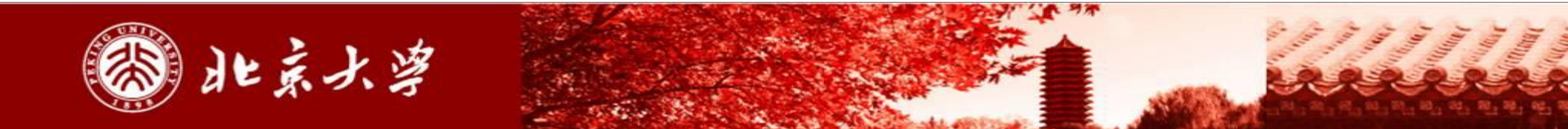
《物理与人工智能》

13. 注意力机制与Transformer

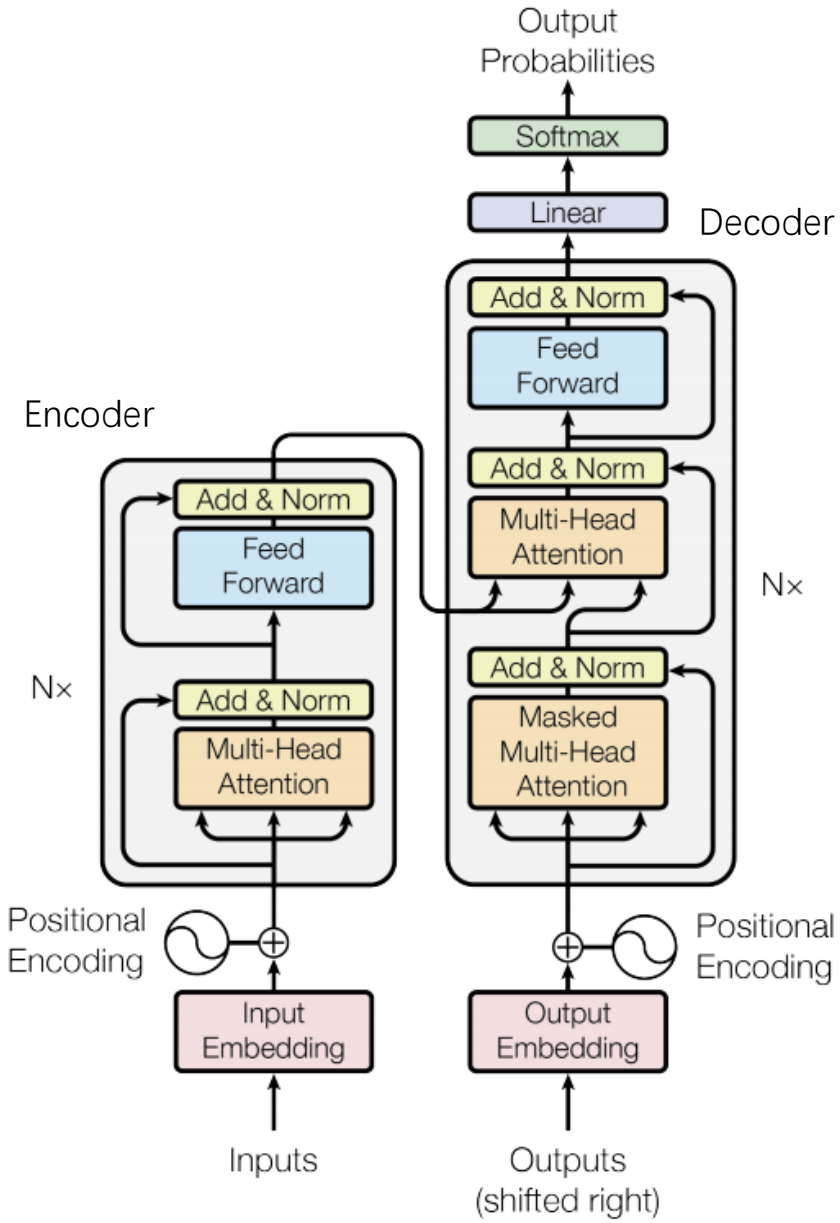
授课教师：马滟青

2025/10/20（第六周）

鸣谢：基于slazebni幻灯片



Attention is all you need!



Outline

- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Transformers

Sequence-to-sequence modeling with RNNs

北京大学 PEKING UNIVERSITY

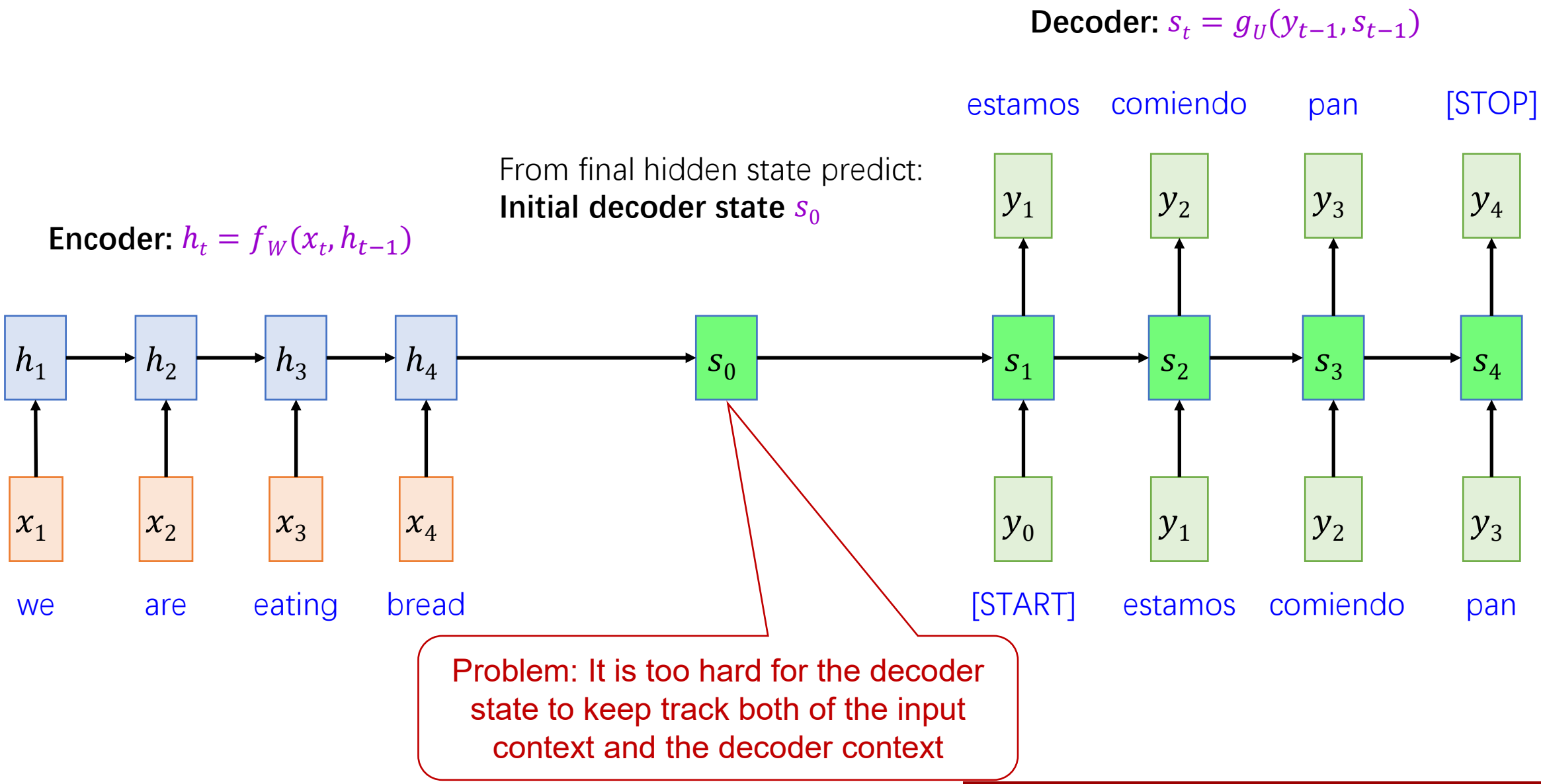
- "We are eating bread"

"Estamos comiendo pan"

I. Sutskever, O. Vinyals, Q. Le, Sequence to Sequence Learning with Neural Networks, NeurIPS 2014

K. Cho, B. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014

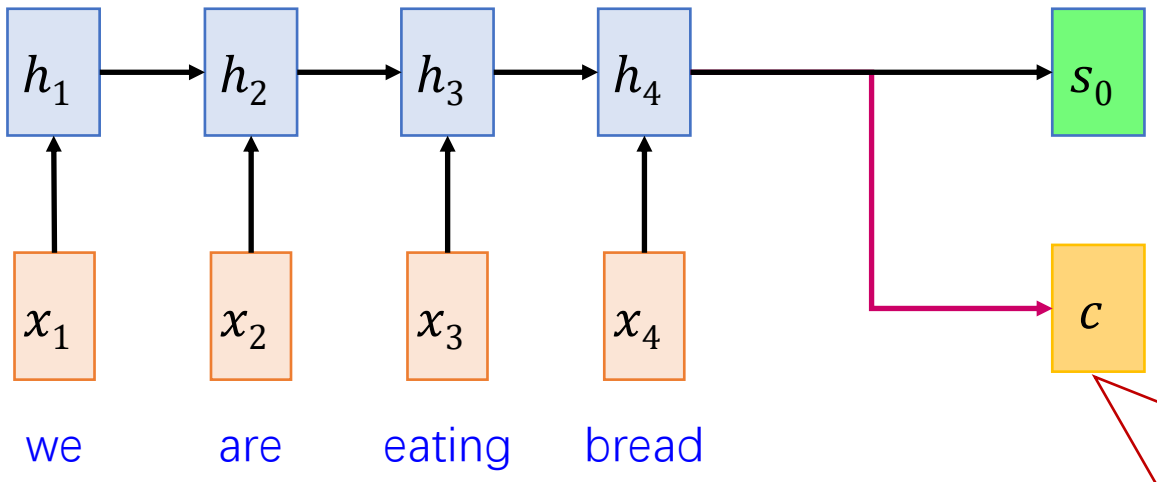
Sequence-to-sequence modeling with RNNs



Sequence-to-sequence modeling with RNNs

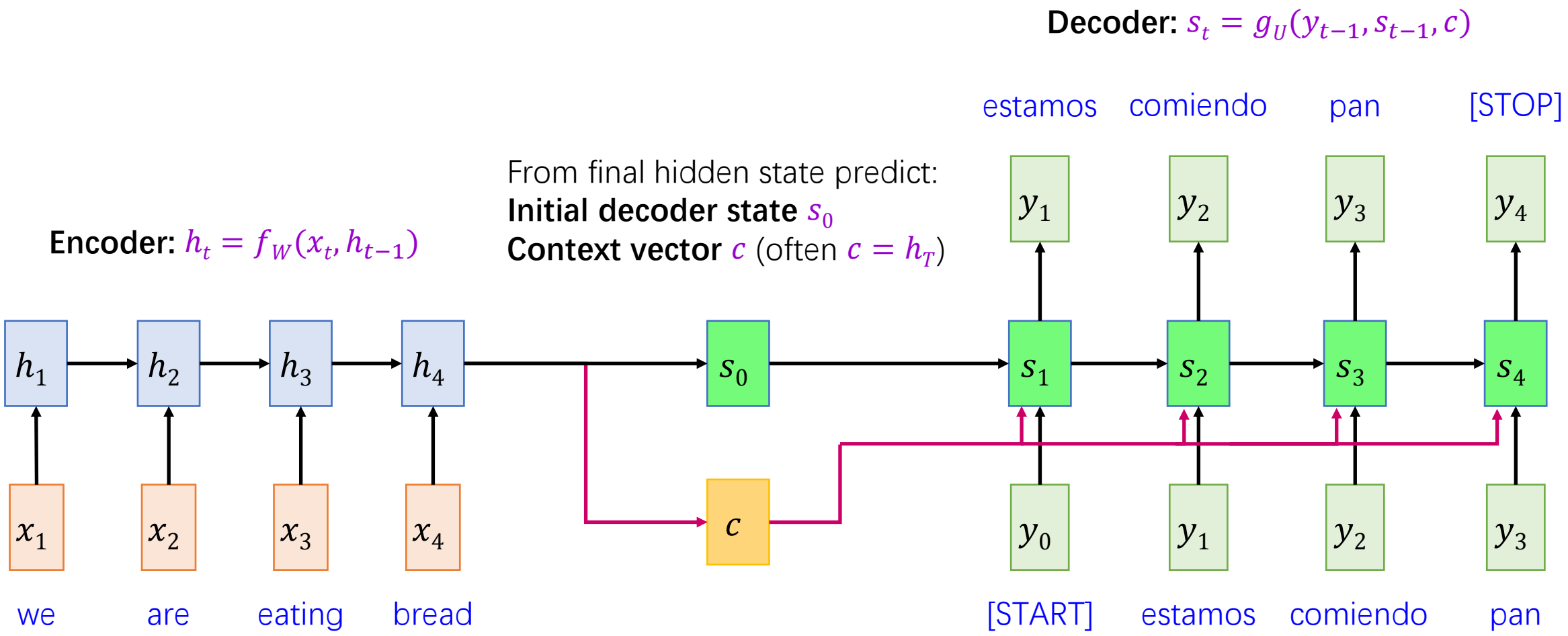
Encoder: $h_t = f_W(x_t, h_{t-1})$

From final hidden state predict:
Initial decoder state s_0
Context vector c (often $c = h_T$)

