SFU NatLangLab

CMPT 413/713: Natural Language Processing

Sequence to Sequence Models (Seq2Seq)

Spring 2025
2025-02-05

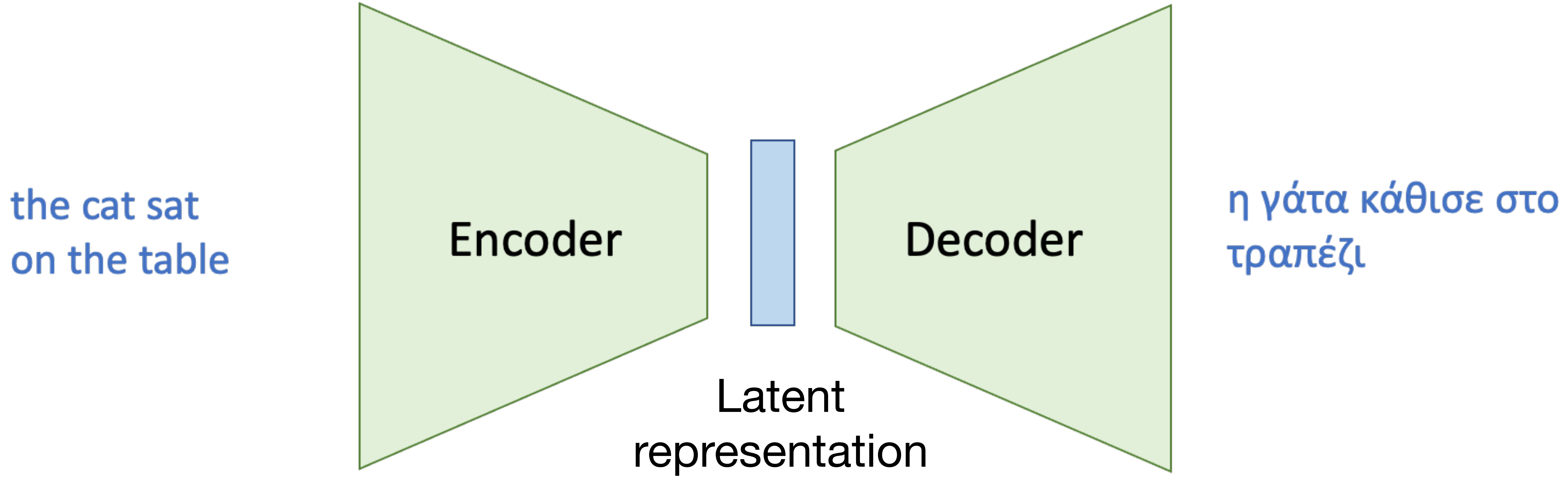
Adapted from slides from Danqi Chen and Karthik Narasimhan
(with some content from slides from Abigail See, Graham Neubig)

Overview

- Sequence generation tasks
- Seq2Seq models - Encoder/Decoder
- Decoding strategies
- Evaluating text generation
- Attention

Sequence Generation

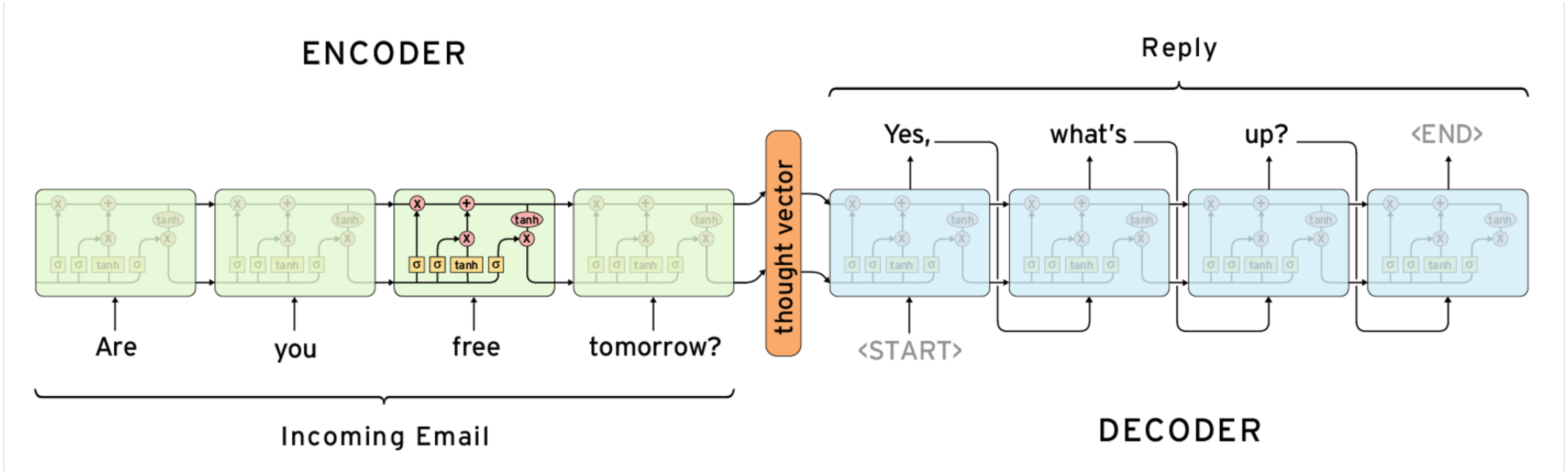
Want computer friendly representation for applications



Understanding what is said
(encoding, parsing, feature extraction)

Deciding what to say
(decoding, generating)

Encoder-Decoder Model



Seq2Seq Tasks and Applications

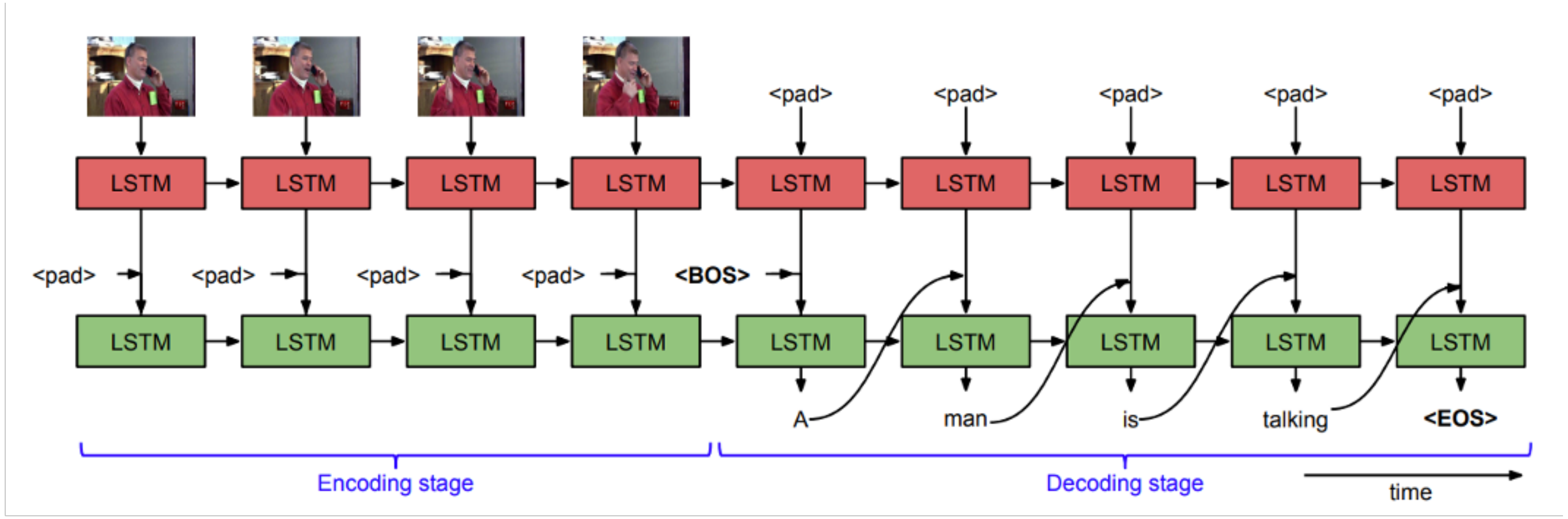
Task/Application	Input	Output
Machine Translation	French	English
Summarization	Document	Short Summary
Dialogue	Utterance	Response
Parsing	Sentence	Parse tree (as sequence)
Question Answering	Context + Question	Answer

Cross-Modal Seq2Seq

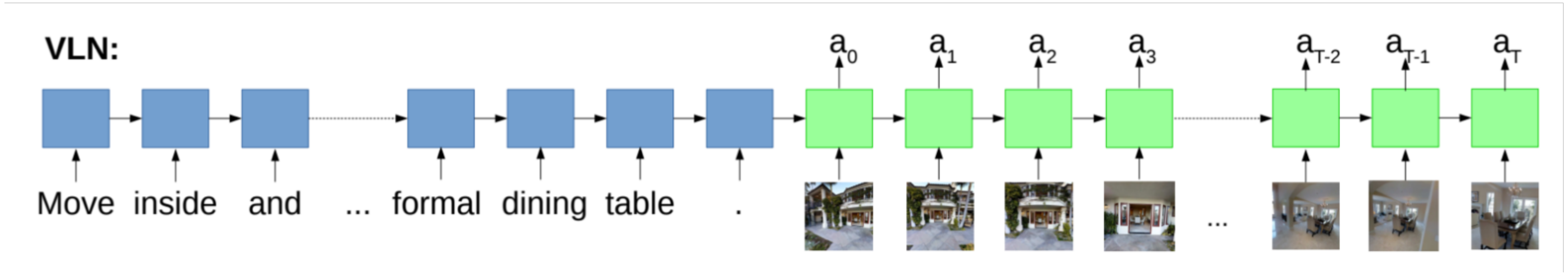
Task/Application	**Input**	**Output**
Speech Recognition	Speech Signal	Transcript
Image Captioning	Image	Text
Video Captioning	Video	Text
Vision-Language Navigation	Text	Actions

Cross-modal sequence generation

- Video captioning (video frames to text)



- Embodied AI (text + frames to actions)



Seq2Seq Tasks and Applications

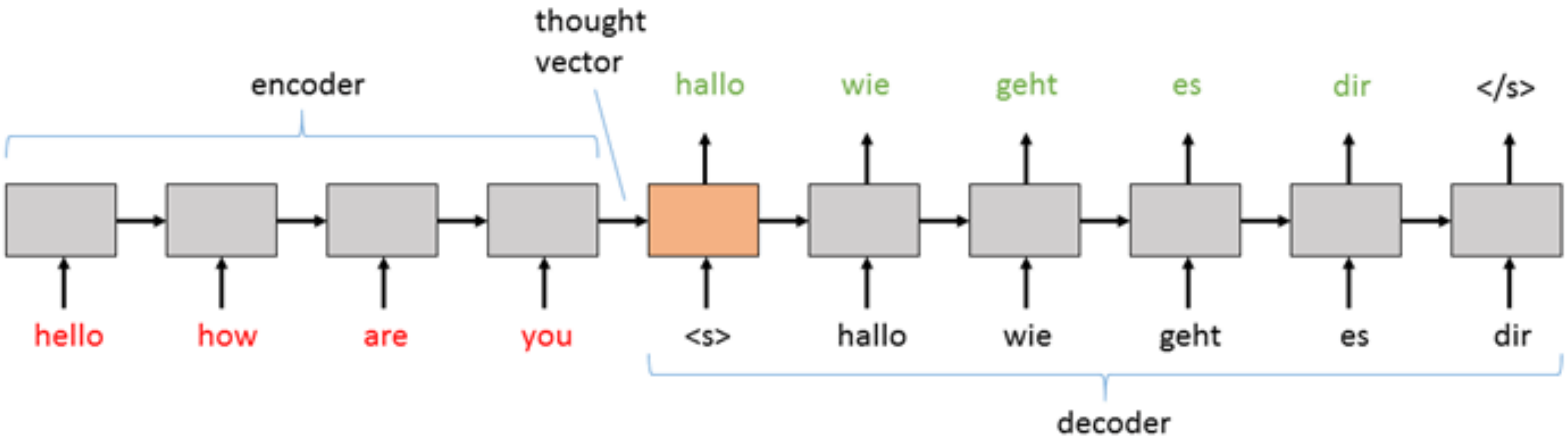
Task/Application	Input	Output
Machine Translation	French	English
Summarization	Document	Short Summary
Dialogue	Utterance	Response
Parsing	Sentence	Parse tree (as sequence)
Question Answering	Context + Question	Answer

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)

