SFU NatLangLab

CMPT 413/713: Natural Language Processing

Attention for Seq2Seq models (Attention)

Spring 2024
2024-02-07

Adapted from slides from Danqi Chen and Karthik Narasimhan
(with some content from slides from Abigail See, Graham Neubig)

Overview

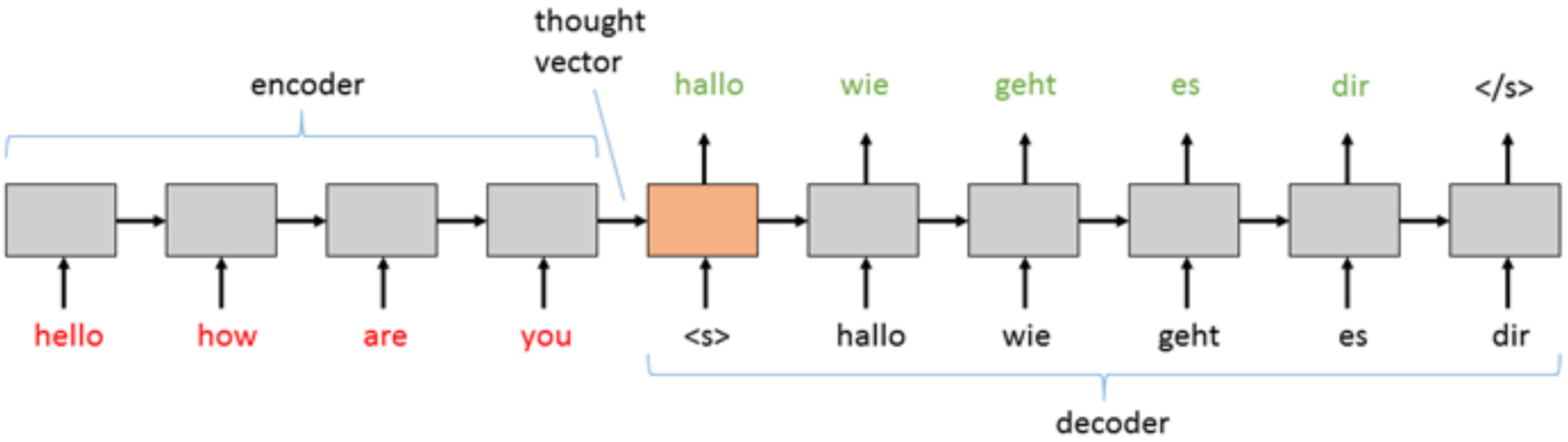
- Review of Seq2Seq models
- Attention

Sequence to sequence models

Neural Machine Translation

- A **single neural network** is used to translate from source to target
- Architecture: Encoder-Decoder
  - Two main components:
    - Encoder: Convert source sentence (input) into a vector/matrix
    - Decoder: Convert encoding into a sentence in target language (output)

Sequence to Sequence learning (Seq2seq)



- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

Seq2seq training

- Similar to training a language model!
- Minimize cross-entropy loss:

$$\sum_{t=1}^{T} - \log P(y_t \mid y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

- Back-propagate gradients through *both decoder and encoder*
- Need a really big corpus

36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

Decoding strategies and evaluation

Decoding Strategies

- Sampling
  - Sample for diverse generation
  - Can sample from top-10 choices or top-50%
- Greedy decoding - Efficient and fast, good starting point
- Beam search - Practical middle ground

Beam decoding

- Different hypotheses may produce $\langle eos \rangle$ (end) token at different time steps
  - When a hypothesis produces $\langle eos \rangle$, stop expanding it and place it aside
- Continue beam search until:
  - All k hypotheses produce $\langle eos \rangle$ OR
  - Hit max decoding limit T
- Select top hypotheses using the *normalized* likelihood score

$$\frac{1}{T}\sum_{t=1}^{T}\log P(y_t \,|\, y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

  - Otherwise shorter hypotheses have higher scores

Evaluating translation quality

- Two main criteria:
  - Adequacy: Translation $w^{(t)}$ should adequately reflect the linguistic content of $w^{(s)}$
  - Fluency: Translation $w^{(t)}$ should be fluent text in the target language

	Adequate?	Fluent?
To Vinay it like Python	yes	no
Vinay debugs memory leaks	no	yes
Vinay likes Python	yes	yes

Different translations of *A Vinay le gusta Python*

BLEU: modified n-gram precision

n-gram precision

$$\text{BLEU-N} = \exp \frac{1}{N} \sum_{n=1}^{N} \log p_n$$

geometric mean over several values of n
(up to N=4)

Two modifications:

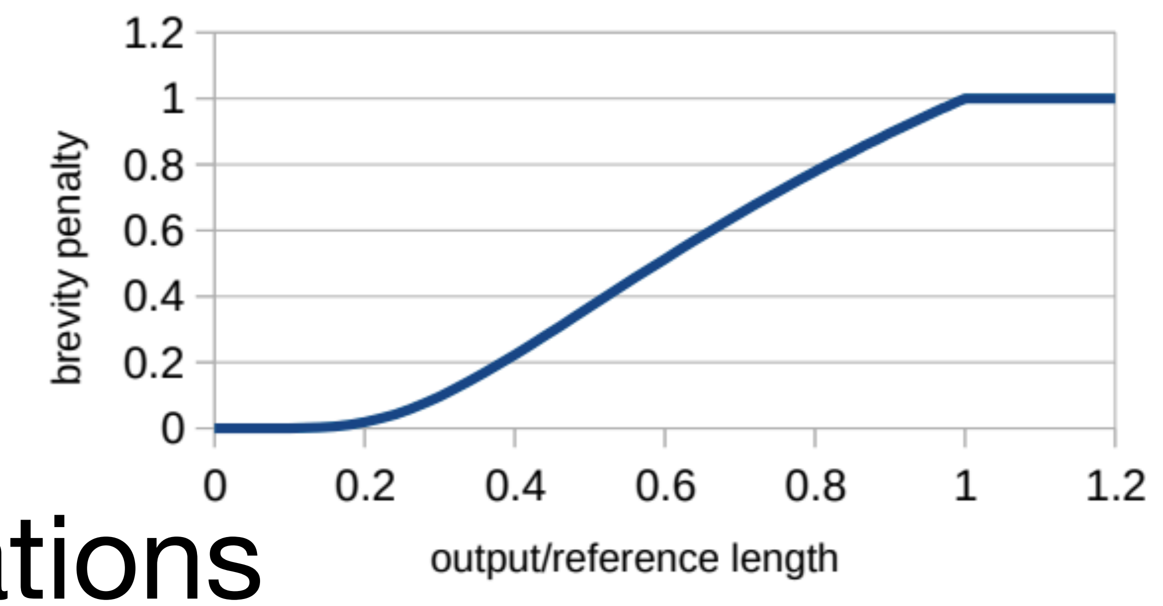
- To avoid $\log 0$, all precisions are smoothed

Various smoothing techniques
add 1 to numerator/denominator

- Each n-gram in reference can be used at most once
  - Ex. **Hypothesis**: *to to to to to* vs **Reference**: *to be or not to be*
    should not get a unigram precision of 1 ($p_1 = 2/5$)

clipped count

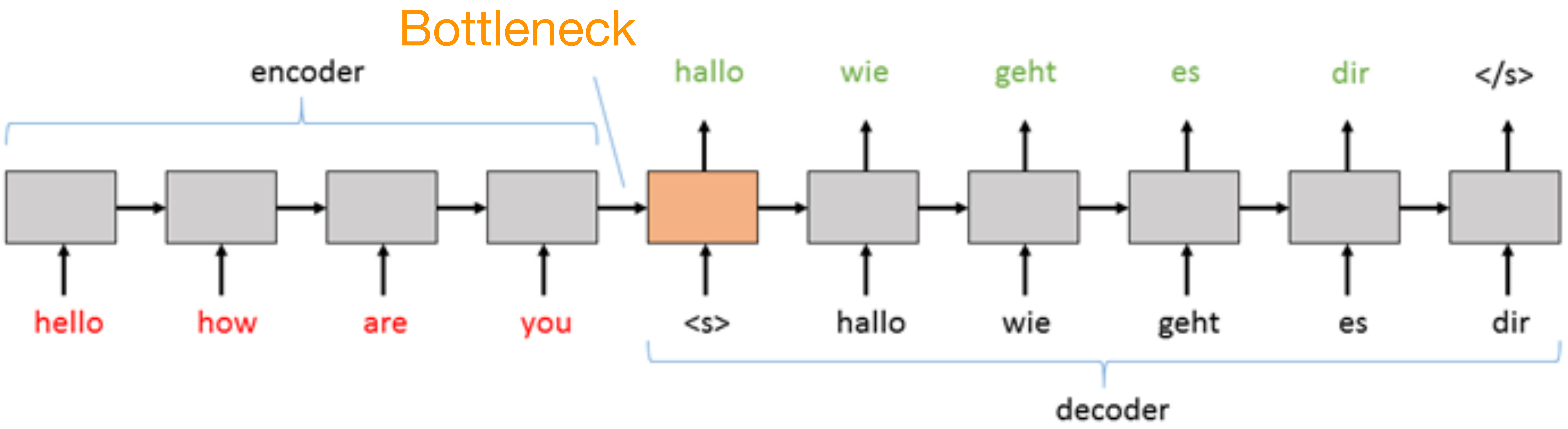
Precision-based metrics favor short translations



- Solution: Multiply score with a brevity penalty for translations shorter than reference, $BP = e^{1-r/h}$

