



DSBA Transformer survey paper study

A Survey of Transformers

#1: Attention – Sparse Attention

arXiv preprint



고려대학교 산업경영공학과

Data Science & Business Analytics Lab

이유경, 김명섭, 윤훈상, 김지나, 허재혁, 김수빈

발표자 : 윤훈상

01 Taxonomy of Transformers

02 Attention

03 Sparse Attention

3-1) Position-based Sparse Attention

3-1-1) Atomic Sparse Attention

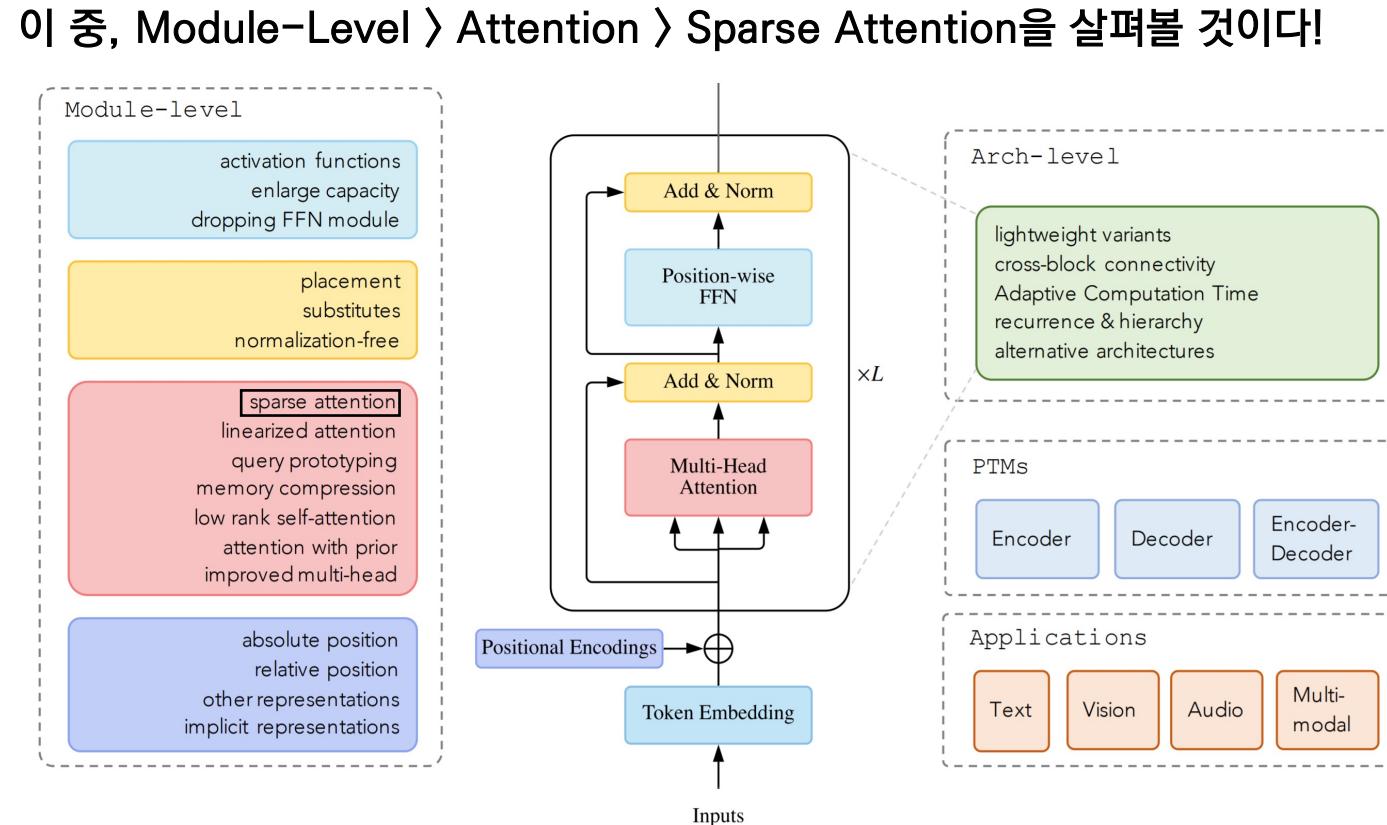
3-1-2) Compound Sparse Attention

3-2) Content-based Sparse Attention

Vanilla Transformer에서 수많은 Variant들이 파생

- Module-Level
- Architecture-Level
- Pretraining-Methods
- Applications

★ 색깔별 Matching



이 중, Module-Level > Attention > Sparse Attention을 살펴볼 것이다!

Attention은 매우 중요하지만, 문제가 있다.

Self-Attention은 Transformer의 중요한 역할을 하지만, 두 가지 문제 발생

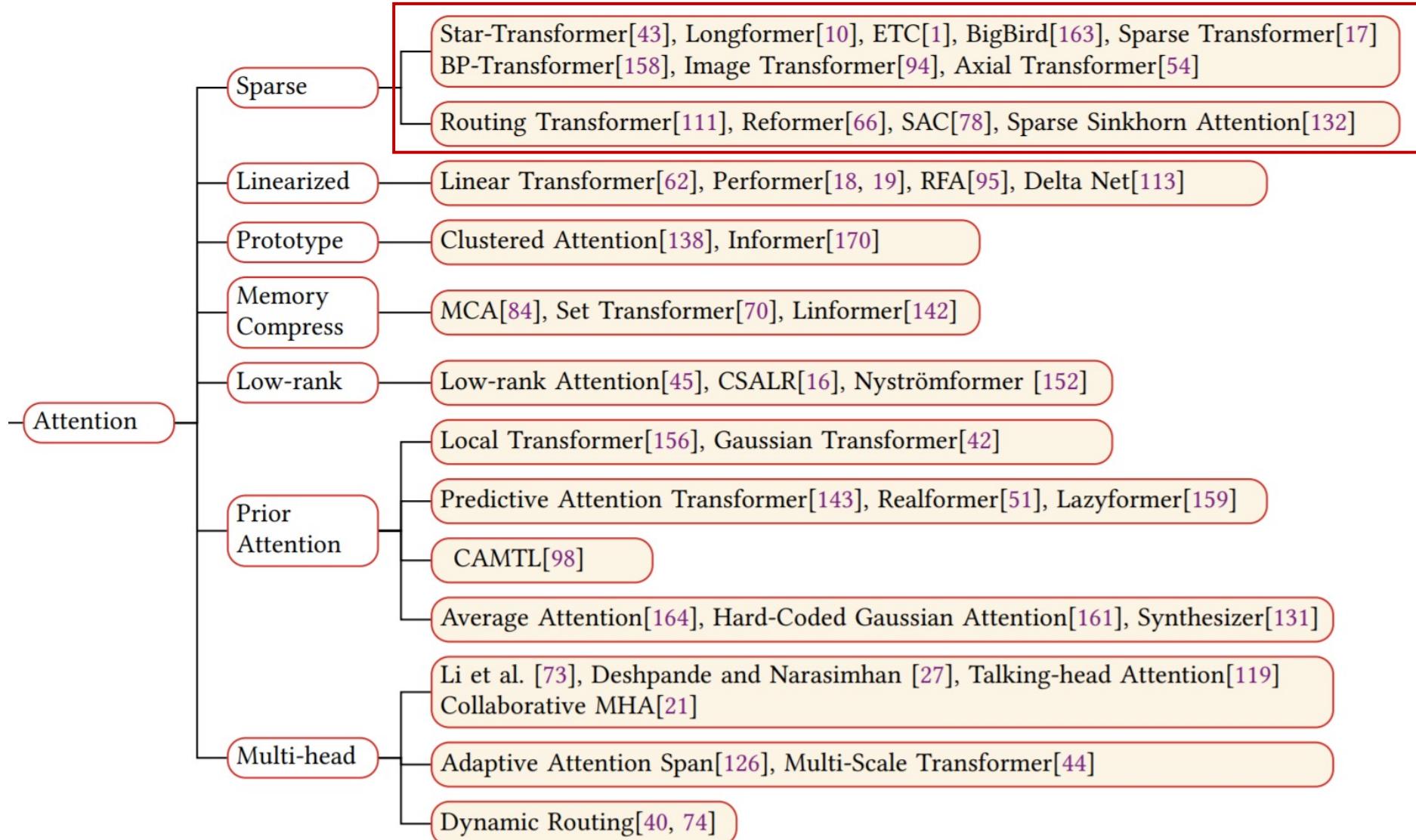
- Complexity: Self Attention의 Complexity ($O(T^2 \cdot D)$)는 문장이 길어지면 Bottleneck이 된다.
- Structural Prior: Input에 대한 어떤 Structural Bias도 없다 (장점인 동시에 단점)

이를 해소하기 위한 Attention Mechanism의 발전은 다음과 같다.

- Sparse Attention
Attention에 Sparsity Bias(Structural Bias)를 추가하여 Complexity 감소
- Linearized Attention
- Prototype and Memory Compression
- Low-rank Self-Attention
- Attention with Prior
- Improved Multi-Head Mechanism

Attention

Attention은 매우 중요하지만, 문제가 있다.



