Advanced RNNs

NLP
Andreas Marfurt

Information Technology
14.03.2025

Overview

- Bidirectional RNNs
- Using RNNs for
  - Classification
  - Generation
  - Sequence-to-sequence tasks
- Attention in RNNs
- Contextual Word Embeddings: ELMo

Advanced RNN Topics

Bidirectionality

- In RNNs, we go through our input sequence from left to right
  - For right-to-left (RTL) languages, the input sequence is simply reversed
- Consider the example sentence:

  “The bank consists of the sides of the river.”

Bidirectionality

- In RNNs, we go through our input sequence from left to right
  - For right-to-left (RTL) languages, the input sequence is simply reversed
- Consider the example sentence:

What type of bank?

"The bank consists of the sides of the river."

Bidirectionality

- In RNNs, we go through our input sequence from left to right
  - For right-to-left (RTL) languages, the input sequence is simply reversed
- Consider the example sentence:

What type of bank?

"The bank consists of the sides of the river."

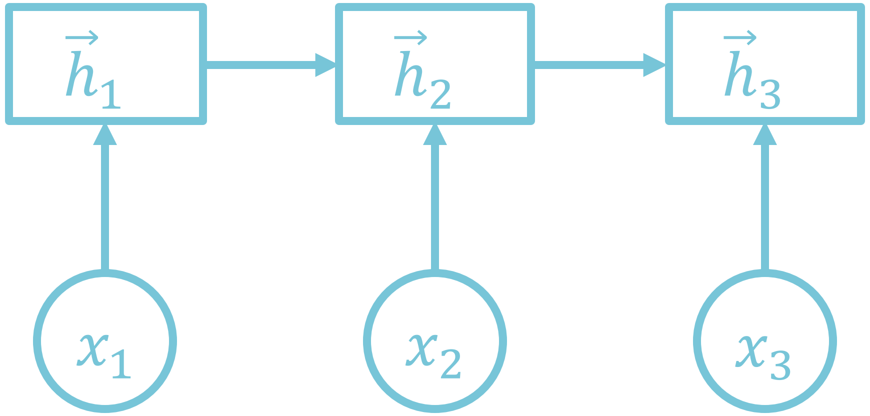
It must be a river bank ← The topic is rivers

Bidirectionality

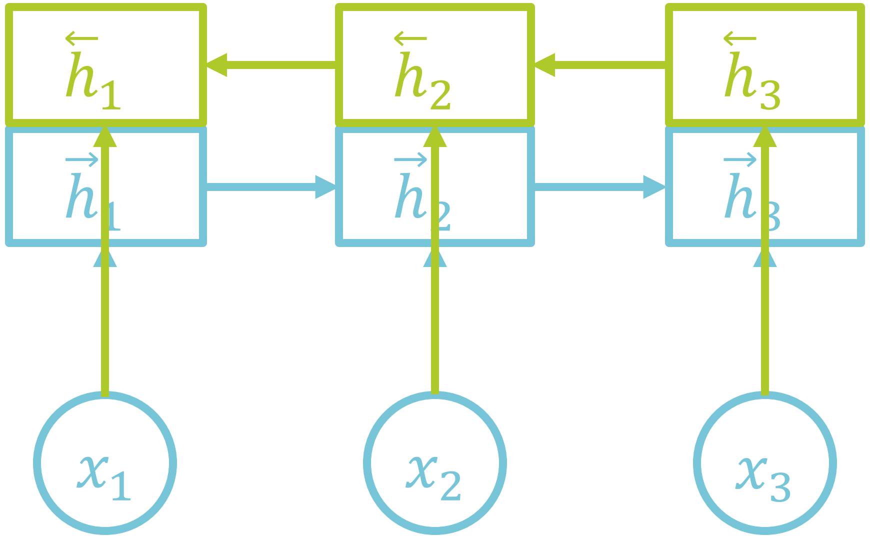
"The bank consists of the sides of the river."

- The right context of bank has information that the left context doesn't have.
- Idea: Use both context directions.

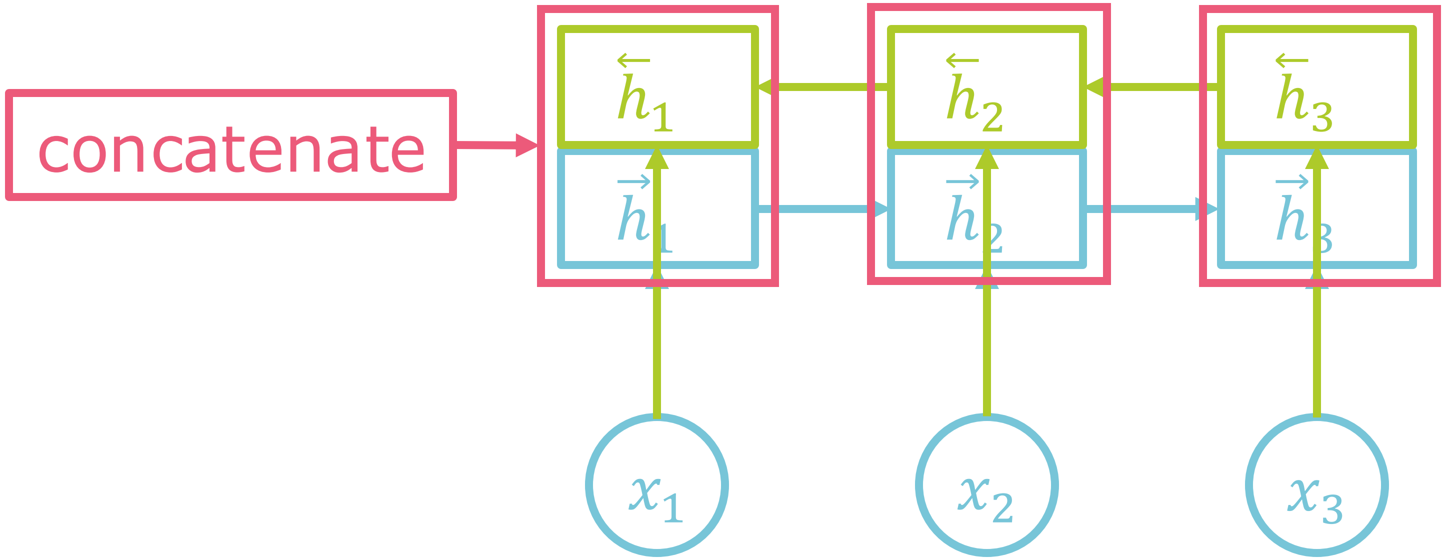
Bidirectional RNN (BiRNN)