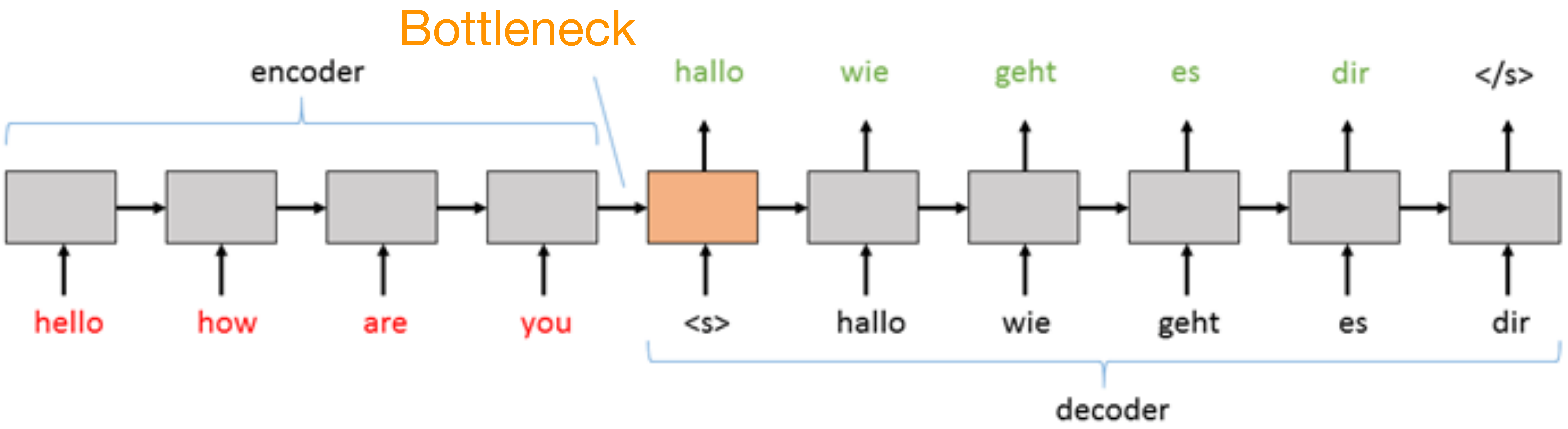
r = reference length, h = hypothesis length

Issues with vanilla seq2seq



- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- Overfitting

Issues with vanilla seq2seq



- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- **Overfitting**

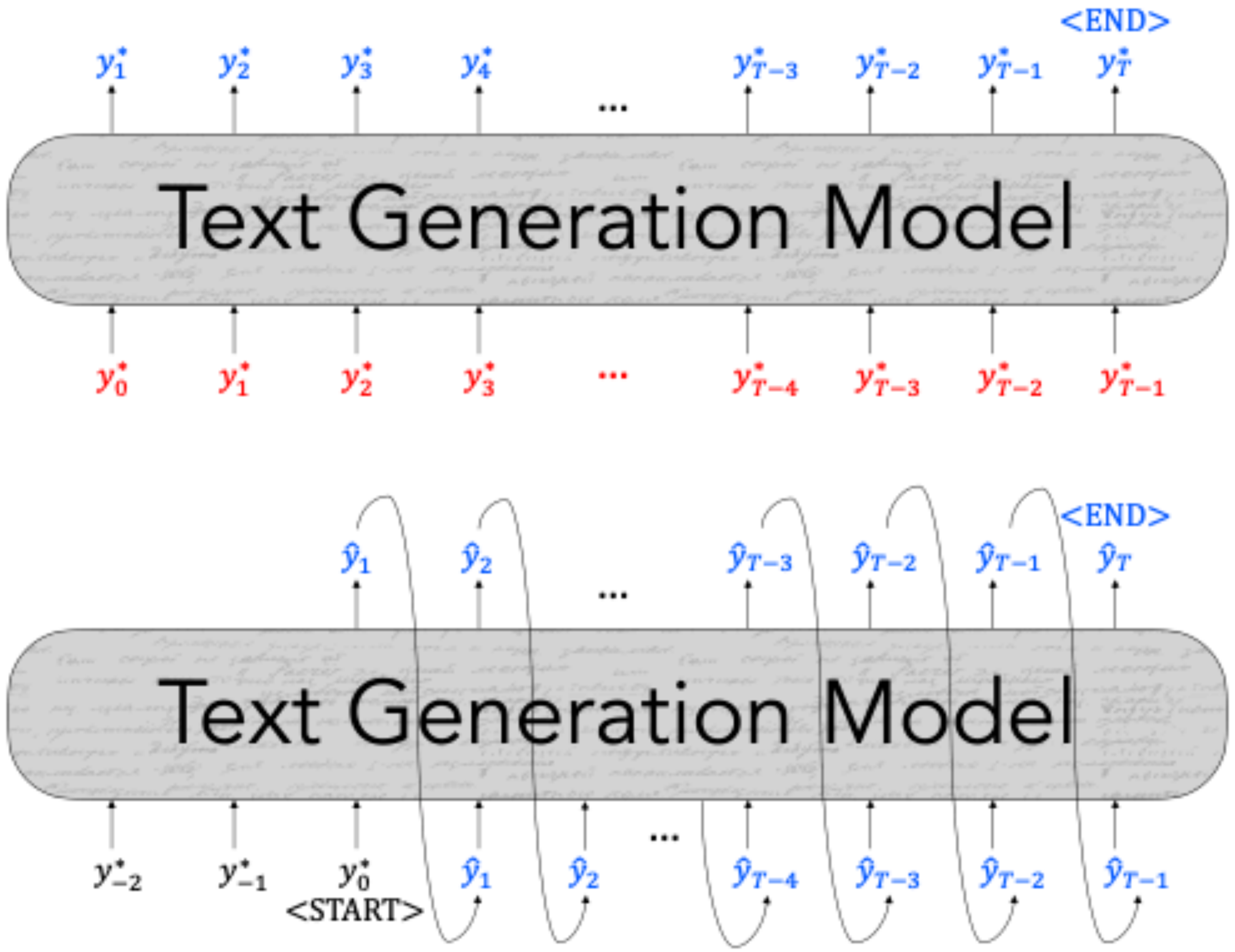
Exposure bias

- Discrepancy in model input between training and generation time
- During training, model inputs are gold context tokens

$$\mathcal{L}_{MLE} = -\sum_{t=1}^{T} \log P(y_t^*|\{y_{<t}^*\})$$

- At generation time, inputs are previously-decoded tokens

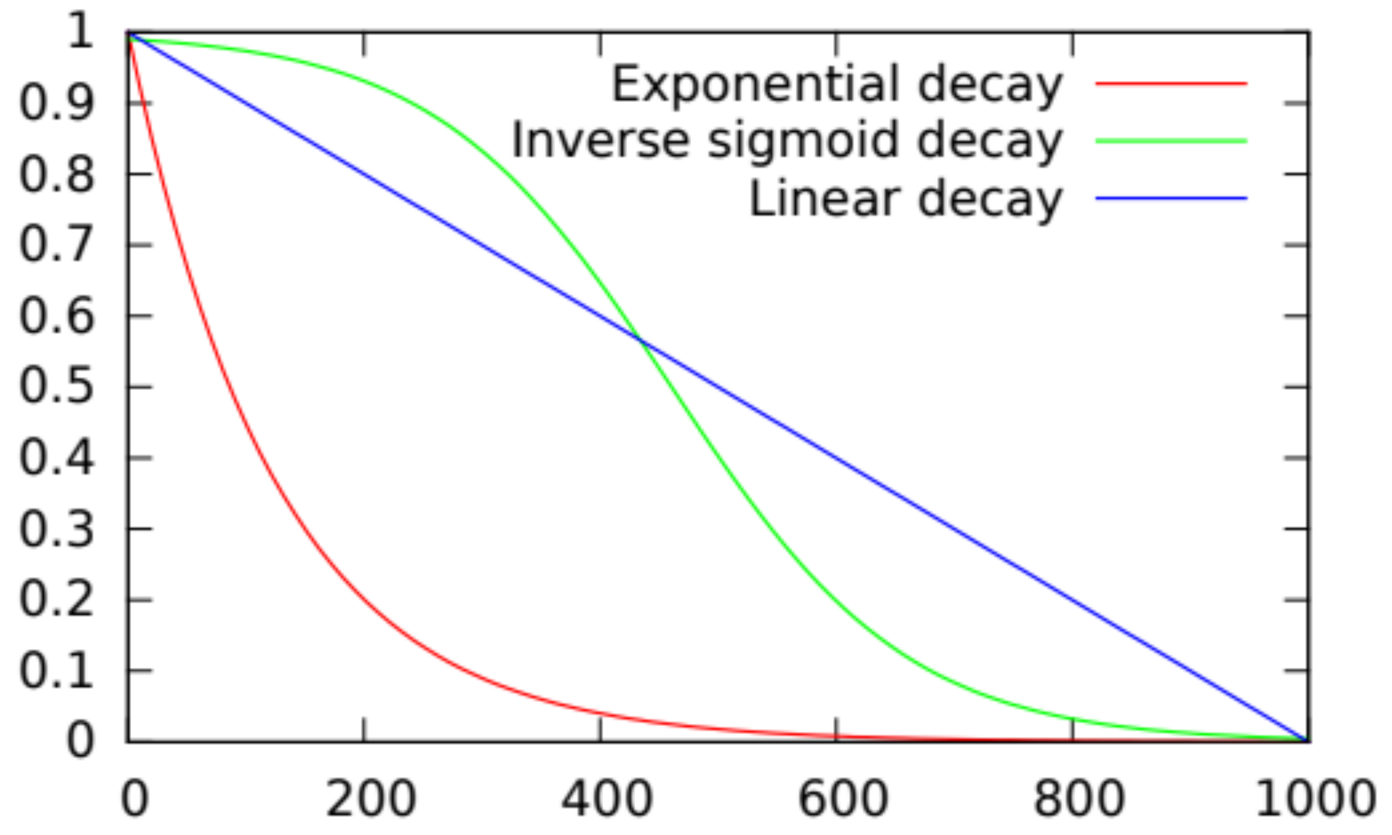
$$\mathcal{L}_{dec} = -\sum_{t=1}^{T} \log P(\hat{y}_t|\{\hat{y}_{<t}\})$$



