Natural Language Processing

Neural semantic parsing

Slides adapted from Richard Socher, Chris Manning, Ofir Press

Plan

- Sequence to sequence models
- Attention
- Pointer networks
- Weak supervision

Sequence to sequence

Conditional generation

Input *X*	Output *Y* (**Text**)	Task
Structured Data	NL Description	NL Generation
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

Slide credit: Graham Neubig

Sequence to sequence

How tall is Lebron James?
HeightOf.LebronJames

[How, tall, is, Lebron, James, ?]
[HeightOf, ., LebronJames]

How tall is Lebron James?
¿Que tan alto es Lebron James?

High level

- We will build a differentiable function that maps a natural language statement *x* to a logical form *y* (or translation, or summary, or answer to a question…)

- We will define a loss (often cross entropy) that tells us how good is our prediction w.r.t the ground truth

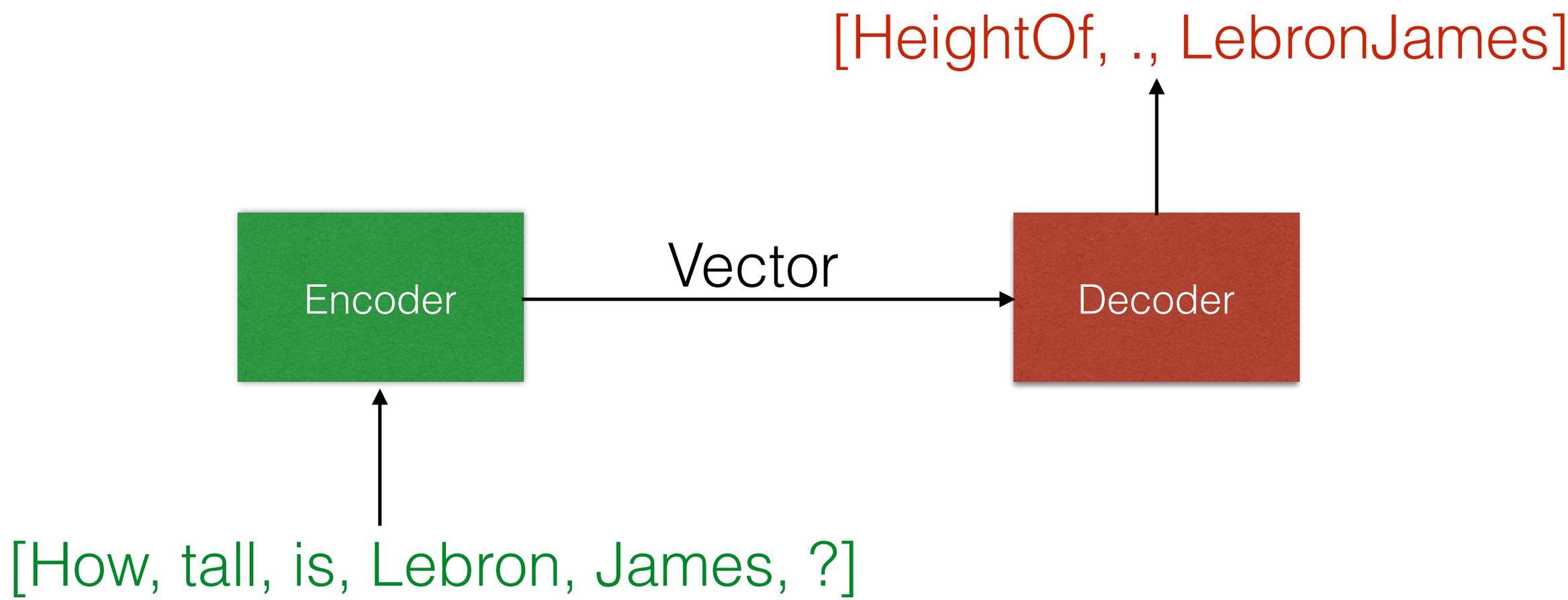
- We will search for parameters that minimize the loss over the training set with SGD.

- We will compute gradients with auto-differentiation packages

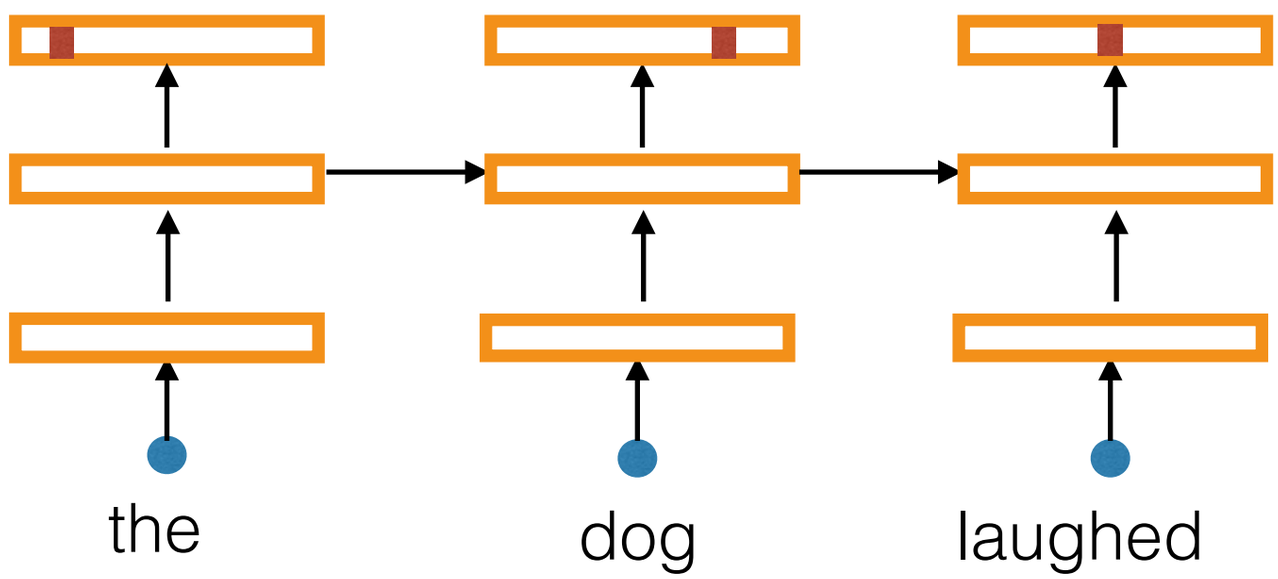
Applications

- Machine translation
- Semantic parsing
- Question answering
- Summarization
- Dialogue
- …

Sequence to sequence



Recurrent neural networks



Input: $w_1, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_T \quad , w_i \in \mathbb{R}^{\mathcal{V}}$

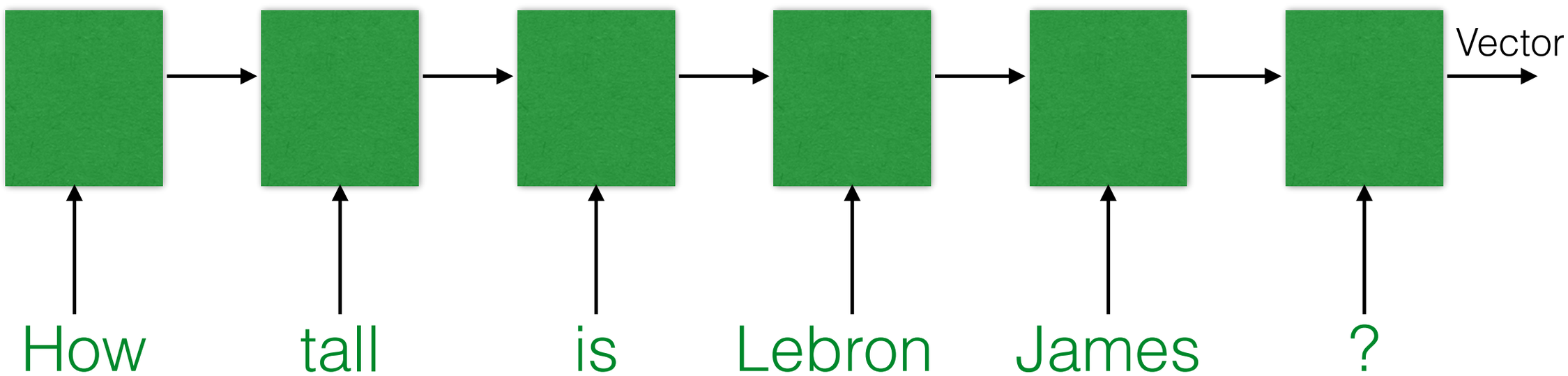
Model: $x_t = W^{(e)} \cdot w_t \quad , W^{(e)} \in \mathbb{R}^{d \times \mathcal{V}}$

$h_t = \sigma(W^{(hh)} \cdot h_{t-1} + W^{(hx)} \cdot x_t), W^{(hh)} \in \mathbb{R}^{D_h \times D_h}, W^{(hx)} \in \mathbb{R}^{D_h \times d}$

$\hat{y}_t = \text{softmax}(W^{(s)} \cdot h_t) \quad , W^{(s)} \in \mathbb{R}^{\mathcal{V} \times D_h}$

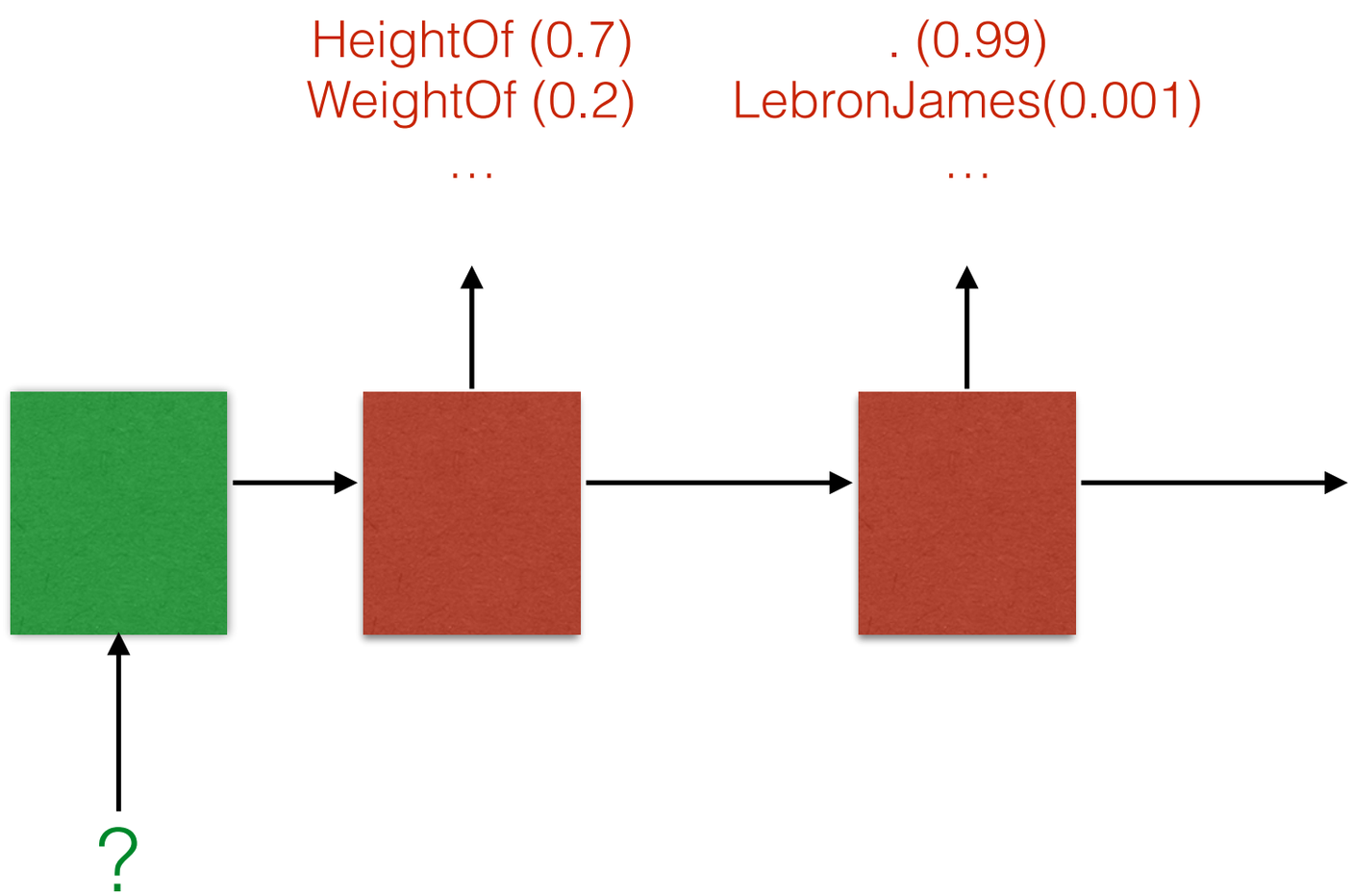
Encoder

An RNN without the output layer



Decoder (v1.0)

An RNN without the input layer



Seq2seq (v1.0)

Encoder:

$h_0^e = 0, h_t^e = \mathrm{RNN}_e(h_{t-1}^e, x_t)$

Decoder:

$h_0^d = \mathrm{RNN}_d(h_{|x|}^e), h_t^d = \mathrm{RNN}_d(h_{t-1}^d)$

$y_t = \mathrm{softmax}(W^{(s)} h_t^d)$

Model:

$$p(y \mid x) = \prod_t p(y_t \mid y_1, \ldots, y_{t-1}, x) = \prod_t p(y_t \mid h_t^d)$$

