CS378: Natural Language Processing

Lecture 15: Neural Network (Sequence) Continued

The University of Texas at Austin

Eunsol Choi

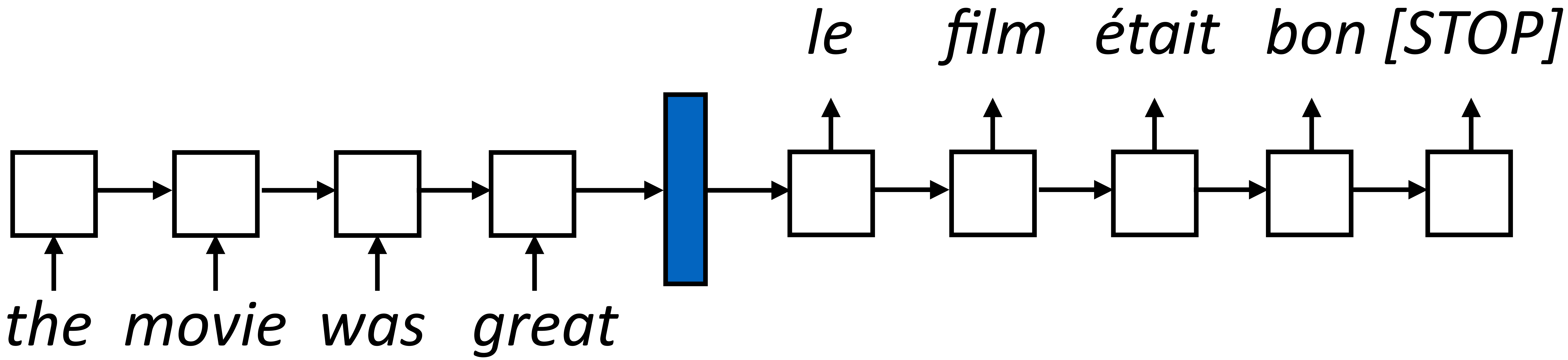
Slides from Greg Durrett, Yoav Artzi, Yejin Choi, Princeton NLP

Overview

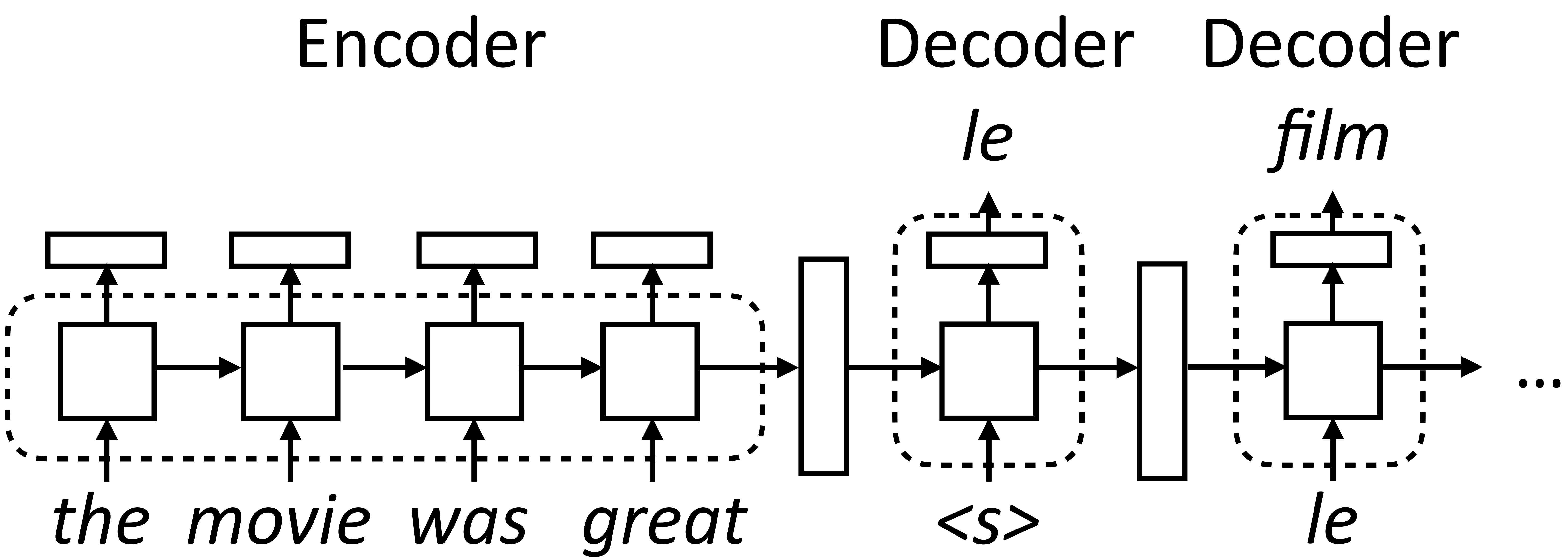
- Sequence to Sequence Model
  - Training
  - Inference
  - Applications
- Improving Seq2Seq Model
  - Attention

Seq2Seq Model

- Input: a sequence of tokens
- Output: a sequence of tokens (of ***arbitrary*** length)

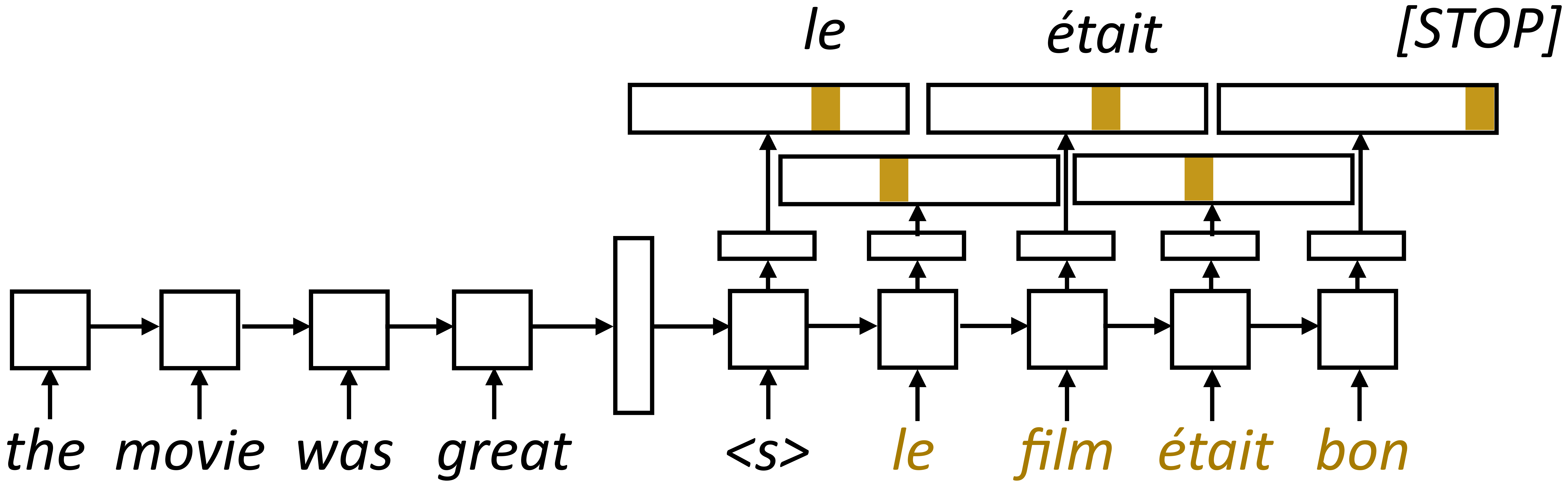


Implementing seq2seq Models



- Encoder: a RNN encoding a sequence of tokens, produces a vector.
- Decoder: separate RNN module (***different*** parameters).
  - Takes two inputs: hidden state and previous token.
  - Outputs token and a new hidden state.

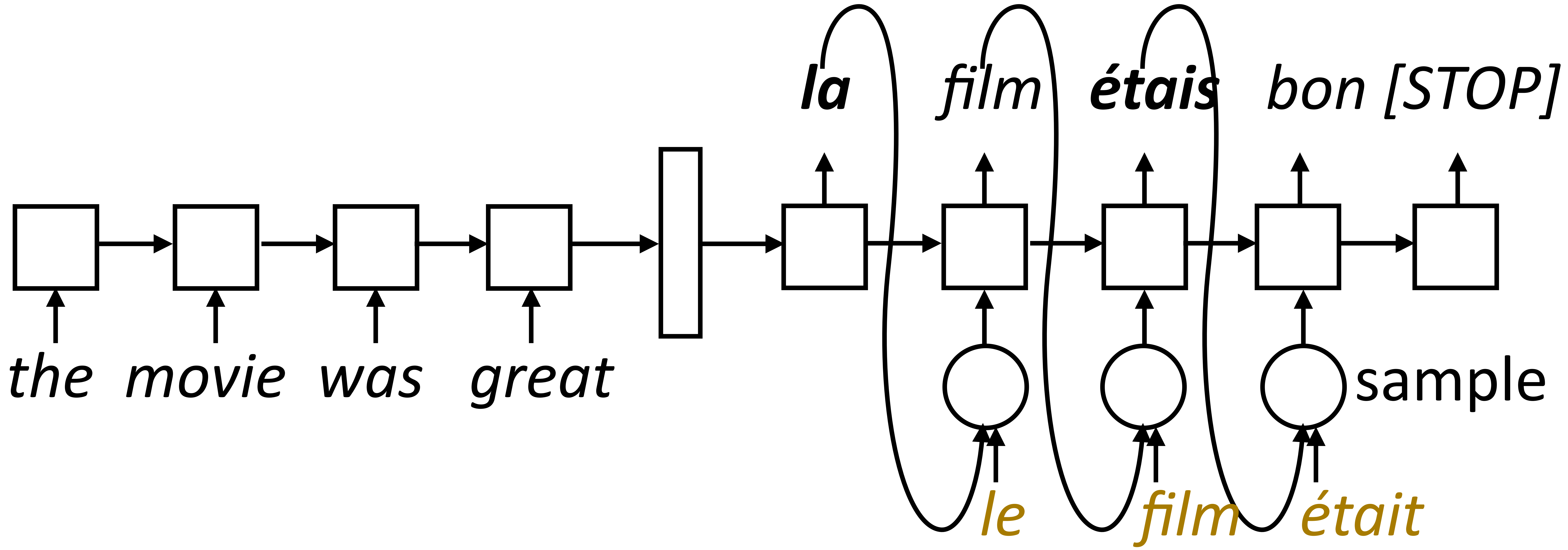
Training



- Objective: maximize $\sum_{(\mathbf{x},\mathbf{y})} \sum_{i=1}^{n} \log P(y_i^* | \mathbf{x}, y_1^*, \ldots, y_{i-1}^*)$
- One loss term for each target-sentence word, feed the correct word regardless of model's prediction (called "teacher forcing")
- Encoder and decoder parameters trained together! "End-to-End" training