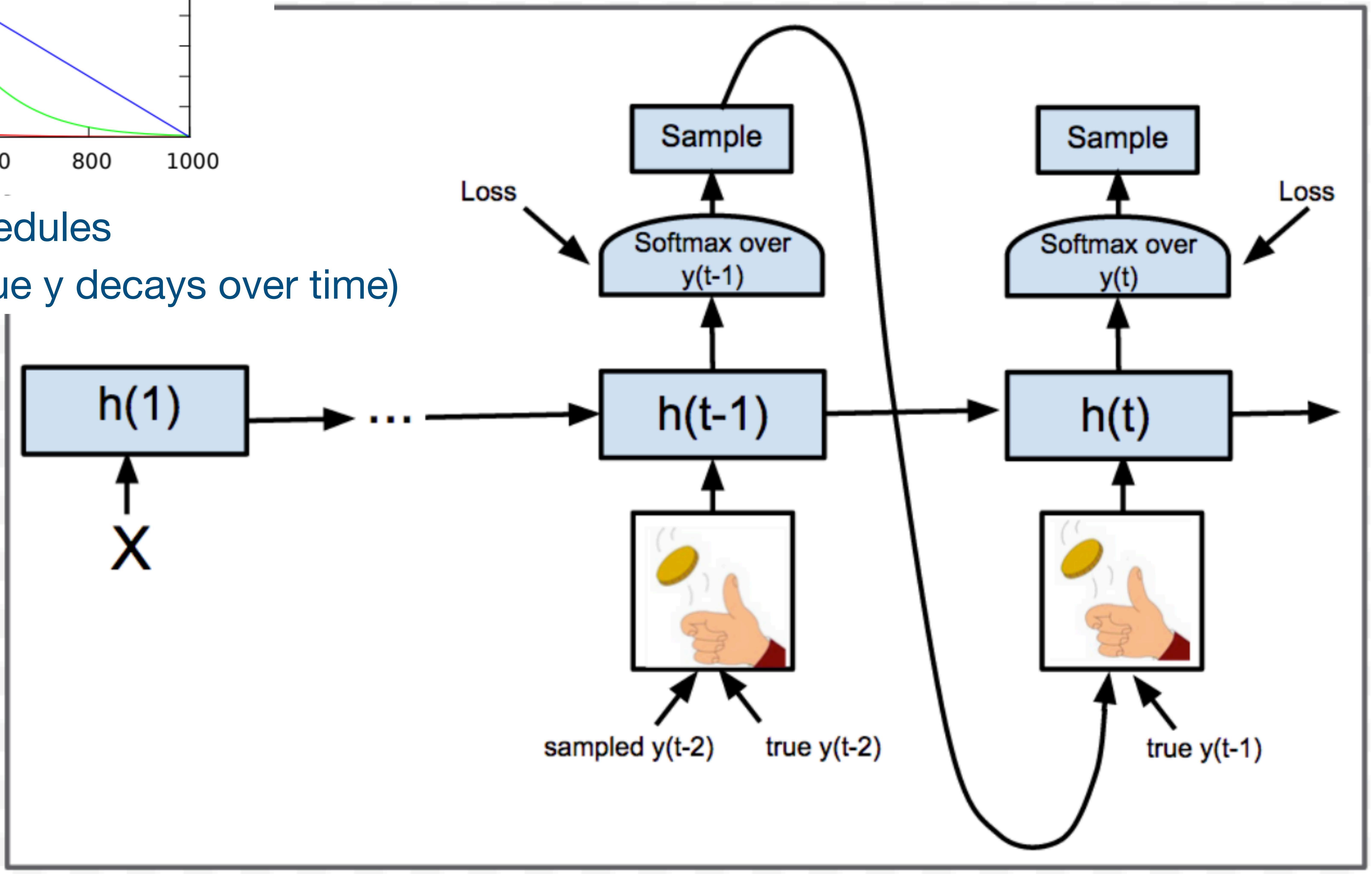
Student forcing: use predicted tokens during training

Scheduled sampling: use decoded token with some probability p, increase p over time

Scheduled Sampling



Possible decay schedules
(probability using true y decays over time)



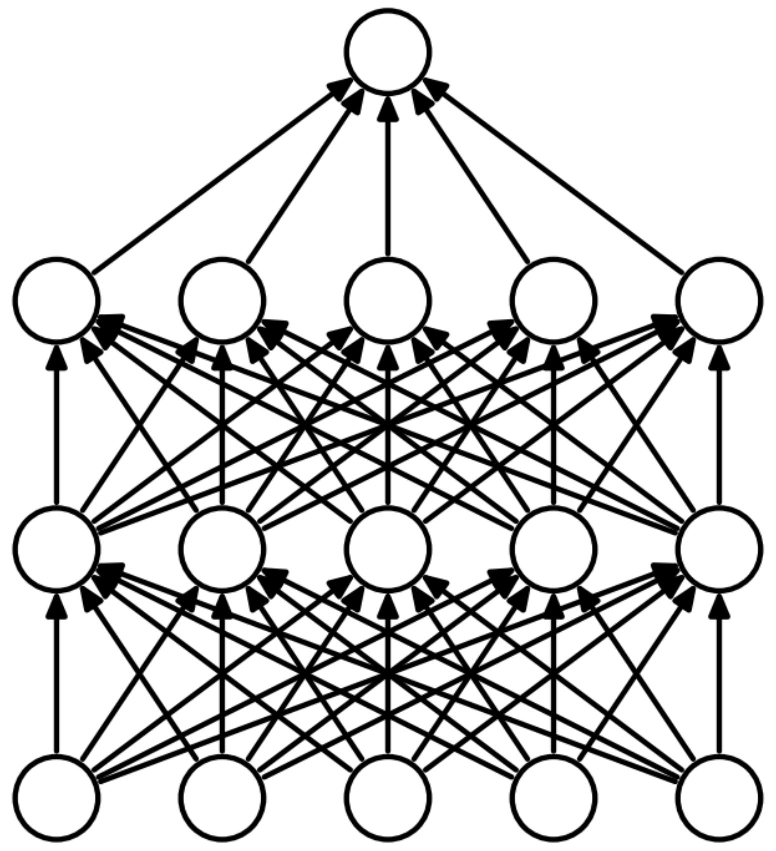
(figure credit: Bengio et al, 2015)

Regularization

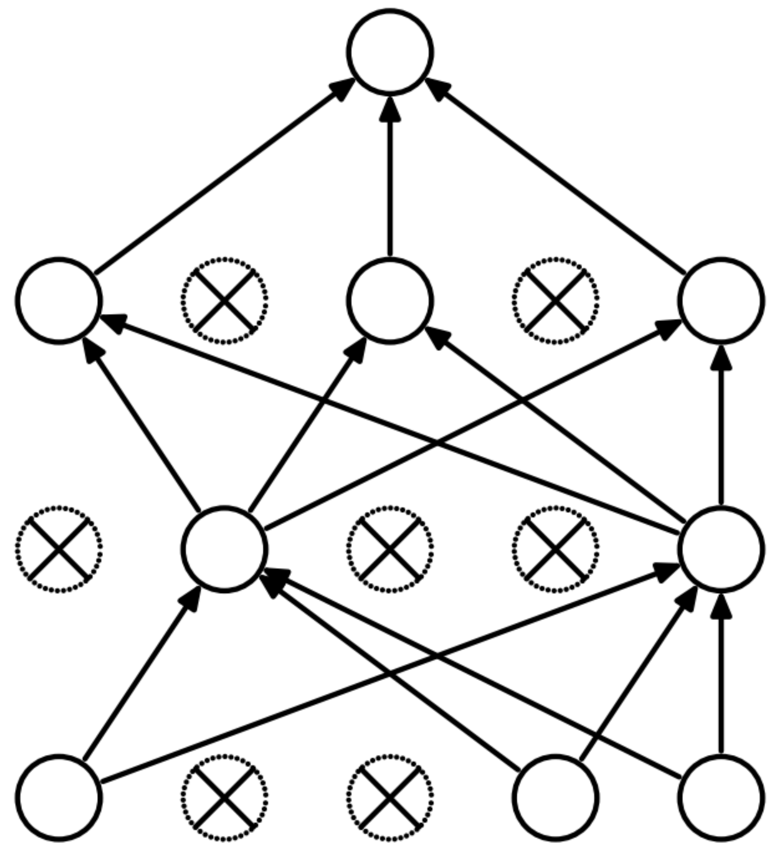
- Weight Decay
- Dropout
- Ensembling

Regularization: Dropout

- Form of regularization for RNNs (and any NN in general)
- **Idea:** “Handicap” NN by removing hidden units **stochastically**
  - set each hidden unit in a layer to 0 with probability p during training ($p = 0.5$ usually works well)
  - scale outputs by $1/(1-p)$
  - hidden units forced to learn more general patterns
- **Test time:** Use all activations (no need to rescale)



(a) Standard Neural Net



(b) After applying dropout.

Dropout and attention improves translation

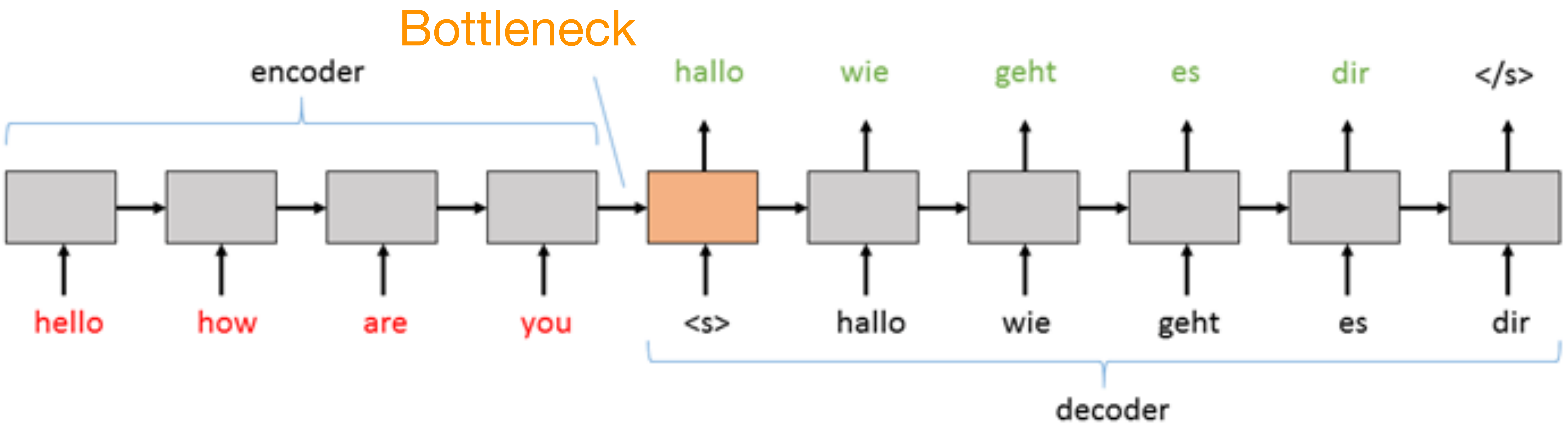
System	Ppl	BLEU
Winning WMT'14 system – *phrase-based* + *large LM* (Buck et al., 2014)		20.7
Existing NMT systems		
RNNsearch (Jean et al., 2015)		16.5
RNNsearch + unk replace (Jean et al., 2015)		19.0
RNNsearch + unk replace + large vocab + *ensemble* 8 models (Jean et al., 2015)		**21.6**
Our NMT systems		
Base	10.6	11.3
Base + reverse	9.9	12.6 (+*1.3*)
Base + reverse + dropout	8.1	14.0 (+*1.4*)
Base + reverse + dropout + global attention (*location*)	7.3	16.8 (+*2.8*)
Base + reverse + dropout + global attention (*location*) + feed input	6.4	18.1 (+*1.3*)
Base + reverse + dropout + local-p attention (*general*) + feed input	5.9	19.0 (+*0.9*)
Base + reverse + dropout + local-p attention (*general*) + feed input + unk replace		20.9 (+*1.9*)
Ensemble 8 models + unk replace		**23.0** (+*2.1*)

