COS 484

L13: Neural Machine Translation

Spring 2022

Final Projects

- Reminder: Proposals due March 29 (next week)
- Course website has guidelines, along with sample proposals and projects from last year

Sample proposals:
https://www.dropbox.com/sh/y0zjhphvyh2e683/AACMgba_AqzwqDgUUZ075vhna?dl=0

Sample final reports:
https://www.dropbox.com/sh/vkde9q2ca1yt1ri/AADxOCyhHRrdQKV5librkLYua?dl=0

Last time: IBM Model 1

- Assume $p(a_m \mid m, M^{(s)}, M^{(t)}) = \frac{1}{M^{(t)}}$

- We then have:

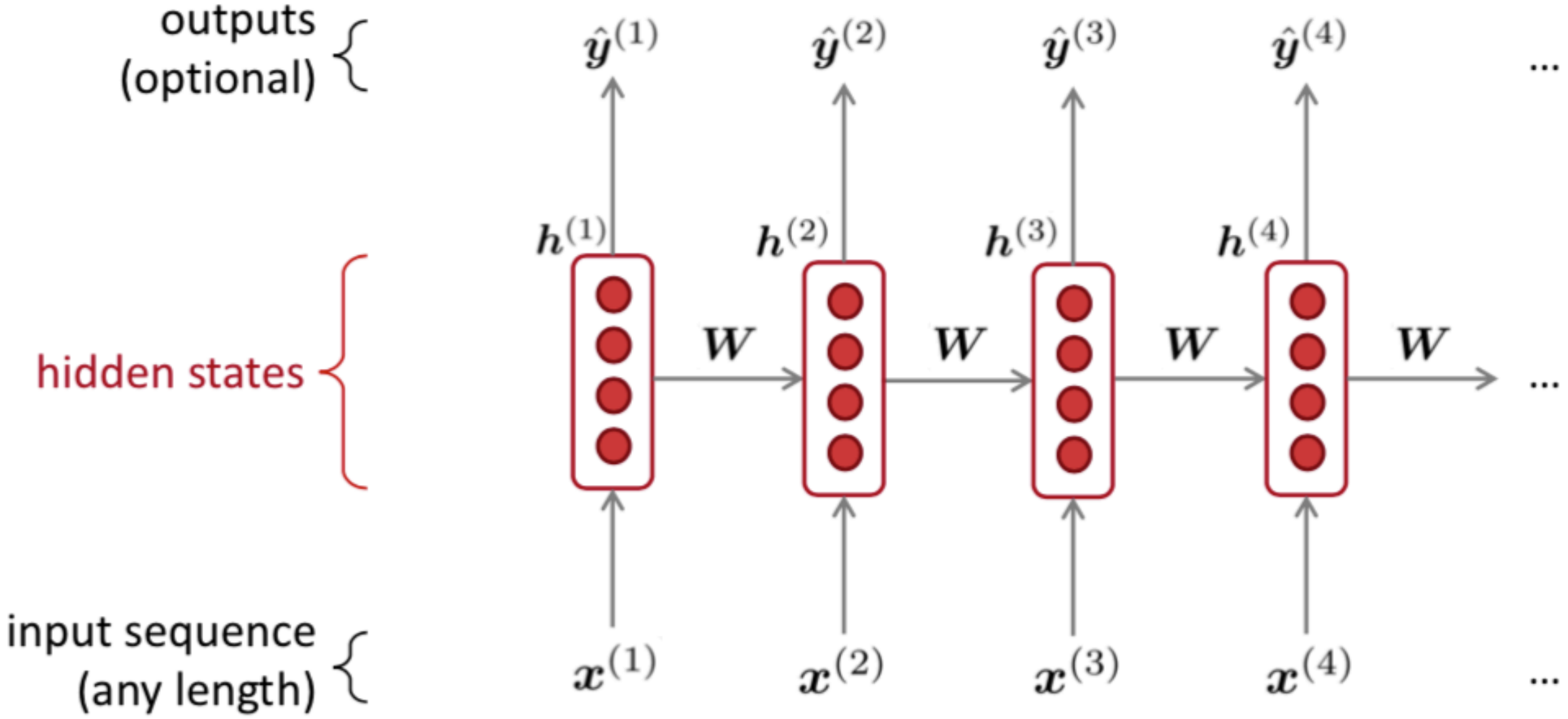
$$p(w^{(s)}, w^{(t)}) = p(w^{(t)}) \sum_{A} (\frac{1}{M^{(t)}})^{M^{(s)}} \; p(w^{(s)} \mid w^{(t)})$$

Neural Machine Translation

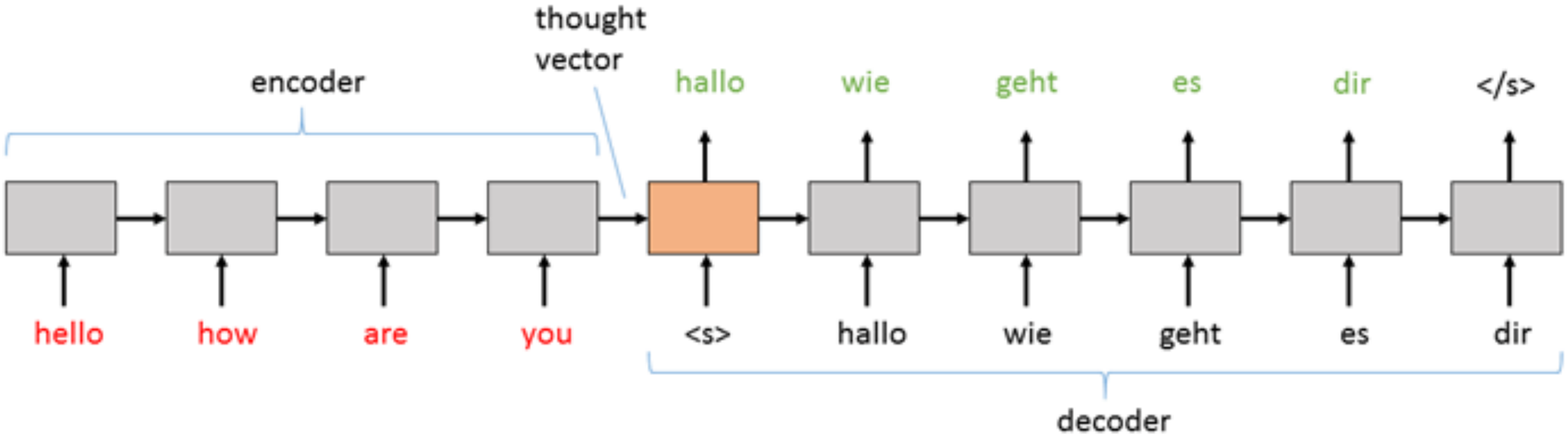
- A **single neural network** is used to translate from source to target language
- Architecture: Encoder-Decoder
  - Two main components:
    - Encoder: Convert source sentence (input) into a vector/matrix
    - Decoder: Convert encoding into a sentence in target language (output)

Recall: RNNs

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \in \mathbb{R}^d$$



Sequence to Sequence learning (Seq2seq)

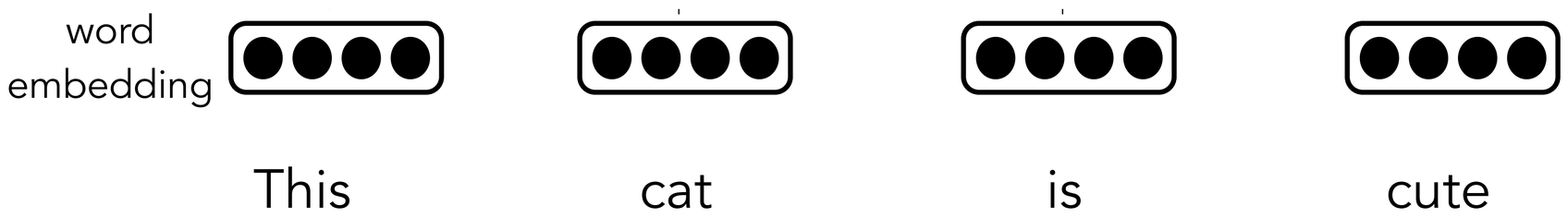


- Encode entire input sequence into a single vector **(using an RNN)**
- Decode one word at a time **(again, using an RNN!)**
- Beam search for better inference
- Learning is not trivial! (vanishing/exploding gradients)

(Sutskever et al., 2014)