Sparse Attention

Overview

Sparse Attention (2021.07 기준)

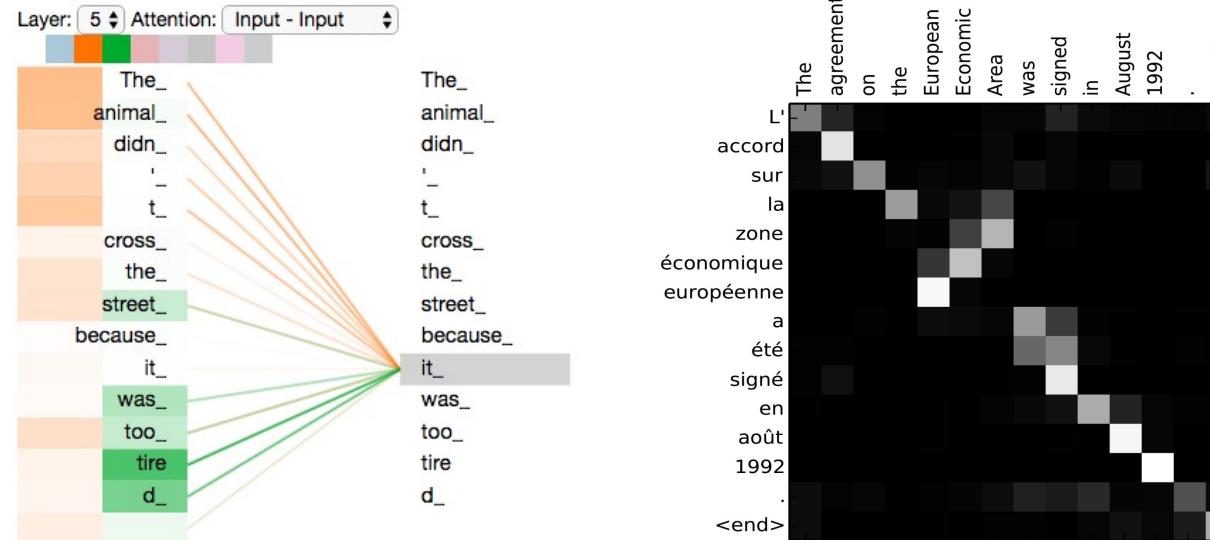
- Position Based Sparse Attention
 - Atomic Sparse Attention
 - Compound Sparse Attention (여러 Atomic Sparse Attention이 모여 구성)
 - Star Transformer (NAACL, 2019, 87회 인용)
 - Longformer (2020, 258회 인용 / AllenAI)
 - ETC (EMNLP, 2020, 24회 인용 / Google Research)
 - BigBird (NeurIPS, 2020, 139회 인용 / Google Research)
 - Sparse Transformer (2019, 353회 인용 / OpenAI)
 - Extended Sparse Attention (Atomic 혼합 대신 새로운 관점 또는 Non-Text Dataset)
 - BP-Transformer (2019, 24회 인용 / AWS AI Lab)
 - Image Transformer (ICML, 2018, 448회 인용)
 - Axial Transformer (2019, 59회 인용 / Google Brain)
 - Content-Based Sparse Attention (Query와 Key Token의 유사도를 통한 Attention)
 - Reformer (ICLR, 2020, 360회 인용 / Google Research)
 - Routing Transformer (TACL, 2020, 66회 인용 / Google Research)
 - Sparse Adaptive Connection, Sparse Sinkhorn Attention

Sparse Attention

Vanilla Transformer Attention도 사실상 Sparse하다.

Vanilla Transformer Attention Matrix는 사실상 Sparse하다?

- Attention은 훈련 뒤에는 볼 대상만 보게 된다.
- 그럼 애초에 Structural Bias를 부여하여, Query-Key Pair 개수를 제한하자!



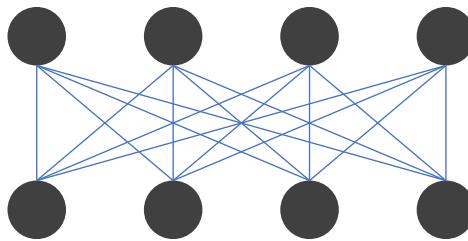
$$\hat{A}_{ij} = \begin{cases} q_i k_j^\top & \text{if token } i \text{ attends to token } j, \\ -\infty & \text{if token } i \text{ does not attend to token } j, \end{cases}$$

Attend하지 않는 대상은 Memory에 담지도 않는다.

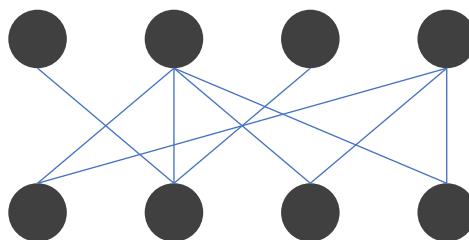
Vanilla Transformer Attention \Rightarrow Complete Bipartite Graph

Sparse Attention \Rightarrow Sparse Graph

Complete Bipartite Graph

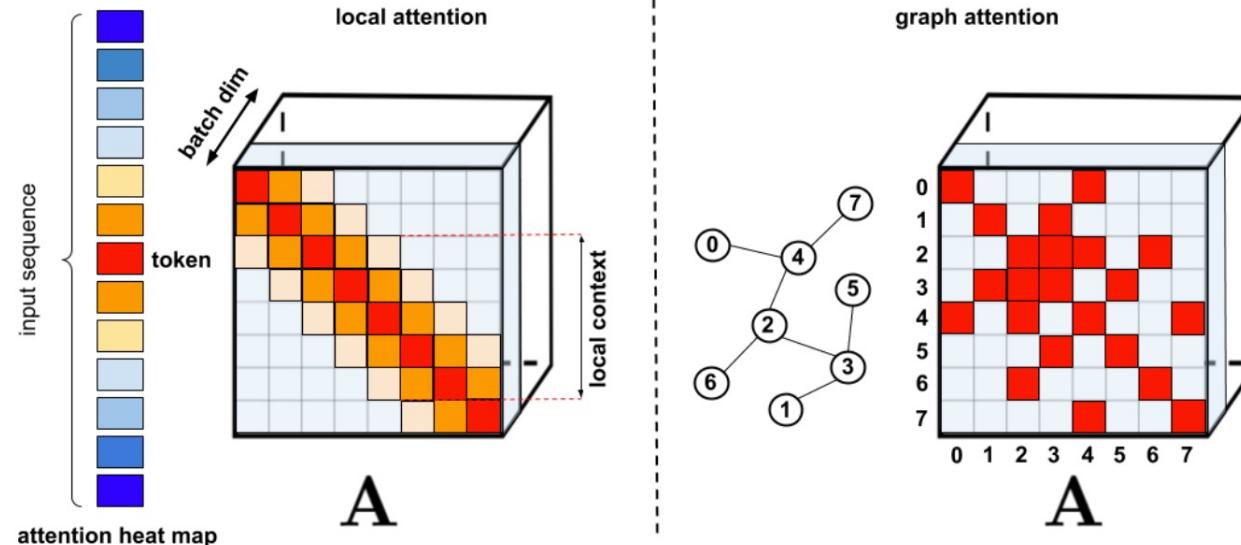


Sparse Graph



Sparse Attention \approx Graph Neural Network

Graph Sparsification



Standard sparsification techniques. Left: Example of a sparsity pattern, where tokens attend only to other nearby tokens. Right: In Graph Attention Networks, tokens attend only to their neighbors in the graph, which should have higher relevance than other nodes. See [Efficient Transformers: A Survey](#) for a comprehensive categorization of various methods.

Sparse Attention의 공통 주장

- Computational Cost: 문장이 길어질수록, Attention에 필요한 비용이 크다.
Input Length에 Quadratic하게 증가하는 비용 대신에, Linear하게 증가하도록 만드는 것이 목적
$$O(n^2) \rightarrow O(n)$$
- Long Range Dependency: 긴 문장에는 Transformer를 적용하기 힘들다 (+Cost 문제)
Global Attention을 통해 Long Range Dependency 해소

Sparse Attention 효과

- 비용 / Dependency를 감소 시키면 512 Token보다 더 긴 문장을 Input으로 활용할 수 있다.
Input 문장을 길게 만들면, Downstream Task에 사용되는 단서가 많아진다.
Downstream Task로 QA & 문서 요약이 많이 사용 (단서 증폭 효과를 살피기 위하여)
- Sparse Attention은 Long Input이 사용되는 Encoder에 적용된다.

Position-based Sparse Attention

Atomic Sparse Attention

Atomic Sparse Attention의 종류

- **Global Attention**

Attention의 Long Range Dependency를 완화하기 위해, Global Node를 설정하여 모든 노드를 Attend

- **Band Attention (Sliding Window Attention / Local Attention)**

언어의 Locality를 반영하기 위하여, 근접한 이웃 노드들에 대해서만 Attend

- **Dilated Attention**

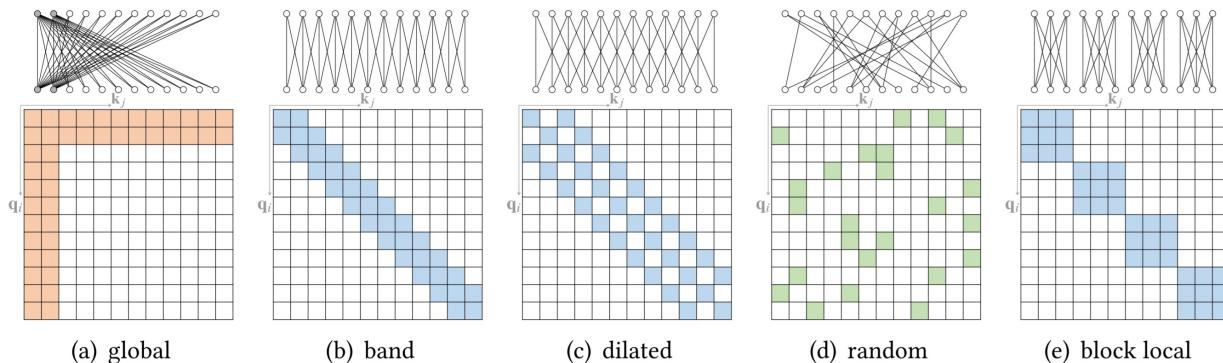
Dilated CNN과 같이 Computation은 같지만 더 넓은 Receptive Field를 위해 한 칸씩 띄운 Band Attention