WMT'14 English to German Results

(Luong et al, 2015)

Sequence to sequence models with attention

Issues with vanilla seq2seq

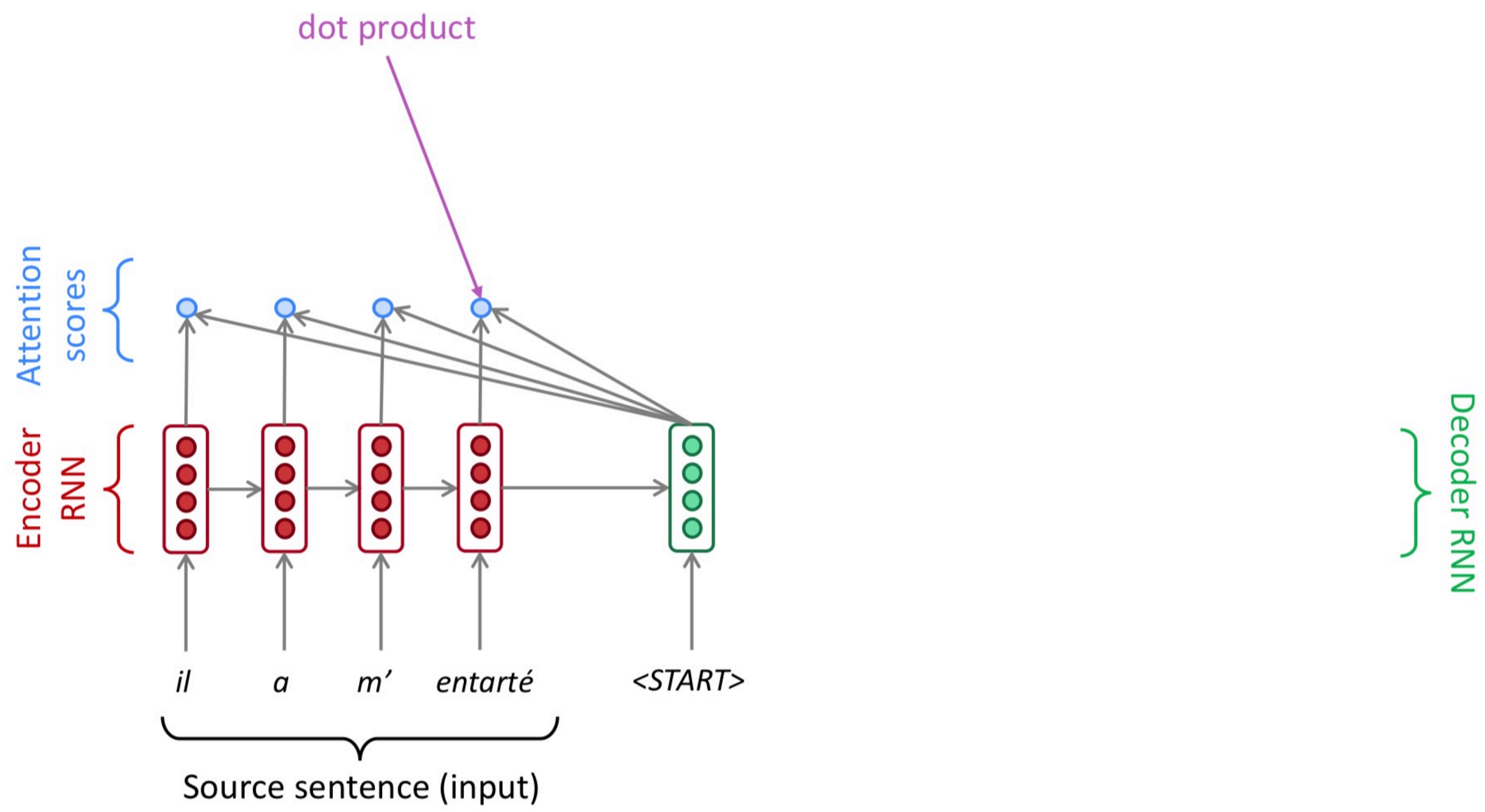


- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- **Longer sequences can lead to vanishing gradients**
- Overfitting

Attention

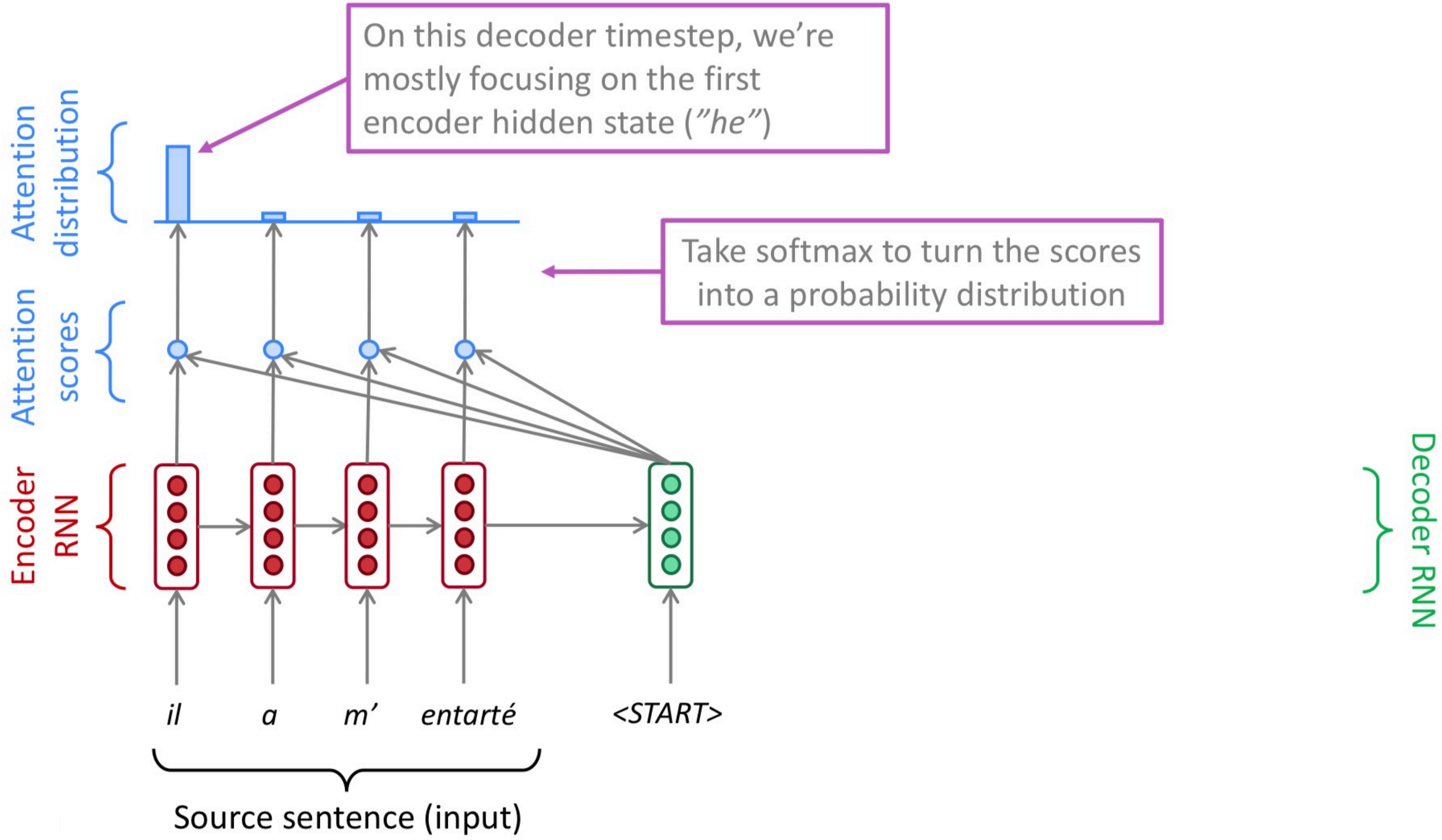
- The neural MT equivalent of alignment models
- Key idea: At each time step during decoding, **focus on a particular part** of source sentence
  - This depends on the decoder's current hidden state (i.e. notion of what you are trying to decode)
  - Usually implemented as a probability distribution over the hidden states of the encoder (h_i^{enc})

