From Shallow to Deep Language Representations

1 Basics · 2 Shallow Models · **3 Transformers** · 4 BERT

KDD'19 Anchorage

Aston Zhang, Haibin Lin, Leonard Lausen, Sheng Zha, Alex Smola

www.d2l.ai gluon-nlp.mxnet.io

Outline (Transformer)

- Encoder-Decoder Architecture
- Attention
- Transformer Architecture
- Sequence Sampling

Encoder-Decoder

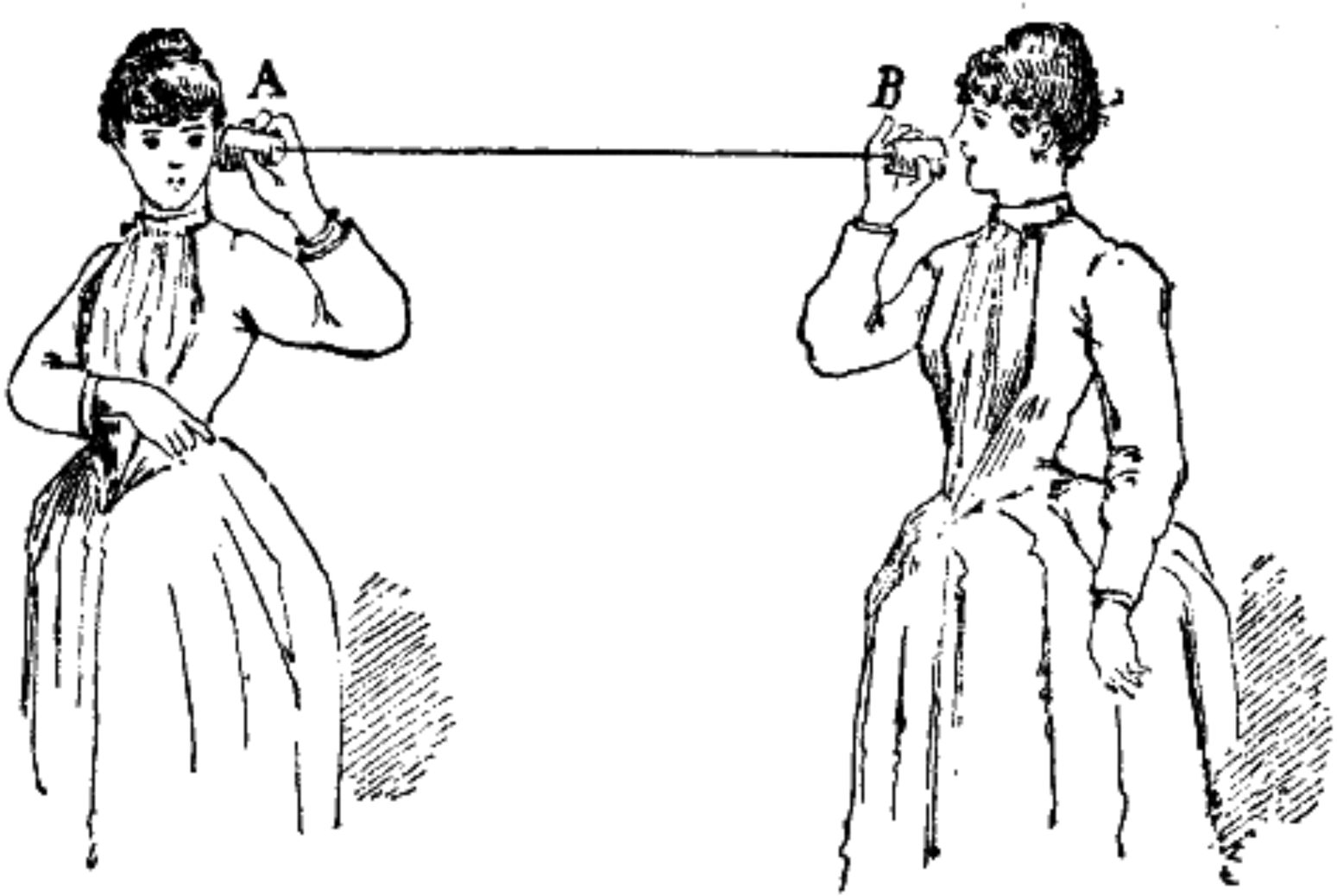
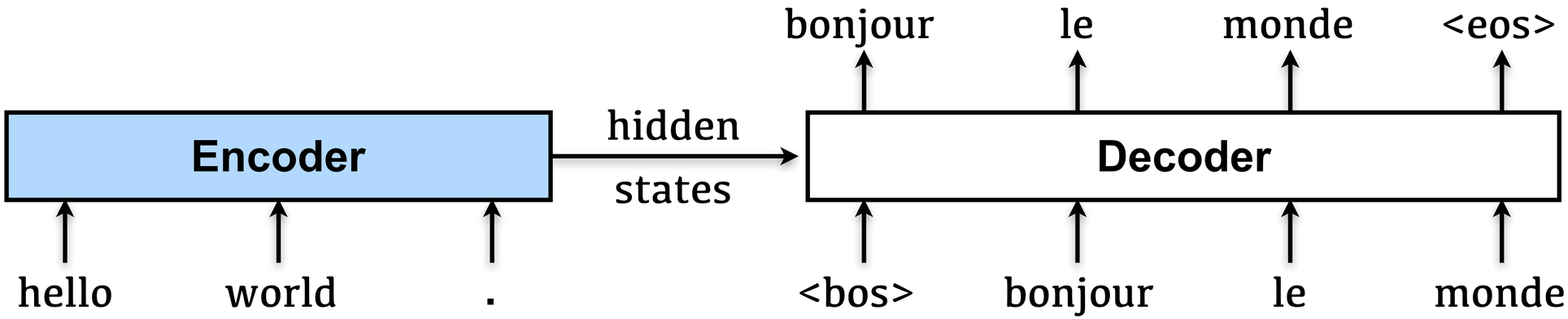


Image Source: https://en.wikipedia.org/wiki/Tin_can_telephone

Encoder-Decoder for Machine Translation

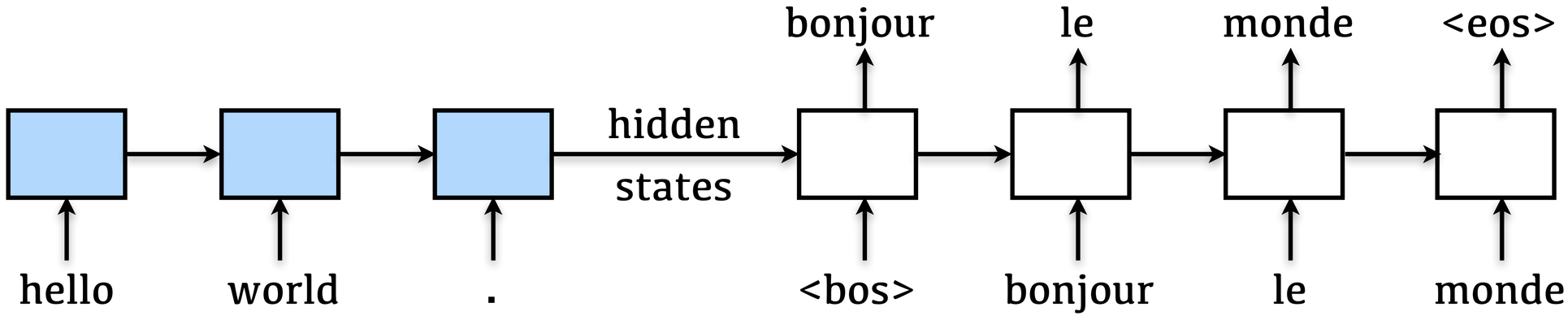


h = Encoder(‘hello’, ‘world’, ‘.’)

