Jump Self-attention: Capturing High-order Statistics in Transformers

Haoyi Zhou[1], Siyang Xiao[1] , Shanghang Zhang[2] , Jieqi Peng[1], Shuai Zhang[1], Jianxin Li[1*]

[1]Beihang University [2]Peking University

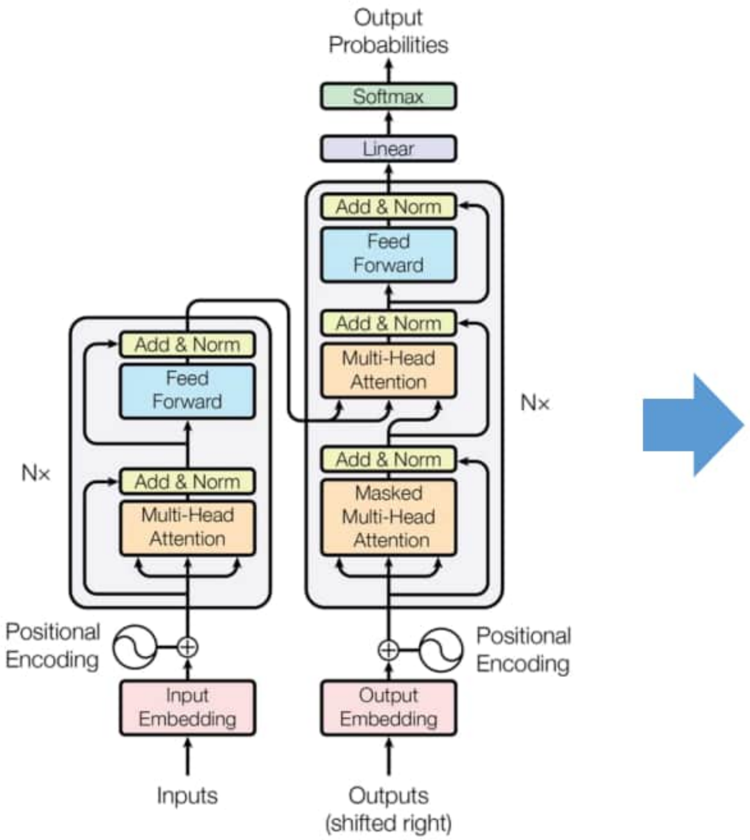
Content

- Motivation
- The Jump Self-attention
- Experimental Results
- Summary

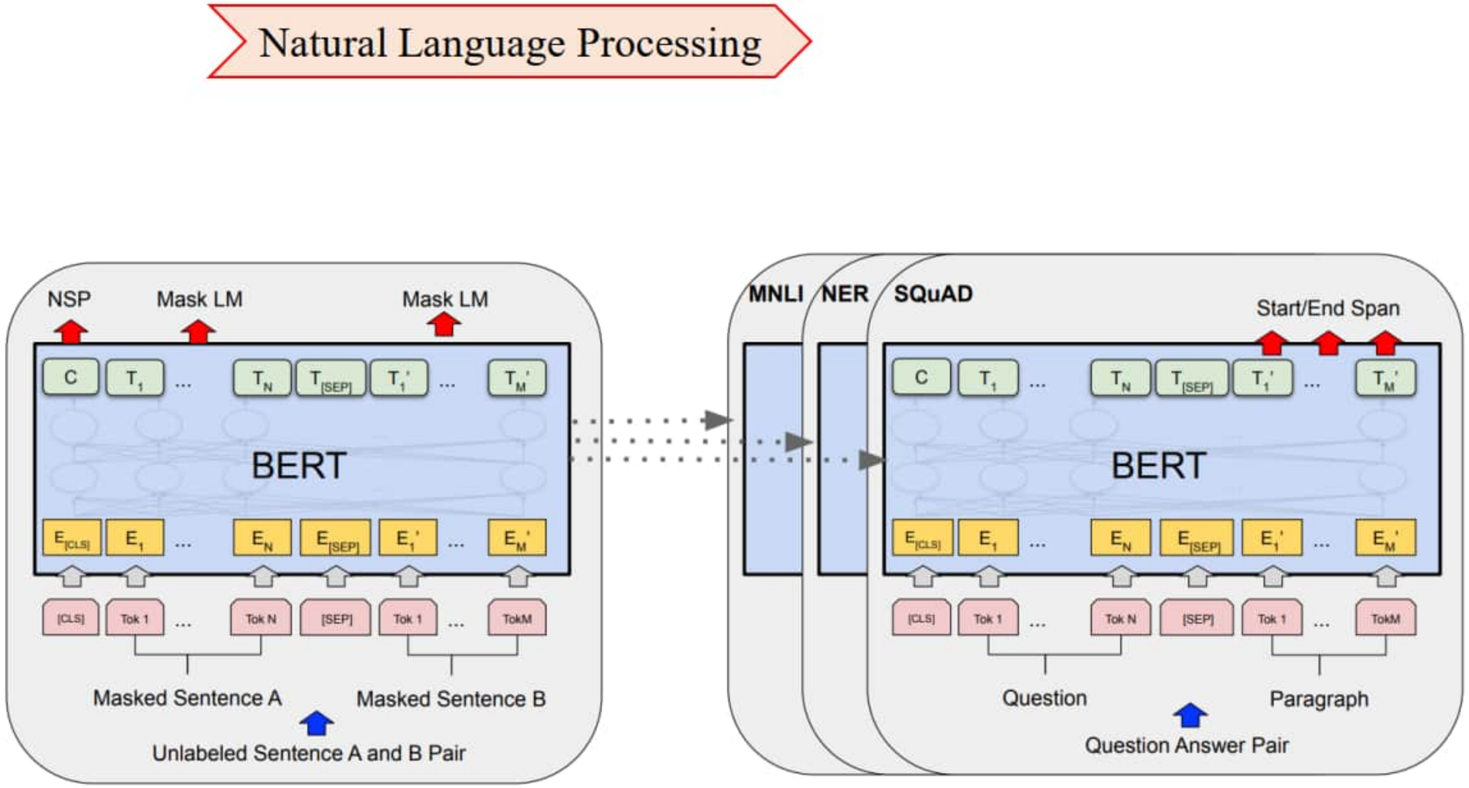
Content

- **Motivation**
- **The Jump Self-attention**
- **Experimental Results**
- **Summary**

