation을 이용하는 변형된 transformer 구조를 제안합니다.

2.3.1 Query Representation: gθ(xz<t,zt)

이전 시점 토큰들의 content와 현재 시점의 위치 정보를 이용하여 계산되는 representation.

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

2.3.2 Context Representation : hθ(xz≤t)

현재 시점 및 이전 시점 토큰들의 content를 이용하여 계산되는 representation.

standard transformer의 hidden state와 동일한 역할.

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

위에 소개된 objective는 permutation을 이용해 가능한 모든 조합의 순서로 maximum likelihood를 수행하는데, 이는 학습 시에 굉장히 속도가 느림.

이를 극복하기 위해, 특정 순서에서 마지막 몇 개의 예측만 이용하는 방법을 사용. ex) 3-2-4-1 순서로 예측한다고 가정하면 아래처럼 마지막 2개만 예측에 이용.

ex) : p(x3)p(x2 | x3)p(x4 | x2,x3)p(x1 | x3,x2,x4)→p(x4 | x2,x3)p(x1 | x3,x2,x4)

3. Incorporationg Ideas from Transformer-XL

긴 문장에 대한 처리를 위해 Transformer-XL에서 사용된 2가지 테크닉을 차용.

3.1 Relative Positional Encoding

Transformer는 input에 절대적 위치에 대한 representation(absolute positional encoding)을 추가하는 방식으로 순서에 대한 모델링이 가능.

이런 transformer의 방식은 하나의 segment 내에서는 위치에 대한 의미를 표현할 수 있으나, 길이가 긴 문장에서 multiple segment에 대해 recurrent 모델링을 할 때 문제가 발생.

$$h_{\tau+1} = f(h_\tau, E_{s_{\tau+1}} + U_{1:L})$$
$$h_\tau = f(h_{\tau-1}, E_{s_\tau} + U_{1:L})$$

- 위 수식을 보면 두개의 다른 순서의 h인데 +U_(1:L)이 동일. (E는 word embedding) - 이러면 모델이 τ번째 segment의 첫 번째 단어와 τ+1번째 segment의 첫 번째 단어를 위치 상으로 같다고 인식할 수 있는 문제가 발생. - 이런 transformer의 방식은 하나의 segment 내에서는 위치에 대한 의미를 표현할 수 있으나, 길이가 긴 문장에서 multiple segment에 대해 recurrent 모델링을 할 때 문제가 발생. - 이런 multiple segments에서 발생하는 문제를 해결하기 위해 input-level이 아닌 self-attention mechanism에서 relative positional encoding을 제안해 단어 간의 상대적 위치 정보를 모델링에 사용.

- 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

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

- 아래가 Relative Positional Encoding을 적용한 Self-attention에서 attention score를 구하는 식. - (a)는 content 기반의 처리. - (b)는 content에 의존한 positional bias를 잡아냄. - (c)는 global content bias를 인코딩. - (d)는 global positional bias를 인코딩.

3.2 Segment Recurrence Mechanism

긴 문장에 대해서 여러 segment로 분리하고 이에 대해서 recurrent하게 모델링.

Segment-level recurrence를 XLNet에 적용하기 위해서 2가지 포인트에 주목.

어떻게 permutation setting에 recurrence를 적용할 것인지 => 긴 문장을 2가지 segment로 나눠서 하나의 segment에 대한 처리를 완료 후 각 layer m으로부터 얻어진 content representation을 caching함.

다른 segment에 대한 계산은 아래의 수식과 같이 나타냄.

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

이를 통해 과거 segment에 대한 factorization order를 고려하지 않고 memory의 caching과 reusing이 가능.