‘le’ = Decoder(‘bonjour’, ‘<bos>’, h)

Decoders are modeled to be predict one word at a time.

RNN as Encoder & Decoder

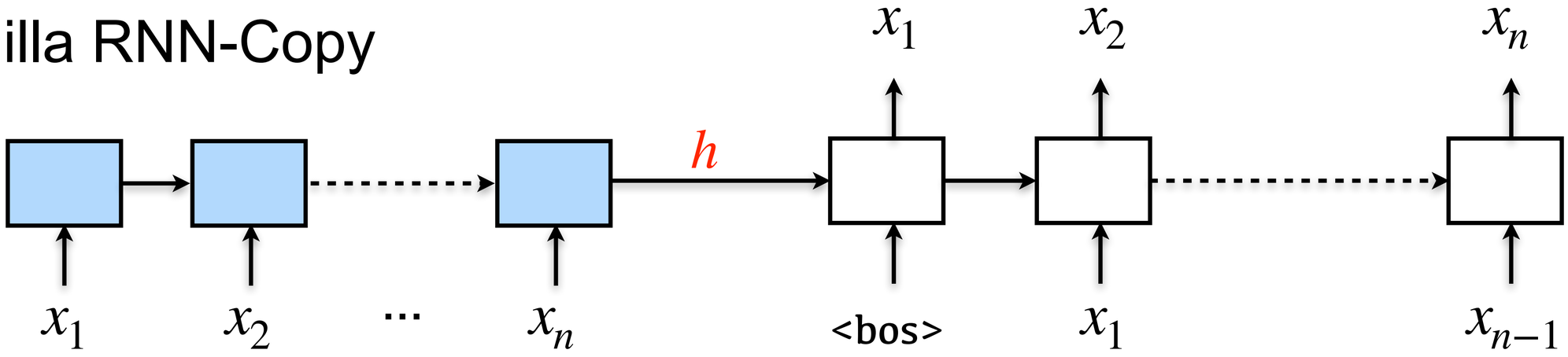


The Limitation of RNN Encoder-Decoder

- Mental Experiment: Sequence Copy

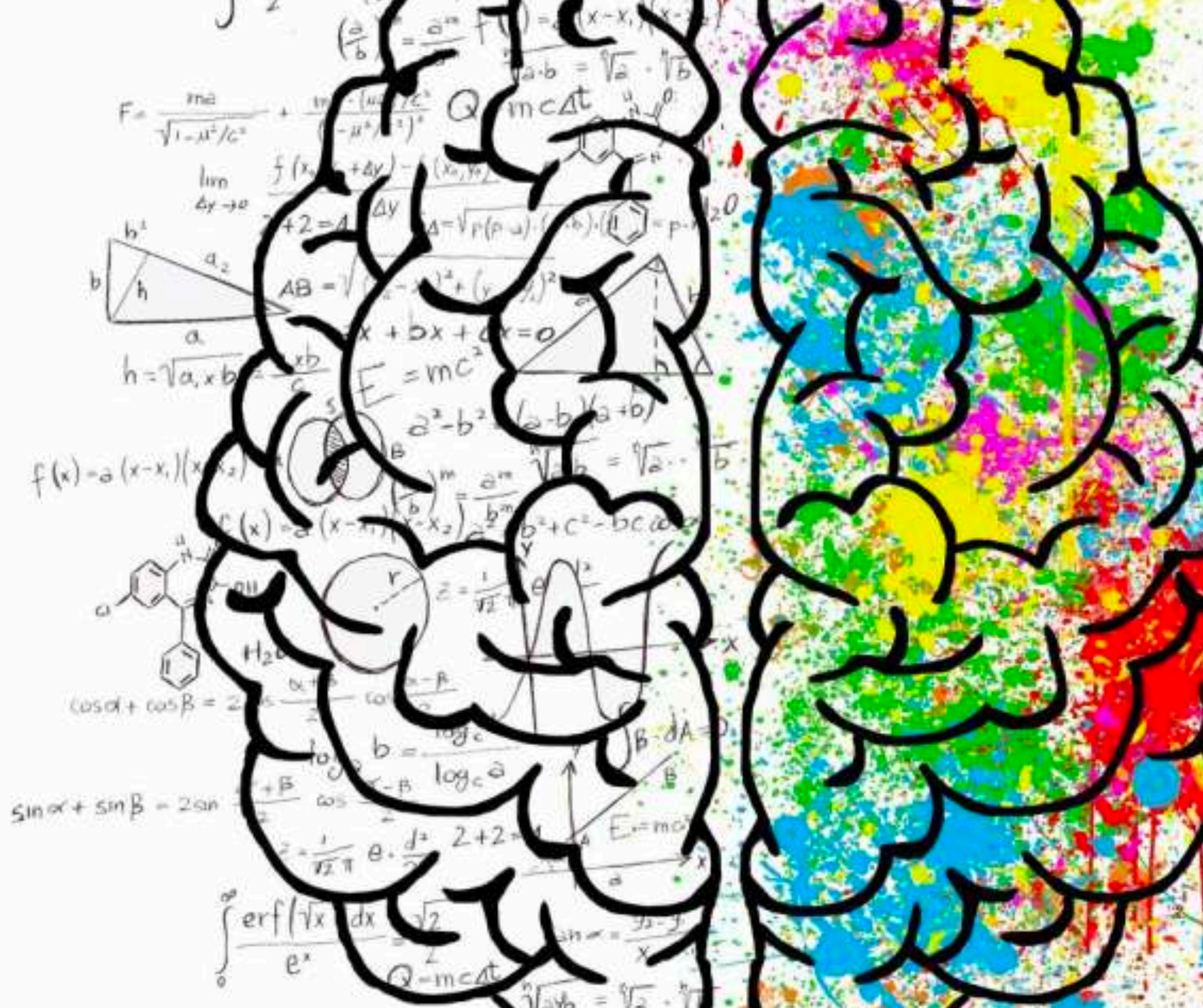
$$x_1, x_2, \ldots, x_n \rightarrow h \rightarrow x_1, x_2, \ldots, x_n$$

- Vanilla RNN-Copy



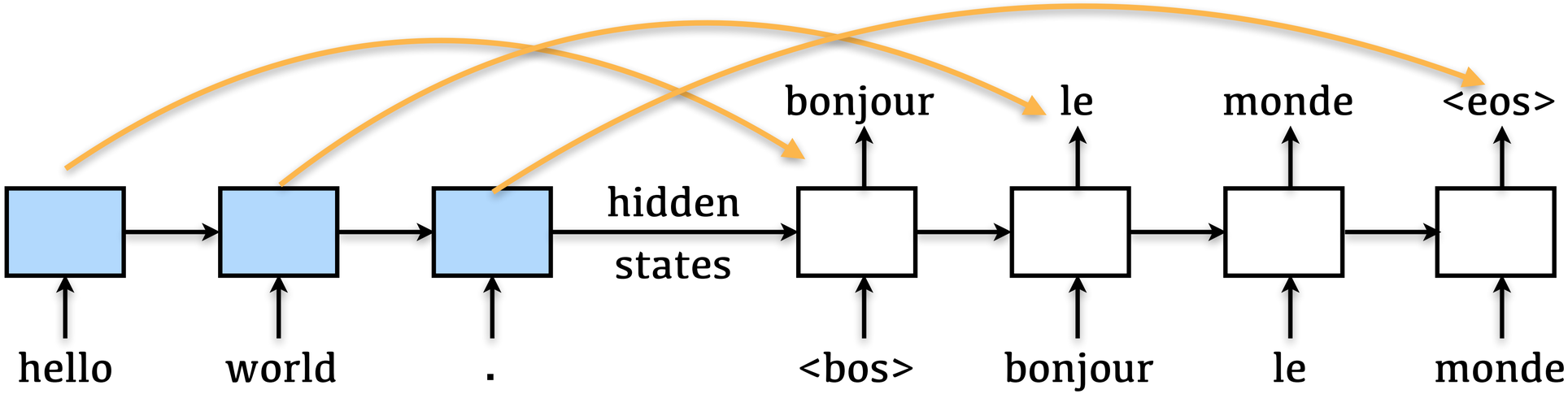
- What if the sequence length approaches infinity?
  - Fixed-dimensional vector has limited capacity
  - Problem is impossible to solve due to the memory capacity

