DSAIL

Data Science Artificial Intelligence Lab.

# 2023 Winter Lab Seminar - Big Bird: Transformers for Longer Sequences

## (2020, NeurIPS)

2023.01.12

Jeon Hyolim
gyfla1512@g.skku.edu

# Content

- 1. Introduction
- 1.1 Related work
- 2. BIGBIRD Architecture
- 3. Theoretical Results about Sparse Attention Mechanism
- 4. Experiments: NLP, Genomics
- 5. Conclusion
- 6. Summary and Take-home message

# Introduction

- Transformers [91], such as BERT [22, 63],
  - wildly successful for a wide variety of Natural Language Processing (NLP) tasks and consequently are mainstay of modern NLP research.
  - versatility and robustness -> wide-scale adoption of Transformers
  - diverse range of sequence based tasks – as a seq2seq model for translation [91], summarization [66], generation [15], etc. or as a standalone encoders for sentiment analysis [83], POS tagging [65], machine reading comprehension [93], etc. – and it is known to vastly outperform previous sequence models like LSTM [37].

# Introduction

- Transformers [91]

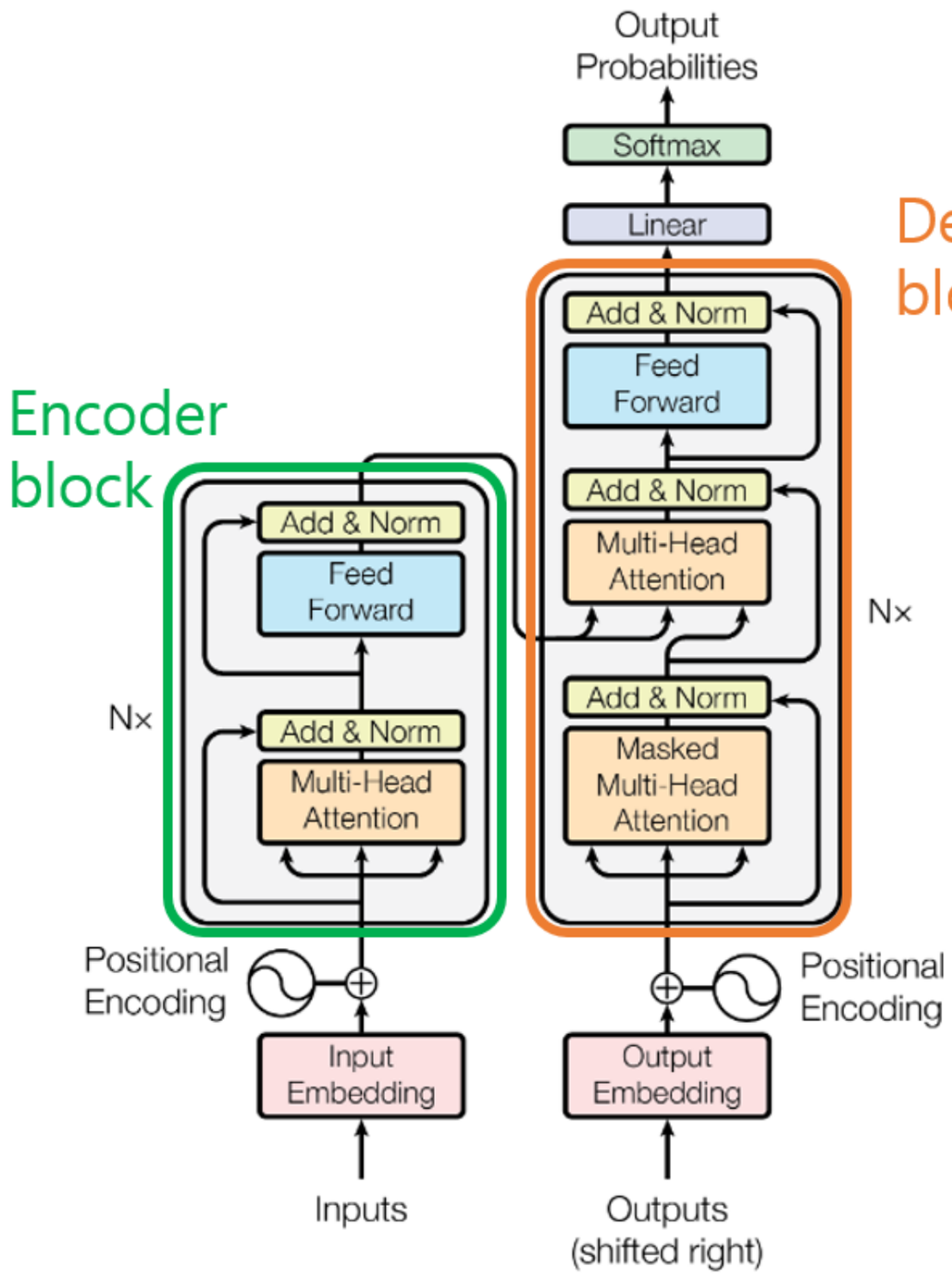


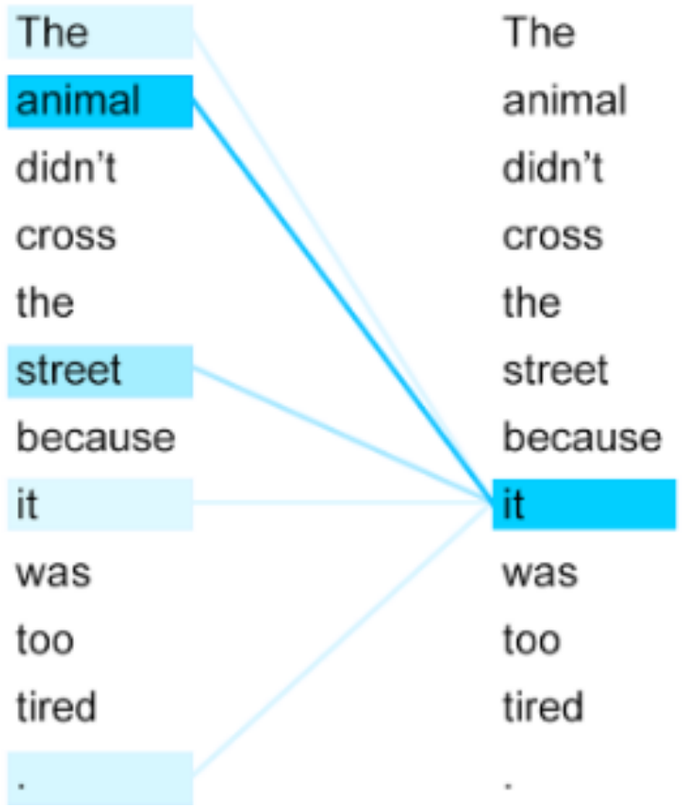

Figure 1: The Transformer - model architecture.

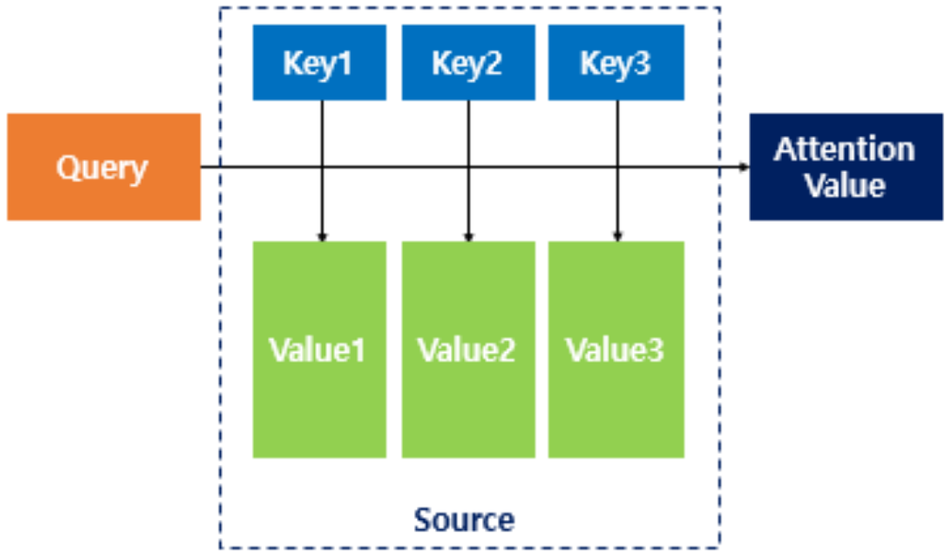
# Introduction

- key innovation in Transformers: introduction of a self-attention mechanism
  - can be evaluated in parallel for each token of the input sequence,
  - eliminating the sequential dependency in recurrent neural networks, like LSTM
- Parallelism -> 1. leverage the full power of modern SIMD hardware accelerators like GPUs/TPUs, 2. facilitating training of NLP models on datasets of unprecedented size

# Introduction

- Transformers [91], such as BERT [22, 63],
  - Self-attention



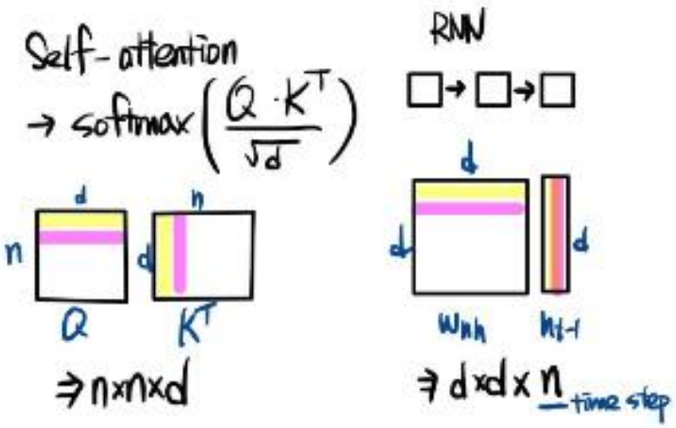




Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

# Introduction

- self-attention mechanism
  - Operation: allowing each token in the input sequence to attend independently to every other token in the sequence
  - Problem: full self-attention
    - quadratic in the sequence length => computational and memory requirement
    - Limitation: input sequences of length 512 tokens
    - QA [60], document classification, etc => reduces its direct applicability

# Introduction

- theoretical understanding about self-attention and Transformers

1. What aspects of the self-attention model are necessary for its performance?
2. What can we say about the expressivity of Transformers and similar models?

# Introduction

- Pérez et al. [72]
  - full transformer is Turing Complete
    - (i.e. can simulate a full Turing machine)

1. Can we achieve the empirical benefits of a fully quadratic self-attention scheme using fewer inner-products?
2. Do these sparse attention mechanisms preserve the expressivity and flexibility of the original network?

# Related Work

- interesting attempts, alleviating the quadratic dependency of Transformers

1. <u>embraces the length limitation and develops method around it</u>
   - Simplest way: employ sliding window [93]
   - general most work: call transformer block multiple time with different contexts each time
     - SpanBERT [42], ORQA [54], REALM [34], RAG [57]
   - Limitation: require significant engineering efforts (like back prop through large scale nearest neighbor search) and are hard to train

