
WHEN PEOPLE USE SILLY MEMES
TO TRY TO GET YOUR ATTENTION
memegenerator.net

Lecture 16: Seq2Seq and Attention

Penn Engineering

Agenda

- Continuing CIS 522 at this point of time
- (Review) Seq2Seq
- Attention Models
  - Seq2Seq
  - Image Captioning

Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Questions about the course

Use the show hand option please

Breakout rooms

How are things going in life?

How are the projects going?

What can we do to make the remainder of the course be as useful as possible?

Part One: Seq2Seq

Penn Engineering

Problems we would like to solve

- Parse language
  - Understand the role of all the words
- Translate language
  - Incidentally, please someone solve this for computer languages
- Summarize text
  - See next slide
- Dialogue
  - Remember AIDungeon?

All these are **Seq2Seq** flavor

Typical automatic summary pre deep learning

Deep learning techniques now touch on data systems of all varieties. The purpose of this course is to deconstruct the hype by teaching deep learning theories, models, skills, and applications that are useful for applications. *Suppose a problem has variable-length input.* We will cover popular models, discuss how to find interesting niche research, and gain intuition for the best approaches. The success of a deep learning project often depends on the interpretation and presentation of results.

Setting

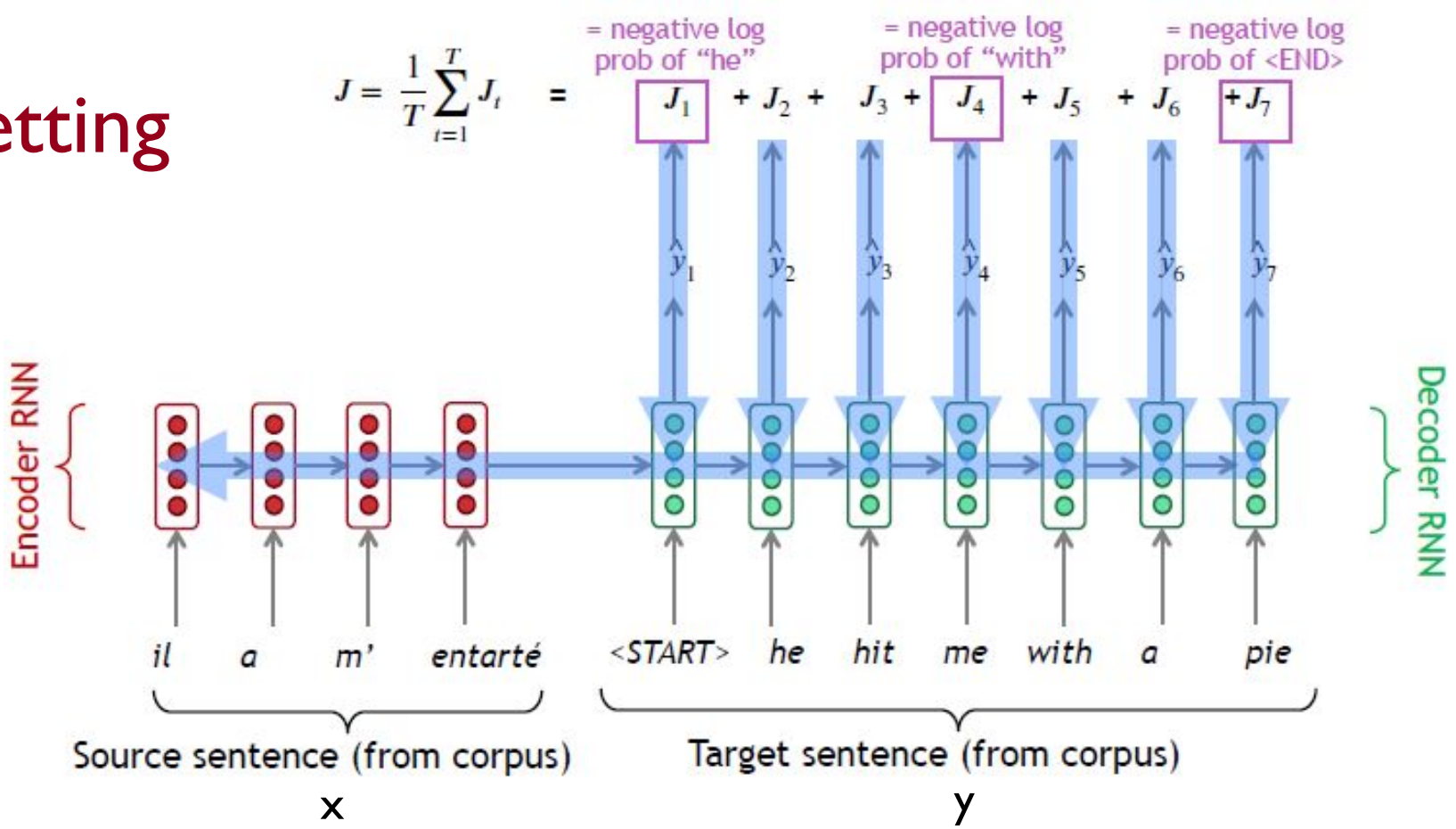


Diagram from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Which y is best?

- We will produce a sequence of length T
- This sequence y should be likely given x
- The sequence y itself should be probable (e.g. good sentence)
- We want to evaluate the probability of y (p(y)) but we can only readily evaluate p(y_i)

$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

Logic adapted from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Perfect algorithm

Enumerate all possible output token sequences.

Choose the one that is most probable.

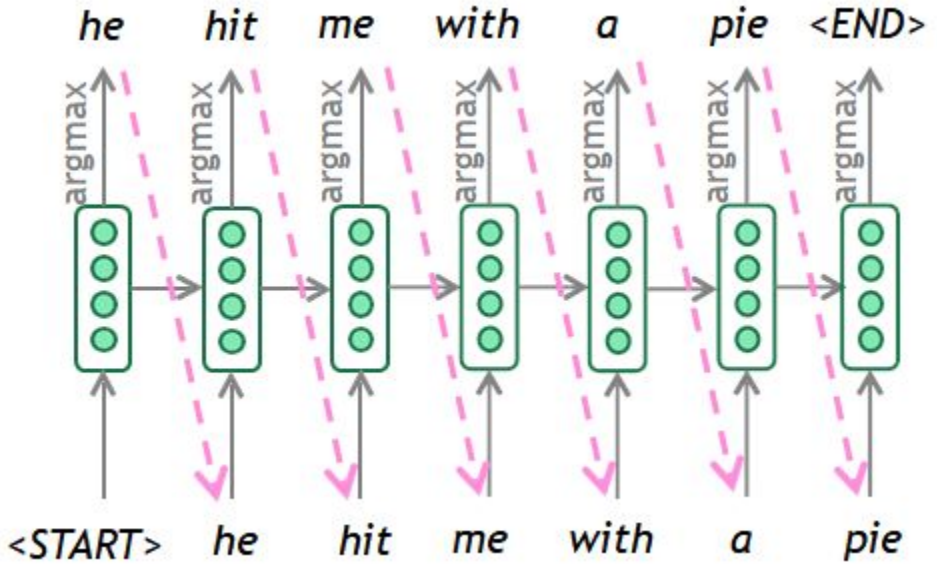
Why is that a bad idea?

Let us zoom summarize (1 word at a time)

Not every problem should be tackled with deep learning. It's data-hungry, catastrophically forgets, extrapolates poorly, cannot easily perform estimates of uncertainty, cannot easily transfer knowledge, and cannot be audited. Deep learning has been wildly successful in certain domains (CV, NLP, and RL come to mind), and students will understand how to determine what kinds of models are best for which problem settings.

Naive Decoder

- The "Greedy" decoder takes the most probable outcome and returns it
- It then uses this outcome to infer the next word in the sequence
- What if it gets one of the earlier inferences wrong?



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search as an alternative

- Beam Search: Keep track of the *k* most probable partial translations (hypotheses) at any given point in time (k from roughly 5 to 10, for example)

$$\text{score}(y_1, \ldots, y_t) = \log P_{\text{LM}}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$$

  - Scores are all negative, and higher score is better
  - We search for high-scoring hypotheses, tracking top k on each step
- Not guaranteed to find the optimal sequence, but a good start!

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

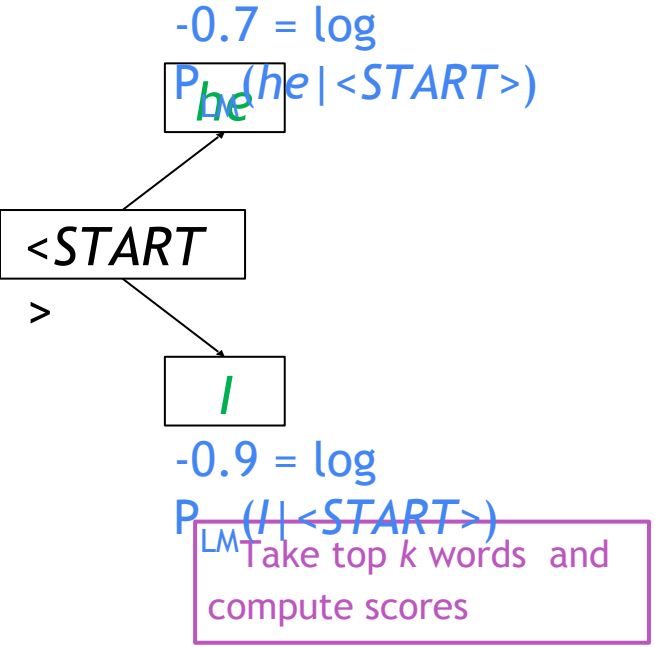
<START>

Calculate prob dist of next word

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

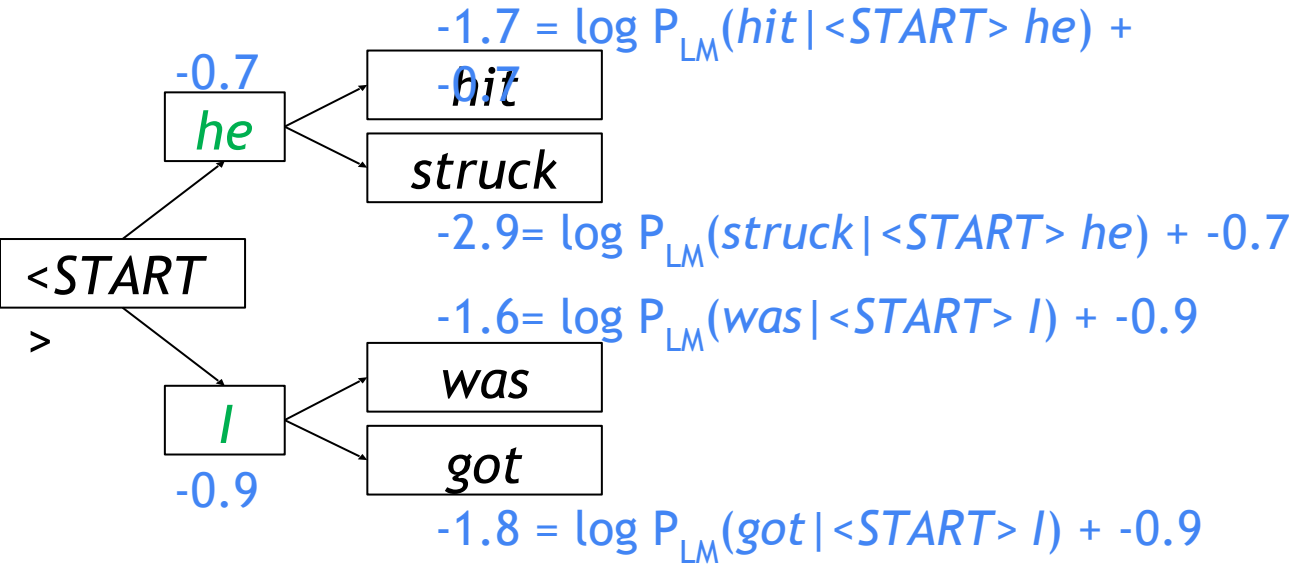
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

