



CMPT 825: Natural Language Processing

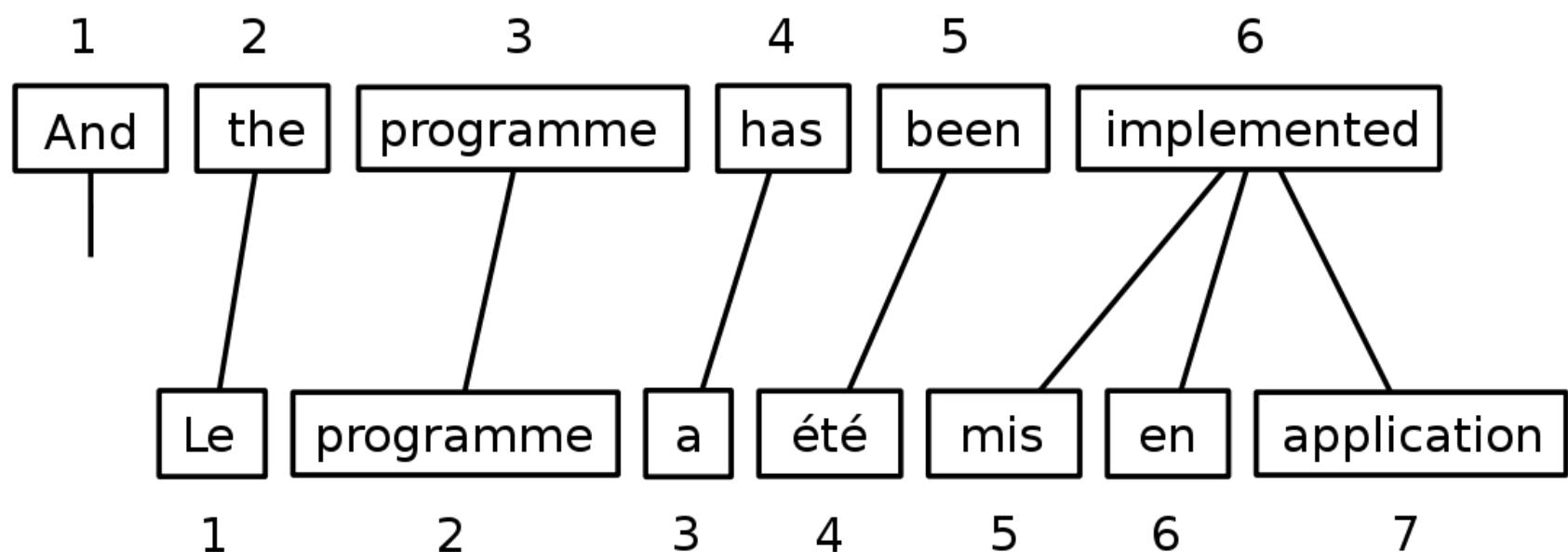
Sequence to Sequence Models (Seq2Seq) and Neural Machine Translation

Fall 2020
2020-10-23

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

Video Lecture

- Statistical MT
- Word-based
- Phrase-based
- Syntactic



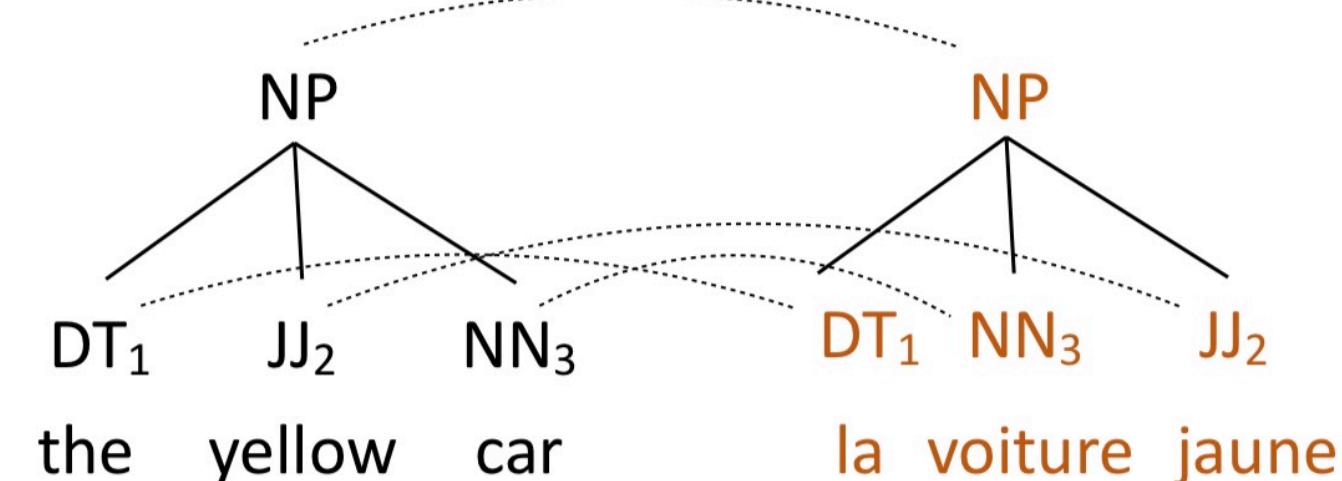
$NP \rightarrow [DT_1 JJ_2 NN_3; DT_1 NN_3 JJ_2]$

$DT \rightarrow [\text{the}, \text{la}]$

$DT \rightarrow [\text{the}, \text{le}]$

$NN \rightarrow [\text{car}, \text{voiture}]$

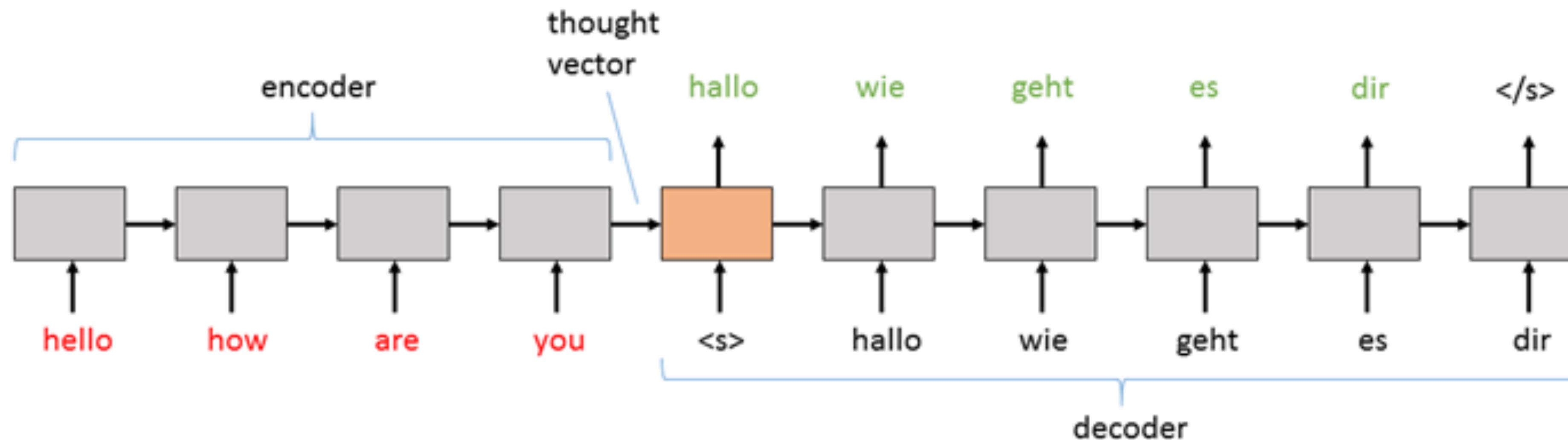
$JJ \rightarrow [\text{yellow}, \text{jaune}]$



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)



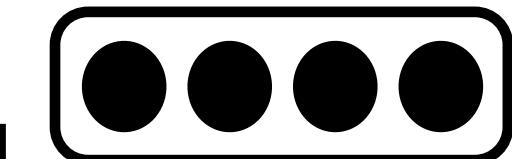
- Encode entire input sequence into a single vector (**using an RNN**)
- Decode one word at a time (**again, using an RNN!**)
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