# Related Work

- SpanBERT [42]
  - Improving Pre-training by Representing and Predicting Spans

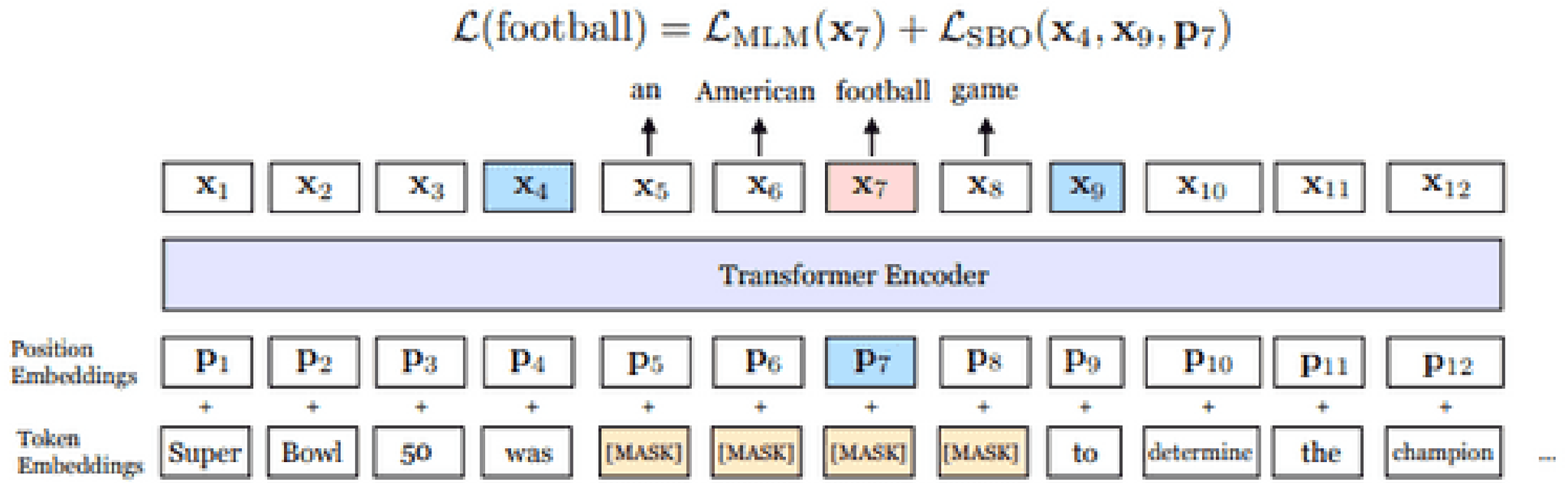
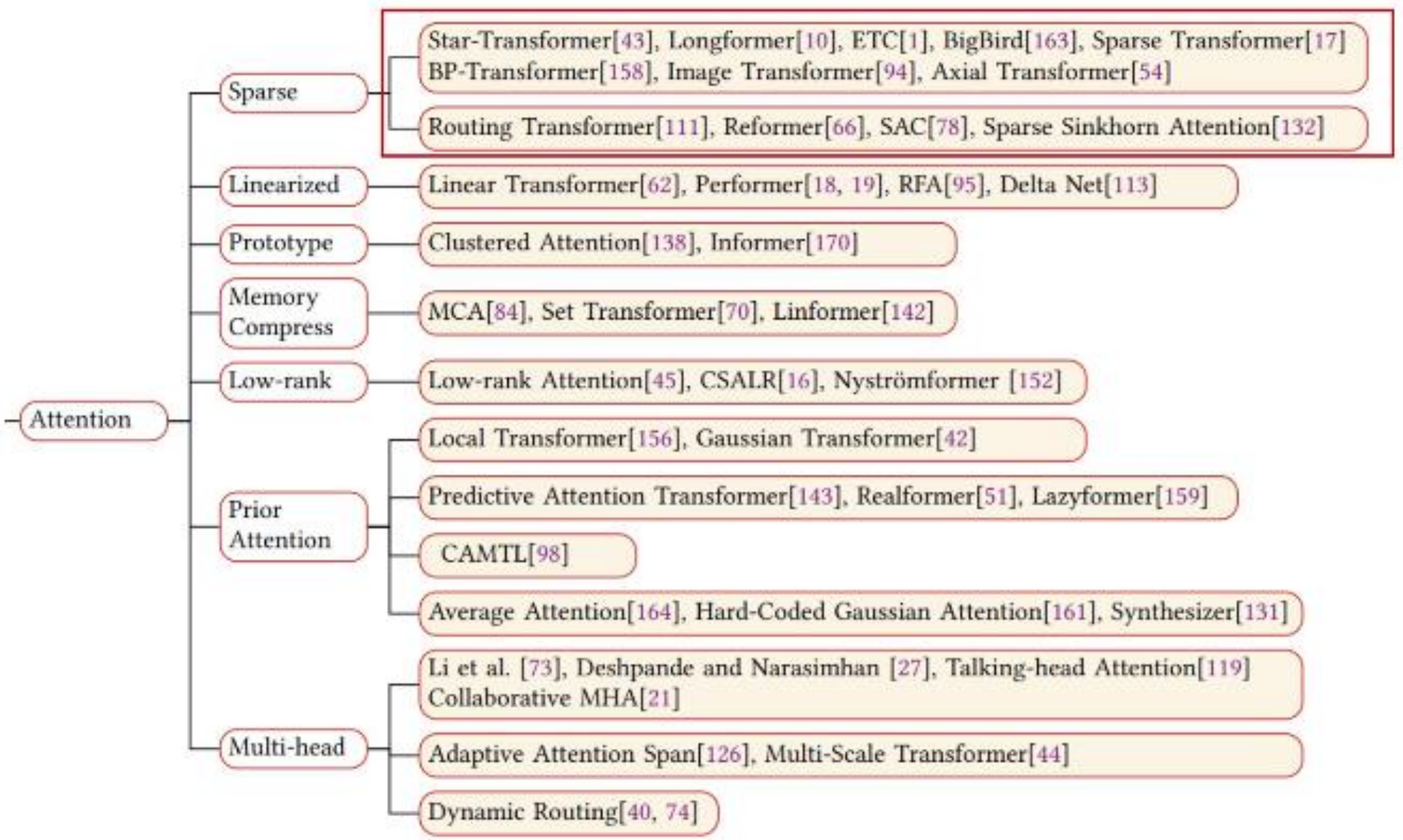
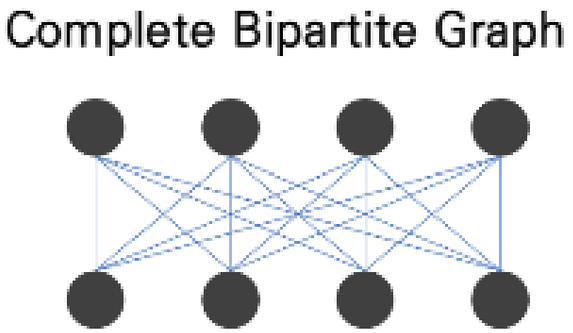


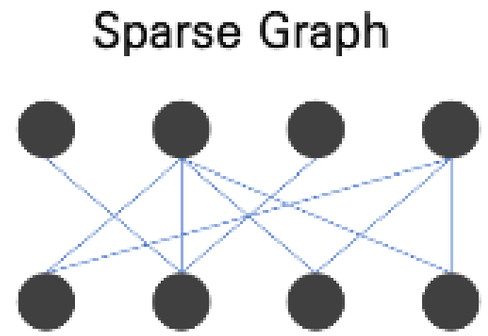

Figure 1: An illustration of SpanBERT training. In this example, the span *an American football game* is masked. The span boundary objective then uses the boundary tokens *was* and *to* to predict each token in the masked span.

# Related Work

2. approaches that do not require full attention



# Related Work

2. approaches that do not require full attention

- Dai et al. [21], Sukhbaatar et al. [82], Rae et al. [74]
  - auto-regresive models
  - work well for left-to-right language modeling but suffer in tasks which require bidirectional context.
- Child et al. [16]
  - sparse model that reduces the complexity to O(n √ n)
- Kitaev et al. [49]
  - reduced the complexity to O(n log(n)) by using LSH to compute nearest neighbors. (Reformer)
- Longformer [8] : introduced a localized sliding window based mask with few global mask to reduce computation and extended BERT to longer sequence based tasks.

# Related Work

2. approaches that do not require full attention

- Extended Transformers Construction [4]
  - designed to encode structure in text for transformers
  - global tokens was used extensively by them to achieve their goals

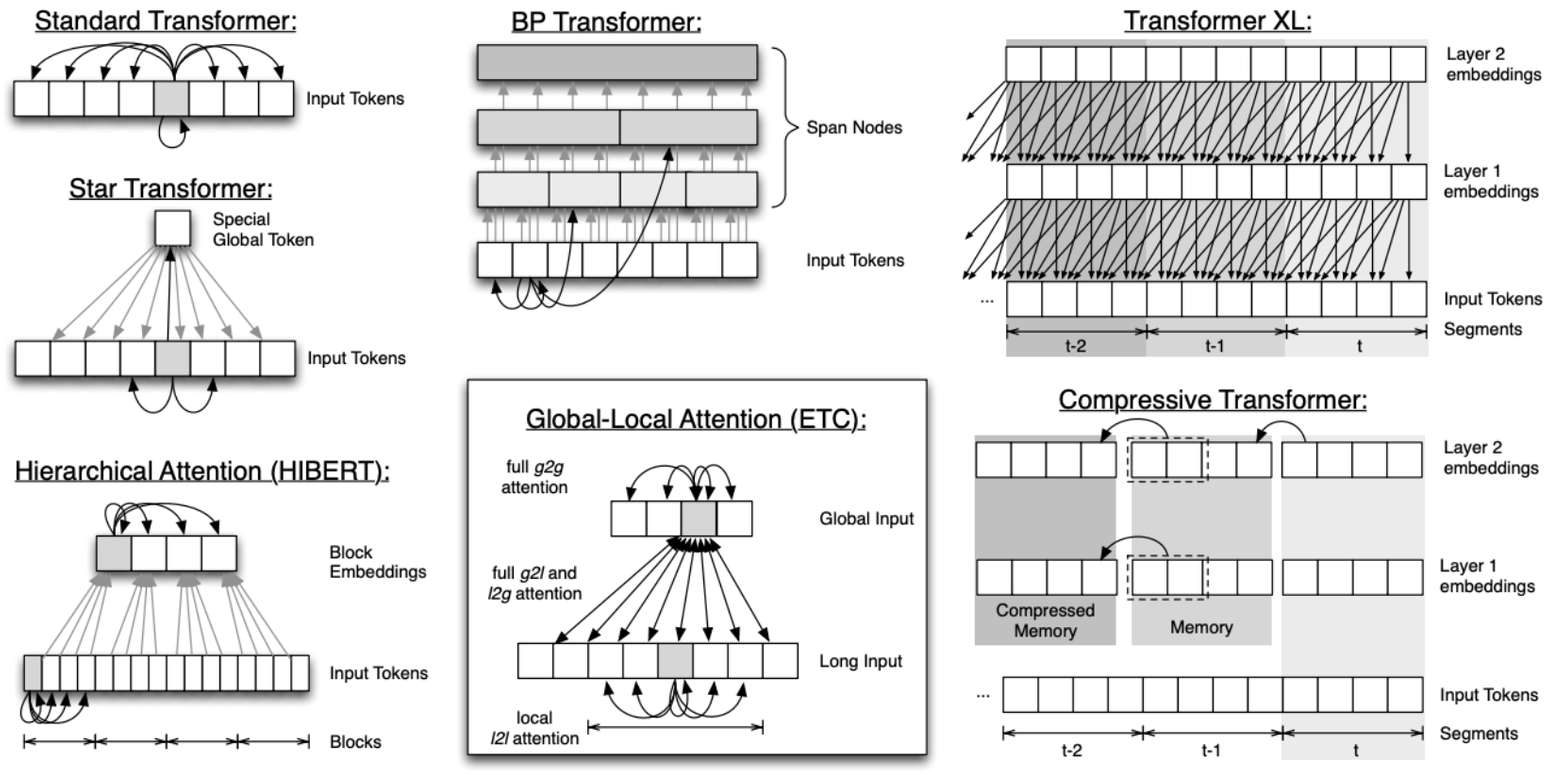



Figure 1: An illustration of mechanisms to scale attention to long inputs, including our proposed model, ETC.

# Related Work

2. approaches that do not require full attention

- A Survey of Transformers(Lin, Tianyang, et al. *AI Open* (2022))

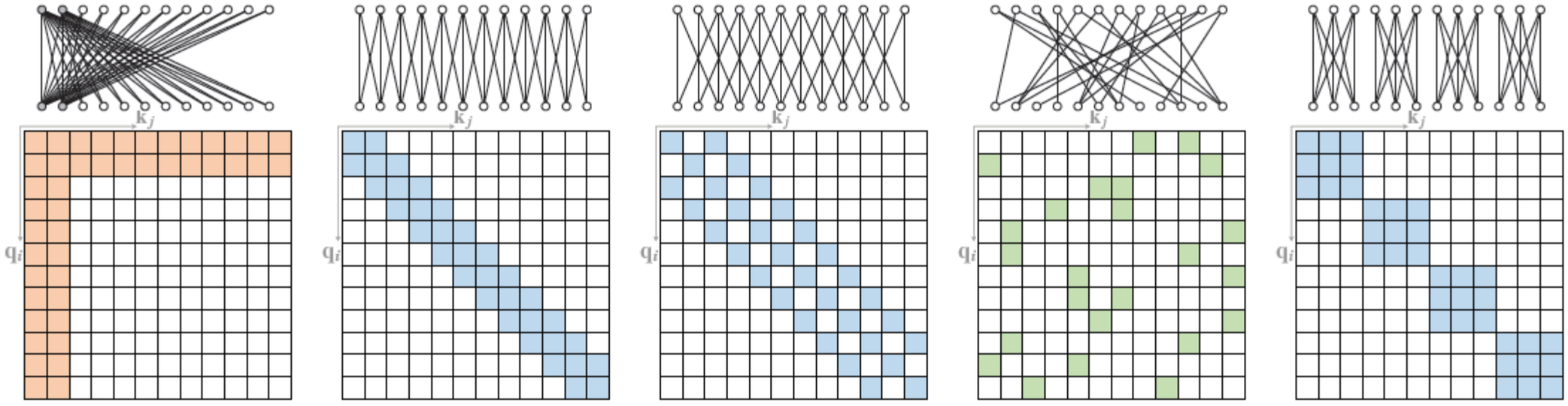
- Sparse attention
  - Computational cost: $O(n^2) \rightarrow O(n)$
  - Long range Dependency: Global attention

- Method
  - Position-based
  - Content-based

# Related Work

2. approaches that do not require full attention

- A Survey of Transformers(Lin, Tianyang, et al. *AI Open* (2022))
- Method: Position-based

