Neural Machine Translation: Attention Model

1/17/2017

JOHNS HOPKINS
UNIVERSITY

Today's Focus

- Neural Machine Translation with Attention
- Specific focus on what you need to know to understand NMT with Attention, as implemented in the Nematus toolkit (used in Edinburgh's WMT'16 submission)

Nematus

If you want to check individual line numbers referenced in these slides, look at this version of the code (most recent at the time of writing): https://github.com/rsennrich/nematus/tree/9b08048e278db46ede288bcca928d176811ee2c2

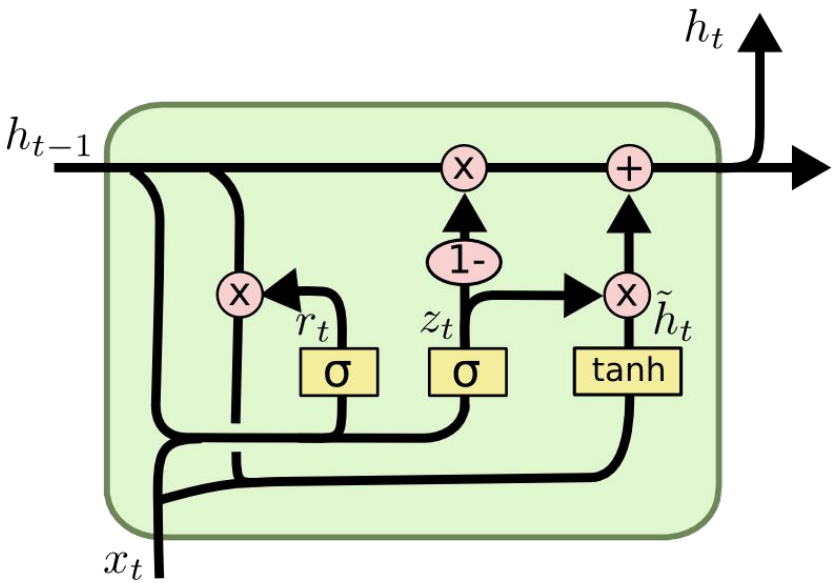
Topics

- GRU
- Basic RNN Encoder-Decoder for NMT
- Attention model
- Other topics
  - Search
  - Ensembling
  - Synthetic data
  - Additional linguistic features
  - Byte-Pair Encoding review

Gated Recurrent Unit (GRU)

- *Reset gate*
  - Determine the tradeoff between previous hidden state and new input.
- *Proposed hidden state*
  - Combine previous hidden state with new input, according to reset gate.
- *Update gate*
  - How much should we update the hidden state? Use the previous hidden state? This proposed one? Some interpolation?

GRU

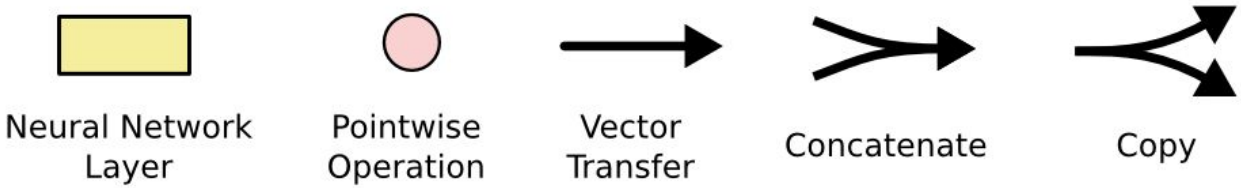


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

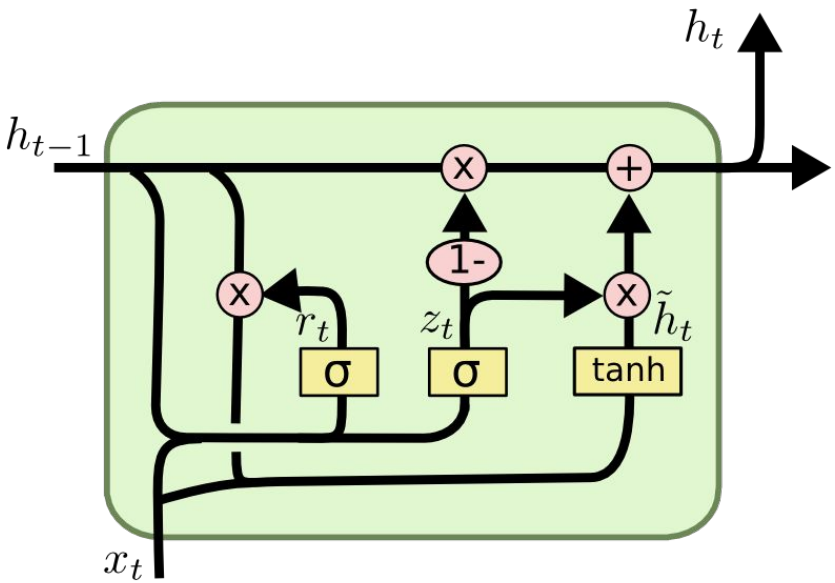
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