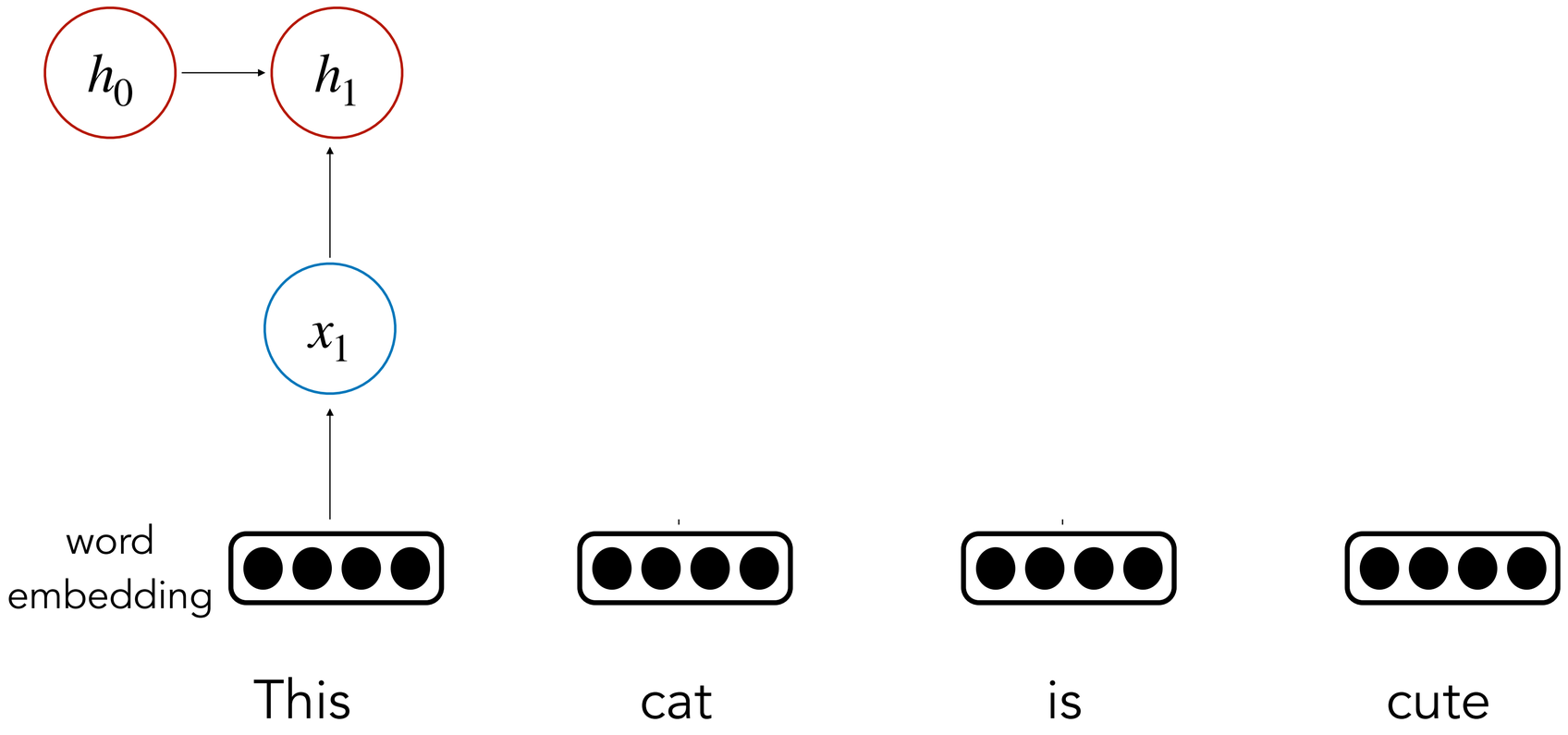
Encoder

Sentence: This cat is cute



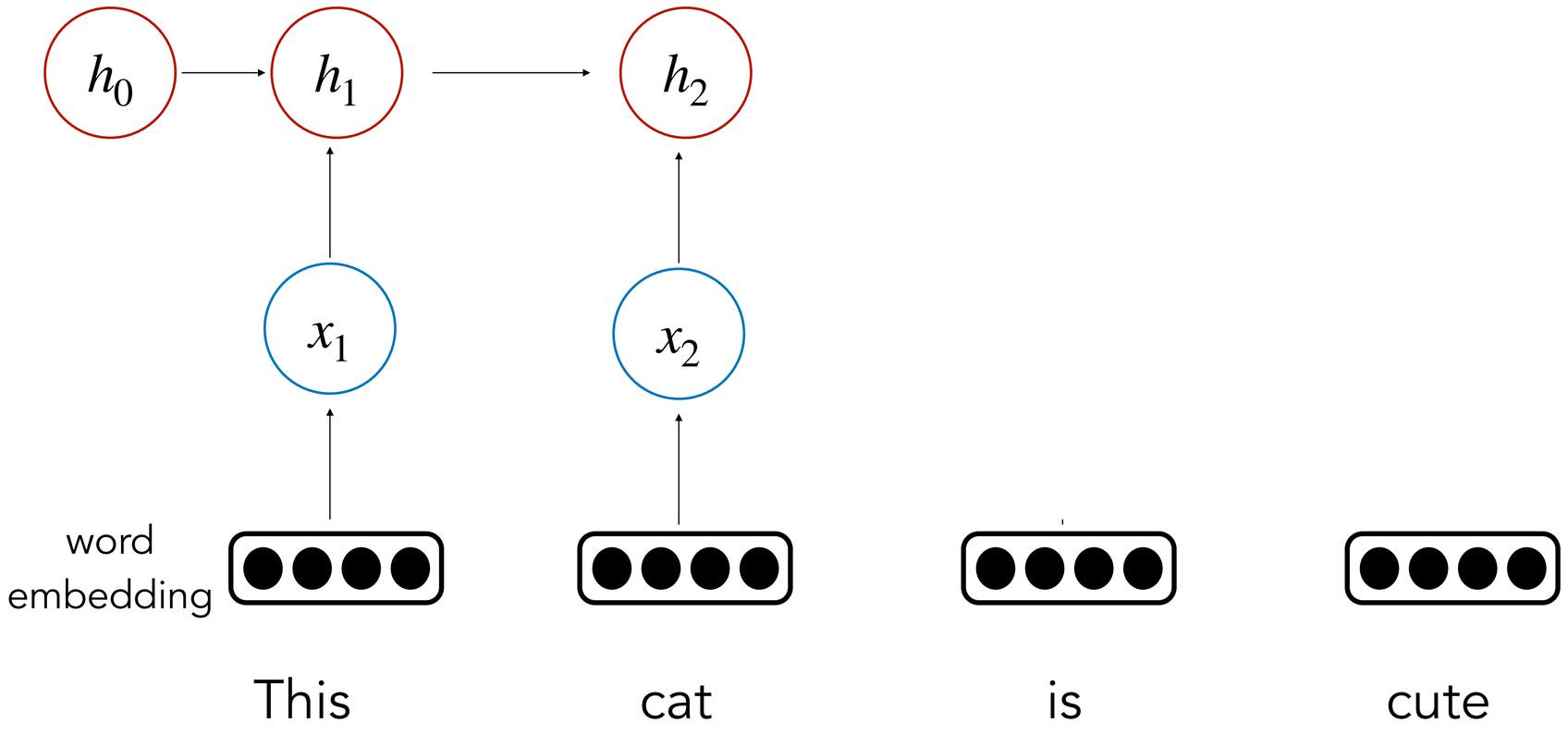
Encoder

Sentence: This cat is cute



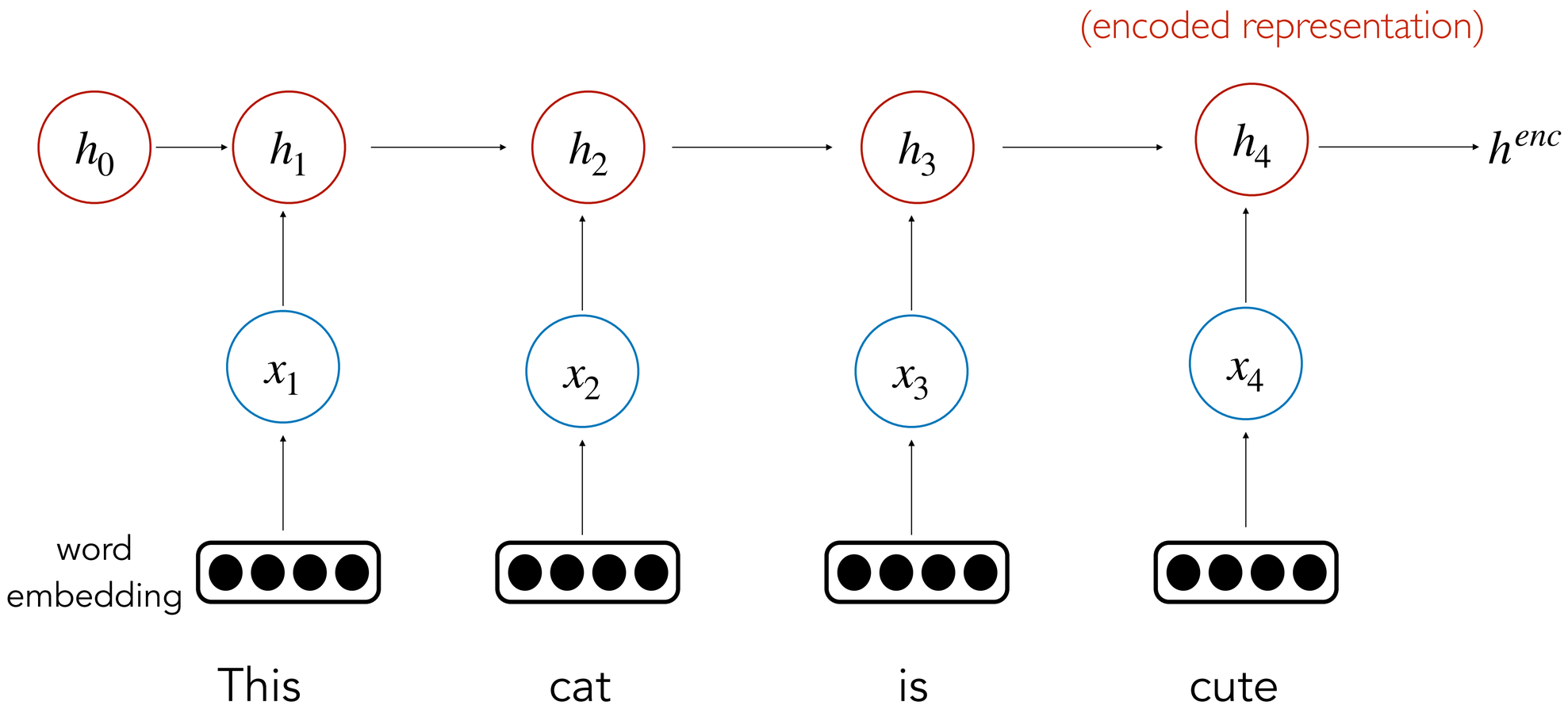
Encoder

Sentence: This cat is cute

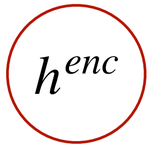


Encoder

Sentence: This cat is cute



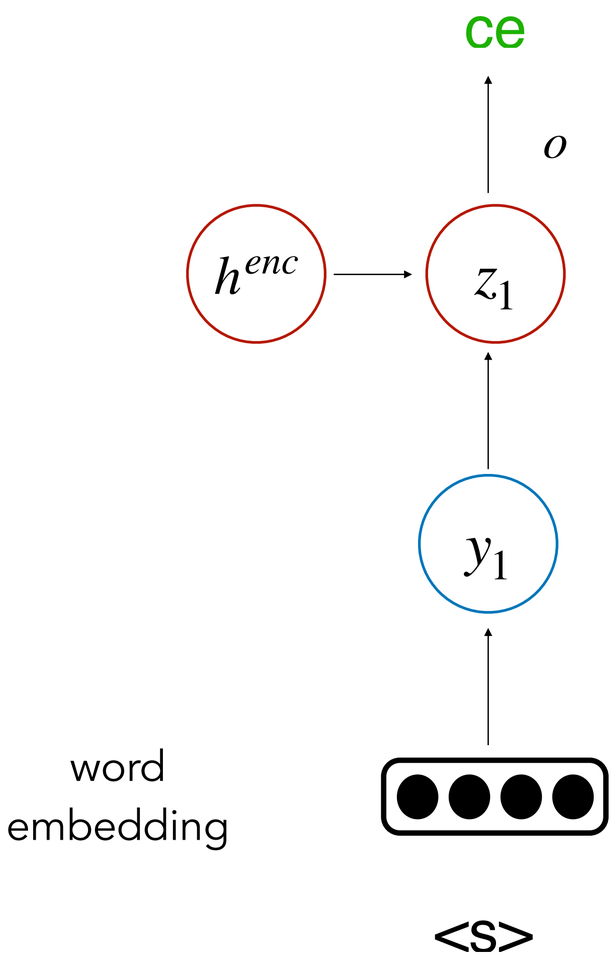
Decoder



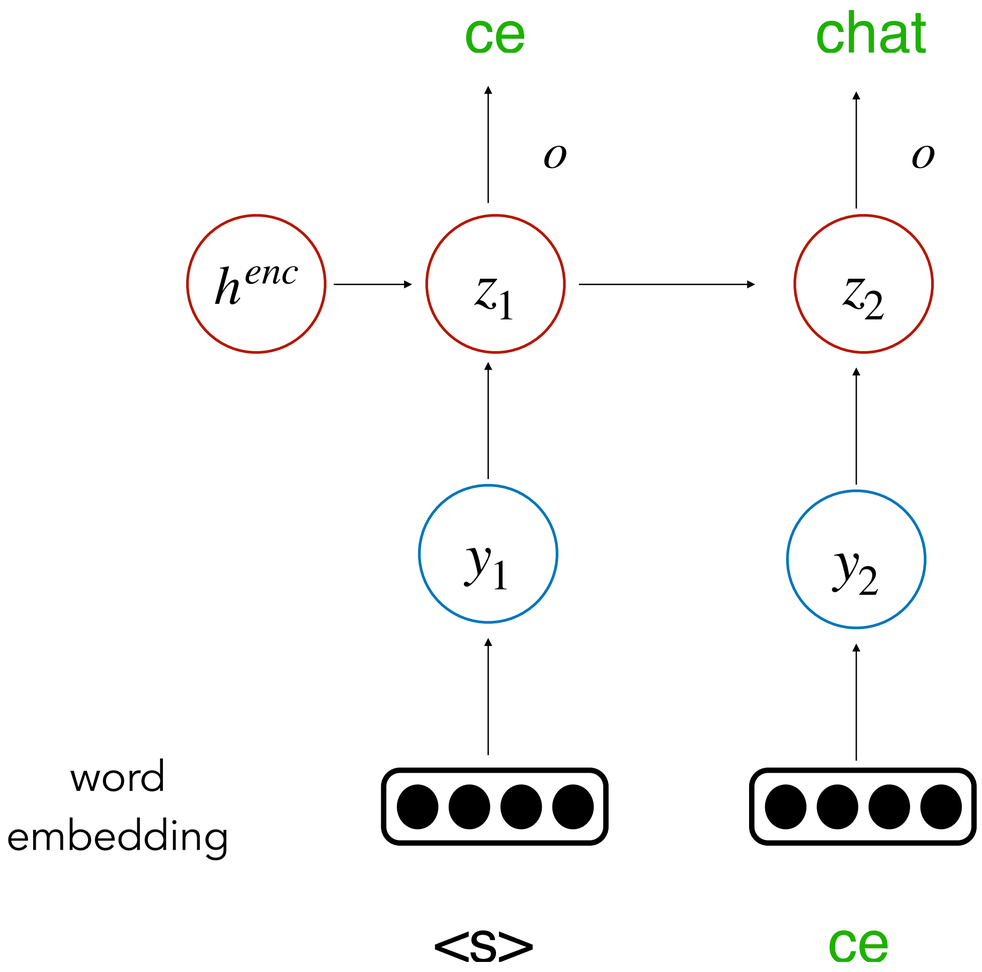
word
embedding

<s>

Decoder

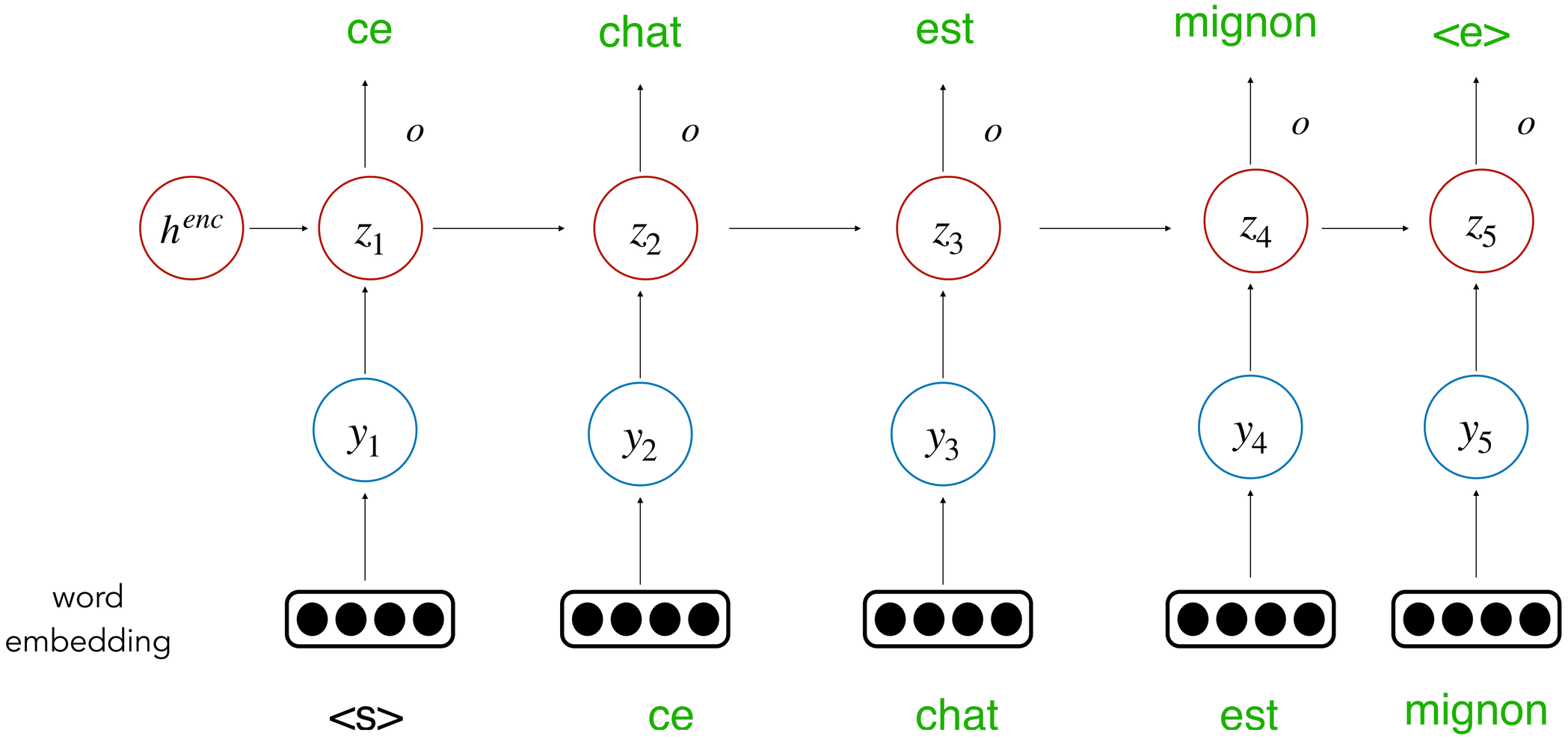


Decoder



Decoder

- A conditioned language model



Seq2seq training

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

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

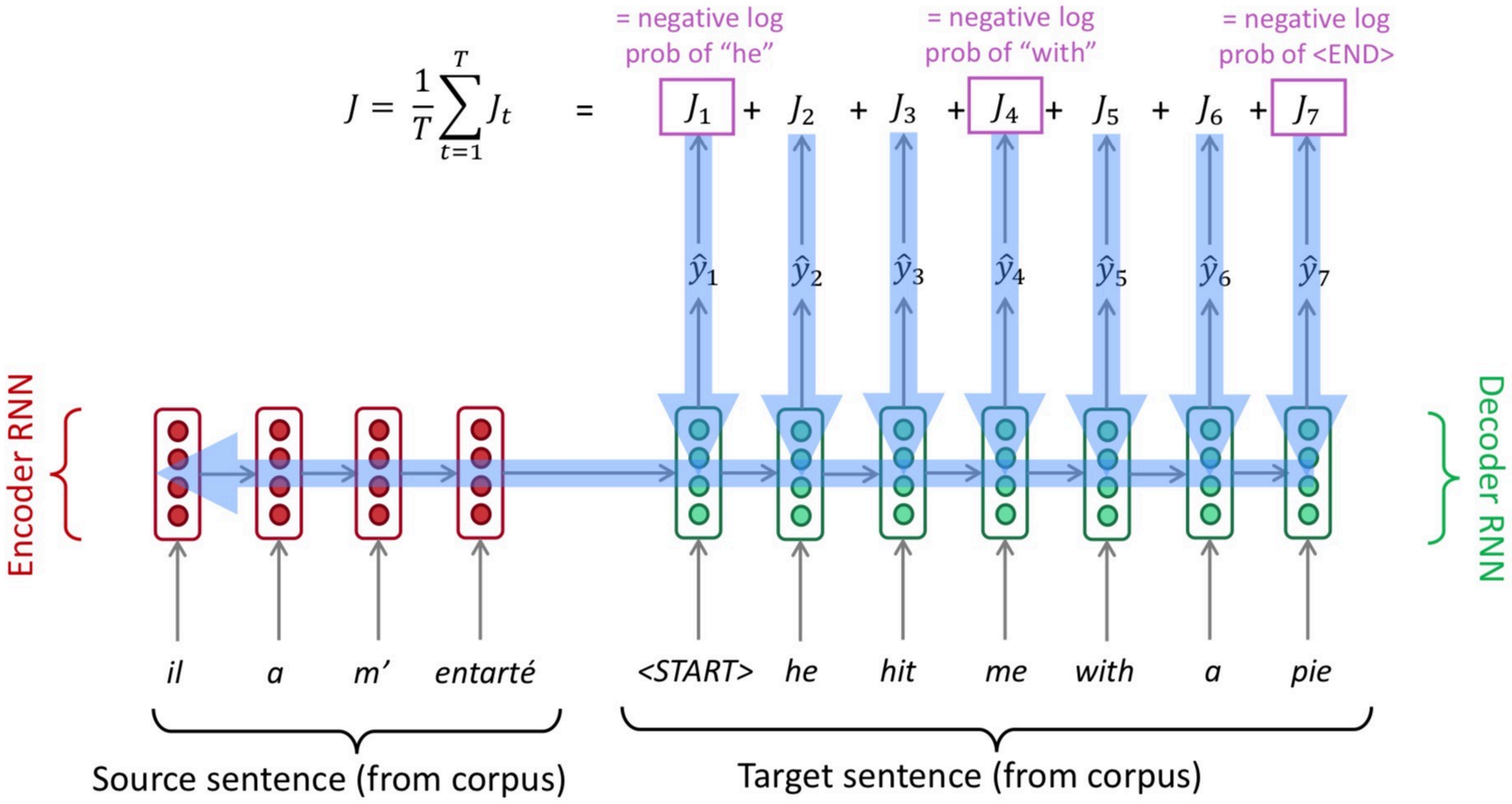
- Back-propagate gradients through *both decoder and encoder*
- Need a really big corpus

36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

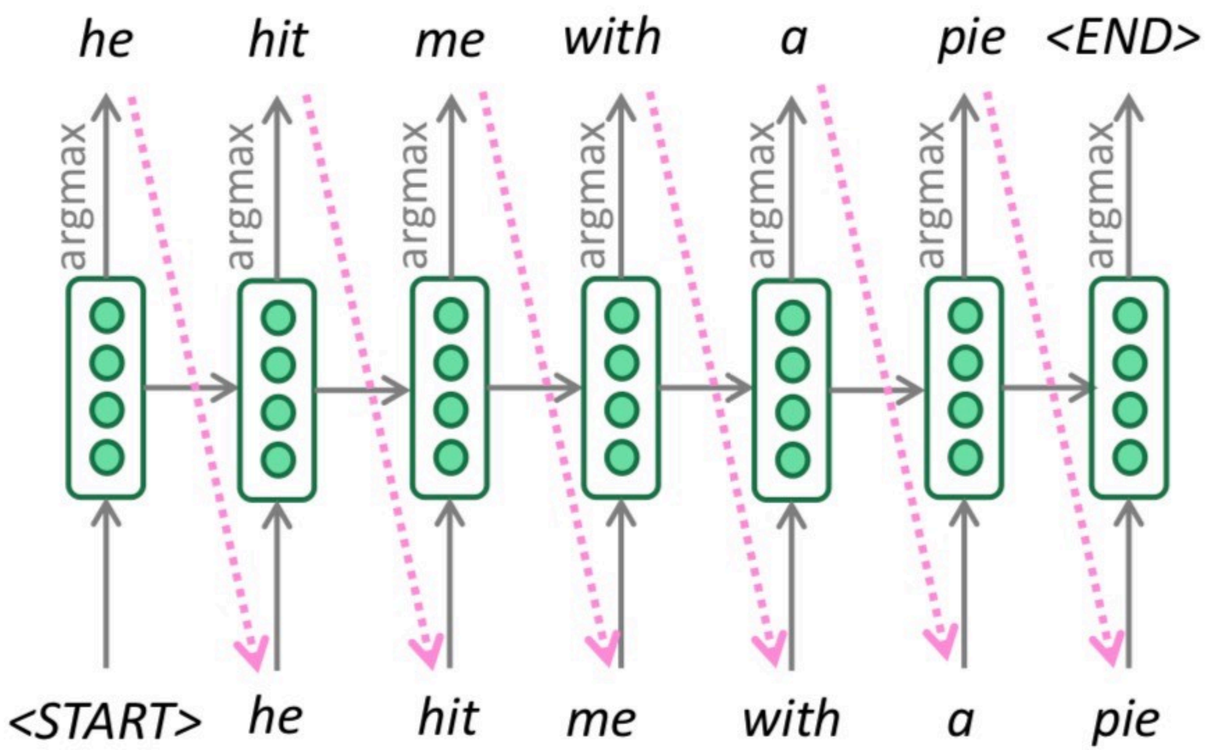
