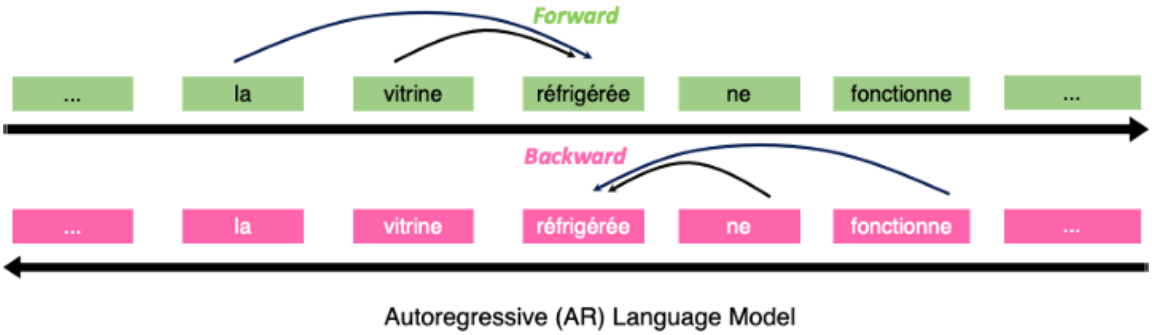
[논문리뷰] XLNet

Introduction

Unsupervised representation learning, 즉 방대한 텍스트 데이터로 pretraining하여 언어 표현을 학습한 뒤 finetuning하는 방식이 좋은 성능을 보이고 있고, 대표적인 학습방법으로 Autoregressive 방법과 AutoEncoding 방법이 있다.

- **Autoregressive:** 이전 토큰들로 다음 토큰을 예측하는 방식으로, 순방향으로하면 forward, 역방향이면 backward로 불린다.



Autoregressive (AR) Language Model

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

연속적으로 들어오는 시퀀스의 likelihood가 최대가 되도록 학습을 진행한다.

해당 방식의 단점은 다른 방향, 즉 양방향에서의 정보는 보지 못한다는 것이다.

- **Autoencoding:** bidirectional model로 임의의 토큰에 마스킹하고 양방향 단어들을 통해 마스킹된 토큰을 예측하는 방식으로 학습한다.

Autoencoding (AE) Language Model

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

예측값과 실제 값의 likelihood를 최대가 되도록 계산하는 과정에서 independent assumption이 들어간다.

Proposed Method

위의 단점들을 극복하기 위해 XLNet에선 새로운 pretraining objective와 architecture를 제안한다.

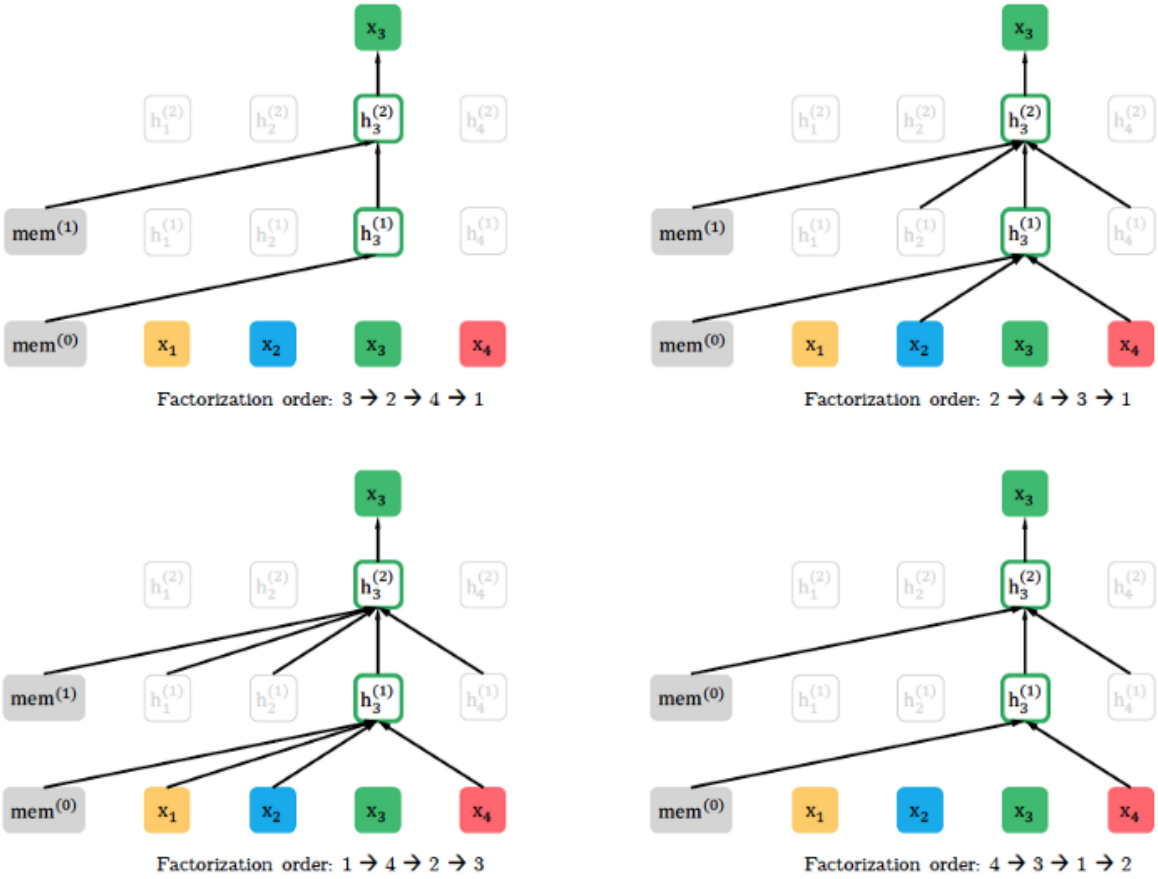
- **Permutation Language Modeling**과 이를 위한 **Two-Stream Self-Attention for Target Aware Representation** → AR과 AE의 문제점 개선
- transforemr-XL의 **Segment-level recurrence mechanism**과 **relative positional encoding** → longer text sequence 성능 향상

