Attention Is All You Need[1]
Learned in Translation: Contextualized Word Vectors[2]

[1]Ahsish Vaswani, Noam Shazeer, Niki Parmer, Jokob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin
(Google Research)

[2] Bryan McCann, James Bradbury, Caiming Xiong, Richard Socher
(Salesforce Research)

Presented by: Dinghan Shen

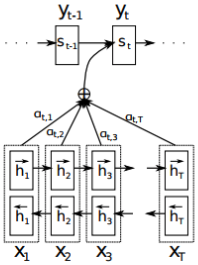
Oct 13, 2017

Main idea (Attention is all you need)

- Motivation:
  - **Recurrent neural networks** are typically employed in an encoder-decoder architecture for machine translation. However, their inherently sequential nature precludes parallelization within training samples;
  - Attention mechanism allows modeling of dependencies **regardless of their distances** in the input and output sequences.

- Contributions:
  - They propose a new simple network architecture, the Transformer, based solely on attention mechanism, **dispensing with recurrence and convolutions entirely**;
  - Allows for significantly more parallelization (**12 hours on eight P100 GPUs**);
  - State-of-the-art results on English-to-German translation quality.

Problem Settings



- Input: a sequence of symbol representations $\boldsymbol{x} = (x_1, ..., x_n)$;
- The encoder maps $\mathbf{x}$ to a sequence of continuous representations $\mathbf{h} = (h_1, ..., h_n)$;
- Given $\mathbf{h}$, the decoder then generates an output sequence $\mathbf{y} = (y_1, ..., y_m)$ of symbols one element at a time.

Model architecture

- Transformer:

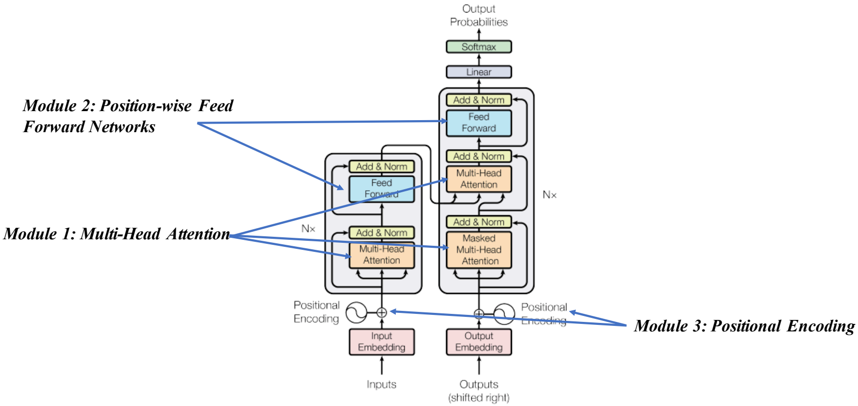


Figure 1: The Transformer - model architecture.

Module 1: Multi-Head Attention

- Queries, Keys, Values:
  - Given the input/output embedding matrix $X \in \mathbb{R}^{n \times E}$, three matrices (queries, keys, values) are computed as $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$;
  - This transformation can be implemented as either an MLP layer or CNN layer.

- Scaled Dot-Product Attention:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where the output $Attention(Q, K, V)$ is also a matrix of the dimension $\mathbb{R}^{n \times d_v}$ ($QK^T \in \mathbb{R}^{n \times n}$). They employ the scale factor $\sqrt{d_k}$ to prevent large magnitude of QK^T, which could lead to extremely small gradients.

Module 1: Multi-Head Attention

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

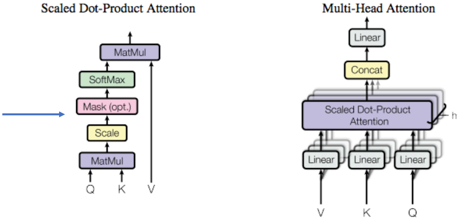


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

- The main intuition here is to learn *h* different linear transformations to project queries, keys and values. the attention is then computed in parallel;
- Allows the model to jointly attend to information from different representation subspaces at different positions.