Training: Scheduled Sampling

- Model needs to do the right thing even with its own predictions



- Scheduled sampling: with probability *p*, take the gold as input, else take the model's prediction
- Starting with *p* = 1 (teacher forcing) and decaying it works best

Bengio et al. (2015)

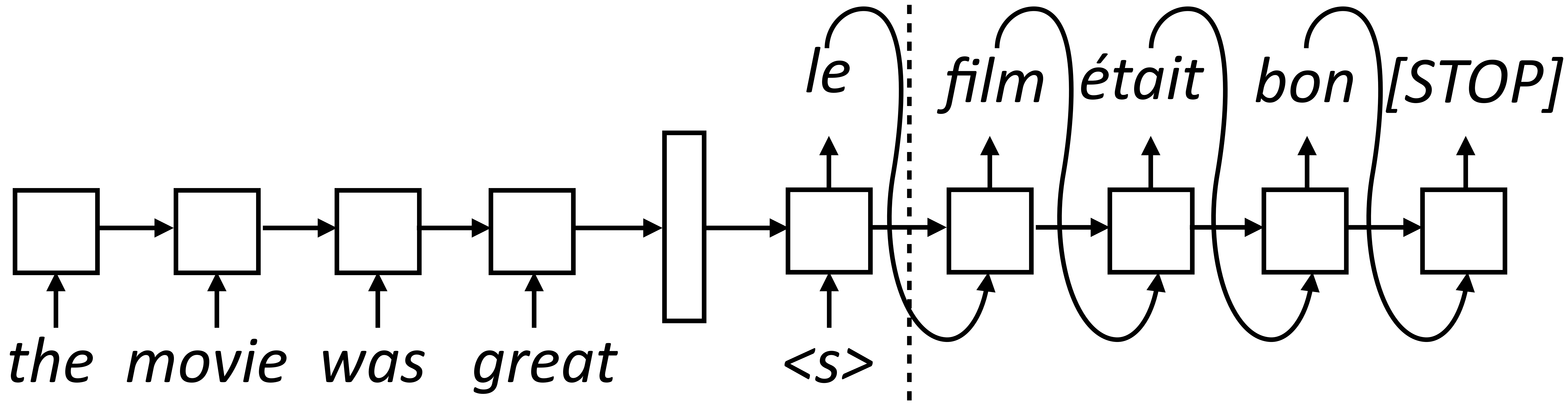
Overview

- Sequence to Sequence Model
  - Training
  - Inference
  - Applications

- Improving Seq2Seq Model
  - Attention

Inference

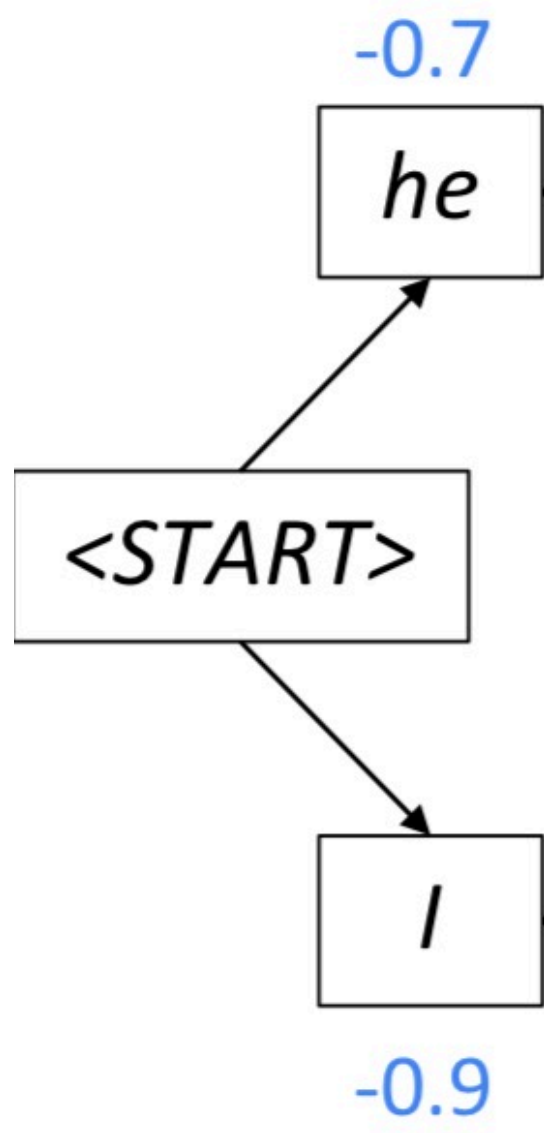
- Generate next word conditioned on previous word as well as hidden state



- Need to compute the argmax over the word predictions and then feed that to the next RNN state
- Decoder is advanced one state at a time until [STOP] is reached
- It’s a greedy decoding! ArgMax at each step!

Beam Search

- At each step, keep track of top K (beam size) hypotheses
- Based on score$(y_1, y_2 \ldots y_t) = \sum_{t=1}^{t} \log P(y_i \mid y_1, \ldots y_{i-1}, \mathbf{x})$

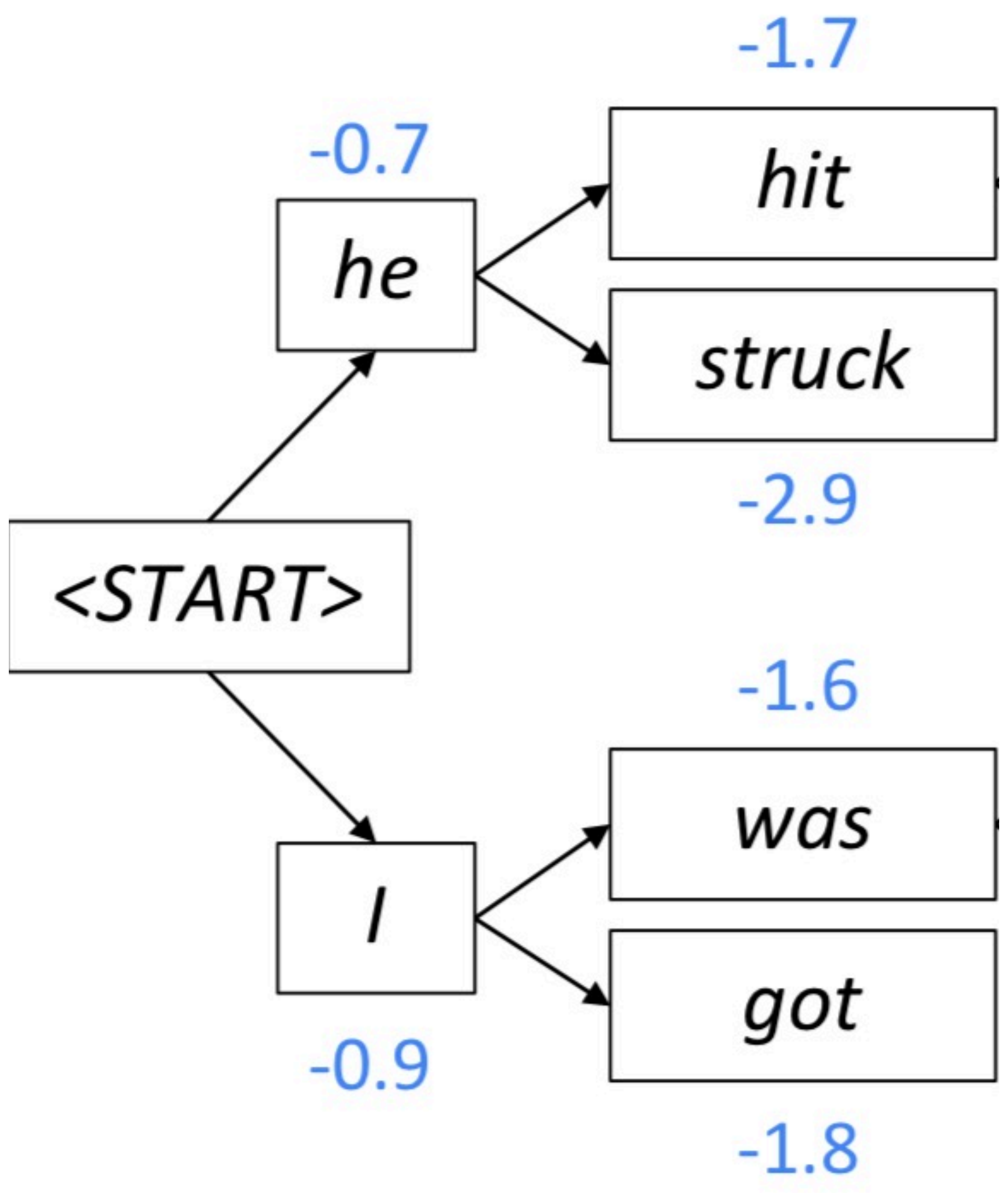


- Beam size: 2

(slide credit: Abigail See)