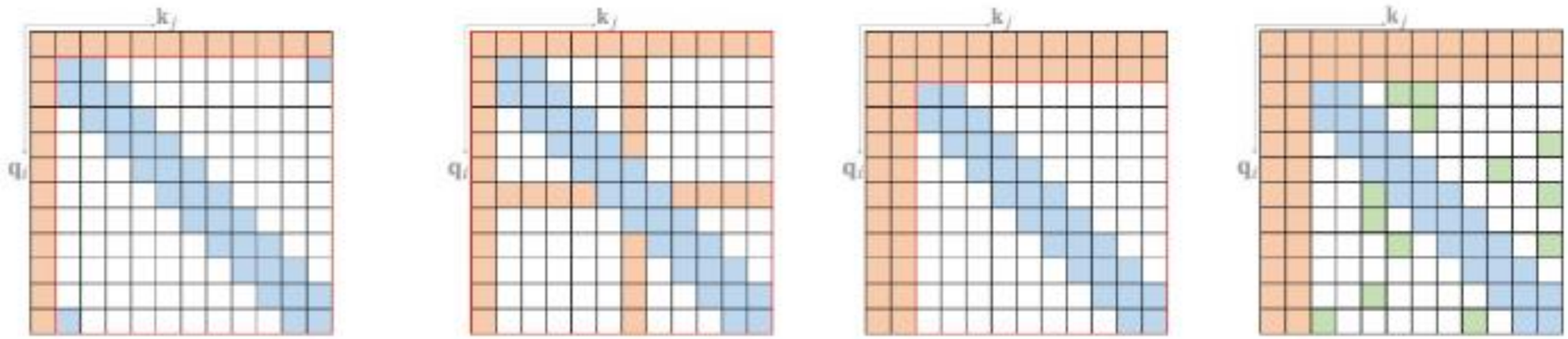



(a) global (b) band (c) dilated (d) random (e) block local

# Related Work

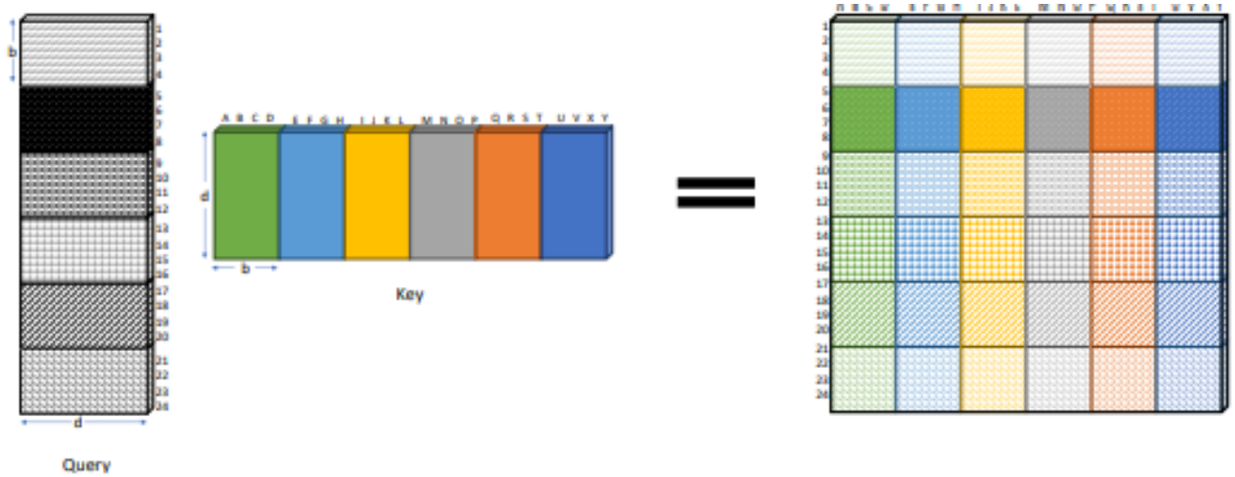
2. approaches that do not require full attention

- A Survey of Transformers(Lin, Tianyang, et al. *AI Open* (2022))
  - Star-Transformer: Band Attention + Global Attention
  - Longformer: Band Attention + Dilated Attention + Global Attention
  - ETC: Band Attention + Global Attention
  - BIGBIRD: Band Attention + Global Attention + Random Attention

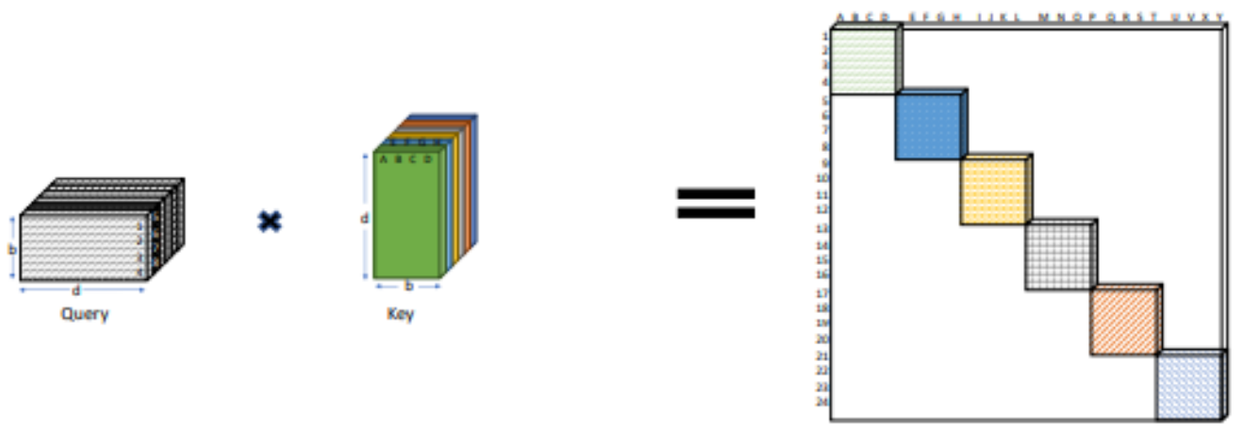


(a) Star-Transformer (b) Longformer (c) ETC (d) BigBird

# Related Work



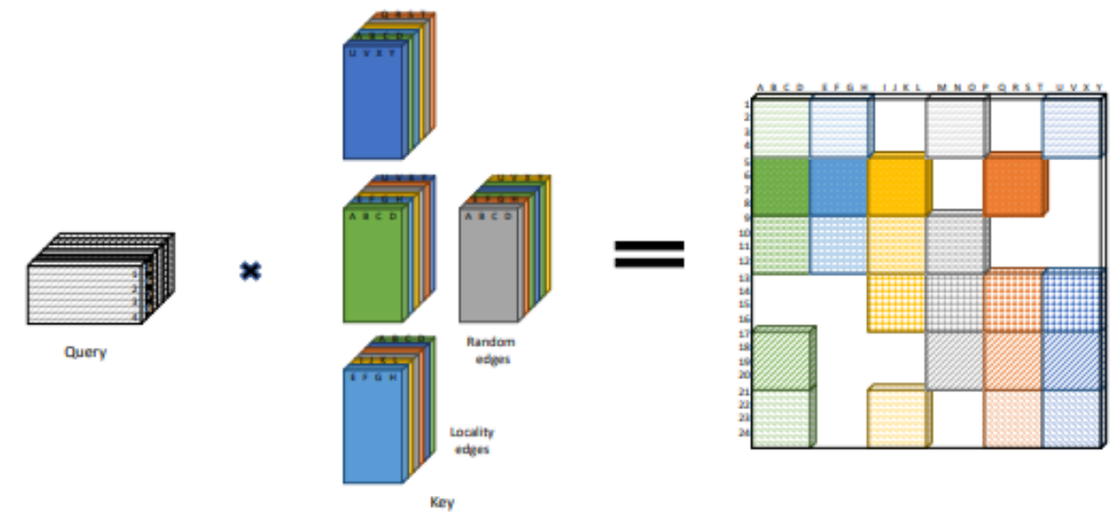

(a) Full all pair attention can be obtained by direct matrix multiplication between the query and key matrix. Groupings just shown for guidance.




(b) Block diagonal attention can be computed by "blockifying" the query and key matrix




(c) Window local attention obtained by "blockifying" the query/key matrix, copying key matrix, and rolling the resulting key tensor (Obtaining rolled key-block tensor is illustrated in detail in Fig. 5). This ensures that every query attends to at least one block and at most two blocks of keys of size $b$ on each side.



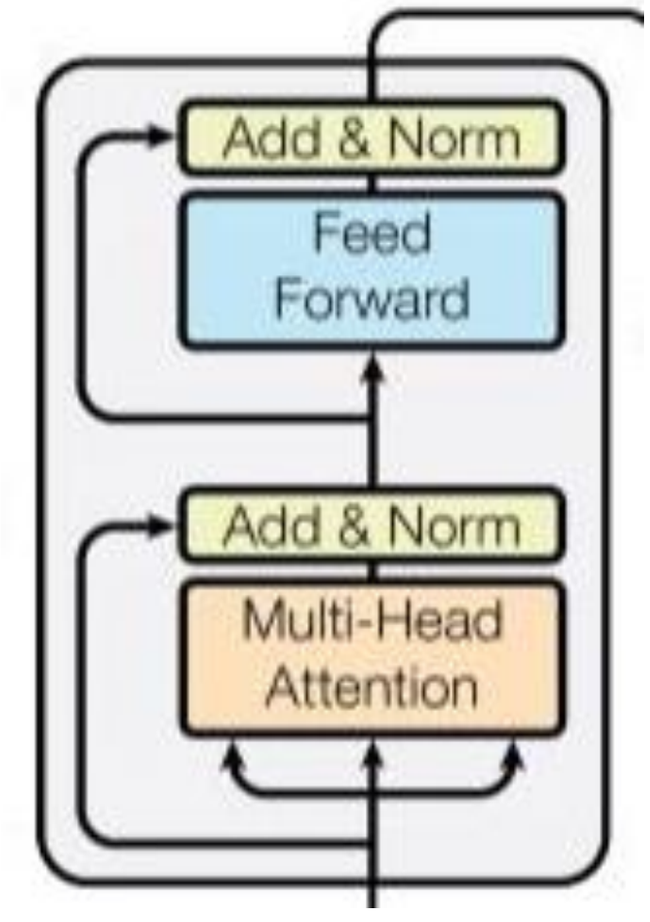

(d) Window + Random attention obtained by following the procedure above along with gathering some random key blocks.

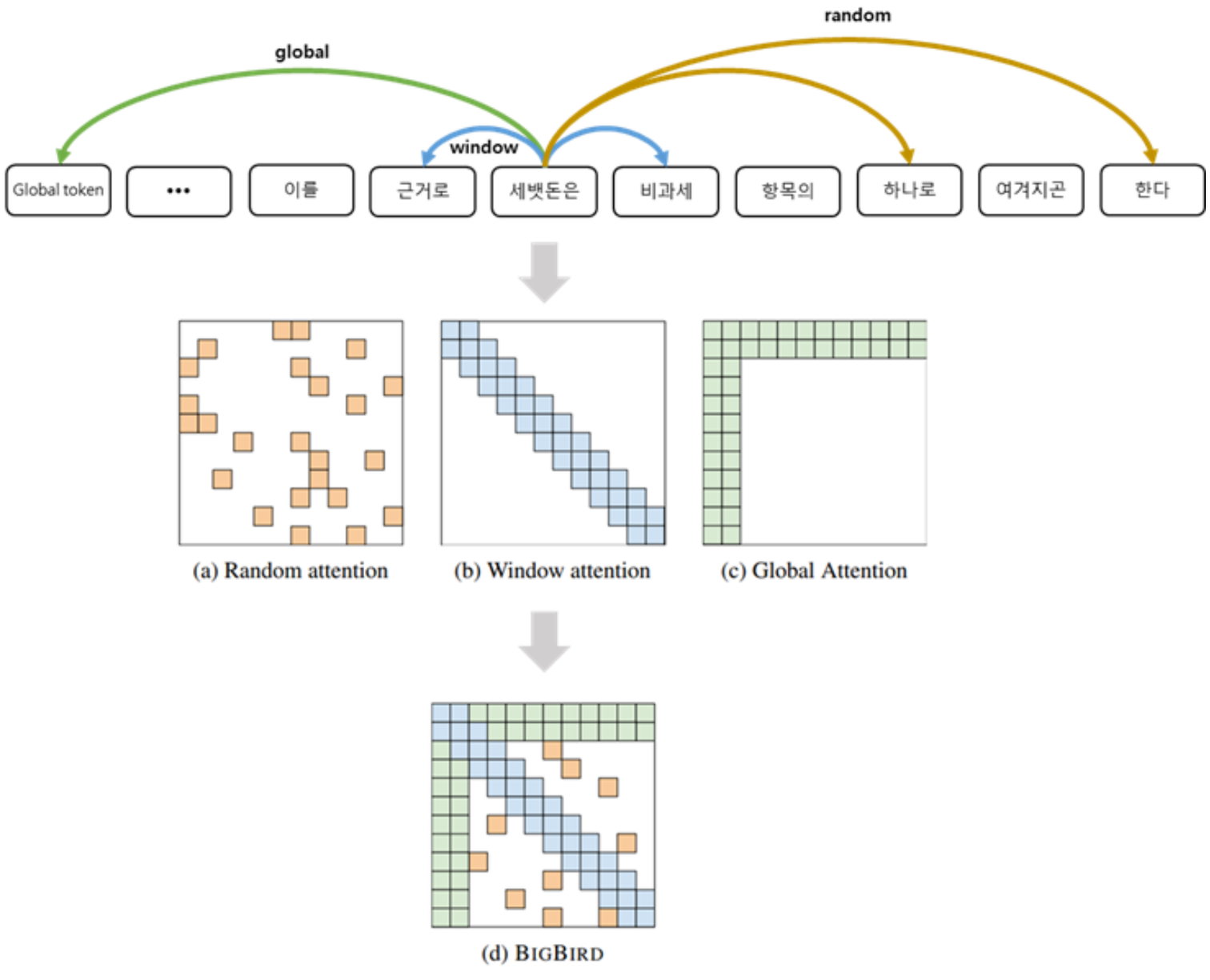
Figure 4: Idea behind fast sparse attention computation in BIGBIRD.

