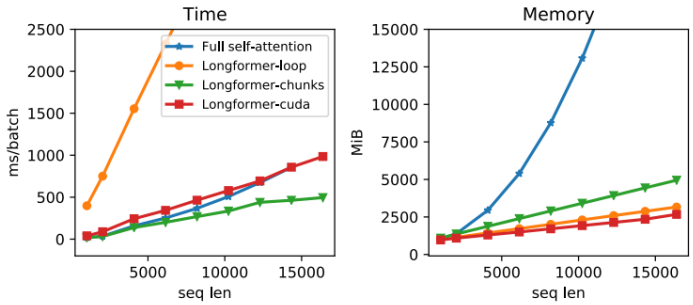
논문 리뷰) Longformer: The long document transformer

〈개요〉

- **Transformer기반 모델들은 긴 시퀀스를 다루지 못한다.**
  - self-attention으로 인해 시퀀스의 길이에 따라 계산량이 quadratic(제곱비례)하게 증가하기 때문.
  - Longformer는 시퀀스의 길이(n)에 따라 계산량이 선형적으로 증가하게 해줌으로써 수 천개 혹은 그 이상의 긴 문서를 쉽게 처리할 수 있도록 하기 위해 만들어졌다.
- 밑의 그래프는 **full self-attention과 Longformer의 실행시간과 메모리를 비교**한 것이다.
  - Longformer-loop
  - Longformer-chunk
  - Longformer-cuda



- Longformer의 attention 메커니즘은 task에 따라 조정하는 global attention과 local window attention을 결합한 것으로, 기존 self-attention을 코드변환없이 바로 대체할 수 있다(drop-in replacement).

> **drop-in replacement**
>
> 컴퓨터공학에서 사용하는 용어로 하드웨어나 소프트웨어의 구성요소를 부정적인 영향을 일으키지 않으면서 코드나 설정 변경없이 다른 요소로 대체할 수 있는 능력을 가리킨다.
>
> https://en.wikipedia.org/wiki/Drop-in_replacement

〈관련연구〉

- 긴 길이의 텍스트를 임베딩하기 위한 self-attention 접근 방법 중 2가지를 비교하였다.
  1. **left-to-right(ltr):** 긴 문서가 입력으로 들어오면 left-to-right방향으로 chunk단위로 쪼개어 임베딩하고, 이렇게 얻어진 임베딩들을 다시 시퀀스로 임베딩하는 방법
     - chunk단위로 쪼개는 과정과 다시 임베딩하는 과정에서 정보 손실이 발생한다.
     - ex) Transformer-XL, Adaptive Span Transformer 등
  2. **sparse attention:** quadratic한 행렬곱 계산을 피하여 긴 문서를 그대로 입력으로 사용하는 방법.
     - 어떻게 효율적으로 sparse한 attention map을 구축할지가 관건이 된다.
     - ex) Longformer, BigBird, Sparse Transformer, Reformer...

Model	attention matrix	char-LM	other tasks	pretrain
Transformer-XL (2019)	ltr	yes	no	no
Adaptive Span (2019)	ltr	yes	no	no
Compressive (2020)	ltr	yes	no	no
Reformer (2020)	sparse	yes	no	no
Sparse (2019)	sparse	yes	no	no
Routing (2020)	sparse	yes	no	no
BP-Transformer (2019)	sparse	yes	MT	no
Blockwise (2019)	sparse	no	QA	yes
Our Longformer	sparse	yes	multiple	yes

Table 1: Summary of prior work on adapting Transformers for long documents. ltr: left-to-right.

많은 task-specific 접근방법들이 BERT와 같은 사전학습 transformer모델의 입력 시퀀스길이의 한계(n=512)를 뛰어넘기 위해 노력하였다.

- 방법1. 분류에 흔히 사용된 방법으로 단순히 문서를 자른다.
- 방법2. 문서를 512길이의 chunk로 쪼개어 각 chunk를 개별적으로 처리한다. (chunk는 서로 겹치게 할 수 있다.)

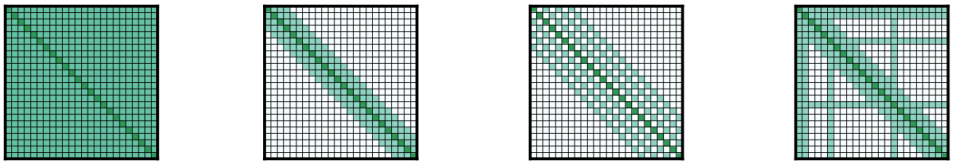
=> 이러한 방법들은 정보손실의 문제가 있지만, Longformer는 긴 시퀀스를 자르거나 chunk로 쪼개는 일 없이 그대로 처리할 수 있다.