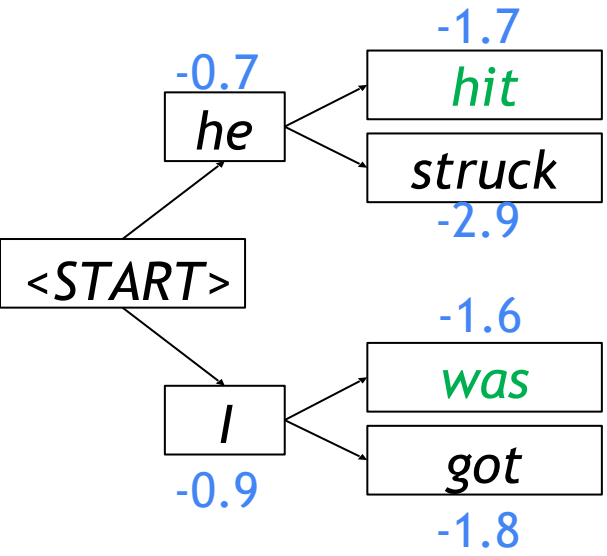
Beam size = k = 2. Blue numbers =



For each of the *k* hypotheses, find top *k* next words and calculate scores

Beam search decoding: example

Beam size = k = 2. Blue numbers =

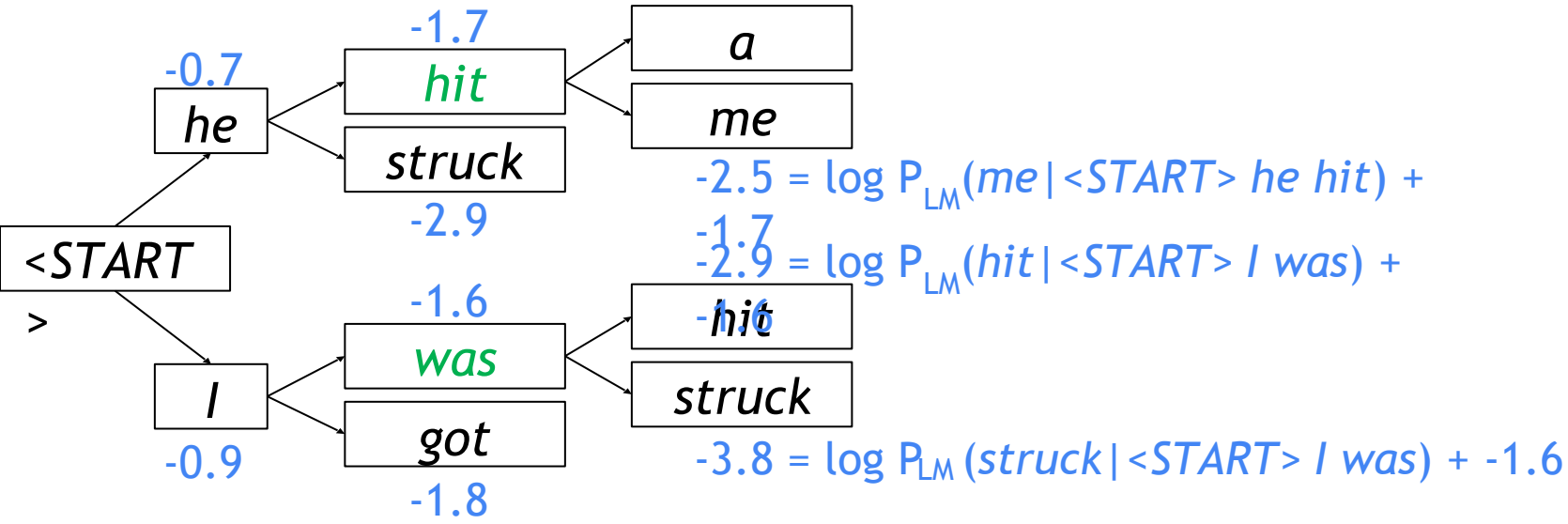


Of these k^2 hypotheses, just keep k with highest scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

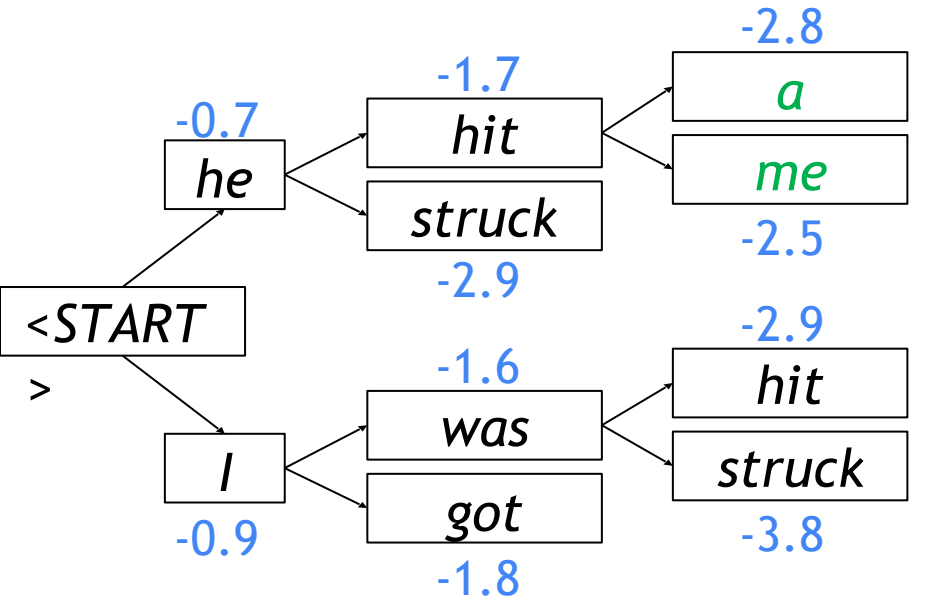


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

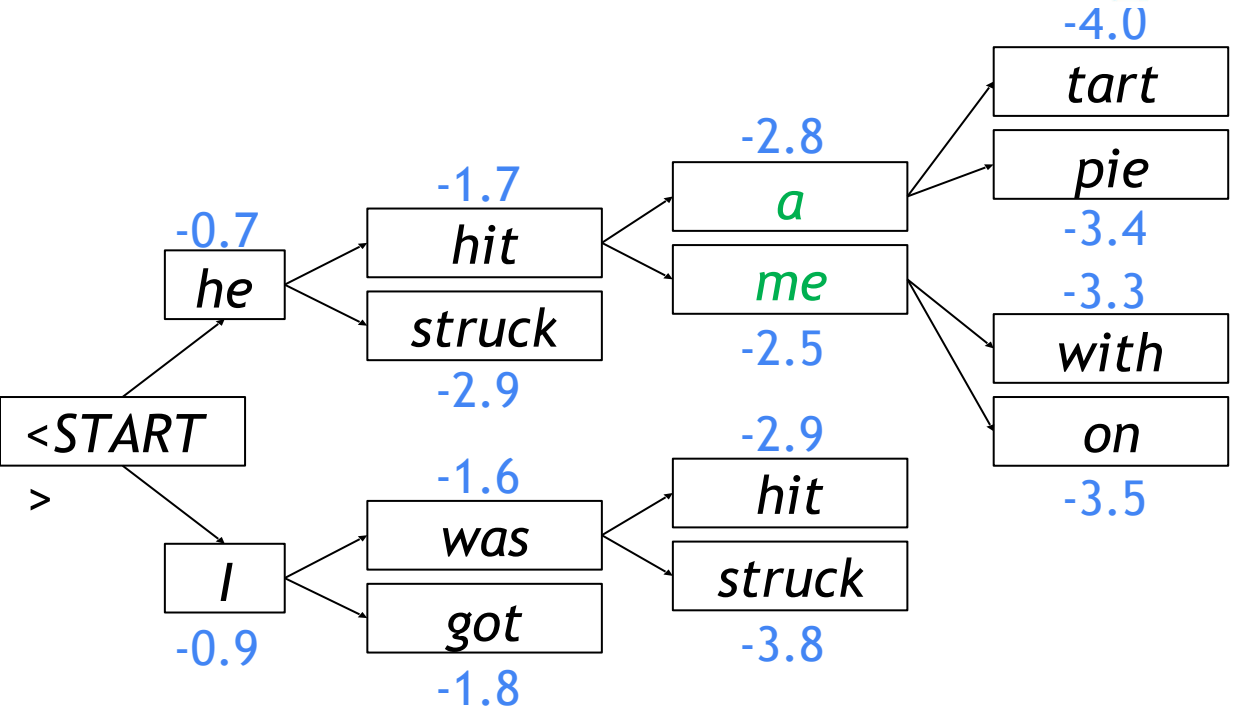


Of these k^2 hypotheses, just keep k with highest scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

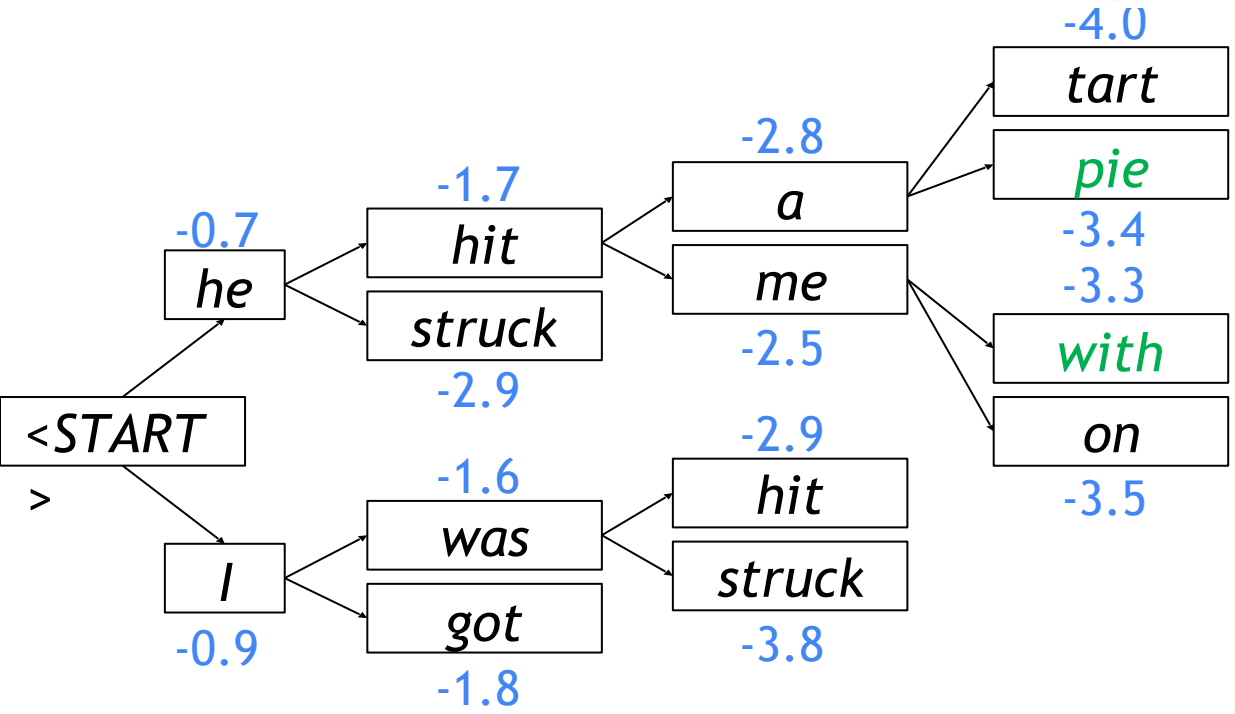


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

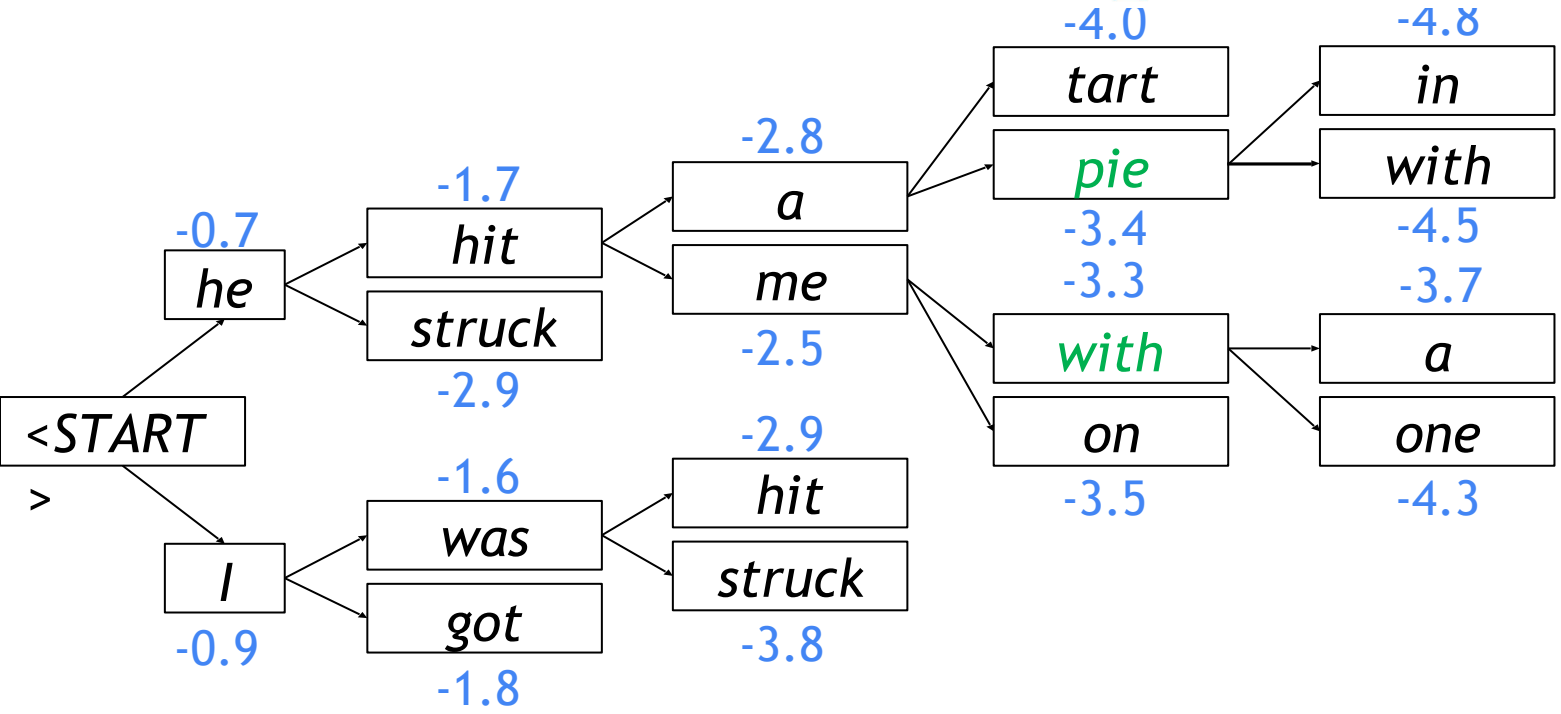
Beam size = k = 2. Blue numbers = $\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} \log P_{\text{LM}}(y_i | y_1, \ldots, y_{i-1}, x)$