# Related Work

- Star Transformer (NACCL, 2019)
- Longformer(2020)
- ETC(EMNLP, 2020)
- BIGBIRD(NeurIPS, 2020)
- Reformer(ICLR, 2020)

# BIGBIRD Architecture



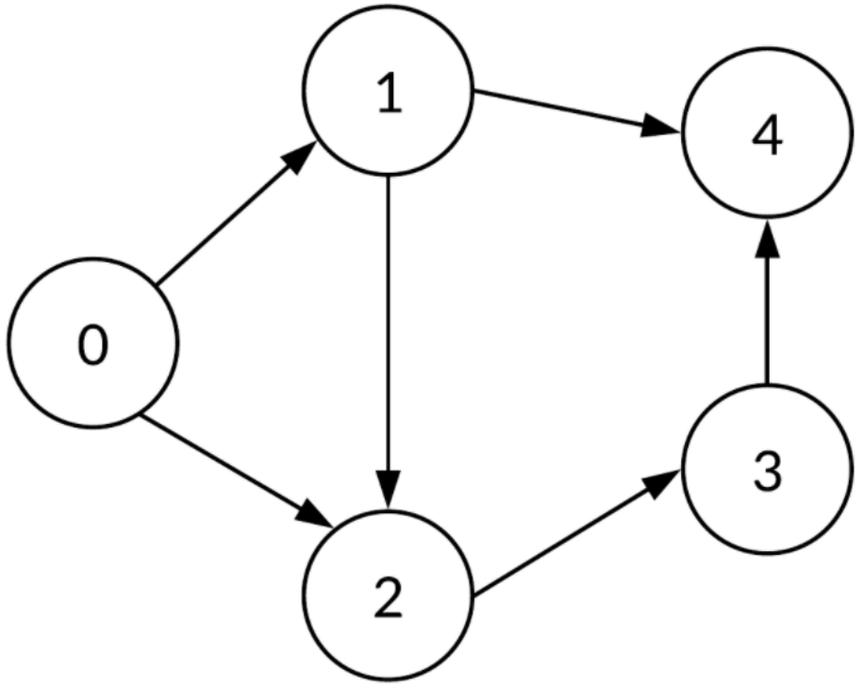
# BIGBIRD Architecture

- Generalized attention mechanism
  - input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \in \mathbb{R}^{n \times d}$
  - directed graph D, vertex set is [n] = {1, . . . , n}.
  - N(i) : the out-neighbors set of node i in D
  - i th output vector of the generalized attention mechanism

$$\text{ATTN}_D(\boldsymbol{X})_i = \boldsymbol{x}_i + \sum_{h=1}^{H} \sigma\left(Q_h(\boldsymbol{x}_i) K_h(\boldsymbol{X}_{N(i)})^T\right) \cdot V_h(\boldsymbol{X}_{N(i)})$$

# BIGBIRD Architecture

- generalised attention mechanism
  - operate on the adjacency matrix A of the graph D




Adjacency Matrix

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |

# BIGBIRD Architecture

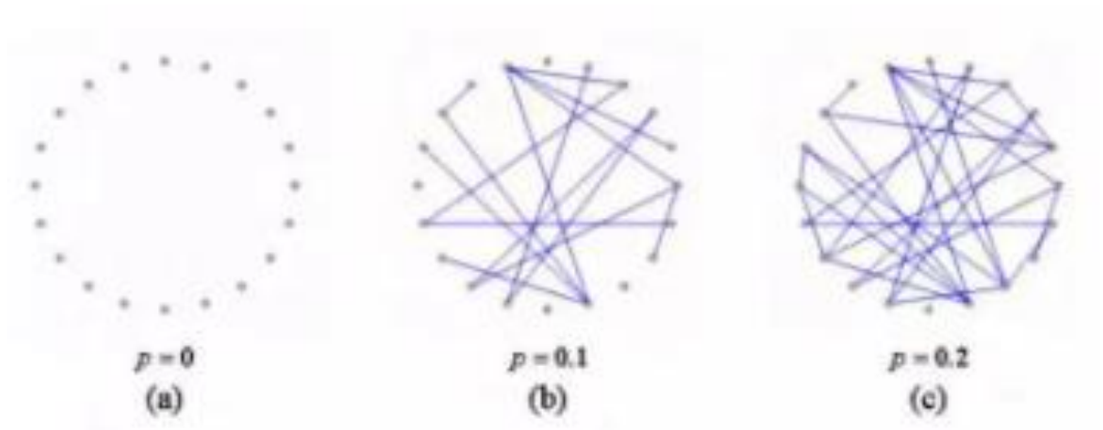
- generalised attention mechanism
  - input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n) \in \mathbb{R}^{n \times d}$
  - directed graph D, vertex set is [n] = {1, . . . , n}.
  - N(i) : the out-neighbors set of node i in D
  - i th output vector of the generalized attention mechanism

$$\text{ATTN}_D(\boldsymbol{X})_i = \boldsymbol{x}_i + \sum_{h=1}^{H} \sigma\left(Q_h(\boldsymbol{x}_i) K_h(\boldsymbol{X}_{N(i)})^T\right) \cdot V_h(\boldsymbol{X}_{N(i)})$$

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

# BIGBIRD Architecture

- generalised attention mechanism
  - quadratic complexity
    - Since all tokens attend on every other token
    - The problem of reducing the quadratic complexity is graph sparsification problem
  - Erdos-Rényi model,




  - each edge is independently chosen with a fixed probability.

# BIGBIRD Architecture

- sparse random graph for attention mechanism should have two desiderata

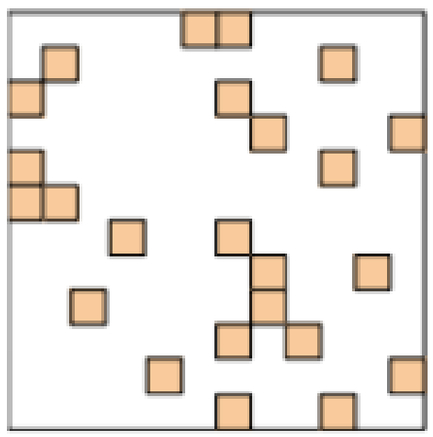
1. **small average path length between nodes**
    - Erdos-Rényi model,
    - each edge is independently chosen with a fixed probability.
    - random graph approximates the complete graph spectrally
    - its second eigenvalue (of the adjacency matrix) is quite far from the first eigenvalue
        - -> rapid mixing time for random walks in the grpah

# BIGBIRD Architecture

- sparse random graph for attention mechanism should have two desiderata

**1. small average path length between nodes**

- sparse attention
- where each query attends over r random number of keys
- i.e. A(i, ·) = 1 for r randomly chosen keys



(a) Random attention

# BIGBIRD Architecture

- sparse random graph for attention mechanism should have two desiderata

2. **locality of reference**

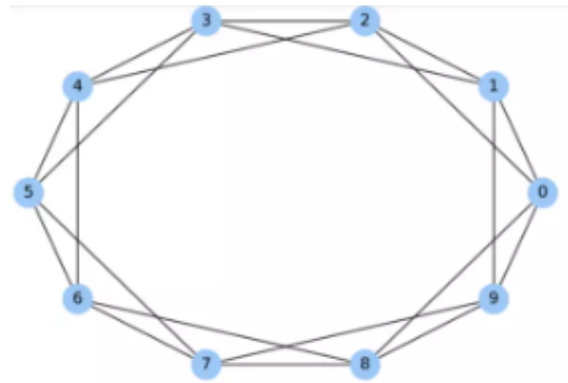
- Locality: proximity of tokens in linguistic structure
- graph theory =>
  - clustering coefficient is a measure of locality of connectivity,
  - high when the graph contains many cliques or near-cliques
- Erdos-Rényi random graphs
  - do not have a high clustering coefficient [84], but a class of random graphs, known as small world graphs, exhibit high clustering coefficient [94]

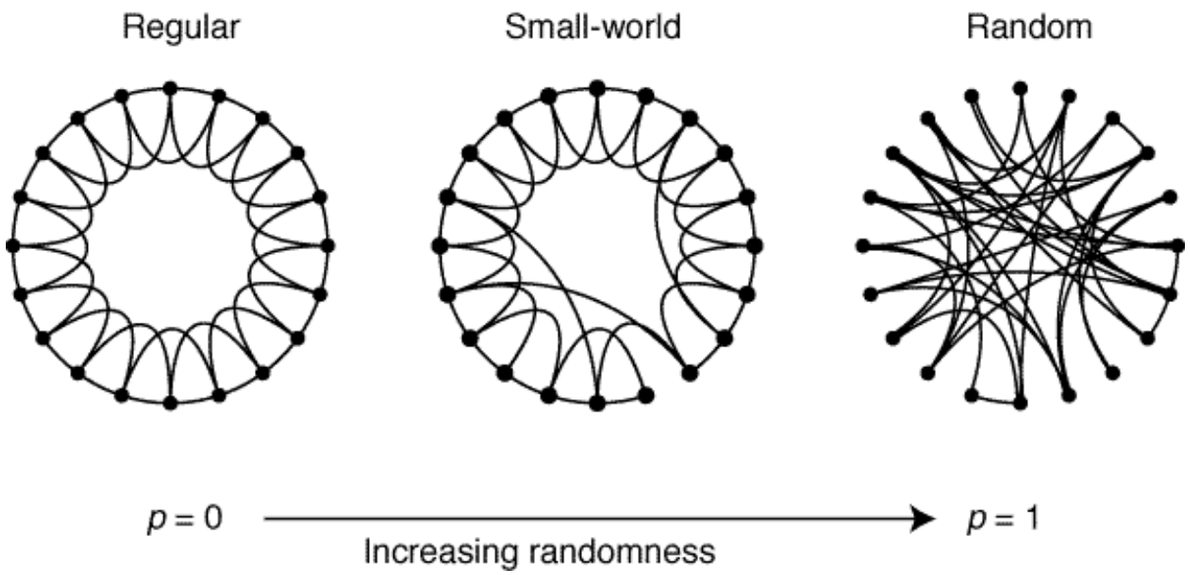
# BIGBIRD Architecture

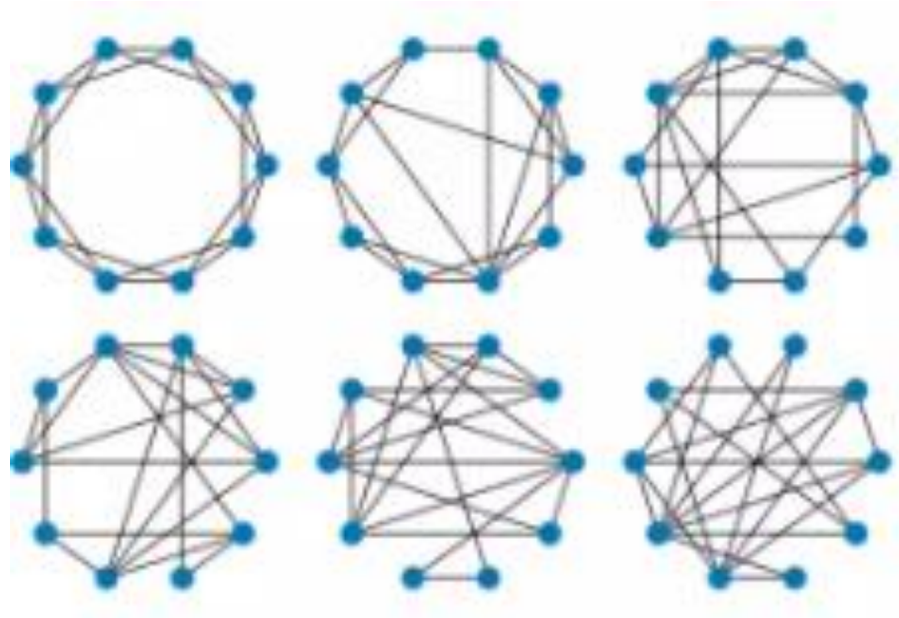
- sparse random graph for attention mechanism should have two desiderata

2. **locality of reference**

- Generative process (Watts and Strogatz [94] )



# BIGBIRD Architecture

- sparse random graph for attention mechanism should have two desiderata

2. **locality of reference**

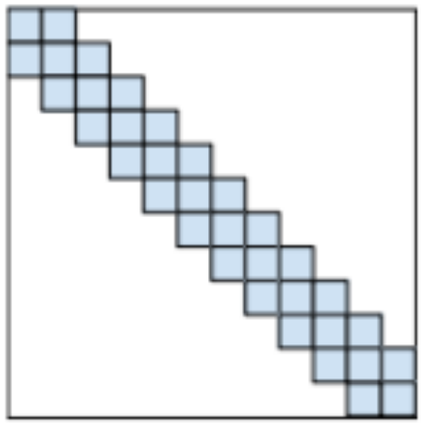
- "sliding window on the nodes"
  - random subset (k%), other (100 - k)%
- Operation
  - width w, query at location i
  - attends from i − w/2 to i + w/2 keys
  - A(i, i−w/2 : i+w/2) = 1

# BIGBIRD Architecture

- sparse random graph for attention mechanism should have two desiderata

2. **locality of reference**

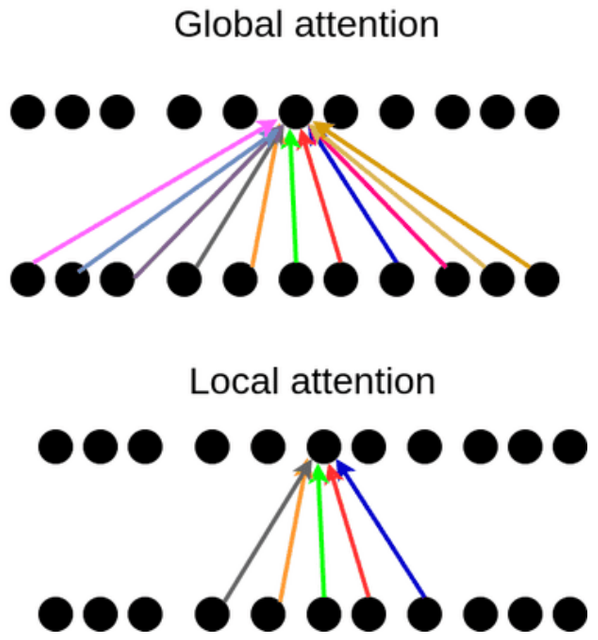
- basic experiments
- random blocks and local window were insufficient



(b) Window attention

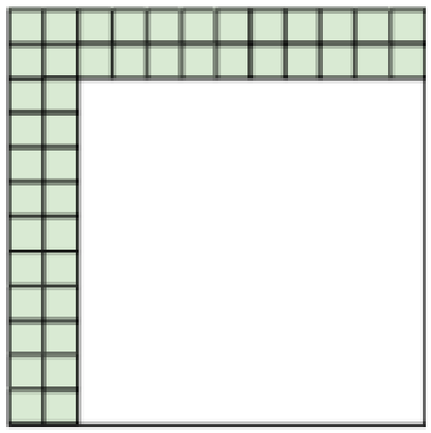
| Model | MLM | SQuAD | MNLI |
|---|---|---|---|
| BERT-base | 64.2 | 88.5 | 83.4 |
| Random (R) | 60.1 | 83.0 | 80.2 |
| Window (W) | 58.3 | 76.4 | 73.1 |
| R + W | 62.7 | 85.1 | 80.5 |