Beam Search

- At each step, keep track of top K (beam size) hypotheses
- Based on score$(y_1, y_2 \ldots y_t) = \sum_{t=1}^{t} \log P(y_i \mid y_1, \ldots y_{i-1}, \mathbf{x})$

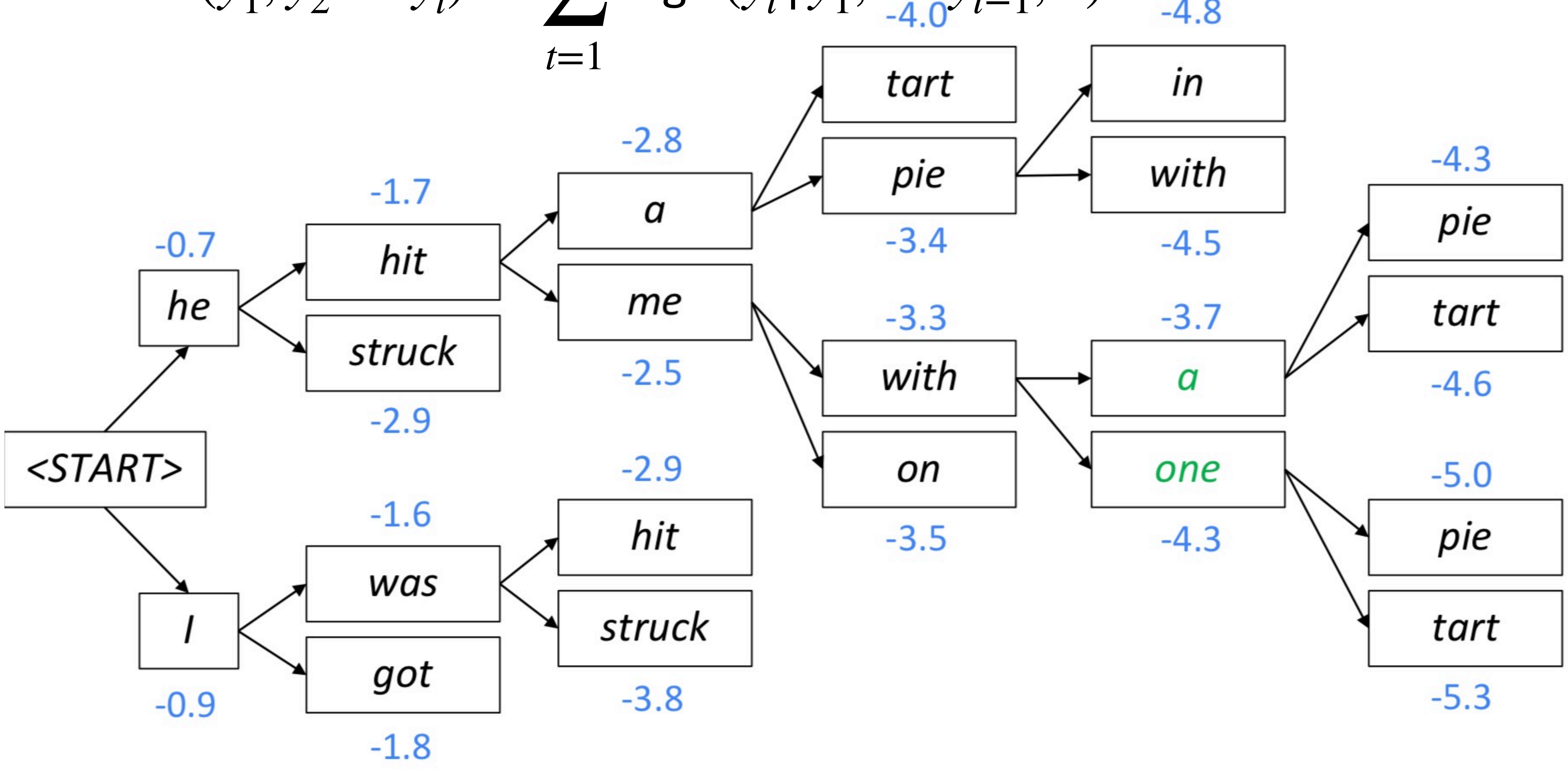


- Beam size: 2

(slide credit: Abigail See)

Beam Search

- At each step, keep track of top K (beam size) hypotheses
- Based on score$(y_1, y_2 \ldots y_t) = \sum_{t=1}^{t} \log P(y_i \mid y_1, \ldots y_{i-1}, \mathbf{x})$

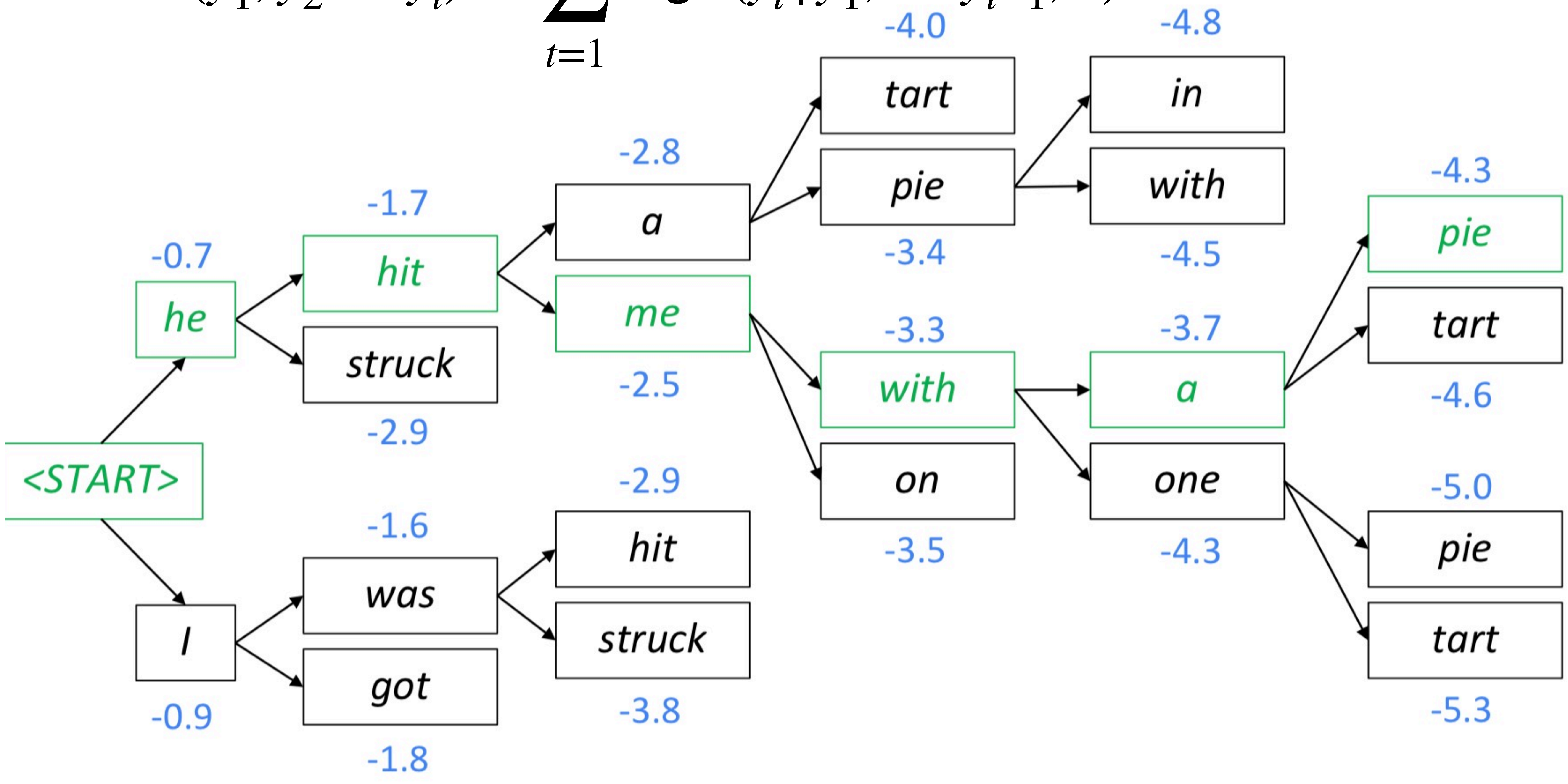


- Beam size: 2

(slide credit: Abigail See)

Beam Search

- At each step, keep track of top K (beam size) hypotheses
- Based on score$(y_1, y_2 \ldots y_t) = \sum_{t=1}^{t} \log P(y_i \mid y_1, \ldots y_{i-1}, \mathbf{x})$



- Beam size: 2

(slide credit: Abigail See)

Applications of Seq2Seq Model

Applications of Seq2Seq

- Semantic parsing:

