# Announcement

- Homework 2
  - Available in Blackboard -> Homework
  - Due: Apr. 2, 11:59pm (one week)

# Sequence to Sequence

SLP3 Ch 10; INLP Ch 18

# Sequence to sequence

- Input a sequence, output another sequence
  - Often abbreviated to seq2seq
- Many applications
  - Machine translation (the most well-known seq2seq task)
    - Input: source language text
    - Output: target language text
  - Paraphrase
    - Input: a sentence
    - Output: a restatement of the meaning using other words
  - Style transfer
    - Input: some text
    - Output: a restatement in a specified style (e.g., formal writing, advertisement, positive sentiment, Shakespearean writing style, etc.)

# Sequence to sequence

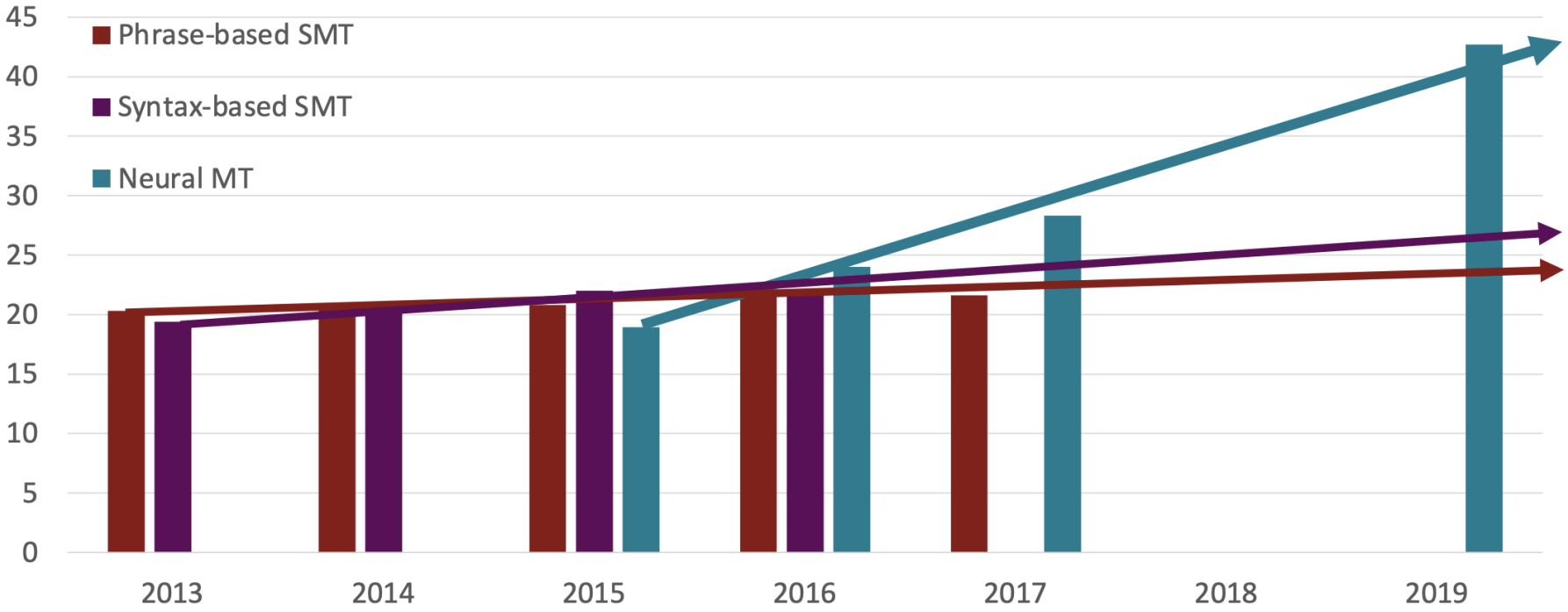
- Input a sequence, output another sequence
  - Often abbreviated to seq2seq
- Many applications
  - Summarization
    - Input: a document
    - Output: a few sentences that summarize the document
  - Question answering
    - Input: a question
    - Output: the answer
  - Dialog
    - Input: previous utterances
    - Output: next utterance

# Sequence to sequence

- More applications
  - Input: Tolkien's epic novel The Lord of the Rings was published in 1954-1955, years after the book was completed.
  - Joint entity and relation extraction
    - [ Tolkien | *person* ]'s epic novel [ The Lord of the Rings | *book* | *author* = Tolkien ] was published in 1954-1955, years after the book was completed.
  - Semantic role labeling
    - Tolkien's epic novel [ The Lord of the Rings | *subject* ] [ was published | *predicate* ] [ in 1954-1955 | *temporal* ], years after the book was completed.
  - Coreference resolution
    - [ Tolkien ]'s epic novel [ The Lord of the Rings ] was published in 1954-1955, years after the [ book | The Lord of the Rings ] was completed.
  - More on these tasks later…

Paolini, et al. "Structured prediction as translation between augmented natural languages." ICLR 2021

# Comparing Methods of Machine Translation




**Sources:** http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf & http://matrix.statmt.org/

SMT: statistical machine translation
NMT: neural machine translation ⇐ neural seq2seq

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal; NMT 2019 FAIR on newstest2019]

# NMT: a big success story of NLP + deep learning

- Neural Machine Translation went from a fringe research attempt in 2014 to the leading standard method in 2016
  - 2014: First seq2seq paper published
  - 2016: Google Translate switches from SMT to NMT - and by 2018 everyone has

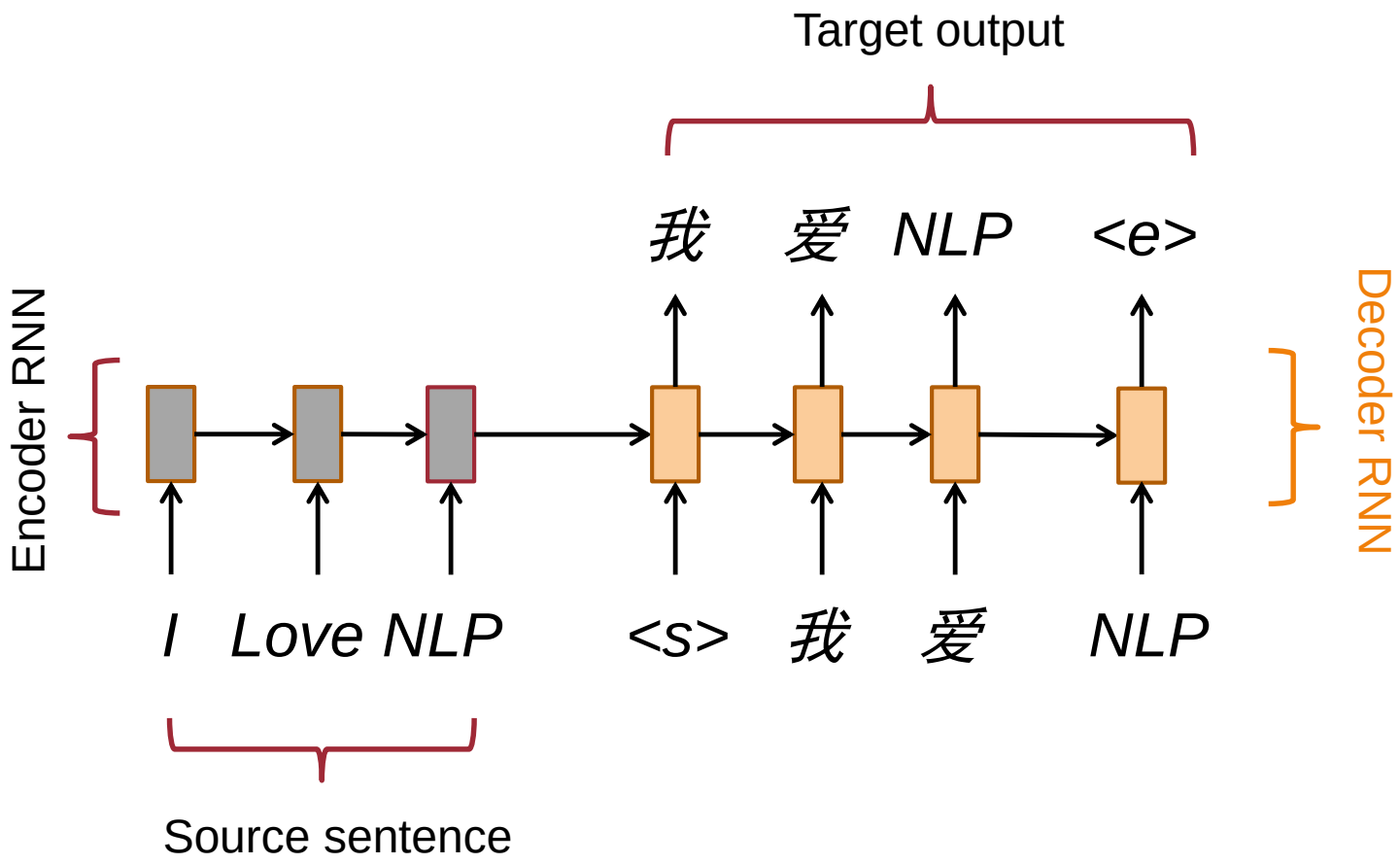

  - SMT systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a small group of engineers in a few months

# Neural Seq2Seq Methods

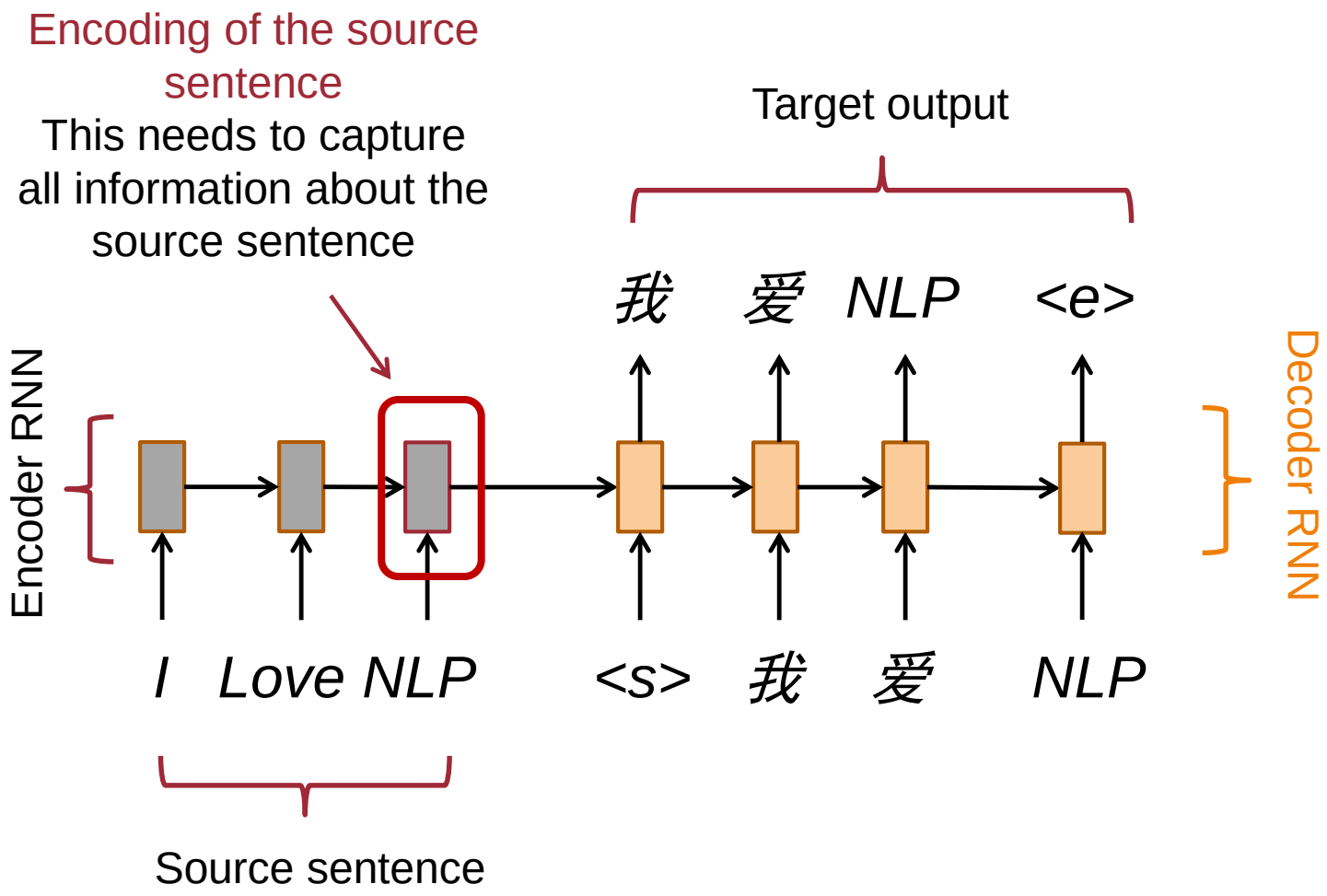
# Neural Seq2seq Methods

- Similar to neural language modeling methods
  - Recurrent neural network
  - Attention
  - Transformer
- Difference: now we have an encoder to process the input sequence and a decoder to produce the output sequence