Table 1: Building block comparison @512

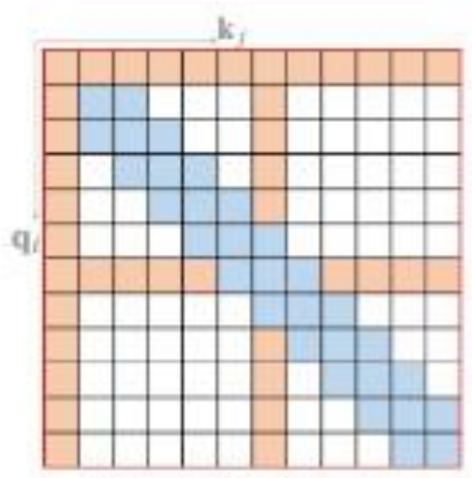


# BIGBIRD Architecture

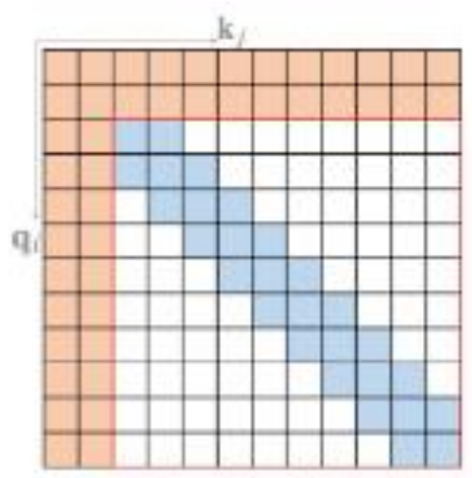
- Global Tokens
  1. BIGBIRD-ITC: In internal transformer construction (ITC)
     - make some existing tokens “global”, which attend over the entire sequence
  2. BIGBIRD-ETC: In extended transformer construction (ETC)
     - include additional “global” tokens such as CLS



(c) Global Attention



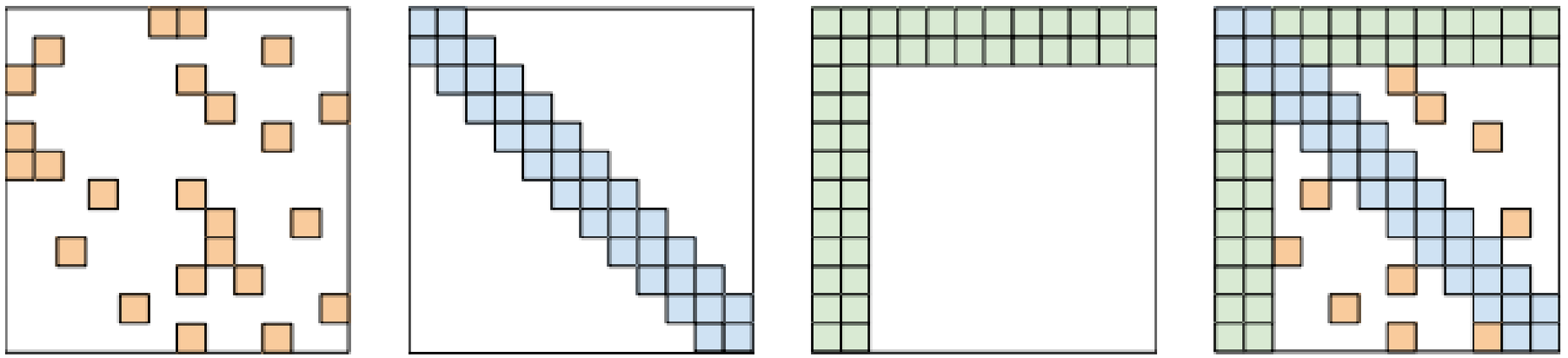

(b) Longformer




(c) ETC

# BIGBIRD Architecture

- final attention mechanism for BIGBIRD
  - queries attend to r random keys,
  - each query attends to w/2 tokens to the left of its location and w/2 to the right of its location
  - they contain g global tokens

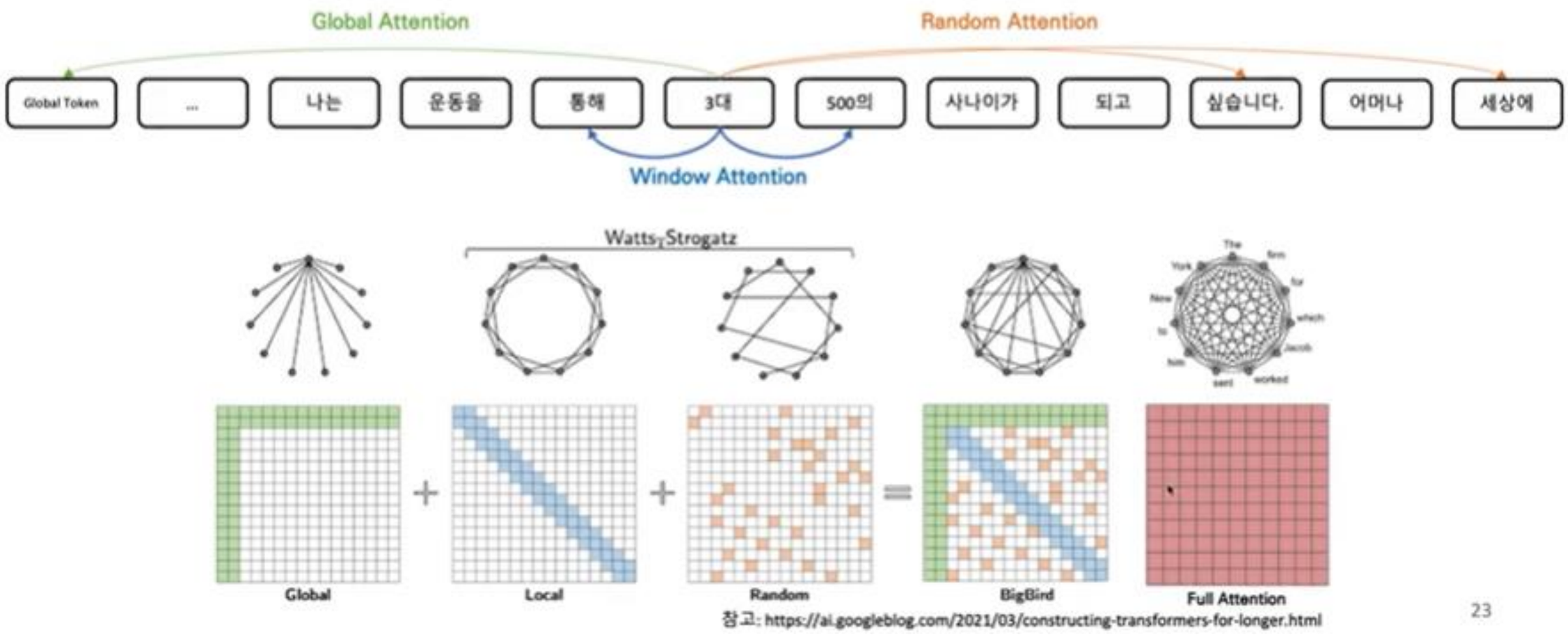


(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Figure 1: Building blocks of the attention mechanism used in BIGBIRD. White color indicates absence of attention. (a) random attention with $r = 2$, (b) sliding window attention with $w = 3$ (c) global attention with $g = 2$. (d) the combined BIGBIRD model.

# BIGBIRD Architecture

- final attention mechanism for BIGBIRD




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

23

# BIGBIRD Architecture

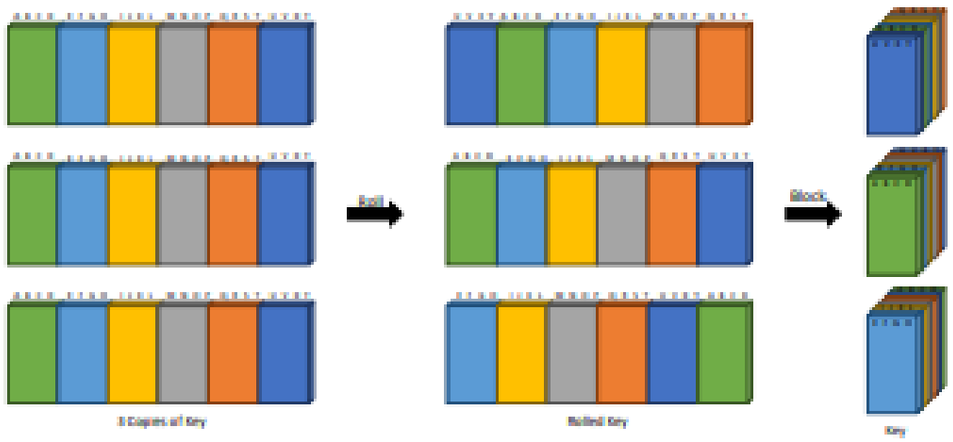


Figure 5: Construction of rolled key-block tensor. Make $w$ copies of the key matrix. Index the copies as $-(w-1)/2 \leq j \leq (w-1)/2$. Roll $j^{\text{th}}$ copy by $j$ blocks. Positive roll means circular shift entries left and likewise for negative roll corresponds to right shift. Finally, reshape by grouping the blocks along a new axis to obtain the key-blocked tensor. For illustration purpose $w = 3$ is chosen.

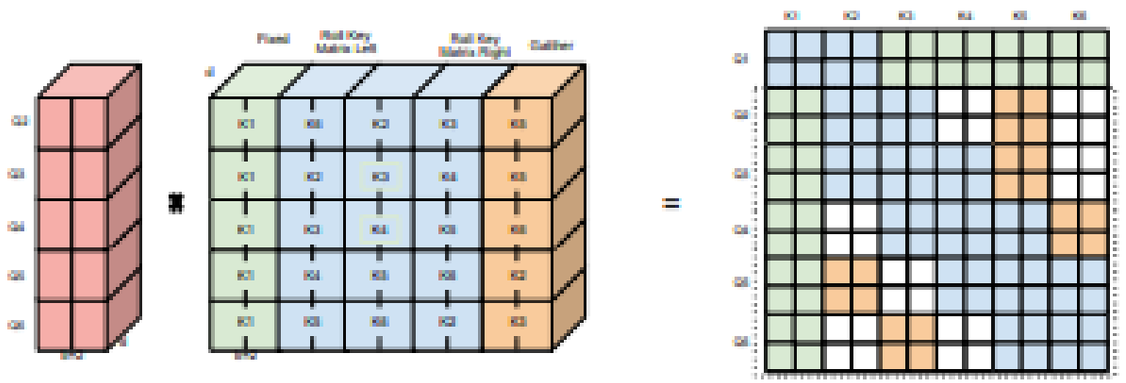


Figure 6: Overview of BIGBIRD attention computation. Structured block sparsity helps in compactly packing our operations of sparse attention, thereby making our method efficient on GPU/TPU. On the left, we depict the transformed dense query and key tensors. The query tensor is obtained by simply blocking and reshaping while the final key tensor by concatenating three transformations: The first green columns, corresponding to global attention, is fixed. The middle blue columns correspond to window local attention and can be obtained by appropriately rolling as illustrated in Fig. 5. For the final orange columns, corresponding to random attentions, we need to use computationally inefficient gather operation. Dense multiplication between the query and key tensors efficiently calculates the sparse attention pattern (except the first row-block, which is computed by direct multiplication), using the ideas illustrated in Fig. 4. The resultant matrix on the right is same as that shown in Fig. 3d.

# Theoretical Results about Sparse Attention Mechanism

- https://velog.io/@gyfla1512/Big-Bird-Transformers-for-Longer-Sequences

- sparse attention mechanisms -> Universal Approximators of sequence to sequence functions in the style of Yun et al. [104]
- sparse encoder-decoder transformers are Turing Complete
- moving to a sparse-attention mechanism incurs a cost

# Experiments: Natural Language Processing

- 1. masked language modeling (MLM; Devlin et al. 22)
  - to check if better contextual representations can be learnt by utilizing longer contiguous sequences.
- 2. Question Answering(QA, with supporting evidence)
  - to handle longer sequence would allow us to retrieve more evidence using crude systems like TF-IDF/BM25
- 3. long document classification
  - where discriminating information may not be located in first 512 tokens
- using sequence length 4096

# Experiments: Natural Language Processing

- 1. Pretraining and MLM

| Parameter | BIGBIRD-ITC | BIGBIRD-ETC |
|---|---|---|
| Block length, $b$ | 64 | 84 |
| # of global token, $g$ | $2 \times b$ | 256 |
| Window length, $w$ | $3 \times b$ | $3 \times b$ |
| # of random token, $r$ | $3 \times b$ | 0 |
| Max. sequence length | 4096 | 4096 |
| # of heads | 12 | 12 |
| # of hidden layers | 12 | 12 |
| Hidden layer size | 768 | 768 |
| Batch size | 256 | 256 |
| Loss | MLM | MLM |
| Activation layer | gelu | gelu |
| Dropout prob | 0.1 | 0.1 |
| Attention dropout prob | 0.1 | 0.1 |
| Optimizer | Adam | Adam |
| Learning rate | $10^{-4}$ | $10^{-4}$ |
| Compute resources | $8 \times 8$ TPUv3 | $8 \times 8$ TPUv3 |

Table 8: Hyperparameters for the two BIGBIRD base models for MLM.