Transformer models' success

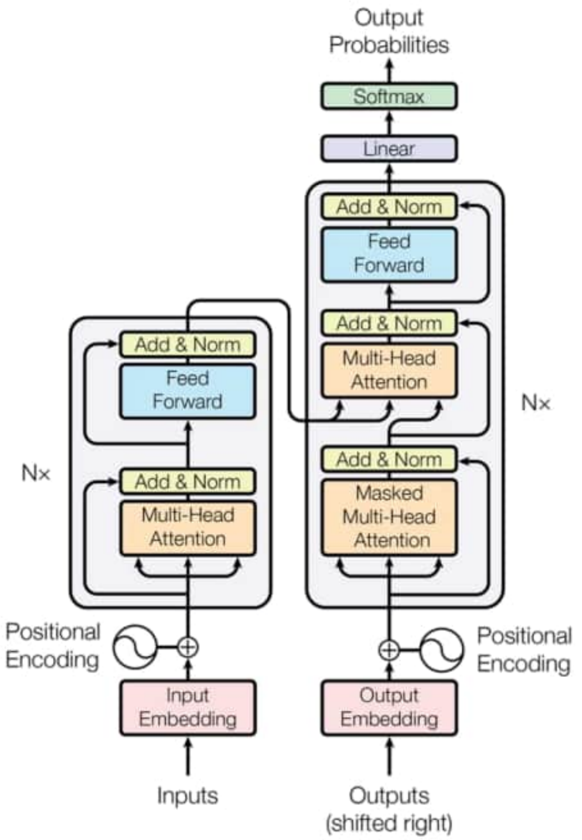


Transformer

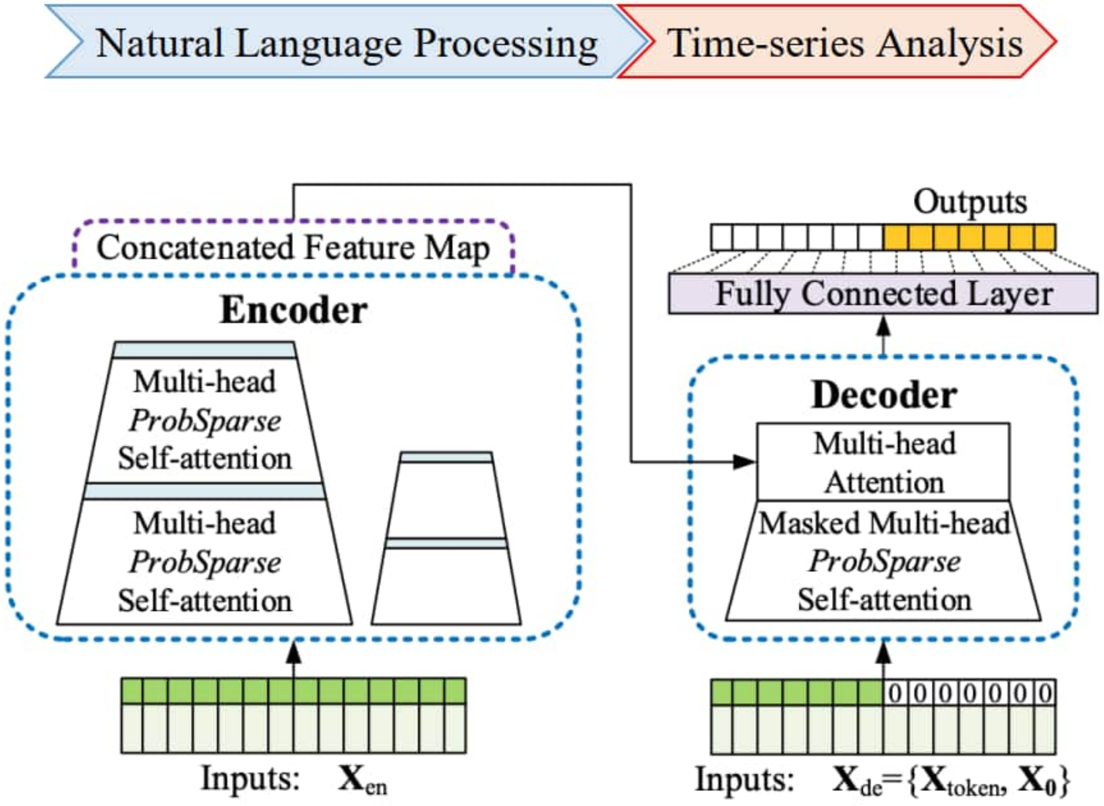


Bert

Transformer models’ success

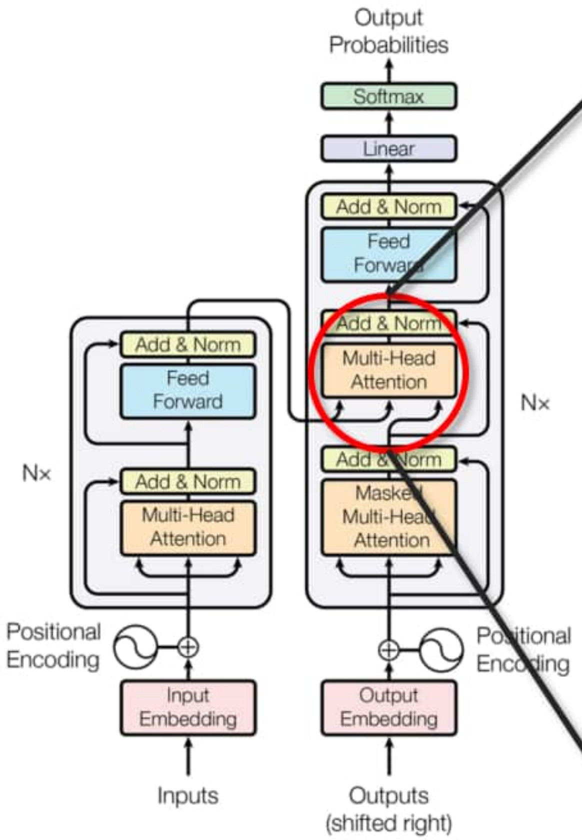


Transformer



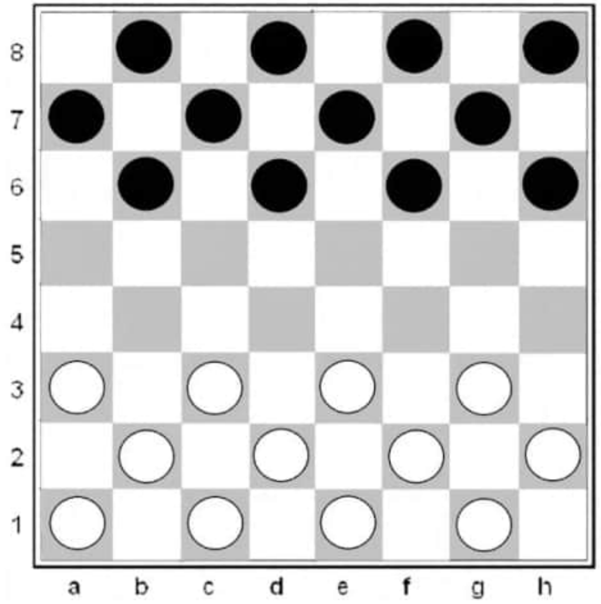
Informer

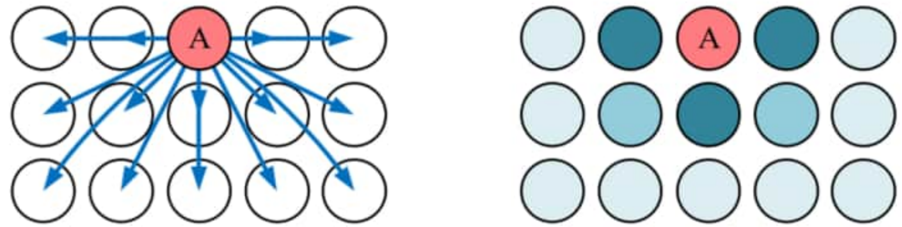
The core of Transformer models



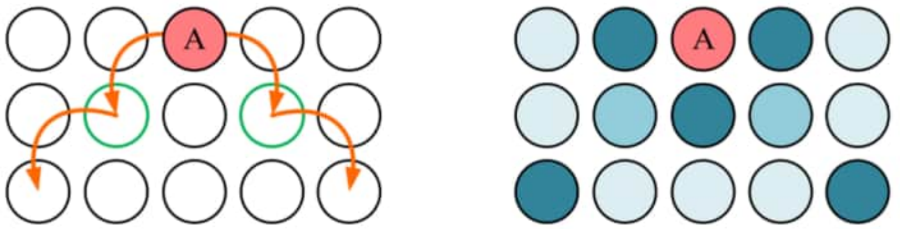
Transformer

Self-attention





(a) Dot-product attention with direct connections.



(b) High-order attention with "jump" connections.

Jump: The pieces moving from the game English Draught

Images are acquired from the Google search engine.