Seq2seq with attention



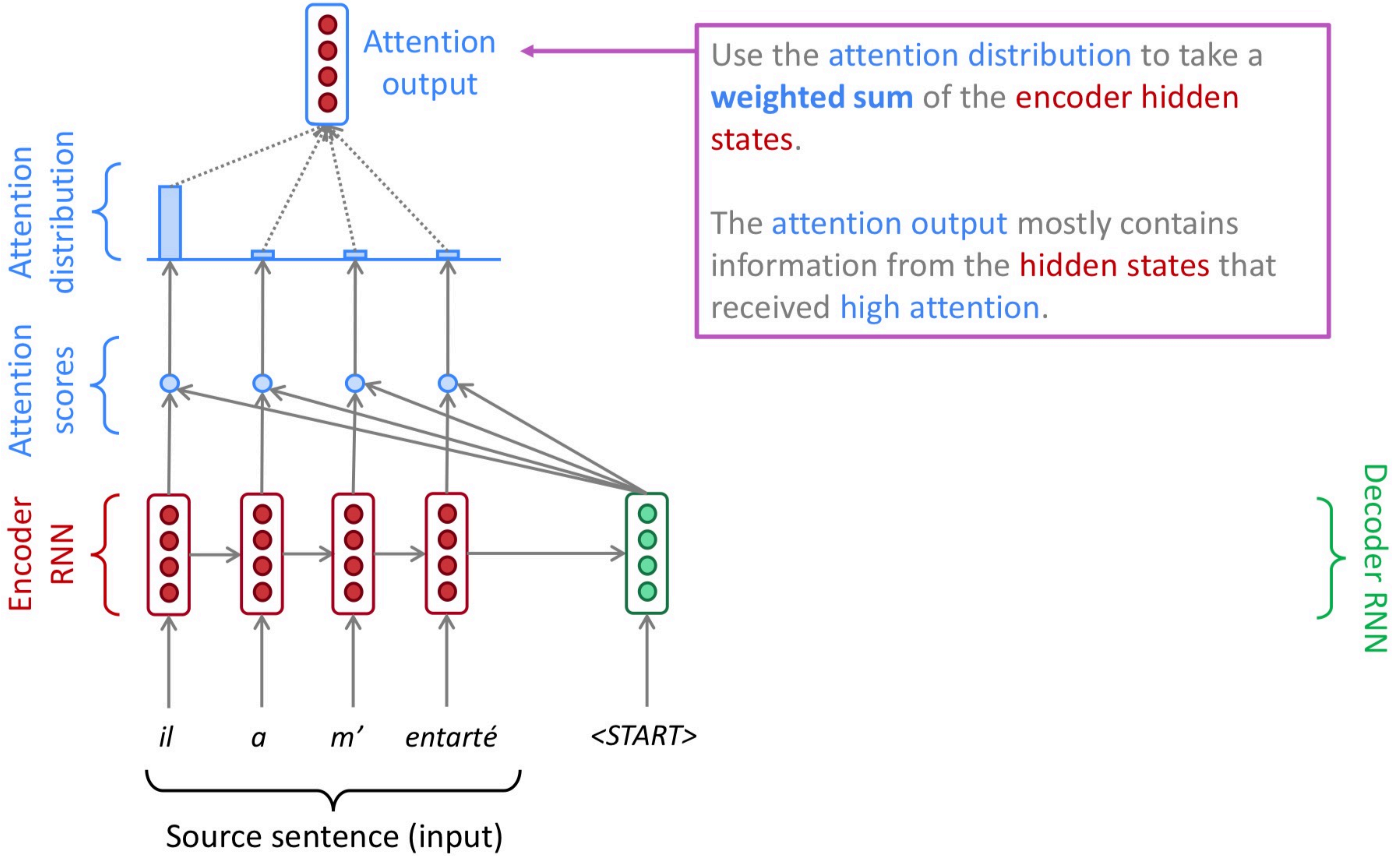
(slide credit: Abigail See)

Seq2seq with attention



(slide credit: Abigail See)

Seq2seq with attention

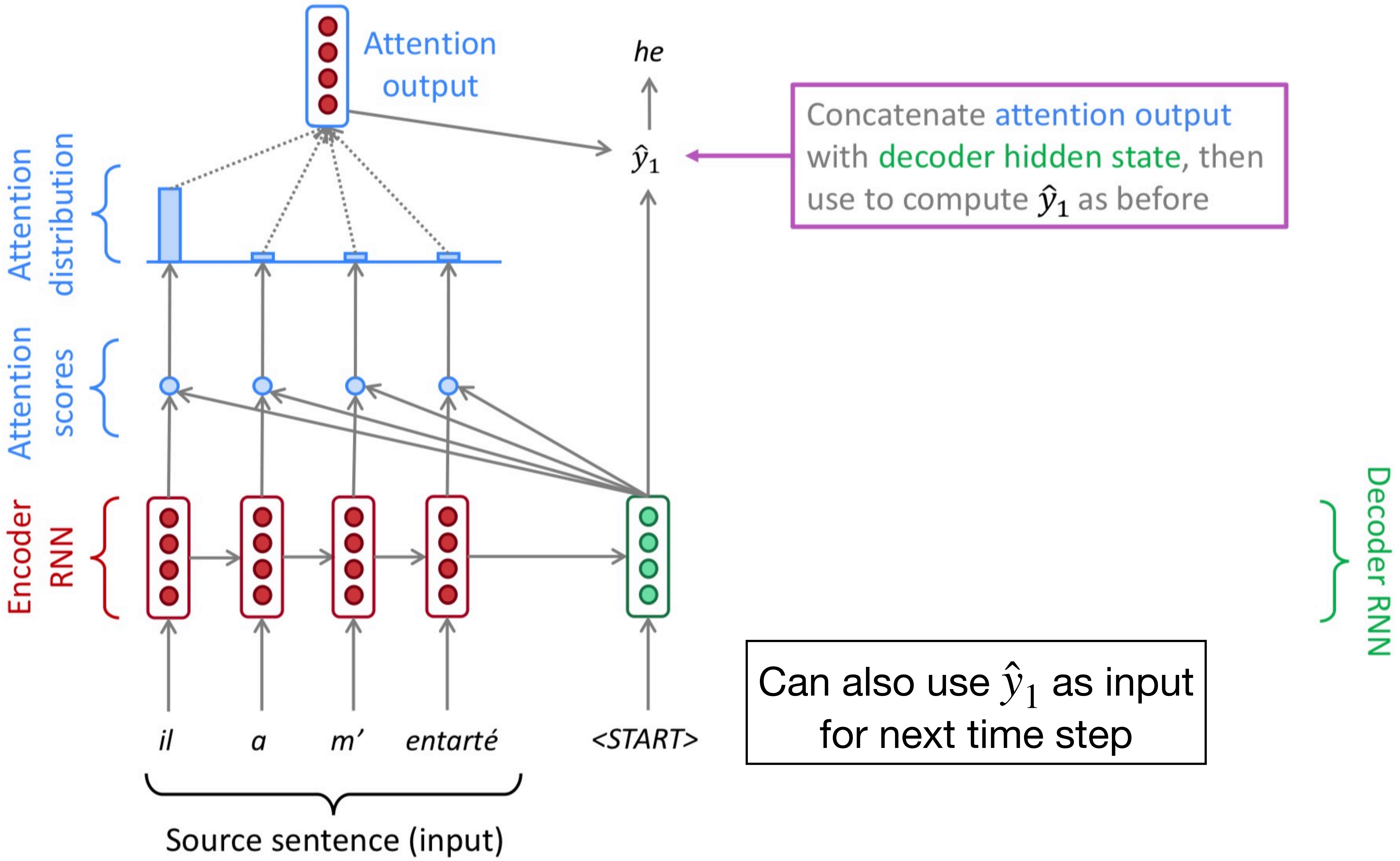


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

(slide credit: Abigail See)

Seq2seq with attention

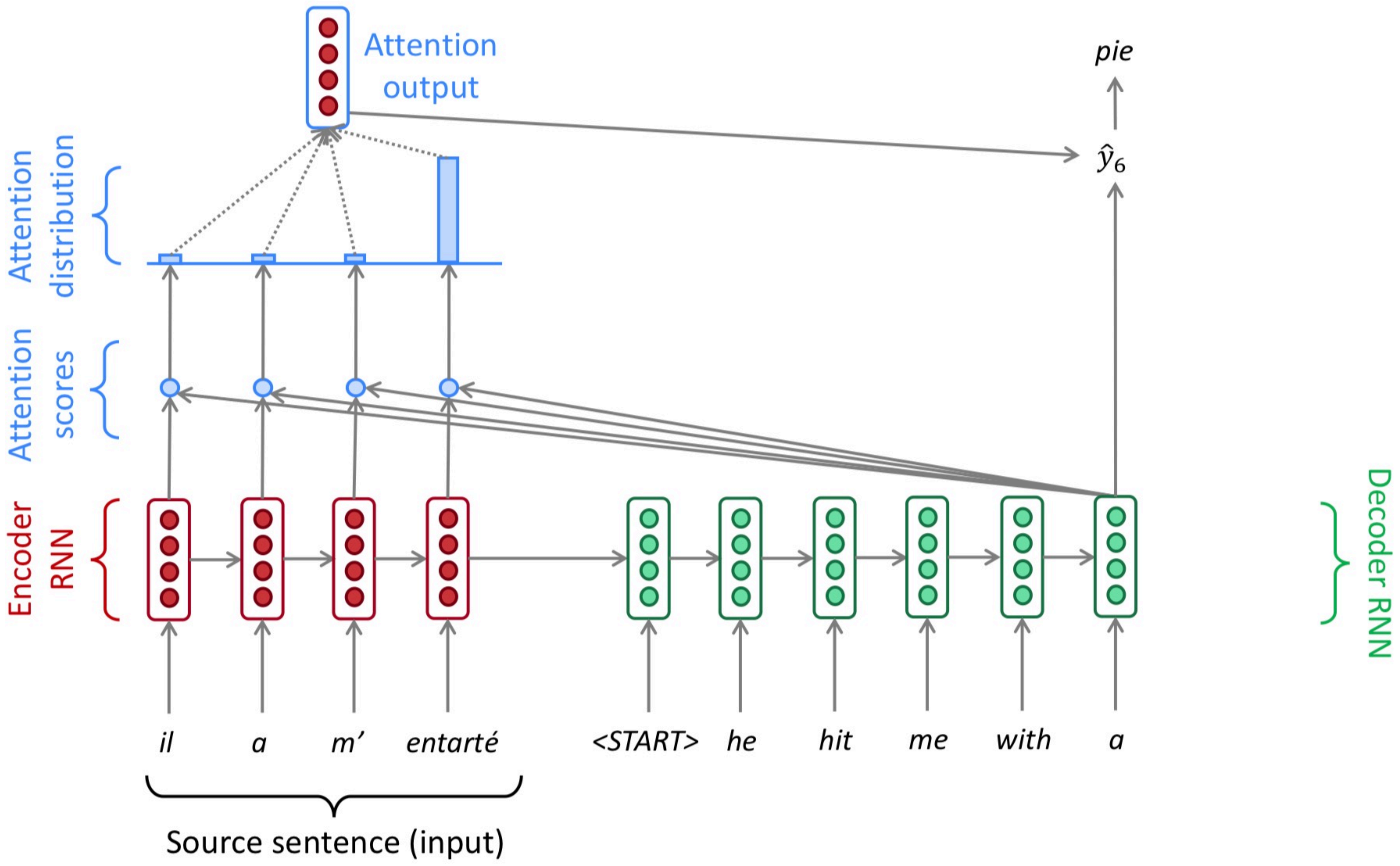


Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Can also use $\hat{y}_1$ as input for next time step

