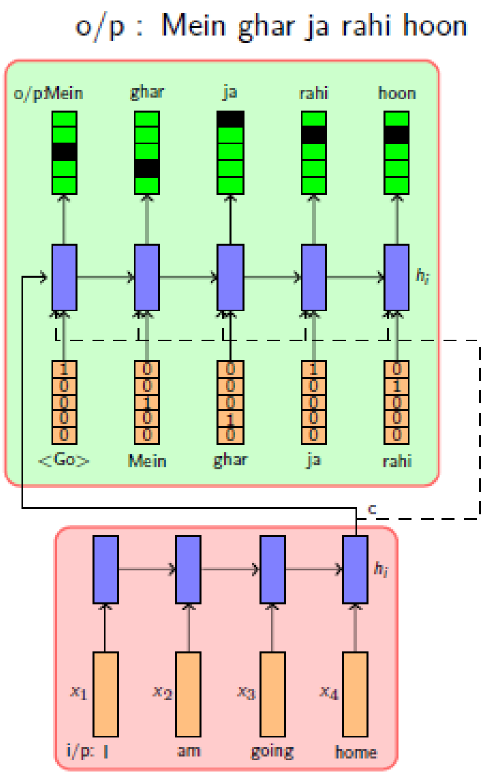
RNN

Attention mechanism

EE5179: Deep learning for Image Processing

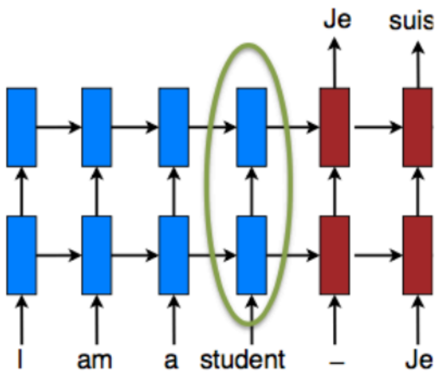
Attention for Long Sequences



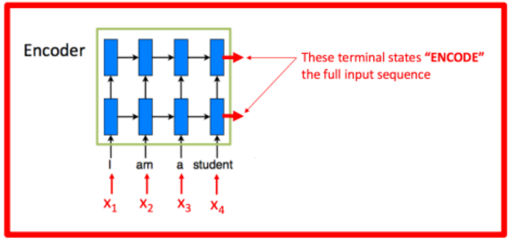
- Let us revisit the decoder that we have seen so far
- We either feed in the encoder information only once(at s_0)
- Or we feed the same encoder information at each time step
- Now suppose an oracle told you which words to focus on at a given time-step t
- Can you think of a smarter way of feeding information to the decoder?

Attention for Long Sequences

The "stuffing" problem



Recall...



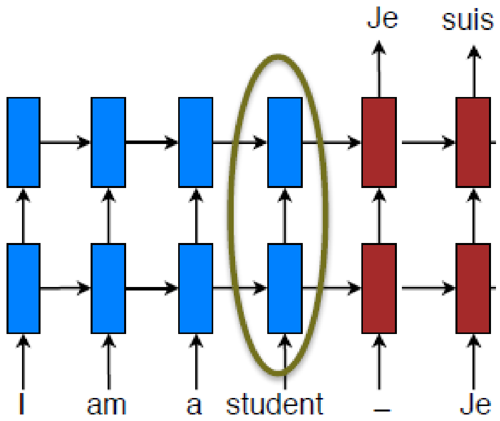
~~Translation~~ quality degrades with long sentences.

Problem: sentence meaning is represented by a fixed-dimensional vector.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Introducing Attention

- Encoder decoder models can be made even more expressive by adding an “attention” mechanism
- We will first motivate the need for this and then explain how to model it.



Problem: fixed-dimensional representation Y

Introducing Attention

The "stuffing" problem

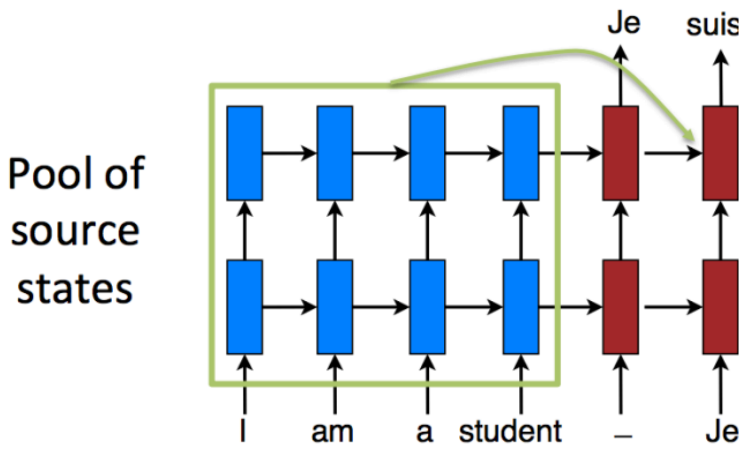
Attention Mechanism

i/p : I am going home
t_1 : [1 0 0 0 0]
t_2 : [0 0 0 0 1]
t_3 : [0 0 0.5 0.5 0]
t_4 : [0 1 0 0 0]
o/p : Main ghar ja rahi hoon