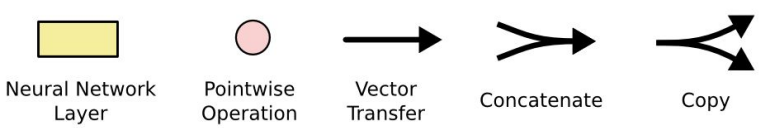


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$\boxed{r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)}$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

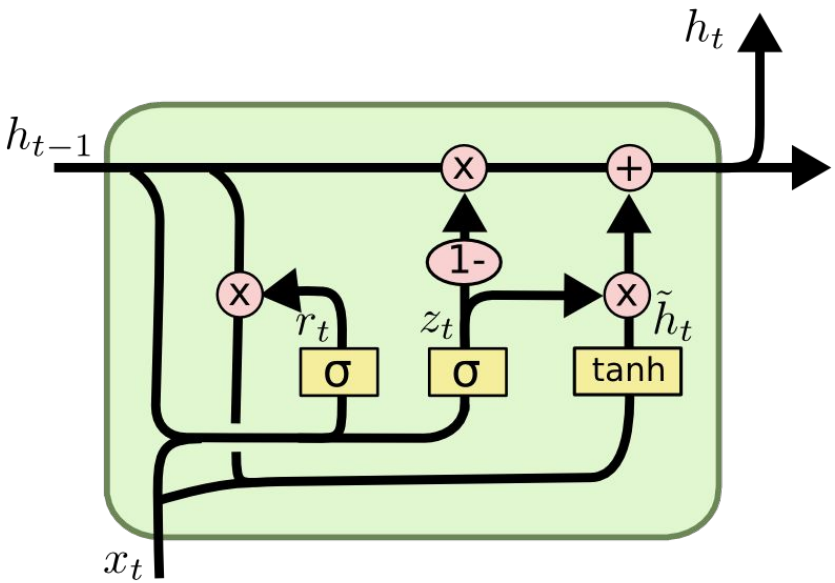
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Reset gate

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

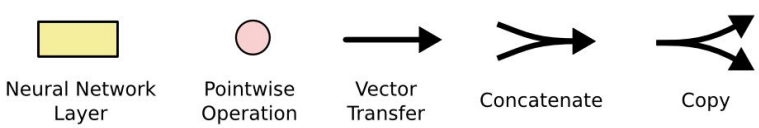


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

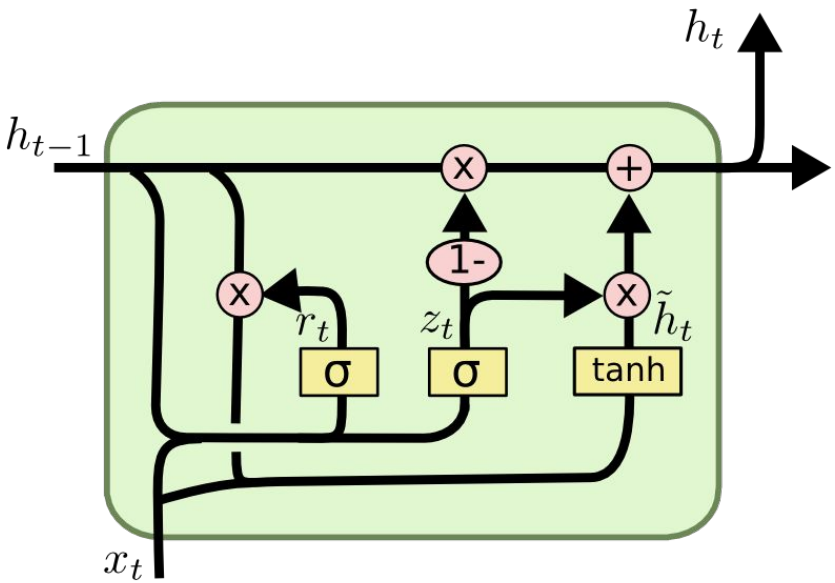


Reset gate

Proposed Hidden State

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

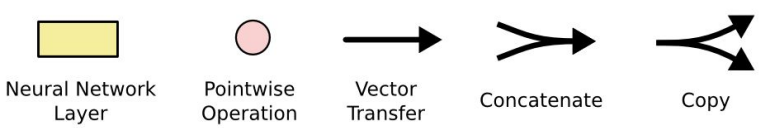


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



Update gate

Reset gate

Proposed Hidden State

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

Reset gate

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Proposed Hidden State

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

Hidden State

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

Reset gate

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Proposed Hidden State

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

Hidden State

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Update tradeoff between:

previous hidden state and proposed hidden state

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

Reset gate

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Proposed Hidden State

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

Hidden State

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Update tradeoff between:

previous hidden state and proposed hidden state

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU

Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

Reset gate

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

Proposed Hidden State

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

Hidden State

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

If z=1:

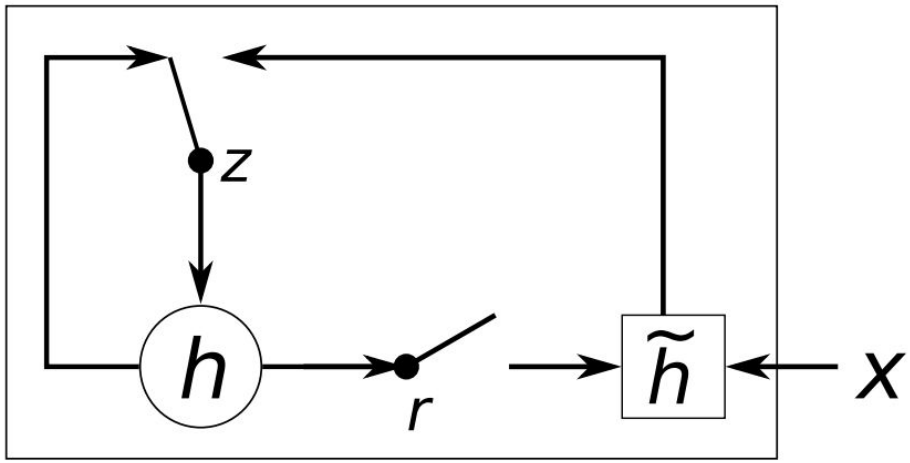
ignore previous state, use proposed state

If z=0:

ignore proposed state, use previous state

Images from: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU



$$r_j = \sigma\left(\left[\mathbf{W}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

$$\tilde{h}_j^{\langle t\rangle} = \phi\left(\left[\mathbf{W}\mathbf{x}\right]_j + \left[\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right)$$

$$z_j = \sigma\left(\left[\mathbf{W}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1-z_j)\tilde{h}_j^{\langle t\rangle}$$

Cho et al. 2014. (http://aclweb.org/anthology/D/D14/D14-1179.pdf)

GRU

Notation: t fixed; j-th element in the vector

"each hidden unit will learn to capture dependencies over different time scales"

$$r_j = \sigma\left(\left[\mathbf{W}_r\mathbf{x}\right]_j + \left[\mathbf{U}_r\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

$$\tilde{h}_j^{\langle t\rangle} = \phi\left(\left[\mathbf{W}\mathbf{x}\right]_j + \left[\mathbf{U}\left(\mathbf{r}\odot\mathbf{h}_{\langle t-1\rangle}\right)\right]_j\right)$$

$$z_j = \sigma\left(\left[\mathbf{W}_z\mathbf{x}\right]_j + \left[\mathbf{U}_z\mathbf{h}_{\langle t-1\rangle}\right]_j\right)$$

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1-z_j)\tilde{h}_j^{\langle t\rangle}$$

Cho et al. 2014. (http://aclweb.org/anthology/D/D14/D14-1179.pdf)

GRU: Nematus Code

nematus/layers.py, lines 71-98, 101-154 (https://github.com/rsennrich/nematus/blob/9b08048e278db46ede288bcca928d176811ee2c2/nematus/layers.py)

Gated Recurrent Unit (GRU)

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder– Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics. (http://aclweb.org/anthology/D/D14/D14-1179.pdf)

Chris Olah. 2015. Understanding LSTMs. (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Basic RNN Encoder-Decoder

Notation note:

In general, neural models tend to be drawn with the input at the bottom and output at the top.

In the rest of this presentation, I will be drawing them with the ***input at the top***.

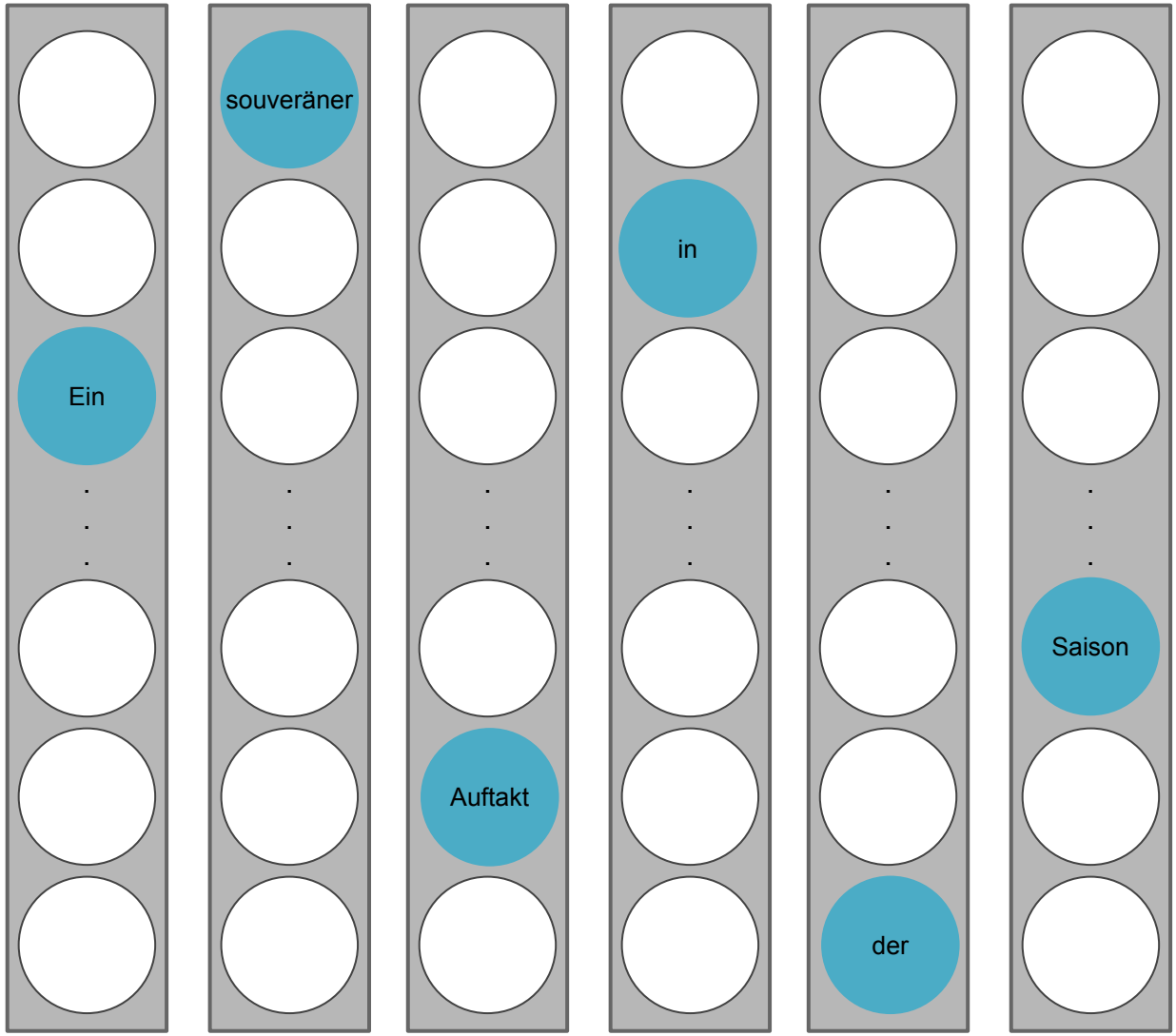
(Flipping them over the horizontal axis will result in the standard version.)

