CMPT 825: Natural Language Processing

Neural Machine Translation

Spring 2020

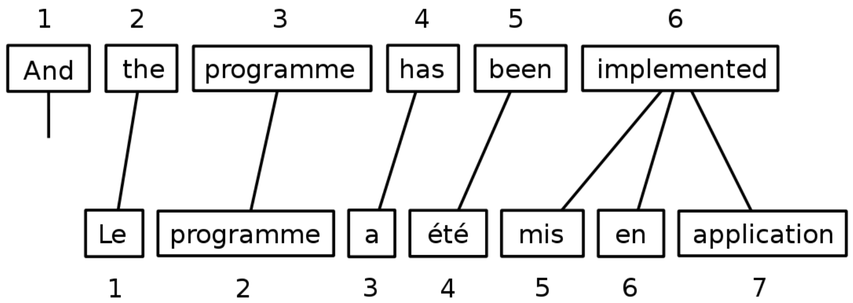
2020-03-12

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

Course Logistics

- Project proposal is due today
  - What problem are you addressing? Why is it interesting?
  - What specific aspects will your project be on?
    - Re-implement paper? Compare different methods?
  - What data do you plan to use?
  - What is your method?
  - How do you plan to evaluate? What metrics?

Last time



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

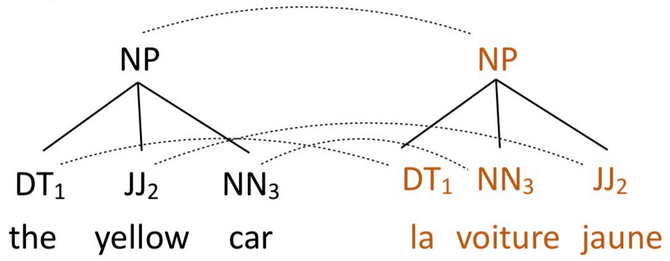
NP → [DT_1 JJ_2 NN_3; DT_1 NN_3 JJ_2]

DT → [the, la]

DT → [the, le]

NN → [car, voiture]

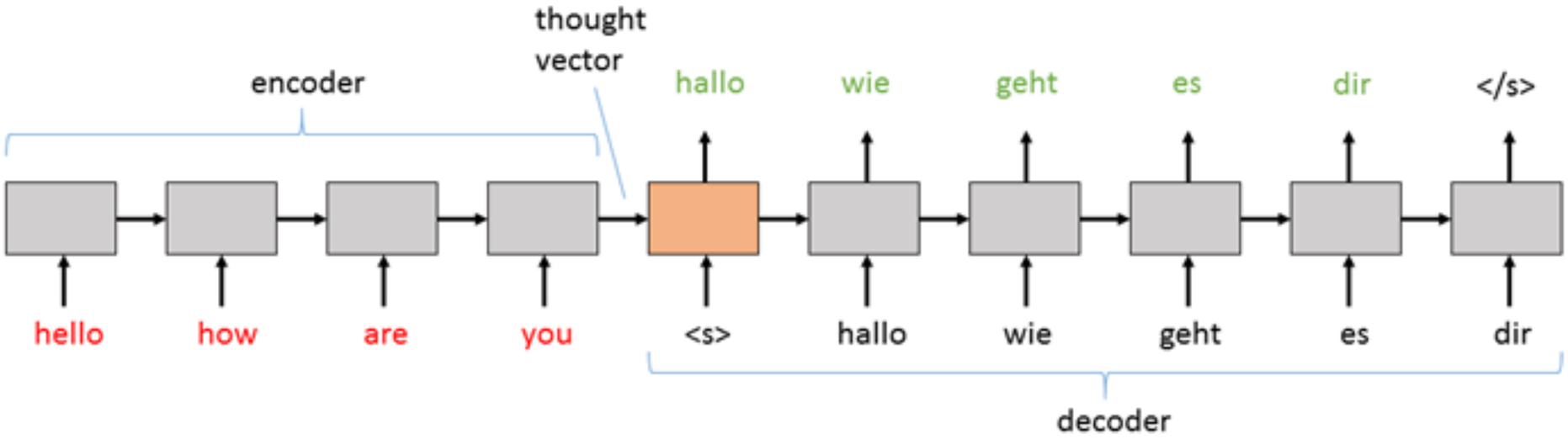
JJ → [yellow, jaune]



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)

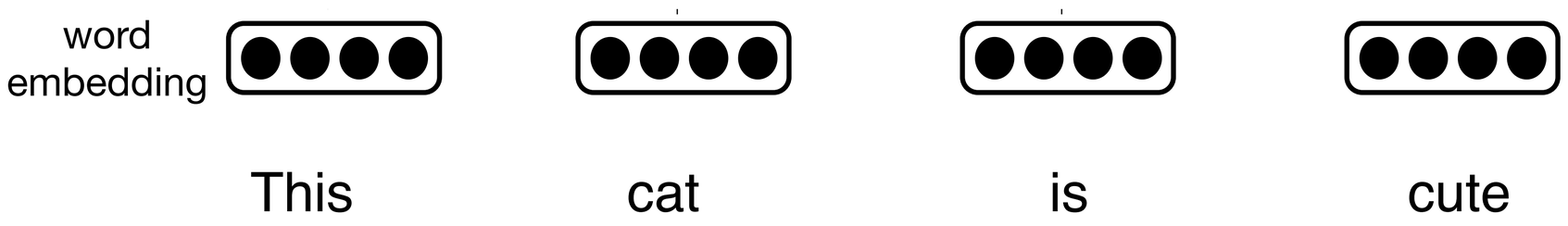


- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

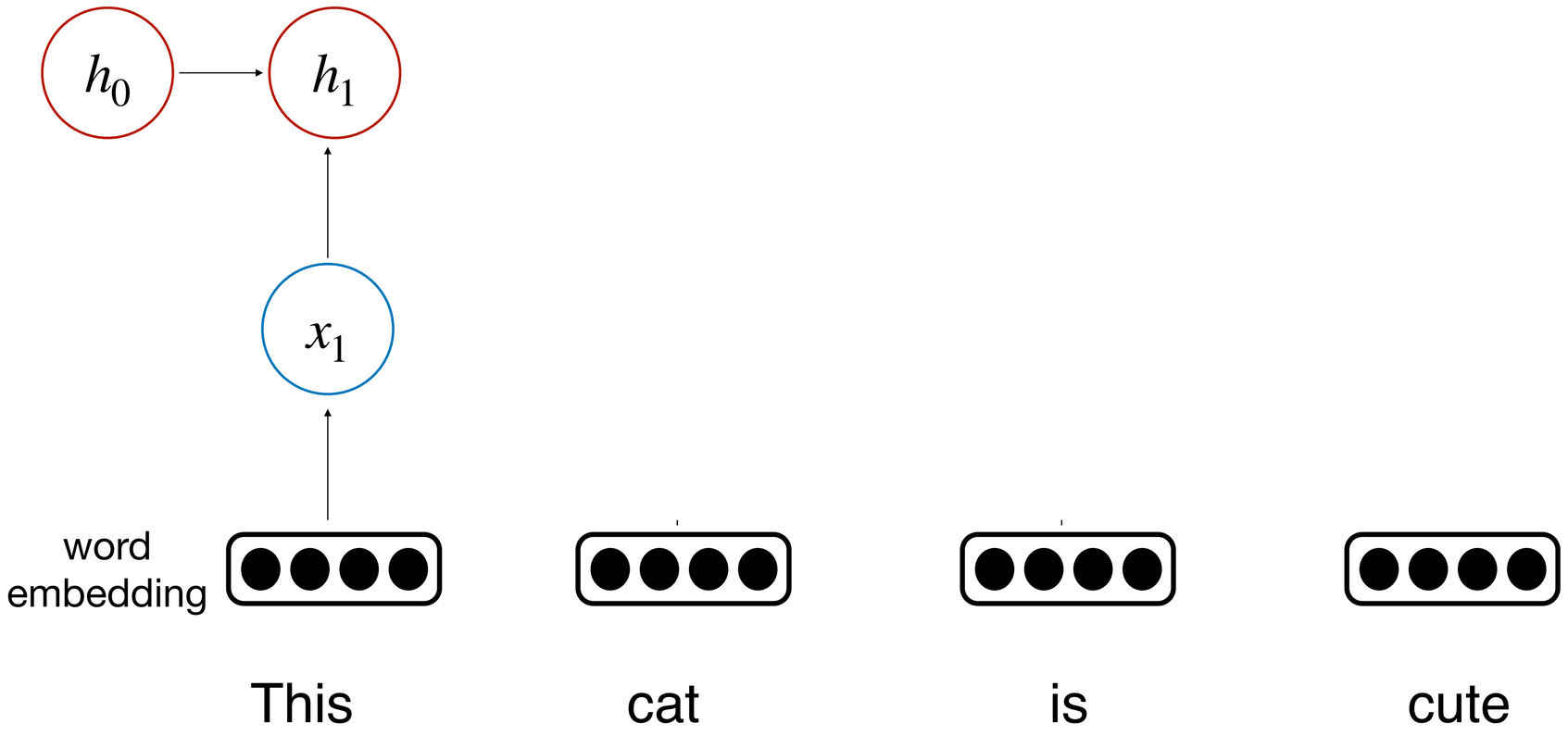
Encoder

Sentence: This cat is cute



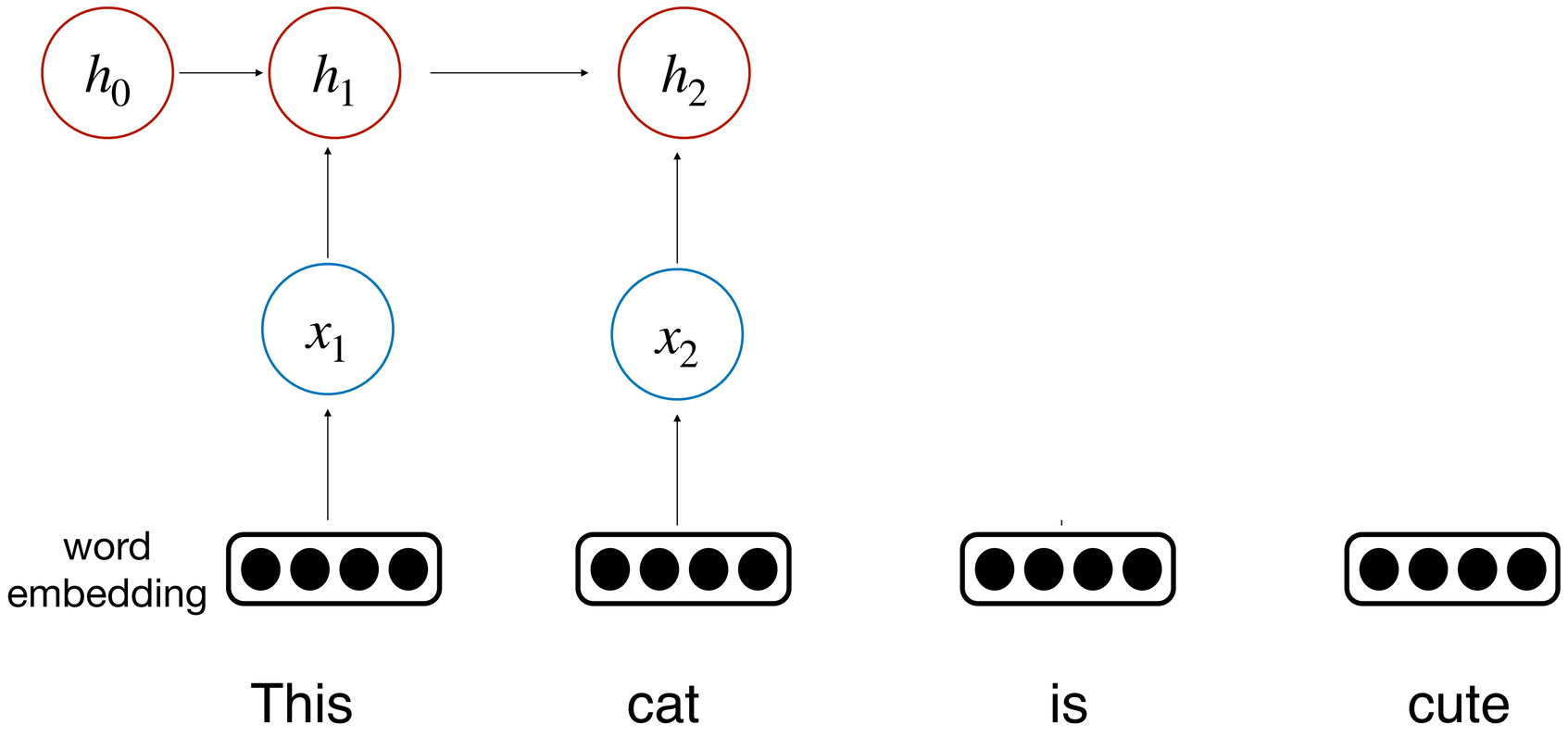
Encoder

Sentence: This cat is cute



Encoder

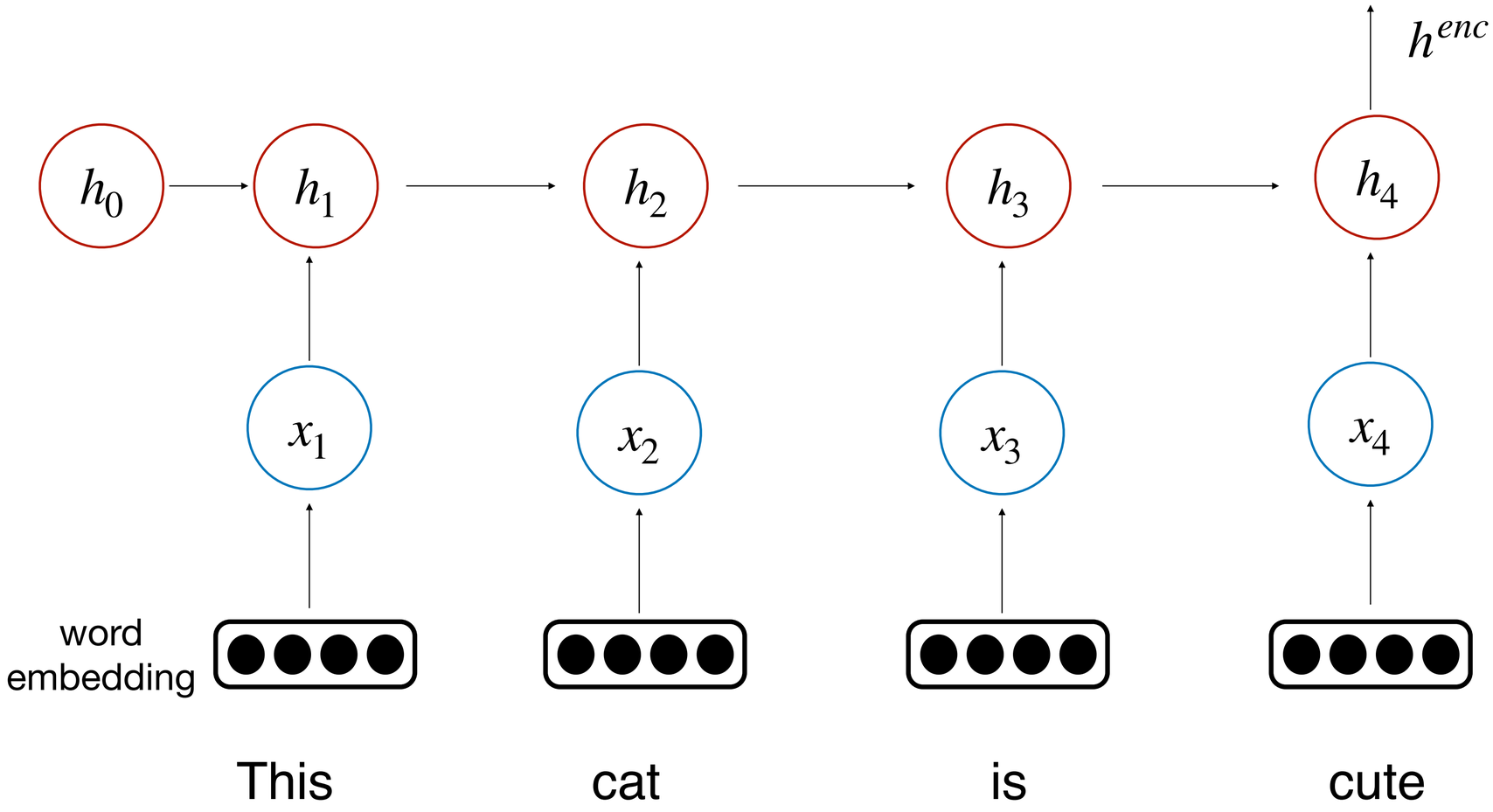
Sentence: This cat is cute



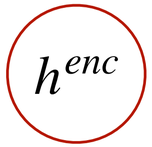
Encoder

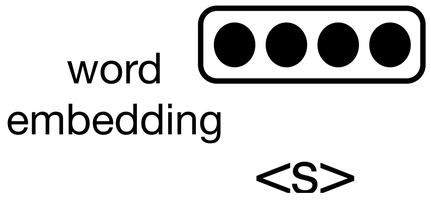
Sentence: This cat is cute

(encoded representation)