# Experiments: Natural Language Processing

- 1. Pretraining and MLM
  - Task: predicting a random subset of tokens which have been masked out.
  - four standard data-sets for pretraining(Tab 9)
  - compare performance in predicting the masked out tokens in terms of **bits per character**(Tab 10)

| Dataset | # tokens | Avg. doc len. |
|---|---|---|
| Books [110] | 1.0B | 37K |
| CC-News [34] | 7.4B | 561 |
| Stories [89] | 7.7B | 8.2K |
| Wikipedia | 3.1B | 592 |

Table 9: Dataset used for pre training.

| Model | Base | Large |
|---|---|---|
| RoBERTa (sqln: 512) | 1.846 | 1.496 |
| Longformer (sqln: 4096) | 1.705 | 1.358 |
| BIGBIRD-ITC (sqln: 4096) | 1.678 | 1.456 |
| BIGBIRD-ETC (sqln: 4096) | **1.611** | **1.274** |

Table 10: MLM performance on held-out set.

# Experiments: Natural Language Processing

- 2. Question Answering(QA)

| | Instances | | Instance Length | |
|---|---|---|---|---|
| Dataset | Training | Dev | Median | Max |
| HotpotQA-distractor [100] | 90447 | 7405 | 1227 | 3560 |
| Natural Questions [52] | 307373 | 7830 | 3258 | 77962 |
| TriviaQA [41] | 61888 | 7993 | 4900 | 32755 |
| WikiHop [95] | 43738 | 5129 | 1541 | 20337 |

Table 11: Question Answering Datasets

| Parameter | HotpotQA | | NaturalQ | | TriviaQA | | WikiHop | |
|---|---|---|---|---|---|---|---|---|
| Global token location | ITC | ETC | ITC | ETC | ITC | ETC | ITC | ETC |
| # of global token, $g$ | 128 | 256 | 128 | 230 | 128 | 320 | 128 | 430 |
| Window length, $w$ | 192 | 252 | 192 | 252 | 192 | 252 | 192 | 252 |
| # of random token, $r$ | 192 | 0 | 192 | 0 | 192 | 0 | 192 | 0 |
| Max. sequence length | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 | 4096 |
| # of heads | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| # of hidden layers | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| Hidden layer size | 768 | 768 | 768 | 768 | 768 | 768 | 768 | 768 |
| Batch size | 32 | 32 | 128 | 128 | 32 | 32 | 64 | 64 |
| Loss | cross-entropy golden spans | | cross-entropy golden spans | | cross-entropy noisy spans [18] | | cross-entropy ans choices | |
| Compute resources | $4 \times 2$ TPUv3 | | $4 \times 8$ TPUv3 | | $4 \times 2$ TPUv3 | | $4 \times 4$ TPUv3 | |

Table 12: Hyperparameters of base BIGBIRD model used for Question Answering i.e. the numbers reported in Tab. 2

# Experiments: Natural Language Processing

- 2. Question Answering(QA)
  - Natural Questions [52], HotpotQA-distractor [100], TriviaQA-wiki [41], WikiHop [95]
  - Experiment result (Tab 2)

| Model | HotpotQA | | | NaturalQ | | TriviaQA | WikiHop |
|---|---|---|---|---|---|---|---|
| | Ans | Sup | Joint | LA | SA | Full | MCQ |
| RoBERTa | 73.5 | 83.4 | 63.5 | - | - | 74.3 | 72.4 |
| Longformer | 74.3 | 84.4 | 64.4 | - | - | 75.2 | 75.0 |
| BIGBIRD-ITC | **75.7** | 86.8 | 67.7 | 70.8 | 53.3 | **79.5** | **75.9** |
| BIGBIRD-ETC | 75.5 | **87.1** | **67.8** | **73.9** | **54.9** | 78.7 | **75.9** |

Table 2: QA Dev results using Base size models. We report accuracy for WikiHop and F1 for HotpotQA, Natural Questions, and TriviaQA.

# Experiments: Natural Language Processing

- 2. Question Answering(QA)
  - Experiment result (Tab 3)
    - compare to top-3 entries from the leaderboard

| 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** |

Table 3: Fine-tuning results on **Test** set for QA tasks. The Test results (F1 for HotpotQA, Natural Questions, TriviaQA, and Accuracy for WikiHop) have been picked from their respective leaderboard. For each task the top-3 leaders were picked not including BIGBIRD-etc. **For Natural Questions Long Answer (LA), TriviaQA, and WikiHop, BIGBIRD-ETC is the new state-of-the-art**. On HotpotQA we are third in the leaderboard by F1 and second by Exact Match (EM).

# Experiments: Natural Language Processing

- 3. Classification
  - Experiments: datasets of different lengths and contents, specifically various document classification and GLUE tasks.

| Parameter | IMDb | Arxiv | Patents | Hyperpartisan | Yelp-5 |
|---|---|---|---|---|---|
| Batch size | 64 | 64 | 64 | 32 | 32 |
| Learning rate | $1 \times 10^{-5}$ | $3 \times 10^{-5}$ | $5 \times 10^{-5}$ | $5 \times 10^{-6}$ | $2 \times 10^{-5}$ |
| Num epochs | 40 | 10 | 3 | 15 | 2 |
| TPUv3 slice | $4 \times 4$ | $4 \times 4$ | $4 \times 4$ | $4 \times 2$ | $4 \times 8$ |
| # of heads | 12 | | | | 16 |
| # of hidden layers | 12 | | | | 24 |
| Hidden layer size | 768 | | | | 1024 |
| Block length, $b$ | 64 | | | | |
| Global token location | ITC | | | | |
| # of global token, $g$ | $2 \times b$ | | | | |
| Window length, $w$ | $3 \times b$ | | | | |
| # of random token, $r$ | $3 \times b$ | | | | |
| Max. sequence length | 4096 | | | | |
| Vocab size | 50358 | | | | |
| Activation layer | gelu | | | | |
| Dropout prob | 0.1 | | | | |
| Attention dropout prob | 0.1 | | | | |
| Loss | cross-entropy | | | | |
| Optimizer | Adam | | | | |

Table 14: Hyperparameters for document classification.

# Experiments: Natural Language Processing

- 3. Classification
  - Experiments: datasets of different lengths and contents, specifically various document classification and GLUE tasks.

| Model | IMDb [64] | Yelp-5 [108] | Arxiv [35] | Patents [53] | Hyperpartisan [47] |
|---|---|---|---|---|---|
| # Examples | 25000 | 650000 | 30043 | 1890093 | 645 |
| # Classes | 2 | 5 | 11 | 663 | 2 |
| Excess fraction | 0.14 | 0.04 | 1.00 | 0.90 | 0.53 |
| SoTA | [88] 97.4 | [3] 73.28 | [69] 87.96 | [69] 69.01 | [40] 90.6 |
| RoBERTa | $95.0 \pm 0.2$ | 71.75 | 87.42 | 67.07 | $87.8 \pm 0.8$ |
| BIGBIRD | $95.2 \pm 0.2$ | 72.16 | **92.31** | 69.30 | $\mathbf{92.2 \pm 1.7}$ |

Table 15: Classification results. We report the F1 micro-averaged score for all datasets. Experiments on smaller IMDb and Hyperpartisan datasets are repeated 5 times and the average performance is presented along with standard deviation.

| System | MNLI-(m/mm) 392k | QQP 363k | QNLI 108k | SST-2 67k | CoLA 8.5k | STS-B 5.7k | MRPC 3.5k | RTE 2.5k |
|---|---|---|---|---|---|---|---|---|
| BERT | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 |
| XLNet | 86.8/- | 91.4 | 91.7 | 94.7 | 60.2 | 89.5 | 88.2 | 74.0 |
| RoBERTa | 87.6/- | 91.9 | 92.8 | 94.8 | 63.6 | 91.2 | 90.2 | 78.7 |
| BIGBIRD | 87.5/87.3 | 88.6 | 92.2 | 94.6 | 58.5 | 87.8 | 91.5 | 75.0 |

Table 16: GLUE Dev results on base sized models. Number of training examples is reported below each task. MCC score is reported for CoLA, F1 score is reported for MRPC, Spearman correlation is reported for STS-B, and accuracy scores are reported for the other tasks.

# Experiments: Natural Language Processing

- 4. Encoder-Decoder Tasks
  - Suffering: quadratic complexity due to the full self attention
    - the sparse attention mechanism of BIGBIRD only at the encoder side
  - Experiment
    - realistic scenarios: that the median output sequence length is ~ 200 whereas the input sequence's median length is > 3000.
    - sparse attention -> encoder, full self-attention -> decoder.

| | Instances | | | Input Length | | Output Length | |
|---|---|---|---|---|---|---|---|
| Dataset | Training | Dev | Test | Median | 90%-ile | Median | 90%-ile |
| Arxiv [20] | 203037 | 6436 | 6440 | 6151 | 14405 | 171 | 352 |
| PubMed [20] | 119924 | 6633 | 6658 | 2715 | 6101 | 212 | 318 |
| BigPatent [78] | 1207222 | 67068 | 67072 | 3082 | 7693 | 123 | 197 |

Table 18: Statistics of datasets used for summarization.

# Experiments: Natural Language Processing

- 5. Summarization
  - using, three long document datasets
  - focus on abstractive summarization of long documents where using a longer contextual encoder should improve performance.
    1. the salient content can be evenly distributed in the long document, not just in first 512 tokens(BigPatents dataset [78]).
    2. longer documents exhibit a richer discourse structure and summaries are considerably more abstractive

# Experiments: Natural Language Processing

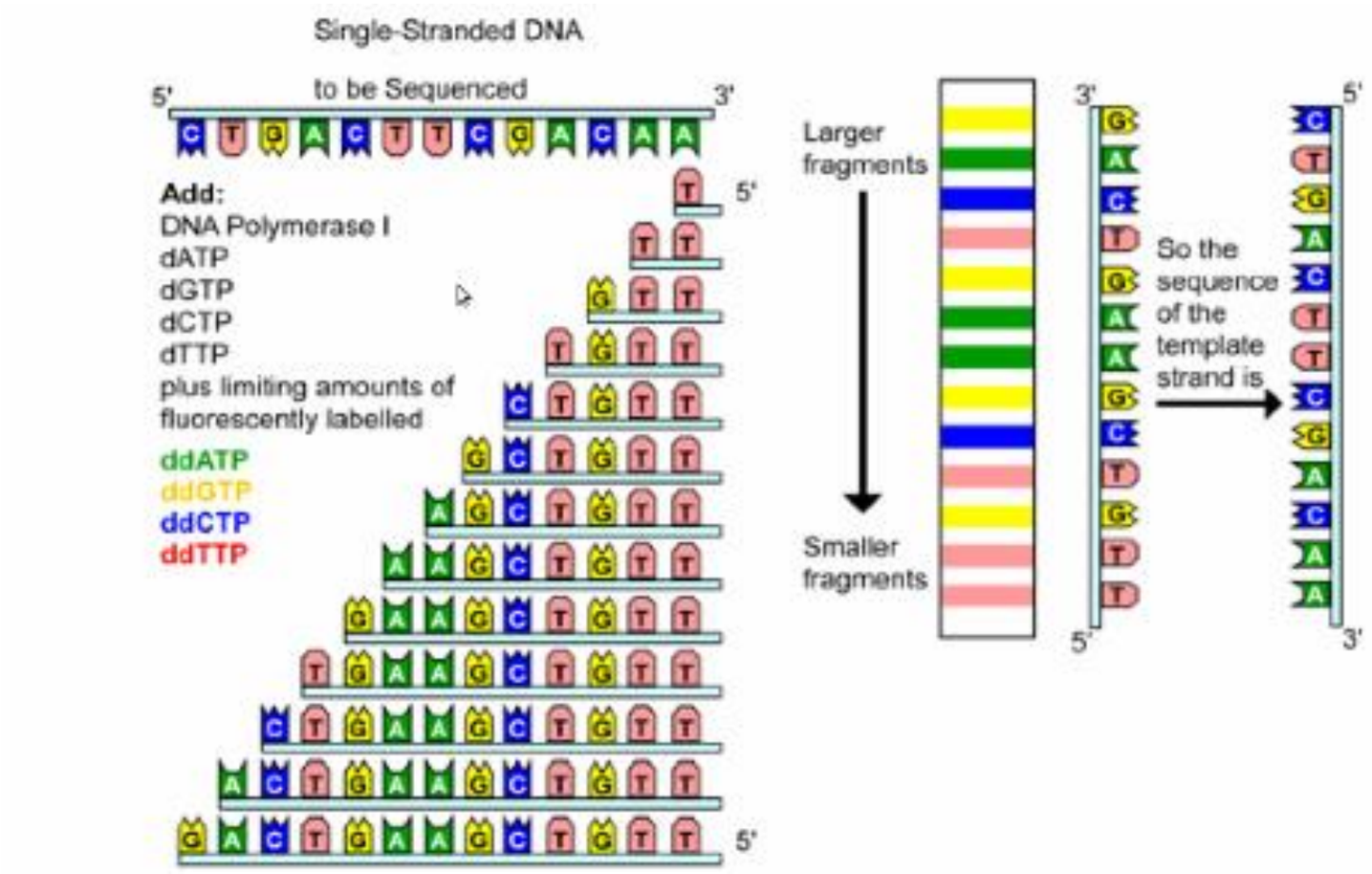
- 5. Summarization
  - ROUGE(Recall-Oriented Understudy for Gisting Evaluation) Score

| | 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 |
| | Dancer [31] | 42.70 | 16.54 | 38.44 | 44.09 | 17.69 | 40.27 | - | - | - |
| Base | 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 |
| | BIGBIRD-RoBERTa | 41.22 | 16.43 | 36.96 | 43.70 | 19.32 | 39.99 | 55.69 | 37.27 | 45.56 |
| Large | 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.