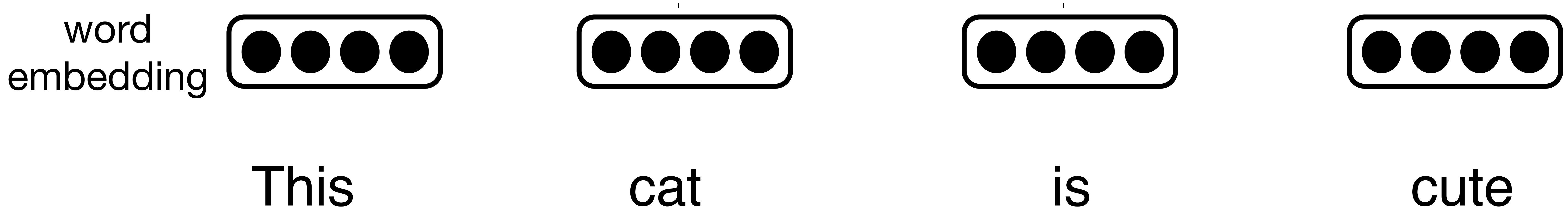


- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

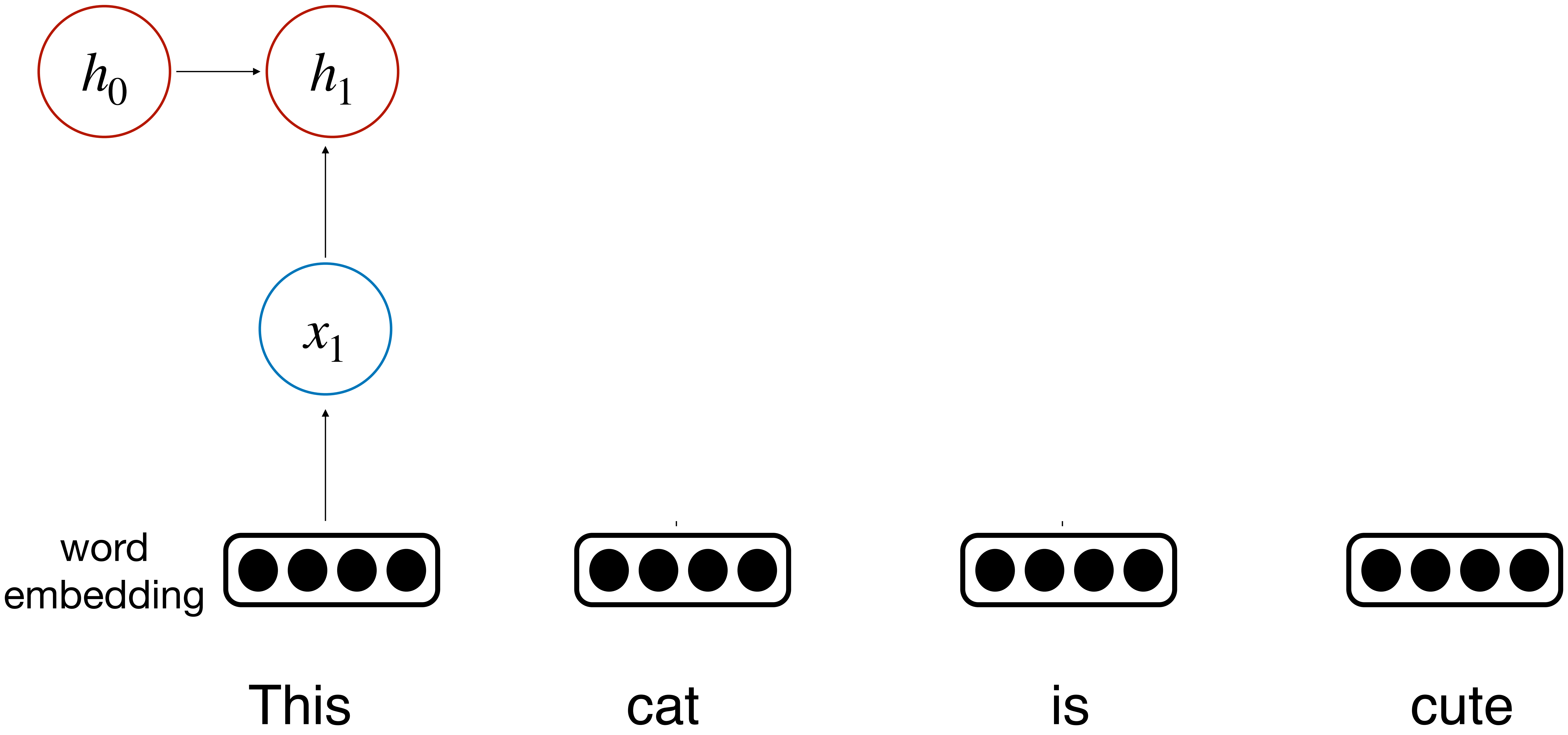
Encoder

Sentence: This cat is cute



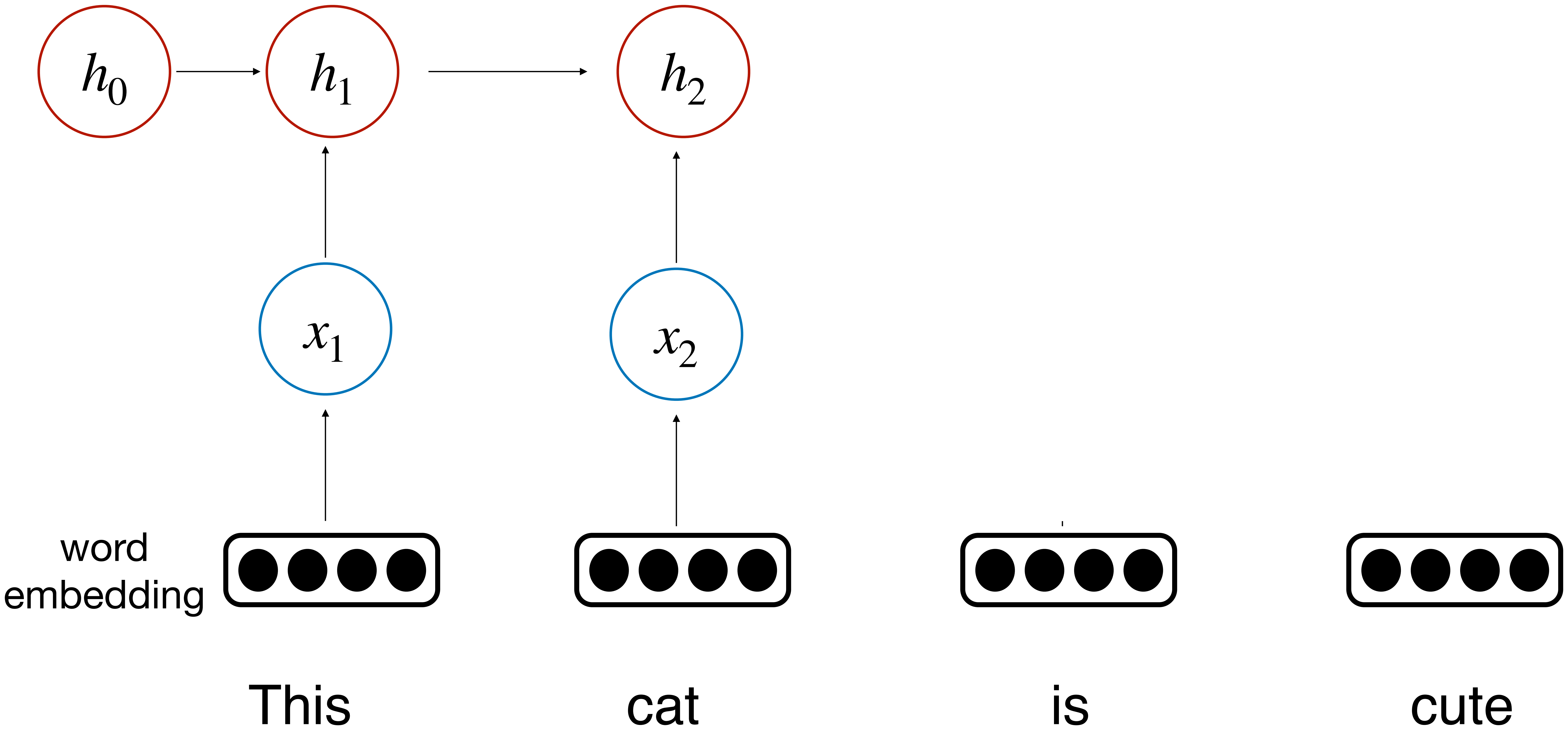
Encoder

Sentence: This cat is cute



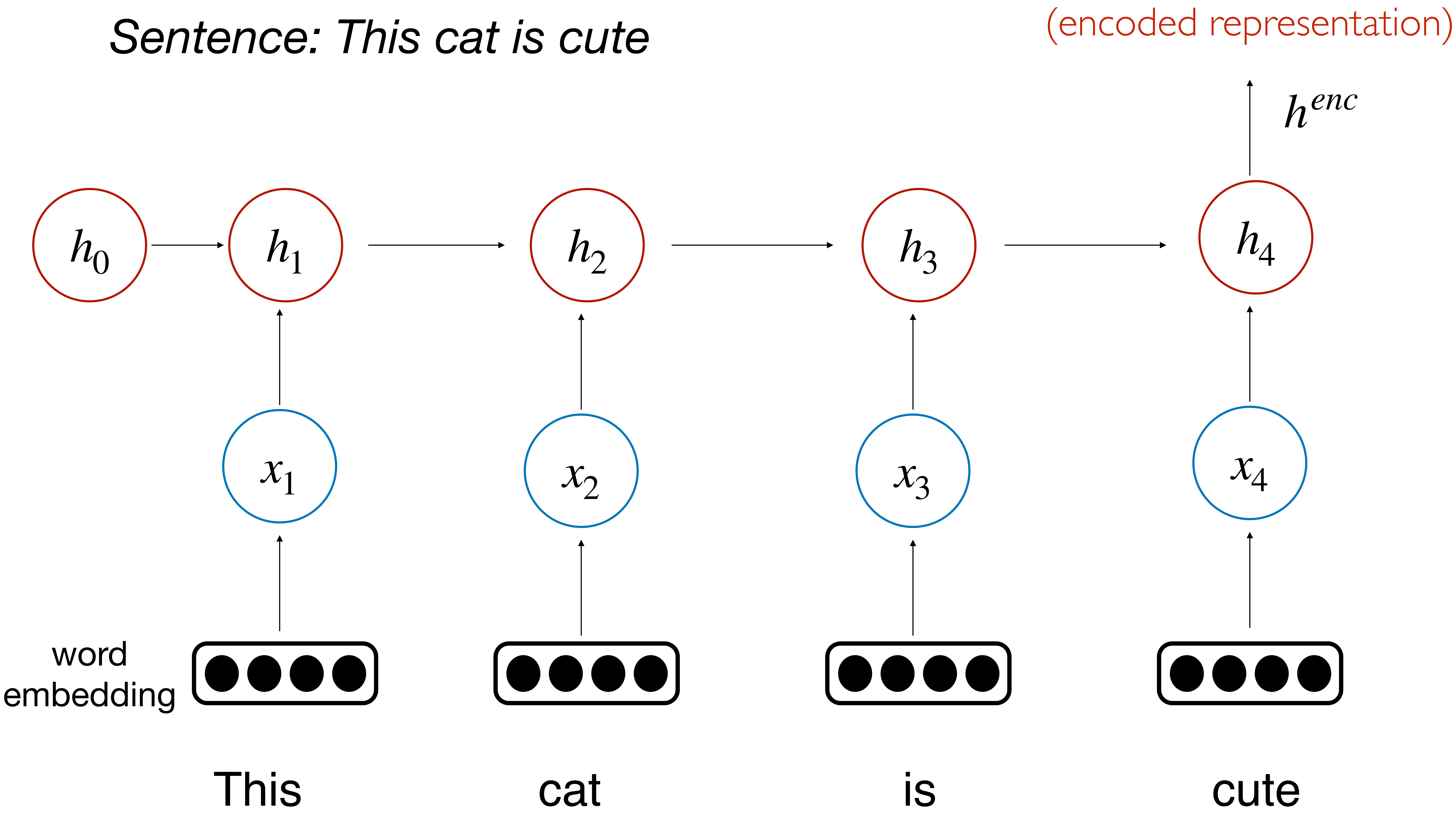
Encoder

Sentence: This cat is cute

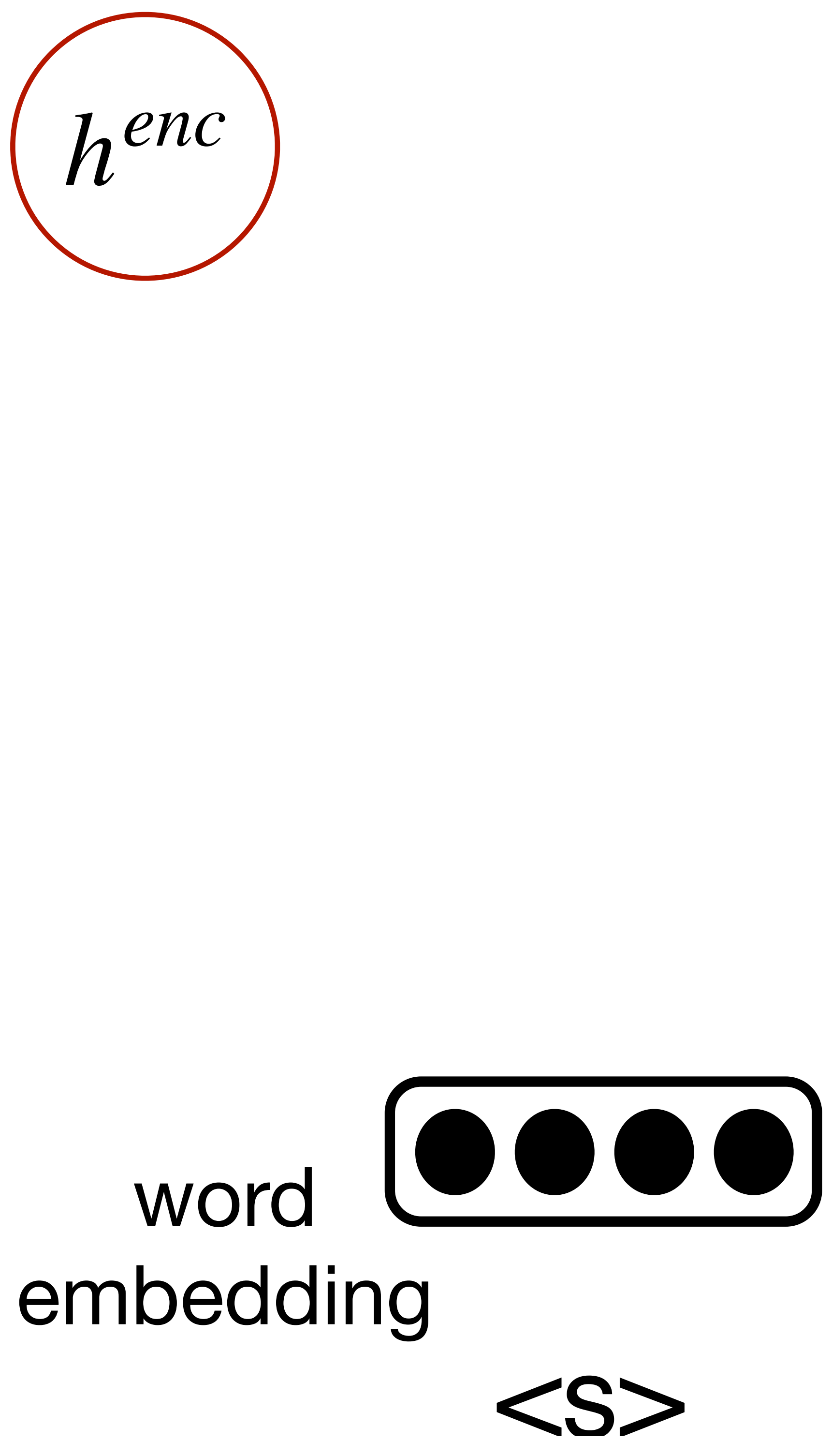


Encoder

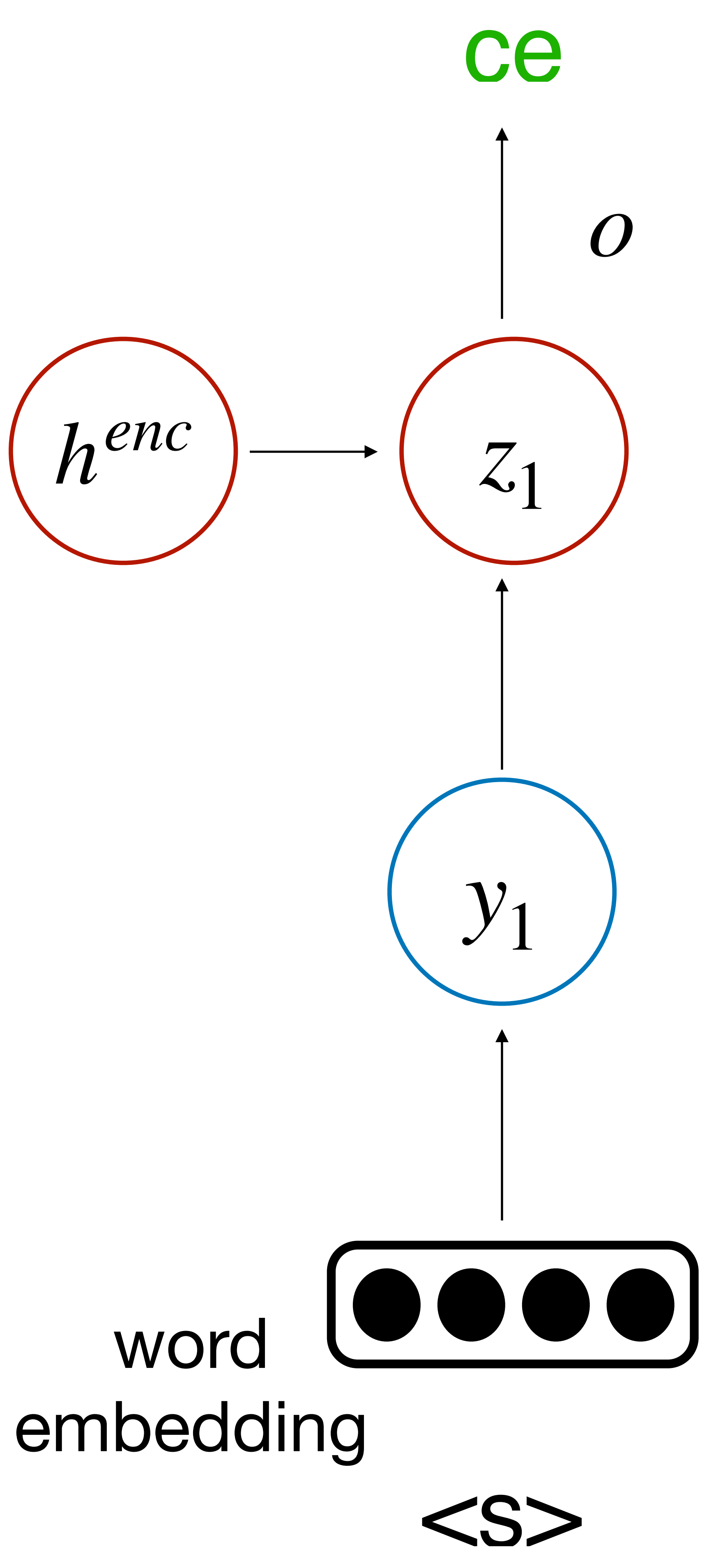
Sentence: This cat is cute



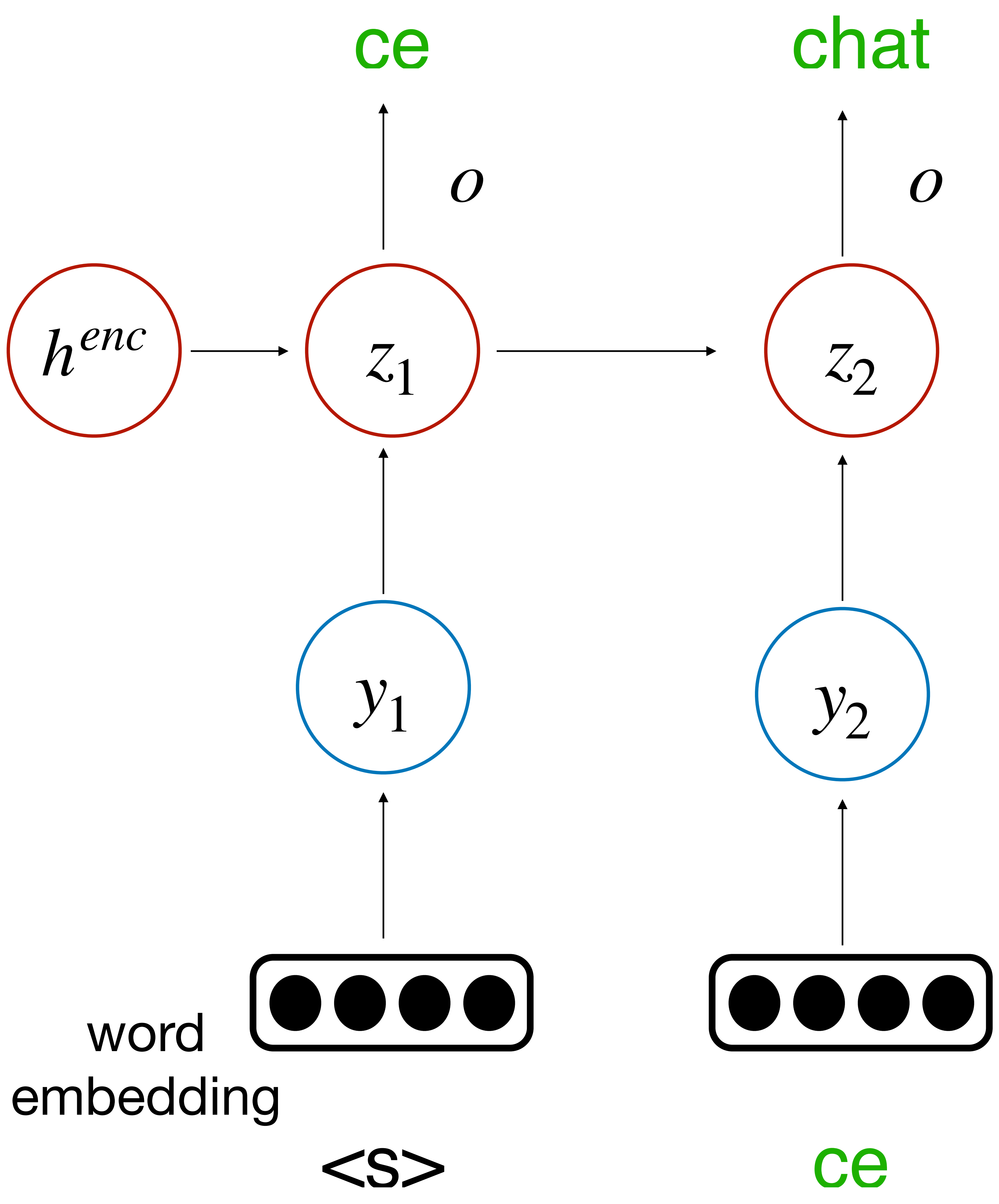
Decoder



Decoder

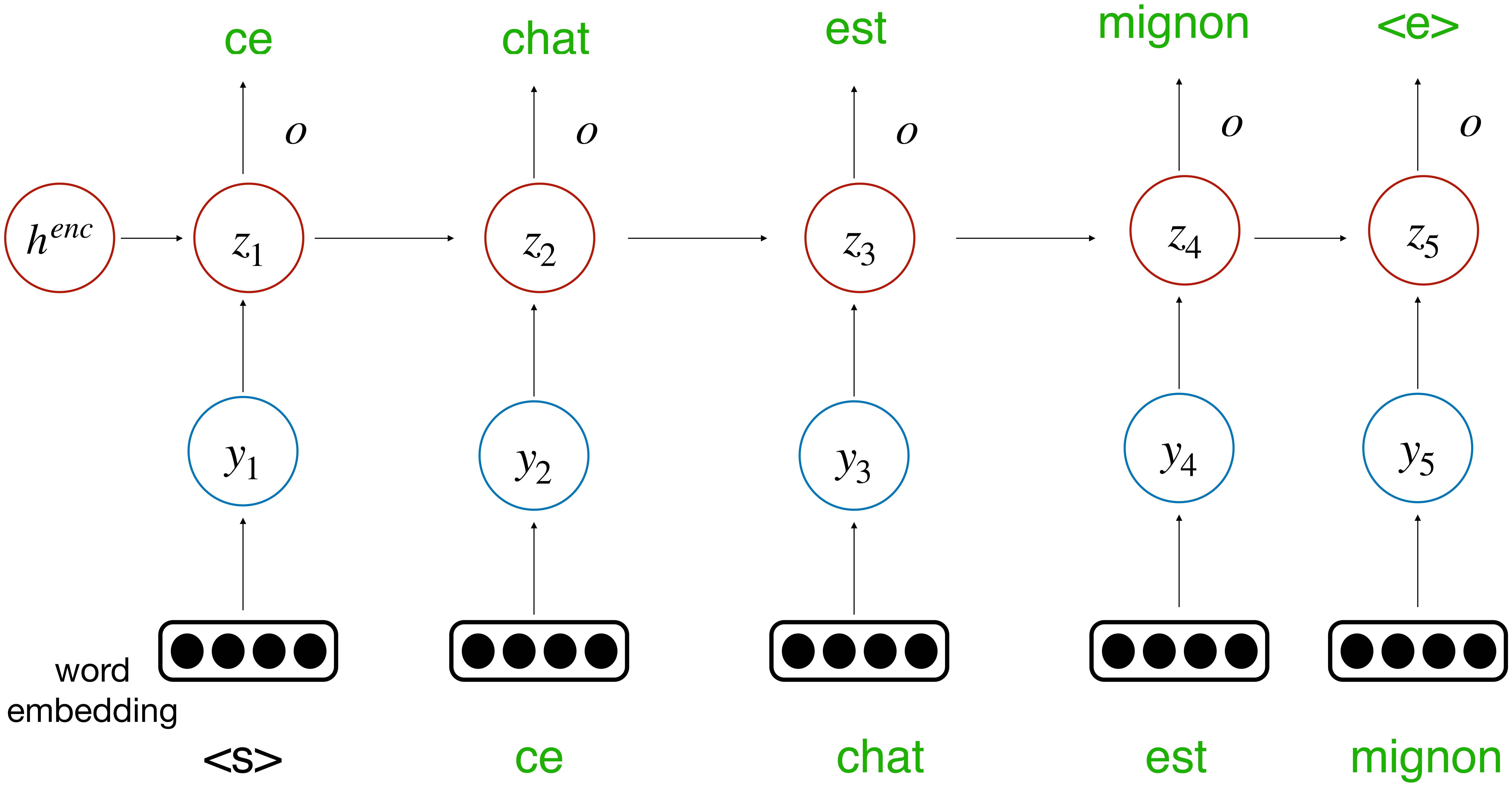


Decoder



Decoder

- A conditioned language model



Seq2seq training

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

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

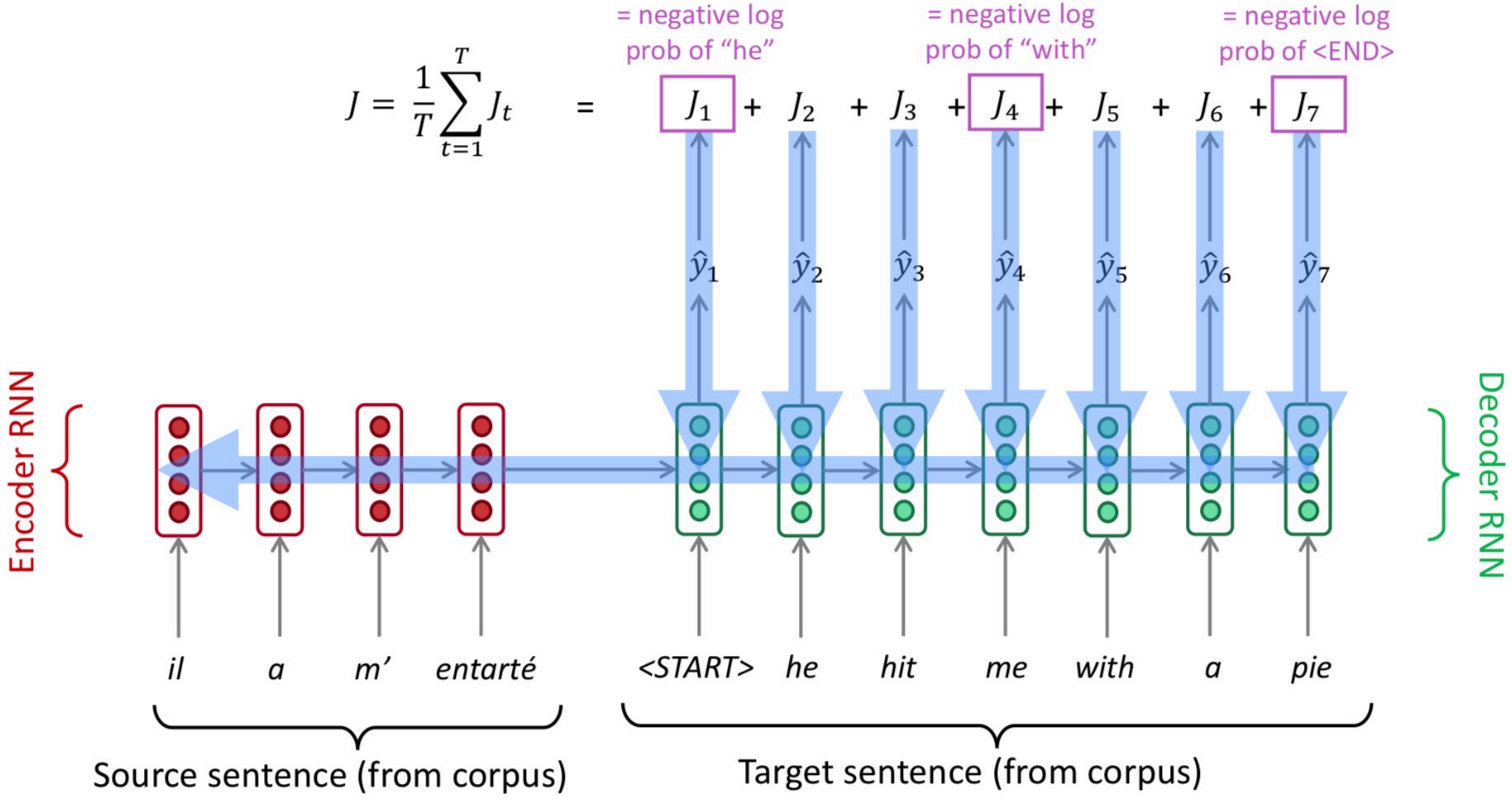
- Back-propagate gradients through *both decoder and encoder*
- Need a really big corpus

36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

Seq2seq training



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

(slide credit: Abigail See)

Efficient Training: Batching

- Apply RNNs to batches of sequences
- Present data as 3D tensor of $(T \times B \times F)$
- Use mask matrix to aid with computations that ignore padded zeros

Padded sequences

1	1	1	1	0	0
1	0	0	0	0	0
1	1	1	1	1	1
1	1	1	0	0	0

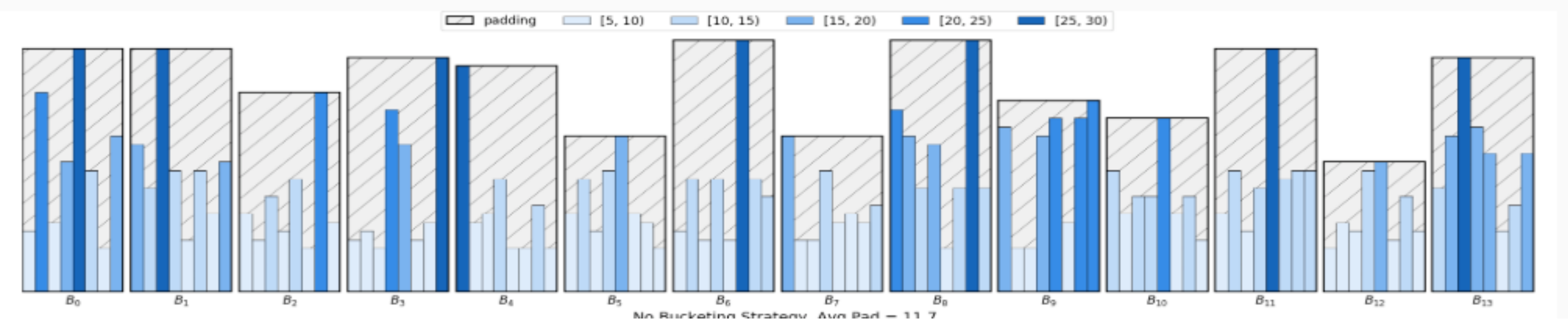
Lengths

4
1
6
3

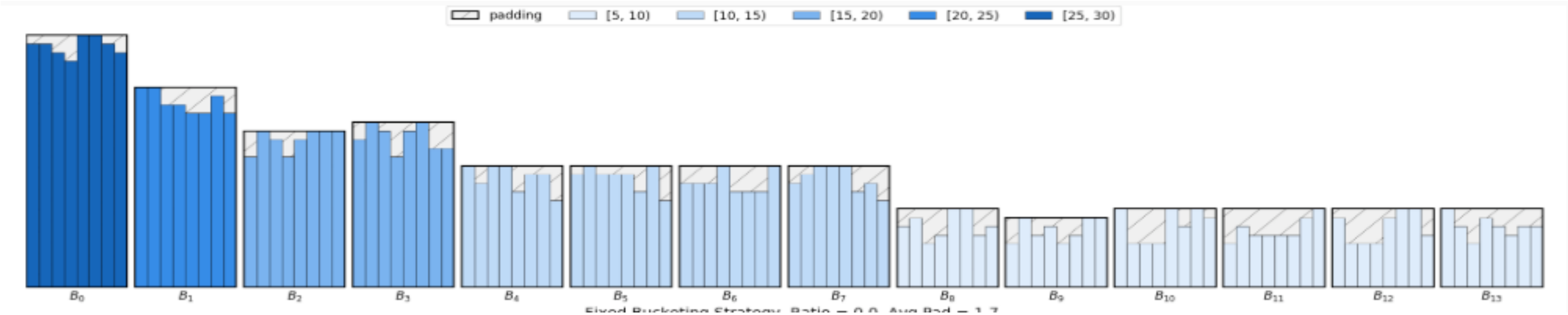
Batching

- Sorting (partially) can help to create more efficient mini-batches
- However, the input is less randomized

Unsorted



Sorted



Decoding strategies

Generation

How can we use our model (decoder) to generate sentences?