Attention



Motivation

- Each output token may be related to specific source tokens

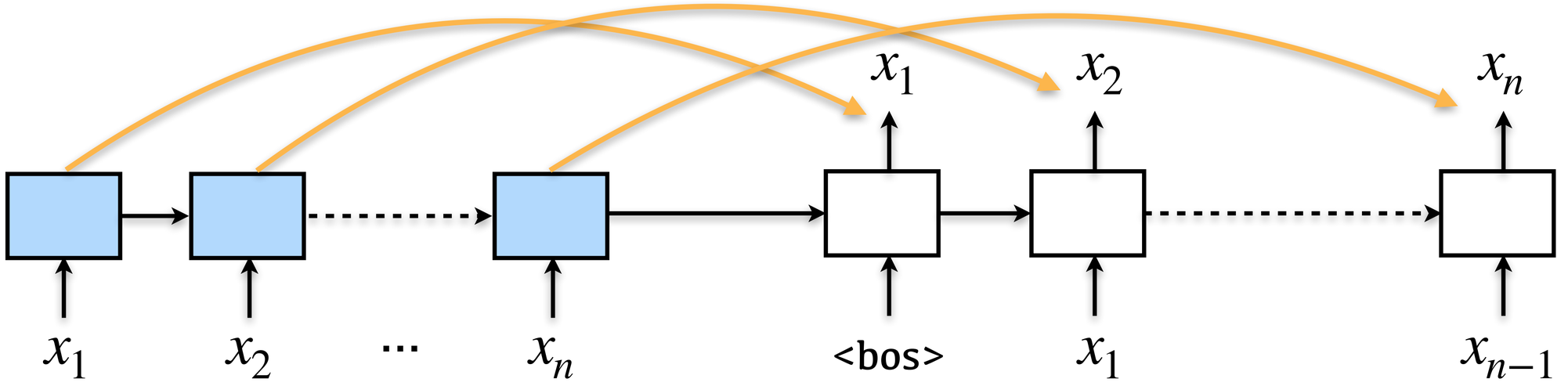


- Vanilla RNN: Use the last encoder state for decoding.
- Attentive RNN: Use all states and learn how to select relevant states.

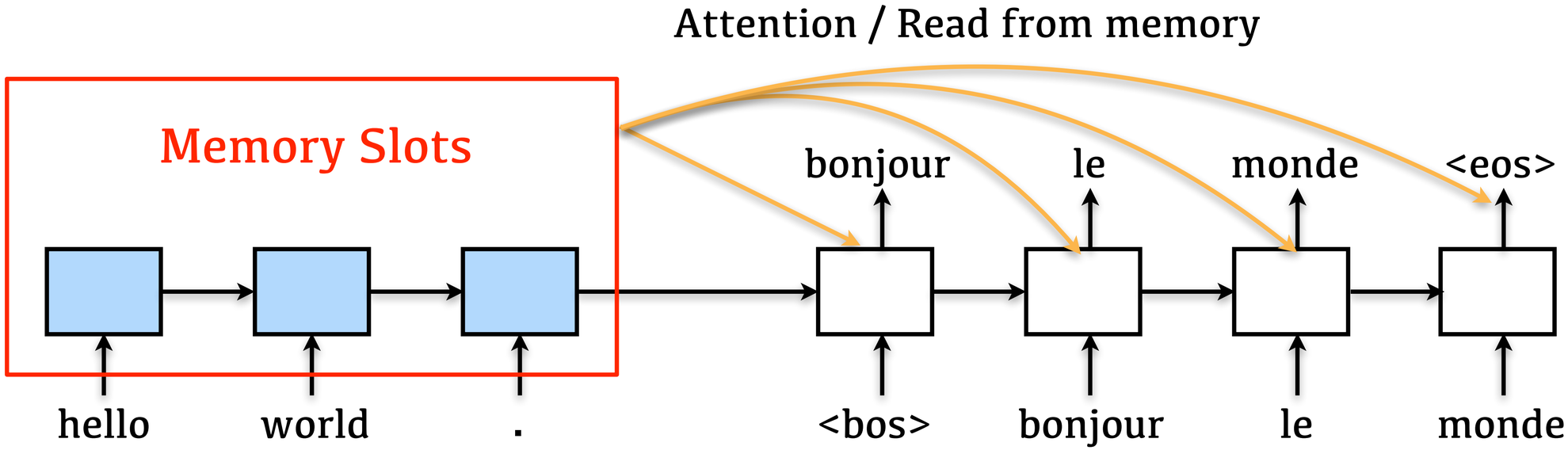
Sequence Copy by Attention

- Sequence Copy is possible with attention
  - Why? All states in the encoder are kept — Memory

$$x_1, x_2, \ldots, x_n \rightarrow h_1, h_2, \ldots, h_n \rightarrow x_1, x_2, \ldots, x_n$$



Attention & Memory

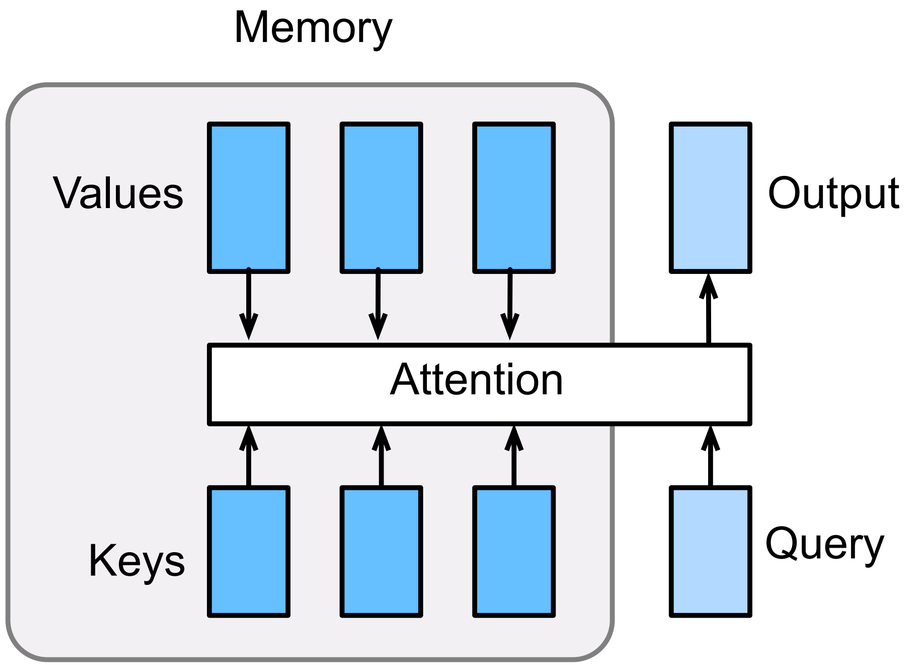


Each state in the RNN can be seen as memory slots

Query the memory for related states at every step in the decoding process.