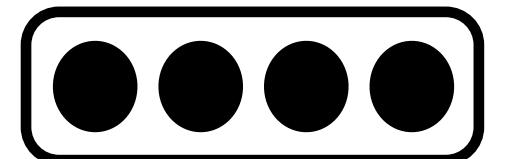
Encoder

Sentence: This cat is cute

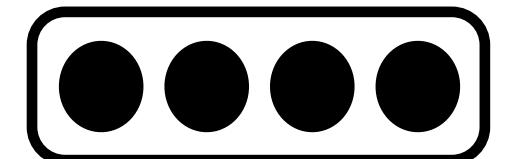
word
embedding



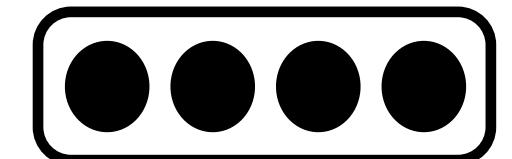
This



cat



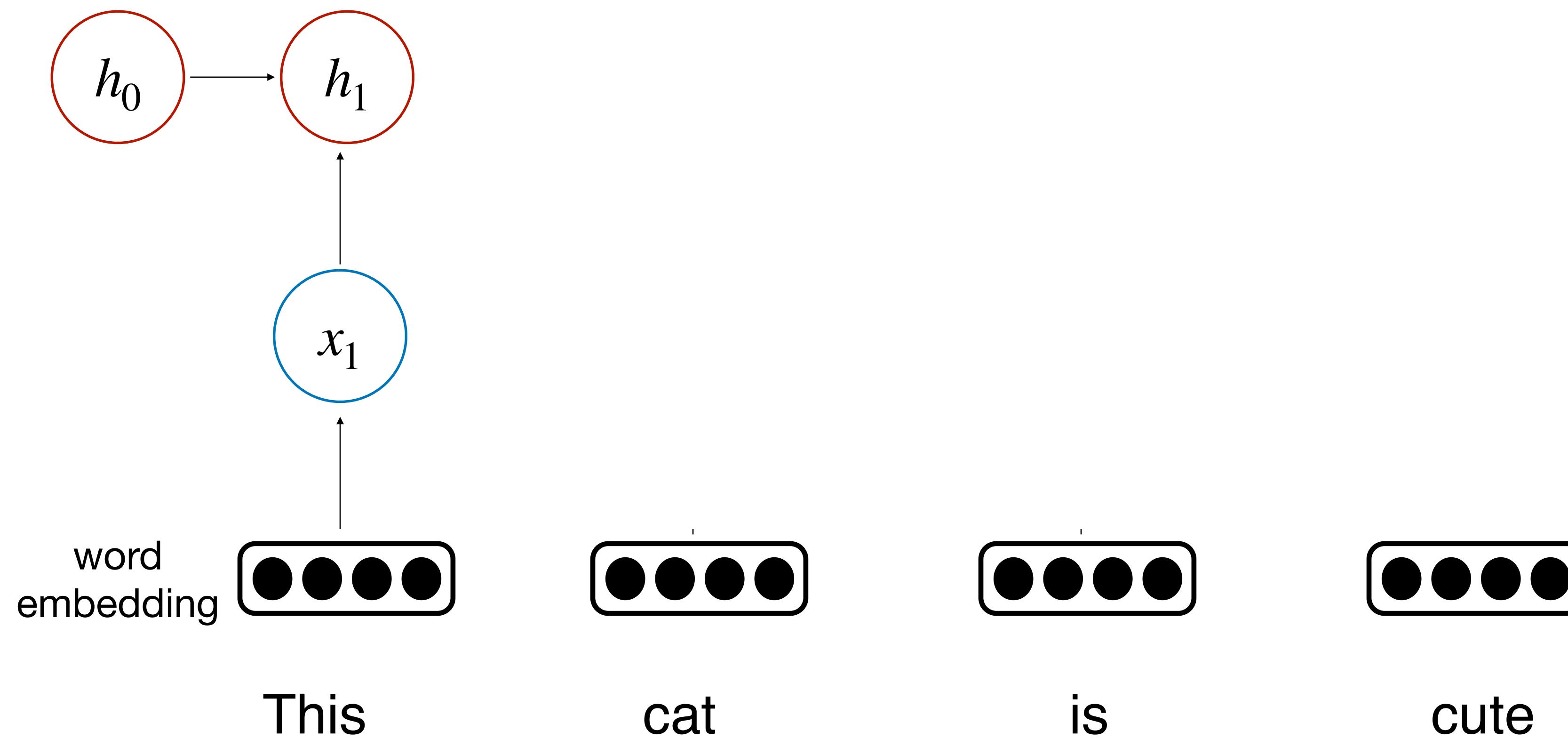
is



cute

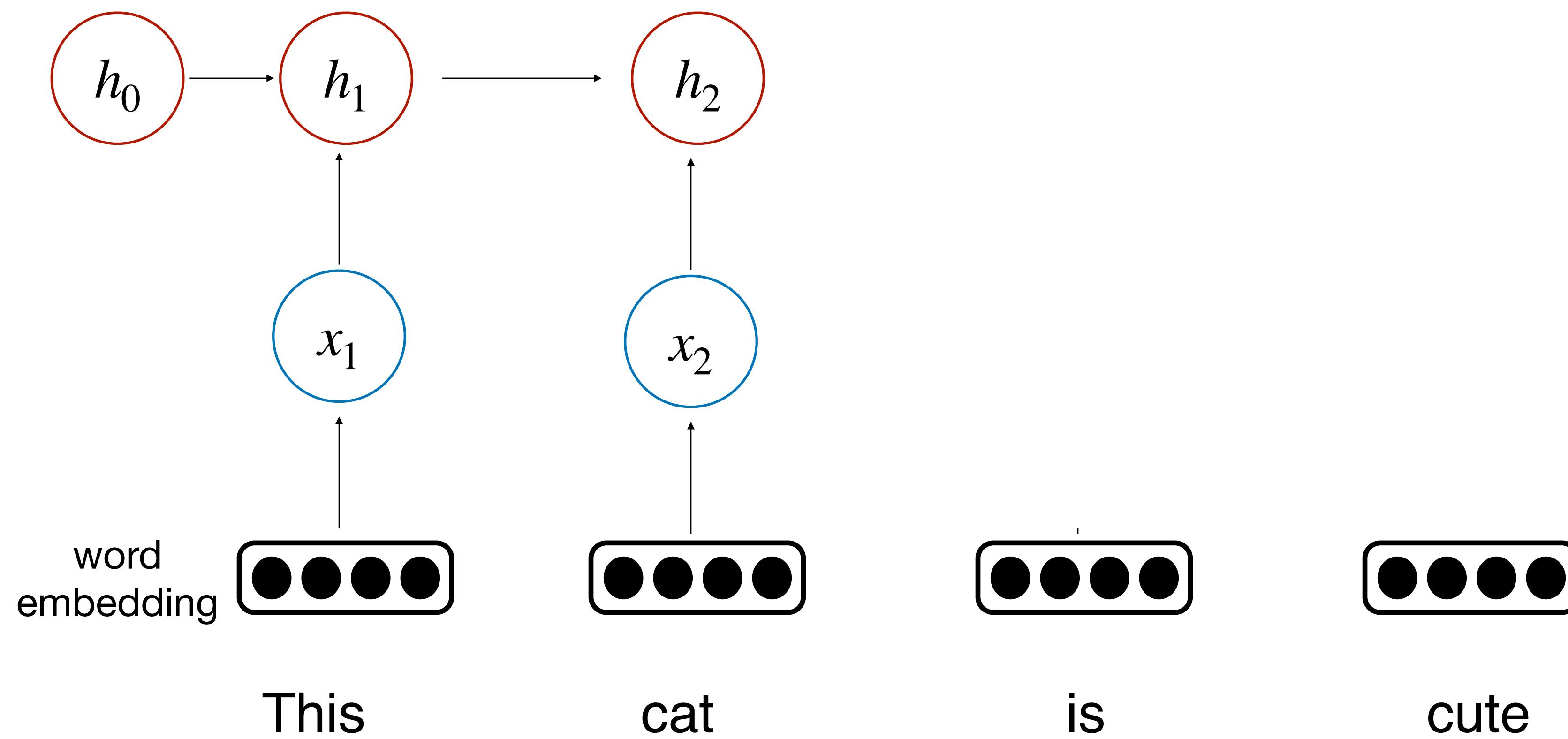
Encoder

Sentence: This cat is cute

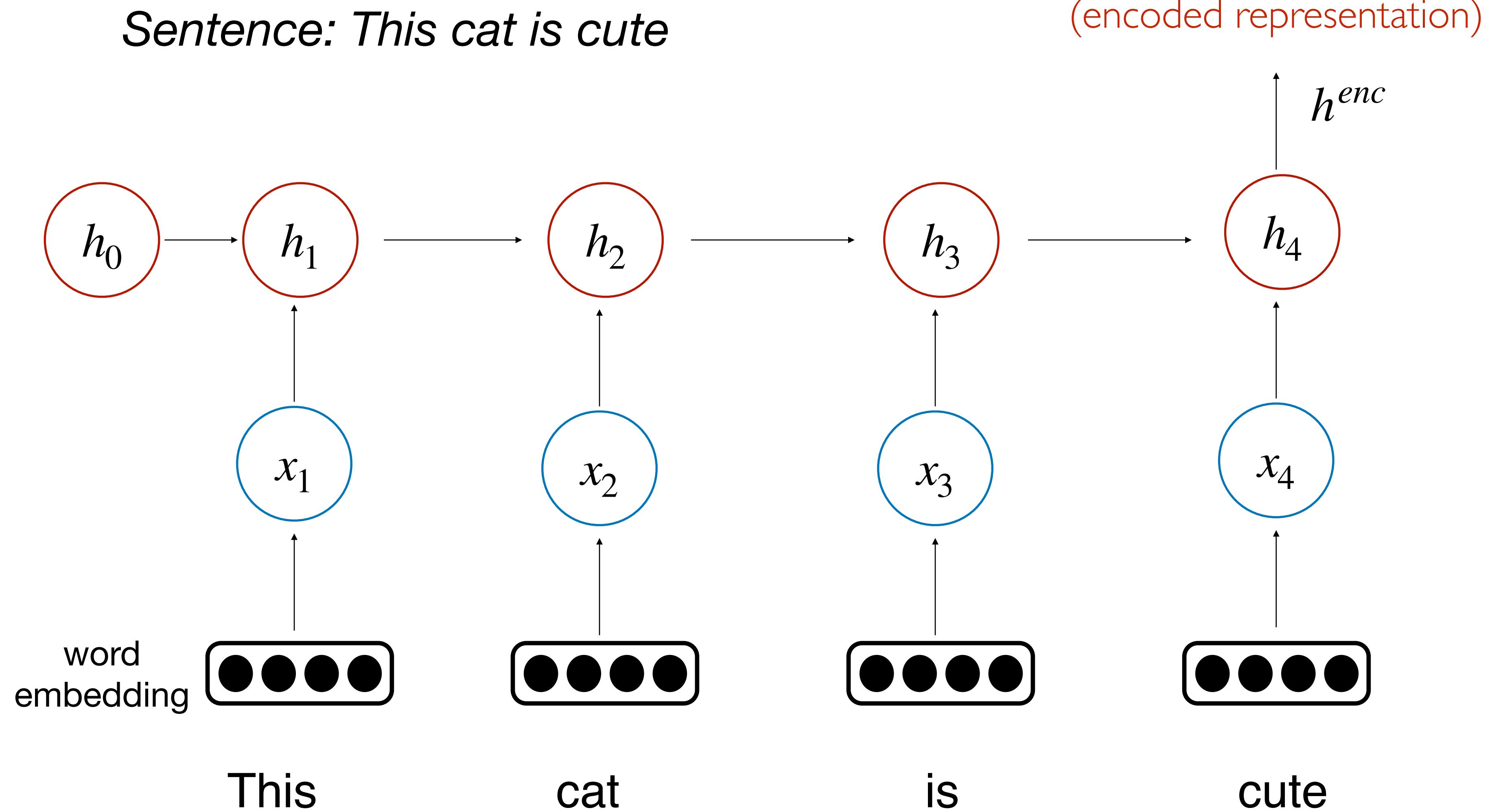


Encoder

Sentence: This cat is cute



Encoder

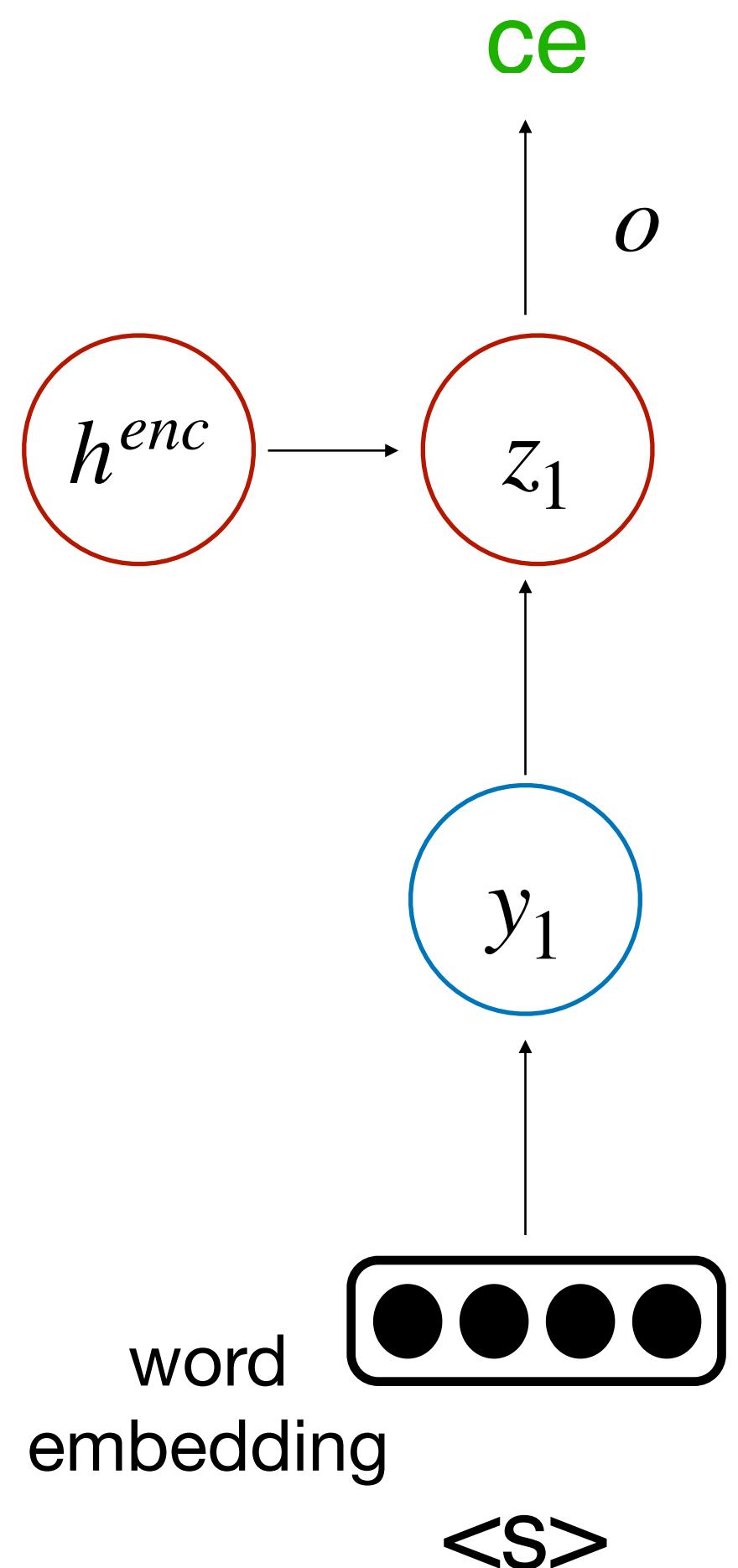


Decoder

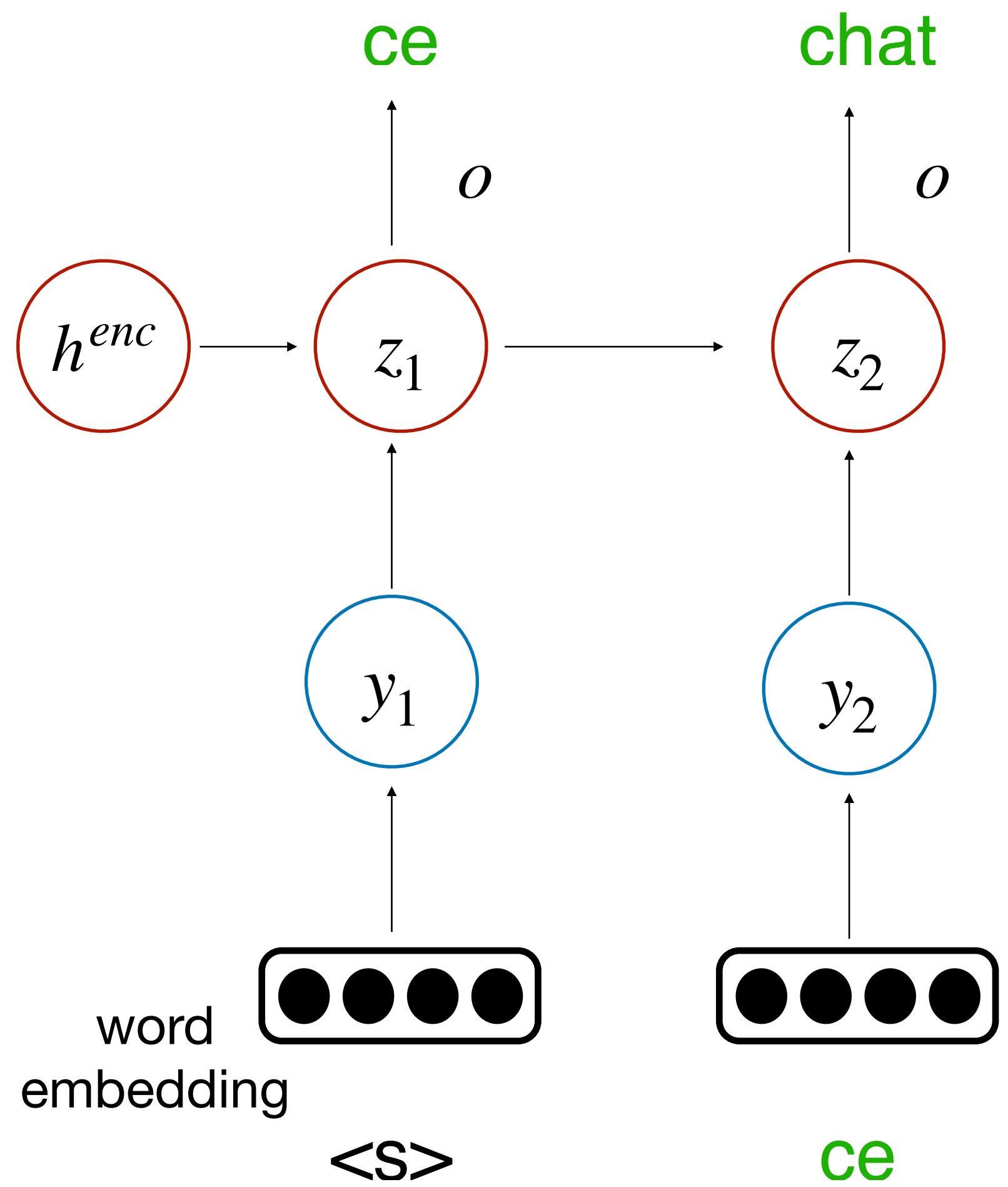
h^{enc}

word
embedding
 $\langle S \rangle$

Decoder

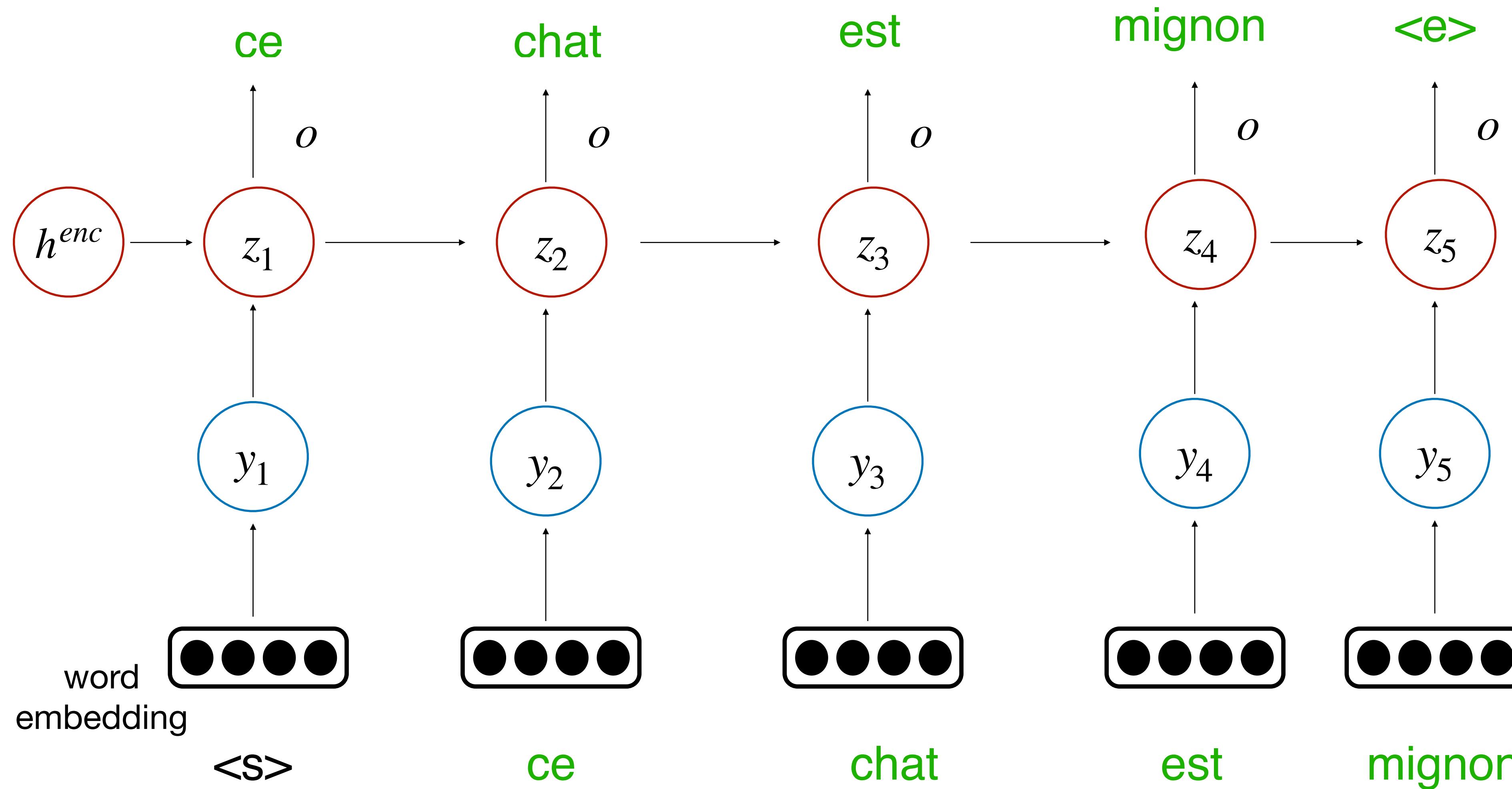


Decoder



Decoder

- A conditioned language model

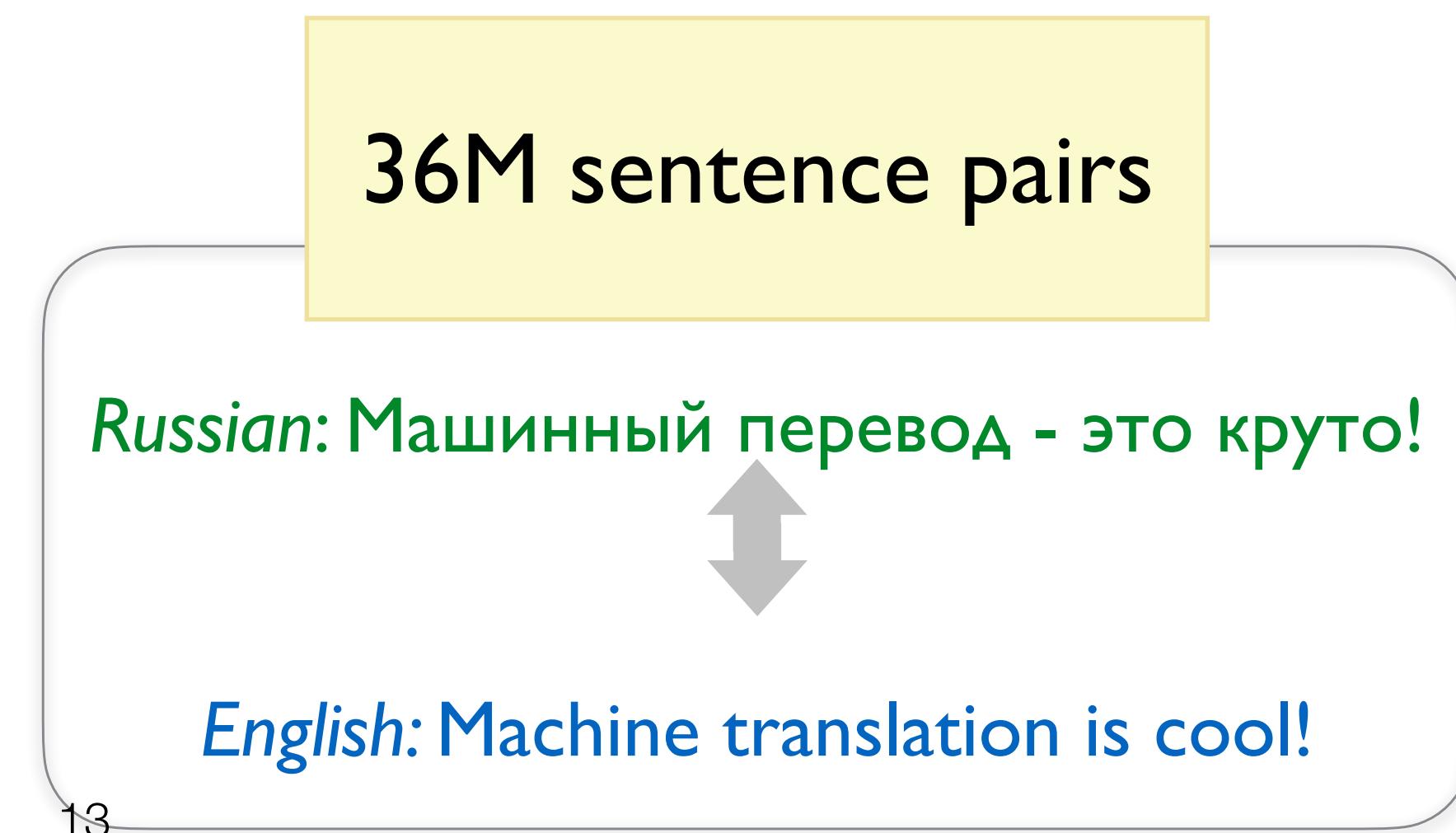


Seq2seq training

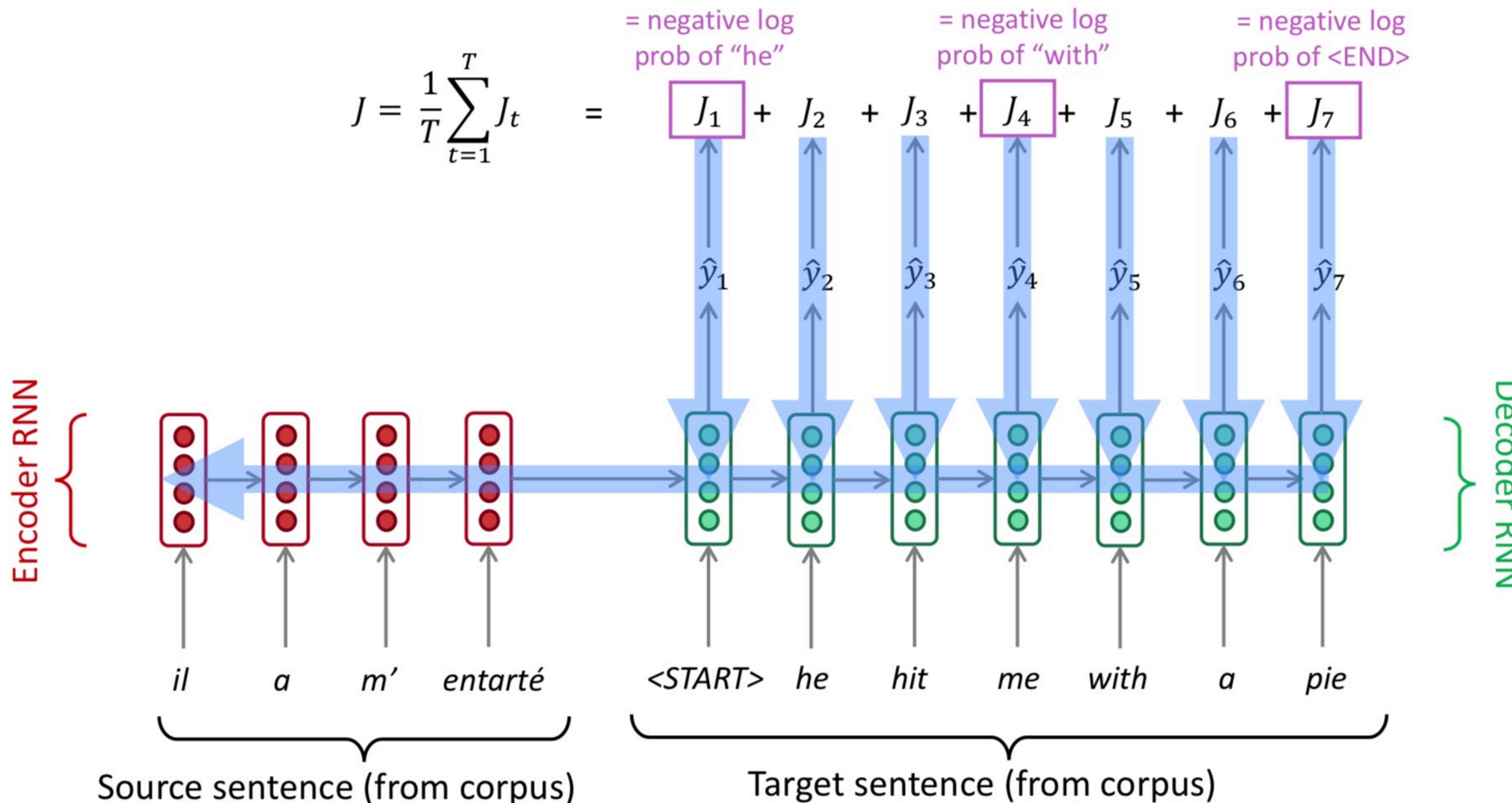
- ▶ Similar to training a language model!
- ▶ Minimize cross-entropy loss:

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

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



Seq2seq training

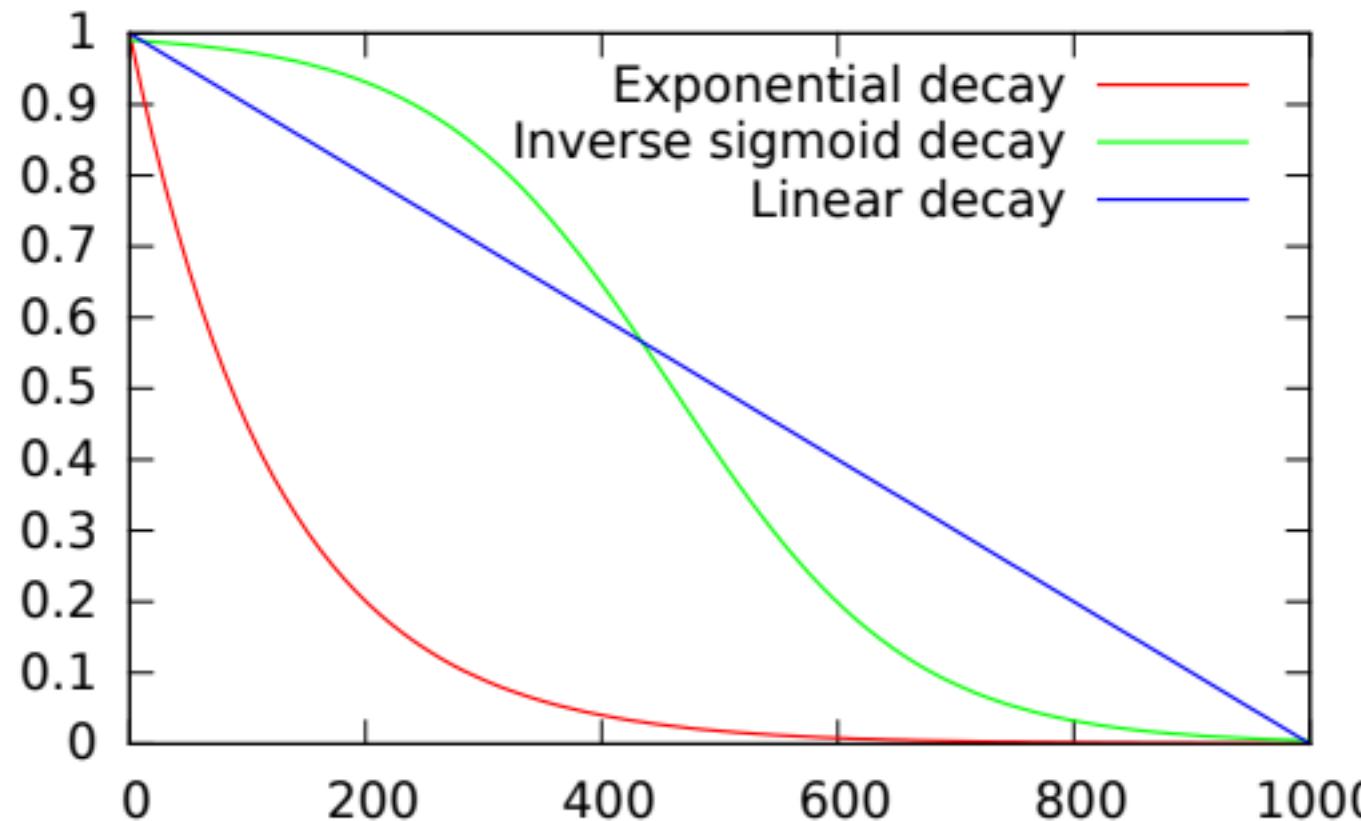


Seq2seq is optimized as a single system.
Backpropagation operates “end-to-end”.

Remember masking

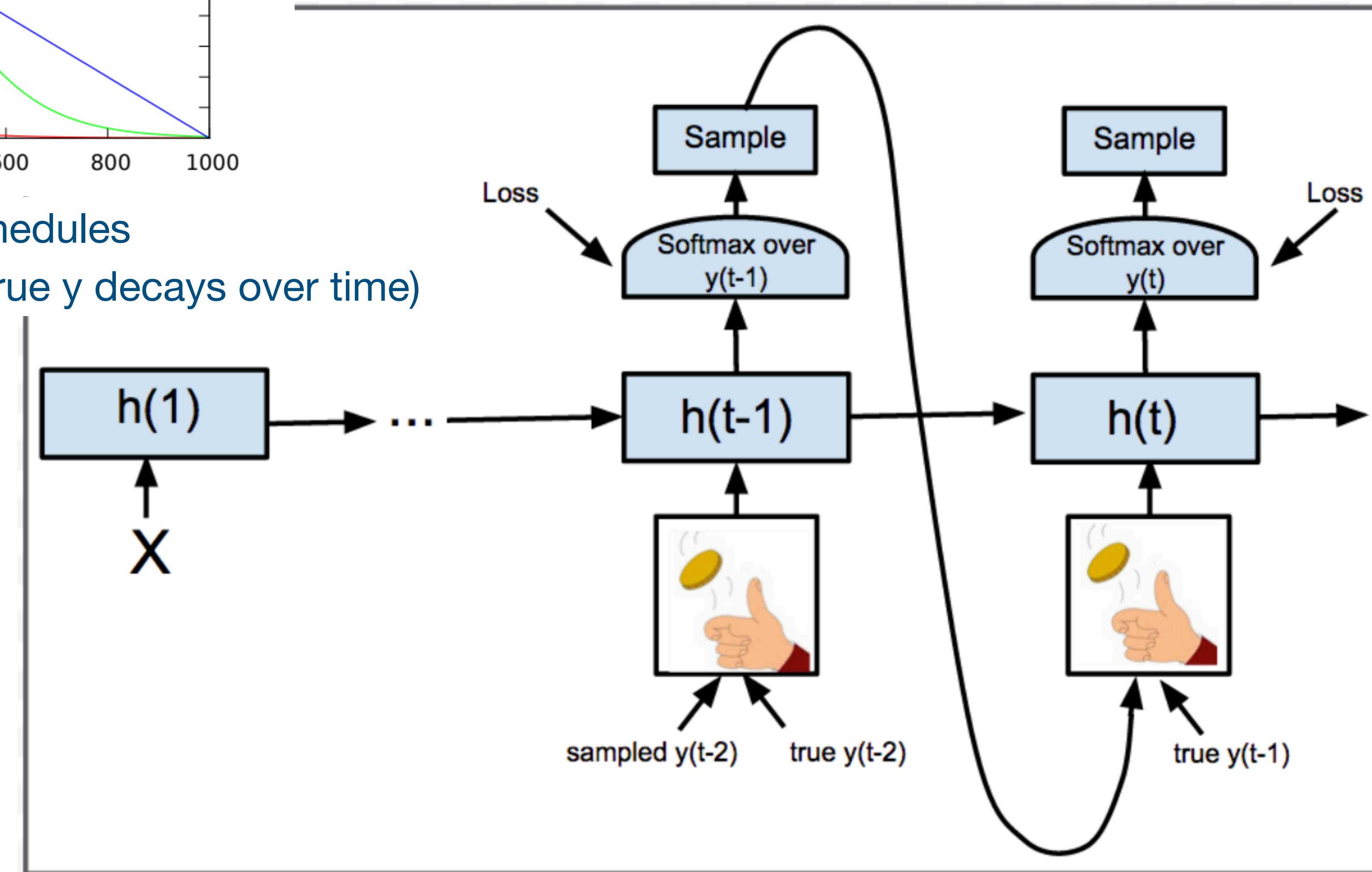
Use masking to help compute loss for batched sequences

Padded sequences						Length
1	1	1	1	0	0	4
1	0	0	0	0	0	1
1	1	1	1	1	1	6
1	1	1	0	0	0	3