- Sampling: Try to generate a *random* sentence according the the probability distribution

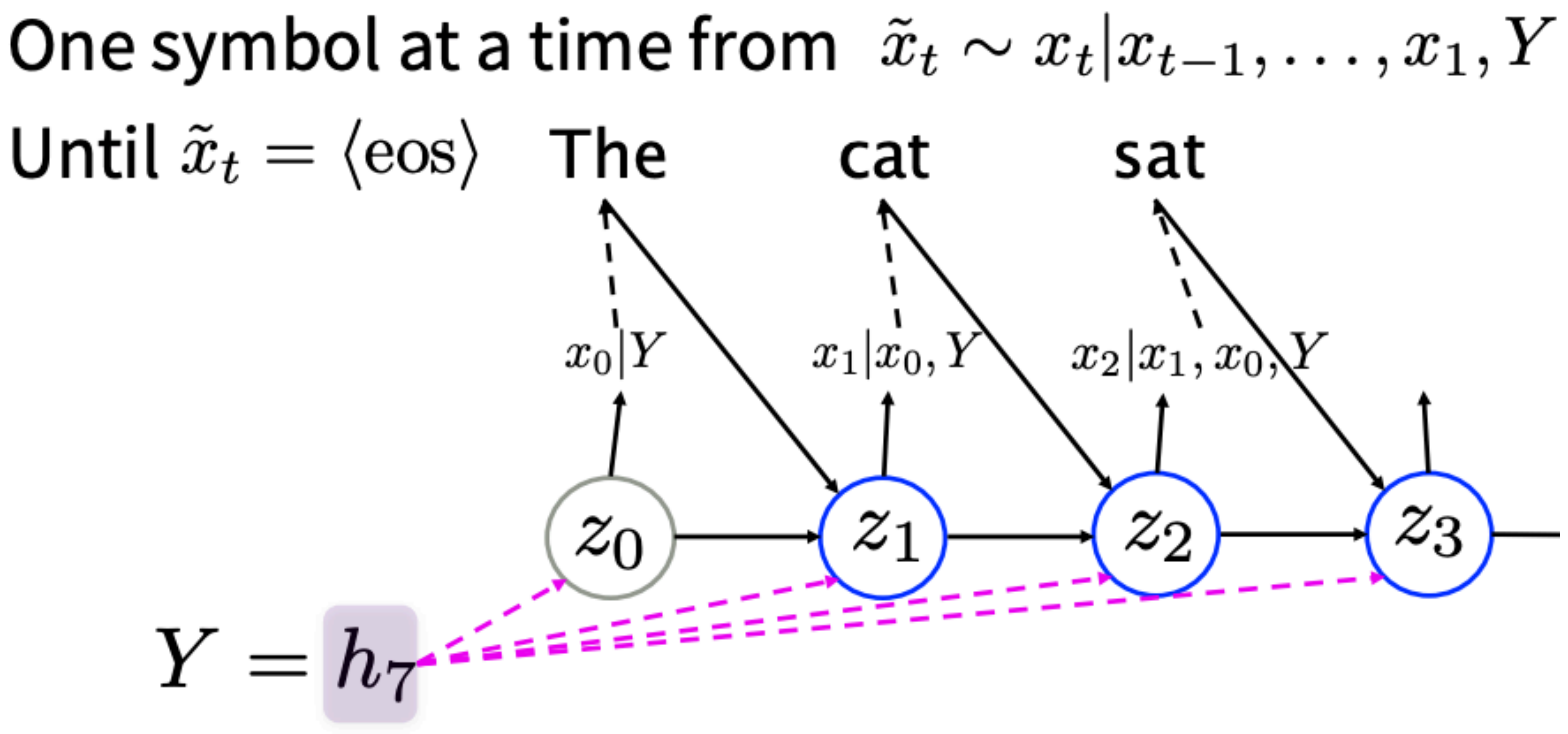
- Argmax: Try to generate the *best* sentence, the sentence with the *highest* probability

Decoding Strategies

- Ancestral sampling
- Greedy decoding
- Exhaustive search
- Beam search

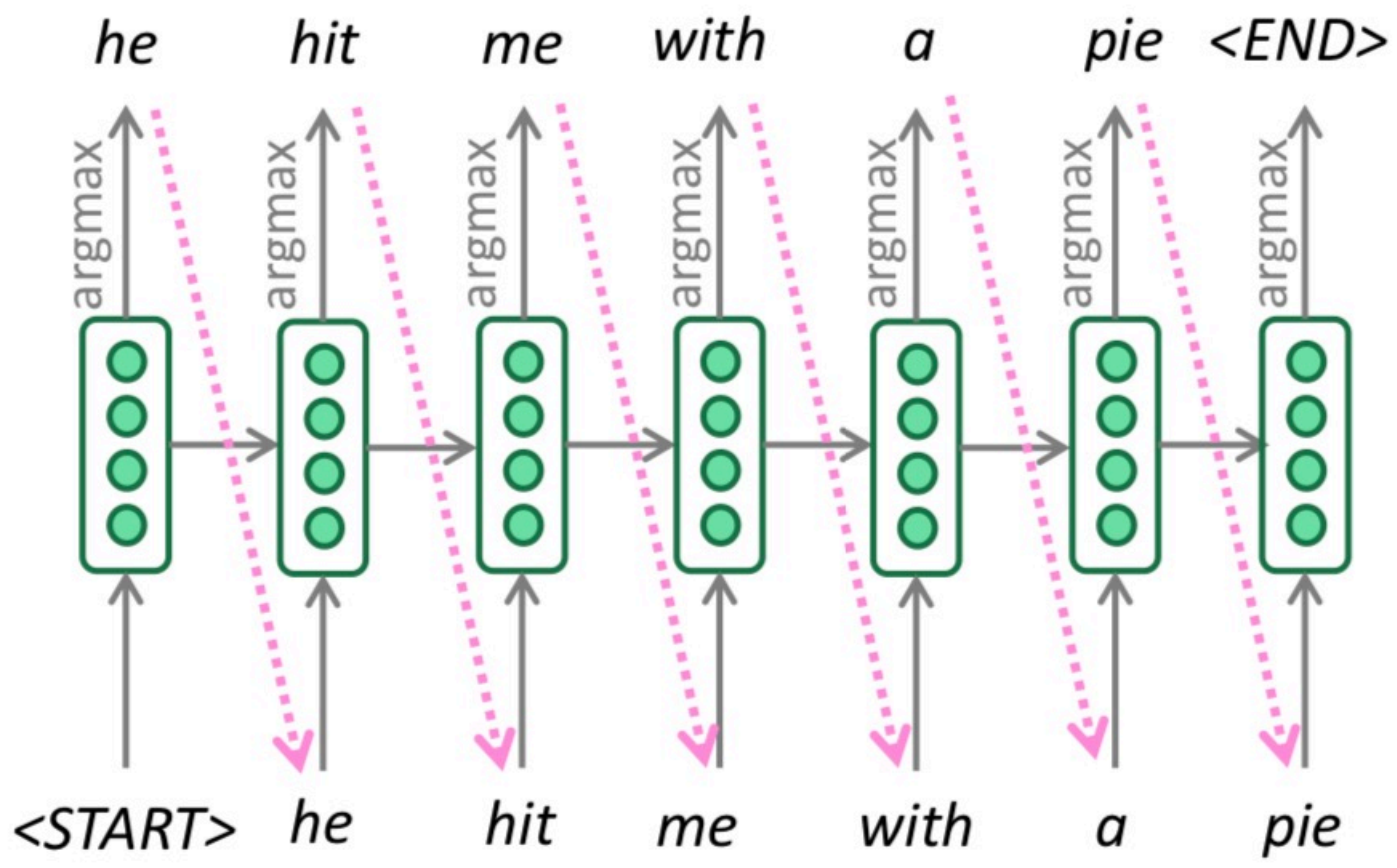
Ancestral Sampling

- Randomly sample words one by one
- Provides diverse output (high variance)



(figure credit: Luong, Cho, and Manning)

Greedy decoding



- Compute argmax at every step of decoder to generate word
- What's wrong?

Exhaustive search?

- Find $\arg\max_{y_1,\ldots,y_T} P(y_1, \ldots, y_T | x_1, \ldots, x_n)$
- Requires computing all possible sequences
  - $O(V^T)$ complexity!
  - Too expensive

Recall: Beam search (a middle ground)

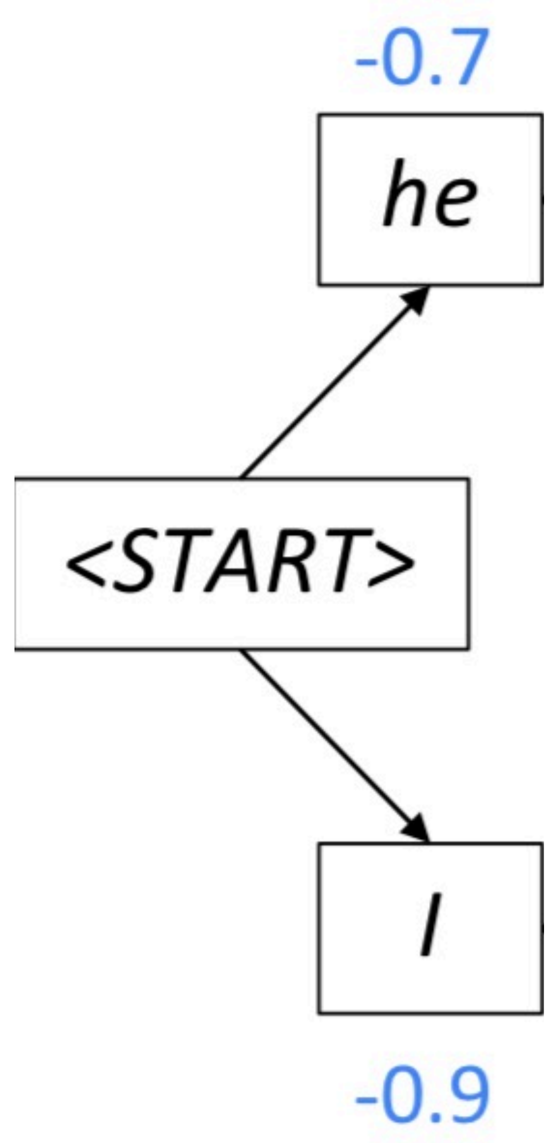
- **Key idea:** At every step, keep track of the k most probable partial translations (hypotheses)
- Score of each hypothesis = log probability

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

- Not guaranteed to be optimal
- More efficient than exhaustive search

Beam decoding

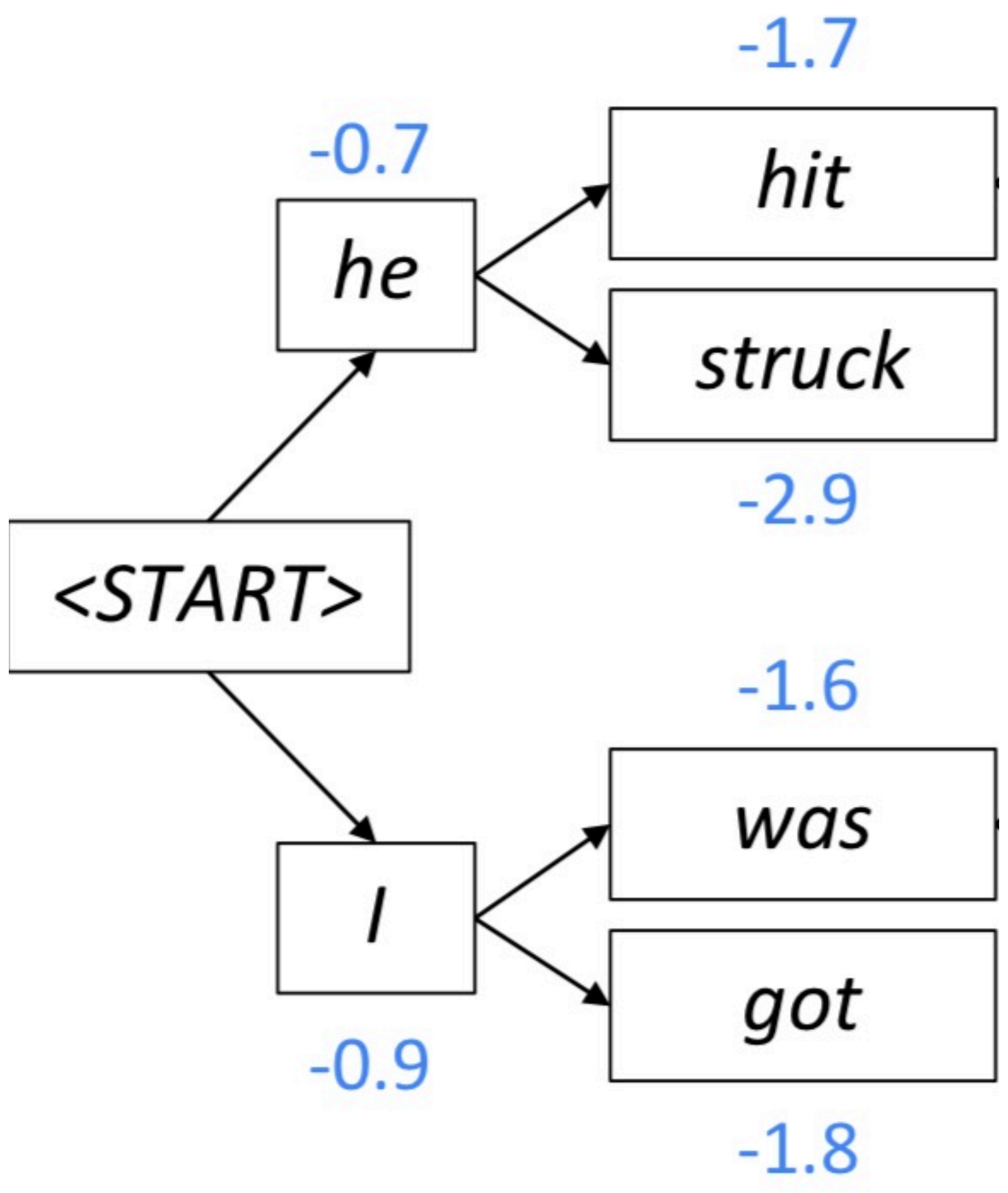
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

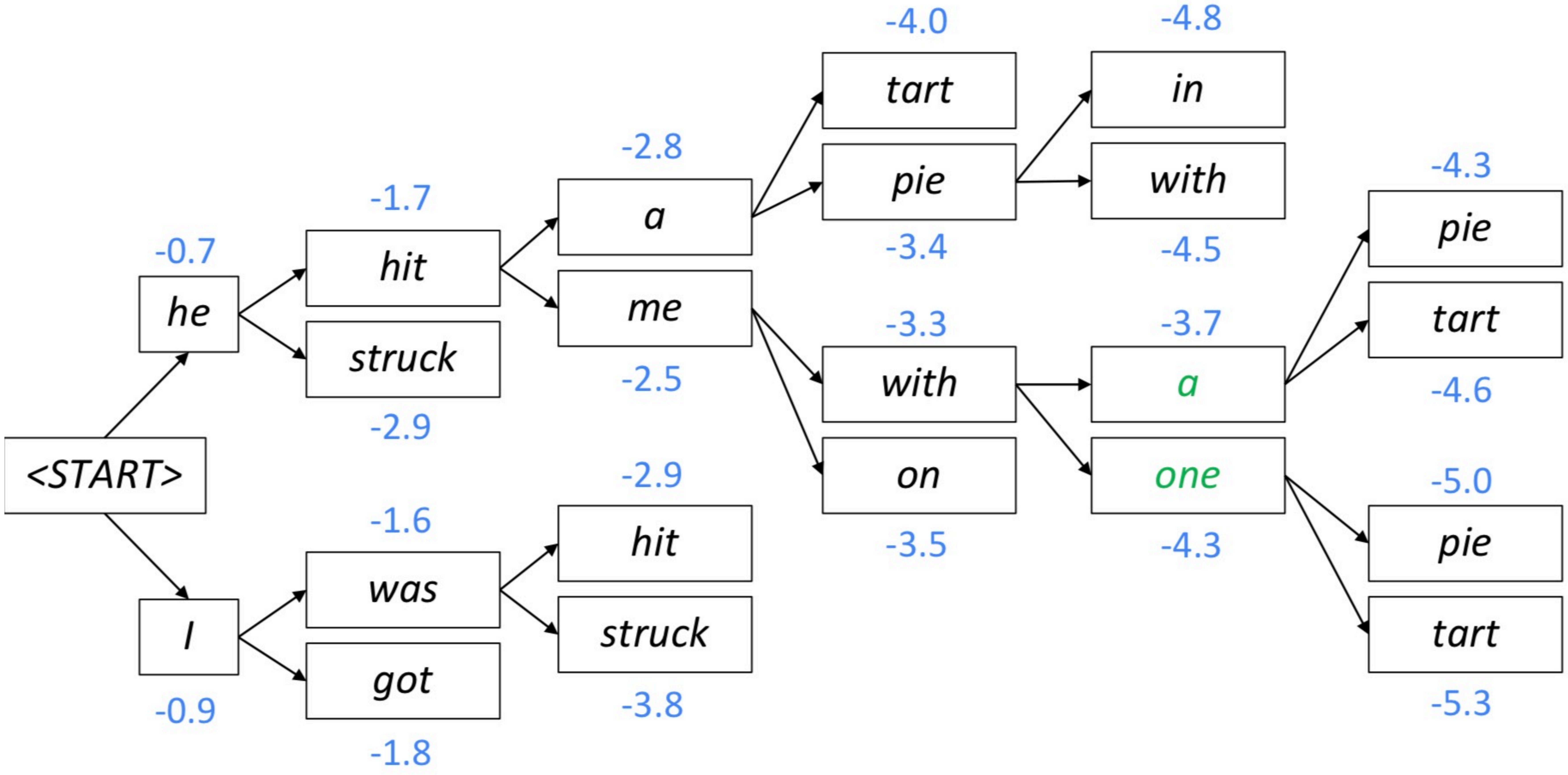
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

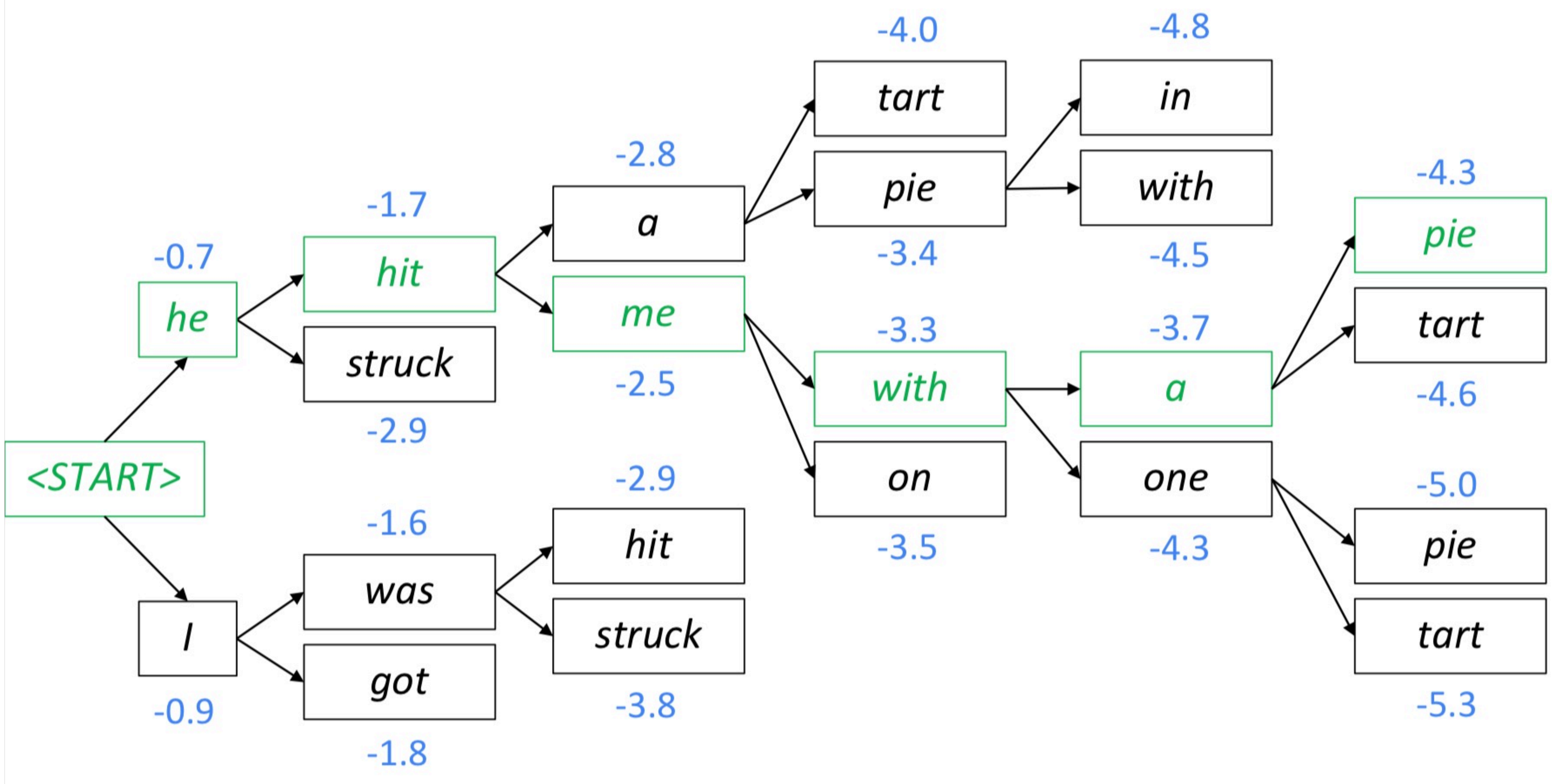
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Backtrack

Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

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

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

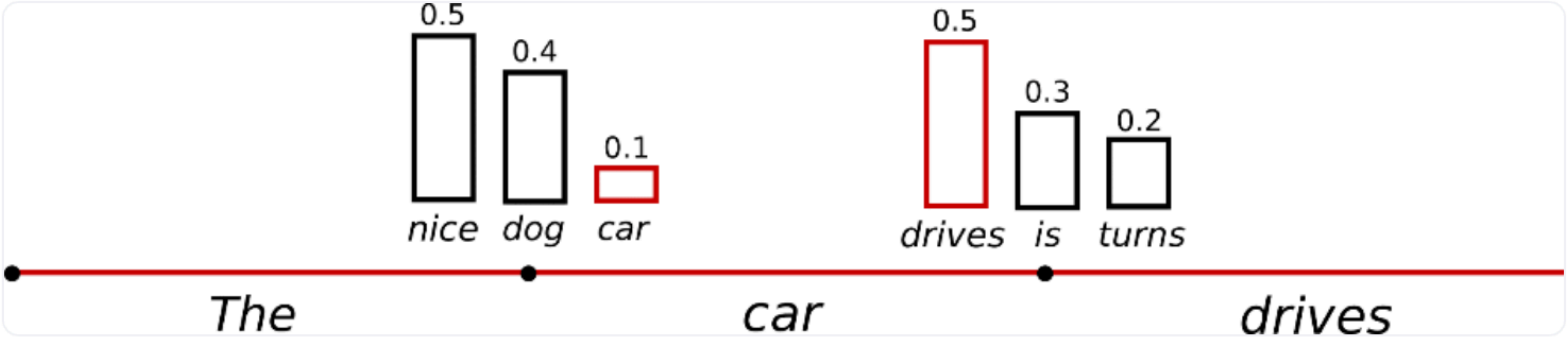
  - Otherwise shorter hypotheses have higher scores

Beam Search Pitfalls

- Beam search can still be very repetitive.
  - Heuristic is to penalize repeated n-grams in the output.
  - Manually set the probability of next words that could create an already seen n-gram to 0
  - n should be greater than 2 or 3
- The choices in beam search may not be very diverse.
  - Similar continuations can happen due to common sub-trees in different branches
- These issues are referred to as **model degeneration**

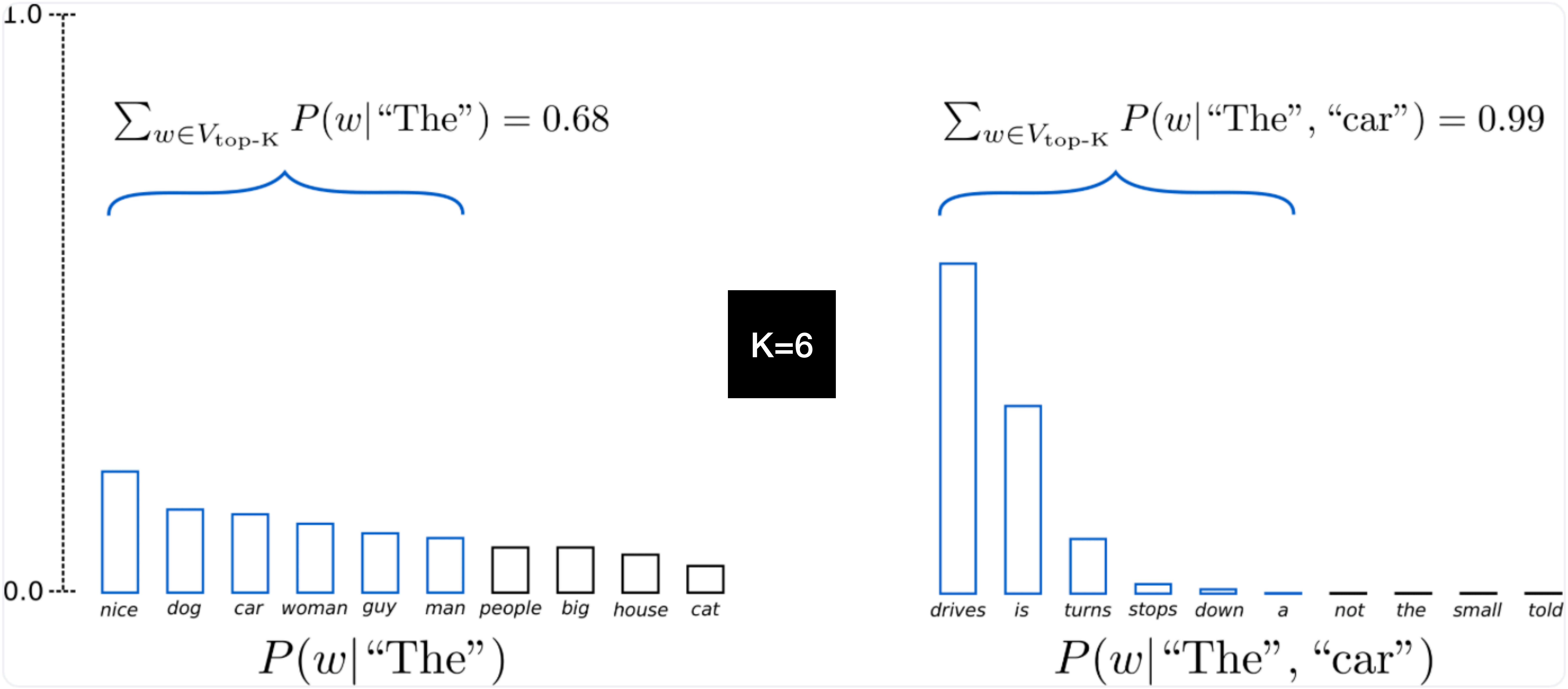
Sampling

- Sampling is represented by the operator $\sim$
- We pick the next word $w_t \sim P(w \mid w_{1:t-1}) = \dfrac{\exp(logits(w \mid w_{1:t-1}))}{\sum_{w'} \exp(logits(w' \mid w_{1:t-1}))}$
- Generation is no longer *deterministic.*
- Sampling can generate gibberish. Solution: use temperature $\dfrac{\exp(logits(w \mid w_{1:t-1})/T)}{\sum_{w'} \exp(logits(w' \mid w_{1:t-1})/T)}$



Top-k Sampling

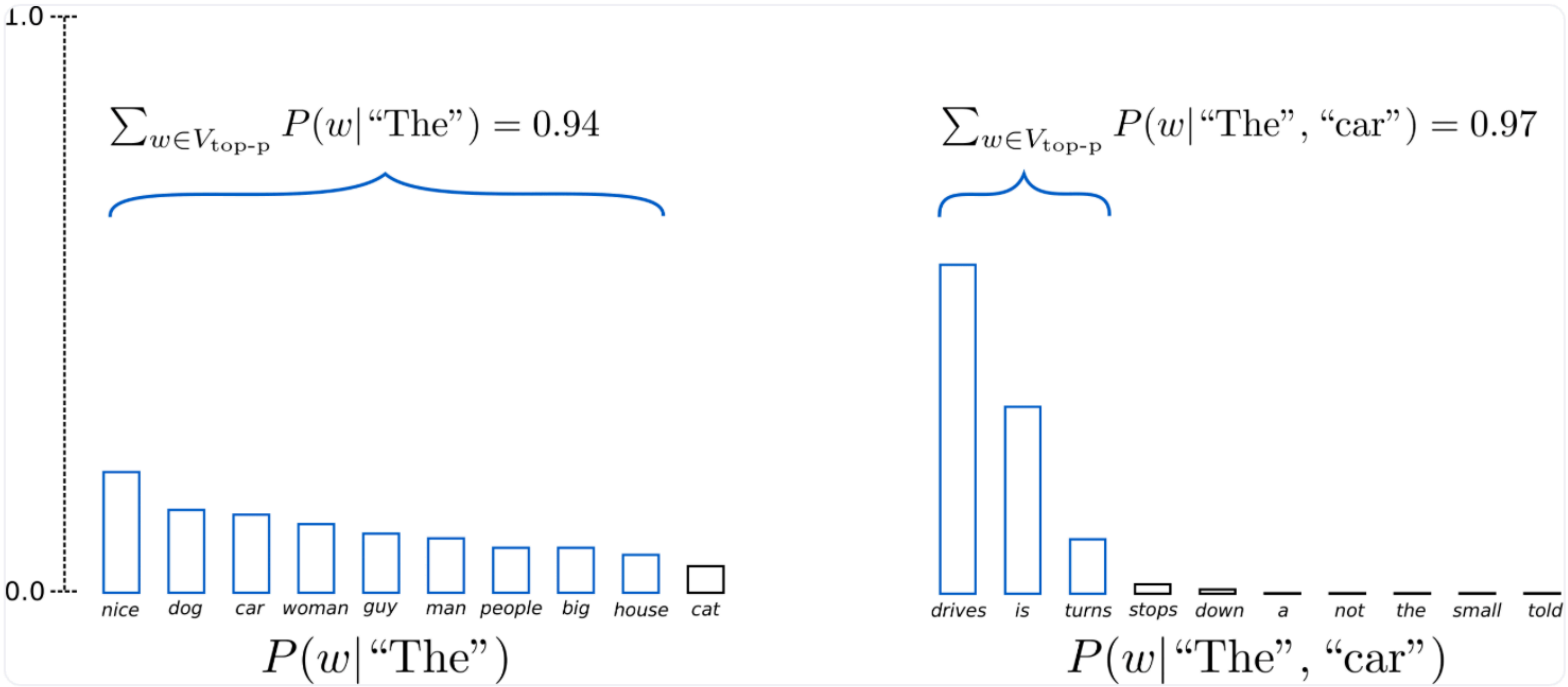
- K most likely next words are filtered and we re-normalize over the K words
- GPT2 showed that this worked better than beam search



Top-p Nucleus Sampling

- Choose the smallest set of words whose cumulative probability exceeds a threshold probability p. The probability mass is redistributed among this set of words.

- The size of the set being sampled from grows and shrinks depending on the probability distribution.



Summary of sampling for text generation

Repetitive

A: Where are you going?
B: I'm going to the restroom.
A: See you later.
B: See you later.
A: See you later.
B: See you later.

Sample and Rank

1. Sample N candidate
2. Rank candidate and select best one

Sampling

Randomly sample words from distribution at each time step t

- **Basic/pure sampling**: sample from $P_t(w)$ directly
  - Can get some very bad samples
  - No control
- **Top-n sampling**: sample from P_t truncated to top n words
  - Increase n to get more diverse/risky output
  - Decrease n to get more generic/safe output
- **Top-p (nucleus) sampling**: sample from P_t restricted to top p proportion of words
  - Better when probability distribution is spread
- **Temperature based**:
  - Increase τ to get more diverse/risky output (P_t is more uniform)
  - Decrease τ to get more generic/safe output (P_t is more spiky)

$$P_t(w) = \frac{\exp(s_w/\tau)}{\sum_{w' \in V} \exp(s_{w'}/\tau)}$$

(adapted from slides: Stanford CS224N, Chris Manning)

Evaluating text generation

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*

Evaluation metrics

- Manual evaluation is most accurate, but expensive
- Automated evaluation metrics:
  - Compare system hypothesis with reference translations
  - BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002)
    - Modified n-gram precision

$$p_n = \frac{\text{number of } n\text{-grams appearing in both reference and hypothesis translations}}{\text{number of } n\text{-grams appearing in the hypothesis translation}}$$

BLEU

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

n-gram precision

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

Example

Reference: Vinay likes programming in Python

Hypothesis/Candidate	p_1	p_2	BLEU-2
Vinay likes Python	3/3	1/2	0.7071
To Vinay it like Python	2/5	0	???

https://www.aclweb.org/anthology/P02-1040.pdf

BLEU

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

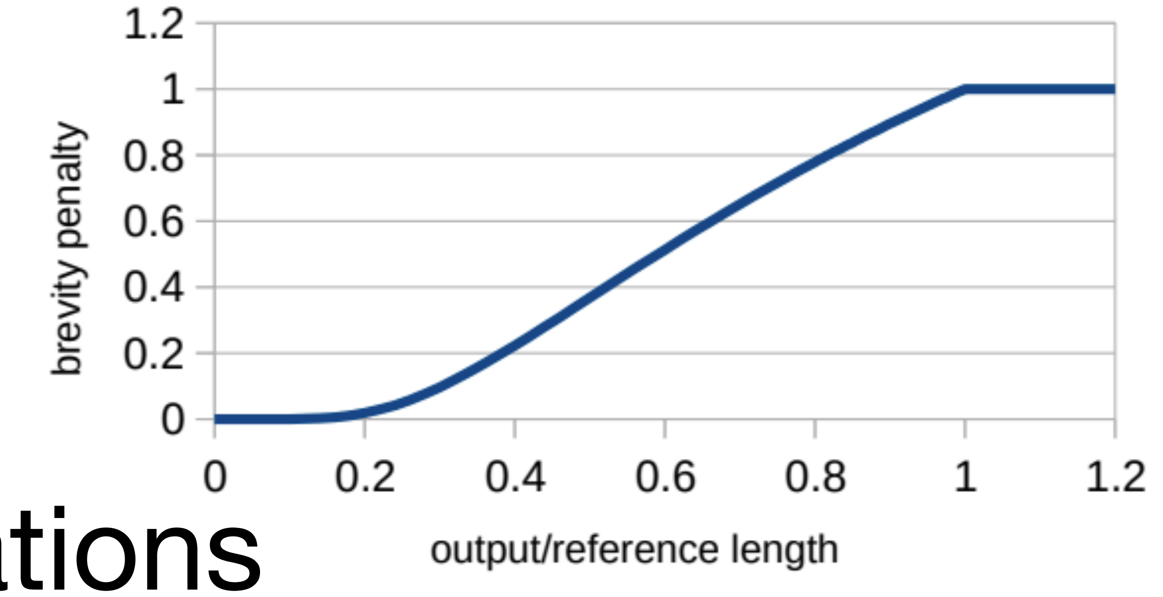
Various smoothing techniques add 1 to numerator/denominator

- 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

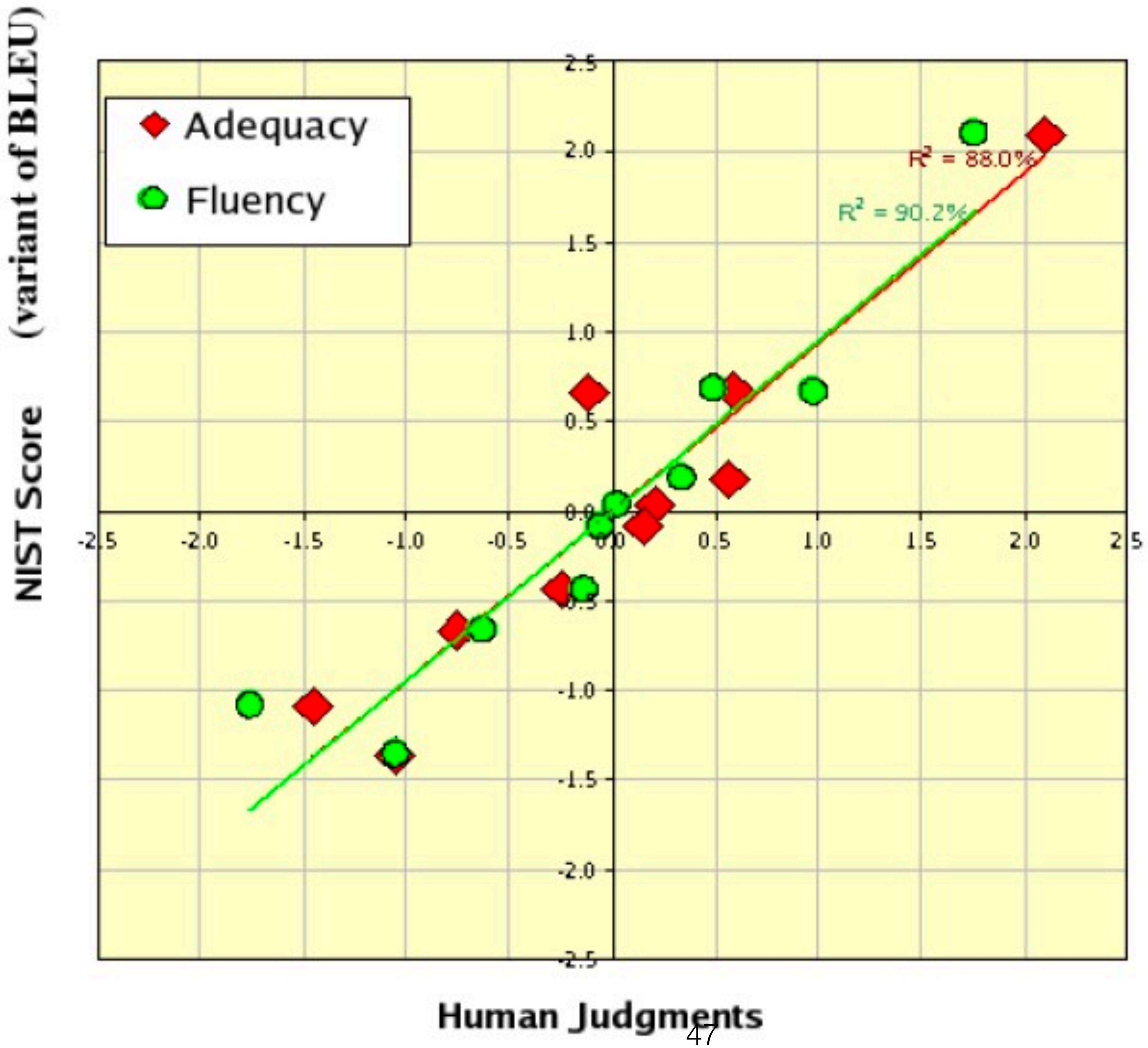
- Solution: Multiply score with a brevity penalty for translations shorter than reference, $BP = e^{1-r/h}$



r = reference length, h = hypothesis length

BLEU

- Correlates somewhat well with human judgements



(G. Doddington, NIST)

BLEU scores

https://www.nltk.org/_modules/nltk/translate/bleu_score.html

Sample BLEU scores for various system outputs $BP = e^{1-r/h}$

Length		Translation	p_1	p_2	p_3	p_4	BP	BLEU
5	*Reference*	*Vinay likes programming in Python*						
7	*Sys1*	*To Vinay it like to program Python*	$\frac{2}{7}$	0	0	0	1	.21
3	*Sys2*	*Vinay likes Python*	$\frac{3}{3}$	$\frac{1}{2}$	0	0	.51	.33
6	*Sys3*	*Vinay likes programming in his pajamas*	$\frac{4}{6}$	$\frac{3}{5}$	$\frac{2}{4}$	$\frac{1}{3}$	1	.76

Example from: https://github.com/jacobeisenstein/gt-nlp-class/tree/master/notes

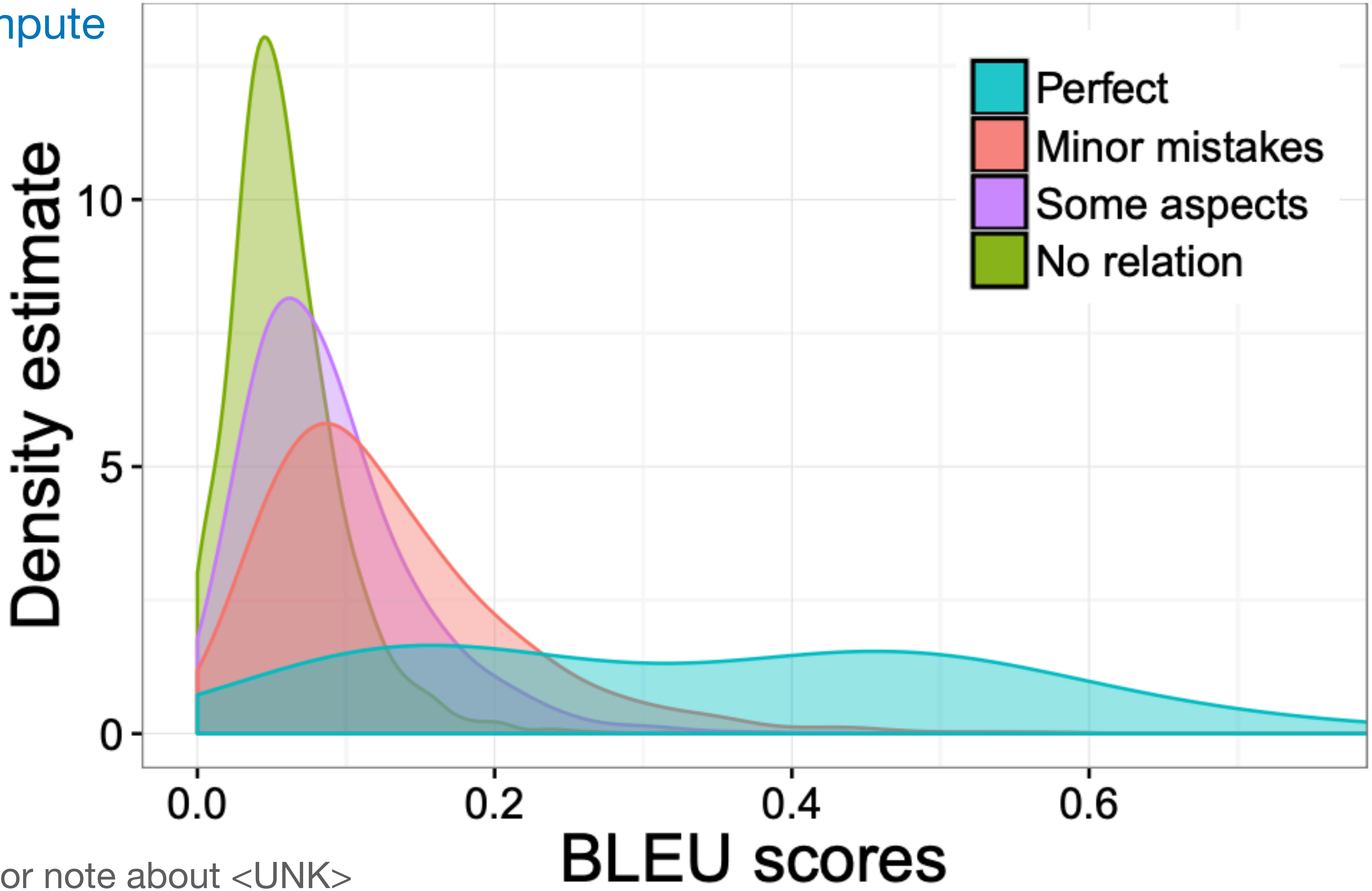
- Alternatives have been proposed:
  - METEOR: weighted F-measure
  - Translation Error Rate (TER): Edit distance between hypothesis and reference

Issues?