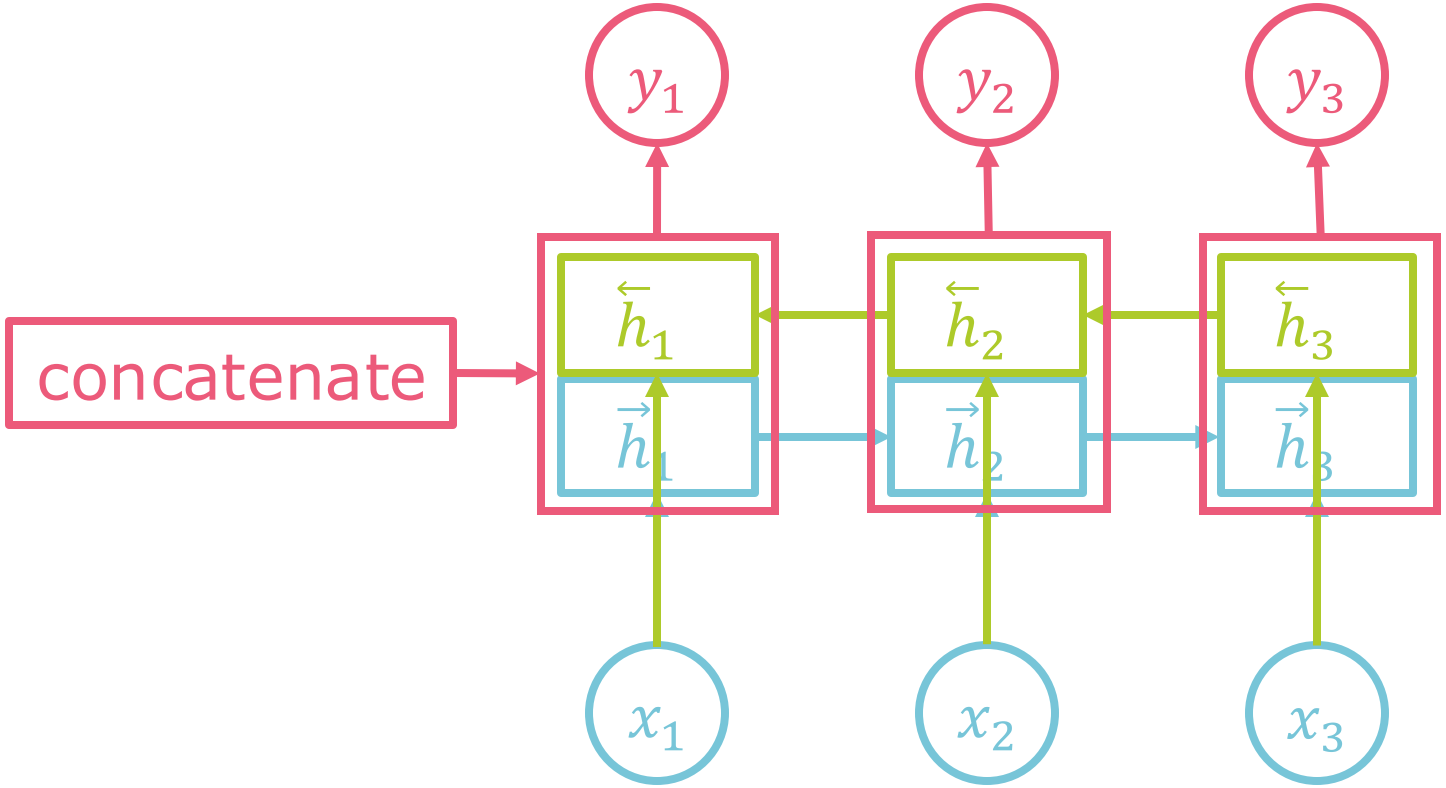
Bidirectional RNN (BiRNN)



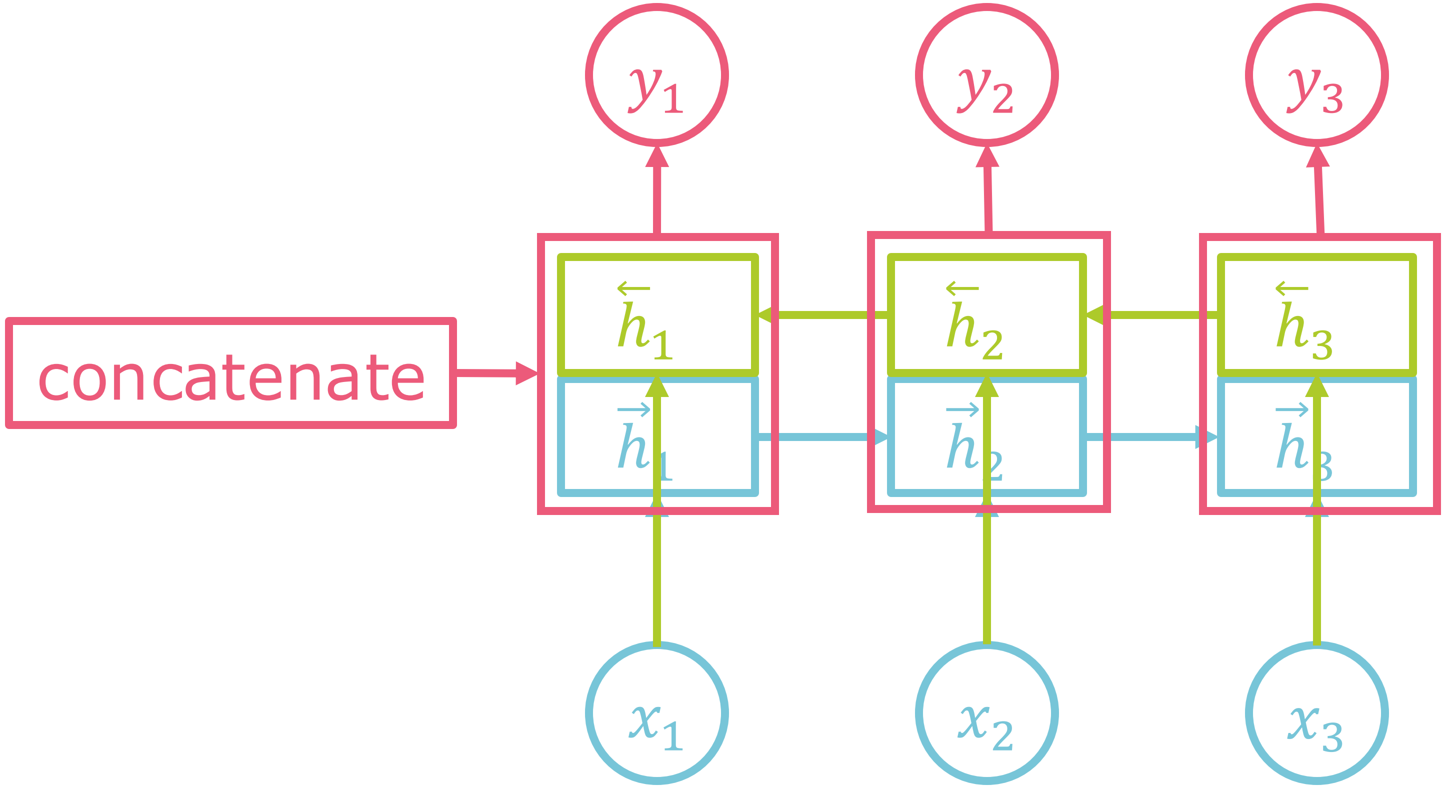
Bidirectional RNN (BiRNN)



Bidirectional RNN (BiRNN)



Bidirectional RNN (BiRNN)



Multiple layers: concatenate $[\overrightarrow{h}_1^1; \overleftarrow{h}_1^1]$, give as input to next layer

BiRNN Definition

- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h)$$
$$y_t = \mathrm{softmax}\left(W_y h_t + b_y\right)$$

- BiRNN:

$$\overrightarrow{h}_t = \sigma\left(\overrightarrow{W}_x x_t + \overrightarrow{W}_h \overrightarrow{h}_{t-1} + \overrightarrow{b}_h\right)$$
$$\overleftarrow{h}_t = \sigma\left(\overleftarrow{W}_x x_t + \overleftarrow{W}_h \overleftarrow{h}_{t-1} + \overleftarrow{b}_h\right)$$
$$y_t = \mathrm{softmax}\left(W_y[\overrightarrow{h}_t; \overleftarrow{h}_t] + b_y\right)$$

BiRNN Definition

- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_h)$$
$$y_t = \mathrm{softmax}\left(W_y h_t + b_y\right)$$

- BiRNN:

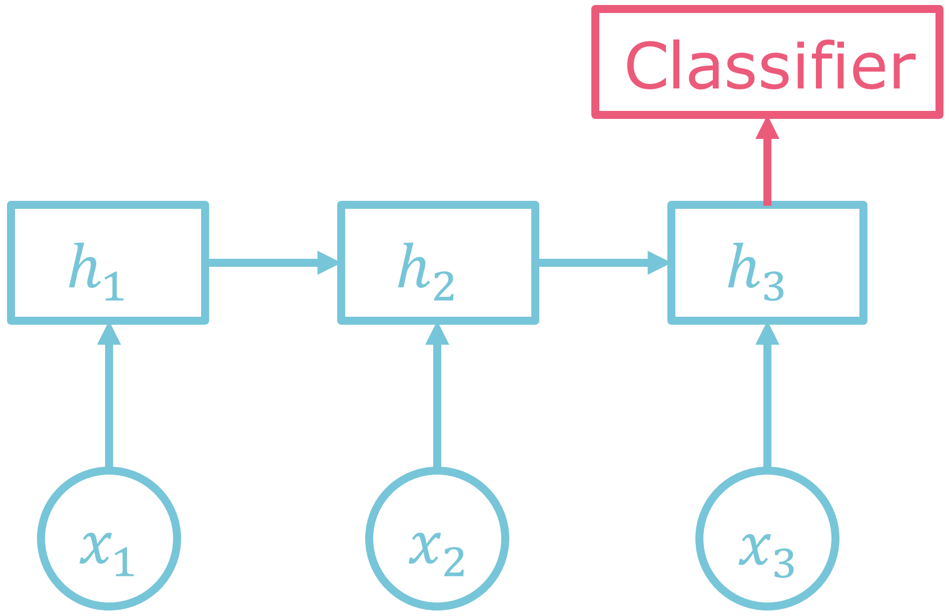
$$\overrightarrow{h}_t = \sigma\left(\overrightarrow{W}_x x_t + \overrightarrow{W}_h \overrightarrow{h}_{t-1} + \overrightarrow{b}_h\right)$$
$$\overleftarrow{h}_t = \sigma\left(\overleftarrow{W}_x x_t + \overleftarrow{W}_h \overleftarrow{h}_{t-1} + \overleftarrow{b}_h\right)$$
$$y_t = \mathrm{softmax}\left(W_y [\overrightarrow{h}_t ; \overleftarrow{h}_t] + b_y\right)$$

Multiple layers:
Input becomes
$[\overrightarrow{h}_1^{(l-1)}; \overleftarrow{h}_1^{(l-1)}]$
(adjust input dim.
to 2x hidden dim.)

Classification

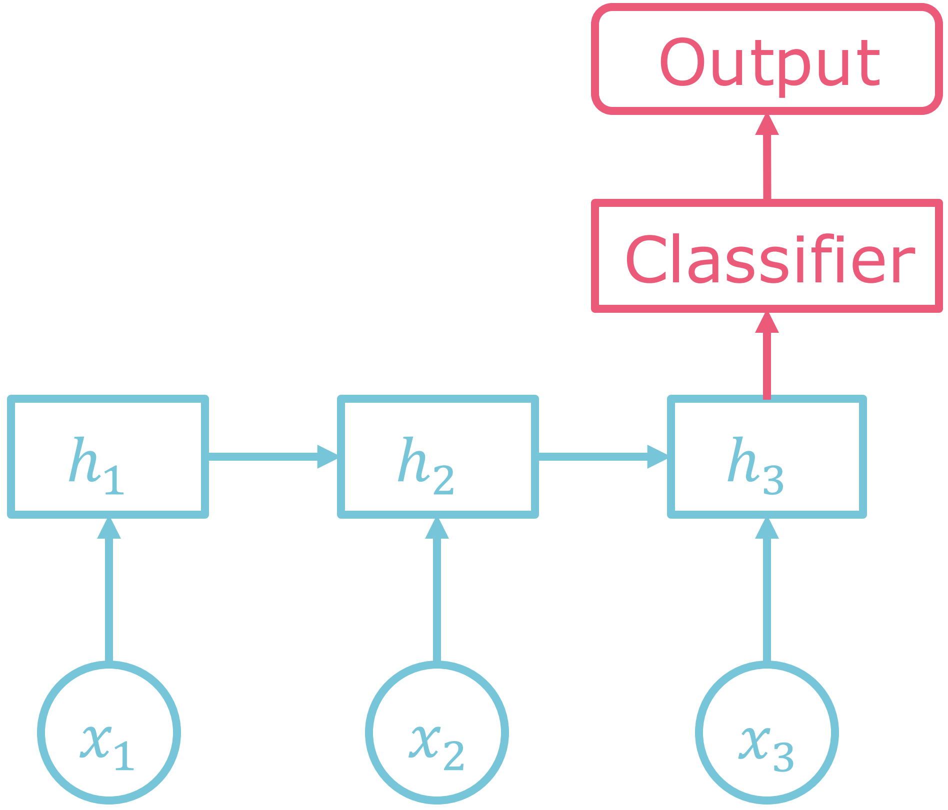
- Final hidden state h_n "encodes" all information of the input sequence

  → Use h_n as a feature for classification



Sequence Classification

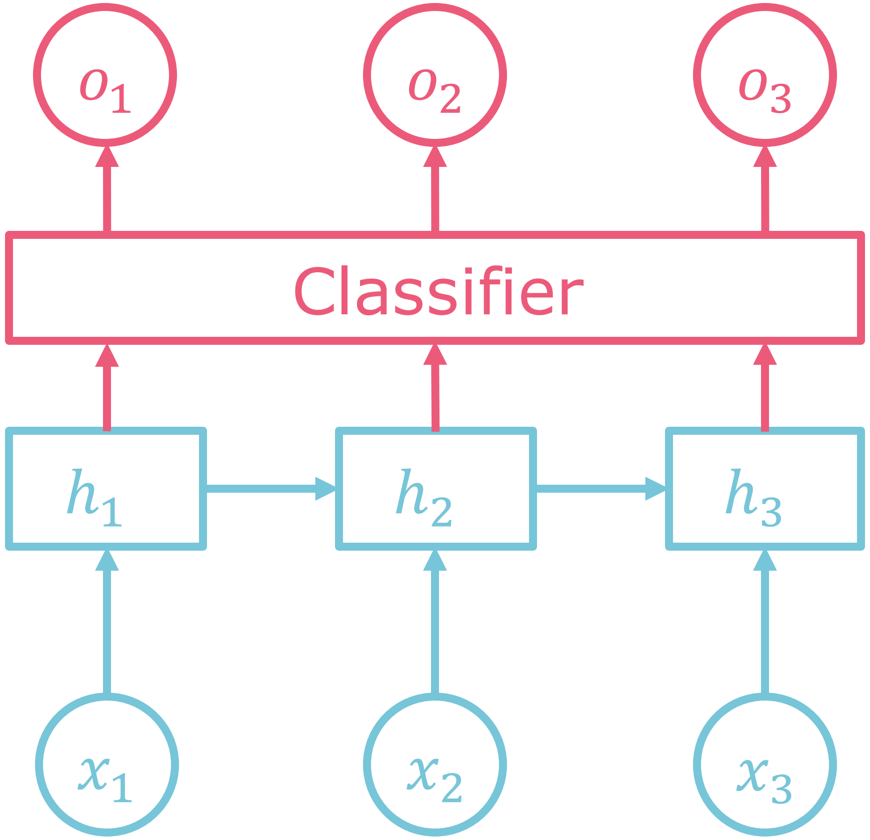
- Classify from h_n:
  - Acceptability: Is this sentence grammatical?
- Compare two sentences (e.g. to determine if one is a paraphrase of the other):
  - Get final hidden states of both sentences: h_n, h'_m
  - Compute similarity (e.g. cosine similarity)