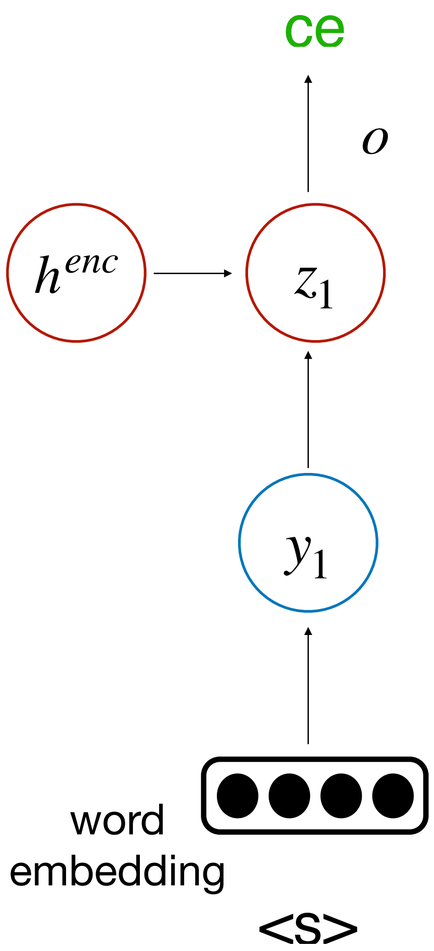


Decoder

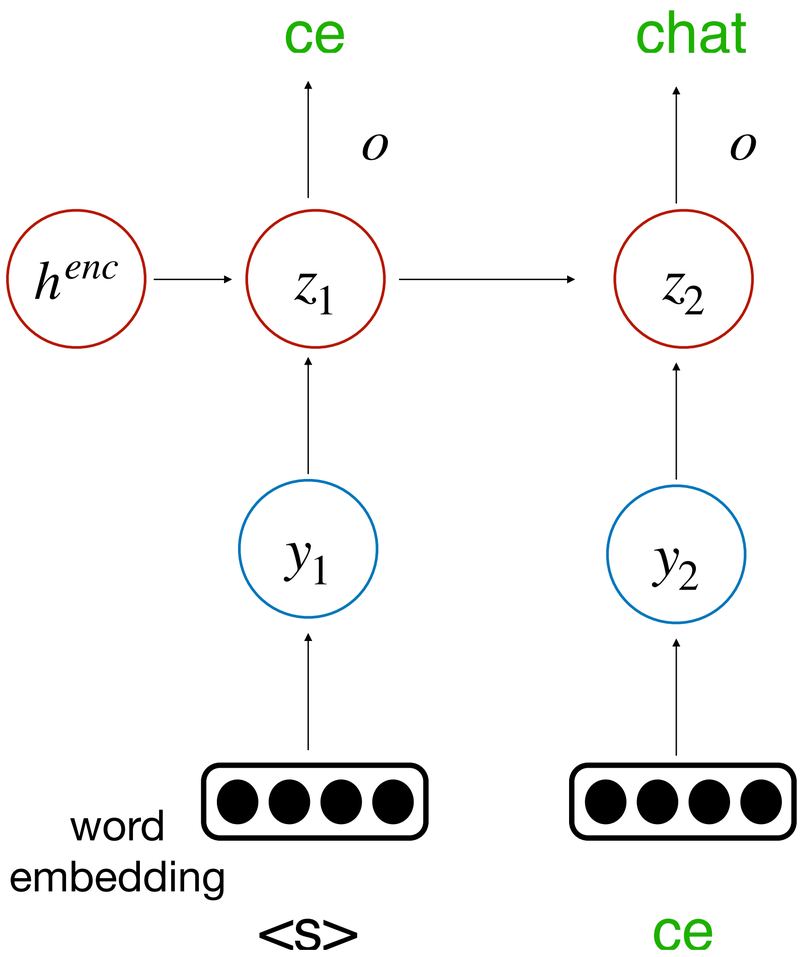




Decoder

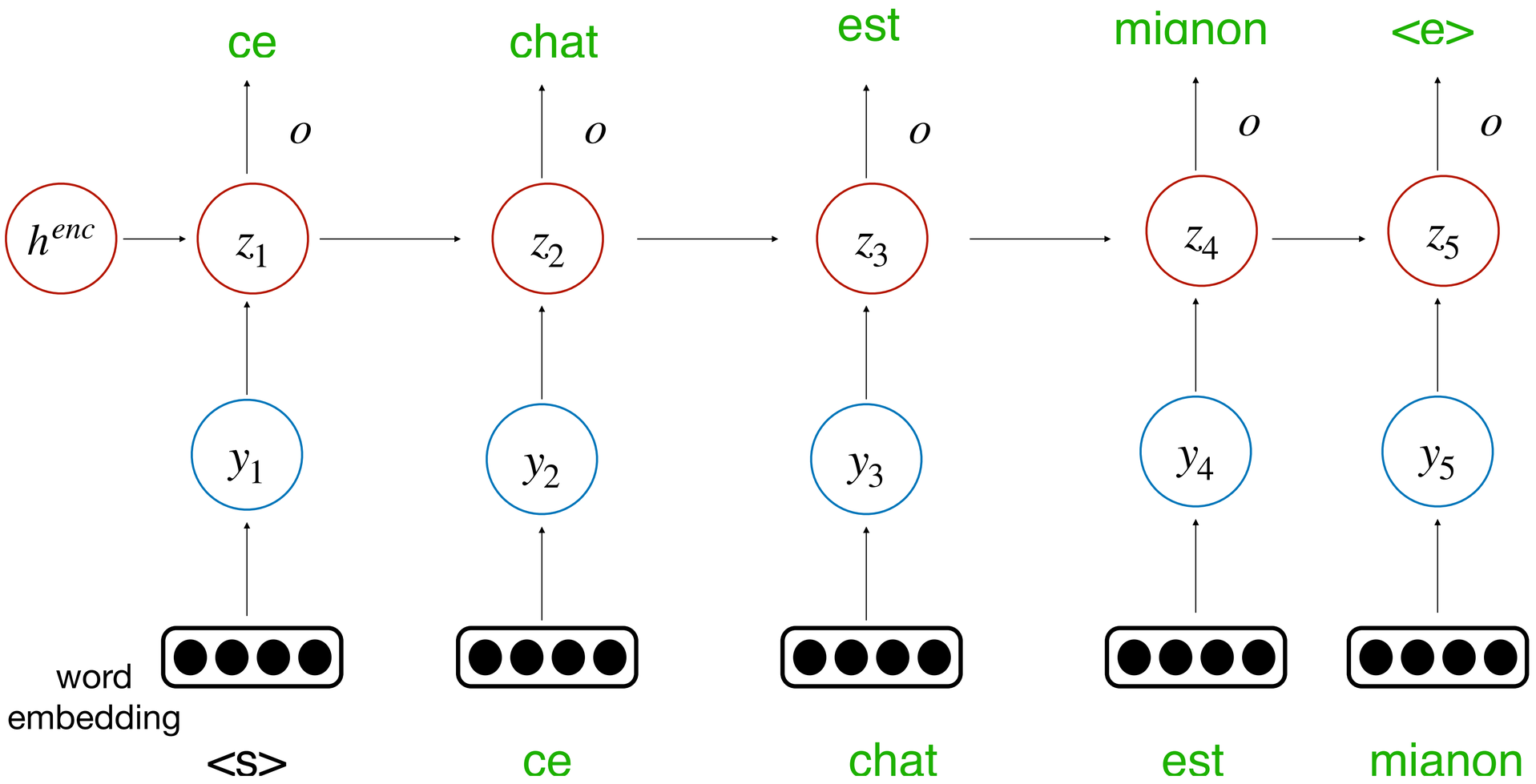


Decoder



Decoder

- A conditioned language model

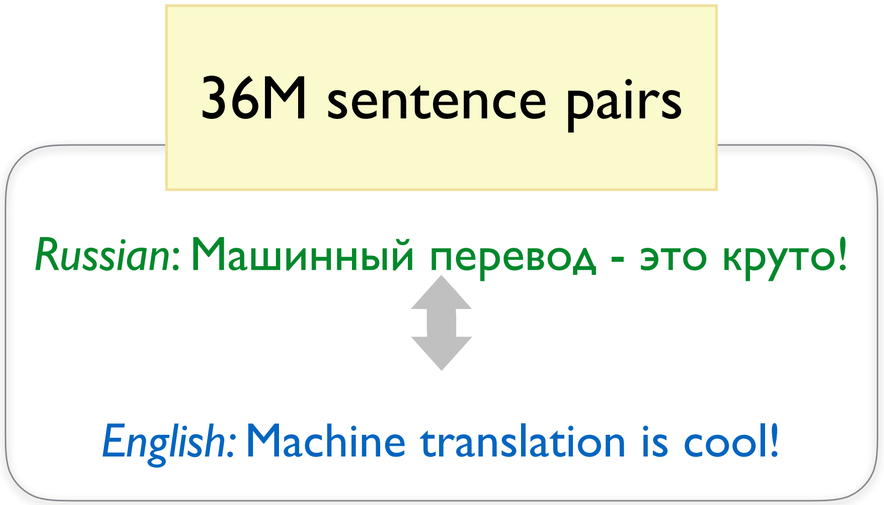


Seq2seq training

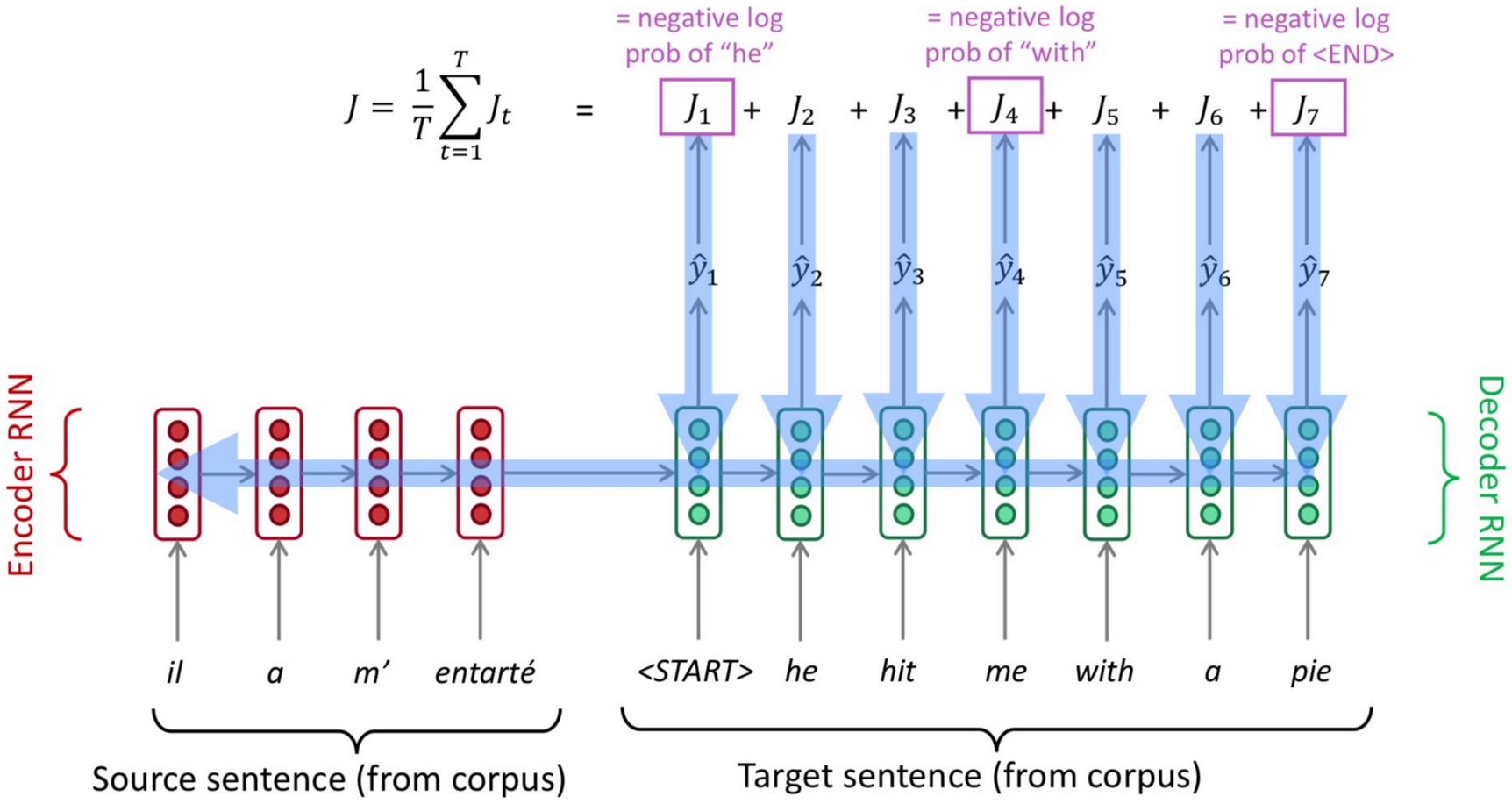
- Similar to training a language model!
- Minimize cross-entropy loss:

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

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



Seq2seq training

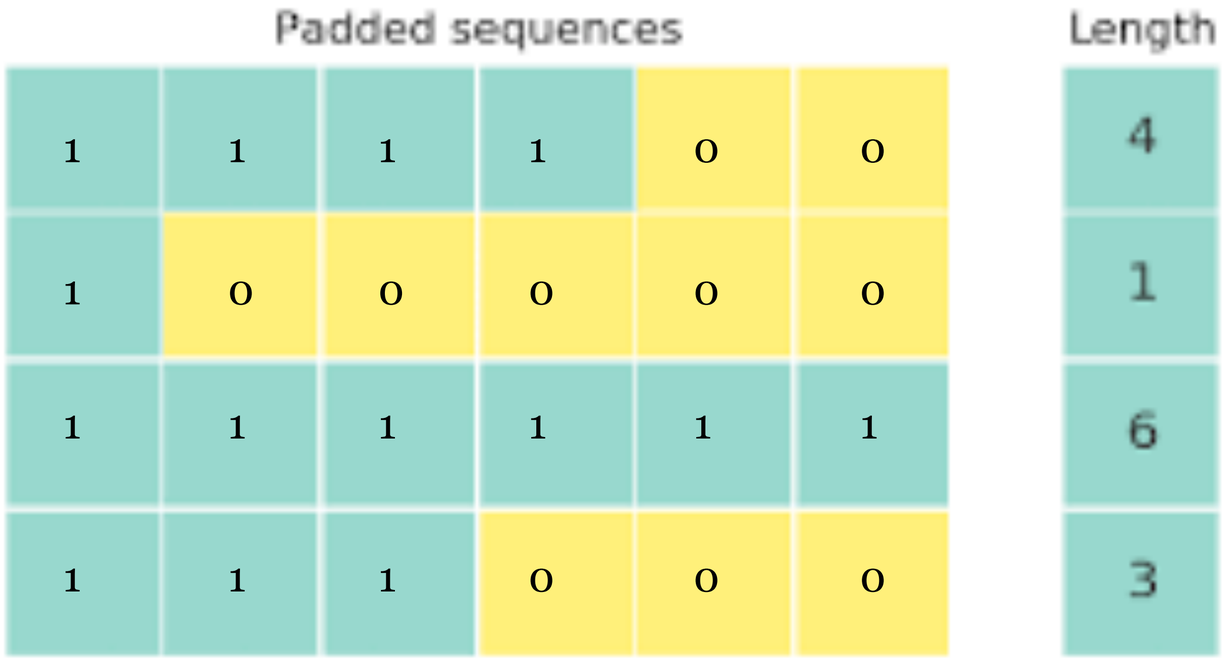


Seq2seq is optimized as a **single system.**
Backpropagation operates "*end-to-end*".

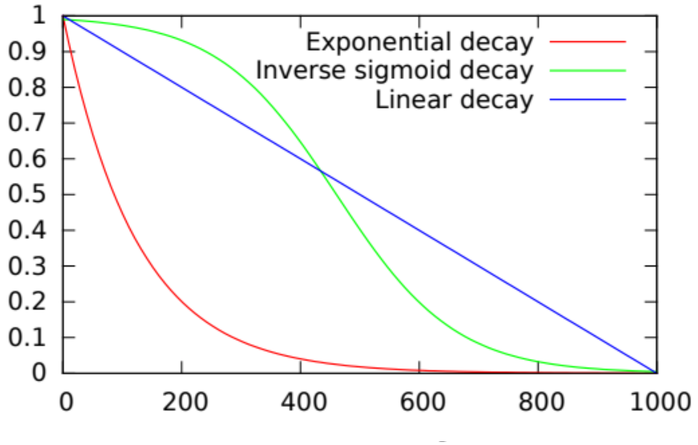
(slide credit: Abigail See)

