Dive into Deep Learning for NLP

4. Machine Translation and Sequence Sampling

Xingjian Shi

gluon-nlp.mxnet.io

13:15-14:15	Natural Language Processing and Deep Learning Basics
14:15-14:25	Break
14:25-15:15	Context-free Representations with Word Embeddings
15:15-15:55	Machine Translation and Sequence Sampling
15:55-16:35	Contextual Representations with BERT
16:35-16:45	Break
16:45-17:15	Model Deployment with TVM

Encoder-Decoder

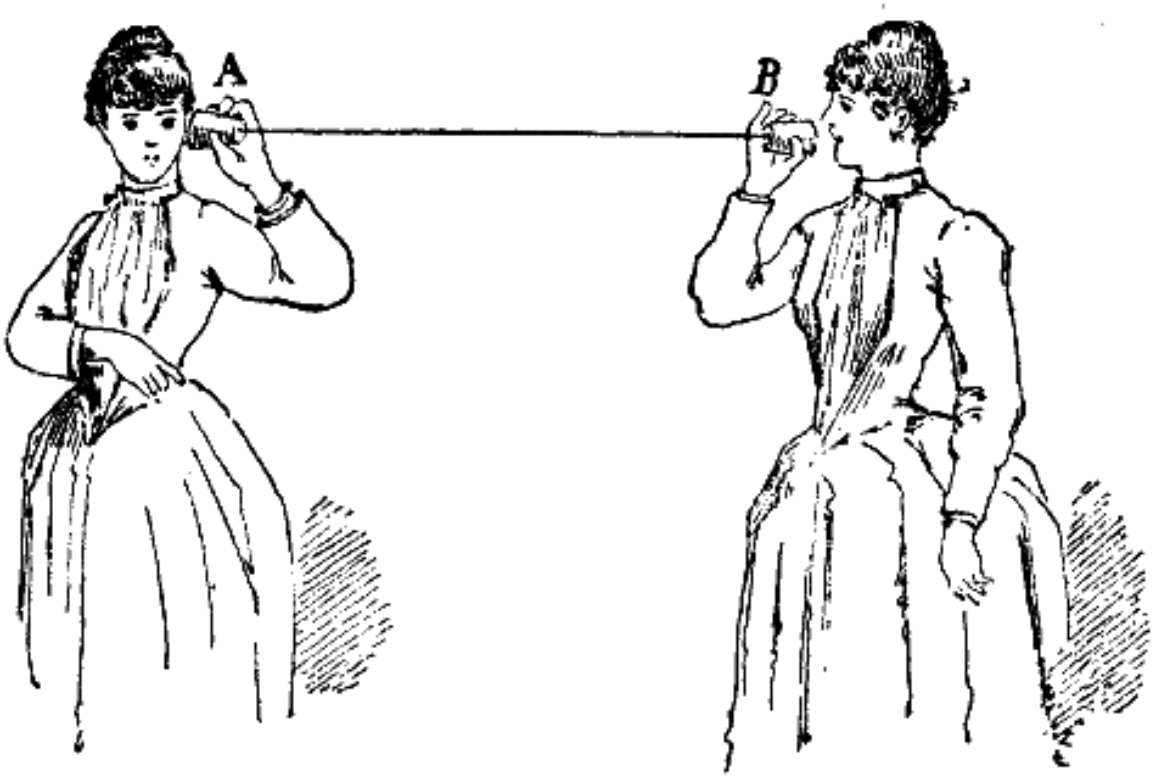
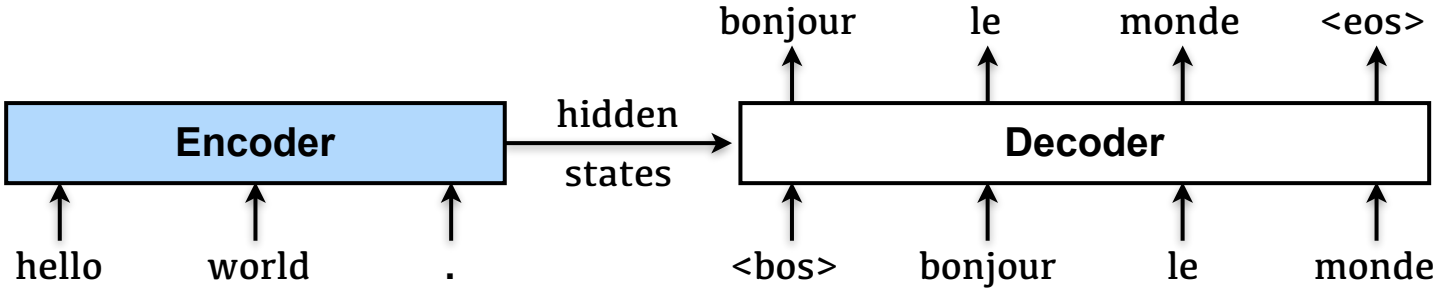