- Humans try to produce each word in the output by focusing only on certain words in the input
- Essentially at each time step we come up with a distribution on the input words
- This distribution tells us how much attention to pay to each input words at each time step
- Ideally, at each time-step we should feed only this relevant information (i.e. encodings of relevant words) to the decoder

Attention for Long Sequences

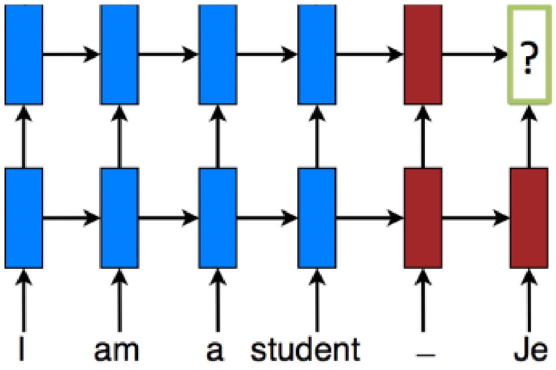
Allow random access to all the states



- Solution: random access memory
  - Retrieve as needed.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences



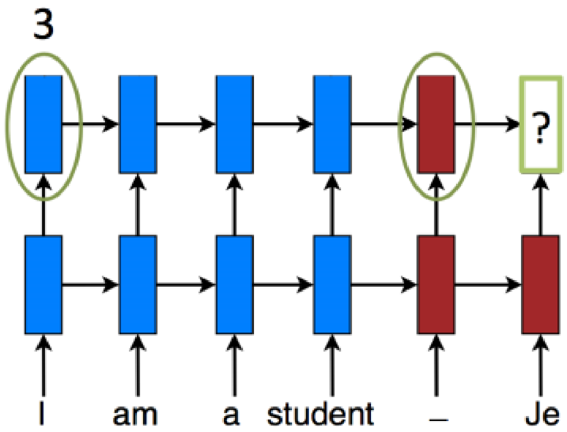
A simplified version of (Bahdanau et al., 2015)

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

Scoring

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



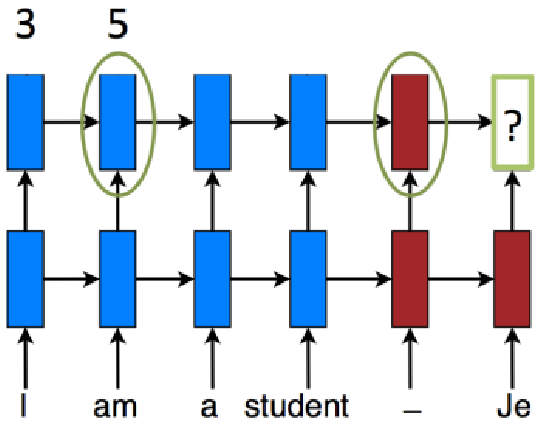
- Compare target and source hidden states.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

Scoring

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$

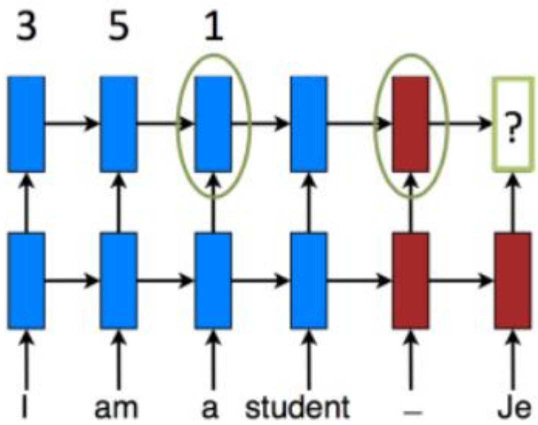


- Compare target and source hidden states.