Seq2seq training



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

(slide credit: Abigail See)

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

V - Vocabulary

T - length of sequence

What is the complexity of doing this search?

A middle ground: Beam search

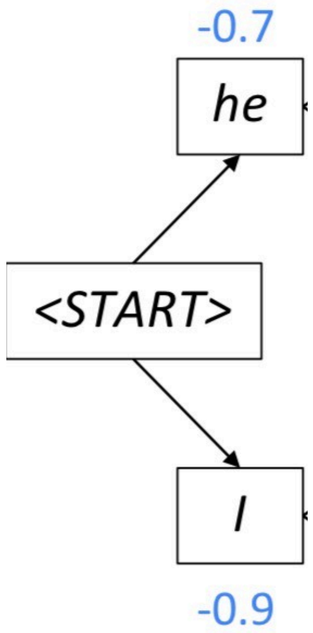
- **Key idea:** At every step, keep track of the k most probable partial translations (hypotheses)
- Score of each hypothesis = log probability of sequence so far

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

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

Beam decoding

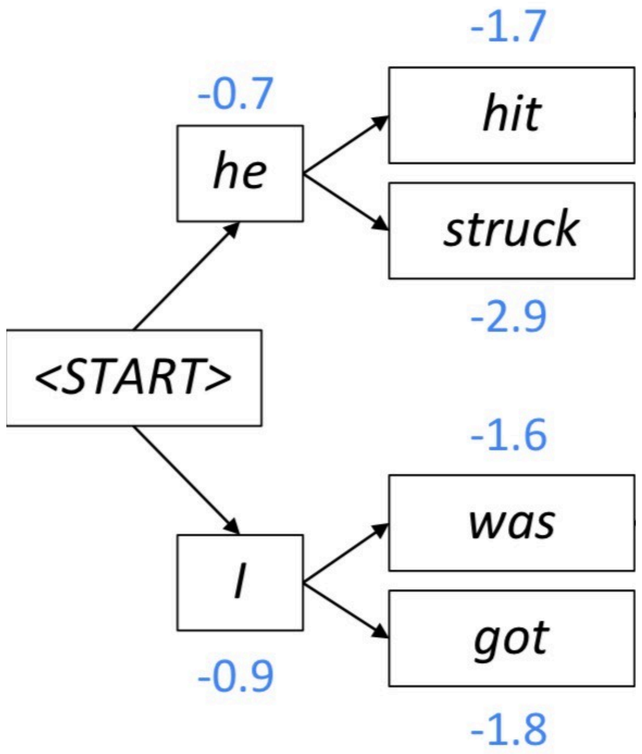
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

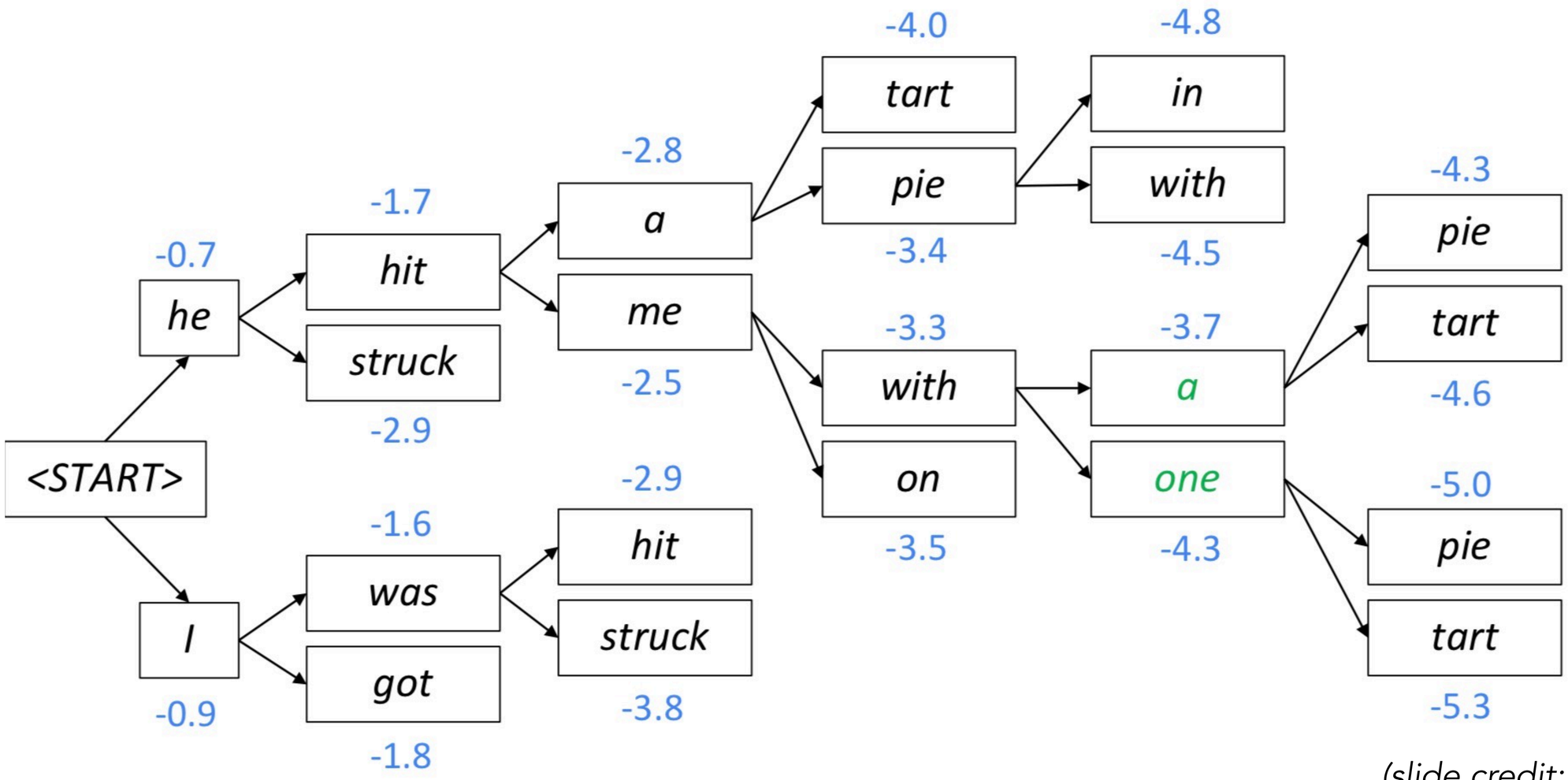
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Beam decoding

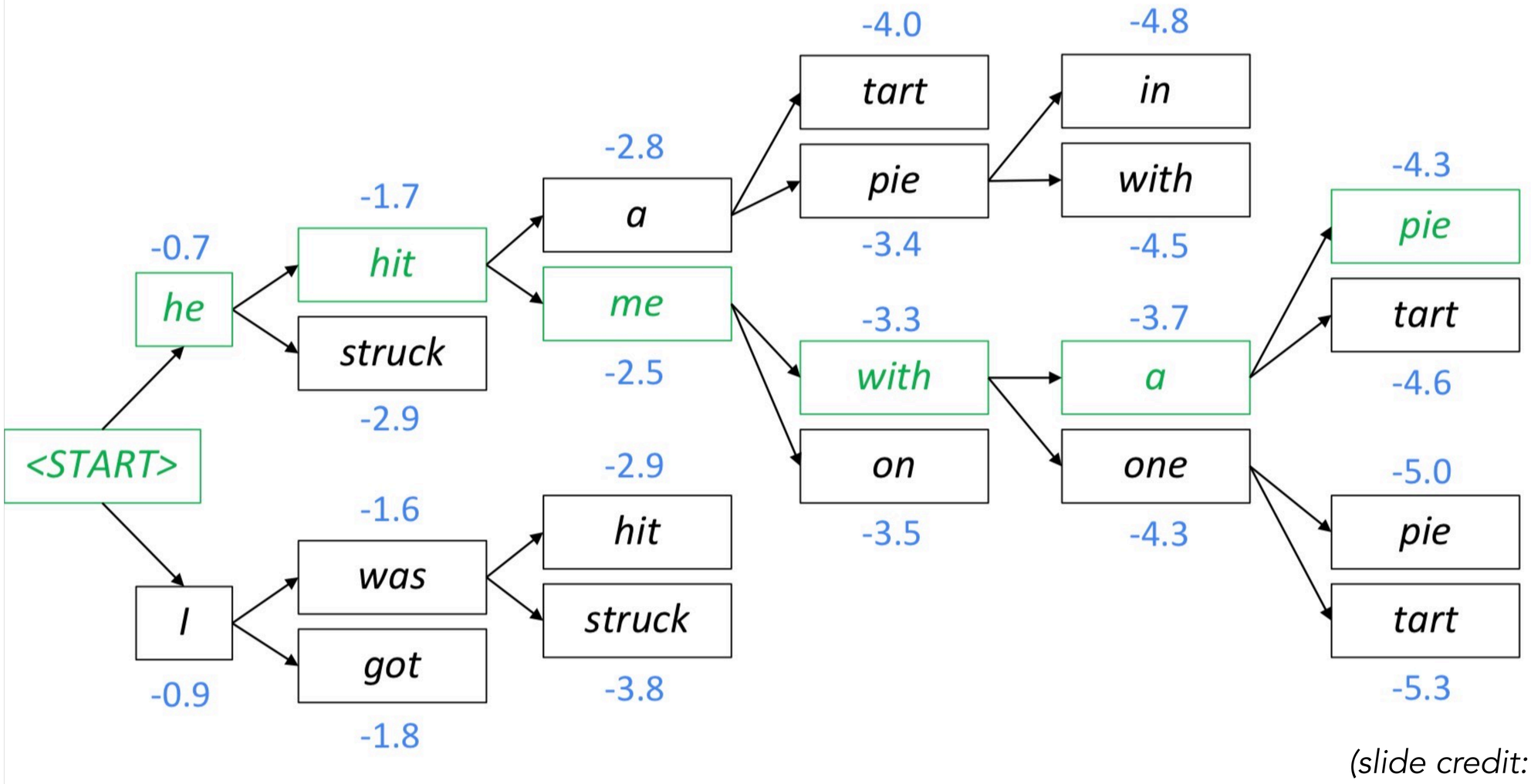
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