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

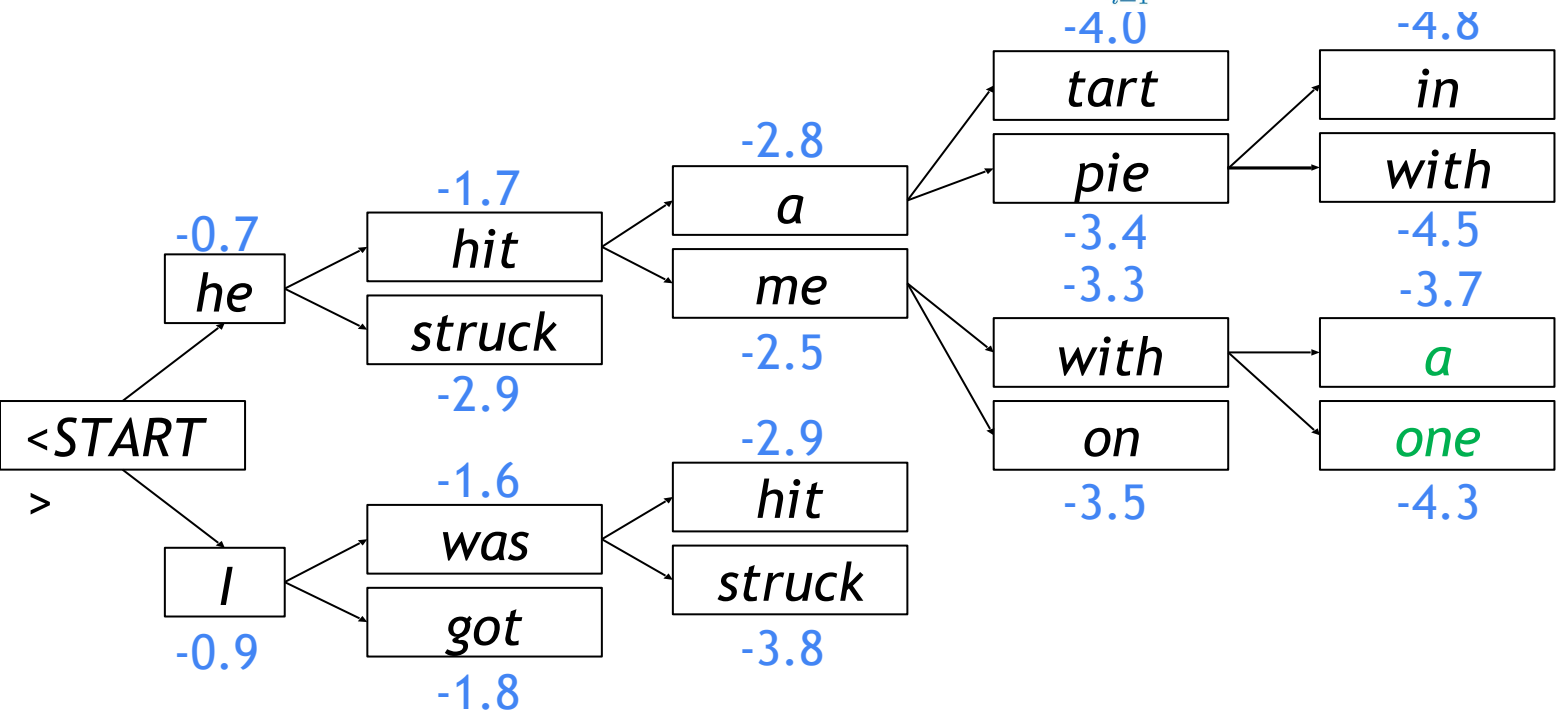


For each of the *k* hypotheses, find top *k* next words and calculate scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

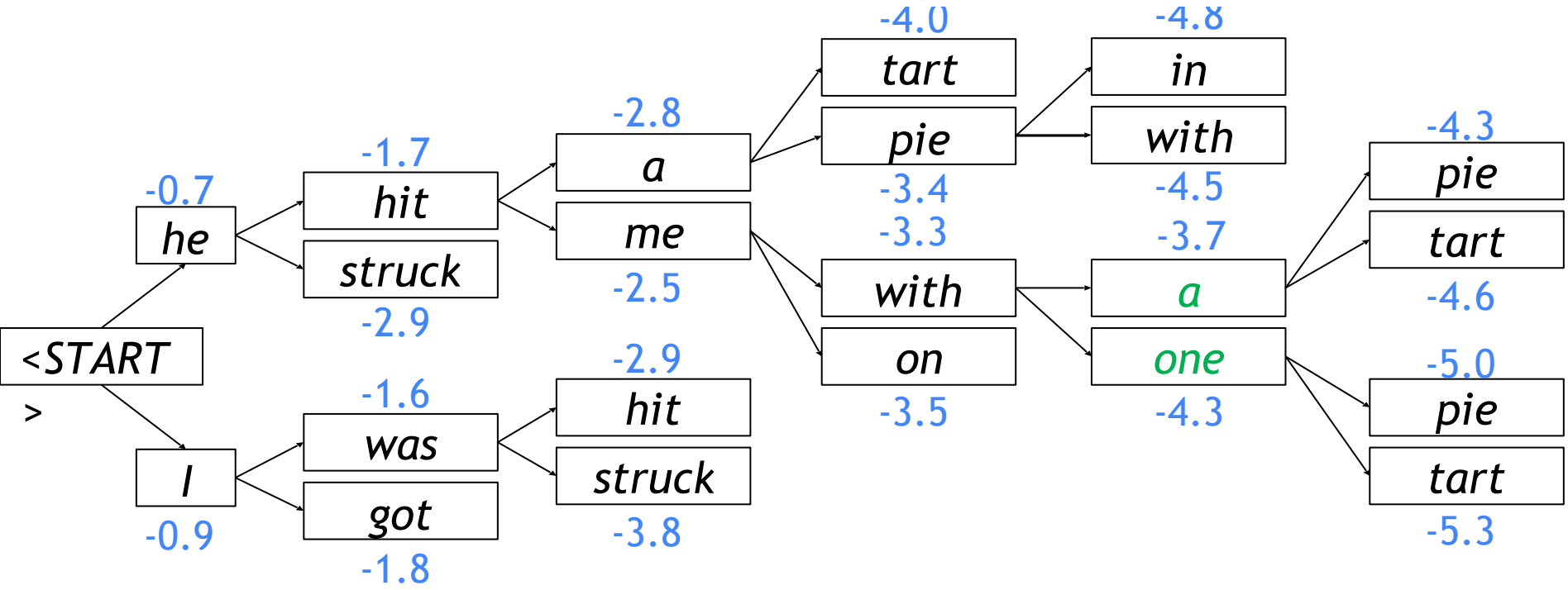


Of these k^2 hypotheses, just keep k with highest scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

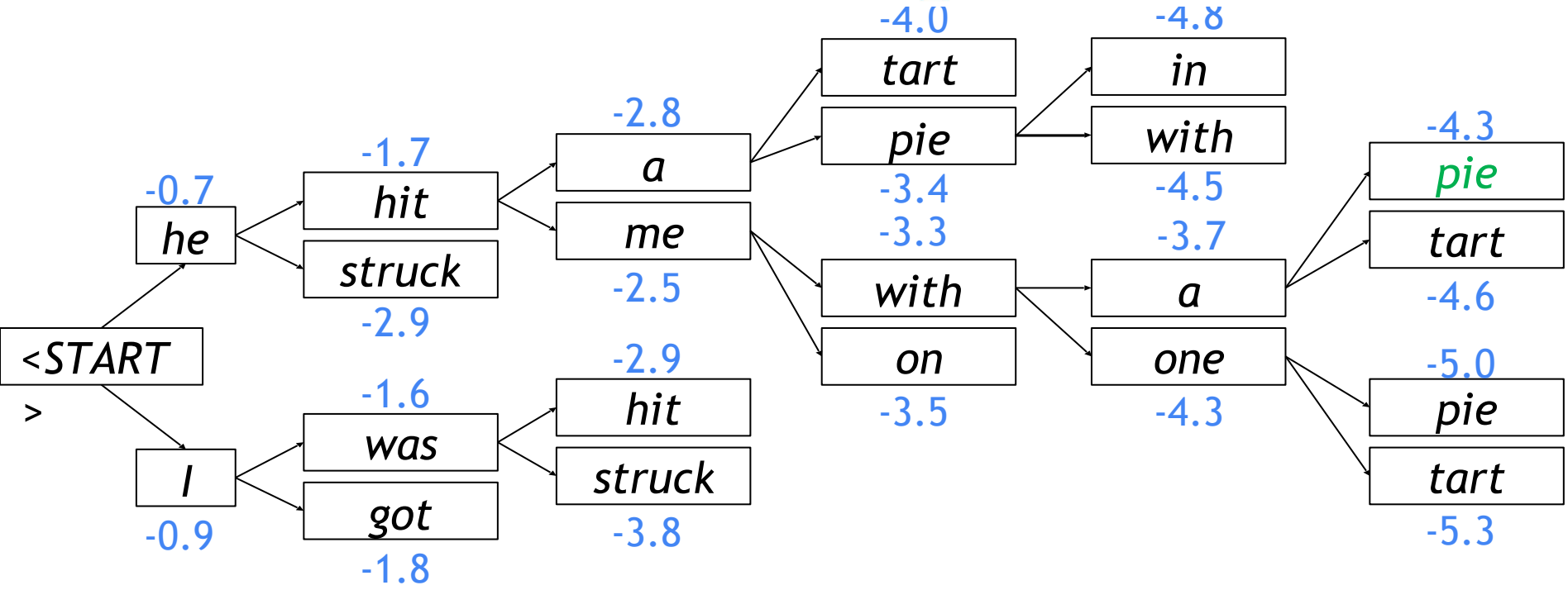


Of these k^2 hypotheses, just keep k with highest scores

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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

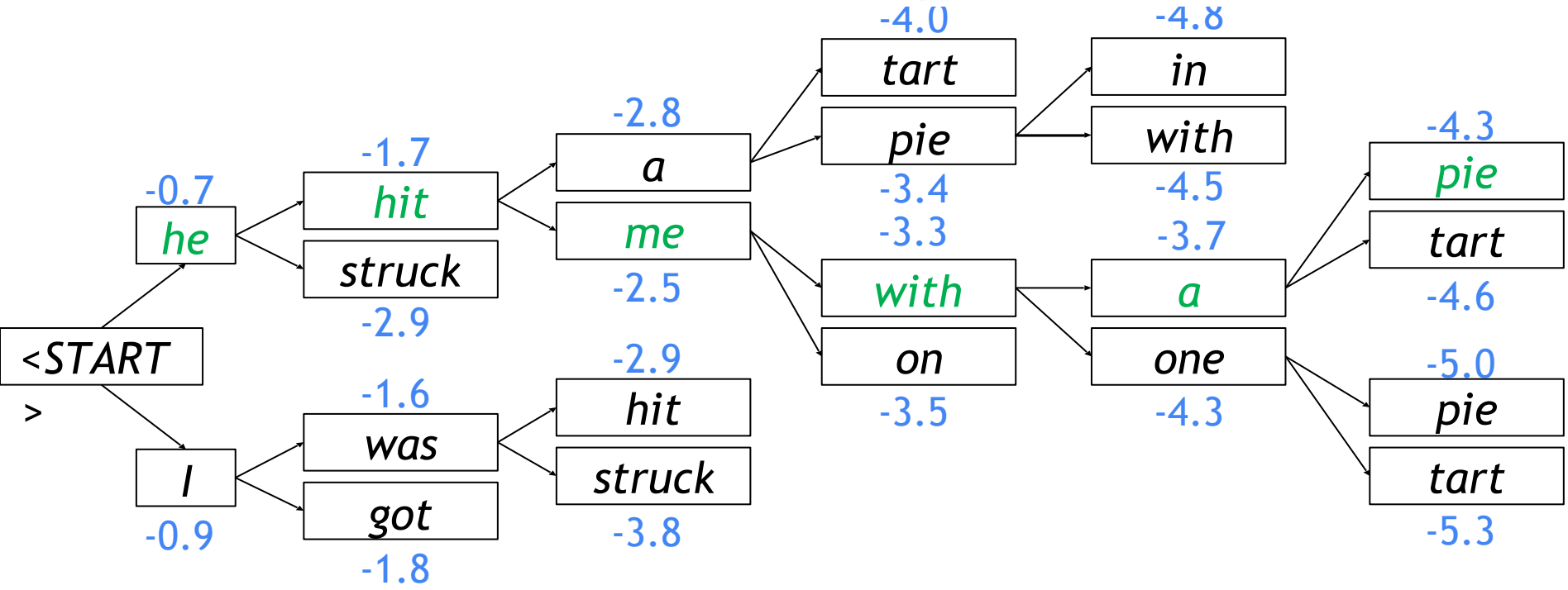


This is the top-scoring hypothesis!