Remember masking

Use masking to help compute loss for batched sequences

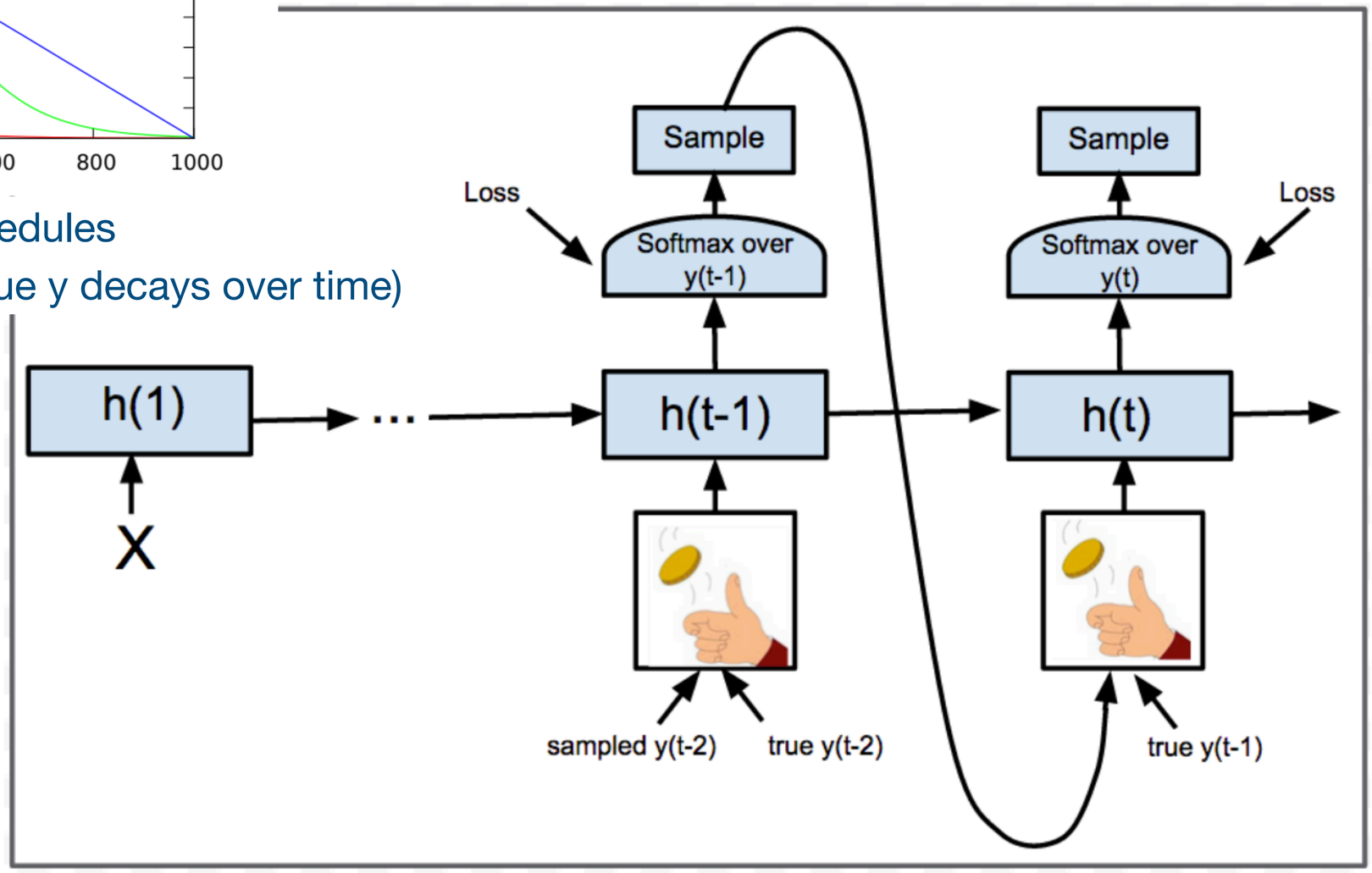


Scheduled Sampling



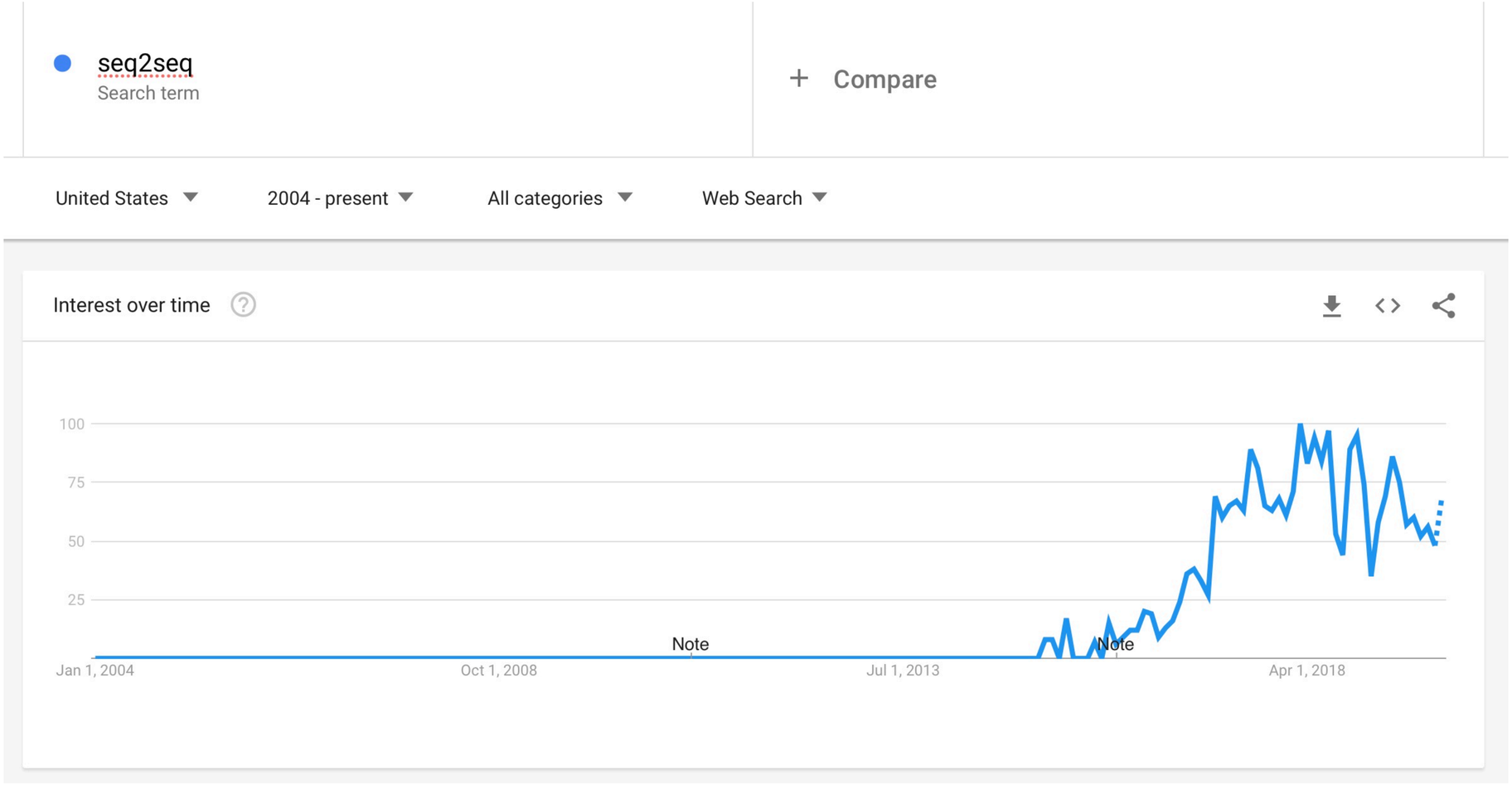
Possible decay schedules

(probability using true y decays over time)

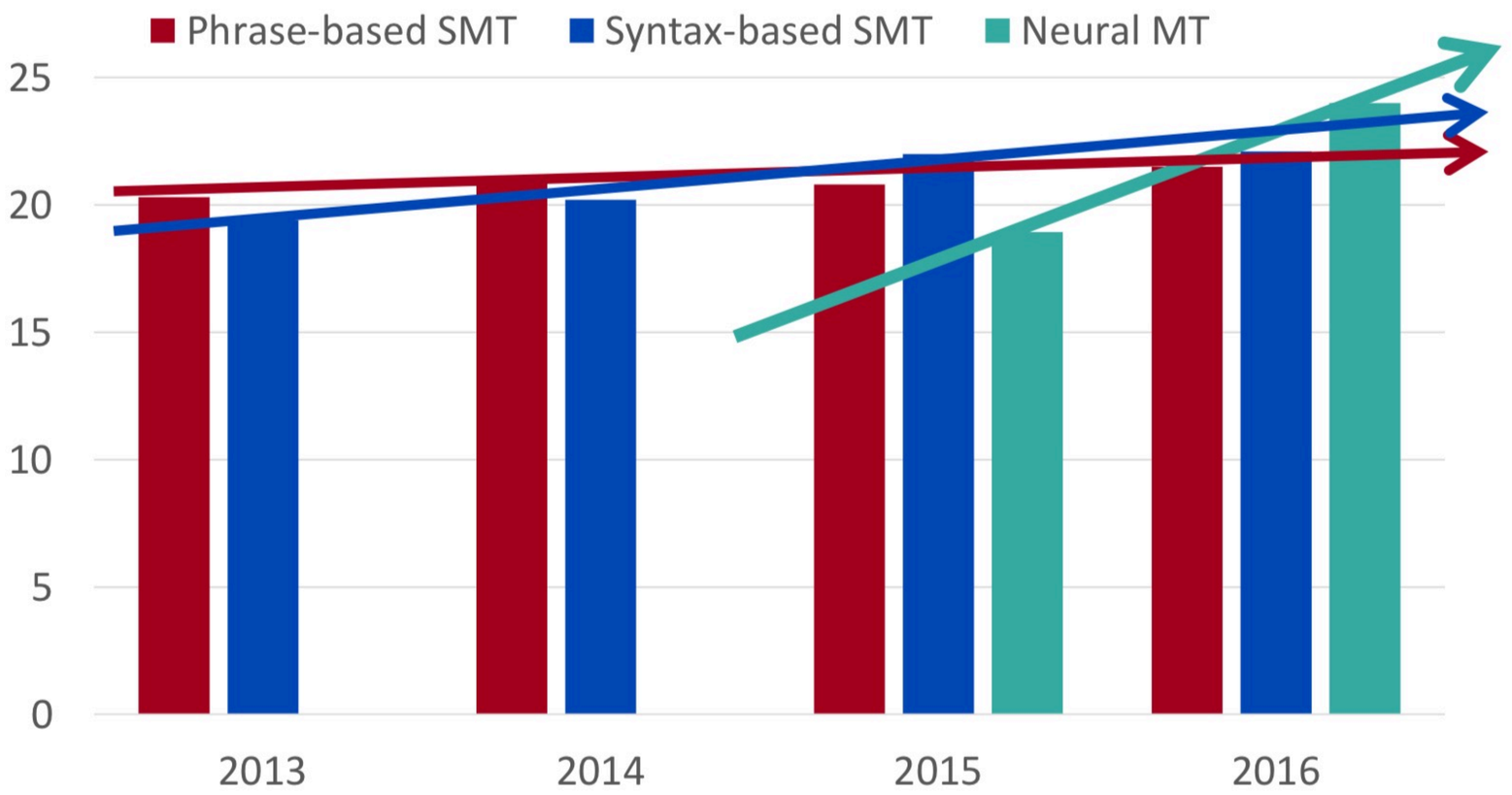


(figure credit: Bengio et al, 2015)

How seq2seq changed the MT landscape



MT Progress



(source: Rico Sennrich)

RESEARCH › PUBLICATIONS ›

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

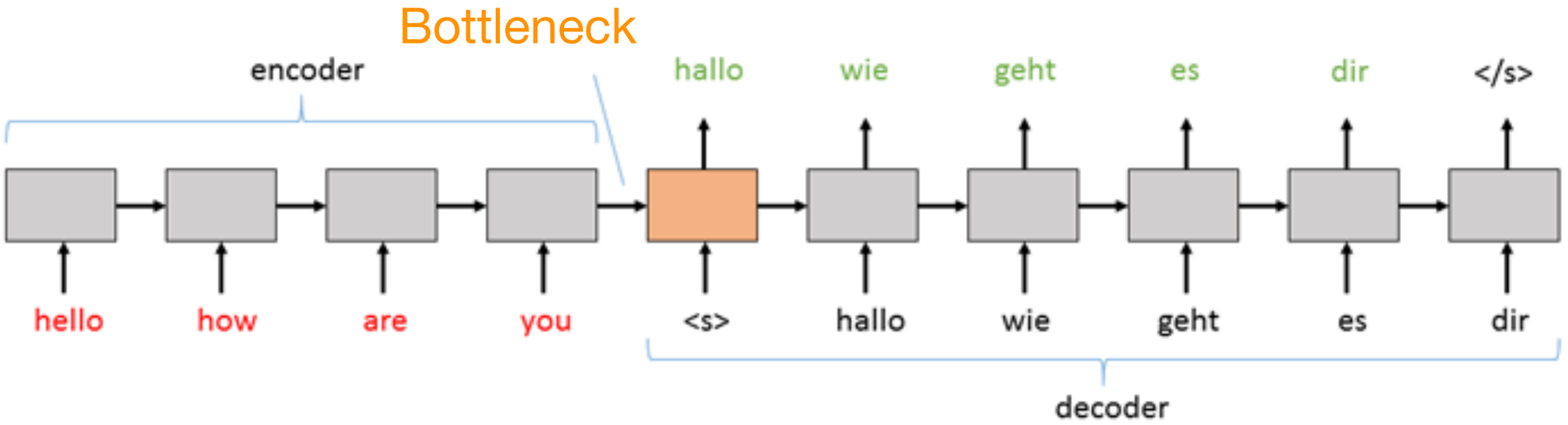
Seq2Seq for more than NMT

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

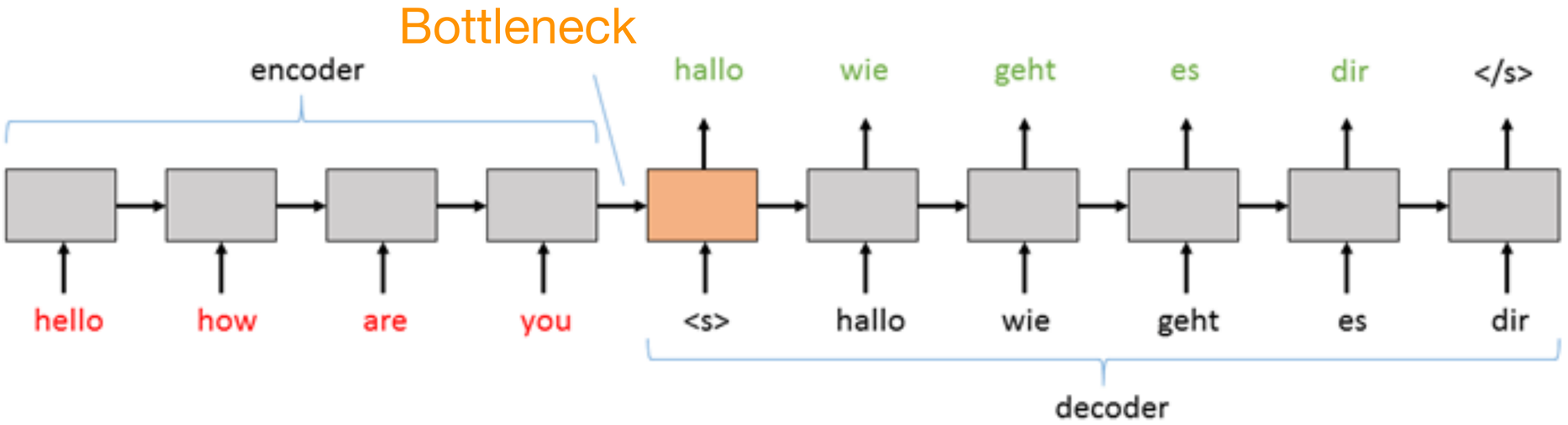
Task/Application	Input	Output
Speech Recognition	Speech Signal	Transcript
Image Captioning	Image	Text
Video Captioning	Video	Text

Issues with vanilla seq2seq



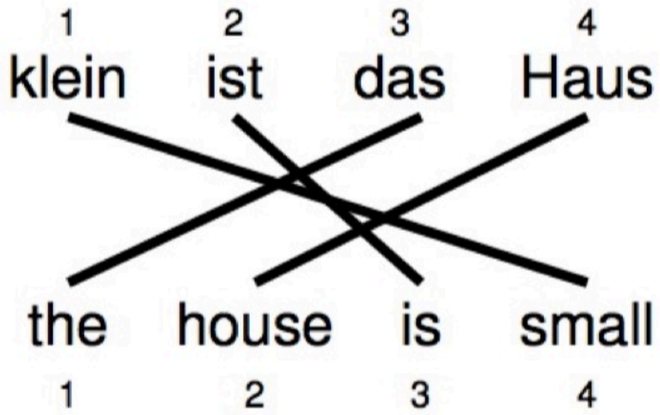
- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- Overfitting

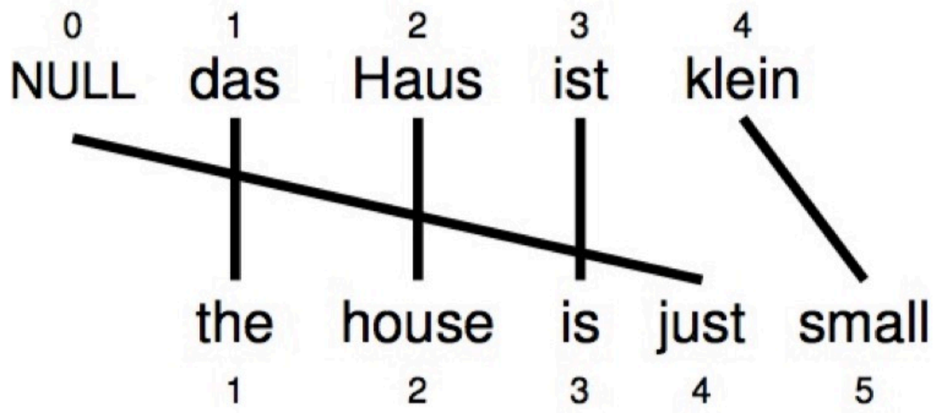
Issues with vanilla seq2seq



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

Remember alignments?