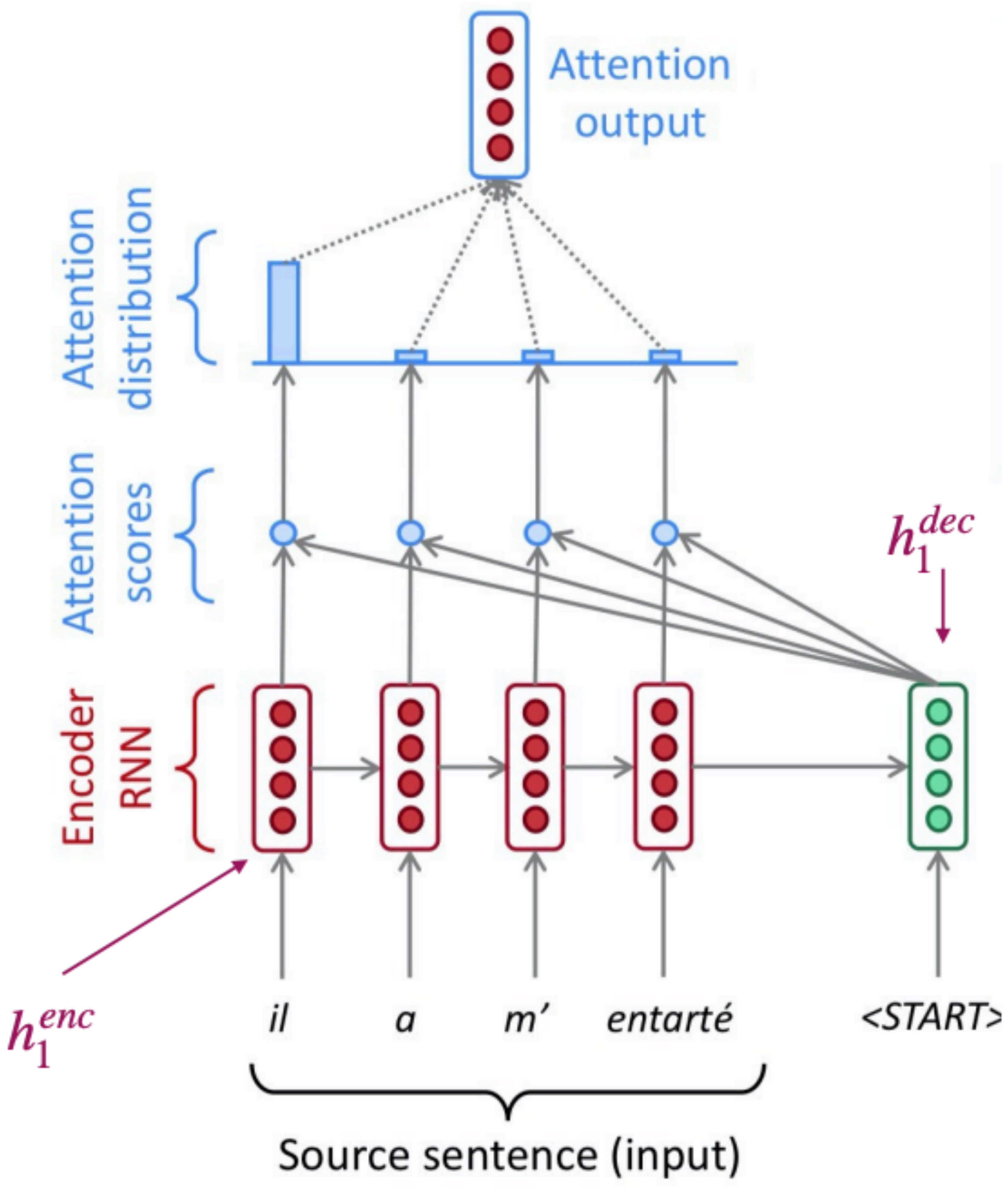
(slide credit: Abigail See)

Seq2seq with attention



(slide credit: Abigail See)

Computing attention



- Encoder hidden states: $h_1^{enc}, \ldots, h_n^{enc}$
- Decoder hidden state at time t: h_t^{dec}
- First, get attention scores for this time step (we will see what g is soon!):

$$e^t = [g(h_1^{enc}, h_t^{dec}), \ldots, g(h_n^{enc}, h_t^{dec})]$$

- Obtain the attention distribution using softmax:

$$\alpha^t = \text{softmax}\,(e^t) \in \mathbb{R}^n$$

- Compute weighted sum of encoder hidden states:

$$a_t = \sum_{i=1}^{n} \alpha_i^t h_i^{enc} \in \mathbb{R}^h$$

- Finally, concatenate with decoder state and pass on to output layer:
  $[a_t; h_t^{dec}] \in \mathbb{R}^{2h}$

Types of attention

- Assume encoder hidden states $h_1, h_2, \ldots, h_n$ and decoder hidden state z

1. **Dot-product attention**:

$$g(h_i, z) = z^T h_i \in \mathbb{R}$$

Simplest (no extra parameters)

requires z and h_i to be same size

more efficient (matrix multiplication)

2. **Bilinear / multiplicative attention:**

$$g(h_i, z) = z^T W h_i \in \mathbb{R}$$, where W is a weight matrix

More flexible than dot-product (W is trainable)

3. **Additive attention (essentially MLP):**

$$g(h_i, z) = v^T \tanh\left(W_1 h_i + W_2 z\right) \in \mathbb{R}$$

where W_1, W_2 are weight matrices and v is a weight vector

Perform better for larger dimensions

Attention can be applied to other modalities

Attention on other modalities

- Images

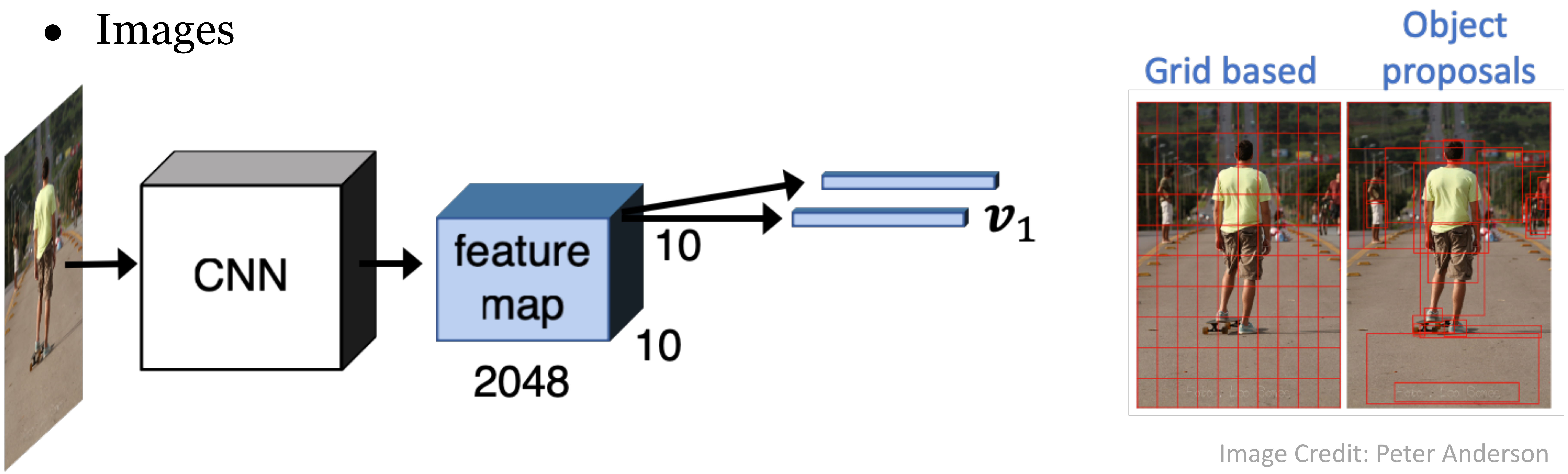
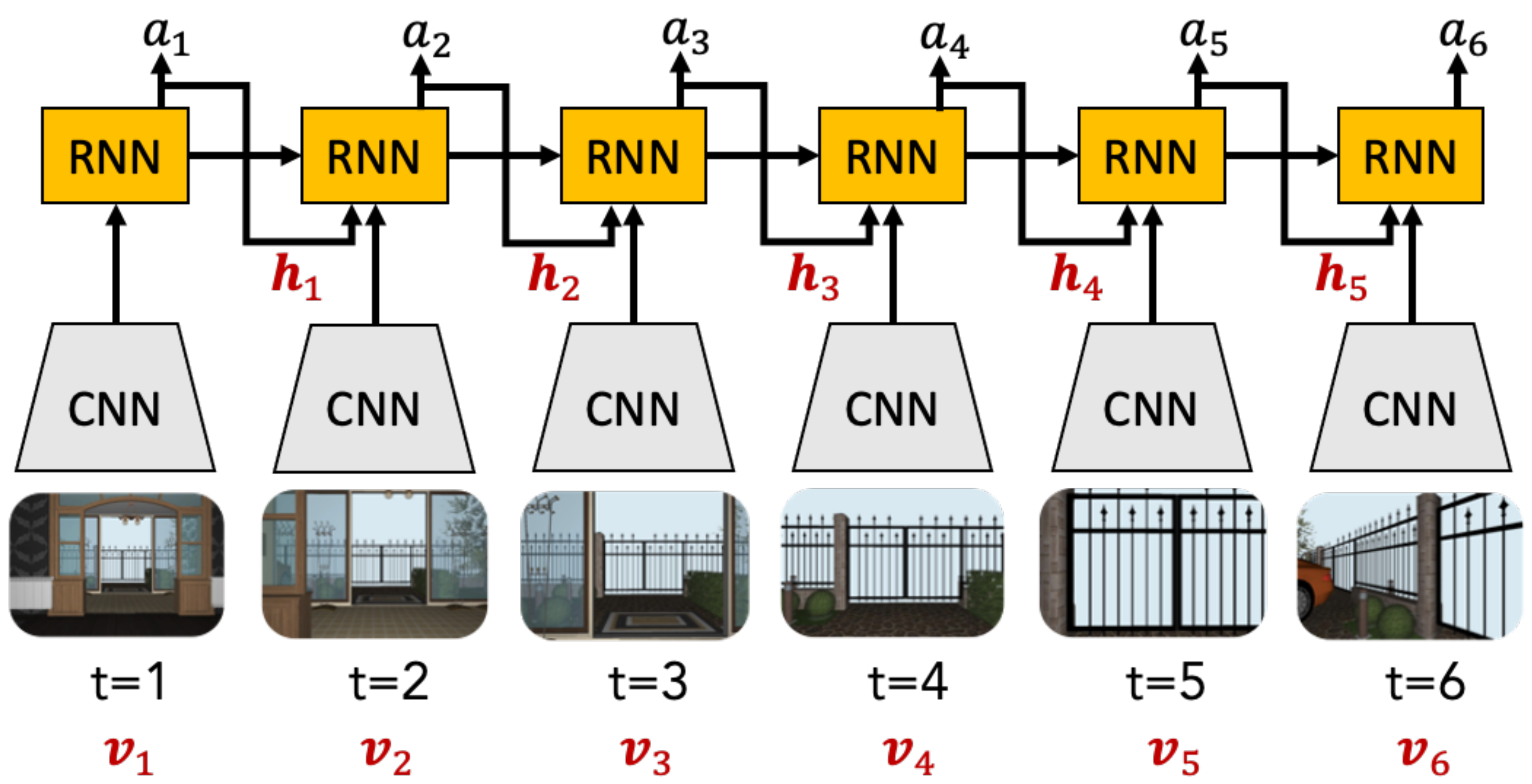