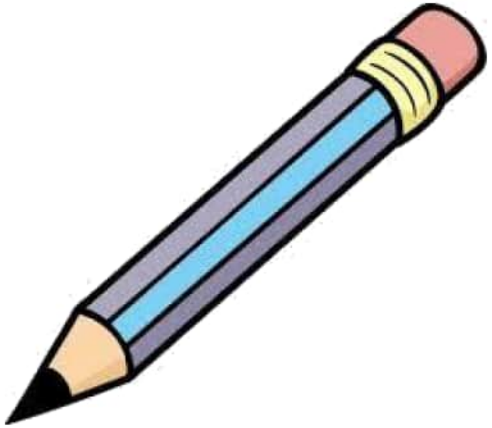
The high-order or Jump connections

"Book costs less than computer"

Book-Computer

Computer ? Pencil

Book-Pencil



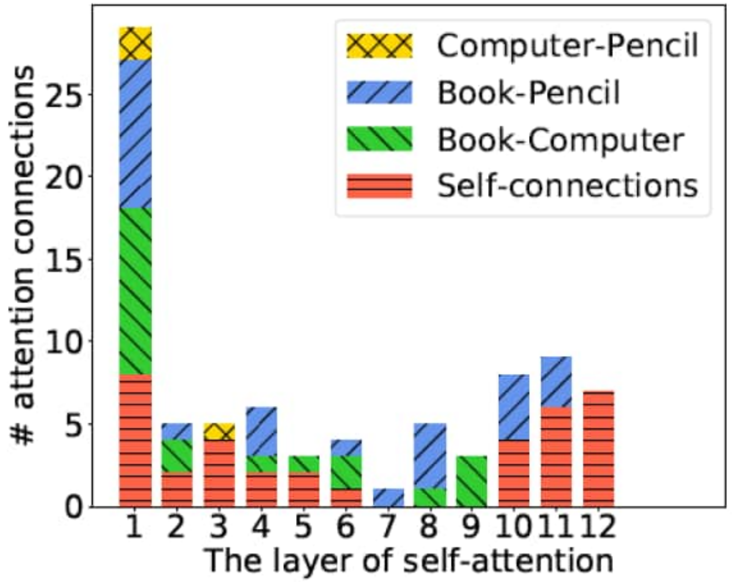
"Book costs more than pencil"

Images are acquired from the Google search engine.

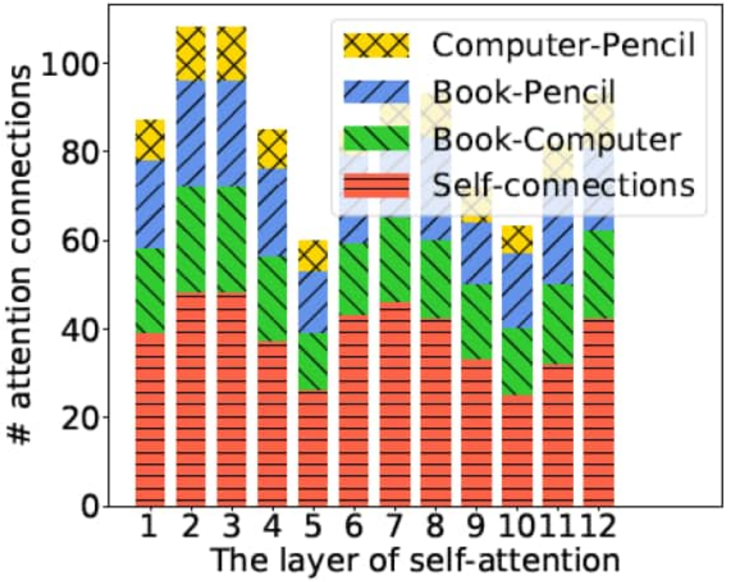
Rethinking the Canonical Self-attention

Model： BERT
with Pre-trained model

Computer Book Pencil



Significant connections (15.2%)

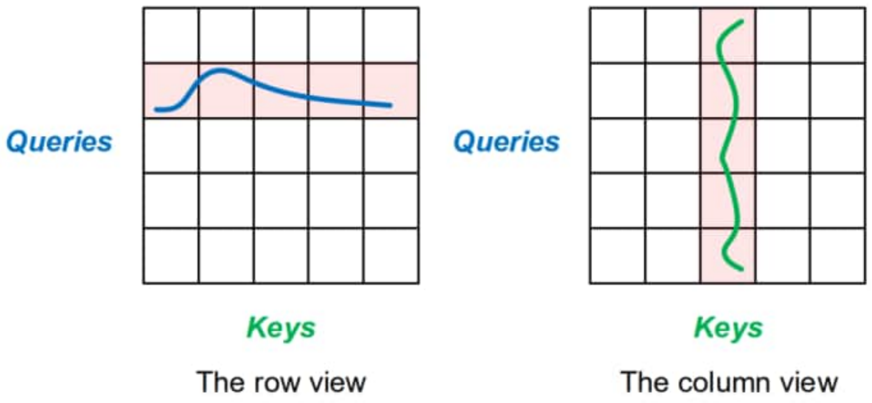


Major connections (70.5%)