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Beam search decoding: example

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



Backtrack to obtain full hypothesis

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Stopping Criterion: Greedy vs. Beam

- In greedy decoding, usually we decode until the model produces a <END> token (for example: <START> he hit me with a pie <END>)
- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete. Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until: we reach timestep T (where T is some pre-defined cutoff), or we have at least n completed hypotheses (where n is pre-defined cutoff)

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Final Touches: Beam Search

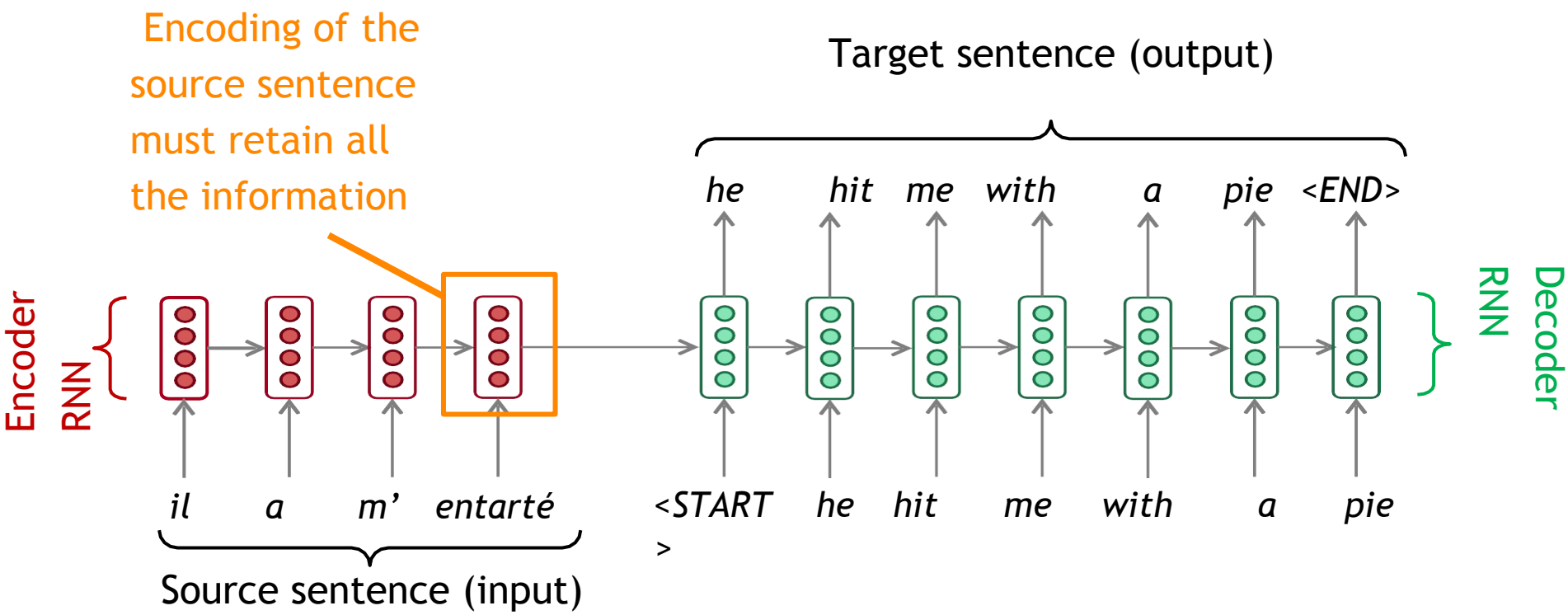
- Our scores for each sequence may be biased towards shorter-length sequences (since scores for longer sequences are lower)
- We must normalize by the length of the sequence:

$$\frac{1}{t}\sum_{i=1}^{t}\log P_{\mathrm{LM}}(y_i|y_1,\ldots,y_{i-1},x)$$

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Part Two: Attention

Sequence-to-sequence: the bottleneck problem

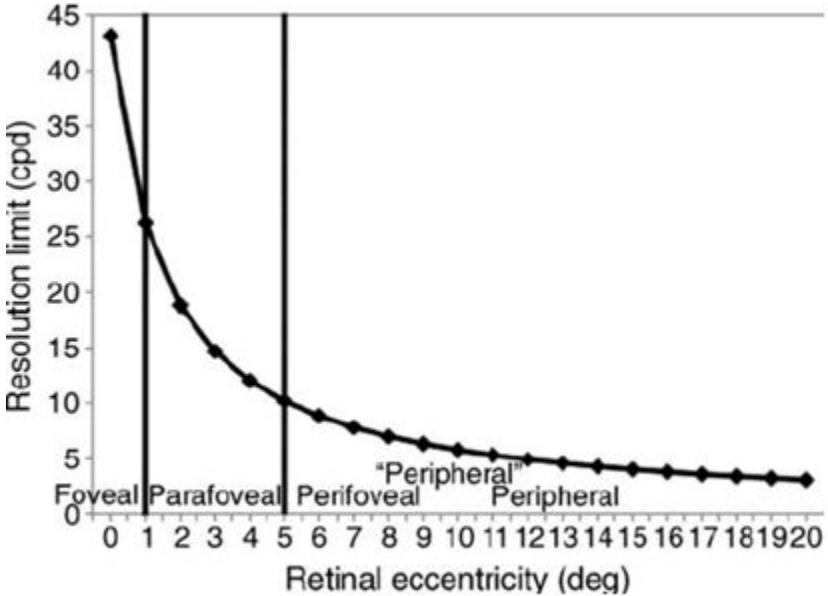


Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Maybe Attention might help?

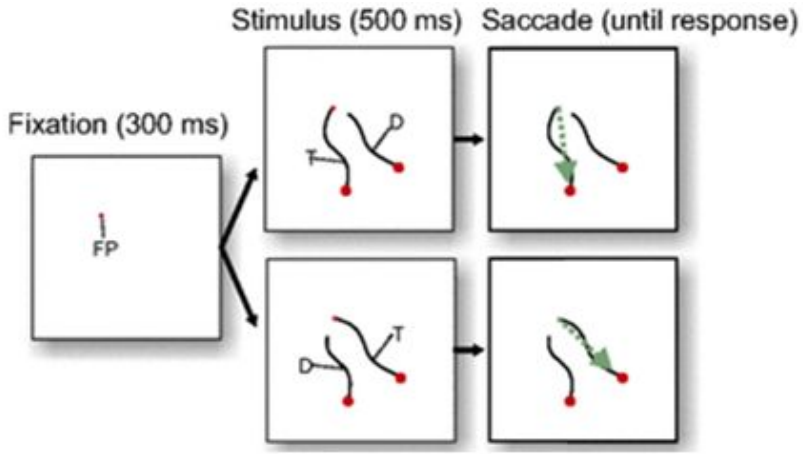
- We want to be able to focus on different parts of the sequence as we decode it
  - Different contextual cues in our input sequence can imply different outputs (e.g. “Let’s table this discussion” vs “I put my cup on the table”)
- But first, is it possible to draw inspiration from neuroscience?