Image Source: https://en.wikipedia.org/wiki/Tin_can_telephone

Encoder-Decoder for Machine Translation

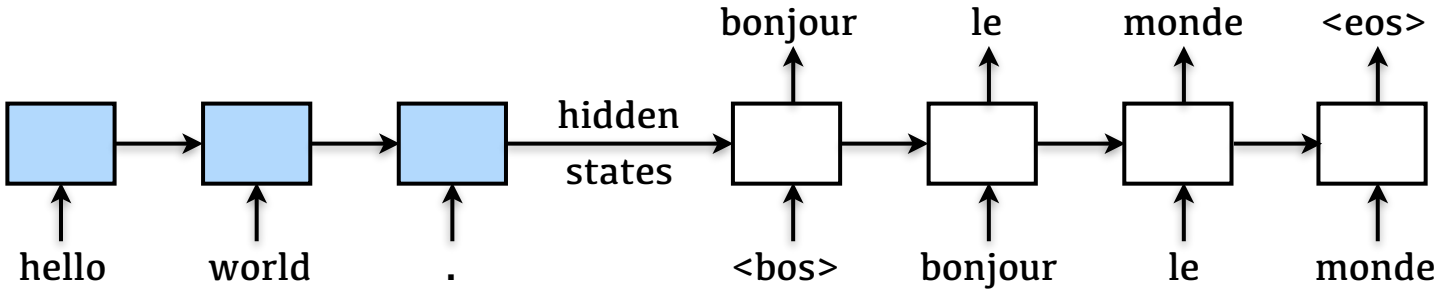


h = Encoder(‘hello’, ‘world’, ‘.’)

‘le’ = Decoder(‘bonjour’, ‘<bos>’, h)

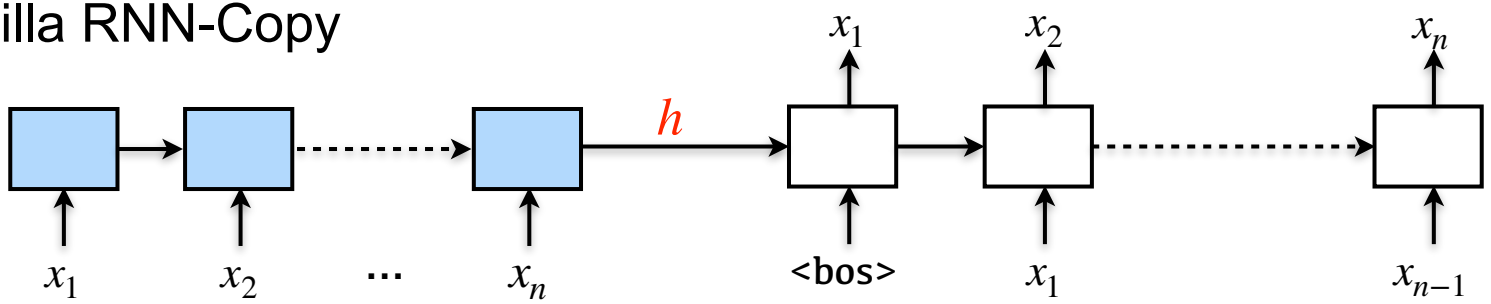
Difficult to generate a sequence. Decoders are modelled to be single-step-ahead.