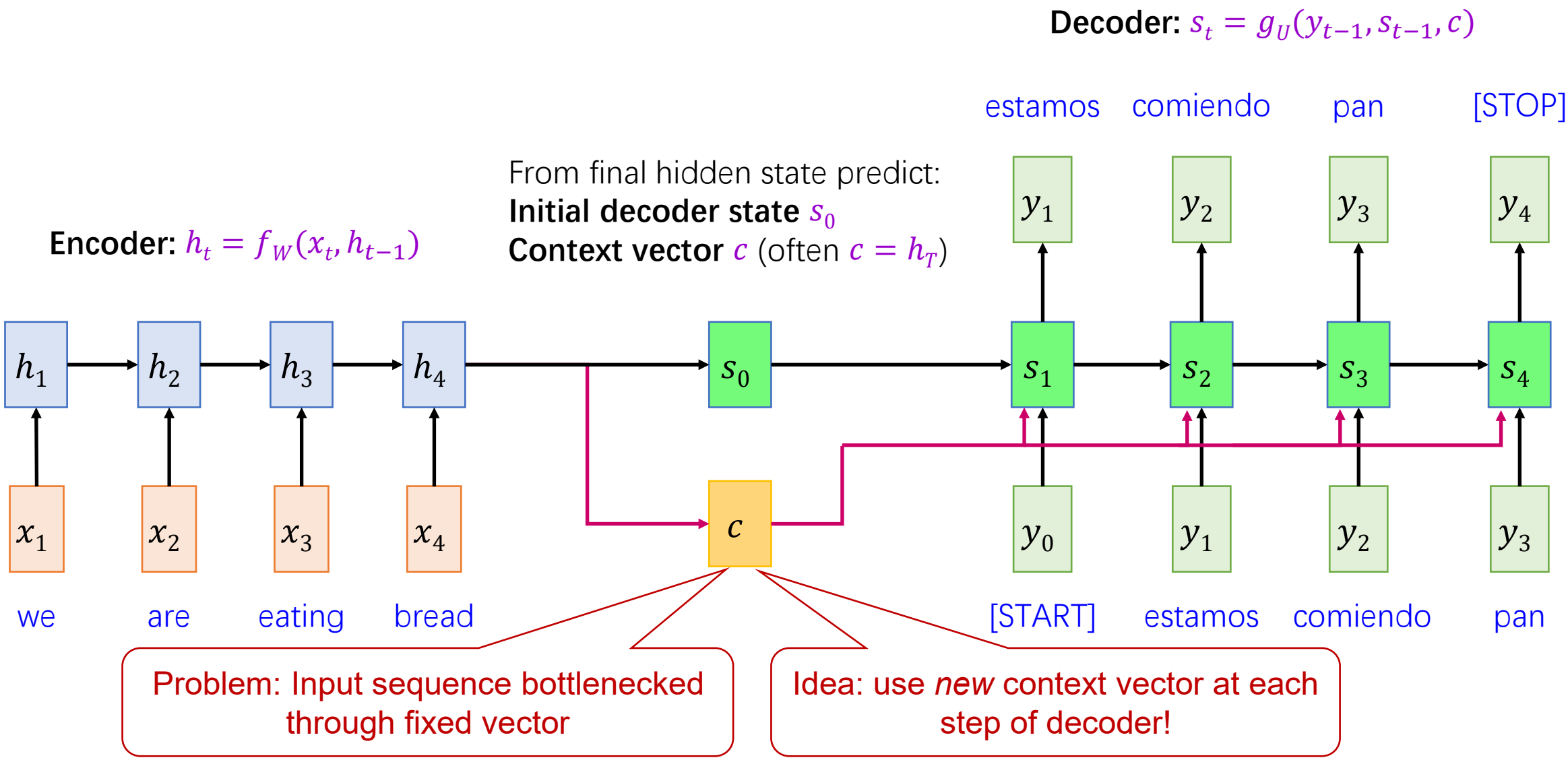


Solution: Introduce a separate context vector to "remember" the input sequence

Sequence-to-sequence modeling with RNNs



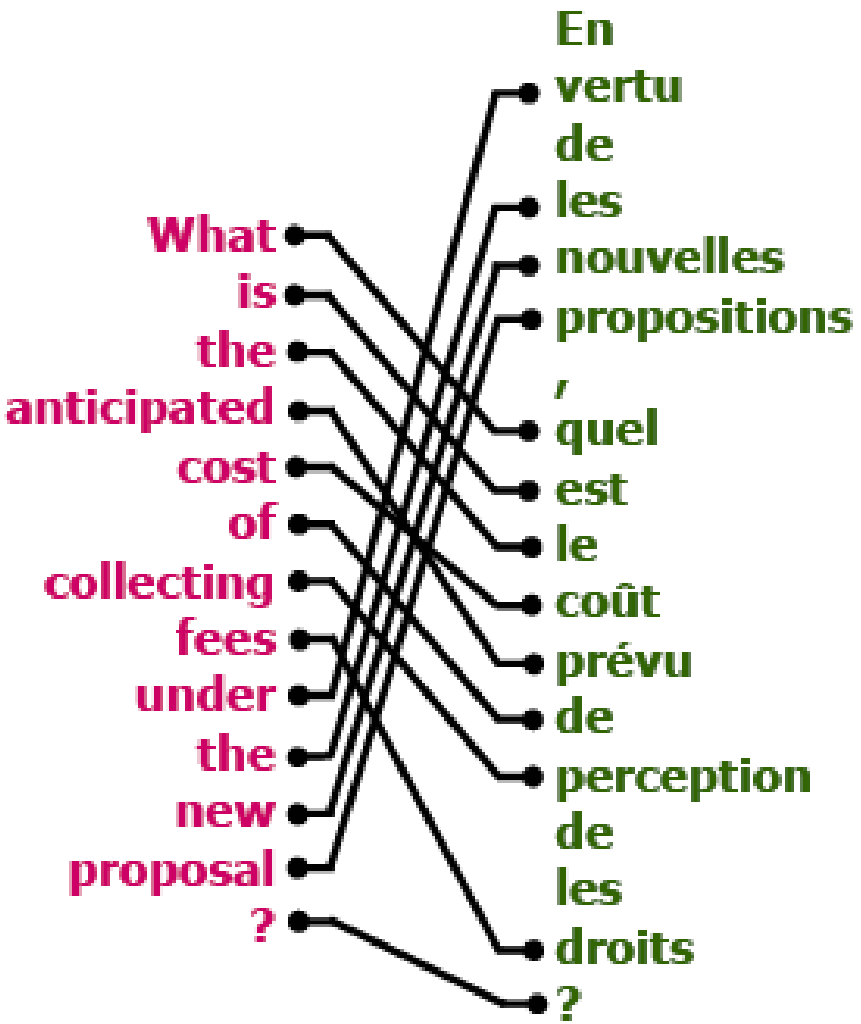
Sequence-to-sequence modeling with RNNs



Sequence-to-sequence w/ RNNs *and attention*

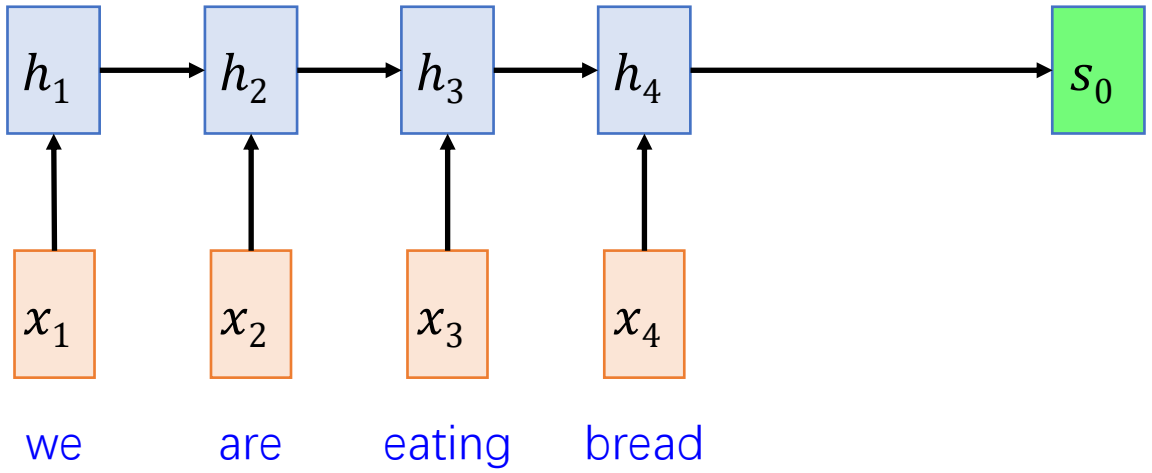
北京大学 PEKING UNIVERSITY

- Intuition: translation requires *alignment*

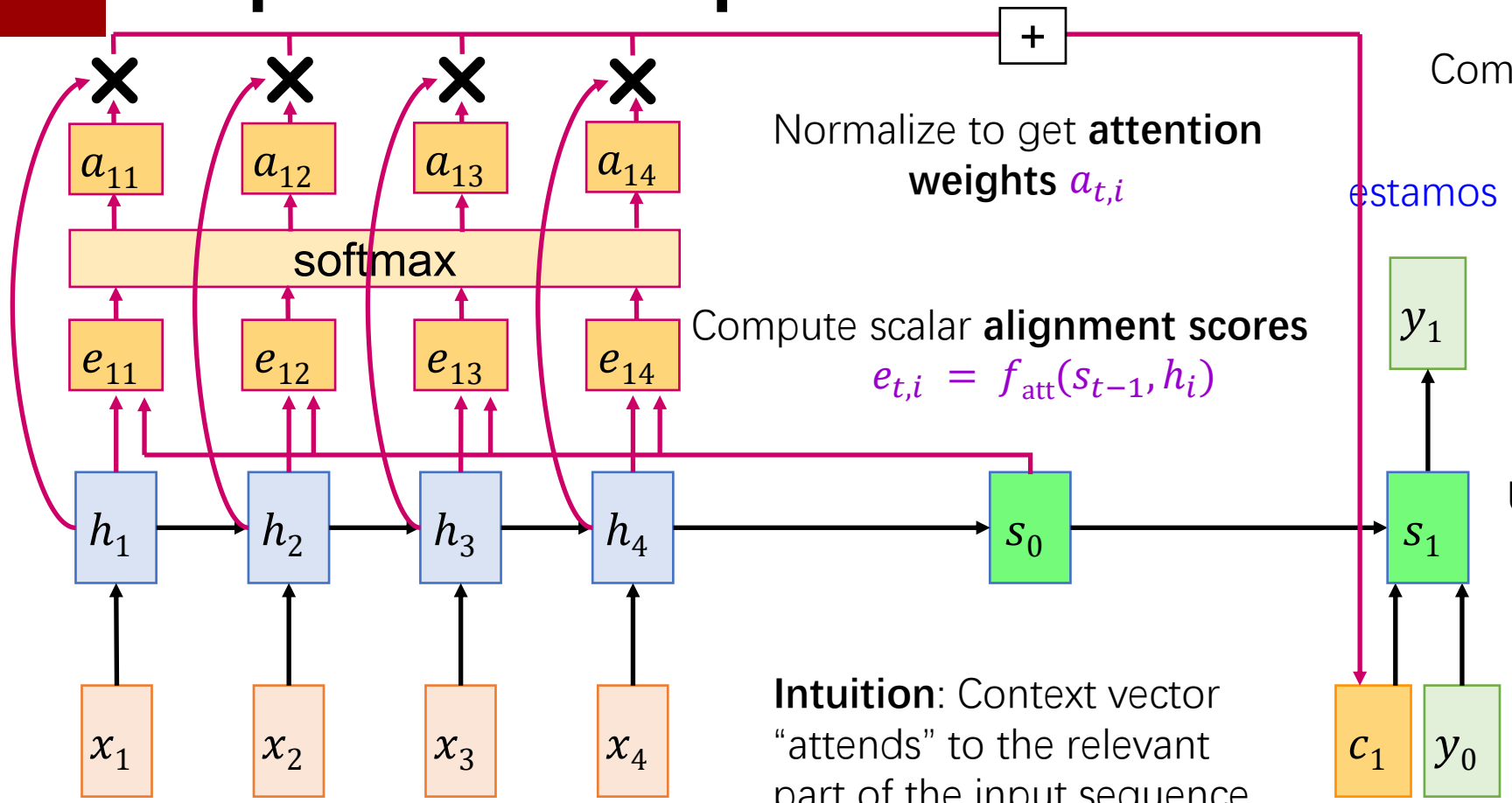


Sequence-to-sequence w/ RNNs and attention

- At each timestep of decoder, context vector “looks at” different parts of the input sequence



