XLNet: Generalized Autoregressive Pretraining for Language Understanding

날짜	@2021년 4월 25일 → 2021년 5월 8일
배정	Yujin Kim 상민 상민 이 유 유경 한 KM S 예진 문
상태	완료

1. Introduction

AR(GPT), AE(BERT) 문제점

AR(AutoRegressive) language model

- 앞의 token을 이용해 문장의 확률 분포를 알아냄
- 순방향(forward)든 역방향(backward)든 단방향 context
- bidirectional context 정보가 필요할 때에는 적합하지 않음

AE(AutoEncoding) based LM = BERT

- 망가진 입력값으로부터 원본을 재구성하는데에 목표로 함 (input sequence의 특정 부분에 `MASK` 적용)
- bidirectional context 사용하기 때문에 성능 향상
- 하지만 `MASK`가 pretrain에서는 사용되는데 finetuning에서는 사용되지 않음 = 불일치 문제 발생
- 예측 token이 `MASK`처리 되어 있어서 AR language modeling에서처럼 joint probability 계산 불가

  = predicted token과 unmasked token이 서로 독립적이다

  = high-order & 장거리 의존성 있는 자연어를 너무 단순화 시켰다는 것

XLNet

: AR과 AE를 최대한 활용할 수 있는 generalized AR method

: permutation-based (AR) language modeling

- 인수분해 순서에 따른 모든 가능한 순열에 대해 sequence의 log likelihood를 최대화

  → 각 위치에 대한 문맥은 양방향 모두 학습 "capturing bidirectional context"

- 데이터 손상에 의지하지 않음 = pretrain-fintune 일치

  → joint probability 계산 가능(token의 독립적 추정 ㄴㄴ)

- pretraining 설계 개선

  1. segment recurrence mechanism & Transformer-XL의 relative encoding scheme을 pretraining에 적용 → 긴 text sequence를 갖는 task 성능 향상
  2. Transformer(-XL)을 reparameterize하여 target의 모호성 제거

     → Transformer? 인수분해(factorization) 순서 무작위 & target 모호함 = XLNet에 적용하기 어려움

2. Proposed Method

2.1 Background

AR language modeling과 BERT의 language model pre-training을 비교

- text sequence X = [X1,X2....,Xt]가 주어지면 다음 식의 likelihood를 maximizing 하는 방향으로 pre-traing을 진행한다.

$$\max_{\theta} \quad \log p_\theta(x) = \sum_{t=1}^{T} \log p_\theta(x_t|x_{<t}) = \sum_{t=1}^{T} \log \frac{exp(h_\theta(x_{1:t-1})^\top e(x_t))}{\sum_{x'} exp(h_\theta(x_{1:t-1})^\top e(x'))} \quad (1)$$

- $h_\theta(x)$: RNN 또는 transformer와 같은 neural model로부터 생성된 content representation e(x) : embedding
- BERT는 text sequence x에 대해 token 일부(e.g., 15%)를 **[MASK]** 로 설정함으로써 손상된 x_hat 를 구성한다. masked token을 x_var 라 할때 training objective는

x_hat로부터 x_var 를 재구성하는 것이다.

$$\max_{\theta} \quad \log p_\theta(\overline{x}|\hat{x}) \approx \sum_{t=1}^{T} m_t \log p_\theta(x_t|\hat{x}) = \sum_{t=1}^{T} m_t \log \frac{exp(H_\theta(\hat{x})_t^\intercal e(x_t))}{\sum_{x\prime} exp(H_\theta(\hat{x})_t^\intercal e(x\prime))} \quad (2)$$

- m_t = 1 이면 x_t는 **[MASK]** 이며, H theta 는 length-T text sequence x로 부터 transformer 를 통해 얻어지는 hidden vector H_theta(x) = [H_theta(x)1, H_theta(x)2, ..., H_theta(x)T] 이다.

두 가지 pre-train의 장단점을 비교

- **Independence Assumption:** BERT는 마스킹된 token *x*가 독립되게 재구성된다는 가정에 기초하여 joint conditional probability *p*(*x* ㅣ *x*^)를 인수분해 한다. 반면에 AR language modeling은 식1과 같이 곱의 규칙을 사용하여 *pθ*(*x*)를 인수분해 한다.
- **Input noise:** BERT의 input에는 downstream task에서는 사용하지 않는 **[MASK]** 와 같은 symbol이 사용되기 때문에 pre-train과 fine-tune간에 차이가 발생한다.
- **Context dependency:** AR representation *hθ*(*x*1:*t*−1) 는 위치 *t*까지의 token에 대해서만 계산되지만 반면에 BERT representation *Hθ*(*x*)*t*는 bidirectional contextual information에 접근할 수 있다. 결과적으로 BERT는 bi-directional context를 더 잘 capture할 수 있도록 pre-train된다.