- Number is not that meaningful (BLEU will be higher for some language than others)
- Does not account for different word choices (synonyms)
- Does not account for morphology
- Does not penalize omitting important words

BLEU useful despite issues

- easy to compute
- automated
- consistent

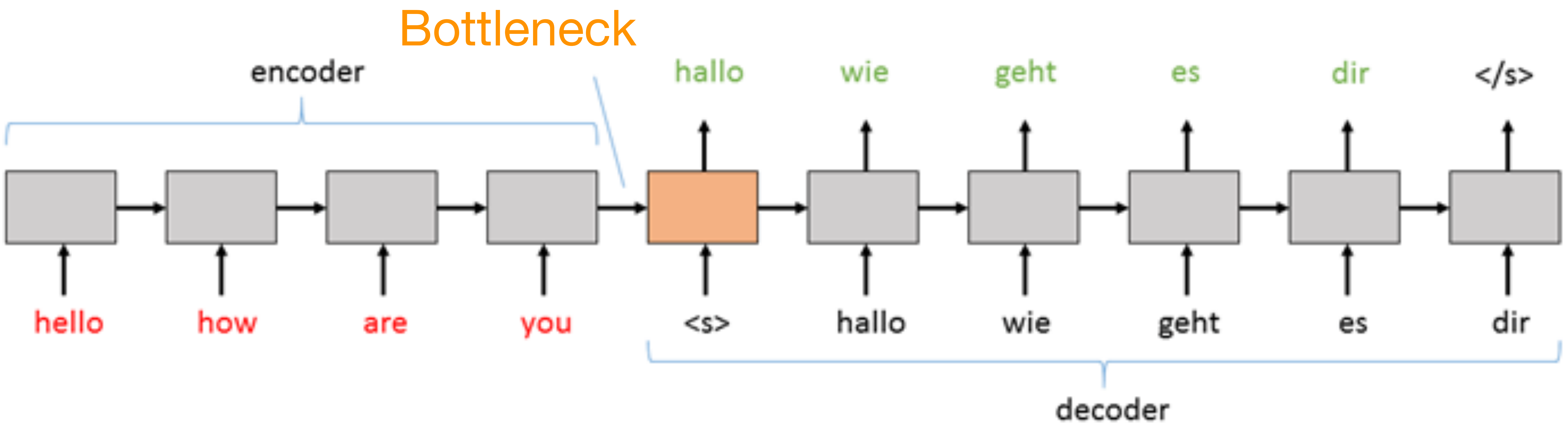


Minor note about <UNK>
Make sure you compare against the original reference
(Don't have <UNK>s in your reference)

Re-evaluating Automatic Metrics for Image Captioning
[Kilickaya et al, EACL 2017]

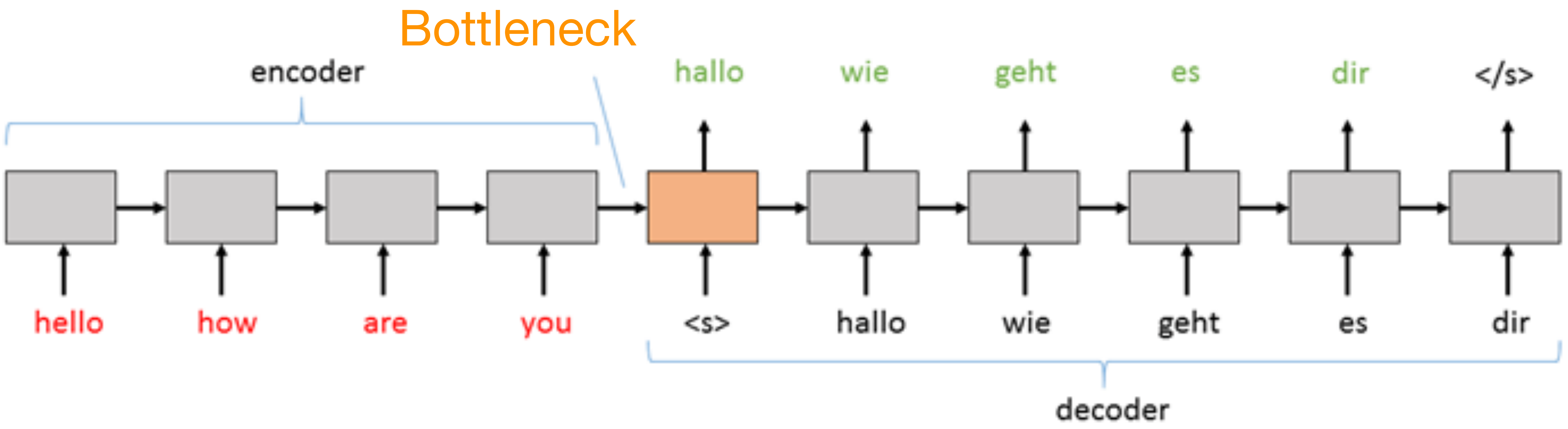
Sequence to sequence models with attention

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

Issues with vanilla seq2seq

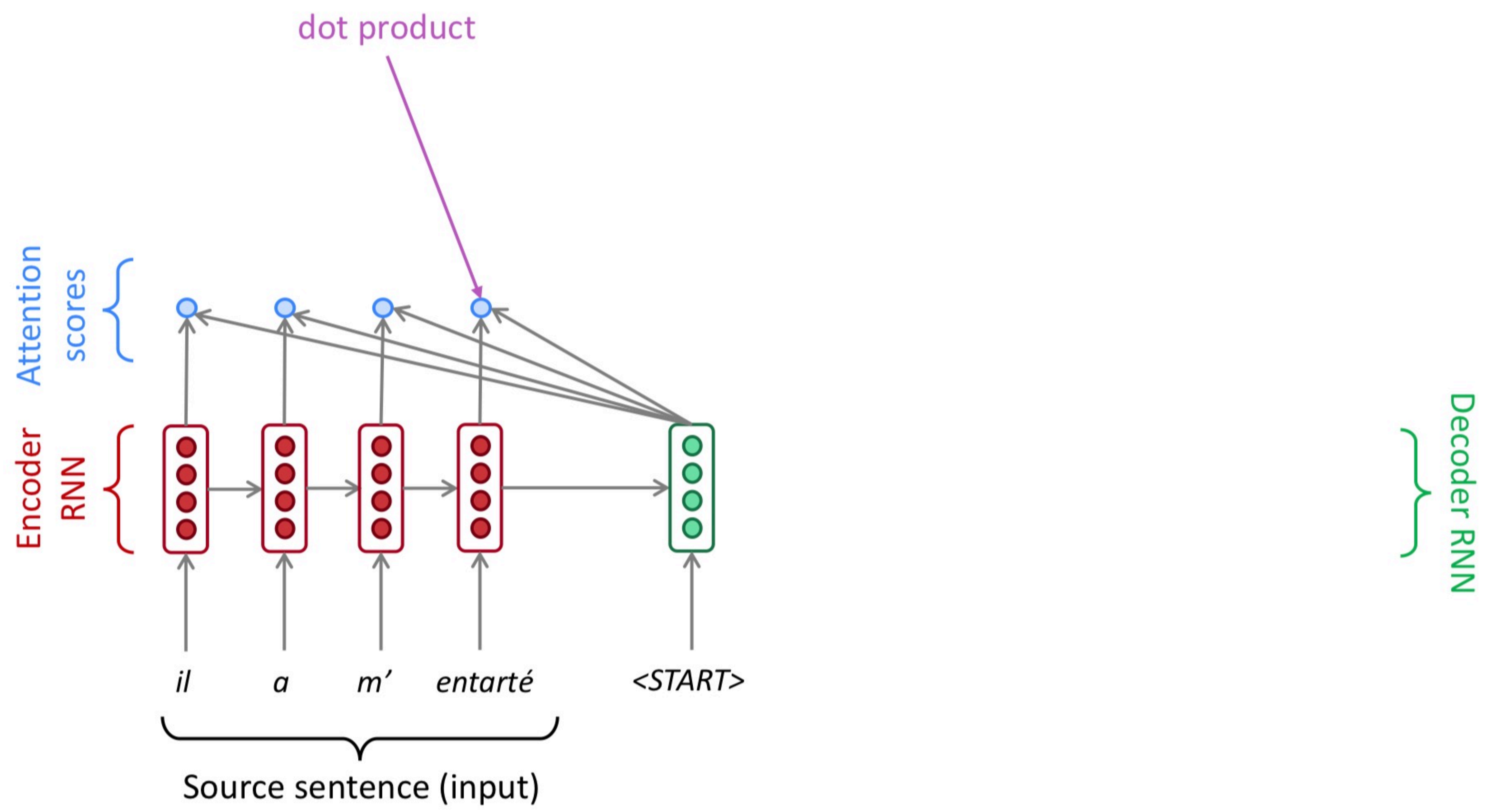


- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- **Longer sequences can lead to vanishing gradients**
- Overfitting

Attention

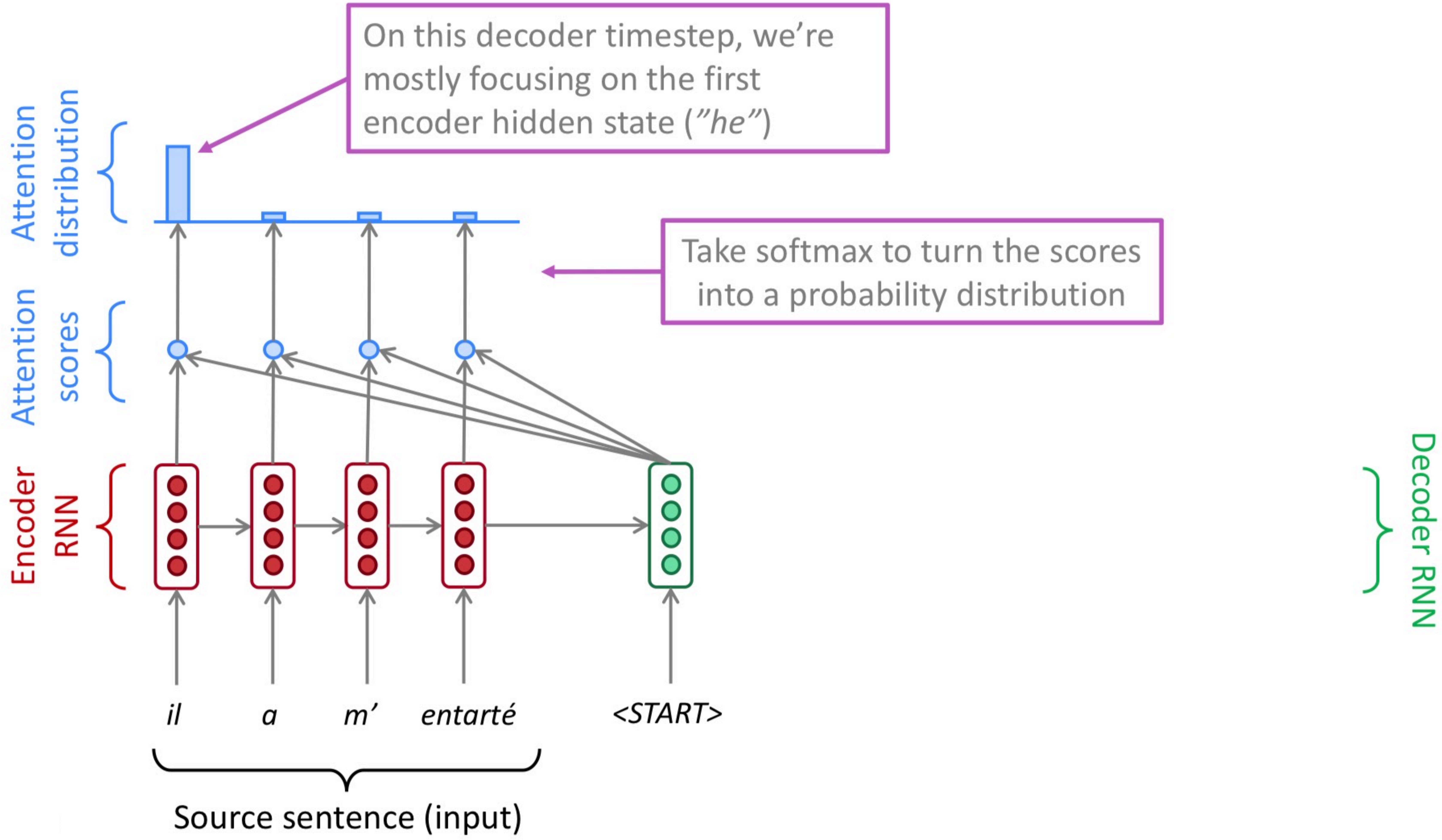
- The neural MT equivalent of alignment models
- Key idea: At each time step during decoding, **focus on a particular part** of source sentence
  - This depends on the decoder's current hidden state (i.e. notion of what you are trying to decode)
  - Usually implemented as a probability distribution over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

Seq2seq with attention



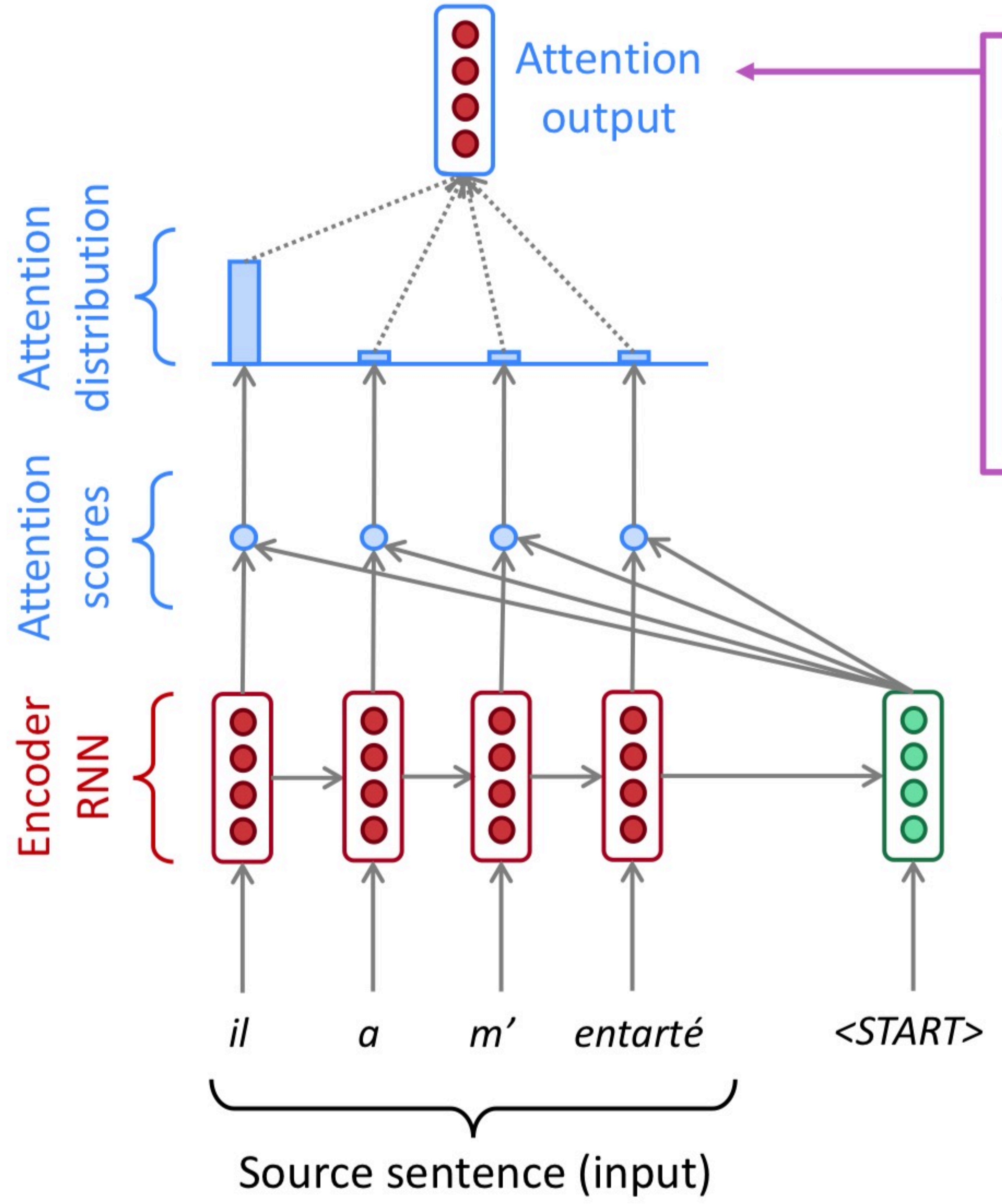
(slide credit: Abigail See)

Seq2seq with attention



(slide credit: Abigail See)

Seq2seq with attention

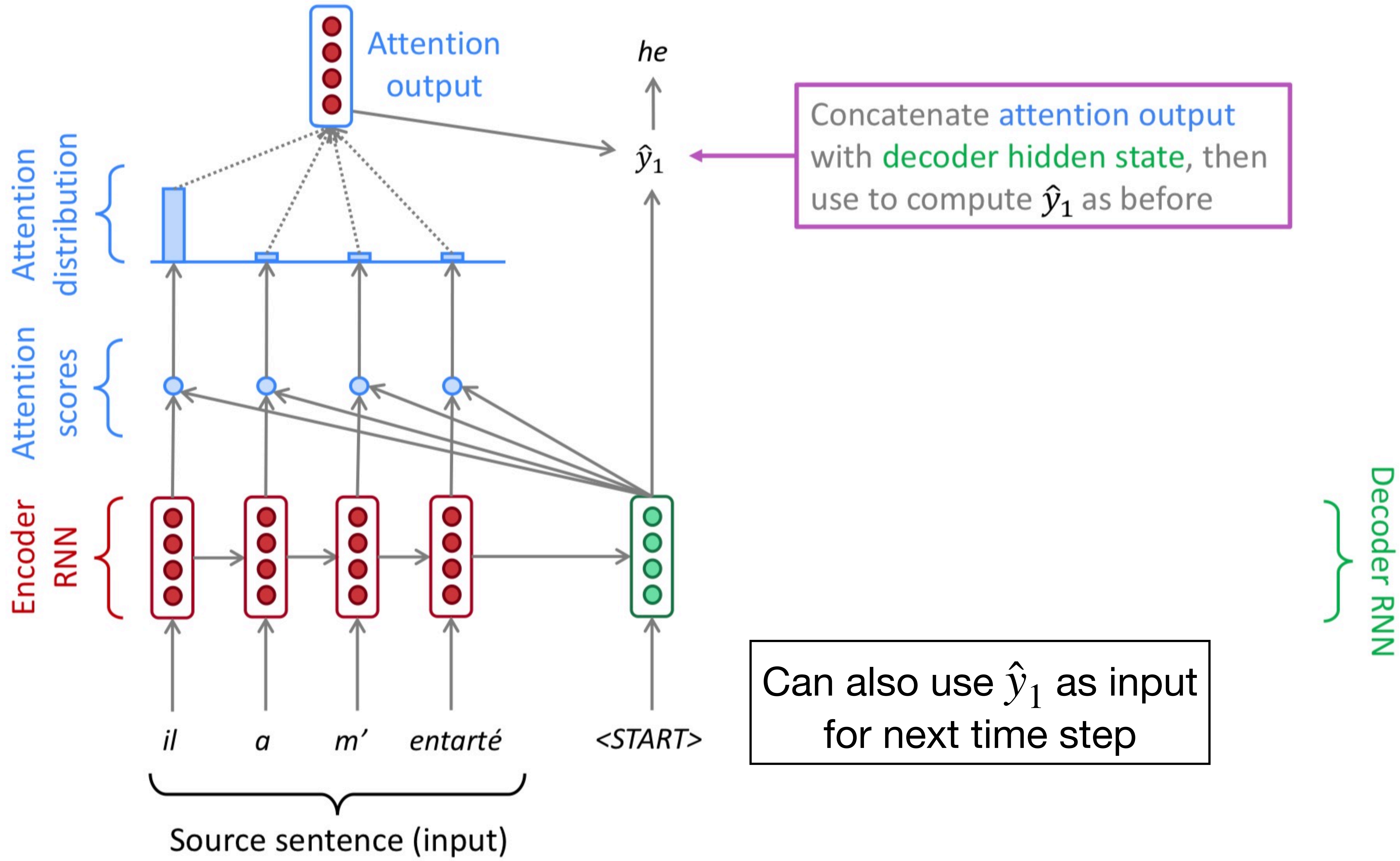


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

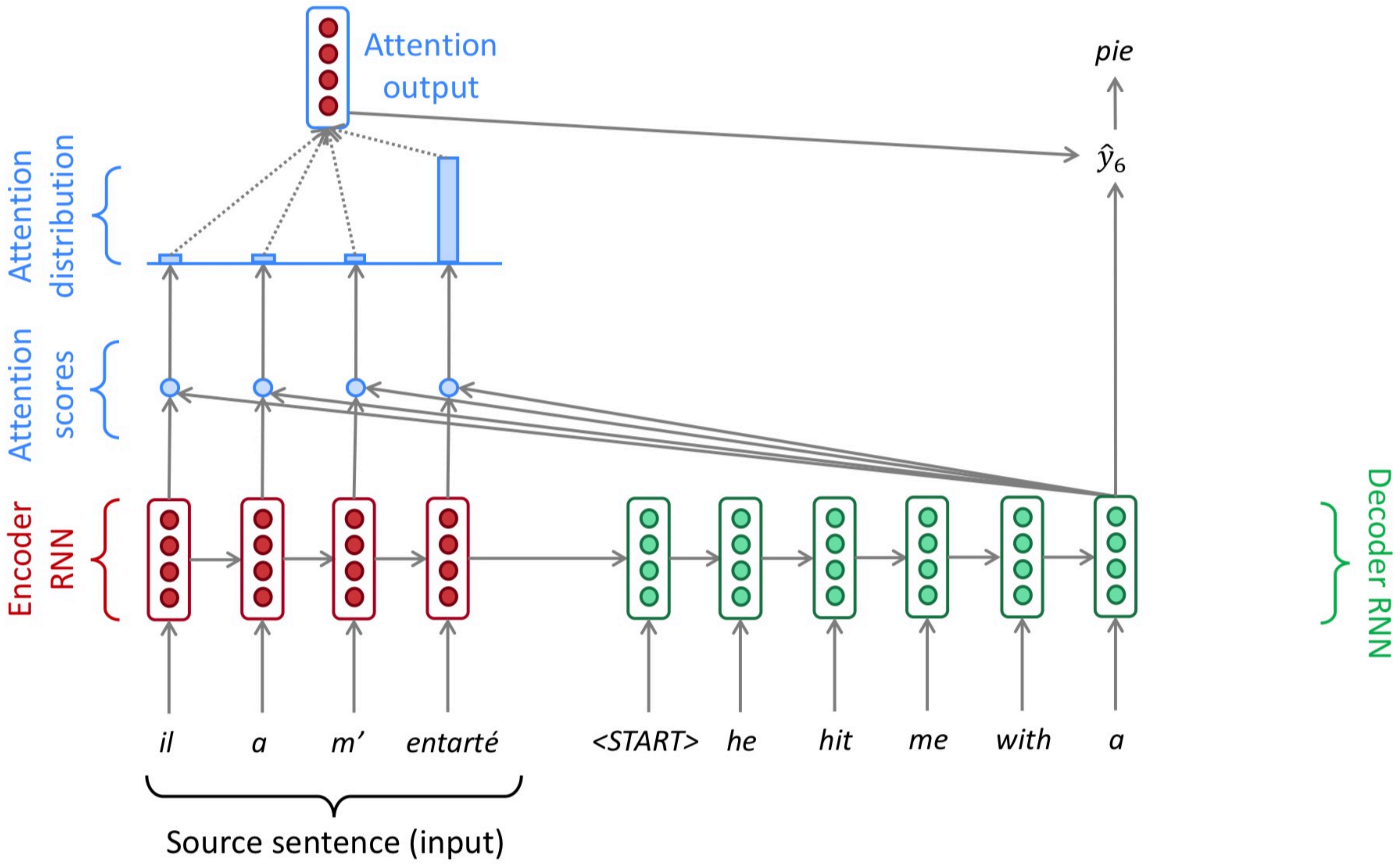
(slide credit: Abigail See)