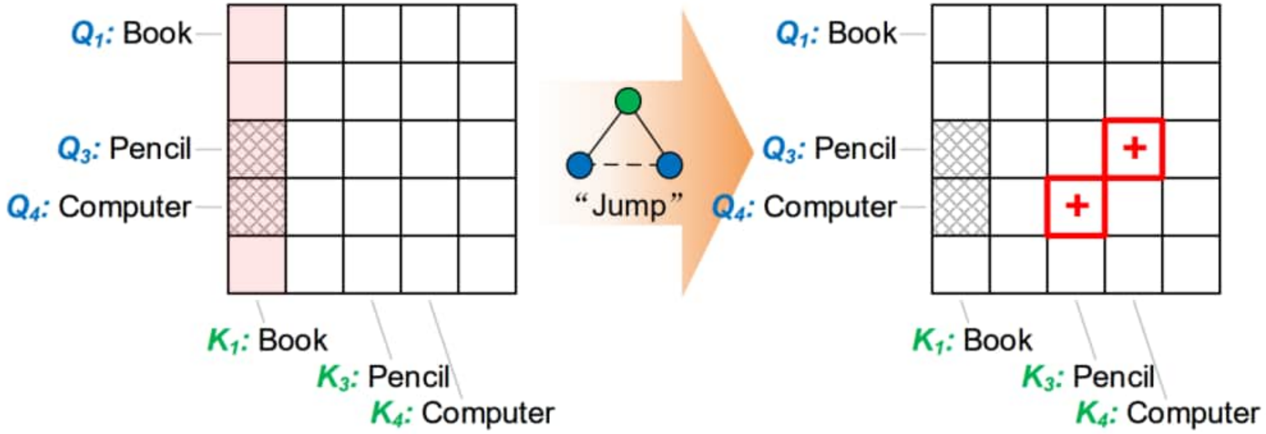
1. There are many self connections.
2. For the connections between "Computer-Pencil", it is much less than self connections.
3. The high order attention decrease with the layer stacking in the BERT model.

Content

The Jump Self-attention



(a) The different views.



(b) The overview of the high-order self-attention.

Under the row view, we build a graph G = (V, E). Queries compose the node set V with the row vector of S as node features, each edge is defined by a adjacency matrix A:

$$\mathbf{A}^{(\mathbf{K}_j)} = [\mathbf{U}_j - \mathrm{diag}(\mathbf{U})_j]_\rho, \quad \text{where} \quad \mathbf{U}_j = \frac{\mathbf{S}_{j^\top} \otimes \mathbf{S}_{j^\top}}{d}$$

$$\mathbf{A} = \frac{1}{L} \sum_j \mathbf{A}^{(\mathbf{K}_j)}$$

We use GCN to implement the jump:

$$\mathbf{Q}^{'} = \hat{\mathbf{A}}\mathbf{Q}\mathcal{W}_Q, \quad \mathbf{K}^{'} = \hat{\mathbf{A}}\mathbf{K}\mathcal{W}_K$$

$\tilde{\mathbf{D}}$ is a diagonal matrix from the column-sum of A

The self-attention score will be updated as:

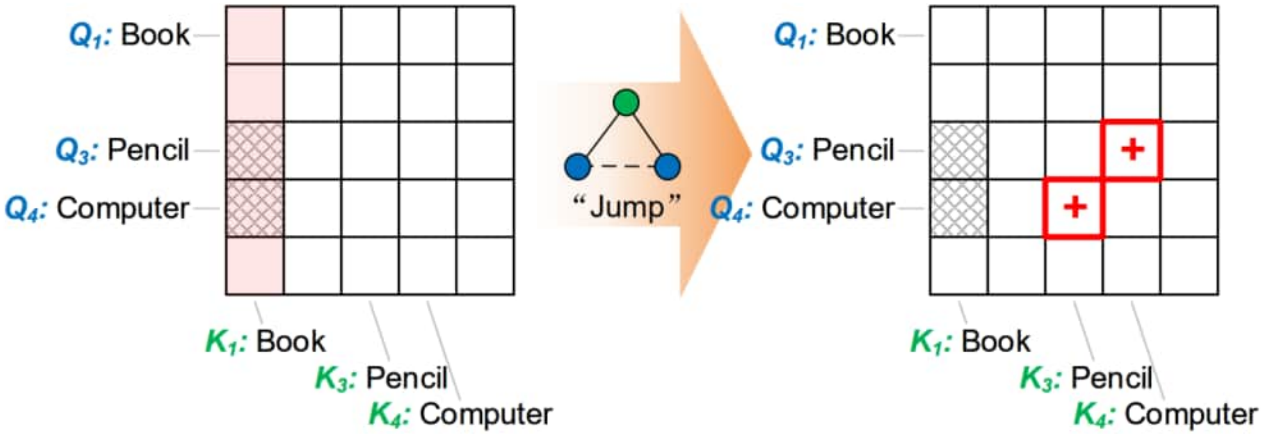
$$\mathbf{S}^{'} = \mathbf{Q}^{'}\mathbf{K}^{'\top} = \hat{\mathbf{A}}\mathbf{X}(\mathbf{W}_Q\mathcal{W}_Q)(\mathbf{W}_K\mathcal{W}_K)^\top\mathbf{X}^\top\hat{\mathbf{A}}^\top$$

Then we can define "jump" operation as: $\Phi(\mathbf{S}) = \hat{\mathbf{A}}\mathbf{S}\hat{\mathbf{A}}^\top$

Thus, the JAT attention is defined:

$$\mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\Phi(\mathbf{Q}\mathbf{K}^\top)}{\sqrt{d}}\right)\mathbf{V}$$

The efficient Jump Self-attention



(b) The overview of the high-order self-attention.