2.2 Objective: Permutation Language Modeling

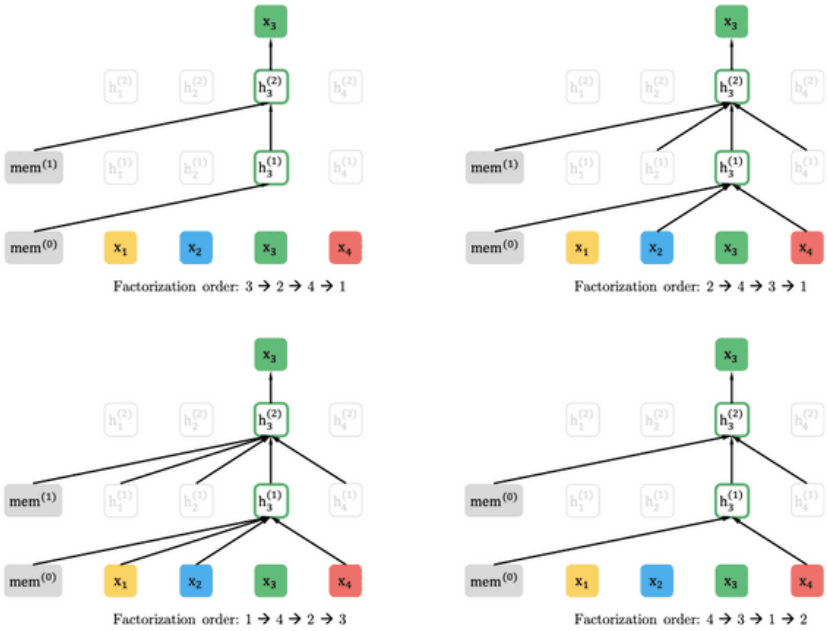

Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence x but with different factorization orders.

- AR language modeling과 BERT는 다른 language modeling보다 고유한 장점을 가지고 있는데, 단점을 피하면서 둘의 장점을 가져오는 pre-training objective에 초점을 맞춤
- orderless NADE에서 아이디어를 차용하여 AR model의 장점을 유지하고 model이 bi-directional context를 capture할 수 있도록 permutation language modeling objective를 제안
- 길이가 T인 시퀀스 x에 대해 T!만큼 autoregressive 인수분해를 수행하여 model parameter들이 모든 인수분해 순서들에 걸쳐 공유되면, model은 양측의 모든 위치에서 정보를 모으는 방법을 train하게 됨
- ZT는 길이 $length-T$의 index sequence [1,2,...,T]의 모든 가능한 permutation 집합이라 정의한다. zt와 $z<t$를 사용하여 permutation $z \in ZT$의 t번째 element와 첫 번째 element $t-1$를 나타낸다
- 제안하는 permutation language modeling은 다음과 같이 표현할 수 있다.

$$\max_{\theta} \quad \mathbb{E}_{z \sim Z_T} \left[\sum_{t=1}^{T} \log P_\theta(x_{z_t} | x_{Z_{<t}}) \right]. \quad (3)$$

text sequence x에 대해 인수분해 순서 z를 sampling하고 인수분해 순서에 따라 likelihood $p\theta(x)$를 decompose 한다. 동일한 parameter θ가 공유되어 학습되는 동안, xt'는 xi != xt인 모든 element를 보기 때문에 bi-directional context를 capture할 수 있다.

- **Remark on Permutation:** 제안하는 방식은 sequence 순서가 아닌 인수분해 순서만 바꾼다. 즉 원래의 sequence 순서를 유지하고 원본 sequence에 해당하는 positional encoding을 사용하여 인수분해 순서 permutation에 해당하는 attention mask를 얻는다.