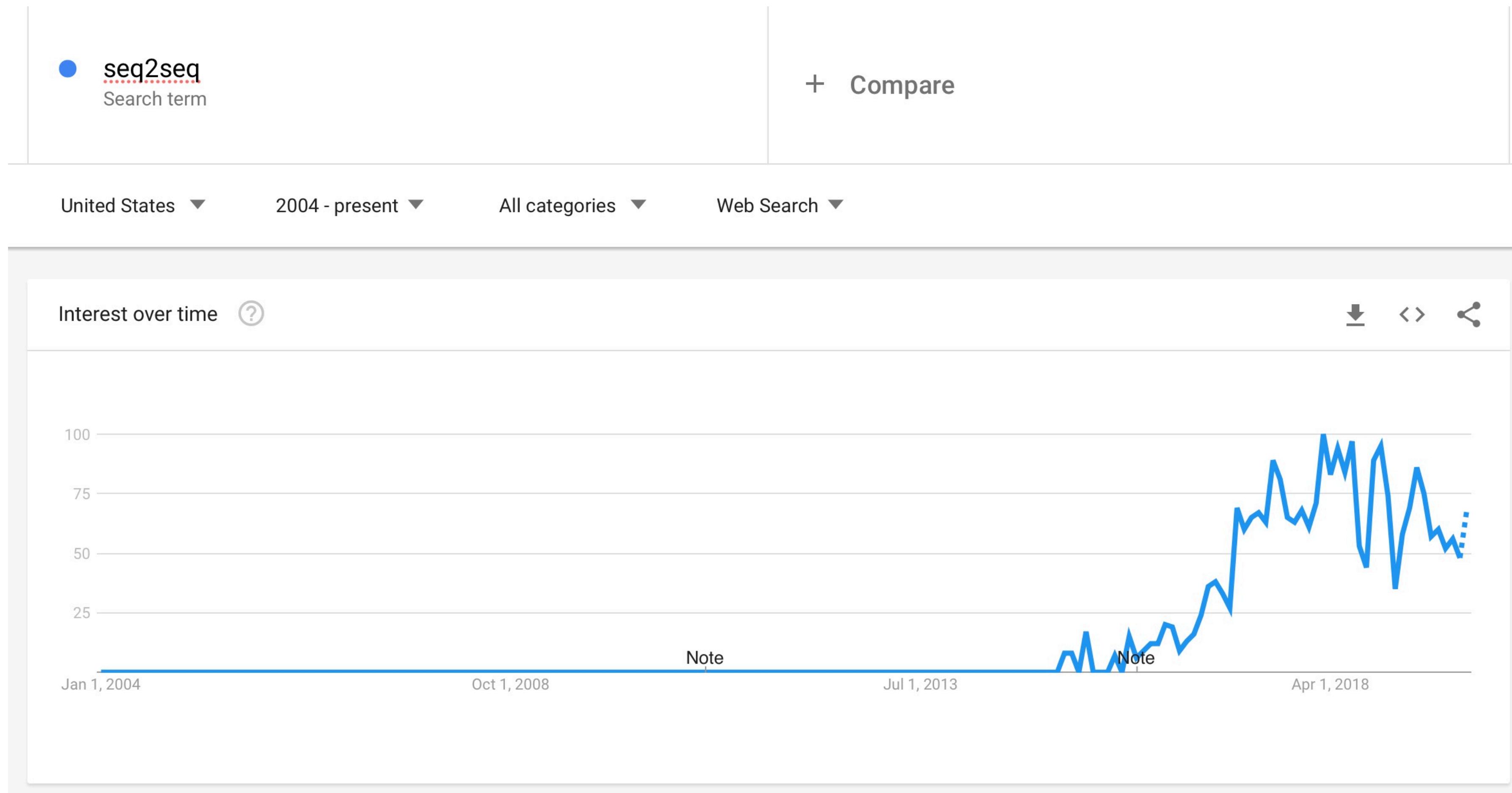
Possible decay schedules
(probability using true y decays over time)

Scheduled Sampling

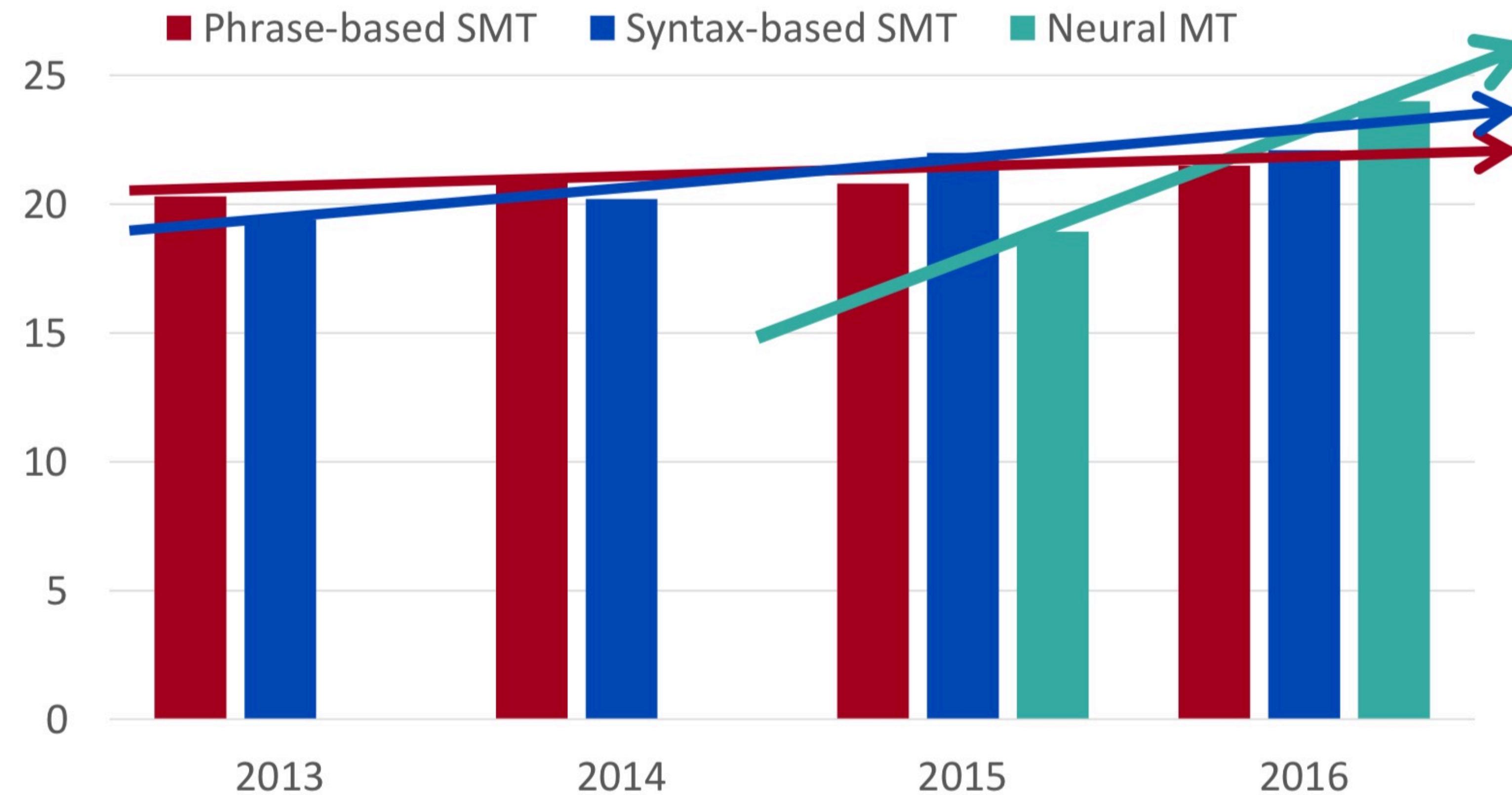


(figure credit: Bengio et al, 2015)

How seq2seq changed the MT landscape



MT Progress



(source: Rico Sennrich)

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

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

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	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%

(Wu et al., 2016)

NMT vs SMT

Pros

- ▶ Better performance
- ▶ Fluency
- ▶ Longer context
- ▶ Single NN optimized end-to-end
- ▶ Less engineering
- ▶ Works out of the box for many language pairs

Cons

- ▶ Requires more data and compute
- ▶ Less interpretable
- ▶ Hard to debug
- ▶ Uncontrollable
- ▶ Heavily dependent on data - could lead to unwanted biases
- ▶ More parameters

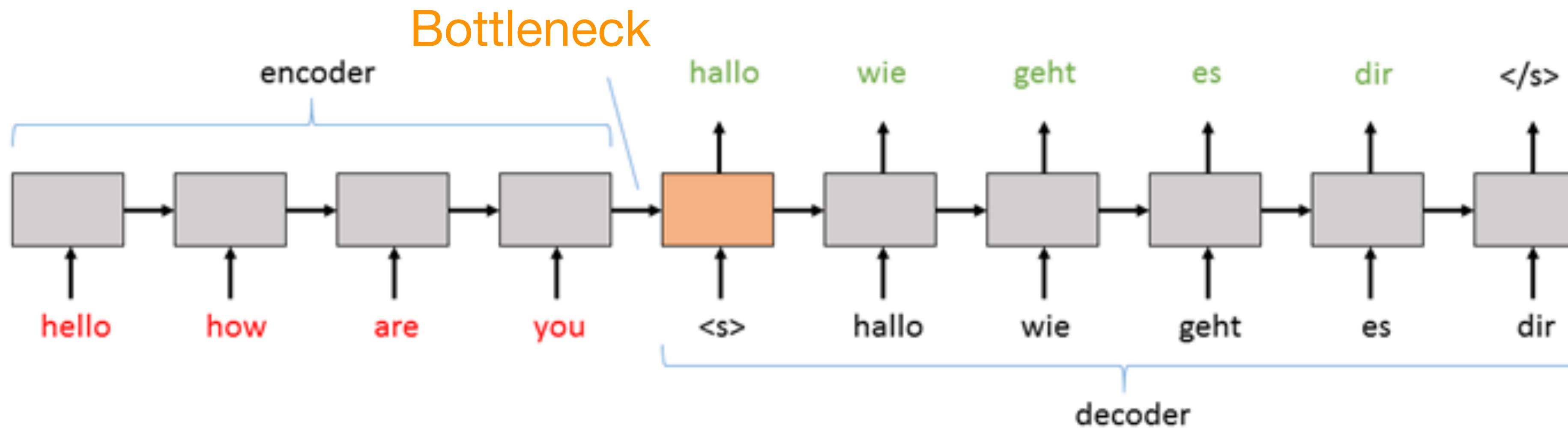
Seq2Seq for more than NMT

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

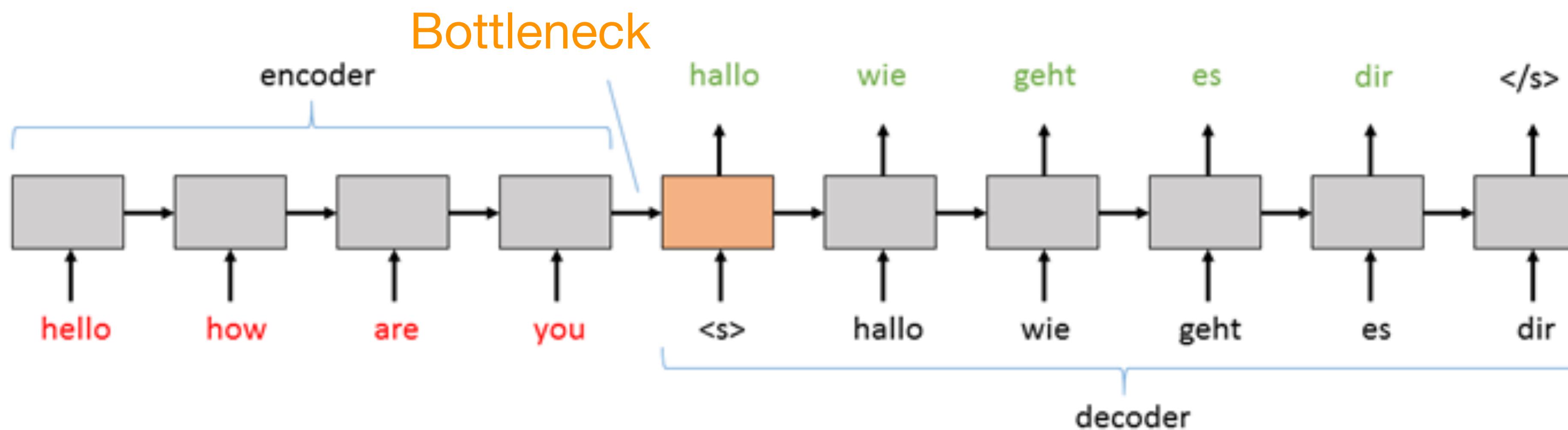
Task/Application	Input	Output
Speech Recognition	Speech Signal	Transcript
Image Captioning	Image	Text
Video Captioning	Video	Text

Issues with vanilla seq2seq



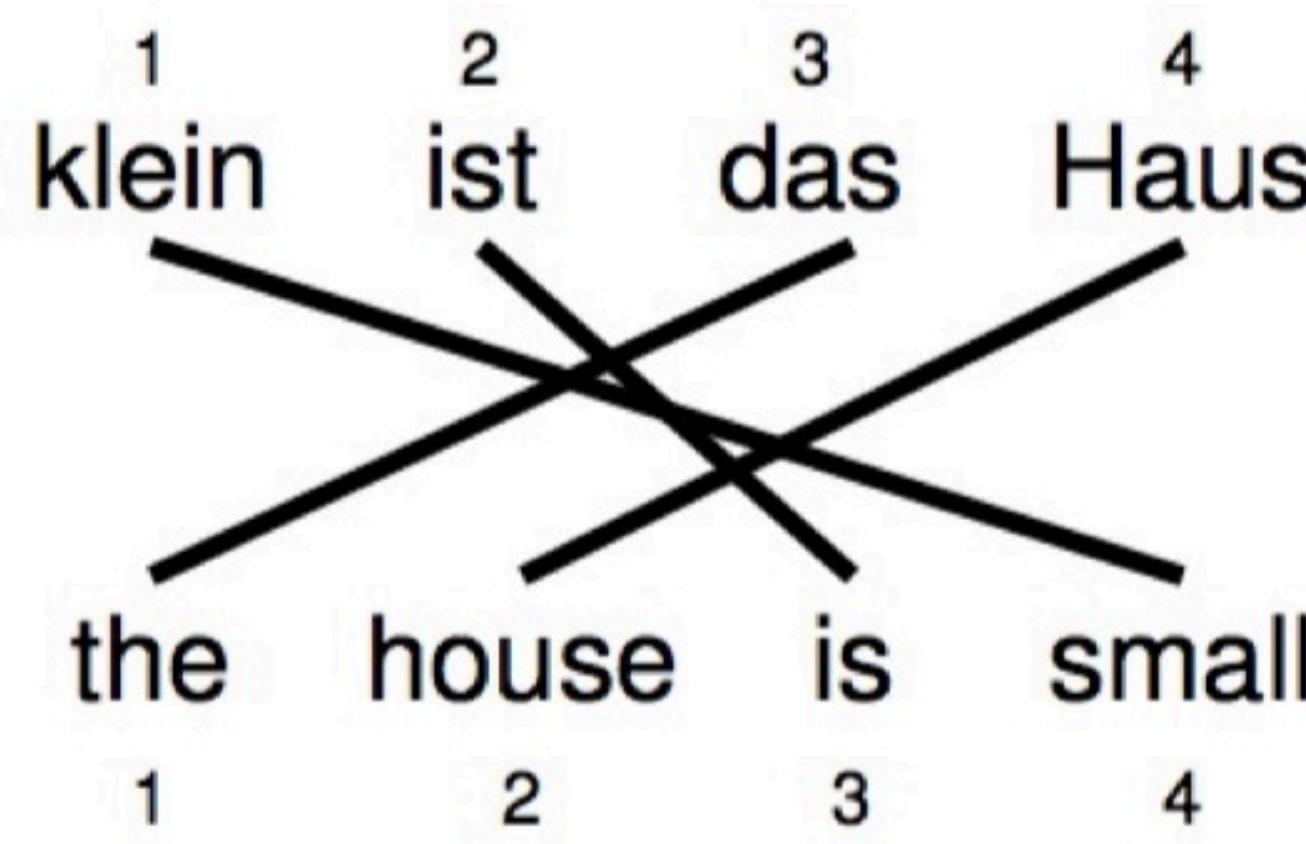
- ▶ A single encoding vector, h^{enc} , needs to capture **all the information** about source sentence
- ▶ Longer sequences can lead to vanishing gradients
- ▶ Overfitting

Issues with vanilla seq2seq

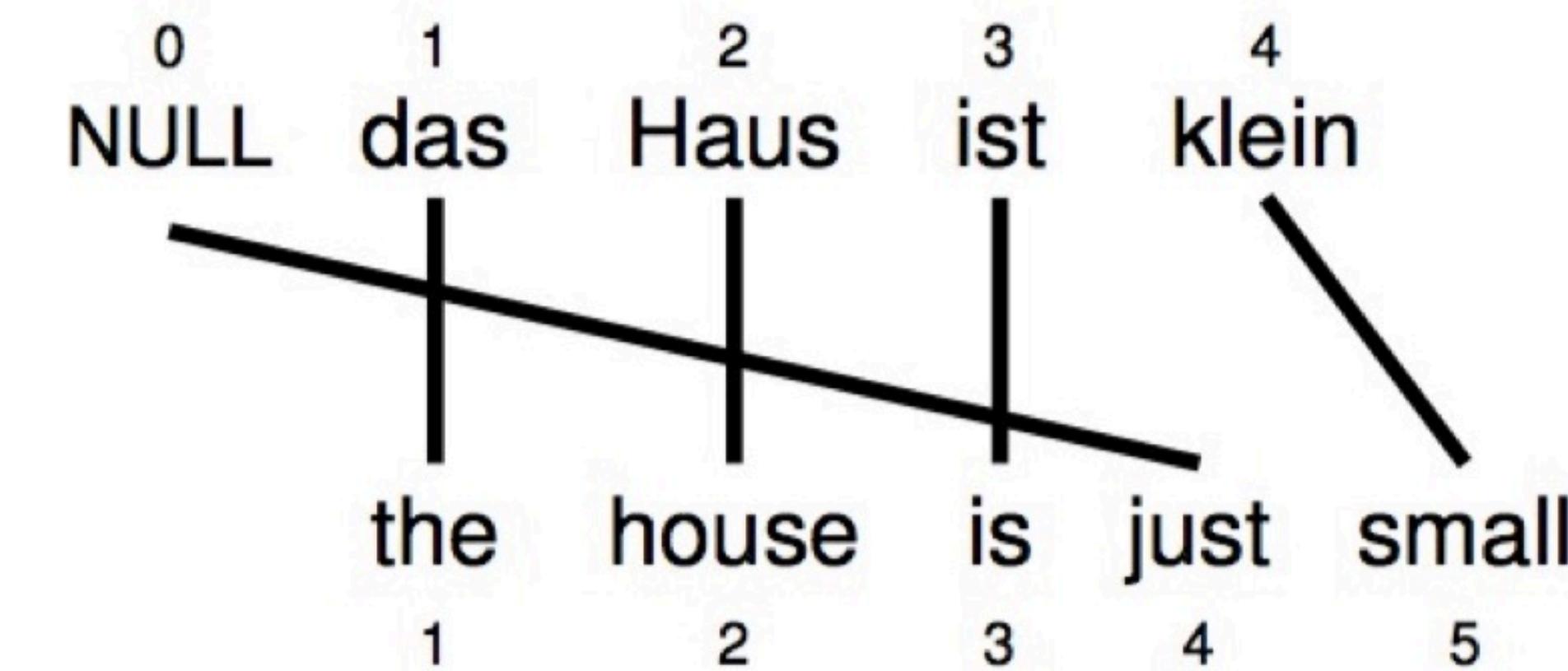


- ▶ A single encoding vector, h^{enc} , needs to capture **all the information** about source sentence
- ▶ **Longer sequences can lead to vanishing gradients**
- ▶ Overfitting

Alignments for MT



$$\mathbf{a} = (3, 4, 2, 1)^\top$$

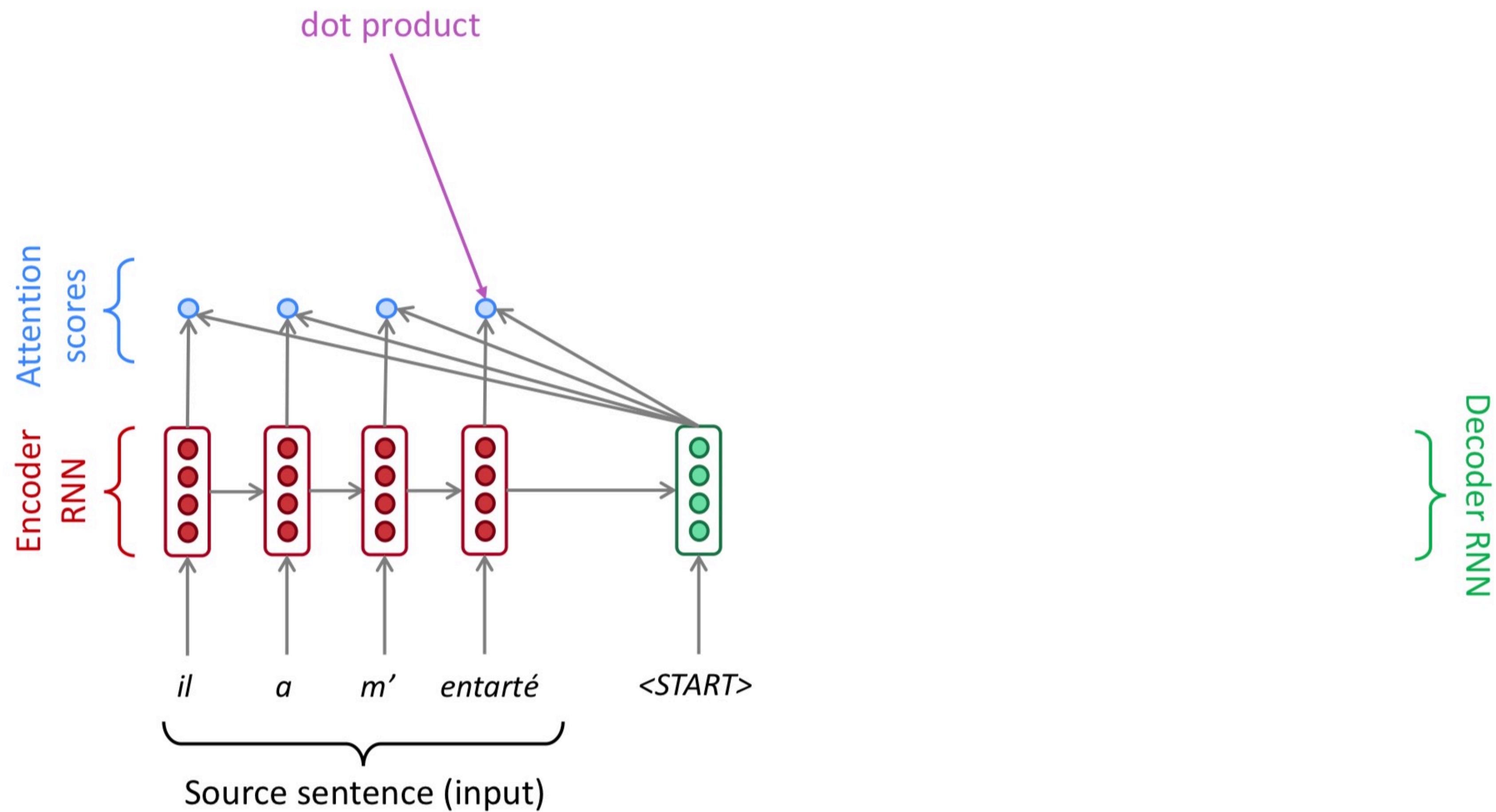


$$\mathbf{a} = (1, 2, 3, 0, 4)^\top$$

Attention

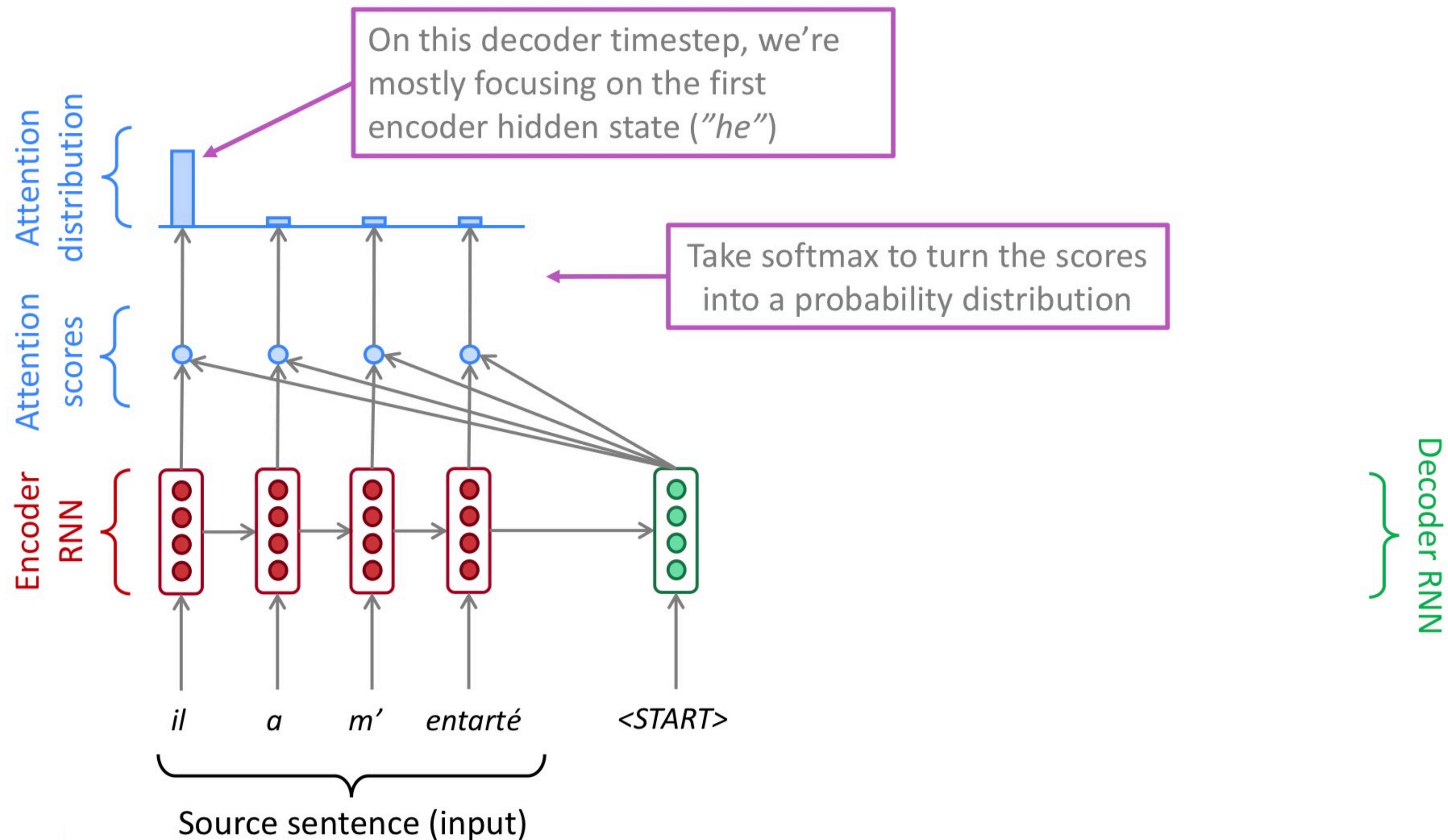
- ▶ The neural MT equivalent of alignment models
- ▶ **Key idea:** At each time step during decoding, **focus on a particular part** of source sentence
 - ▶ This depends on the **decoder's current hidden state** (i.e. notion of what you are trying to decode)
 - ▶ Usually implemented as a **probability**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