Permutation Language Modeling

인풋 시퀀스의 모든 permutation을 활용한 Autoregressive 방식으로 학습한다.

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

- **학습과정**
  - T 길이의 시퀀스가 있을 때 T!개수 만큼의 index 순서가 다른 순열 조합 ZT*ZT*를 만들어 사용한다. (각 token은 원래 순서에 따라 positional encoding이 부여되고 permutation은 index에 대해서만 진행되었다)
  - 순열 조합에서 샘플링을 통해 학습에 사용될 시퀀스를 무작위로 뽑는다.
  - 각 ZT*ZT*에 대해 시퀀스의 likelihood가 최대가 되도록 파라미터를 학습한다. (이 때 파라미터는 모든 순열 조합에서 공유되어 양방향 context를 고려할 수 있도록 한다)



이를 통해 AR 방식에서도 양방향 context를 고려하게 될 수 있음으로써, AR 모델의 한계를 극복할 수 있었다. AR 방식이기에 AE에서 문제가 되었던 independent assumption을 고려하지 않을 수 있게 되었다.

또한 마스크 토큰을 사용하지 않기 때문에 AE에서 지적되었던 pretraining과 finetuning간 불일치를 극복할 수 있었다.

그런데, 해당 방식을 그대로 활용할 수는 없다. 왜냐하면 예측하고자 하는 target token의 index와 상관없이 같은 분포를 갖게 되는 문제가 있기 때문이다.

$$p_\theta(x_{z_t} \mid x_{z<t}) = \frac{exp(e(x)^T h_\theta(x_{z<t})}{\Sigma_{x'} exp(e(x')^T h_\theta(x_{z<t}))}$$

