How to train a Large Language Model from Scratch

Ph.D. Dang Tran Thai

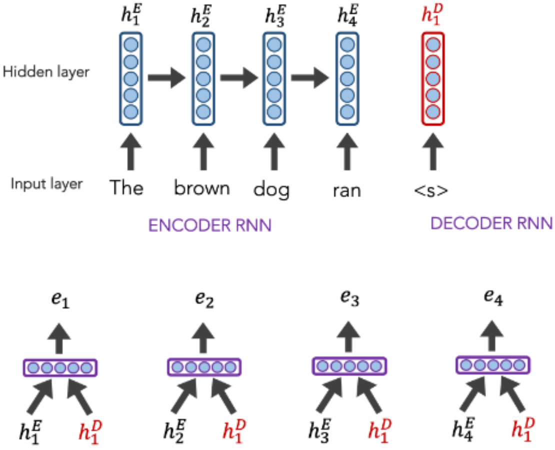
Department of Natural Language Processing
Virtual Assistant Technology Division
VinBigdata

August 18, 2024

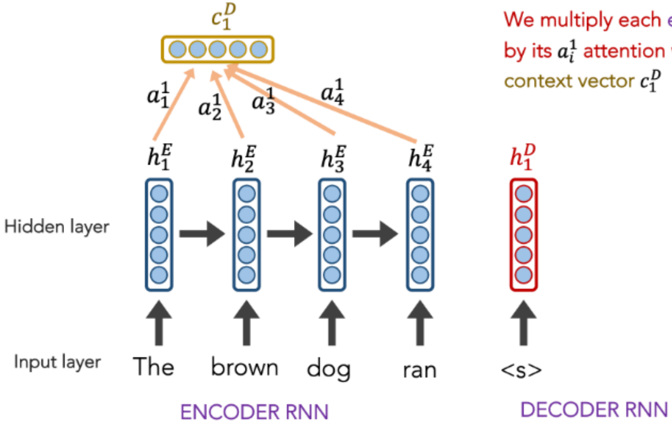
Outline

Attention in Seq2Seq (1)

- Input sequence X, encoder f_{enc}, decoder f_{dec}
- $f_{enc}(X)$ produces hidden states $h_1^E, ..., h_N^E$
- On time step t, have decoder hidden state h_t^D
- Attention score: $e_i = h_t^{D\intercal} h_i^E$
- Attention distribution: $a_i^t = \frac{\exp(e_i)}{\sum_i^N \exp(e_i)}$



Attention in Seq2Seq (2)

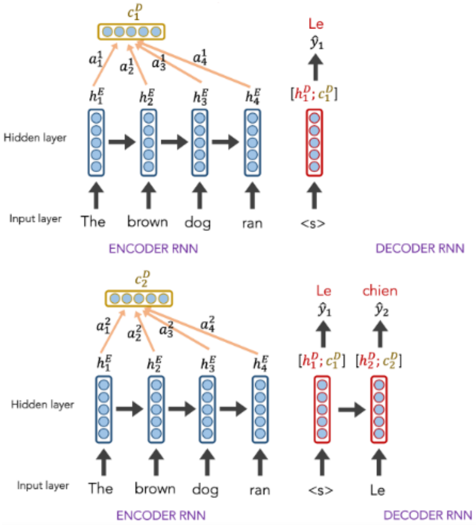


We multiply each encoder's hidden layer by its a_i^1 attention weights to create a context vector c_1^D

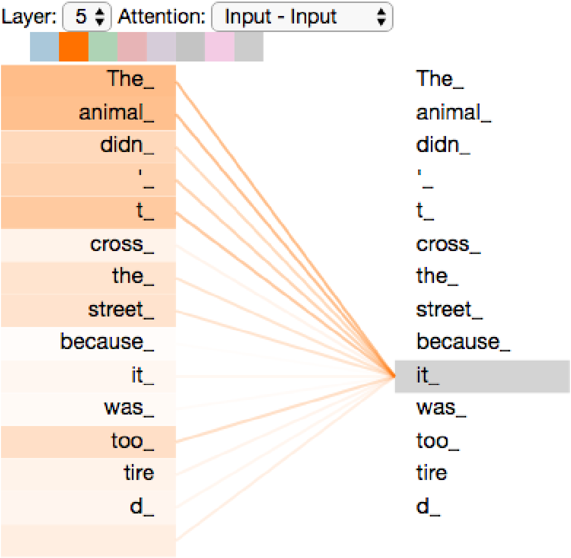
Attention in Seq2Seq (3)