Interpretation — Key Value Memory Network

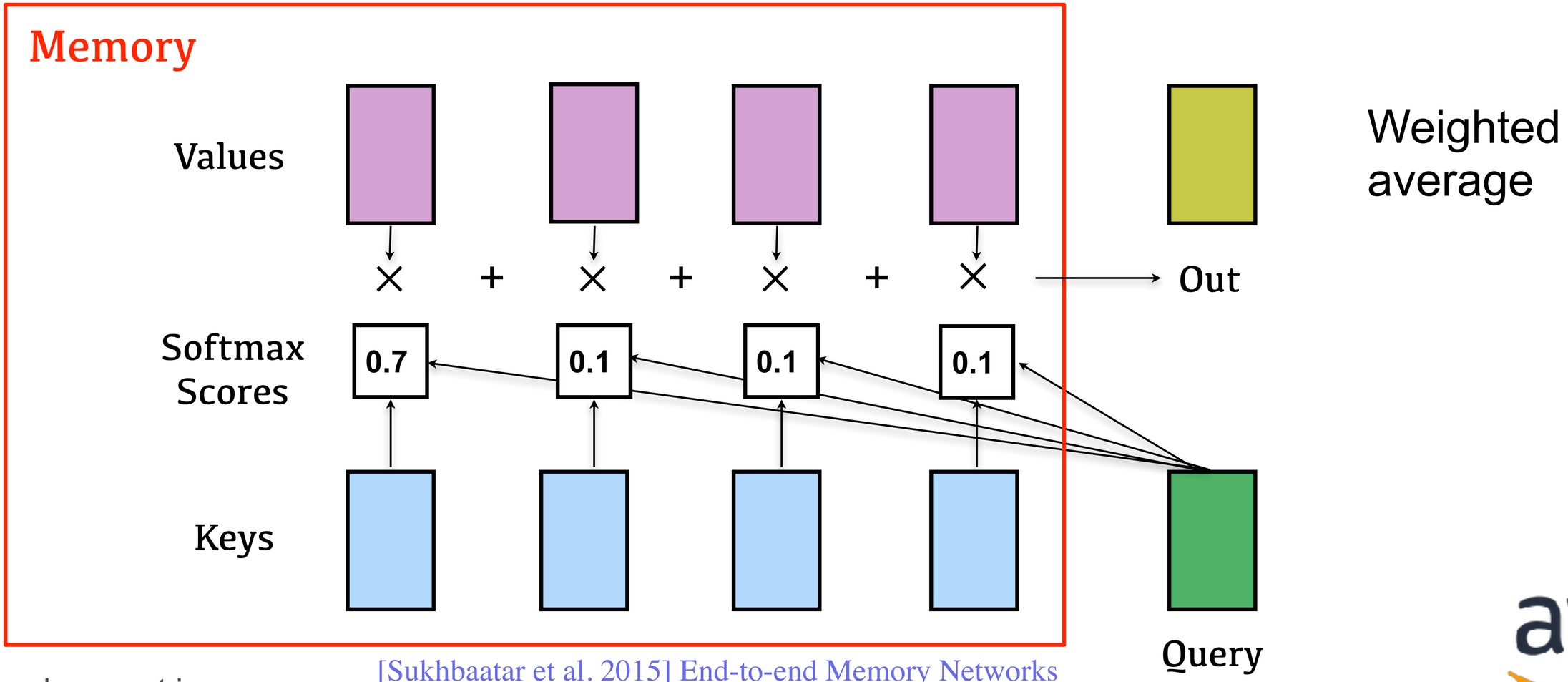
- Attention is a type of key-value memory structure
  - Memory consists of key-value pairs.
  - Fetch values based on respective query-key similarity (weighted average of values).



Interpretation — Key Value Memory Network

Out = Attention(Query, Key, Value)

Out = Attention(h_i^{dec}, $f(H^{enc})$, $g(H^{enc})$)