What states border Texas ⟶ `λ x state( x ) ∧ borders( x , e89 )`

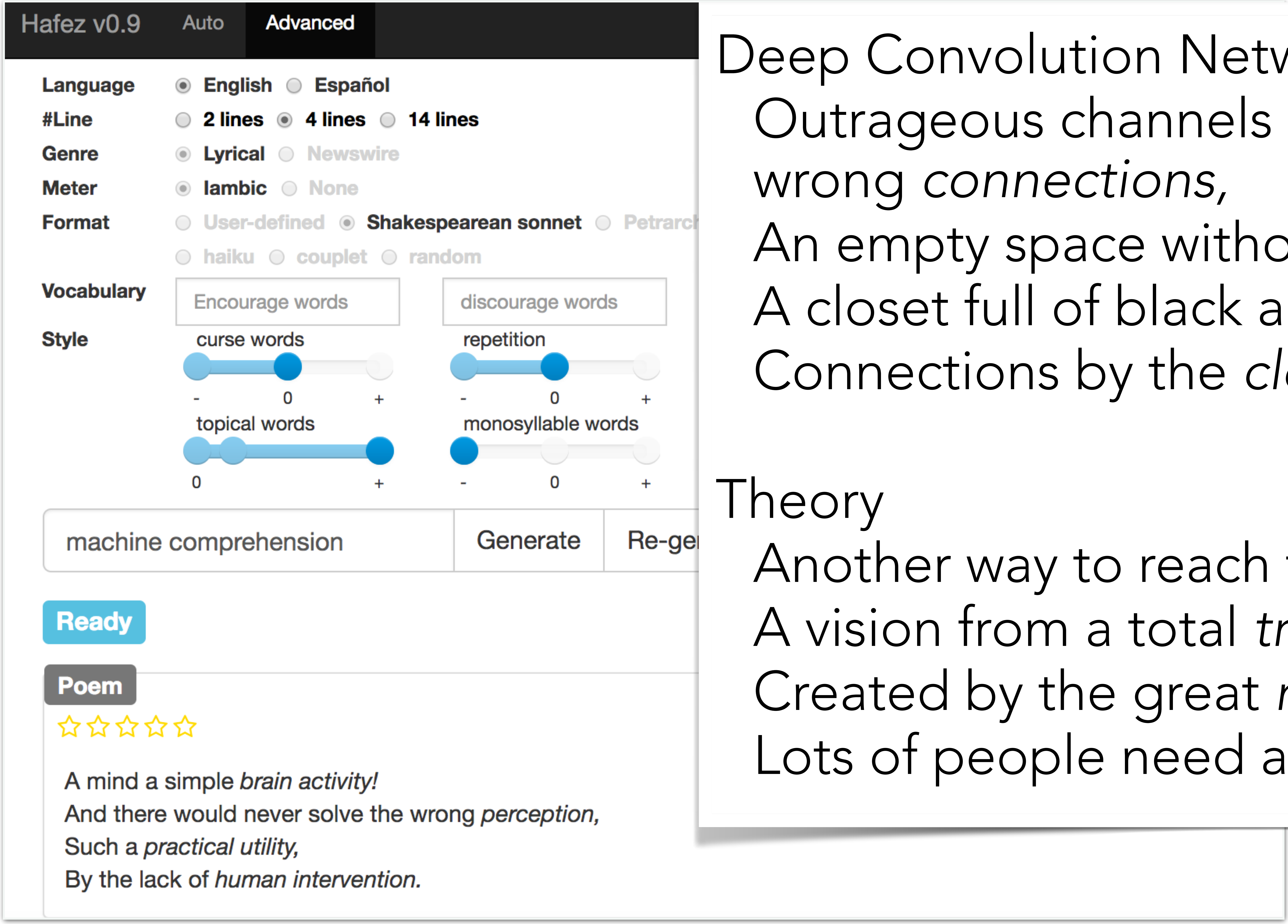
- Syntactic parsing

The dog ran ⟶ `(S (NP (DT the) (NN dog) ) (VP (VBD ran) ) )`

(but what if we produce an invalid tree or one with different words?) 🤔

- Machine translation, summarization, dialogue can all be viewed in this framework as well

Applications of Seq2Seq

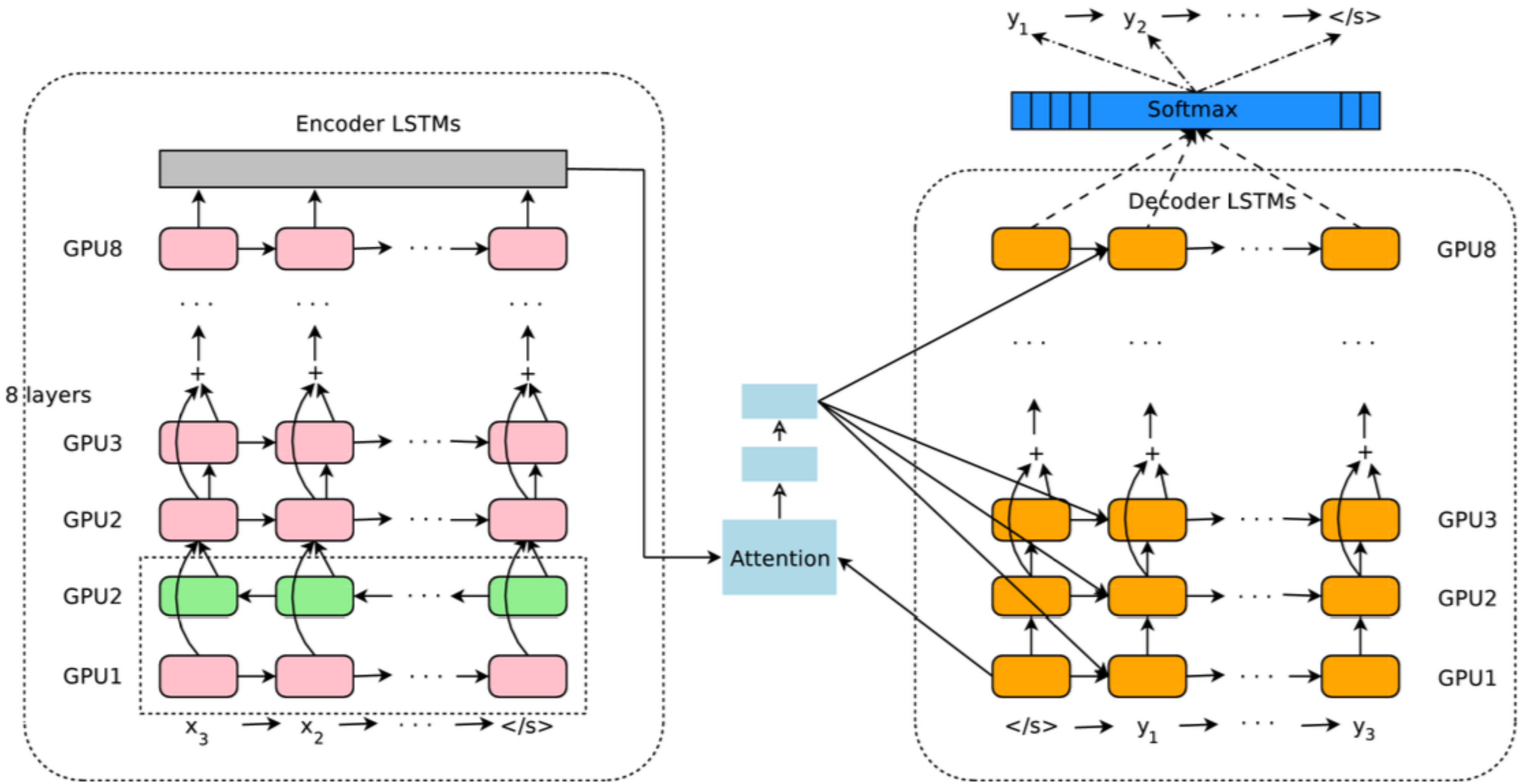


Deep Convolution Network
Outrageous channels on the
wrong *connections*,
An empty space without an open *layer*,
A closet full of black and blue *extensions*,
Connections by the *closure operator*.

Theory
Another way to reach the wrong *conclusion!*
A vision from a total *transformation*,
Created by the great *magnetic fusion*,
Lots of people need an *explanation*.

Hafez: Neural Sonnet Writer
(Ghazvininejad et al. 2016)

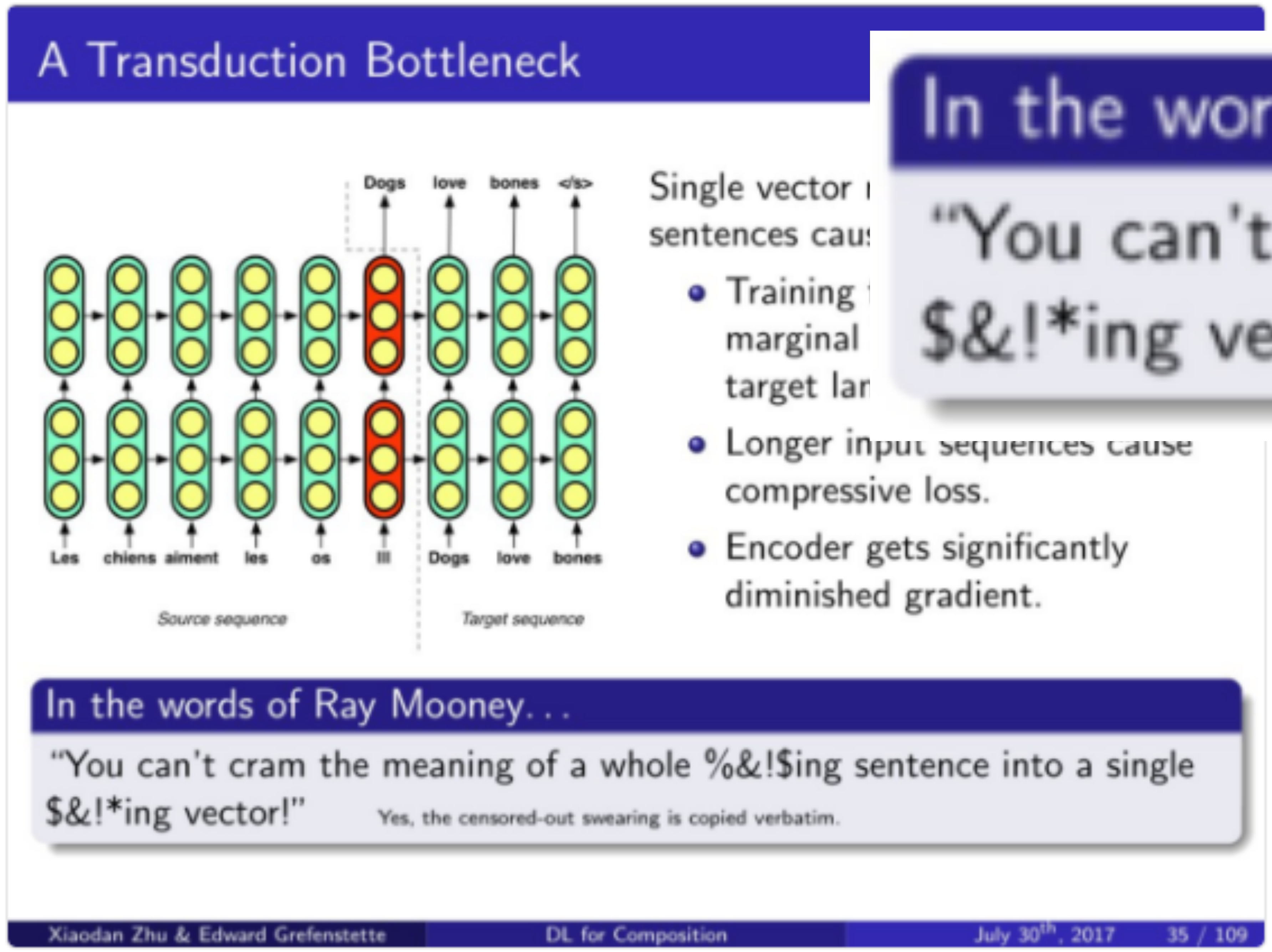
Applications: Google NMT (Oct 2016)