RNN as Encoder & Decoder



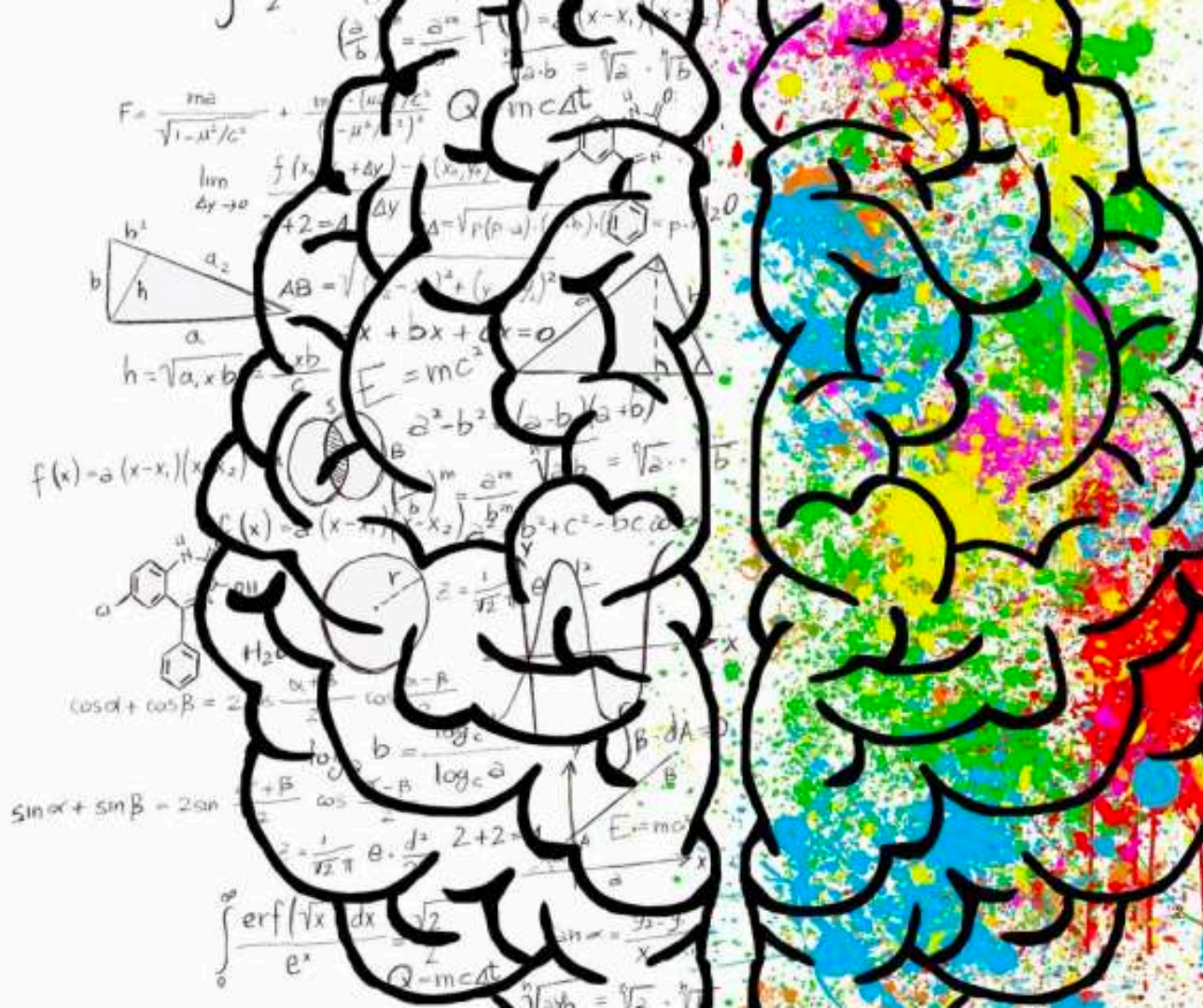
The Limitation of RNN Encoder-Decoder

- Mental Experiment: Sequence Copy
  - $x_1, x_2, \ldots, x_n \rightarrow h \rightarrow x_1, x_2, \ldots, x_n$
- Vanilla RNN-Copy



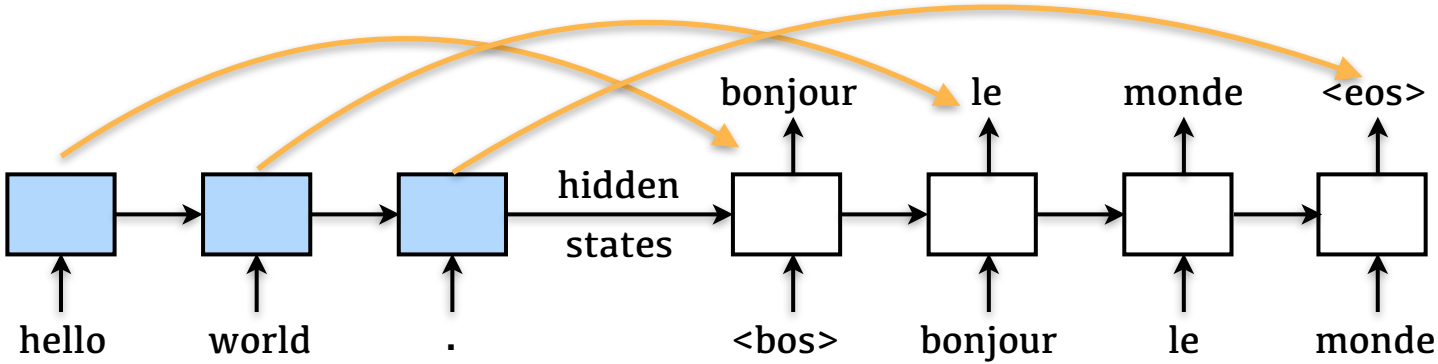
- What if the sequence length approaches infinity?
  - Encoding to a fixed-dimensional vector will have information loss
  - Can never solve the problem due to the memory capacity

Attention



Motivation

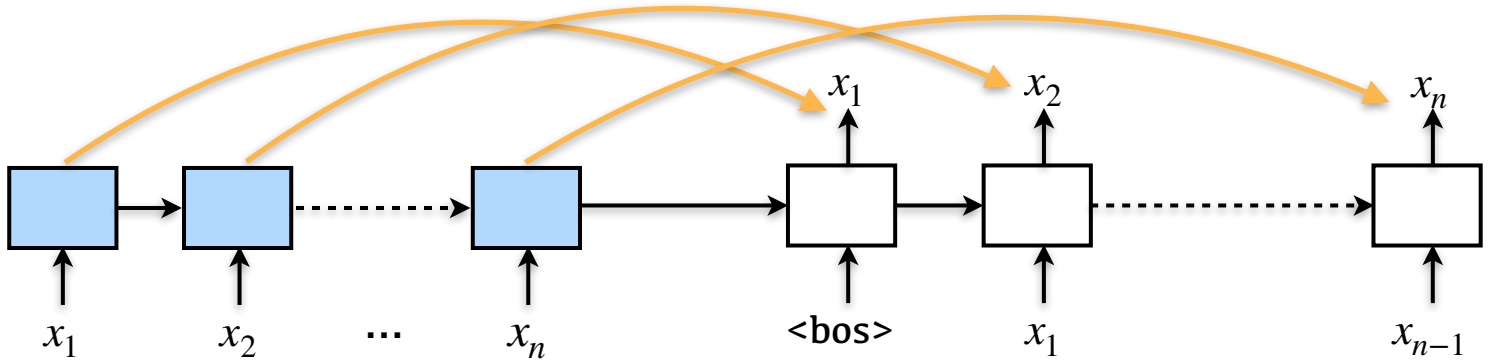
- Each generated token might be related to some specific source tokens.