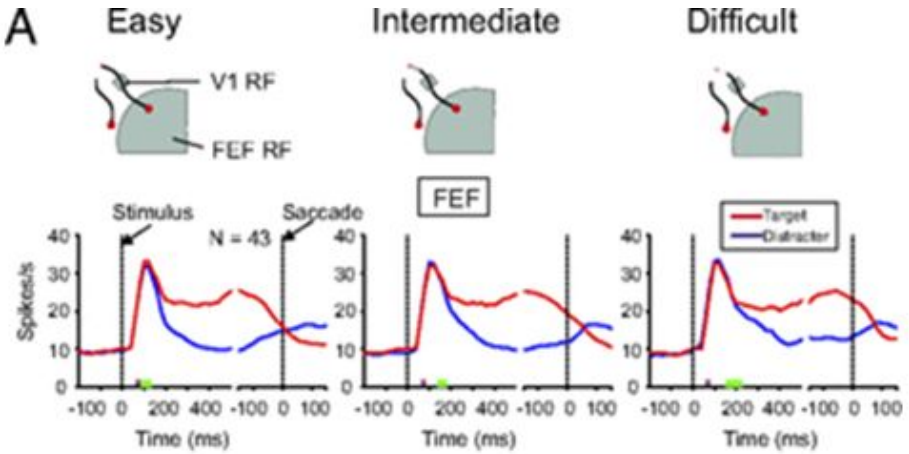
Overt Attention



Covert Attention

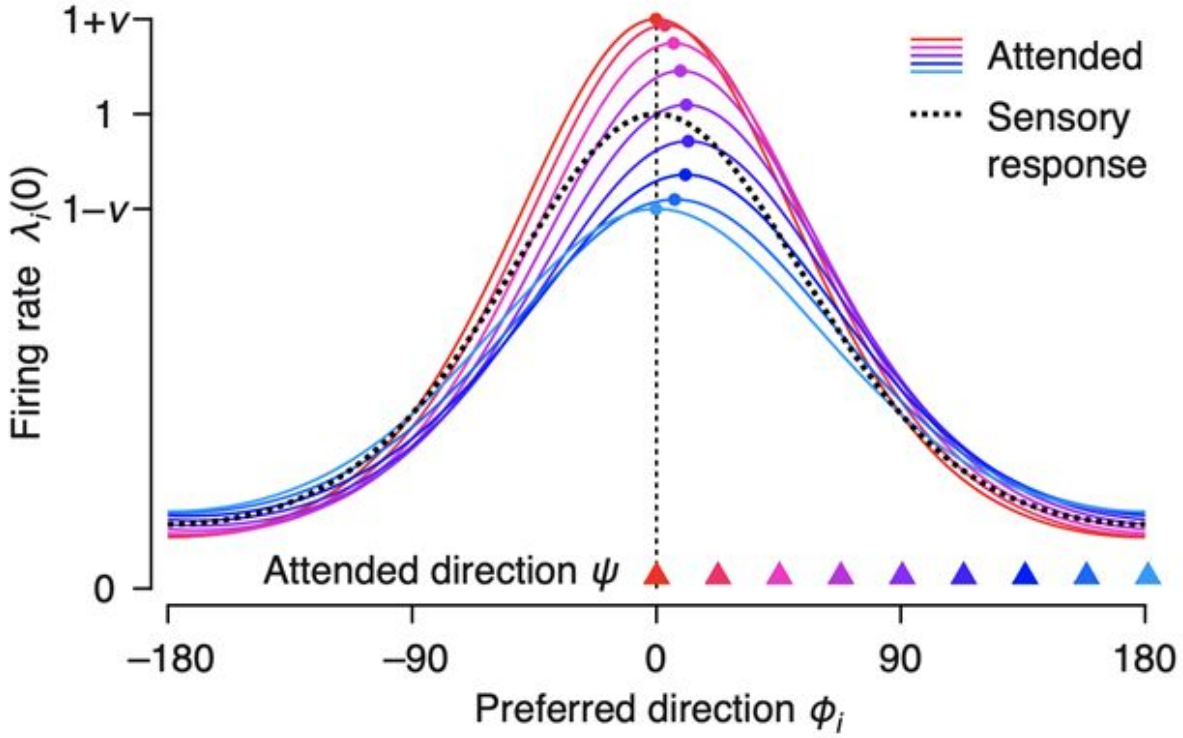
Object-based Attention





Pooresmaeili, et al 2014

Spotlight

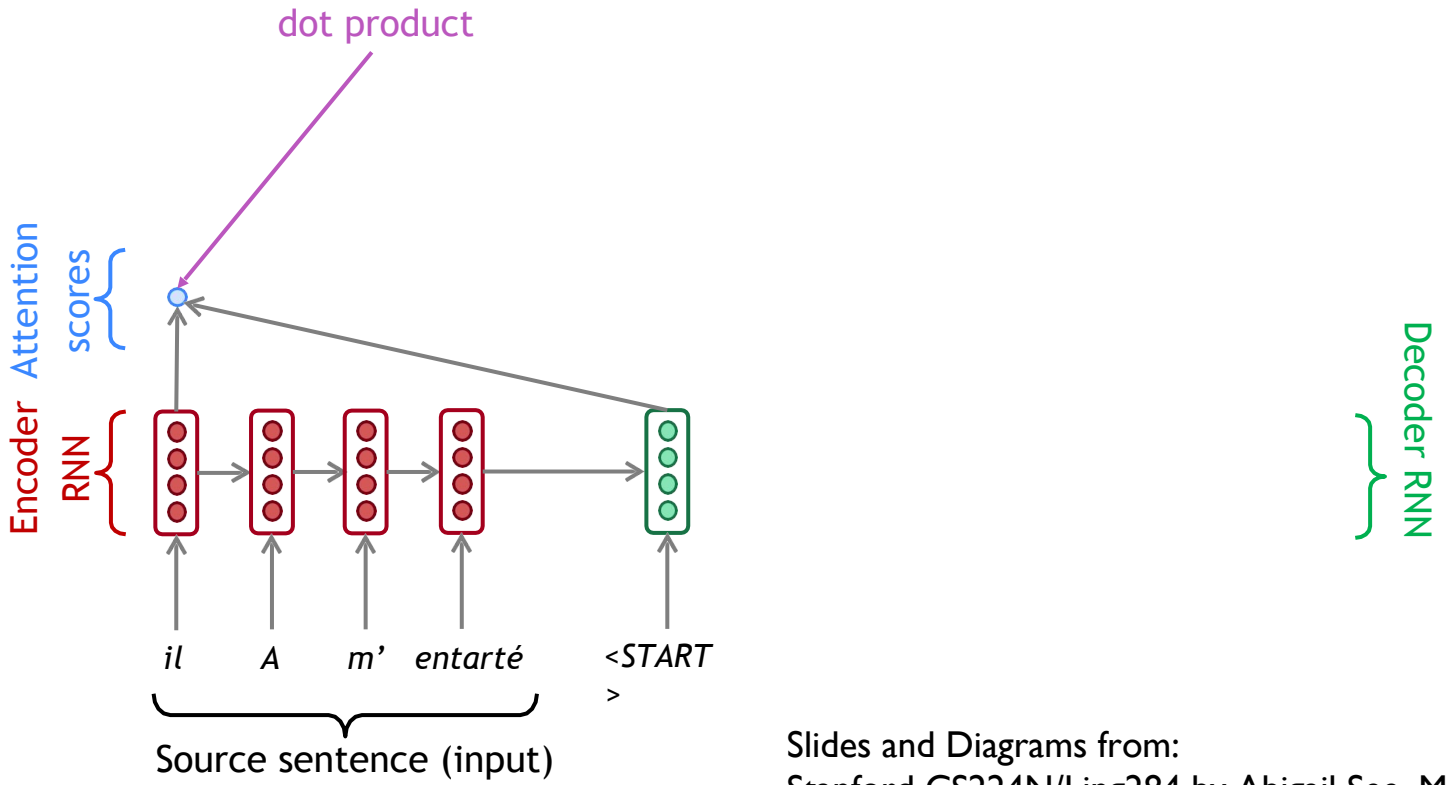


Eckert, et al 2015

Attention in Deep Learning

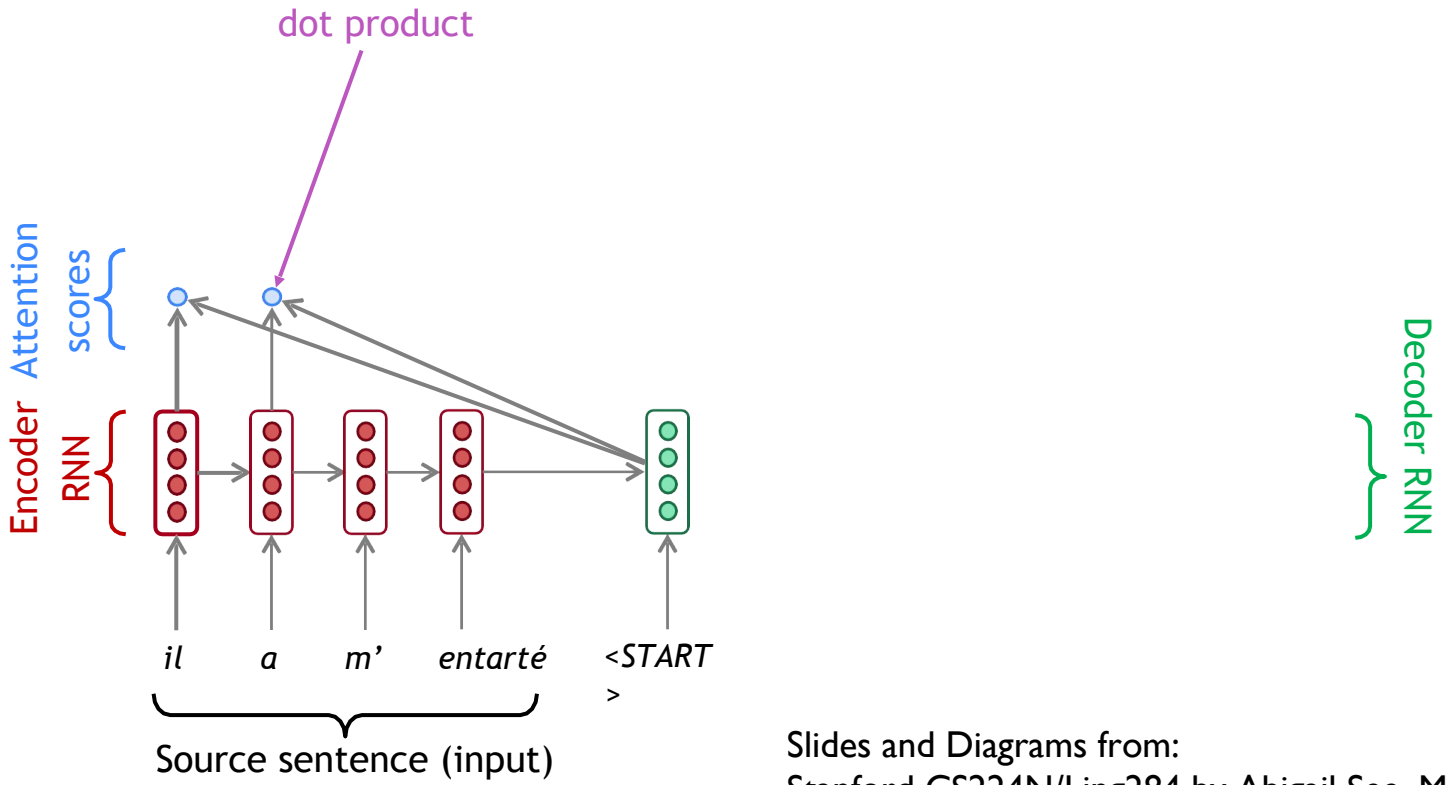
- **Big Idea:** we can use some weighted sum of our encoder hidden states to inform us about our current prediction in the decoding phase
  - N → One and One → M is too constricting - we need a "bridge" of information between N and M
  - We can give the network the ability to selectively focus on different parts of the input
  - This solves the bottleneck problem because we have a direct connection to our encoding phase

Sequence-to-sequence with attention



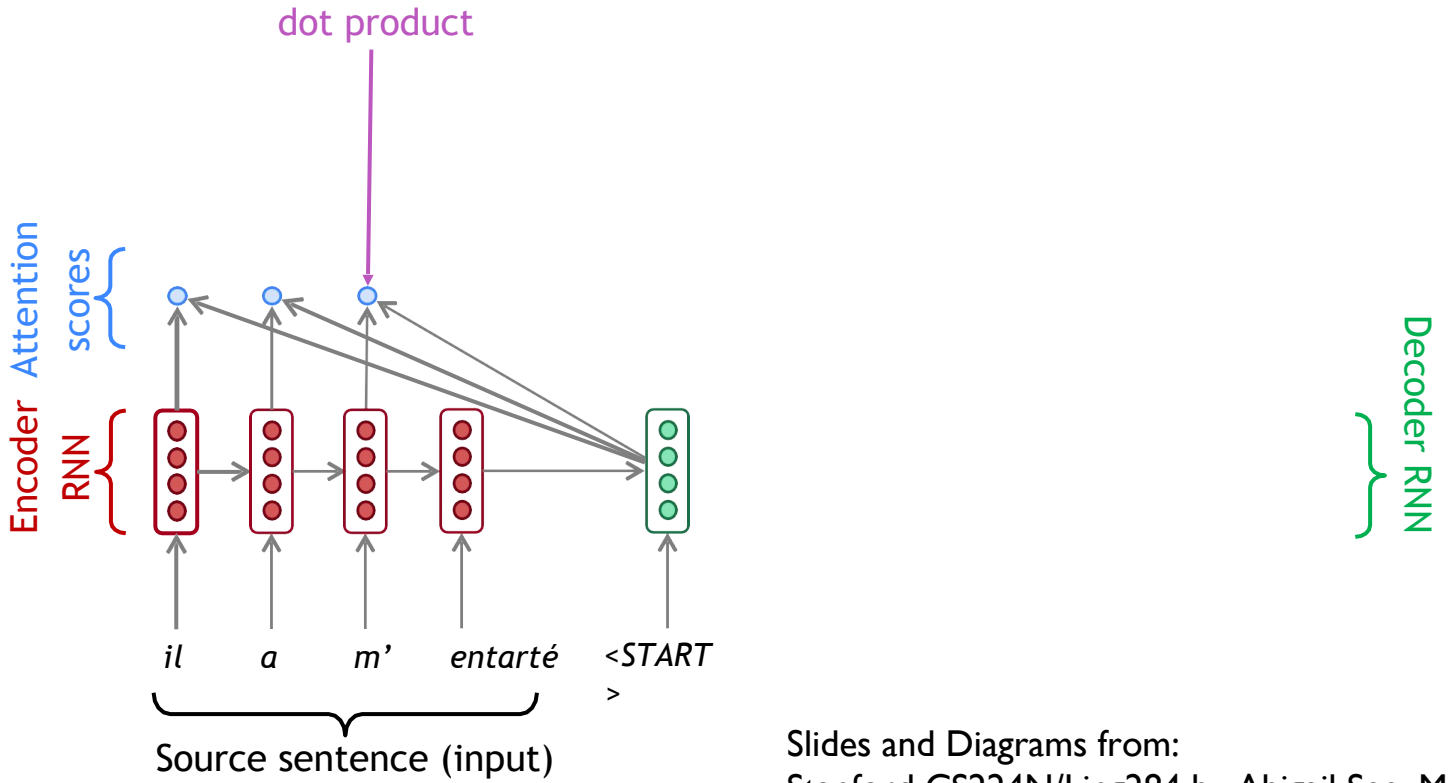
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



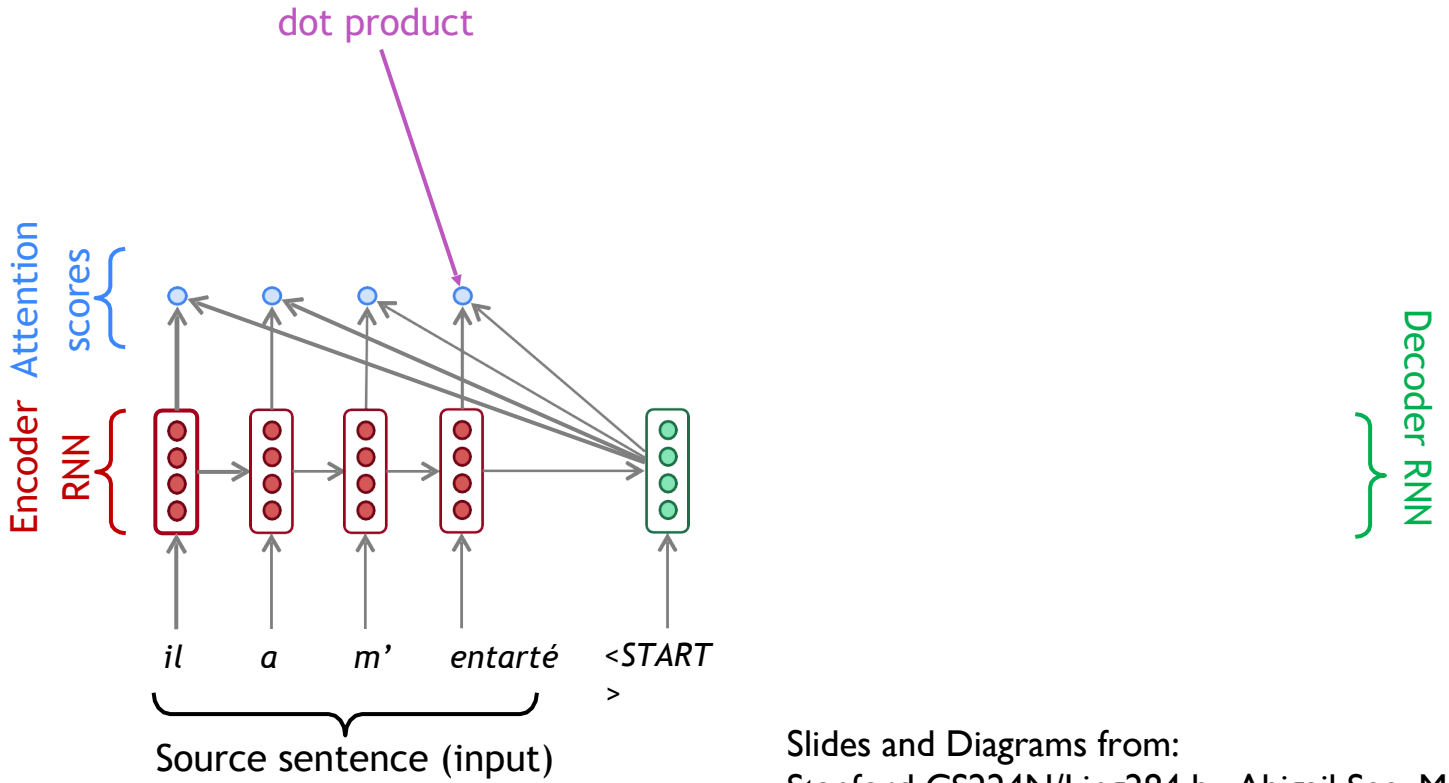
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