Sample output $\hat{y}_t$ using both h_t^D and c_t^D:

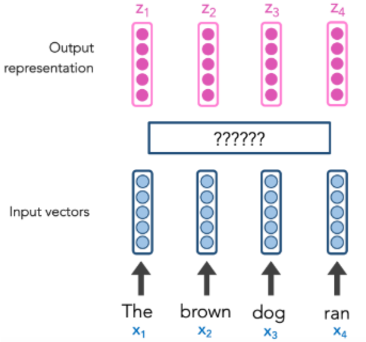
$$\hat{y}_t \sim f_{dec}(h_t^D, c_t^D, \theta) \qquad (1)$$

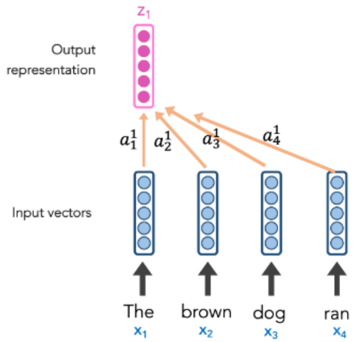


Self-Attention (1)



Self-Attention (2)





- Self-Attention's goal is to create great representations, z_i of the input
- z_i is based on a weighted contribution of each token in the input

Self-Attention (3)

- Given the input sequence: $x_1, x_2, ..., x_n$
- Each word x_i has 3 associated vectors: *Query vector* q_i, *Key vector* k_i, *Value vector* v_i:

$$q_i = w_q x_i$$
$$k_i = w_k x_i$$
$$v_i = w_v x_i$$

- For word x_i, let's calculate the scores $s_i^1, ..., s_i^n$ which represent how much attention to pay to each respective v_i:

$$s_i^j = q_i k_j \tag{2}$$

  where $j = 1, ..., n$
- Let's divide s_i^j by $\sqrt{d}$ where d is the dimension of k_i and softmax it:

$$a_i^j = softmax(\frac{s_i^j}{\sqrt{d}}) \tag{3}$$

Self-Attention (3)

- Compute z_i as the following:

$$z_i = \sum_{j=1}^{n} a_i^j v_j \tag{4}$$

Self-Attention is powerful that allows to create context-aware representations.

Transformer Architecture (1)

- The encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$
- Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time.
- Encoder: a stack of 6 identical layers. Each layer contains: multi-head self-attention; position-wise fully connected feed-forward network.
- Decoder: a stack of 6 identical layers. Each layer also contains sub-layers as encoder with adding a multi-head attention over the output of the encoder stack.

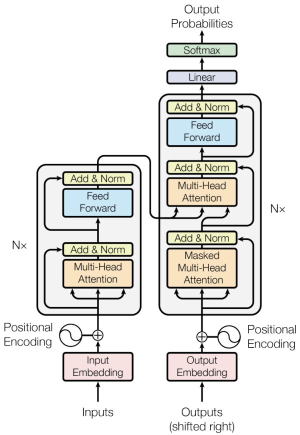


Figure: The Transformer - model architecture

Transformer Architecture (2)

Attention

- Scaled Dot-Product Attention

$$Attention(Q, K, V) = softmax(\frac{QK^\intercal}{\sqrt{d_k}})V \tag{5}$$

where d_k is the dimension of query and key vectors

- Multi-Head Attention: Linearly project queries, keys, and values h times:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, .., head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{6}$$

Positional Encoding

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{7}$$

Model Architectures

- Language Model (e.g., GPT): use Decoder-only architecture
- BERT-style models: use Encoder-only architecture
- T5, BART: use Encoder-Decoder architecture
- The major distinguishing factor for different architectures is the "mask" used by different attention mechanisms in the model

Attention Masks (1)

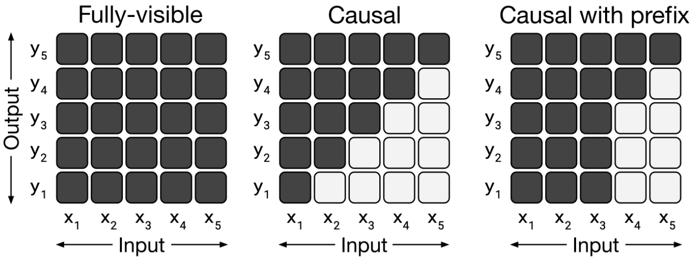


Figure 3: Matrices representing different attention mask patterns. The input and output of the self-attention mechanism are denoted x and y respectively. A dark cell at row i and column j indicates that the self-attention mechanism is allowed to attend to input element j at output timestep i. A light cell indicates that the self-attention mechanism is *not* allowed to attend to the corresponding i and j combination. Left: A fully-visible mask allows the self-attention mechanism to attend to the full input at every output timestep. Middle: A causal mask prevents the ith output element from depending on any input elements from "the future". Right: Causal masking with a prefix allows the self-attention mechanism to use fully-visible masking on a portion of the input sequence.