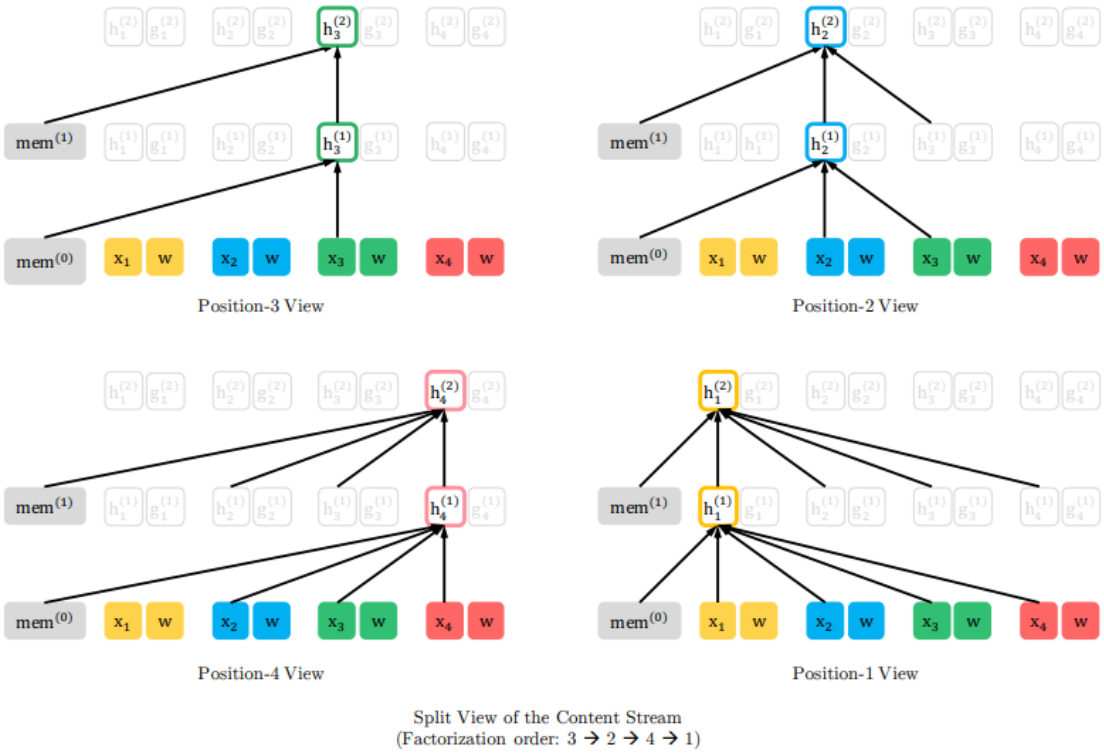
기존 AR방식과 달리 Permutation LM은 index의 순서가 섞여있기 때문에 예측해야하는 토큰의 index가 무엇인지 알 수 없다. 따라서 타겟 토큰의 위치에 따라 다른 분포로 학습되어야 함에도 불구하고 같은 분포로 학습되는 문제가 발생한다.

Two-Stream Self-Attention for Target-Award Representation

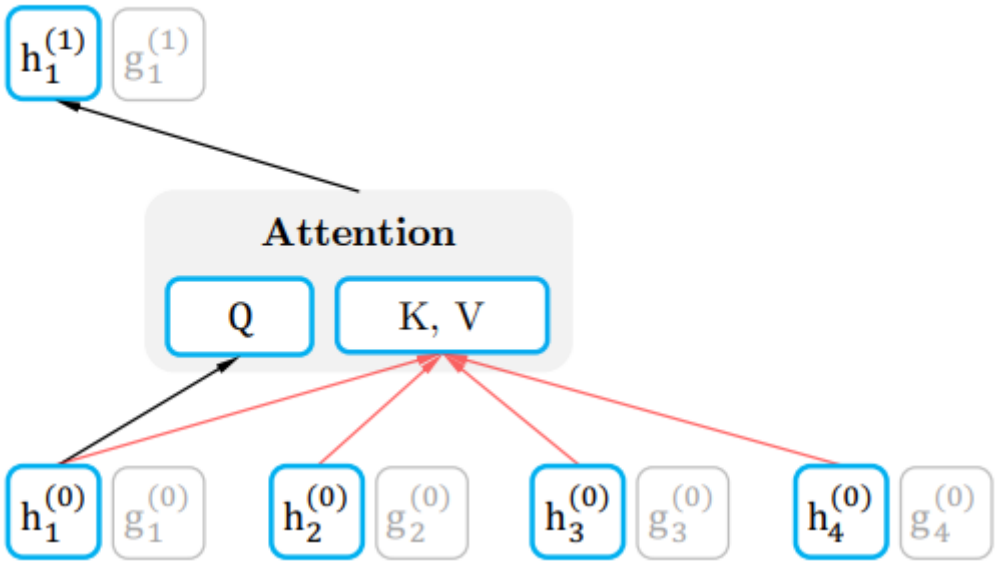
위 문제를 해결하기 위해 예측하려는 토큰 위치 정보 역시 함께 사용하여 target token을 예측하는 새로운 아키텍처를 제안한다.

다음 두개의 stream으로 학습을 진행한다. 본래 standard transformer에선 한 토큰 당 하나의 representation을 가지지만 두개의 stream으로 학습하기 위해 한 token당 두개의 hidden representation을 사용하게 된다.