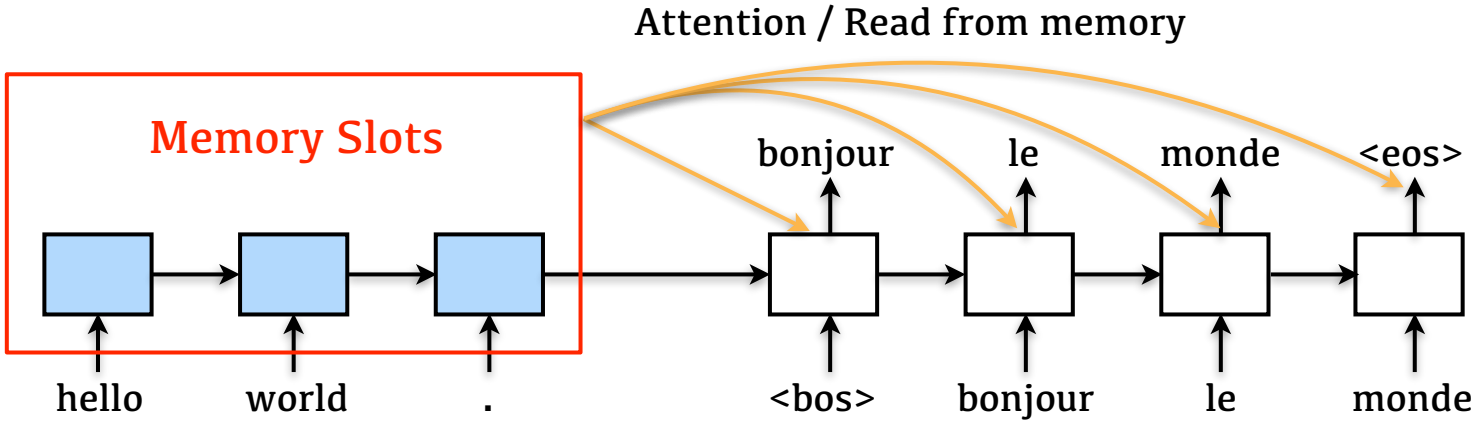
- Instead of just using the last state, all states are used in the decoding process. Search for related states output by encoder.

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

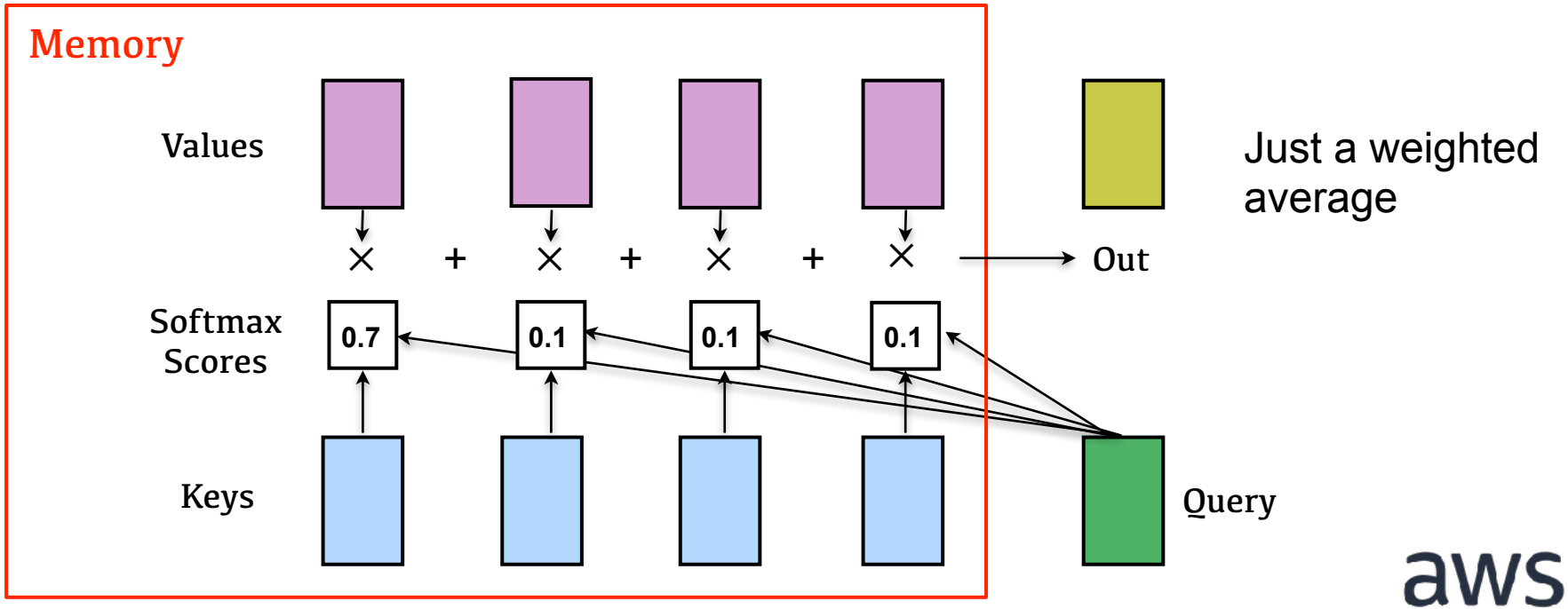


Treat each state in the RNN as memory slots

In each decoding step, the network will search the states that are related to the query.