Sequence-to-sequence w/ RNNs and attention



Normalize to get **attention weights** $a_{t,i}$

Compute scalar **alignment scores**

$$e_{t,i} = f_{\text{att}}(s_{t-1}, h_i)$$

Compute **context vector** as

$$c_t = \sum_i a_{t,i} h_i$$

Use context vector in decoder:

$$s_t = g_U(y_{t-1}, s_{t-1}, c_t)$$

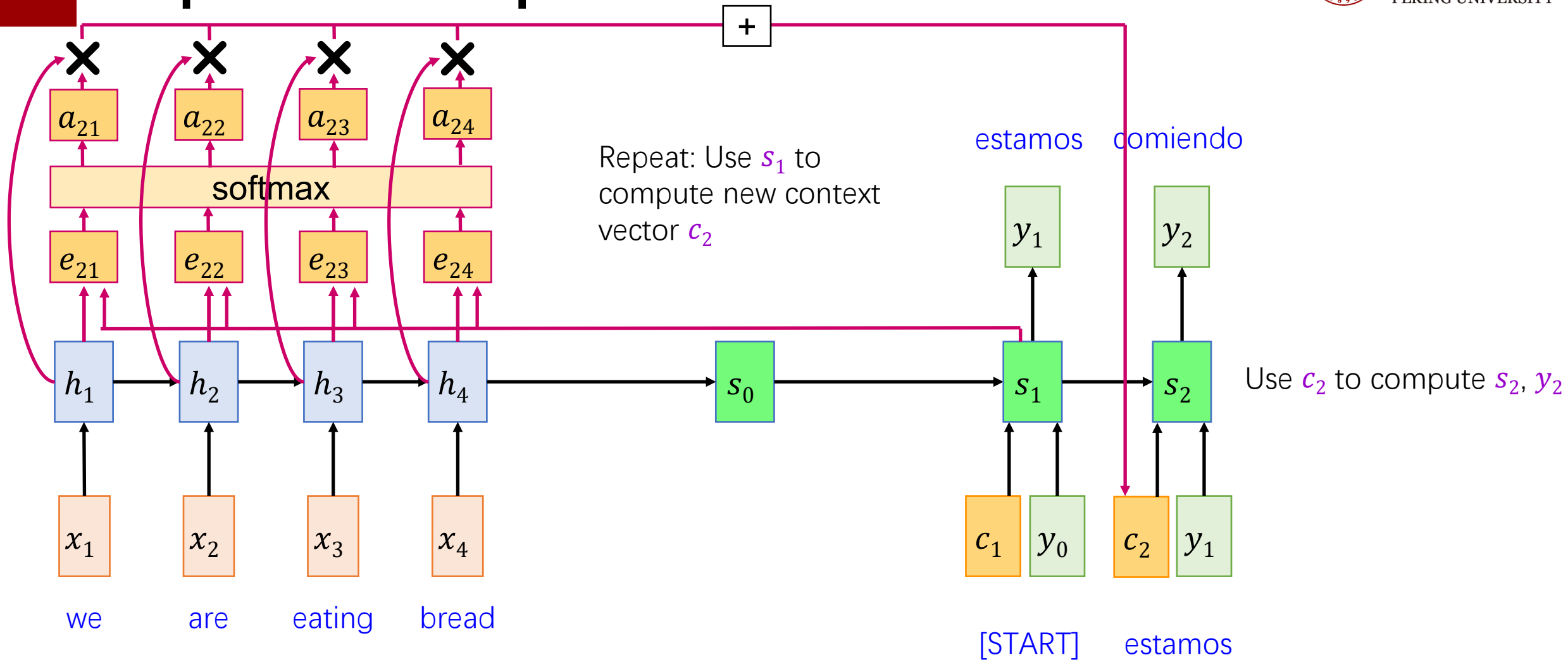
Intuition: Context vector "attends" to the relevant part of the input sequence

"estamos" = *"we are"*

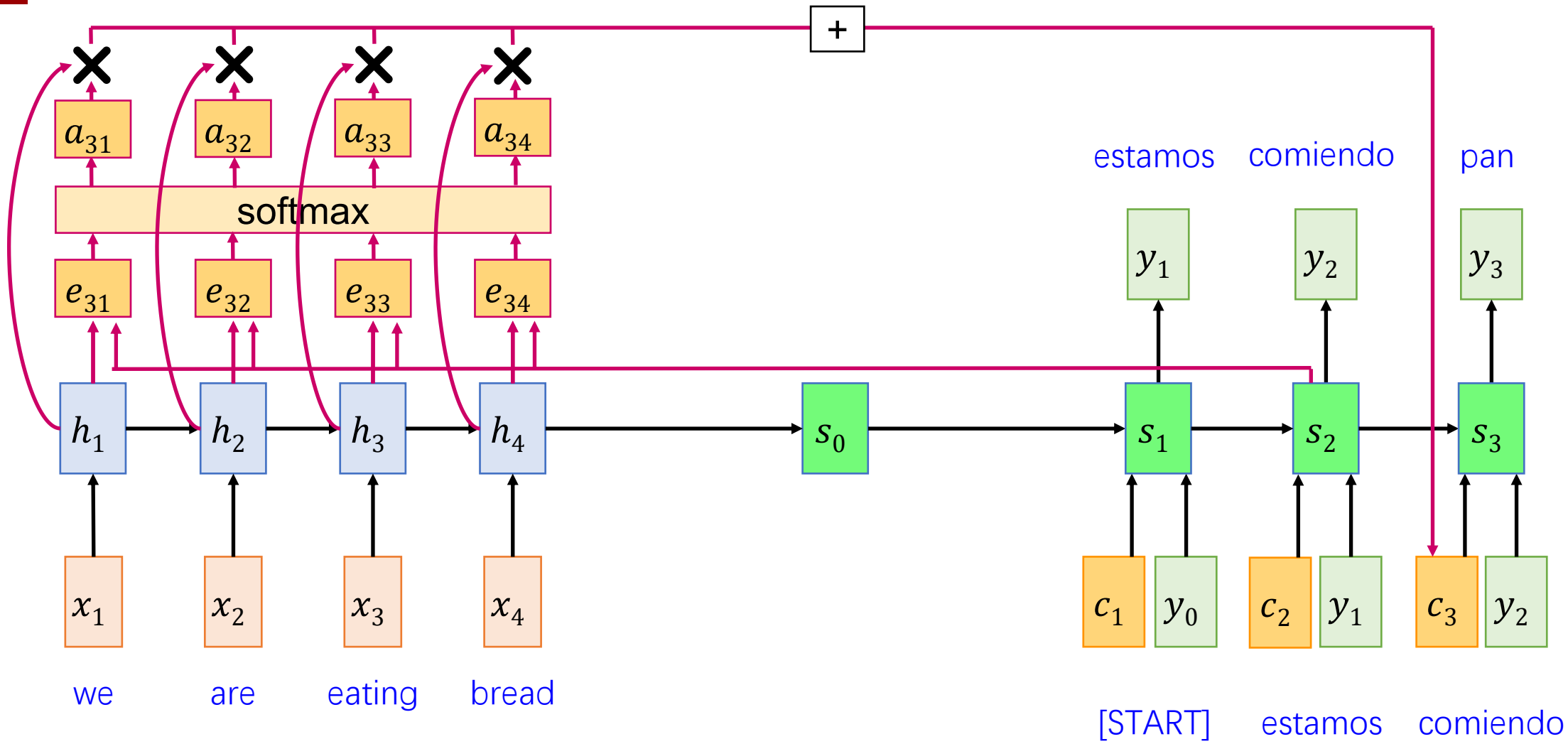
so maybe $a_{11} = a_{12} = 0.45$, $a_{13} = a_{14} = 0.05$

D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

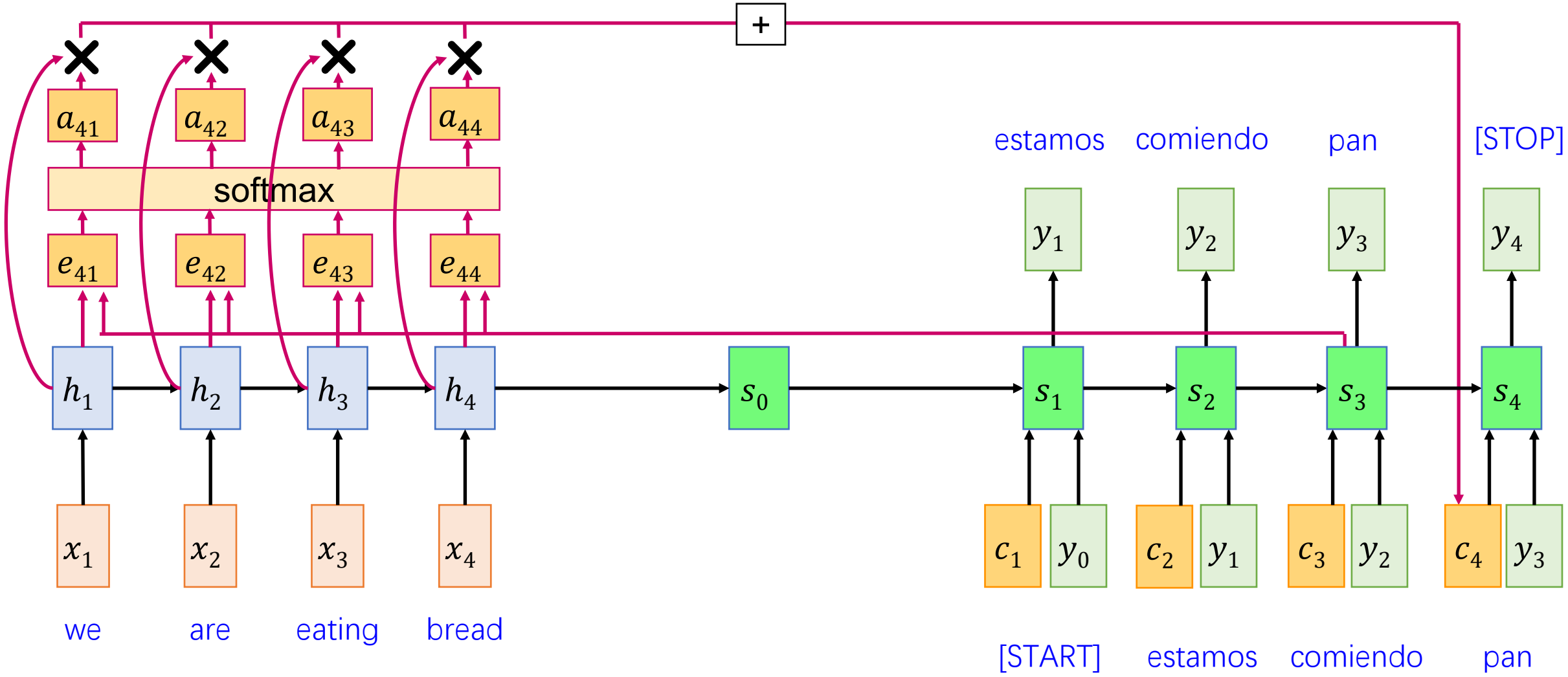
Sequence-to-sequence w/ RNNs and attention



Sequence-to-sequence w/ RNNs and attention



Sequence-to-sequence w/ RNNs and attention



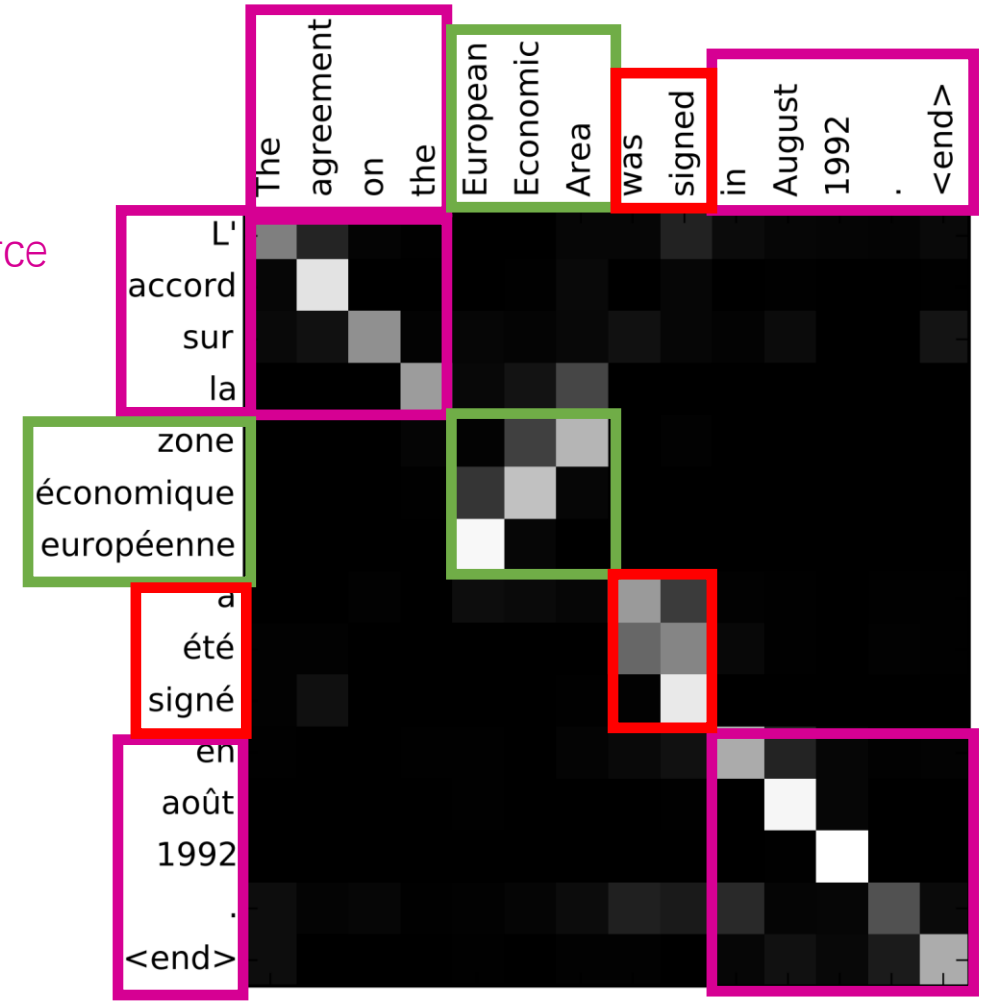
Sequence-to-sequence w/ RNNs and attention