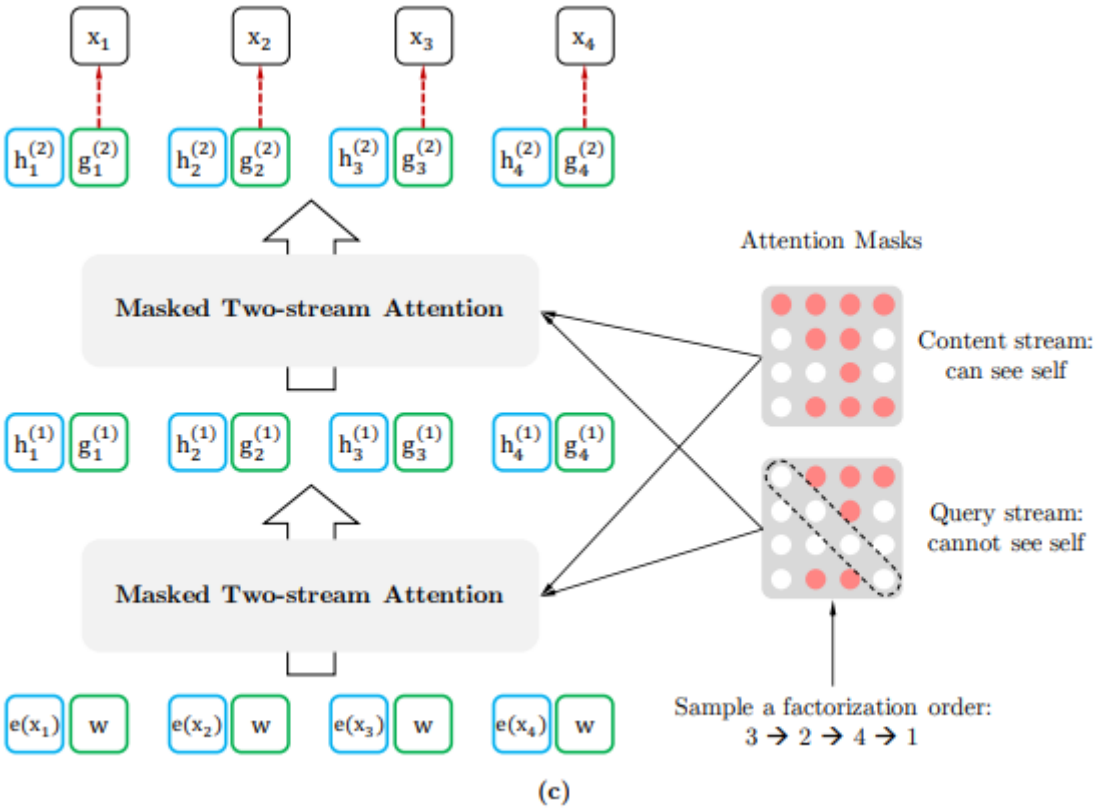
- **Content Stream**



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



- 기존 self-attention구조와 동일한 구조로, 현재시점의 content(word embedding)와 이전 시점의 content를 모두 활용한다.
  - content stream에서는 각 토큰의 임베딩에 positional embedding을 추가한 값으로 초기화한다.
  - 마지막 layer까지, 이전 layer에서 나온 현재 시점의 hidden state와 이전 시점의 hidden state를 모두 활용해 attention을 계산한다.



마지막 layer에서 나온 query representation으로 해당 위치에 어떤 토큰이 오는지 예측한다.

- **Partial Prediction**

  모든 조합의 순서로 maximum likelihood를 계산하기 때문에 느리게 수렴하는 문제를 극복하기 위해 특정 순서에서 마지막 몇개만 예측에 활용하는 방법을 사용했다.

$$p(x_3)p(x_2 \mid x_3)p(x_4 \mid x_2, x_3)p(x_1 \mid x_3, x_2, x_4) \rightarrow p(x_4 \mid x_2, x_3)p(x_1 \mid x_3, x_2, x_4)$$

  하이퍼파라미터 K로 개수를 정해 K=2이면 1/2 토큰만 사용해 예측에 활용한다.

Incorporating Ideas from Transformer-XL

긴 문장을 잘 처리할 수 있도록 Transformer-XL에서 사용된 두가지 방법을 차용했다.