- **Random Attention**

인접하지 않은 단어들에 대한 상호작용을 반영하기 위하여 각 Query에서 랜덤으로 Attend

- **Block Local Attention**

문장을 여러 non-overlapping query block으로 나누고 해당 Block내에서만 Attention 적용



(a) global

(b) band

(c) dilated

(d) random

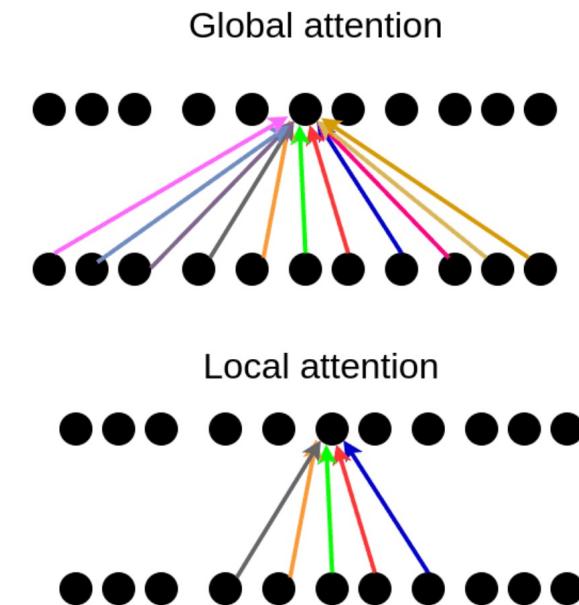
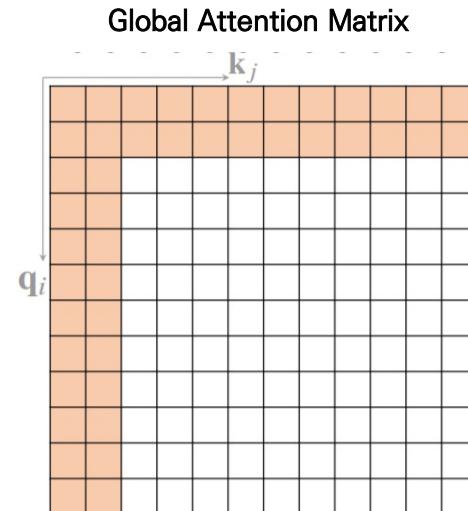
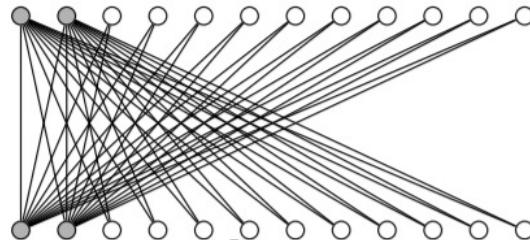
(e) block local

Position-based Sparse Attention

Atomic Sparse Attention

- Global Attention

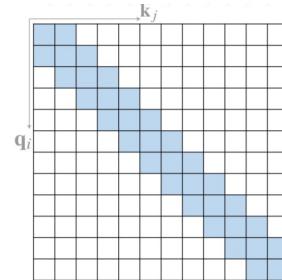
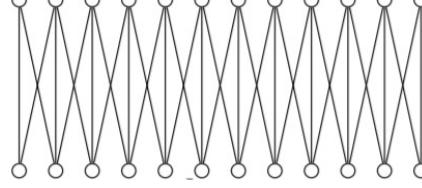
Attention의 Long Range Dependency를 완화하기 위해, Global Node를 설정하여 모든 노드를 Attend
모든 노드를 *Attend*한다는 의미와 다르다.



Position-based Sparse Attention

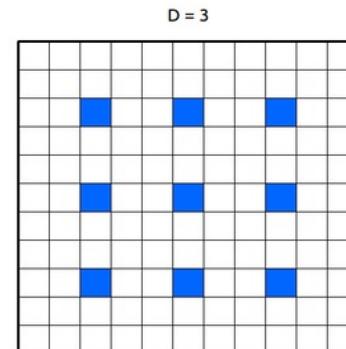
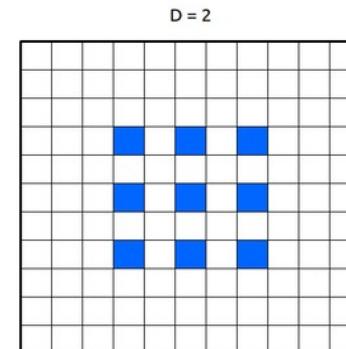
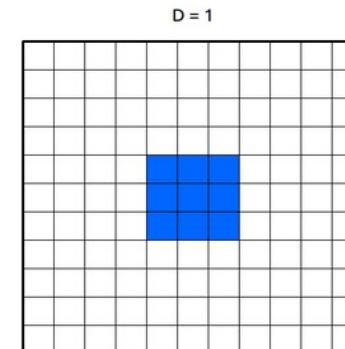
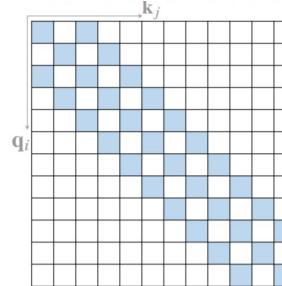
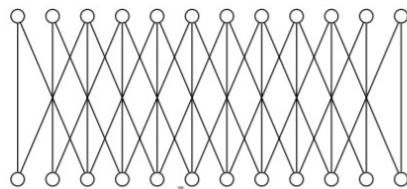
Atomic Sparse Attention

- Band Attention (Sliding Window Attention / Local Attention)
언어의 Locality를 반영하기 위하여, 근접한 이웃 노드들에 대해서만 Attend



- Dilated Attention

Dilated CNN과 같이 Computation은 같지만 더 넓은 Receptive Field를 위해 한 칸씩 띄운 Band Attention

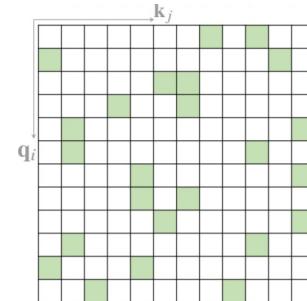
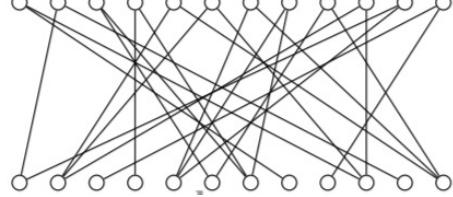


Position-based Sparse Attention

Atomic Sparse Attention

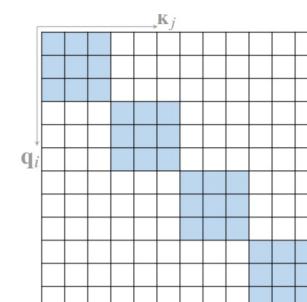
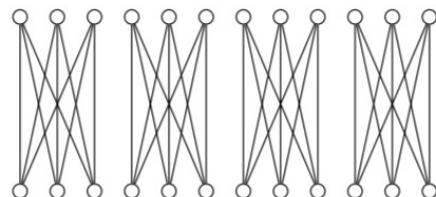
- Random Attention

인접하지 않은 단어들에 대한 상호작용을 반영하기 위하여 각 Query에서 랜덤으로 Attend



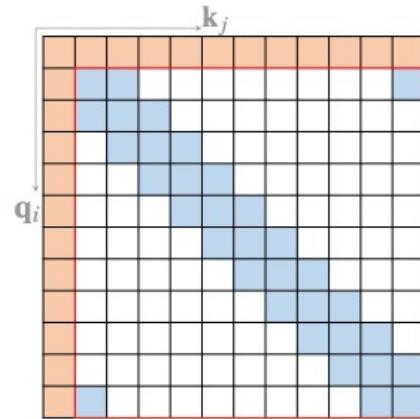
- Block Local Attention

문장을 여러 non-overlapping query block으로 나누고 해당 Block내에서만 Attention 적용

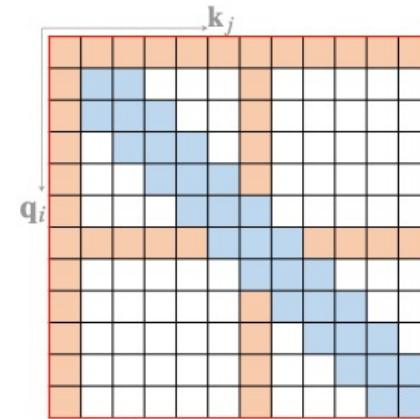


Compound Sparse Attention

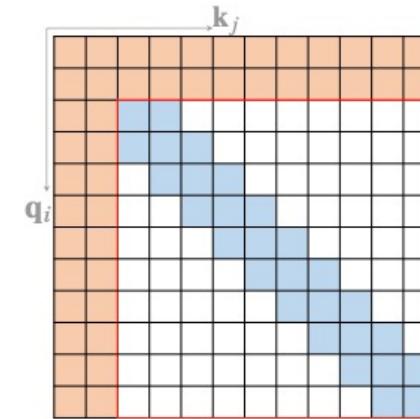
Atomic Sparse Attention의 Combination



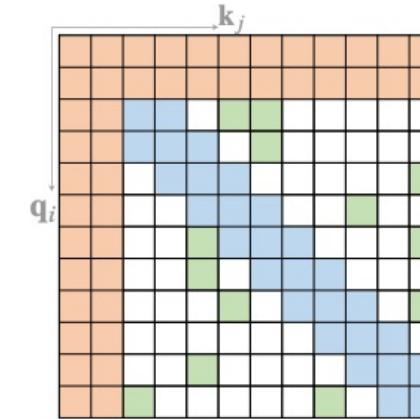
(a) Star-Transformer



(b) Longformer



(c) ETC



(d) BigBird

Star-Transformer: Band Attention + Global-Node Attention

Longformer: Band Attention + Dilated Attention + (Internal) Global-Node Attention