- Visualizing attention weights (English source, French target):

Same word order in source and target languages

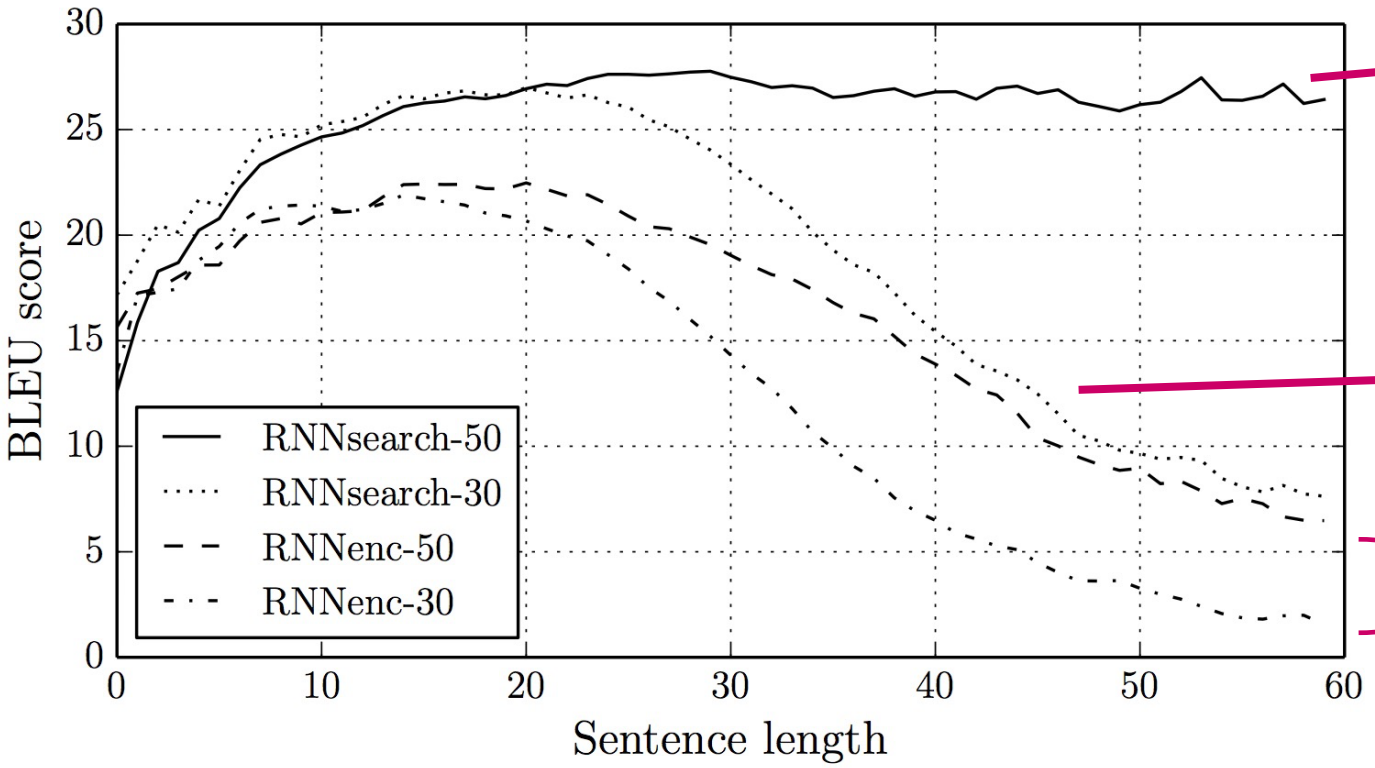
Flipped word order

Verb conjugation is different



D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

Quantitative evaluation



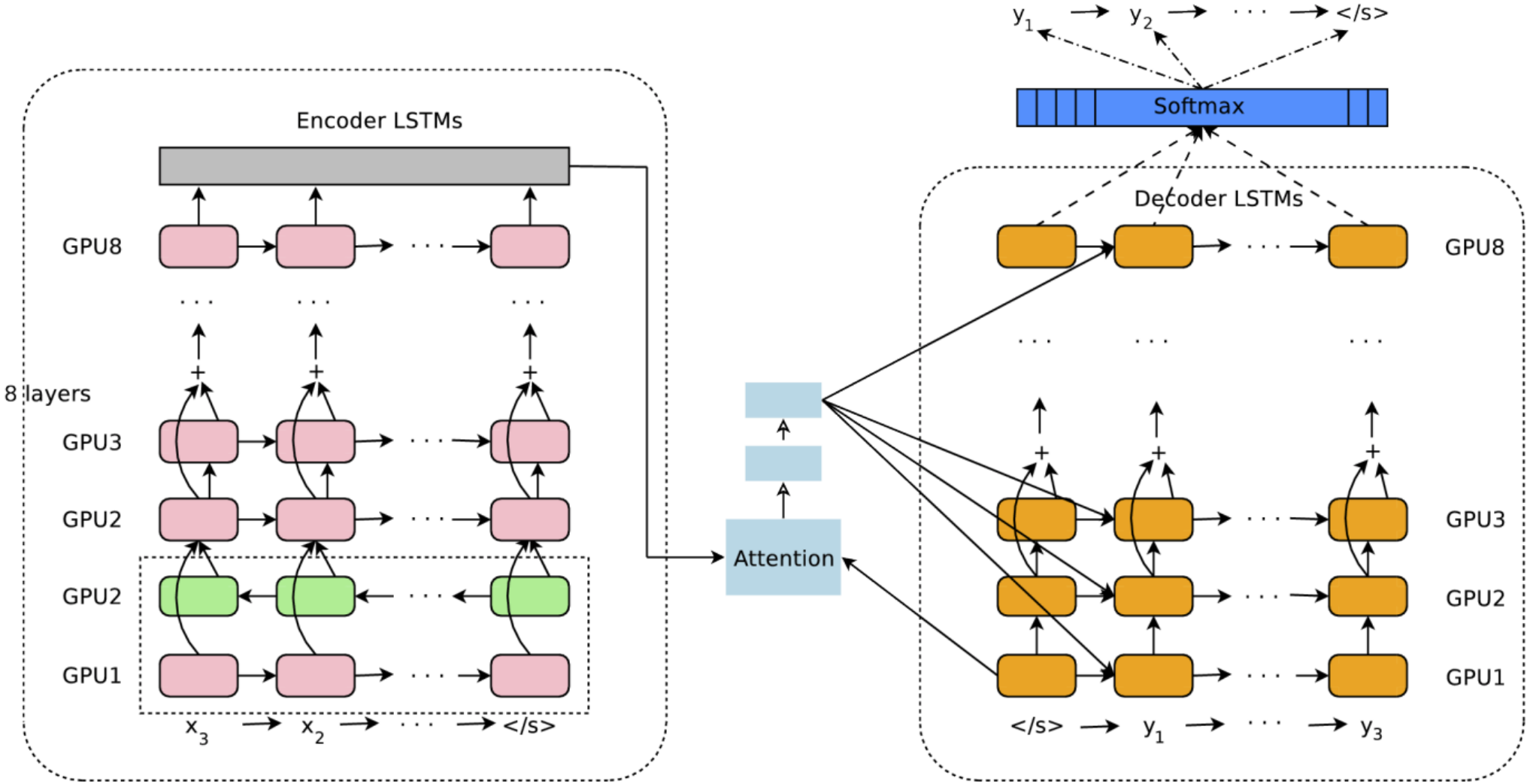
With attention (trained with sentence length <= 50)

With attention (trained with sentence length <= 30)

No attention

D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015

Google Neural Machine Translation (GNMT)



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

Google Neural Machine Translation (GNMT)

北京大学 PEKING UNIVERSITY

- Human evaluation results on production data (500 randomly sampled sentences from Wikipedia and news websites)

Table 10: Mean of side-by-side scores on production data

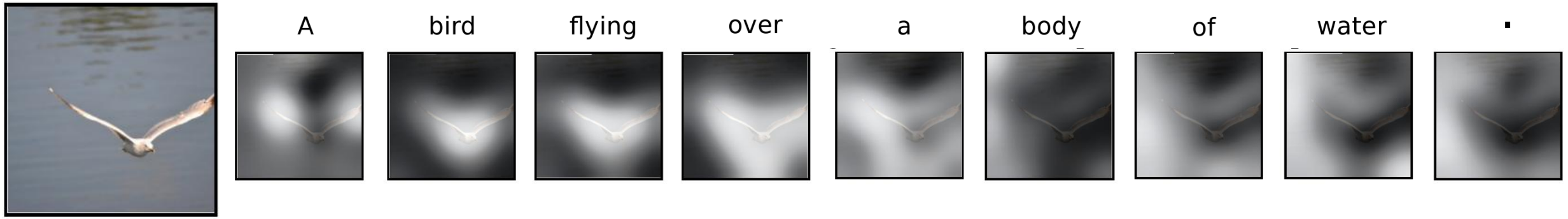
	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.550	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

Side-by-side scores: range from 0 ("completely nonsense translation") to 6 ("perfect translation"), produced by human raters fluent in both languages

PBMT: Translation by phrase-based statistical translation system used by Google
GNMT: Translation by GNMT system
Human: Translation by humans fluent in both languages

Image captioning with RNNs and attention

- Idea: pay attention to different parts of the image when generating different words
- Automatically learn this *grounding* of words to image regions without direct supervision



K. Xu et al., Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Outline

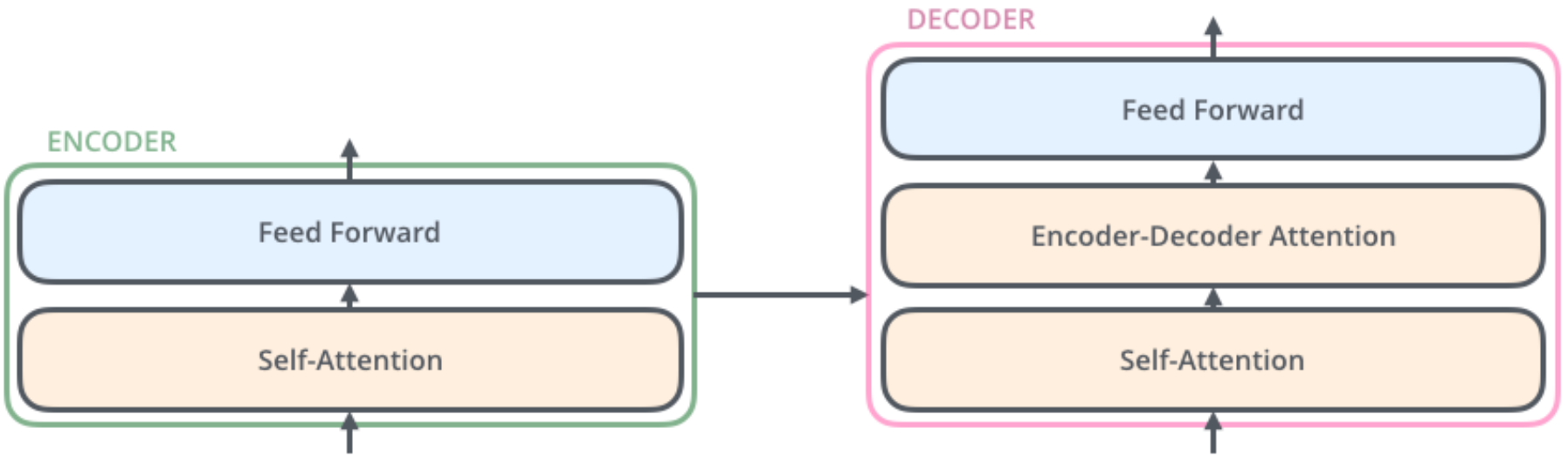
- Vanilla seq2seq with RNNs
- Seq2seq with RNNs and attention
- Transformers

Basic transformer model

- Sequence-to-sequence architecture using *only point-wise processing and attention* – no recurrent units or convolutions

Encoder: receives entire input sequence and outputs encoded sequence of the same length