Seq2seq with attention



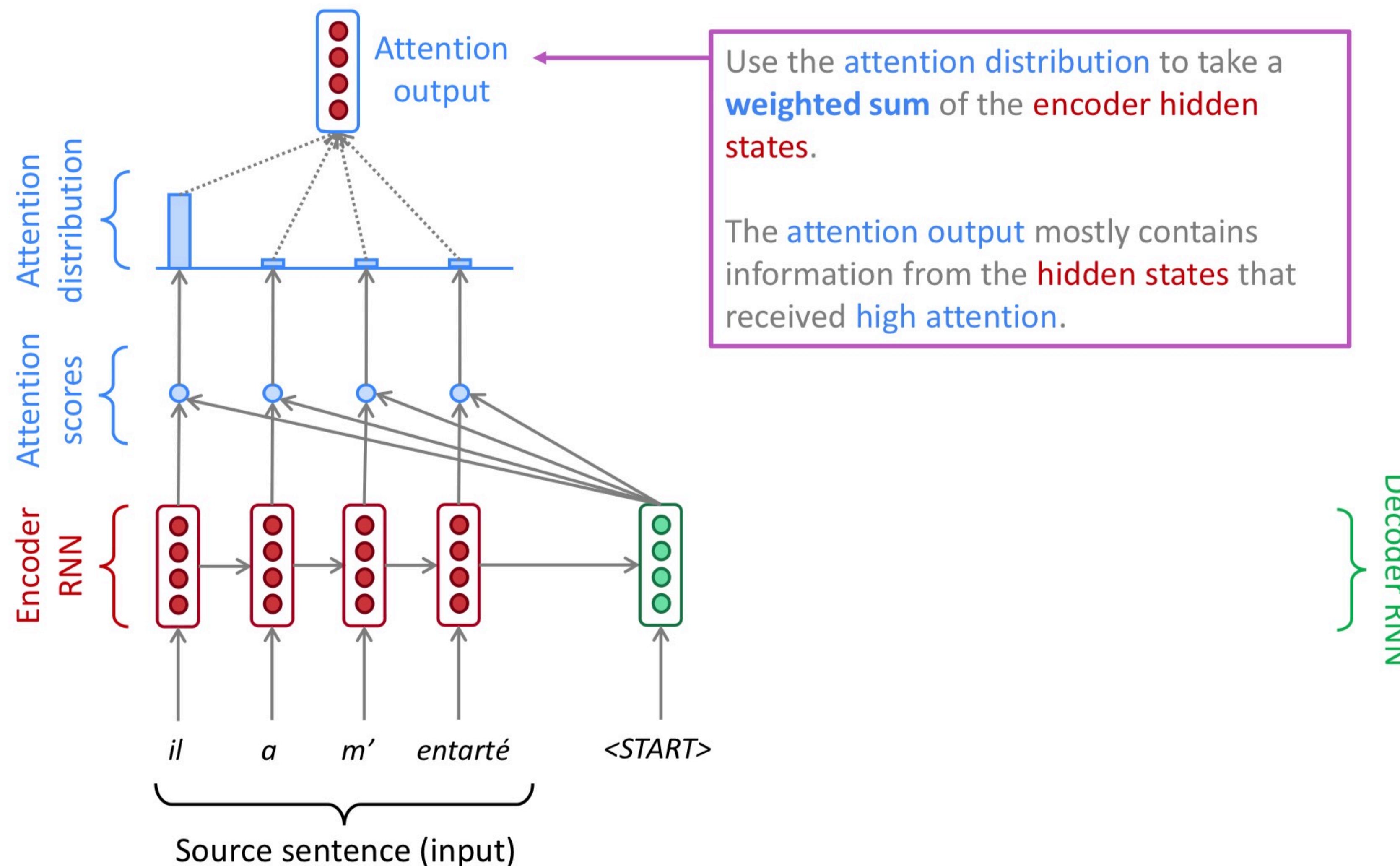
(slide credit: Abigail See)

Seq2seq with attention



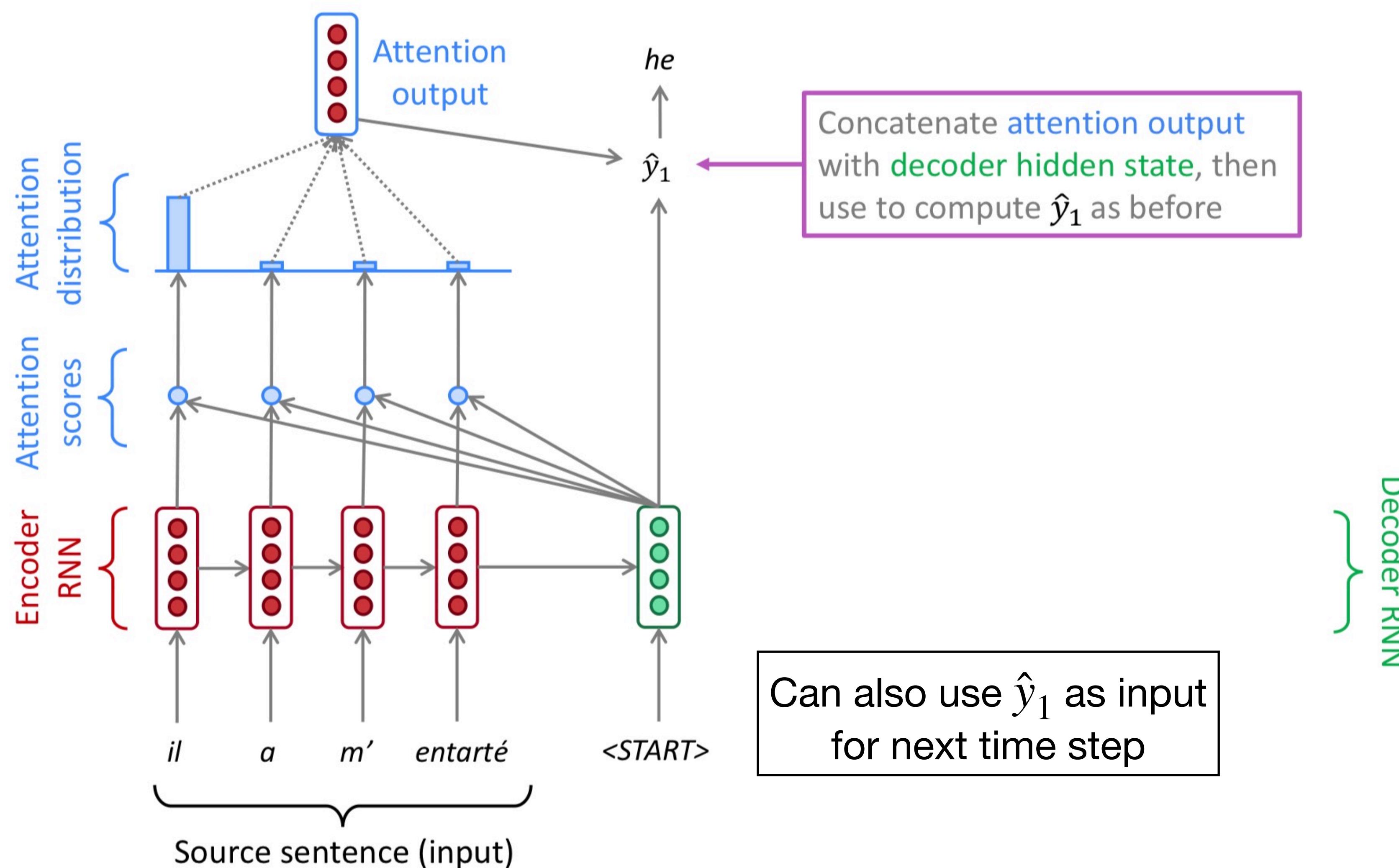
(slide credit: Abigail See)

Seq2seq with attention

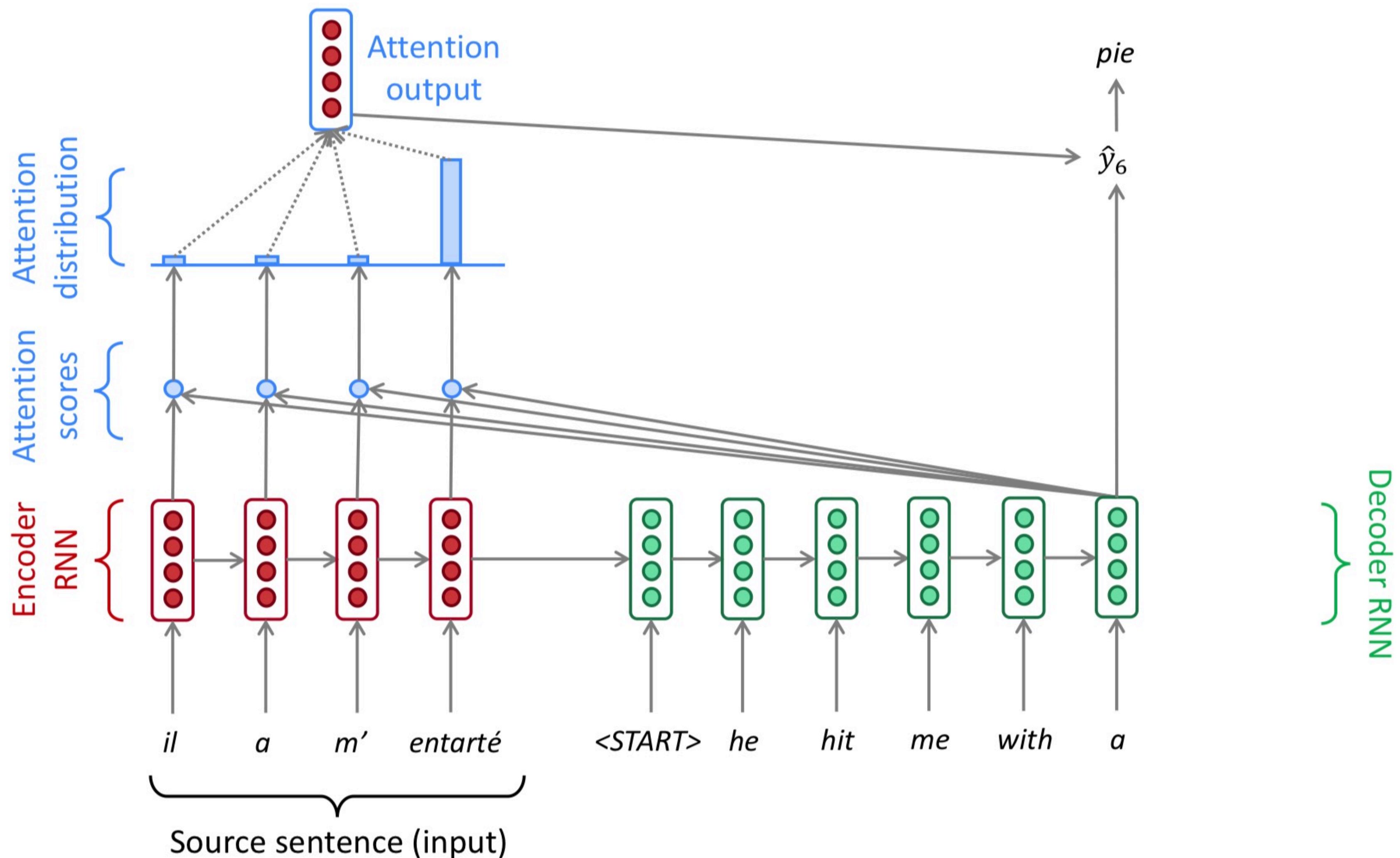


(slide credit: Abigail See)

Seq2seq with attention



Seq2seq with attention



(slide credit: Abigail See)

Computing attention

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

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

- ▶ 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, \dots, 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:

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

More flexible
than dot-product
(W is trainable)

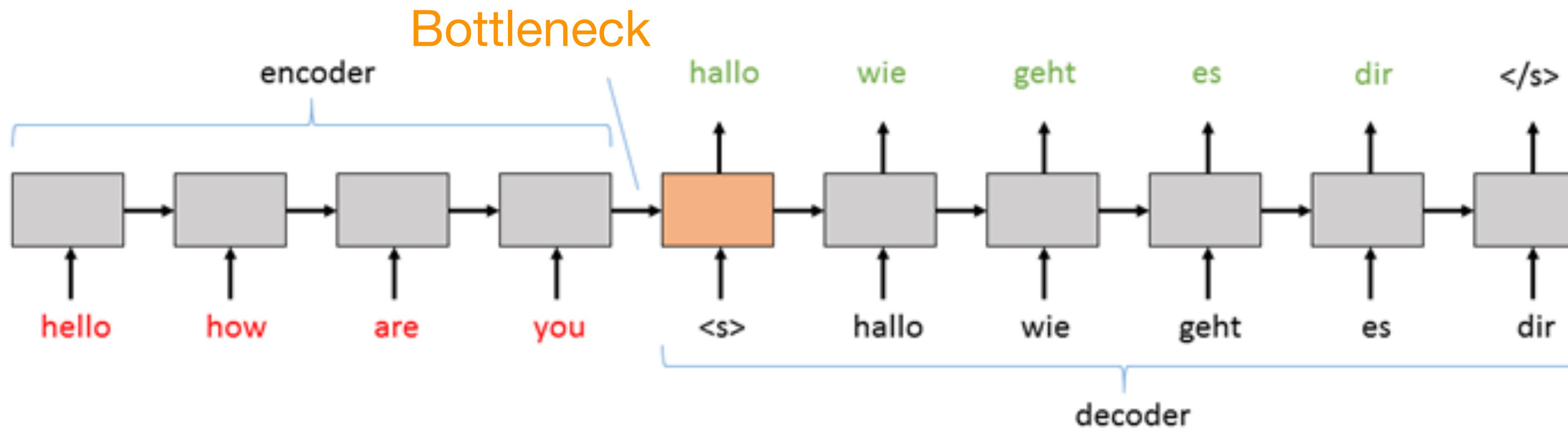
3. Additive attention (essentially MLP):

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

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

Perform better for
larger dimensions

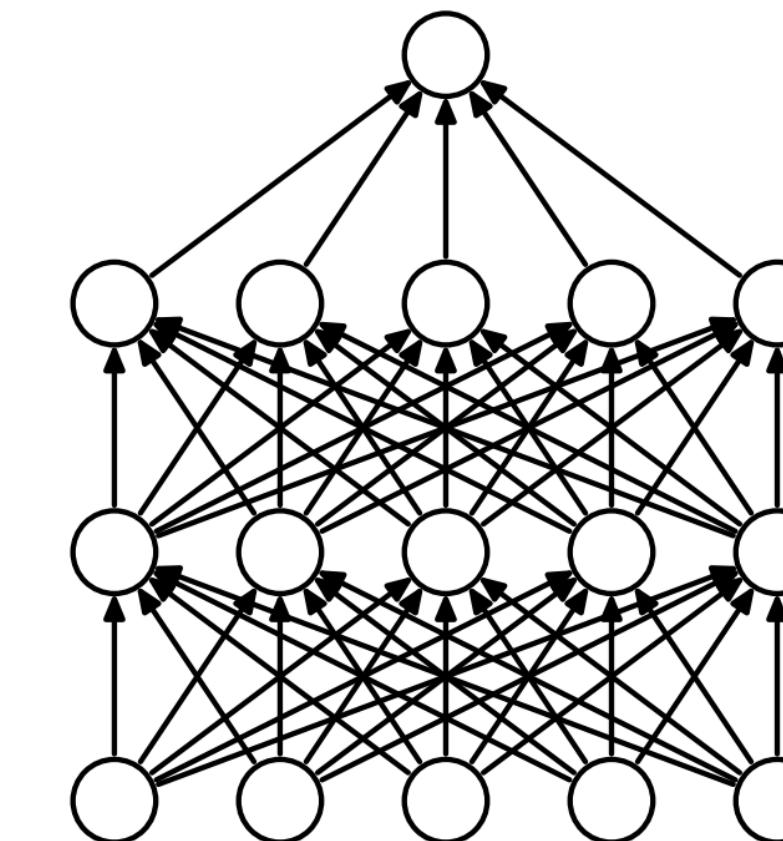
Issues with vanilla seq2seq



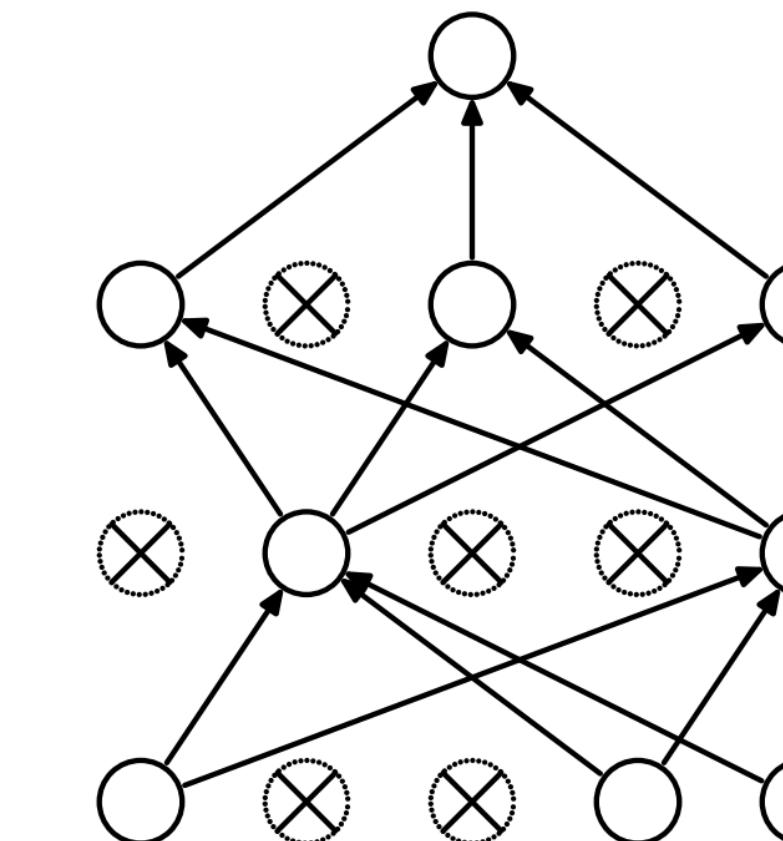
- ▶ A single encoding vector, h^{enc} , needs to capture **all the information** about source sentence
- ▶ Longer sequences can lead to vanishing gradients
- ▶ **Overfitting**

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.

Handling large vocabularies

- ▶ Softmax can be expensive for large vocabularies

$$P(y_i) = \frac{\exp(w_i \cdot h + b_i)}{\sum_{j=1}^{|V|} \exp(w_j \cdot h + b_j)}$$

← Expensive to compute

- ▶ English vocabulary size: 10K to 100K

Approximate Softmax

- ▶ Negative Sampling
- ▶ Structured softmax
- ▶ Embedding prediction

Negative Sampling

- Softmax is expensive when vocabulary size is large

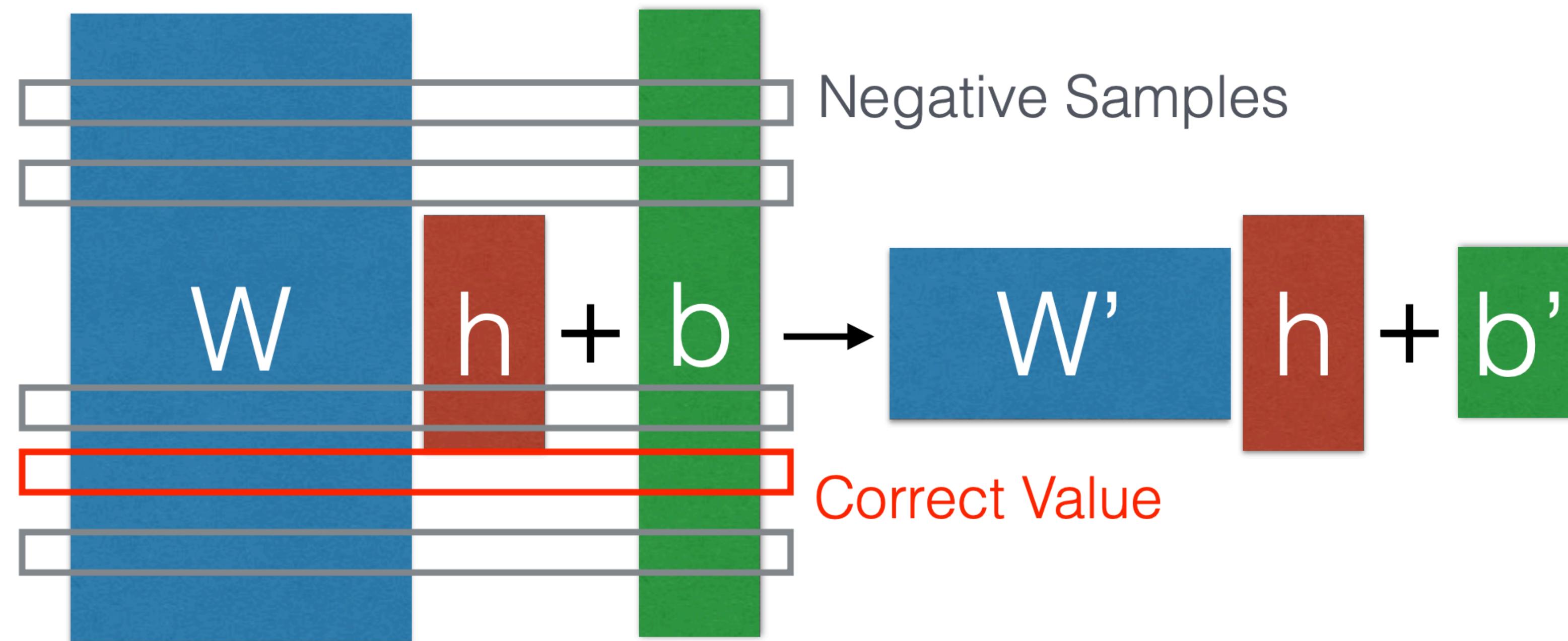
$$p = \text{softmax}(W h + b)$$

The diagram illustrates the softmax formula $p = \text{softmax}(W h + b)$. It consists of five vertical bars of different colors and widths. From left to right, they represent the input vector h (grey), the weight matrix W (blue), the bias vector b (green), the addition operator $+$ (white), and the softmax function softmax (orange). The width of each bar corresponds to its dimension in the formula.