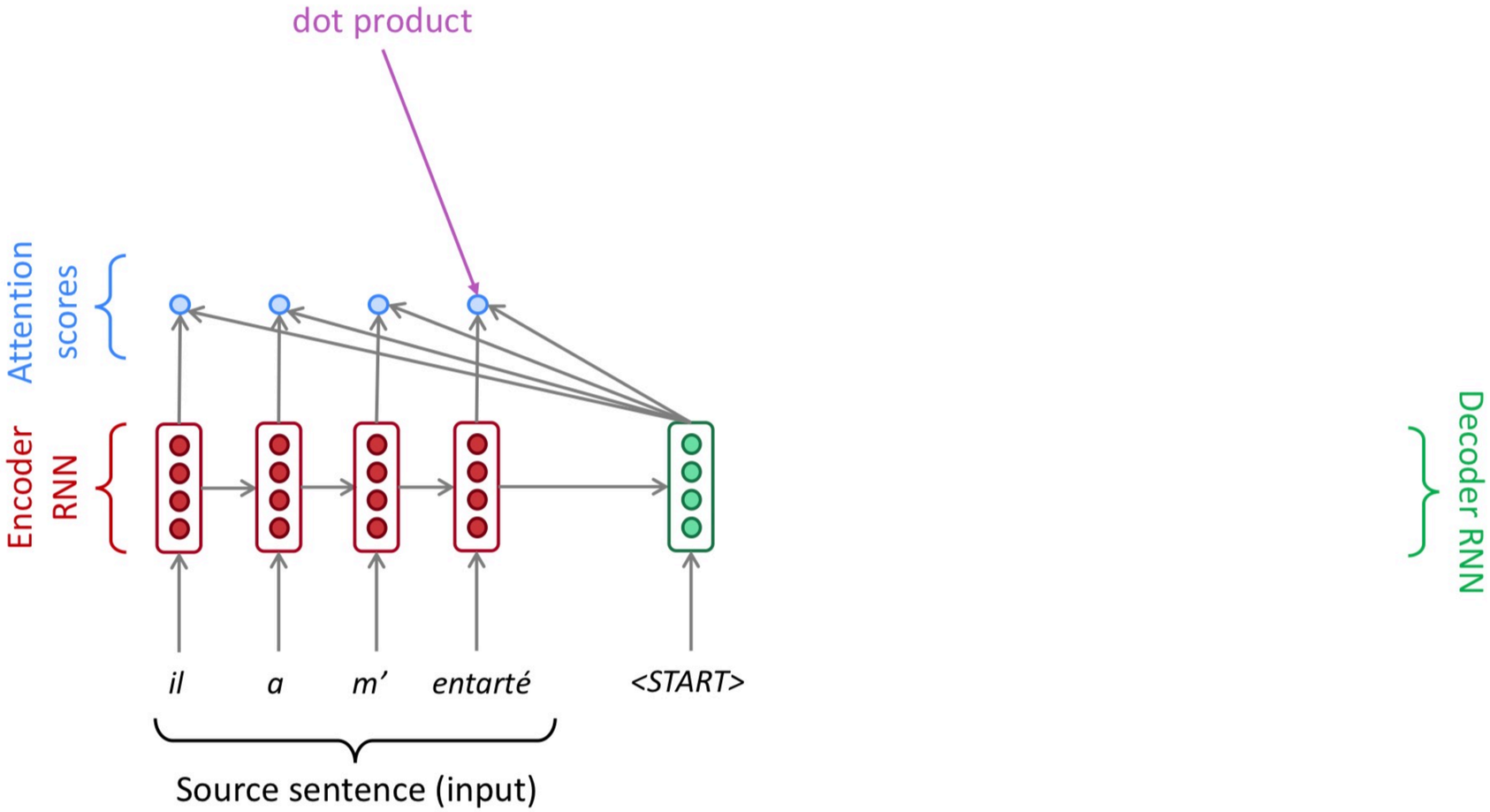
$$\mathbf{a} = (3, 4, 2, 1)^\top$$

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

Attention

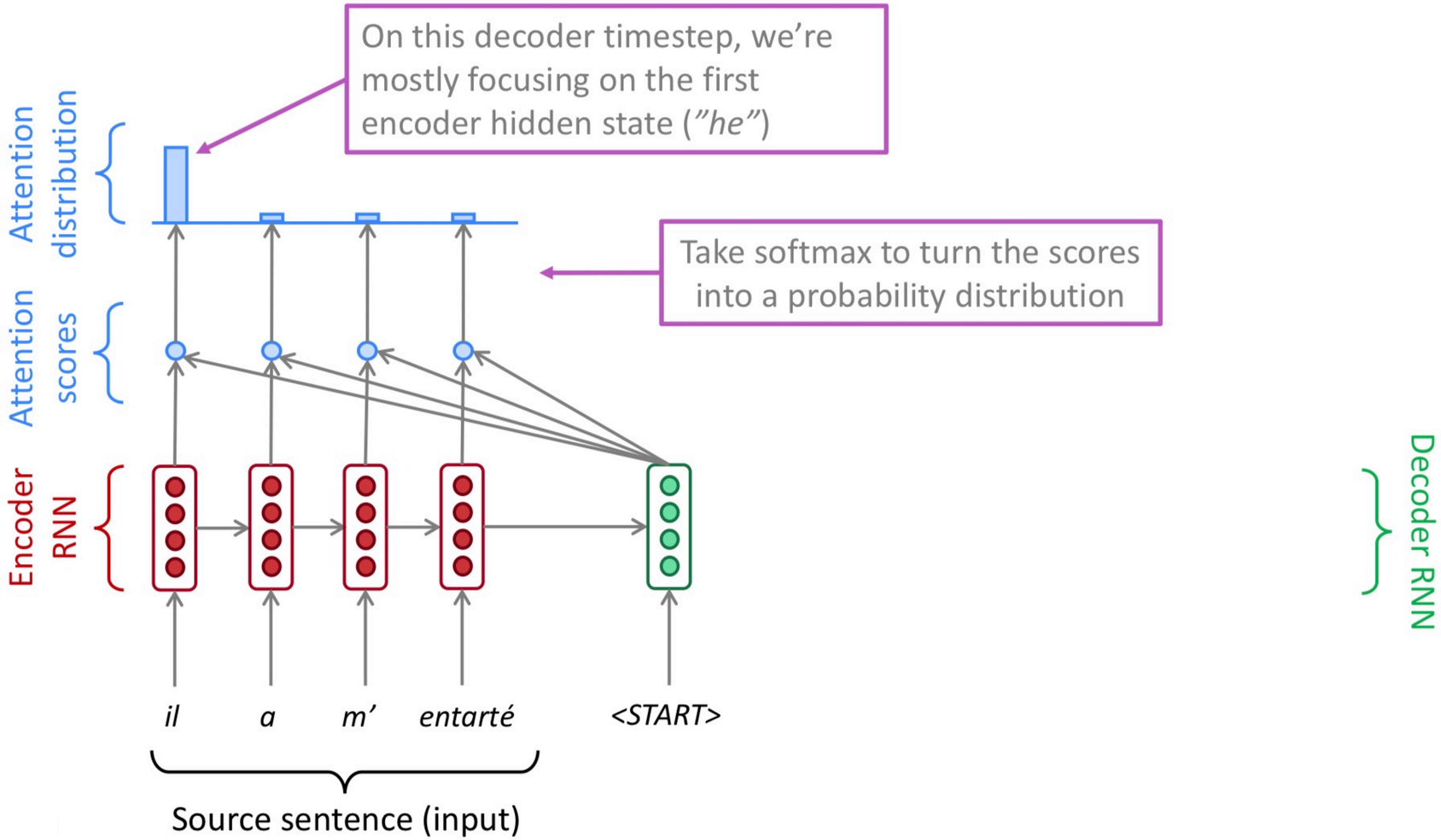
- The neural MT equivalent of alignment models
- Key idea: At each time step during decoding, **focus on a particular part** of source sentence
  - This depends on the decoder's current hidden state (i.e. notion of what you are trying to decode)
  - Usually implemented as a probability distribution over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

Seq2seq with attention



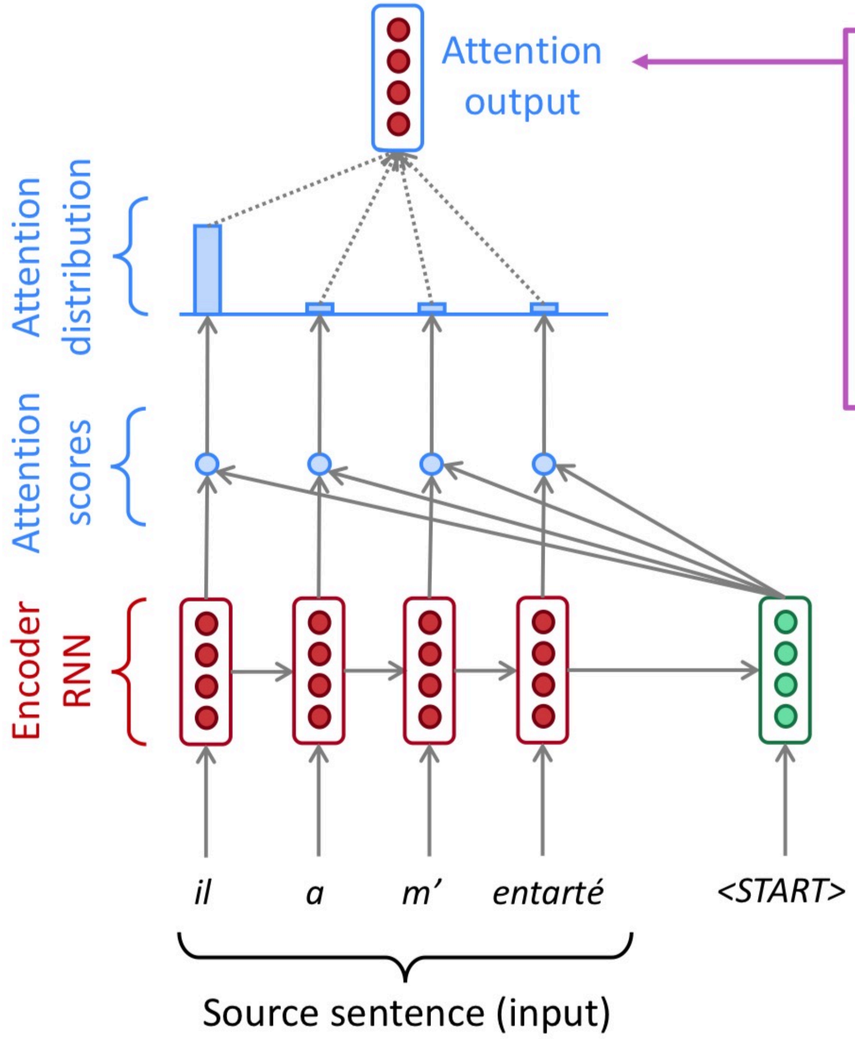
(slide credit: Abigail See)

Seq2seq with attention



(slide credit: Abigail See)

Seq2seq with attention



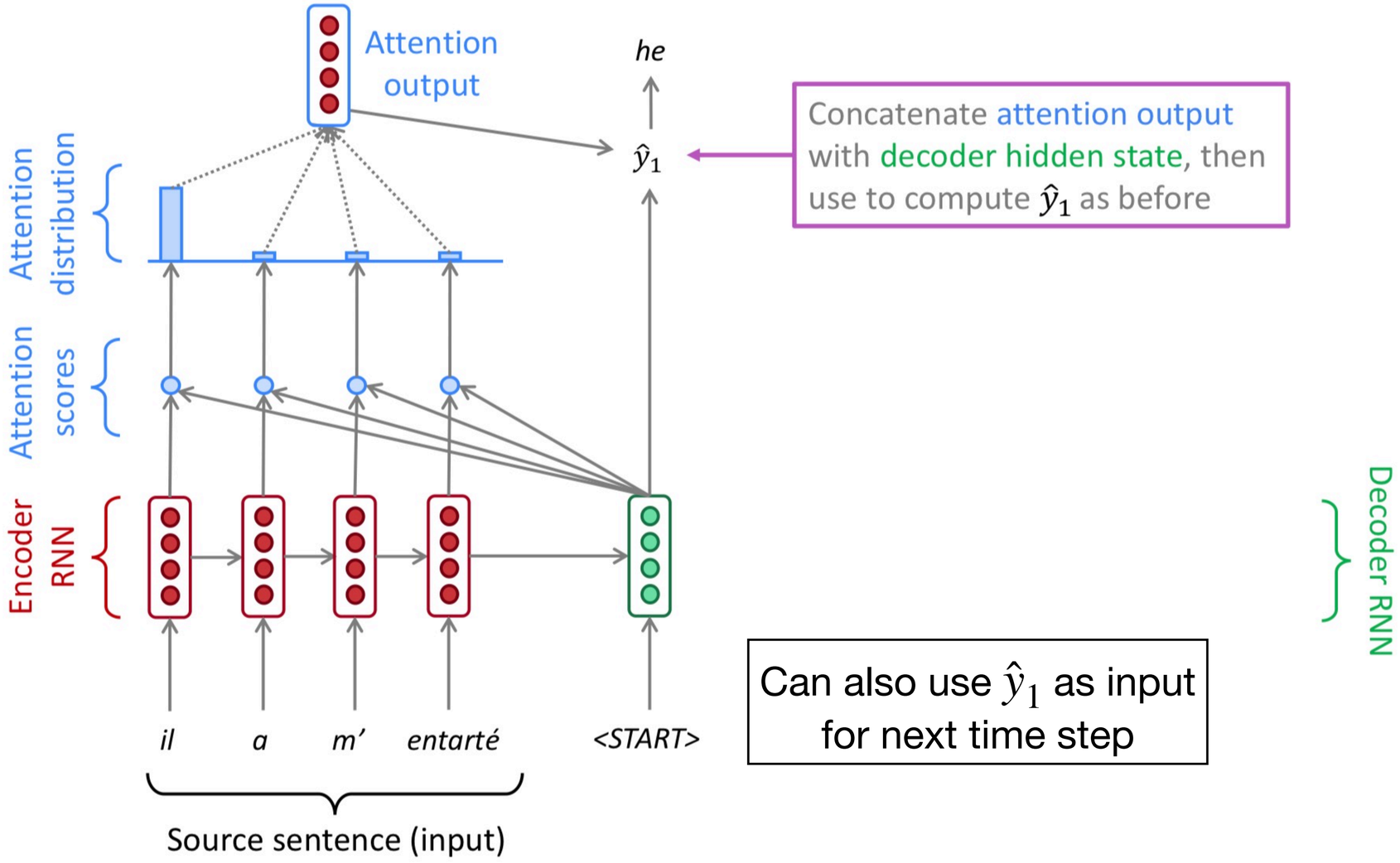
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

Decoder RNN

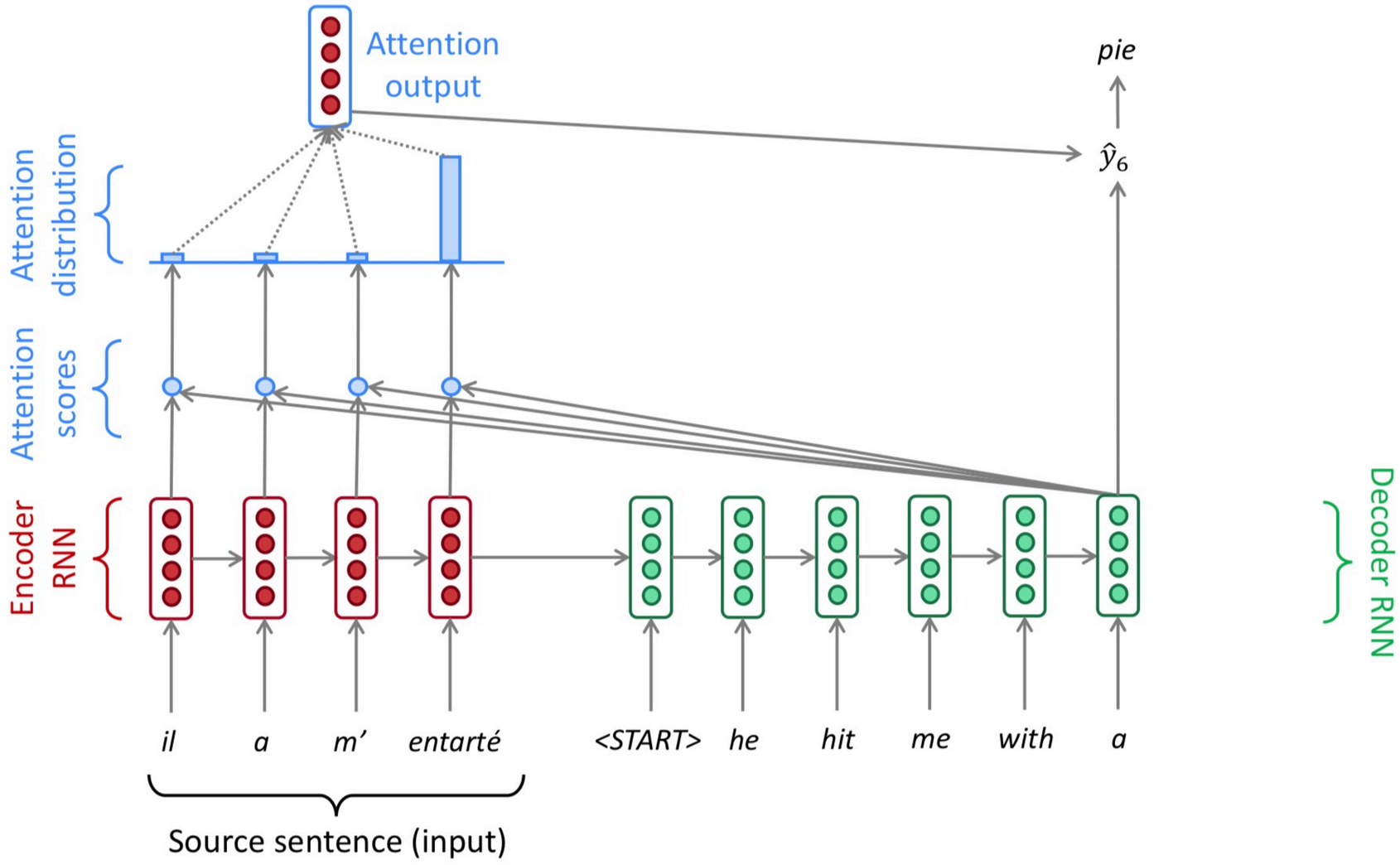
(slide credit: Abigail See)

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** (assumes equal dimensions for a and b:

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

2. **Multiplicative attention:**

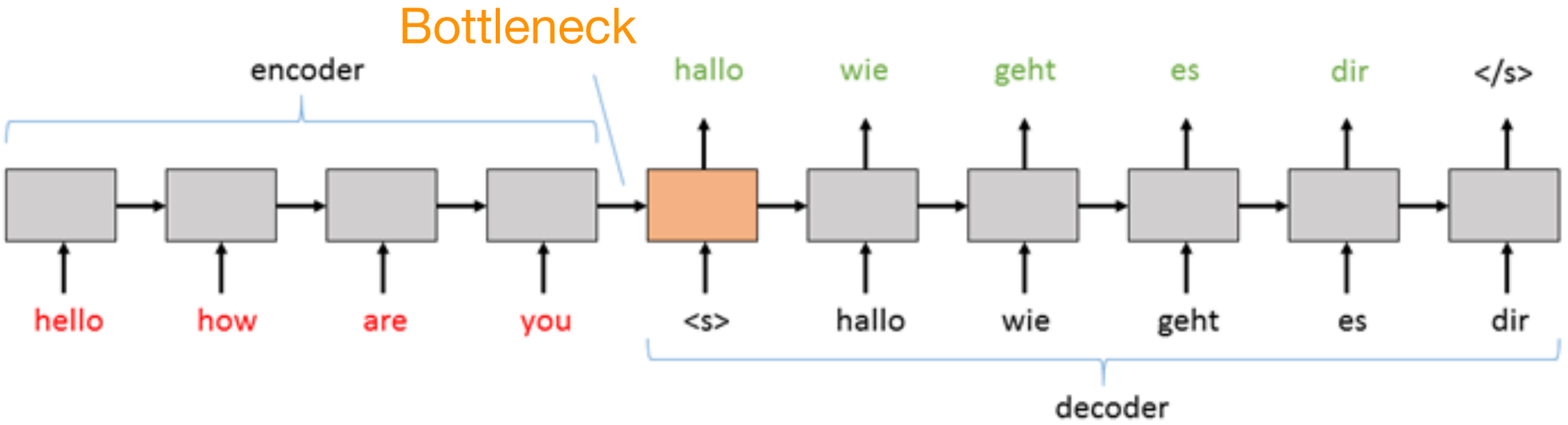
$$g(h_i, z) = z^T W h_i \in \mathbb{R}, \text{ where } W \text{ is a weight matrix}$$

3. **Additive attention:**

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

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

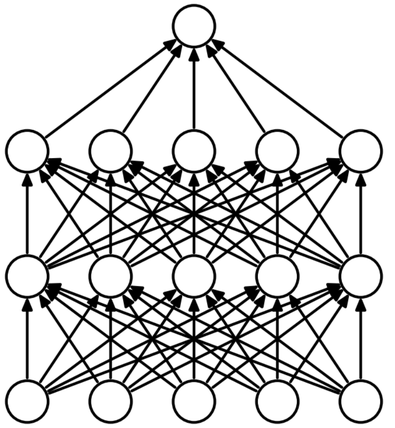
Issues with vanilla seq2seq



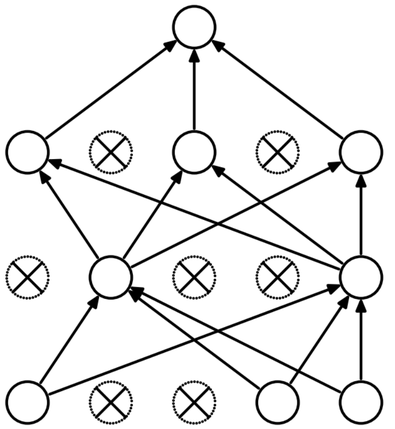
- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- **Overfitting**