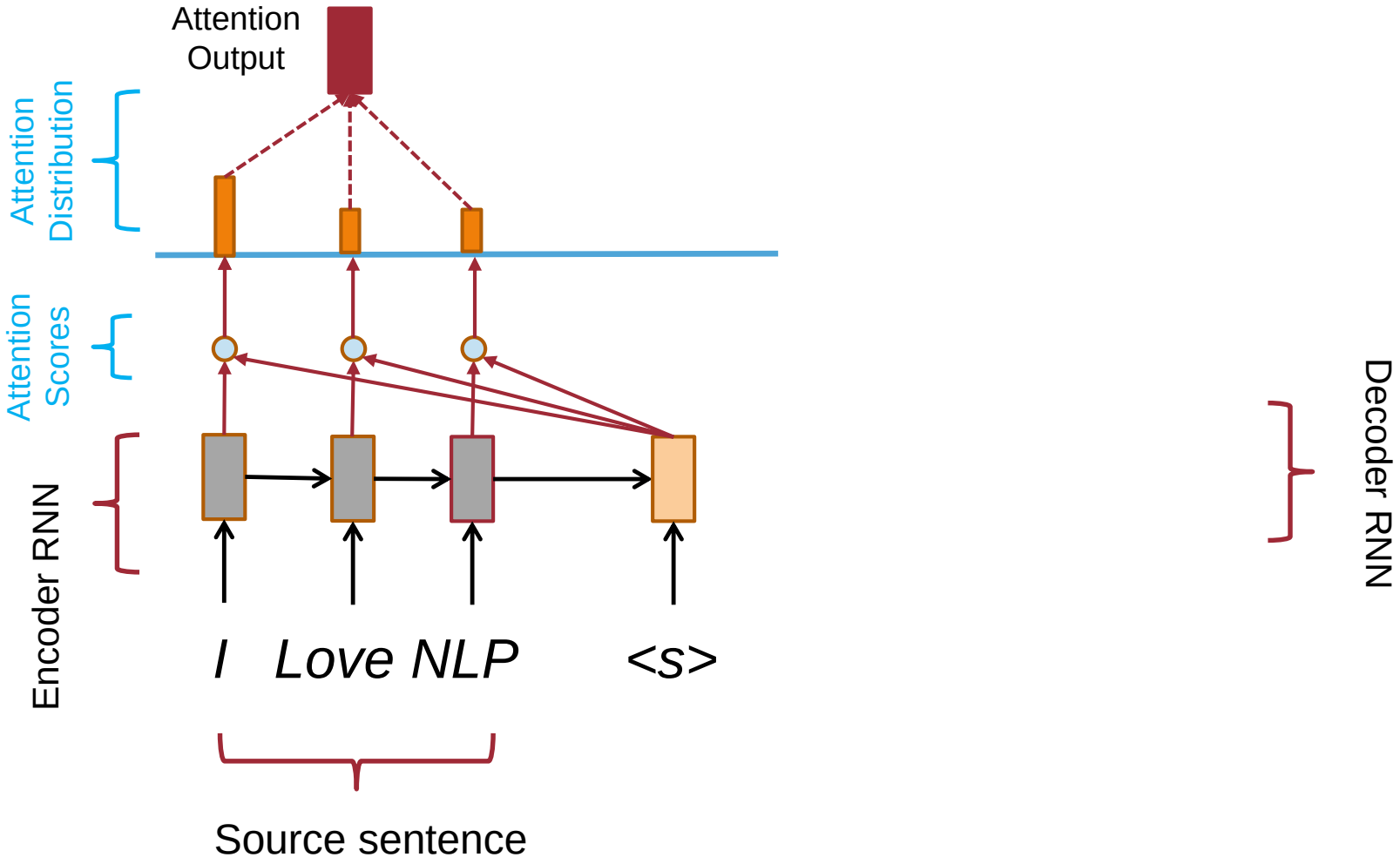
# RNN


Target output
我 爱 NLP <e>
Encoder RNN
Decoder RNN
I Love NLP
<s> 我 爱 NLP
Source sentence