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

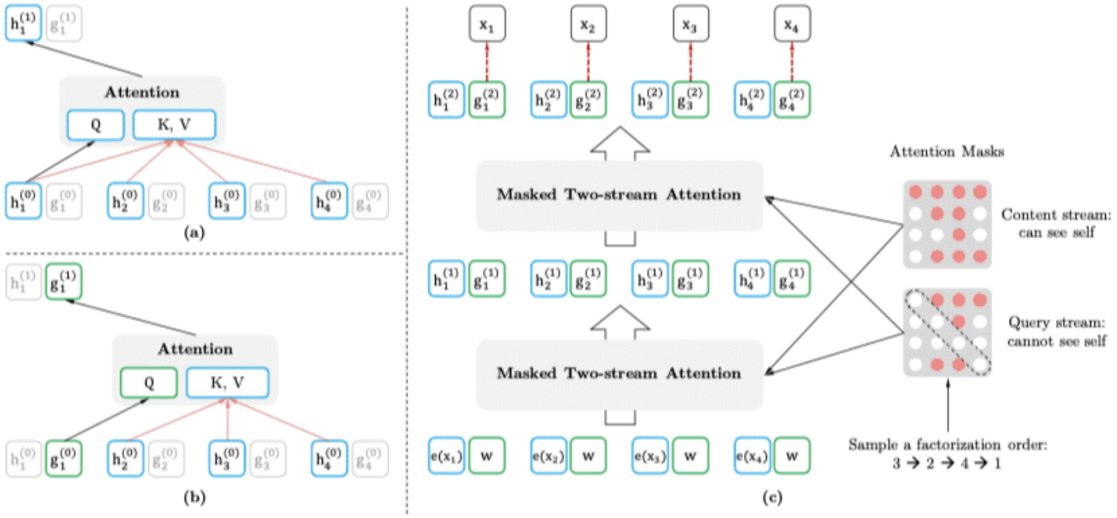

Figure 2: (a): Content stream attention, which is the same as the standard self-attention. (b): Query stream attention, which does not have access information about the content x_{z_t}. (c): Overview of the permutation language modeling training with two-stream attention.

$$p_\theta(X_{z_t} = x | x_{z_{<t}}) = \frac{exp(e(x)^\top h_\theta(x_{z_{<t}}))}{\sum_{x\prime} exp(e(x\prime)^\top h_\theta(x_{z_{<t}}))}$$

permutation language modeling을 기존 transofrmer에 적용하기는 어려움이 있다. 그래서 softmax를 사용하여 next token의 distribution p를 parameter화 함. h는 masking 후 transformer로부터 생성된 x의 hidden representation을 의미하는데, zt가 식에 없음(의존성이 없음). 결과적으로 유용한 representation을 배울 수 없는 위치에 상관없이 동일한 distribution 예측. 그래서 아래와 같이 next-token의 distirubtion이 target postion을 인식할 수 있게 재구성

$$p_\theta(X_{z_t} = x | x_{z_{<t}}) = \frac{exp(e(x)^\top g_\theta(x_{z_{<t}}, z_t))}{\sum_{x\prime} exp(e(x\prime)^\top g_\theta(x_{z_{<t}}, z_t))}$$

제안하는 수식을 transformer architecture에 사용할 수 없는 두 가지 이유가 있다. 첫 번째는 toekn xzt, g세타 를 예측하기 위해 postion zt만 사용해야 한다. context xzt를 사용하면 objective는 너무 간단해짐. 두 번째는 j > t일 때 다른 token xzj를 예측하기 위해 xzt는 완전한 context information을 제공하기 위해 content를 encoding 해줘야 함.

$$h_{zt}^{(m)} \leftarrow Attention(Q = h_{zt}^{(m-1)}, KV = h_{z\leq t}^{(m-1)}; \theta$$

(a) : content stream attention : 예측하고자 하는 토큰의 실제 값 정보를 같이 사용하여 예측 / z는 원래 문장 순서를 random shuffle한 index list, zt는 z의 t번째 요소

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

(b) : query stream attention : 토근, position 정보를 활용한 self attetnion 기법. 예측하고자 하는 target토큰 이전 정보들의 값(postion embedding, random initialization)을 가지고 예측

Parital Prediction

permutation languagemodeling은 순열로 인해, 최적화가 어렵고 수렴이 오래걸린다. 최적화의 어려움을 해결하기 위해 인수분해 순서에서 마지막 token만 예측하고 z를 non target subsequence(z≤c)와 target subsequence(z>c)로 분할한다. 목적은 non target에서 targe의 log-likelihood를 maximize하는 것이다.