Seq2seq with attention



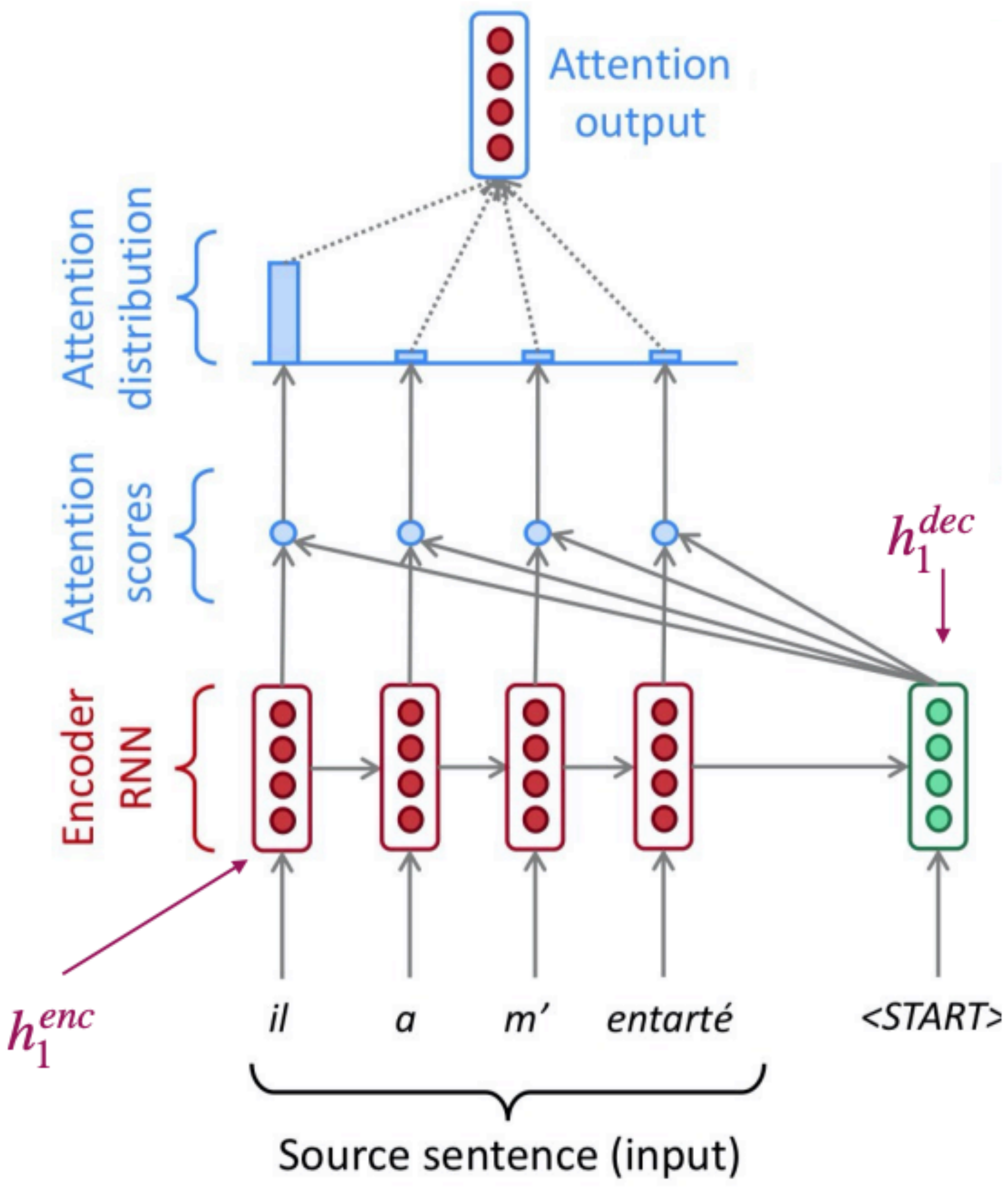
(slide credit: Abigail See)

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):**

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

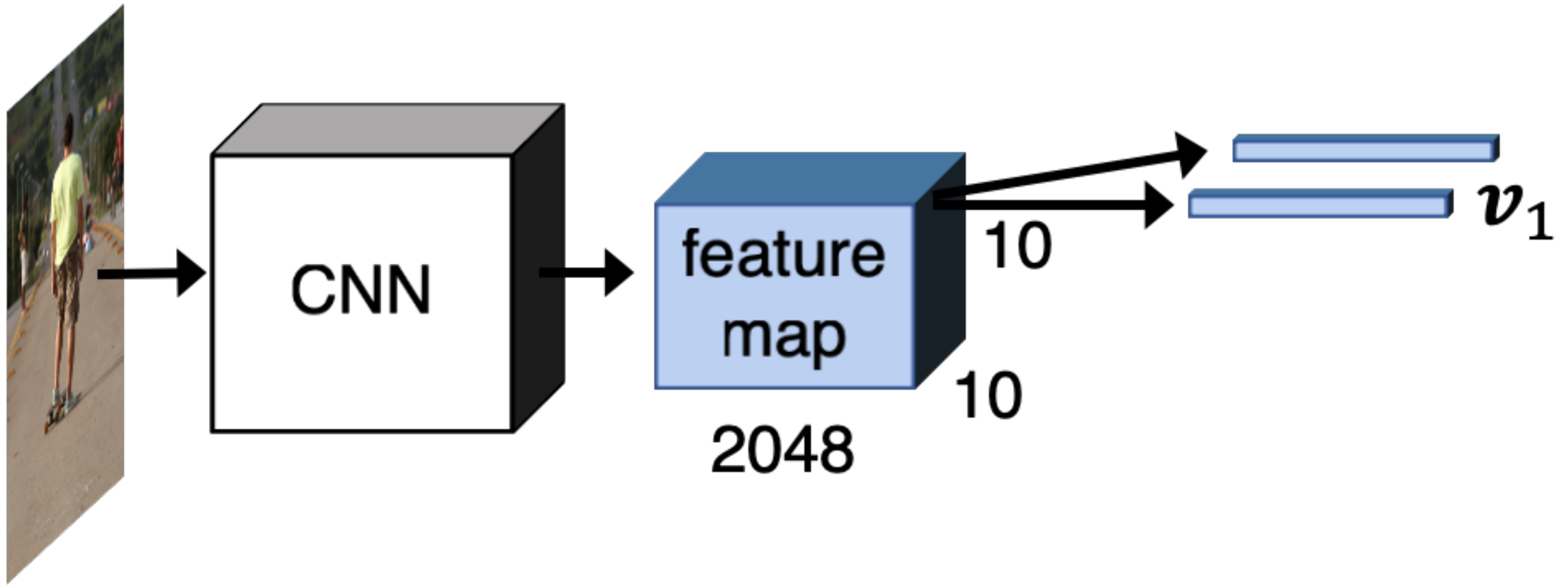
where W_1, W_2 are weight matrices and v is a weight vector

Perform better for larger dimensions

Attention can be applied to other modalities

Attention on other modalities

- Images



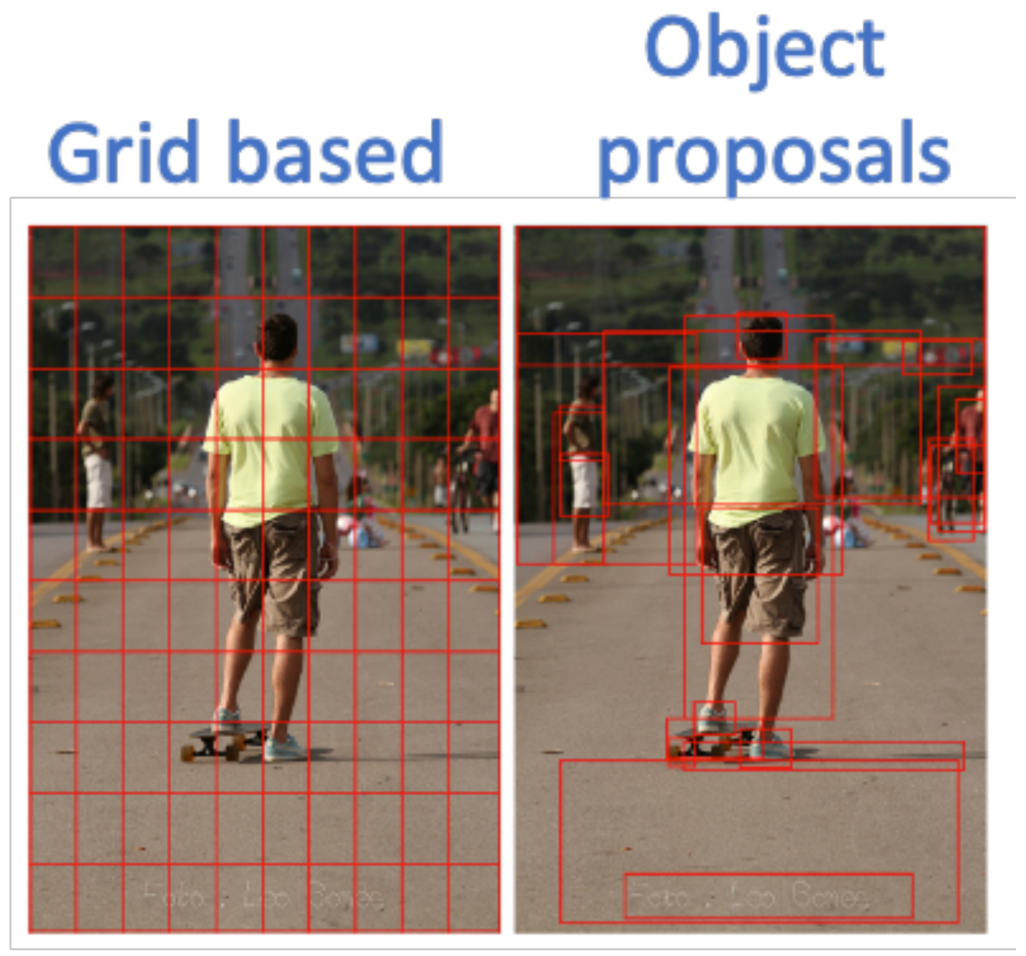
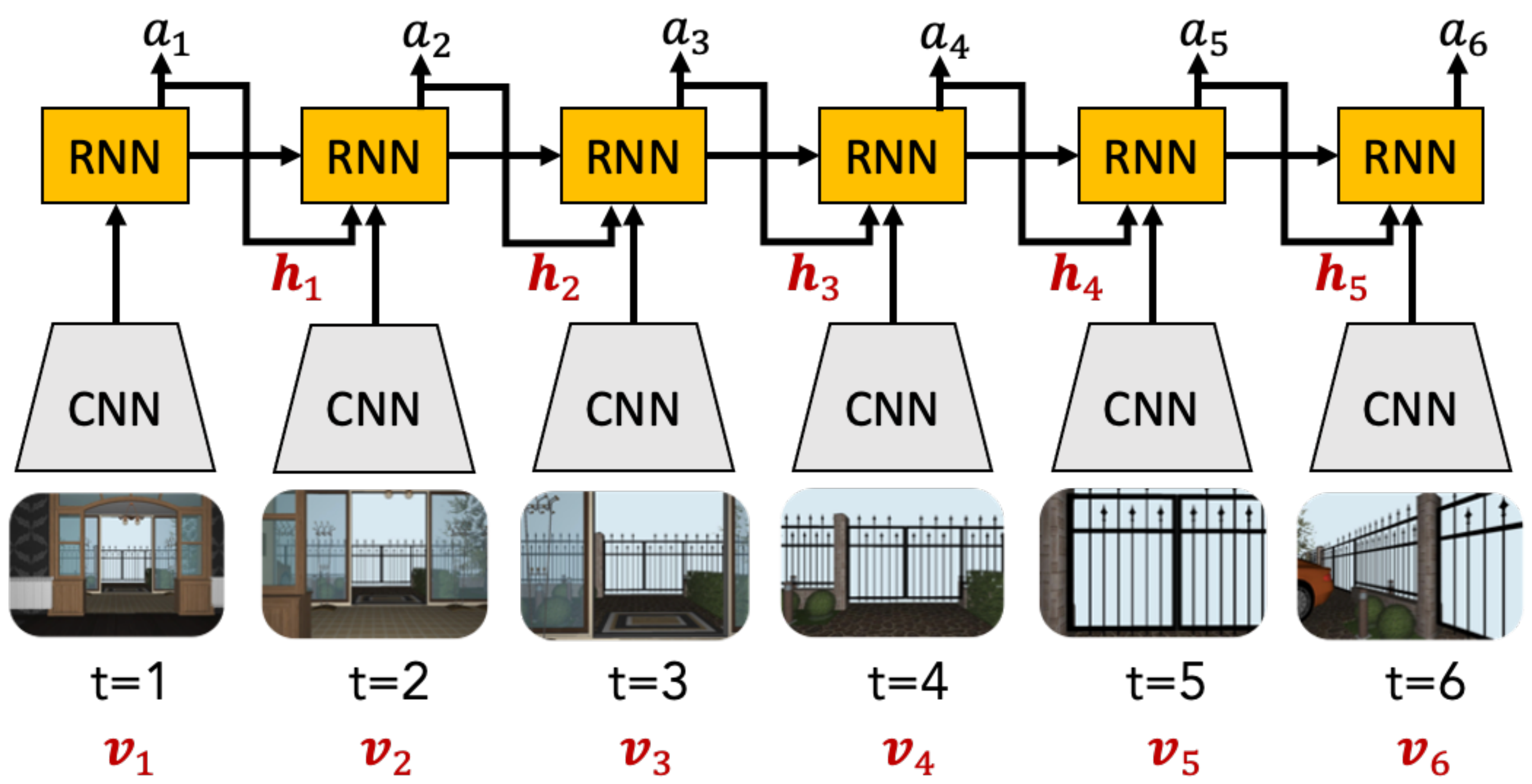


Image Credit: Peter Anderson

- Agent experience

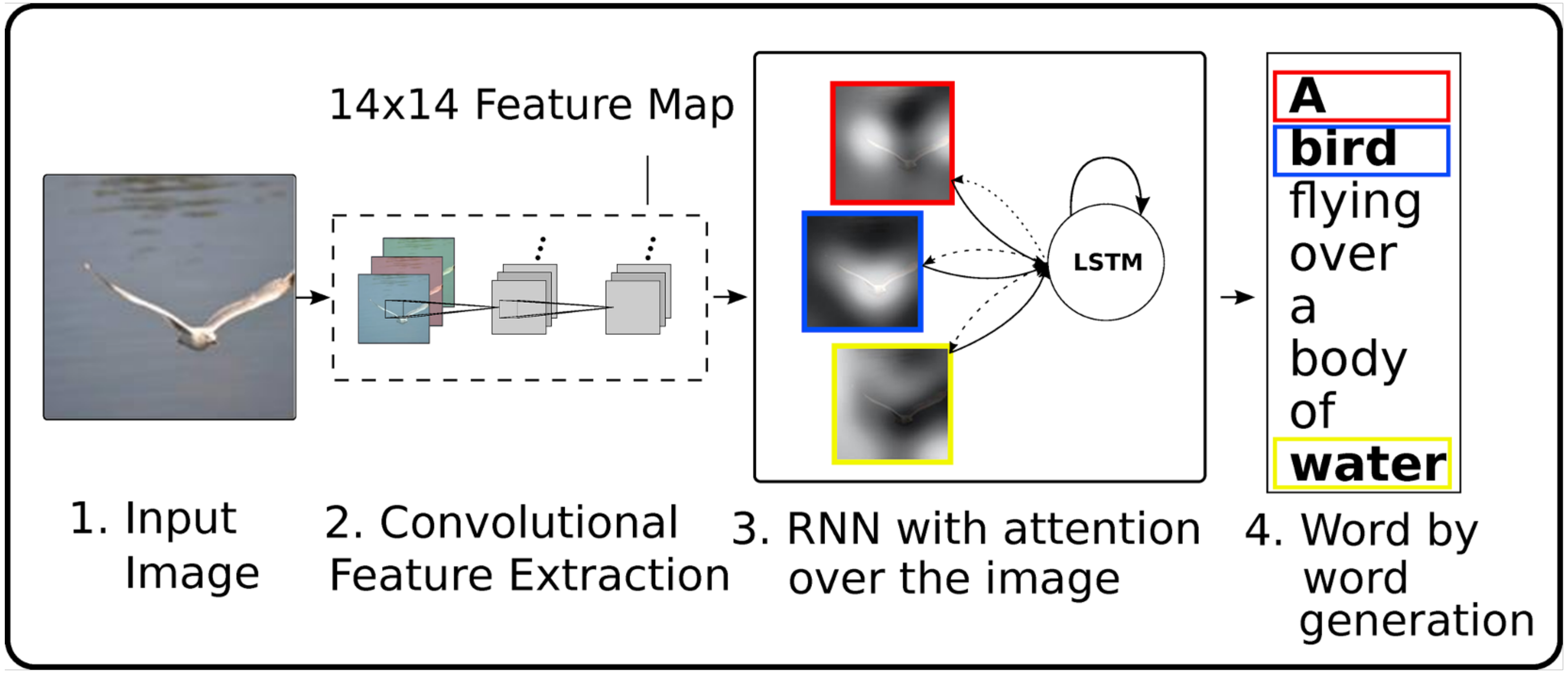


$$C = \{h_1, \ldots, h_5\}$$

or

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

Image captioning example



Xu et al. ICML 2015

Soft vs Hard Attention

- Soft: Each attention candidate is weighted by α_i

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

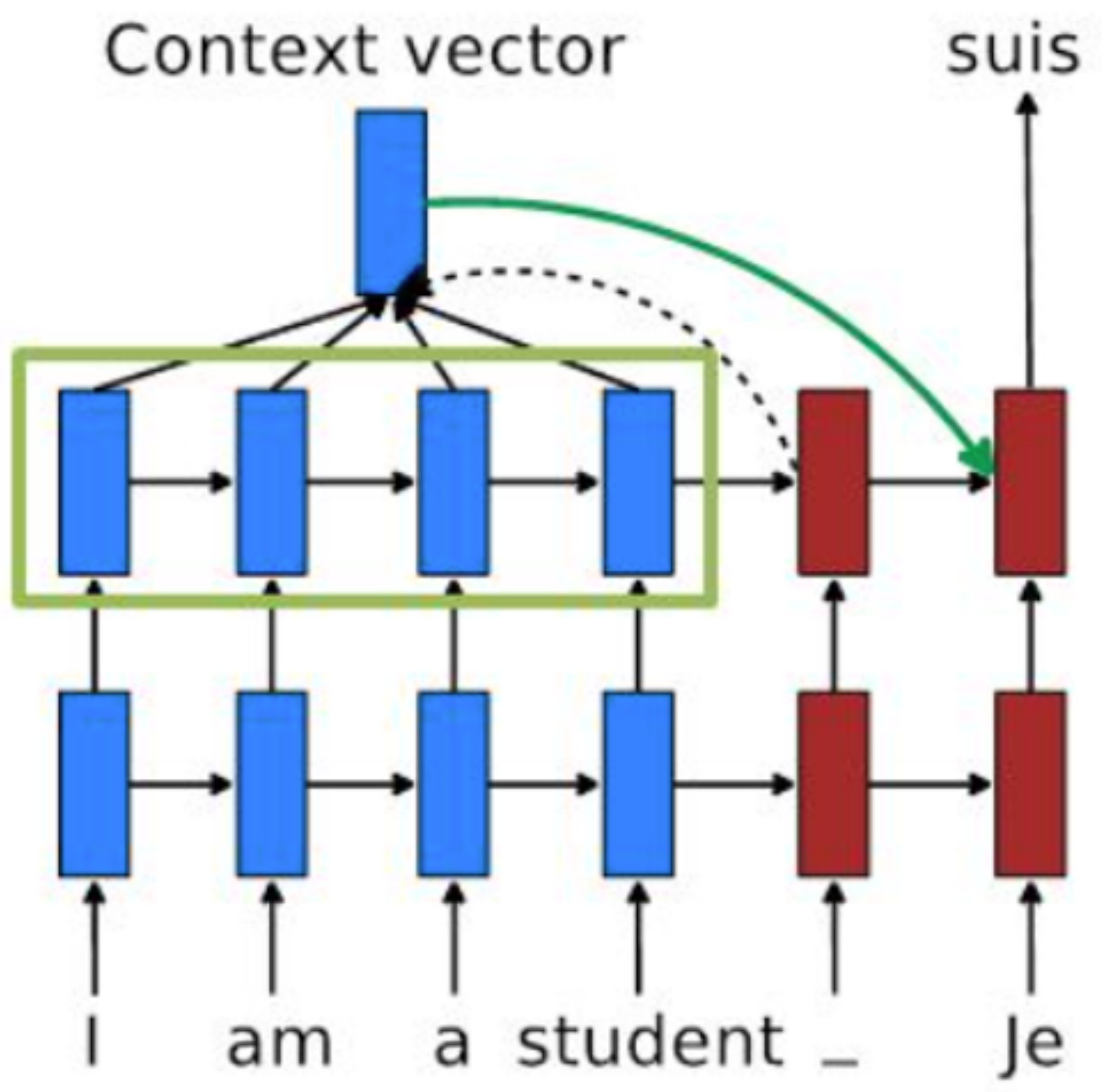
  - Easy to train (smooth and differentiable)
  - But can be expensive over large input