Token Classification

- One decision for each input token
  - PoS tagging
  - Named entity recognition

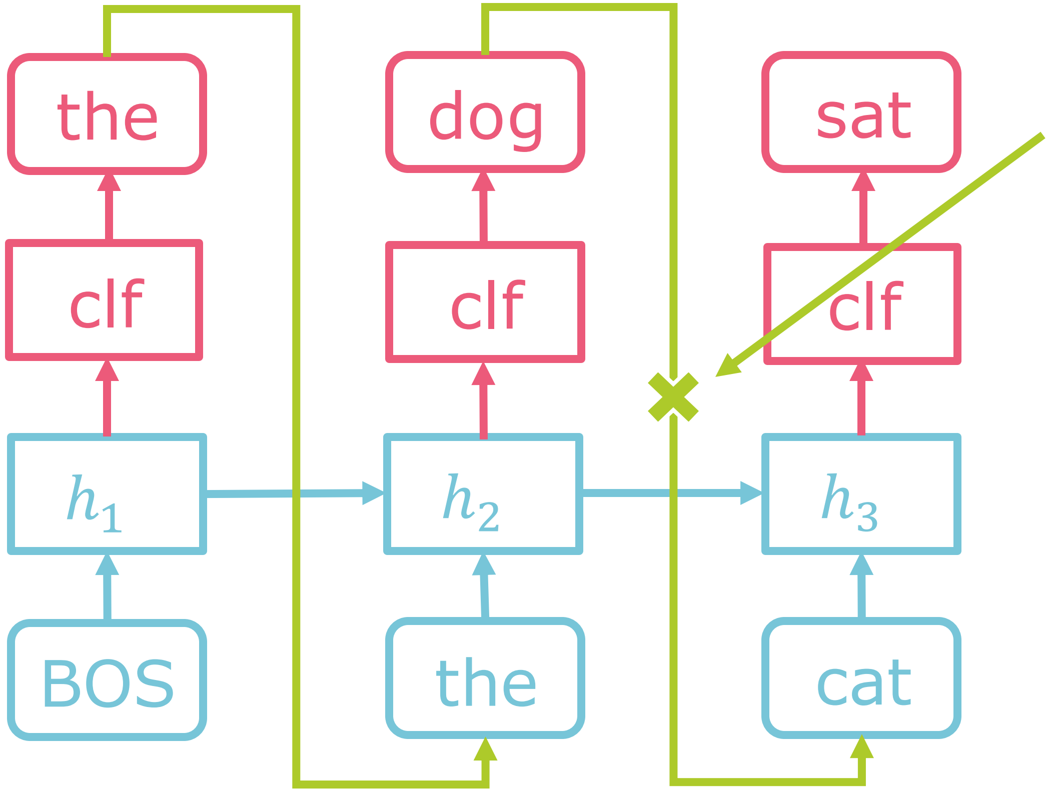
- … or text generation!



Text Generation

- Text generation is also a token classification task:
  - Next word prediction is just a probability distribution over the vocabulary
  - One strategy: Choose the vocabulary item with the highest probability
    - This is called *greedy decoding*
    - We will see later why this isn't always a good idea

Text Generation



Teacher forcing: During training, the true word is given to the network as the next input, irrespective of what it predicted.

Sequence-to-sequence (seq2seq)

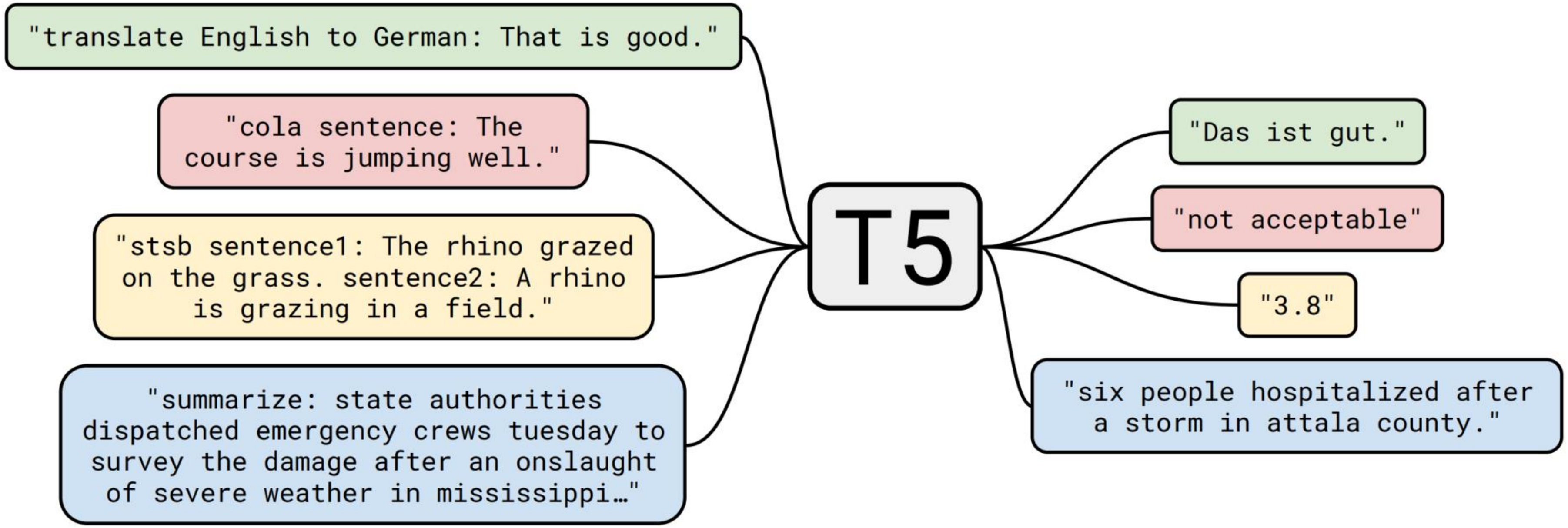
- Sequence-to-sequence tasks are very popular right now
  - Sequence as input
  - Sequence as output
  - Network does the transformation
- Tasks:
  - Machine translation
  - Summarization
  - Question answering
  - … and nearly every task: Just formulate it in a text-to-text format.

Sequence-to-sequence (seq2seq)

- Arithmetic operations
  - Input: 1 + 2 = ?
  - Output: “3” (as a string)

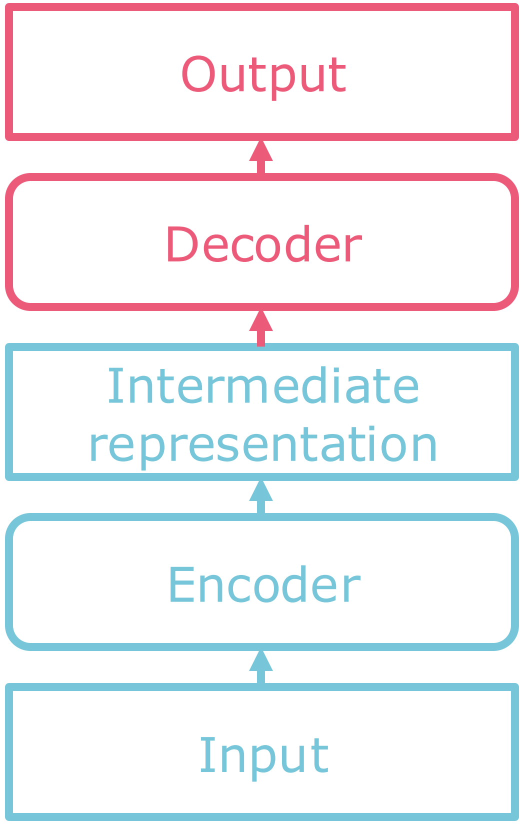
Sequence-to-sequence (seq2seq)

- Arithmetic operations
  - Input: 1 + 2 = ?
  - Output: “3” (as a string)