Modern Interpretation — Key Value Memory Network

Out = Attention(Query, Key, Value)

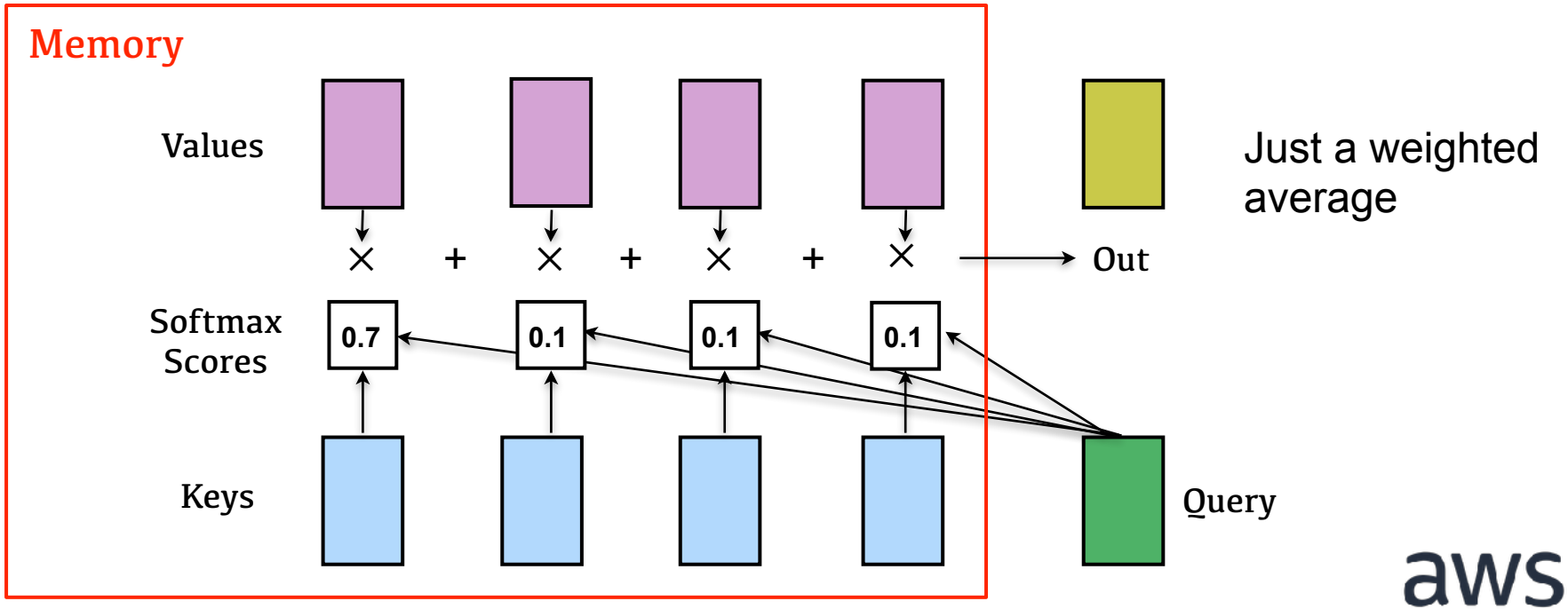


[Sukhbaatar et al. 2015] End-to-end Memory Networks

Modern 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

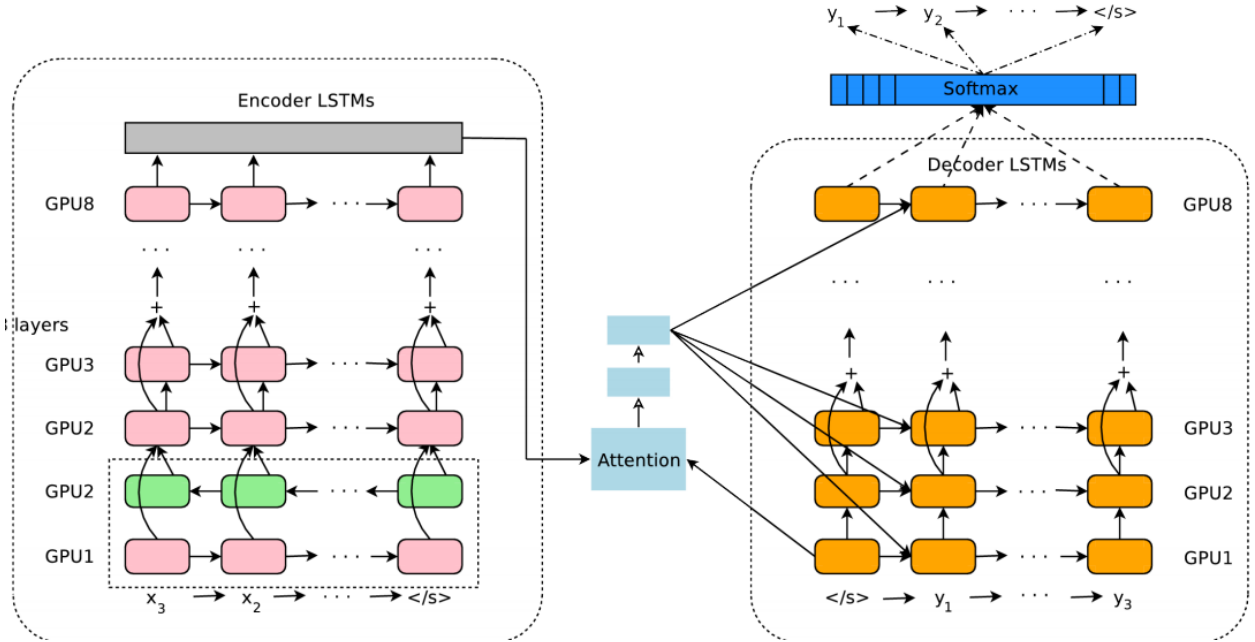
Implement Different Attention Scores in GluonNLP