usselected token들을 query representation이 계산되지 않기 때문에 메모리를 아끼고 속도를 향상시킬 수 있다.

$$\max_{\theta} \quad \mathbb{E}_{z\sim Z_T}\left[\log p_\theta(x_{Z_{>c}}|x_{Z_{\le c}})\right] = \mathbb{E}_{z\sim Z_T}\left[\sum_{t=c+1}^{|z|}\log p_\theta(x_{z_t}|x_{z_{<t}})\right]$$

2.4 Incorporating Ideas from Transformer-XL

transformer-XL에서 사용되는 두가지 중요한 기술 `relative positional encoding scheme`과 `segment recurrence mechasnism`을 pre-training framework에 포함시켰다.

긴 sequence에서 x˜ = s1:T 와 x = sT +1:2T 두개의 segment를 입력으로 받으며, z˜ 와 z의 [1 · · · T] 와 [T + 1 · · · 2T]의 permutation이라고 가정한다.

z˜를 통해 첫번째 segment를 계산하고, 각 layer m에 대해 얻어진 content representation h˜(m) 를 캐싱한다. next segment x에서 memory를 포함하는 attention update는 다음과 같다.

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

[.,.]은 sequence dimension사이에 concatenation 의미한다. positional encoding은 original sequence의 실제 position에만 의존하기 때문에 $\tilde{h}^{(m)}$이 계산되면 $\tilde{z}$와는 독립적이다. 이를 통해 previous segment의 인수분해 순서에 대한 정보 없이 memory 를 caching하여 다시 사용할 수 있다. model은 last segment의 모든 인수분해 순서에대해 메모리를 활용하는 방법을 train한다고 추측한다. (자세한 그림은 부록 A.7 참고) query stream도 동일한 방법으로 계산할 수 있다.

2.5 Modeling Multiple Segments

XLNet이 autoregressive framework에서 multiple segment를 어떻게 pre-train하는지 설명한다. BERT와 마찬가지로 random하게 선택된 segment를 concatenation하여 하나의 sequence로 permutation language modeling을 진행한다. XLNet의 input은 [CLS, A, SEP, B, SEP]이다. ablation study(Section 3.7참고)에 따라 XLNet-Large는 next sentence prediction을 수행하지 않았다.

Relative Segment Encoding

- architecture상으로 차이점은 BERT의 경우 word embedding에 각 position에 따라 absolute segment embedding을 적용하였고, XLNet은 transformer-XL에서 제안한 아이디어를 확장하여 relative segment encoding을 사용.
- sequence에서 *i*와 *j*인 position pair가 있고 *i*와 *j*가 같은 segment에 있을때, segment encoding $s_{ij} = s_+$또는 $sij = s-$를 사용. $s+$ 와 $s-$는 각 attention head의 learnable model parameter.
- *i*가 *j*에 attend될 때, segment encoding s_{ij}는 attention weight $a_{ij} = (q_i + b)^\top s_{ij}$를 계산하는데 사용되며, q_i는 query vector같은 standard attention operation이고, *b*는 learnable head-specific bias vector이다. 다음으로 value a_{ij}가 normal attention weight에 더해짐.
- **relative segment encoding은 두가지 이점이 있음.**
  1. relative encoding의 inductive bias는 일반화를 향상.
  2. absolute segment encoding을 사용하여 불가능한 두 개 이상의 segment가 있는 task에 대한 fine-tune 가능성을 열어줌.

2.6 Discussion

[New, York, is, a, city]라는 문장(sequence of words)이 주어졌을 때, BERT와 XLNet 모두 예측할 token으로 [New, York] 2개를 선택하여 log p(New York | is a city)를 maximize 해야 하는 상황을 가정하면,

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

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

XLNet은 BERT에서 누락되는 (New, York) pair의 dependency를 capture할 수 있다. 예시를 통해 BERT는 (New, city)와 (York, city)와 같은 일부 dependency를 학습하지만 XLNet은 항상 동일한 대상에서 항상 더 많은 dependenct pair를 학습하고 “denser”한 효과적인 train을 하게 된다.