이외에도 local attention과 global attention을 결합하여 사용하고, 긴 문서 자연어 task를 위해 사전학습 시키는 Longformer와 유사한 아이디어를 가지고 진행한 연구들이 있다.

- **ETC**는 full attention대신 local+global attention을 사용하였는데, Longformer와 global attention설정이 조금 다르고 상대적 position embedding과 추가적인 목적함수로 CPC loss를 도입하였다.
- **BigBird**는 ETC를 확장한 모델로, 이론적으로 sparse transformer가 시퀀수함수의 universal approximators 라는 것과 full self-attention의 특성을 보존한다는 것을 증명하였다.

<Attention 패턴>

- 가능한 입력 시퀀스의 길이(n)를 늘리기 위해 $O(n^2)$의 계산복잡도를 가진 full self-attention을 희소화 (sparsify)하여 Longformer의 attention 패턴은 선형적으로 확장(O(n))되며 더 긴 입력 시퀀스를 효율적으로 학습시킬 수 있도록 하였다.
- **Longformer의 Attention 패턴**
  - sliding window attention
  - dilated window attention
  - global attention

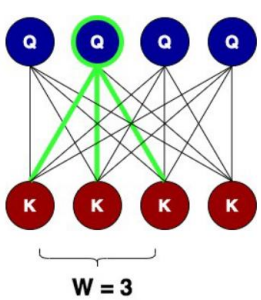


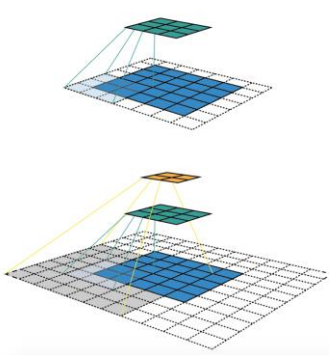
(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

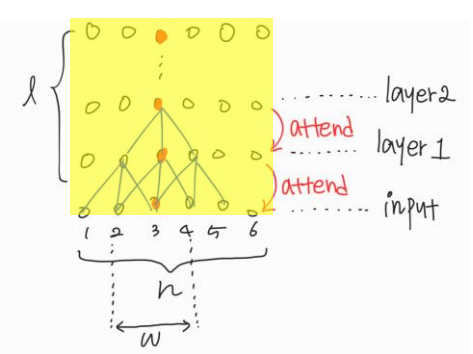
Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

- Sliding Window (1)

- 하나의 토큰이 양 옆으로 w/2만큼만 attention하는 방법이다. (w: window의 크기)
- window attention은 CNN에서 레이어가 쌓임에 따라 각 커널이 더 넓은 receptive field를 가지는 것에서 착안하였다.
  - 첫 번째 레이어의 토큰이 갖는 receptive field는 w이지만, 레이어를 쌓을수록 receptive field가 커져서 레이어를 충분히 쌓으면 최상위층의 노드는 모든 입력 토큰에 접근 가능하며, 전체 입력의 정보를 아우르는 representation을 만들 수 있게 된다.

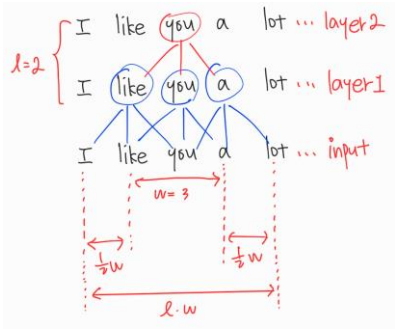






- Sliding Window (2)

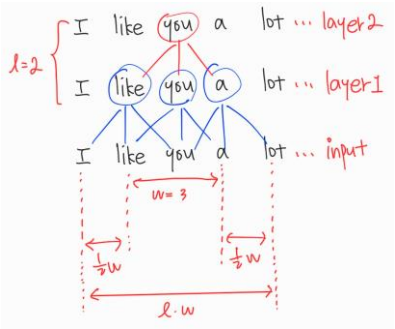
- 하나의 레이어가 쌓일 때마다 양끝에 w/2만큼 receptive field가 넓어진다.