Image Credit: Peter Anderson

- Agent experience

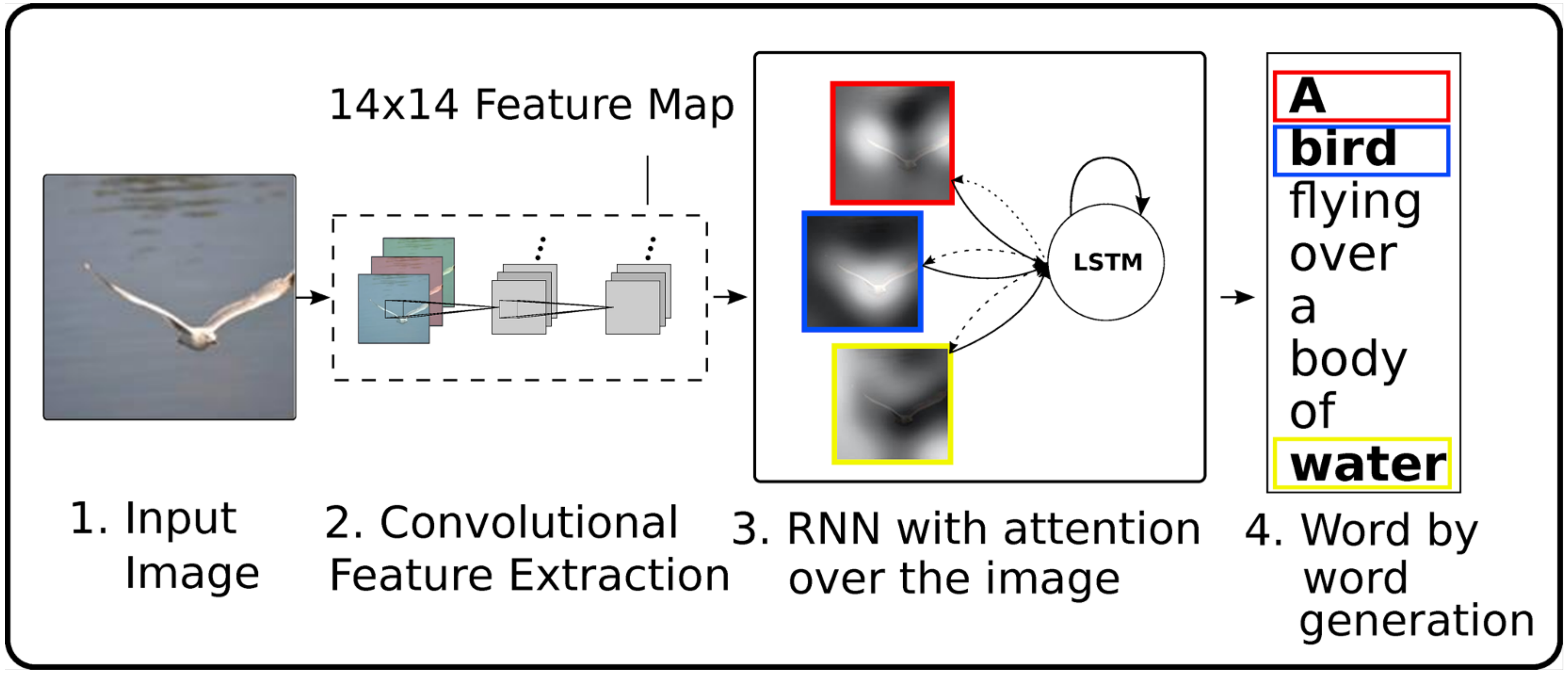


$C = \{h_1, \ldots, h_5\}$

or

$C = \{v_1, \ldots, v_6\}$

Image captioning example



Xu et al. ICML 2015

Different types of attention

Soft vs Hard Attention

- Soft: Each attention candidate is weighted by α_i

$$\hat{\boldsymbol{v}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

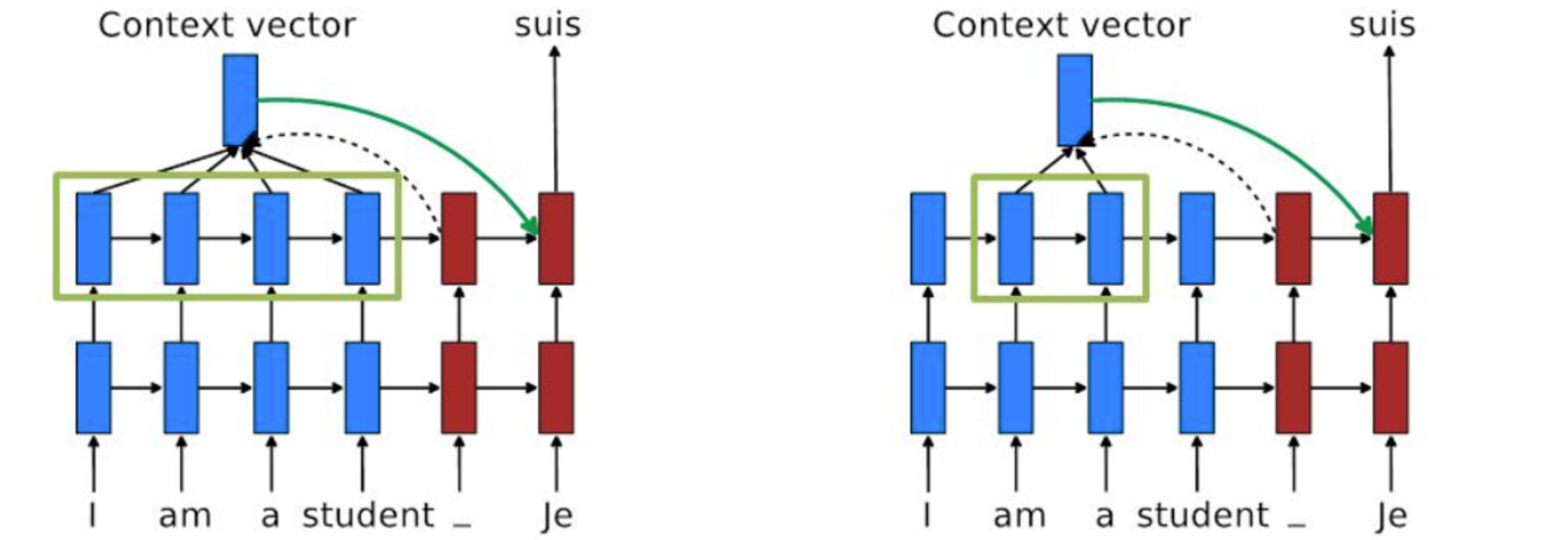
  - Easy to train (smooth and differentiable)
  - But can be expensive over large input

- Hard: Use α_i as a sample probability to pick *one* attention candidate as input to subsequent layers
  - Trainable with REINFORCE approaches (Xu et al. ICML 2015), or Gumbel-Softmax (Jang et al. ICLR 2017)

Xu et al. ICML 2015

Global vs Local Attention

- Global: attention over the entire input
- Local: attention over a window (or subset) of the input