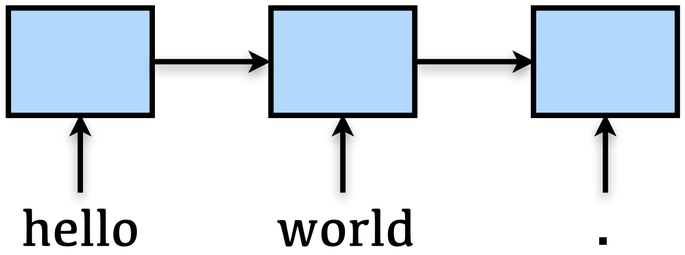


[Sukhbaatar et al. 2015] End-to-end Memory Networks

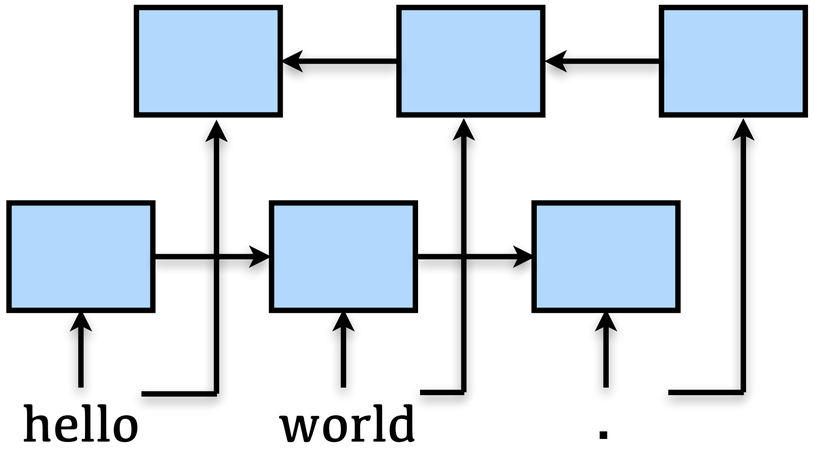
Transformer

Recurrence is bad

RNN



Bidirectional RNN

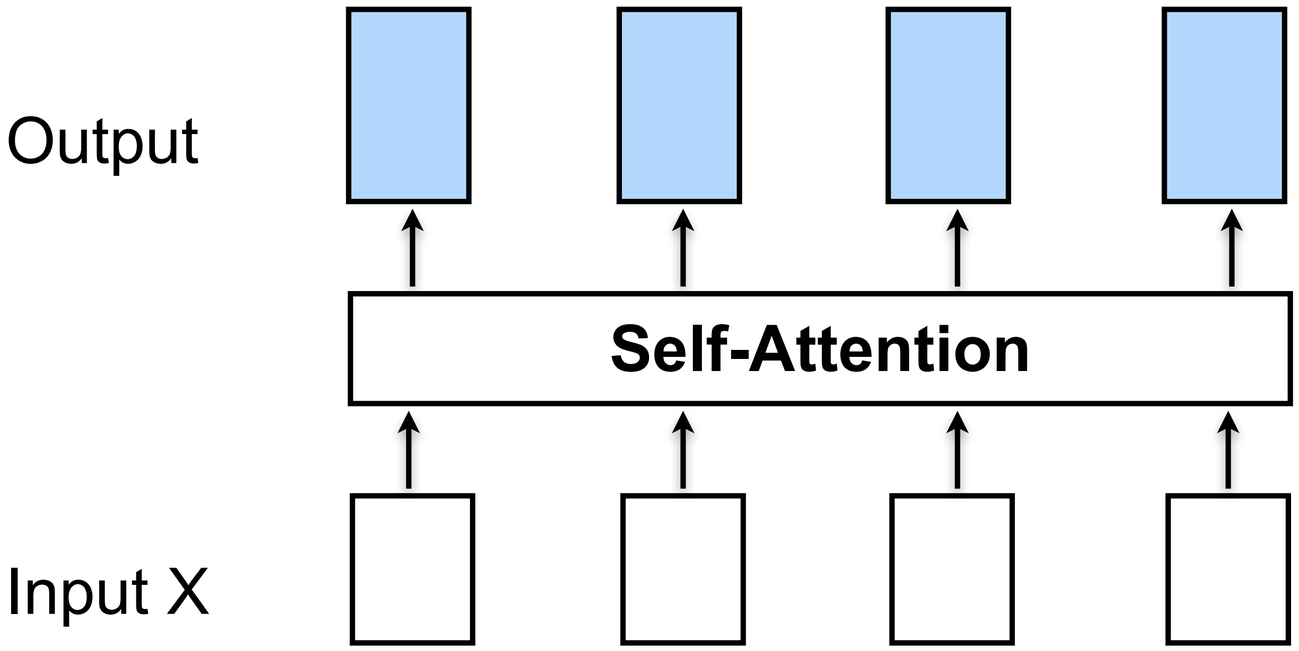


Recurrence introduces sequential dependencies that hinder parallel computation

Self-Attention is all you need

$$\text{Out} = \text{Attention}(\text{Query}, \text{Key}, \text{Value})$$

$$\text{Self-Attention: Attention}(W_q X, W_k X, W_v X)$$



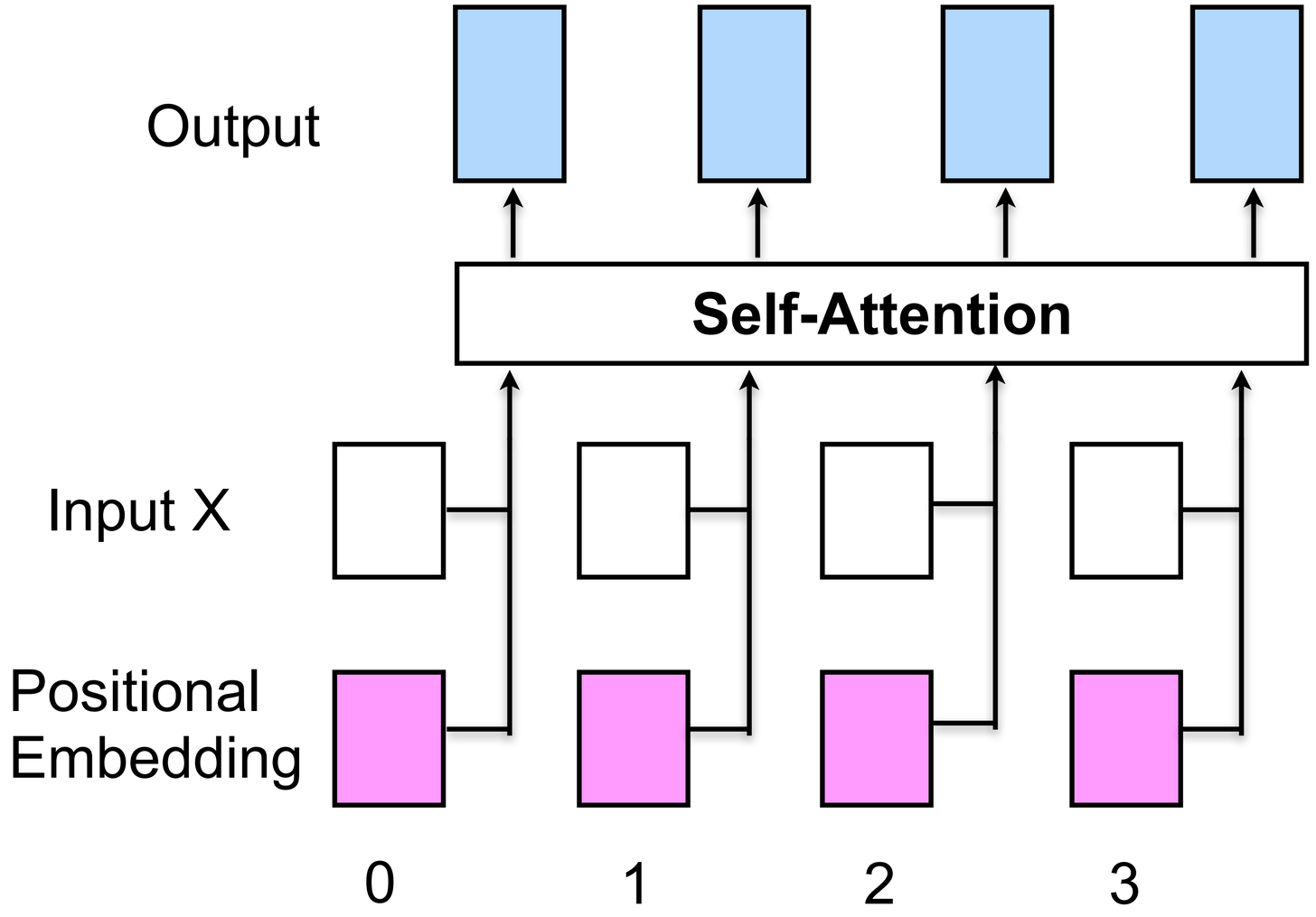
Attention in the encoder.
Run in parallel.

Permutational invariant?