(slide credit: Abigail See)

Backtrack

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 $\langle e \rangle$ (end) token at different time steps
  - When a hypothesis produces $\langle e \rangle$, stop expanding it and place it aside
- Continue beam search until:
  - All k hypotheses produce $\langle e \rangle$ 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

NMT vs SMT

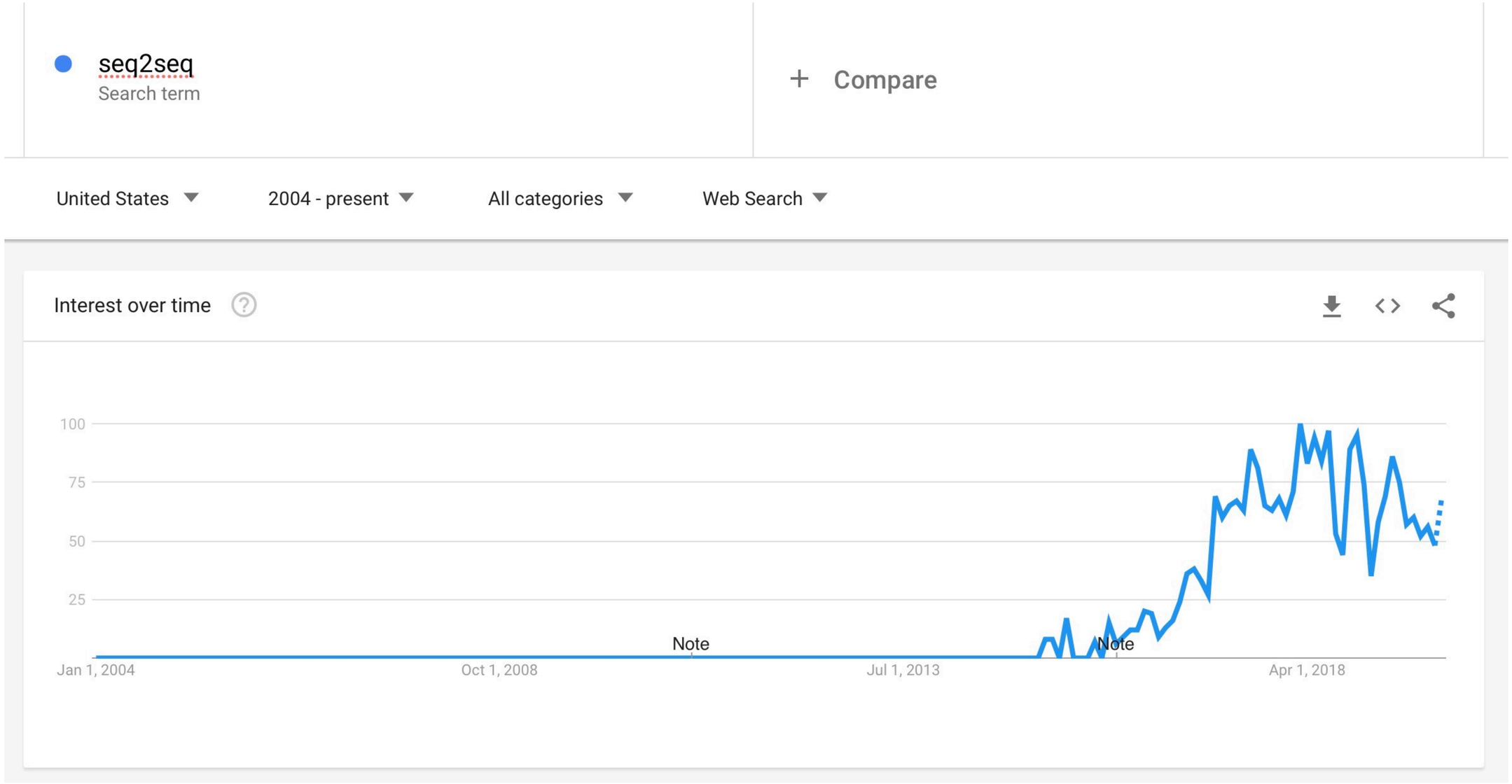
Pros

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

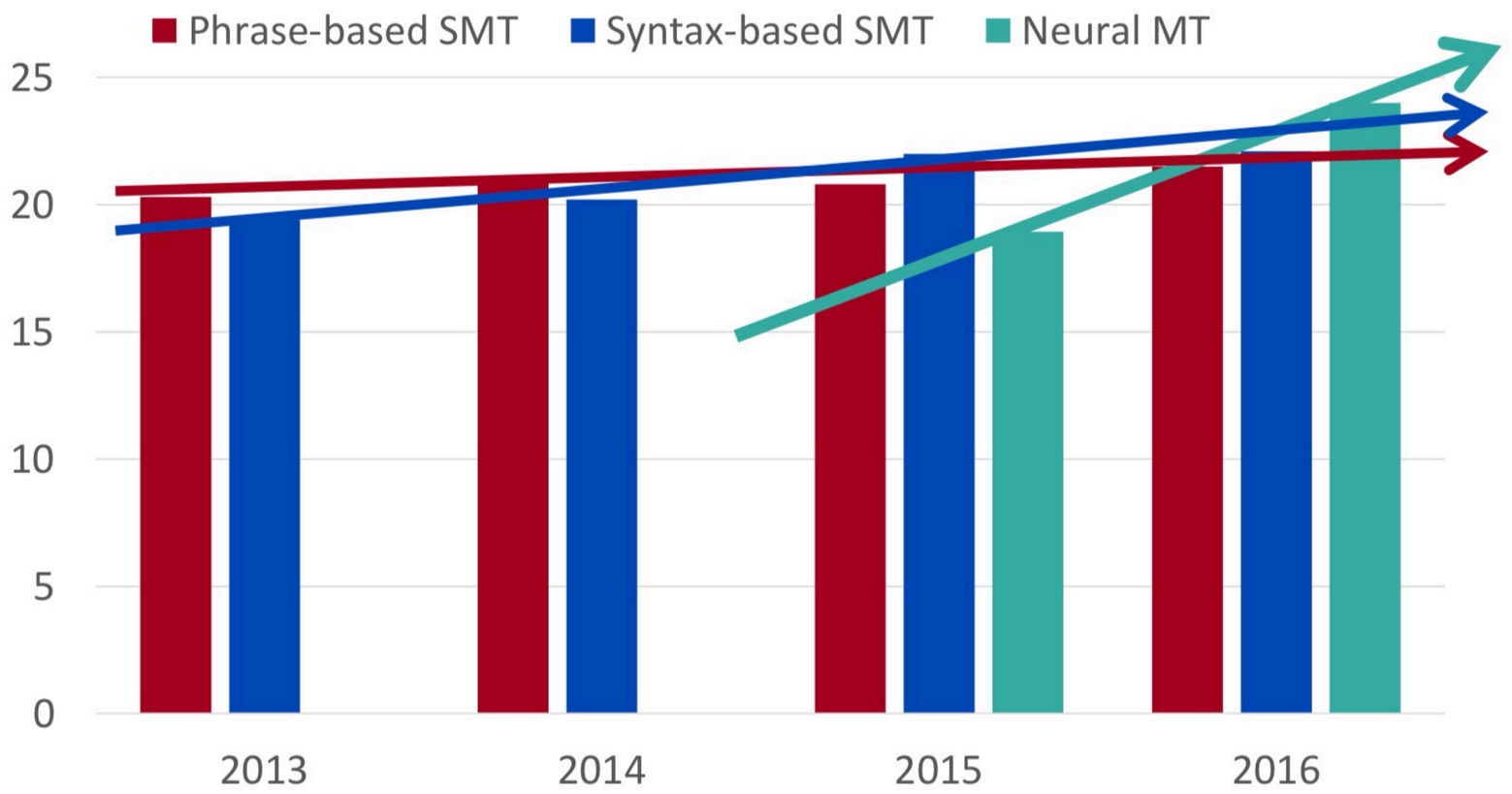
Cons

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

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

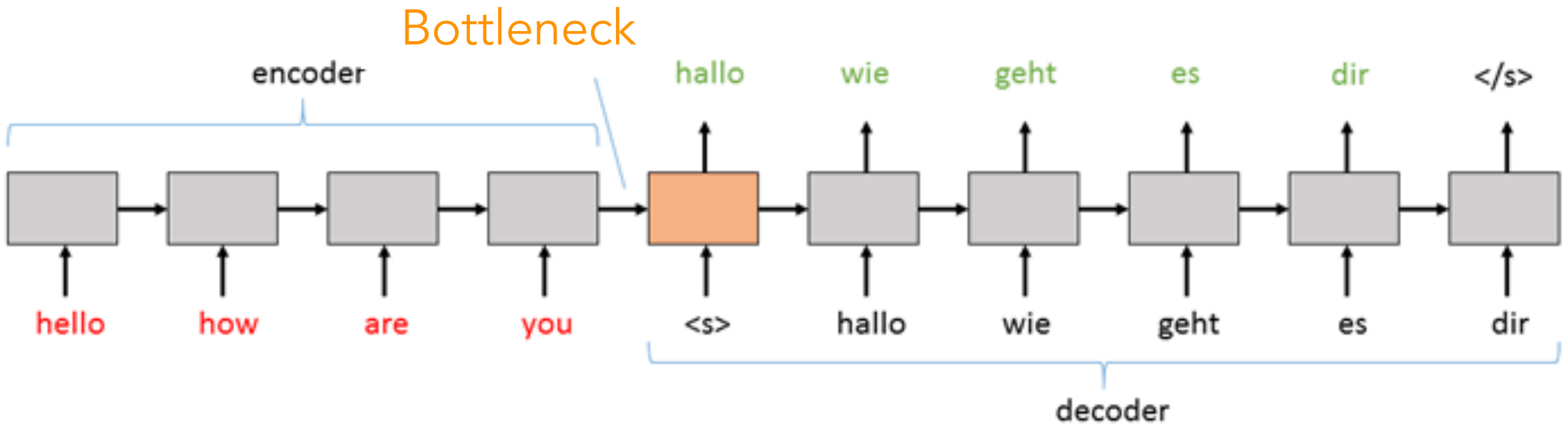
	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

(Wu et al., 2016)

Versatile seq2seq

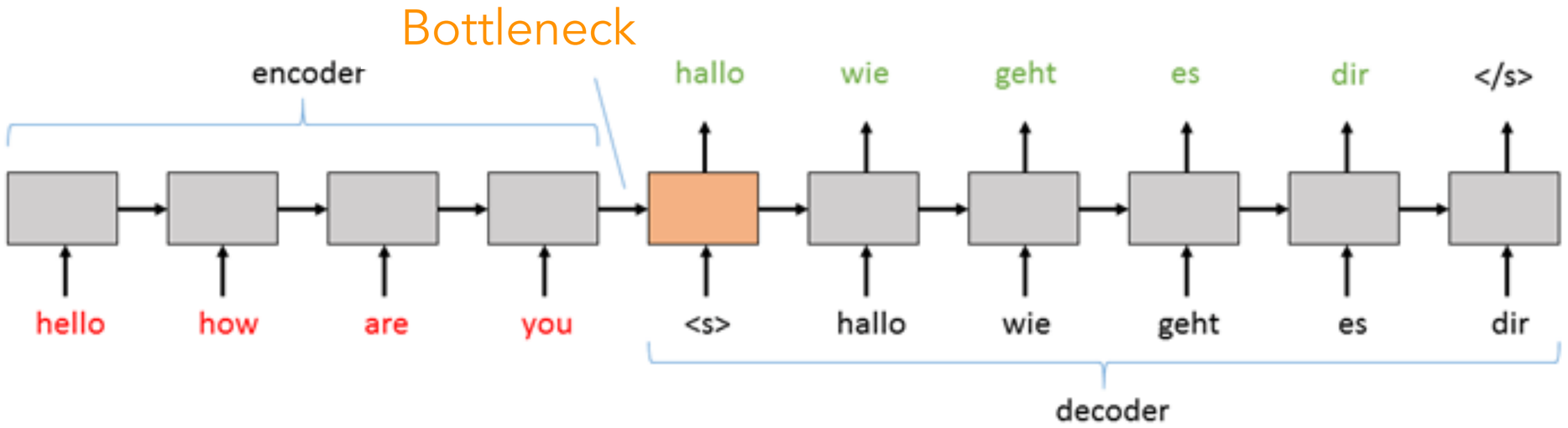
- Seq2seq finds applications in many other tasks!
- Any task where inputs and outputs are sequences of words/ characters
  - Summarization (input text → summary)
  - Dialogue (previous utterance → reply)
  - Parsing (sentence → parse tree in sequence form)
  - Question answering (context+question → answer)

Issues with vanilla seq2seq



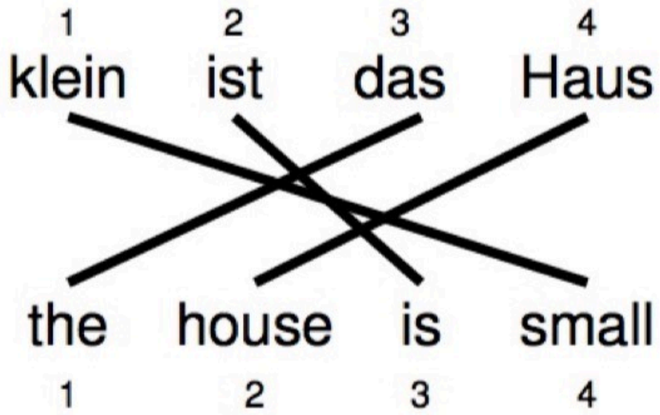
- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- Model may “overfit” to training sequences

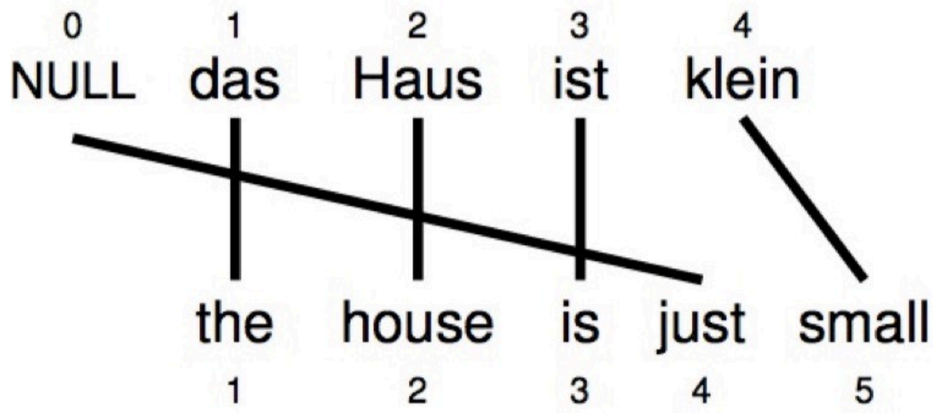
Issues with vanilla seq2seq



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

Remember alignments?