Attention masks (2)

- "Fully-visible" attention masking
  - Allowing a self-attention mechanism to attend to any entry of the input when producing each entry of its output
  - BERT also uses a fully-visible masking pattern and appends a special "classification" token to the input
- "Causal" attention masking
  - used in the self-attention operations in the Transformer's decoder
  - When producing the i^{th} entry of the output sequence, causal masking prevents the model from attending to the j^{th} entry of the input sequence for $j > i$
  - During the training the model can't "see into the future" as it produces its output
- Prefix LM
  - Use fully-visible masking during the prefix portion of the sequence
  - This architecture is similar to an encoder-decoder model with parameters shared across the encoder and decoder

Training Objectives

Causal Language Modeling

- The model is trained to predict the next token in the sequence given the previous tokens.
- The input tokens are fed into the model, and the model predicts the probability distribution of the next token
- The loss is calculated based on the model's predictions and the actual target tokens

Masked Language Modeling (a.k.a., denoising)

- The model is trained to predict masked tokens within the input sequence. During the preprocessing, a certain percentage of tokens are randomly masked, and the model is trained to predict the original tokens at those masked positions.
- The loss is calculated based on the model's predictions and the actual target tokens (the original tokens masked)

Pros & Cons of Training Objectives

Causal Language Modeling

- **Pros**: models are designed for auto-regressive text generation that helps to generate coherent and contextually documents or responses in chatbot
- **Cons**: Do not explicitly capture bidirectional context and the only generate tokens based on previous ones

Masked Language Modeling

- **Pros**: Model can potentially capture bidirectional context that help the model understand the context more effectively
- **Cons**: Cannot generate text auto-regressively

Unifying Language Learning Paradigms (UL2)

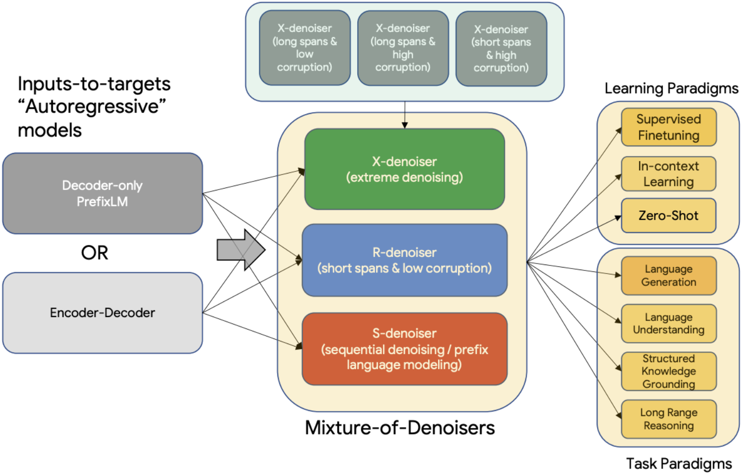


Figure 2: An overview of UL2 pretraining paradigm. UL2 proposes a new pretraining objective that works well on a diverse suite of downstream tasks.

What Language Model Architecture and Pre-training Objective work best for zero-shot generalization

- **Finding 1** [1]: The causal decoder-only models pretrained with a full language modeling objective achieve best zero-shot generalization when evaluated immediately after unsupervised pre-training
- **Finding 2**: Encoder-decoder models trained with masked language modeling achieve the best zero-shot performance after multitask finetuning.
- **Finding 3**: Decoder-only models can be efficiently adapted from one architecture/objective prior to the other. Specifically, to obtain both a generative and a multitask model with the smallest total compute budget possible, they recommend starting with a causal decoder-only model, pre-training it with a full language modeling objective, then using non-causal masked language modeling adaptation before taking it through multitask finetuning.

[1] https://arxiv.org/pdf/2204.05832