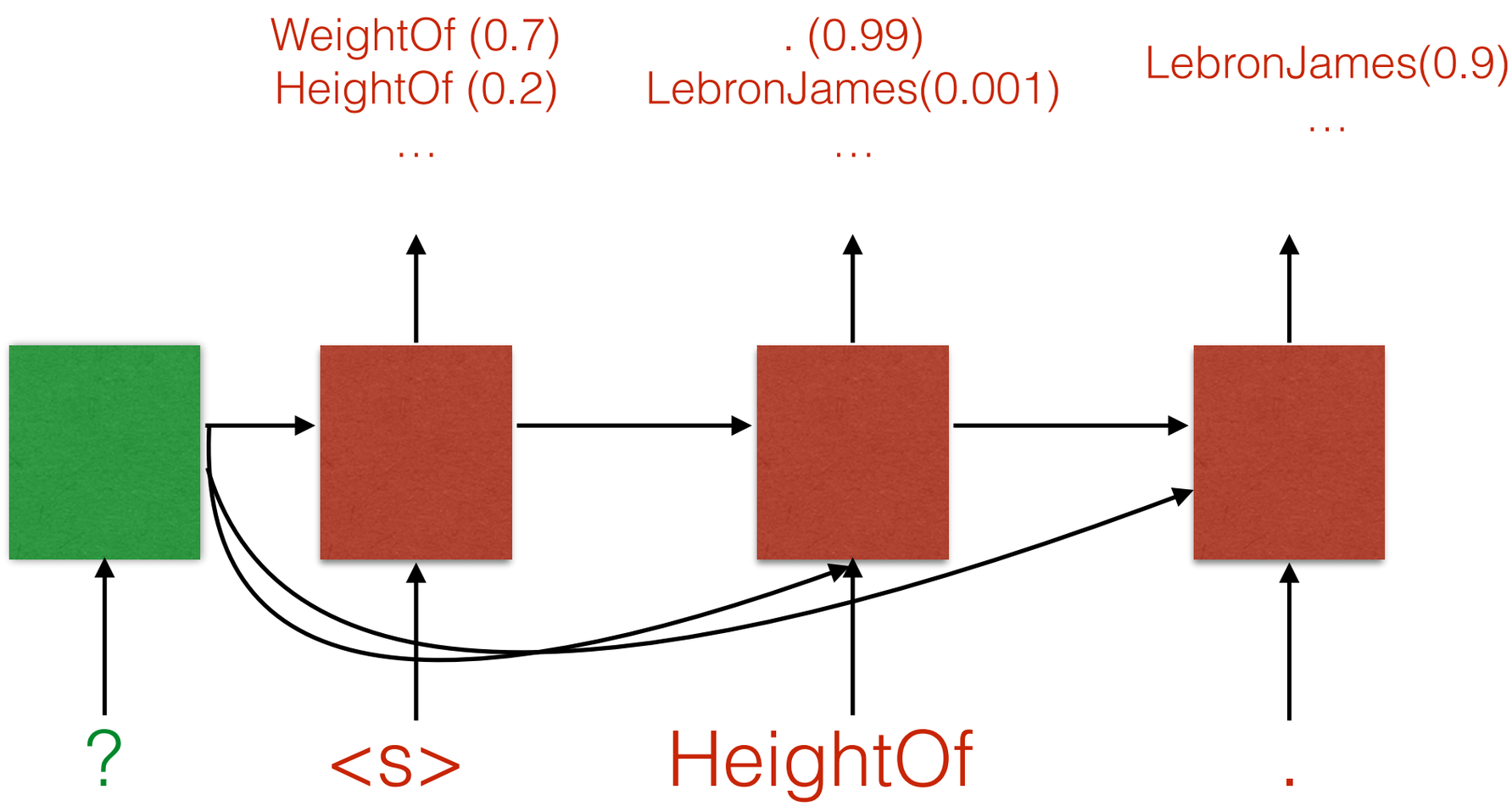
Training is finding parameters that minimize cross entropy over tokens:

$$\sum_i \log p_\theta(y^{(i)} \mid x^{(i)})$$

Seq2seq (v1.0)

- Training is done with SGD on top of standard auto-diff packages
- At training time decoding is done as many steps as the training example (with a stopping symbol)
- At test time we output the argmax token of every time step and stop when we output the stopping symbol.

Seq2seq (v2.0)


WeightOf (0.7)
HeightOf (0.2)
…
. (0.99)
LebronJames(0.001)
…
LebronJames(0.9)
…
?
<s>
HeightOf
.

Seq2seq (v2.0)

Encoder:

$h_0^e = 0, h_t^e = \mathrm{RNN}_e(h_{t-1}^e, x_t)$

Decoder:

$\mathbf{h_t^d} = \mathrm{RNN}_\mathbf{d}(\mathbf{h_{t-1}^d}, \mathbf{h_{|x|}^e}, \mathbf{y_{t-1}})$

$y_t = \mathrm{softmax}(W^{(s)} h_t^d)$

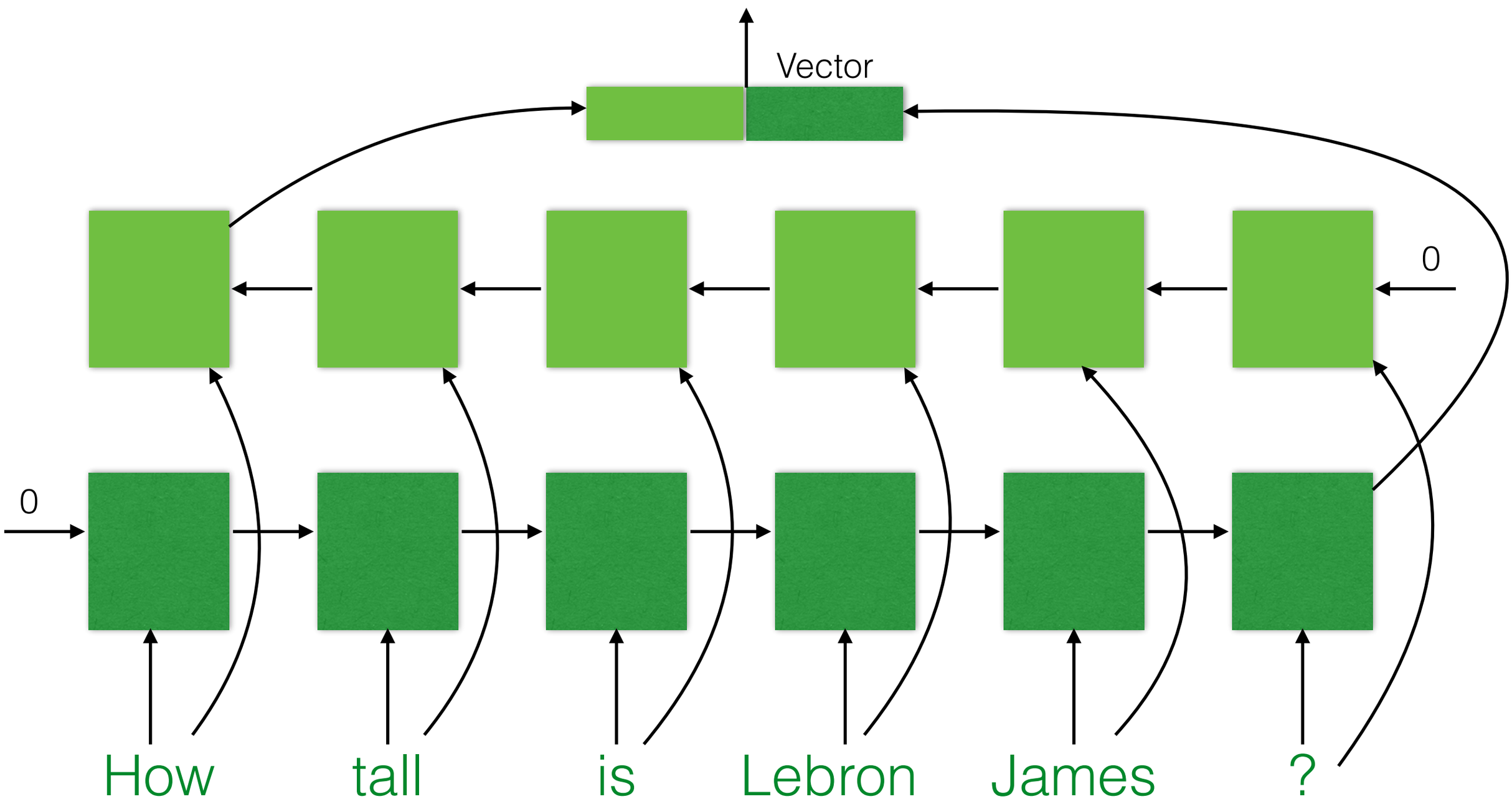
Model:

$$p(y \mid x) = \prod_t p(y_t \mid y_1, \ldots, y_{t-1}, x) = \prod_t p(y_t \mid h_t^d)$$

Training is finding parameters that minimize cross entropy over tokens:

$$\sum_i \log p_\theta(y^{(i)} \mid x^{(i)})$$

Bidirectional encoder



Bidirectional encoder

Encoder:

$$h_0^f = 0, h_t^f = \mathrm{RNN}_f(h_{t-1}, x_t)$$

$$h_{|x|}^b = 0, h_t^b = \mathrm{RNN}_b(h_{t+1}, x_t)$$

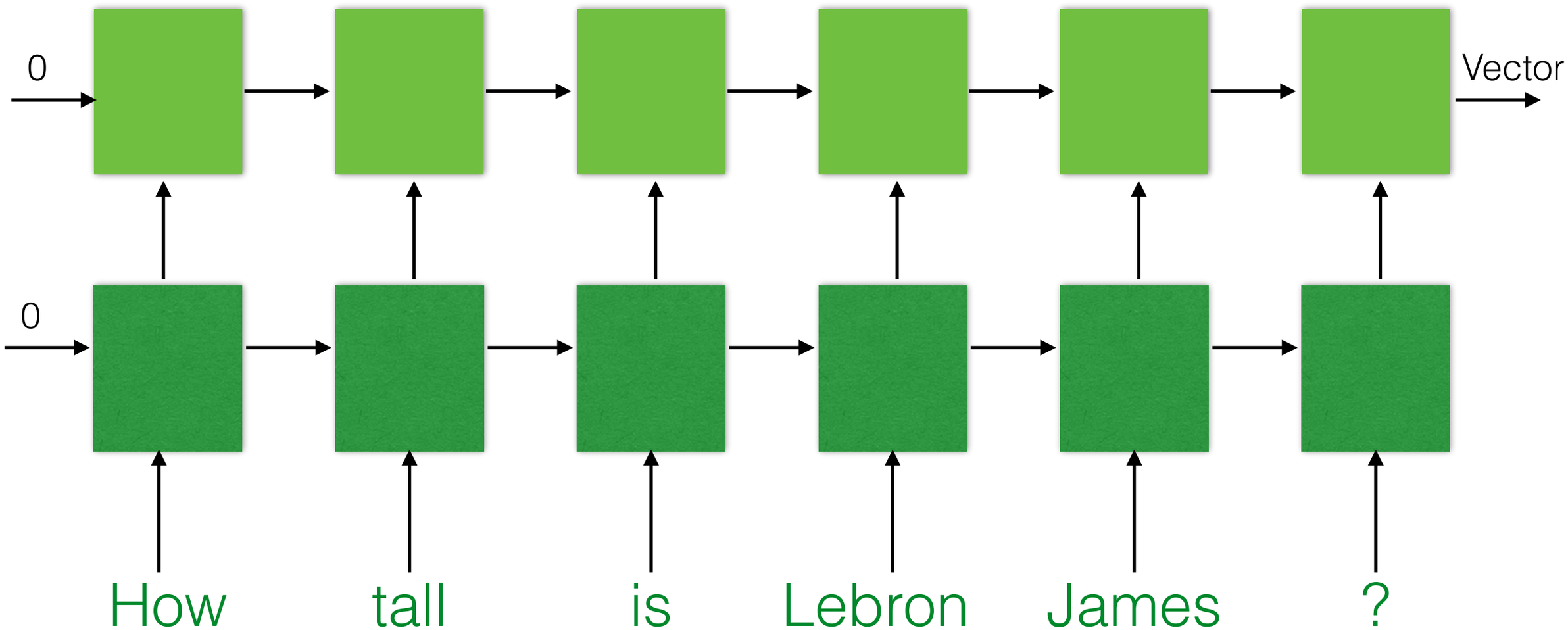
Decoder:

$$h_t^d = \mathrm{RNN}_d(h_{t-1}^d, h_{|x|}^f, h_0^b, y_{t-1})$$

An extremely successful model (state-of-the-art), when using more sophisticated cells (LSTMs, GRUs).

Stacked RNNs

- For encoder or decoder



Wu et al, 2016

Stacked RNN

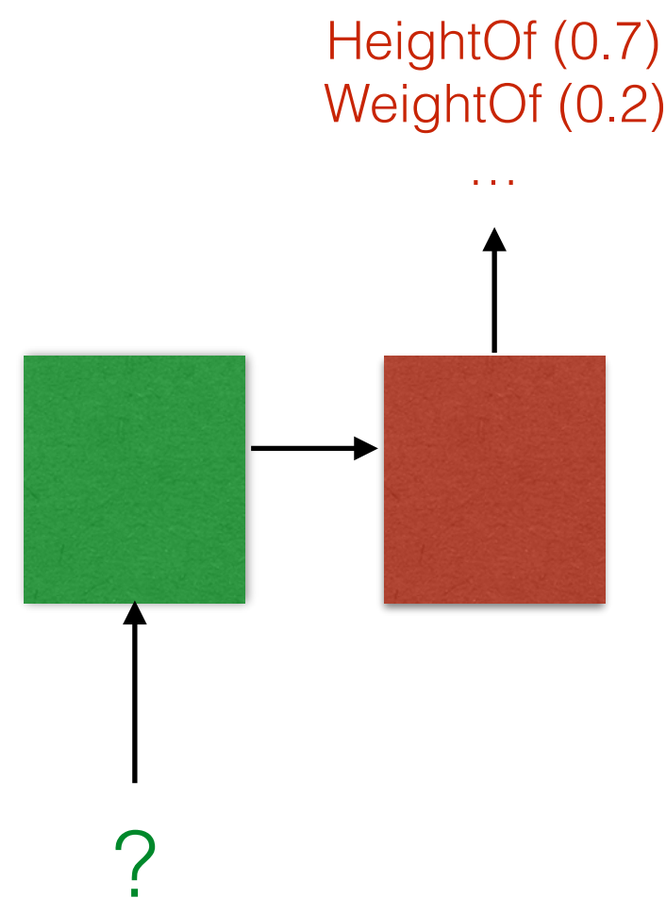
- For stacked RNN, we need to have an output to each state, usually the hidden state itself is used as the input to the next layer
- Empirically stacking RNNs is often better than just increasing the dimensionality
- For example, Google's NMT system uses 8 layers at both encoding and decoding time.

Efficiency

- RNNs are not very efficient in terms of parallelization
- You can not compute h_t before computing h_{t-1} (compared to bag of words or convolutional neural networks)
- This becomes a problem for tasks where one needs to read long documents (summarization).
- Today we have the transformer architecture that can replace the RNN encoder

Beam search

- At test time, decoding is greedy - we output the symbol that has highest probability. **Not** guaranteed to produce the highest probability sequence
- Improved substantially with a small beam.
- At decoding step *t*, we consider *K* most probable sequence prefixes, and compute all possible continuations, score them, and keep the *top-K*
- Burden shift from search to learning again



Advantages of seq2seq

- Simplicity
- Distributed representations of words and phrases
- Better use of context (history) at encoding and decoding time
- Neural networks seem to be very good at generating text (for MT, summarization, QA, etc.)