3 Experiments

3.1 Pretraining and Implementation

- 13GB 텍스트를 결합한 BookCorpus와 English Wikipedia를 pretraining data로 사용
- Bert보다 10배 많은 데이터 사용
- 가장 큰 모델인 XLNet-Large는 BERT-Large와 같은 구조의 하이퍼파라미터 매개 변수를 가지므로 비슷한 모델 사이즈를 가짐

3.2 Fair Comparison with BERT

Model	SQuAD1.1	SQuAD2.0	RACE	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B
BERT-Large (Best of 3)	86.7/92.8	82.8/85.5	75.1	87.3	93.0	91.4	74.0	94.0	88.7	63.7	90.2
XLNet-Large-wikibooks	88.2/94.0	85.1/87.8	77.4	88.4	93.9	91.8	81.2	94.4	90.0	65.2	91.1

BERT와 XLNet을 공정한 환경에서 비교 (3가지의 가장 좋은 성능을 낸 BERT와 같은 데이터와 하이퍼파라미터로 학습된 XLNet을 비교)

모든 데이터셋에서 BERT를 능가함!

3.3 Comparison with RoBERTa: Scaling Up

RACE	Accuracy	Middle	High	Model	NDCG@20	ERR@20
GPT [28]	59.0	62.9	57.4	DRMM [13]	24.3	13.8
BERT [25]	72.0	76.6	70.1	KNRM [8]	26.9	14.9
BERT+DCMN* [38]	74.1	79.5	71.8	Conv [8]	28.7	18.1
RoBERTa [21]	83.2	86.5	81.8	BERT†	30.53	18.67
XLNet	**85.4**	**88.6**	**84.0**	XLNet	**31.10**	**20.28**

RACE Dataset

GPT, BERT 앙상블 모델들보다 성능이 좋음!

SQuAD2.0	EM	F1	SQuAD1.1	EM	F1
Dev set results (single model)					
BERT [10]	78.98	81.77	BERT† [10]	84.1	90.9
RoBERTa [21]	86.5	89.4	RoBERTa [21]	88.9	94.6
XLNet	**87.9**	**90.6**	XLNet	**89.7**	**95.1**
Test set results on leaderboard (single model, as of Dec 14, 2019)					
BERT [10]	80.005	83.061	BERT [10]	85.083	91.835
RoBERTa [21]	86.820	89.795	BERT* [10]	87.433	93.294
XLNet	**87.926**	**90.689**	XLNet	**89.898‡**	**95.080‡**

SQuAD Dataset

Single model들 중에서도 최고 성능을 보임

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [15]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [15]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [31, 23]	4.32	-	-	0.70	4.95	-	-
ULMFiT [14]	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	**3.20**	**1.37**	**27.05**	**0.60**	**4.45**	**2.11**	**31.67**

Text Classification

오분류율도 모두 SOTA 달성!

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	-
RoBERTa [21]	90.2/90.2	94.7	92.2	**86.6**	96.4	**90.9**	68.0	92.4	-
XLNet	**90.8/90.8**	**94.9**	**92.3**	85.9	**97.0**	90.8	**69.0**	**92.5**	-
Multi-task ensembles on test (from leaderboard as of Oct 28, 2019)									
MT-DNN* [20]	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0
RoBERTa* [21]	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0
XLNet*	**90.9/90.9**†	**99.0**†	**90.4**†	**88.5**	**97.1**†	**92.9**	**70.2**	**93.0**	**92.5**

GLUE Dataset

9개의 NLU task를 모은 데이터셋

9개 중 7개가 SOTA, 9개 모두 기존의 BERT보다 더 좋은 성능을 보임!

3.4 Ablation Study

#	Model	RACE	SQuAD2.0		MNLI	SST-2
			F1	EM	m/mm	
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

Line1,2 : 기존 BERT에 Transformer-XL을 반영하면 성능이 좋아짐.

Line 3,4 : BERT + Transformer-XL + Permutation LM 에서 성능이 더 좋아짐.

Line 8 : RACE에서는 next-sent pred를 추가하는게 성능이 더 좋아짐.

4 Conclusions

- XLNet은 AR pretrainig 방법을 일반화한 것으로, Permutation language modeling objective를 사용해 AR과 AE의 장점을 결합한 것이다.
- XLNet의 구조는 AR objective을 작업하는데 적용되고 Transformer-XL와 two-stream attention mechanism을 결합하여 설계되었다.
- 다양한 작업에서 이전 pretraining 목표들보다 상당한 개선을 달성했다.