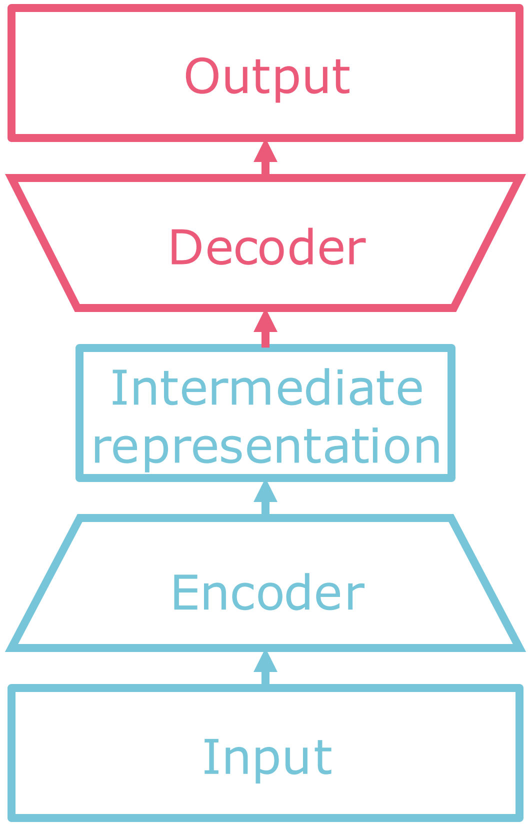
Sequence-to-sequence (seq2seq)

- Encoder:
  - Understands the text
  - Extracts the important information
  - Outputs the important information in a compressed format (= vector, “representation of the input”)
- Decoder:
  - Can read the compressed representation from the encoder
  - Generates the output based on that information



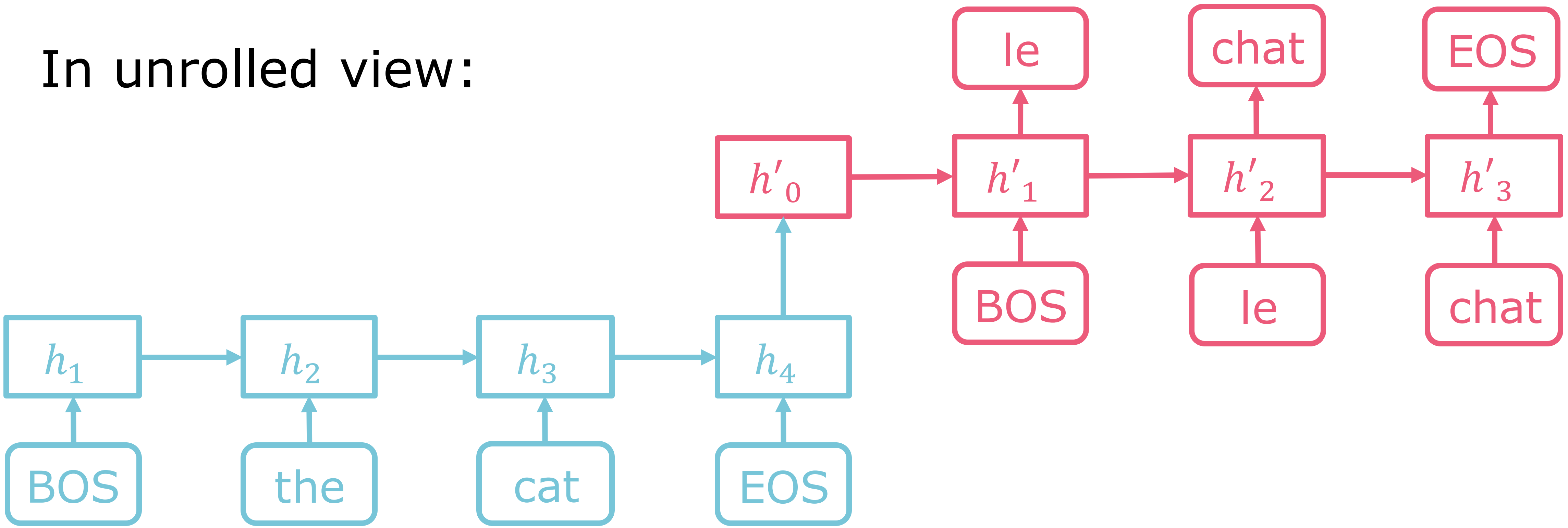
Sequence-to-sequence (seq2seq)

… or more like this:



Sequence-to-sequence (seq2seq)

In unrolled view:



HSLU

Exercise: Seq2seq RNN

Attention in RNNs

Motivation

- Attention is the basis for Transformer (2017)
  - Start of a new era in NLP (and with a bit of delay also in other fields)
- The idea comes up in a machine translation paper written in 2014
- We will also see the first steps in pretraining (2018)
  - … at least in NLP
  - computer vision was already routinely using features from models trained on ImageNet

Human Attention

“Attention is the behavioral and cognitive process of selectively concentrating on a discrete aspect of information, whether considered subjective or objective, while ignoring other perceivable information.”

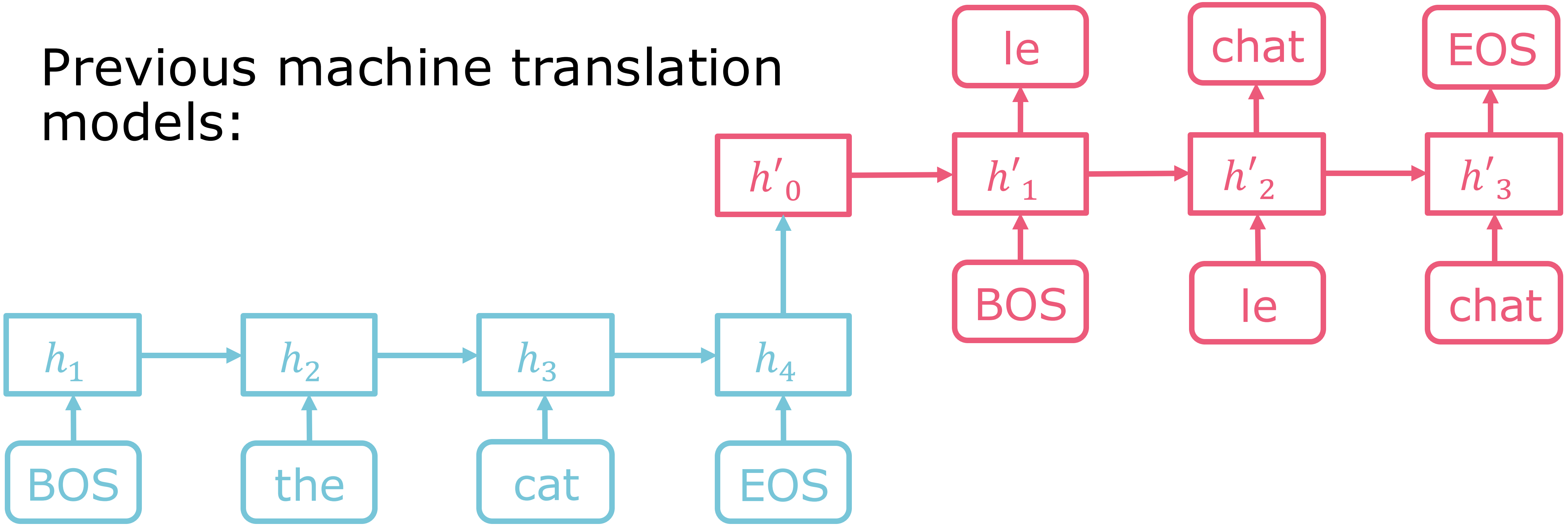
Attention Motivation

- If we just use the final hidden state h_n, don't we throw away a lot of information computed in the earlier parts of the sequence?
  - Especially for long sequences: Information can be removed from the cell/hidden state at every time step t!

- Idea: Use a linear combination of all hidden states as the sequence's representation.