Attention for Long Sequences

Scoring

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



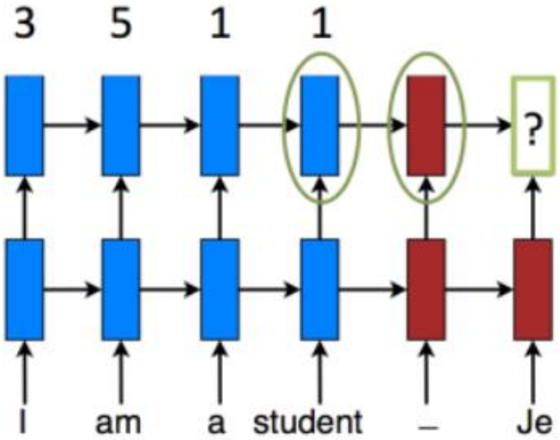
- Compare target and source hidden states.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

Scoring

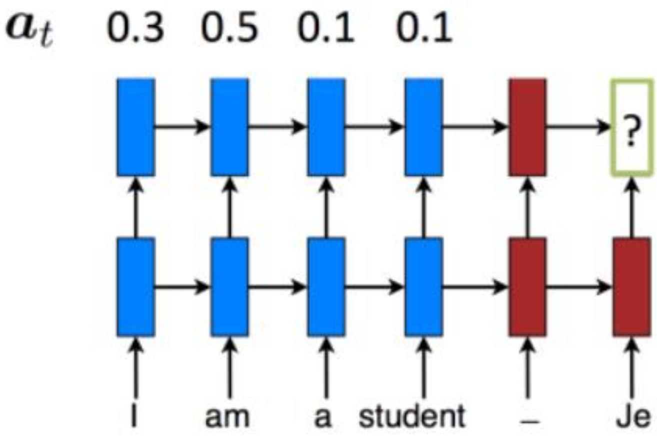
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$



- Compare target and source hidden states.

Attention for Long Sequences

Score Normalization → Alignment Vectors (visualized later)

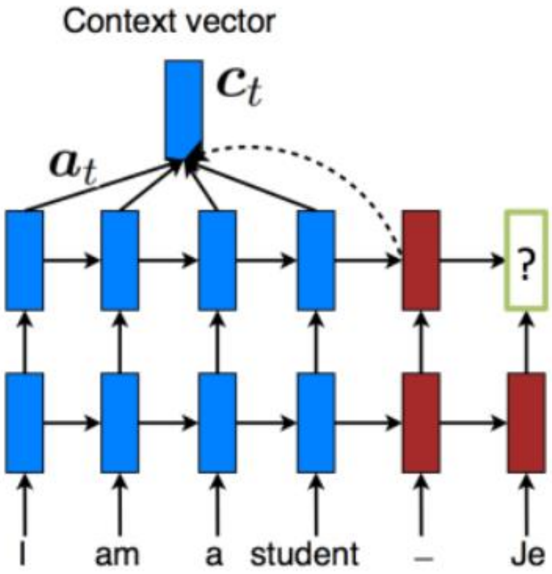


- Convert into alignment weights.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

Context Vector

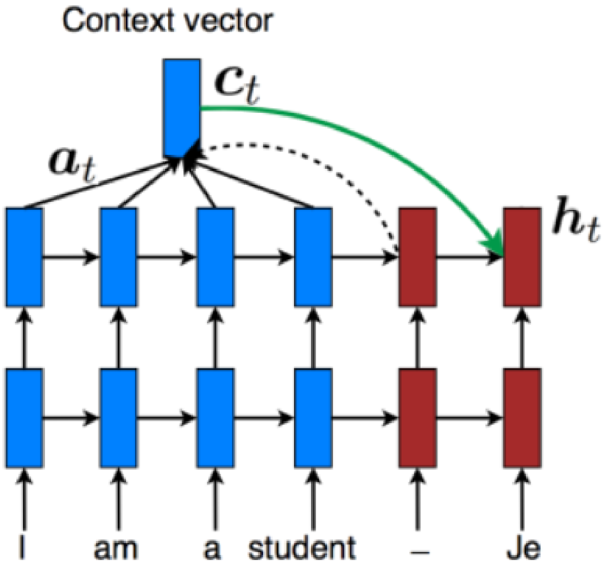


- Build context vector: weighted average.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