# Experiments: Genomics

- 5. Genomics
  - recent upsurge in using deep learning for genomics data [86, 106, 13]
  - Inputs: DNA sequence fragments



# Experiments: Genomics

- 5. Genomics
  - recent upsurge in using deep learning for genomics data [86, 106, 13]
  - Inputs: DNA sequence fragments
    - longer input sequence handling capability of BIGBIRD would be beneficial as many functional effects in DNA are highly non-local [12]
- MLM pretraining: taking inspiration from NLP, we learn powerful contextual representations for DNA fragments utilizing abundant unlabeled data
- long input BIGBIRD along with the proposed pretraining significantly improves performances in two downstream tasks

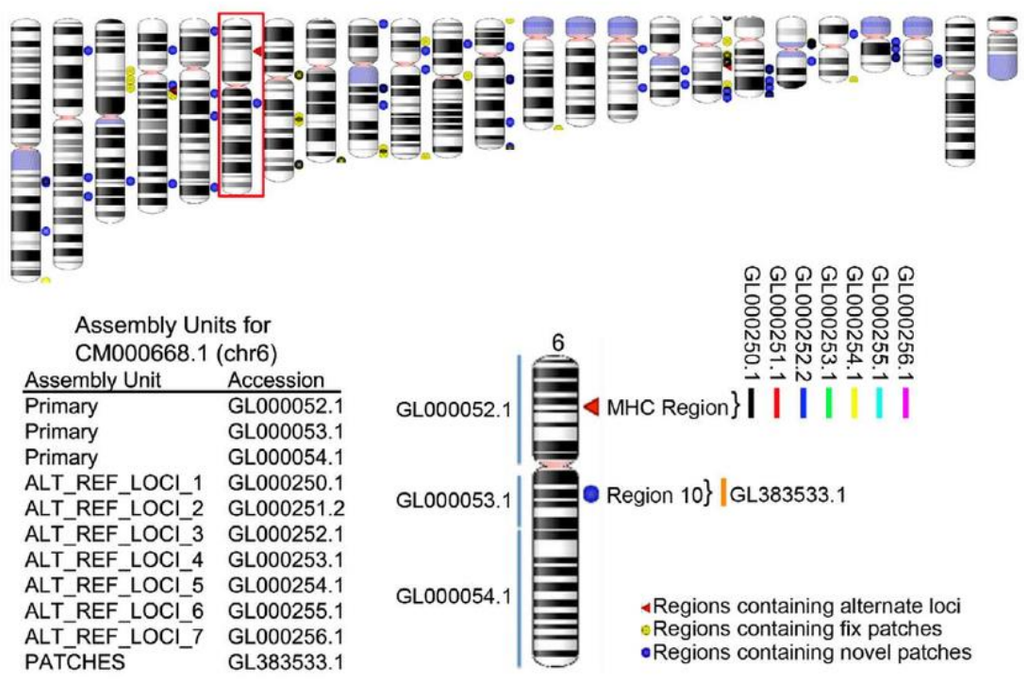
# Experiments: Genomics

- 5. Genomics
  - 1. Pre-training and MLM
    - Liang [58], first segment DNA into tokens so as to further increase the context length
      - raw DNA sequence: GRCh37


Figure 7: Visual description of how the masked language modeling data was generated from raw DNA dataset. The raw DNA sequences of GRCh37, where split at random positions to create documents with 50-100 sentences where each sentence was 500-1000 base pairs (bps). Thus each document had a continuous strand of 25000-100,000 bps of DNA. This process was repeated 10 times to create 10 sets of document for each chromosome of GRCH37. The resulting set of documents was then passed through Sentencepiece that created tokens of average 8bp. For pretraining we used masked language model and masked 10% of the tokens and trained on predicting the masked tokens.

# Experiments: Genomics

- 5. Genomics
  - 1. Pre-training and MLM
    - Liang [58], first segment DNA into tokens so as to further increase the context length
    - raw DNA sequence: GRCh37



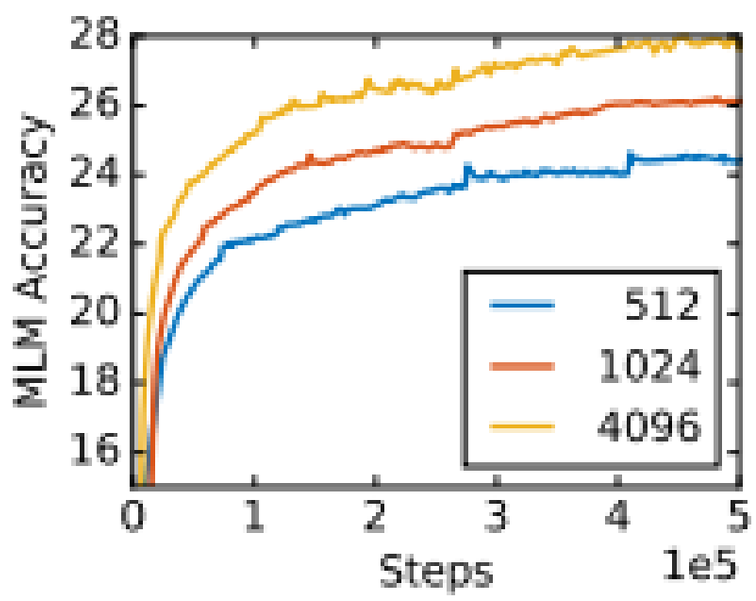




Figure 8: BIGBIRD accuracy with context length.

# Experiments: Genomics

- 5. Genomics
  1. Pre-training and MLM
     - Experiments Result
     - bits per character (BPC)

| Model | BPC |
|---|---|
| SRILM [58] | 1.57 |
| BERT (sqln. 512) | 1.23 |
| BIGBIRD (sqln. 4096) | **1.12** |

Table 5: MLM BPC

# Experiments: Genomics

- 5. Genomics
  2. Promoter Region Prediction
  - Promoter: DNA region typically located upstream of the gene
    - Multiple methods have been proposed to identify the promoter regions in a given DNA sequence.
  - The corresponding machine learning task is to classify a given DNA fragment as promoter or non-promoter sequence.
  - Dataset: Oubounyt et al. [71]

| Model | F1 |
|---|---|
| CNNProm [90] | 69.7 |
| DeePromoter [71] | 95.6 |
| BIGBIRD | **99.9** |

Table 6: Comparison.

# Experiments: Genomics

- 5. Genomics
  - 3. Chromatin-Profile Prediction

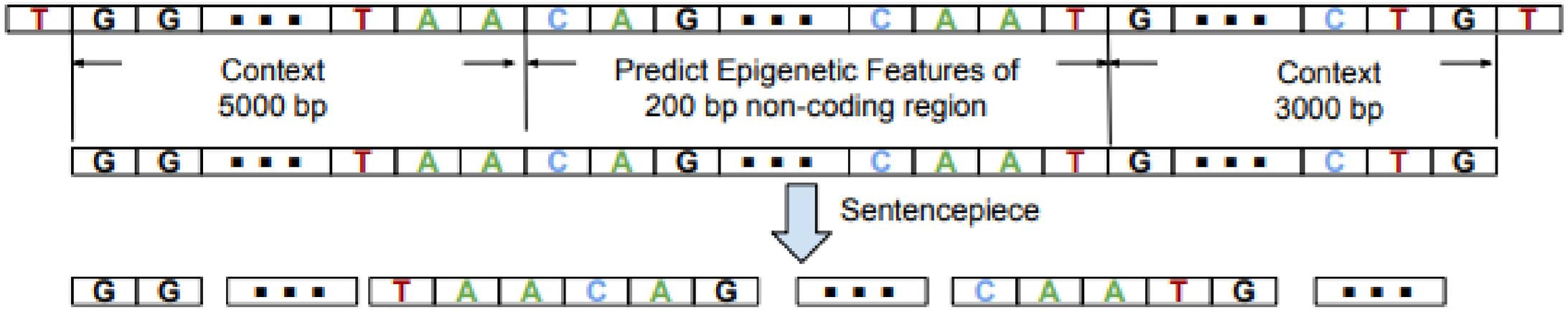



Figure 9: Visual description of the DNA segment from which we predict the chromatin profile for a given non-coding region of the raw DNA sequences of GRCh37. We take 8000 bps of DNA before and after the given non-coding region as context. The complete fragment of DNA including the context on both side, is then tokenized to form our input sequence of tokens. The task is to predict 919 chromatin profile including 690 transcription factors (TF) binding profiles for 160 different TFs, 125 DNase I sensitivity (DHS) profiles and 104 histone-mark (HM) profiles

# Experiments: Genomics

- 5. Genomics

  3. Chromatin-Profile Prediction

  - understanding the functional effects of non-coding regions of DNA is a very important task
  - Method: We jointly learn 919 binary classifiers to predict these functional effects from sequence of DNA fragments.

| Model | TF | HM | DHS |
|---|---|---|---|
| gkm-SVM [30] | 89.6 | - | - |
| DeepSea [109] | 95.8 | 85.6 | **92.3** |
| BIGBIRD | **96.1** | **88.7** | 92.1 |

Table 7: Chromatin-Profile Prediction

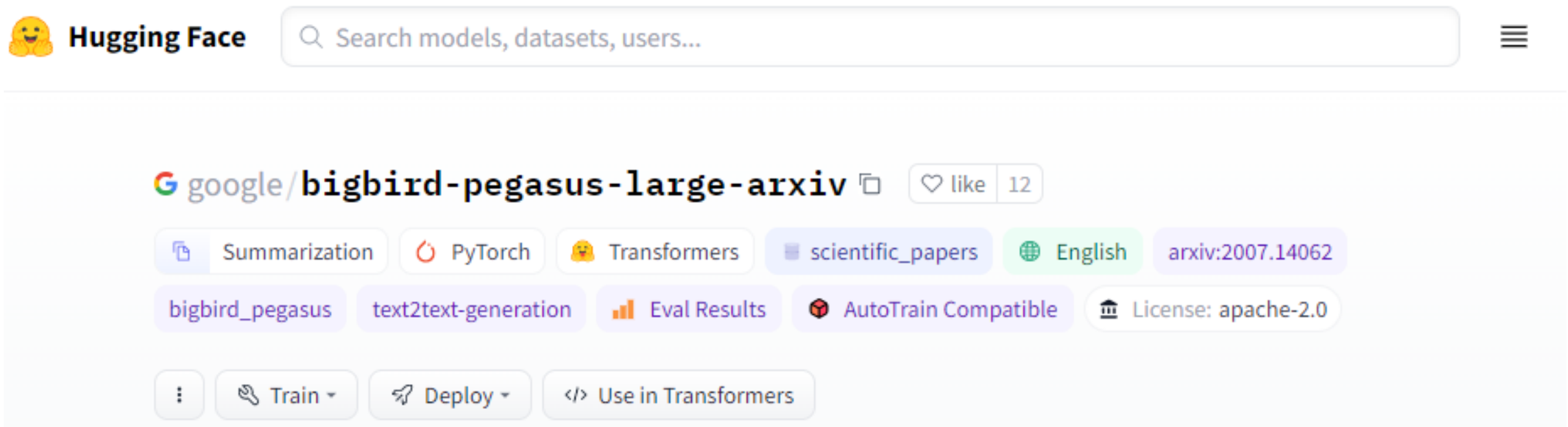
# Conclusion

- BIGBIRD: a sparse attention mechanism that is linear in the number of tokens
  - universal approximator of sequence to sequence functions
  - is also Turing complete.
- Method
  - use the power of extra global tokens preserve the expressive powers of the model.
- BIGBIRD gives state-of-the-art performance on a number of NLP tasks such as question answering and long document classification
- introduce attention based contextual language model for DNA and fine-tune it for down stream tasks

# BIGBIRD_ Hugging Face

- https://huggingface.co/transformers/v4.7.0/model_doc/bigbird.html#overview

- https://huggingface.co/google/bigbird-pegasus-large-arxiv/blob/main/README.md



# BIGBIRD_ Hugging Face

```python
from transformers import BigBirdPegasusForConditionalGeneration, AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("google/bigbird-pegasus-large-arxiv")

# by default encoder-attention is 'block_sparse' with num_random_blocks=3, block_size=64
model = BigBirdPegasusForConditionalGeneration.from_pretrained("google/bigbird-pegasus-large-arxiv")

# decoder attention type can't be changed & will be "original_full"
# you can change 'attention_type' (encoder only) to full attention like this:
model = BigBirdPegasusForConditionalGeneration.from_pretrained("google/bigbird-pegasus-large-arxiv",
                                                               attention_type="original_full")

# you can change 'block_size' & 'num_random_blocks' like this:
model = BigBirdPegasusForConditionalGeneration.from_pretrained("google/bigbird-pegasus-large-arxiv",
                                                               block_size=16, num_random_blocks=2)

text = "Replace me by any text you'd like."
inputs = tokenizer(text, return_tensors='pt')
prediction = model.generate(**inputs)
prediction = tokenizer.batch_decode(prediction)
```

In [13]:
```python
prediction
```

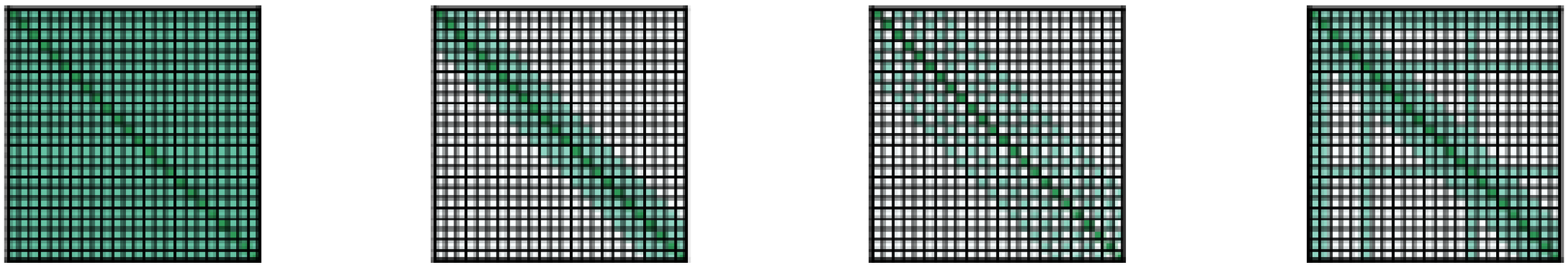
Out[13]:
```
['<s> we present a new method for the generation of non - abelian gauge fields from a non - abelian gauge field.<n> we show that the non - abelian gauge field can be efficiently generated from a non - abelian gauge field.<n> the method is based on the use of a non - linear coupling between the non - abelian and the abelian gauge field.<n> the non - abelian gauge field can be efficiently generated from a non - abelian gauge field. <n> @xmath0 non - abelian gauge field.<n> @xmath1 non - abelian gauge field.<n> @xmath2 non - abelian gauge field.<n> @xmath3 non - abelian gauge field.<n> @xmath4 non - abelian gauge field.<n> @xmath5 non - abelian gauge field.<n> @xmath6 non - abelian gauge field.<n> @xmath7 non - abelian gauge field.<n> @xmath8 non - abelian gauge field.<n> @xmath9 non - abelian gauge field.<n> @xmath10 non']
```

# Summary and Take-home message

- 좋은 점: 42쪽의 분량으로 인해서 어펜딕스가 풍부하다 !!! 그 덕에 궁금한 부분이 있다면 어펜딕스에서 그 어려움을 해결할 수 있다(다만, 수학적인 내용과 기본 지식을 알고 있다는 전제 하에 논문이 작성된 것 같아서 이해가 많이 어려울 수도 있다)
- 아쉬운 점: 내용이 어렵다ㅠㅠㅠ 예전에 Attention is all you need 논문을 3주 가량 공부했는데 그만큼 시간이 주어지지 못해 이해하지 못한 부분이 많다는 점에서 아쉬움이 남는다
- 흥미로운 점: Transformer가 NLP에 끼친 영향이 어마어마 하다는 것은 알고 있었지만, 정말 그 이상이구나 싶었고, 그로 인한 다양한 연구들이 진행되어왔다는 현행 상황들까지 살펴볼 수 있어서 좋았다 !!!