Summary

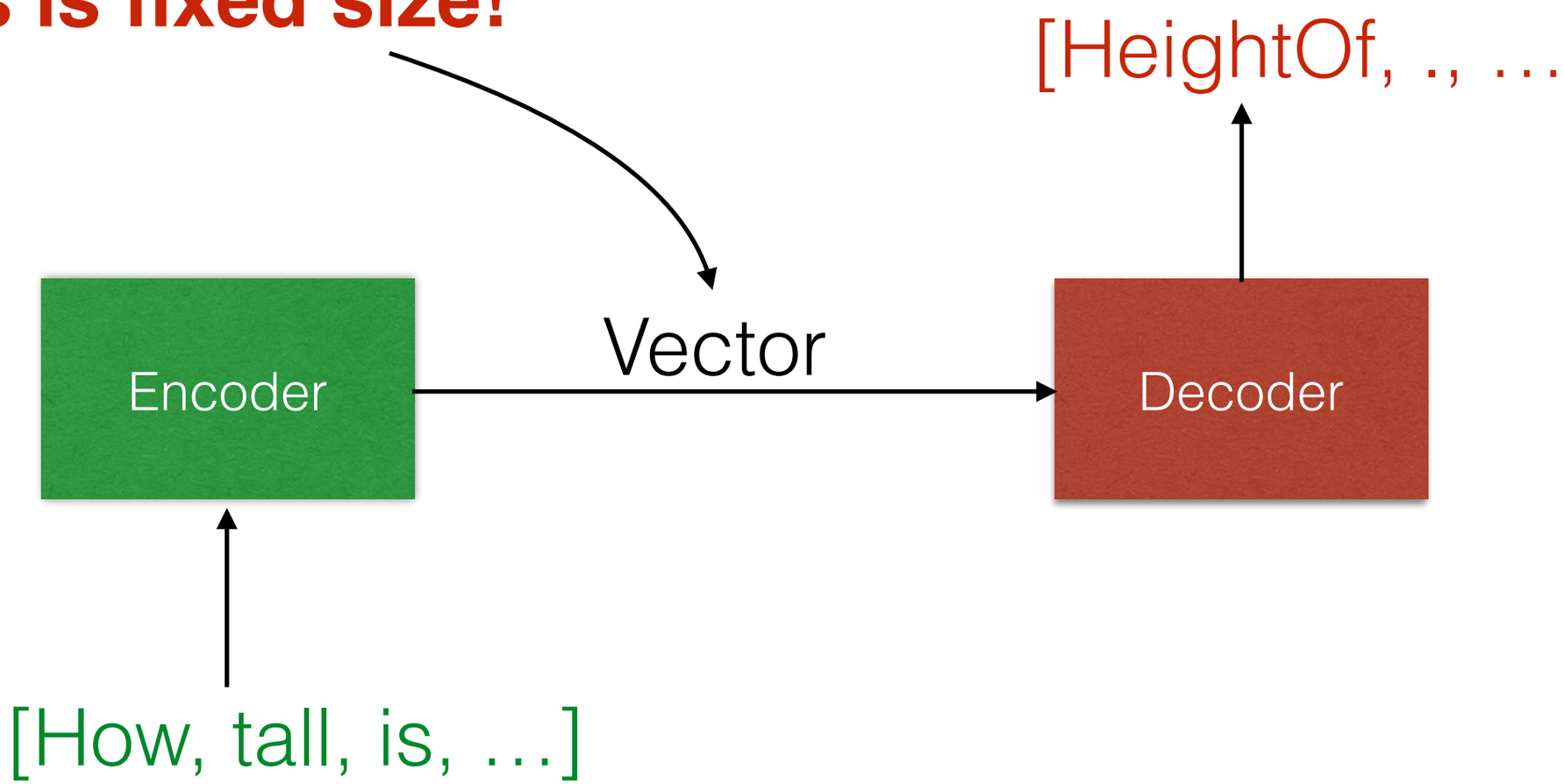
- Sequence to sequence models map a sequence of symbols to another sequence of symbols - very common in NLP!
- LSTMs and GRUs allow this to work for long sequences (~100 steps)
- Results in state-of-the-art performance in many cases
- But not always! Attention!

Attention

The problem

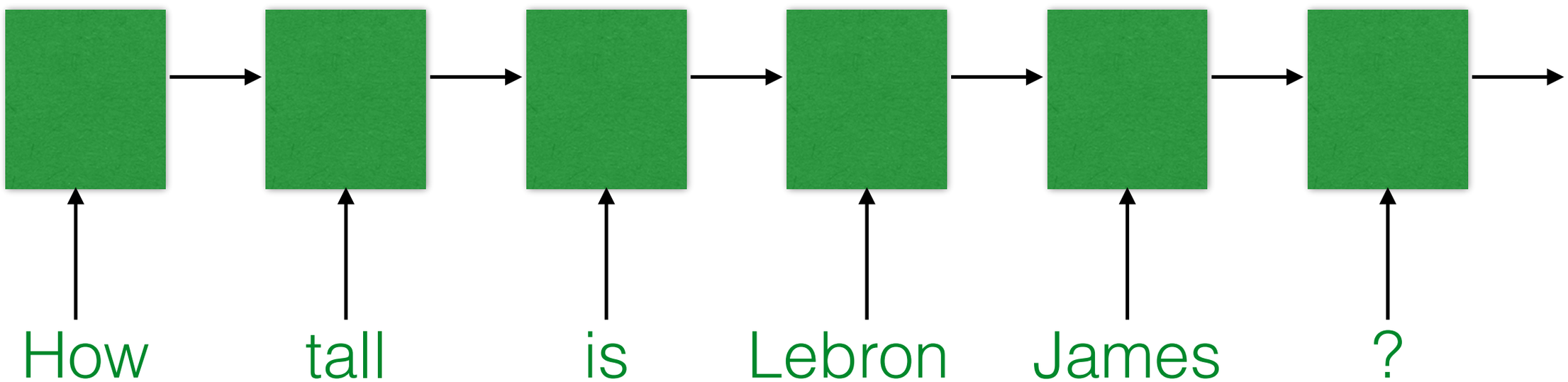
What if the source is very long? "
how tall is the NBA player that has won the most NBA titles before he reached the age of 28?"

This is fixed size!



Attention

Treat source representations as memory

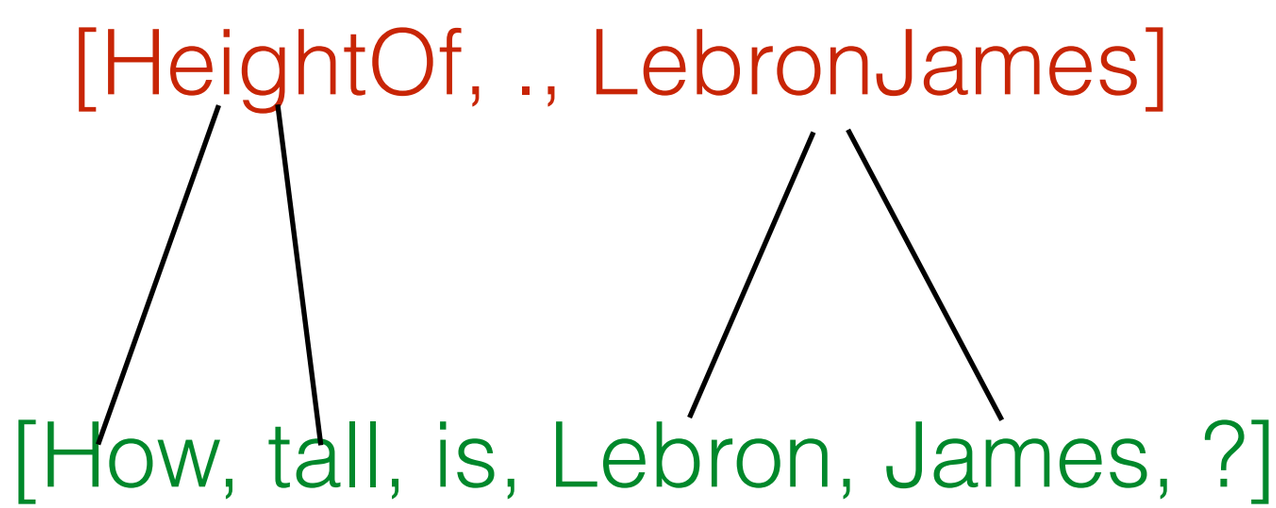


Decide what to read from memory when decoding

Alignment

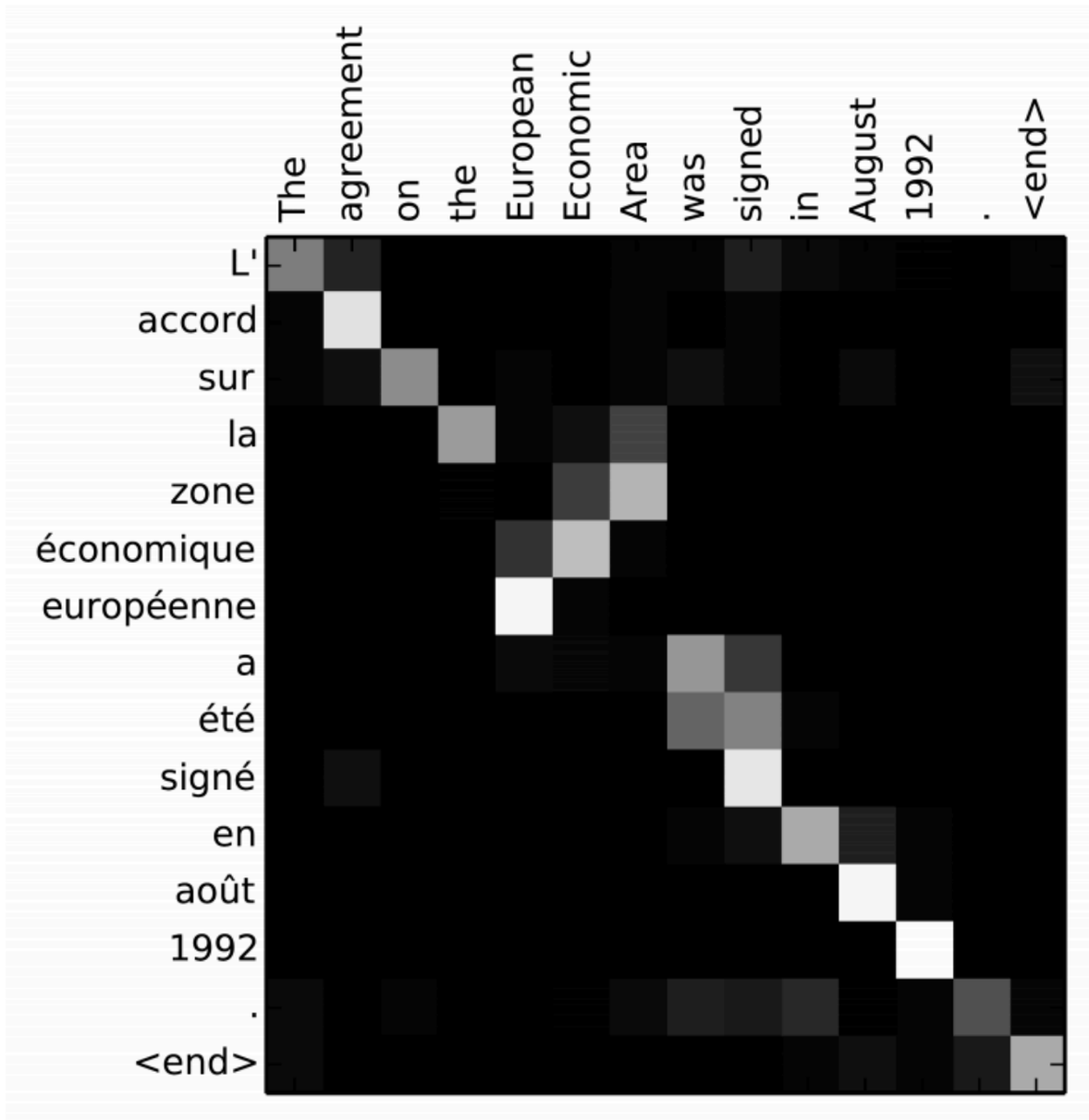
- What are the important words when we decide on the next symbol at decoding time?

- Alignments are heavily used in traditional MT



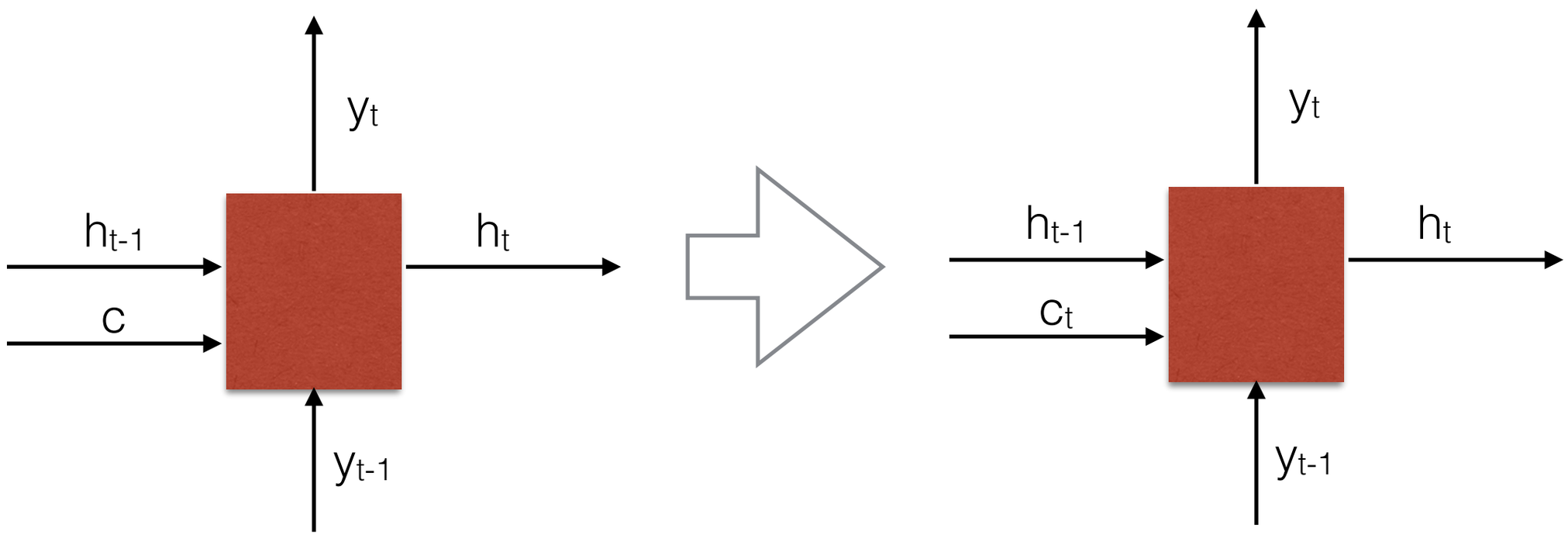
- We will learn to perform the alignment as we decode