(figure credit: Graham Neubig)

Negative Sampling

- Sample just a subset of the vocabulary for negative
- Saw simple negative sampling in word2vec (Mikolov 2013)



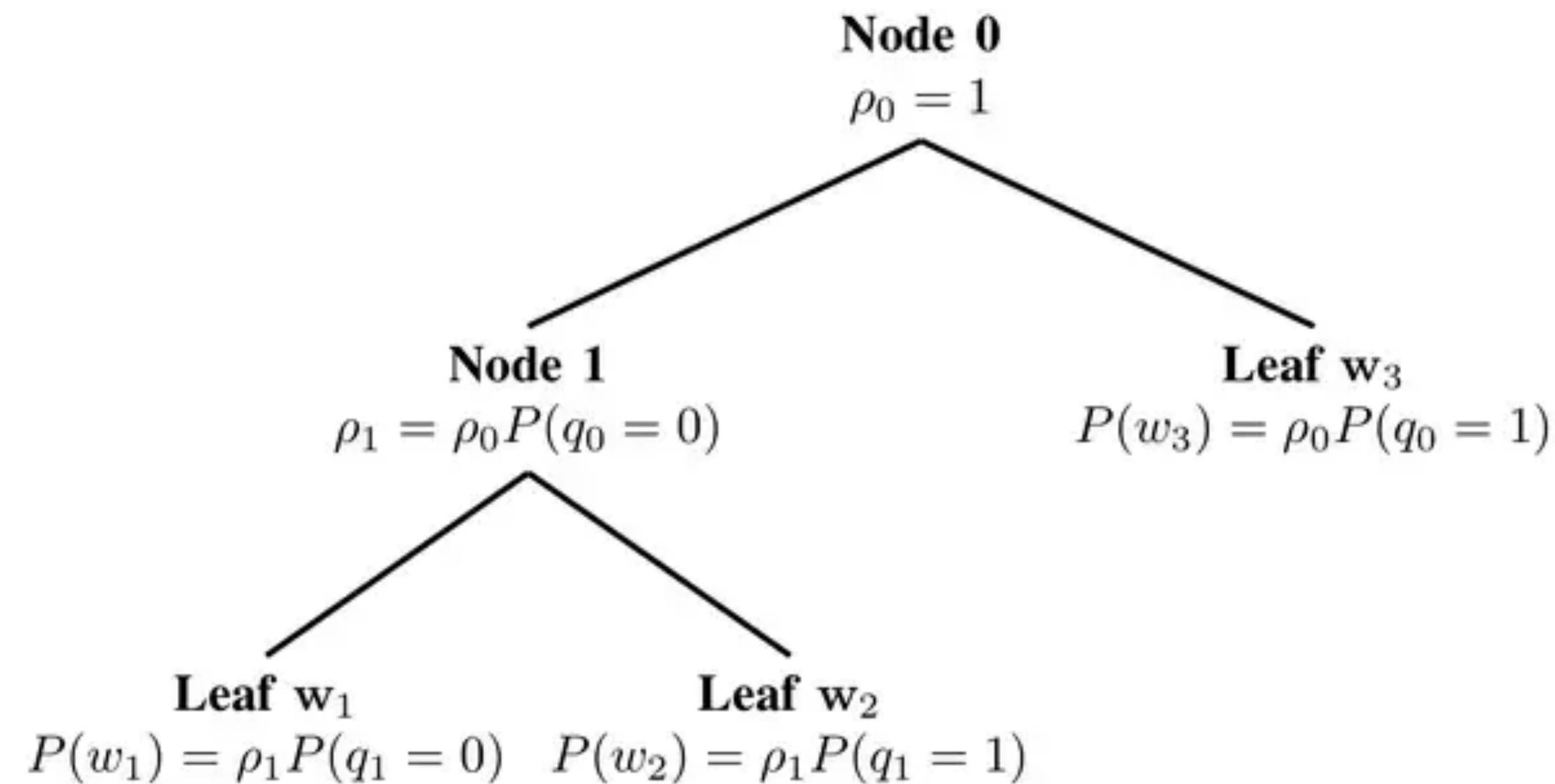
Other ways to sample:

Importance Sampling (Bengio and Senecal 2003)

Noise Contrastive Estimation (Mnih & Teh 2012)

Hierarchical softmax

(Morin and Bengio 2005)



(figure credit: [Quora](#))

Class based softmax

- ▶ Two-layer: cluster words into **classes**, predict class and then predict word.

$$P(c|h) = \text{softmax}(W_c h + b_c)$$

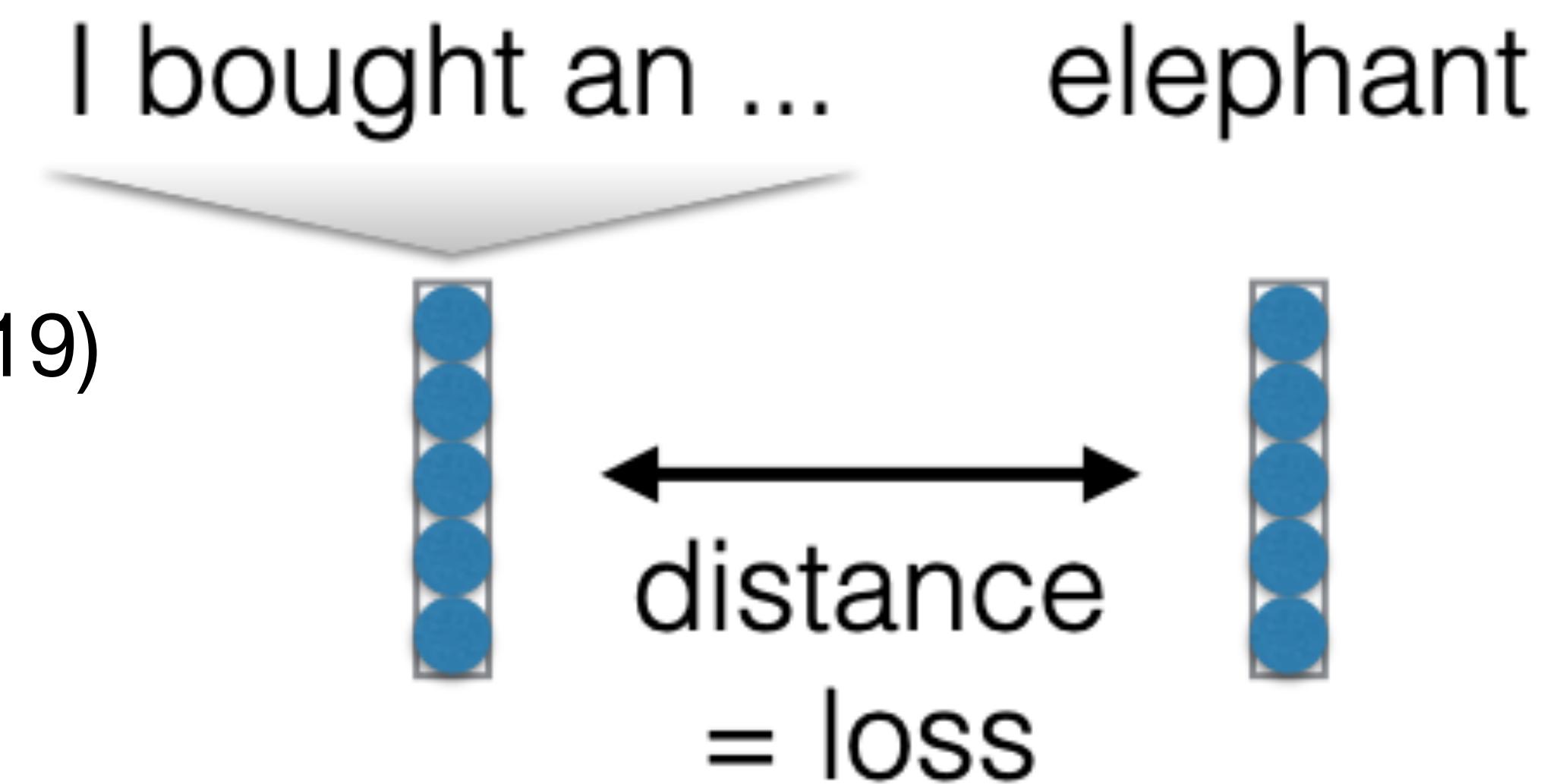
$$P(x|c,h) = \text{softmax}(W_x h + b_x)$$

(figure credit: Graham Neubig)

- ▶ Clusters can be based on *frequency*, *random*, or *word contexts*.

Embedding prediction

- ▶ Directly predict embeddings of outputs themselves
- ▶ What loss to use? (Kumar and Tsvetkov 2019)
 - ▶ L2? Cosine?
 - ▶ Von-Mises Fisher distribution loss, make embeddings close on the unit ball



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

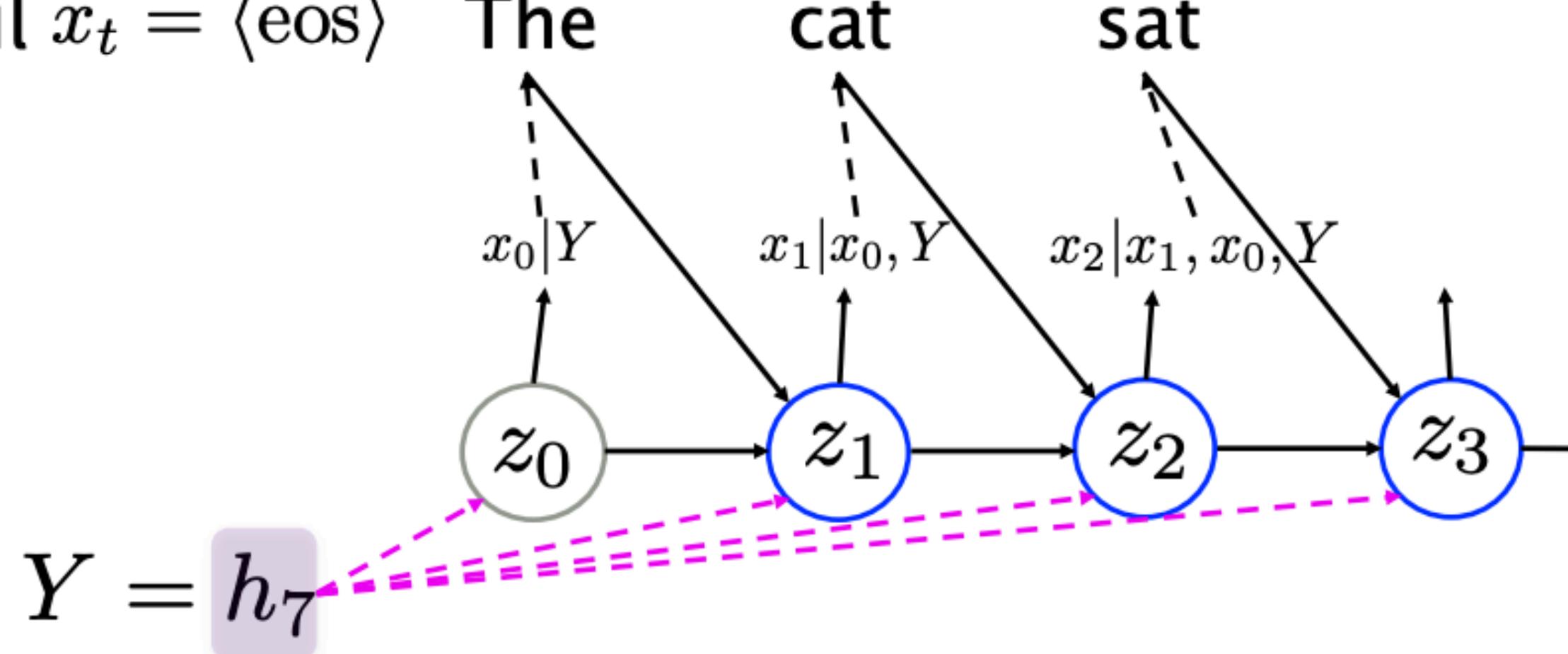
- ▶ Ancestral sampling
- ▶ Greedy decoding
- ▶ Exhaustive search
- ▶ Beam search

Ancestral Sampling

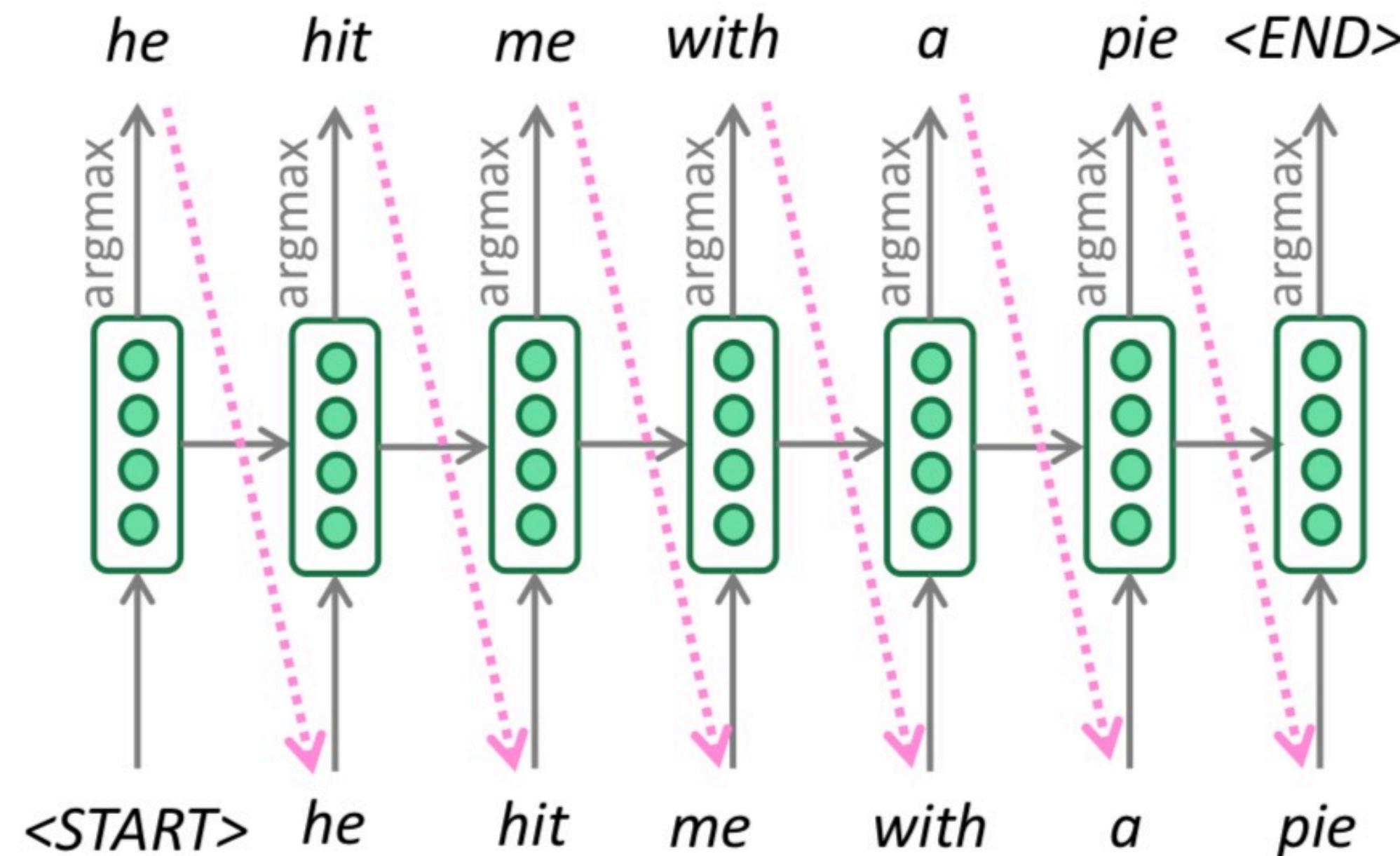
- Randomly sample words one by one
- Provides diverse output (high variance)

One symbol at a time from $\tilde{x}_t \sim x_t | x_{t-1}, \dots, x_1, Y$

Until $\tilde{x}_t = \langle \text{eos} \rangle$



Greedy decoding



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

Exhaustive search?

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

Recall: Beam search (a middle ground)

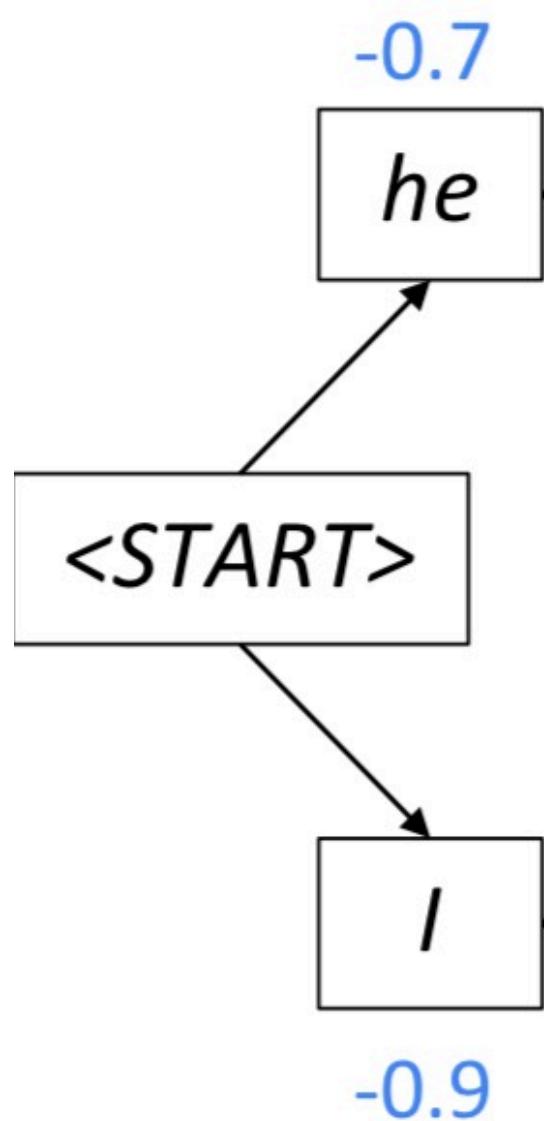
- ▶ **Key idea:** At every step, keep track of the **k most probable** partial translations (hypotheses)
- ▶ Score of each hypothesis = log probability

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

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

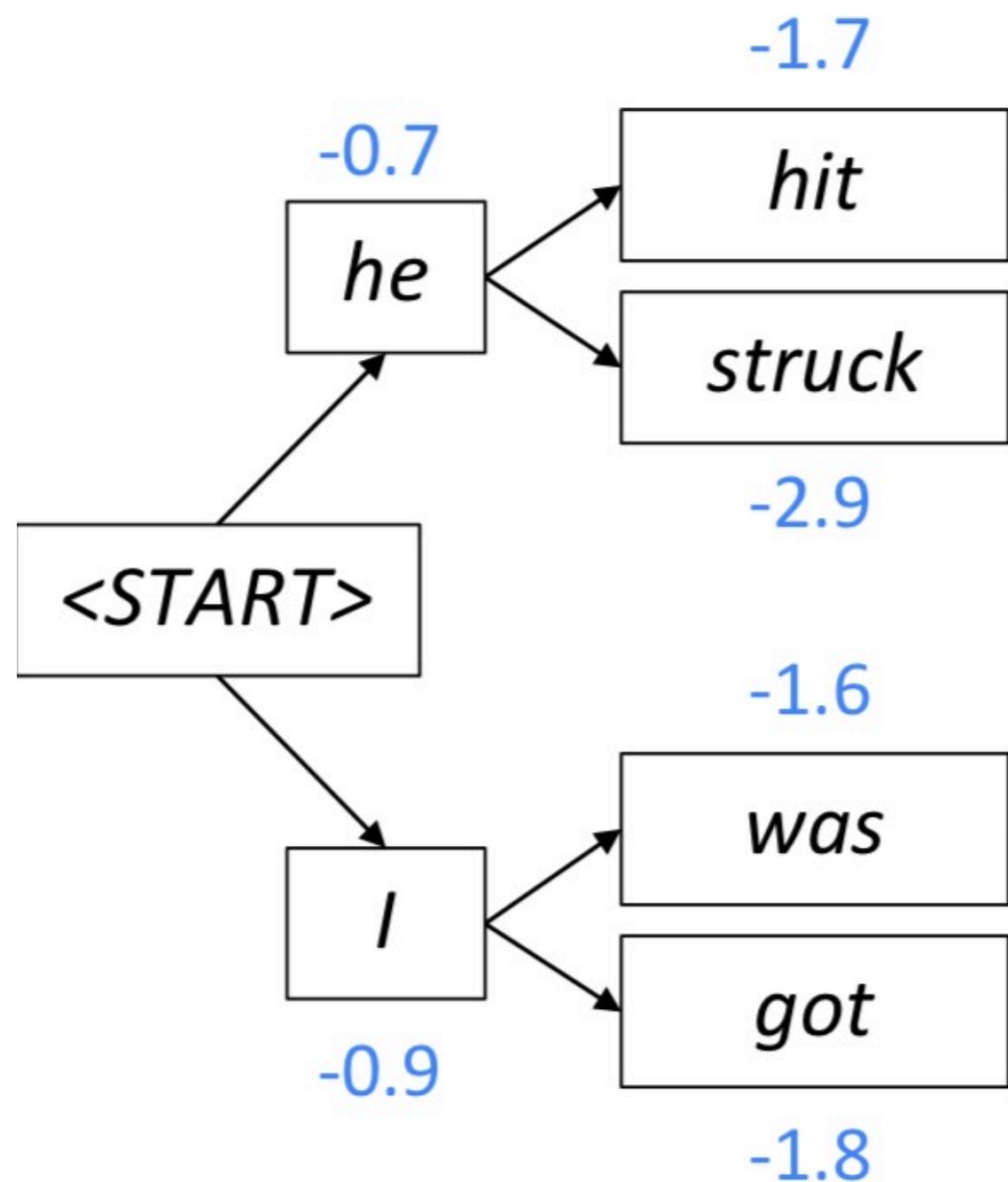
Beam decoding

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



Beam decoding

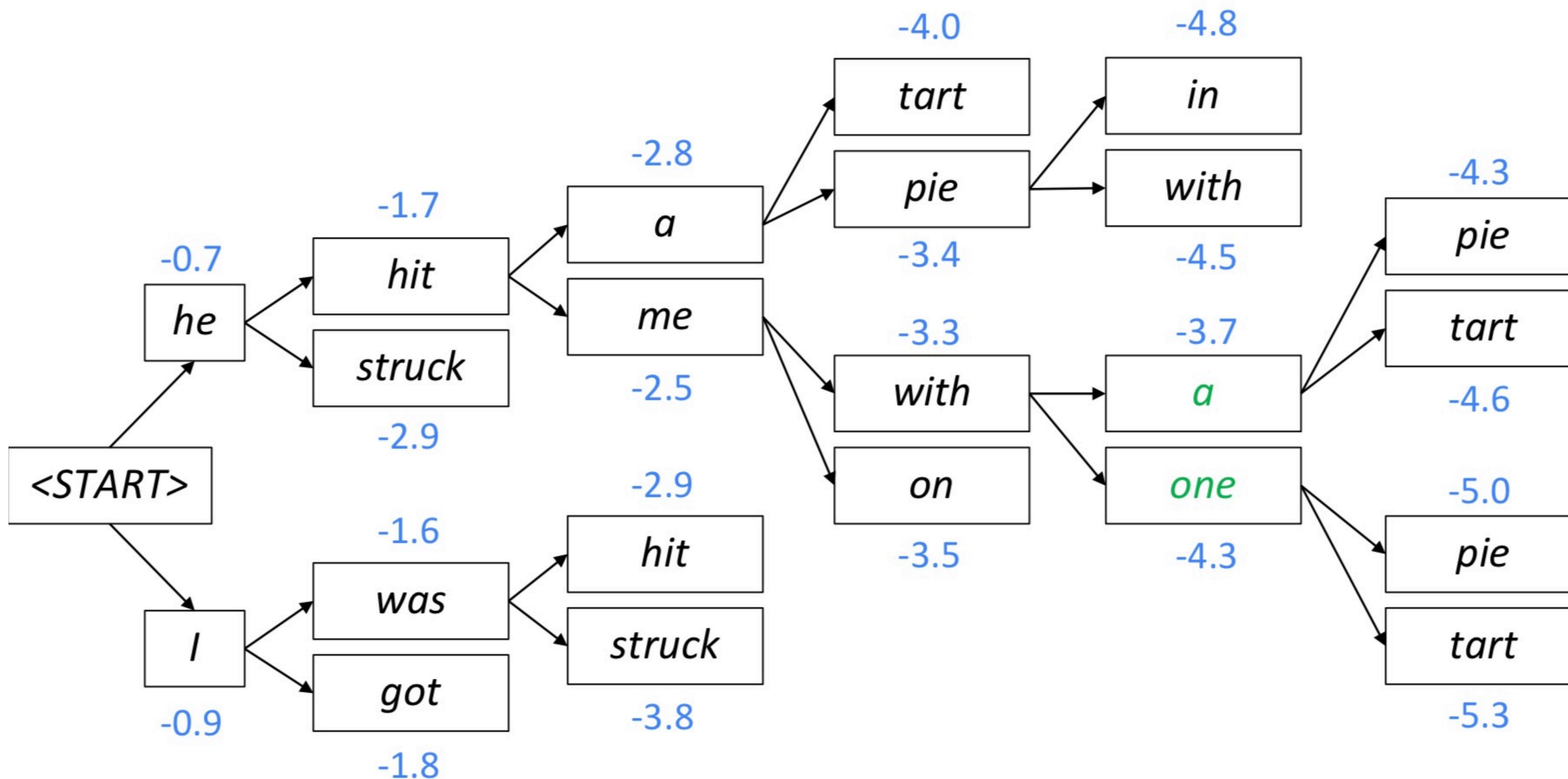
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