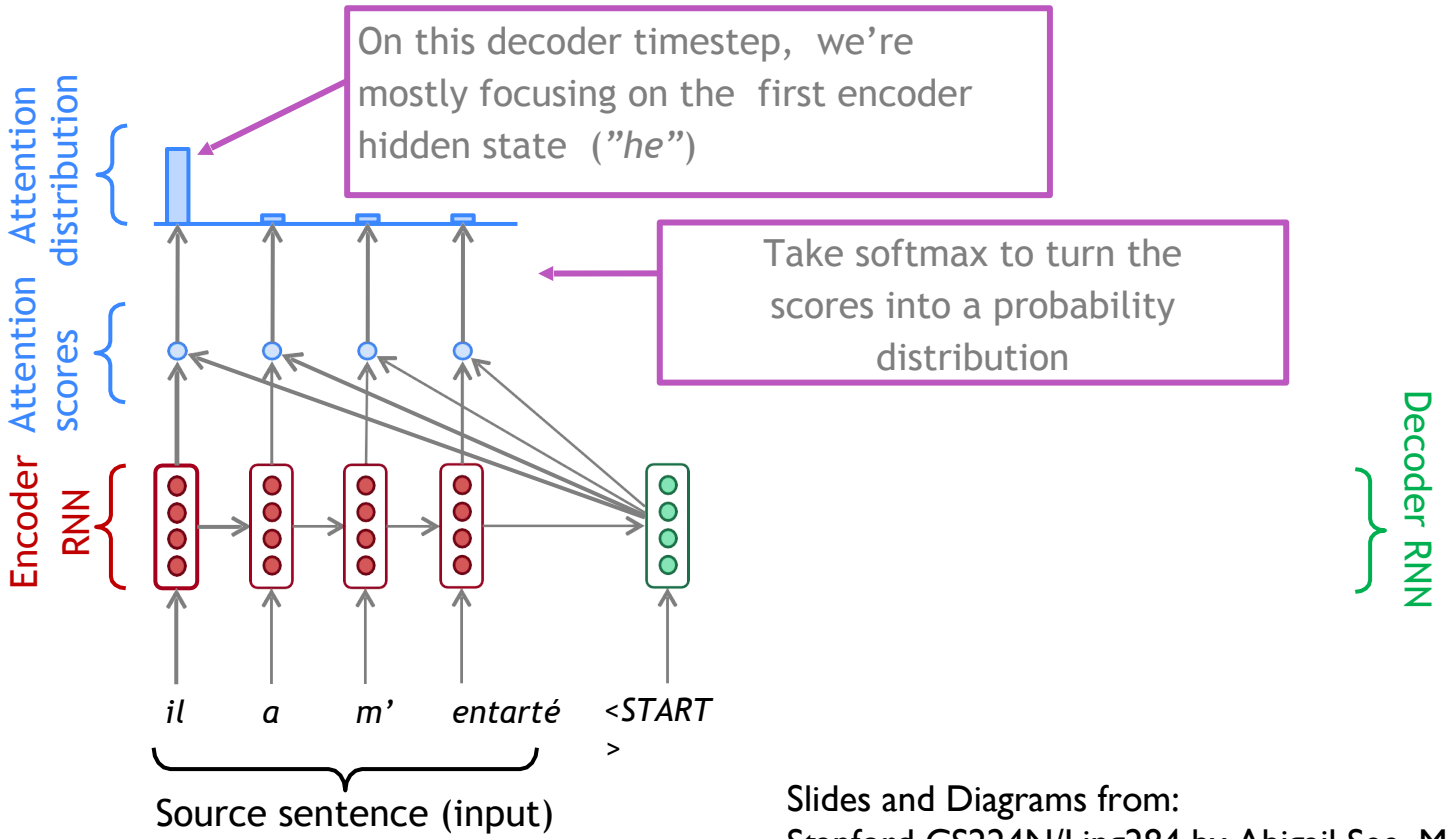
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



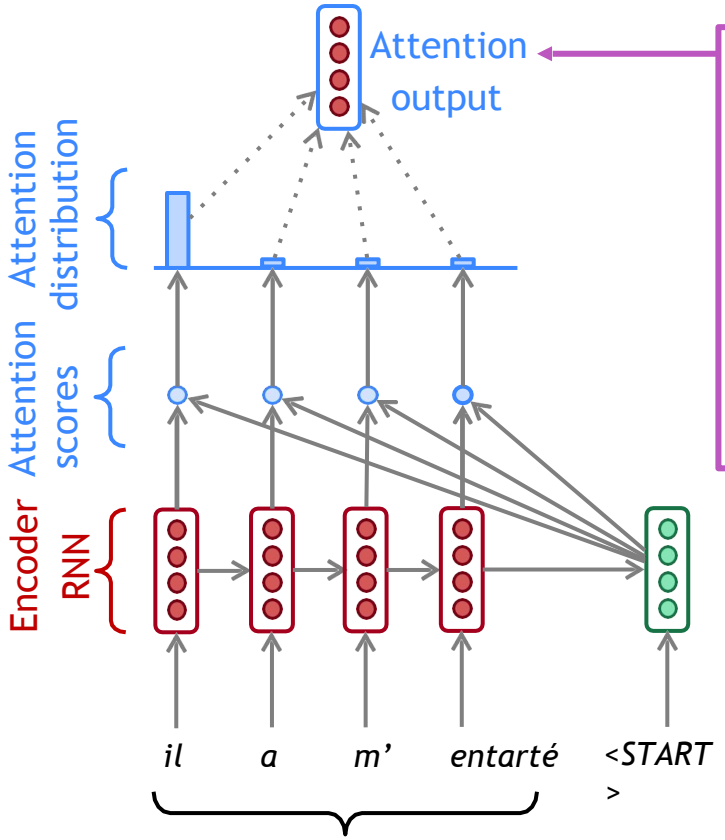
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



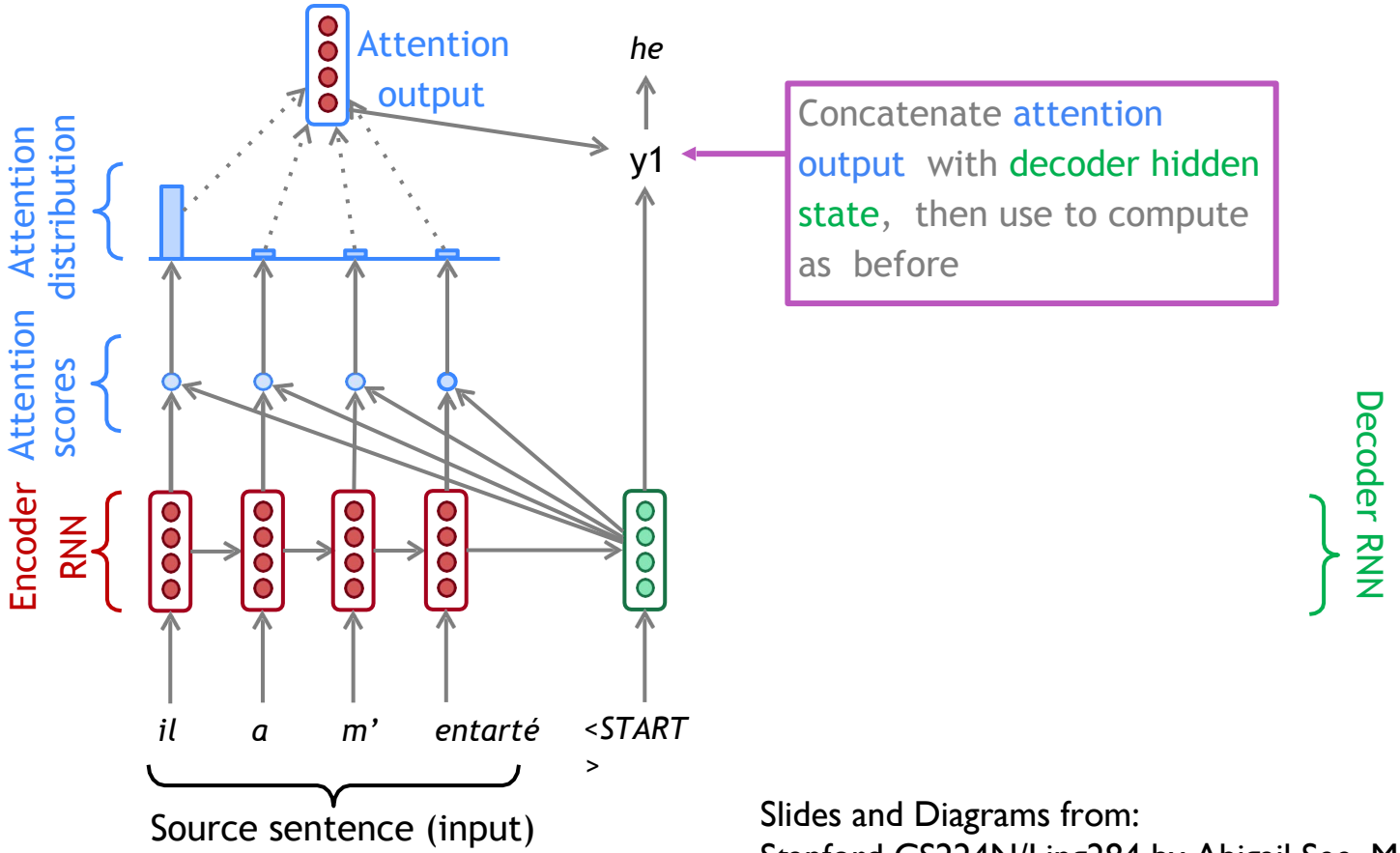
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