# Review

- Longformer: The Long-Document Transformer



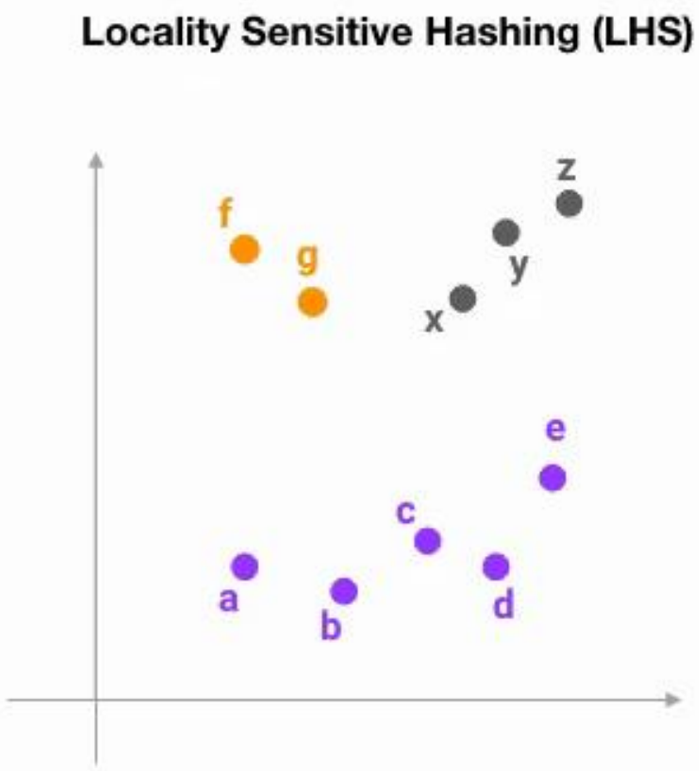
(a) Full $n^2$ attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

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

# Review

- Reformer: The Efficient Transformer
  - O(L log L)
  - LSH

  (Locality-Sensitive-Hashing)




| Point | Hash |
|---|---|
| a | |
| b | |
| c | |
| d | |
| e | |
| f | |
| g | |
| x | |
| y | |
| z | |

# Review

- Reformer: The Efficient Transformer

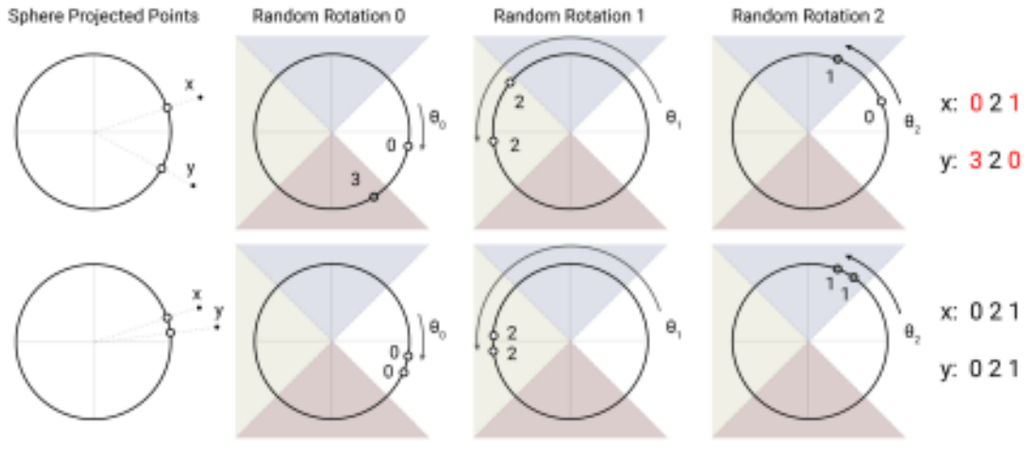



Figure 1: An angular locality sensitive hash uses random rotations of spherically projected points to establish buckets by an argmax over signed axes projections. In this highly simplified 2D depiction, two points $x$ and $y$ are unlikely to share the same hash buckets (above) for the three different angular hashes unless their spherical projections are close to one another (below).

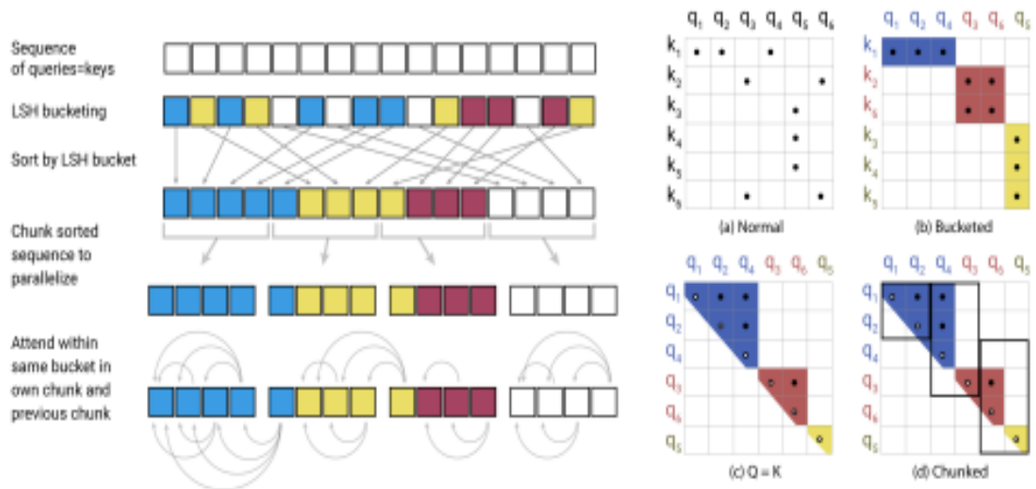



Figure 2: Simplified depiction of LSH Attention showing the hash-bucketing, sorting, and chunking steps and the resulting causal attentions. (a-d) Attention matrices for these varieties of attention.

# Review

- Extended Transformers Construction
- ETC: Encoding Long and Structured Inputs in Transformers

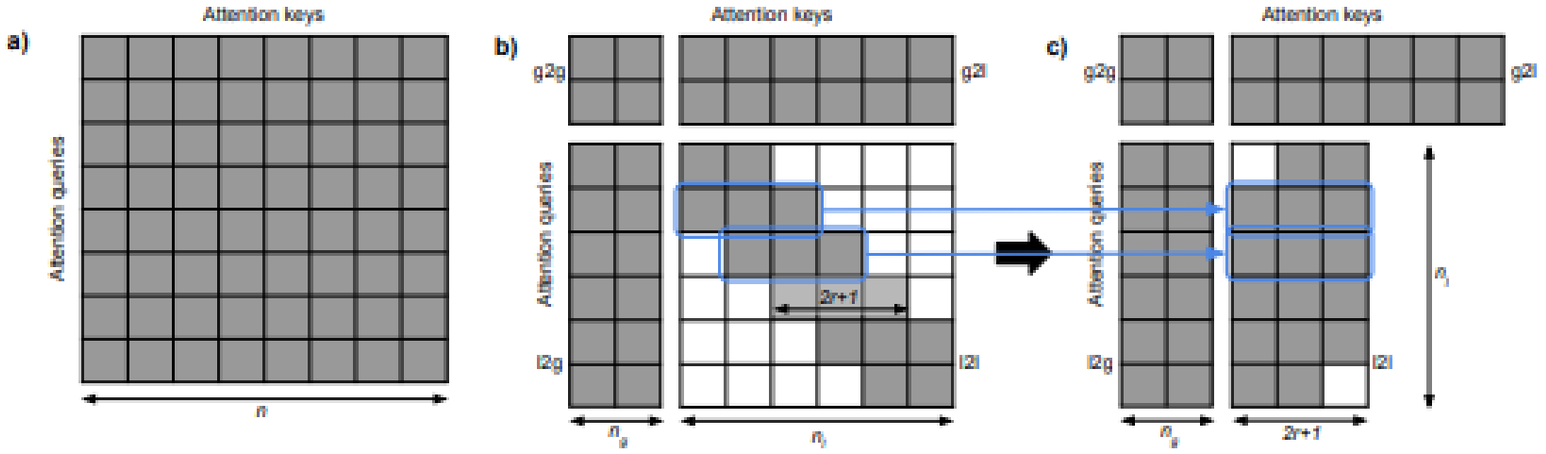



Figure 2: Sparsity diagram showing which attention queries (rows) can attend to which attention keys (columns) a) for standard Transformer attention with input size $n$; b) for global-local attention with input sizes $n_g$, $n_l$, and radius $r$; c) how the l2l attention piece is reshaped into a much smaller attention matrix, limited by local radius.

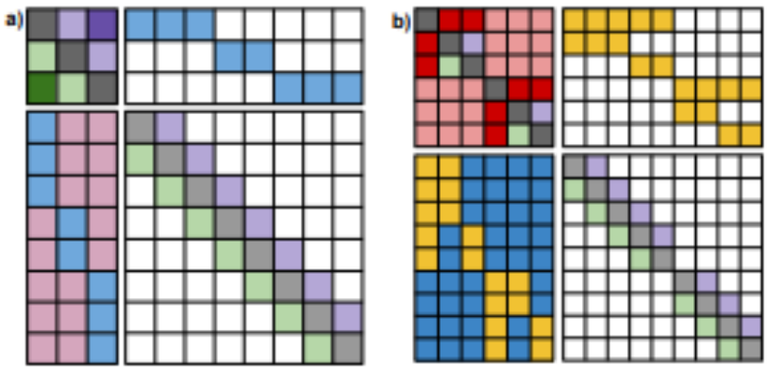



Figure 3: Example attention patterns for handling (a) long inputs and (b) structured inputs. White background means attention is masked via $M$, and the other colors indicate different relative position labels.

# Review

- LittleBird: Efficient Faster & Longer Transformer for Question Answering(2022, EMNLP)

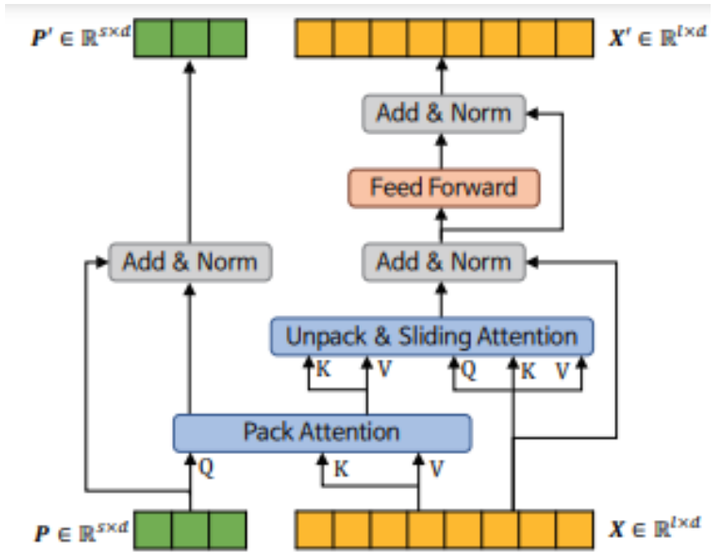



Figure 1: LittleBird Layer

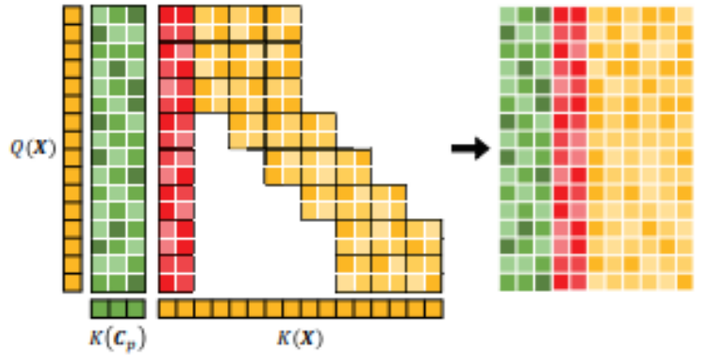



Figure 2: Unpack & Sliding Window Attention of LittleBird


DSAIL
Data Science Artificial Intelligence Lab.