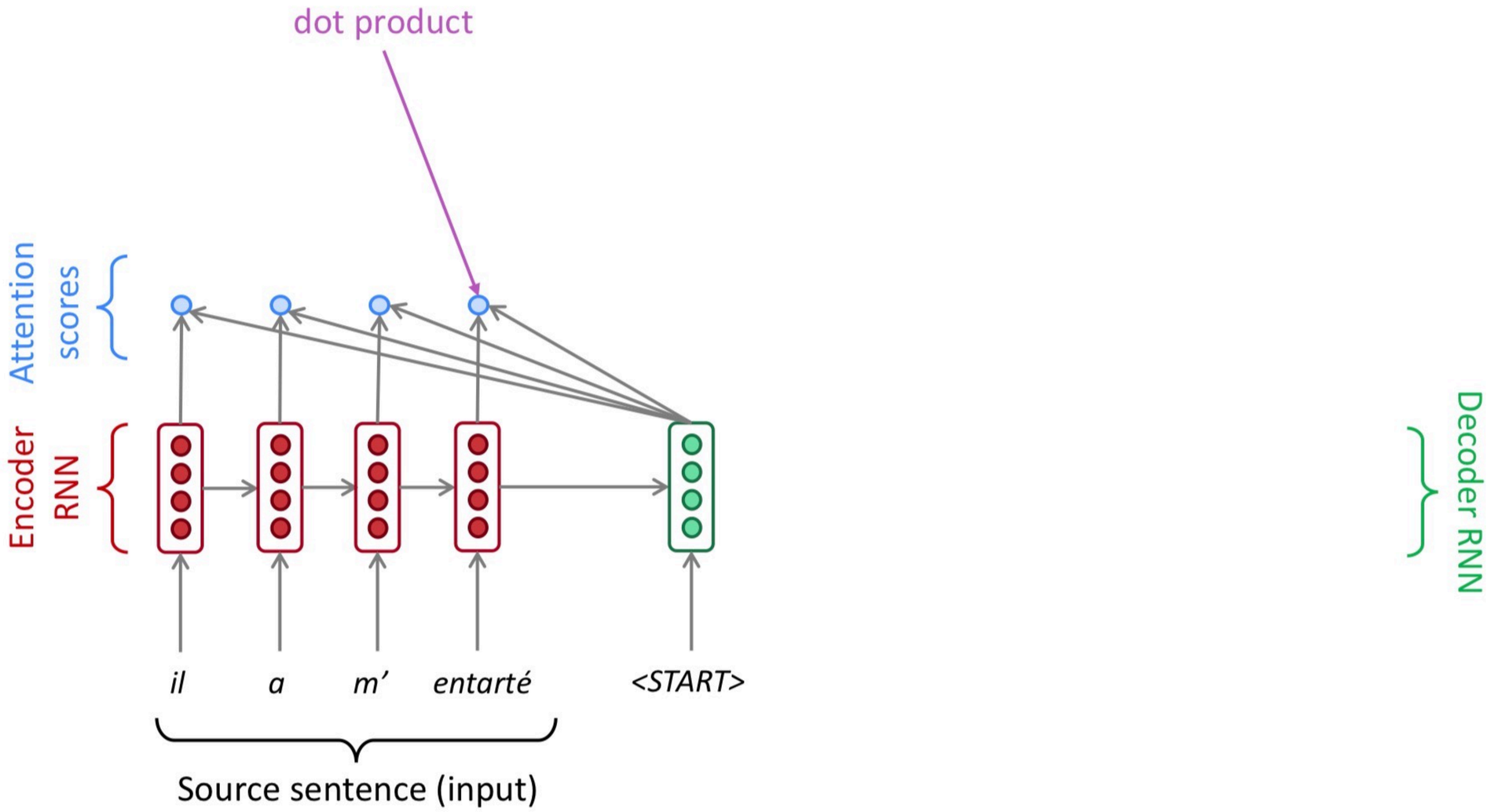
$$\mathbf{a} = (3, 4, 2, 1)^\top$$

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

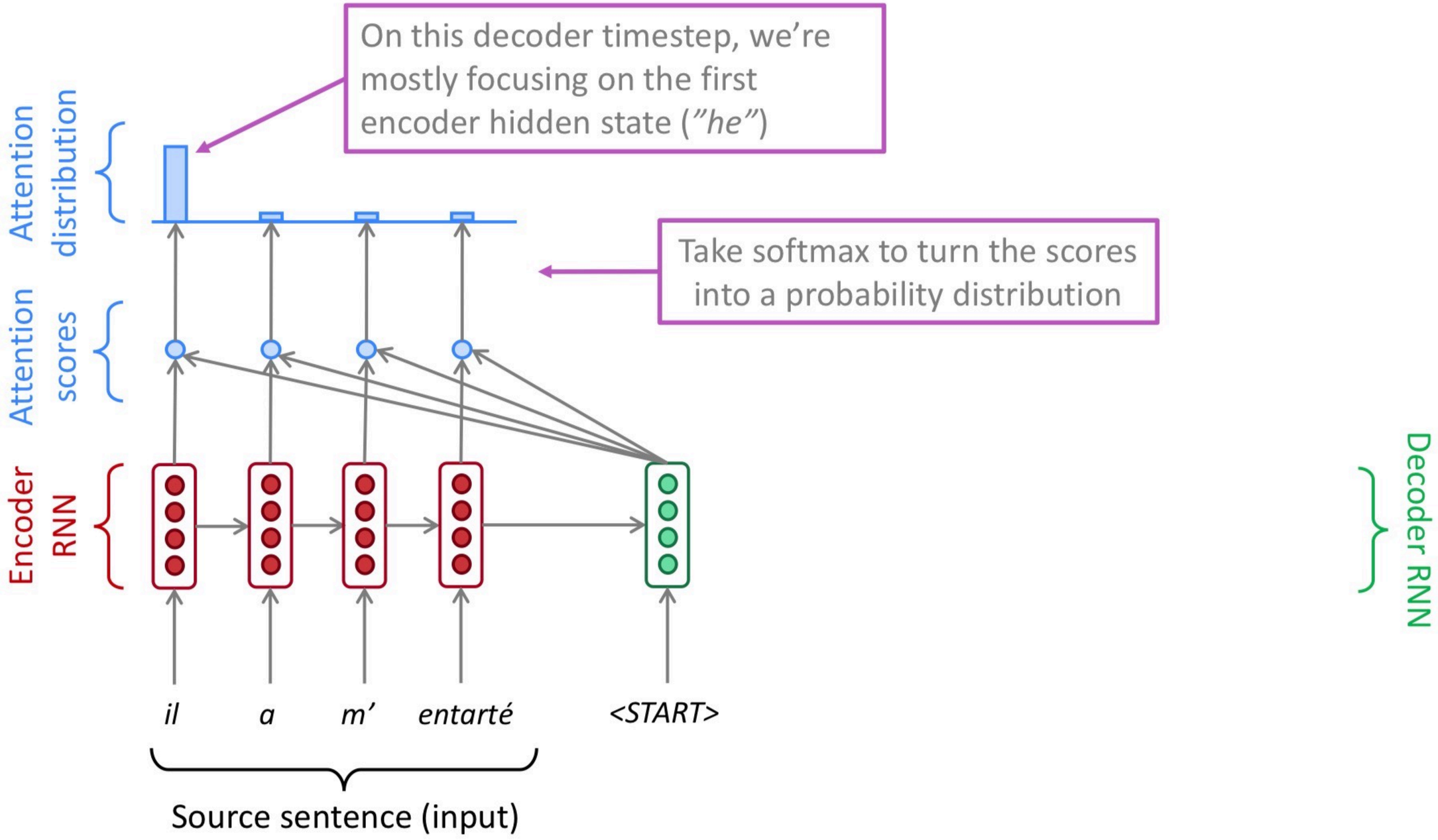
Attention

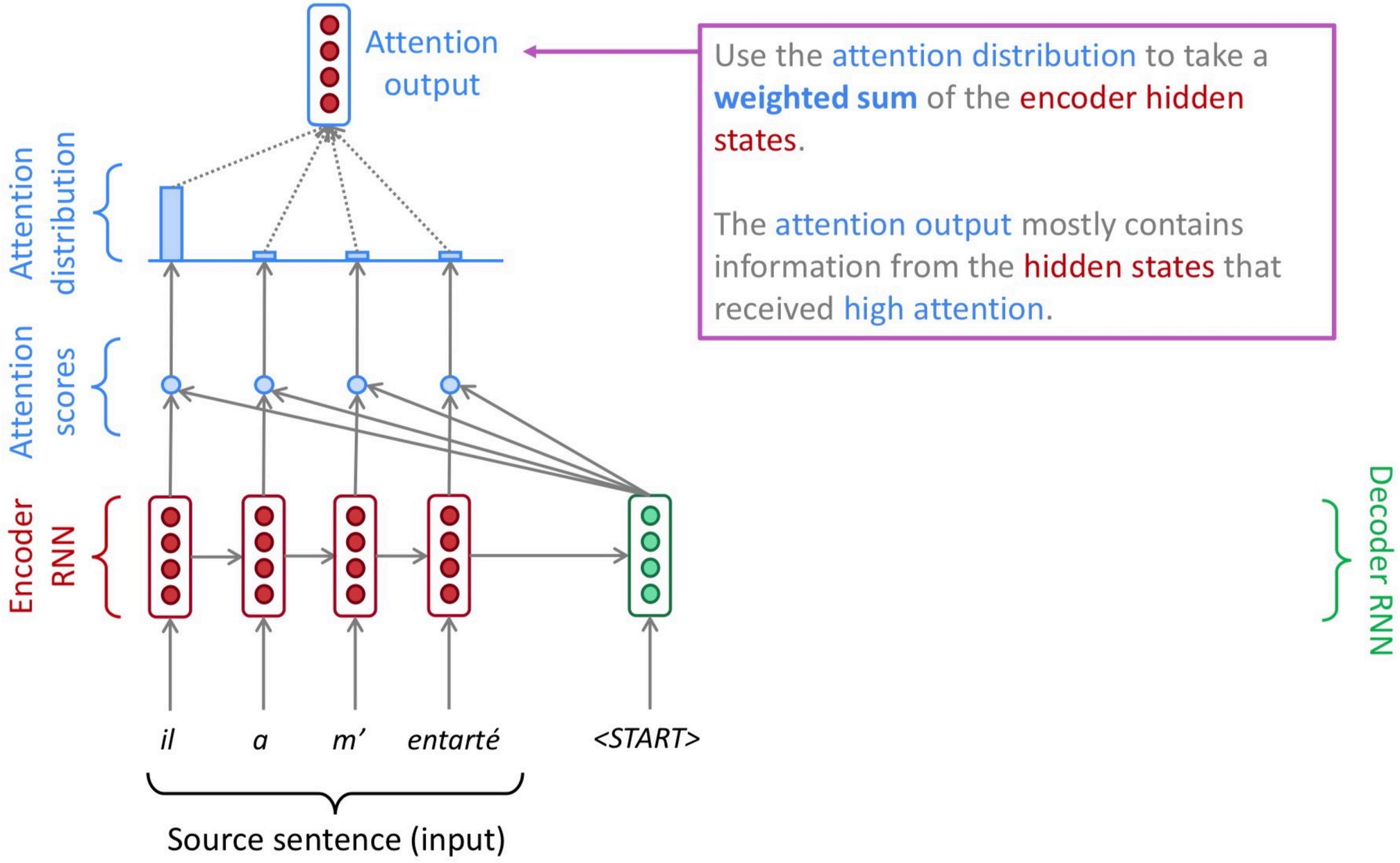
- The neural MT equivalent of alignment models
- Key idea: At each time step during decoding, **focus on a particular part** of source sentence
  - This depends on the **decoder's** current hidden state h^{dec} (i.e. an idea of what you are trying to decode)
  - Usually implemented as a probability distribution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