Thus, the JAT attention is defined:

$$\boxed{\mathcal{A}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \operatorname{softmax}\left(\frac{\Phi(\mathbf{Q}\mathbf{K}^\top)}{\sqrt{d}}\right)\mathbf{V}}$$

Using a sparsity measurement to find the most significant keys:

$$M^{(\mathbf{K}_j)}(\mathbf{S}) = \max_i(\mathbf{S}_{j\top}) - \operatorname*{mean}_i(\mathbf{S}_{j\top})$$

We follow the sampling strategy and choose Top-u Keys, thus the original self-attention S reduces to $\bar{\mathrm{S}} \in \mathbb{R}^{L\times u}$

Thus, the efficient jump self-attention is defined: $A(Q,K,V) = softmax\left(\frac{\Phi(QK'^T)}{\sqrt{d}}\right)V$

Finally, we combine the JAT and the original self-attention to get the final attention score.

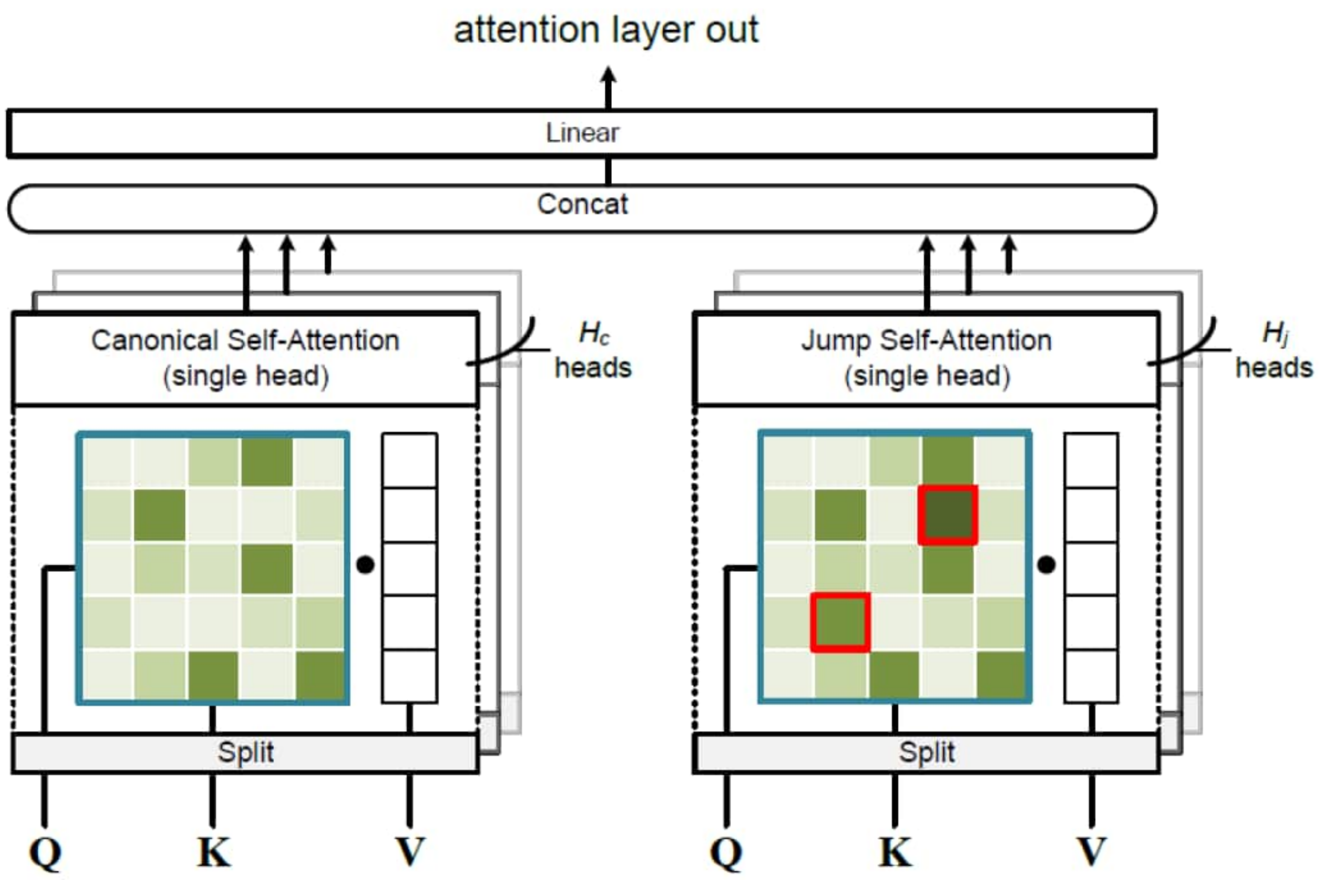
Overall architecture

The multi-head jump self-attention:

$$matAttention(Q, K, V) = concat(head_1, \dots head_{H_j}, \dots head_m)$$

$$Where\ head_i = \begin{cases} JatAttention(Q_i, K_i, V_i), if\ i \le H_j \\ Attention(Q_i, K_i, V_i), otherwise \end{cases}$$

JAT is compatible with canonical self-attention and can be used interchangeably.