Positional encoding

[Vaswani et al., 2017] Attention is all you need

Positional Encoding

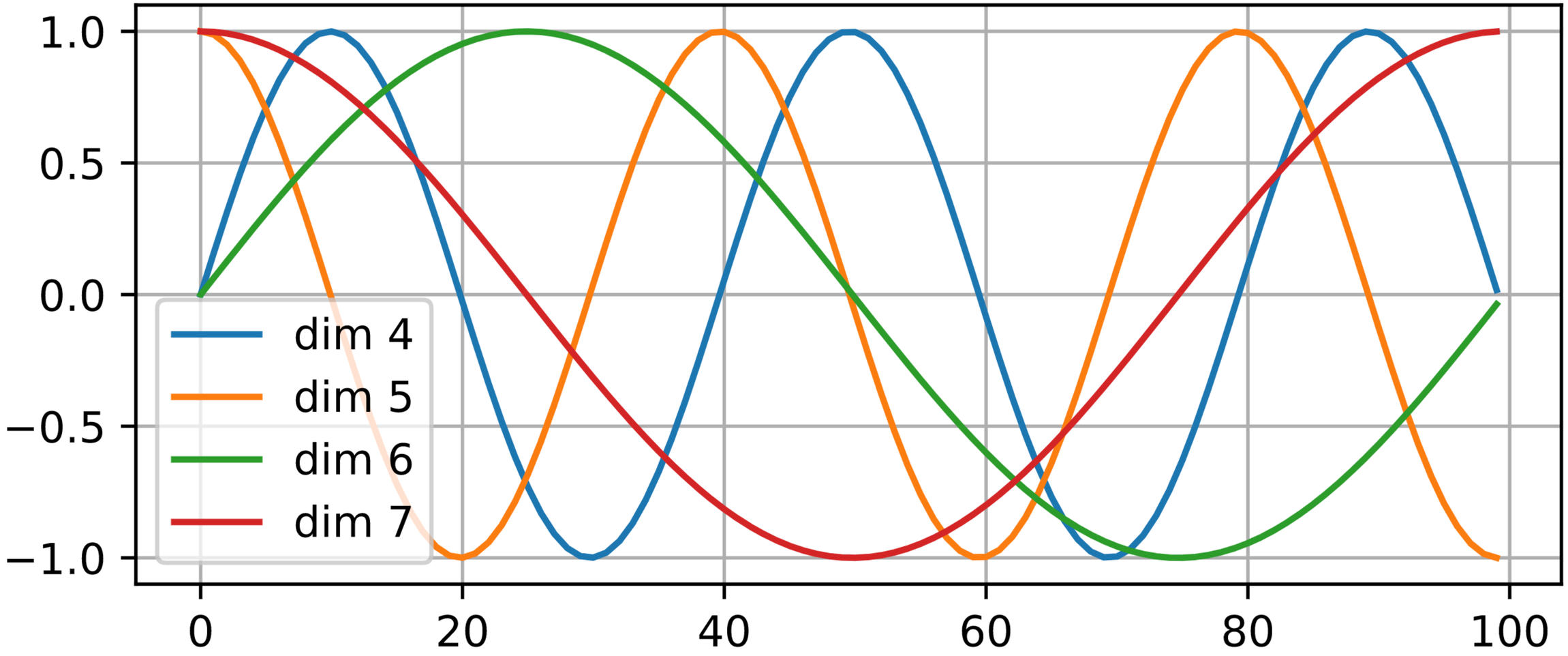


Many options:

- Learn the positional embeddings

- Interpolating a Sinusoidal function

Sinusoidal Positional Encoding



Multi-head Attention

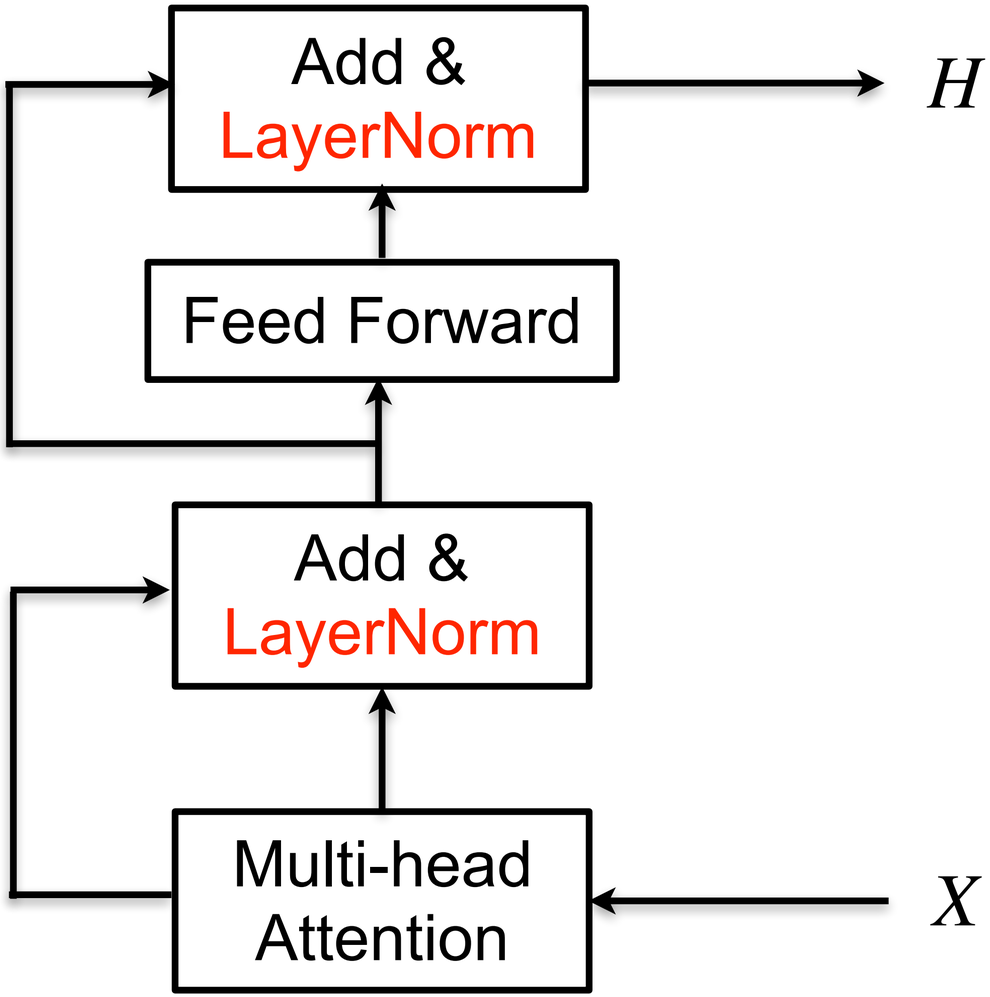
Increase the representational power of Attention

One-head: Similarity measurement in one subspace

Multi-head: Multiple subspaces!

$$\text{Multi-head Attention:} \quad W_o \begin{bmatrix} \text{Attention}(W_q^1 X,\ W_k^1 X,\ W_v^1 X) \\ \text{Attention}(W_q^2 X,\ W_k^2 X,\ W_v^2 X) \\ \dots \\ \text{Attention}(W_q^K X,\ W_k^K X,\ W_v^K X) \end{bmatrix}$$

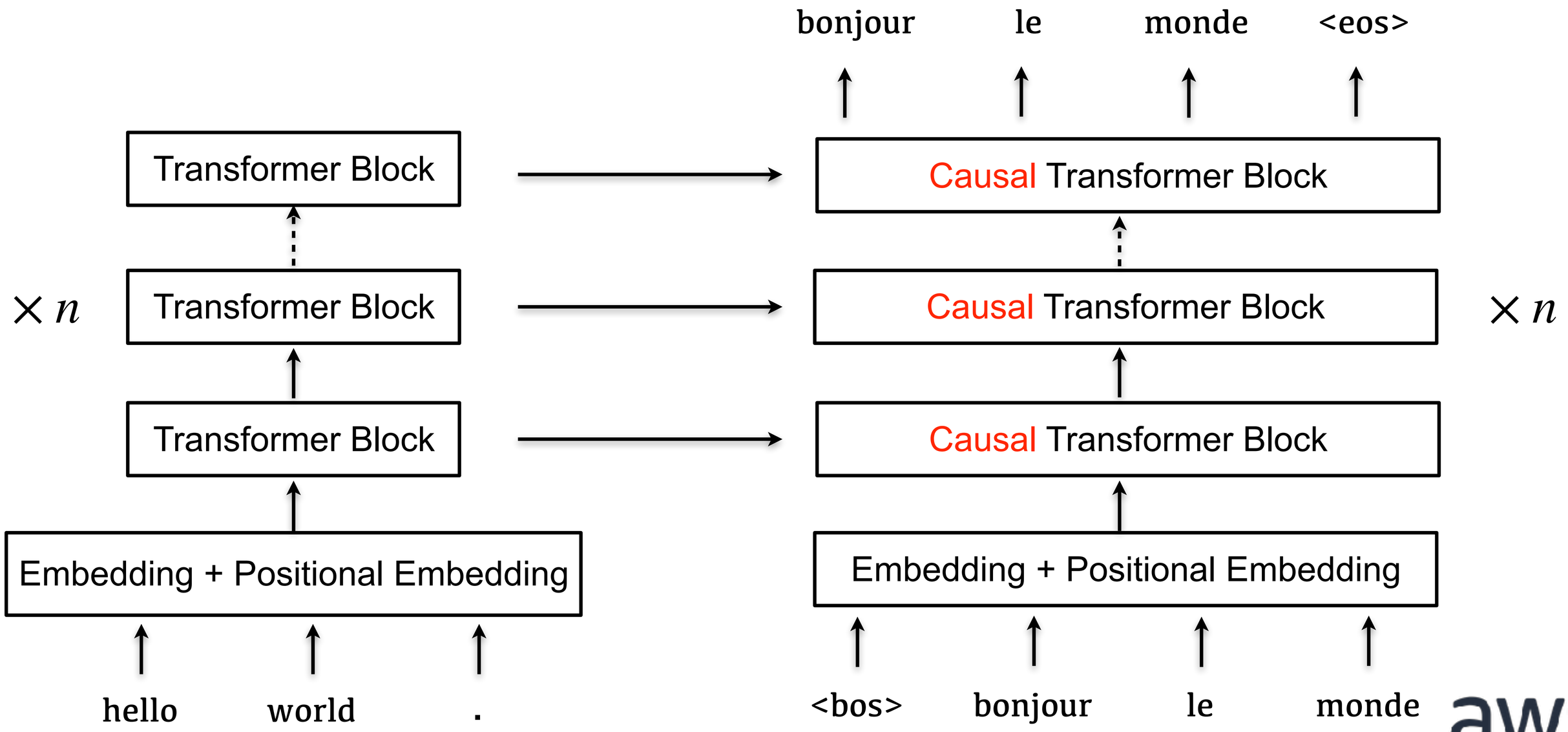
Transformer Block = Multi-head Attention + Residual + Layer Normalization