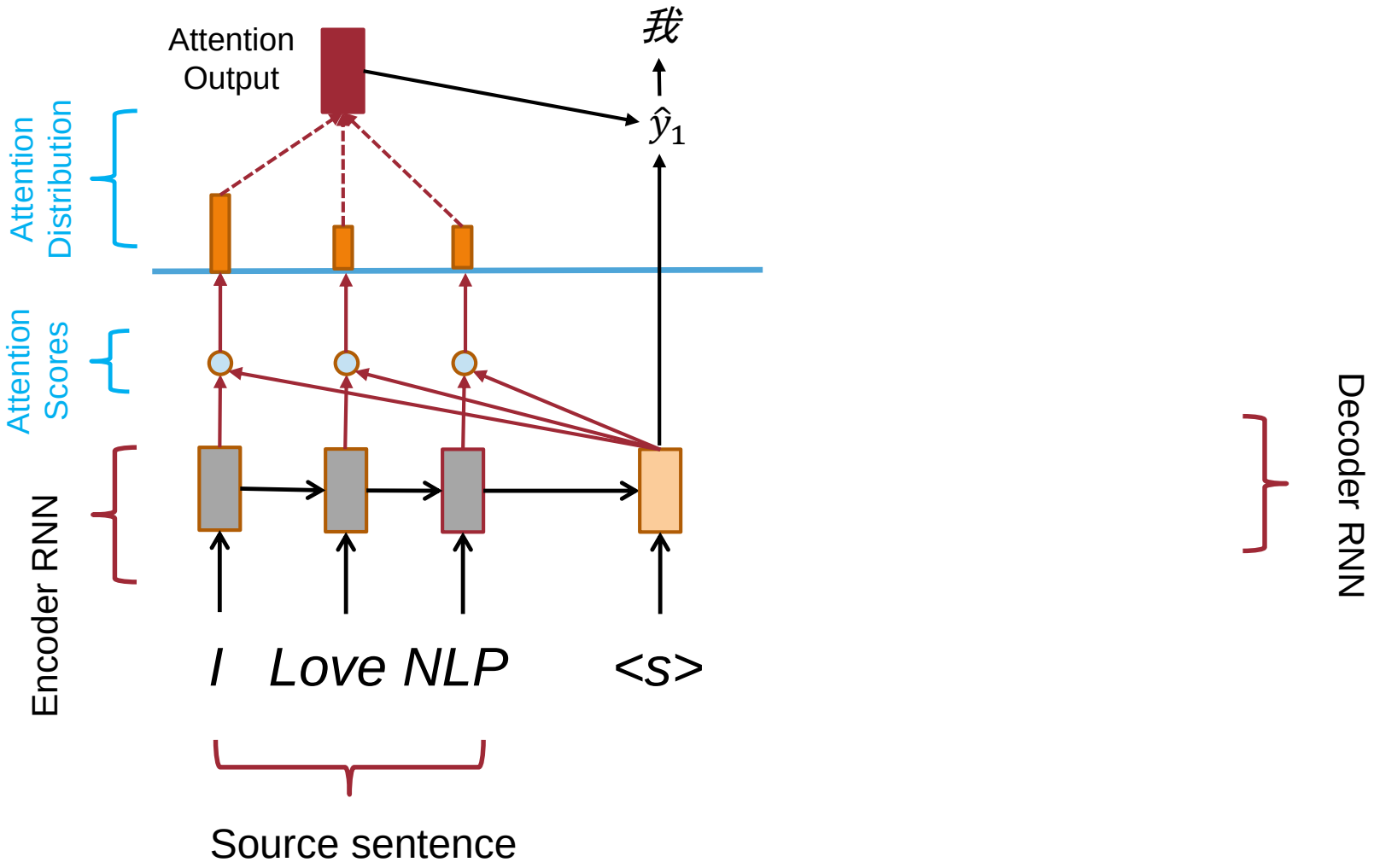
# RNN: the bottleneck problem



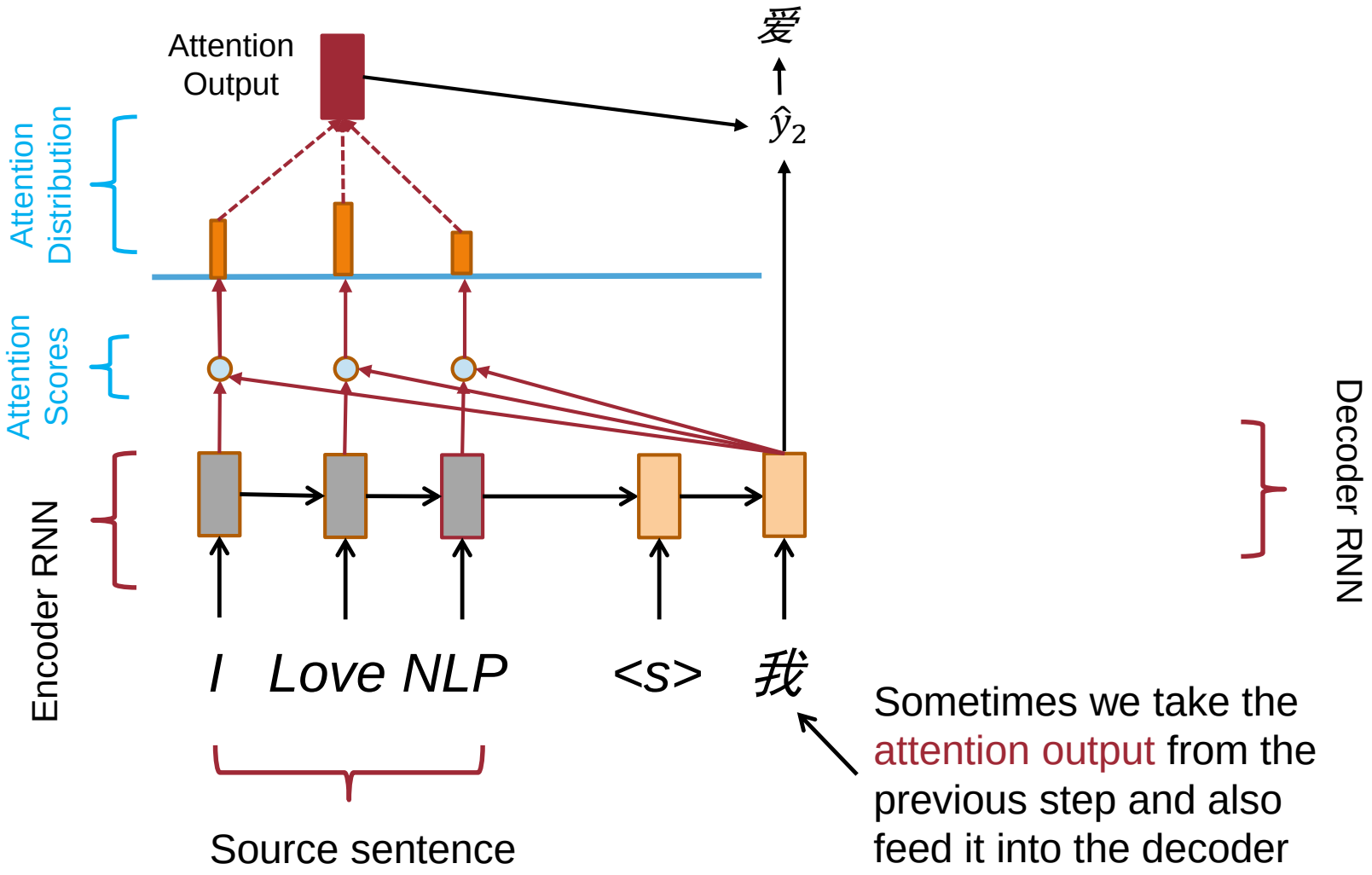
# Seq2seq with attention



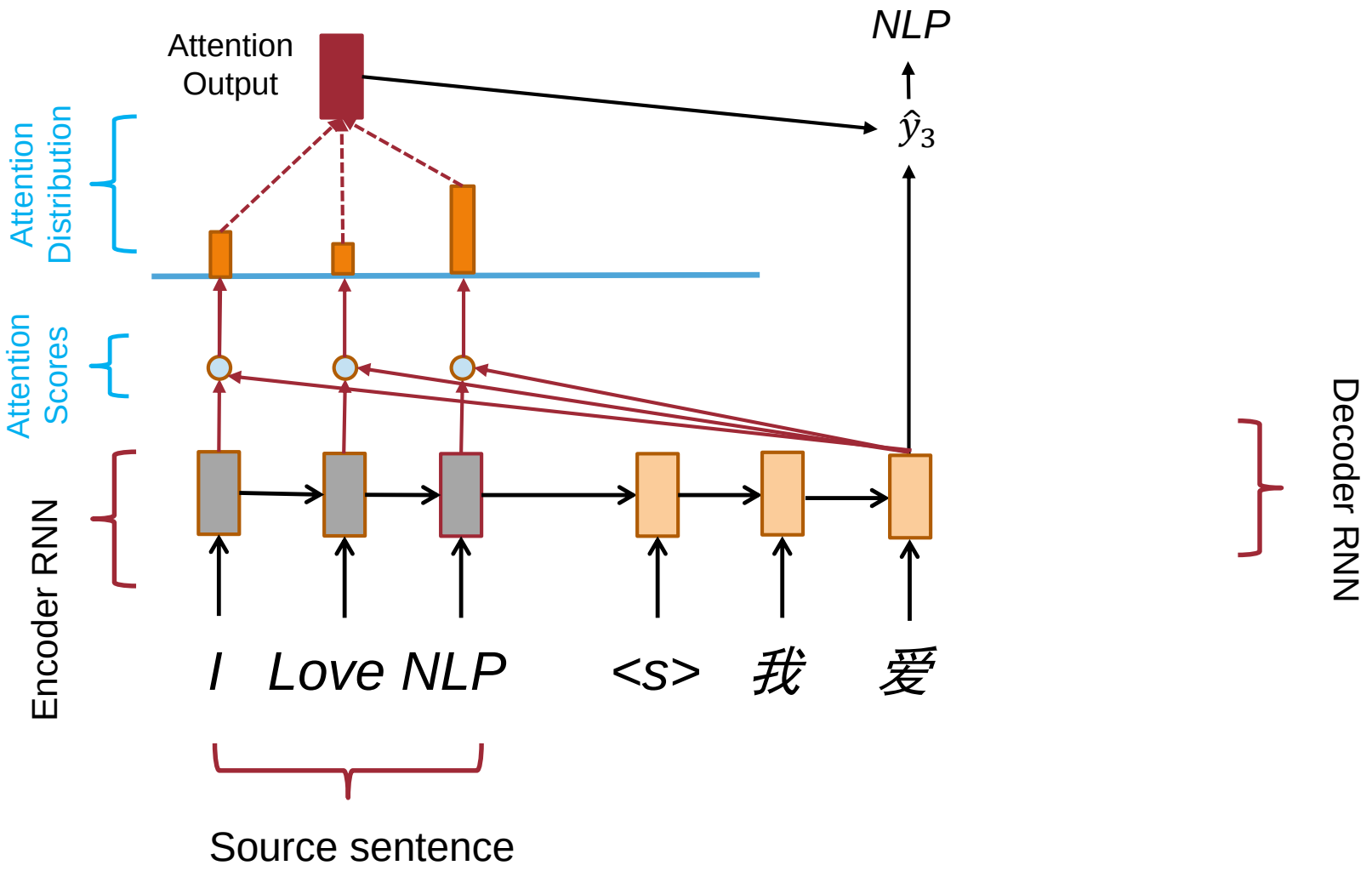
# Seq2seq with attention



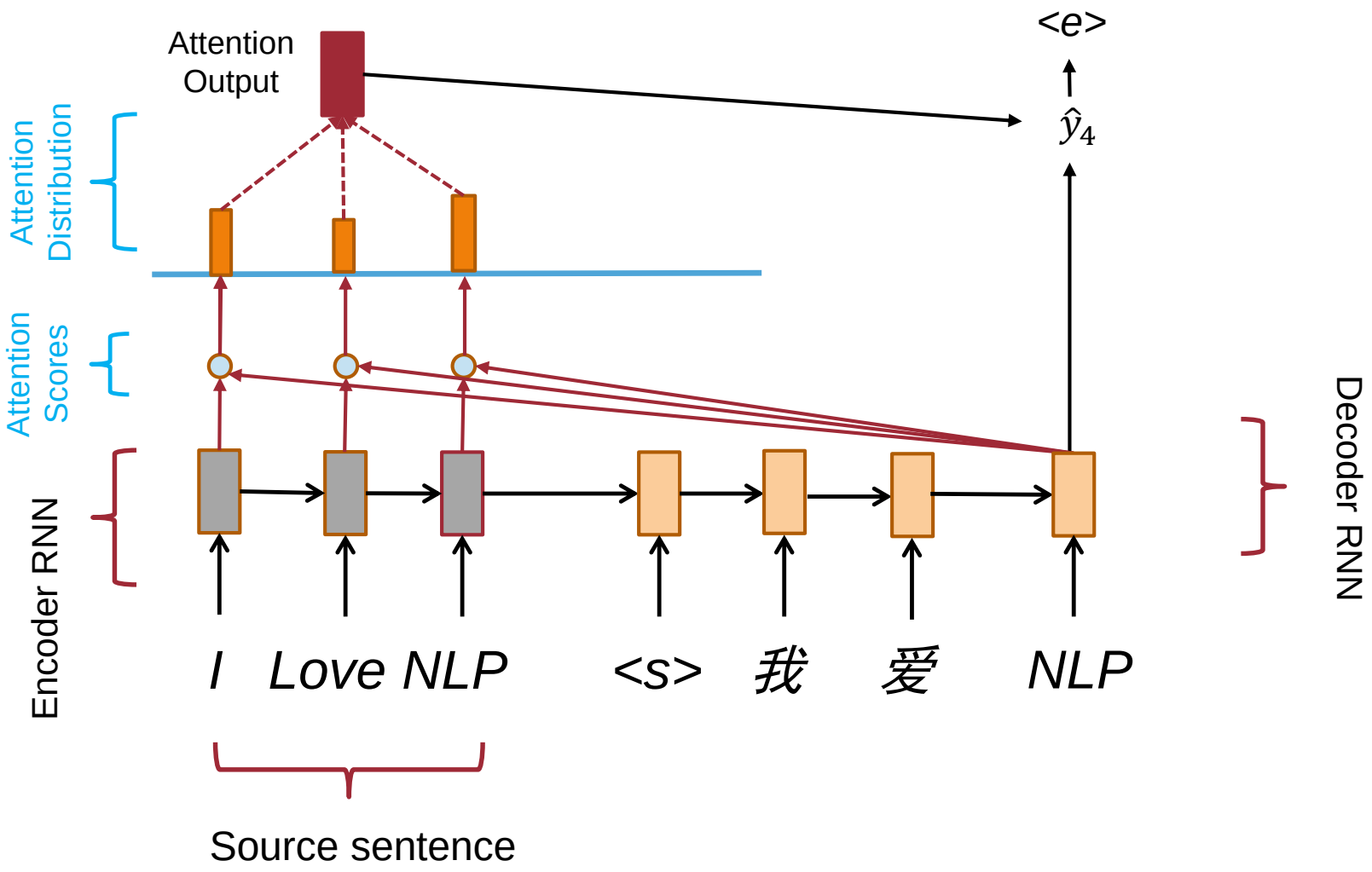
# Seq2seq with attention



# Seq2seq with attention

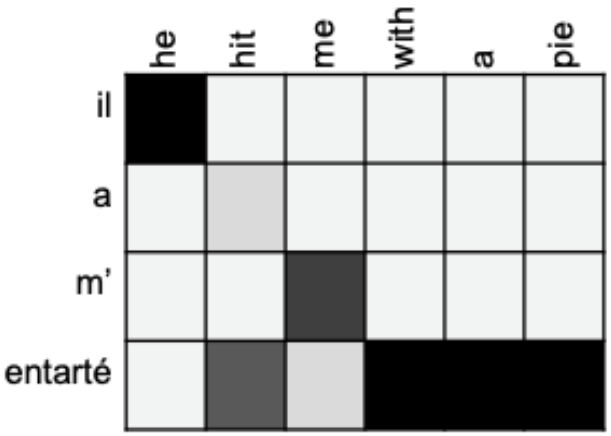


# Seq2seq with attention



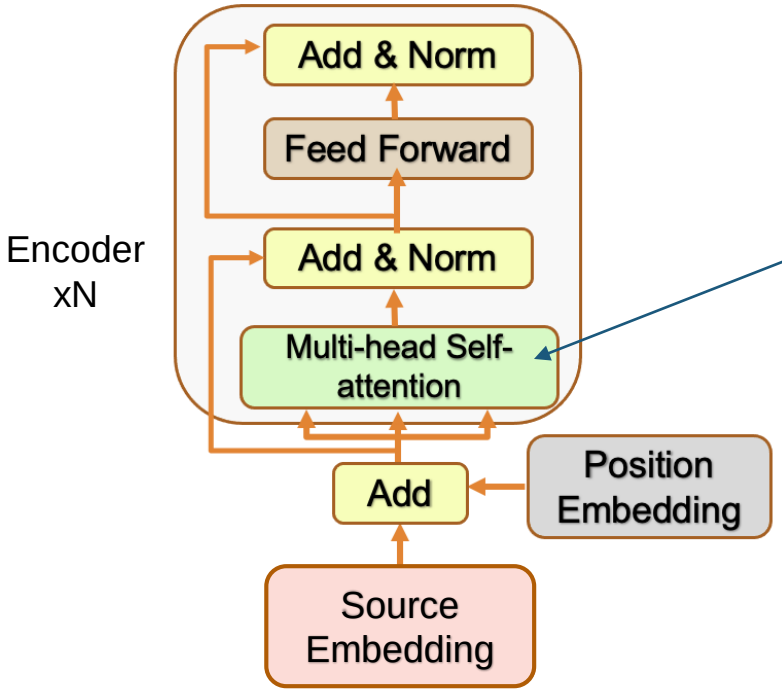
# Attention is great

- Attention solves the bottleneck problem
  - It allows decoder to look directly at source, bypassing bottleneck
- Attention helps with the vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides more "human-like" model of seq2seq
  - You can look back at the source sentence while translating, rather than needing to remember it all
- Attention provides some interpretability
  - Attention distribution reveals what the decoder was focusing on
  - = (soft) source-target sentence alignment.




# Replace RNN with Transformer

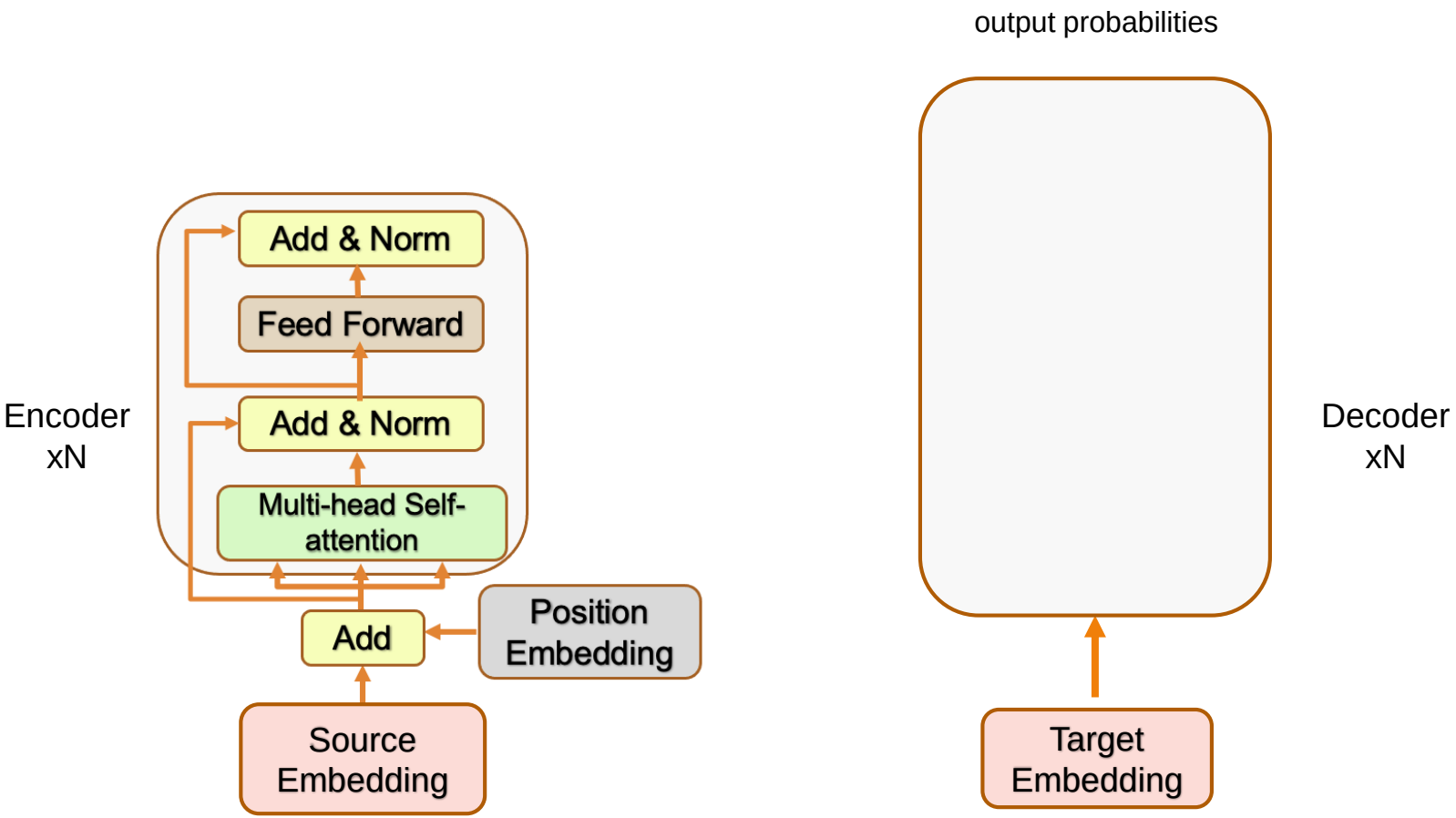
- We have introduced transformer encoder in LM.




No mask, because we are not doing LM and are allow to see the complete sentence.