Input

Ein souveräner Auftakt in der Saison

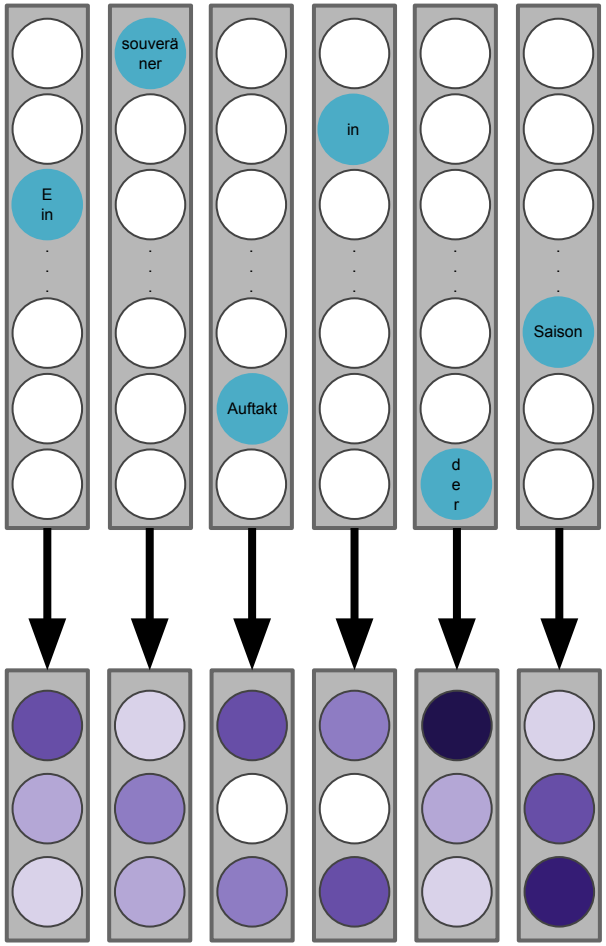
Input

One-hot representation



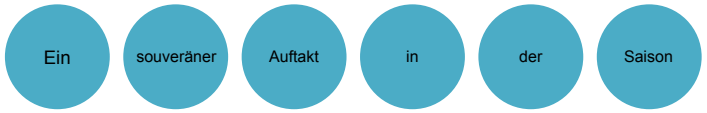
Input

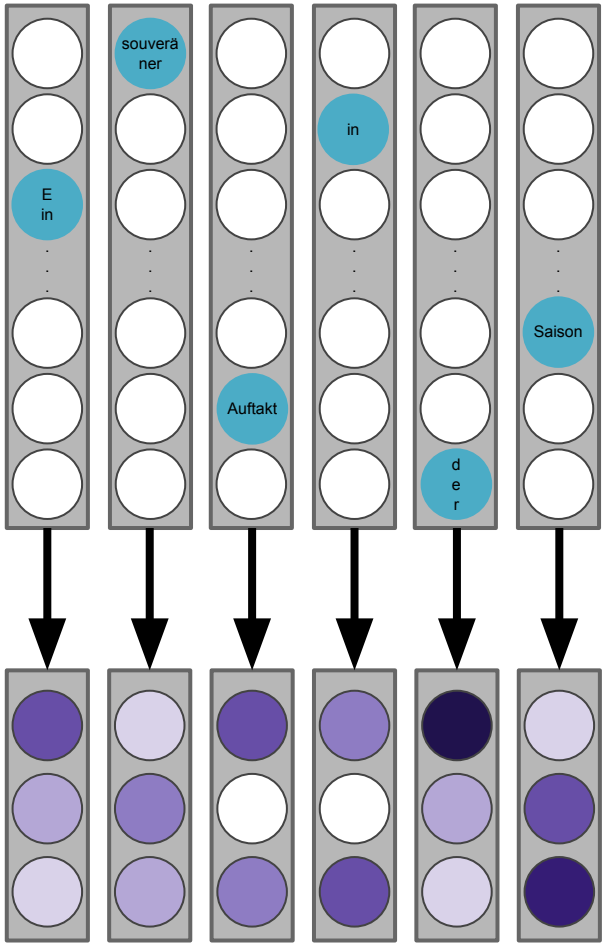
Continuous representation
(word embeddings)