Decoder RNN

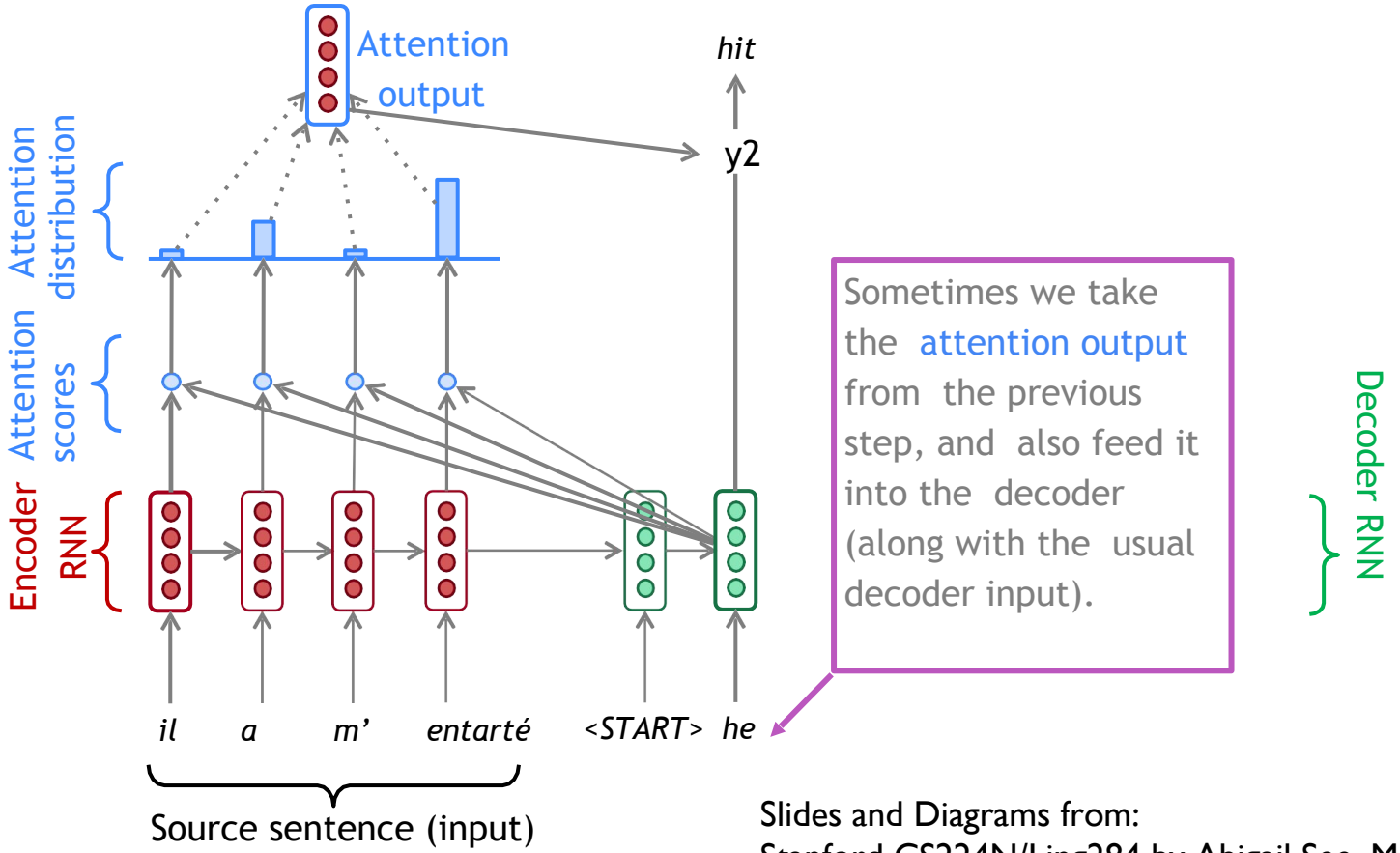
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



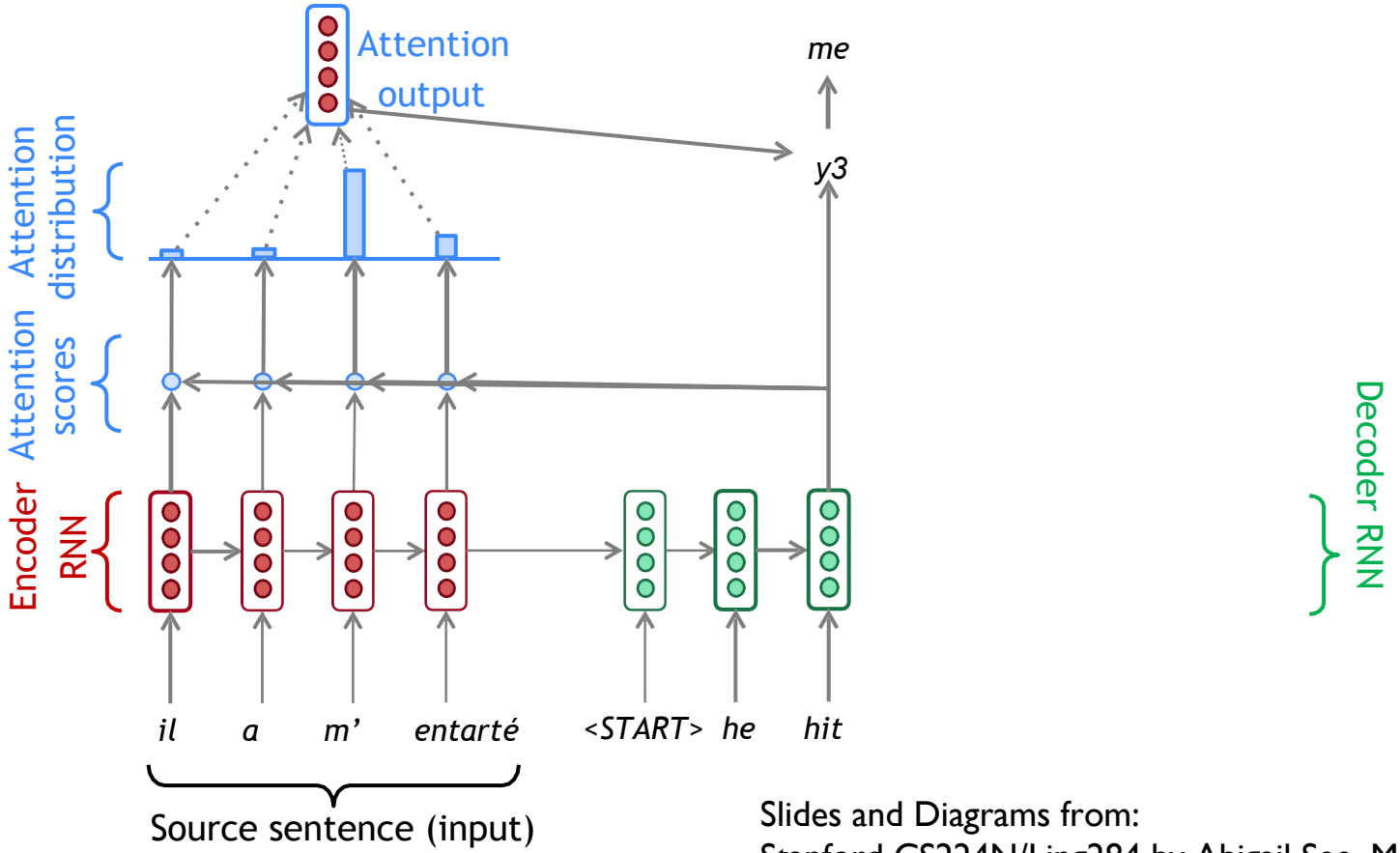
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



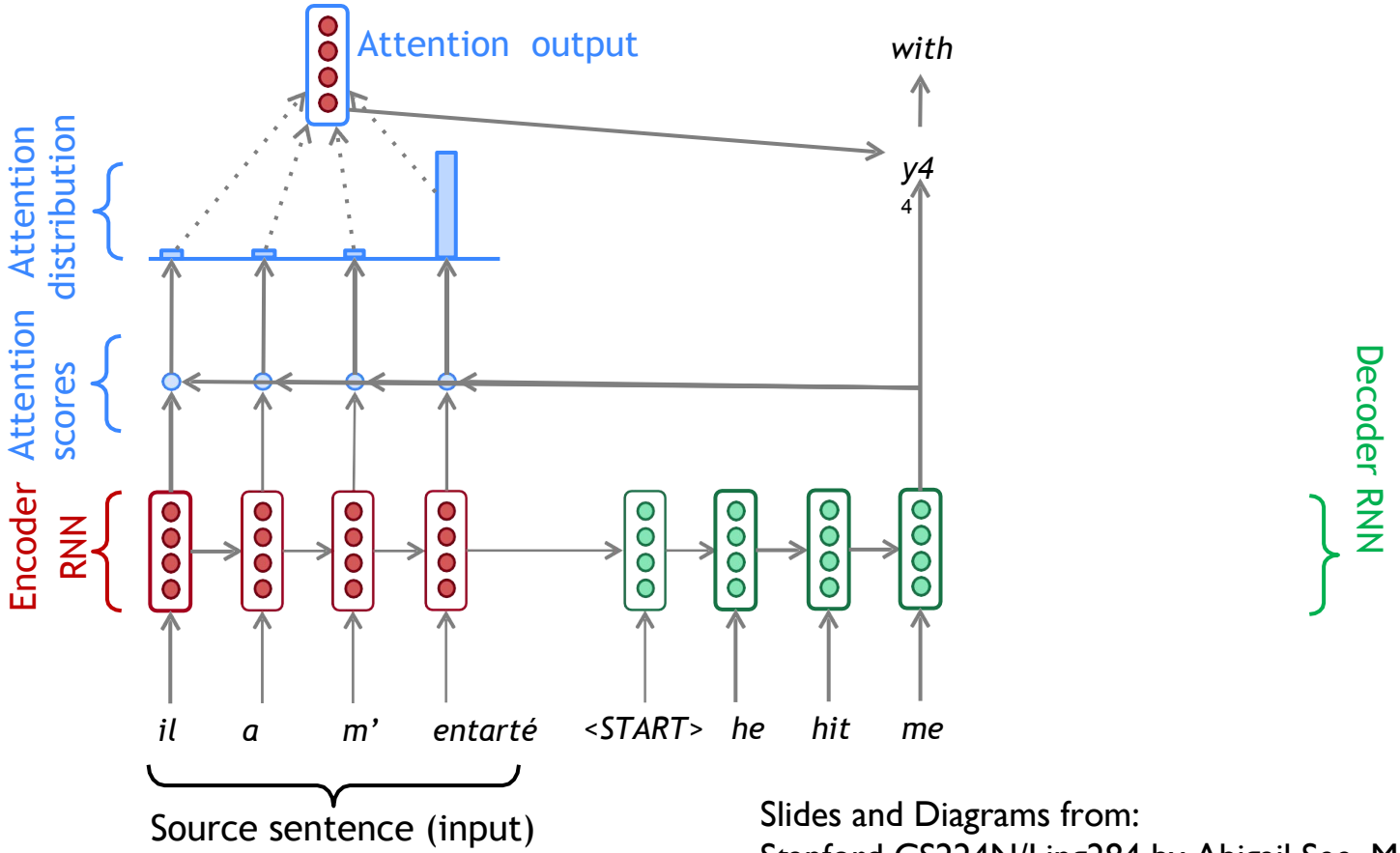
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



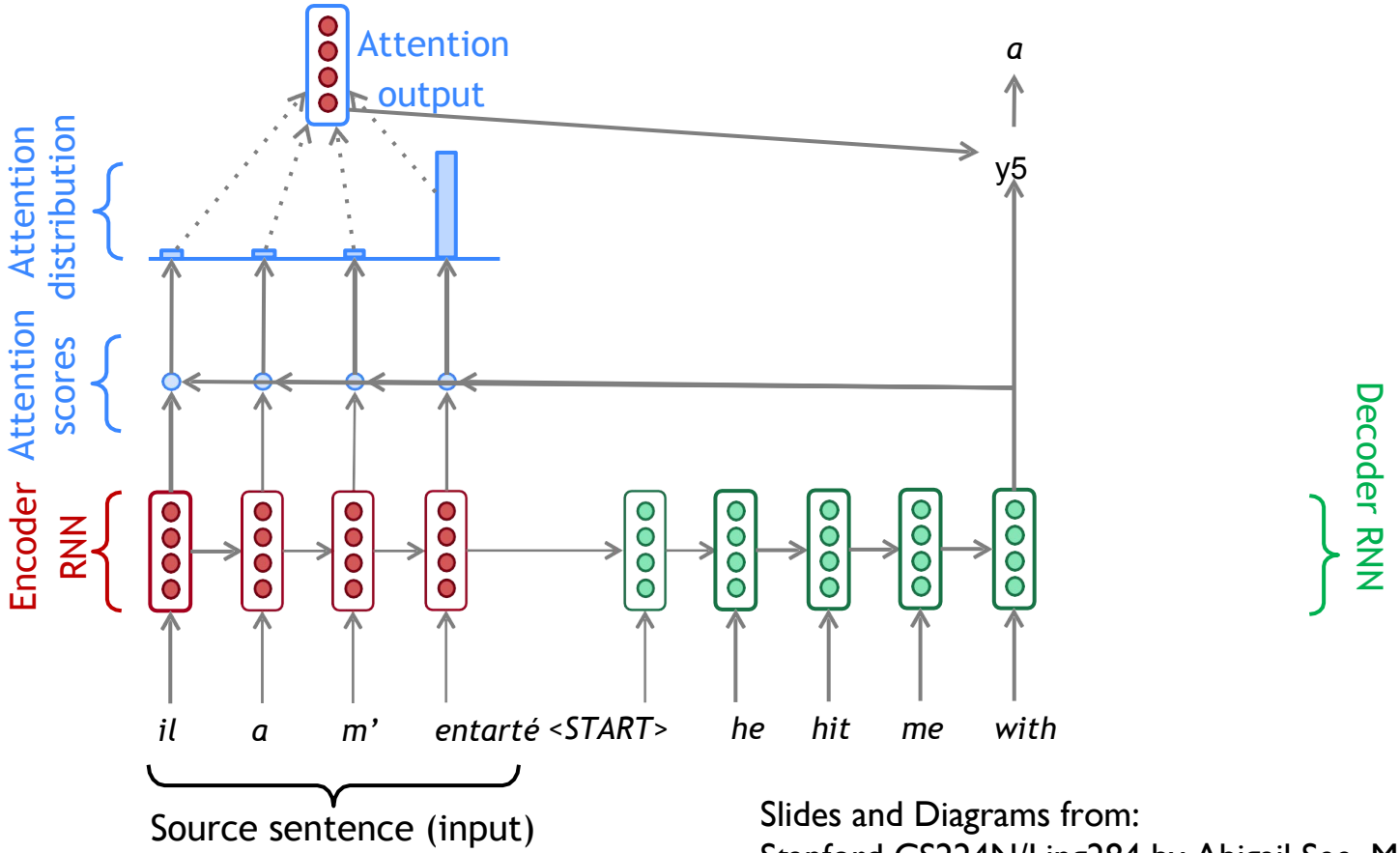
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