- 계산복잡도: O(n x w)
- receptive field 크기: l x w (l: 레이어 개수)
  - w가 모든 레이어에서 같은 값이라고 가정
- 그러나 실제로는 w값을 각 레이어마다 다르게 설정한다.
  - 효율성과 모델의 표현능력 사이의 균형을 조정

- Sliding Window의 한계

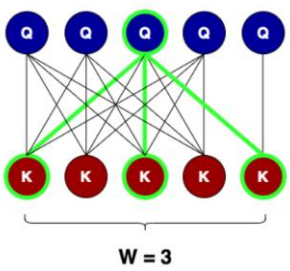
- window 크기가 너무 작거나, 시퀀스 길이가 너무 길면
  전체 토큰을 아우르는 receptive field를 만들기 위해 쌓아야 하는 레이어 수가 너무 많아진다.
- 이는 연산량의 증가를 야기한다.

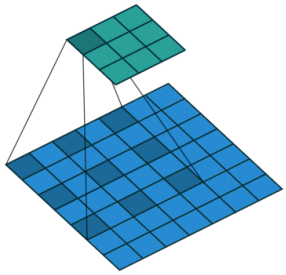


- Dilated Sliding Window

- 기본적인 sliding window와 같은 window 크기를 사용하여 연산량은 그대로 유지하면서, window의 각 칸마다 d만큼의 masking을 두어서 더 넓은 receptive field를 가지도록 하는 방법이다. (d: dilation 크기)
- dilated CNN에서 착안된 아이디어이다.

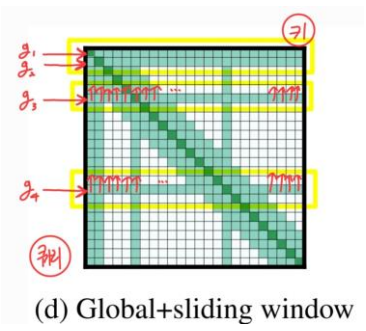
- 계산복잡도: O(n x w x d)
- receptive field 크기: l x d x w
  - 모든 레이어가 동일한 d와 w값을 가진다고 가정



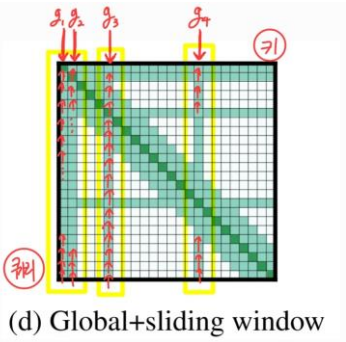


- Global Attention

- global 토큰을 도입한 방법이다.
  - global 토큰은 같은 레이어에 있는 모든 토큰들을 attend하고, 다른 모든 토큰들은 global 토큰을 attend한다.
  - global 토큰은 직접 선택한다.
- global 토큰은 시퀀스 길이n에 비해 상대적으로 적기 때문에 두 attention을 결합해도 여전히 계산복잡도는 O(n)이다.



(d) Global+sliding window



(d) Global+sliding window

<실험>

1) Attention 패턴

- 레이어마다 다른 window 크기를 사용하였다.
- window크기는 하위 레이어일수록 작게, 상위 레이어일수록 크게 설정했다.
  - 이는 하위 레이어는 local 정보를 모으고 상위 레이어는 전체 시퀀스에 대한 representation을 학습하도록 한다.
  - 또한 window크기가 작을수록 nonzero값이 적으므로 계산량이 적고, window크기가 클수록 풍부한 representation 능력을 가지므로 효율성과 성능사이의 균형을 맞추는 역할을 한다.
- 하위 레이어에서는 즉시 local context를 학습하고 활용할 수 있도록 dilated sliding window를 사용하지 않았다.

2) 실험 설정

- 이전 연구들과 비교를 위해 text8,enwik8데이터셋에 대한 character-level 언어모델에 주목하였다.

<Training>

- GPU 메모리에 맞춰서 최대한의 window size와 시퀀스 길이로 모델을 훈련시키는 것이 이상적이겠지만 모델이 보다 긴 context를 학습하기 이전에 우선 local context 학습을 필요로 한다는 것을 알게 되었다.
- 따라서 우리는 단계적 학습 절차(staged training procedure)를 채택하였다.
  - attention window size와 시퀀스 길이를 phase를 거치며 점점 증가시킨다.
- 총 5개 phase로 모델을 학습시켰다.
  - 시퀀스길이: 2048로 시작하여 마지막 phase에서는 23040
- 두가지 규모의 모델을 사용했다.
  - small: 12 layers, 512 hidden size
  - large: 30 layers, 512 hidden size

<Evaluation>

- 32256시퀀스로 평가함.
- 겹쳐지도록 쪼갬. 그리고 마지막 512토큰에 대한 결과를 보고하도록 함.