Content

- Motivation
- The Jump Self-attention
- **Experimental Results**
- Summary

Experiment settings

Datasets

- General Language Understanding Evaluation (GLUE)
- Stickier Benchmark for General-Purpose Language Understanding Systems (SuperGLUE)
- Stanford Question Answering Dataset (SQuAD v1.1)

Baselines

GLUE: ELMo, BERT_{base} , RoBERTa_{base}
SuperGLUE: CBOW, M.F.C, BERT_{base} , RoBERTa_{base}
SQuAD v1.1: BiDAF-ELMo, XLNet_{base}, BERT_{base}, RoBERTa_{base}

Metrics

GLUE: The Matthews Correlation Coefficient for CoLA, Pearson Correlation Coefficient for STS-B, Accuracy for others
superGLUE: Exact Match (EM) and F1 score for MultiRC , Accuracy and F1 score for CB , Accuracy for others
SQuAD v1.1: Exact Match (EM) and F1 score

Platform

All methods are run on four Nvidia V100 GPUs.

(1) GLUE Experiment Results

Model	CoLA 8.5k	MRPC 3.5k	RTE 2.5k	STS-B 5.7k	QNLI 108k	QQP 363k	SST-2 67k	WNLI 0.64k	MNLI 392k	Average -
ELMo	44.1	76.6	53.4	70.4	71.1	86.2	91.5	56.3	68.6	68.7
$BERT_{base}$	56.3	88.6	69.3	89.0	91.9	89.6	92.7	53.5	86.7	79.7
BERT-A^3	**62.3**	**90.2**	**73.9**	90.1	91.1	90.6	92.9	55.6	**87.3**	81.5
BERT-JAT (ours)	61.6	89.1	73.3	**90.5**	**93.2**	**91.6**	**93.5**	**57.3**	85.6	**81.7**
BERT-JAT† (ours)	61.7	89.6	72.8	90.4	92.9	91.3	93.3	56.8	85.5	81.6
$RoBERTa_{base}$	63.6	90.2	78.7	91.2	92.8	**91.9**	94.8	-	87.6	86.4
RoBERTa-JAT (ours)	65.9	91.4	80.2	91.3	**93.2**	91.7	**95.4**	-	**87.7**	**87.1**
RoBERTa-JAT† (ours)	**66.7**	**91.7**	**80.9**	**92.3**	92.9	91.7	94.9	-	87.5	**87.1**

[1] JAT† represents the efficient variant of JAT. And the '-' indicates abandoned experiments.

JAT mechanism

- JAT enhance the complementary dependency of self-attention mechanism and gain more competitive scores.
- BERT-JAT outperform BERT*base* respectively on 9 tasks.
- RoBERTa-JAT outperform RoBERTa*base* respectively on 7 tasks.
- BERT-JAT achieves 9.4% score rising on CoLA and 7.1% on WNLI dataset.

(2) SuperGLUE Experiment Results

Model	CB Acc/F1	BoolQ Acc	COPA Acc	MultiRC F1/EM	WiC Acc	WSC Acc	RTE Acc	Average
CBOW	71.4/49.6	62.4	63.0	20.3/0.3	55.3	61.5	54.2	55.4
M.F.C	50.0/22.2	62.2	55.0	59.9/0.8	50.0	63.5	52.7	56.2
BERT_{base}	94.6/93.7	77.7	69.0	70.5/24.7	**74.9**	**68.3**	75.8	75.8
RoBERTa_{base}	92.8/93.7	81.5	74.0	70.7/28.4	69.1	64.4	78.7	75.9
RoBERTa-JAT	×	×	**82.0**	×	70.2	65.4	80.2	-
RoBERTa-JAT†	**98.2/98.5**	**82.3**	79.0	**71.6/29.5**	70.5	67.3	**80.9**	**78.5**

JAT mechanism

[1] JAT† represents the efficient variant of JAT.
[2] The '-' indicates abandoned experiments, and '×' happens when reaching out-of-memory.

- The efficient variant RoBERTa-JAT† achieves the best average scores.
- RoBERTa-JAT† outperform RoBERTa_{base} respectively on 7 tasks.
- RoBERTa-JAT† achieves 5% score(Acc) rising on CB dataset.

(3) SQuAD Experiment Results

Model	SQuAD v1.1		SQuAD v2.0	
	EM	F1	EM	F1
BiDAF-ELMo	-	85.6	63.4	66.2
XLNet_{base}	89.7	95.1	87.9	90.6
BERT_{base}	81.2	87.9	75.9	79.3
BERT-A^3	81.8	89.3	75.9	79.3
BERT-JAT	82.1	89.3	76.4	82.0
RoBERTa_{base}	88.9	94.6	86.5	89.4
RoBERTa-A^3	89.2	94.8	86.6	89.7
RoBERTa-JAT	**90.1**	**95.2**	**87.0**	**90.2**

JAT mechanism

BERT-JAT and RoBERTa-JAT achieve better performance in both EM and F1 scores.

(4) Ablation study

Table 4: The scores of different JAT's layer deployment.