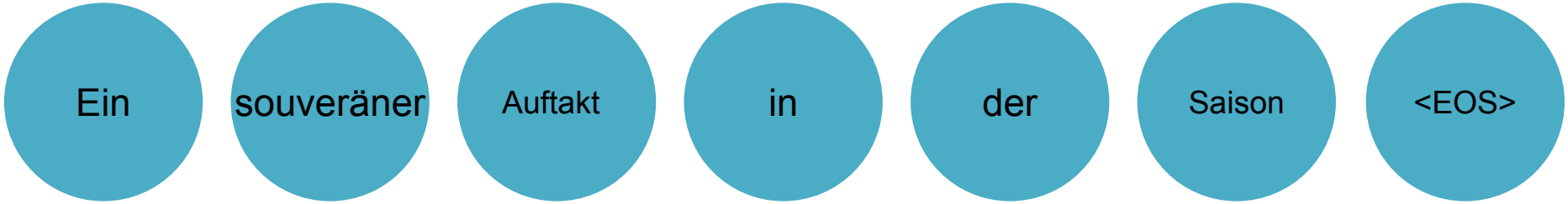
Input

As visual shorthand for the input (one-hot into continuous), we'll just use this to save space:

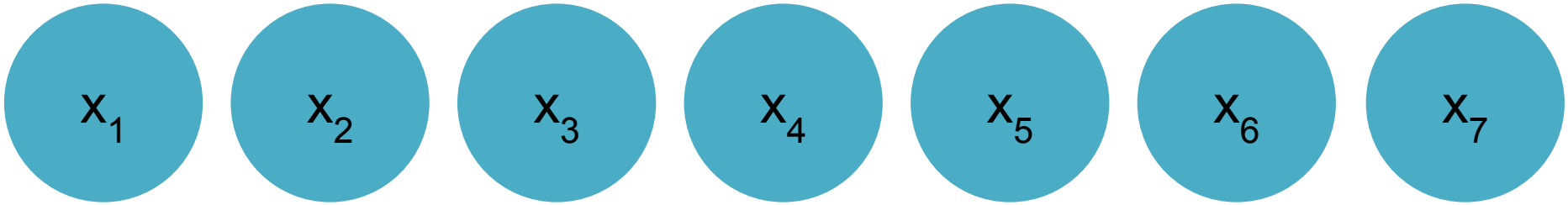




Input



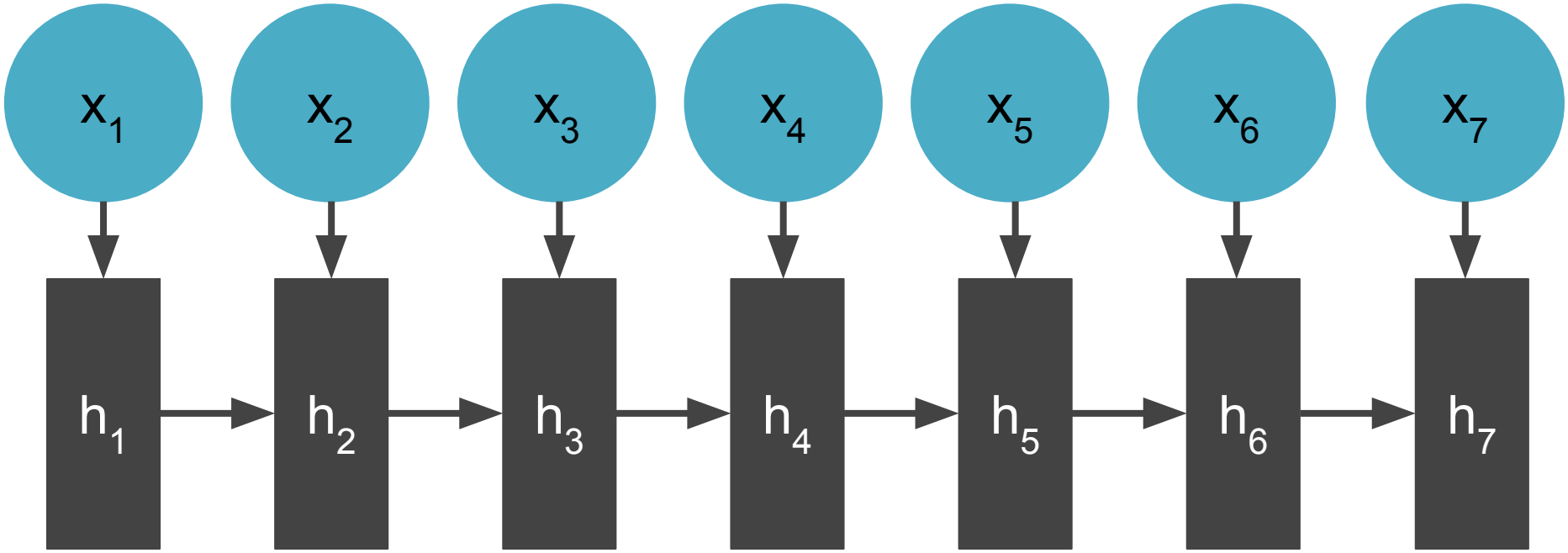
Input



$$\mathbf{x} = (x_1, \ldots, x_T)$$

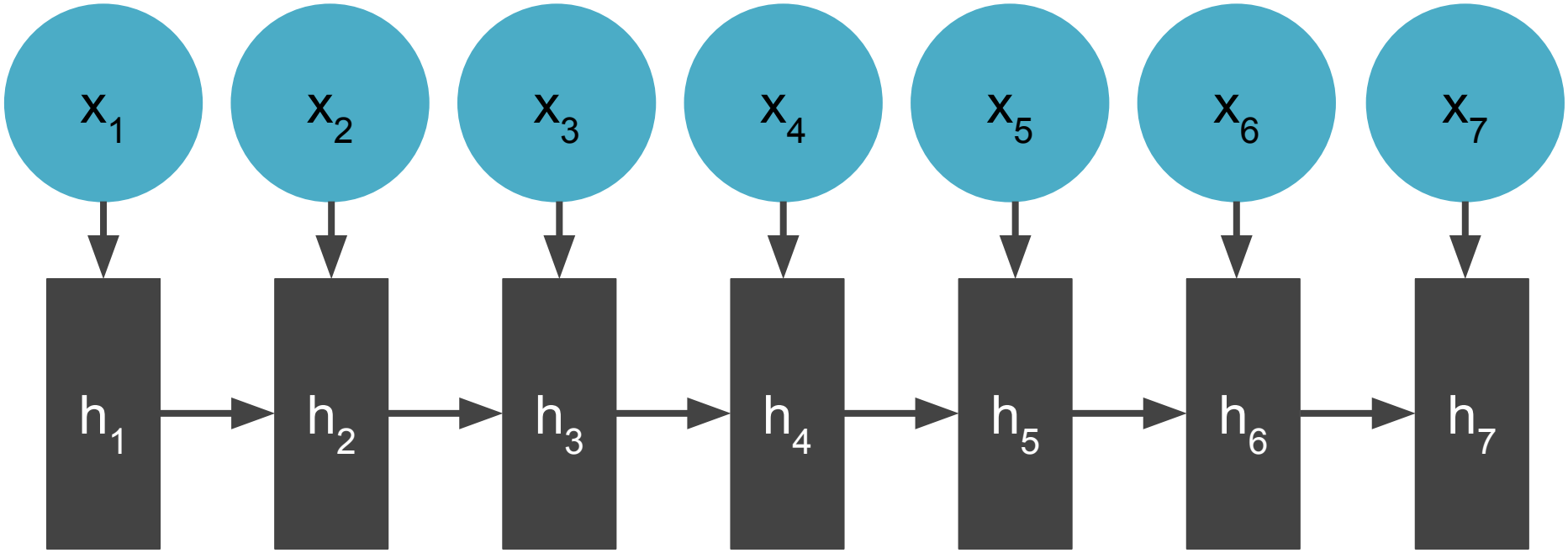
(here, T=7)