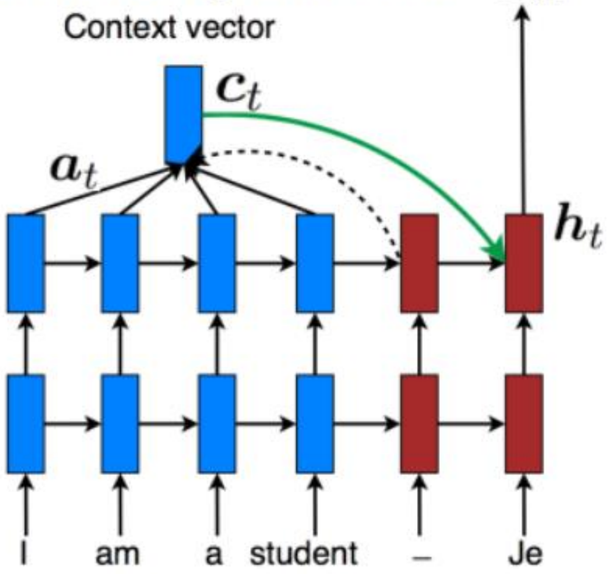
Hidden State / Push through RNN Cell



- Compute the next hidden state.

Attention for Long Sequences

Hidden State / Push through RNN Cell



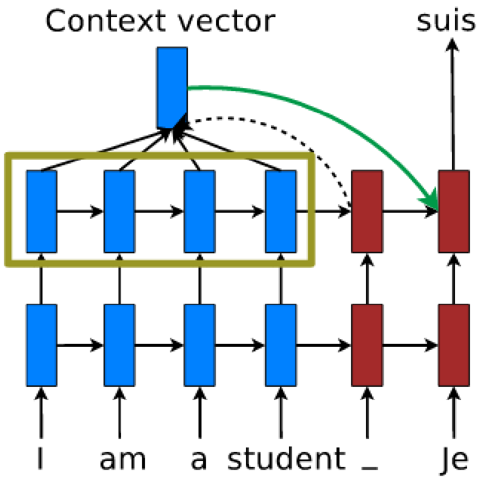
$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) \end{cases}$$

- Predict the next word.

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention for Long Sequences

- Simplified mechanism & more functions:

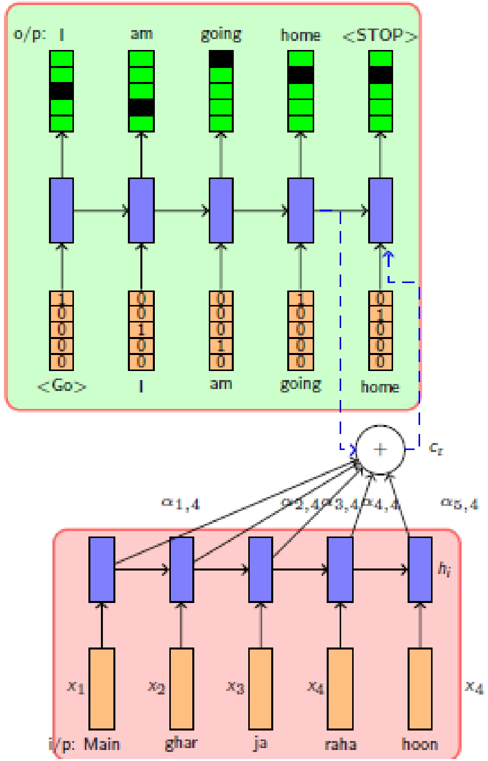


Bilinear form: well-adopted.

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t ; \bar{\boldsymbol{h}}_s]\right) \end{cases}$$

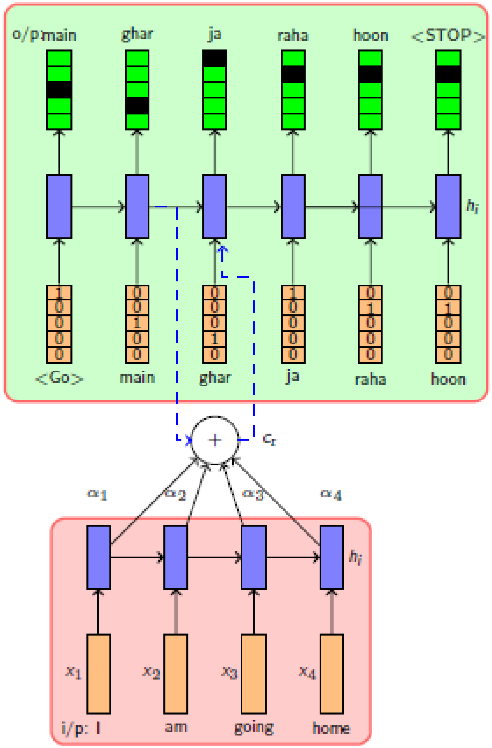
GitHub, Inc. [US] https://github.com/harvardnlp/seq2seq-attn

Attention Mechanism



- We could just take a weighted average of the corresponding word representations and feed it to the decoder
- For example at timestep 3, we can just take a weighted average of the representations of 'ja' and 'rahi'
- Intuitively this should work better because we are not overloading the decoder with irrelevant information (about words that do not matter at this time step)
- How do we convert this intuition into a model ?