Problem?

Edward Grefenstette
@egrefen
Follow

It's not an ACL tutorial on vector representations of meaning if there isn't at least one Ray Mooney quote.



In the words of Ray Mooney...

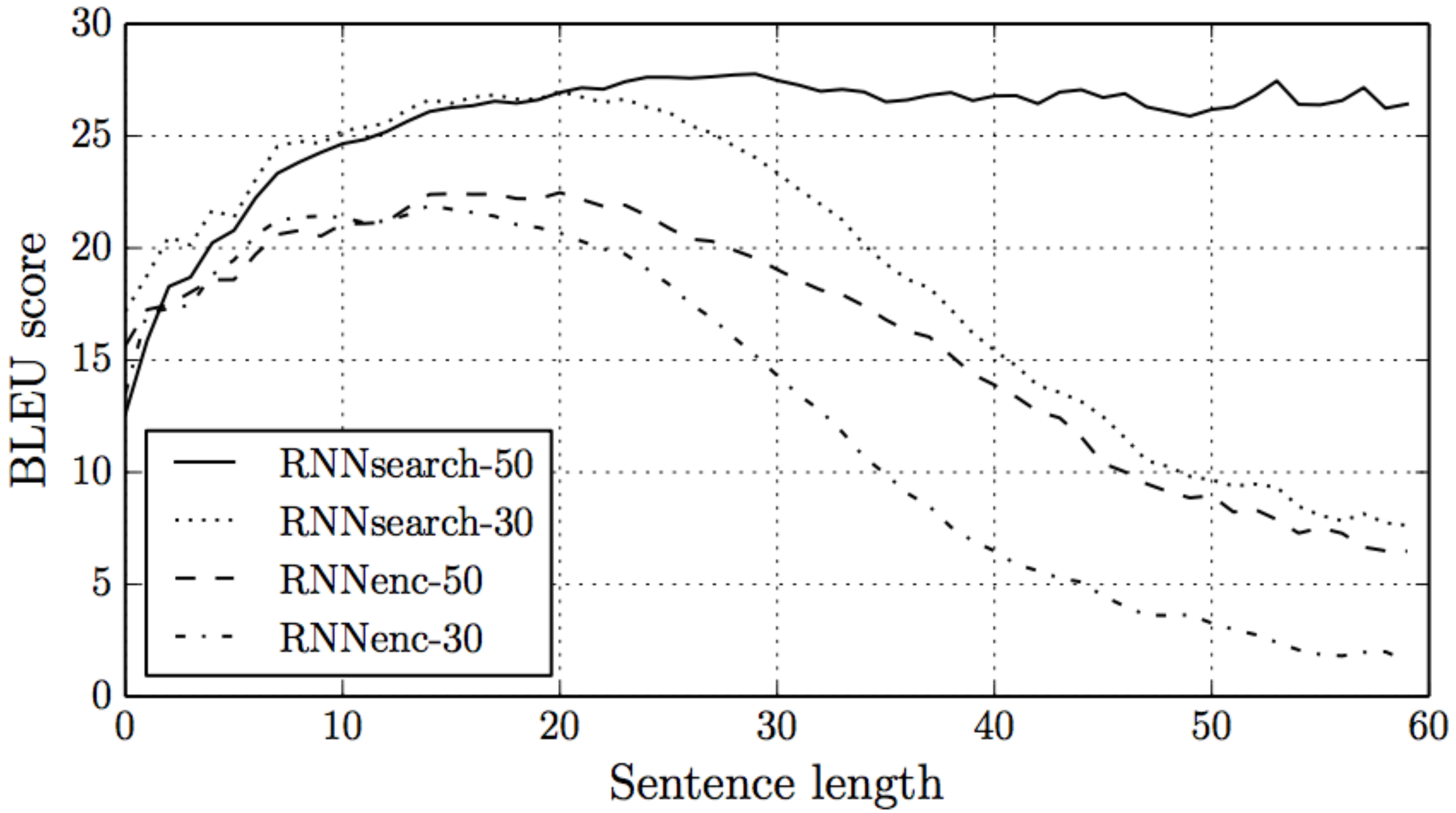
"You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!" Yes, the censored-out swearing is copied verbatim.

12:27 AM - 11 Jul 2017

20 Retweets 127 Likes

Problems with Seq2seq Models

- Bad at long sentences



RNNenc: the model we've discussed so far

RNNsearch: uses attention

Bahdanau et al. (2014)

Vanilla Seq2Seq Likes to Repeat

- Encoder-decoder models like to repeat themselves:

Un garçon joue dans la neige → A boy plays in the snow **boy plays boy plays**

sausage sandwiches → Cut each sandwich in halves.
Sandwiches with sandwiches.
Sandwiches, sandwiches, Sandwiches, sandwiches, sandwiches
sandwiches, sandwiches, sandwiches, sandwiches, sandwiches, sandwiches, or sandwiches or triangles, a griddle, each sandwich.....

Kiddon et al., 2016

Problems with Seq2seq Models

- Why does such repeating happen?
  - Models trained poorly
  - Input is forgotten by the LSTM so it gets stuck in a “loop” of generating the same output tokens again and again
- Need some notion of input coverage or what input words we’ve translated

Problems with Seq2seq Models

- Unknown words:

en: The *ecotax* portico in *Pont-de-Buis* , . . . [truncated] . . . , was taken down on Thursday morning

fr: Le *portique* *écotaxe* de *Pont-de-Buis* , . . . [truncated] . . . , a été *démonté* jeudi matin

nn: Le unk de unk à unk , . . . [truncated] . . . , a été pris le jeudi matin

- Encoding these rare words into a vector space is really hard

- In fact, we don't want to encode them, we want a way of directly looking back at the input and copying them

Overview

- Sequence to Sequence Model
  - Training
  - Inference
  - Applications
- Improving Seq2Seq Model
  - Attention

Attention

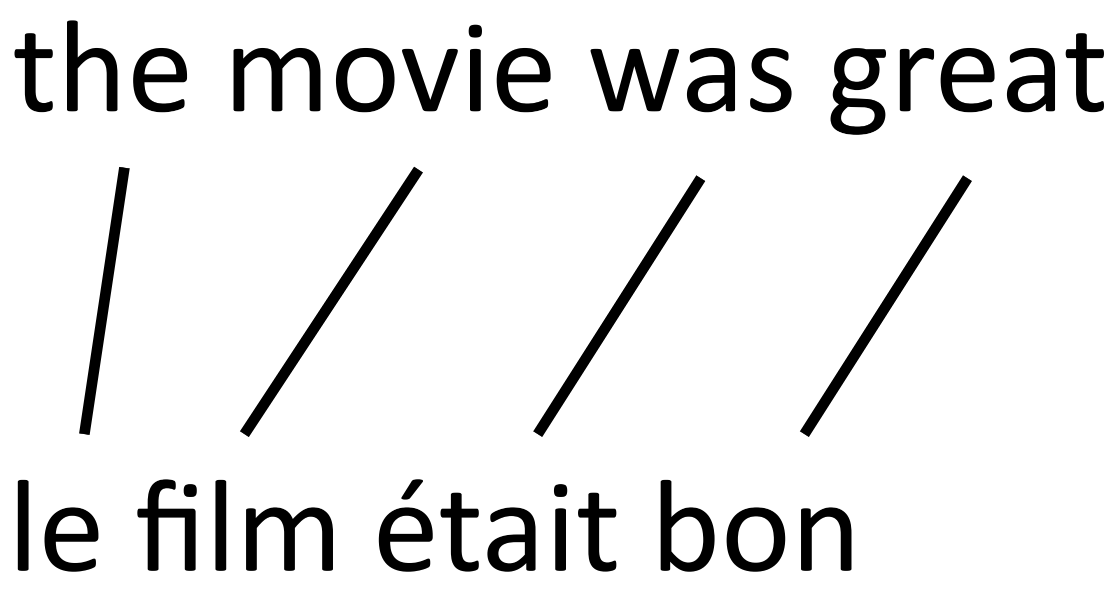
“*what states border* *Texas*” ⟶ lambda x (state (x) and border (x , e89)))