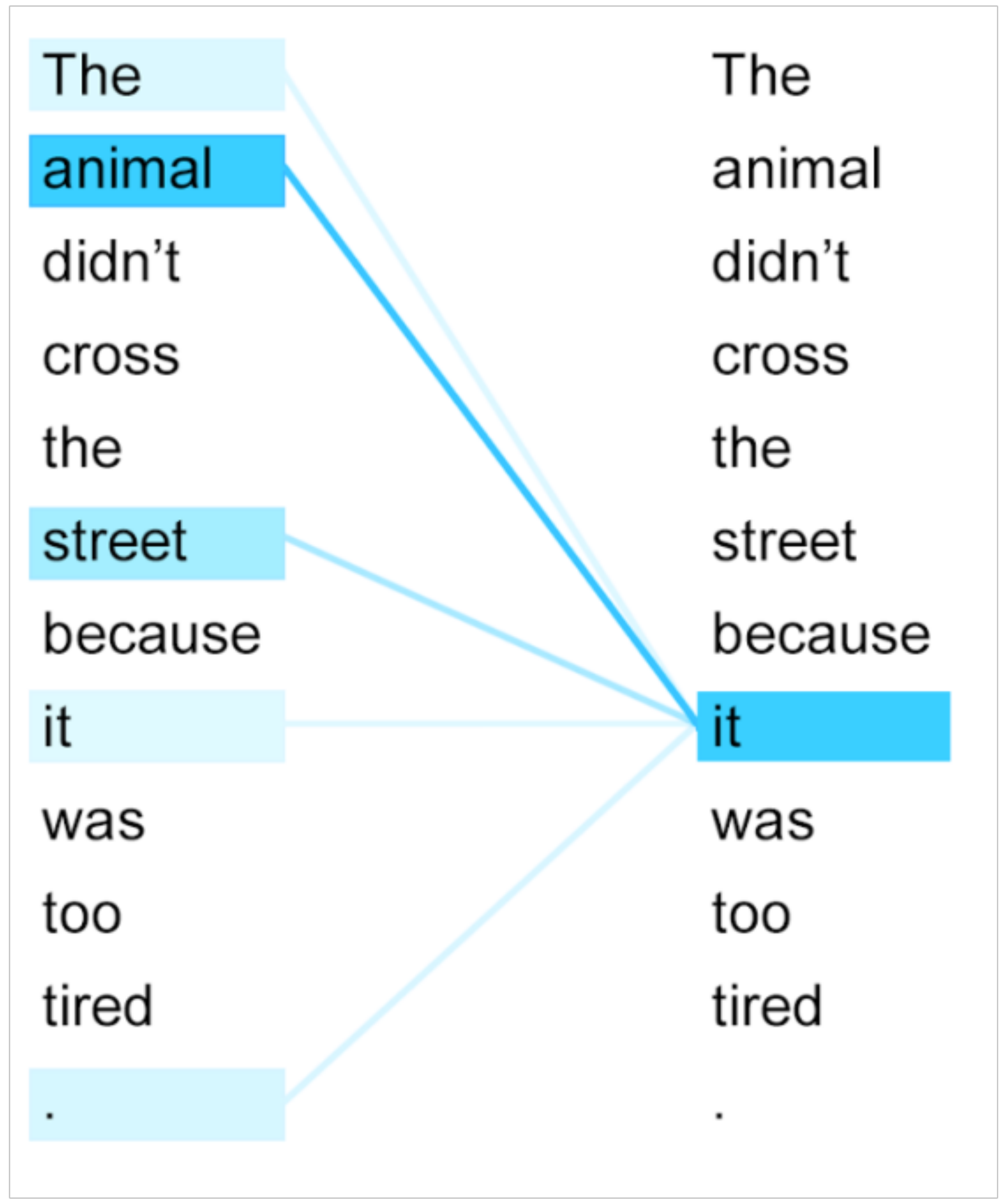


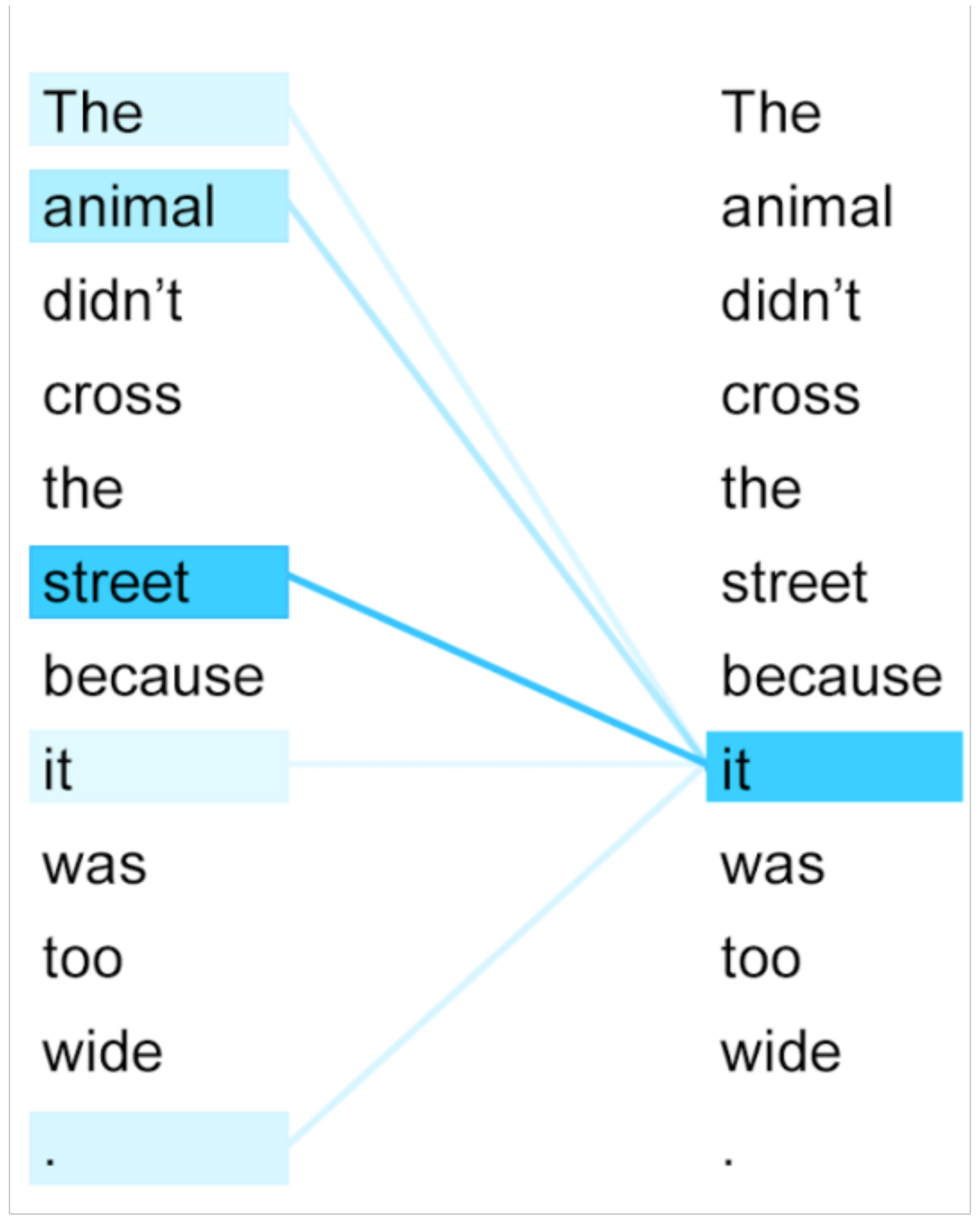
Global: ***all*** source states. Local: ***subset*** of source states.

Luong et al, 2015

Self-Attention

- Attention (correlation) with different parts of itself

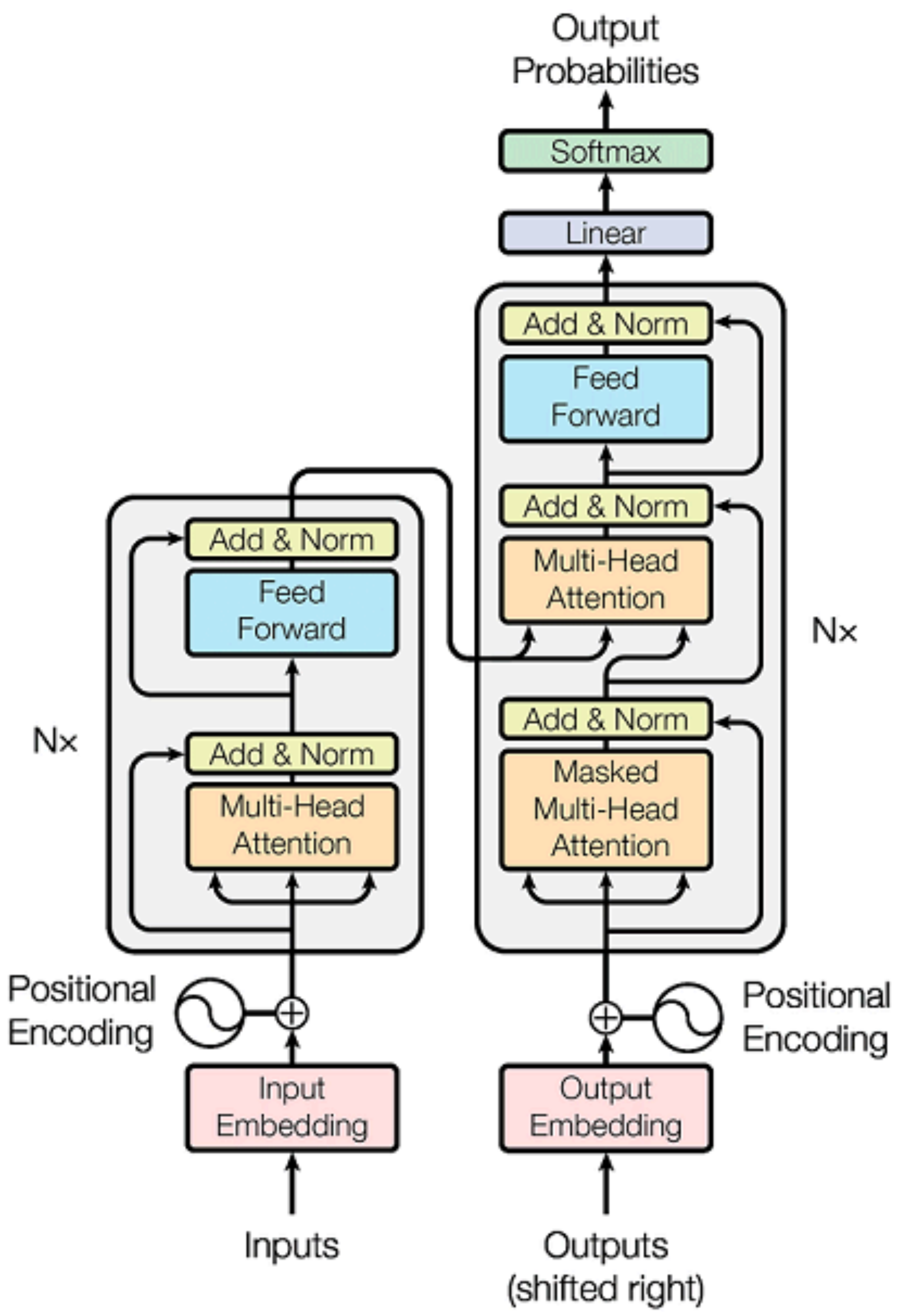




https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

- Transformers: modules with scaled dot-product self-attention

Transformers: self-attention



- More recent models (e.g. Transformer, Vaswani et al., 2017) have replaced RNNs entirely with attention mechanisms
- Theoretically limiting (since recurrence can help handle arbitrarily long sequences)
- Huge gains in practical performance