Module 2: Position-wise Feed Forward Networks

- For each word/position, the following transformations are applied:

$$FFN(x) = W_2 ReLU(W_1 X + b_1) + b_2 \tag{2}$$

  - The linear transformations are identical across different positions (words), but vary from layer to layer;
  - The dimensionality of input and output is the same: $d_{model} = 512$.

Module 3: Positional Encoding

- Without recurrent or convolutional operations, the **word-order information** of a sequence is ignored. In this regard, they propose to inject some information about tokens' positions within the sequence:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{3}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{4}$$

  - For any given offset k, PE_{pos+k} can be considered as a linear function of PE_{pos}. Thus, the model can easily learn to attend by relative positions;
  - This strategy can be applied to sequences of arbitrary lengths.

Model architecture

- Transformer (decoding process, parallelization):

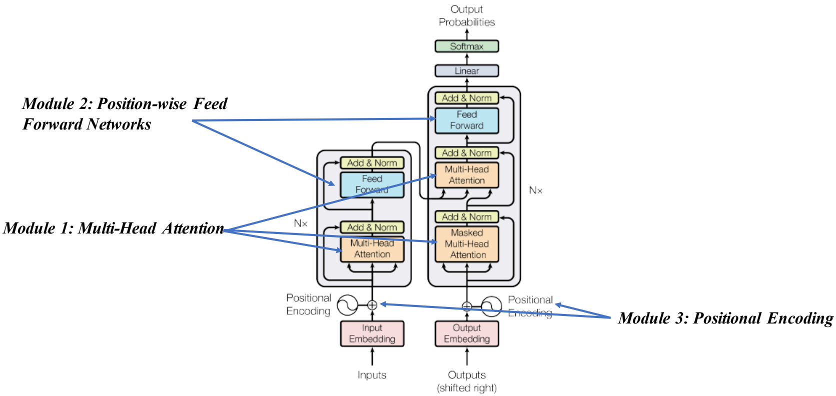


Figure 1: The Transformer - model architecture.

Why 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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- Maximum path length: the largest path length between any two input and output positions in networks.
- Typically n is smaller than d, thus self-attention has smaller complexity per layer. Besides, it is parallelable.
- Self-attention model is more interpretable.

Experiments (Machine Translation)

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [17]	23.75			
Deep-Att + PosUnk [37]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [36]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [31]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [37]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [36]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

Experiments (English Constituency Parsing)

- The output is subject to strong structural constraints;
- The output is significantly longer than the input.

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [35]	WSJ only, discriminative	88.3
Petrov et al. (2006) [28]	WSJ only, discriminative	90.4
Zhu et al. (2013) [38]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [38]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [25]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [35]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Dyer et al. (2016) [8]	generative	93.3

Main idea (Learned in Translation: Contextualized Word Vectors)

- Motivation:
  - Inspired by the successful transfer of CNNs trained on ImageNet to other tasks in computer vision, we aim to train an encoder from a large NLP task and **transfer** that encoder to other tasks;
  - Machine translation task could improve the quality of word embeddings due to its ability to share a common representation of words **in the context of sentences**.
- Contributions:
  - They introduce an approach for transferring knowledge from an encoder pretrained on machine translation to a variety of downstream NLP tasks;
  - The CoVe embeddings pretrained from their translation model outperforms other popular word embeddings, such as GloVe or character embeddings on many tasks.

How to transfer knowledge from machine translation?