Attention Mechanism



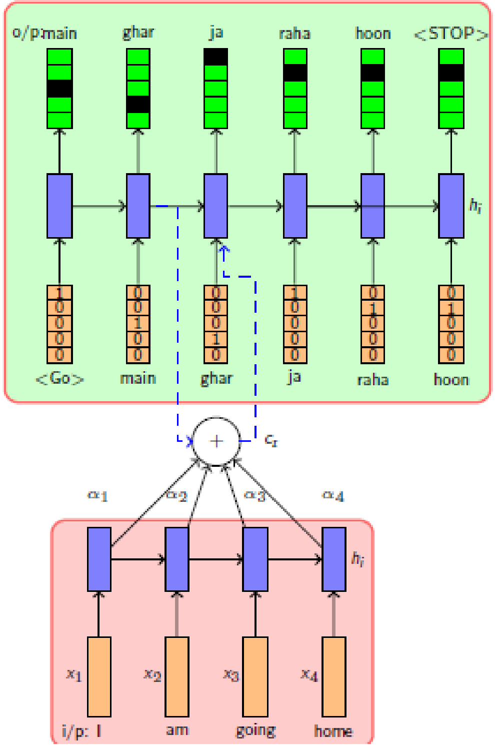
- Of course in practice we will not have this information given to us.
- The machine will have to learn this from the data
- To enable this we define a function

$$e_{jt} = f_{ATT}(h_{t-1}, c_j)$$

- This quantity captures the importance of the j^{th} input word for decoding the t^{th} output word (we will see the exact form of f_{ATT} later)
- We can normalize these weights by using the softmax function

$$\alpha_{jt} = \frac{exp(e_{jt})}{\sum_{j=1}^{M} exp(e_{jt})}$$

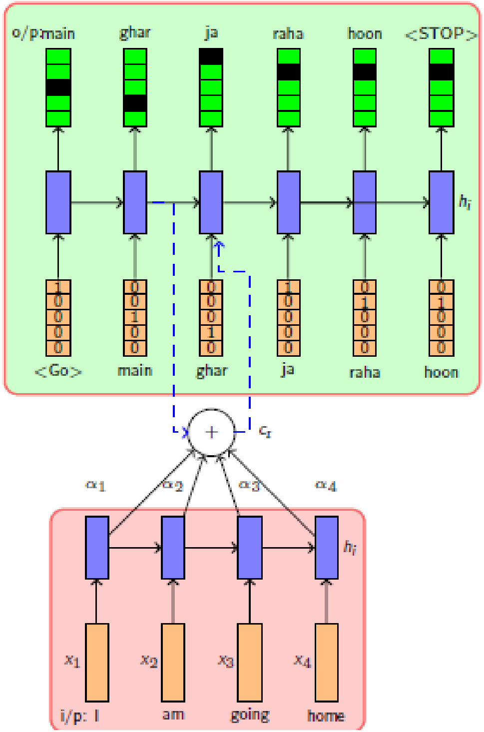
Attention Mechanism



- α_{jt} denotes the probability of focusing on the j^{th} word to produce the t^{th} output word
- We are now trying to learn the α's instead of an oracle informing us about the α's
- Learning would always involve some parameters
- So let's define a parametric form for α's

(just one of the ways of formulating the concept of attention)

Attention Mechanism

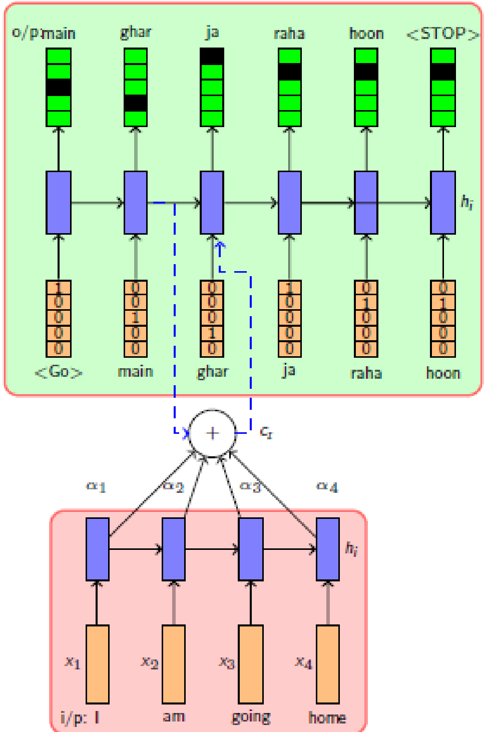


- α_{jt} denotes the probability of focusing on the j^{th} word to produce the t^{th} output word
- Given these new notations, one (among many) possible choice for f_{ATT} is