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

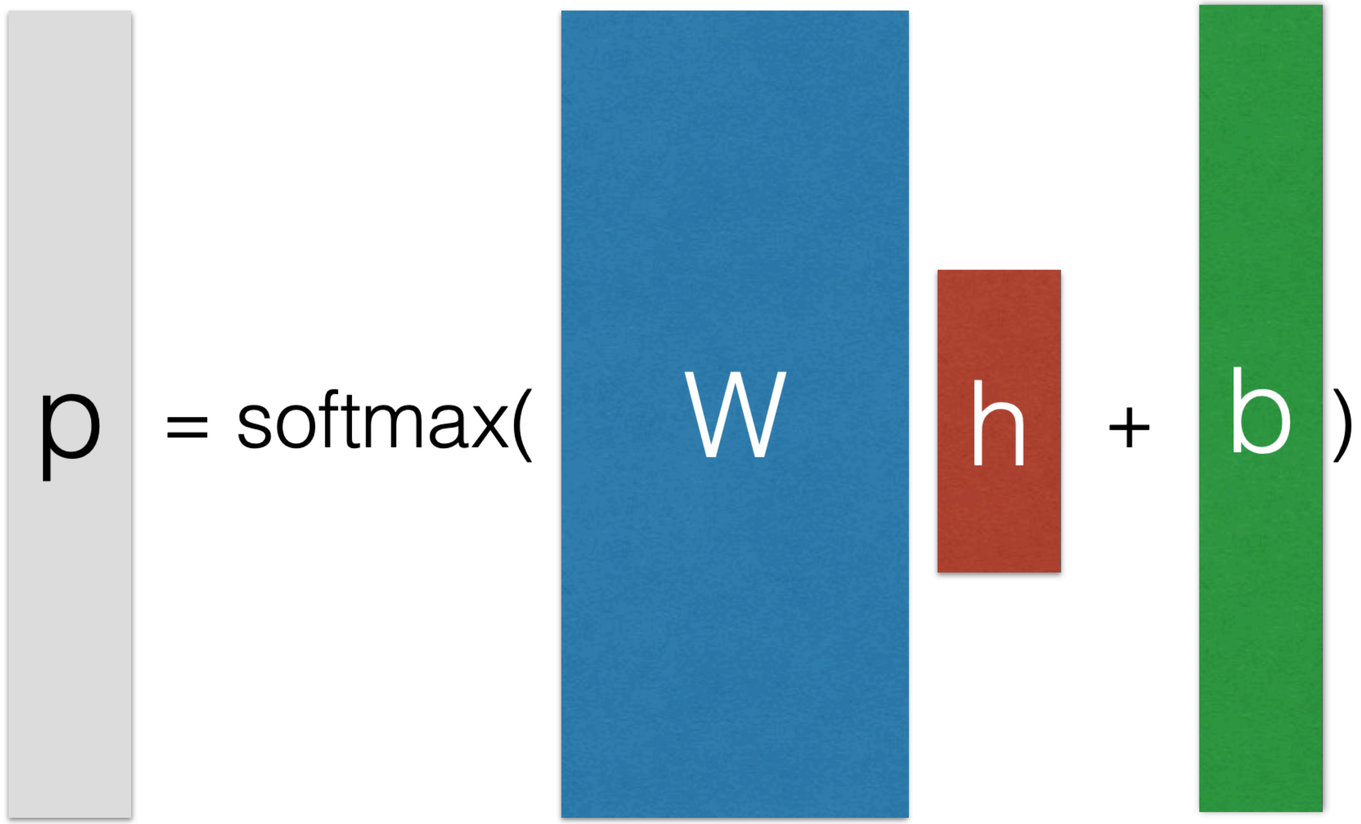
- English vocabulary size: 10K to 100K

Approximate Softmax

- Negative Sampling
- Structured softmax
- Embedding prediction

Negative Sampling

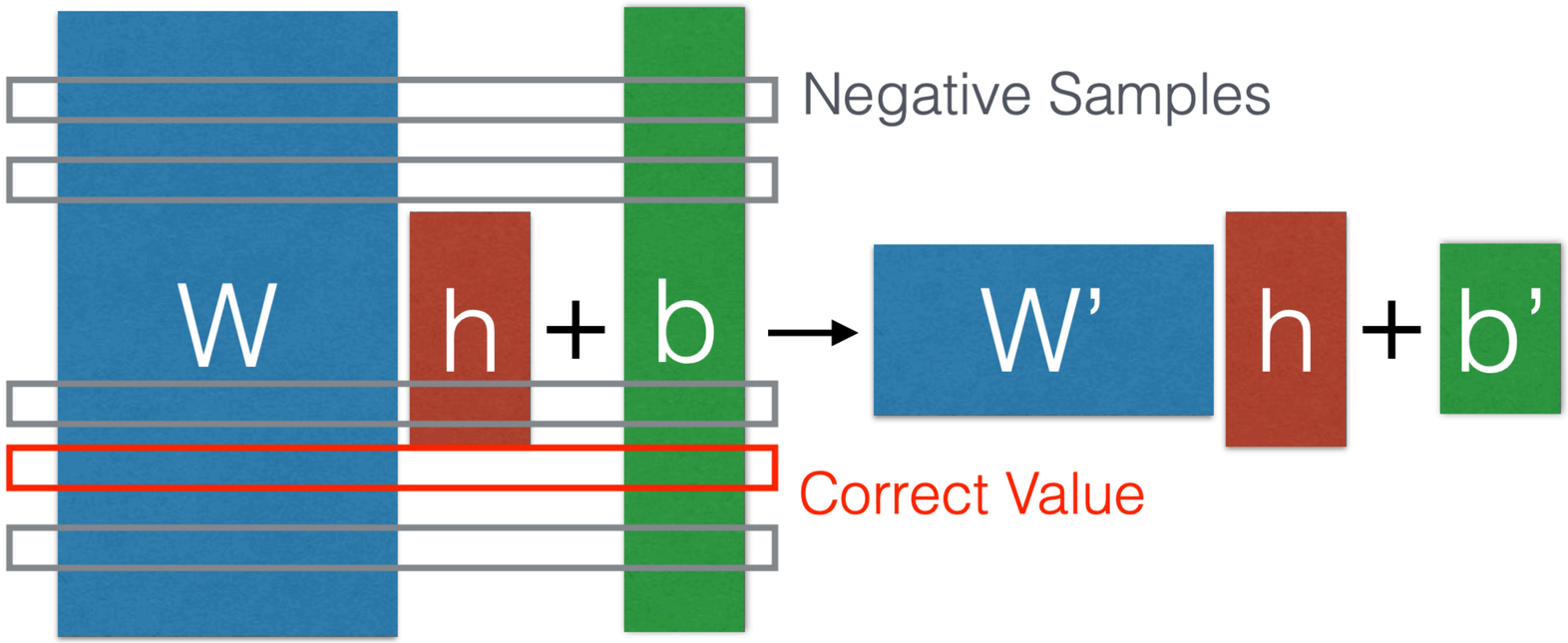
- Softmax is expensive when vocabulary size is large



(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:

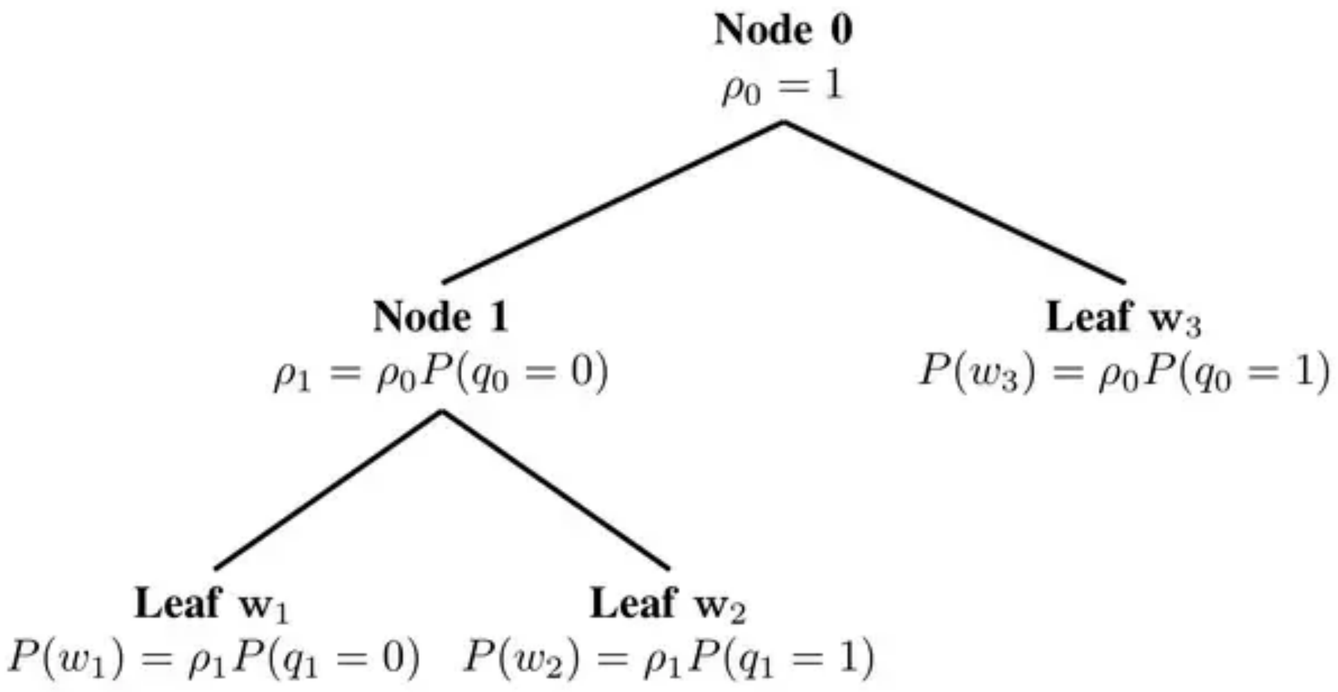
Importance Sampling (Bengio and Senecal 2003)

Noise Contrastive Estimation (Mnih & Teh 2012)

(figure credit: Graham Neubig)

Hierarchical softmax

(Morin and Bengio 2005)



(figure credit: Quora)

Class based softmax

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

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

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

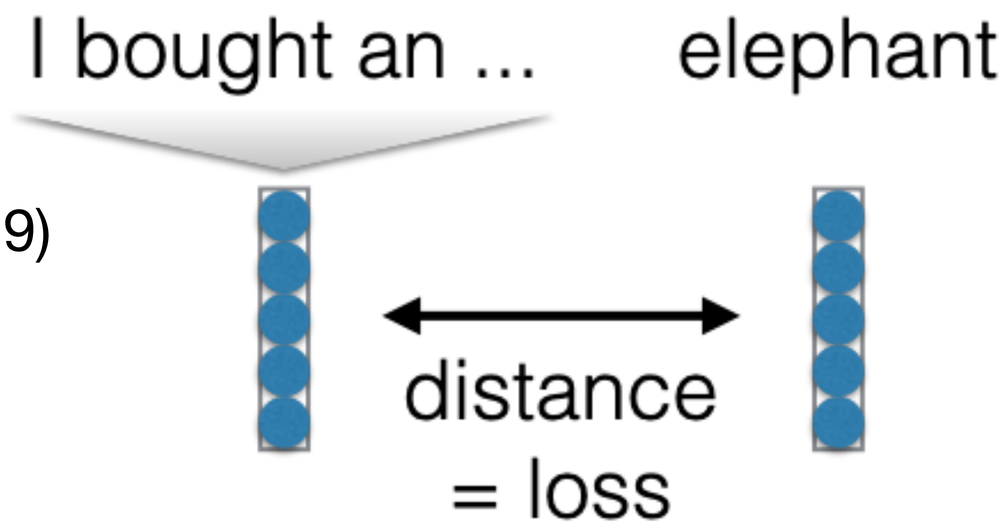
(figure credit: Graham Neubig)

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

(Gooding 2001, Mikolov et al 2011)

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



(slide credit: Graham Neubig)

Generation

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

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

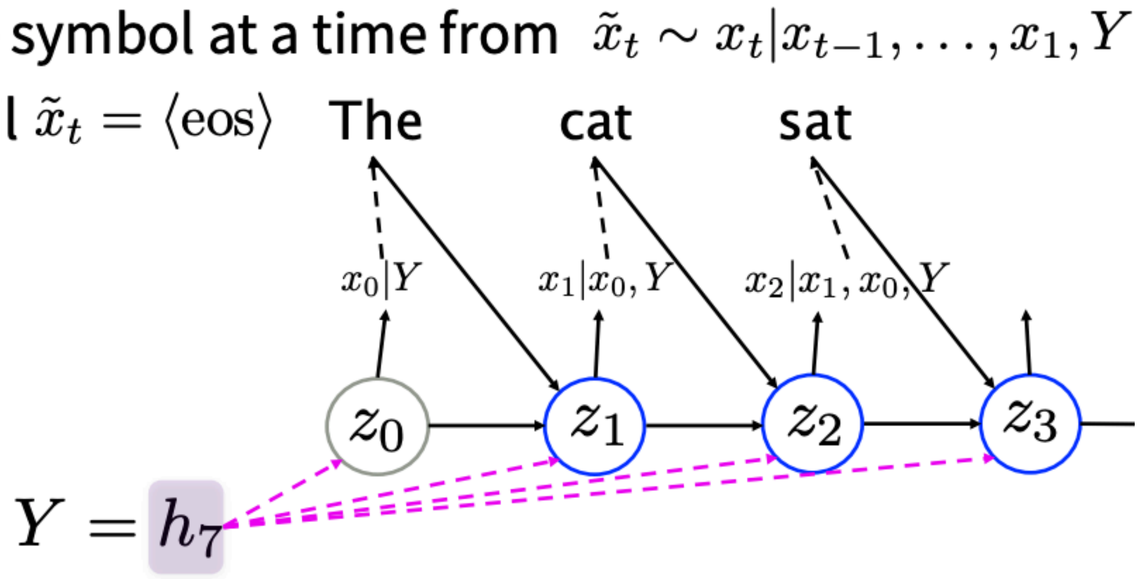
- Argmax: Try to generate the *best* sentence, the sentence with the *highest* probability

Decoding Strategies

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

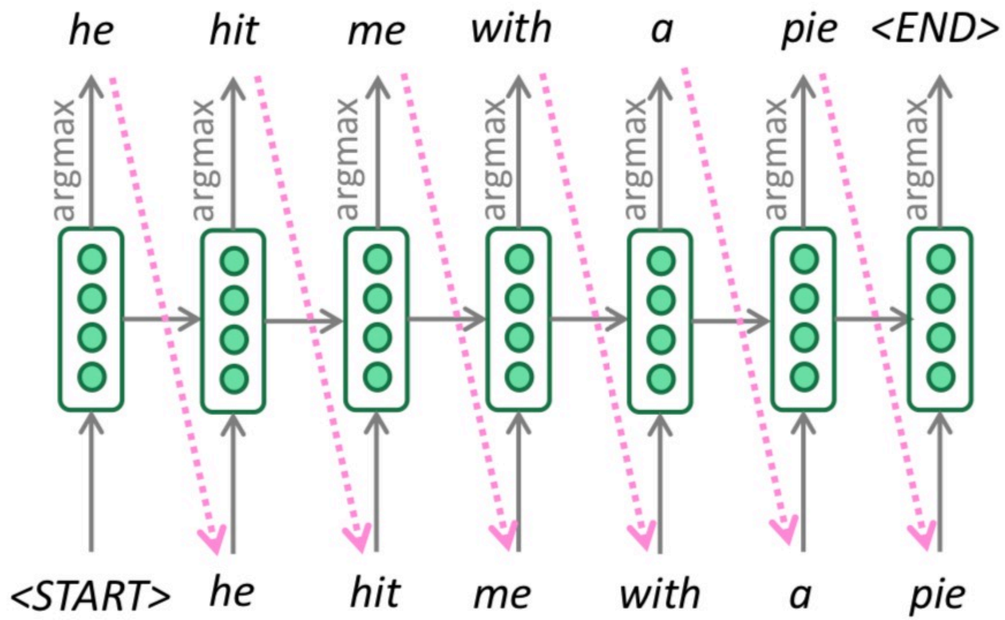
Ancestral Sampling

- 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)

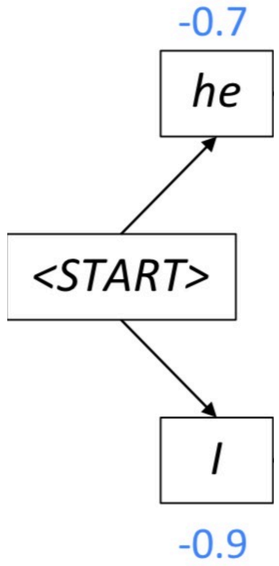
- **Key idea:** At every step, keep track of the k most probable partial translations (hypotheses)
- Score of each hypothesis = log probability

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

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

Beam decoding

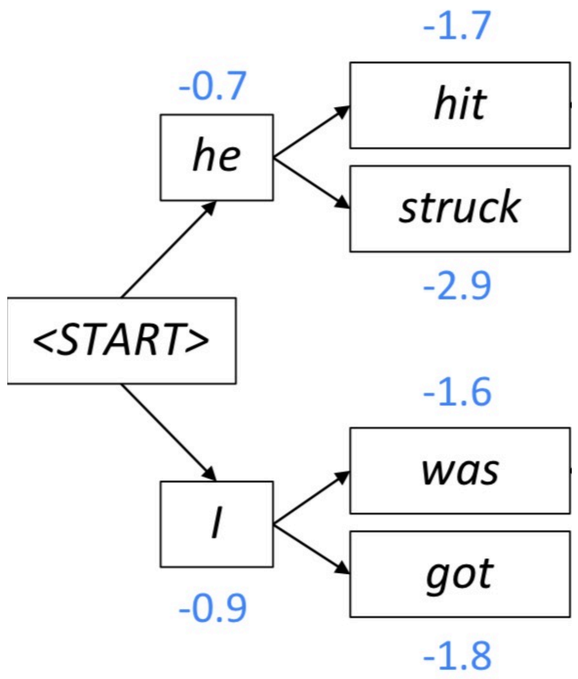
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

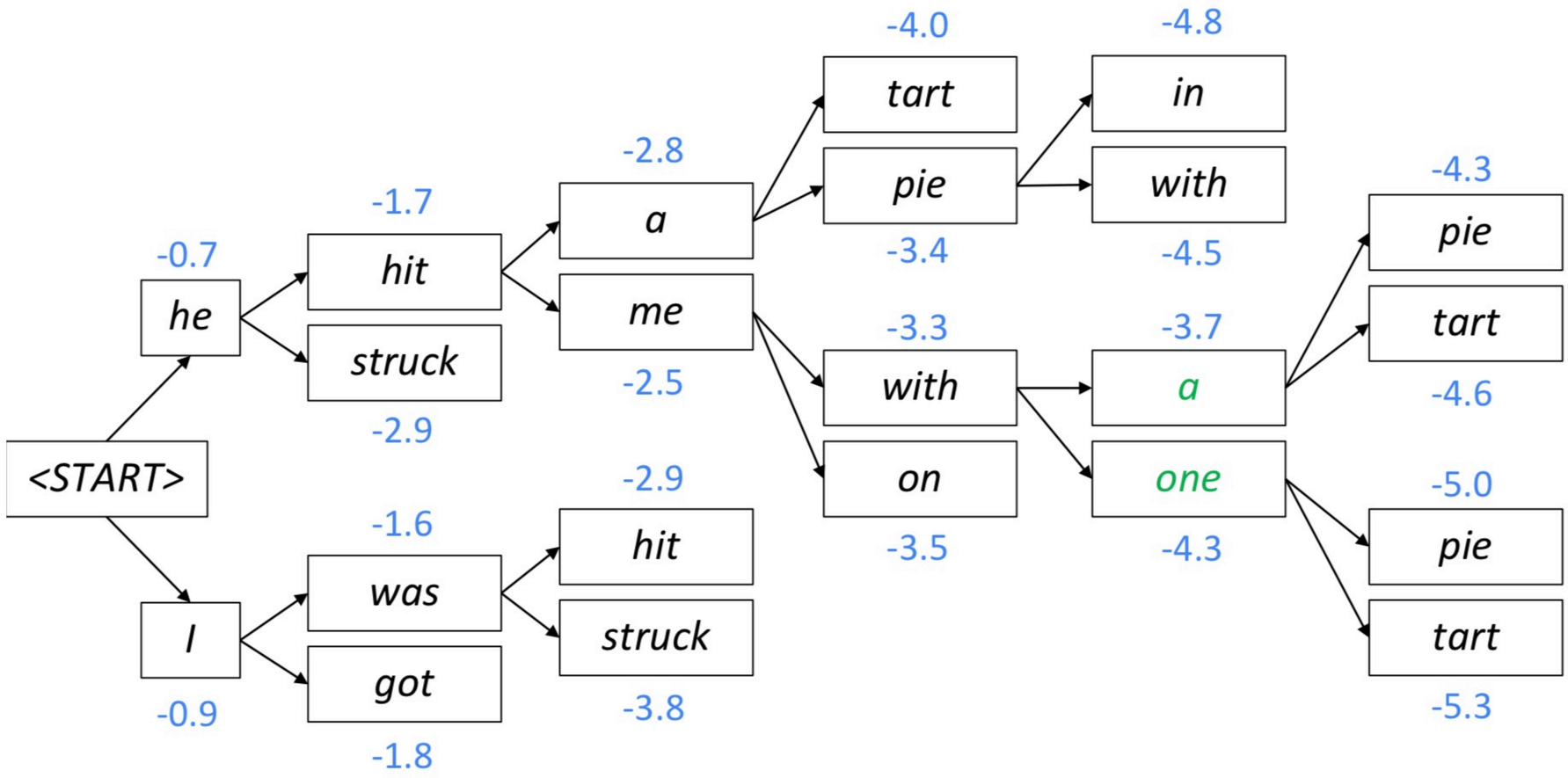
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