Extended Transformer Construction: Band Attention + (External) Global-Node Attention

BigBird: Band Attention + Global-Node Attention + Random Attention

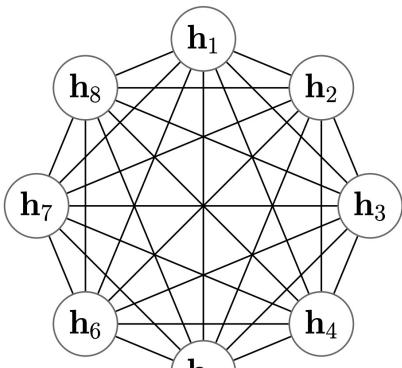


Star Transformer

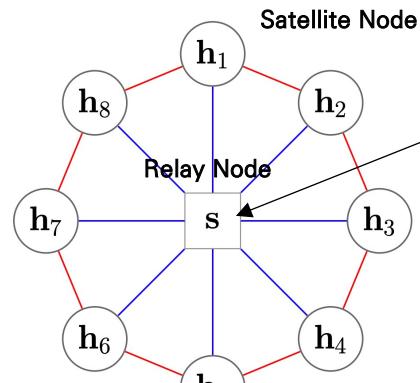
Band Attention + Global-Node Attention

Text Inputs

$$\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \dots, \mathbf{h}_8$$

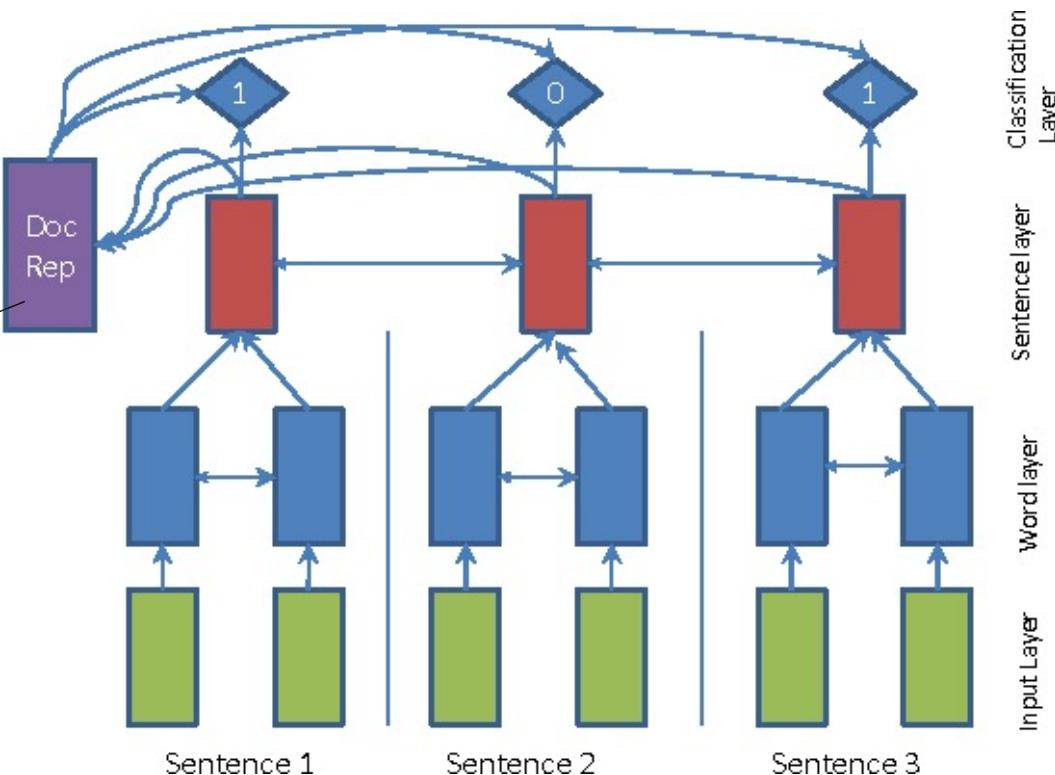


Original Transformer



Star Transformer

SummaRuNNer

Global Attention > Radical Connections: Relay Node – Satellite Node \approx Document Representation

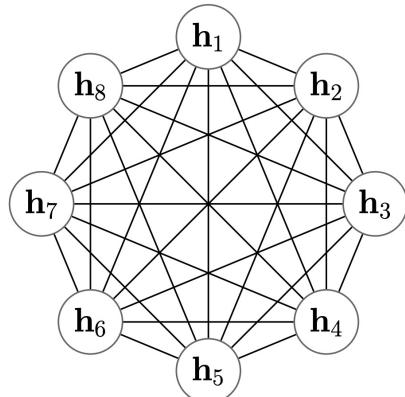
Band Attention > Ring Connections: Satellite Node – Satellite Node



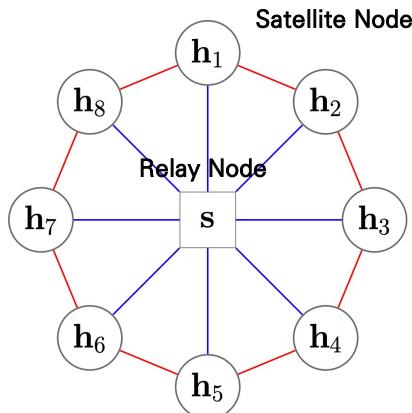
Star Transformer

Band + Global Attention

Text Inputs

 $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3, \dots, \mathbf{h}_8$ 

Original Transformer



Star Transformer

Radical Connections: Relay Node – Satellite Node

Ring Connections: Satellite Node – Satellite Node

절차

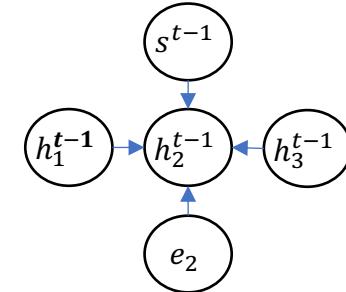
1. Satellite Node의 Update

$$\mathcal{C}_i^t = [h_{i-1}^{t-1}; h_i^{t-1}; h_{i+1}^{t-1}; e^i; s^{t-1}]$$

C_i^t에 포함된 요소들로만 Attention 구성

$$h_i^t = \text{MultiAtt}(h_i^{t-1}, \mathcal{C}_i^t)$$

$$h_i^t = \text{LayerNorm}(\text{ReLU}(h_i^t))$$



2. Relay Node의 Update

Update된 Satellite Node (H^t)들에 대한 Multi-head Att.

$$s^t = \text{MultiAtt}(s^{t-1}, [s^{t-1}; H^t])$$

$$s^t = \text{LayerNorm}(\text{ReLU}(s^t))$$

Implications

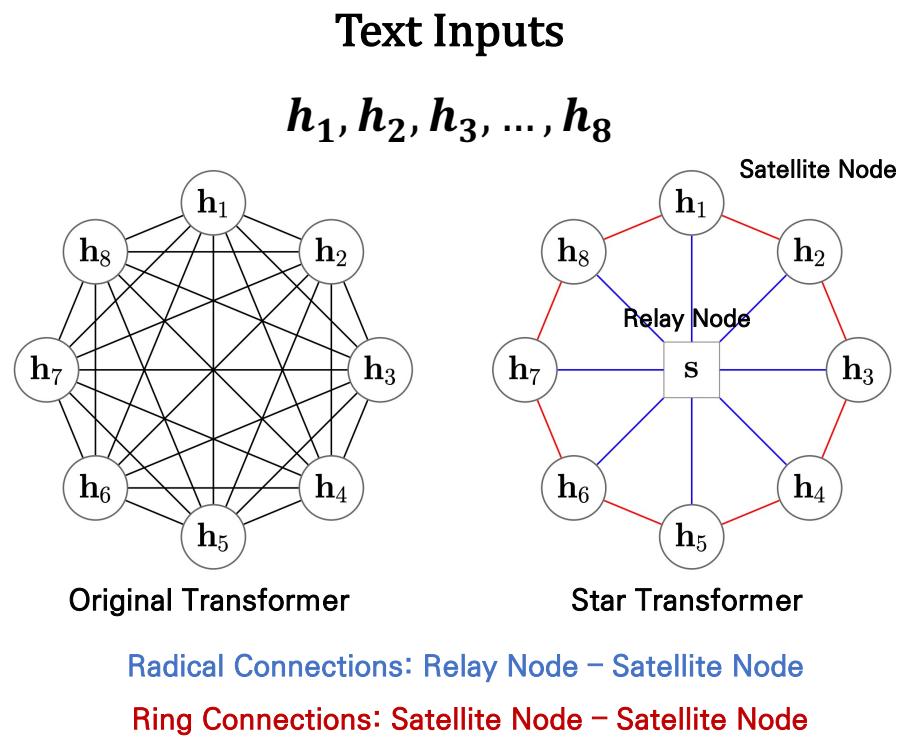
⇒ Band Attention과 Global Attention이 상호작용한다.