- 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)

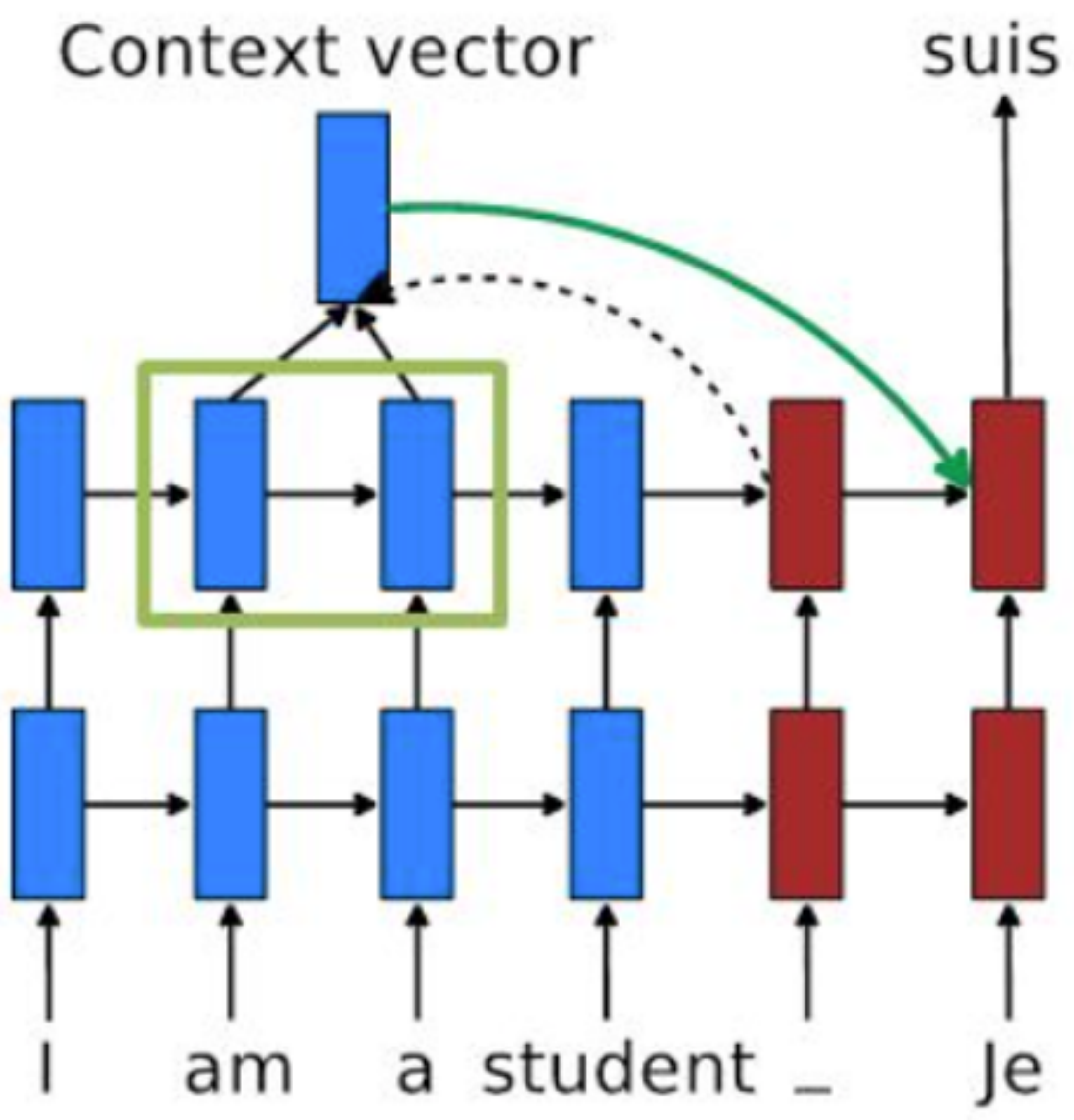
Xu et al. ICML 2015

Global vs Local Attention

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



Global: ***all*** source states.

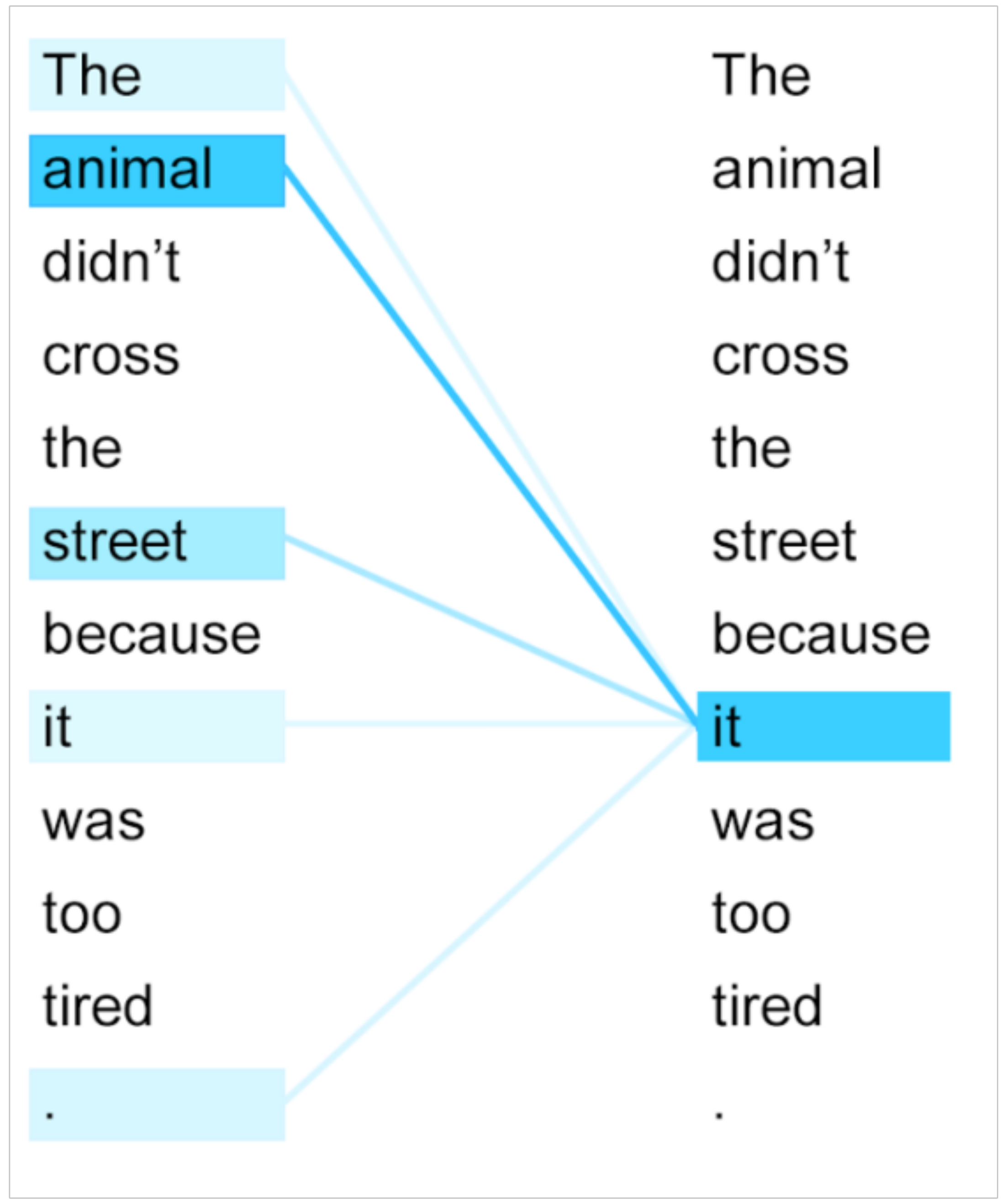


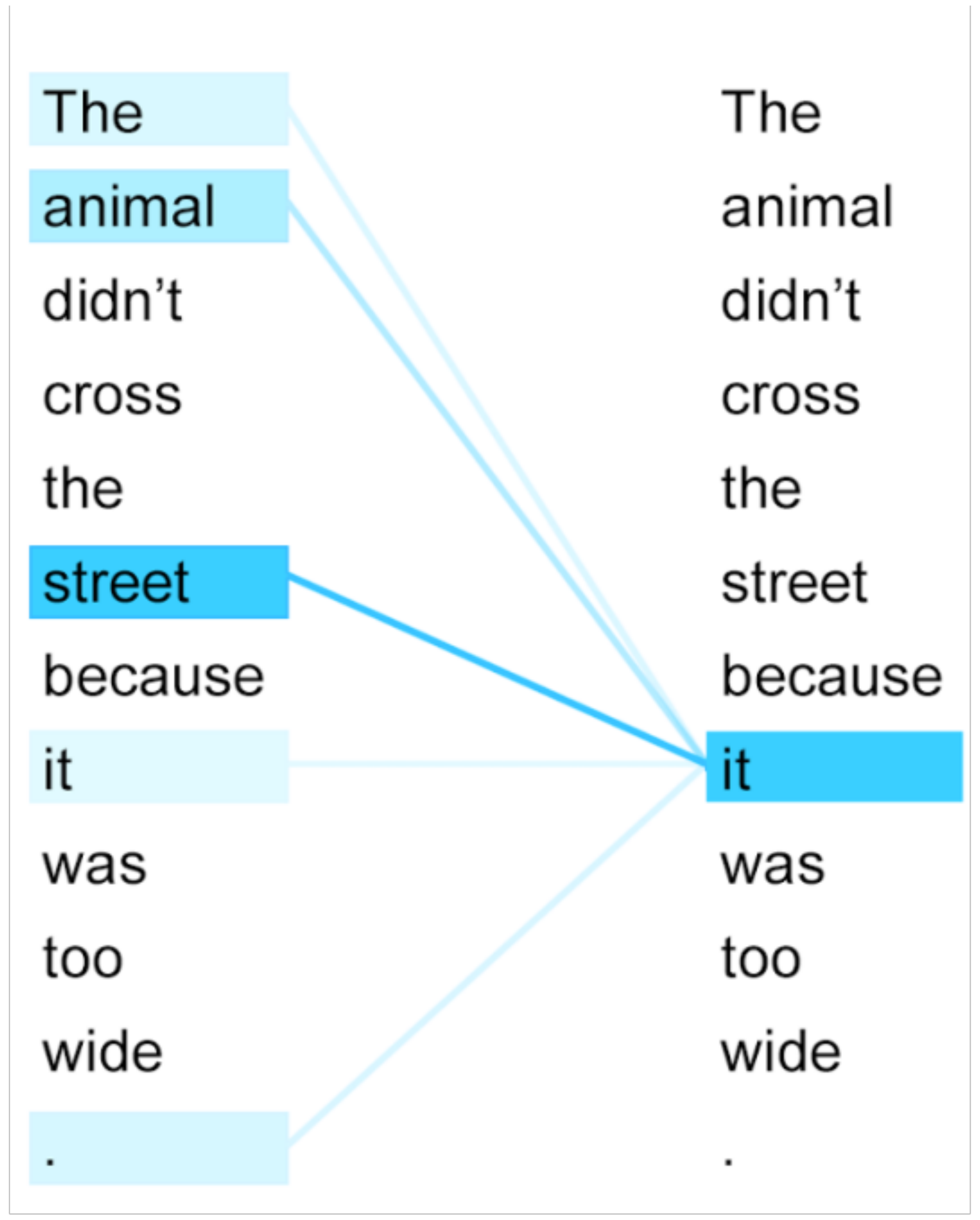
Local: ***subset*** of source states.

Luong et al, 2015

Self-Attention

- Attention (correlation) with different parts of itself

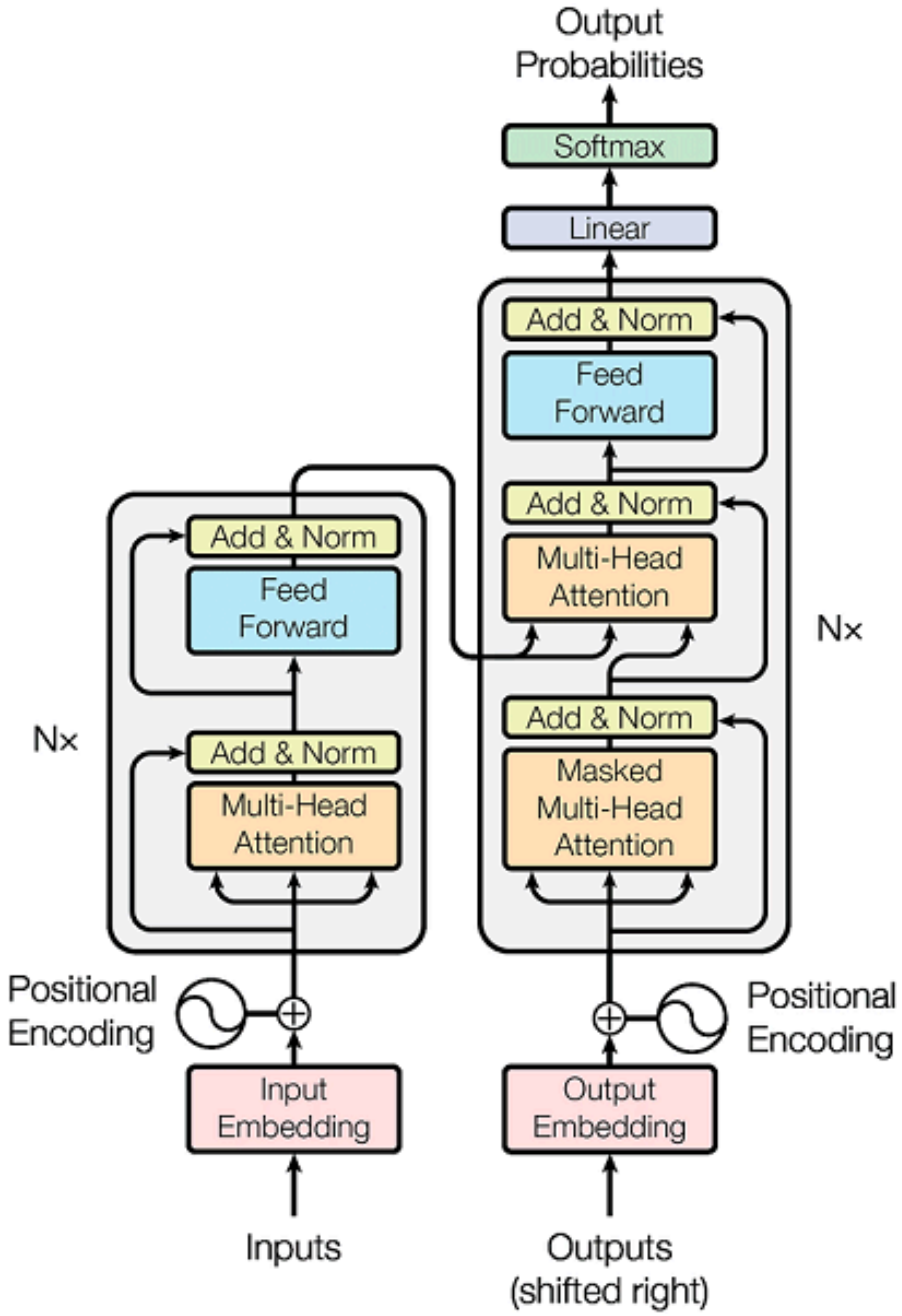




https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

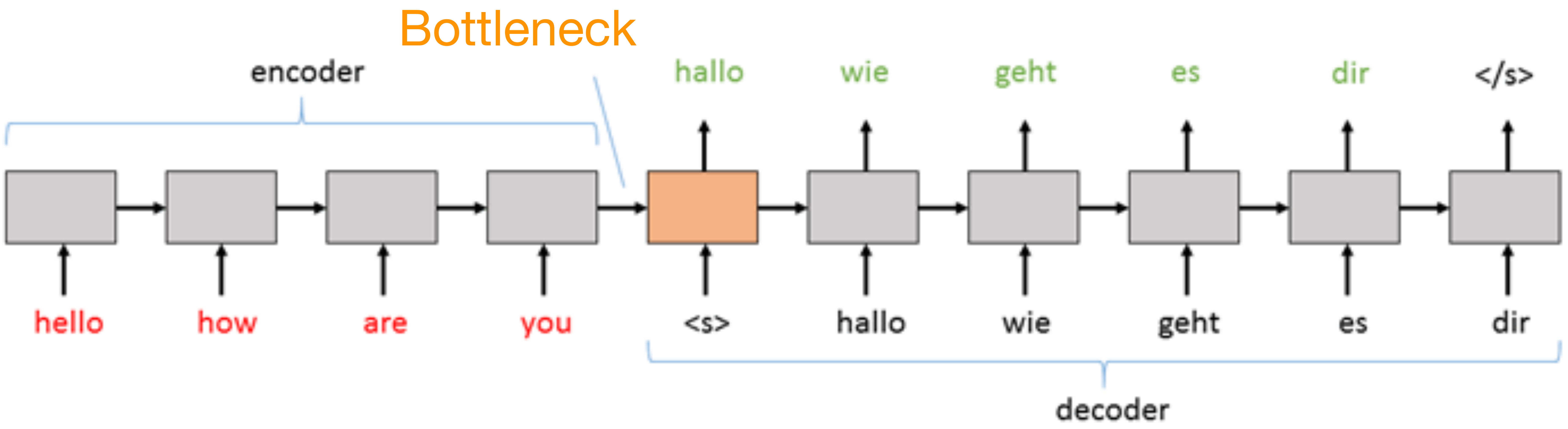
- Transformers: modules with scaled dot-product self-attention