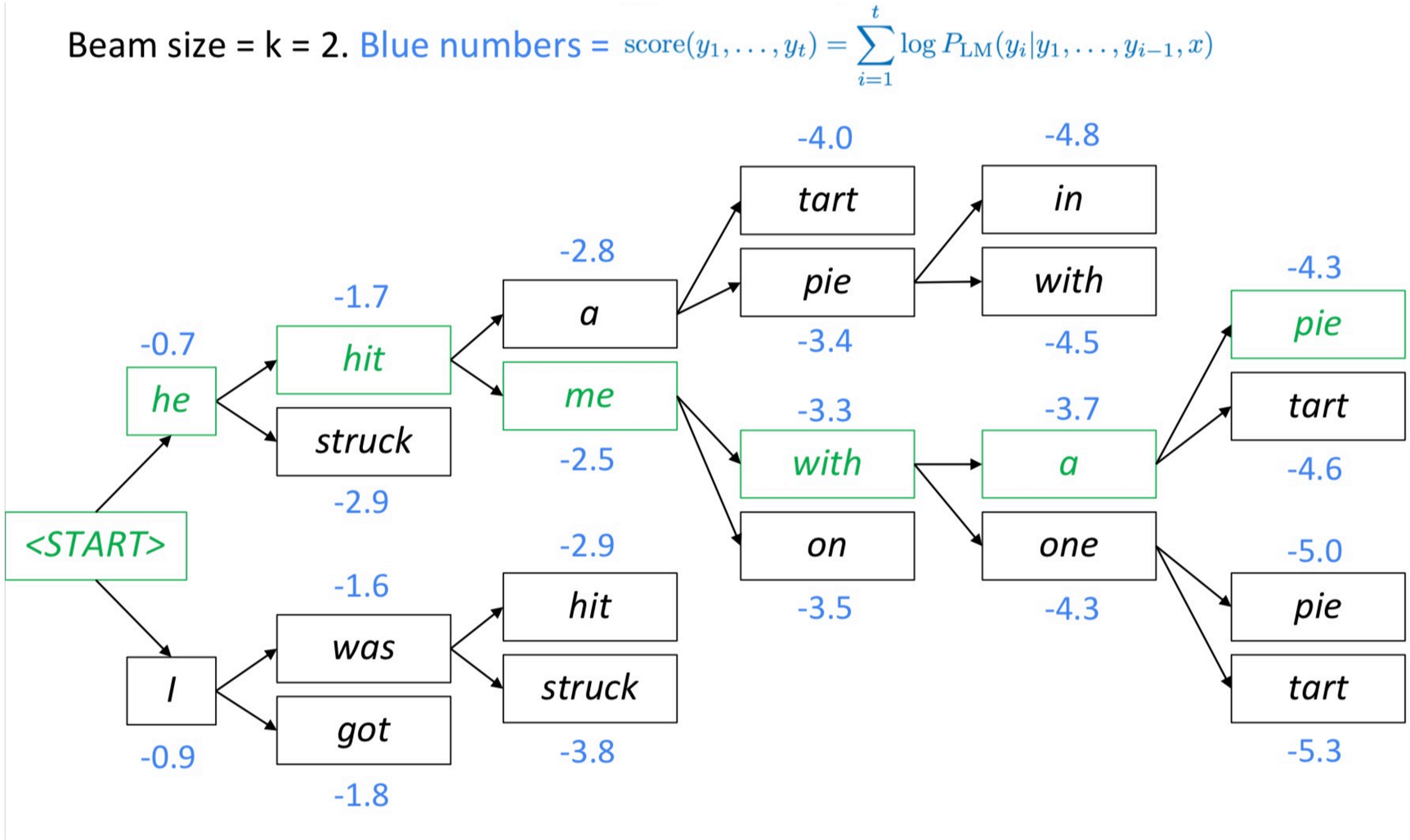
Beam decoding

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



(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

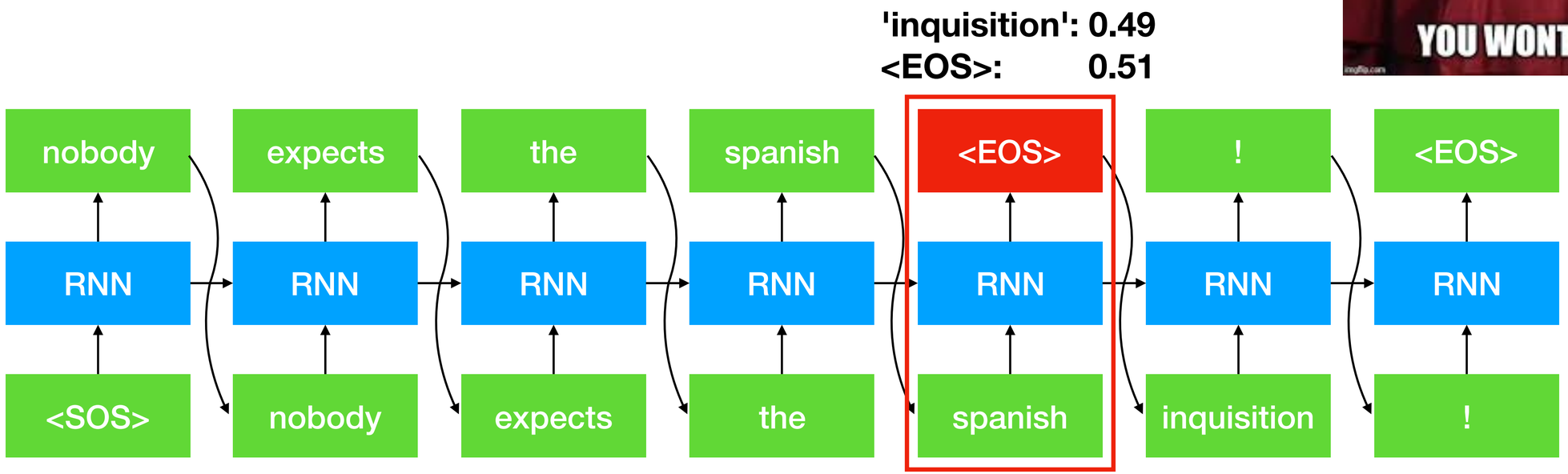
$$\frac{1}{T} \sum_{t=1}^{T} \log P(y_t \,|\, y_1, \ldots, y_{t-1}, x_1, \ldots, x_n)$$

  - Otherwise shorter hypotheses have higher scores

Under translation

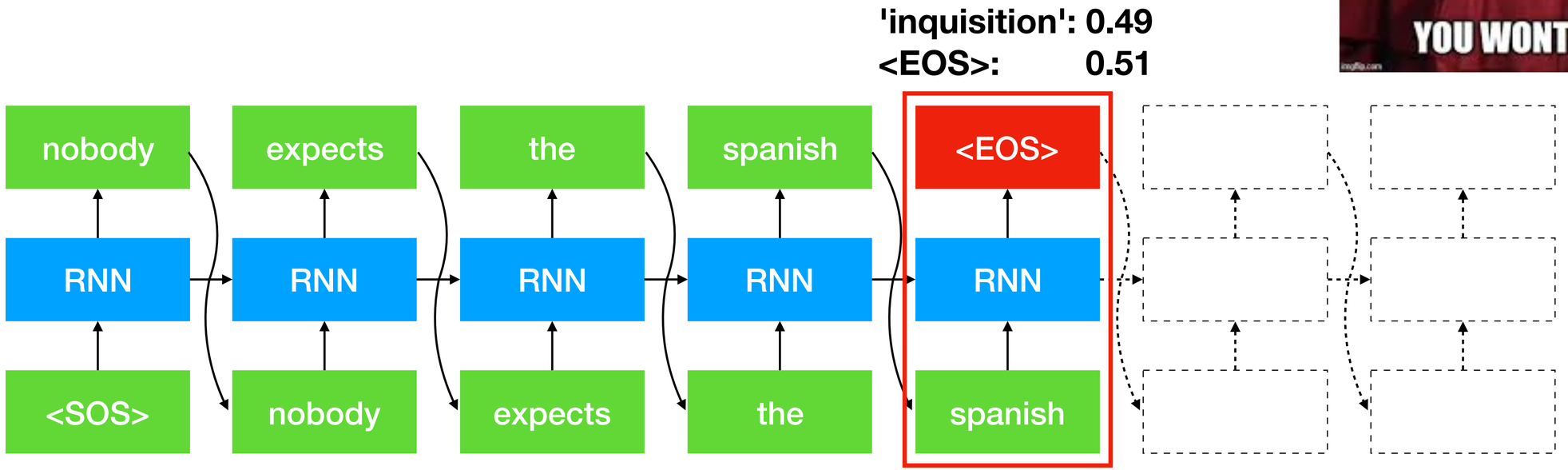
- Crucial information are left untranslated; premature <EOS> generation
  - Beam search
  - Ensemble
  - Coverage models