$$e_{jt} = V_{att}^T \tanh(U_{att} h_{t-1} + W_{att} c_j)$$

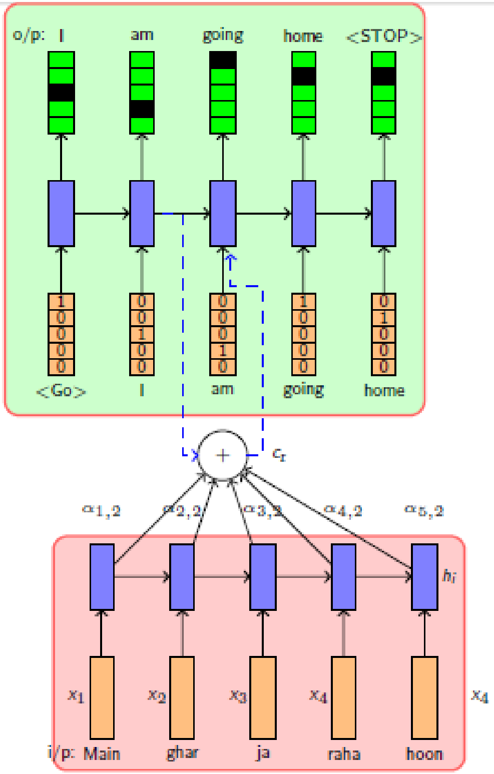
- $V_{att} \in \mathbb{R}^d$, $U_{att} \in \mathbb{R}^{d \times d}$, $W_{att} \in \mathbb{R}^{d \times d}$ are additional parameters of the model
- These parameters will be learned along with the other parameters of the encoder and decoder

Attention Mechanism



- It works because it is a better modeling choice
- This is a more informed model
- We are essentially asking the model to approach the problem in a better (more natural) way
- Given enough data it should be able to learn these attention weights just as humans do
- That's the hope (and hope is a good thing)
- And in practice indeed these models work better than the vanilla encoder decoder models

Attention Mechanism



- Data: $\{x_i = source_i,\ y_i = target_i\}_{i=1}^{N}$
- Encoder:

$$h_t = RNN(h_{t-1}, x_t)$$
$$s_0 = h_T$$

- Decoder:

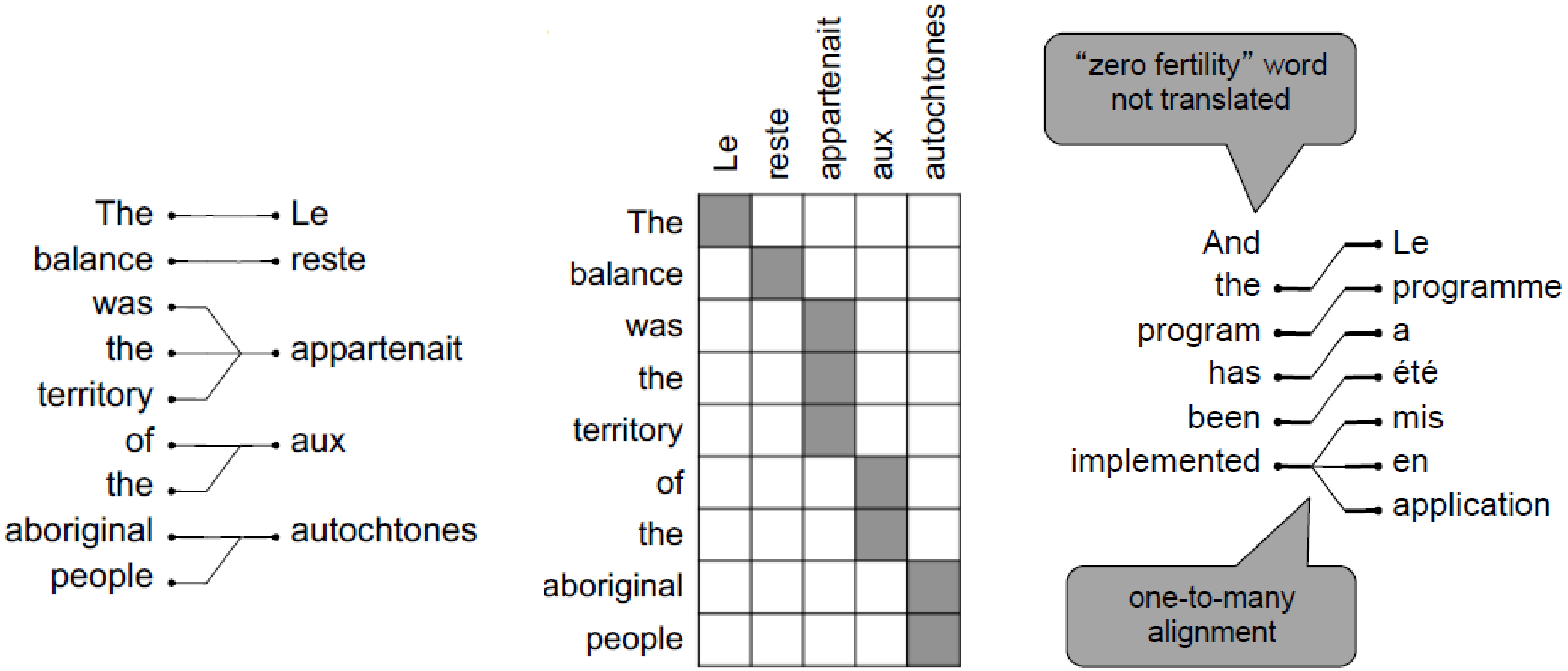
$$e_{jt} = V_{attn}^{T} tanh(U_{attn} h_j + W_{attn} s_t)$$
$$\alpha_{jt} = softmax(e_{jt})$$
$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$
$$s_t = RNN(s_{t-1}, [e(\hat{y}_{t-1}), c_t])$$
$$\ell_t = softmax(V s_t + b)$$