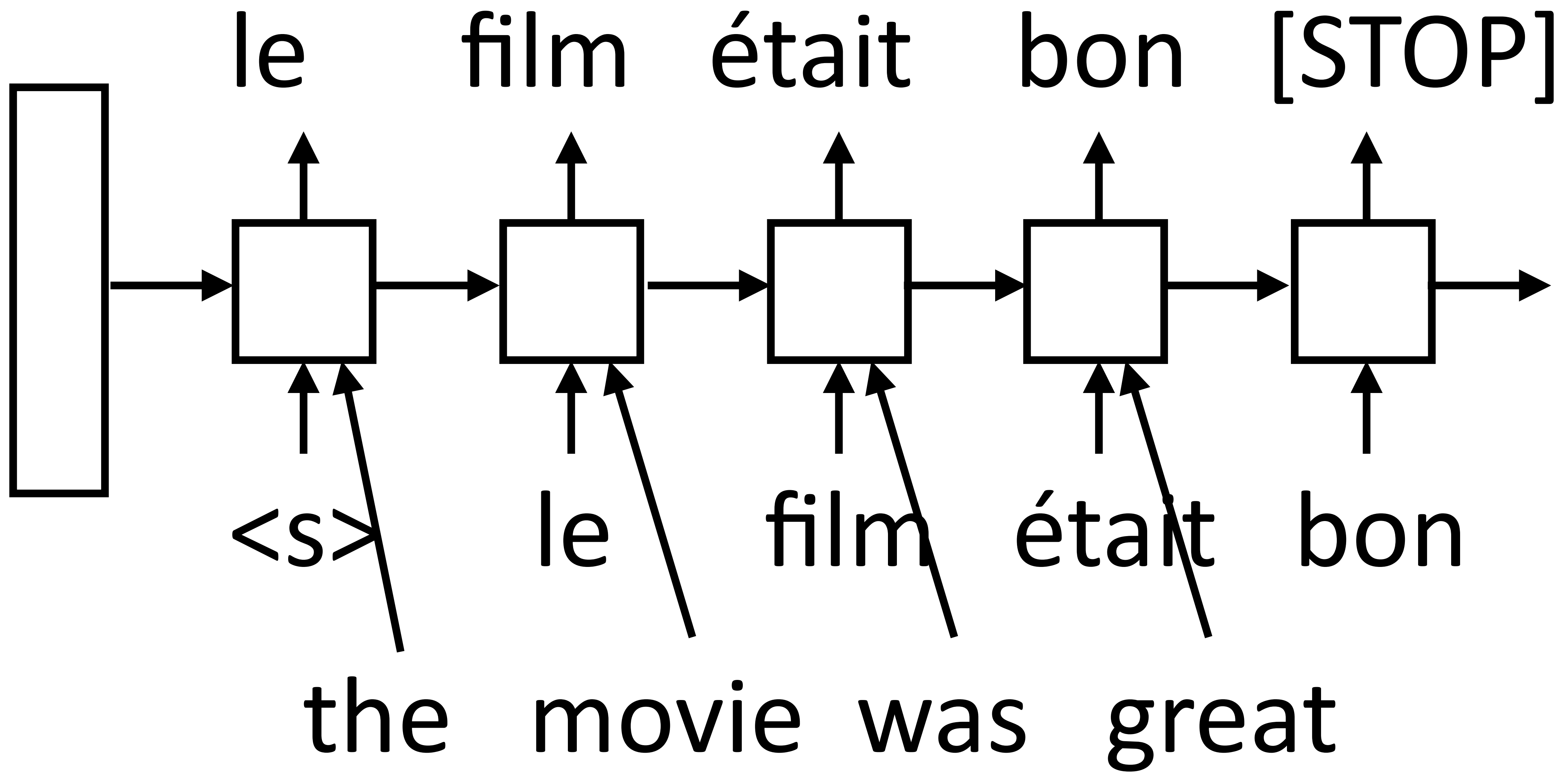
- Orange pieces are probably reused across many problems

- Not too hard to learn to generate: start with lambda, always follow with x, follow that with paren, etc.

- LSTM has to remember the value of Texas for 13 steps!

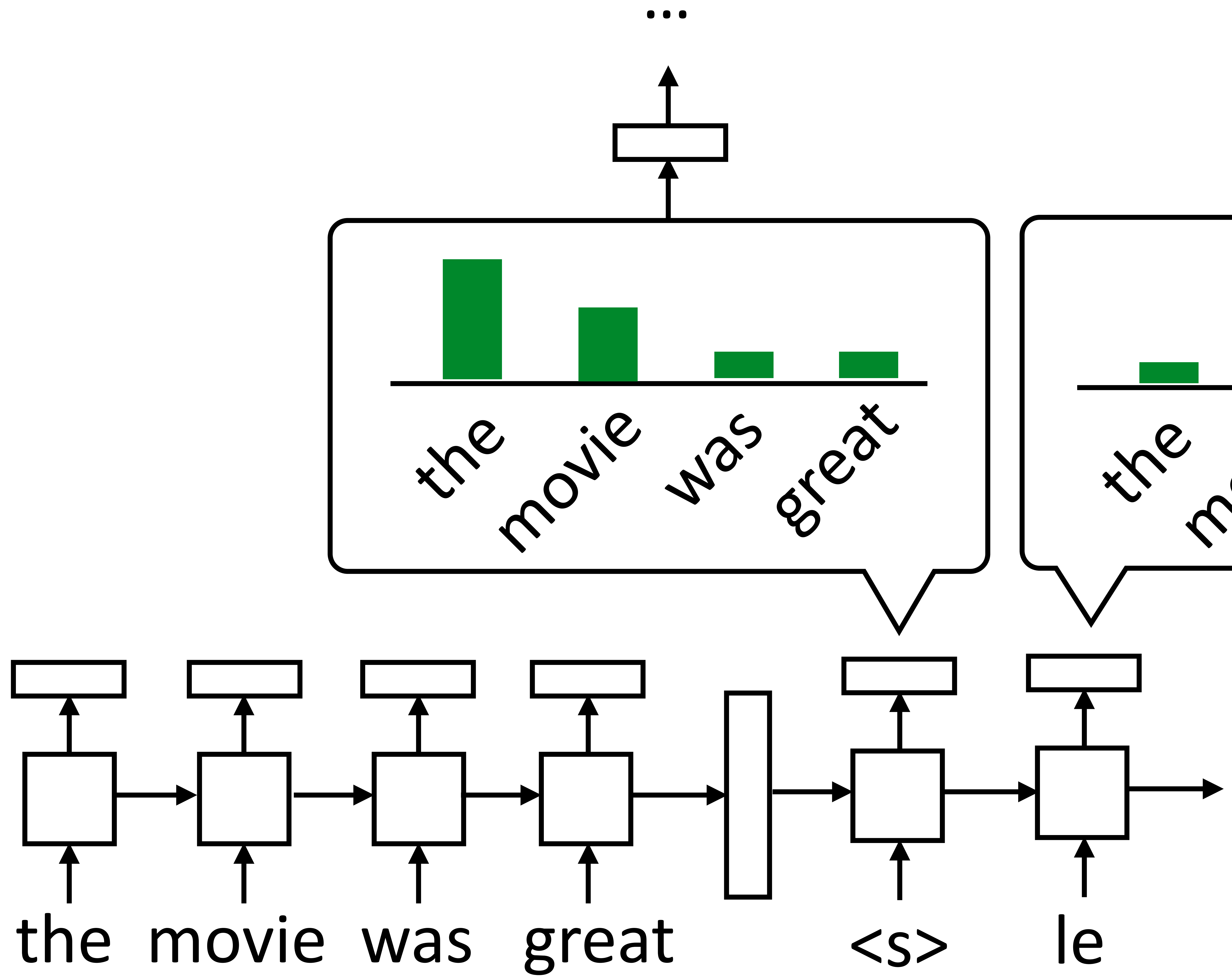
First Try: Aligned Inputs





Problem?