- GPU 메모리에 맞춰서 최대한의 window size와 시퀀스 길이로 모델을 훈련시키는 것이 이상적이겠지만 모델이 보다 긴 context를 학습하기 이전에 우선 local context 학습을 필요로 한다는 것을 알게 되었다.
- 따라서 우리는 단계적 학습 절차(staged training procedure)를 채택하였다.
  - attention window size와 시퀀스 길이를 phase를 거치며 점점 증가시킨다.
- 총 5개 phase로 모델을 학습시켰다.
  - 시퀀스길이: 2048로 시작하여 마지막 phase에서는 23040
- 두가지 규모의 모델을 사용했다.
  - small: 12 layers, 512 hidden size
  - large: 30 layers, 512 hidden size

Param	Value
Position Embeddings	Relative and Sinusoidal as in Dai et al. (2019)
Small model config	12 layers, 8 heads, 512 hidden size as in Dai et al. (2019)
Large model config	30 layers, 8 heads, 512 hidden size as in Child et al. (2019)
Optimizer	AdamW
Dropout	0.2 (small model), 0.4 (large model)
Gradient clipping	0.25
Weight Decay	0.01
Layernorm Location	pre-layernorm (Xiong et al., 2020)
Activation	GeLU
Number of phases	5
Phase 1 window sizes	32 (bottom layer) - 8,192 (top layer)
Phase 5 window sizes	512 (bottom layer) - (top layer)
Phase 1 sequence length	2,048
Phase 5 sequence length	23,040 (gpu memory limit)
Phase 1 LR	0.00025
Phase 5 LR	000015625
Batch size per phase	32, 32, 16, 16, 16
#Steps per phase (small)	430K, 50k, 50k, 35k, 5k
#Steps per phase (large)	350K, 25k, 10k, 5k, 5k
Warmup	10% of the phase steps with maximum 10K steps
LR scheduler	constant throughout each phase
Dilation (small model)	0 (layers 0-5), 1 (layers 6-7), 2 (layers 8-9), 3 (layers 10-11)
Dilation (large model)	0 (layers 0-14), 1 (layers 15-19), 2 (layers 20-24), 3 (layers 25-29)
Dilation heads	2 heads only

Table 12: Hyperparameters for the best performing model for character-level language modeling

3) 결과

- BPC (bits-per-character): character단위의 perplexity
- Table 2에서 small model로 SOTA에 달성함으로써 Longformer의 효율성을 증명하였다.
- Table 3에서 Longformer가 Transformer-XL모델보다 좋은 성능을 보임.
- 약소하게 Longformer보다 성능이 좋은 모델들이 있으나 관련 연구 파트에서 언급했듯이 pretraining하고 finetuning하기에 적합하지 않음.

Model	#Param	Dev	Test
Dataset `text8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.18
Adaptive (Sukhbaatar et al., 2019)	38M	1.05	1.11
BP-Transformer (Ye et al., 2019)	39M	-	1.11
Our Longformer	41M	1.04	**1.10**
Dataset `enwik8`			
T12 (Al-Rfou et al., 2018)	44M	-	1.11
Transformer-XL (Dai et al., 2019)	41M	-	1.06
Reformer (Kitaev et al., 2020)	-	-	1.05
Adaptive (Sukhbaatar et al., 2019)	39M	1.04	1.02
BP-Transformer (Ye et al., 2019)	38M	-	1.02
Our Longformer	41M	1.02	**1.00**

Table 2: *Small* model BPC on `text8` & `enwik8`

Model	#Param	Test BPC
Transformer-XL (18 layers)	88M	1.03
Sparse (Child et al., 2019)	≈100M	0.99
Transformer-XL (24 layers)	277M	0.99
Adaptive (Sukhbaatar et al., 2019)	209M	0.98
Compressive (Rae et al., 2020)	277M	0.97
Routing (Roy et al., 2020)	≈223M	0.99
Our Longformer	102M	0.99

Table 3: Performance of *large* models on `enwik8`

4) Ablation Study

- attention 패턴 설계 요소들의 중요성을 보여주는 파트이다.