Seq2seq with attention



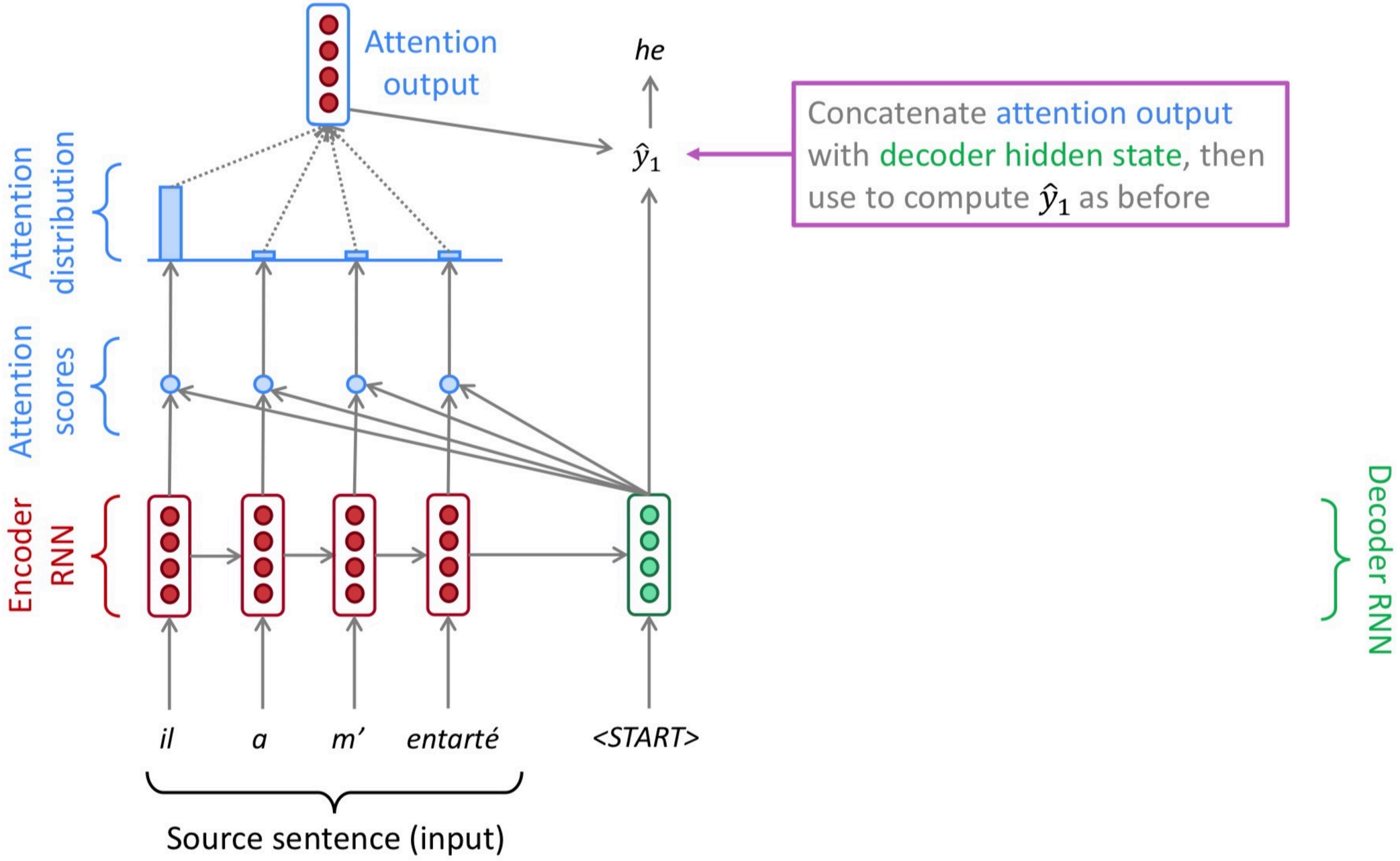
(slide credit: Abigail See)

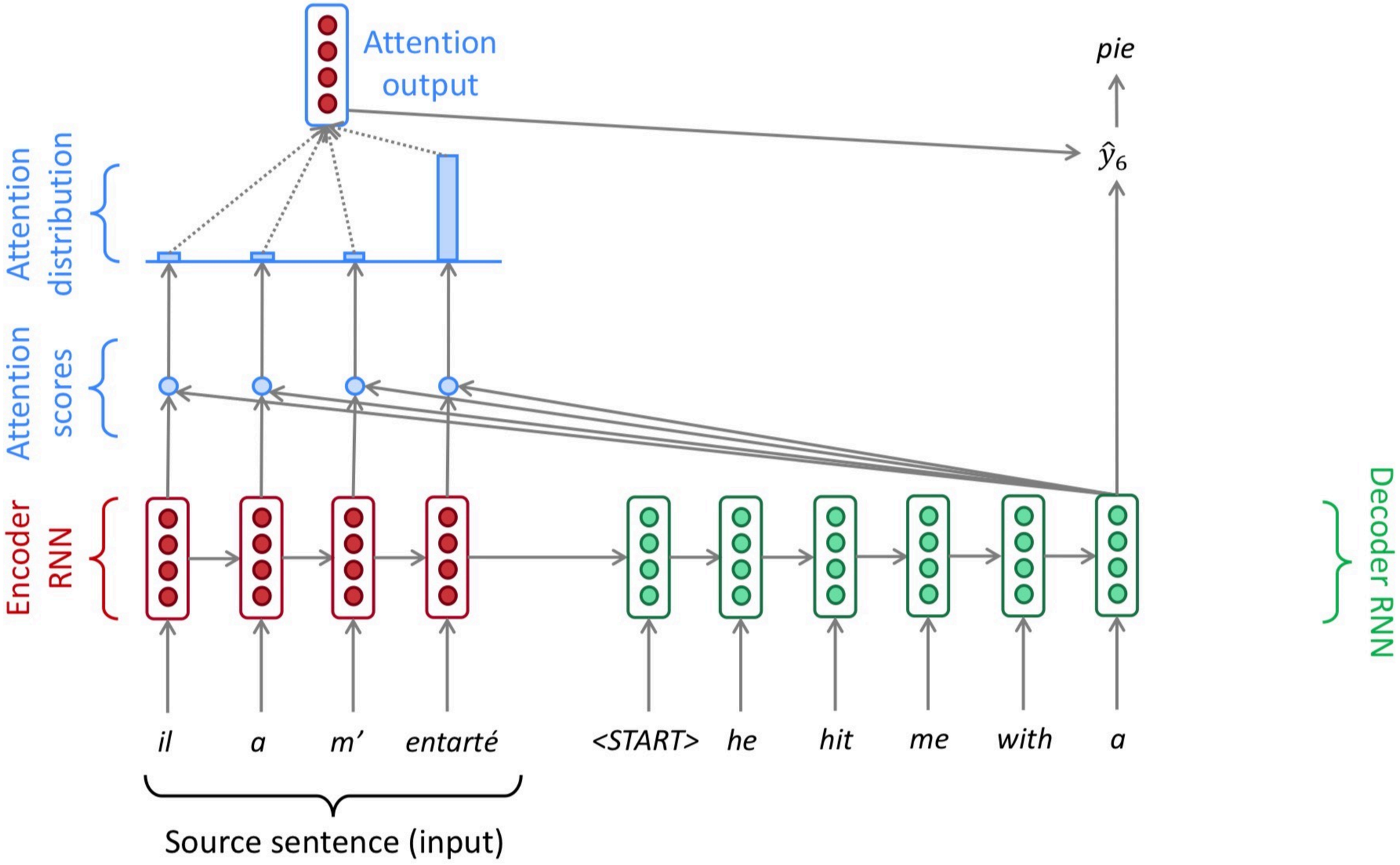

On this decoder timestep, we're mostly focusing on the first encoder hidden state ("*he*")
Take softmax to turn the scores into a probability distribution
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)



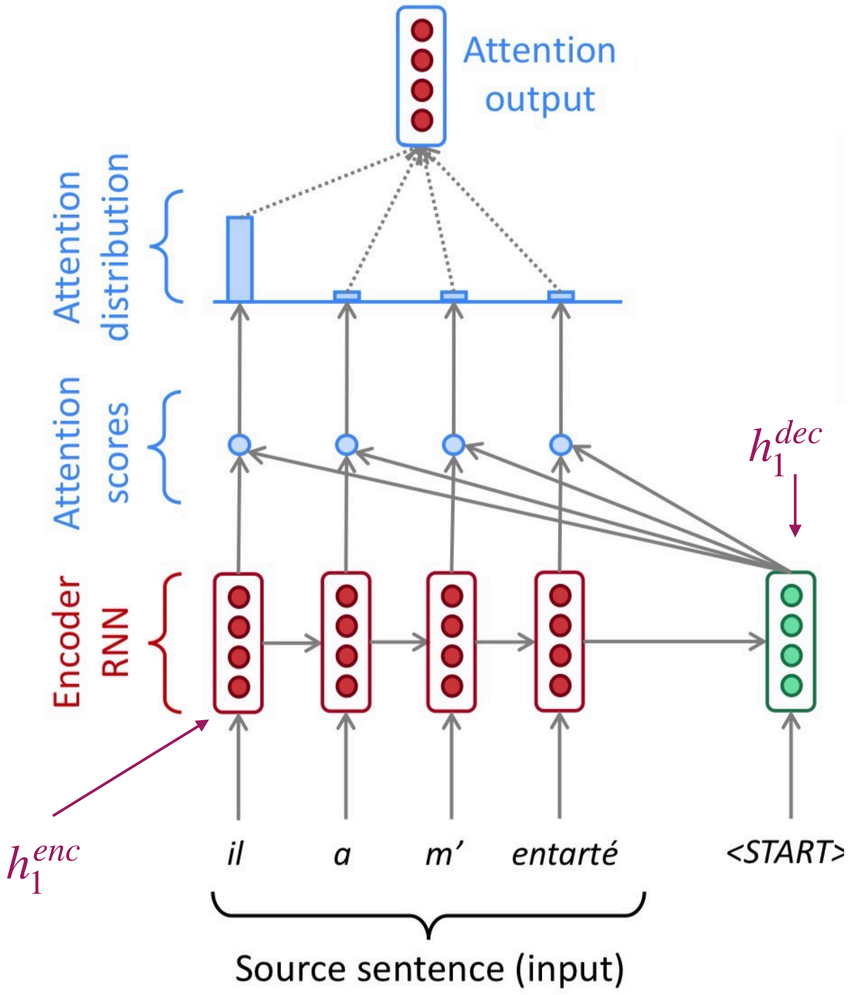
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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


Attention output
he
$\hat{y}_1$
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)


Attention output
pie
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
a
Source sentence (input)

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 of decoder (we'll define g soon):

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

- Obtain the attention distribution using softmax:

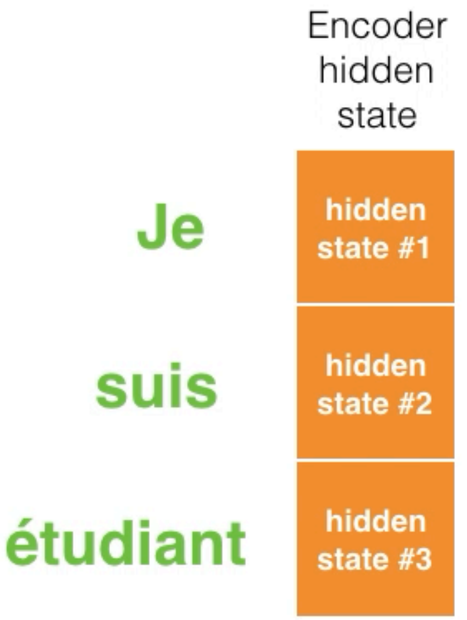
$$\alpha^t = \text{softmax}\,(e^t) \in \mathbb{R}^n$$

- 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:

$$[a_t; h_t^{dec}] \in \mathbb{R}^{2h}$$



(credits: Jay Alammar)

Types of attention

- Assume encoder hidden states $h_1^{enc}, h_2^{enc}, \ldots, h_n^{enc}$ and a decoder hidden state h^{dec}

1. **Dot-product attention** (assumes equal dimensions for h^{enc} and h^{dec}):

$$g(h_i^{enc}, h^{dec}) = (h^{dec})^T \; h_i^{enc} \in \mathbb{R}$$

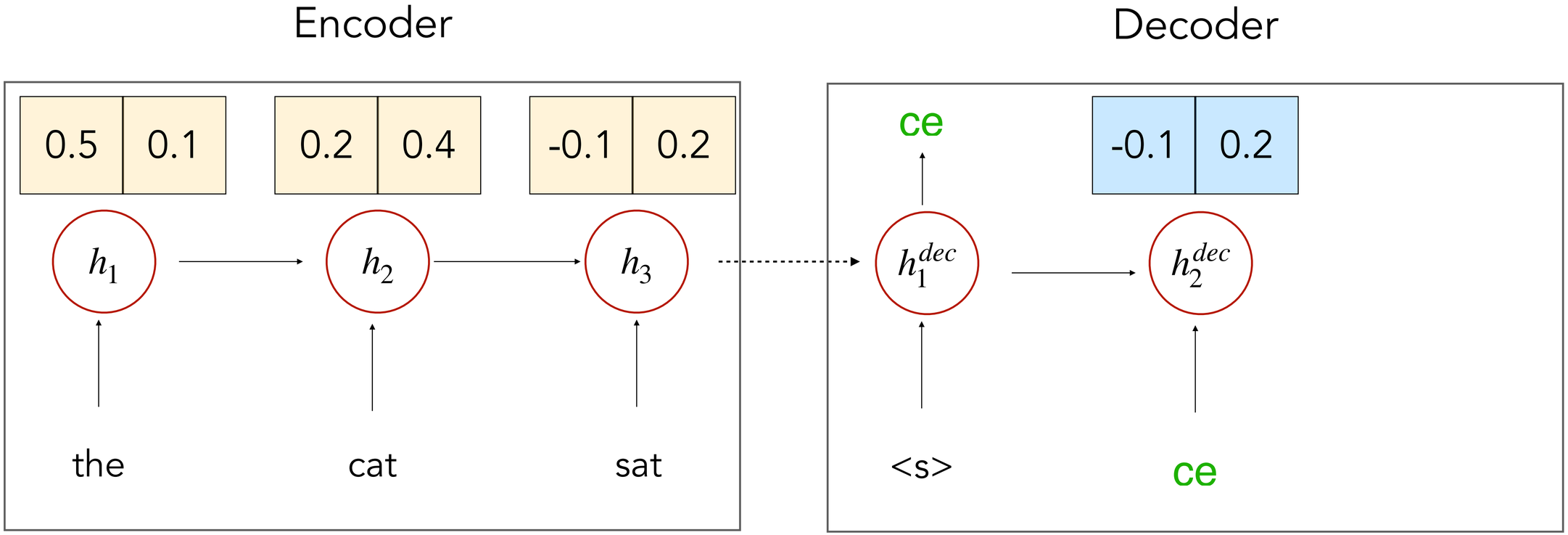
2. **Multiplicative attention:**

$$g(h_i^{enc}, h^{dec}) = (h^{dec})^T \; W \; h_i^{enc} \in \mathbb{R}$$, where W is a weight matrix (learned)

3. **Additive attention:**

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

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



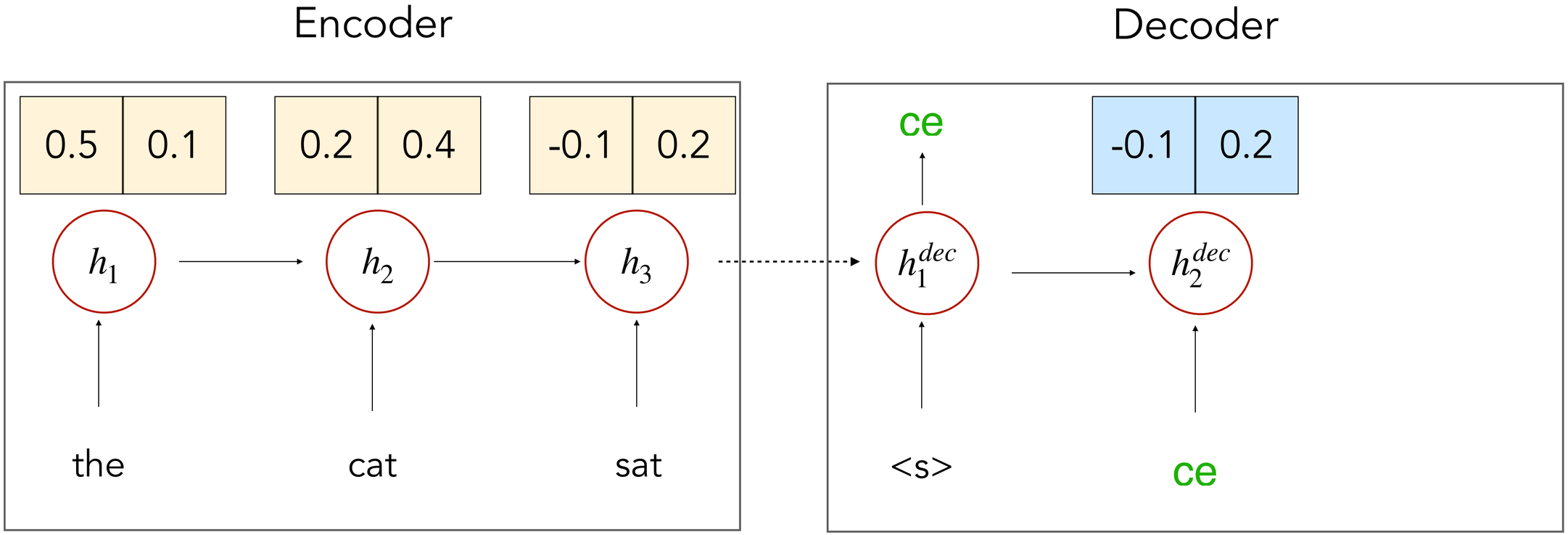
Dot-product attention:

$$g(h_i^{enc}, h^{dec}) = h^{dec} \cdot h^{enc}$$

Assuming we use dot product attention, which input word will have the highest attention value at current time step?