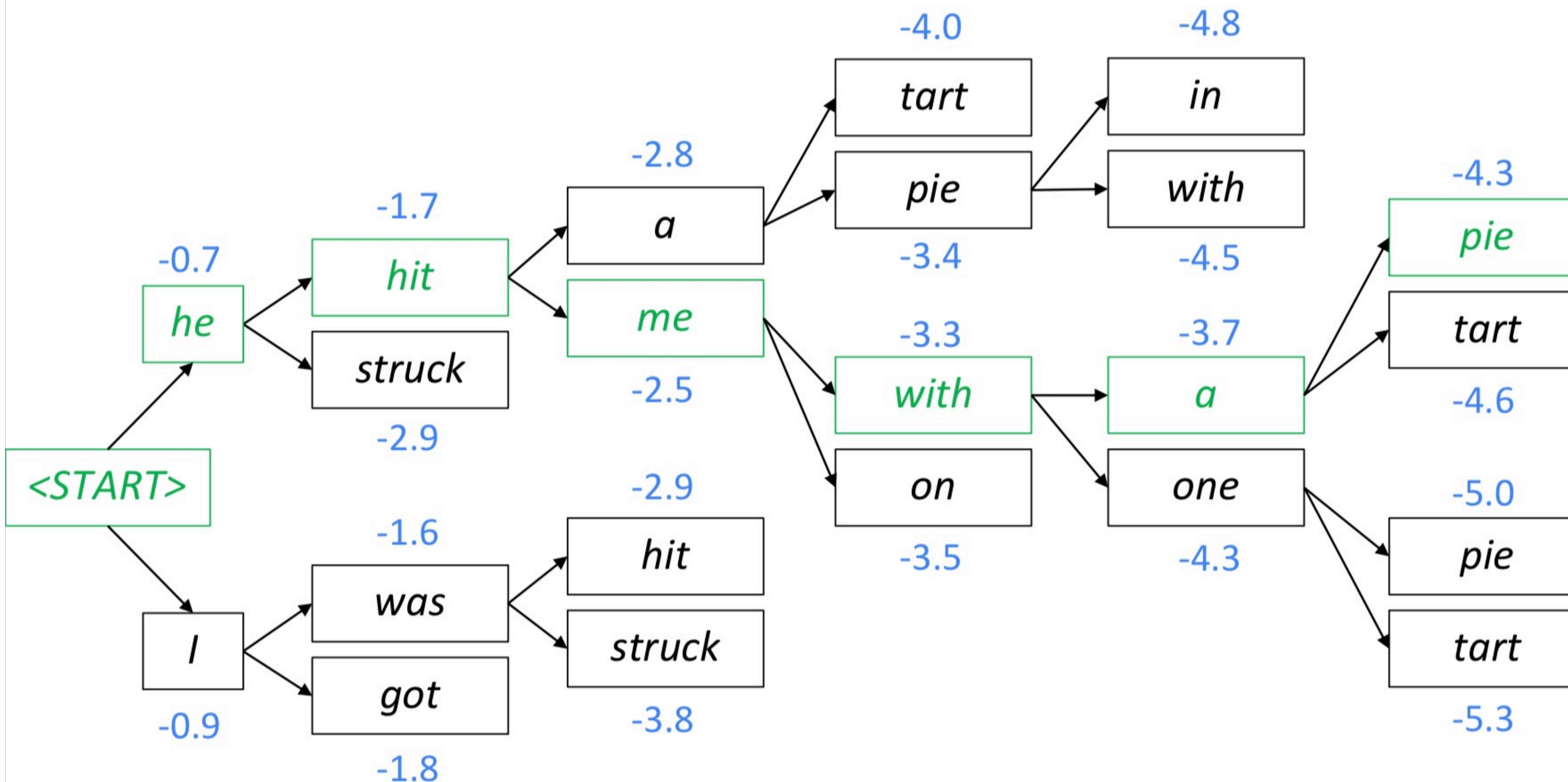
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



(slide credit: Abigail See)

Backtrack

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



(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*

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

- ▶ Otherwise shorter hypotheses have higher scores

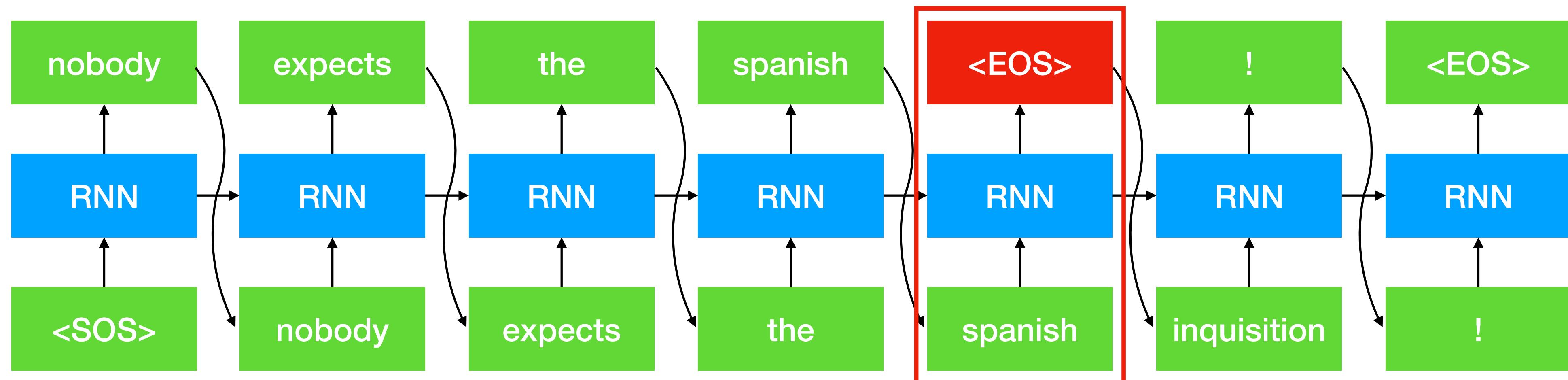
Under translation

- Crucial information are left untranslated; premature `<EOS>` generation
 - Beam search
 - Ensemble
 - Coverage models

Under-trans<EOS>



'inquisition': 0.49
<EOS>: 0.51

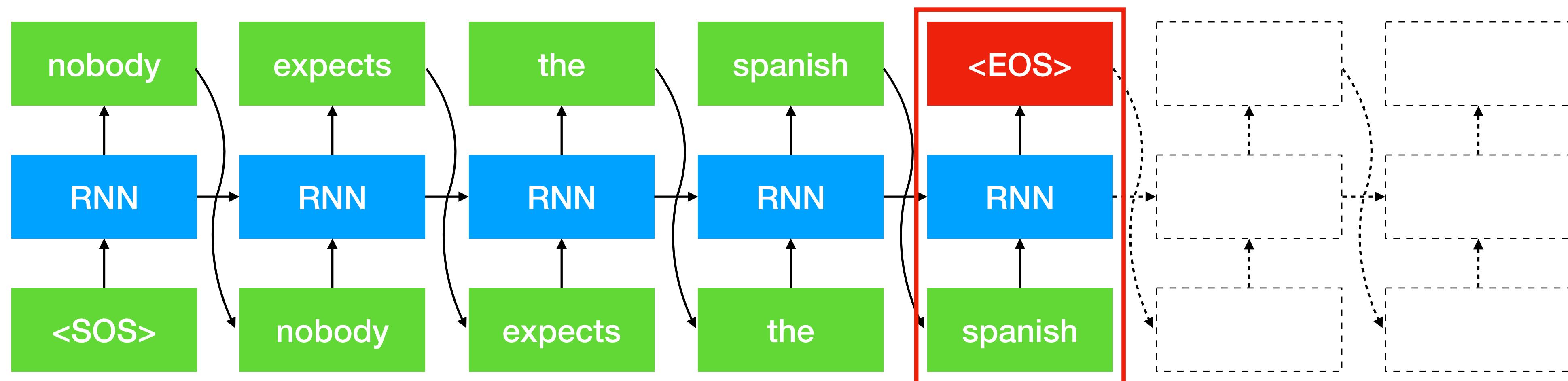


- Under translation
- Crucial information are left untranslated; premature <EOS> generation

Under-trans<EOS>



'inquisition': 0.49
<EOS>: 0.51

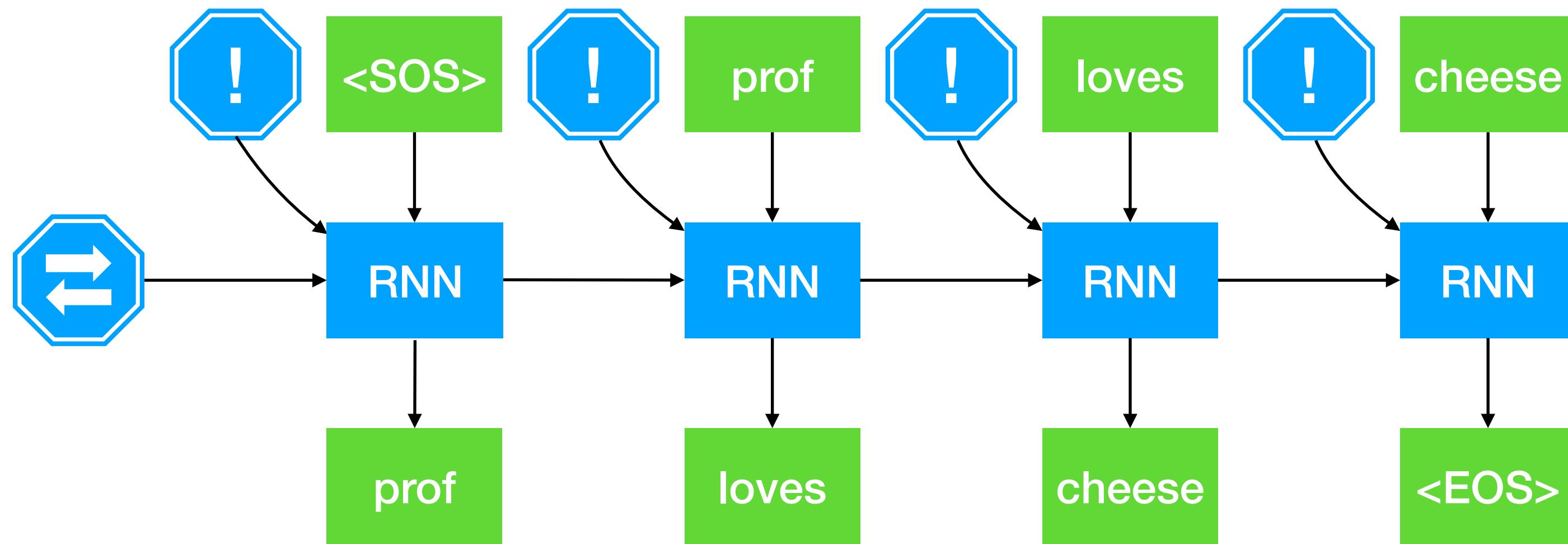


- Under translation
- Crucial information are left untranslated; premature <EOS> generation

Ensemble

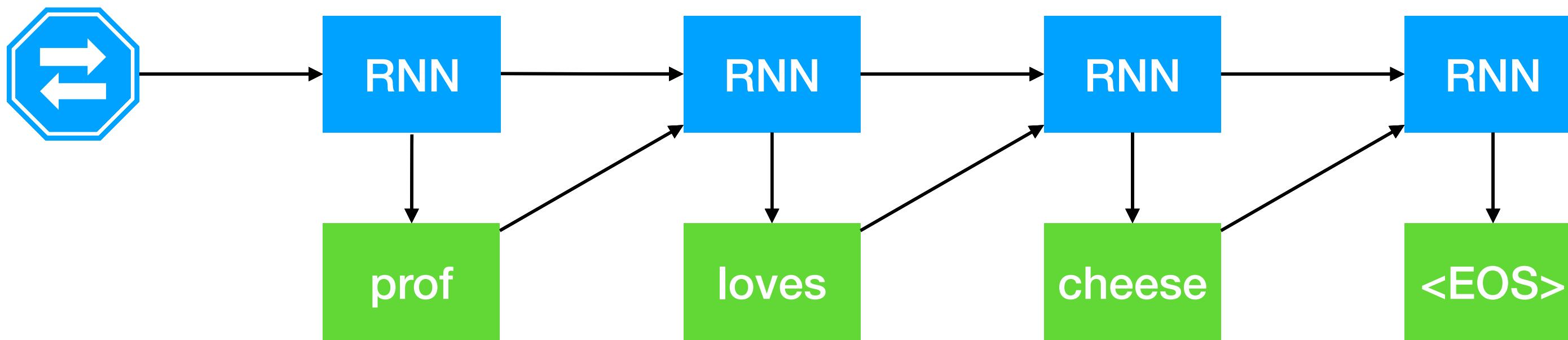
- Similar to voting mechanism, but with probabilities
- multiple models of different parameters (usually from different checkpoints of the same training instance)
- use the output with the highest probability across all models