- Query: q, Keys: $k_{1:n}$

gluonnlp.model.attention_cell

Attention Score Before Softmax	GluonNLP
$s_i = \mathbf{q}^T \mathbf{k}$	DotProductAttentionCell
$s_i = W_o \tanh(W_i[\mathbf{q} \parallel \mathbf{k}] + b)$	MLPAttentionCell

Google Neural Machine Translation



Residual: X + f(X)

Implementation available in GluonNLP

scripts/machine_translation

Encoder: Bidirectional LSTM + Residual

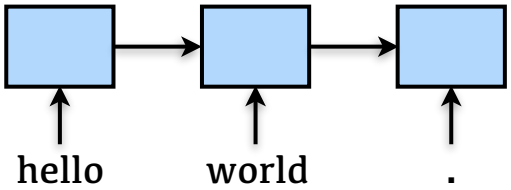
Decoder: LSTM + Residual + Attention

[Wu et al., 2016] Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Transformer

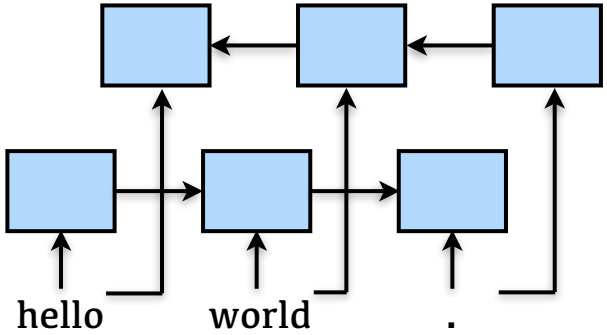
RNN is slow in the encoding phase

RNN



Sequential, Hard to make parallel

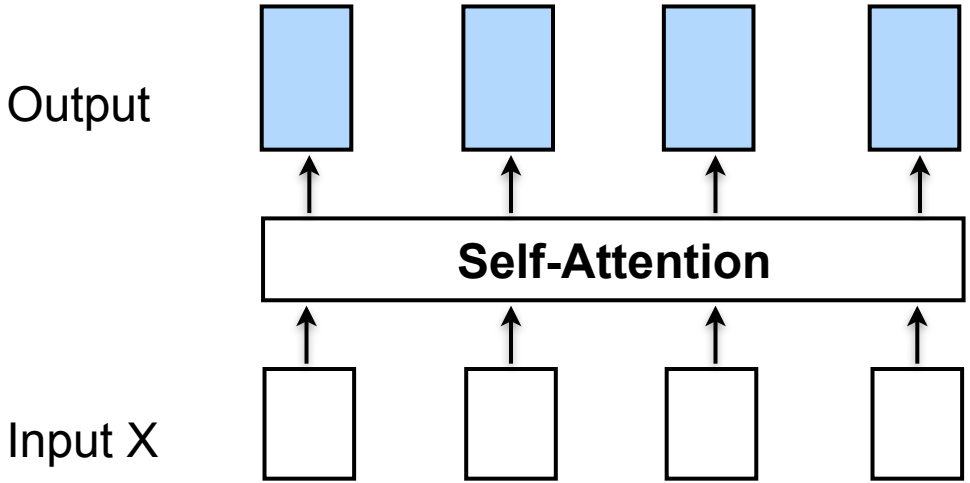
Bidirectional RNN



Self-Attention

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

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



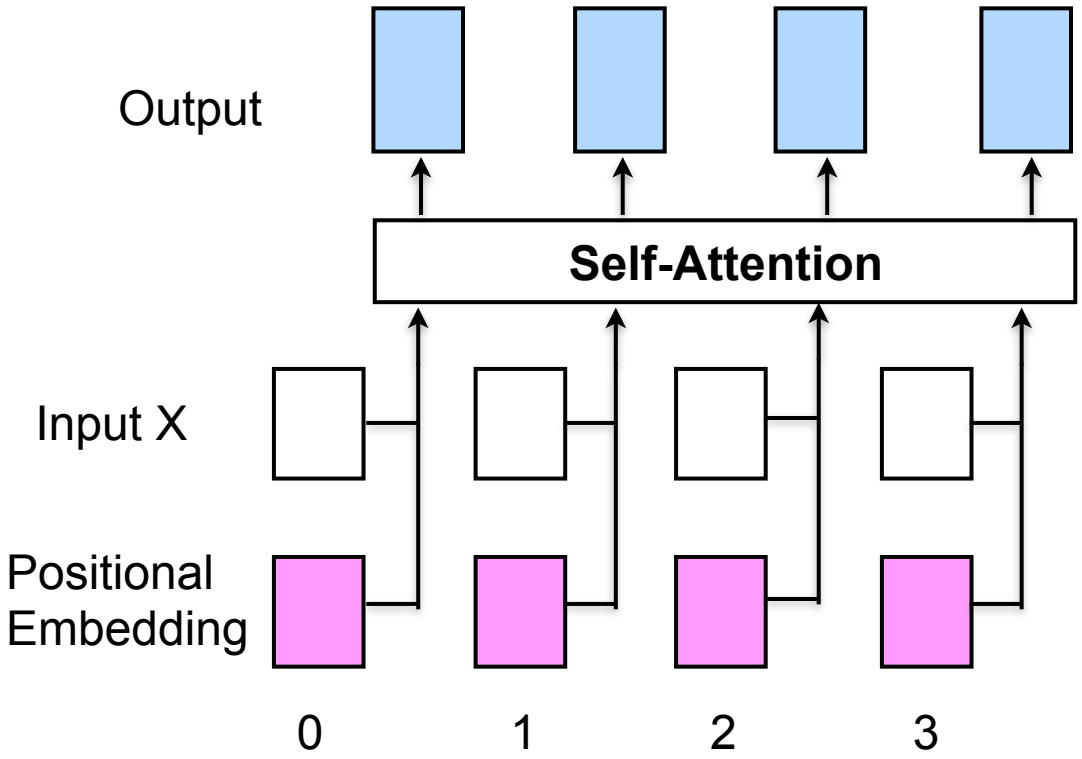
Attention in the encoder.
Run in parallel.