⇒ Original Transformer Complexity: $O(n^2d)$ Star Transformer (ring conn): $O(5nd)$ Star Transformer (radical conn): $O(nd)$



Star Transformer

Band + Global Attention



절차

1. Satellite Node의 Update

$$\mathcal{C}_i^t = [h_{i-1}^{t-1}; h_i^{t-1}; h_{i+1}^{t-1}; e^i; s^{t-1}]$$

\mathcal{C}_i^t 에 포함된 요소들로만 Attention 구성

$$h_i^t = \text{MultiAtt}(h_i^{t-1}, \mathcal{C}_i^t)$$

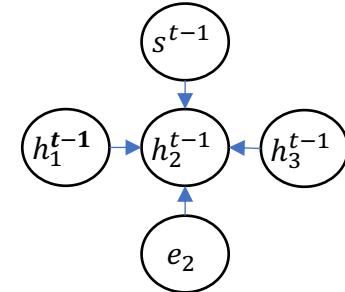
$$h_i^t = \text{LayerNorm}(\text{ReLU}(h_i^t))$$

2. Relay Node의 Update

Update된 Satellite Node (H^t)들에 대한 Multi-head Att.

$$s^t = \text{MultiAtt}(s^{t-1}, [s^{t-1}; H^t])$$

$$s^t = \text{LayerNorm}(\text{ReLU}(s^t))$$



Implications

⇒ Band Attention과 Global Attention이 상호작용한다.

⇒ Original Transformer Complexity: $O(n^2d)$

Star Transformer (ring conn): $O(5nd)$

Star Transformer (radical conn): $O(nd)$

⇒ Relay Node를 활용한 분류 Task 진행

Compound Sparse Attention

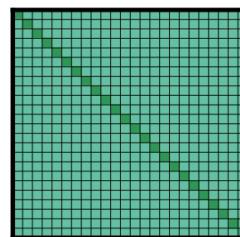
Longformer (2020, 258회 인용 / AllenAI)

Longformer

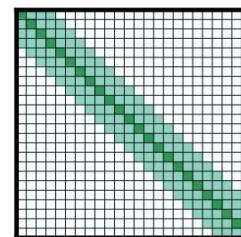
Band Attention + Dilated Attention + (Internal) Global-Node Attention

Long Sequence를 다루기 위한 Transformer (512 token 제한 벗어나고자 함)

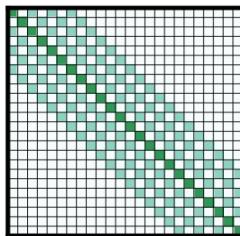
- LongFormer Attention을 통해 Self-Attention 보완
- Transformer의 Encoder 성능 향상 용이 아닌, Seq2Seq 성능 향상 용도



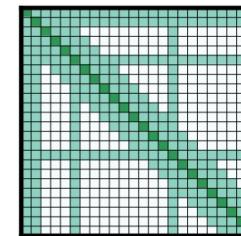
(a) Full n^2 attention



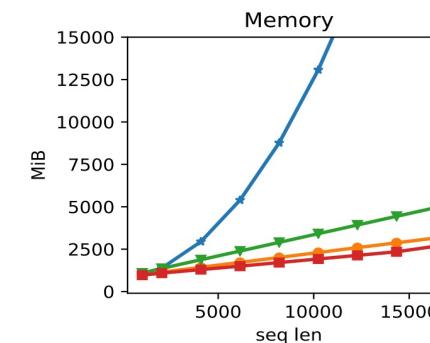
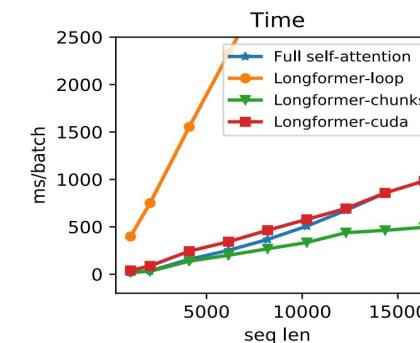
(b) Sliding window attention



(c) Dilated sliding window



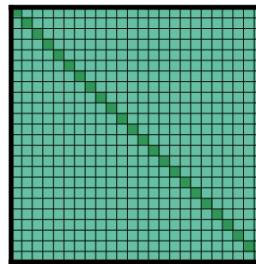
(d) Global+sliding window



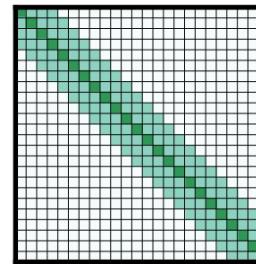
Model	QA			Coref.		Classification	
	WikiHop	TriviaQA	HotpotQA	OntoNotes	IMDB	Hyperpartisan	
RoBERTa-base	72.4	74.3	63.5	78.4	95.3	87.4	
Longformer-base	75.0	75.2	64.4	78.6	95.7	94.8	

Compound Sparse Attention

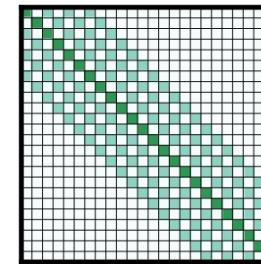
Longformer (2020, 258회 인용 / AllenAI)



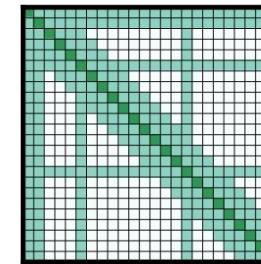
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

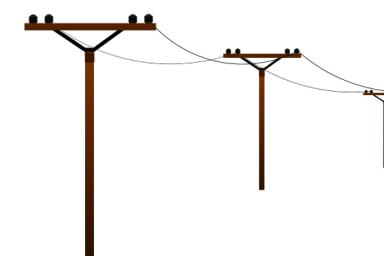
Attention Pattern: Transformer Layer의 일반적인 특성 반영

Local Attention

- Lower Layer: Local Information → Sliding Window Attention (낮은 층에서는 문장의 지역적인 정보를 인코딩한다).
- Higher Layer: Higher-level representation → Dilated Sliding Window Attention (높은 층에서는 문장의 더 넓은 정보를 인코딩한다).

Global Attention

- Sliding Window와 Dilated Sliding Window Attention은 [CLS] Token과 같이 Task Specific Representation을 구성할 수 없다.
- [CLS] token과 같이 단 하나만 사용하는 것이 아니라, 선택된 위치에 복수 개 삽입 (like 전봇대)



Compound Sparse Attention

ETC: Encoding Long and Structured Inputs in Transformers (EMNLP, 2020, 24회 인용 / Google Research)

Extended Transformer Construction

Band Attention + (External) Global–Node Attention

Key Ideas

- **Global–Local Attention**

Global Node를 통해 전체 문장의 Attention을 계산

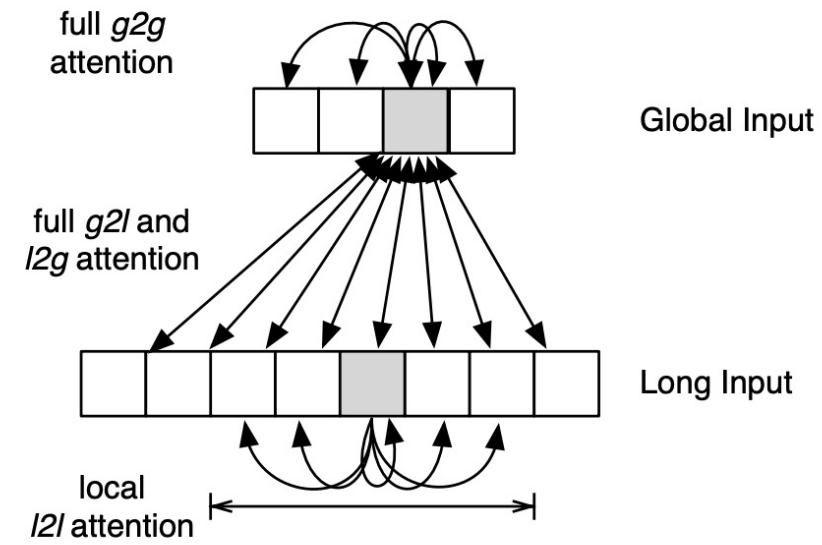
Local Attention을 통해 인접 Token들 간의 Attention 계산