Learning alignment in MT



Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. ***Neural Machine Translation by Jointly Learning to Translate and Align****. ICLR'15.*

Decoding



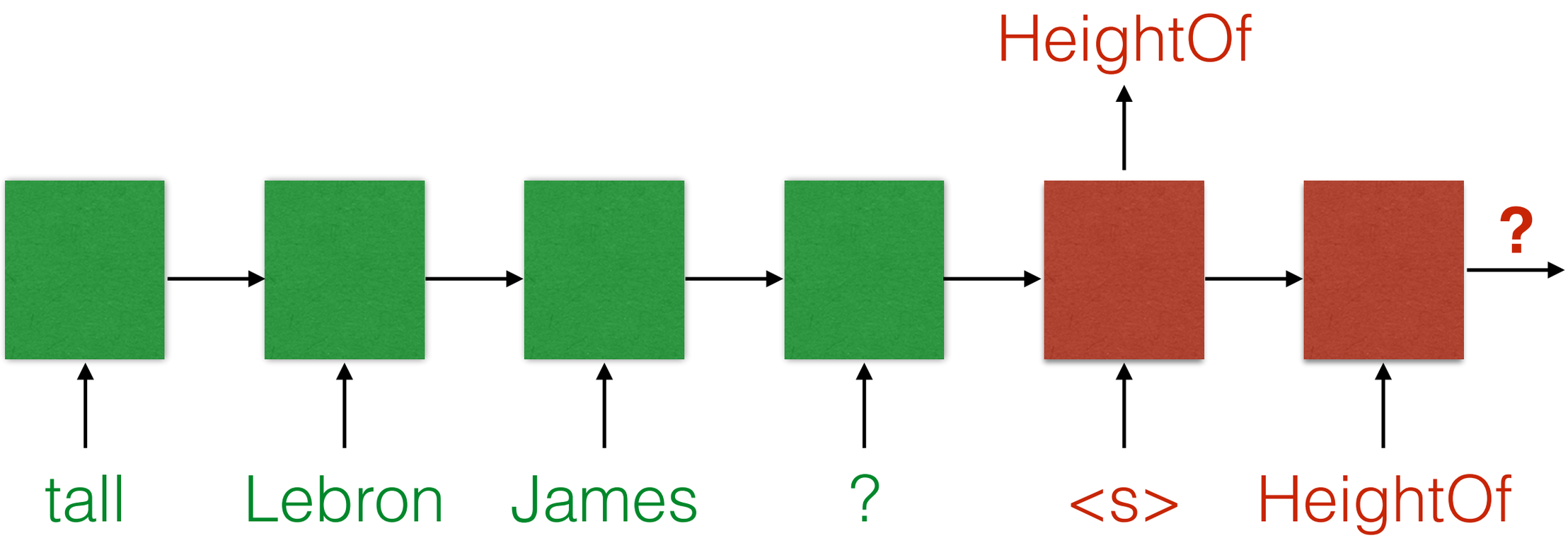
Replace a fixed vector with a time-variable vector

Intuition: before generating a word we softly align to the relevant words in the source!

Attention

To compute c_t:

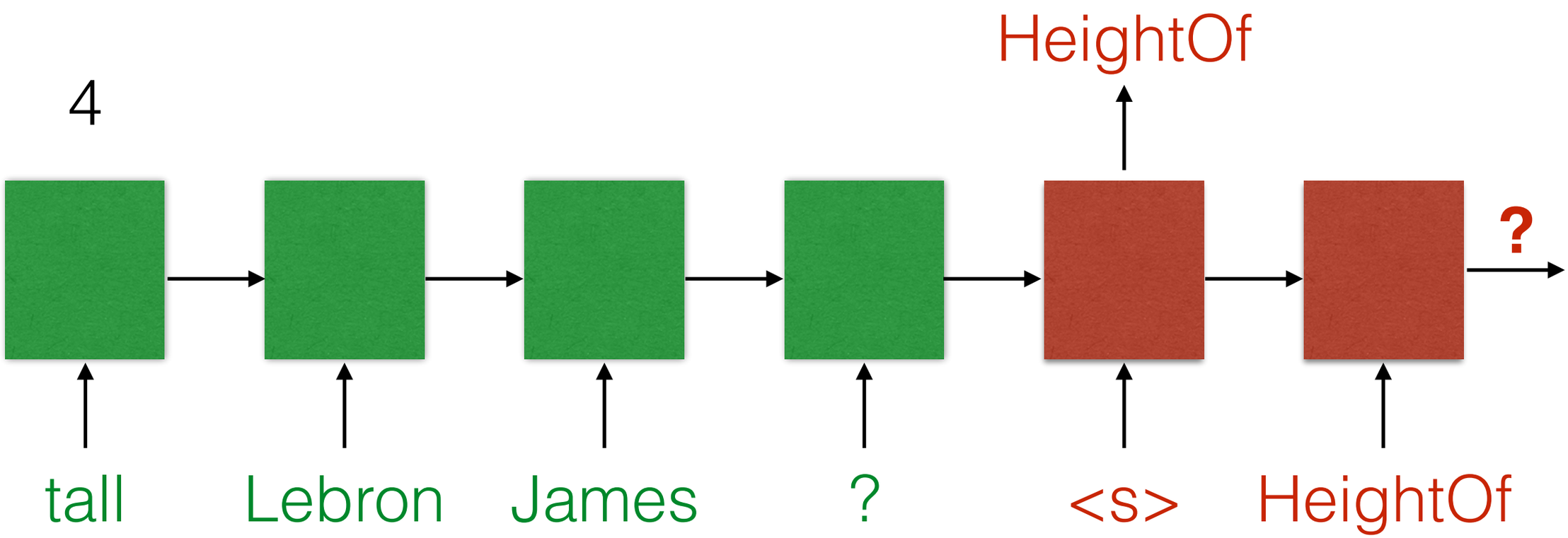
$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$
$$\alpha = \text{softmax}(s)$$



Attention

To compute c_t:

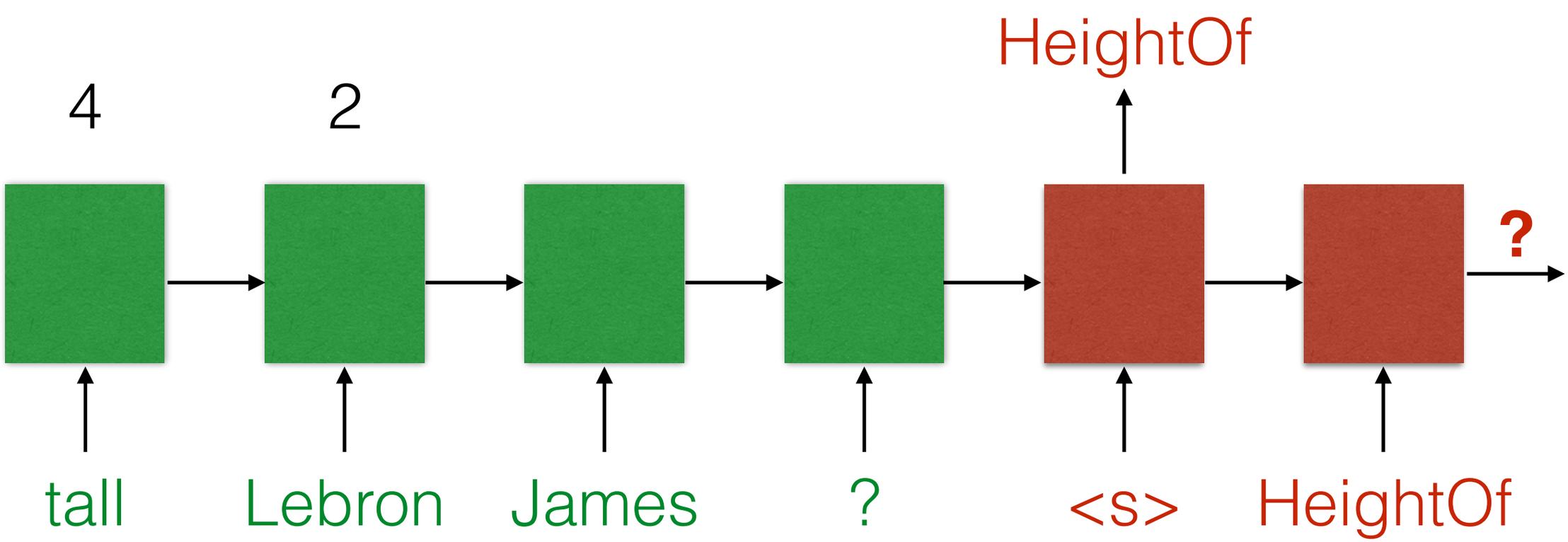
$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$
$$\alpha = \text{softmax}(s)$$



Attention

To compute c_t:

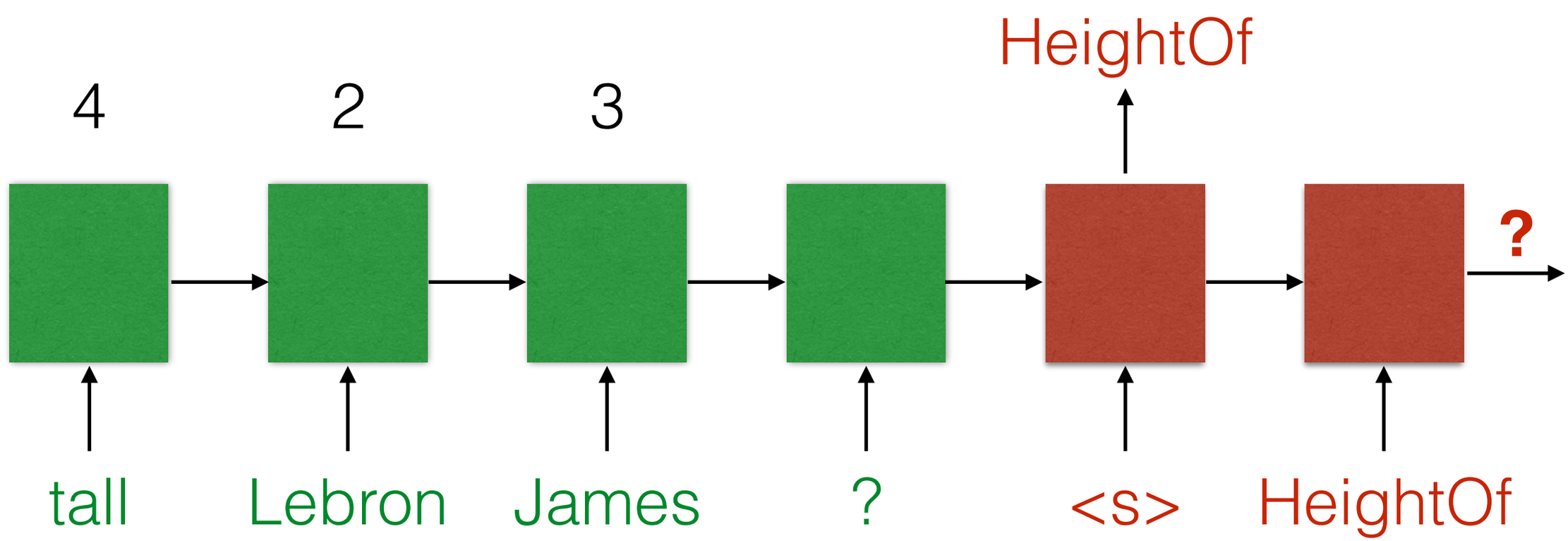
$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$
$$\alpha = \text{softmax}(s)$$



Attention

To compute c_t:

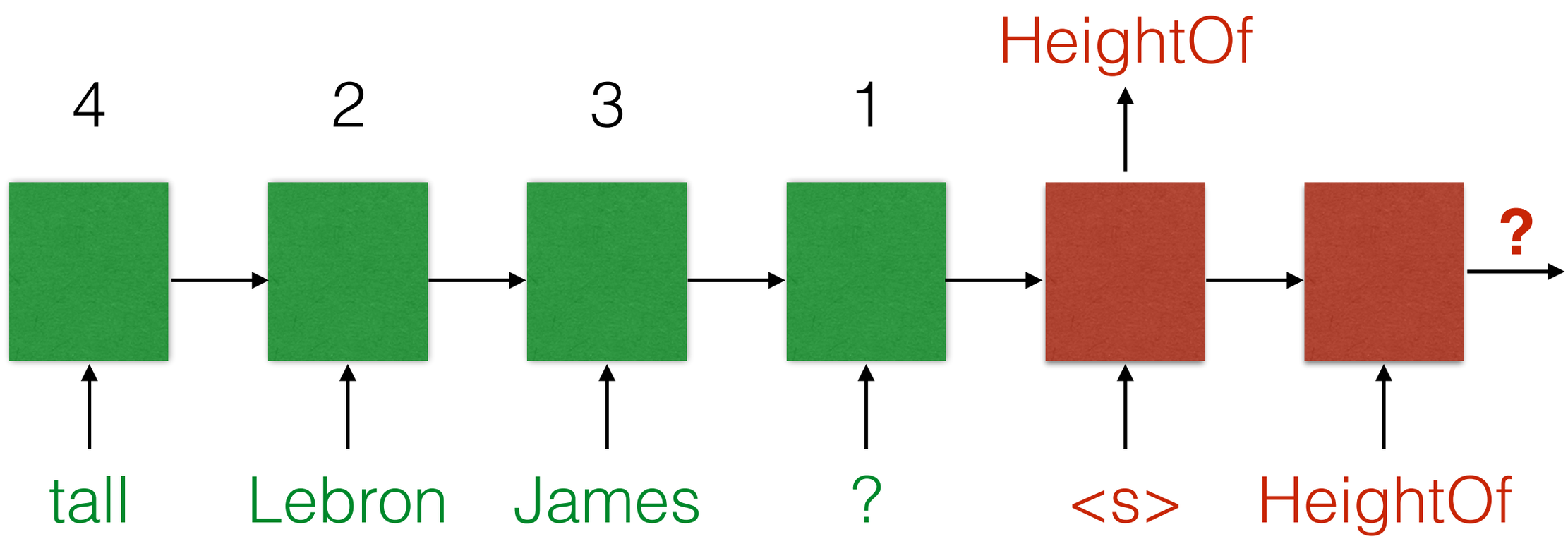
$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$
$$\alpha = \text{softmax}(s)$$



Attention

To compute c_t:

$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$
$$\alpha = \text{softmax}(s)$$