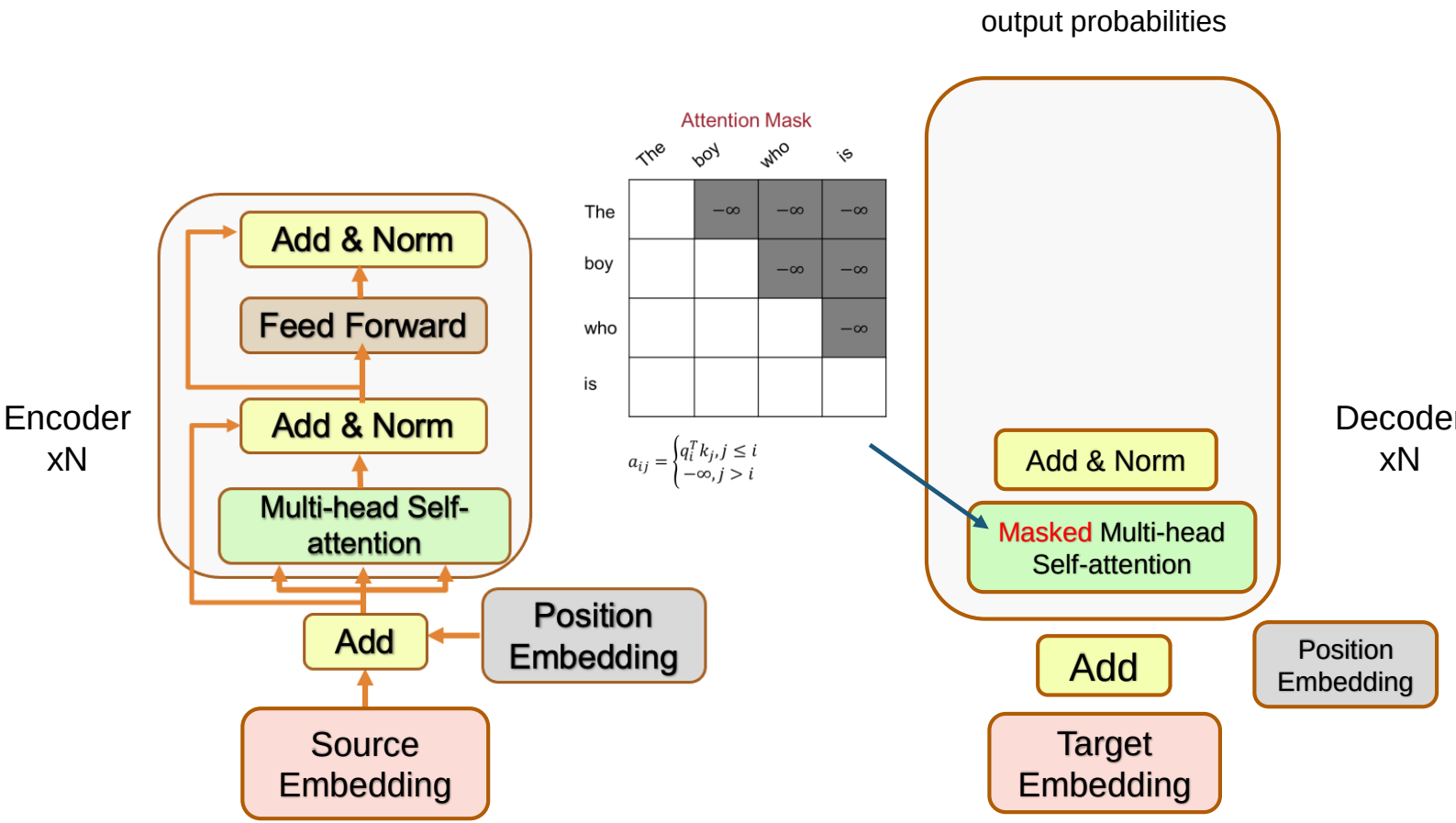
# Replace RNN with Transformer

- What about the decoder?



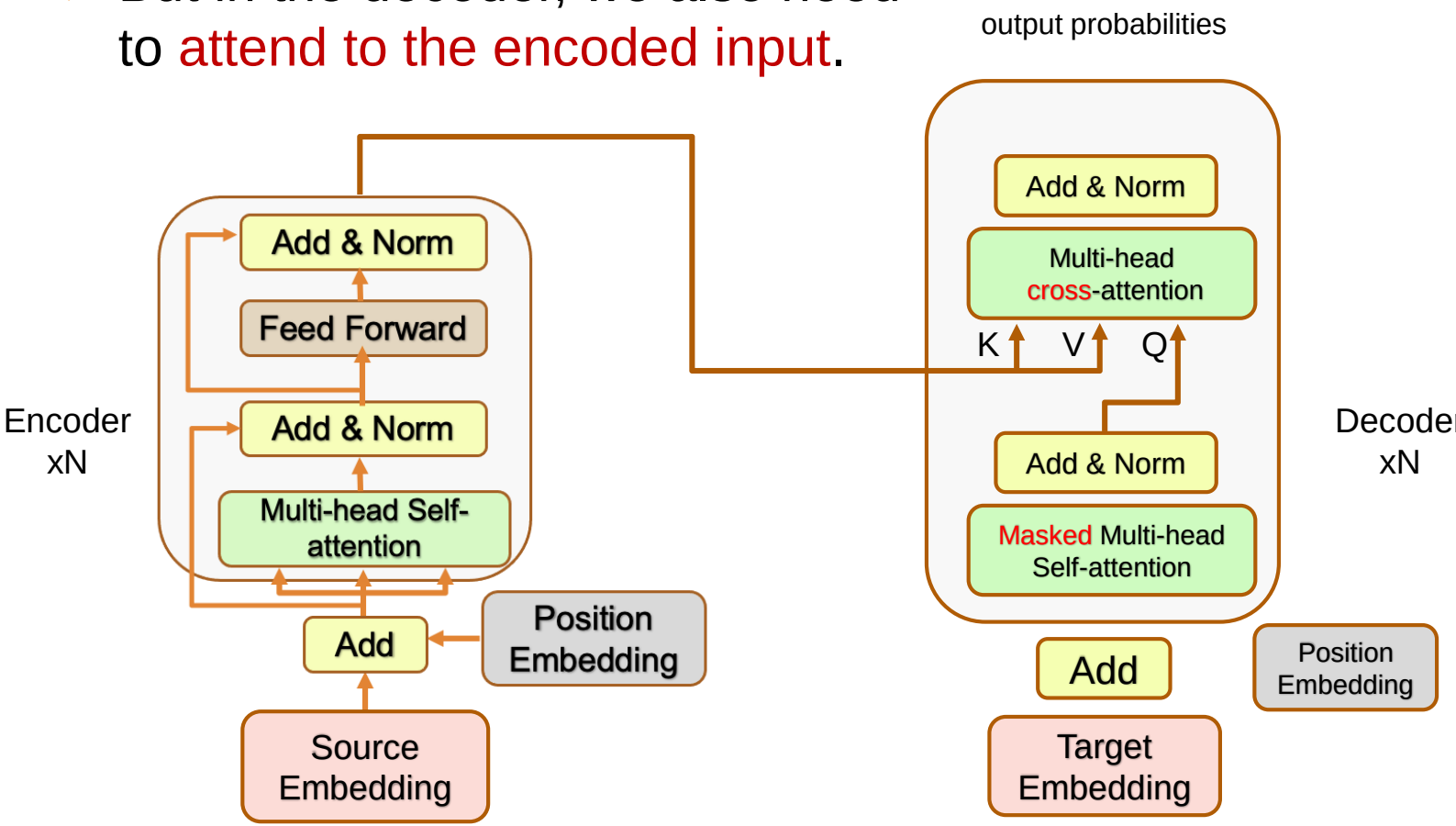
# Seq2seq with Transformer

- We ignore the arrows in the decoder for simplicity.



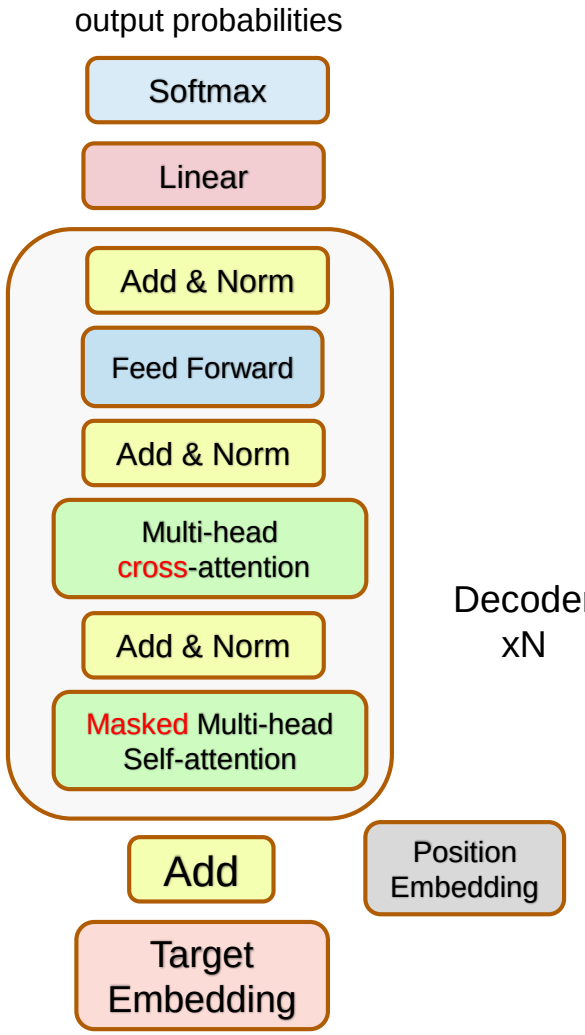
# Seq2seq with Transformer

- In self-attention, K/Q/V come from the same sequence.
- But in the decoder, we also need to attend to the encoded input.

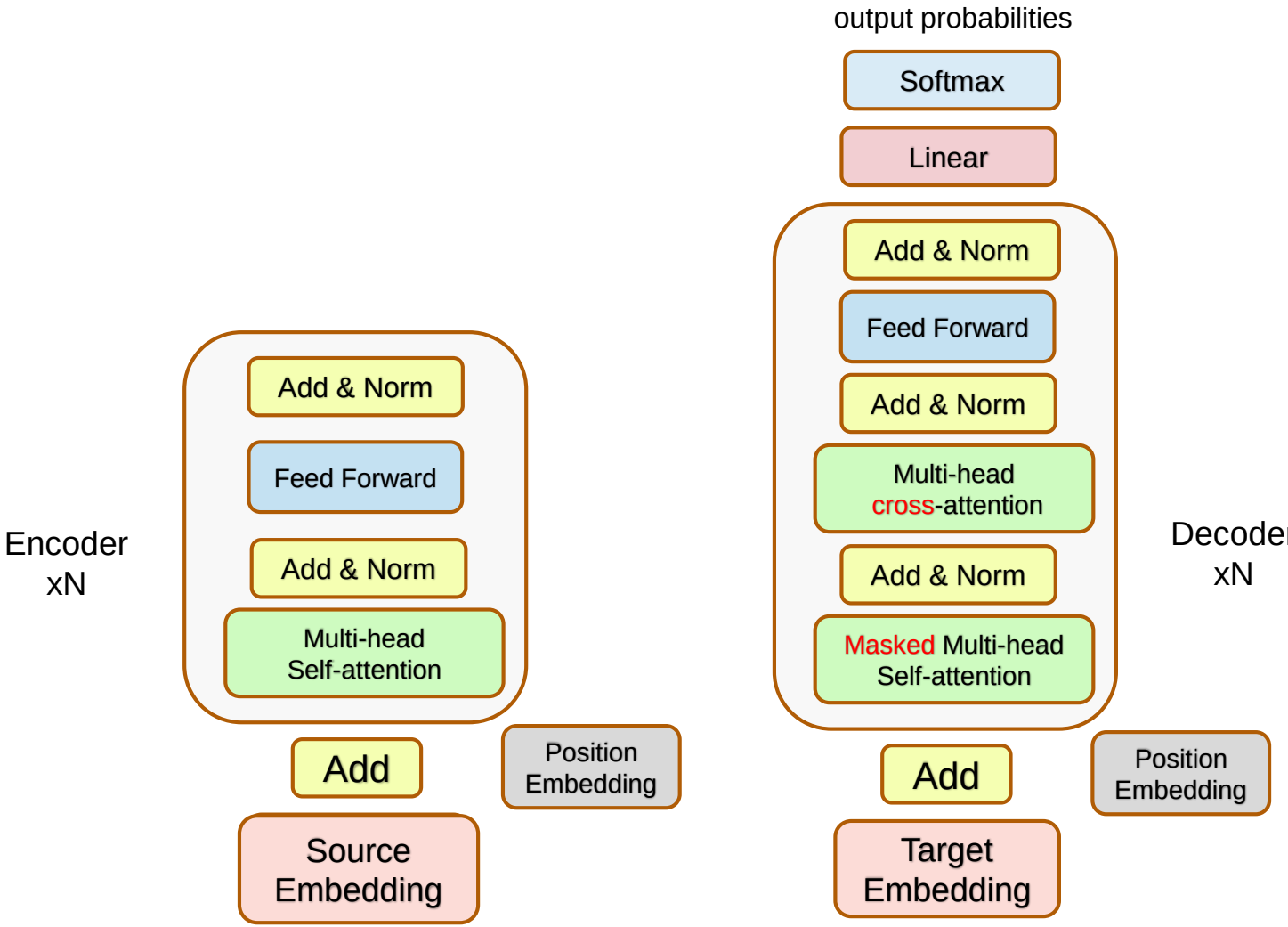


# Transformer decoder: finishing touches

- Add a feed forward layer (with residual connections and layer norm)
- Add a final linear layer to project the embeddings into a much longer vector of length vocab size (logits)
- Add a final softmax to generate a probability distribution of next word