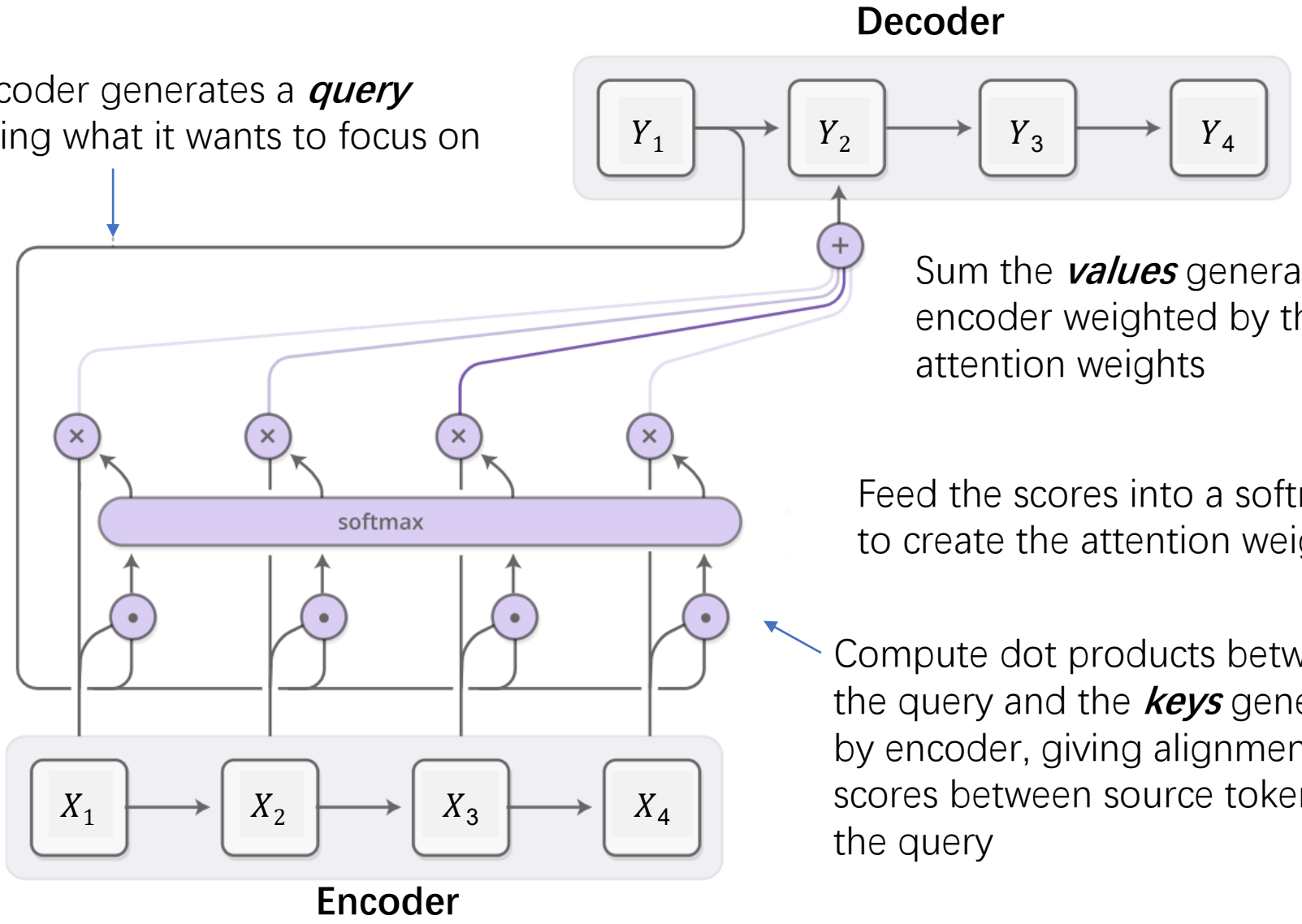
Decoder: predicts next token conditioned on encoder output and previously predicted tokens



A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, NeurIPS 2017

Image source

Key-Value-Query attention model



The decoder generates a ***query*** describing what it wants to focus on

Sum the ***values*** generated by encoder weighted by the attention weights

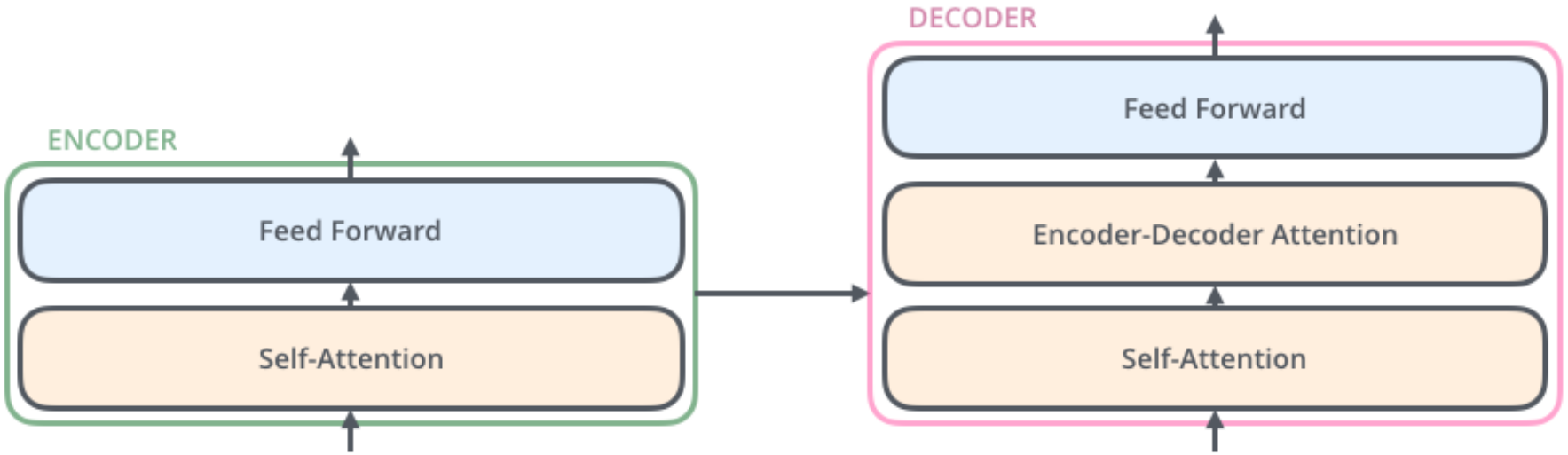
Feed the scores into a softmax to create the attention weights

Compute dot products between the query and the ***keys*** generated by encoder, giving alignment scores between source tokens and the query

Image source

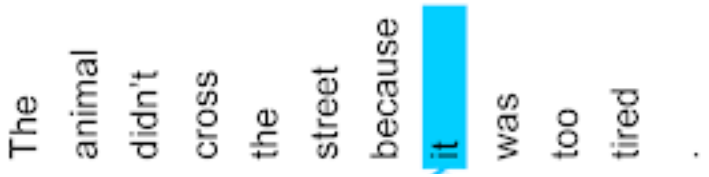
Attention mechanisms

- **Encoder self-attention:** queries, keys, and values come from previous layer of encoder
- **Decoder self-attention:** values corresponding to future decoder outputs are masked out
- **Encoder-decoder attention:** queries come from previous decoder layer, keys and values come from output of encoder

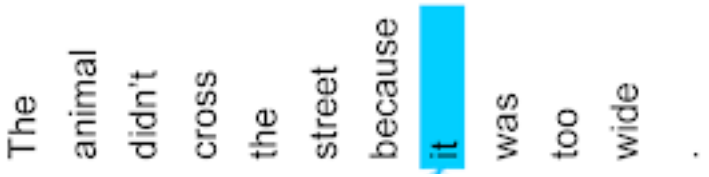


Self-attention

- Used to capture context *within the sequence*



As we are encoding “it”, we should focus on “the animal”



As we are encoding “it”, we should focus on “the street”

Image source

Self-attention layer

- Query vectors: $Q = XW_Q$
- Key vectors: $K = XW_K$
- Value vectors: $V = XW_V$
- Similarities: *scaled dot-product attention*

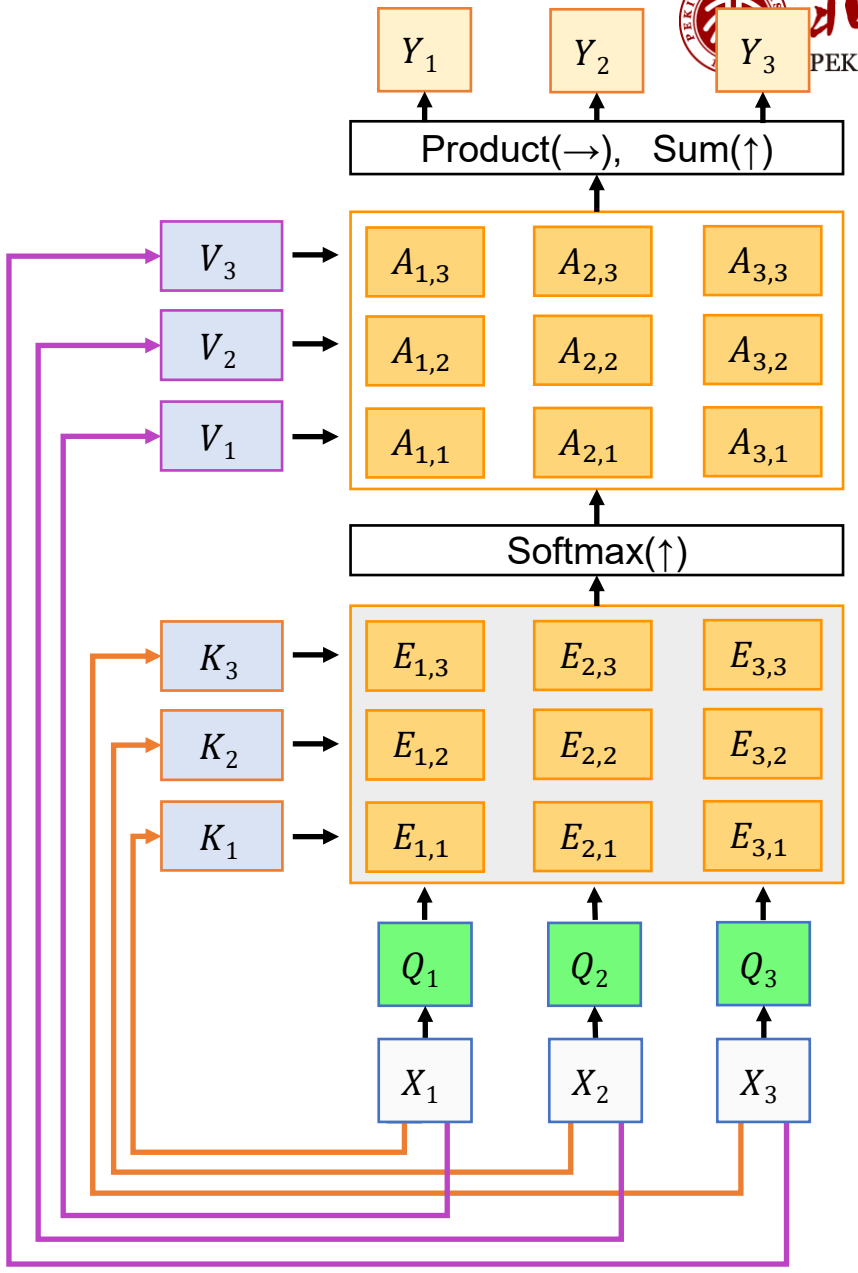
$$E_{i,j} = \frac{(Q_i \cdot K_j)}{\sqrt{D}} \text{ or } E = QK^T/\sqrt{D}$$

(D is the dimensionality of the keys)