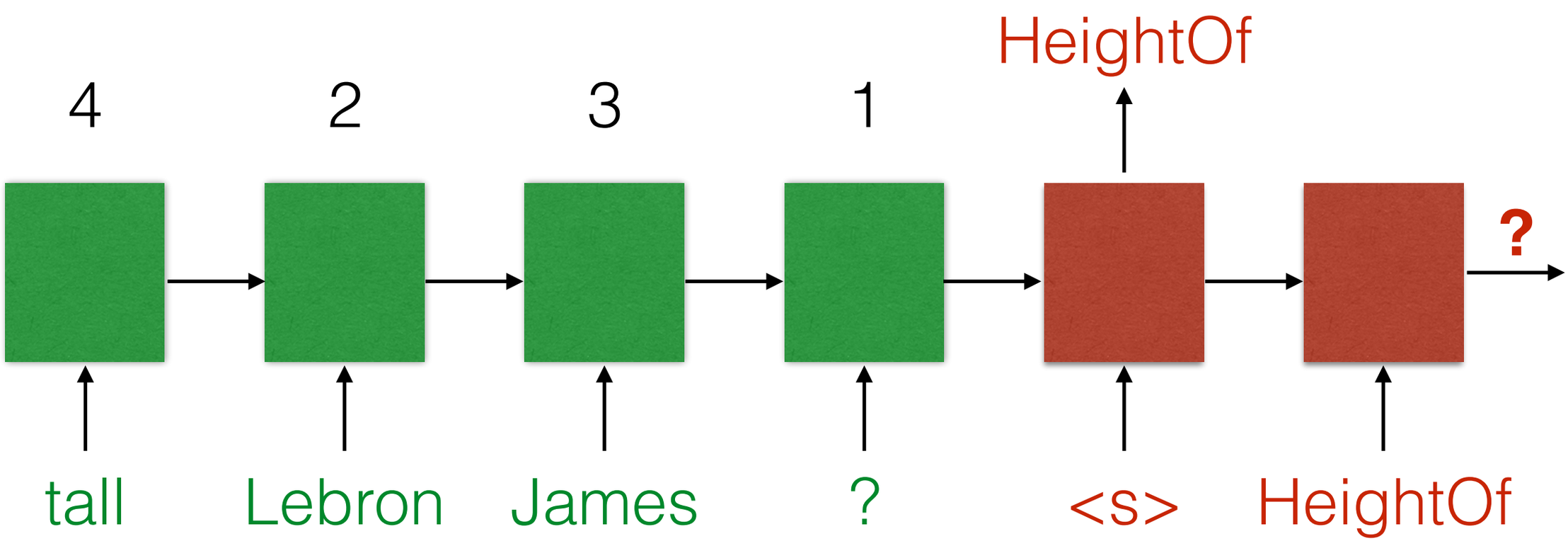


Attention

To compute c_t:

$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$

$$\alpha = \text{softmax}(s)$$



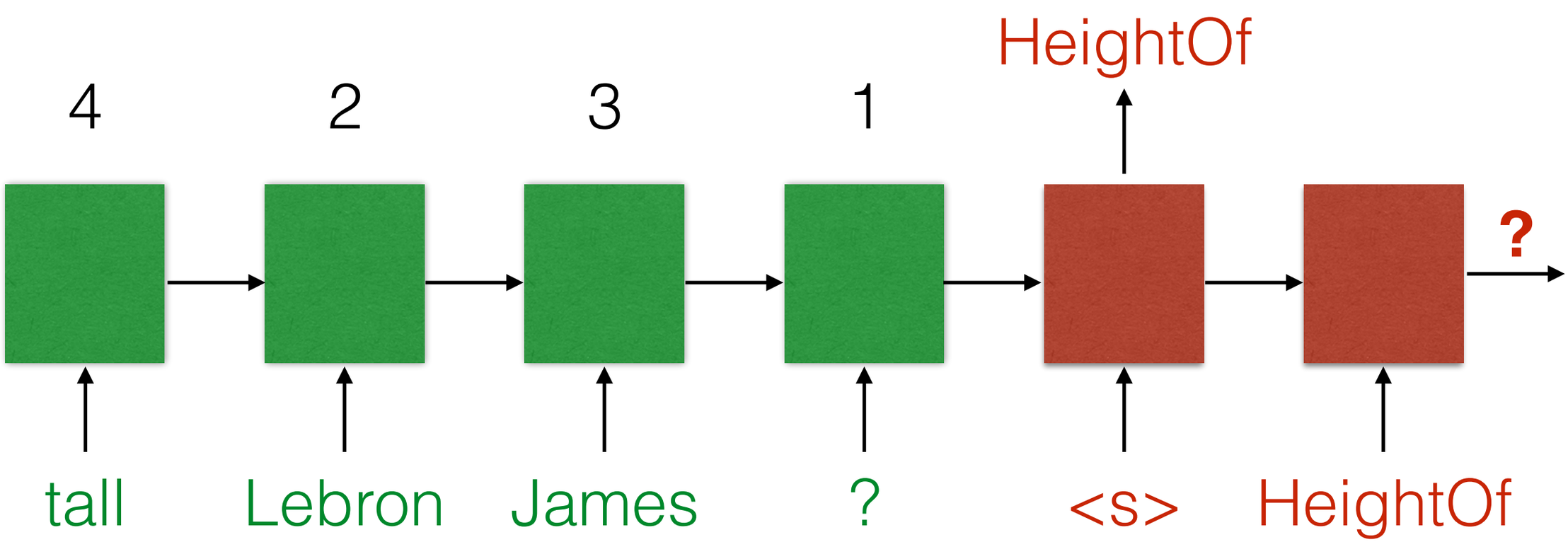
Attention

To compute c_t:

$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$

$$\alpha = \text{softmax}(s)$$

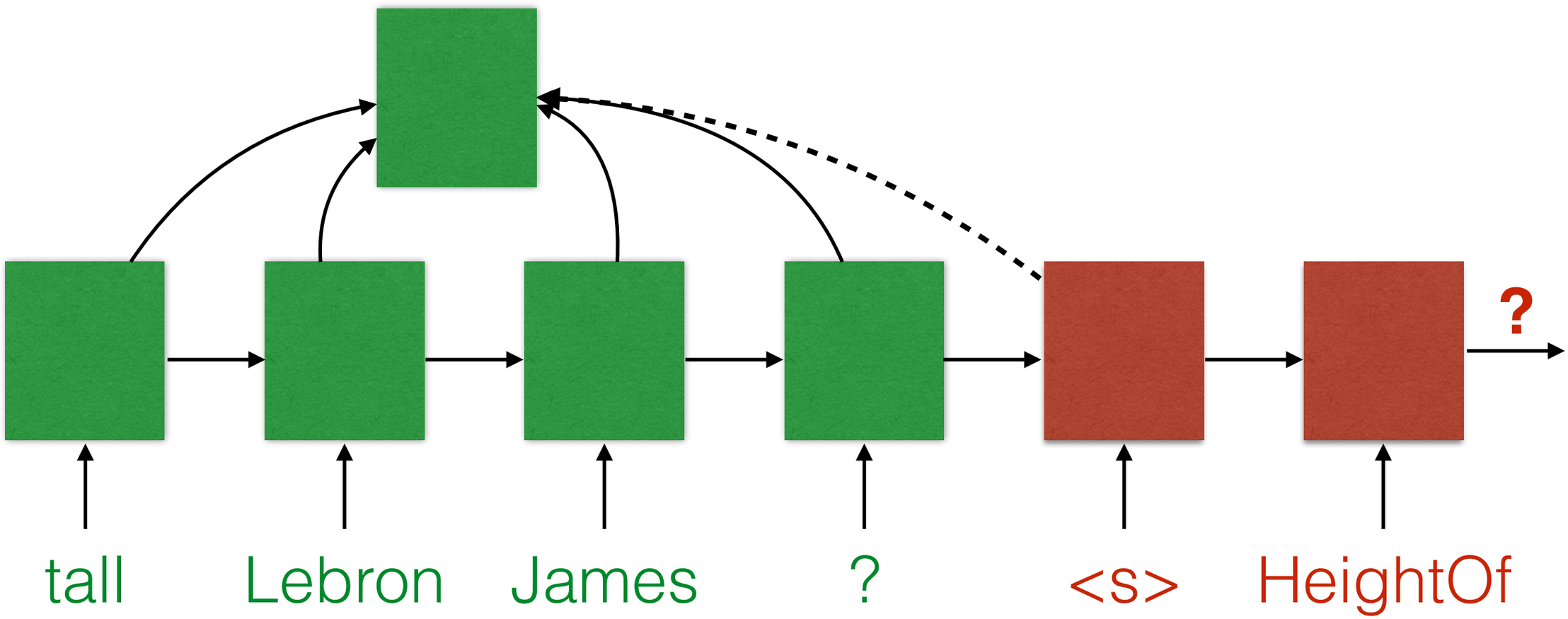
soft alignment!



Attention

To compute c_t:

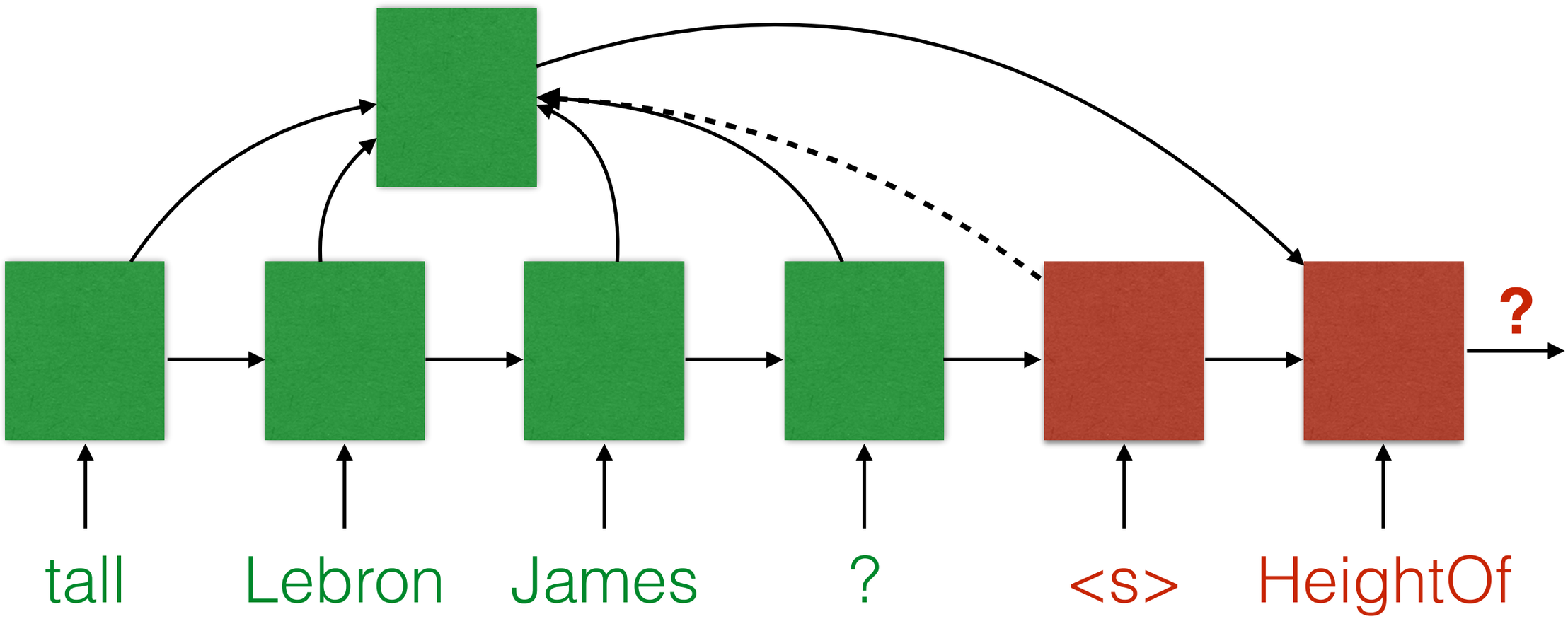
$$c_t = \sum_i \alpha_i h_e^i$$



Attention

To compute c_t:

$$c_t = \sum_i \alpha_i h_e^i$$



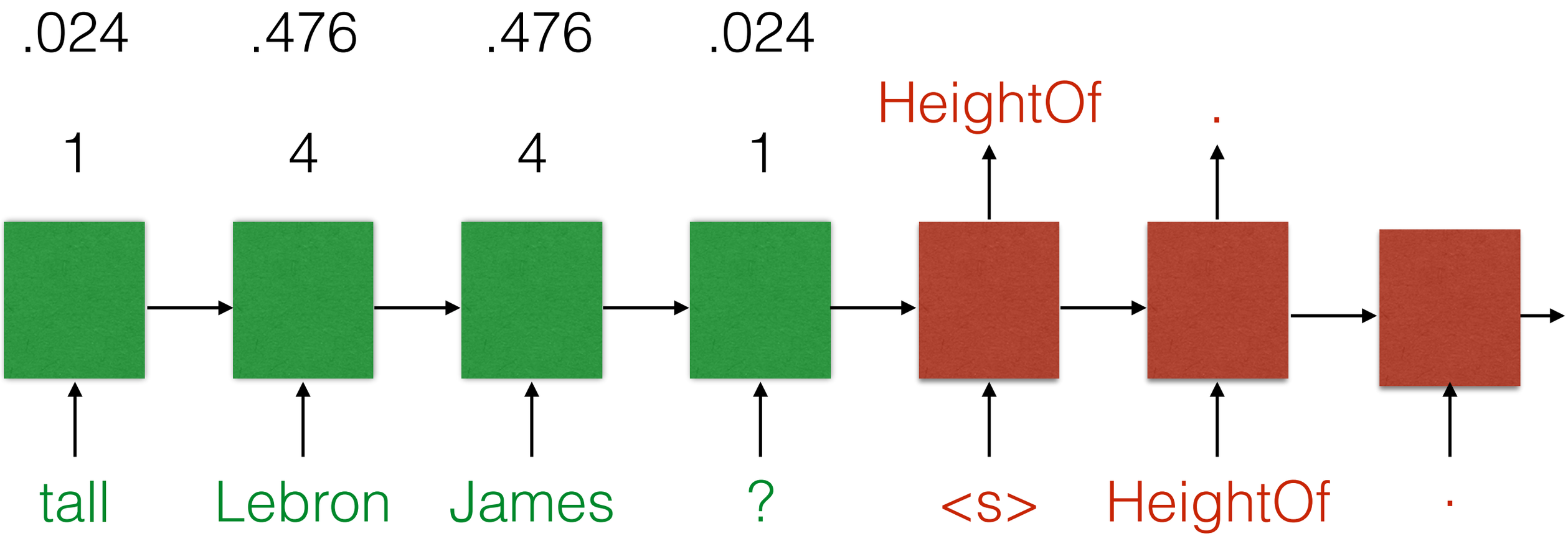
Attention

To compute c_t:

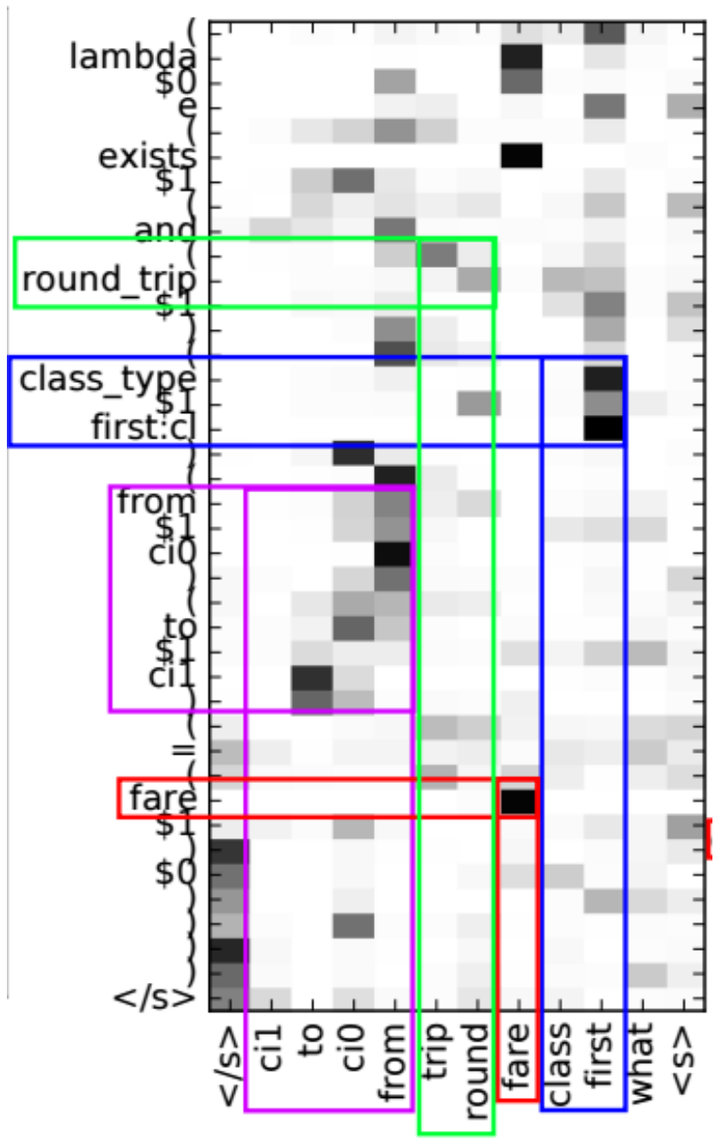
$$\forall i, s_i = \text{score}(h_d^{t-1}, h_e^i)$$

$$\alpha = \text{softmax}(s)$$

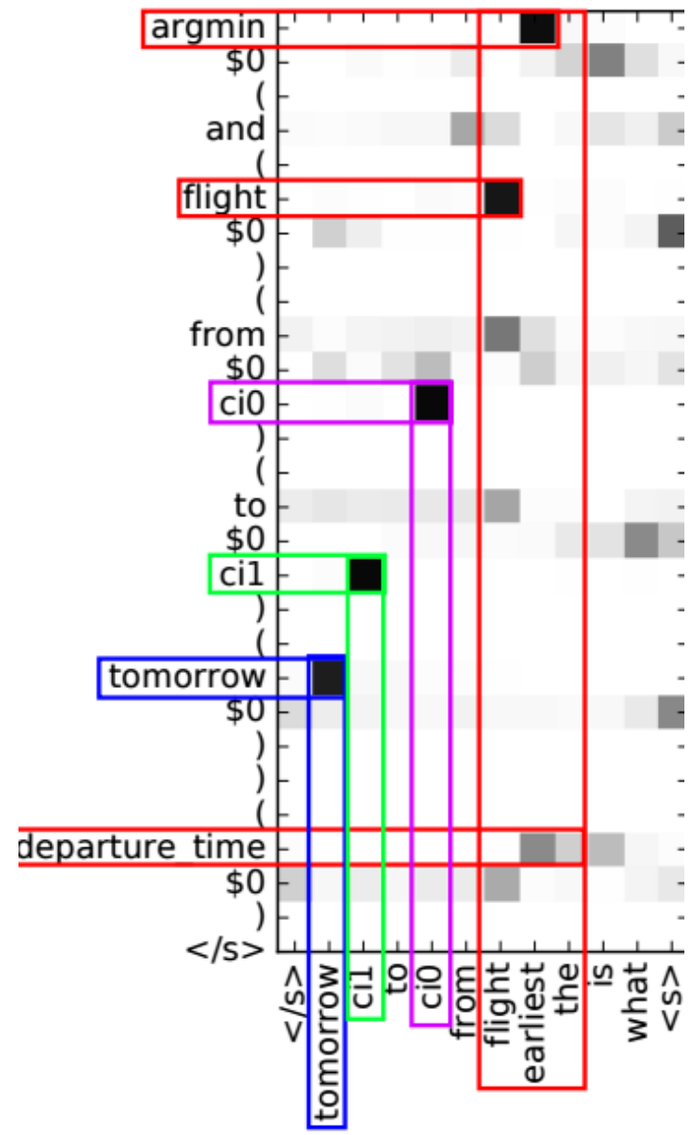
soft alignment!



Attention example



(b) what's first class fare round trip from ci0 to ci1



(c) what is the earliest flight from ci0 to ci1 tomorrow

Attention scoring function

- Options for scoring function:
  - Dot-product
  - Bilinear map
  - Single layer neural net
  - Multiple layer neural net

$$\text{score}(h_d^{t-1}, h_e^i) = {h_d^{t-1}}^\top h_e^i$$
$$\text{score}(h_d^{t-1}, h_e^i) = {h_d^{t-1}}^\top W h_e^i$$
$$\text{score}(h_d^{t-1}, h_e^i) = v^\top \tanh(W_1 h_d^{t-1} + W_2 h_e^i)$$

Dong and Lapata, 2016

Semantic parsing

Method	Accuracy
SCISSOR (Ge and Mooney, 2005)	72.3
KRISP (Kate and Mooney, 2006)	71.7
WASP (Wong and Mooney, 2006)	74.8
λ-WASP (Wong and Mooney, 2007)	86.6
LNLZ08 (Lu et al., 2008)	81.8
ZC05 (Zettlemoyer and Collins, 2005)	79.3
ZC07 (Zettlemoyer and Collins, 2007)	86.1
UBL (Kwiatkowski et al., 2010)	87.9
FUBL (Kwiatkowski et al., 2011)	88.6
KCAZ13 (Kwiatkowski et al., 2013)	89.0
DCS+L (Liang et al., 2013)	87.9
TISP (Zhao and Huang, 2015)	88.9
SEQ2SEQ	84.6
– attention	72.9
– argument	68.6
SEQ2TREE	87.1
– attention	76.8

Table 3: Evaluation results on GEO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZC05 is used for all other systems.

Method	Accuracy
ZC07 (Zettlemoyer and Collins, 2007)	84.6
UBL (Kwiatkowski et al., 2010)	71.4
FUBL (Kwiatkowski et al., 2011)	82.8
GUSP-FULL (Poon, 2013)	74.8
GUSP++ (Poon, 2013)	83.5
TISP (Zhao and Huang, 2015)	84.2
SEQ2SEQ	84.2
– attention	75.7
– argument	72.3
SEQ2TREE	84.6
– attention	77.5

Table 4: Evaluation results on ATIS.

Method	Channel	+Func	F1
retrieval	28.9	20.2	41.7
phrasal	19.3	11.3	35.3
sync	18.1	10.6	35.1
classifier	48.8	35.2	48.4
posclass	50.0	36.9	49.3
SEQ2SEQ	54.3	39.2	50.1
– attention	54.0	37.9	49.8
– argument	53.9	38.6	49.7
SEQ2TREE	55.2	40.1	50.4
– attention	54.3	38.2	50.0

(a) Omit non-English.

Method	Channel	+Func	F1
retrieval	36.8	25.4	49.0
phrasal	27.8	16.4	39.9
sync	26.7	15.5	37.6
classifier	64.8	47.2	56.5
posclass	67.2	50.4	57.7
SEQ2SEQ	68.8	50.5	60.3
– attention	68.7	48.9	59.5
– argument	68.8	50.4	59.7
SEQ2TREE	69.6	51.4	60.4
– attention	68.7	49.5	60.2

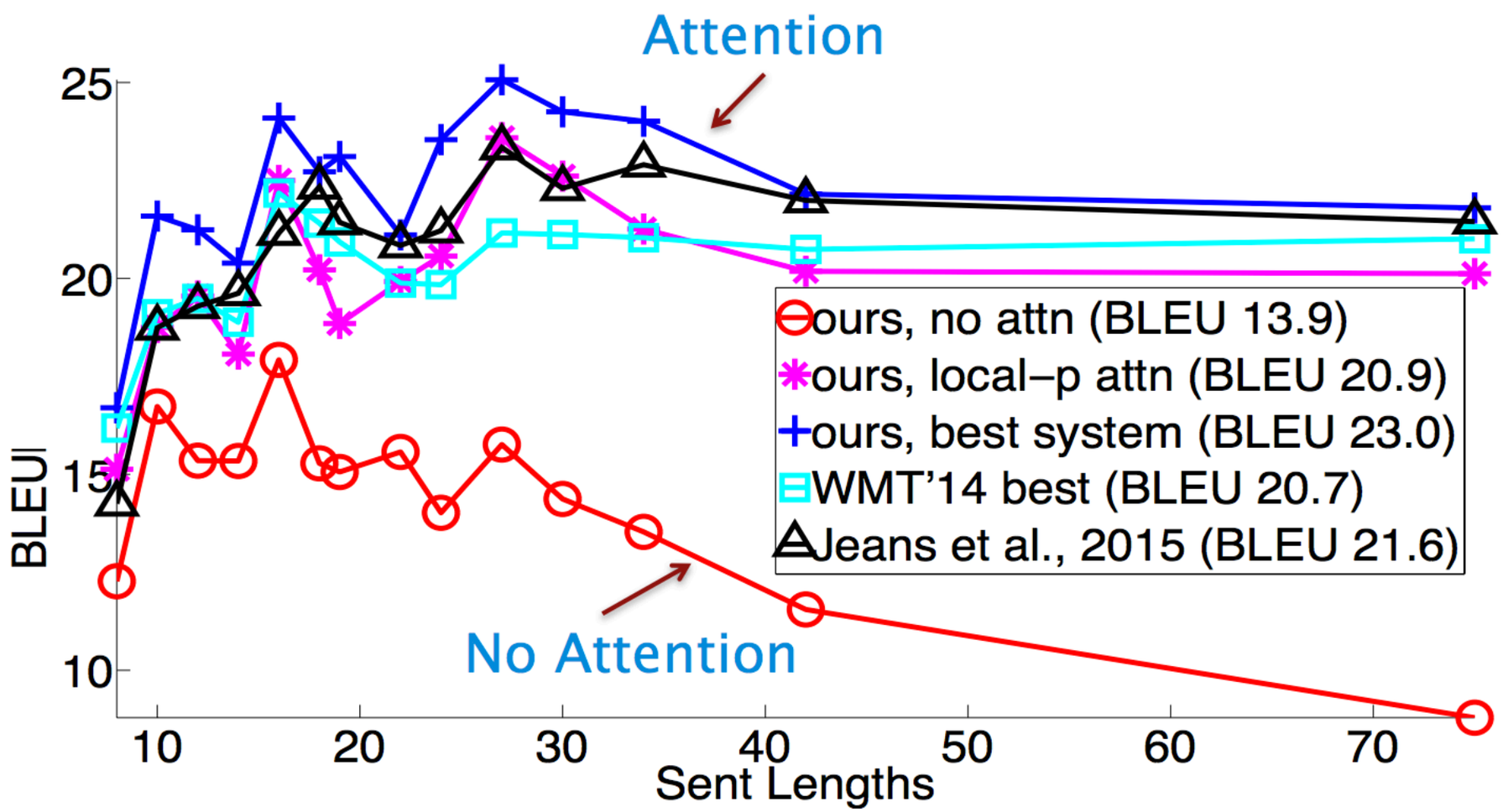
(b) Omit non-English & unintelligible.

Method	Channel	+Func	F1
retrieval	43.3	32.3	56.2
phrasal	37.2	23.5	45.5
sync	36.5	24.1	42.8
classifier	79.3	66.2	65.0
posclass	81.4	71.0	66.5
SEQ2SEQ	87.8	75.2	73.7
– attention	88.3	73.8	72.9
– argument	86.8	74.9	70.8
SEQ2TREE	89.7	78.4	74.2
– attention	87.6	74.9	73.5

(c) ≥ 3 turkers agree with gold.

Table 5: Evaluation results on IFTTT.