# Recap of the Transformer Architecture



# Learning and Decoding

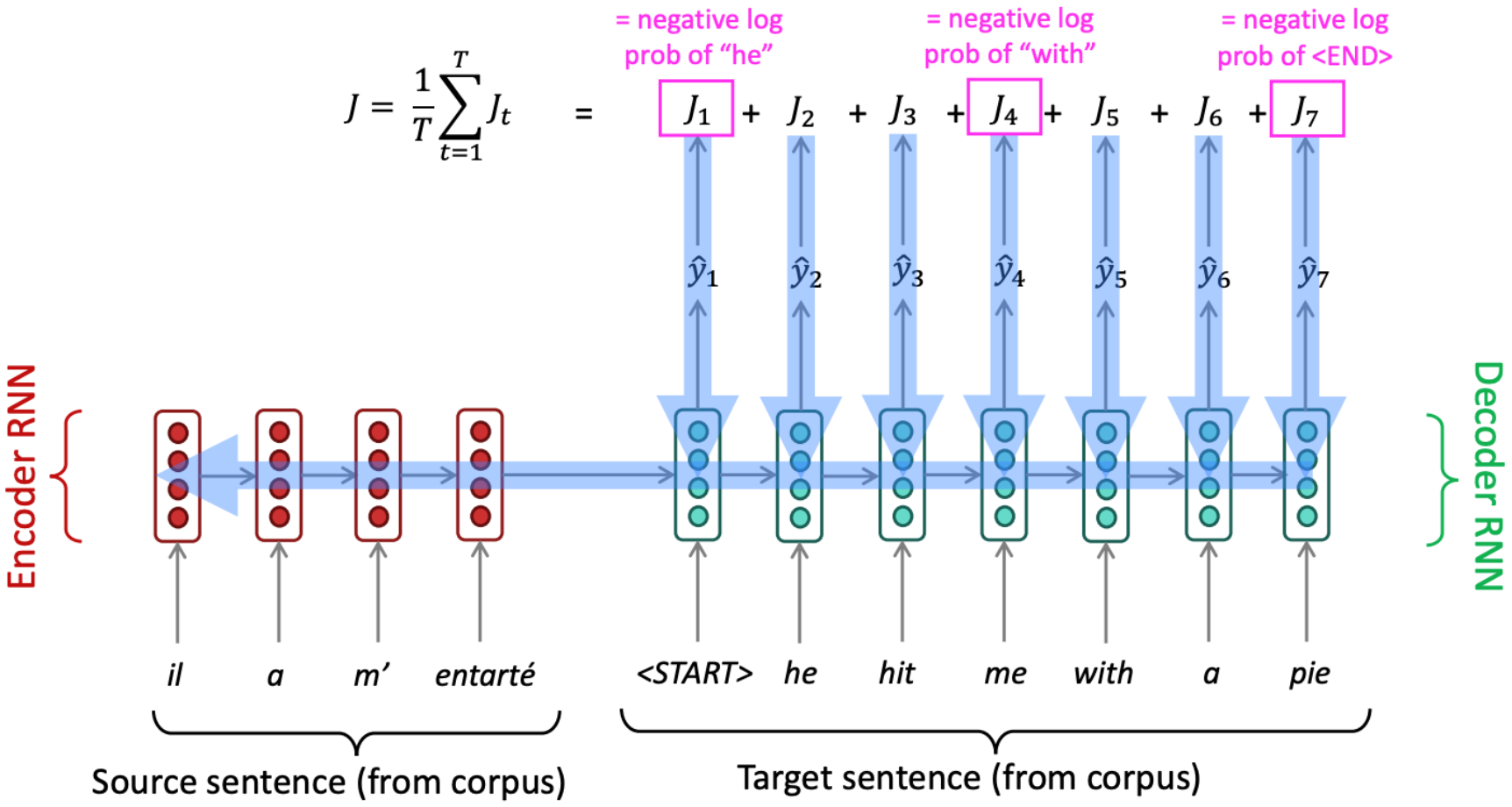
# Learning of seq2seq

- The sequence-to-sequence model is an example of a conditional language model
  - "Language model" because the decoder is predicting each word of the target sentence $y$ from left to right
  - "Conditional" because its predictions are conditioned on the source sentence $x$
- Seq2seq directly calculates $P(y|x)$:

$$P(y_{1:T}|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_1, y_2, x) \dots P(y_T|y_1, \dots y_{T-1}, x)$$

- Learning
  - Training data: a parallel corpus (paired input-output sequences)
  - Objective: maximizing the conditional likelihood

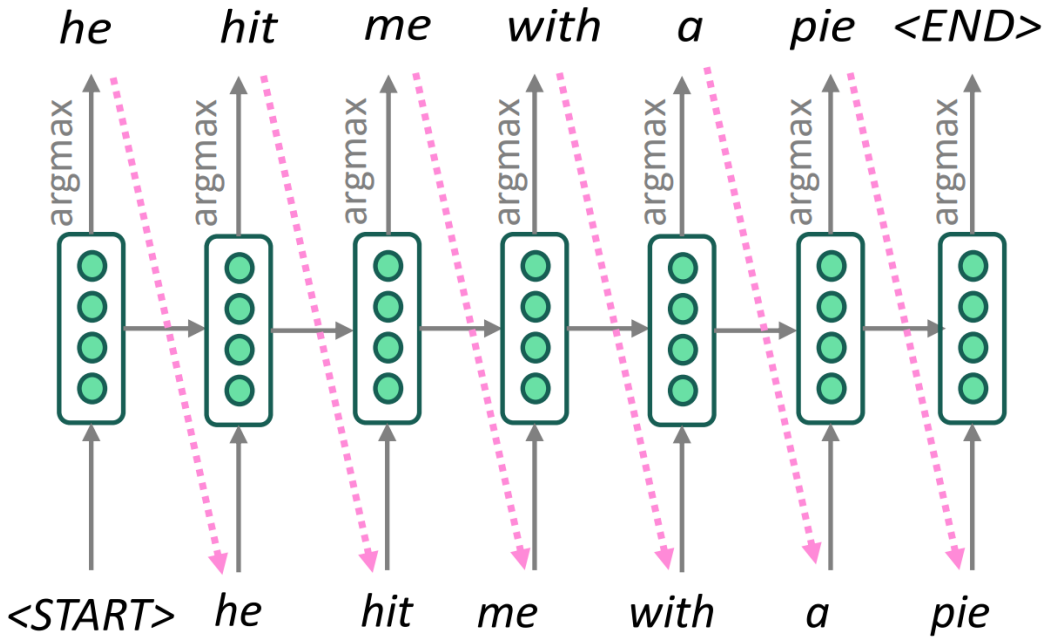
# Learning of seq2seq




Seq2seq is optimized as a **single system.** Backpropagation operates "*end-to-end*".

# Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder
  - Predict the most probable word at each step




# Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
  - Input: *il a m'entarté* *(he hit me with a pie)*
  - → *he* ____
  - → *he hit* ____
  - → *he hit a* ____ *(whoops! no going back now…)*
- How to fix this?

# Beam search decoding

- At each step, keep track of the $k$ best partial translations (i.e., hypotheses)
  - $k$ is the beam size (e.g., 5 to 10 in NMT)
- A hypothesis $y_1, \dots, y_t$ has a score which is its log probability:

$$\text{score}(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t | x) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$

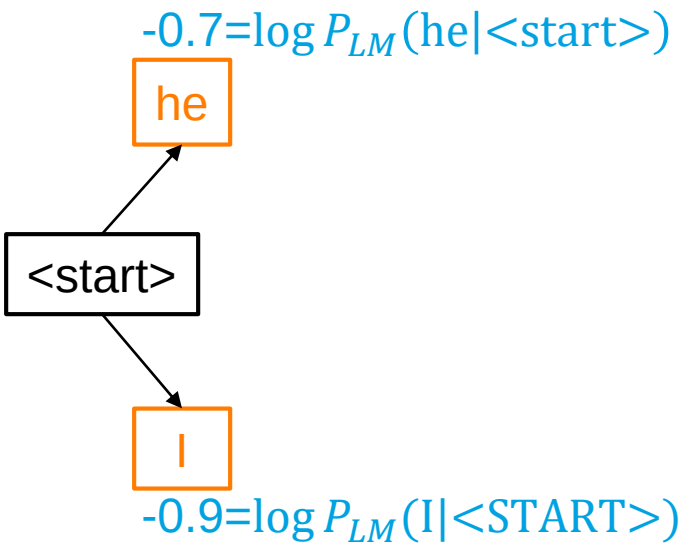
<start>

Calculate prob dist of next word

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$

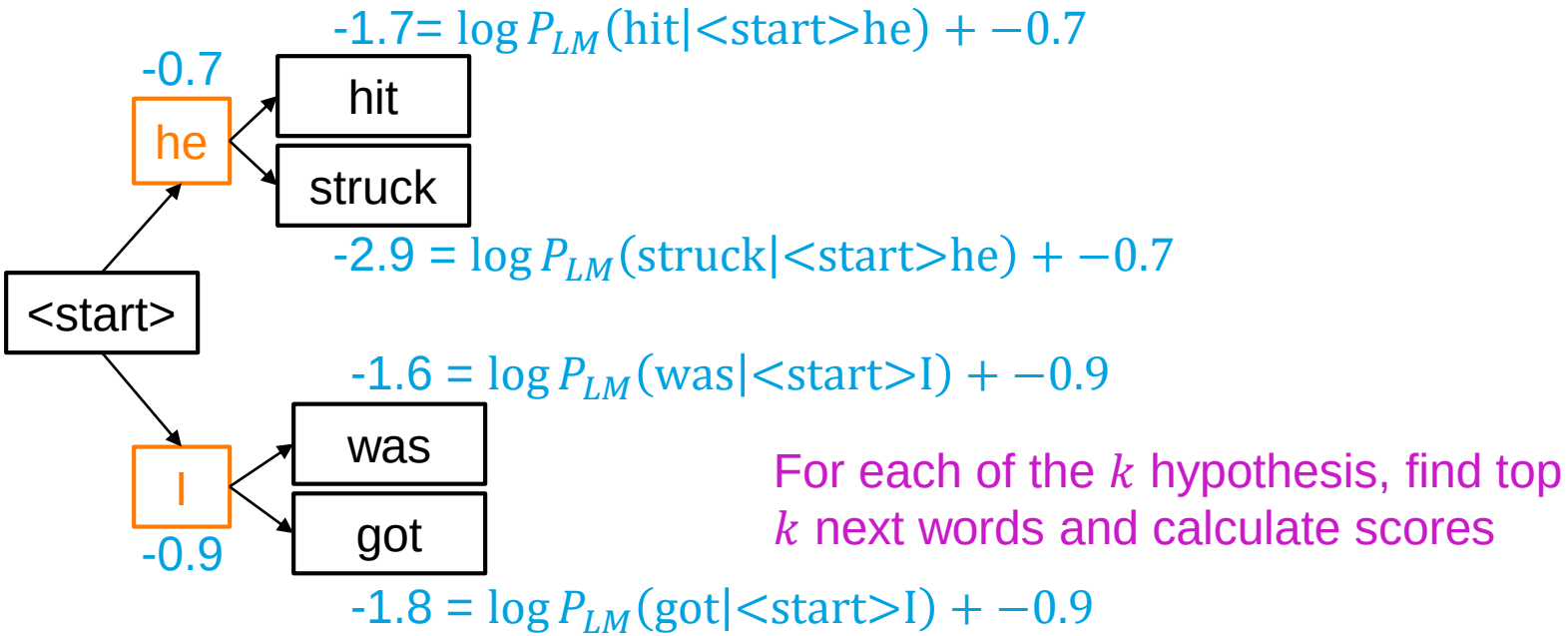



Take top $k$ words

# Beam search decoding: example

Beam size = $k$ = 2.

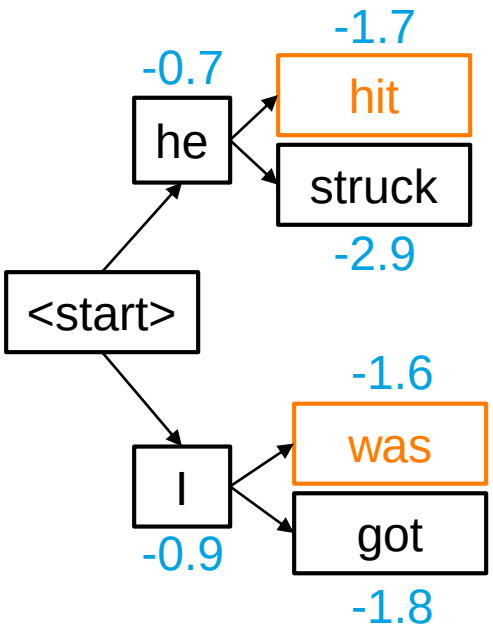
Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$



# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$

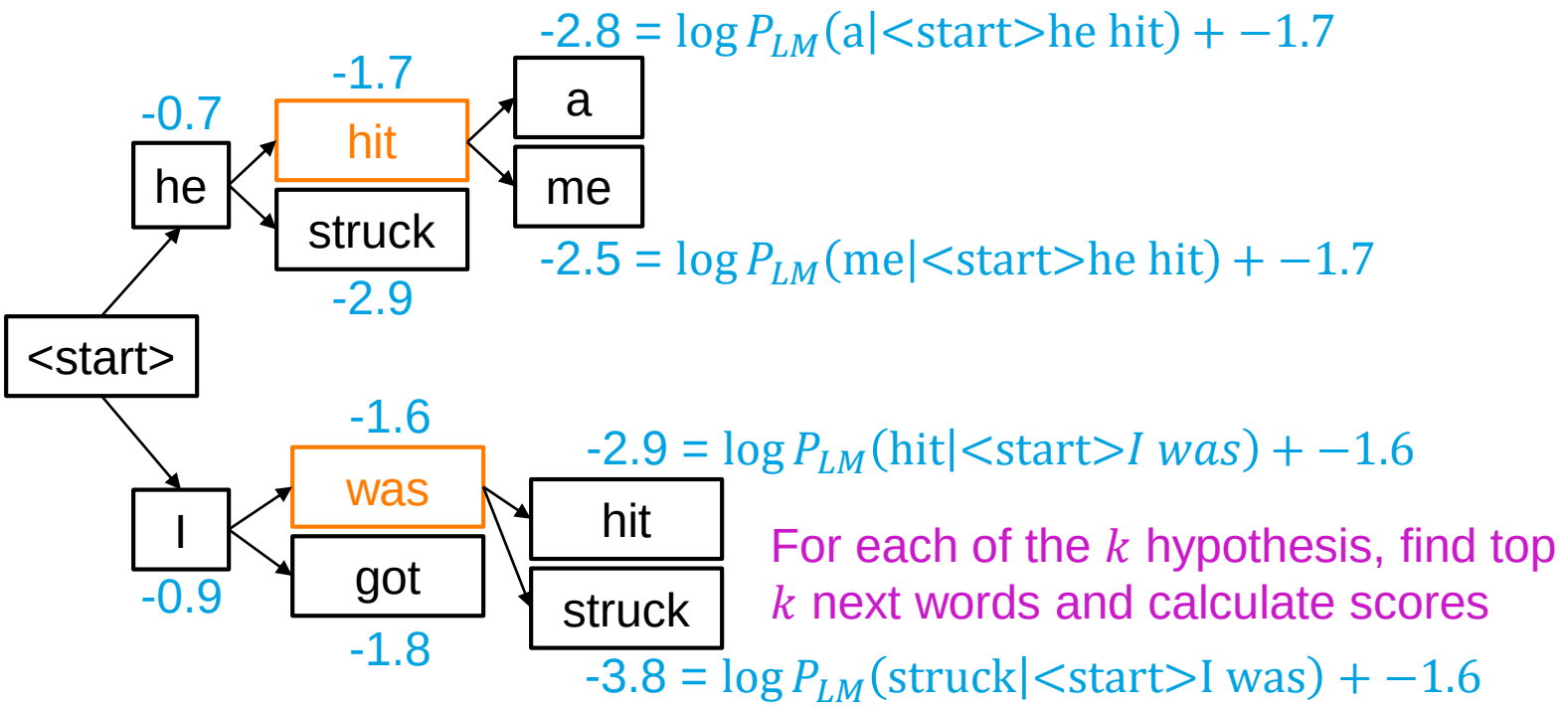



Of these $k^2$ hypotheses,
just keep $k$ with highest scores

# Beam search decoding: example

Beam size = $k$ = 2.