- Attn. weights: $A = \text{softmax}(E, \text{dim} = 1)$
- Output vectors:

$$Y_i = \sum_j A_{i,j} V_j \text{ or } Y = AV$$

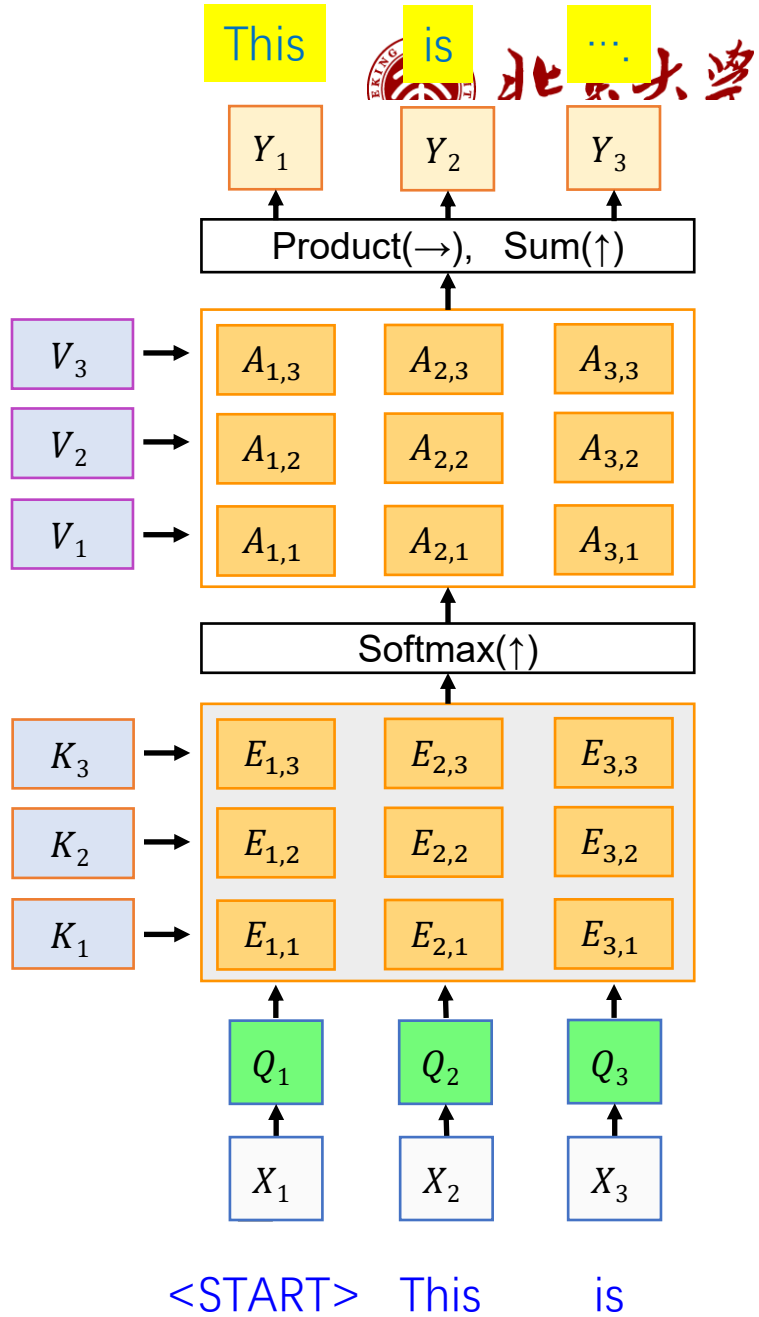
One query per input vector



Adapted from J. Johnson

Masked self-attention layer

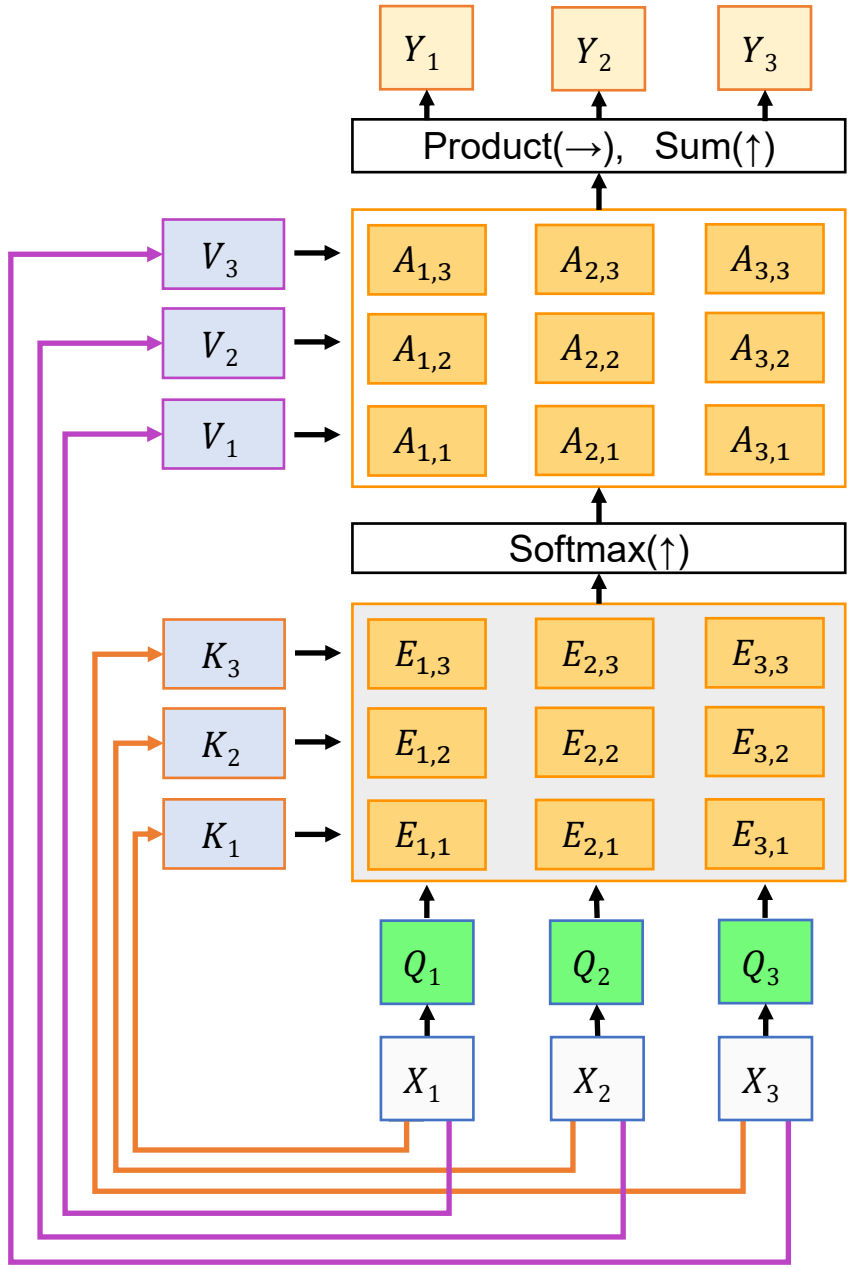
- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

Masked self-attention layer

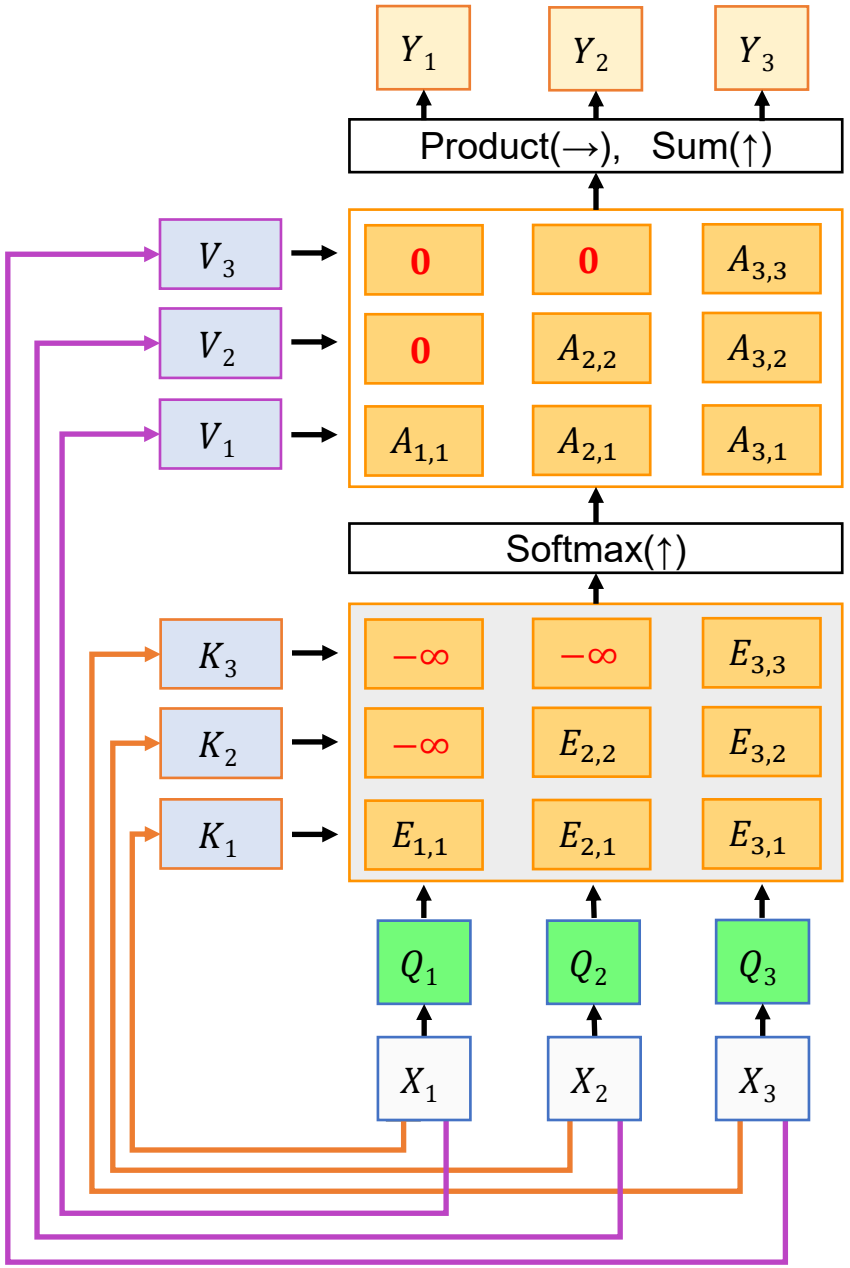
- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

Masked self-attention layer

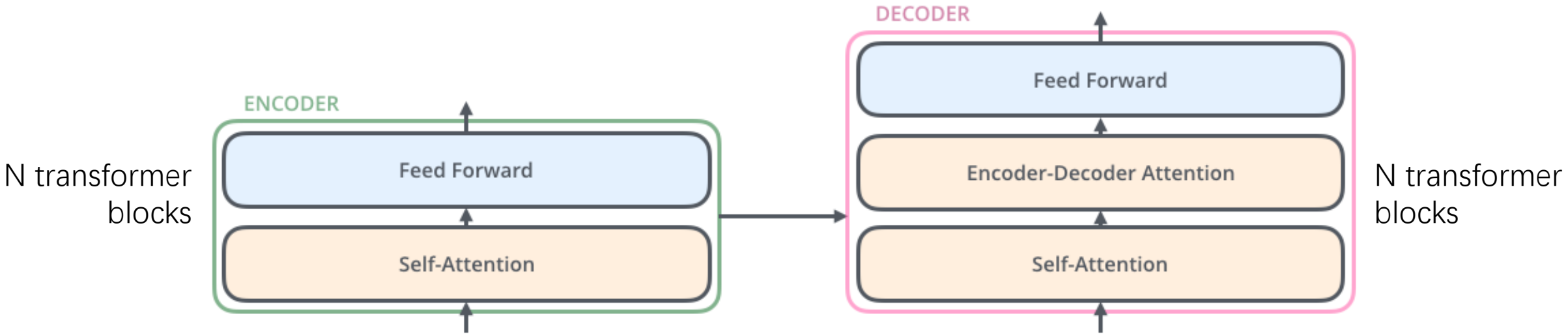
- The decoder should not “look ahead” in the output sequence



Adapted from J. Johnson

Attention mechanisms: Summary

- **Encoder self-attention:** queries, keys, and values come from previous layer of encoder
- **Decoder self-attention:** values corresponding to future decoder outputs are masked out
- **Encoder-decoder attention:** queries come from previous decoder layer, keys and values come from output of encoder

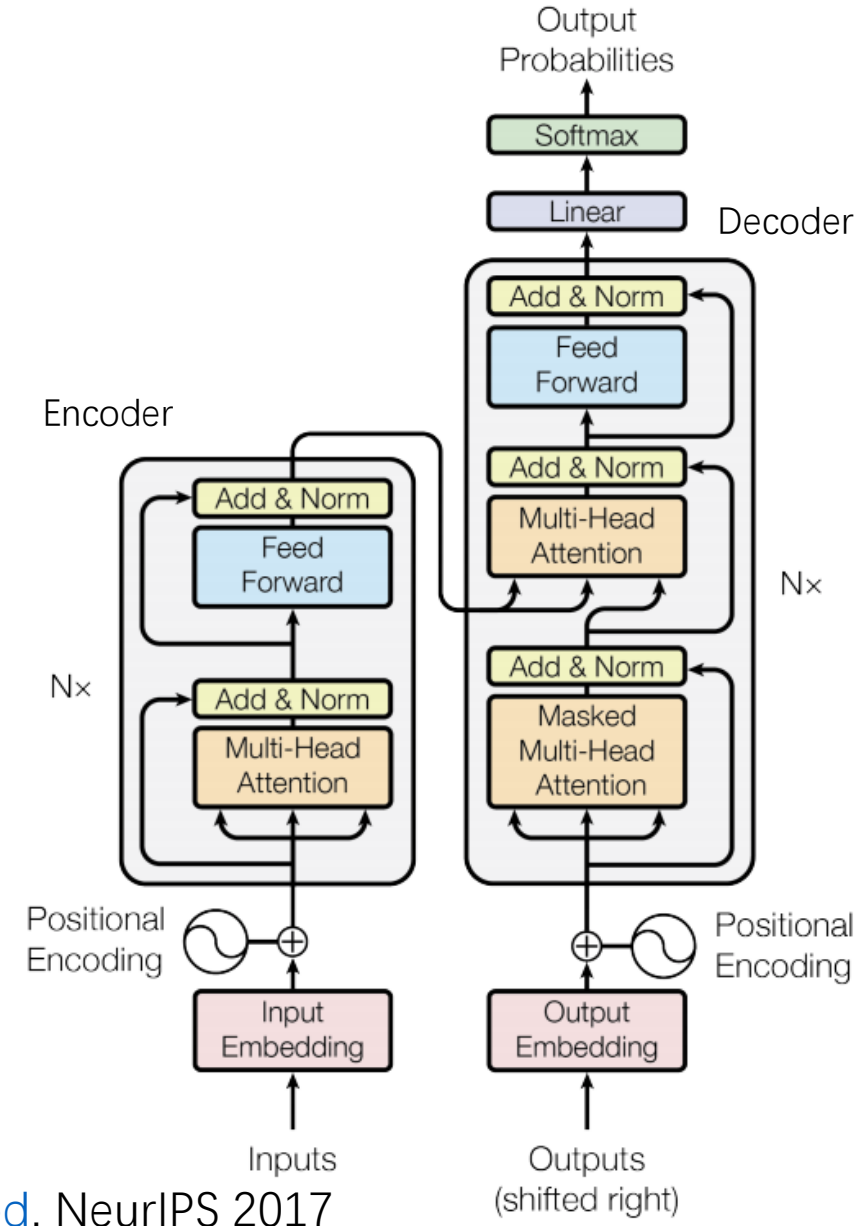


Attention mechanisms: Illustration

北京大学 PEKING UNIVERSITY

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformer architecture: Details