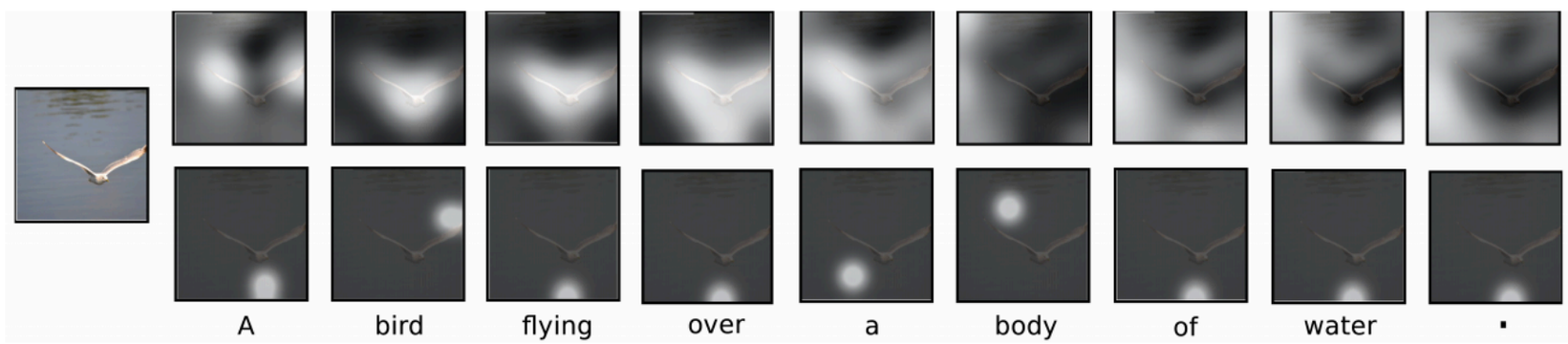
Machine translation



Xu et al., 2015

Coverage

- Caption generation



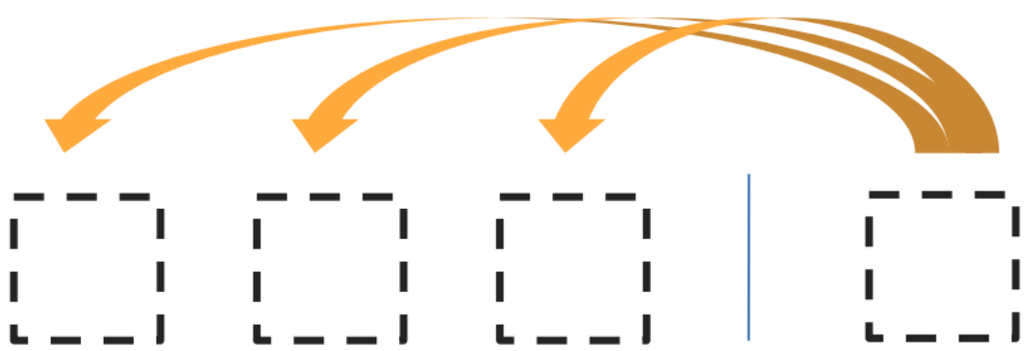
- How do we make sure we cover the source? (also relevant for summarization)
  - Penalize for source patches/words that are not aligned to any target word

$$\sum_{\text{patch}} (1 - \sum_{\text{word}} \alpha_{\text{patch,word}})^2$$

[vaswani et al., 2018]

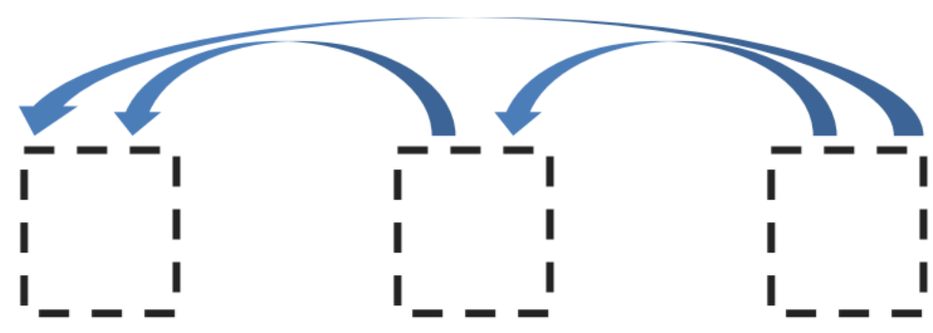
Attention is all you need

- Attention-based encoders **and decoders**



Encoder-Decoder Attention

Encoder Self-Attention



MaskedDecoder Self-Attention

Slide credit: Lukasz Kaiser

Summary

- Attention has enabled getting state-of-the-art performance for transduction scenarios
- Allows to softly align each token in a sequence of text to another sequence

Class 10

Projects

- Not everybody sent project descriptions
- You have until the end of the day

Next week

- Each team doing the research project will describe their project: motivation, plan, evaluation in 3-5 min.
- Please put a pdf with up to two slides in the shared folder.
- Not presenting will result in a 5-point deduction from the project grade. If you have good reason not to present, contact asap.

Attention

- Motivation: eliminate the bottleneck of a single vector in sequence to sequence models
- Framework:
  - Query vector h_q
  - Memory with key, value pairs h_k, h_v
  - h_q is used to score each key h_k and create a distribution over the memory
  - The distribution is used to compute a weighted average over the memory values h_v

Attention

- Seq2seq with LSTMs - one type of attention:
  - h_q: decoder hidden state, h_k: encoder hidden states, h_v: encoder hidden states
- Transformer has two additional attention types:
  - Encoder self-attention: $h_q = h_k = h_v$ are encoder hidden states
  - Decoder self-attention $h_q = h_k = h_v$ are encoder hidden states (when decoding token *t*, the decoder attends only to tokens *1…t-1*)

Pointer networks

Problem

- Often at test time you need to translate entities you have never seen
- If we define the target vocabulary with the training set, we will never get it right
- In addition, translation for those entities is often simply copying

How tall is Dreymond Green?
HeightOf.DreymondGreen

Solution 1

- Mask entities
- Translate
- Bring back entities
  - But if there are many entities
  - How do you identify entities?

How tall is <e>?
HeightOf.<e>

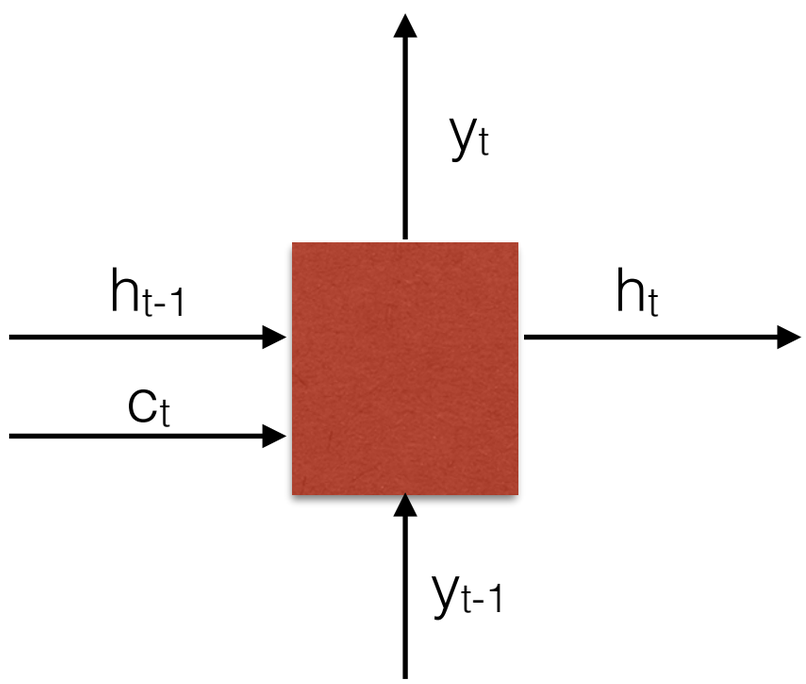
Idea

- When we translate a sentence, the probability of a word increases once we see it.
  - P(“pokemen") is low
  - P(“pokemon" | “the pokemon company”) is high
- Let’s allow outputting either words from a fixed target vocabulary or any word from the source sentence

Vinyals, et al, 2015, Jia and Liang, 2016

Regular model

$$p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \exp(U_w h_t)$$



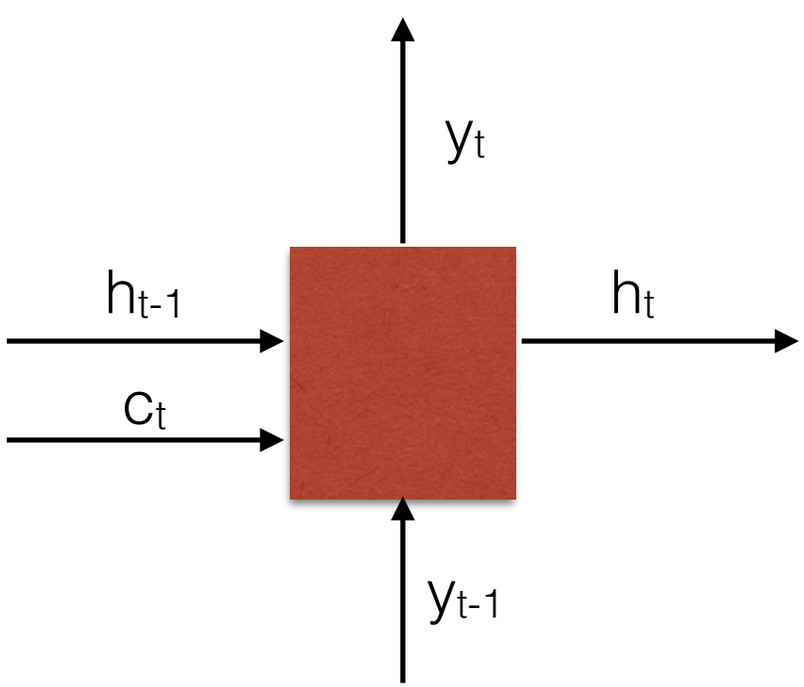
HeightOf [3]
WeightOf [-1]
NumAssists [40]
([9]
) [5.8]
. [3.7]
and [13]

Vinyals, et al, 2015, Jia and Liang, 2016

Copying model

$$p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \exp(U_w h_t)$$
$$p(y_t = x_i \mid x, y_1, \ldots, y_{t-1}) \propto \exp(s_{ti})$$

How tall is Dreymond Green ?



HeightOf [3]
WeightOf [-1]
NumAssists [40]
([9]
) [5.8]
. [3.7]
and [13]
How [5]
tall [1]
is [-2]
Dreymond [100]
Green [100]
? [0]

Copying model

- Need to marginalize over the words since there could be repetitions
- At training this means that the true distribution is uniform over all correct tokens
- At test time we choose the highest probability token, but marginalize over the same instances of a token

Slight improvement

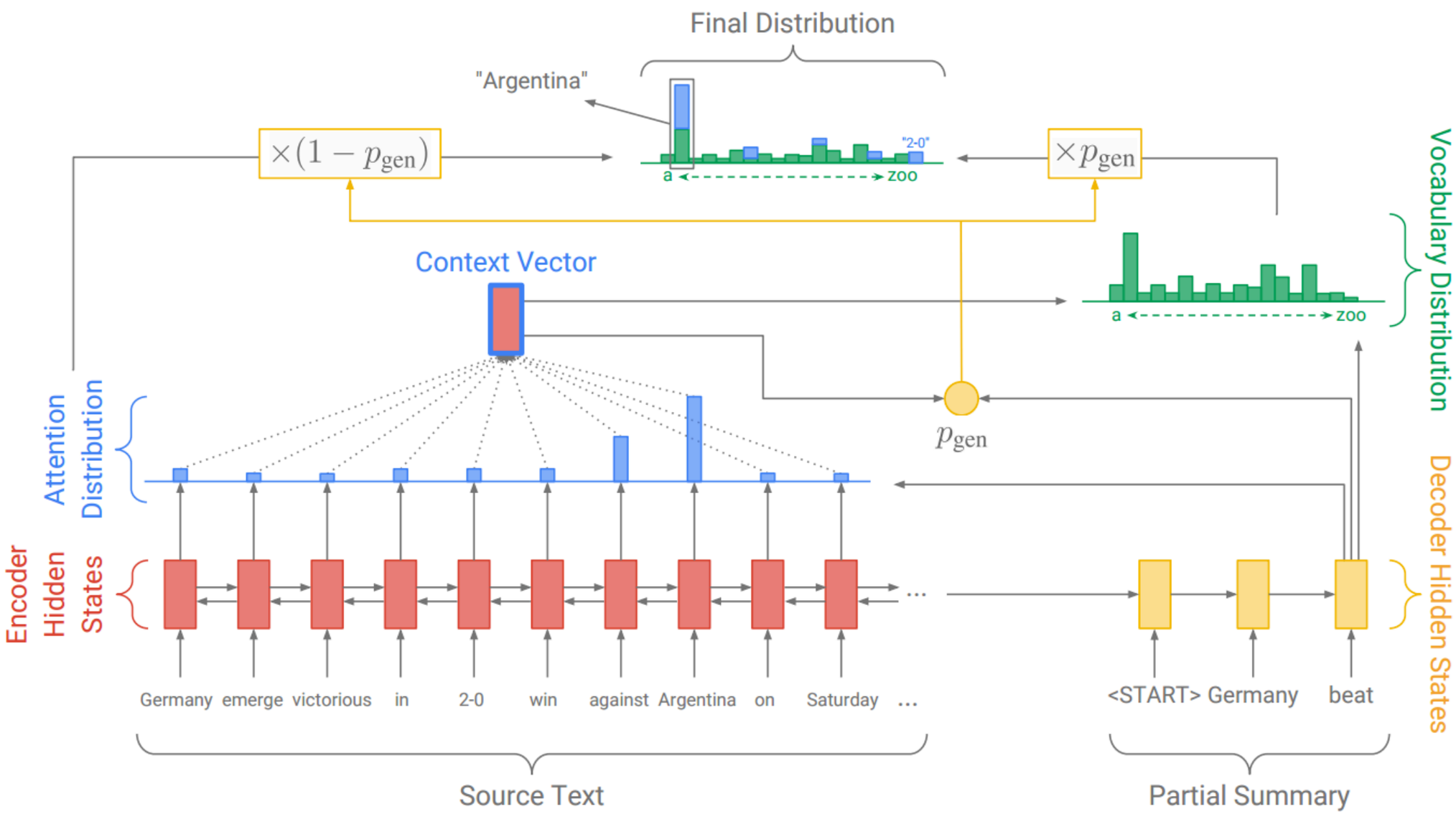
$$p(y_t = w \mid x, y_1, \ldots, y_{t-1}) \propto \boxed{\exp(U_w h_t)}$$
$$p(y_t = x_i \mid x, y_1, \ldots, y_{t-1}) \propto \boxed{\exp(s_{ti})}$$

- These scores need to be calibrated
- We can just interpolate two distributions after normalization

$$p_{\text{vocab}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(U h_t)$$
$$p_{\text{copy}}(y_t \mid x, y_1, \ldots, y_{t-1}) = \text{softmax}(s_t)$$
$$p(y_t \mid x, y_1, \ldots, y_{t-1}) = p_{\text{gen}} \cdot p_{\text{vocab}} + (1 - p_{\text{gen}}) \cdot p_{\text{copy}}$$
$$p_{\text{gen}} = \sigma(w_1^\top h_t + w_2^\top c_t + w_3^\top y_{t-1})$$

See et al., 2016

Illustration



Summary

- Neural network for semantic parsing are based on sequence to sequence models
- These models are useful also for summarization, dialogue, question answering, paraphrasing, and other transduction tasks
- Attention added memory to circumvent the constant representation problem
- Pointer networks help in handling new words at test time
- Together you can often get models that are comparable to state-of-the-art without a grammar

Weak supervision

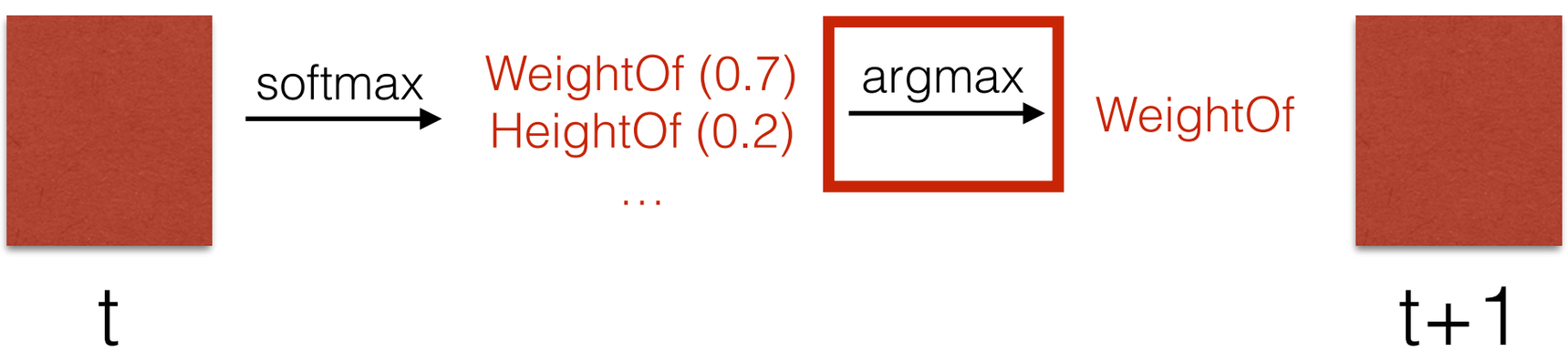
Weak supervision

- We have assumed that we have as input pairs of natural language and logical form
- In practice those are hard to collect and we usually have (language, denotation) pairs

Heavy supervision	**Light supervision**
How tall is Lebron James?	*How tall is Lebron James?*
HeightOf.LebronJames	203cm
What is Steph Curry's daughter called?	*What is Steph Curry's daughter called?*
ChildrenOf.StephCurry $\sqcap$ Gender.Female	Riley Curry
Youngest player of the Cavaliers	*Youngest player of the Cavaliers*
$\arg\min$(PlayerOf.Cavaliers, BirthDateOf)	Kyrie Irving
...	...