1. 레이어별 window크기 설정
   - 상위 레이어로 갈수록 window를 증가시키는 방법이 제일 성능이 좋았다.
2. dilation 여부
   - dilation을 2개 head에 추가했을 때가 사용하지 않을 때보다 성능이 개선되었다.

Model	Dev BPC
Decreasing w (from 512 to 32)	1.24
Fixed w (= 230)	1.23
Increasing w (from 32 to 512)	**1.21**
No Dilation	1.21
Dilation on 2 heads	**1.20**

Table 4: Top: changing window size across layers. Bottom: with/without dilation (@ 150K steps on phase1)

5) Pretraining and Finetuning

- 최근 많은 NLP에서 SOTA를 달성하는 기술들은 BERT와 같이 사전학습된 모델을 fine-tuning한다.
- 본 논문의 주목적 또한 긴 문서 task들에 적합한 모델을 만드는 것이므로
  Longformer를 문서 말뭉치로 사전학습시킨 후, 6개 task에 대하여 fine-tuning하였다.
  - classification, QA, coreference solution
- 결과적으로 모델은 4096개의 긴 토큰을 처리할 수 있게 되었다. (BERT의 8배)

- 시퀀스에서 랜덤하게 마스킹된 토큰을 예측하는 것을 목표로 하는 MLM(masked language modeling)으로 사전학습을 한다.
- MLM은 비용이 많이 들기 때문에 RoBERTa checkpoint를 가져와서
  Longformer의 attention 메커니즘을 지원하기 위해 필요한 최소한의 변경 후 사전학습시켰다.
  - 초반에 언급했듯이 Longformer의 attention 패턴을 사전학습된 transformer모델에 적용하기 위해 모델 구조를 변경할 필요가 없다는 점에 주목하자.

- 베이스라인: RoBERTa
- Attention Pattern: window 크기로 512를 사용하여 RoBERTa와 같은 연산량을 수행한다.
- Position Embedding: RoBERTa는 최대 512까지 absolute position embedding을 사용하는데, 더 긴 텍스트를 지원하기 위해 최대 4096까지 position embedding을 추가하였다.
  - 단, 단순히 랜덤하게 초기화하는 것이 아니라 학습된 512개의 position embedding을 복사하여 초기화한다. 간단한 방법이지만 매우 효율적이라는 것을 확인하였다.(Tab 5)

Model	base	large
RoBERTa (seqlen: 512)	1.846	1.496
Longformer (seqlen: 4,096)	10.299	8.738
+ copy position embeddings	1.957	1.597
+ 2K gradient updates	1.753	1.414
+ 65K gradient updates	1.705	1.358
Longformer (train extra pos. embed. only)	1.850	1.504

Table 5: MLM BPC for RoBERTa and various pretrained Longformer configurations.

- 베이스라인: RoBERTa
- Continued MLM Pretraining: base 모델과 large 모델 모두 4096길이의 시퀀스를
  배치 크기 64, 65K번의 gradient 업데이트 등으로 하이퍼파라미터를 조정하여 학습시켰다.
  그리고 Table 5를 통해 더 지속적으로 사전학습 시키는 것이 성능개선에 영향을 준다는 것을 확인할 수 있다.
- Frozen RoBERTa Weights: Longformer를 사전학습할 때 RoBERTa의 가중치를 모두 고정시키고, 새로운 position embedding만 학습시킨다. 이러한 설정은 RoBERTa의 short document에 대한 성능을 그대로 가져오기 위함이다.

Model	base	large
RoBERTa (seqlen: 512)	1.846	1.496
Longformer (seqlen: 4,096)	10.299	8.738
+ copy position embeddings	1.957	1.597
+ 2K gradient updates	1.753	1.414
+ 65K gradient updates	1.705	1.358
Longformer (train extra pos. embed. only)	1.850	1.504

Table 5: MLM BPC for RoBERTa and various pretrained Longformer configurations.

6) Tasks

- Question answering
- Coreference Resolution
- Document Classification

- Results
  - Longformer가 지속적으로 베이스라인인 RoBERTa보다 좋은 성능을 보인다.