- **Segment Recurrence Mechanism**

  기존에는 길이가 긴 시퀀스의 경우 max len으로 자르고 학습을 수행하는 문제가 있었다. 이런 문제를 해결하고자 긴 문장을 여러 segment로 나눠 학습할 수 있도록 하는 매커니즘이다.

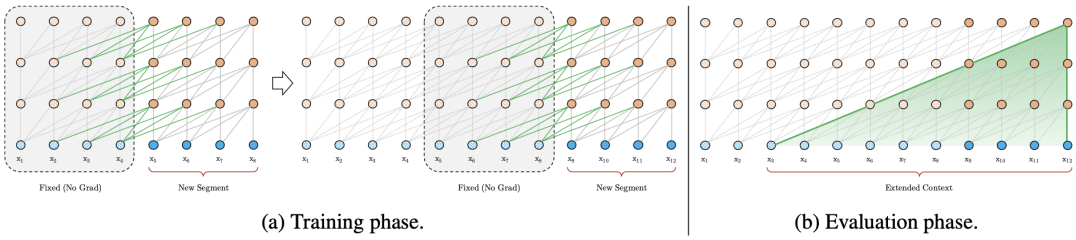


Figure 2: Illustration of the Transformer-XL model with a segment length 4.

각 세그먼트의 hidden state를 memory에 저장해두었다가 다음 세그먼트에서 concat 하여 사용하는 방식으로, 이전 세그먼트에 대해서는 파라미터를 freeze하여 그레디언트가 전파되지 않도록 한다.

- **Relative Positional Encoding**

  현재 세그먼트의 메모리 사이의 positional encoding을 구분하기 위해 K벡터와 Q벡터의 상대적 위치 차이 정보를 위치 인코딩으로 사용한다.

Experiments

- bert: BookCorpus + English wikipedia → 16GB
- xlnet: BookCorpus + English Wiki + Giga5 + ClueWeb + Common Crawl → 19GB , 78GB
- model size는 Bert large와 동일
- 약 500k step을 돌며 학습되었는데, 학습데이터 양에 비해서 많은 학습을 하지 않았고, 저자 역시 모델이 학습데이터에 대해 underfit했음을 밝혔다.
  - 그러나 pretrain을 더 하더라도 downstream task에선 크게 도움이 되지 않았다고 함
  - 모델이 데이터 스케일을 충분이 커버하지 못했음을 그 원인으로 추측

Results

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**

Table 2: Comparison with state-of-the-art results on the test set of RACE, a reading comprehension task, and on ClueWeb09-B, a document ranking task. ∗ indicates using ensembles. † indicates our implementations. "Middle" and "High" in RACE are two subsets representing middle and high school difficulty levels. All BERT, RoBERTa, and XLNet results are obtained with a 24-layer architecture with similar model sizes (aka BERT-Large).

- RACE: 100K개의 중국 중고등학생을 위한 질문과 정답으로 이루어진 QA dataset이다. 평균 passage 길이가 300을 넘기 때문에 긴 문장을 이해할 수 있는지를 판단할 수 있다.

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‡**

Table 3: Results on SQuAD, a reading comprehension dataset. † marks our runs with the official code. ∗ indicates ensembles. ‡: We are not able to obtain the test results of our latest model on SQuAD1.1 from the organizers after submitting our result for more than one month, and thus report the results of an older version for the SQuAD1.1 test set.

- SQuAD: QA Dataset로 역시 비교적 문장 길이가 긴 태스크이다.

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**

Table 4: 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).

- Text Classification task에서 유명한 dataset에 대해 수행한 실험결과이다.

Ablation Study

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
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

Table 6: The results of BERT on RACE are taken from [38]. We run BERT on the other datasets using the official implementation and the same hyperparameter search space as XLNet. K is a hyperparameter to control the optimization difficulty (see Section 2.3).

Bert base 하이퍼파라미터와 동일하게 맞추었고, BookCorpus와 Wikipedia로 학습했다.

- permutation LM을 수행했을 때 성능이 좋아졌다.
- Transformer XL도 BERT보다 성능이 좋아졌다.
- 메모리 캐싱 기법을 빼면 RACE에서 성능저하가 발생한다.
- next-sent pred는 RACE를 제외하곤 오히려 성능이 떨어진다.

Conclusion

- Permutation을 학습하기 위해 새로운 기법을 고안하여 양방향 정보를 Autoregressive 하게 학습한 모델로 AR과 AE의 단점을 개선한 모델이다.