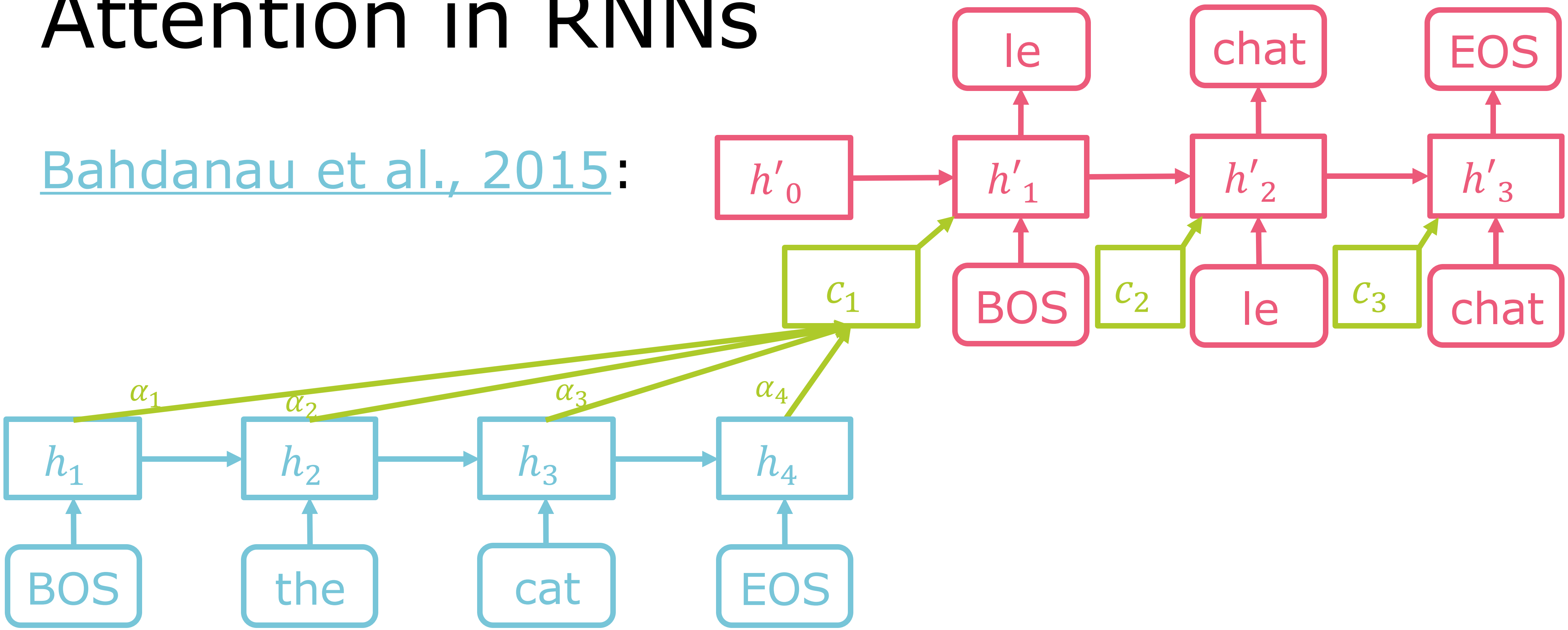
Attention in RNNs

Previous machine translation models:



Attention in RNNs

Bahdanau et al., 2015:



Attention Definition

- Context vector c_i
  - a weighted sum of encoder hidden states
  - computed for each decoding time step

$$c_i = \sum_{j=1}^{n} \alpha_{ij} h_j^{(\mathrm{enc})}$$

i: time step in decoder
j: time step in encoder

- Attention weights α_{ij}

$$\alpha_{ij} = \mathrm{softmax}\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{n} \exp\left(e_{ik}\right)}$$

- Attention function f_{att}

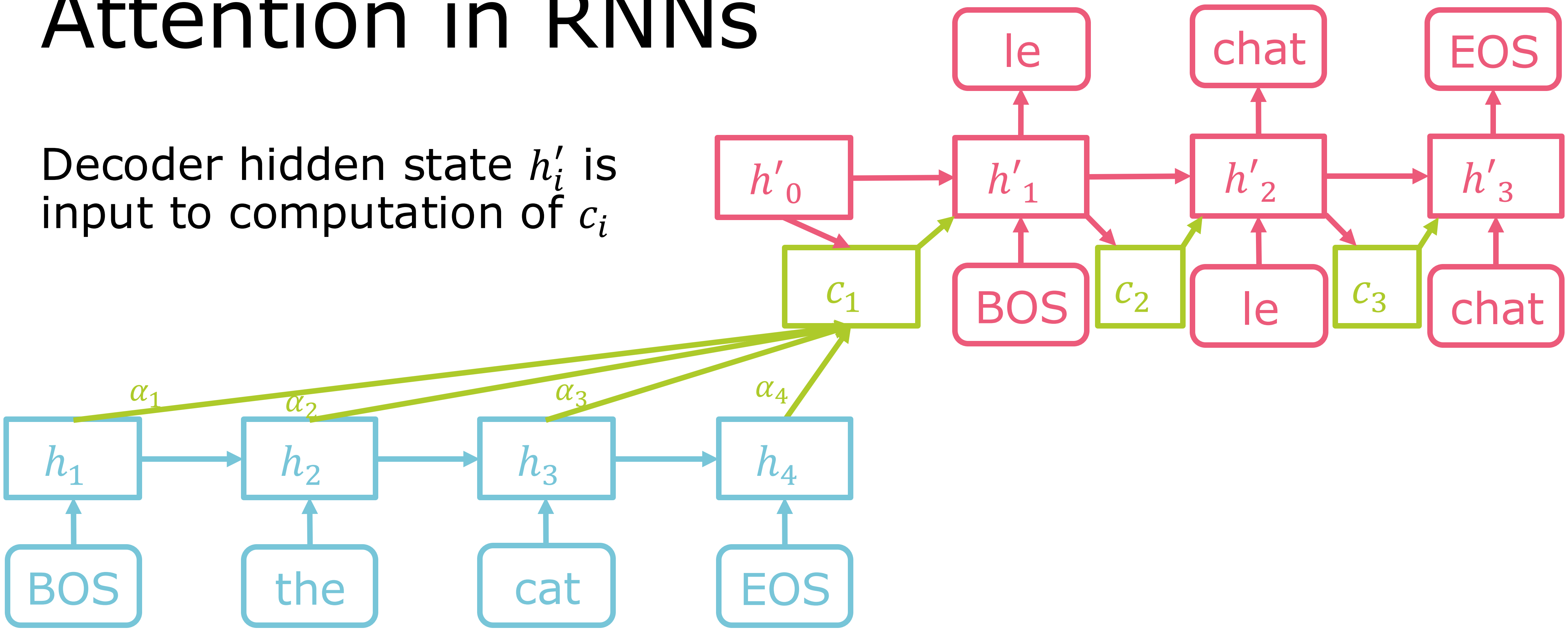
$$e_{ij} = f_{\mathrm{att}}\left(h_{i-1}^{(\mathrm{dec})}, h_j^{(\mathrm{enc})}\right)$$

previous decoder hidden state

current encoder hidden state

Attention in RNNs

Decoder hidden state h_i' is input to computation of c_i



Attention Functions

- Additive attention

$$f_{\text{att}}\left(h_{i-1}^{(\text{dec})}, h_j^{(\text{enc})}\right) = v^{\top} \tanh\left(W h_{i-1}^{(\text{dec})} + U h_j^{(\text{enc})}\right)$$