- **Relative Position Representations**

문서 내 문장 내에서 단어의 상대적 위치를

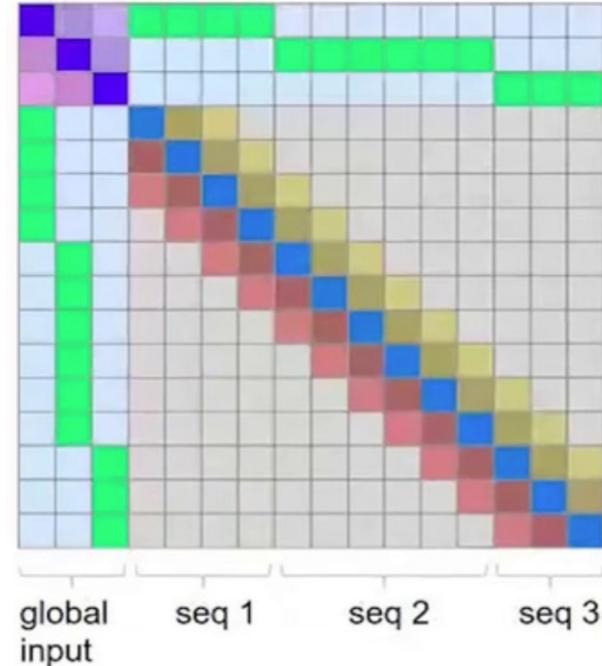
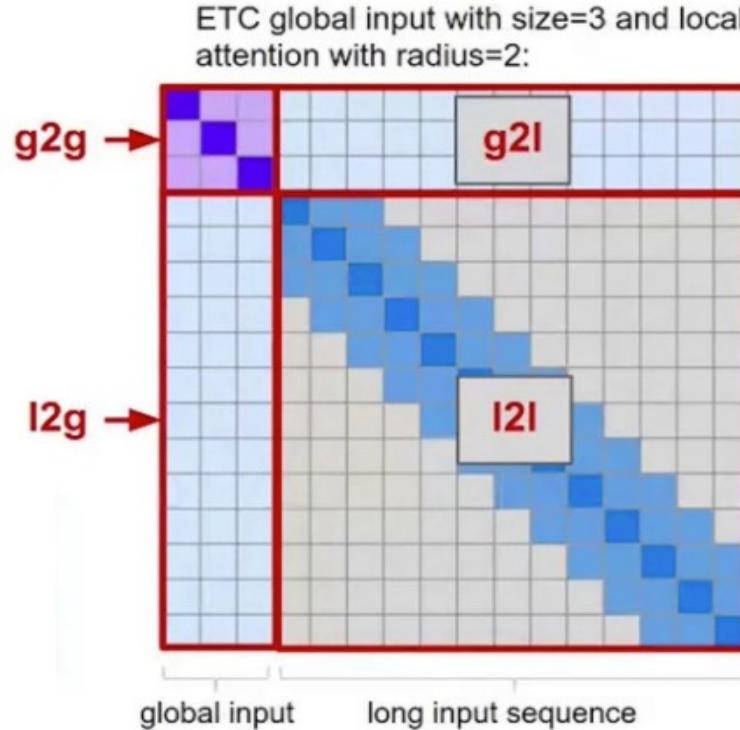
Position Encoding에 반영한다.

Global-Local Attention (ETC):



Compound Sparse Attention

ETC: Encoding Long and Structured Inputs in Transformers (EMNLP, 2020, 24회 인용 / Google Research)



Global Node 하나마다 하나의 문장 Attend

색깔 별로 Relative Positional Encoding

QA Task에서는 Question과 Text의 Separation이 중요

Compound Sparse Attention

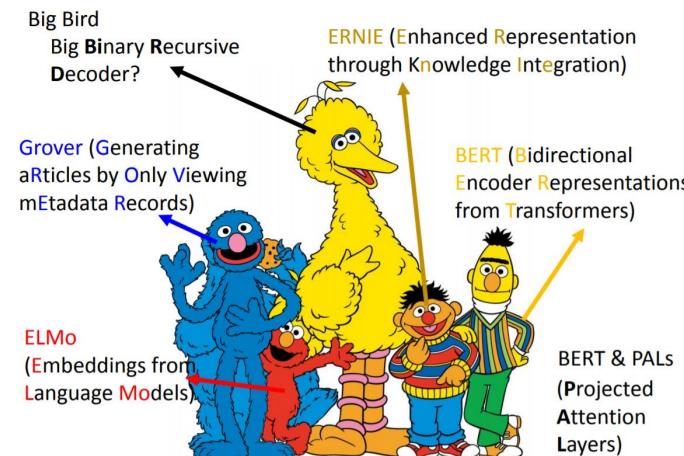
BigBird (NeurIPS, 2020, 139회 인용 / Google Research)

BigBird

Band Attention + Global–Node Attention + Random Attention

Question

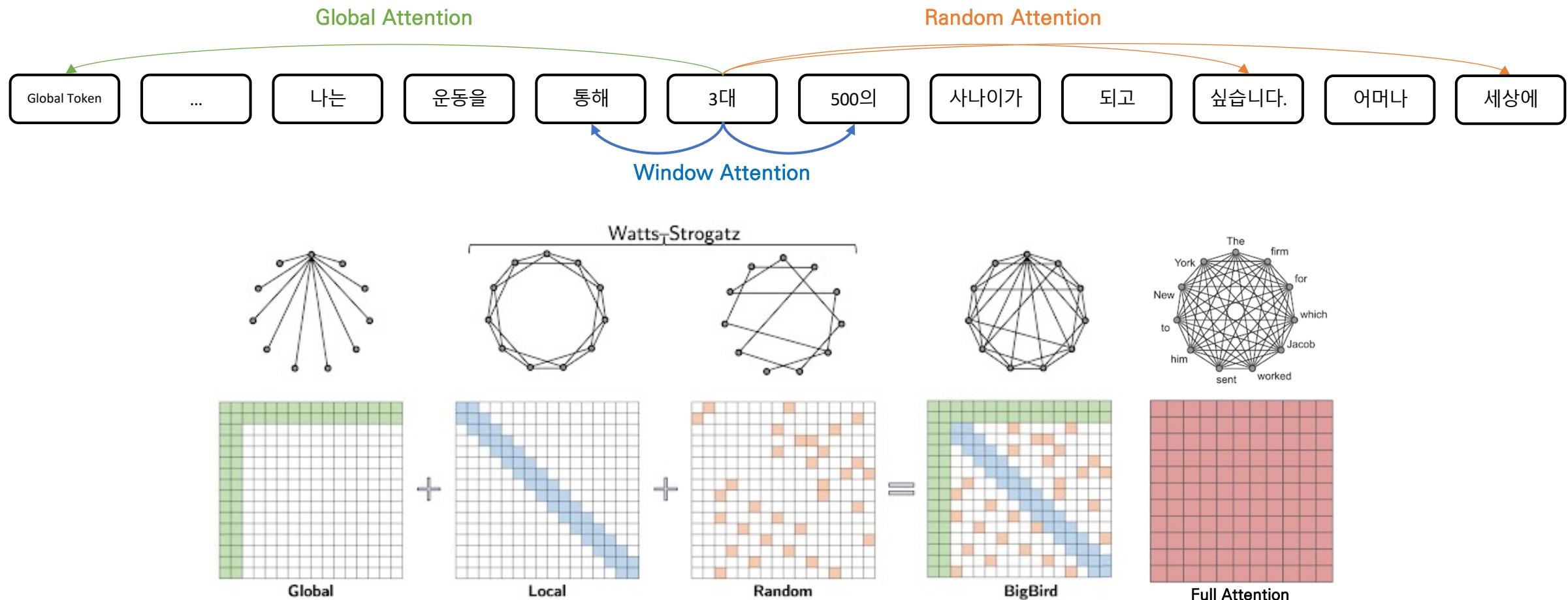
- Sparse Attention이 원래 Attention의 성능과 동일할 수 있는가?
- 더 적은 Inner–product로 Full Quadratic Self–Attention의 성능을 보존할 수 있는가?
⇒ Sparsity를 생성하면서도 최대한 Full Attention Model과 유사한 성능을 만들고자 한다.



Compound Sparse Attention

BigBird (NeurIPS, 2020, 139회 인용 / Google Research)

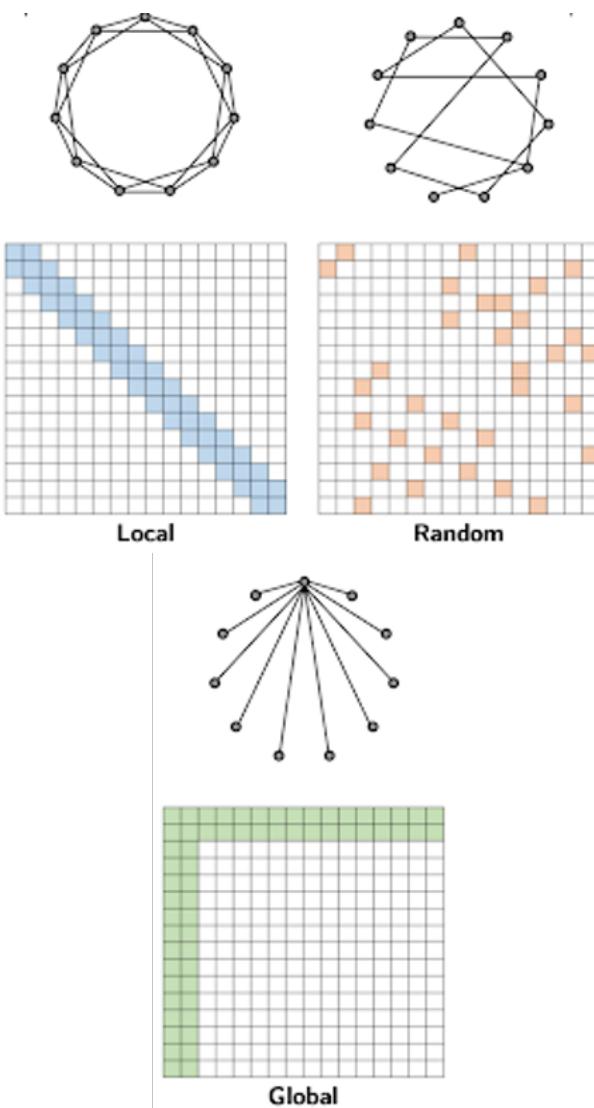
BigBird's Attentions



참고: <https://ai.googleblog.com/2021/03/constructing-transformers-for-longer.html>

Compound Sparse Attention

BigBird (NeurIPS, 2020, 139회 인용 / Google Research)



- Random Attention (Random Blocks) \approx Graph Sparsification Problem

Graph Theory에서 Random Graph는 Complete Graph를 잘 근사하기 때문에, Full Attention Graph를 Random Graph로 치환해도 문제가 없다.

(Erdos–Renyi model, Spectral Graph Theory)

- Local Attention (Local Window)

Linguistic Structure를 반영하기 위해 인접한 Token에 대한 Attention이 필요하다.

실험 결과

Random Blocks와 Local Window만으로는 BERT에 상응하는 성능을 달성할 수 없었음.