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

Multi-head Attention

How to increase the representational power?

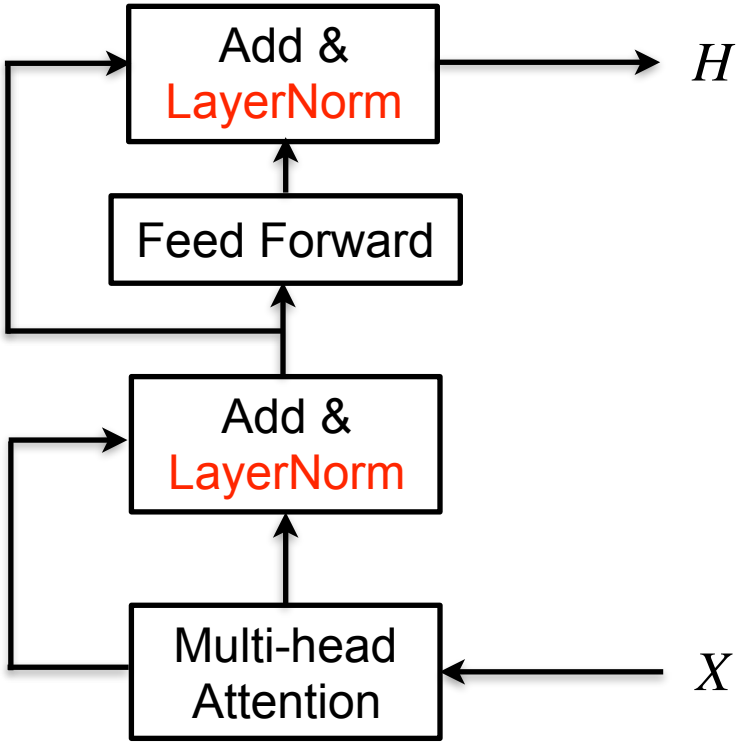
One-head: Similarity measurement in one subspace

Multi-head: Multiple subspaces!

Multi-head Attention:

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

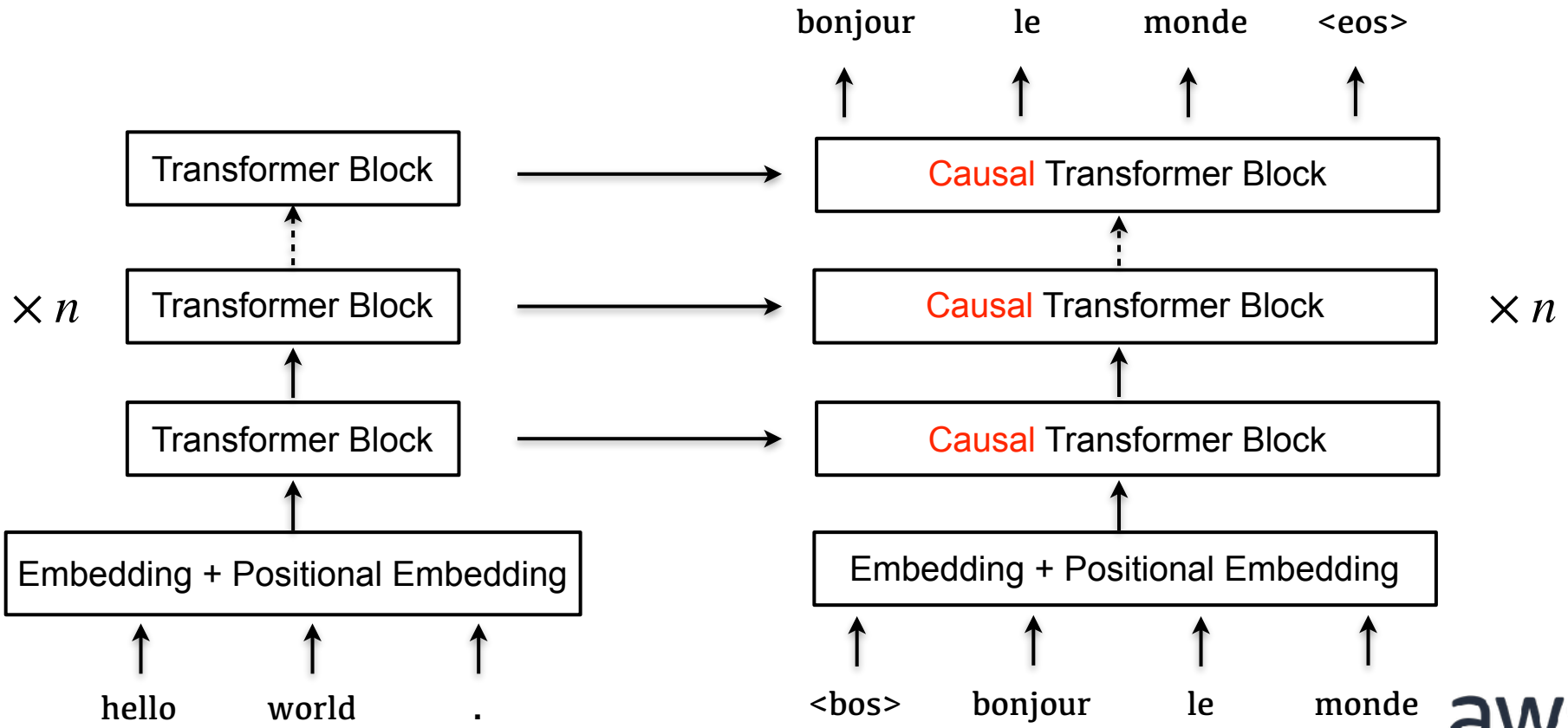
Transformer Block = Multi-head Attention + Residual + Layer Normalization