	QA			Coref.	Classification	
Model	WikiHop	TriviaQA	HotpotQA	OntoNotes	IMDB	Hyperpartisan
RoBERTa-base	72.4	74.3	63.5	78.4	95.3	87.4
Longformer-base	75.0	75.2	64.4	78.6	95.7	94.8

Table 7: Summary of finetuning results on QA, coreference resolution, and document classification. Results are on the development sets comparing our Longformer-base with RoBERTa-base. TriviaQA, Hyperpartisan metrics are F1, WikiHop and IMDB use accuracy, HotpotQA is joint F1, OntoNotes is average F1.

7) Longformer-encoder-decoder (LED)

- encoder만 사용하는 Transformer모델들이 다양한 NLP task에서 효율성을 보인 반면에,
  사전 학습된 encoder-decoder 구조인 Transformer 모델들(BART, T5)은 요약에 강점을 보였다.
  그러나 아직 긴 입력을 사용하는 seq2seq task에서는 효율적으로 scaling하지 못한다.
- 따라서 Longformer의 variant(변형)으로 Transformer의 encoder와 decoder를 갖지만
  full self-attention대신 local+global attention 패턴을 사용하는 Longformer-encoder-decoder(LED)를 제안한다.

- LED를 사전학습시키는 것은 비용이 많이 들기때문에 BART의 구조(레이어 수, hidden size)를 가져와서 파라미터를 초기화 하였다. 추가적으로 사전학습은 시키지 않았다.
- position embedding은 BART의 1K position embedding을 16번 반복하여 복사함으로써
  16K 토큰의 position embedding으로 확장하였다.
- encoder는 local attention(w=1024)과 첫번째 토큰을 global 토큰으로 하는 global attention을 사용하고,
  decoder는 full attention을 사용한다.
  - encoder는 긴 문서를 읽고, decoder는 출력할 요약문을 생성한다.

	R-1	R-2	R-L
Discourse-aware (2018)	35.80	11.05	31.80
Extr-Abst-TLM (2020)	41.62	14.69	38.03
Dancer (2020)	42.70	16.54	38.44
Pegasus (2020)	44.21	16.95	38.83
LED-large (seqlen: 4,096) (ours)	44.40	17.94	39.76
BigBird (seqlen: 4,096) (2020)	**46.63**	19.02	41.77
LED-large (seqlen: 16,384) (ours)	**46.63**	**19.62**	**41.83**

Table 11: Summarization results of Longformer-Encoder-Decoder (LED) on the arXiv dataset. Metrics from left to right are ROUGE-1, ROUGE-2 and ROUGE-L.

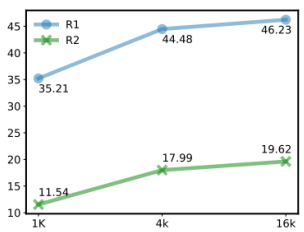


Figure 3: ROUGE-1 and ROUGE-2 of LED when varying the input size (arXiv validation set).

<결론 및 향후 연구방향>

- 긴 문서를 처리하기 위해 긴 입력을 chunk로 나누거나 줄이지 않고
  모든 정보를 결합하여 복잡한 구조없이 넓은 범위의 document-level NLP task를 쉽게 수행하는
  Transformer기반 모델인 Longformer를 선보였다.
- Longformer는 local attention과 global attention을 결합함으로써
  시퀀스 길이에 따라 선형적으로 확장하는(O(n)) attention 패턴을 사용한다.
- Longformer는 character-level 언어모델 task인 text8, enwik8에서 SOTA를 달성했고,
  사전학습된 Longformer는 긴 문서 task들에서 RoBERTa보다 지속적으로 좋은 성능이 나왔다.
- LED(Longformer-encoder-decoder)를 제안하여 arXiv 긴 문서 요약 task에서 SOTA를 달성했다.

- LED를 사전학습하기 위한 연구와, 본 논문에서 제안한 모델로 좋은 성능을 낼 수 있는 다른 task를 더 탐색한다.

[Reference]

- https://arxiv.org/pdf/2004.05150.pdf
- https://en.wikipedia.org/wiki/Drop-in_replacement
- https://ko.wikipedia.org/wiki/%EB%9D%A0%ED%96%89%EB%A0%AC
- https://towardsdatascience.com/review-dilated-convolution-semantic-segmentation-9d5a5bd768f5
- https://www.youtube.com/watch?v=i7aiBMDExmA&t=1399s
- https://www.youtube.com/watch?v=_8KNb5iqblE