- Global Attention (Global Tokens)

› BIGBIRD-ITC (기존 Corpus에 존재하는 Token들 중 Global Token을 지정)

ex) “global”

› BIGBIRD-ETC (문장에 [CLS]와 같은 Global Token 추가)

Results

BigBird의 Sparse Attention으로 더 긴 길이의 문장을 Transformer에 적용 가능 → DownStream Task의 성능

- Basic Masked Language Modeling을 통해 더 좋은 Representation을 학습할 수 있었는지
- QA Task에서 긴 문장의 Input이 정답을 유도할 수 있는 더 많은 Evidence를 제공하는지
- 문서 요약에서 512 token보다 더 긴 문장의 효과를 볼 수 있는지

를 실험을 통해 보이고자 한다.

Encoder–Decoder Task Setup

- Sparse Attention을 Encoder에만 적용
- Encoder Input이 Decoder Output보다 높은 수준으로 차이가 있기 때문에, Decoder에서는 Full Attention을 사용한다. (Input Median Len: 30000이상 / Decoder Median Len: 200)

Compound Sparse Attention

BigBird (NeurIPS, 2020, 139회 인용 / Google Research)

문서 요약 ROUGE Score

Model	Arxiv			PubMed			BigPatent			
	R-1	R-2	R-L	R-1	R-2	R-L	R-1	R-2	R-L	
Prior Art	SumBasic [68]	29.47	6.95	26.30	37.15	11.36	33.43	27.44	7.08	23.66
	LexRank [25]	33.85	10.73	28.99	39.19	13.89	34.59	35.57	10.47	29.03
	LSA [97]	29.91	7.42	25.67	33.89	9.93	29.70	-	-	-
	Attn-Seq2Seq [85]	29.30	6.00	25.56	31.55	8.52	27.38	28.74	7.87	24.66
	Pntr-Gen-Seq2Seq [77]	32.06	9.04	25.16	35.86	10.22	29.69	33.14	11.63	28.55
	Long-Doc-Seq2Seq [20]	35.80	11.05	31.80	38.93	15.37	35.21	-	-	-
	Sent-CLF [81]	34.01	8.71	30.41	45.01	19.91	41.16	36.20	10.99	31.83
	Sent-PTR [81]	42.32	15.63	38.06	43.30	17.92	39.47	34.21	10.78	30.07
	Extr-Abst-TLM [81]	41.62	14.69	38.03	42.13	16.27	39.21	38.65	12.31	34.09
Base	Dancer [31]	42.70	16.54	38.44	44.09	17.69	40.27	-	-	-
	Transformer	28.52	6.70	25.58	31.71	8.32	29.42	39.66	20.94	31.20
	+ RoBERTa [76]	31.98	8.13	29.53	35.77	13.85	33.32	41.11	22.10	32.58
	+ Pegasus [107]	34.81	10.16	30.14	39.98	15.15	35.89	43.55	20.43	31.80
Large	BIGBIRD-RoBERTa	41.22	16.43	36.96	43.70	19.32	39.99	55.69	37.27	45.56
	Pegasus (Reported) [107]	44.21	16.95	38.83	45.97	20.15	41.34	52.29	33.08	41.75
	Pegasus (Re-eval)	43.85	16.83	39.17	44.53	19.30	40.70	52.25	33.04	41.80
	BIGBIRD-Pegasus	46.63	19.02	41.77	46.32	20.65	42.33	60.64	42.46	50.01

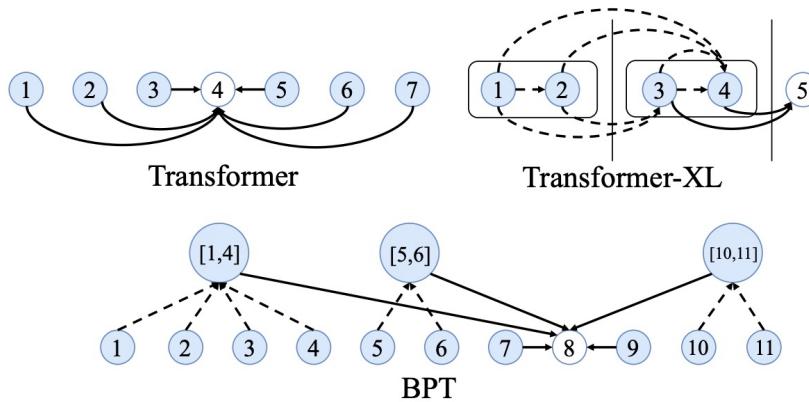
Table 4: Summarization ROUGE score for long documents.

Model	HotpotQA			NaturalQ		TriviaQA		WikiHop	
	Ans	Sup	Joint	LA	SA	Full	Verified	MCQ	
HGN [26]	82.2	88.5	74.2	-	-	-	-	-	-
GSAN	81.6	88.7	73.9	-	-	-	-	-	-
ReflectionNet [32]	-	-	-	77.1	64.1	-	-	-	-
RikiNet-v2 [61]	-	-	-	76.1	61.3	-	-	-	-
Fusion-in-Decoder [39]	-	-	-	-	-	84.4	90.3	-	-
SpanBERT [42]	-	-	-	-	-	79.1	86.6	-	-
MRC-GCN [87]	-	-	-	-	-	-	-	78.3	-
MultiHop [14]	-	-	-	-	-	-	-	76.5	-
Longformer [8]	81.2	88.3	73.2	-	-	77.3	85.3	81.9	-
BIGBIRD-ETC	81.2	89.1	73.6	77.8	57.9	84.5	92.4	82.3	-

QA Task Exact Match

Extended Sparse Attention

BP-Transformer (2019, 24회 인용 / AWS AI Lab)



BP-Transformer

- Binary Partitioning을 통해 문장을 Graph의 형태로 나타낸다.

Binary Partitioning: 문장을 재귀적으로 나누어 단일 Token이 될 때 까지 나눈다.

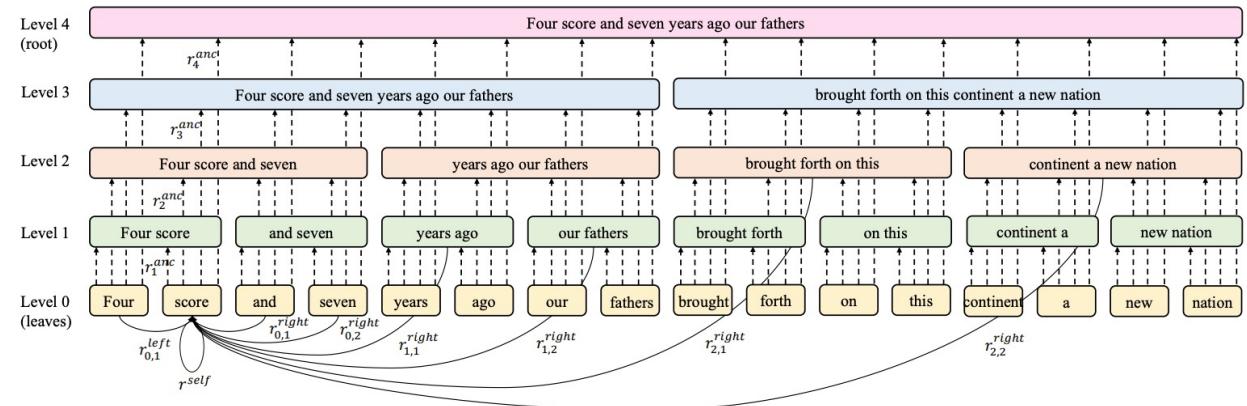
- Graph Self Attention을 통해 Representation을 Update

Edge Construction

- Affiliated Edges: Span Node에 대하여 포함되어 있는 Token들과 모두 연결 (Hierarchical Linguistic Structure 반영)
- Contextual Edges: 모든 이웃 Token을 Edge로 연결하지 않고, 상위 Span Node들을 연결

파라미터 k 를 통해 Layer별 몇 개의 노드와 연결할 지 결정

Binary Partitioning



Content-based Sparse Attention

위치 기반이 아닌, Token의 내용 기반

Content Based Sparse Attention의 목적

- 모든 Query-Key Pair에 대하여 내적을 실시하지 않고, 최대 내적 값을 갖는 Key값 탐색
- 가장 유사한 Content들 간의 Attention으로 Sparse Attention을 달성하고자 한다.
- 유사한 Content Search Algorithm: Maximum Inner Product Search(MIPS) problem
$$\text{MIPS} = \operatorname{argmax}_{k \in K} (q^T k)$$
- (Exact) MIPS를 근사하기 위한 여러 전략이 있으며, Transformer에서는
 - Locality-Sensitive Hashing(LSH): Reformer
 - Nearest Neighbor Search(NNS): Routing Transformer

Content-based Sparse Attention

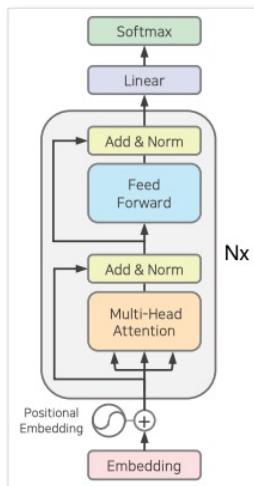
Reformer (ICLR, 2020, 356회 인용 / Google Research)

Reformer: The Efficient Transformer → Transformer의 극한의 효율을 달성하고자 한다.