- Parameters: $U_{dec},\ V,\ W_{dec},\ U_{enc},\ W_{enc},\ b,\ U_{attn},\ V_{attn}$
- Loss and Algorithm remains same

Attention Mechanism

- Can we check if the attention model actually learns something meaningful ?
- In other words does it really learn to focus on the most relevant words in the input at the time-step t?
- We can check this by plotting the attention weights as a heatmap (we will see some examples on the next slide)

Input-Output Alignment



Input-Output Alignment

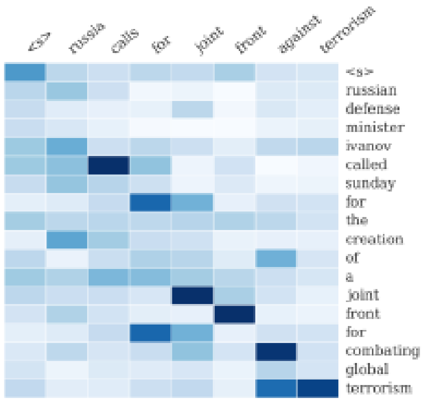


Figure: Example output of attention-based summarization system [Rush et al. 2015.]

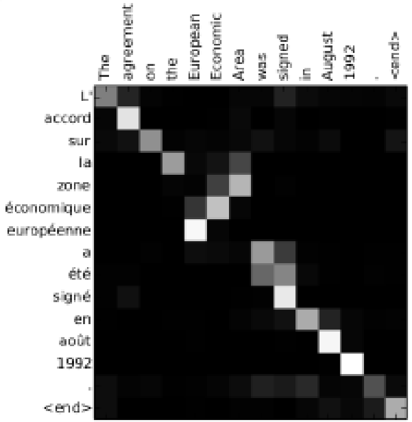
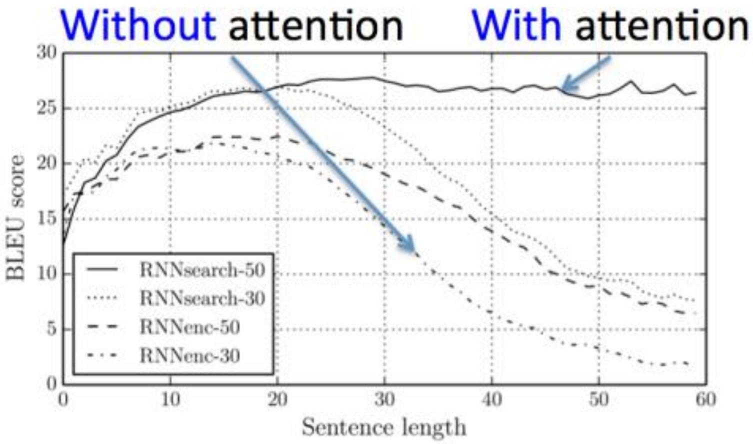