Under-trans<EOS>



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

Under-trans<EOS>



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

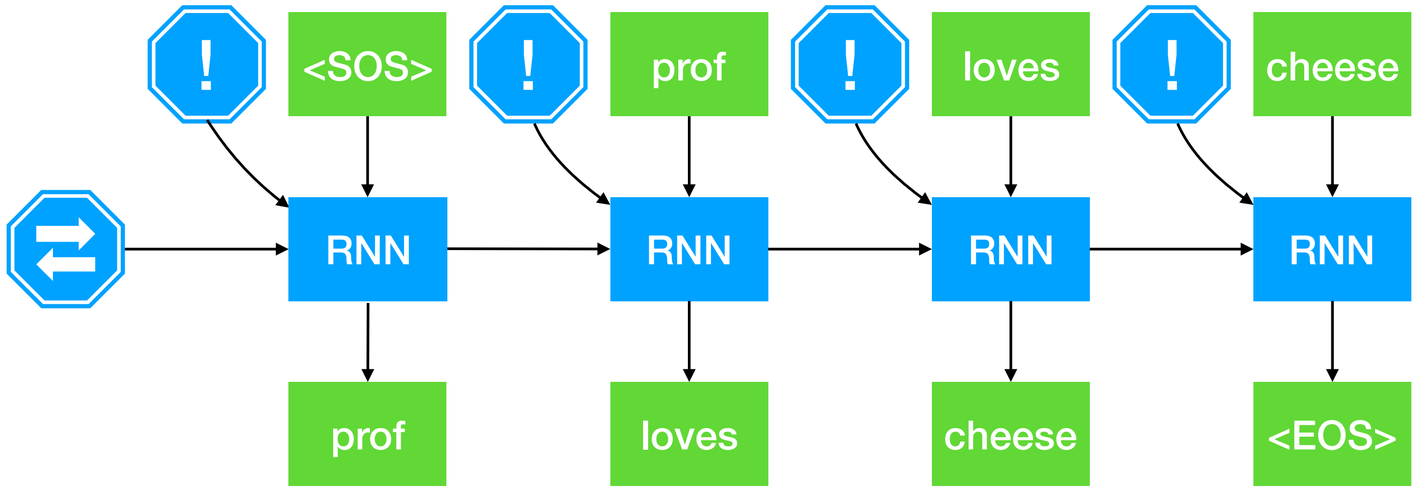
Ensemble

- 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

Concept

P2
Ensemble

Ensemble

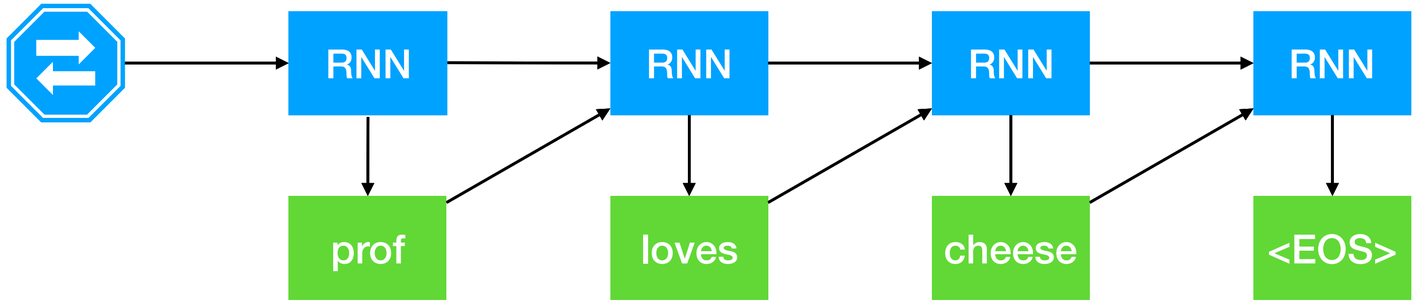


Demo

P2
Ensemble

Ensemble

seq2seq_E049.pt



Demo

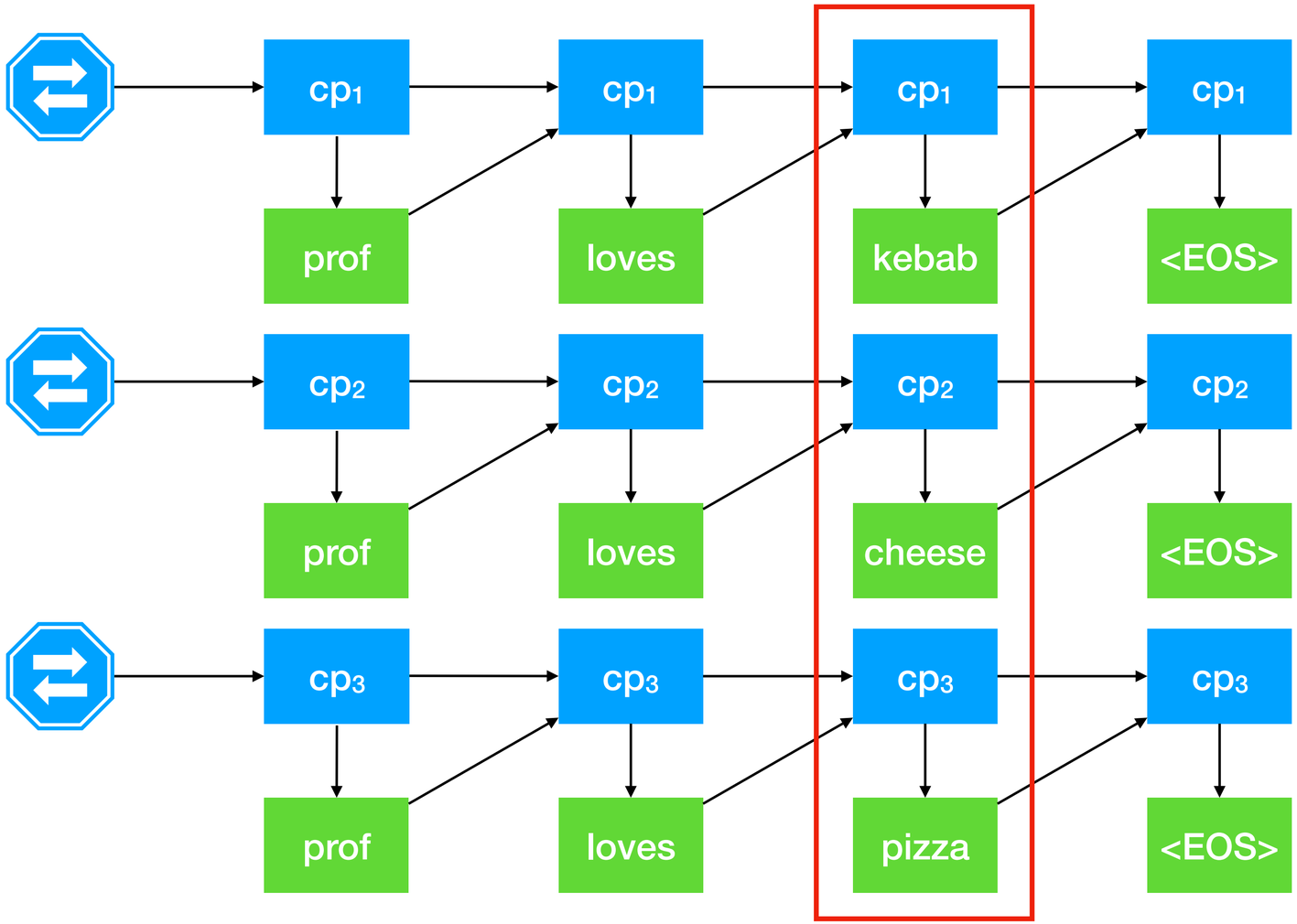
P2
Ensemble

Ensemble

seq2seq_E049.pt

seq2seq_E048.pt

seq2seq_E047.pt



Demo

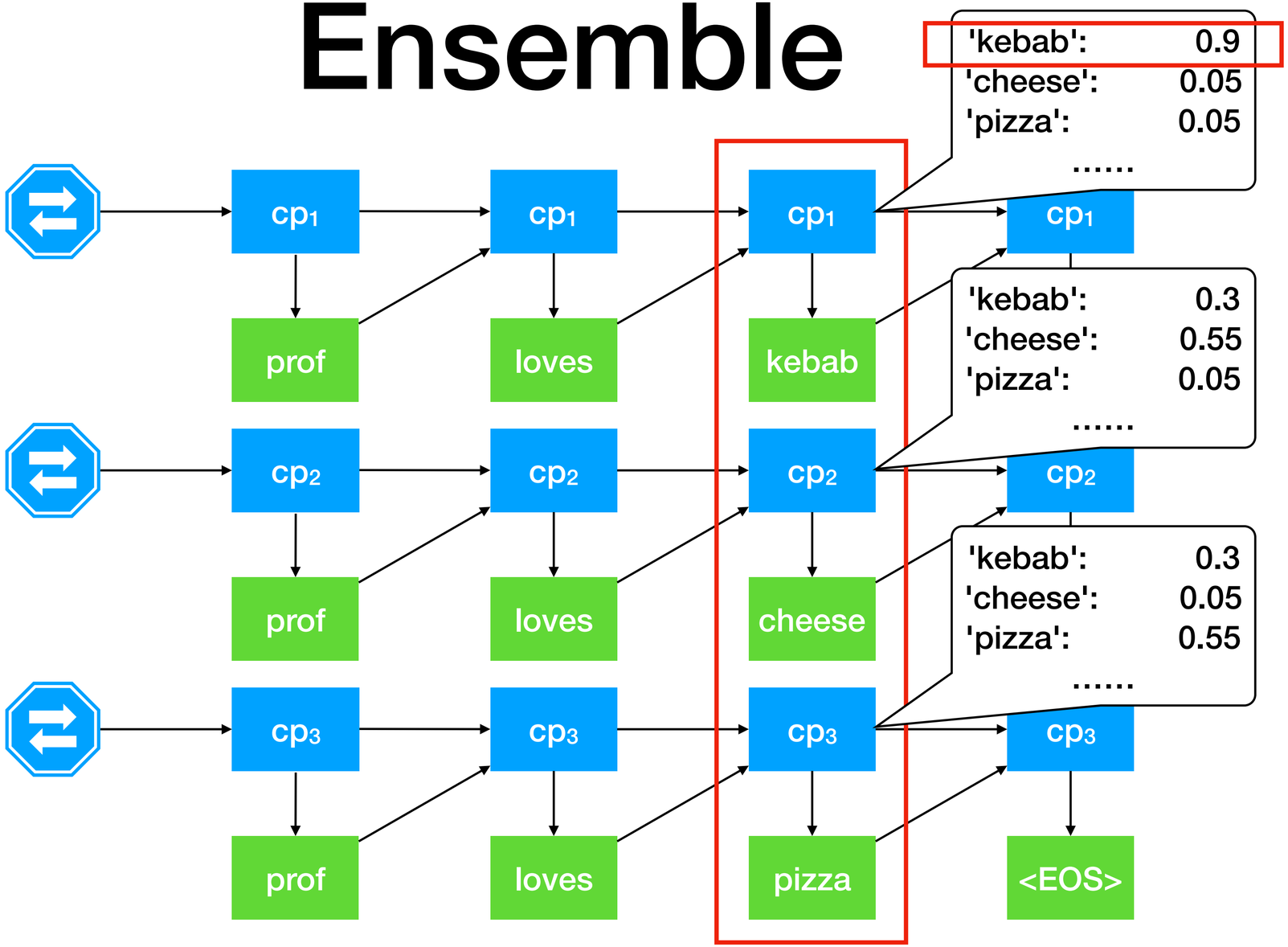
P2
Ensemble

Ensemble

seq2seq_E049.pt

seq2seq_E048.pt

seq2seq_E047.pt



Demo

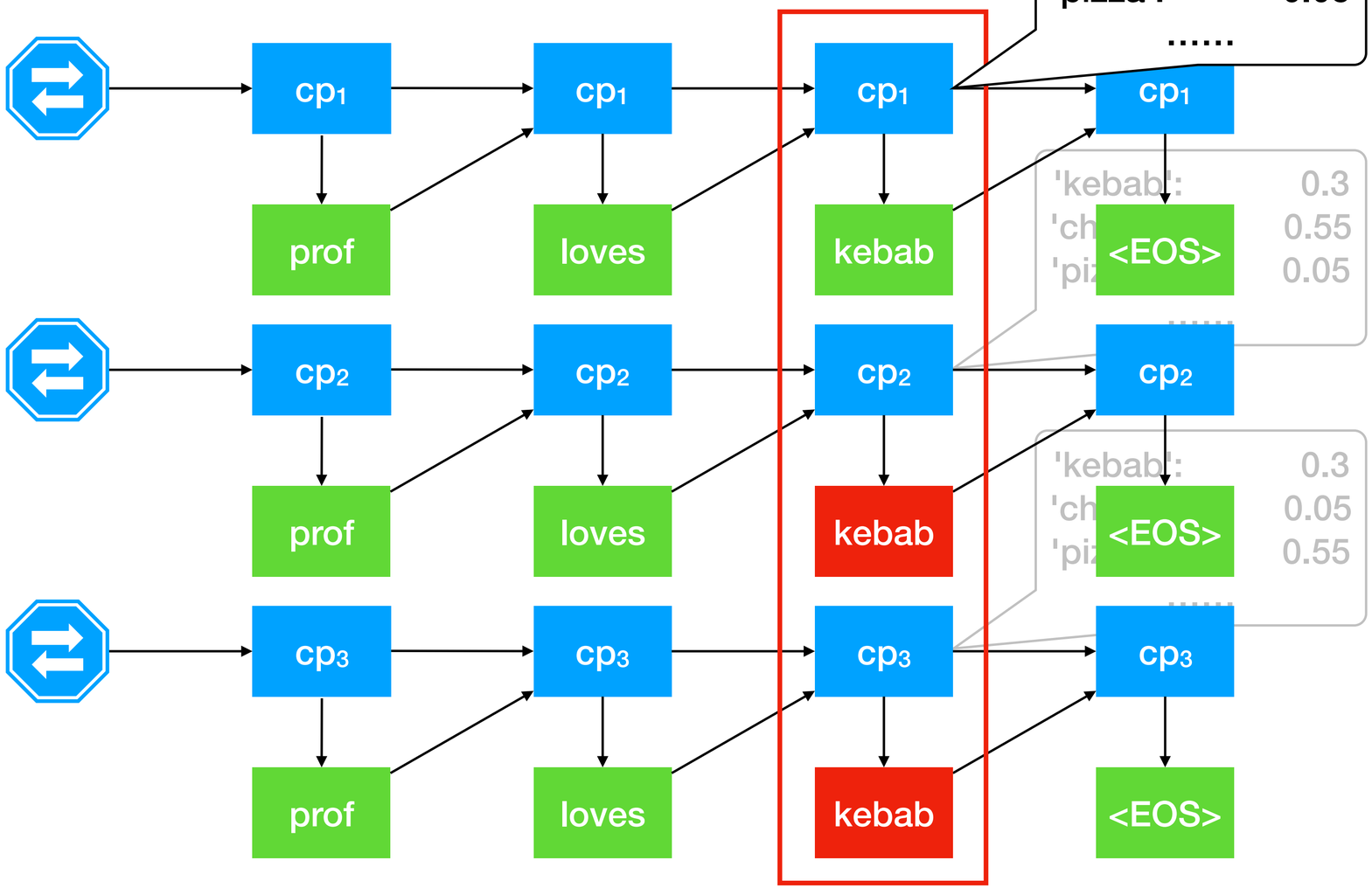
P2
Ensemble

Ensemble

seq2seq_E049.pt

seq2seq_E048.pt

seq2seq_E047.pt



Demo

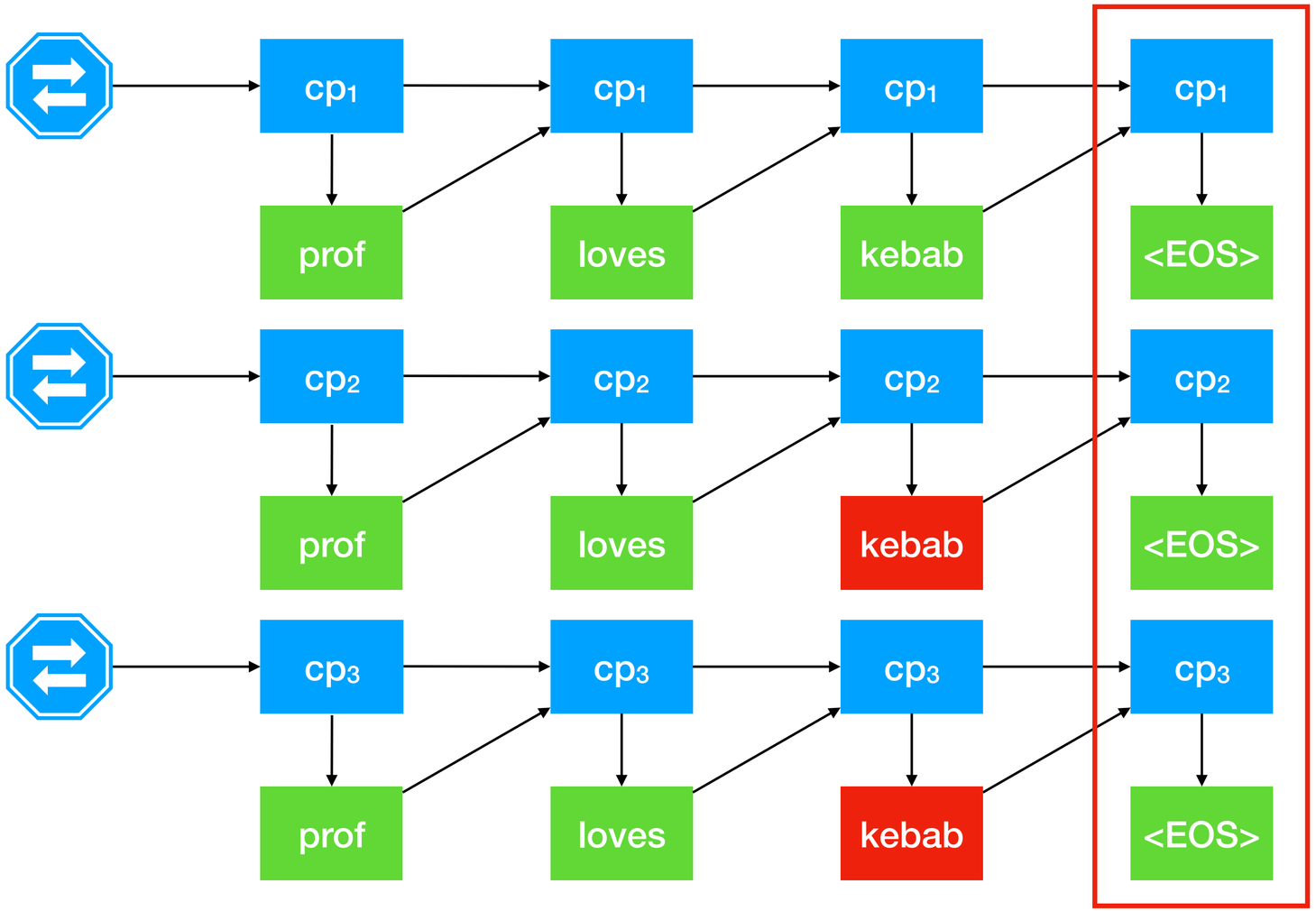
P2
Ensemble

Ensemble

seq2seq_E049.pt

seq2seq_E048.pt

seq2seq_E047.pt



Demo

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

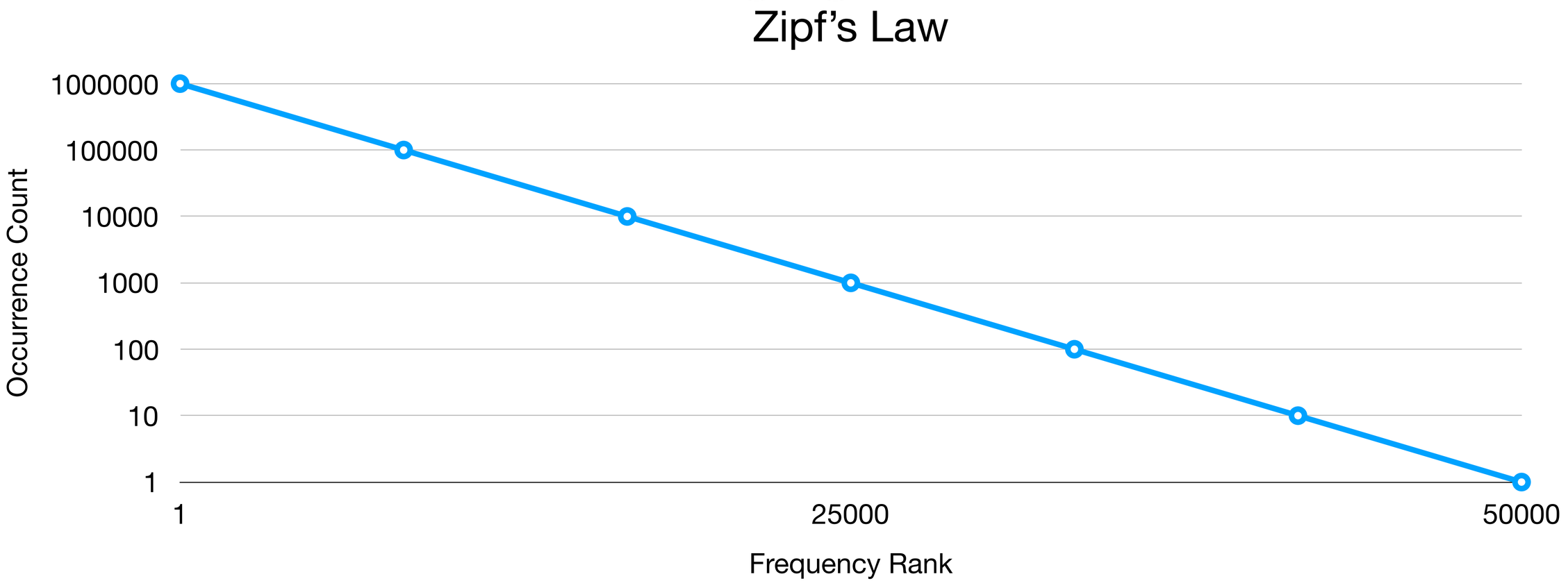
Existing challenges with NMT

- 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 word are translated correctly, rare words are not
  - In any corpus, word frequencies are exponentially unbalanced

OOV



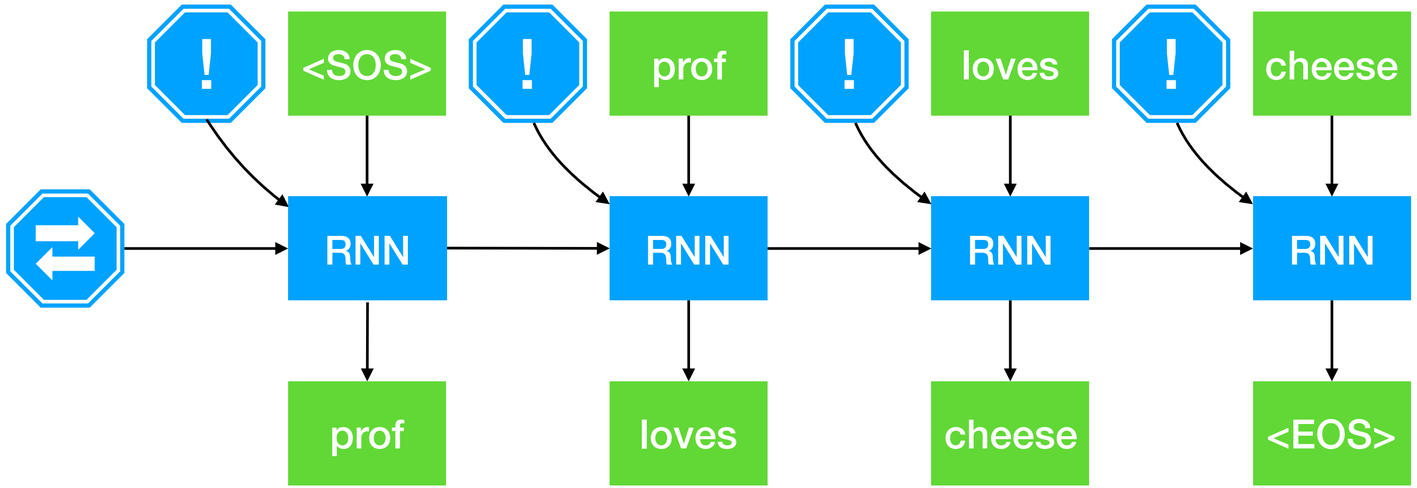
- rare word are exponentially less frequent than frequent words
- e.g. in HW4, 45%|65% (src|tgt) of the unique words occur once