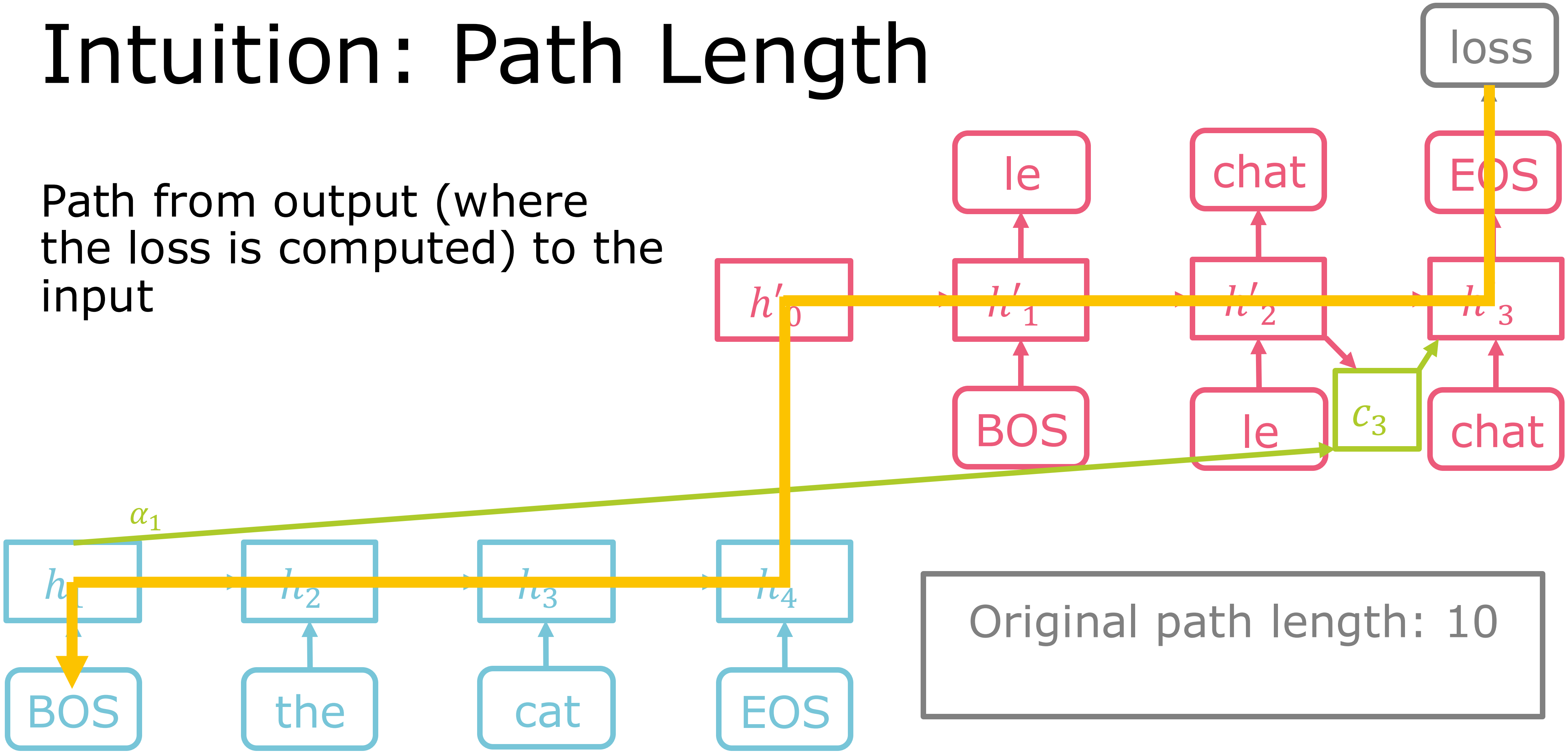
- Multiplicative attention

learnable weights

$$f_{\text{att}}\left(h_{i-1}^{(\text{dec})}, h_j^{(\text{enc})}\right) = h_{i-1}^{(\text{dec})} W h_j^{(\text{enc})}$$

Intuition: Path Length

Path from output (where the loss is computed) to the input

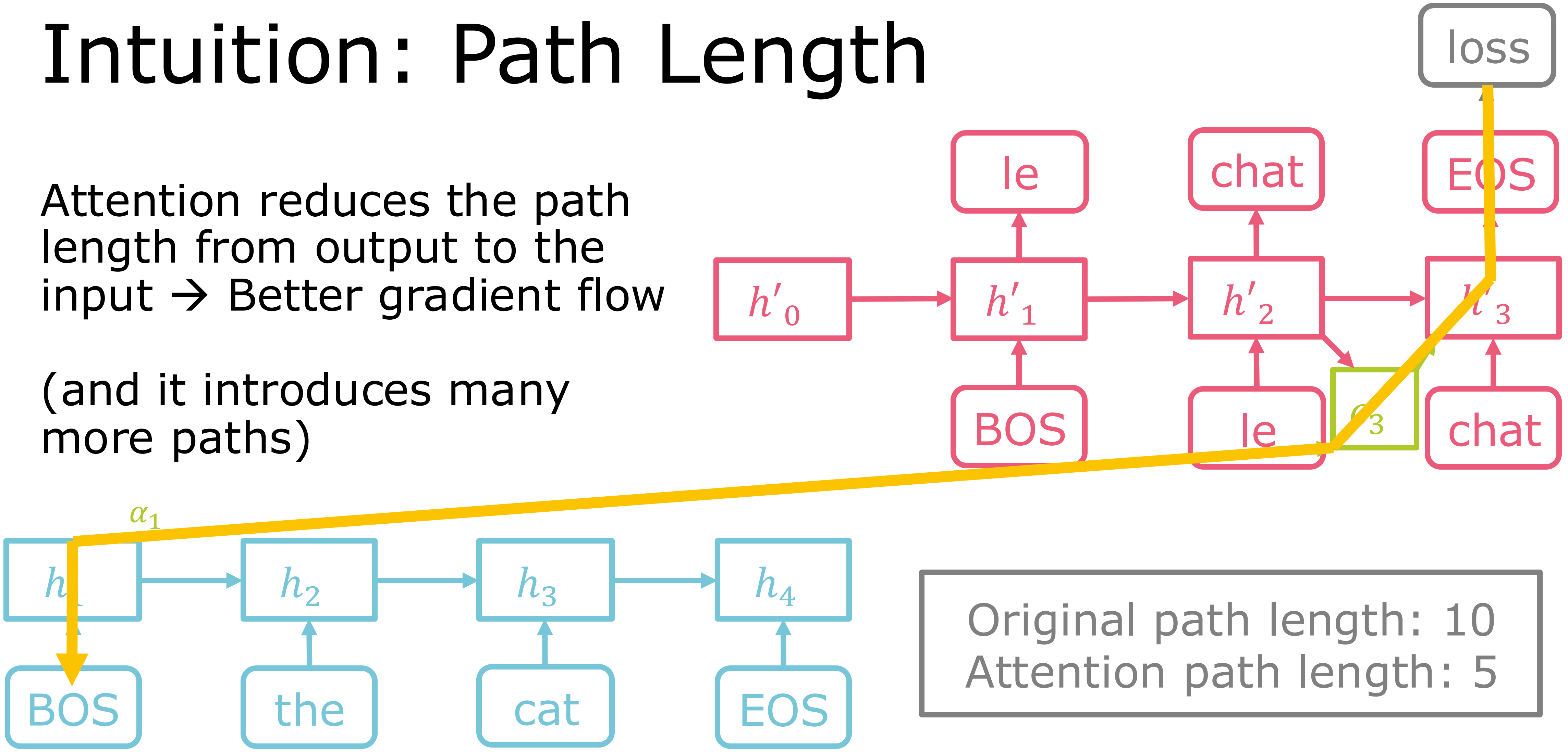


Original path length: 10

Intuition: Path Length

Attention reduces the path length from output to the input → Better gradient flow

(and it introduces many more paths)