Main Idea

- Transformer 모델에서 Memory 효율적으로 사용하는 방법
 - ~~모델깊이, 모델넓이, 문장길이 줄이기~~
 - ~~Batch size를 줄이기~~
 - 메모리에 저장되는 **중간결과물** 줄이기

$$\text{Memory} \approx \text{모델깊이} \times \text{모델넓이} \times \text{문장길이} \times \text{Batch}$$



Attention Layer 개선

쓸모 없는 Attention을
줄일 수 있도록
LSH Attention 적용

문장길이

필요 없는 연결 제거

Residual Block 개선

중간 값 저장없이 역전파
할 수 있는 Reversible
Network 구조 적용

모델깊이

메모리 저장 구조 변경

Feedforward Layer 개선

계산하는 단위를 나누어
메모리를 사용할 수 있는
Chunking 적용

모델넓이

단계별로 메모리 사용

DSBA 연구실 김정희 석사과정의 자료가 가장 우수하여 그대로 사용하고자 함

참고자료: <http://dsba.korea.ac.kr/seminar/?mod=document&pageid=1&keyword=reformer&uid=385>

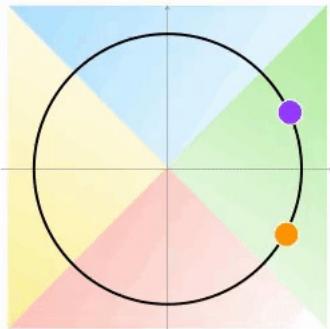
Content-based Sparse Attention

Reformer (ICLR, 2020, 356회 인용 / Google Research)

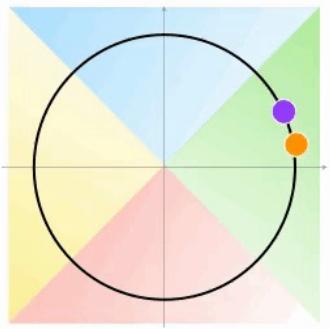
Locality-Sensitive Hashing을 적용한 뒤, 같은 Hashing Bucket 내에서 Attention 적용

Locality-Sensitive Hashing: Token들에 Hashing Bucket 할당

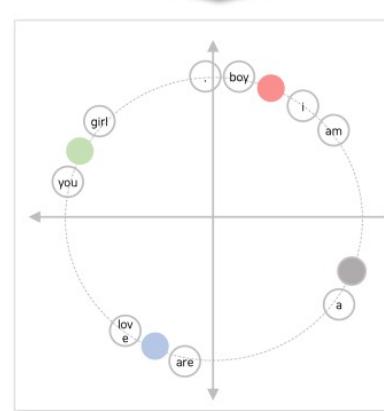
Angular Locality Sensitive Hashing



Angular Locality Sensitive Hashing



i am a boy you are a girl . i love you



i am a boy you are a girl . i love you

LSH Attention

i am a boy you are a girl . i love you

i am boy . i are love you girl you a a

i am boy . i are love you girl you a a

query i am boy . i are love you girl you a a

query i am boy . i are love you girl you a a

query i am boy . i are love you girl you a a

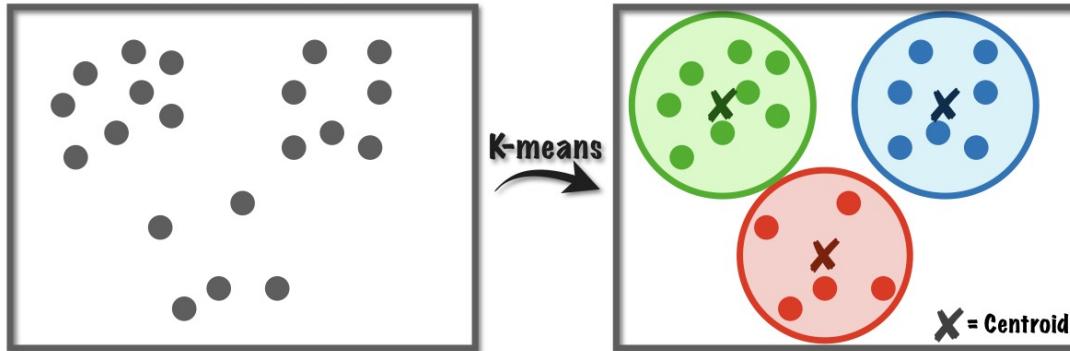
Content-based Sparse Attention

Routing Transformer (TACL, 2020, 66회 인용 / Google Research)

Routing Transformer

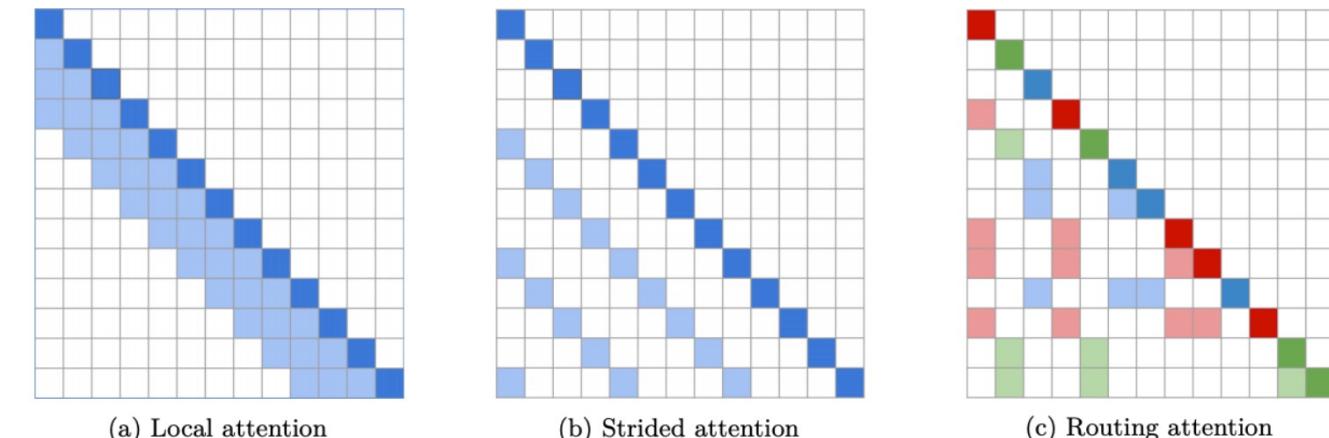
- Query가 Attention을 구할 대상인 Key들에게 Routed(연결)되어 대상이 제한된다.
- Maximum Inner Product Search (MIPS)를 근사하려는 목적에서 LSH보다 spherical k-means가 더 우수

Query & Key K-Means



각 Token에 대한 K-Means를 통해 Clustering 진행

Attention Matrix



(c)에서 같은 색깔 (Cluster)에 대해서만 Attention이 적용되는 것을 볼 수 있다.

Sparse Attention 정리

일반 Self-Attention은 Transformer의 중요한 역할을 하지만, 두 가지 문제 발생

- Complexity: Self Attention의 Complexity ($O(T^2 \cdot D)$)는 문장이 길어지면 Bottleneck이 된다.
- Structural Prior: Input에 대한 어떤 Structural Bias도 없다 (장점인 동시에 단점)

Vanilla Transformer Attention Matrix는 사실상 Sparse하다?

→ Attention은 훈련 뒤에는 볼 대상만 보게 된다.

→ 그럼 애초에 **Structural Bias**를 부여하여, Query-Key Pair 개수를 제한하자!

Sparse Attention의 공통 주장

- Computational Cost:** 문장이 길어질수록, Attention에 필요한 비용이 크다.
Input Length에 Quadratic하게 증가하는 비용 대신에, Linear하게 증가하도록 만드는 것이 목적
 $O(n^2) \rightarrow O(n)$
- Long Range Dependency:** 긴 문장에는 Transformer를 적용하기 힘들다 (+Cost 문제)
Global Attention을 통해 Long Range Dependency 해소

Sparse Attention 효과

- 비용 / Dependency를 감소 시키면 512 Token보다 더 긴 문장을 Input으로 활용할 수 있다.
Input 문장을 길게 만들면, Downstream Task에 사용되는 단서가 많아진다.
Downstream Task로 QA & 문서 요약이 많이 사용 (단서 증폭 효과를 살피기 위하여)
- Sparse Attention은 Long Input이 사용되는 Encoder에 적용된다.

Sparse Attention (2021.07 기준)

- Position Based Sparse Attention**
 - Atomic Sparse Attention**
 - Compound Sparse Attention** (여러 Atomic Sparse Attention이 모여 구성)
 - Star Transformer (NAACL, 2019, 87회 인용)
 - Longformer (2020, 258회 인용 / AllenAI)
 - ETC (EMNLP, 2020, 24회 인용 / Google Research)
 - BigBird (NeurIPS, 2020, 139회 인용 / Google Research)
 - Sparse Transformer (2019, 353회 인용 / OpenAI)
- Extended Sparse Attention** (Atomic 혼합 대신 새로운 관점 또는 Non-Text Dataset)
 - BP-Transformer (2019, 24회 인용 / AWS AI Lab)
 - Image Transformer (ICML, 2018, 448회 인용)
 - Axial Transformer (2019, 59회 인용 / Google Brain)
- Content-Based Sparse Attention** (Query와 Key Token의 유사도를 통한 Attention)
 - Reformer (ICLR, 2020, 360회 인용 / Google Research)
 - Routing Transformer (TACL, 2020, 66회 인용 / Google Research)
 - Sparse Adaptive Connection, Sparse Sinkhorn Attention

- 1) 이렇게 좋은데 왜 기존 Transformer 대신 Default로 쓰지 아니한가?
- 2) QA / Text Summarization과 같이 Long Sequence가 필요한 Task에는 필요할 Attention이라고 생각
- 3) HuggingFace에 등록된 모델
 - Longformer
 - BigBird (+Pegasus)
 - Reformer