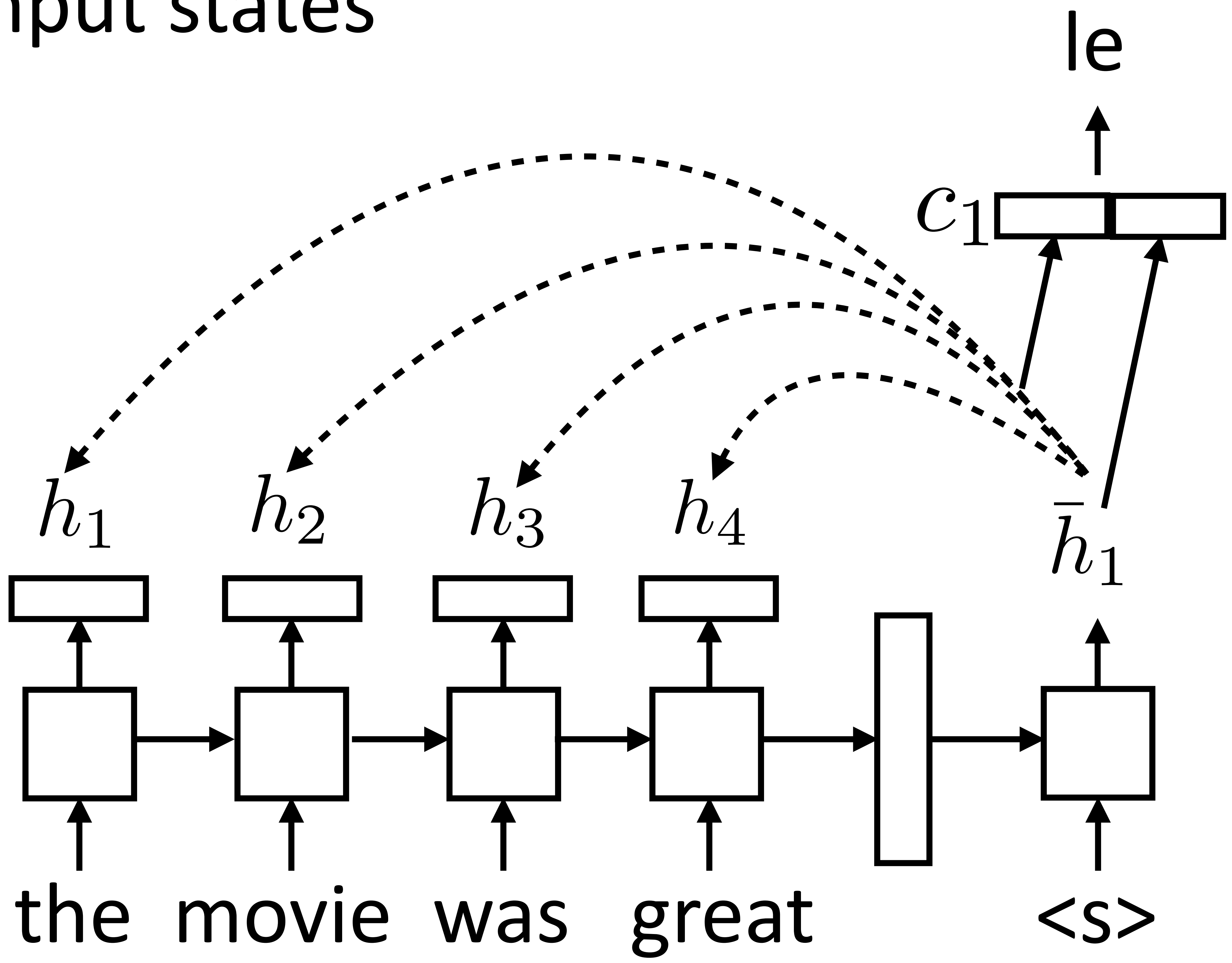
Attention



- At each decoder state, compute a distribution over source inputs based on current decoder state, use that in output layer

Attention

- For each decoder state, compute weighted sum of input states



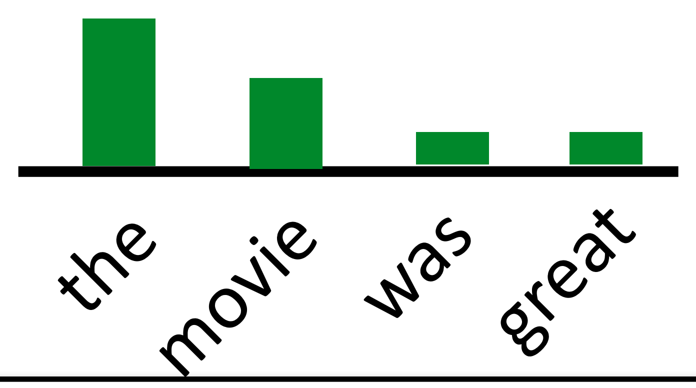
- No attn: $P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h}_i)$

$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W[c_i; \bar{h}_i])$$

$$c_i = \sum_j \alpha_{ij} h_j$$

Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

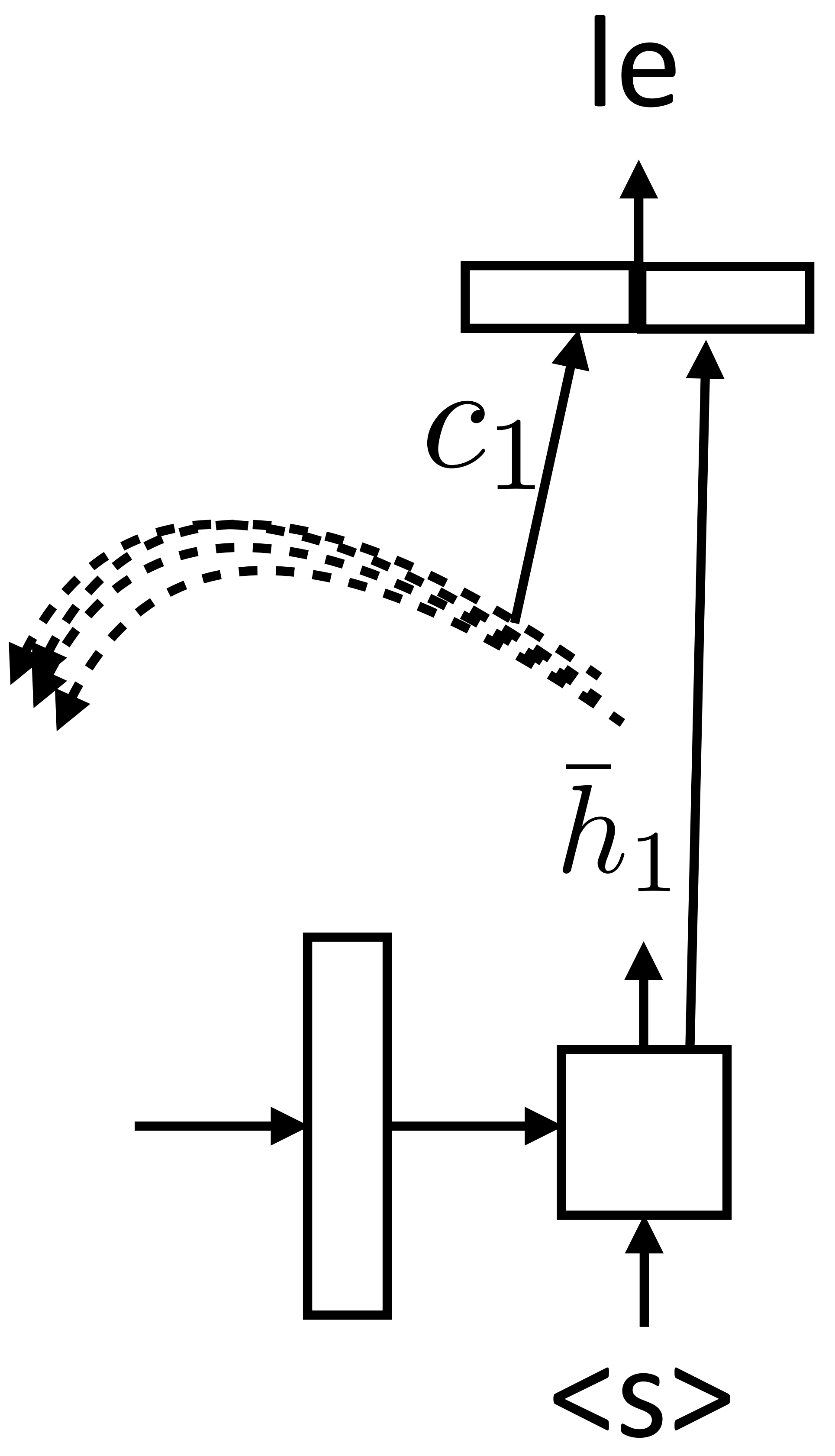


Attention weight for input x_j at decoding y_i

$$e_{ij} = f(\bar{h}_i, h_j)$$

- Some function f (next slide)

Attention



$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

$$e_{ij} = f(\bar{h}_i, h_j)$$

$h_i, h_j \in R^{d_k}$

- Bahdanau+ (2014): additive

$$f(\bar{h}_i, h_j) = \tanh(W[\bar{h}_i, h_j])$$

- Luong+ (2015): dot product

$$f(\bar{h}_i, h_j) = \bar{h}_i \cdot h_j$$

$$f(\hat{h}_i, h_j) = \frac{\bar{h}_i \cdot h_j}{\sqrt{d_k}}$$

- Luong+ (2015): bilinear

$$f(\bar{h}_i, h_j) = \bar{h}_i^\top W h_j$$

Luong et al. (2015)