Original path length: 10
Attention path length: 5

Exercise: Seq2seq RNN with Attention

Contextual Embeddings: ELMo

Embeddings from Language Models (ELMo)

Deep contextualized word representations

Matthew E. Peters†, **Mark Neumann**†, **Mohit Iyyer**†, **Matt Gardner**†,
{matthewp,markn,mohiti,mattg}@allenai.org

Christopher Clark*, **Kenton Lee***, **Luke Zettlemoyer**†*
{csquared,kentonl,lsz}@cs.washington.edu

†Allen Institute for Artificial Intelligence
*Paul G. Allen School of Computer Science & Engineering, University of Washington

Embeddings from Language Models (ELMo)

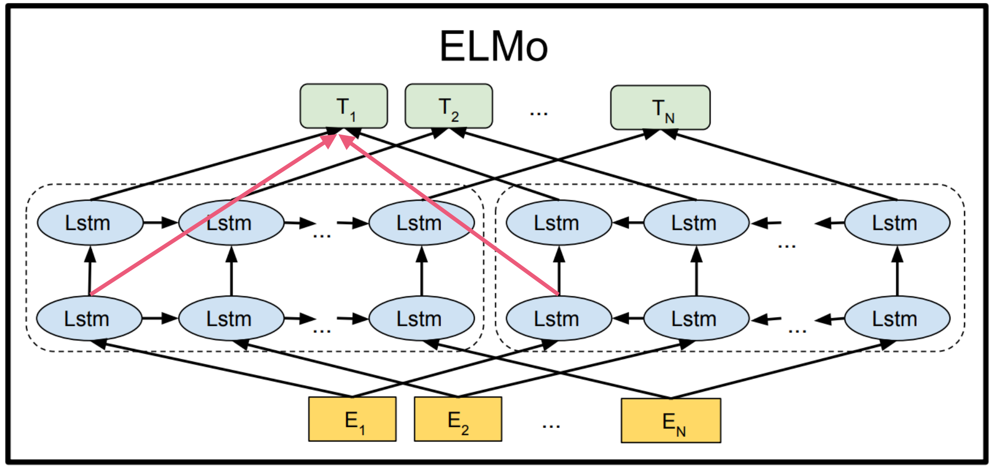
Contextual(ized) Word Embeddings

- Peters et al., 2018
- Idea: Use the hidden states as embeddings of words in context

- word2vec & GloVe: static word embeddings, stays the same in each context
- ELMo: dynamic word embeddings, changes with the context

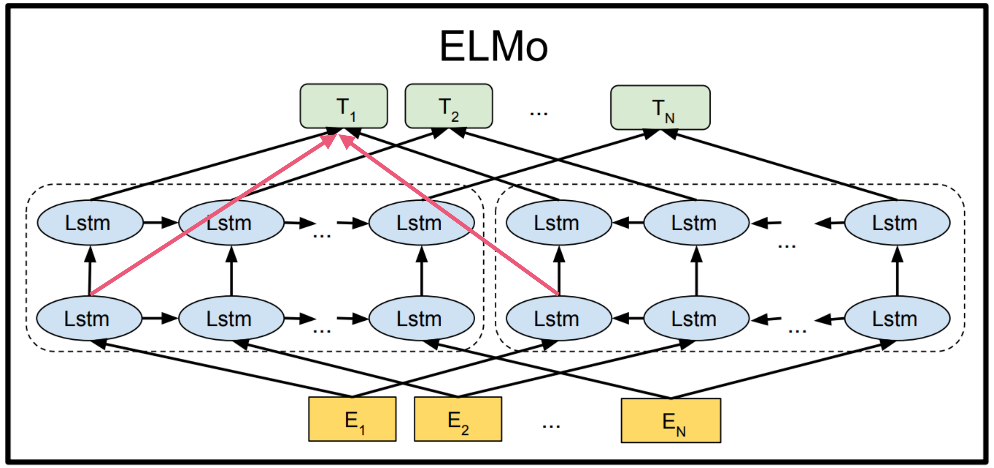
Contextual Word Embeddings

- 2-layer bidirectional LSTM (BiLSTM)
- Word representation: Combination of LSTM’s hidden states
  - Both directions
  - Both layers (imagine red arrows for each T_i in the image)



Contextual Word Embeddings

- 2-layer BiLSTM
  - Layer 1 hidden states: Do well on syntax-related tasks (e.g. part-of-speech tagging)
  - Layer 2 hidden states: Do well on semantic (= meaning) tasks (word sense disambiguation)



Contextual Word Embeddings

- 2-layer BiLSTM
  - Layer 1 hidden states: Do well on syntax-related tasks (e.g. part-of-speech tagging)
  - Layer 2 hidden states: Do well on semantic (= meaning) tasks (word sense disambiguation)

Using both layers performs better than just using layer 2

Task	Baseline	Last Only	All layers λ=1	All layers λ=0.001
SQuAD	80.8	84.7	85.0	**85.2**
SNLI	88.1	89.1	89.3	**89.5**
SRL	81.6	84.1	84.6	**84.8**

Table 2: Development set performance for SQuAD, SNLI and SRL comparing using all layers of the biLM (with different choices of regularization strength λ) to just the top layer.

Recipe for Contextual Embeddings

1. Train a BiLSTM on the 1B Word Benchmark for 10 epochs
2. Freeze the model's weights
3. For each example of your task:
   a) Run the current input through the BiLSTM to get contextual word embeddings
   b) Concatenate with the static word embedding and feed as input to the task network (in ELMo's case a different RNN)

Recipe for Contextual Embeddings

1. Train a BiLSTM on the 1B Word Benchmark for 10 epochs
2. Freeze the model's weights
3. For each example of your task:
   a) Run the current input through the BiLSTM to get contextual word embeddings
   b) Concatenate with the static word embedding and feed as input to the task network (in ELMo's case a different RNN)

Pretraining!

Task-specific Combinations

- We can combine the embeddings of each layer, depending on the task
  → combine their strengths
- ELMo learns a scaling factor for each of
  - static word embedding
  - contextual embedding from L1
  - contextual embedding from L2
- Concatenate with input x_t (“input layer”) or hidden state h_t (“output layer”)

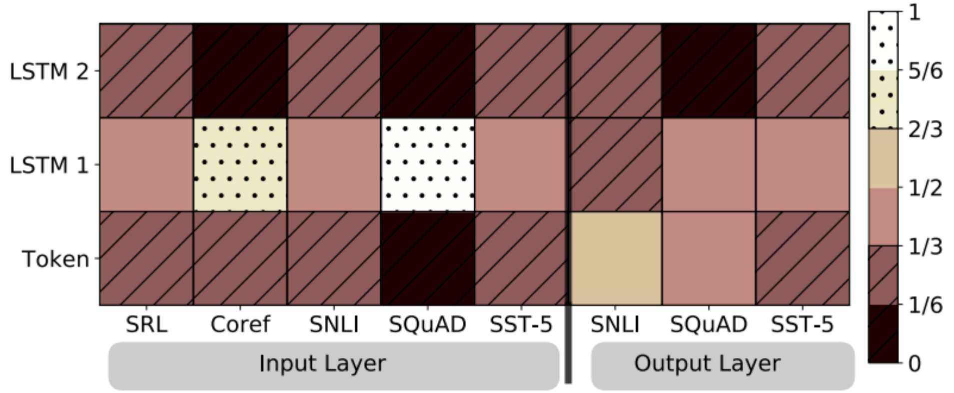


Figure 2: Visualization of softmax normalized biLM layer weights across tasks and ELMo locations. Normalized weights less then 1/3 are hatched with horizontal lines and those greater then 2/3 are speckled.

ELMo Performance

- ELMo improves the state-of-the-art (SOTA) on various tasks

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Nearest Neighbors for Contextual Word Embeddings

- Disambiguation of polysemous words (same word with multiple meanings)

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.