Ensemble



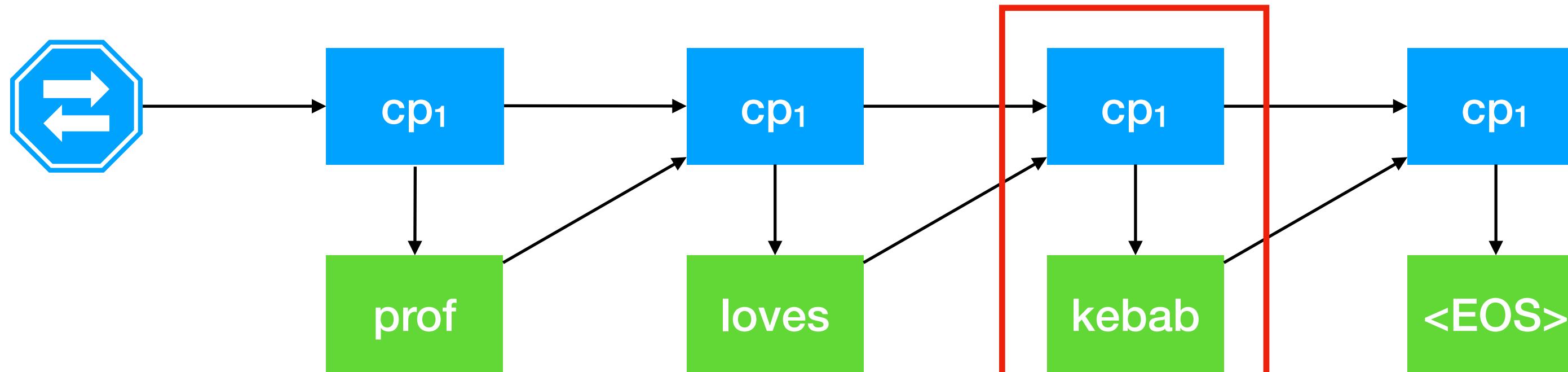
Ensemble

seq2seq_E049.pt

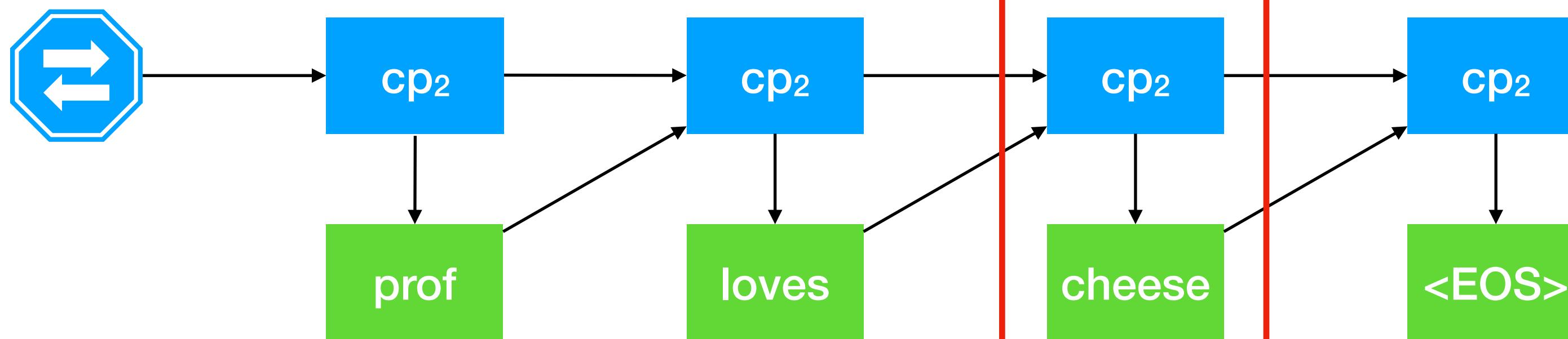


Ensemble

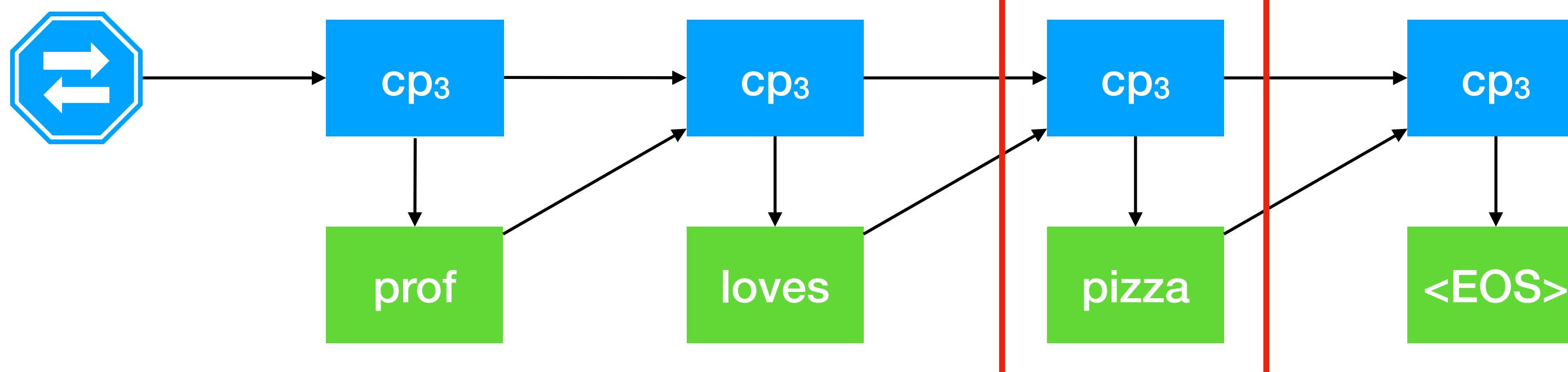
seq2seq_E049.pt



seq2seq_E048.pt

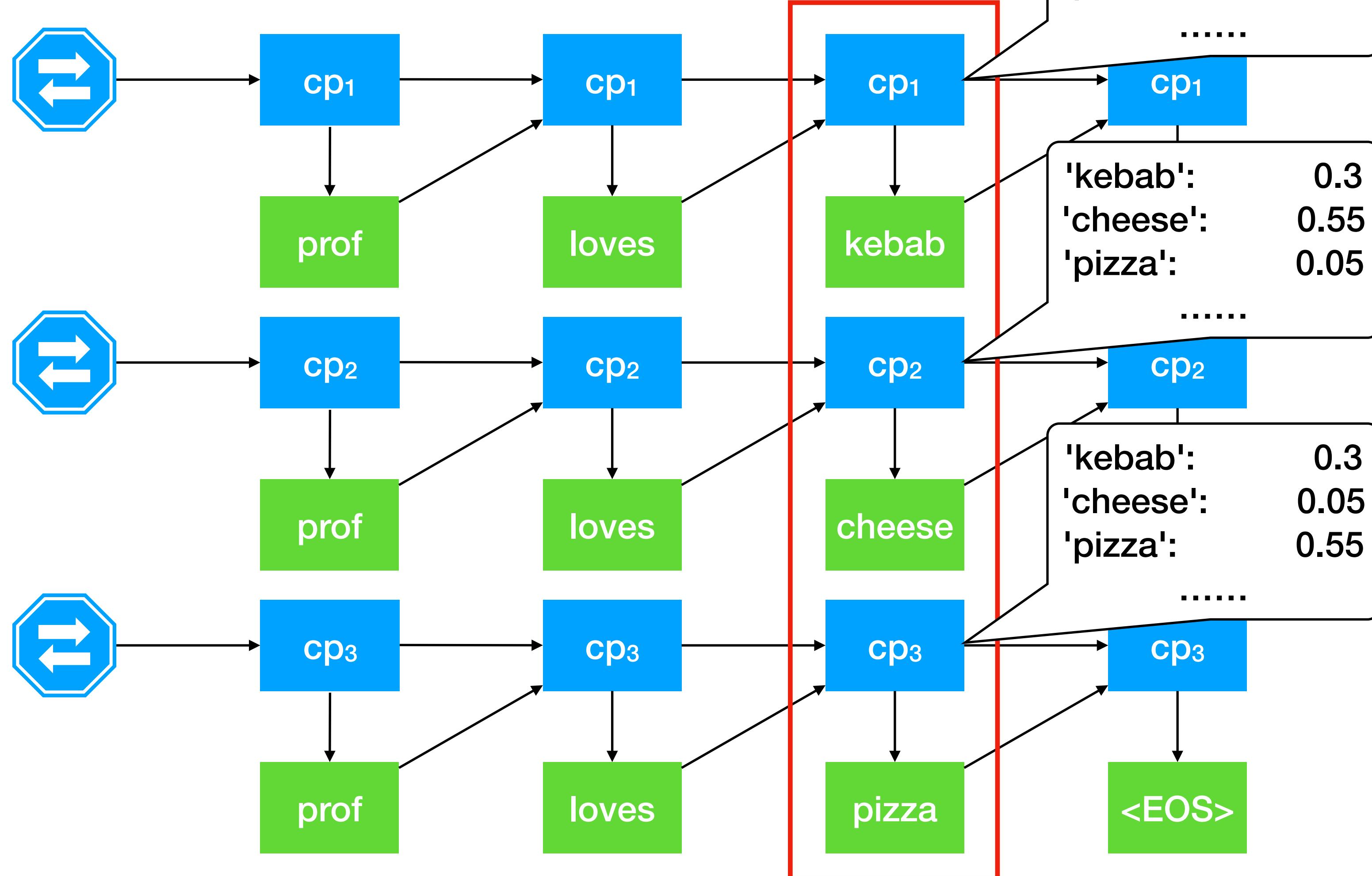


seq2seq_E047.pt



Ensemble

seq2seq_E049.pt

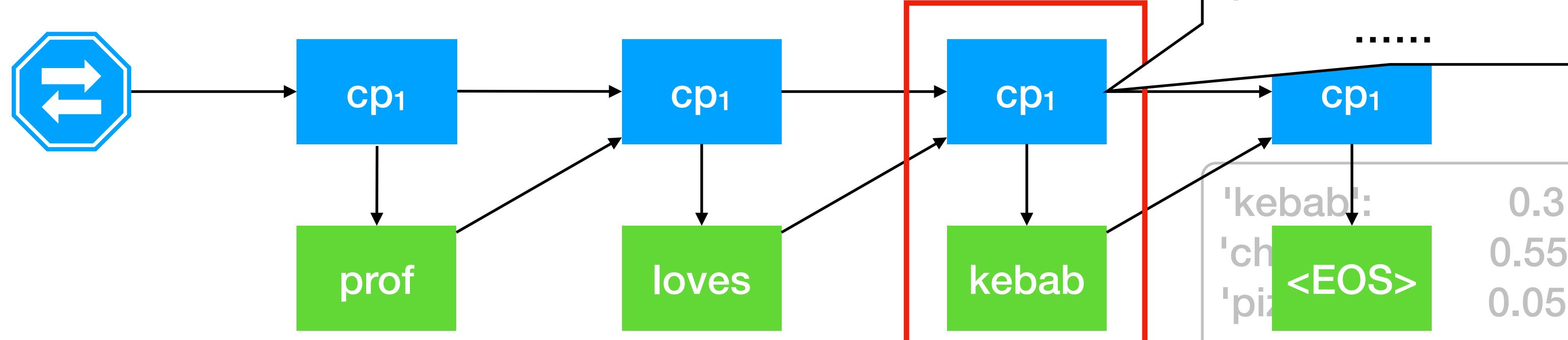


seq2seq_E048.pt

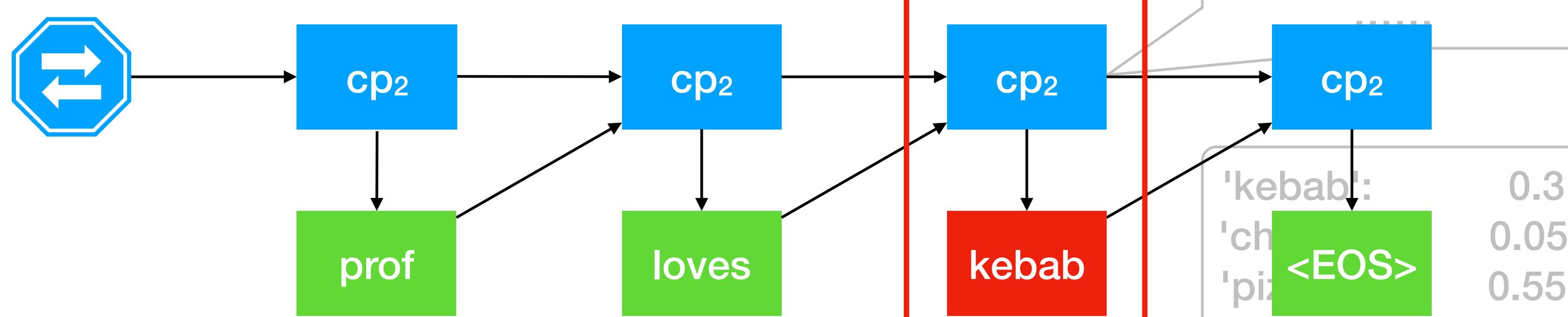
seq2seq_E047.pt

Ensemble

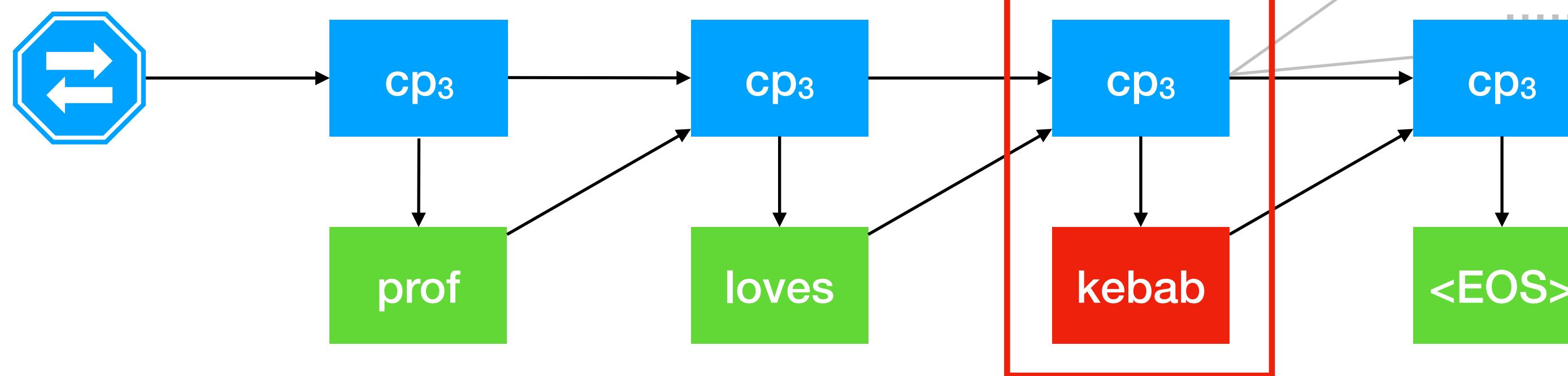
seq2seq_E049.pt



seq2seq_E048.pt

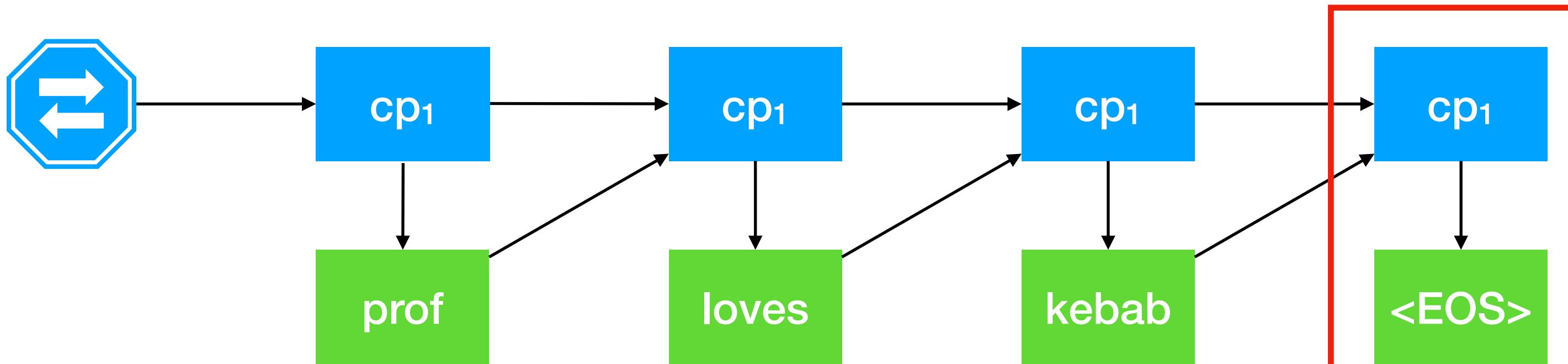


seq2seq_E047.pt

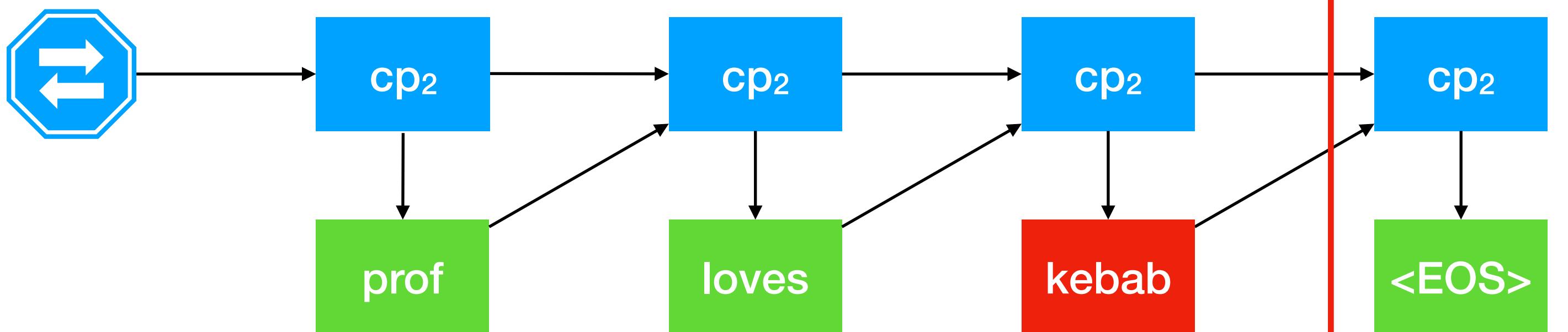


Ensemble

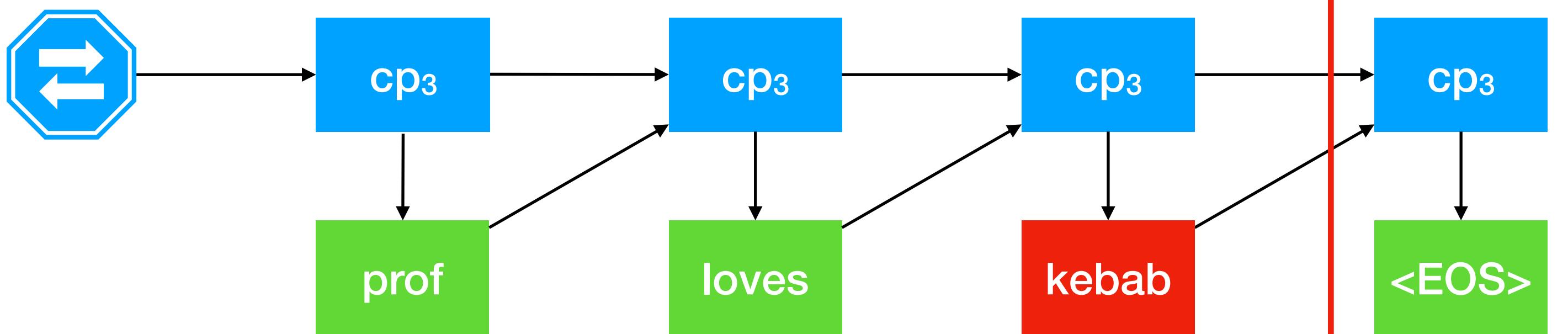
seq2seq_E049.pt



seq2seq_E048.pt



seq2seq_E047.pt



Existing challenges with NMT

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

Existing challenges with NMT

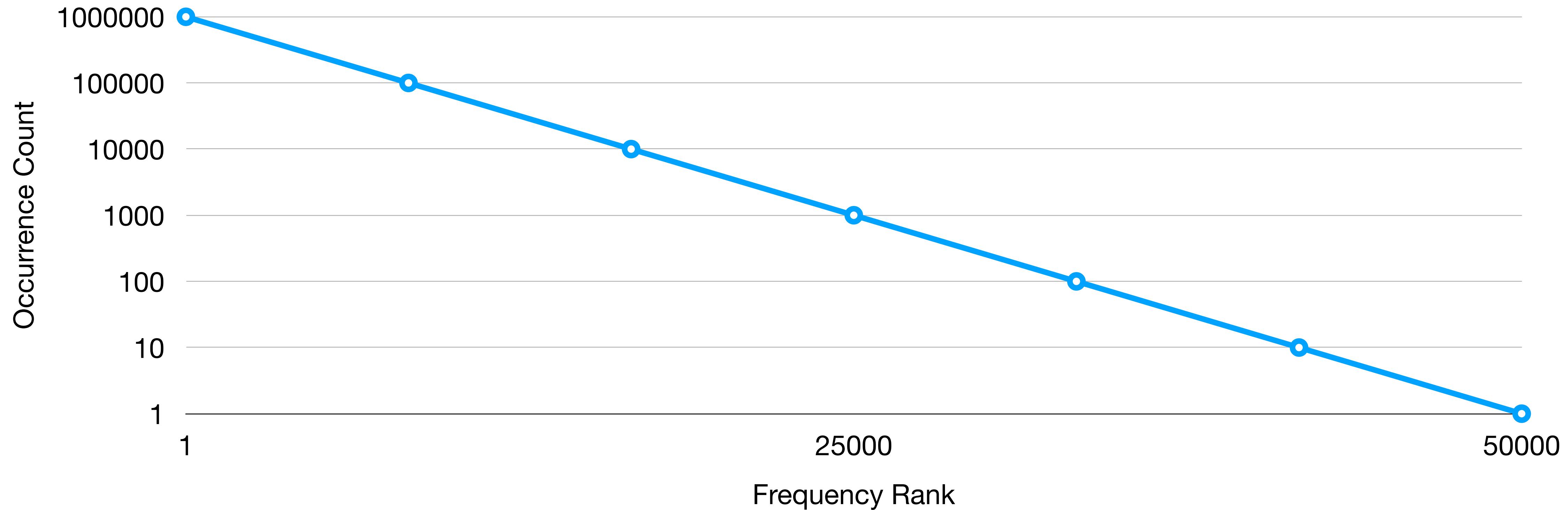
- ▶ Out-of-vocabulary words
- ▶ Low-resource languages
- ▶ Long-term context
- ▶ Common sense knowledge (e.g. hot dog, paper jam)
- ▶ Fairness and bias
- ▶ Uninterpretable

Out-of-vocabulary words

- Out-of-Vocabulary (OOV) Problem; Rare word problem
 - Frequent words are translated correctly, rare words are not
 - In any corpus, word frequencies are exponentially unbalanced

OOV

Zipf's Law

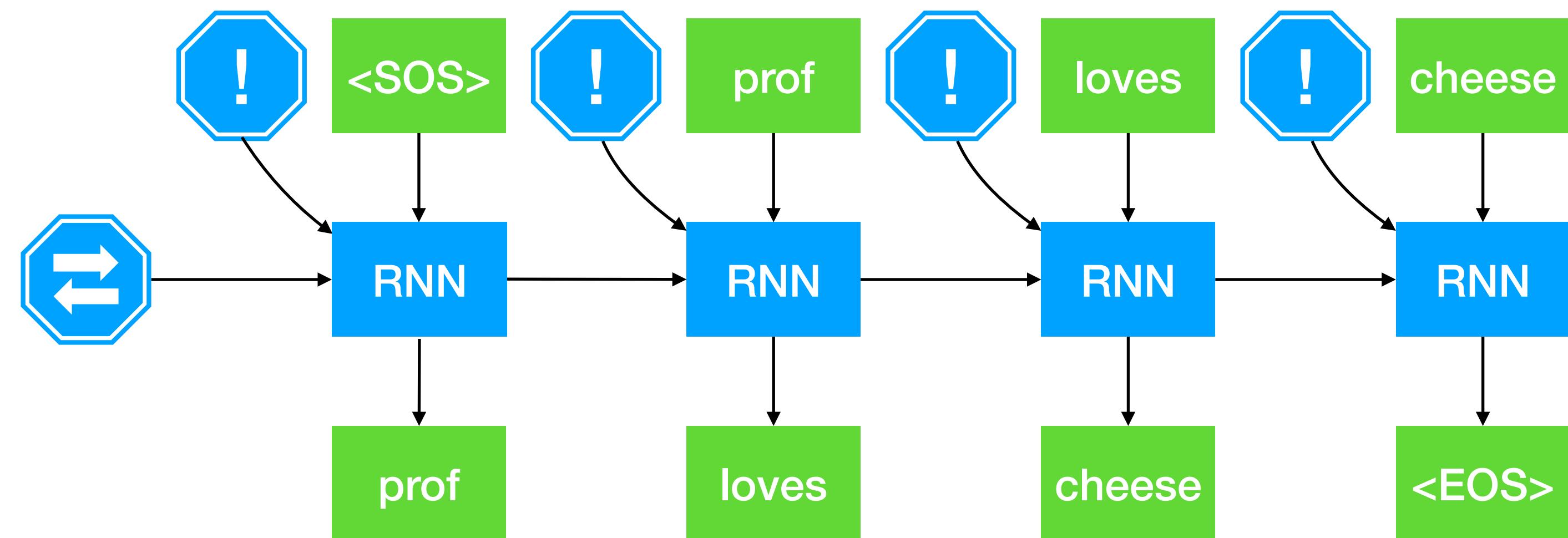


- rare words are exponentially less frequent than frequent words
- e.g. in HW4, 45%|65% (src|tgt) of the unique words occur once

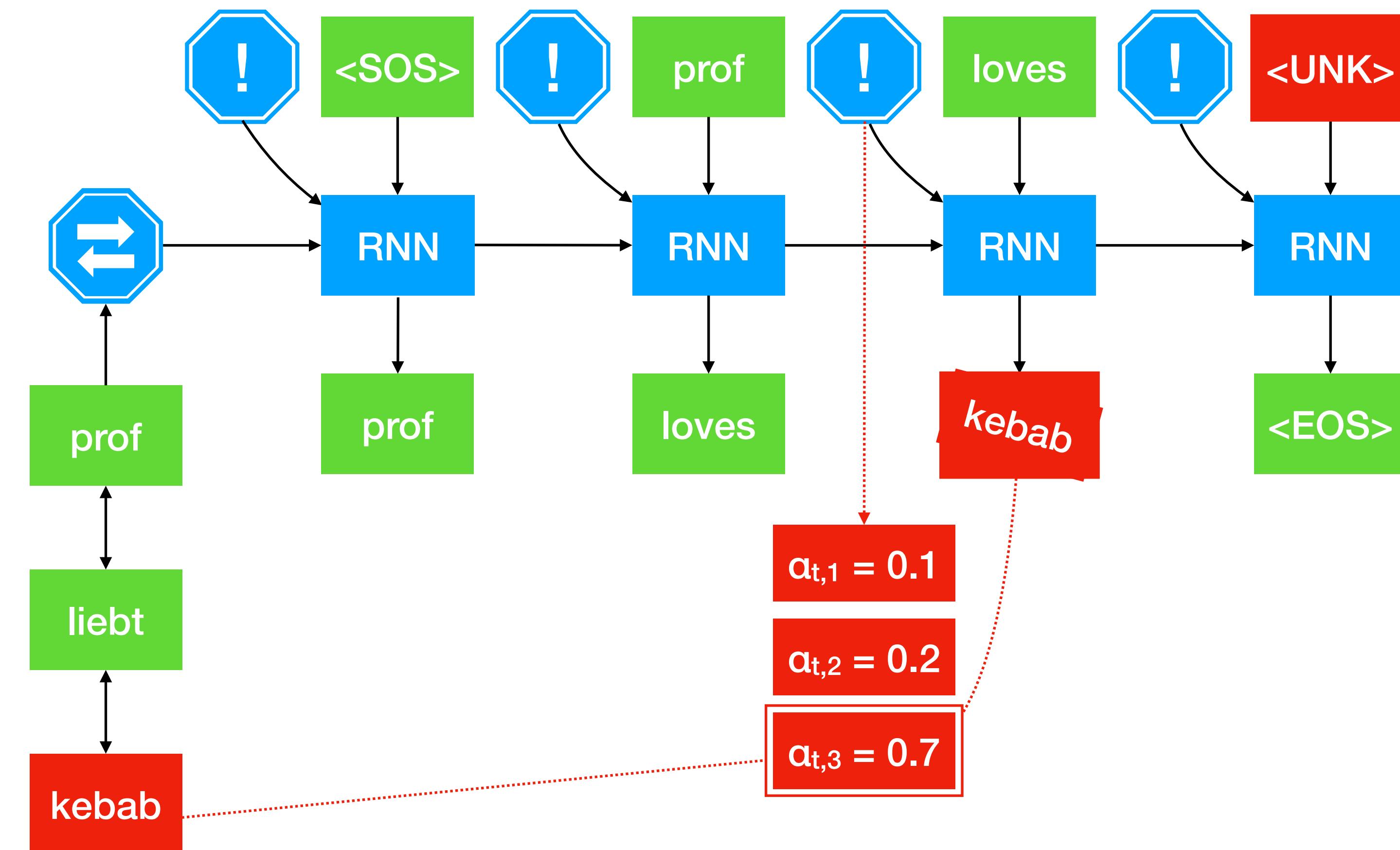
Out-of-vocabulary words

- Out-of-Vocabulary (OOV) Problem; Rare word problem
 - Copy Mechanisms
 - Subword / character models