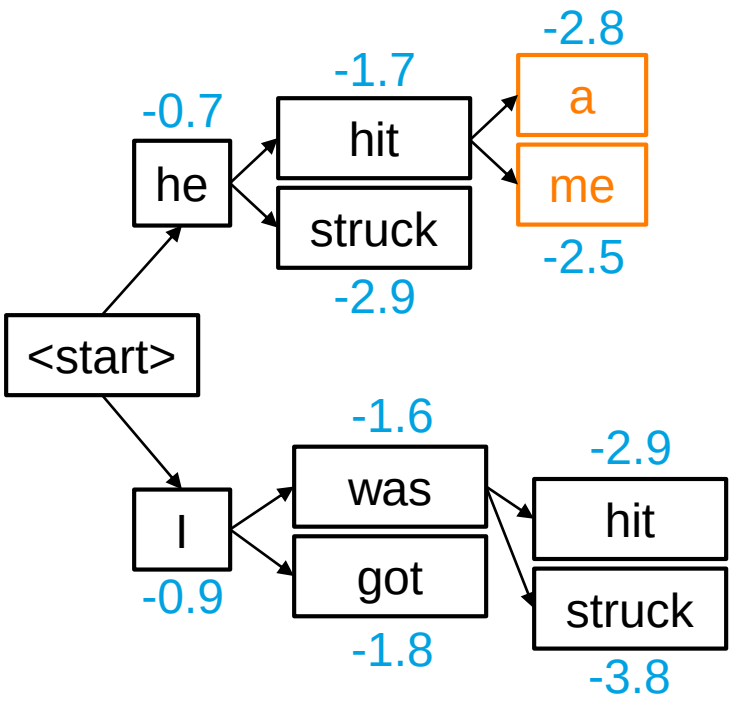
Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$



# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$

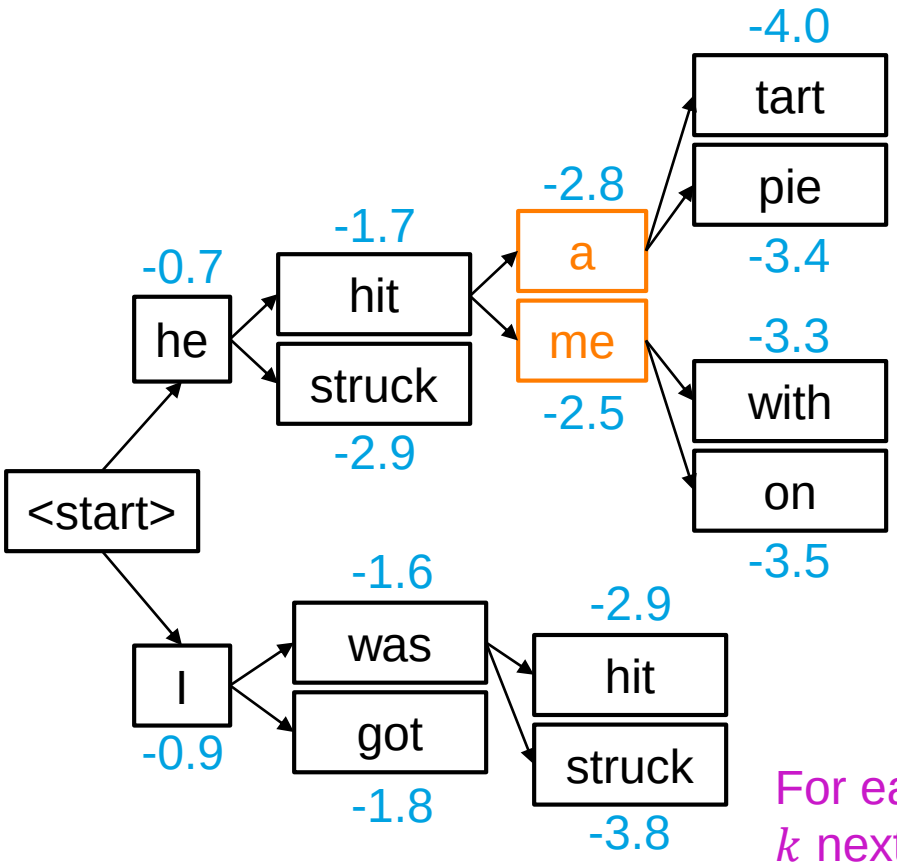



Of these $k^2$ hypotheses,
just keep $k$ with highest scores

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$

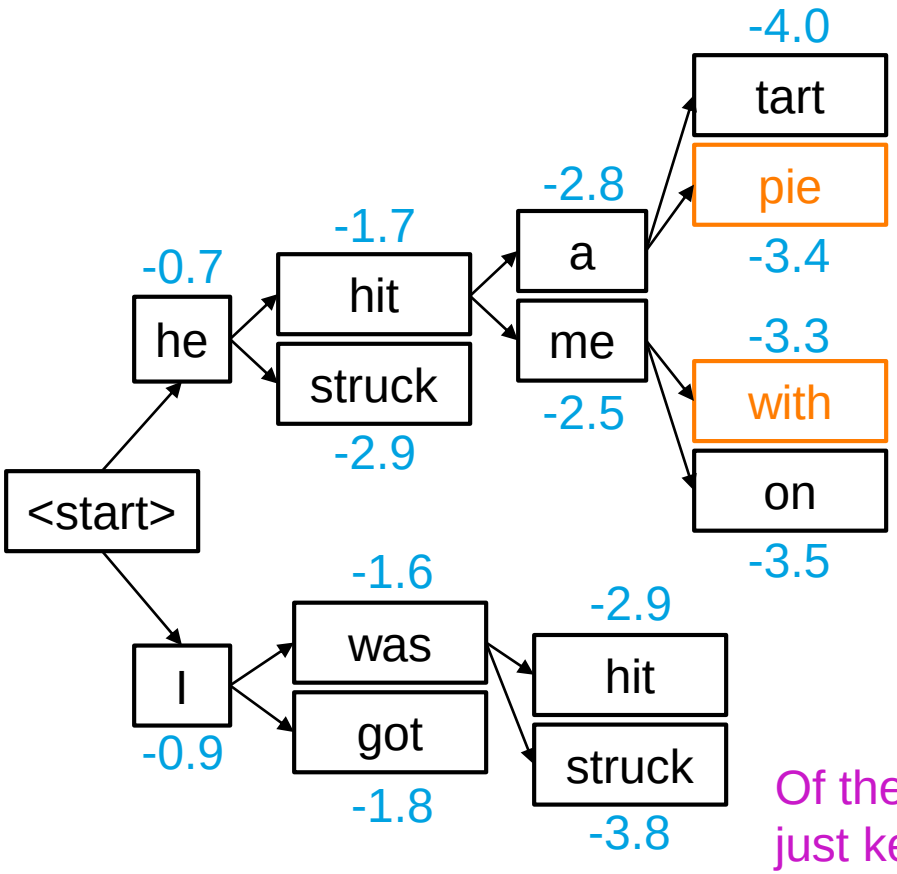



For each of the $k$ hypothesis, find top $k$ next words and calculate scores

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$

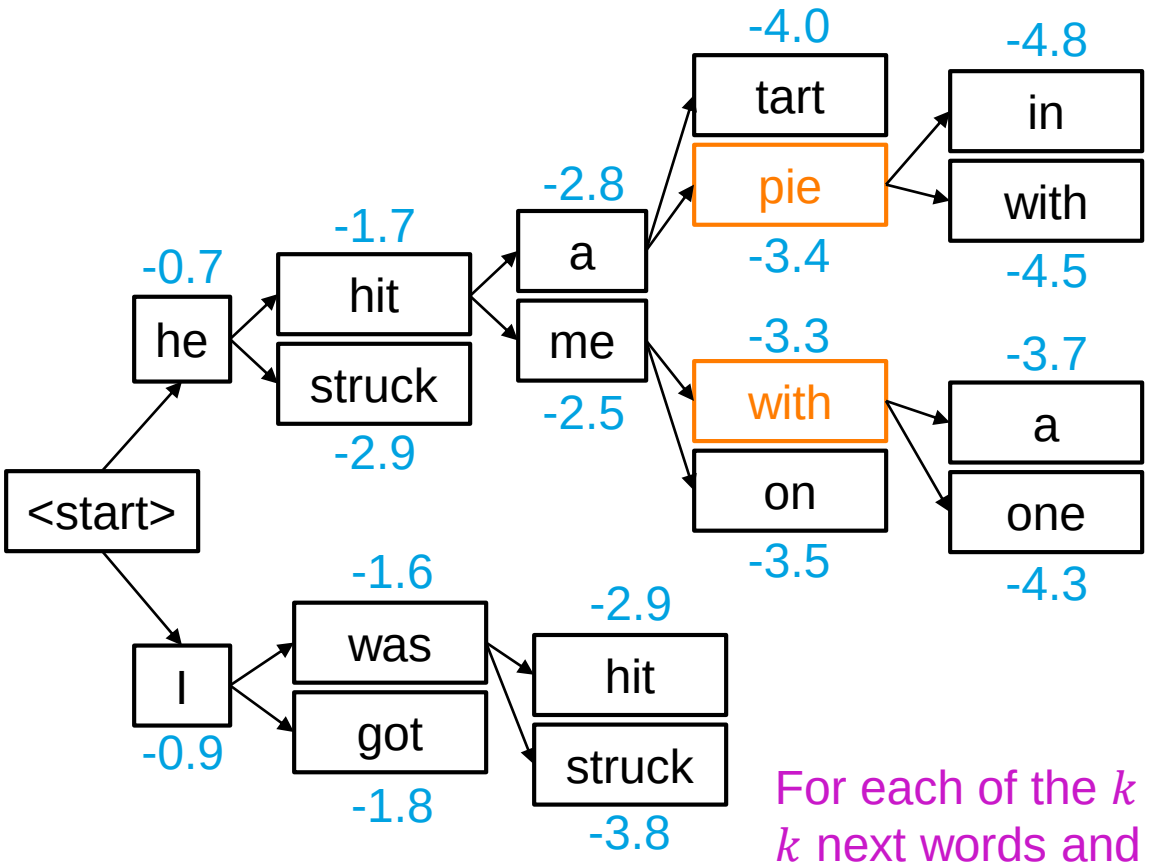



Of these $k^2$ hypotheses,
just keep $k$ with highest scores

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$

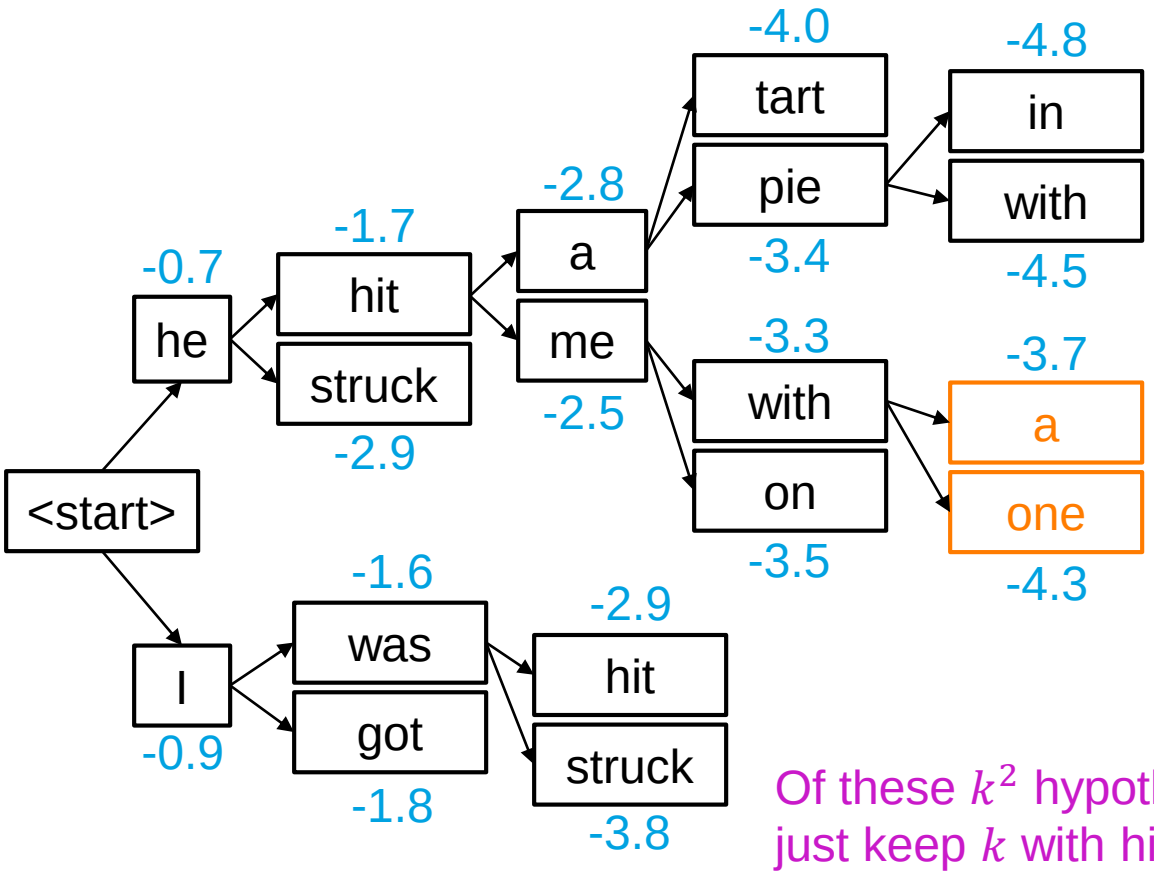



For each of the $k$ hypothesis, find top $k$ next words and calculate scores

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$

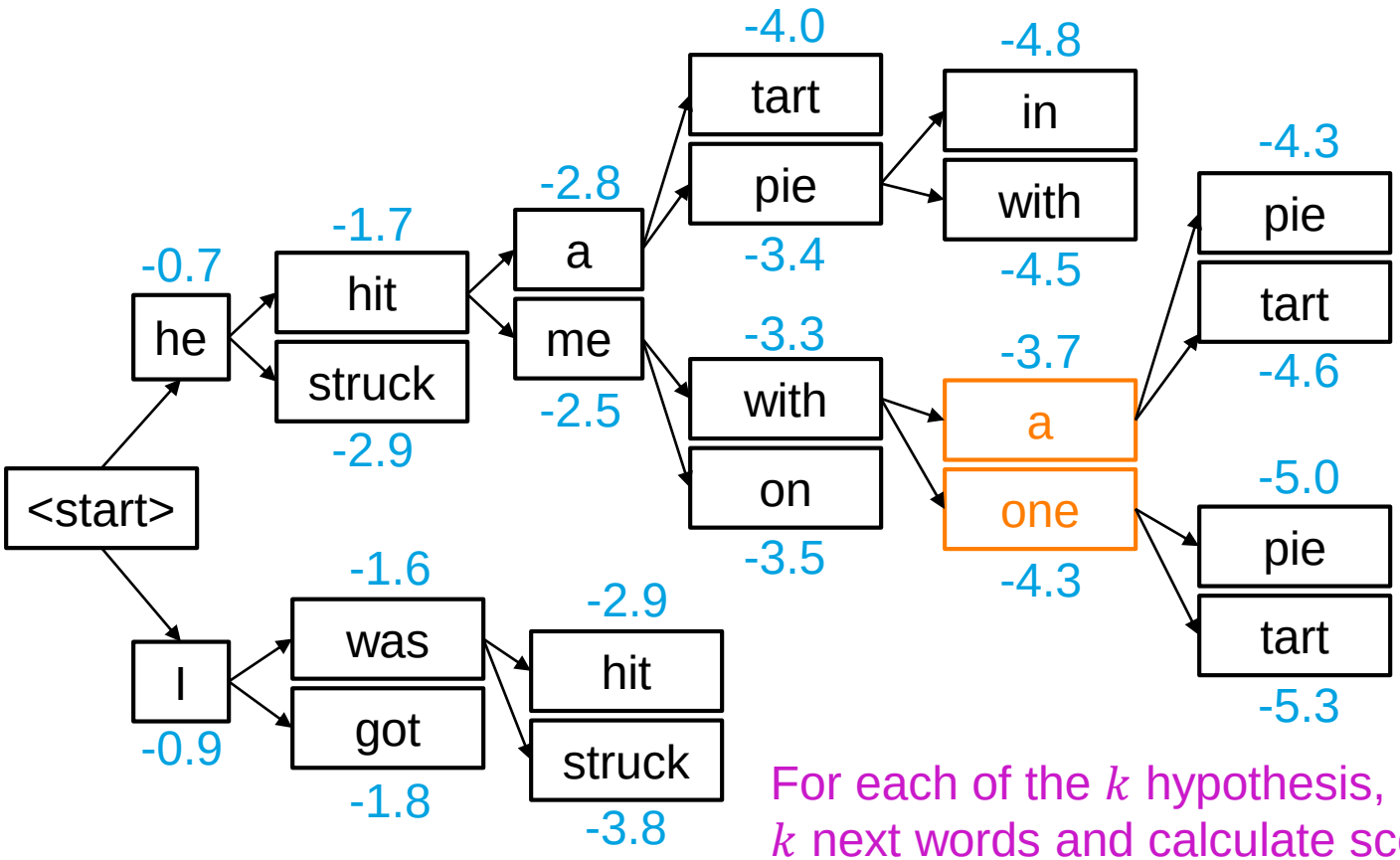



Of these $k^2$ hypotheses,
just keep $k$ with highest scores

# Beam search decoding: example

Beam size = $k$ = 2.

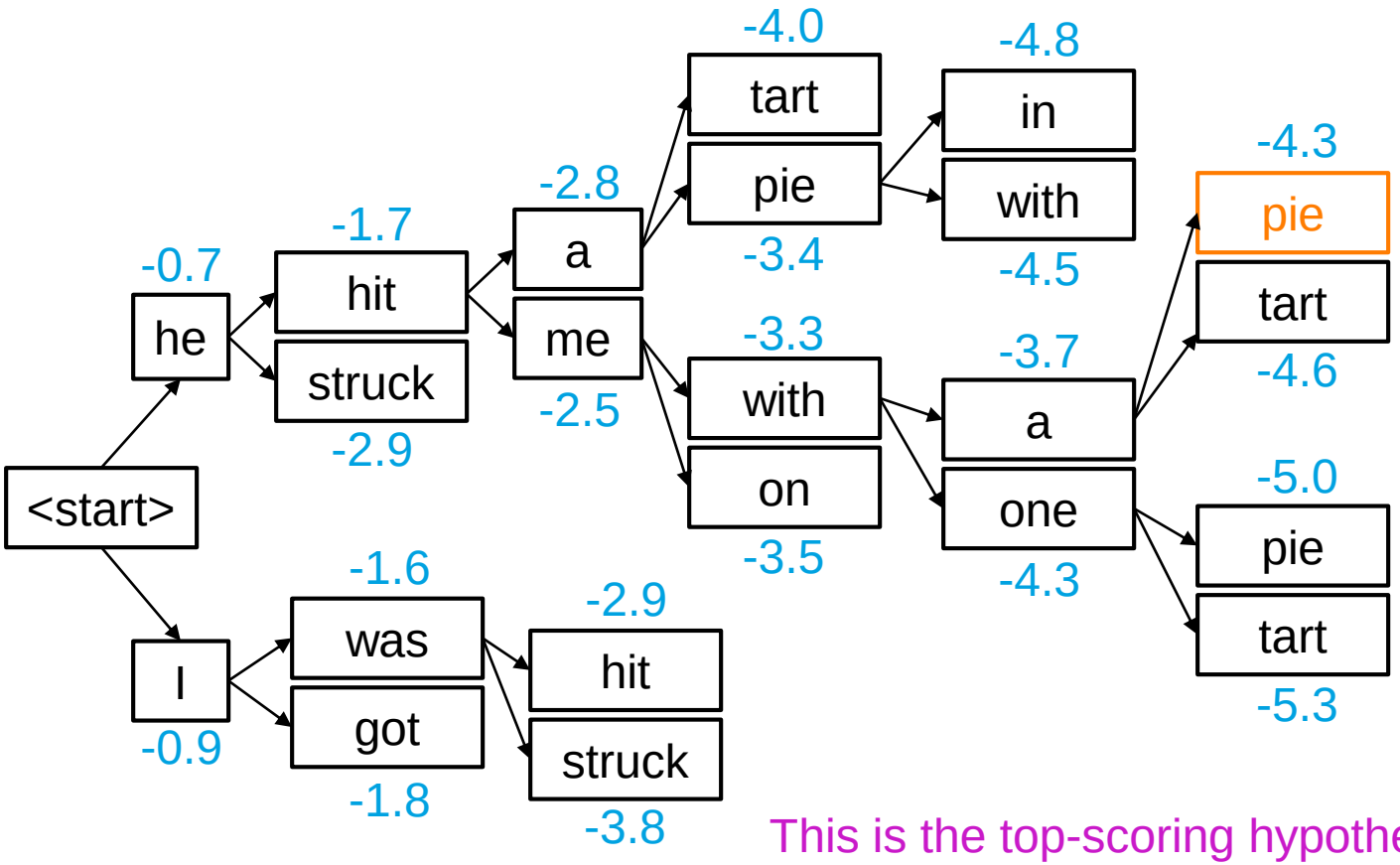
Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$



# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$

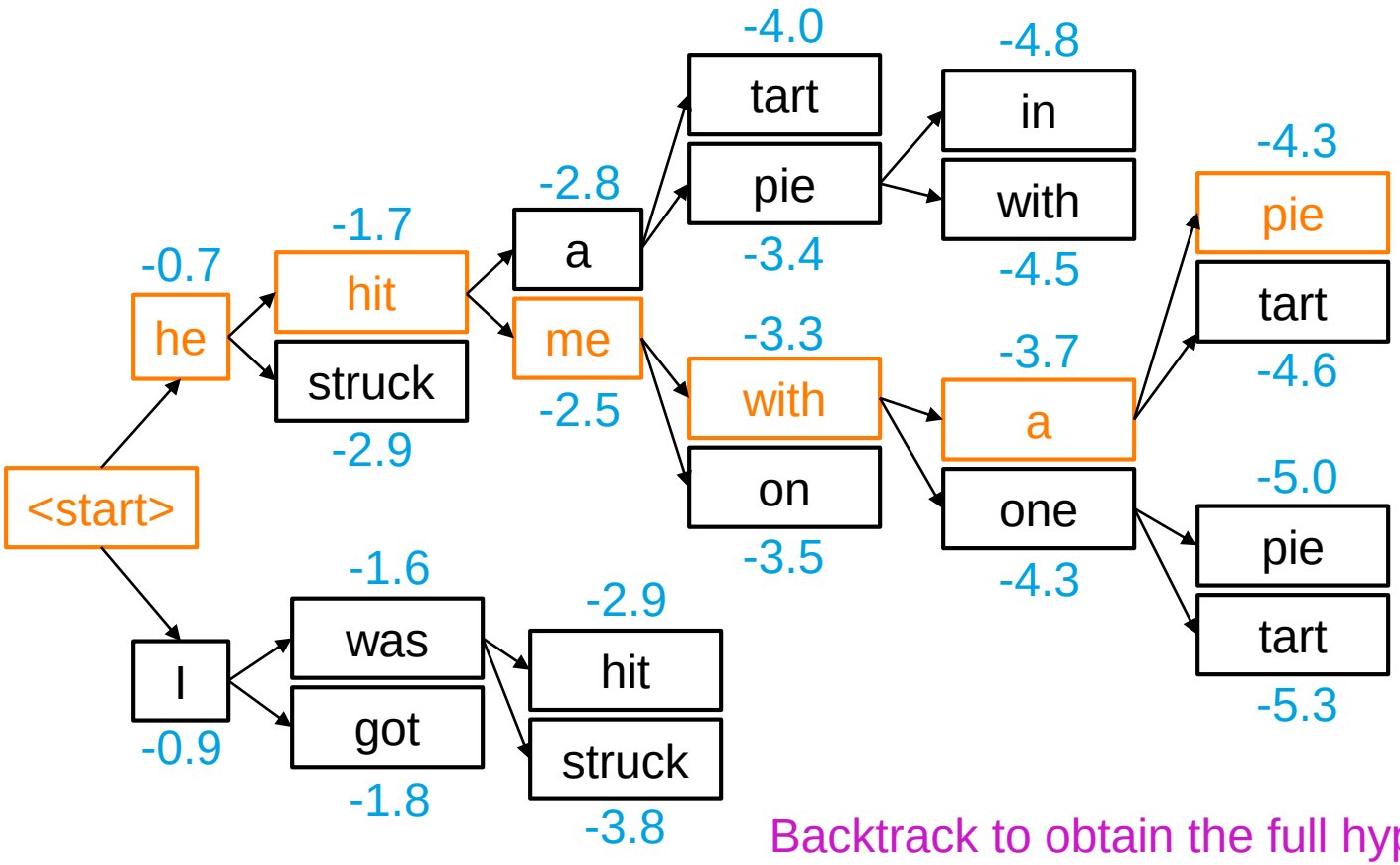



This is the top-scoring hypothesis!

# Beam search decoding: example

Beam size = $k$ = 2.

Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$




Backtrack to obtain the full hypothesis

# Beam search decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces an <END> token
  - For example, "<*START*> *he hit me with a pie* <*END*>"
- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete.
  - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
  - We reach timestep $T$, or
  - We have at least $n$ completed hypotheses

$T$ and $n$ are some pre-defined cutoffs

# Beam search decoding: finishing up

- With a list of completed hypotheses, how do we select the best one?
- Selecting one with the highest score?

$$\text{score}(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t | x) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$

  - Problem with this?
  - Longer hypotheses have lower scores
- Fix: Normalize by length. Use this to select the top one:

$$\frac{1}{t} \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$

# Beam search decoding

- Beam search is not guaranteed to find optimal solution
- It is even not guaranteed to outperform greedy decoding
  - But it is often better

# Decoding

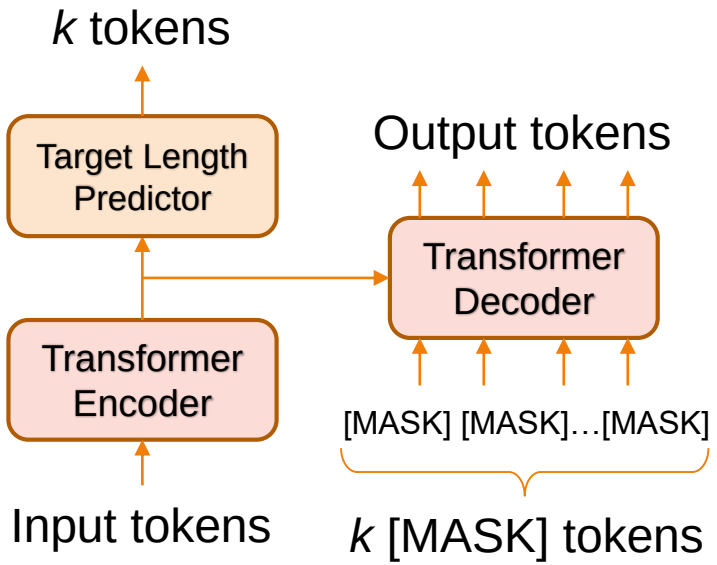
- Other desiderata
  - Reduce repetition
    - Penalize generation of already-seen tokens
      - Can also be incorporated into training (unlikelihood objective)
    - Prevent attending to the same words
  - Encourage diversity for certain tasks (e.g., dialog)
    - Sample, instead of being greedy
  - Avoid hallucination (generating things not in the input) for translation, summarization, etc.
    - Multiple causes and solutions

# Autoregressive vs. Non-Autoregressive Decoding

- Autoregressive: output tokens one by one
  - Slow
- Non-Autoregressive Decoding (NAT)
  - Transformers are parallelizable
  - Why not generating all the tokens simultaneously?

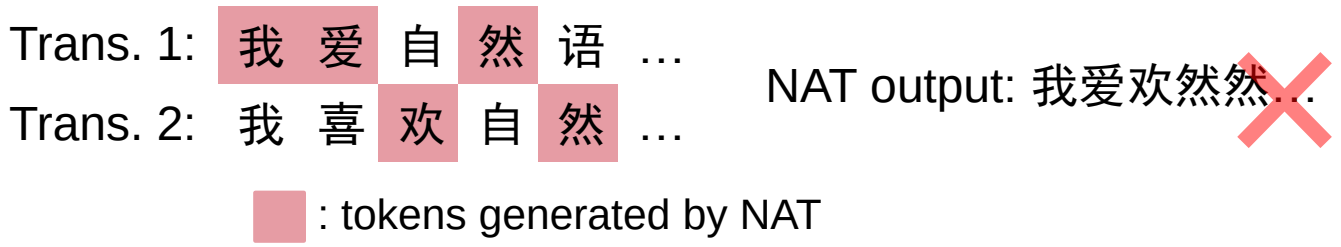
# Non-Autoregressive Decoding

- Encoder
  - Same as AT
- Decoder
  - Predict the target sequence length *k* from encoder output
    - Usually with a linear classifier
  - Input *k* special symbol to the decoder
  - Run decoding for all positions in parallel
  - No masking of future tokens




# Non-Autoregressive Decoding

- Multi-modal problem of naïve NAT
  - Tokens are generated independently.
  - Often fails when one input can be mapping to multiple possible outputs.
    - Ex: translate “I love NLP” into Chinese.




  - Solution: directed acyclic transformer, iterative NAT …

# Extensions

# Pointer Net / Copy Mechanism

- In some cases, we may wish to copy part of the input to the output
  - Ex: named entities, numbers

Source sentence: Federer suffered a 1-3 defeat to Nadal

Target sentence: Federer a subi une défaite 1-3 contre Nadal

# Pointer Net / Copy Mechanism

- In some cases, we may wish to copy part of the input to the output
  - Ex: semantic role labeling as seq2seq

Source sentence:

Tolkien's epic novel The Lord of the Rings was published in 1954-1955 …

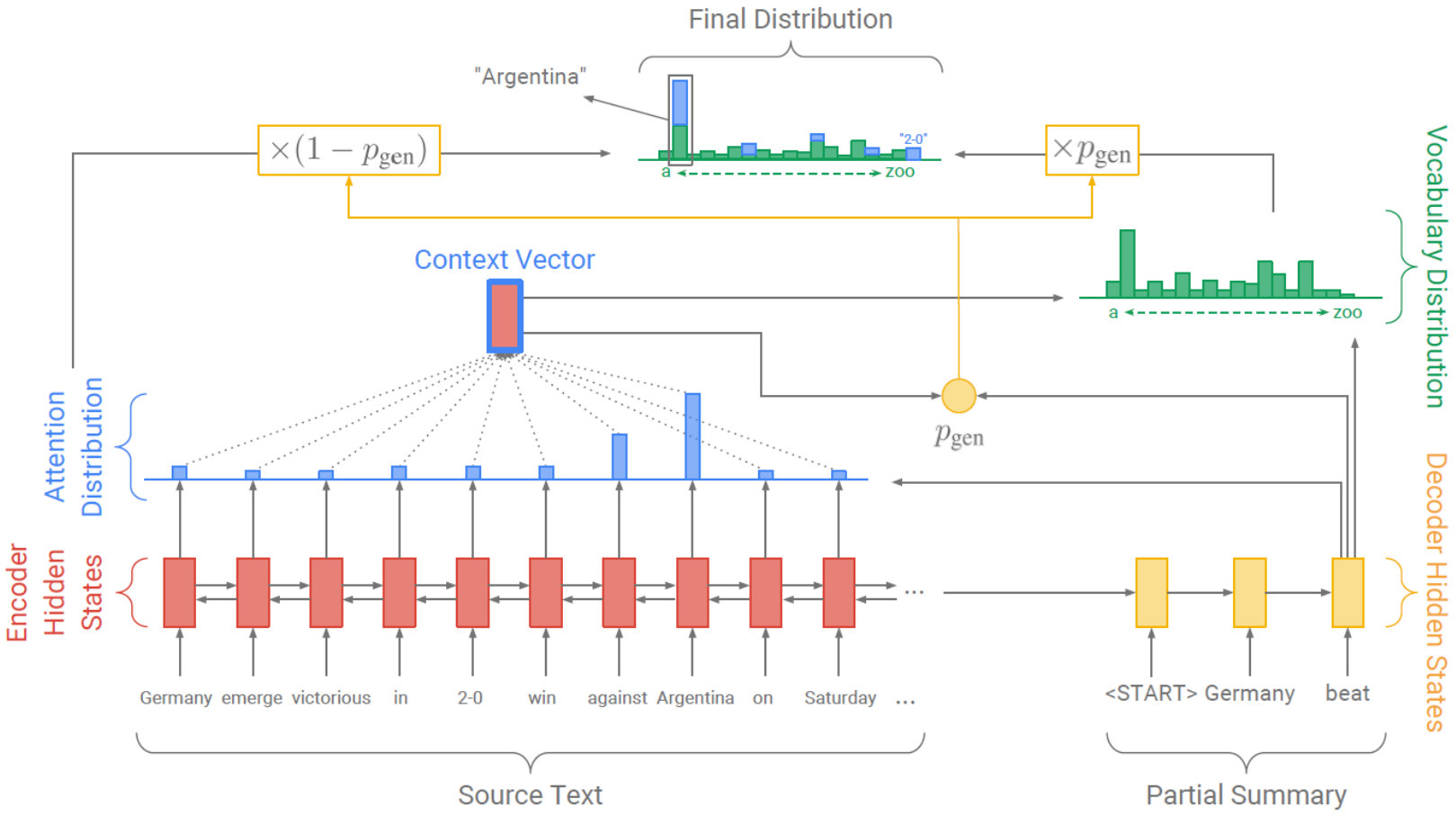
Target output:

Tolkien's epic novel [ The Lord of the Rings | *subject* ] [ was published | *predicate* ] [ in 1954-1955 | *temporal* ]

# Pointer Net / Copy Mechanism

- At each decoding step, calculate a Bernoulli distribution over generation vs. copying and sample from it
  - Generation: the old way of predicting a token
  - Copying: predict a distribution over the input tokens (i.e., a pointer)
    - This is similar to computing the attention distribution

# Pointer Net / Copy Mechanism

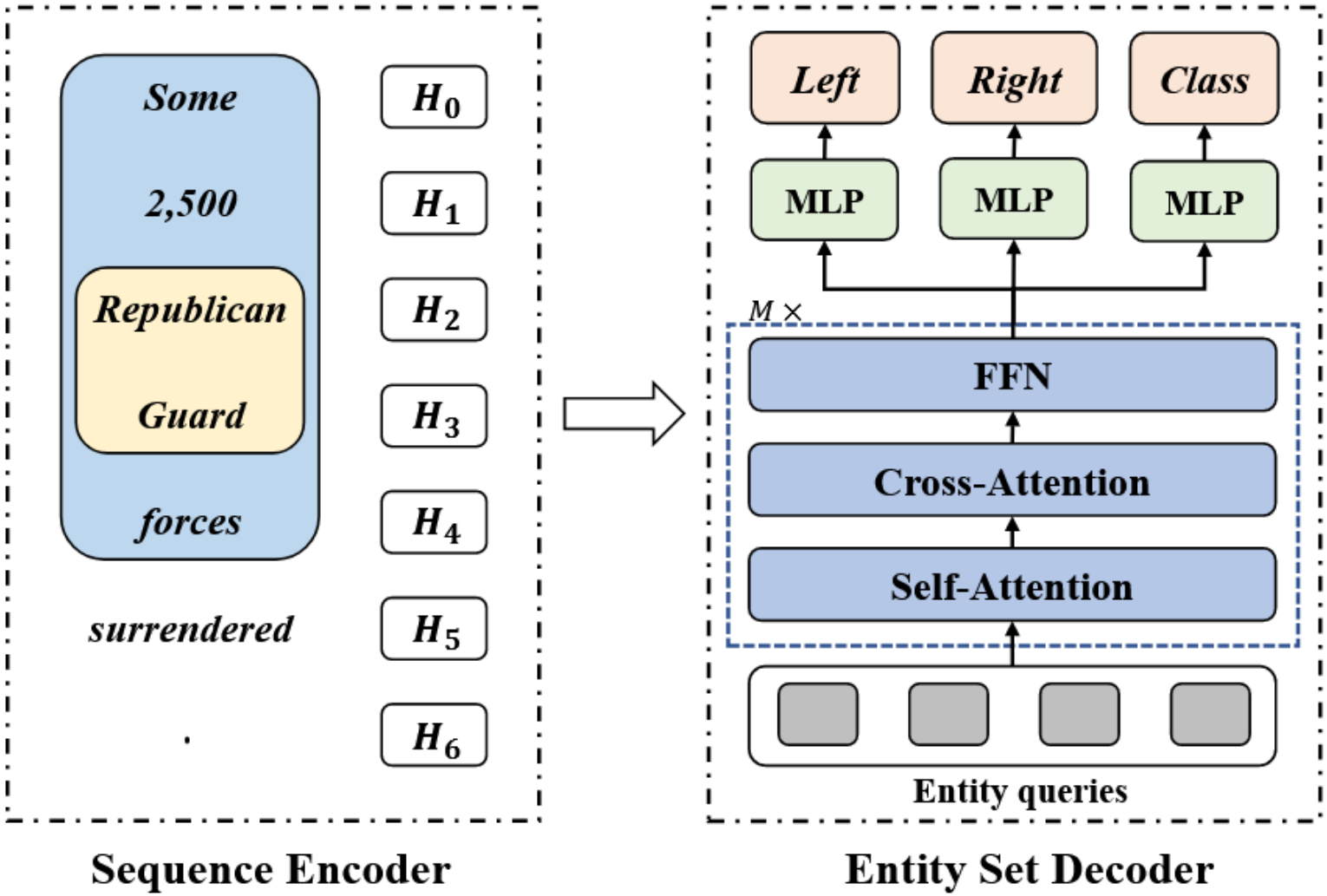



See, et al., “Get To The Point: Summarization with Pointer-Generator Networks”, ACL 2017

# Extensions of Seq2Seq

- Seq2Set
  - Decoding to a set (e.g., named entities)
  - Still use a Transformer decoder, but the input is a fixed number of “queries”
    - Query vectors are learnable model parameters
    - No position embedding, no mask
    - Simultaneous decoding across all the positions
    - An output can be null, allowing variable set sizes

# Extensions of Seq2Seq



# Extensions of Seq2Seq

- X2Seq (requiring an X-encoder)
  - Image captioning
  - Visual question answering
  - Structured data (e.g., table) to text
  - …

# Summary

# Sequence to Sequence

- Many applications
  - MT, paraphrase, summarization, ...
- Methods: encoder-decoder
  - Recurrent neural network
  - Attention
  - Transformer: cross-attention
- Learning
  - Maximizing conditional likelihood on a parallel corpus
- Decoding
  - Greedy, beam-search, non-autoregressive
- Extensions
  - Pointer Net / Copy Mechanism
  - Seq2Set, X2Seq