X.shape = (#Batch, #Channels)

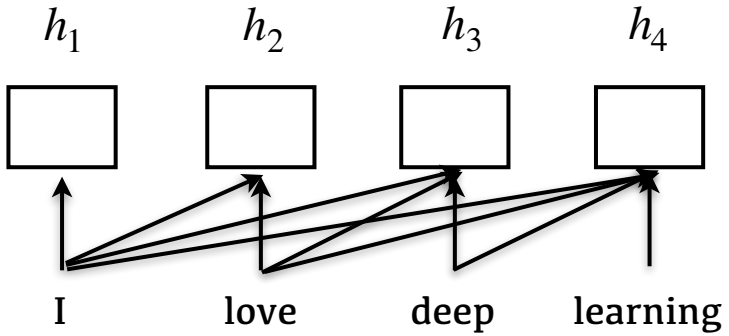
$$\text{LayerNorm: } \frac{\text{X} - \text{X.mean(axis=-1)}}{\text{X.std(axis=-1)}}$$

Transformer Architecture for Machine Translation



Causal Attention

- Never attend to the future. Avoid information leak.



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

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

Search-based Algorithms for Inference

$$\arg\max_{y_{1:m}} \log p(y_{1:m} \mid x_{1:n}) \xrightarrow{\text{Greedy}} \arg\max_{y_i} \log p\left(y_i \mid y_{1:i-1}, x_{1:n}\right)$$

$O(|V|T)$ Accumulative Error

Search-based Algorithms for Inference

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

Greedy →

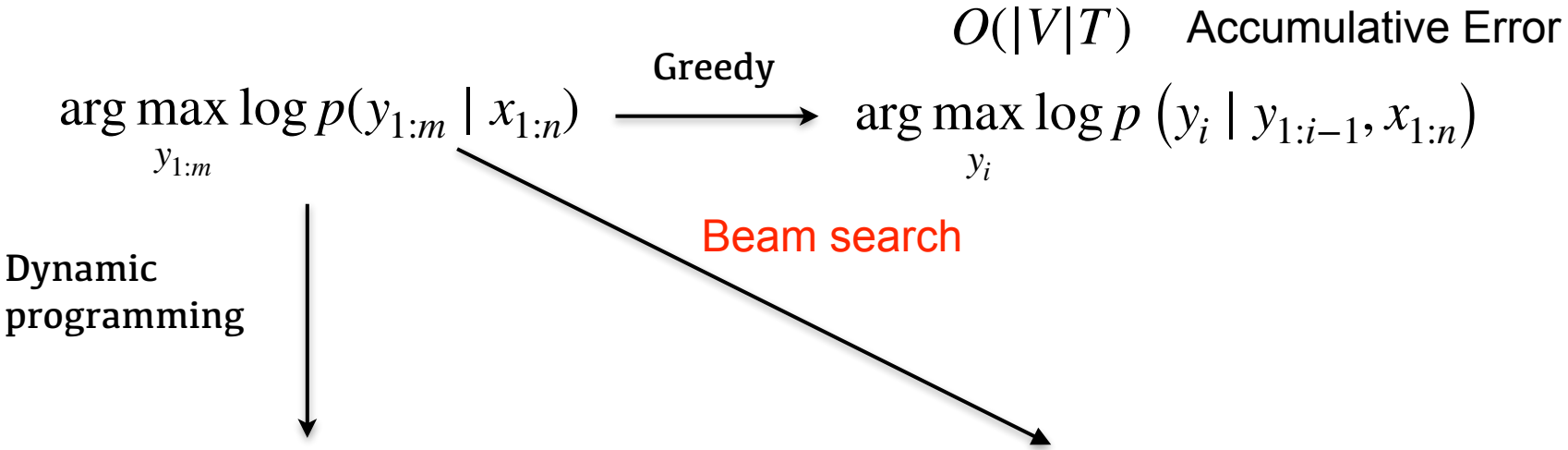
$$\arg\max_{y_i} \log p\left(y_i \mid y_{1:i-1}, x_{1:n}\right)$$

$O(|V|T)$ Accumulative Error

Dynamic programming ↓

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

Search-based Algorithms for Inference



Length Penalty

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

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!

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.

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

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

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

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

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

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

How to evaluate the prediction?

- Bilingual Evaluation Understudy (BLEU) Score
- Based on a modified version of n-gram precision
- Not go to the details, rely on sacreBLEU to evaluate

[Papineni et al., 2002] BLEU: a Method for Automatic Evaluation of Machine Translation

Demo: Machine Translation

04_machine_translation/transformer.ipynb