1. https://en.wikipedia.org/wiki/Zipf%27s_law

Out-of-vocabulary words

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

Copy Mechanism

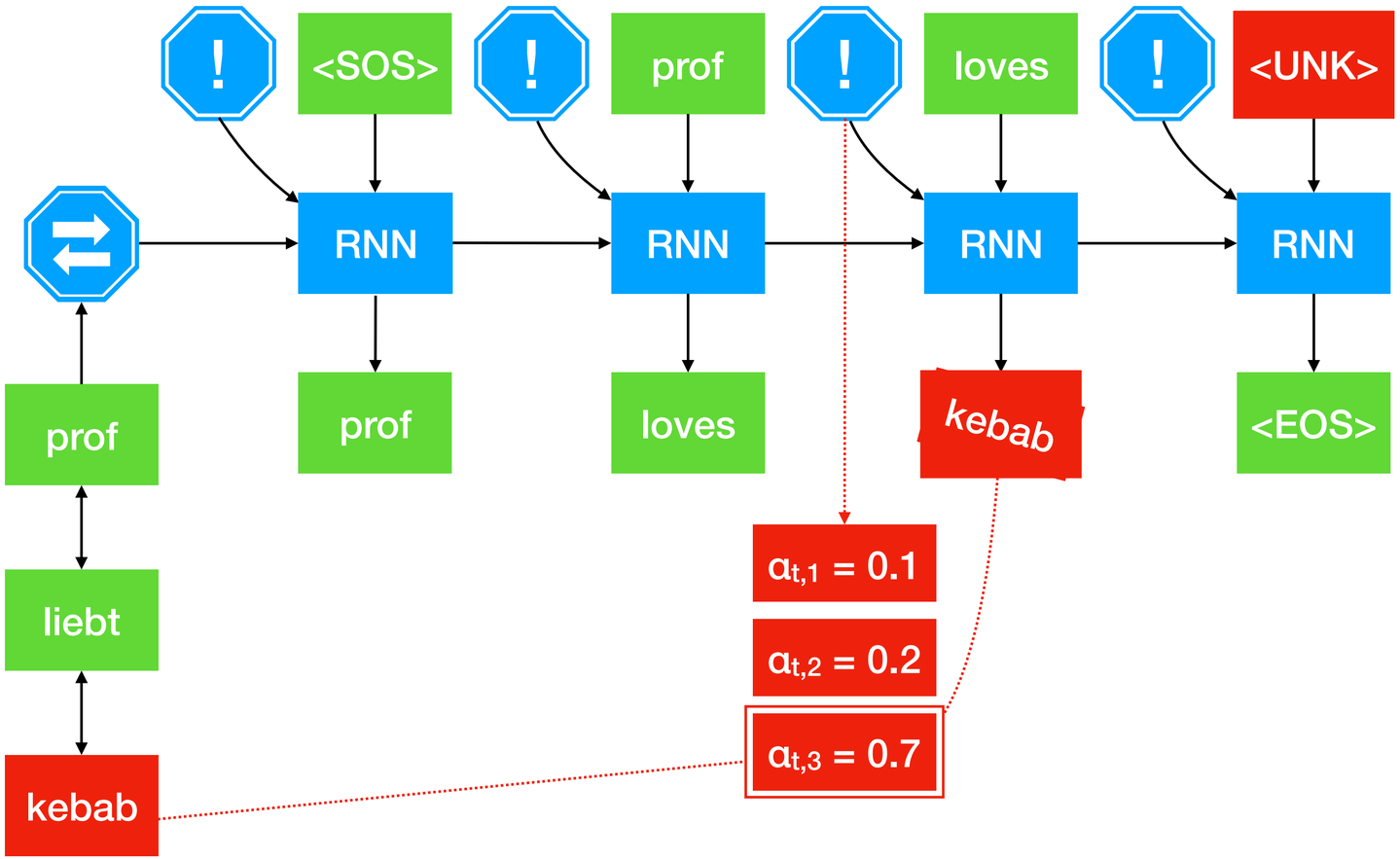


1. CL2015015 [Luong et al.] Addressing the Rare Word Problem in Neural Machine Translation

Concept

P1
Copy

Copy Mechanism

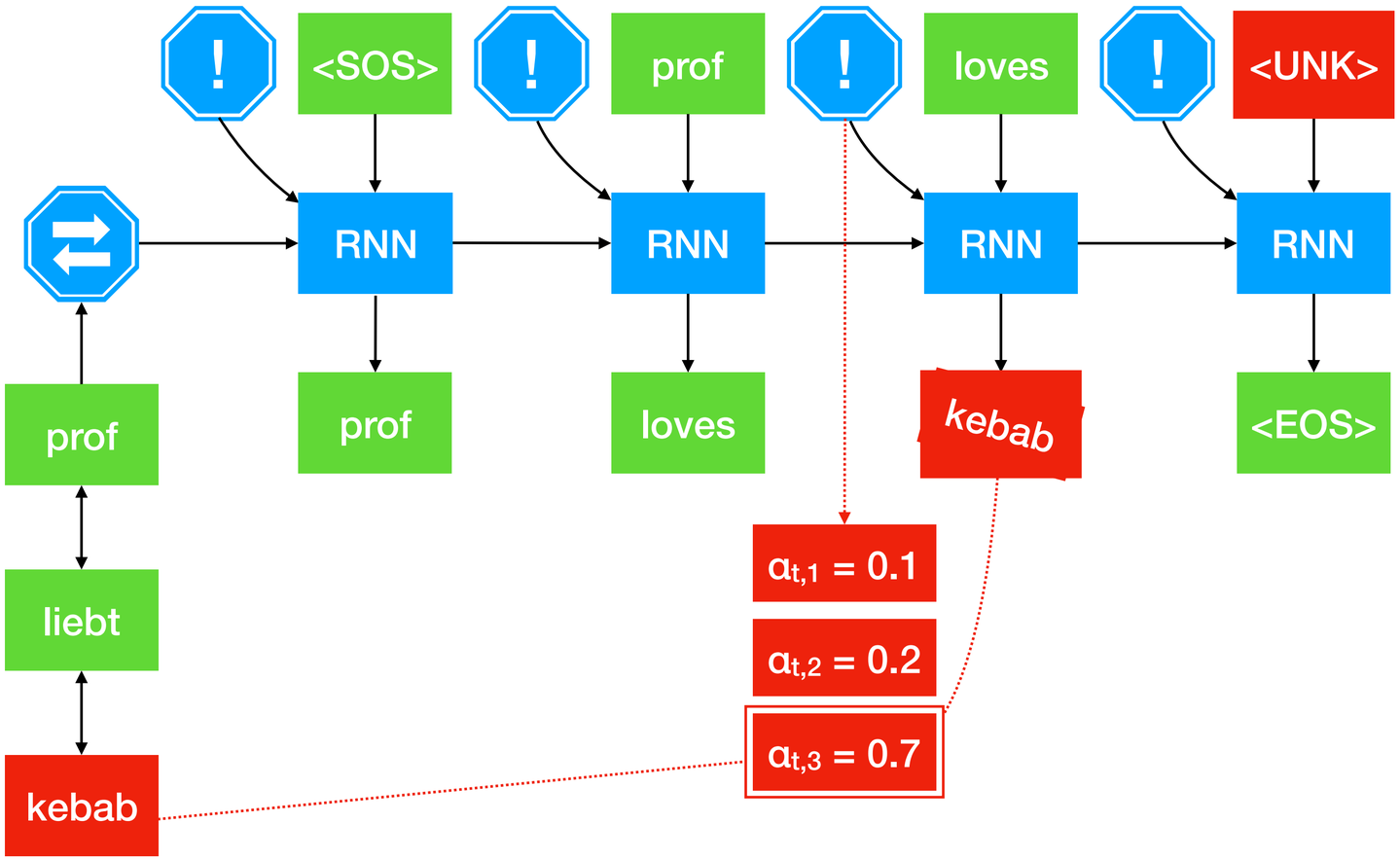


Concept

1. CL2015015 [Luong et al.] Addressing the Rare Word Problem in Neural Machine Translation

P1
Copy

Copy Mechanism

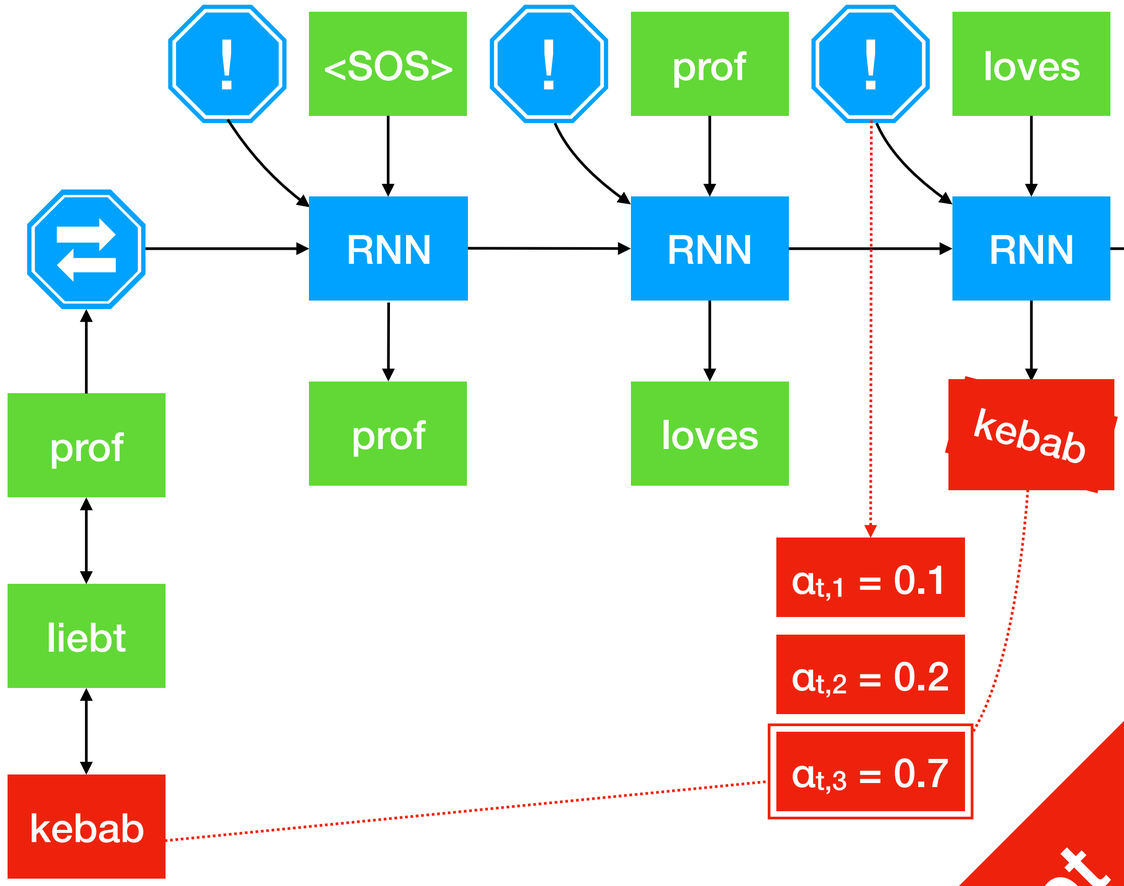


Concept

1. CL2015015 [Luong et al.] Addressing the Rare Word Problem in Neural Machine Translation

Copy Mechanism

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

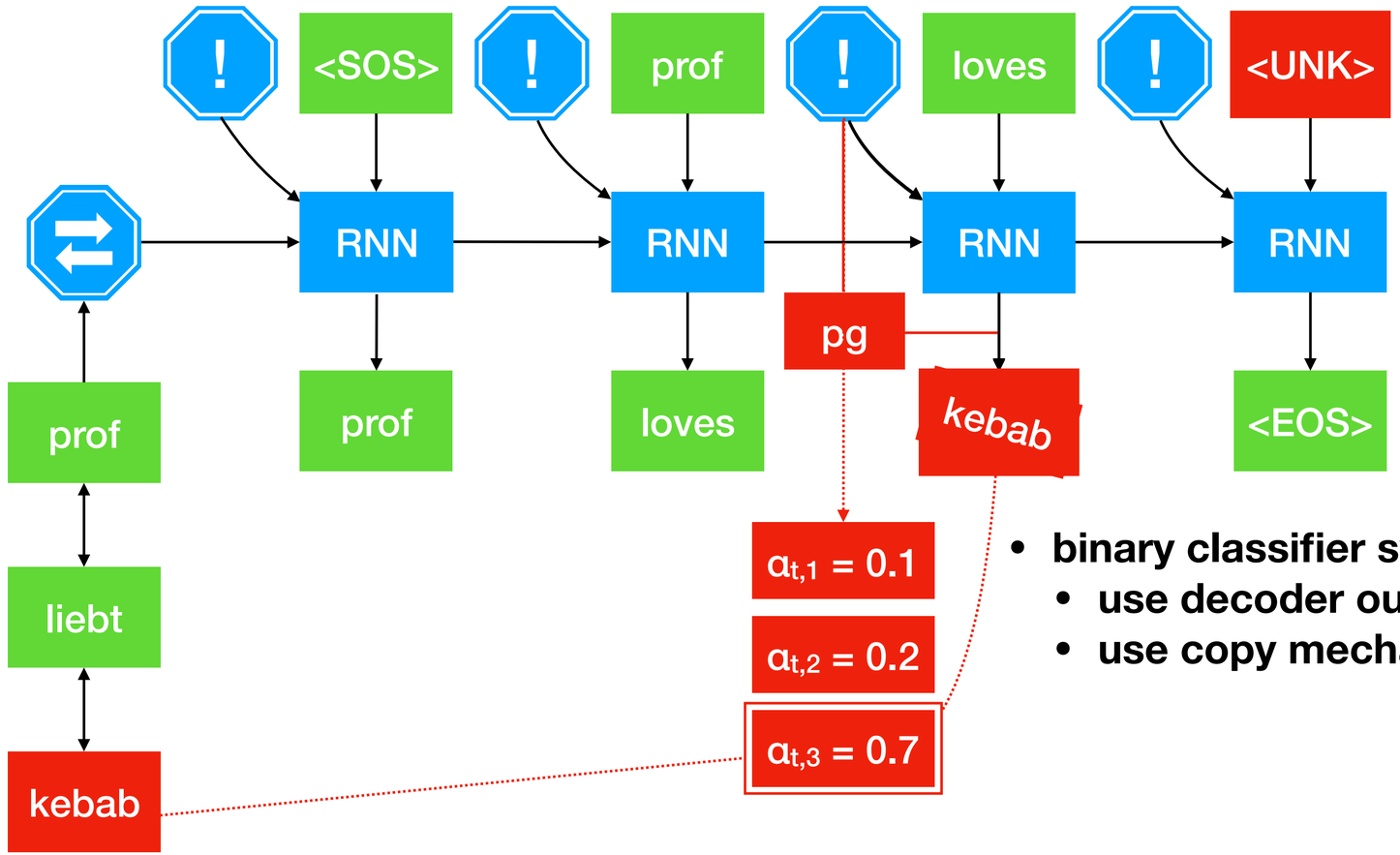


Concept

1. CL2015015 [Luong et al.] Addressing the Rare Word Problem in Neural Machine Translation

P1
Copy

*Pointer-Generator Copy Mechanism



- **binary classifier switch**
  - **use decoder output**
  - **use copy mechanism**

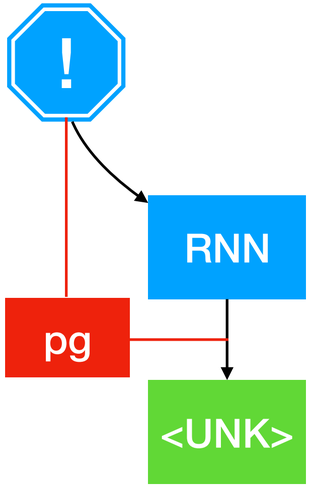
Concept

1. CL2016032 [Gulcehre et al.] Pointing the Unknown Words

P1
Copy

*Pointer-Generator Copy Mechanism

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



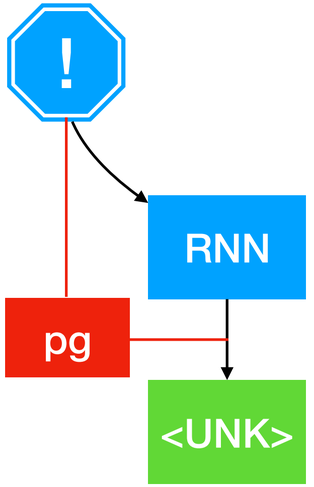
Concept

1. CL2016032 [Gulcehre et al.] Pointing the Unknown Words

P1
Copy

*Pointer-Based Dictionary Fusion

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



Concept

1. CL2019331 [Gū et al.] Pointer-based Fusion of Bilingual Lexicons into Neural Machine Translation

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:

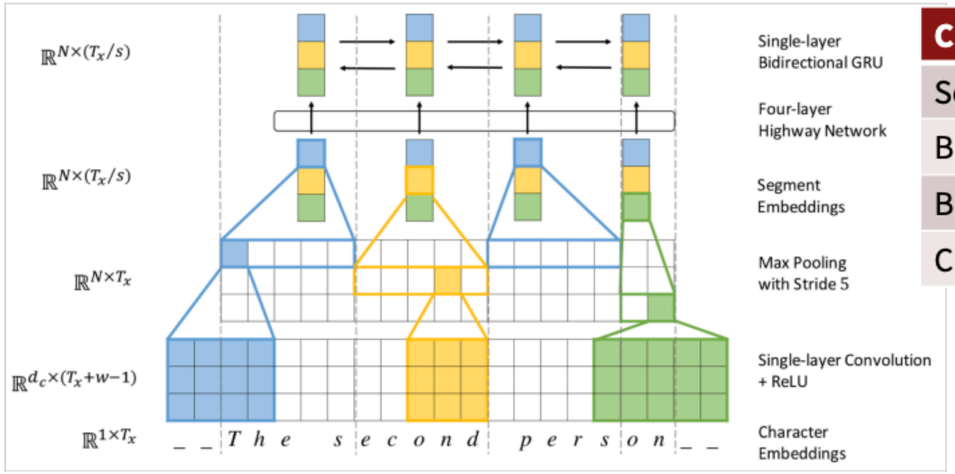
Subword / character models

- 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



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

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 ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w e s t
3 w i d e s t

Vocabulary

l, 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 ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, 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 ↦ a new ngram

Dictionary

5 l o w
2 l o w e r
6 n e w **est**
3 w i d **est**

Vocabulary

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

Add a pair (es, t) with freq 9

(Example from Sennrich)

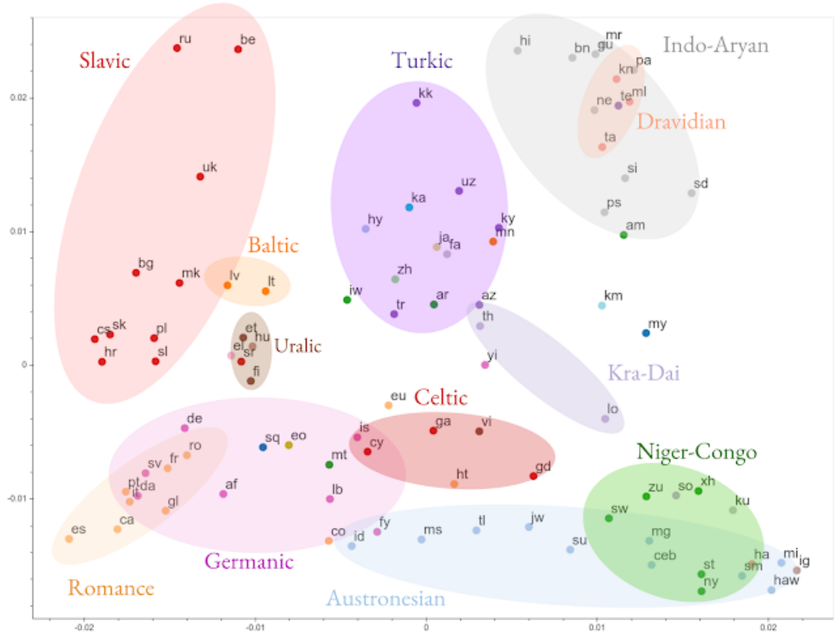
Subword / character models

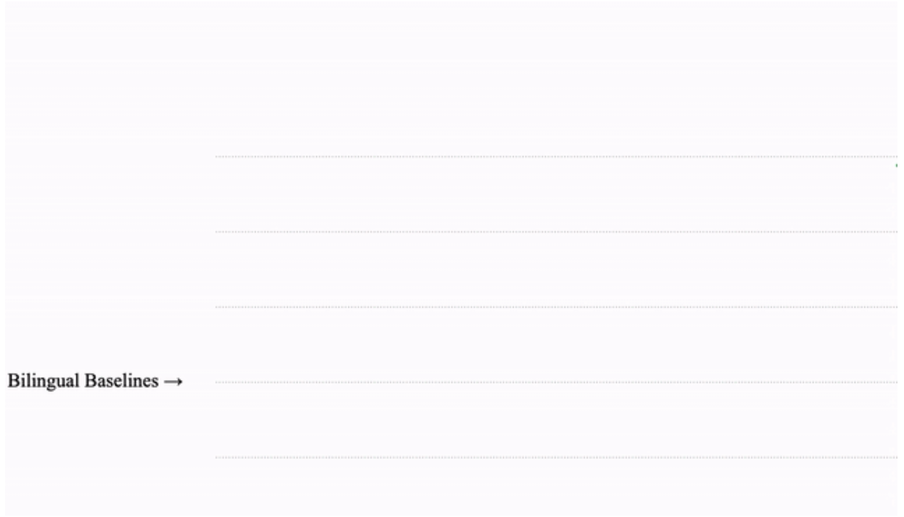
- Character based seq2seq models
- Byte pair encoding (BPE)
- Subword units help model morphology

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