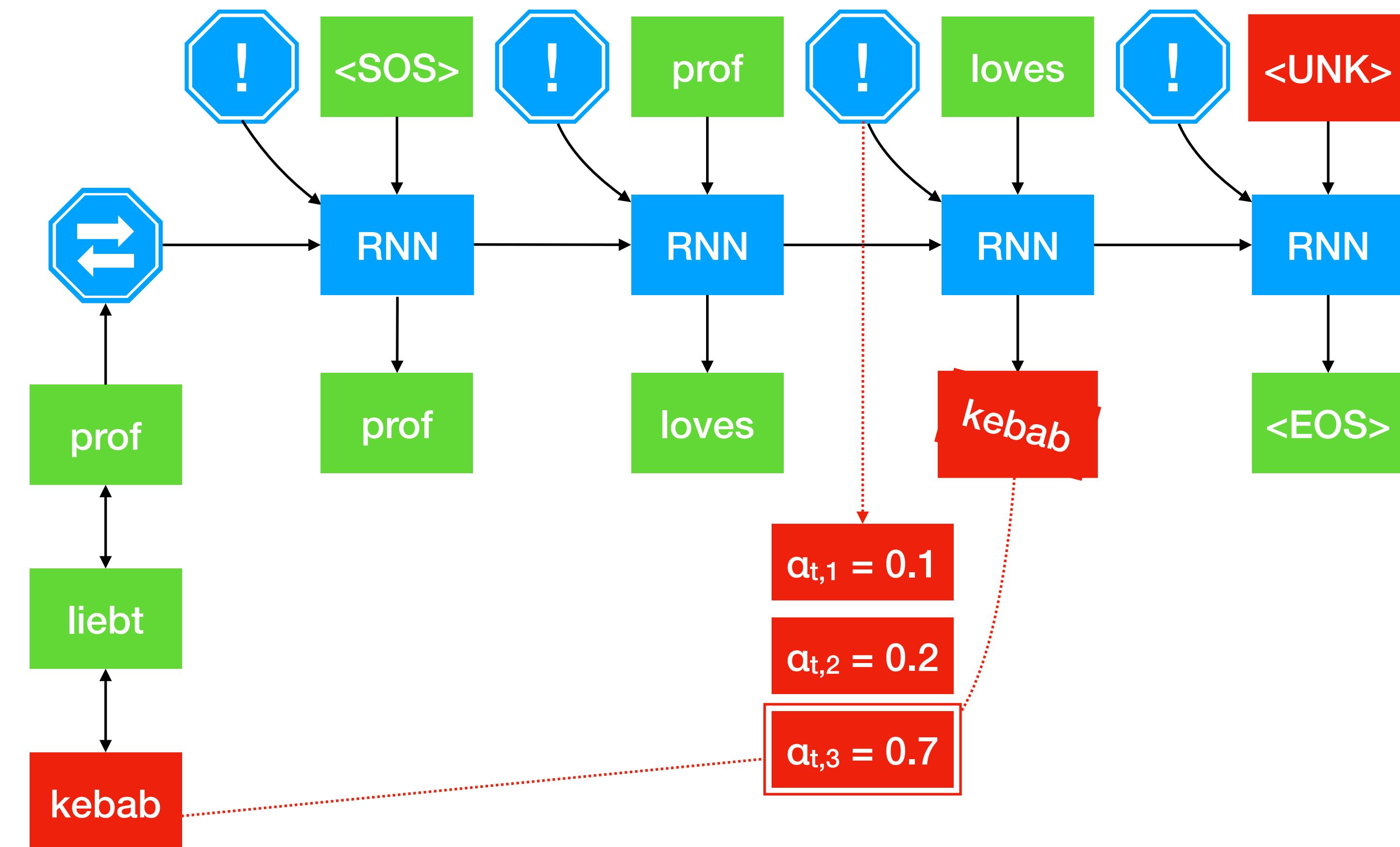
Copy Mechanism



Copy Mechanism

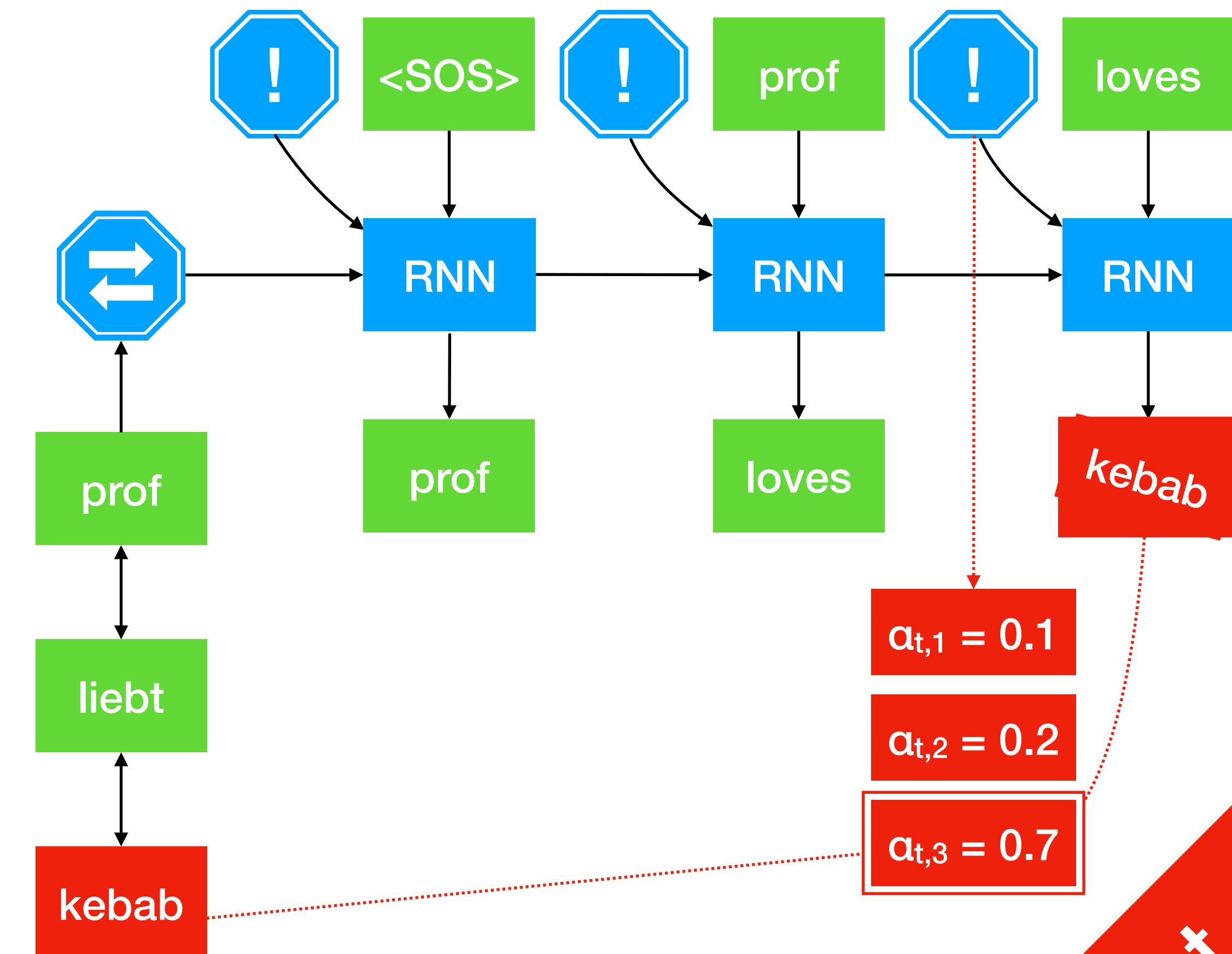


Copy Mechanism

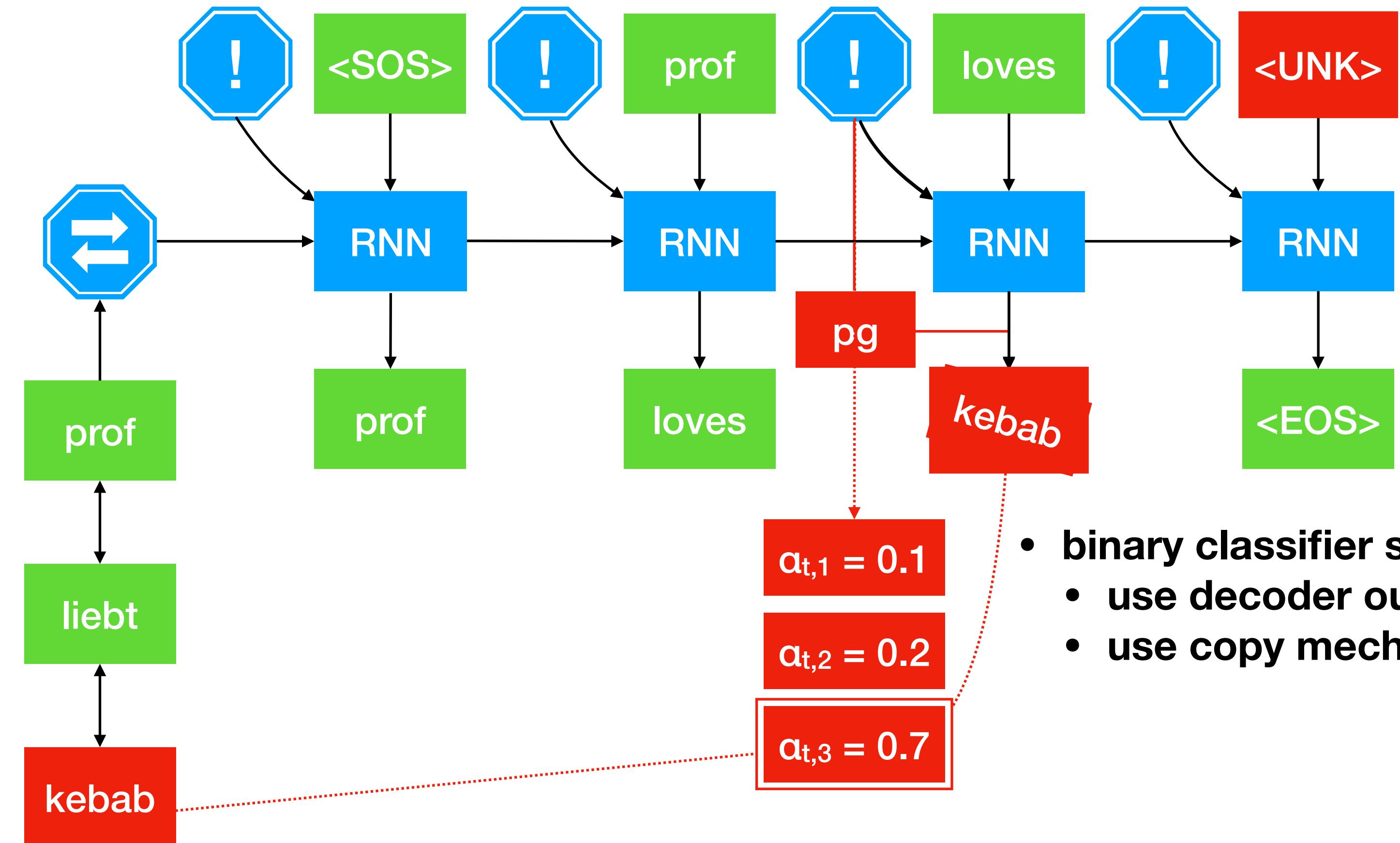


Copy Mechanism

- Sees **<UNK>** at step t
 - looks at attention weight α_t
 - replace **<UNK>** with the source word
 $f_{\text{argmax}_i \alpha_{t,i}}$

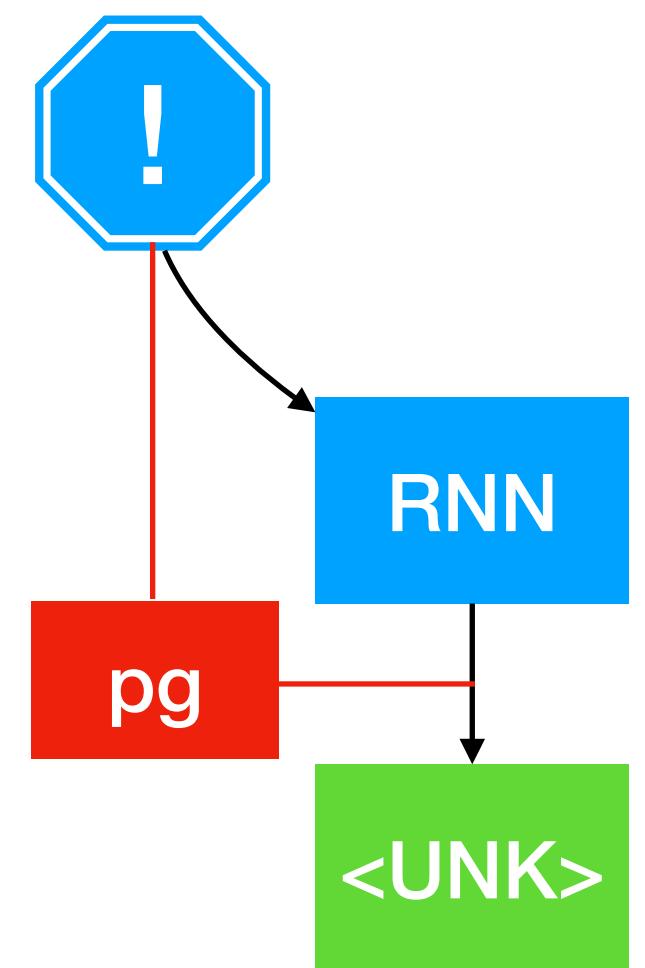


*Pointer-Generator Copy Mechanism



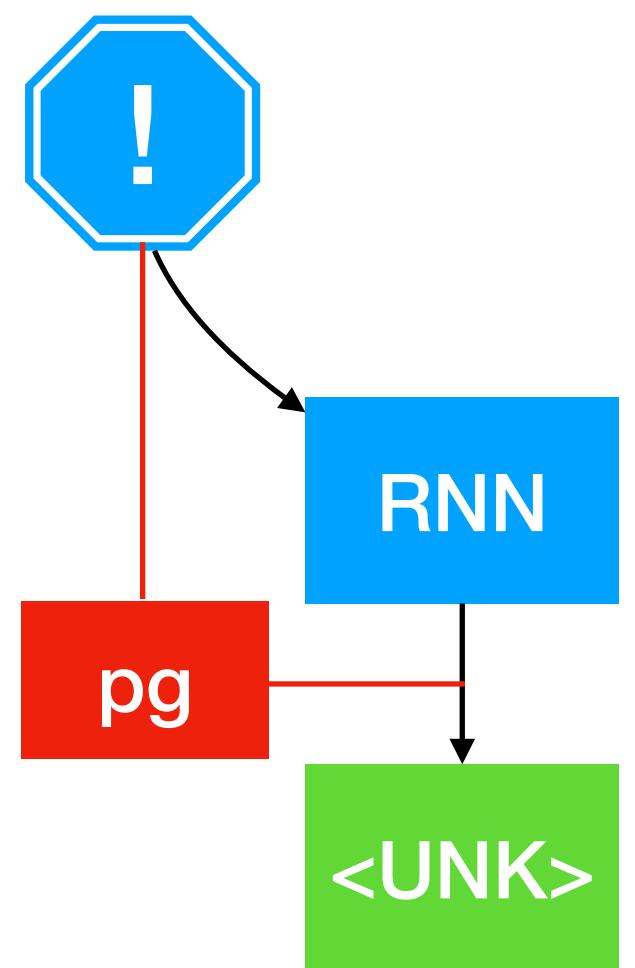
*Pointer-Generator Copy Mechanism

- Sees $\langle \text{UNK} \rangle$ at step t , or $\text{pg}([h_t^{dec}; context_t]) \leq 0.5$
 - looks at attention weight α_t
 - replace $\langle \text{UNK} \rangle$ with the source word $f_{\text{argmax}_i \alpha_{t,i}}$



*Pointer-Based Dictionary Fusion

- Sees $\langle\text{UNK}\rangle$ at step t , or $pg([h_t^{dec}; context_t]) \leq 0.5$
 - looks at attention weight α_t
 - replace $\langle\text{UNK}\rangle$ with **translation of** the source word $\text{dict}(f_{\text{argmax}_i \alpha_{t,i}})$



Out-of-vocabulary words

- Out-of-Vocabulary (OOV) Problem; Rare word problem
 - Copy Mechanisms
 - Subword / character models

OOV

- Rich morphology: **nejneobhospodařovávatelnějšímu**
("to the worst farmable one")
- Transliteration: **Christopher ↳ Kryštof**
- Informal spelling:

The image shows two tweets from the Twitter mobile application interface. Both posts have blacked-out user handles and URLs.

Brianna @_parsimonia_ · 24h
Gooooood Vibesssssss

@J0YUS · 1m
When idc, I really don't care.
Like my "I want space" is me shutting you out. My "**imma** go, u want something?" And u don't say nothing, then I'm not coming back sumn 4 u

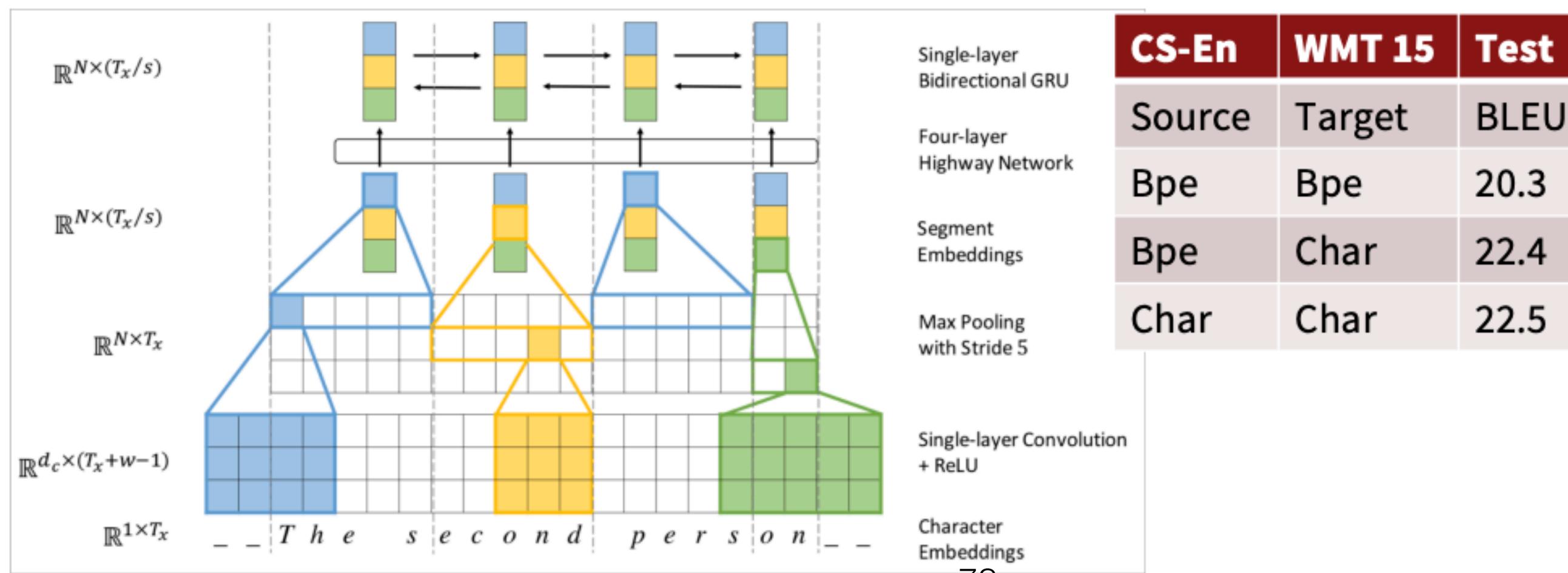
Subword / character models

- Character based seq2seq models

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



Subword / character models

- Character based seq2seq models
- Byte pair encoding (BPE)

A word segmentation algorithm:

- Start with a vocabulary of characters
- Most frequent ngram pairs \mapsto a new ngram

Dictionary

5 low
2 lower
6 newest
3 widest

Vocabulary

I, o, w, e, r, n, w, s, t, i, d

Start with all characters
in vocab

(Example from Sennrich)

Subword / character models

- Character based seq2seq models
- Byte pair encoding (BPE)

A word segmentation algorithm:

- Start with a vocabulary of characters
- Most frequent ngram pairs \mapsto a new ngram

Dictionary

5 low
2 lower
6 newes t
3 wide es t

Vocabulary

I, o, w, e, r, n, w, s, t, i, d, es

Add a pair (e, s) with freq 9

(Example from Sennrich)

Subword / character models

- Character based seq2seq models
- Byte pair encoding (BPE)

A **word segmentation** algorithm:

- Start with a vocabulary of **characters**
- Most frequent **ngram pairs** \mapsto a new **ngram**

Dictionary

5 low
2 lower
6 newest
3 widest

Vocabulary

I, o, w, e, r, n, w, s, t, i, d, es, est

Add a pair (es, t) with freq 9

(Example from Sennrich)

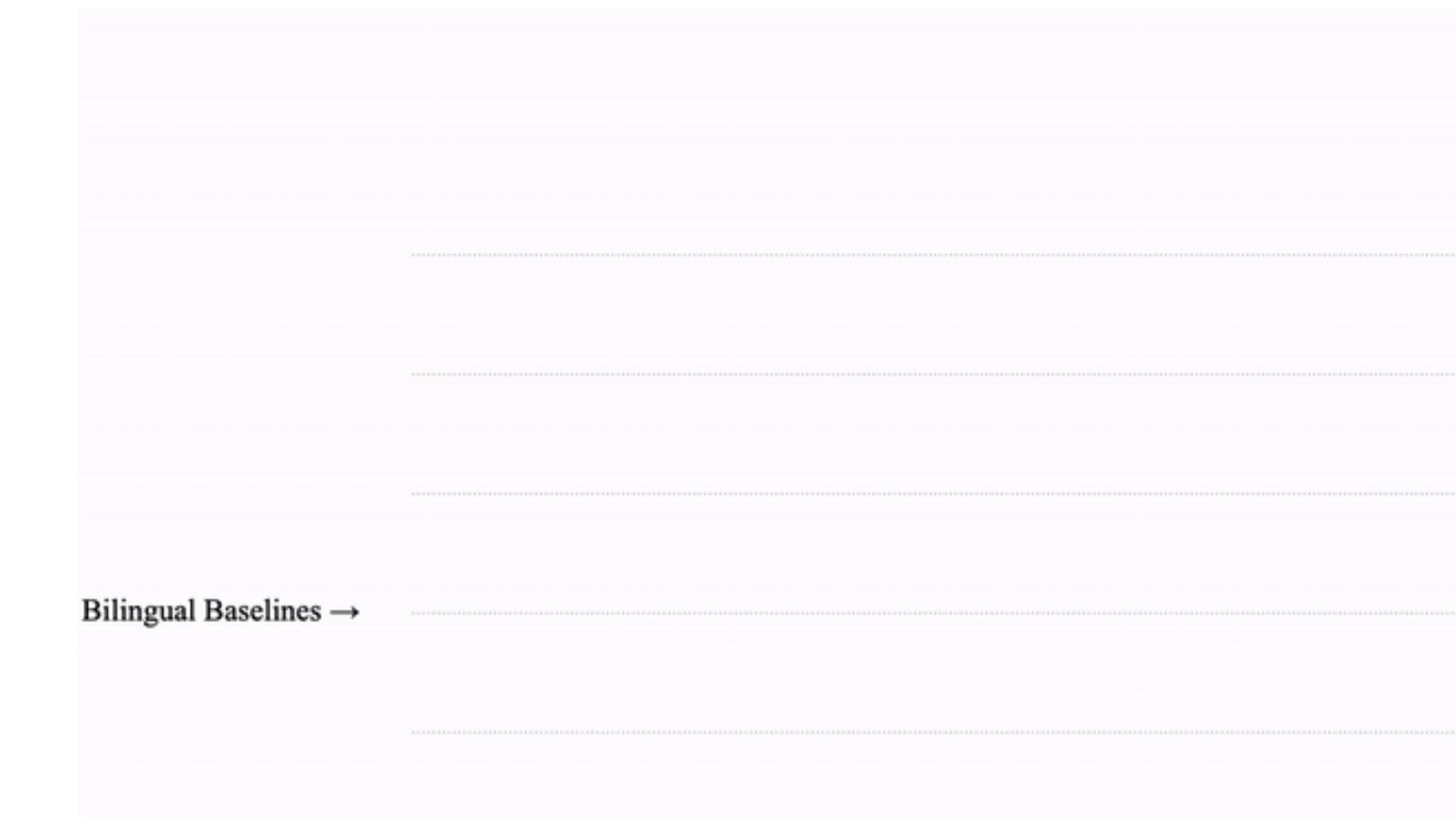
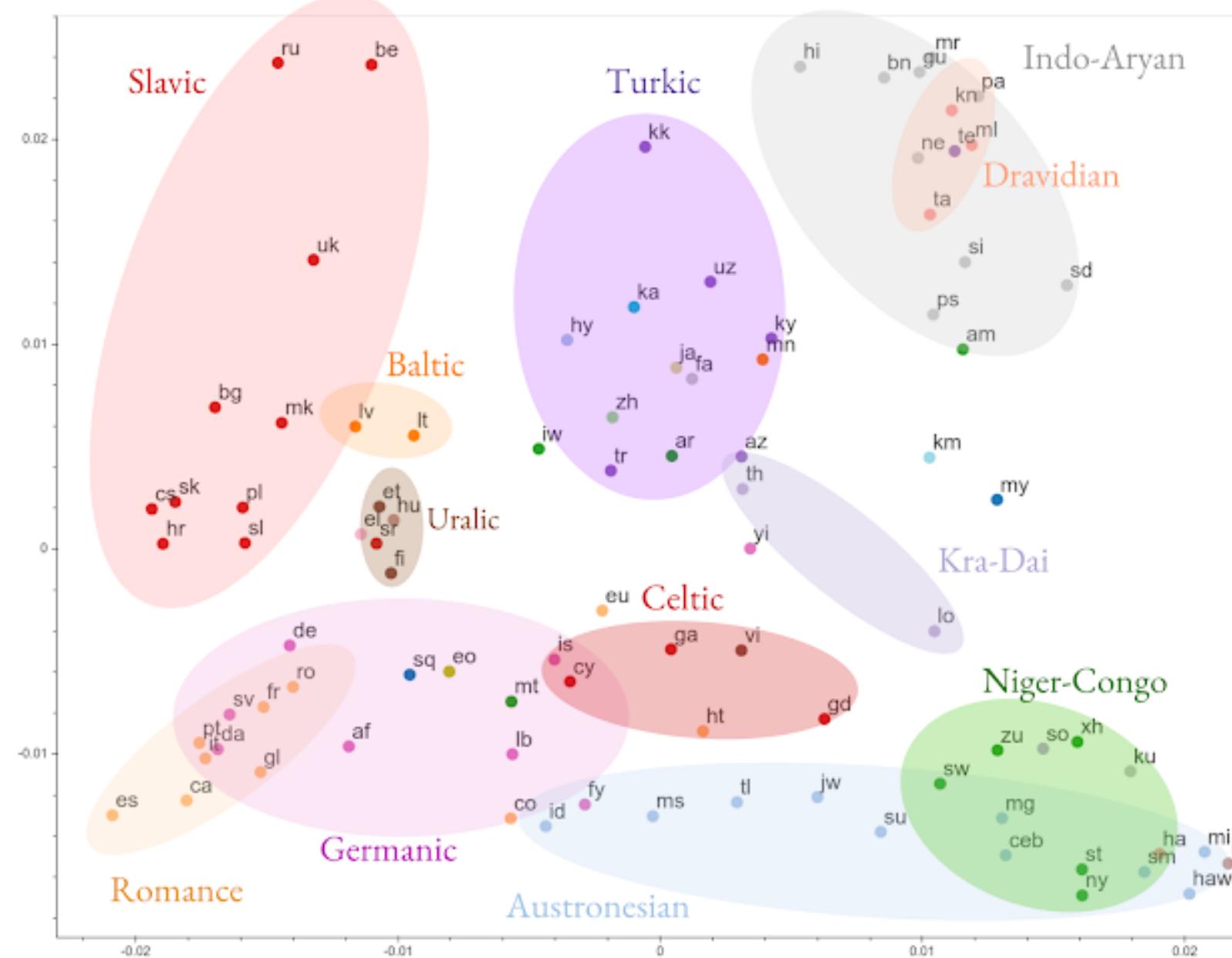
Subword / character models

- Character based seq2seq models
- Byte pair encoding (BPE)
- Subword units help model morphology

Existing challenges with NMT

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

Massively multilingual MT



- ▶ Train a *single* neural network on 103 languages paired with English (remember Interlingua?)
- ▶ Massive improvements on low-resource languages

(Arivazhagan et al., 2019)

Existing challenges with NMT

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

Next time

- ▶ Contextualized embeddings
 - ▶ ELMO, BERT, and friends
- ▶ Transformers
 - ▶ Multi-headed self-attention