X.shape = (#Batch, #Channels)

$$\text{LayerNorm: } \frac{\text{X} - \text{X.mean(axis=-1)}}{\text{X.std(axis=-1)}}$$

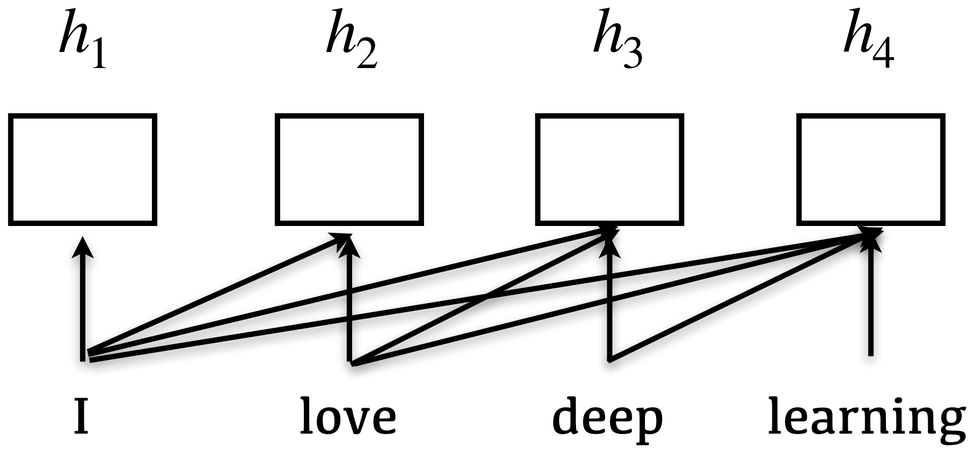
Transformer Architecture for Machine Translation



gluon-nlp.mxnet.io

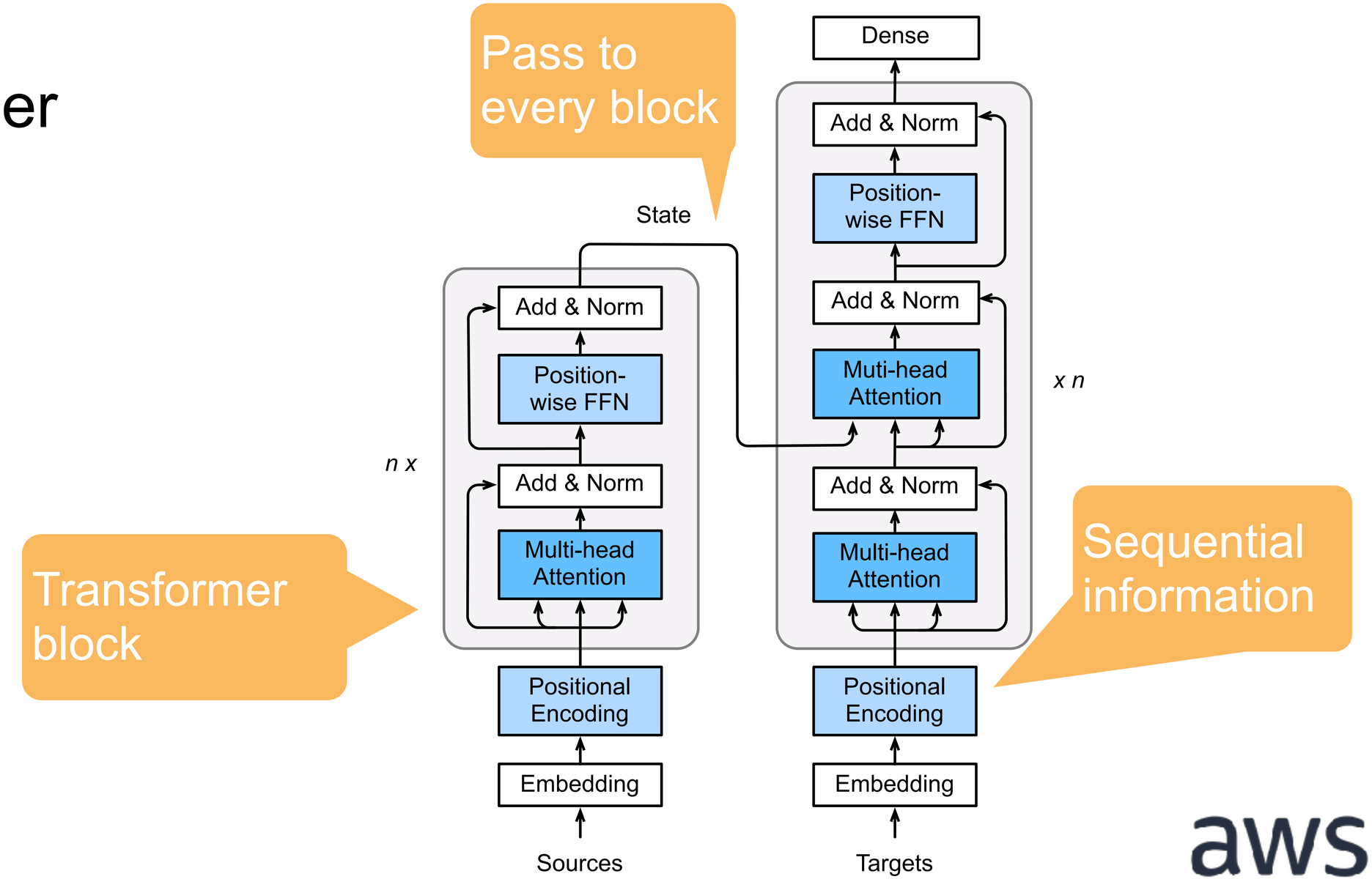
Causal Attention

- Never attend to the future.
- Avoid information leak.