A. Vaswani et al., Attention is all you need, NeurIPS 2017

Positional encoding

- To give transformer information about ordering of tokens, add function of position (based on sines and cosines) to every input

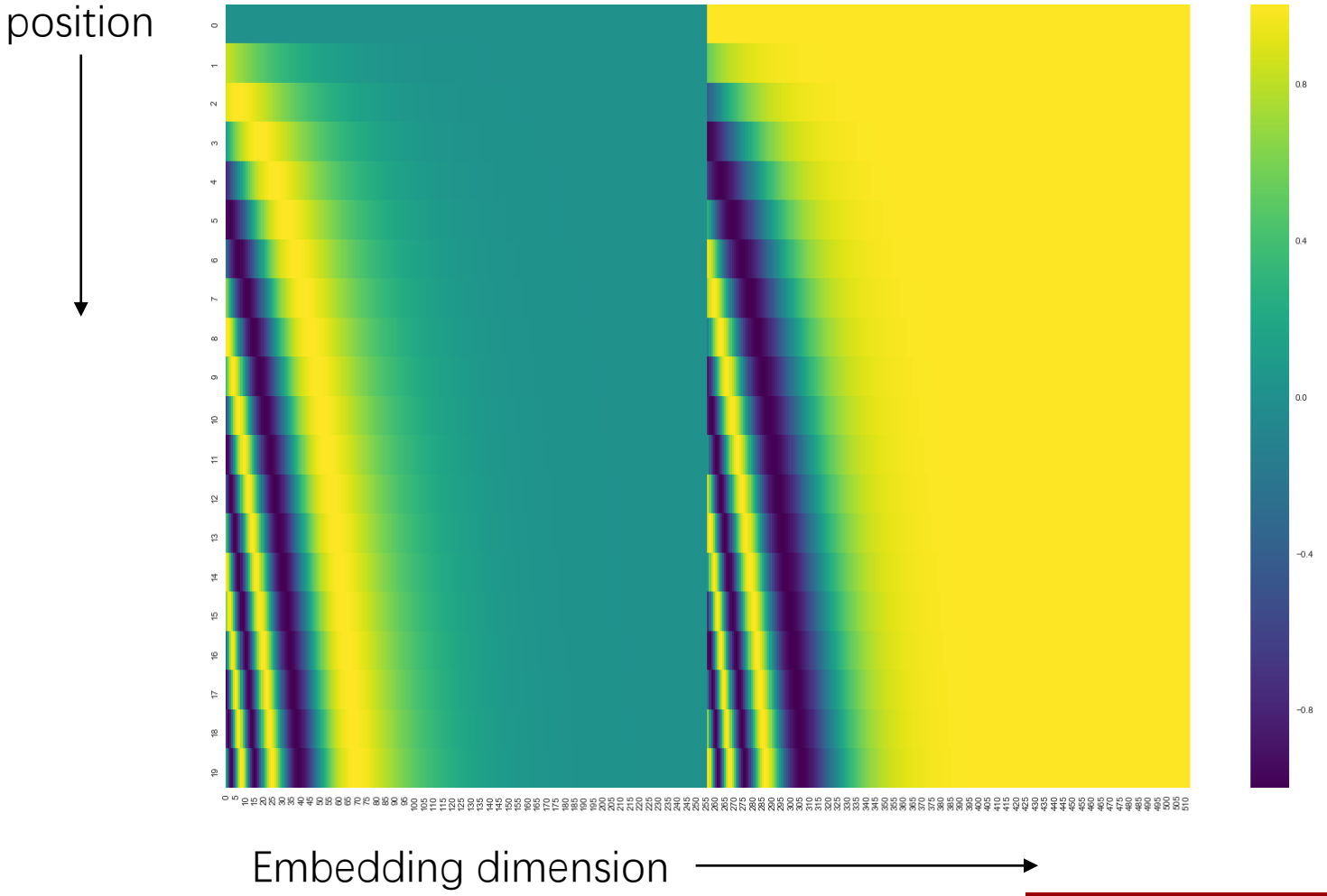
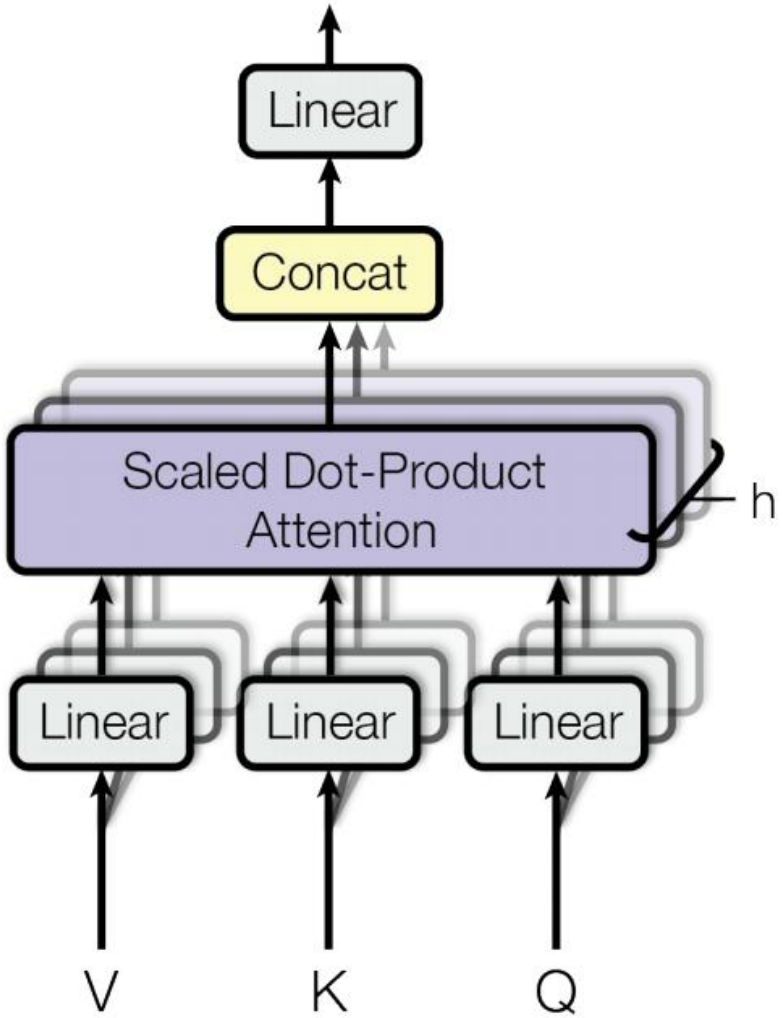


Image source

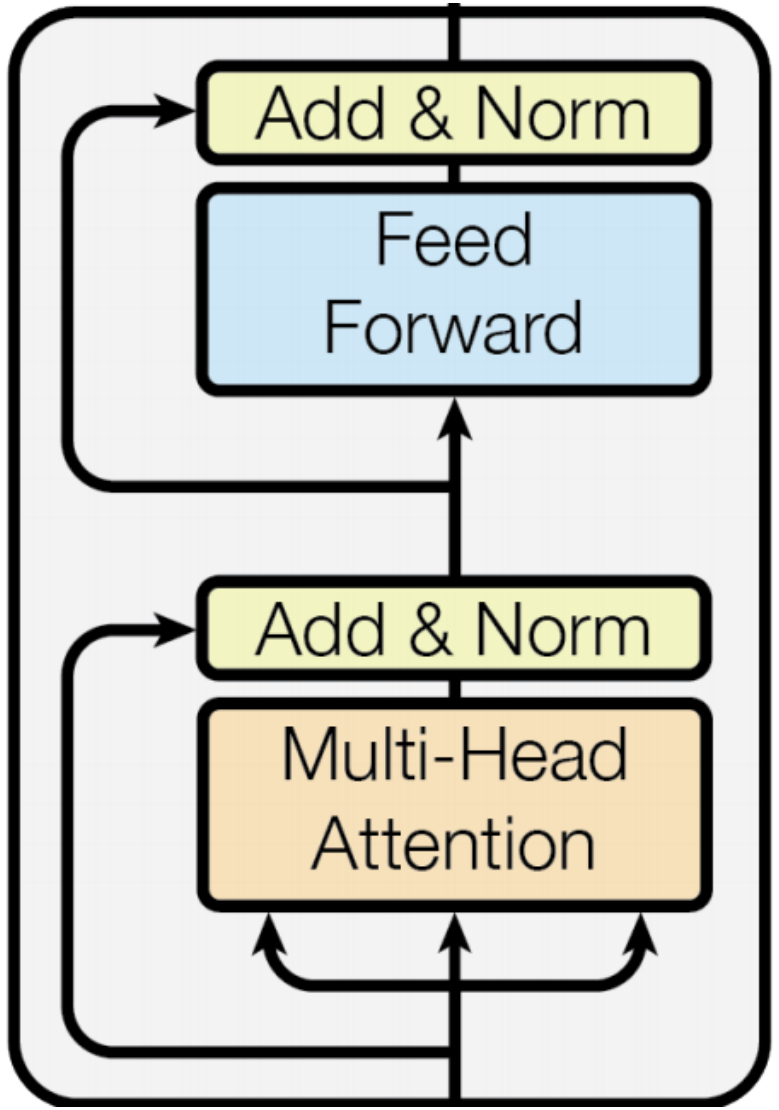
Multi-head attention

- Run h attention models in parallel on top of different linearly projected versions of Q, K, V; concatenate and linearly project the results
- Intuition: enables model to attend to different kinds of information at different positions (see visualization tool)

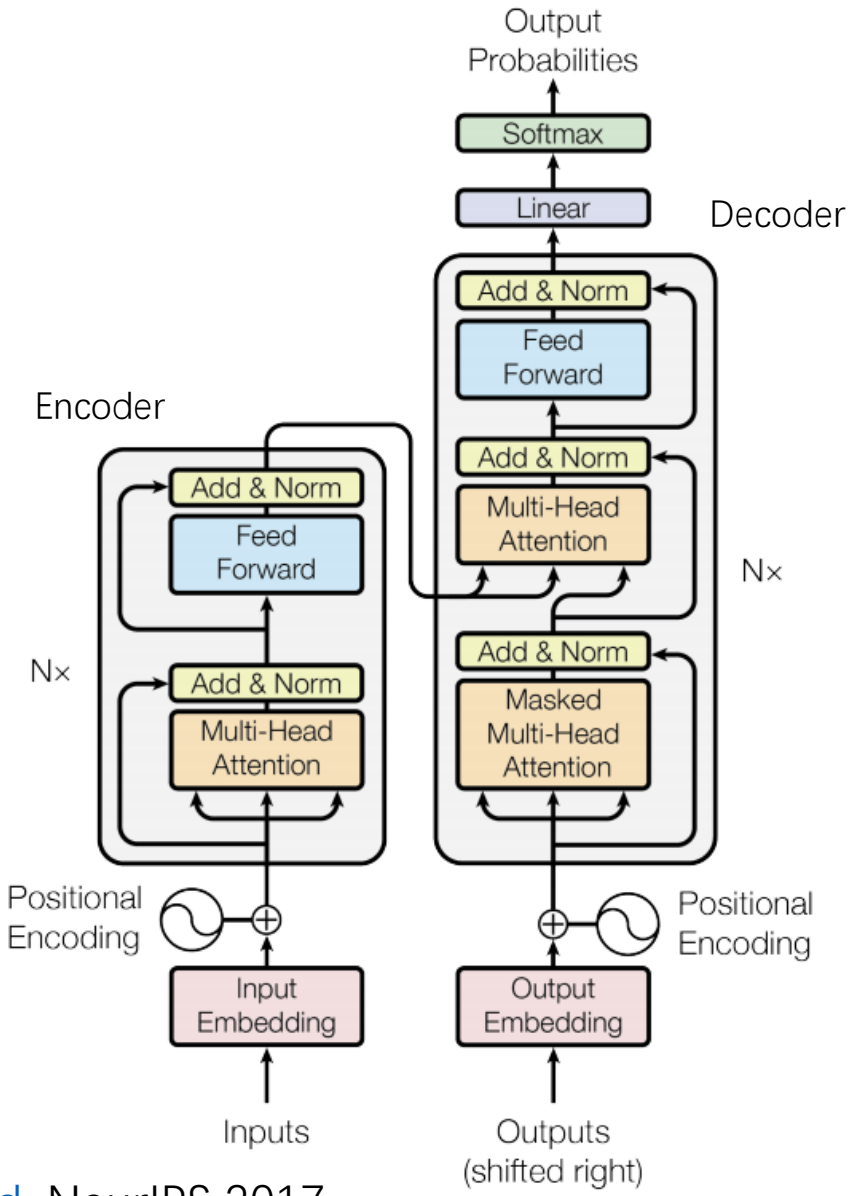


Transformer blocks

- A **Transformer** is a sequence of transformer blocks
  - Vaswani et al.: N=12 blocks, embedding dimension = 512,
    6 attention heads
  - **Add & Norm:** residual connection followed by layer normalization
  - **Feedforward:** two linear layers with ReLUs in between, applied independently to each vector
- Attention is the only interaction between inputs!