A) the
B) cat
C) sat

the -> -0.05 + 0.02
cat -> -0.02 + 0.08
sat -> 0.01 + 0.04



What if we use multiplicative attention with $W = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$?

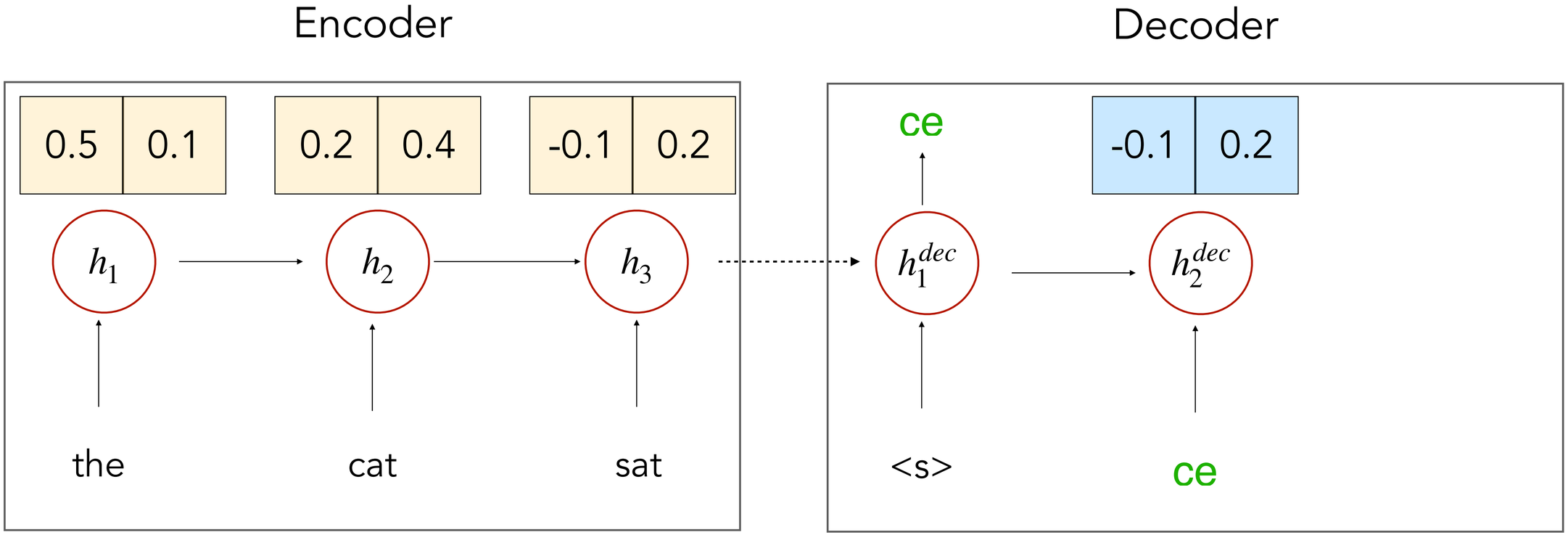
Which input word will have the highest attention value at current time step?

A) the
B) cat
C) sat

the -> -0.05
cat -> -0.02
sat -> 0.01

Multiplicative attention:

$$g(h_i^{enc}, h^{dec}) = (h^{dec})^T \; W \; h_i^{enc}$$



Which value of W in multiplicative attention will provide the same word with highest attention value as dot-product attention?

Multiplicative attention:

$g(h_i^{enc}, h^{dec}) = (h^{dec})^T \; W \; h_i^{enc}$

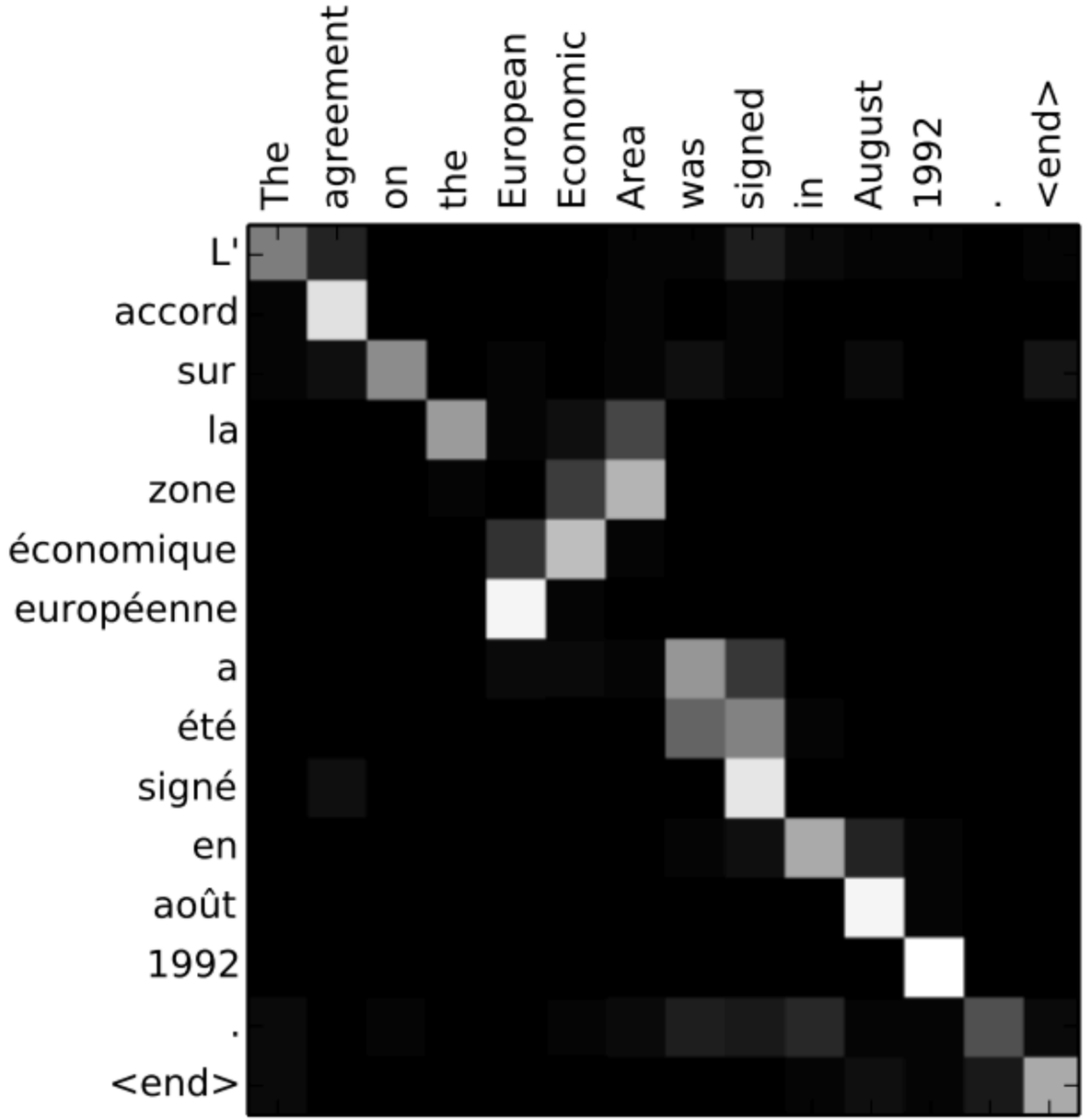
A) $W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ B) $W = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$ C) both

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*)

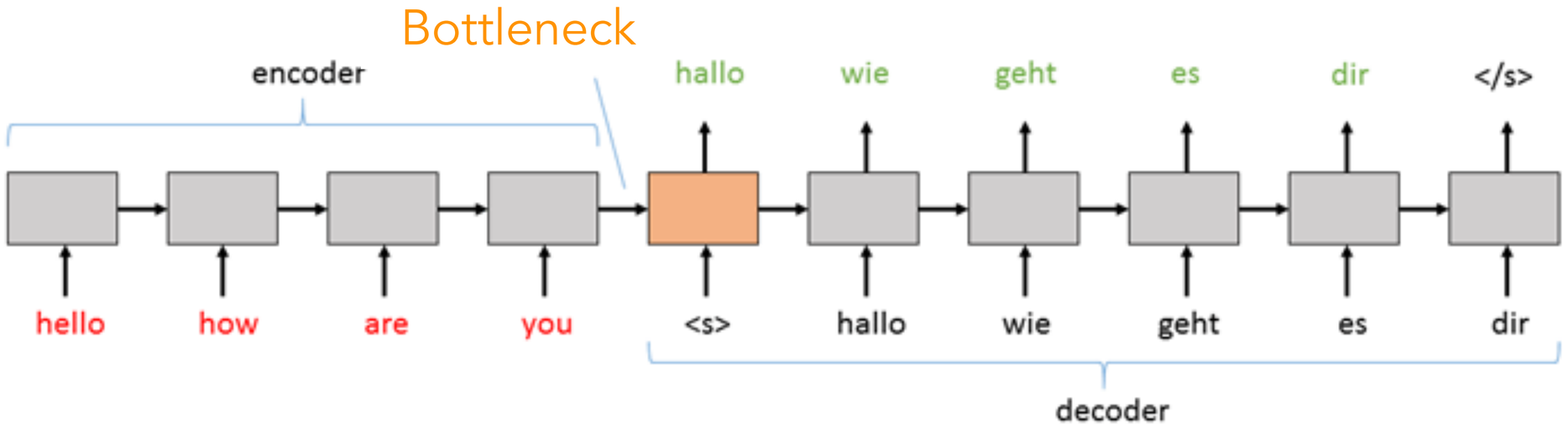
(Luong et al., 2015)

Visualizing attention



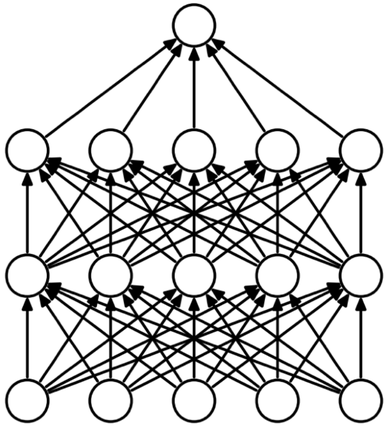
(credits: Jay Alammar)

Issues with vanilla seq2seq

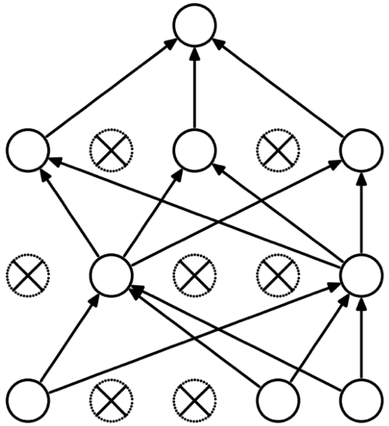


- A single encoding vector, h^{enc}, needs to capture all the information about source sentence
- Longer sequences can lead to vanishing gradients
- **Model may "overfit" to training sequences**

Dropout (advanced)



(a) Standard Neural Net



(b) After applying dropout.