RNN Encoder



$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, x_t)$$

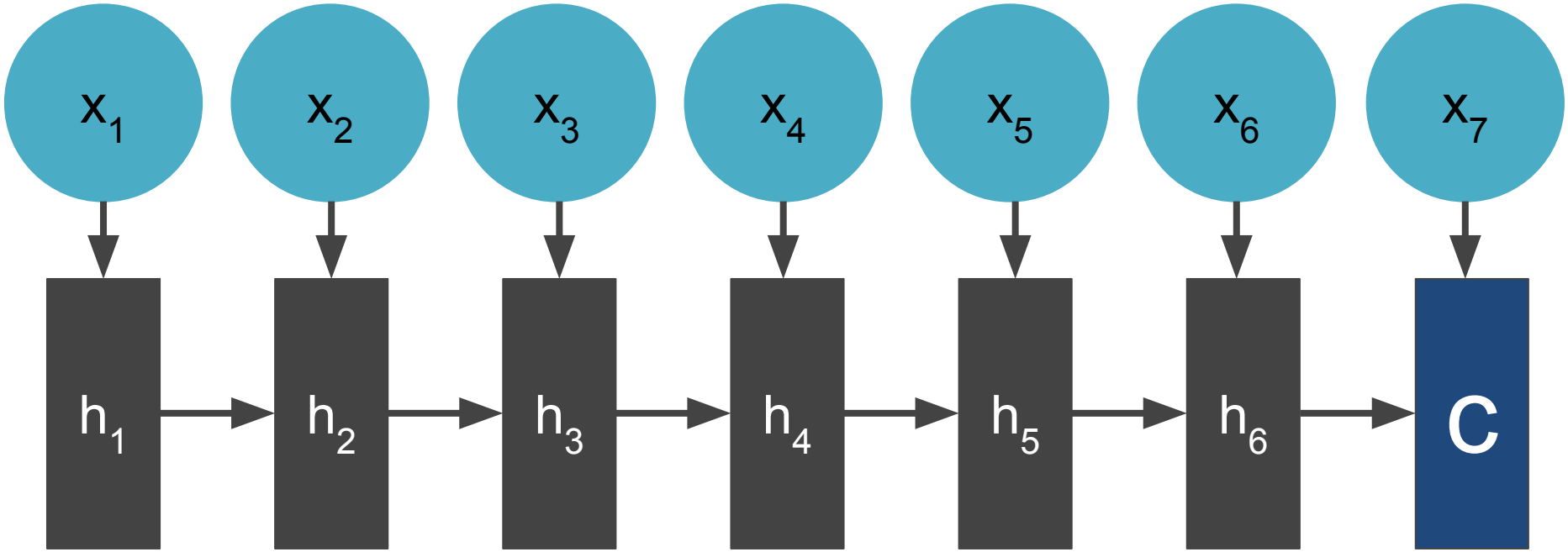
RNN Encoder



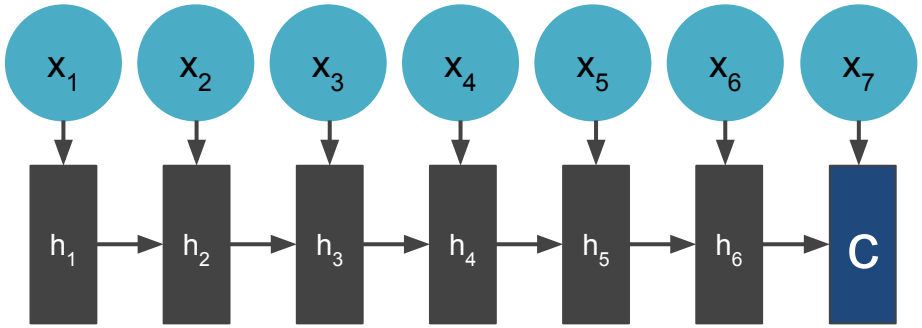
$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, x_t)$$

Let's assume *f* is a GRU (or LSTM).