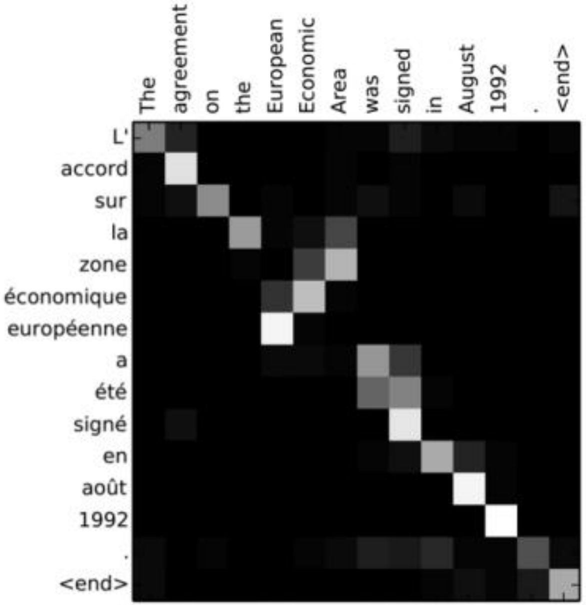
Figure: Example output of attention-based neural machine translation model [Cho et al. 2015].

- The heat map shows a soft alignment between the input and the generated output.
- Each cell in the heat map corresponds to α_{tj} (i.e., the importance of the j^{th} input word for predicting the t^{th} output word as determined by the model)

Better Translation of long sentences

Soft Attention



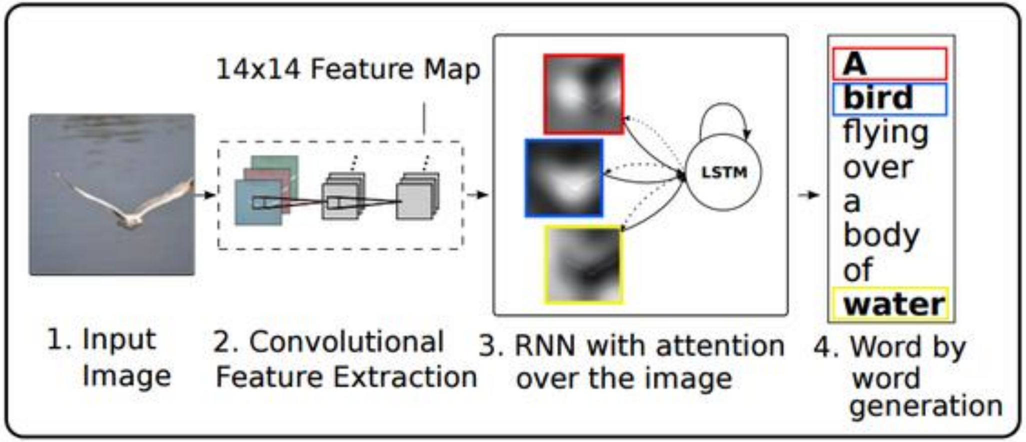


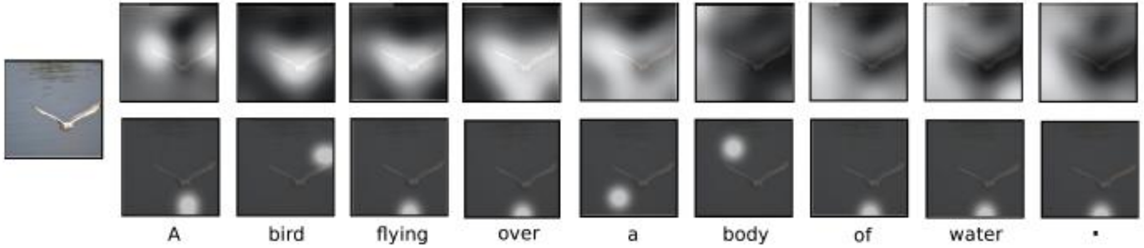
Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. ***Neural Machine Translation by Jointly Learning to Translate and Align****. ICLR 2015.*

Slide adapted from Thang Luong's Stanford CS224d lecture 15

Attention in Images

The idea of coverage





How to not miss an important image patch?

Xu, Ba, Kiros, Cho, Courville, Salakhutdinov, Zemel, Bengio.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML'15

Results


A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.
Two hockey players are fighting over the puck.
A little girl in a pink hat is blowing bubbles.
A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.
A close up of a cat laying on a couch.
A red motorcycle parked on the side of the road.
A yellow school bus parked in a parking lot.