- They first trained an attentional sequence-to-sequence model for English-to-German translation (standard settings), with a two-layer, bidirectional LSTM as the encoder networks (referred to as MT-LSTM);
- Transferring knowledge: for a sequence of words w, the CoVe vectors are represented as:

$$\mathrm{CoVe}(w) = \text{MT-LSTM}(\mathrm{GloVe}(w)) \tag{5}$$

- The hidden units are concatenated with GloVe as the final word vectors:

$$\tilde{w} = [\mathrm{GloVe}(w); \mathrm{CoVe}(w)] \tag{6}$$

Model for downstream tasks (classification, question answering)

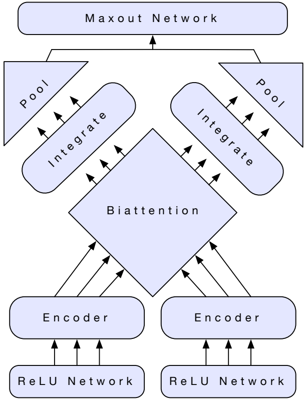
For input sequences w^x and w^y:

$$x = \text{biLSTM}\left(f(\tilde{w}^x)\right)$$
$$y = \text{biLSTM}\left(f(\tilde{w}^y)\right)$$
$$A_x = \text{softmax}\left(A\right) \qquad A_y = \text{softmax}\left(A^\top\right)$$
$$C_x = A_x^\top X \qquad C_y = A_y^\top Y$$

$$X_{|y} = \text{biLSTM}\left([X; X - C_y; X \odot C_y]\right)$$
$$Y_{|x} = \text{biLSTM}\left([Y; Y - C_x; Y \odot C_x]\right)$$

$$\beta_x = \text{softmax}\left(X_{|y} v_1 + d_1\right) \qquad \beta_y = \text{softmax}\left(Y_{|x} v_2 + d_2\right)$$
$$x_{\text{self}} = X_{|y}^\top \beta_x \qquad y_{\text{self}} = Y_{|x}^\top \beta_y$$

$$x_{\text{pool}} = \left[\max(X_{|y}); \text{mean}(X_{|y}); \min(X_{|y}); x_{\text{self}}\right]$$
$$y_{\text{pool}} = \left[\max(Y_{|x}); \text{mean}(Y_{|x}); \min(Y_{|x}); y_{\text{self}}\right]$$

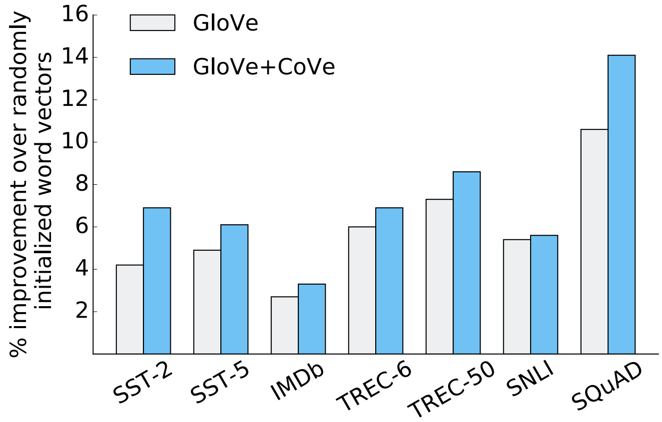


Experiments

			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

Table 2: CoVe improves validation performance. CoVe has an advantage over character n-gram embeddings, but using both improves performance further. Models benefit most by using an MT-LSTM trained with MT-Large (CoVe-L). Accuracy reported for classification tasks; F1 for SQuAD.

Experiments



Experiments

- Performance on SQuAD dataset:

Model	Reference	EM	F1
LR	Rajpurkar et al. [2016]	40.0	51.0
DCR	Yu et al. [2017]	62.5	72.1
M-LSTM+AP	Wang and Jiang [2017]	64.1	73.9
DCN+Char	Xiong et al. [2017]	65.4	75.6
BiDAF	Seo et al. [2017]	68.0	77.3
R-NET	Wang et al. [2017]	71.1	79.5
DCN+Char+CoVe	***Ours***	***71.3***	***79.9***

Table 4: Validation exact match and F1 for single-model question answering.

Takeaways

- Transformer architecture replaced RNN/CNN encoder/decoder with self-attention layers, achieving state-of-the-art MT results while allowing parallelization during training;
- Machine Translation can be employed as a great source towards learning transferable sentence encoders.