Learned Attention

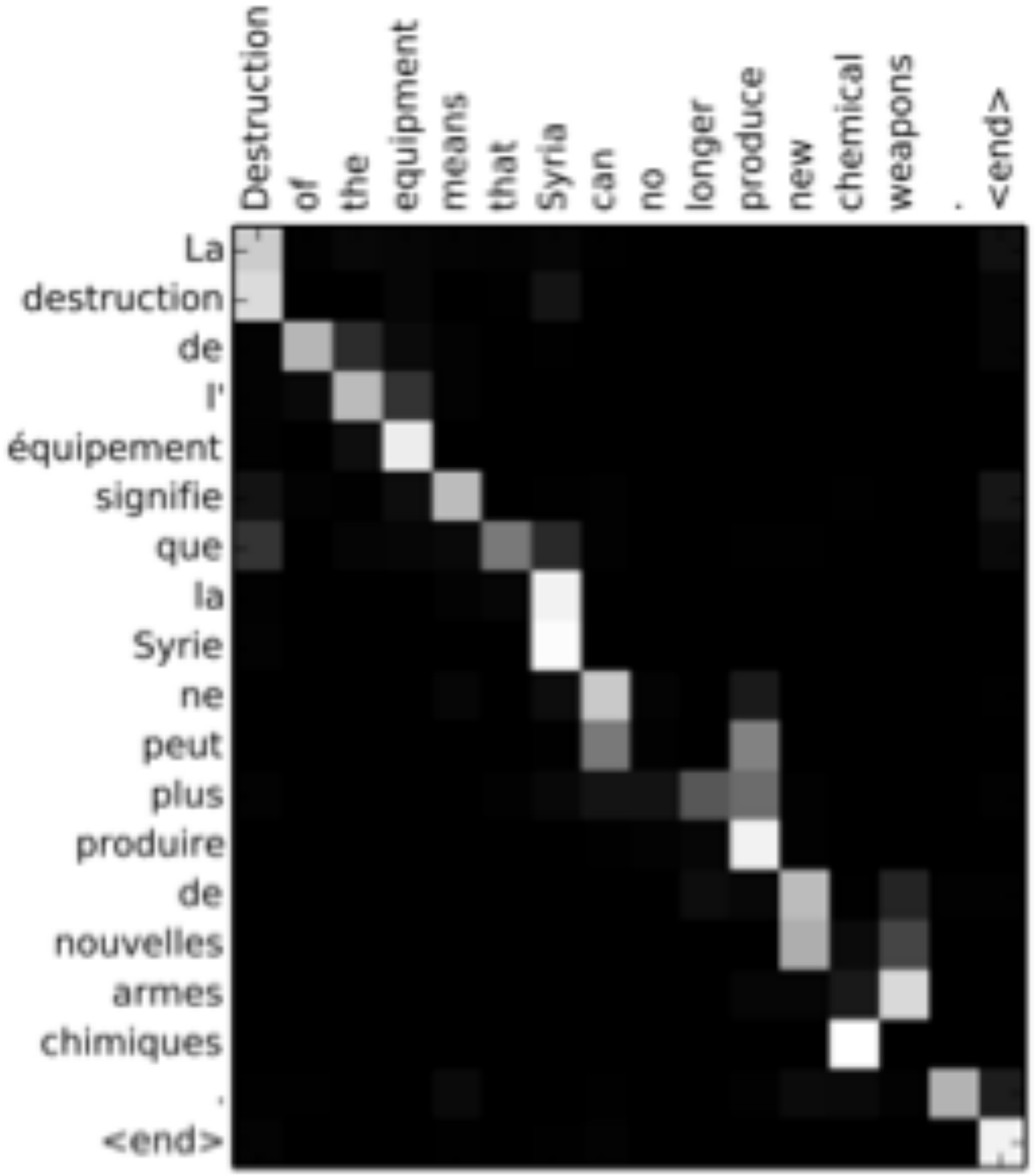


Diagram borrowed from Alex Rush

Attention for Captioning

Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

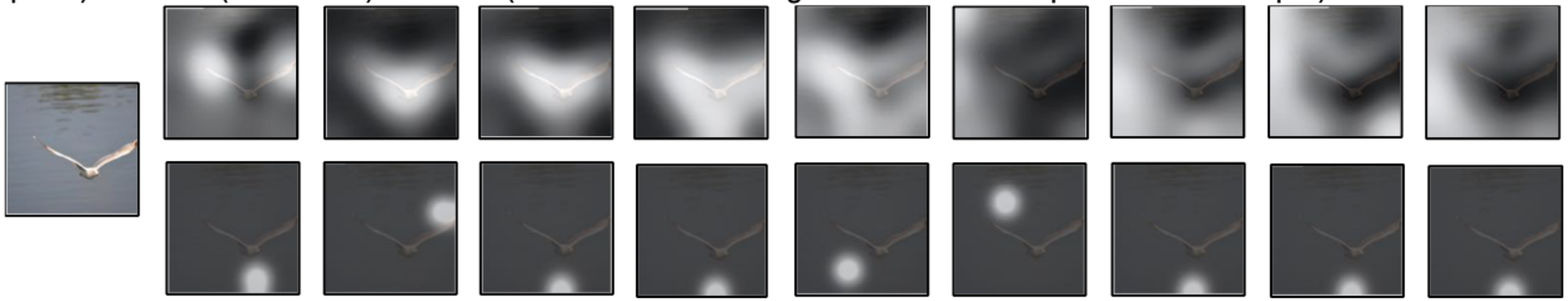
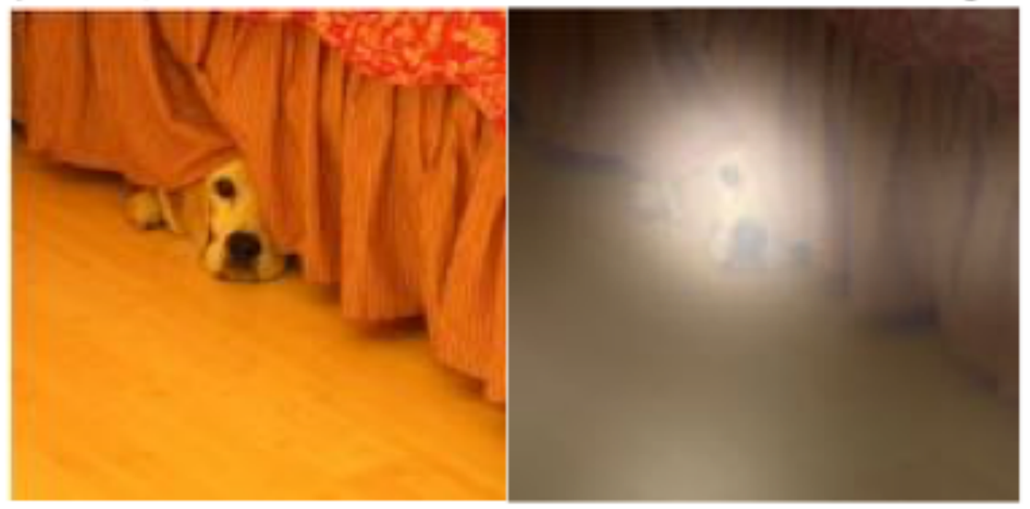


Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.

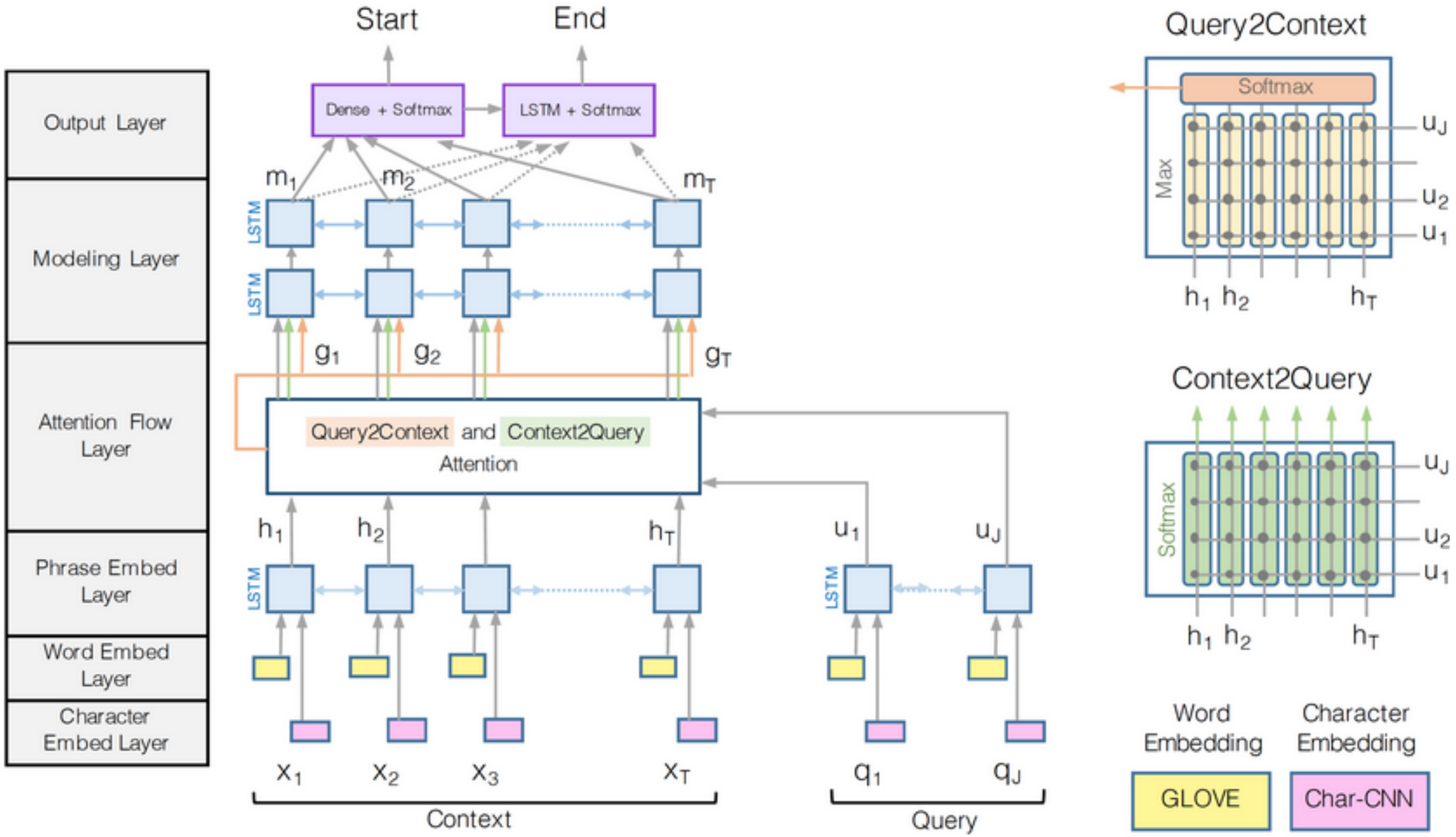
A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Attention for Reading Comprehension



Takeaways

- Seq2seq models are a very flexible framework
- Vanilla version of seq2seq model has shortcomings, such as handling rare words, repeating itself, bad at long sequences
- Attention mechanism is a powerful and general solution
  - Applied to many problems, including vision!