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

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**

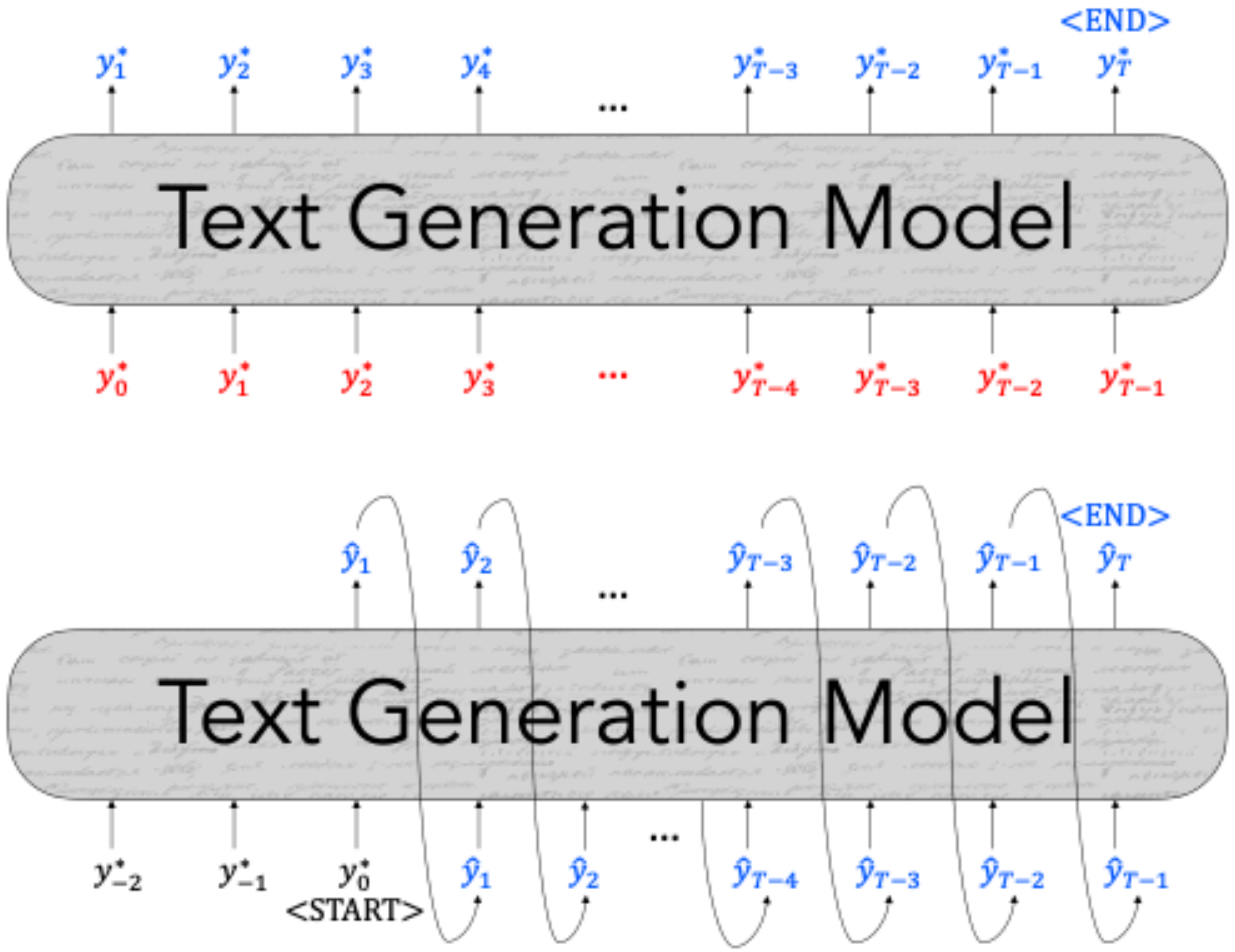
Exposure bias

- 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

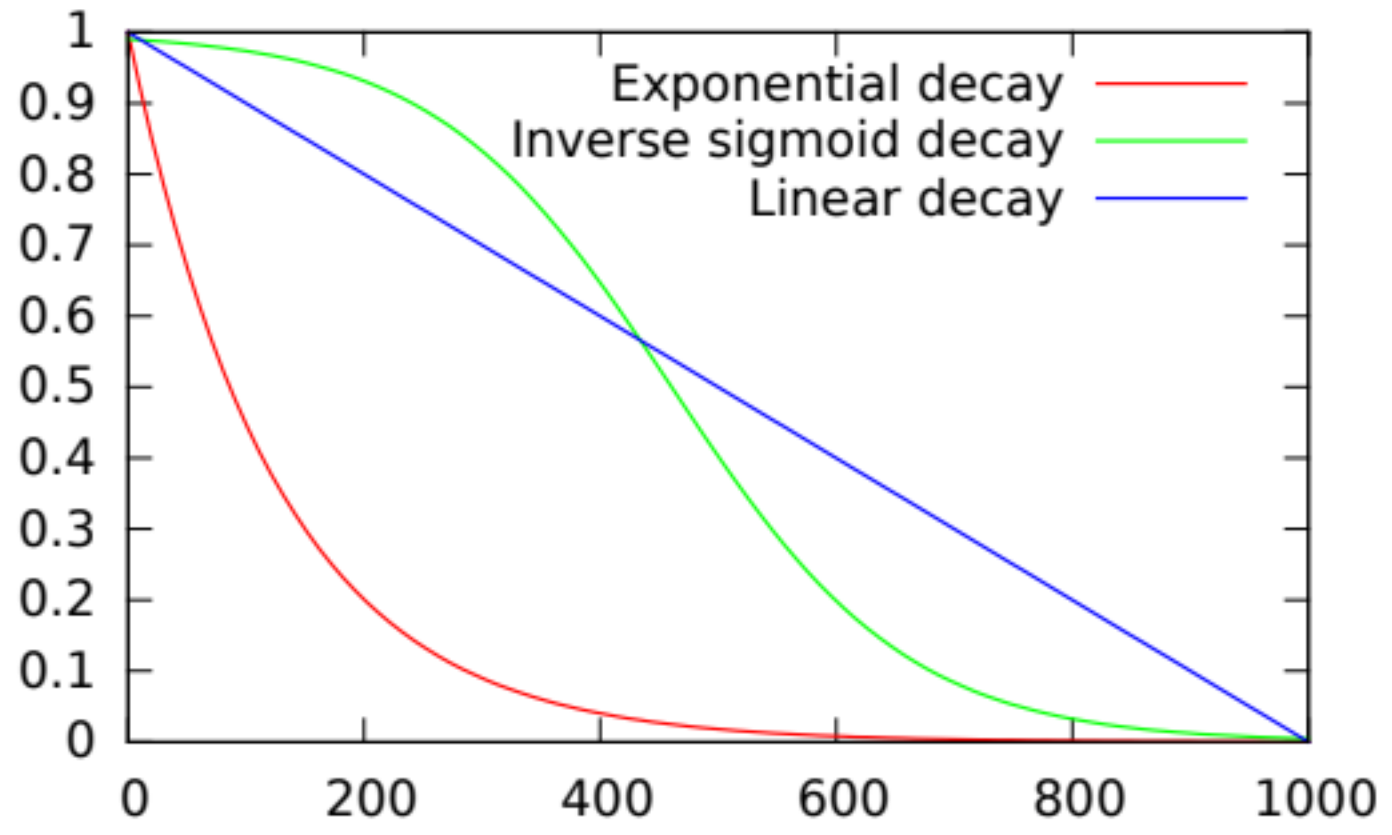
$$\mathcal{L}_{dec} = -\sum_{t=1}^{T} \log P(\hat{y}_t | \{\hat{y}_{<t}\})$$



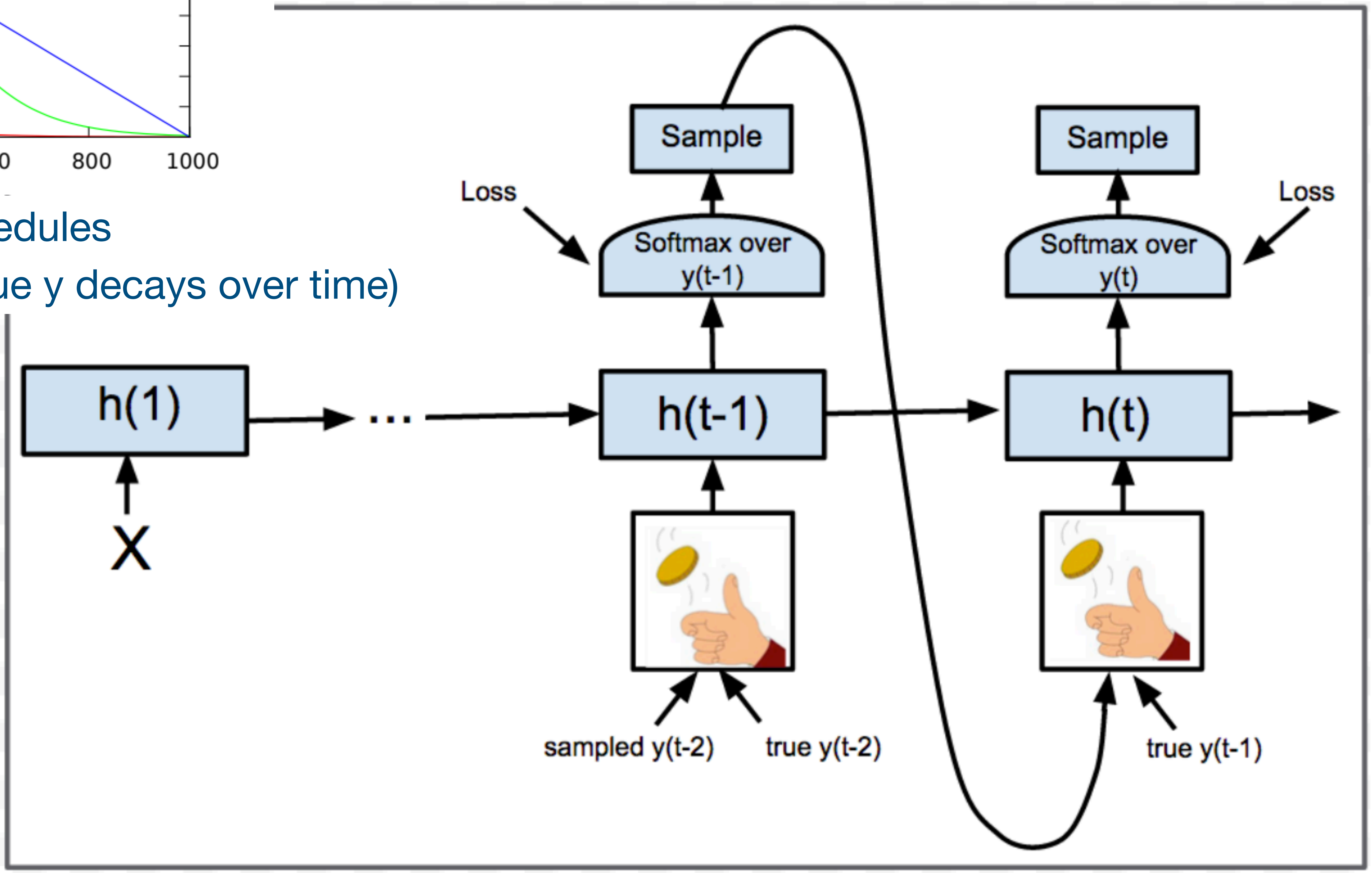
Student forcing: use predicted tokens during training

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
- Label smoothing
- Dropout
- Ensembling

Weight decay

- Weight decay
  - Decays weights θ exponentially
  - $\theta^{t+1} = (1-\lambda)\theta^t - \eta \frac{d}{d\theta} L(\theta)$
- For SGD, weight decay and L2 regularization are equivalent

Weight decay and SGD

- SGD
  - $\theta_{t+1} = \theta_t - \eta \frac{d}{d\theta} L(\theta)$
- L2 regularization
  - $L_{\text{L2}} = L(\theta) + \alpha ||\theta||_2^2$
  - $\frac{dL_{\text{L2}}}{d} = \frac{dL(\theta)}{d\theta} + 2\alpha\theta$
- SGD with L2 regularization
  - $\theta_{t+1} = \theta_t - \eta \frac{d}{d\theta} L_{\text{L2}}(\theta)$
  - $\theta_{t+1} = (1 - 2\eta\alpha)\theta_t - \eta \frac{d}{d\theta} L(\theta)$
- L2 regularization with $\alpha = \frac{\lambda}{2\eta}$ gives
  - $\theta_{t+1} = (1 - \lambda)\theta_t - \eta \frac{d}{d\theta} L(\theta)$

Weight decay

- Weight decay
  - Decays weights θ exponentially
  - $\theta^{t+1} = (1-\lambda)\theta^t - \eta \frac{d}{d\theta} L(\theta)$
- For SGD, weight decay and L2 regularization are equivalent
  - But for this to hold, the weight decay and learning rate are coupled for a desired L2 regularization
- Weight decay and L2 regularization are not necessarily equivalent for adaptive optimizers
- Can decouple weight decay and learning rate parameters
  - AdamW

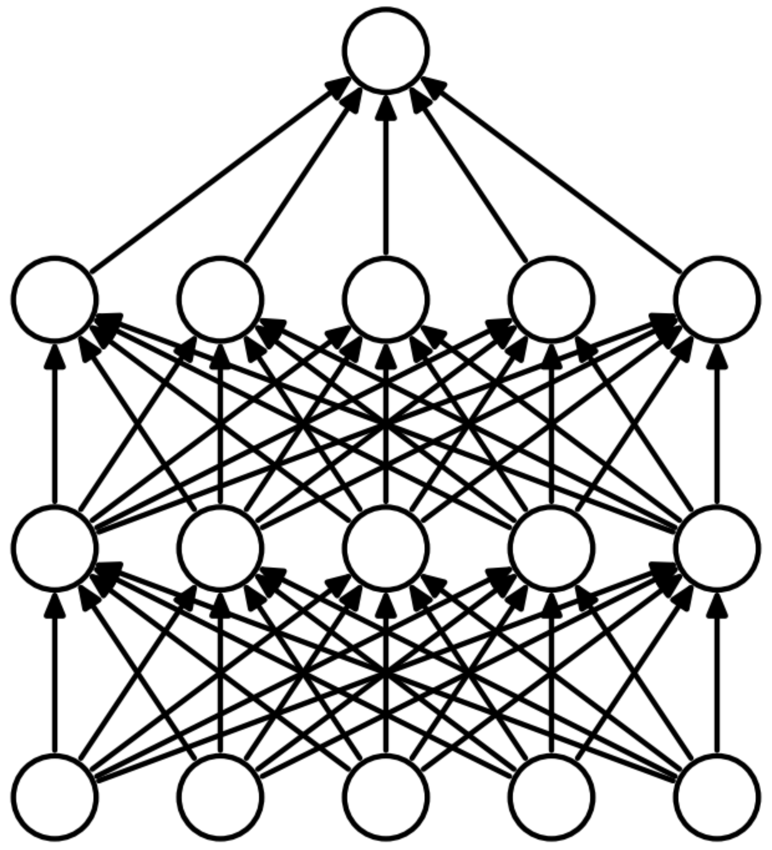
Label smoothing

- Cross entropy loss
  - $$L = -\sum_{k=1}^{K} q(k) \log p(k)$$
  - Ground-truth $q(k) = \delta(y) = 1[y = k]$
- Label smoothing
  - Smoothed distribution for training
  - $q(k) = \epsilon\delta(y) + (1-\epsilon)u(k)$
  - $u(k)$ is prior - simplest prior is the uniform distribution: $u(k) = \dfrac{1}{K}$

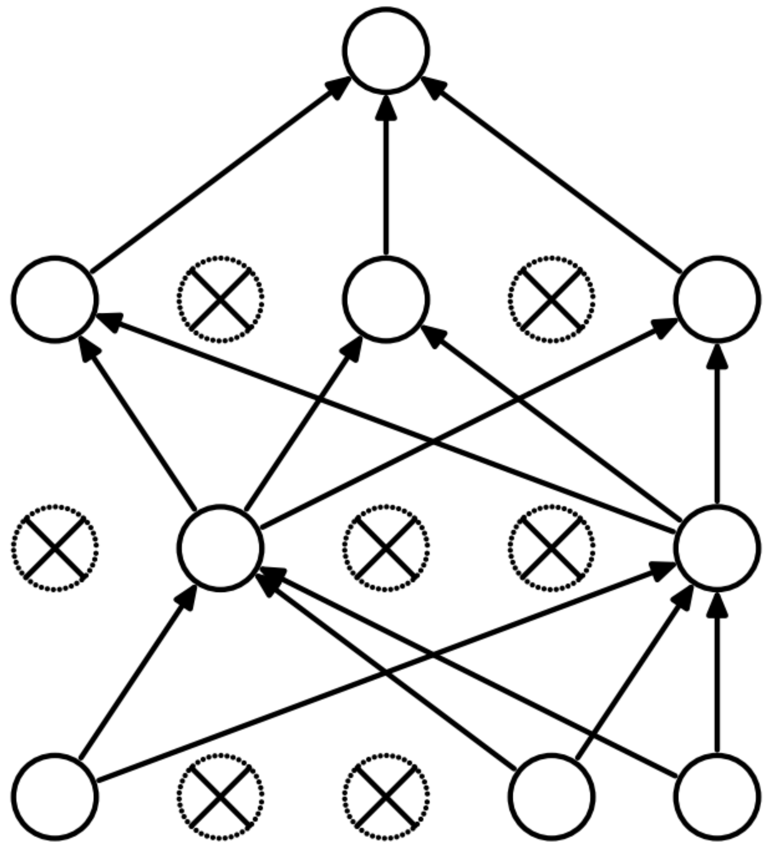
Figure from https://blog.dailydoseofds.com/p/label-smoothing-the-overlooked-and

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)

Other challenges with NMT

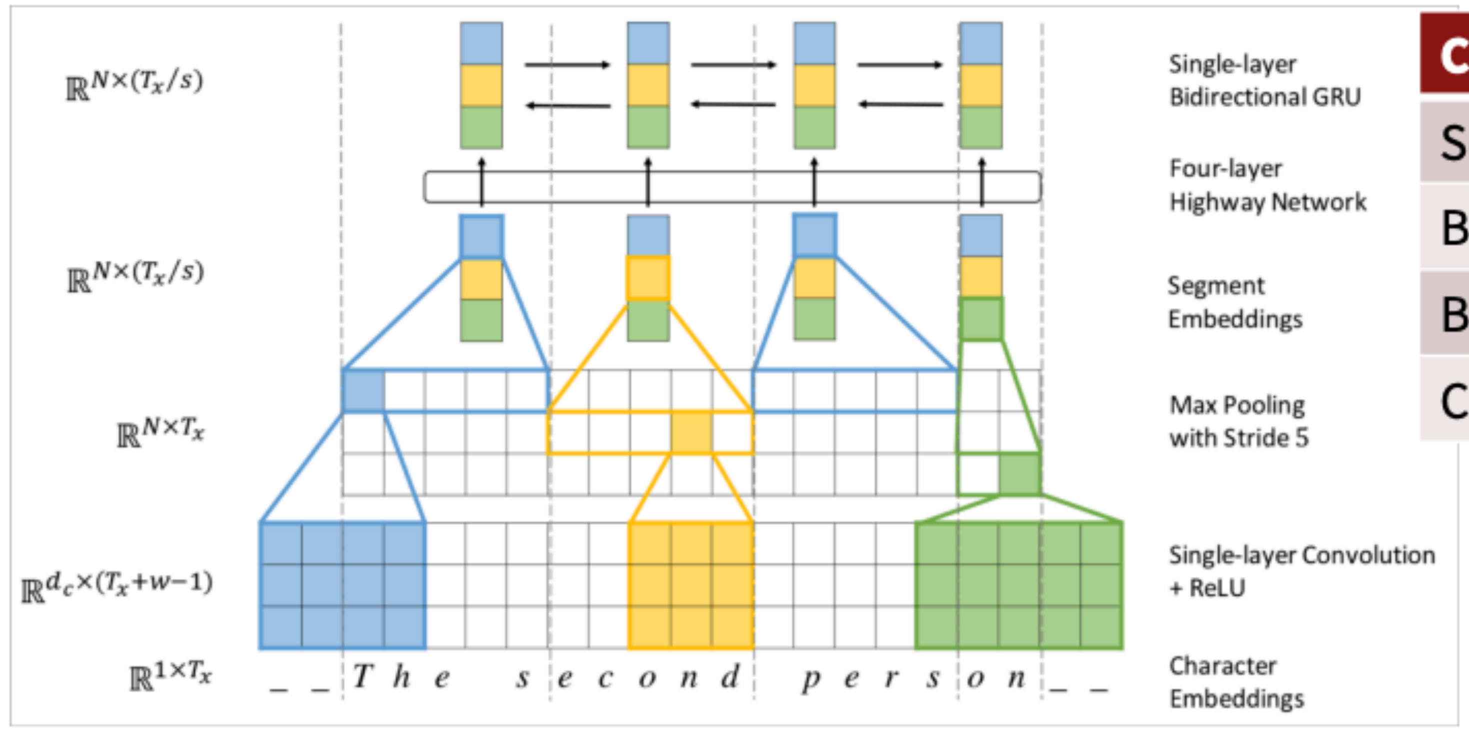
- Out of vocabulary (OOV)
- Low-resource languages
- Long-term context
- Common sense knowledge (e.g. hot dog, paper jam)
- Fairness and bias
- Interpretability

Out of vocabulary (OOV)

- Subword-modeling
  - Character level GRU
  - Byte-pair encoding

Fully Character-Level Neural Machine Translation without Explicit Segmentation

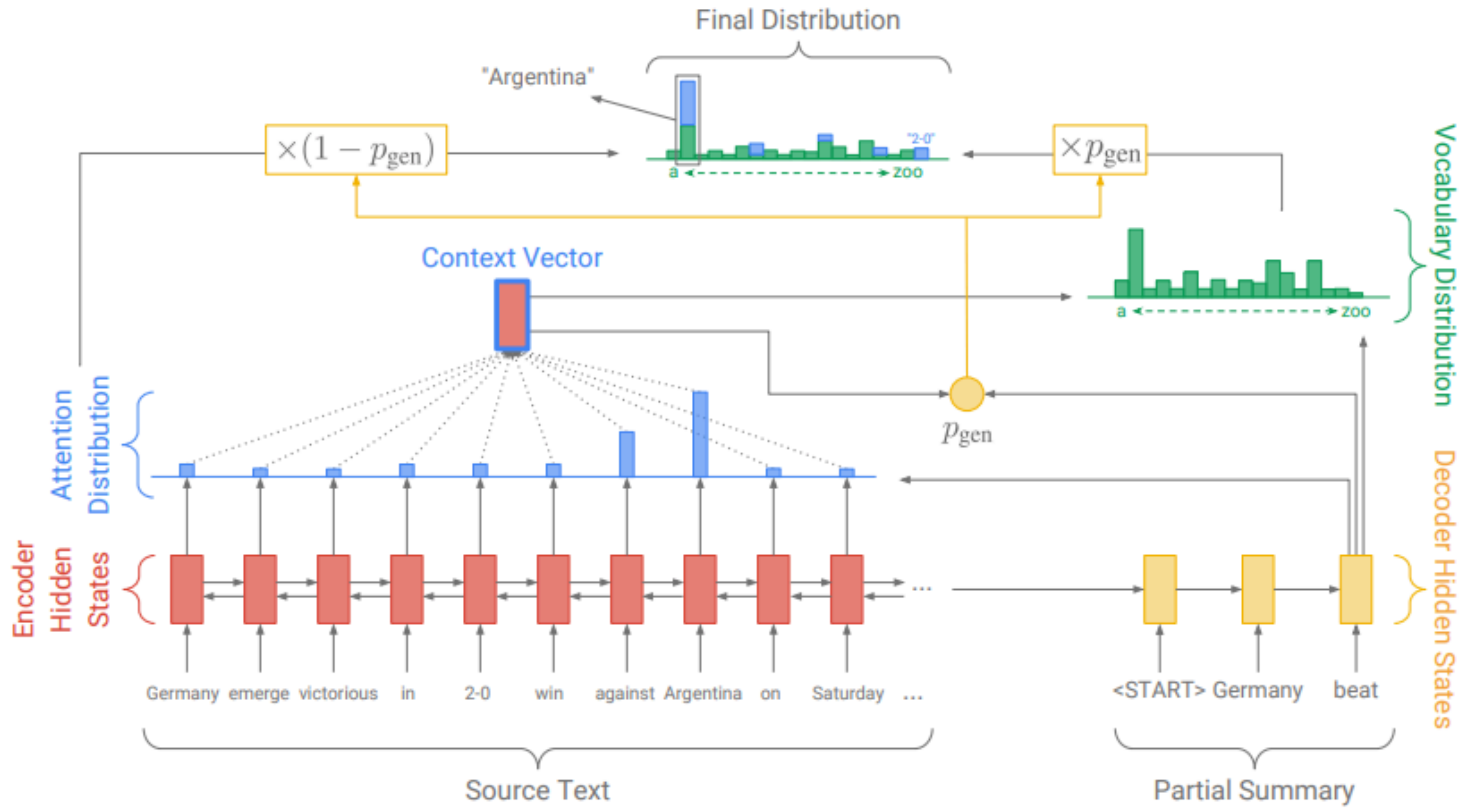
Jason Lee, Kyunghyun Cho, Thomas Hoffmann. 2017.
Encoder as below; decoder is a char-level GRU



CS-En	WMT 15	Test
Source	Target	BLEU
Bpe	Bpe	20.3
Bpe	Char	22.4
Char	Char	22.5

(Lee et al, 2017)

- Copy mechanism



  - Probability of generating from vocabulary or copying from input
  - Probability of copying specific word (similar to attention)

(See et al, 2017)