Transformer Architecture

Encoder-decoder architecture



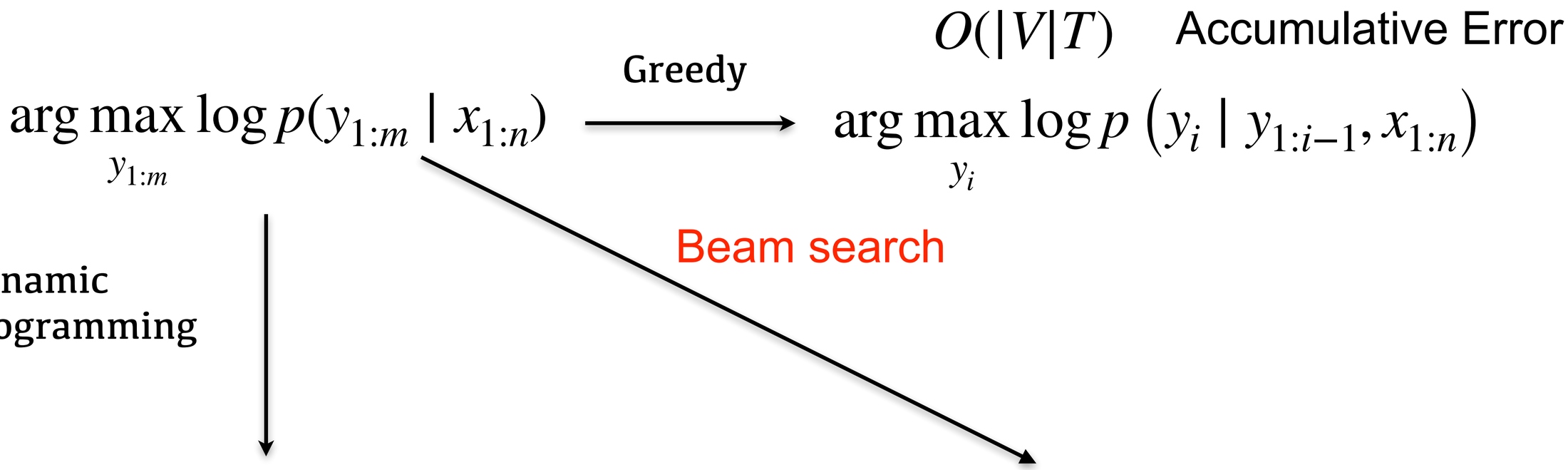
Sequence Sampling

How to Predict with a Machine Translation Model?

- Decoder returns the probability of single-step ahead prediction: $p\left(y_i \mid y_{1:i-1}, x_{1:n}\right)$
- One-step ahead — Multi-step ahead

$$\arg\max_{y_{1:m}} \log p(y_{1:m} \mid x_{1:n}) = \arg\max_{y_{1:m}} \sum_{i=1}^{m} \log p(y_i \mid y_{1:i-1}, x_{1:n})$$

Search-based Algorithms for Inference

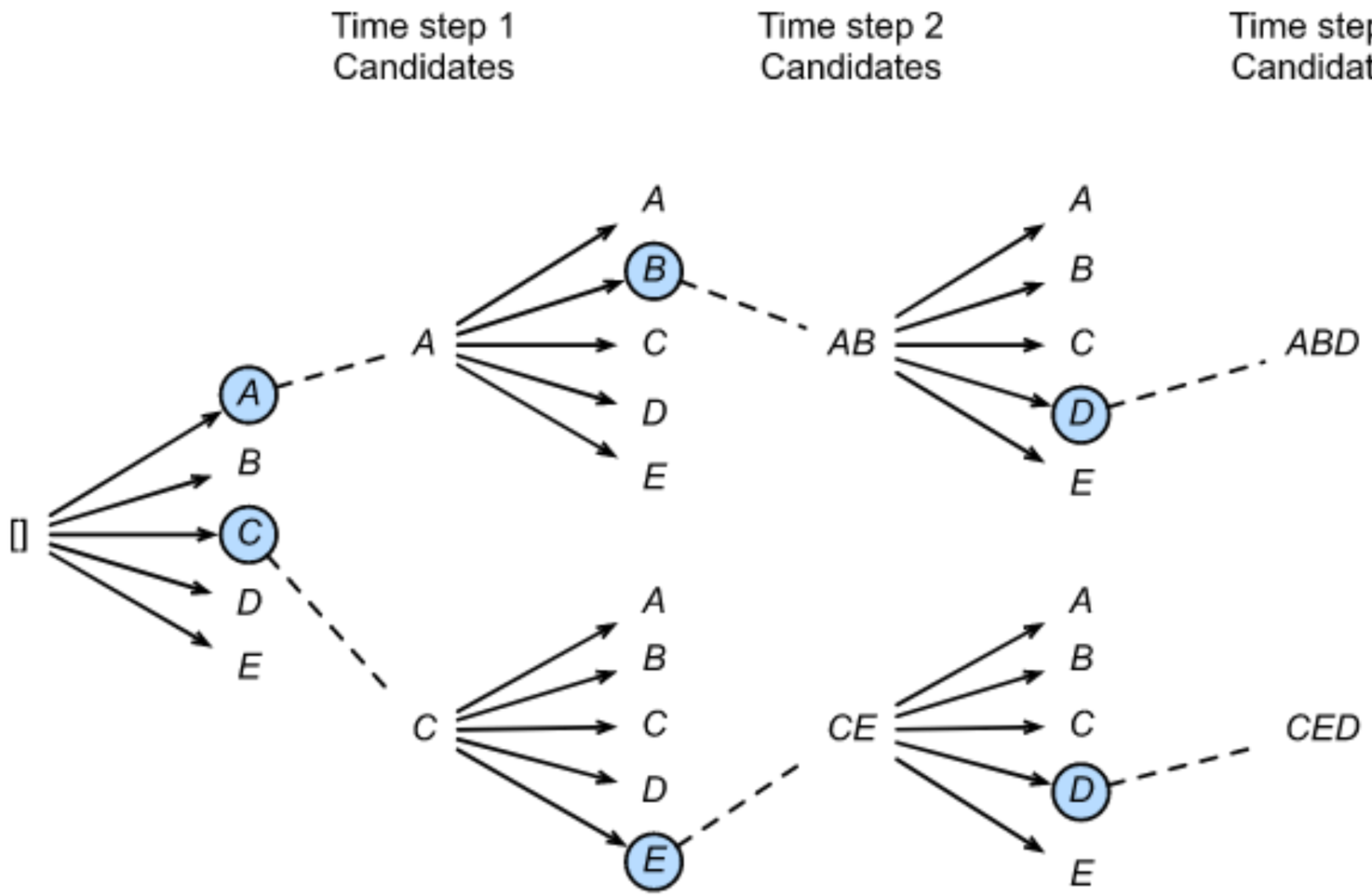


Exact inference.
Too many states, space complexity $O(|V|^T)$.

$O(K|V|T)$

Accumulative error, but better than greedy. Set K to trade off.

Beam Search for Sequence Sampling



Always keep sequences with the top K scores.

Beam Size = 2, |V| = 5

Length Penalty

$\log p(y_i \mid y_{1:i-1}, x_{1:n})$ is always negative

$$\arg\max_{y_{1:m}} \sum_{i=1}^{m} \log p(y_i \mid y_{1:i-1}, x_{1:n})$$ prefers shorter sequences!

How to solve? Penalize shorter sequences.

$$\arg\max_{y_{1:m}} \frac{\log p(y_{1:m} \mid x_{1:n})}{lp(m)}$$

Log-likelihood Score

Length penalty function

$$lp(m) = \frac{(K+m)^{\alpha}}{(K+1)^{\alpha}}$$

E.g., K = 5, alpha = 0.6