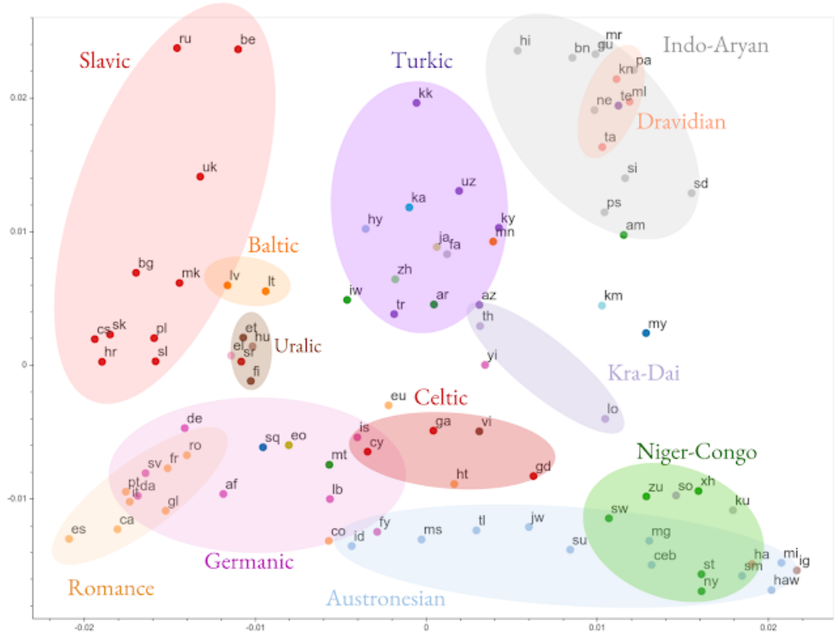
- Form of regularization for RNNs (and any NN in general)
- Idea: "Handicap" NN by removing hidden units **stochastically**
  - set each hidden unit in a layer to 0 with probability p during training ($p = 0.5$ usually works well)
  - scale outputs by $1/(1-p)$
  - hidden units forced to learn more general patterns and improve redundancy
- Test time: Simply compute identity

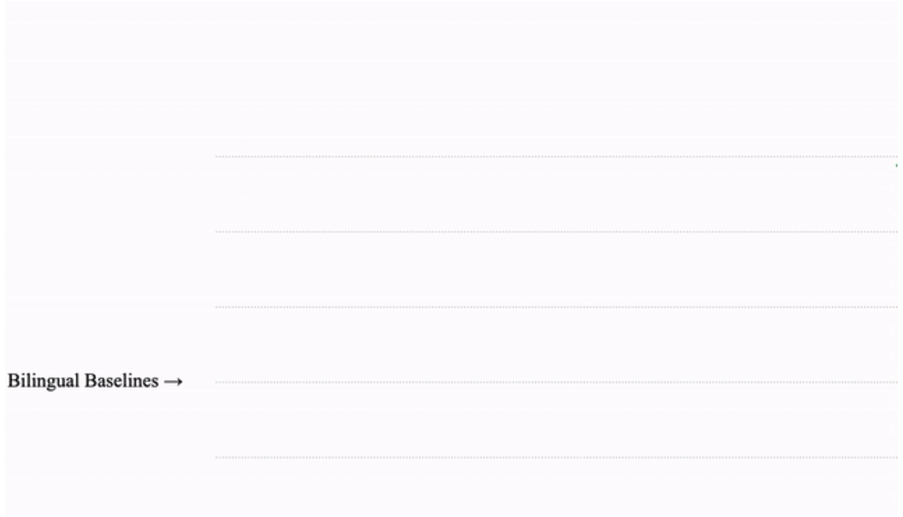
(Srivastava et al., 2014)

Other 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

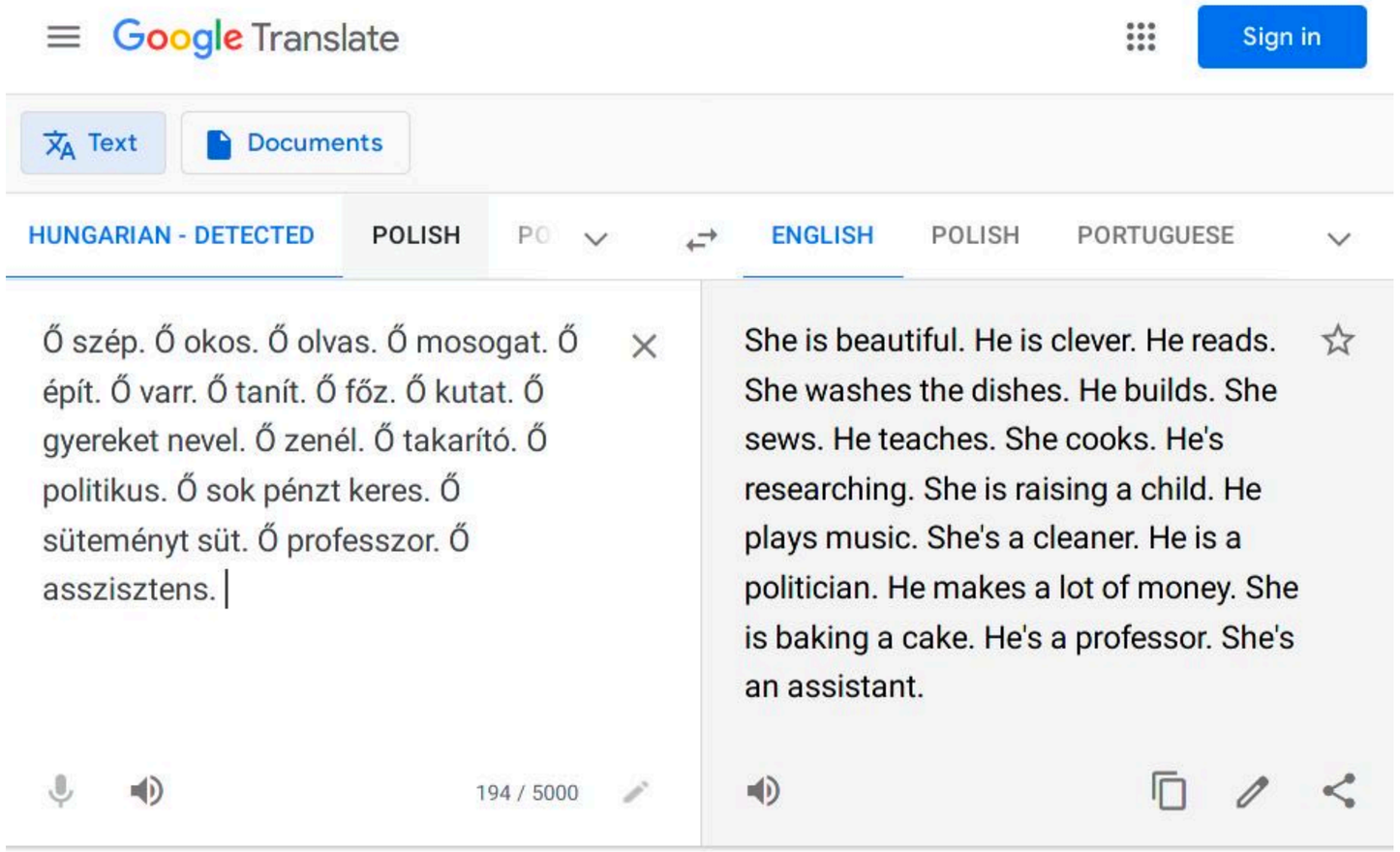
Massively multilingual MT

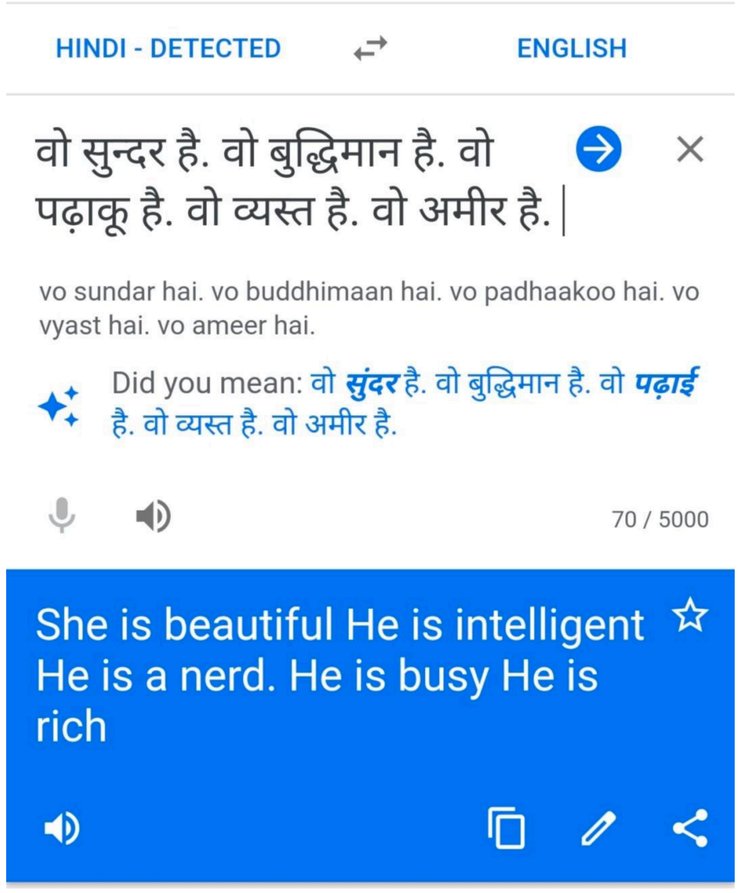




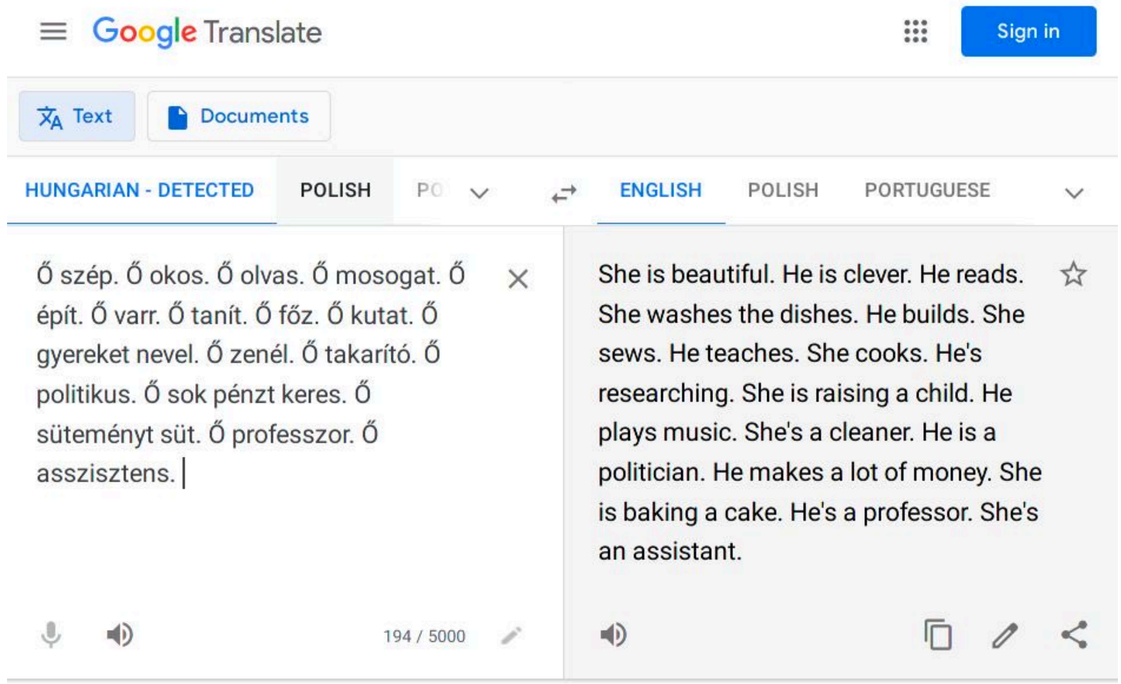
- Train a *single* neural network on 103 languages paired with English (remember Interlingua?)
- Massive improvements on low-resource languages

(Arivazhagan et al., 2019)


Google Translate
Sign in
Text
Documents
HUNGARIAN - DETECTED
POLISH
PO
ENGLISH
POLISH
PORTUGUESE
Ő szép. Ő okos. Ő olvas. Ő mosogat. Ő épít. Ő varr. Ő tanít. Ő főz. Ő kutat. Ő gyereket nevel. Ő zenél. Ő takarító. Ő politikus. Ő sok pénzt keres. Ő süteményt süt. Ő professzor. Ő asszisztens.
194 / 5000
She is beautiful. He is clever. He reads. She washes the dishes. He builds. She sews. He teaches. She cooks. He's researching. She is raising a child. He plays music. She's a cleaner. He is a politician. He makes a lot of money. She is baking a cake. He's a professor. She's an assistant.


HINDI - DETECTED
ENGLISH
वो सुन्दर है. वो बुद्धिमान है. वो पढ़ाकू है. वो व्यस्त है. वो अमीर है.
vo sundar hai. vo buddhimaan hai. vo padhaakoo hai. vo vyast hai. vo ameer hai.
Did you mean: वो सुंदर है. वो बुद्धिमान है. वो पढ़ाई है. वो व्यस्त है. वो अमीर है.
70 / 5000
She is beautiful He is intelligent He is a nerd. He is busy He is rich

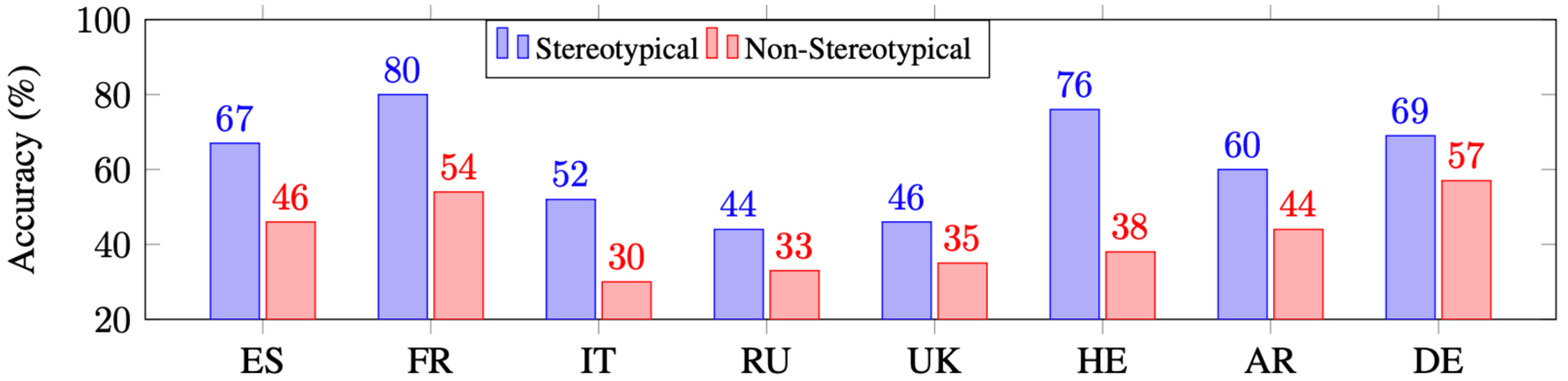
Bias and Fairness



- NMT systems suffer from issues of systematic bias (e.g. gender)
- Evident when translating from/to a language with gender-specific (or gender-agnostic) terms
- Models learn (and amplify) stereotypes from data

(Farkas and Nemeth, 2020)

Measuring bias in MT



- WinoMT: Stanovsky et al. (2019) use coreference resolution to construct a dataset of non-stereotypical gender roles
  - e.g. “The doctor asked the nurse to help **her** in the operation”
- Systems consistently performed worse on non-stereotypical gender translation

Source	[Target lang.] Predicted translation	Phenomenon
The janitor does not like **the baker** because **she** always messes up the kitchen.	[ES] Al conserje no le gusta **el panadero** porque **ella** siempre desordena la cocina.	Biased translation, giving "baker" a male inflection, with a mismatched pronoun reference.
The janitor does not like **the pretty baker** because **she** always messes up the kitchen.	[ES] Al conserje no le gusta **la panadera bonita** porque **ella** siempre desordena la cocina.	Adding a stereotypically female adjective "fixes" the translation.
The counselor asked **the guard** a few questions and praised **her** for the good work.	[FR] Le conseiller a posé quelques questions à **la garde** et **l'a louée** pour le bon travail.	French uses "garde" for both male and female guards, allowing for a more direct translation from English.

Table 5: Examples of Google Translate's output for different sentences in the WinoMT corpus. Words in **blue**, **red**, and **orange** indicate male, female and neutral entities, respectively.

(Stanovsky et al. 2019)

Anonymous feedback form:
https://forms.gle/875aEkJqodZcDx8H6