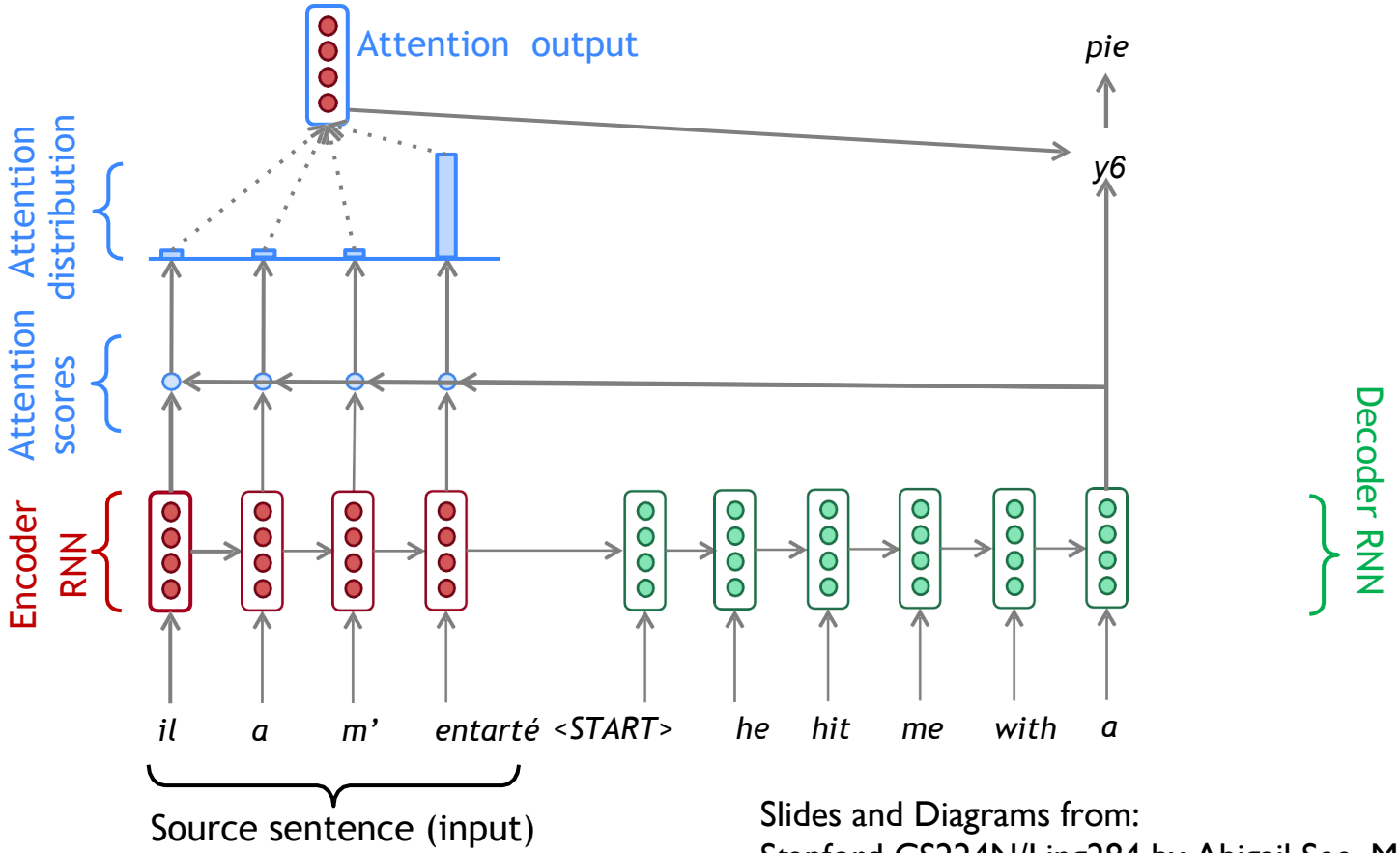
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Sequence-to-sequence with attention



Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

The Mathematics Behind Attention (1)

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep t, we have some decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We'll take a softmax of these scores to get a probability distribution

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

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

The Mathematics Behind Attention (2)

- We use this attention distribution to generate a weighted sum of encoder hidden states

$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- We concatenate the output of the attention with the decoder hidden state and do the same operations as in the non-attention model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Benefits of Attention

- Improves Performance - allows decoder to focus on certain parts of the source input
- Solves the bottleneck problem - allows decoder to look directly at the source, bypassing bottleneck
- Helps with vanishing gradients - similar to skip connections
- Provides interpretability
  - We can see what the decoder is focusing on
  - We get (soft) alignment between terms for free

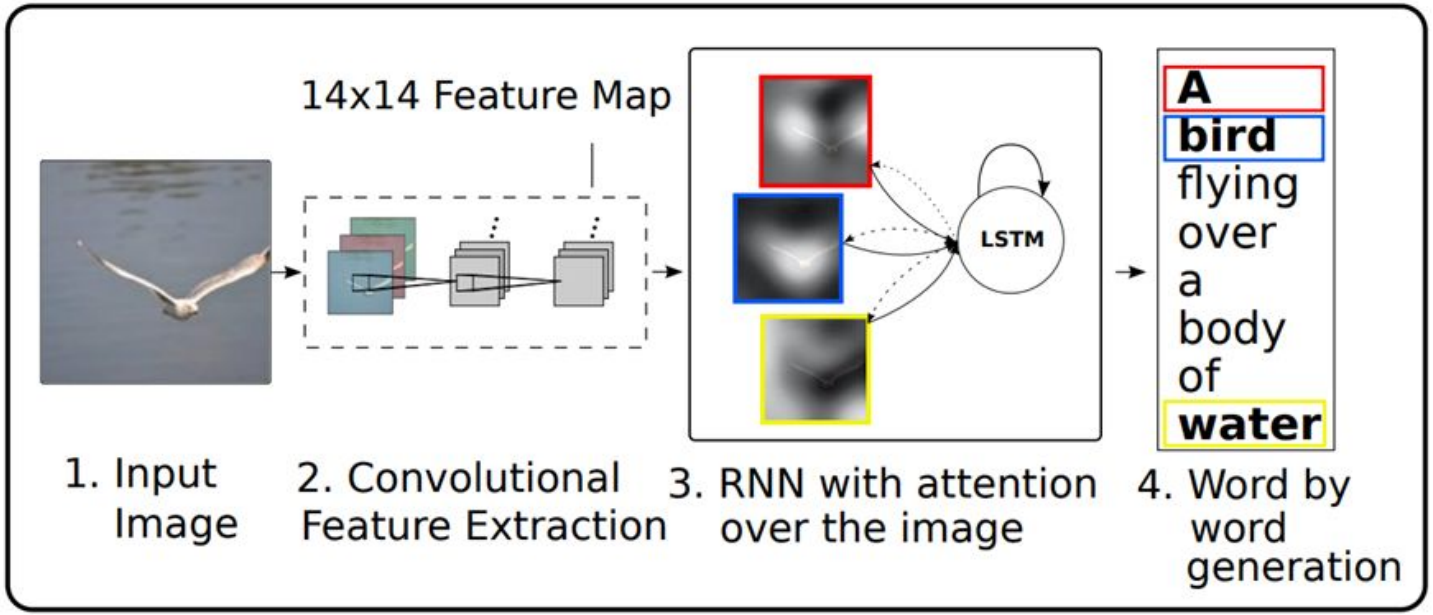
Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

Attention, Generalized

- More broadly, attention is defined in the following way:
  - Given a set of **vector values,** and a **vector query**, attention is a technique to **compute a weighted sum of the values, dependent on the query.**
- There are several variants on attention, many of which involve changing how the attention weights are computed (dot-product, multiplicative, additive, etc)

Slides and Diagrams from:
Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

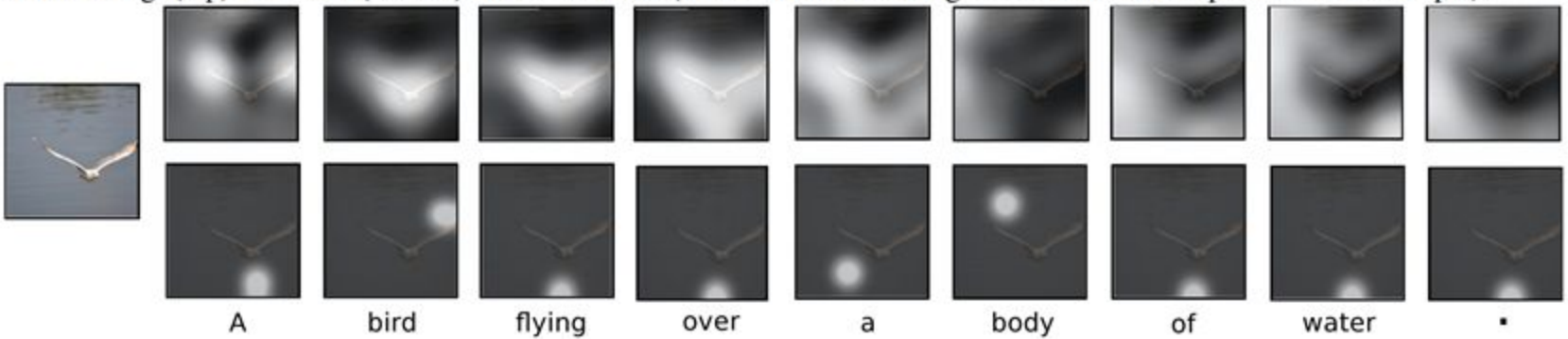
Show, Attend, Tell: Image Captioning



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015.

Show, Attend, Tell

Figure 3. Visualization of the attention for each generated word. The rough visualizations obtained by upsampling the attention weights and smoothing. (top)"soft" and (bottom) "hard" attention (note that both models generated the same captions in this example).



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015.

Show, Attend, Tell

Figure 4. 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.

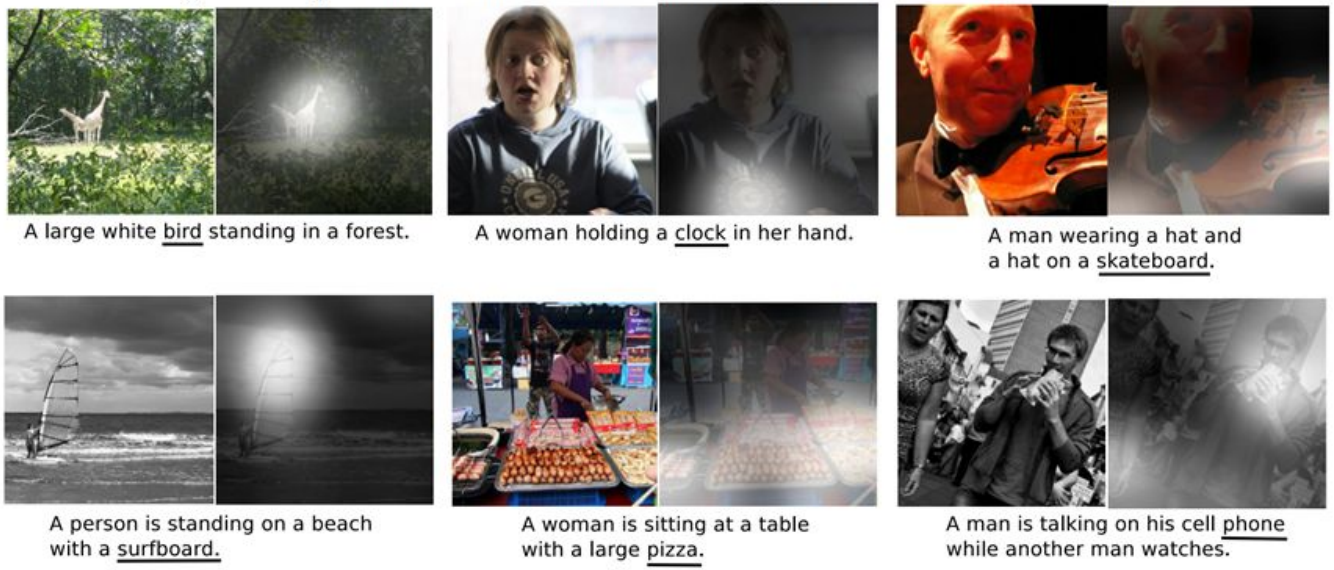
A group of people sitting on a boat in the water.

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

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015.

Show, Attend, Tell

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015.

What we learned today

- Clarity about how we are going forward
- Seq2Seq
- Attention Models
  - Seq2Seq
  - Image Captioning

Stanford CS224N/Ling284 by Abigail See, Matthew Lamm

How could the Seq2seq/ Attention lecture have been better?