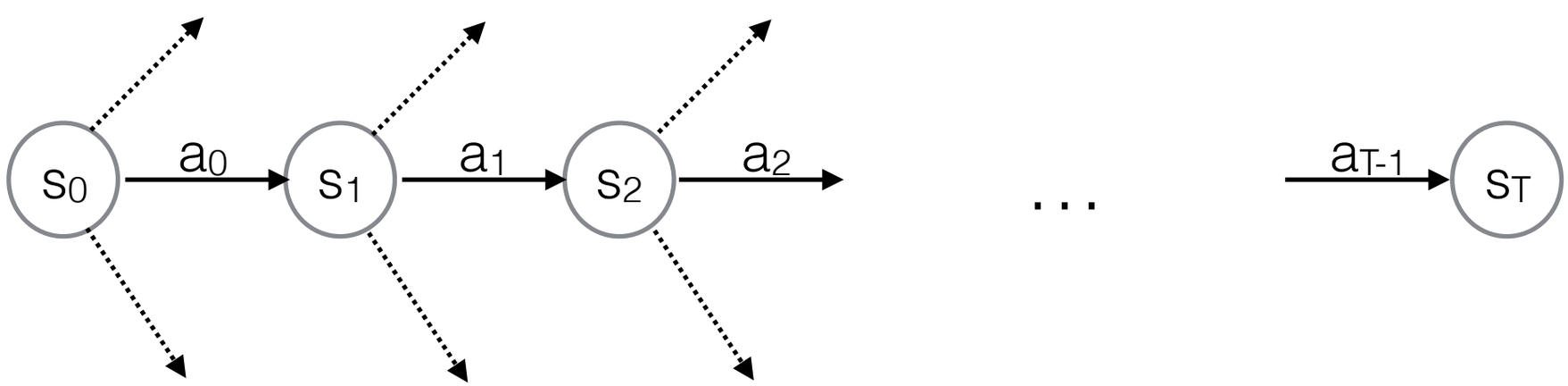
The problem

- In sequence to sequence we trained end-to-end with SGD, minimizing the cross entropy loss of every token
- Here we don't have tokens
- <u>Suggestion</u>: generate the program token by token, execute, and minimize cross entropy over denotations
- <u>Problem</u>: The loss is not a differentiable function of the input because we don't input gold tokens



Markov Decision Process

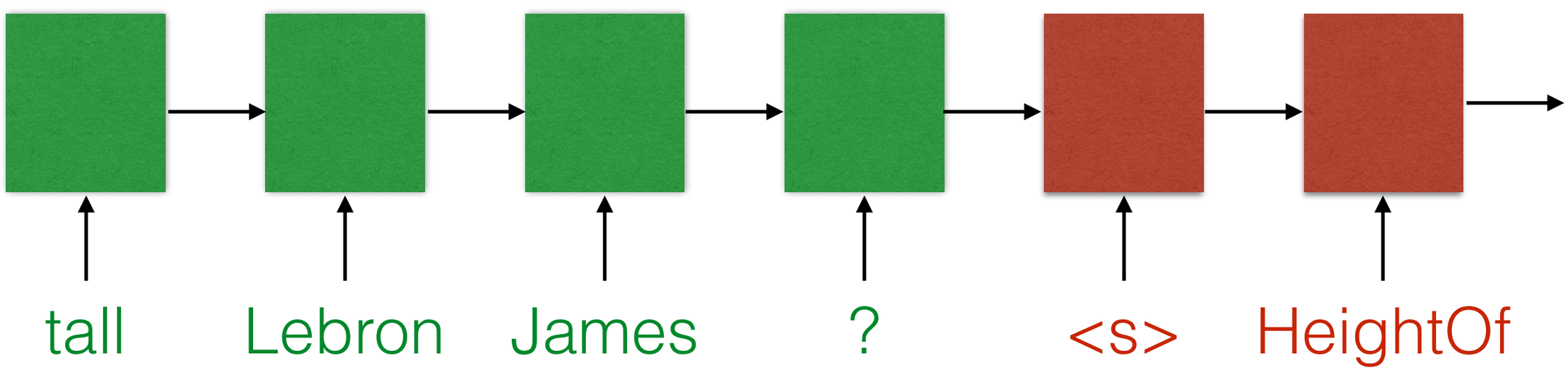
- Sequence of states, actions and rewards
  - $s_0, s_1, s_2, \ldots, s_T$ from a set S
  - $a_0, a_1, a_2, \ldots, a_T$ from a set A
    - Let's assume a deterministic transition function $f{:}SxA{-}{>}S$
  - $r_0, r_1, r_2, \ldots, r_T$ given by a reward function $r(s,a)$
- We want a policy $\pi(a \mid s)$ providing a distribution over actions that will maximize future reward



Liang et al, 2017, Guu et al., 2017

Seq2seq as MDP

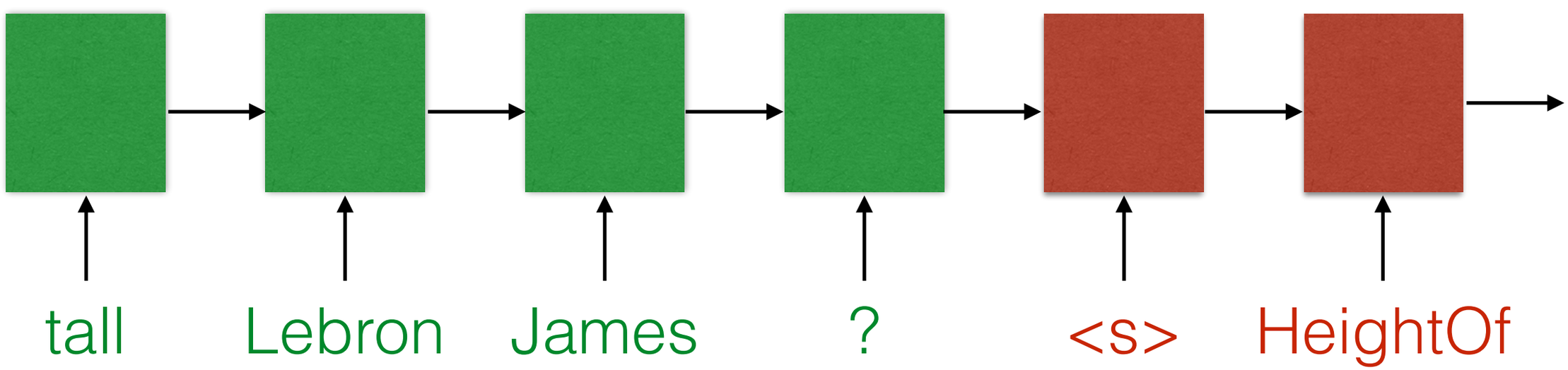
- s_t: h_t
- a_t is in $A(s_t)$
  - Either all symbols in the target vocabulary
  - All **valid** symbols if we check grammaticality
- r_t is zero in all steps except the last. Then, it is 1 if execution results in a correct answer and 0 otherwise.



Seq2seq as MDP: policy

$$
\begin{aligned}
p(z \mid x) &= \prod_t p(z_t \mid x, z_0, \ldots, z_{t-1}) \\
&= \prod_t p(a_t \mid x, a_0, \ldots, a_{t-1}) \\
&= \prod_t \pi(a_t \mid s_t)
\end{aligned}
$$

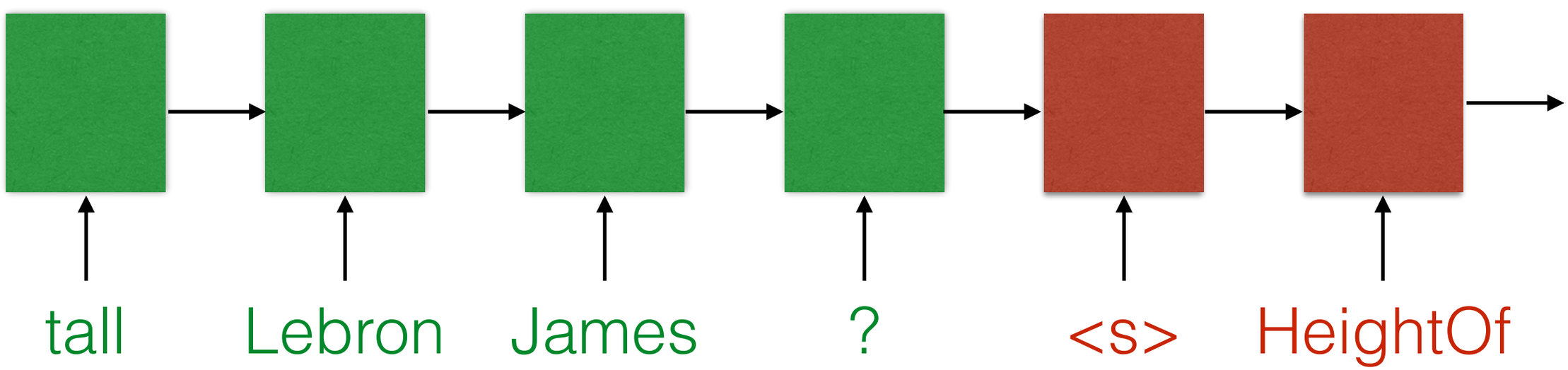
$$\pi(a_t \mid s_t) = \mathrm{softmax}(W^{(s)} h_t)$$



Seq2seq as MDP: policy

$$
\begin{aligned}
p(z \mid x) &= \prod_t p(z_t \mid x, z_0, \ldots, z_{t-1}) \\
&= \prod_t p(a_t \mid x, a_0, \ldots, a_{t-1}) \\
&= \prod_t \pi(a_t \mid s_t)
\end{aligned}
$$

$$\pi(a_t \mid s_t) = \mathrm{softmax}(W^{(s)} h_t)$$



How do we learn?

Option 1: Maximum marginal likelihood

- Our data is language-dentation pairs *(x,y)*
- We obtain y by constructing a logical form *z*
- We can use maximum marginal likelihood by interleaving search and learning:
  - Search to get candidate logical forms
  - Learn from these candidates

Maximum marginal likelihood

- *z* is independent of *x* conditioned on *y*

$$p_\theta(y \mid x) = \sum_z p_\theta(z \mid x) \cdot p(y \mid z)$$

$$= \sum_z p_\theta(z \mid x) R(z) = E_{p_\theta(z|x)}[R(z)]$$

$$\mathcal{L}_{\mathrm{MML}}(\theta) = \log \prod_{(x,y)} p_\theta(y \mid x) = \log \prod_{(x,y)} E_{p_\theta(z|x)}[R(z)]$$

$$= \sum_{(x,y)} \log \sum_z p_\theta(z \mid x) \cdot R(z)$$

Gradient of MML

- Gradient has similar form to what we have seen in the past, except that we are not in a log-linear model. Let's assume a binary reward:

$$\nabla_\theta \log \sum_z p_\theta(z \mid x) \cdot R(z) = \sum_z \frac{p_\theta(z \mid x) R(z) \nabla \log p_\theta(z \mid x)}{\sum_z' p_\theta(z' \mid x) \cdot R(z')}$$

$$= \sum_z p(z \mid x, R(z) = 1) \nabla \log p_\theta(z \mid x)$$

- Compute the gradient of the log probability for every logical form, and weight the gradient using the reward.

Computing the gradient

- We can not enumerate all of the logical forms
- Instead we perform beam search as usual and get a beam *Z* containing *K* logical forms.
- We imagine that this beam is the entire set of possible logical forms

$$\sum_{z \in Z} p(z \mid x, R(z) = 1) \nabla \log p_\theta(z \mid x)$$

- For every *z* we can compute the gradient of *log p(z | x)* since this is now the usual seq2seq setup.

Option 2: policy gradient

- We would like to simply maximize our expected reward

$$E_{p_\theta(z|x)}[R(z)] = \sum_z p_\theta(z \mid x) R(z)$$

$$\mathcal{L}_{\mathrm{RL}}(\theta) = \sum_{(x,y)} \sum_z p_\theta(z \mid x) R(z) = \sum_{(x,y)} E_{p_\theta(z|x)}[R(z)]$$

$$\nabla \mathcal{L}_{\mathrm{RL}}(\theta) = \sum_{(x,y)} \sum_z p_\theta(z \mid x) R(z) \nabla \log p_\theta(z \mid x)$$

$$= \sum_{(x,y)} E_{p_\theta(z|x)}[R(z) \nabla \log p_\theta(z \mid x)]$$

- Weight the gradient by the product of the reward and the model probability

Computing the gradient

- Again, we can not sum over all logical forms
- But the gradient for every example is an expectation over a distribution we can sample from!
- So we can sample many logical forms, compute the gradient and sum them weighted by the product of the model probability and reward
- Again, for every sample this is regular seq2seq and we can compute an approximate gradient

Some differences

- Using MML with beam search is a biased estimator and has less exploration - we only observe the approximate top-K logical forms

- Using RL could be harder to train. If we have a correct logical form z* that has low probability at the beginning of training, then the contribution to the gradient would be very smaller and it would be hard to bootstrap.

Summary

- Training with a seq2seq model with weak supervision is problematic because the loss function is not a differentiable function of the input
- We saw both MML and RL approaches for getting around that
- In both we find a set of logical forms, compute the gradient for them like in supervised learning, and weight them in some way to form a final gradient
- This let's us train with SGD
- It is still often hard to train