Model	CoLA	MRPC	RTE	STS-B
RoBERTa$_{base}$	63.6	90.2	78.7	91.2
RoBERTa-JAT$_{l[1,4]}$	63.9	**90.9**	78.8	**91.2**
RoBERTa-JAT$_{l[5,8]}$	63.6	90.4	78.7	90.9
RoBERTa-JAT$_{l[9,12]}$	63.6	90.2	76.1	90.9
RoBERTa-JAT$_{l[1,6]}$	**64.6**	90.4	**79.4**	91.1
RoBERTa-JAT$_{l[7,12]}$	63.8	89.7	78.4	91.0

Table 5: The scores of different JAT's heads grouping.

Model	CoLA	MRPC	RTE	STS-B
RoBERTa$_{base}$	63.6	90.2	78.7	91.2
RoBERTa-JAT$_{h2}$	63.7	**91.4**	**80.2**	**91.1**
RoBERTa-JAT$_{h4}$	63.9	91.2	**80.2**	90.6
RoBERTa-JAT$_{h6}$	63.6	90.4	79.7	90.8
RoBERTa-JAT$_{h8}$	63.8	90.2	78.9	90.7
RoBERTa-JAT$_{h10}$	64.1	90.2	77.2	90.5
RoBERTa-JAT$_{h12}$	**64.7**	90.0	77.8	90.5

Table 6: The scores of JAT's increasing order.

Model	CoLA	MRPC	RTE	STS-B
RoBERTa$_{base}$	63.6	90.2	78.7	91.2
RoBERTa-JAT^{o1}	**65.4**	**91.4**	79.9	**91.3**
RoBERTa-JAT^{o2}	64.4	90.7	**80.2**	90.8
RoBERTa-JAT^{o3}	64.0	91.2	78.7	90.9

- Using JAT heads in lower or middle layers leads to better scores.
- Adding JAT heads enhance the high-order global dependencies but too strong high-order inductive bias suppressing the canonical self-attention.
- the second-order self-attention is sufficient to discover more higher-order connections.

(5) Case Study: Layer Stacking Degradation

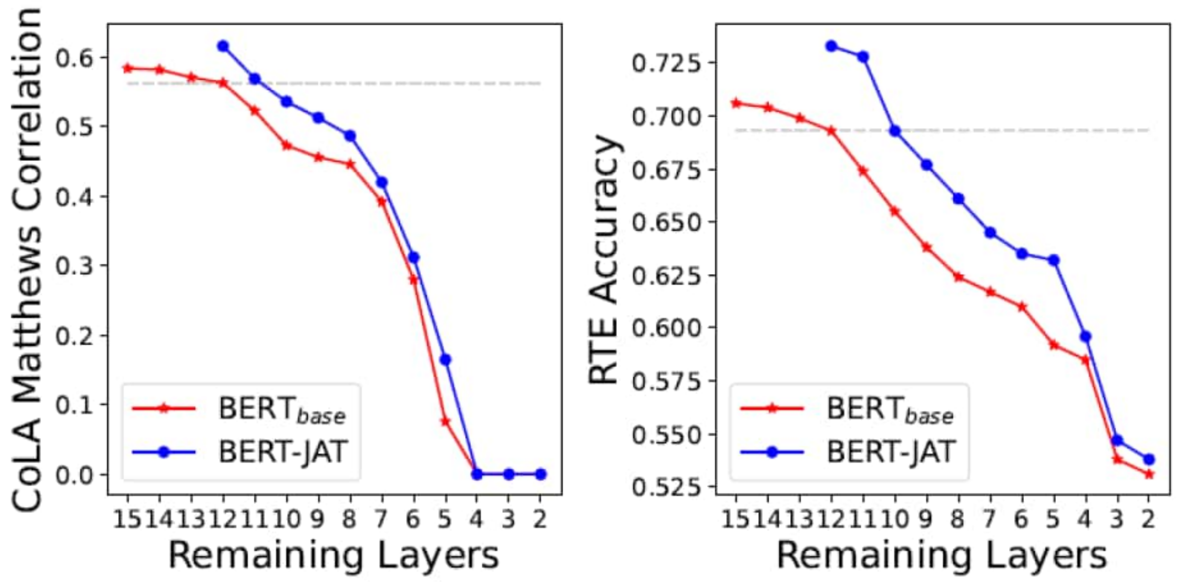


Figure 4: The performance decreases when the overall layers degrade from 12 to 2.

- BERT-JAT consistently outperforms BERT*base* with fewer layers.
- BERT-JAT with only 12 layers outperforms BERT*base* with complete 15 layers on both datasets.

Content

- Background
- The Jump Self-attention
- Experimental Results
- **Summary**

Summary

- Jump Self-attention mechanism allows for attention on dissimilarity pairs and it contributes to building the high-level dependency.
- Jump Self-attention mechanism can be deployed with canonical self-attention through proper configurations.
- JAT†: efficient variants for long inputs on large-scale models.
- Experimental results on three benchmarks demonstrate that JAT outperforms the baselines and it shows the benefits of introducing jump self-attention into Transformers.

Thank you!

Presented by: Haoyi Zhou, haoyi@buaa.edu.cn
www.zhouhaoyi.com
Discussion (online)

BEIHANG
UNIVERSITY