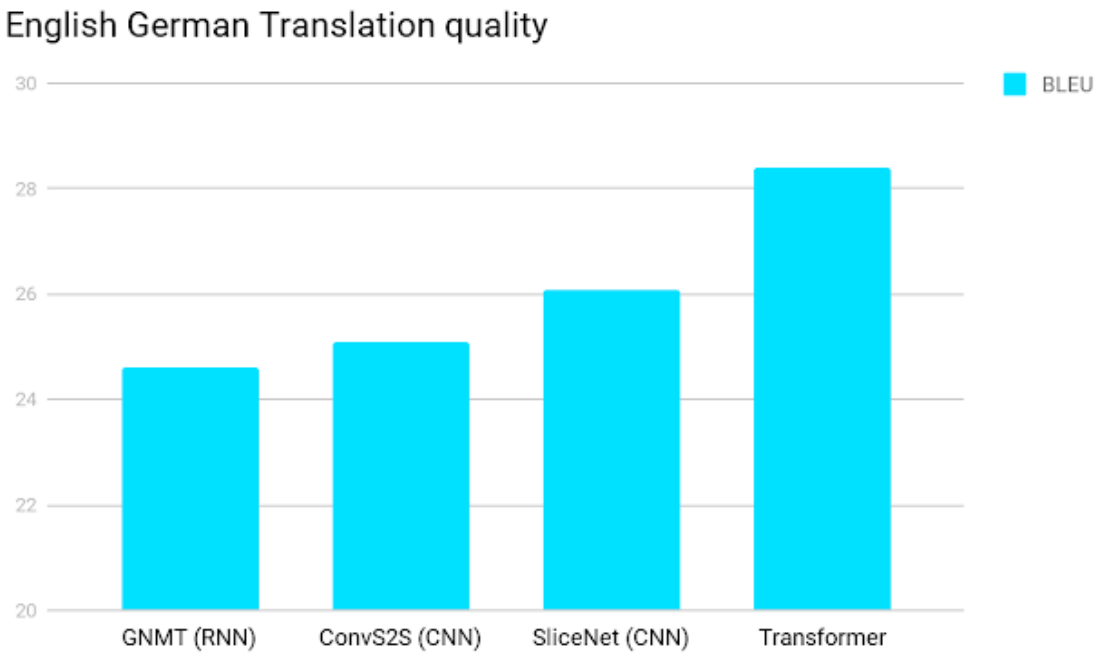


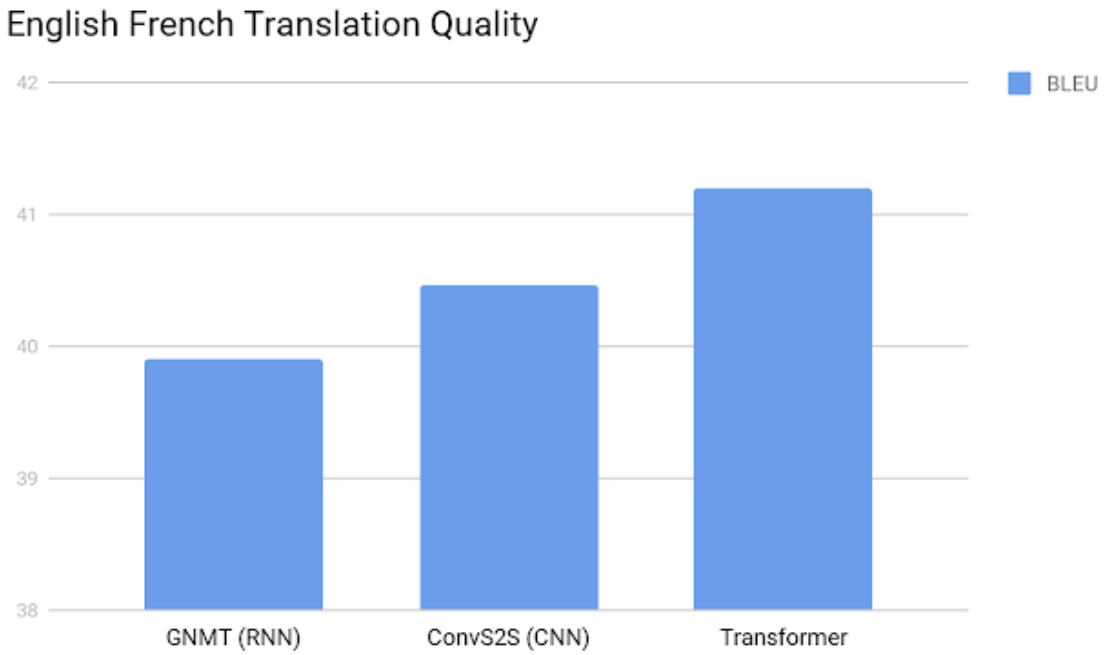
Transformer architecture: Zooming back out



A. Vaswani et al., Attention is all you need, NeurIPS 2017

Results

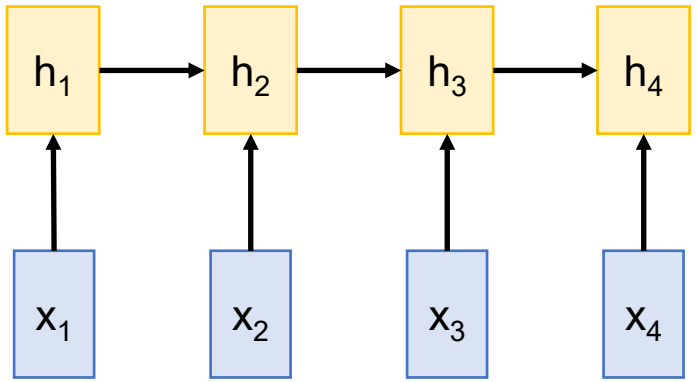




https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformers: Pros and cons

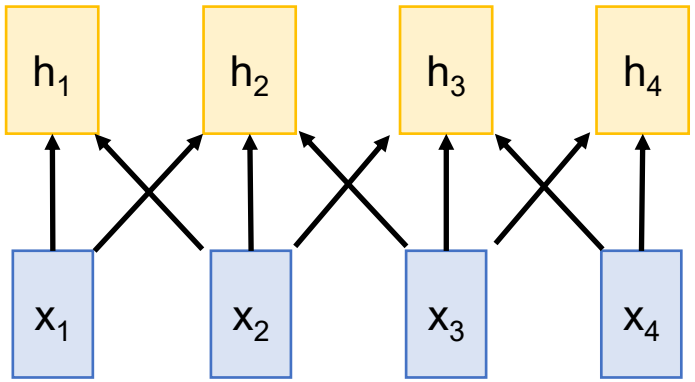
RNNs



Works on **ordered sequences**

- Pros: Not limited by fixed context size (in principle): After one RNN layer, h_T "sees" the whole sequence
- Con: Not parallelizable: need to compute hidden states sequentially
- Con: Hidden states have limited expressive capacity

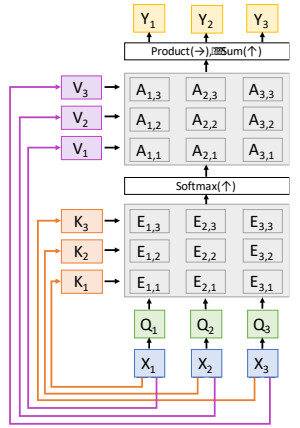
1D convolutional networks



Works on **multidimensional grids**

- Pro: Each output can be computed in parallel (at training time)
- Con: Need to stack many conv layers for outputs to "see" the whole sequence

Transformers



- Works on **sets of vectors**
- Pro: Good at long sequences: after one self-attention layer, each output "sees" all inputs!
- Pro: Each output can be computed in parallel (at training time)
- Con: Memory-intensive: cost of attention operator is *quadratic* in input size

Making transformers more efficient

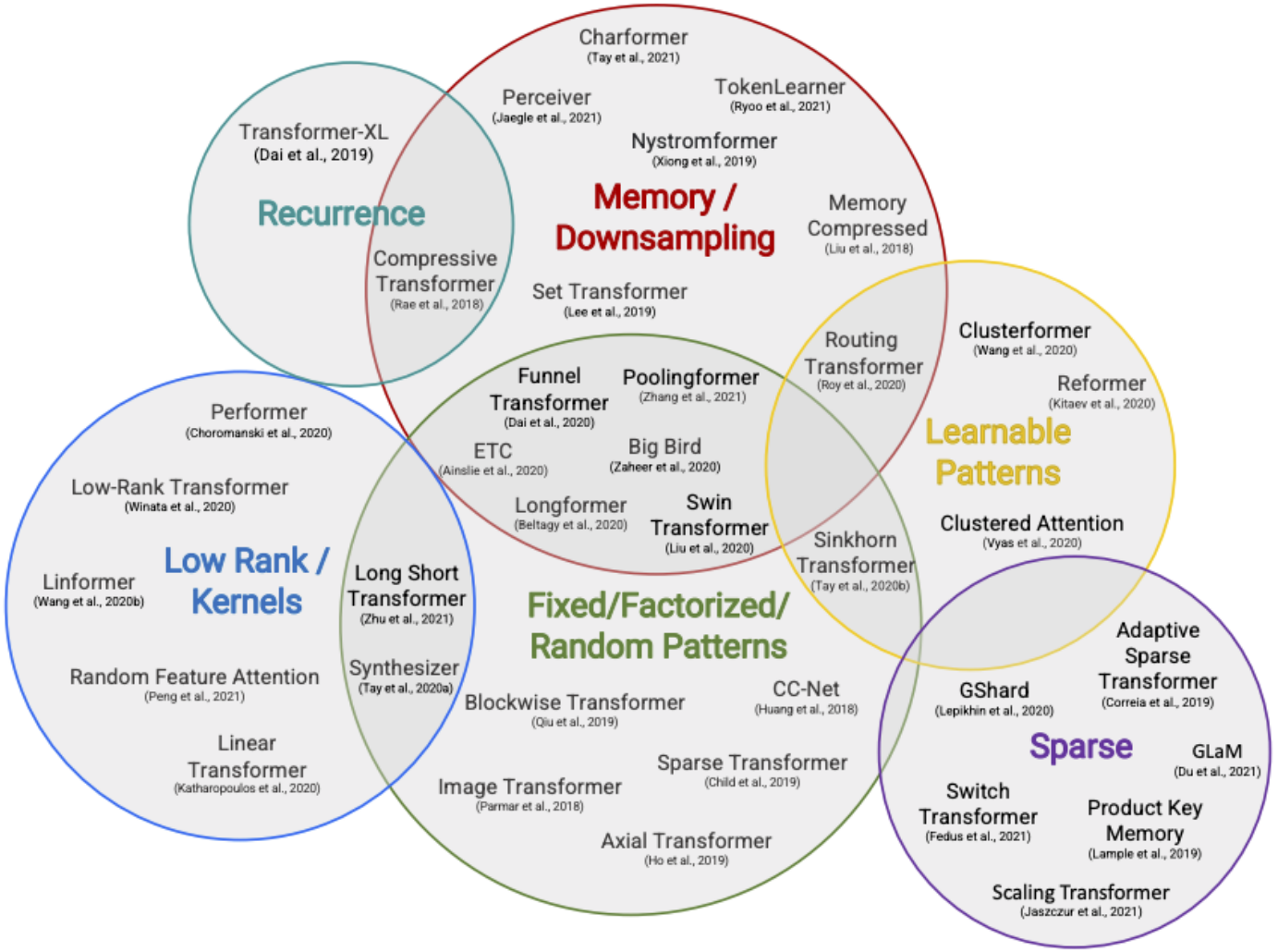


Figure 2: Taxonomy of Efficient Transformer Architectures.

Y. Tay et al. Efficient transformers: A survey. arXiv 2022

谢谢

北京大学
PEKING UNIVERSITY