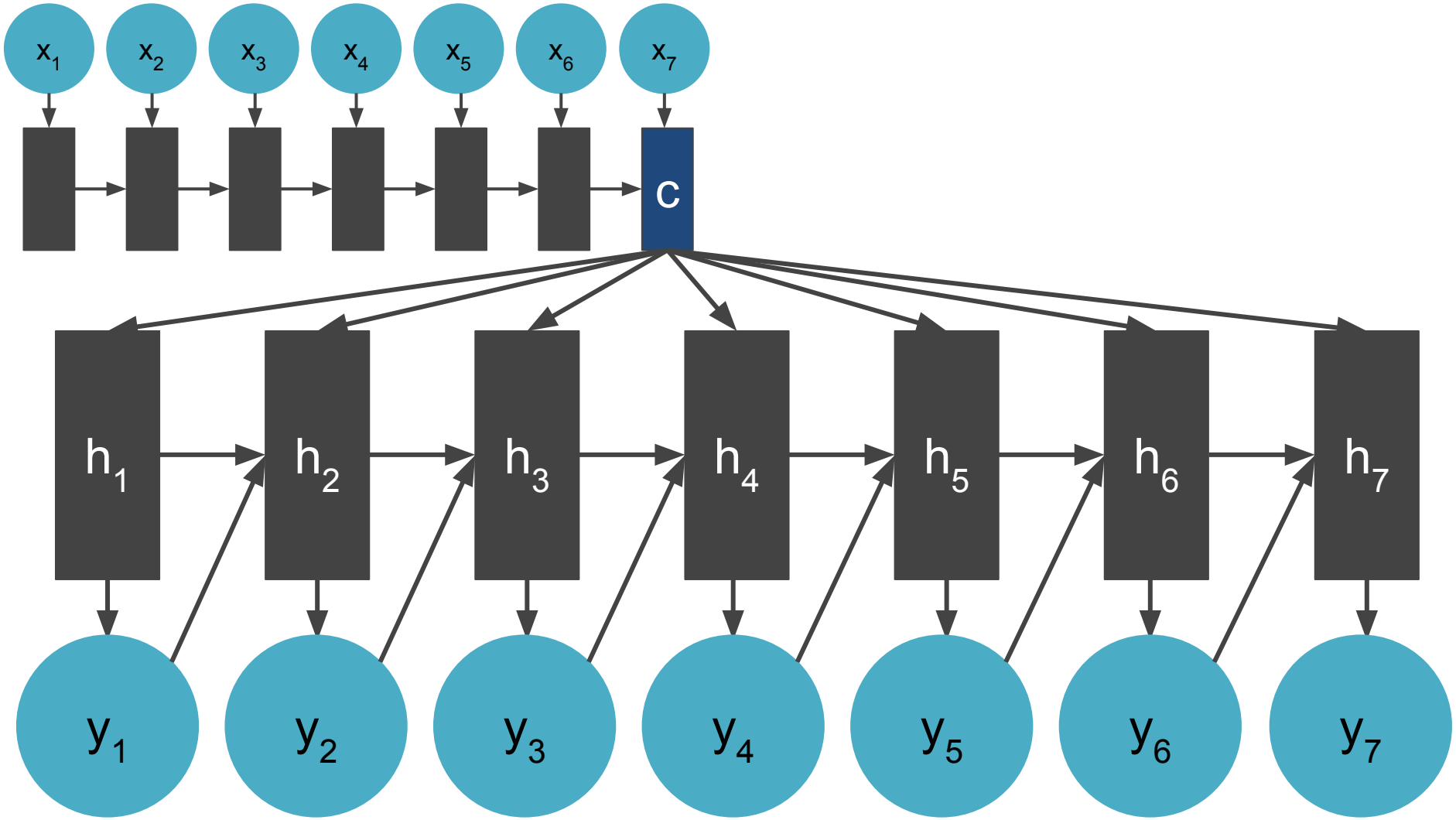
RNN Encoder



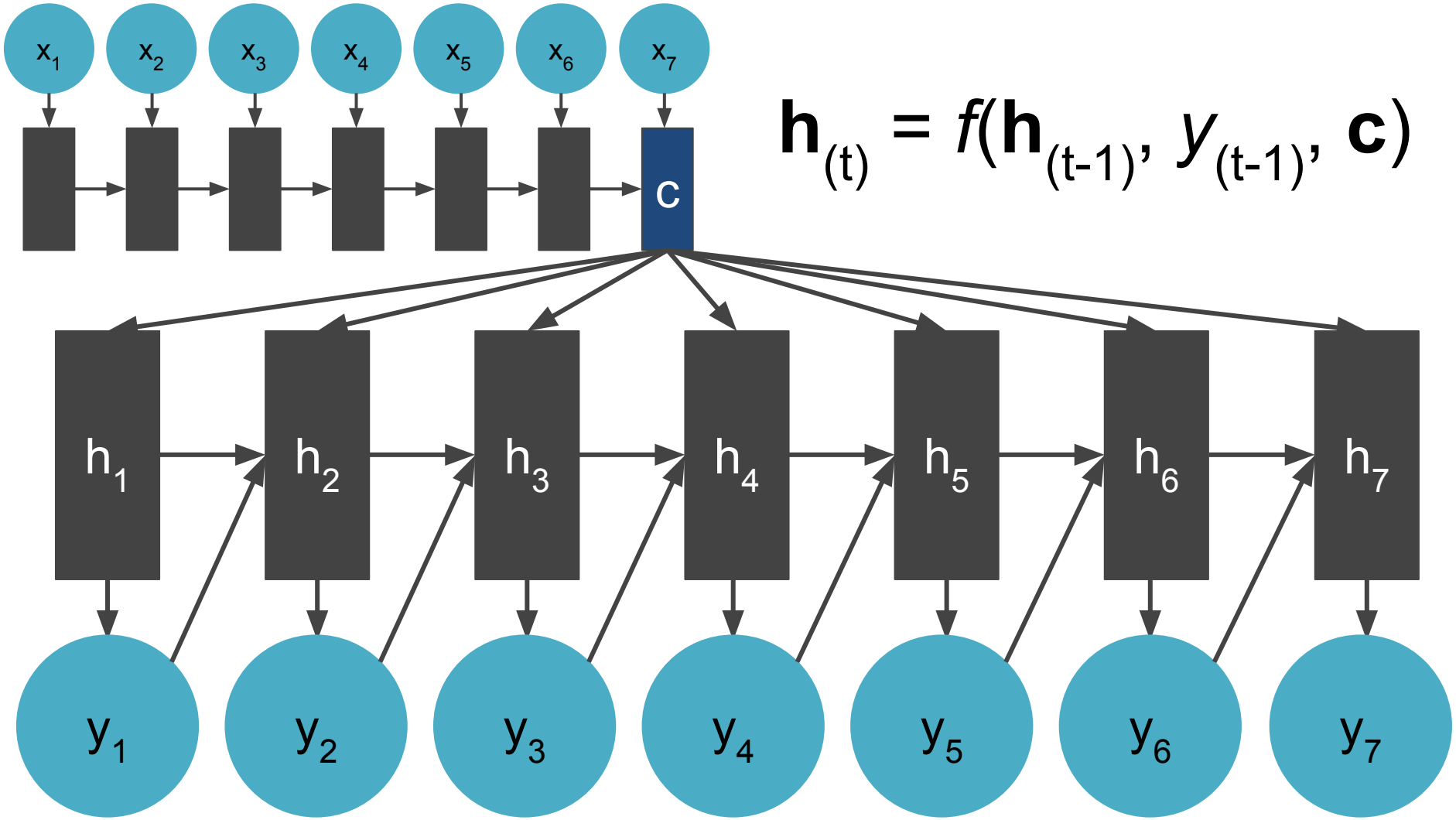
The last hidden state of the RNN, **c** (formerly h_7), contains a *summary* of the input sentence **x**.


x1
x2
x3
x4
x5
x6
x7
h1
h2
h3
h4
h5
h6
C

RNN Decoder

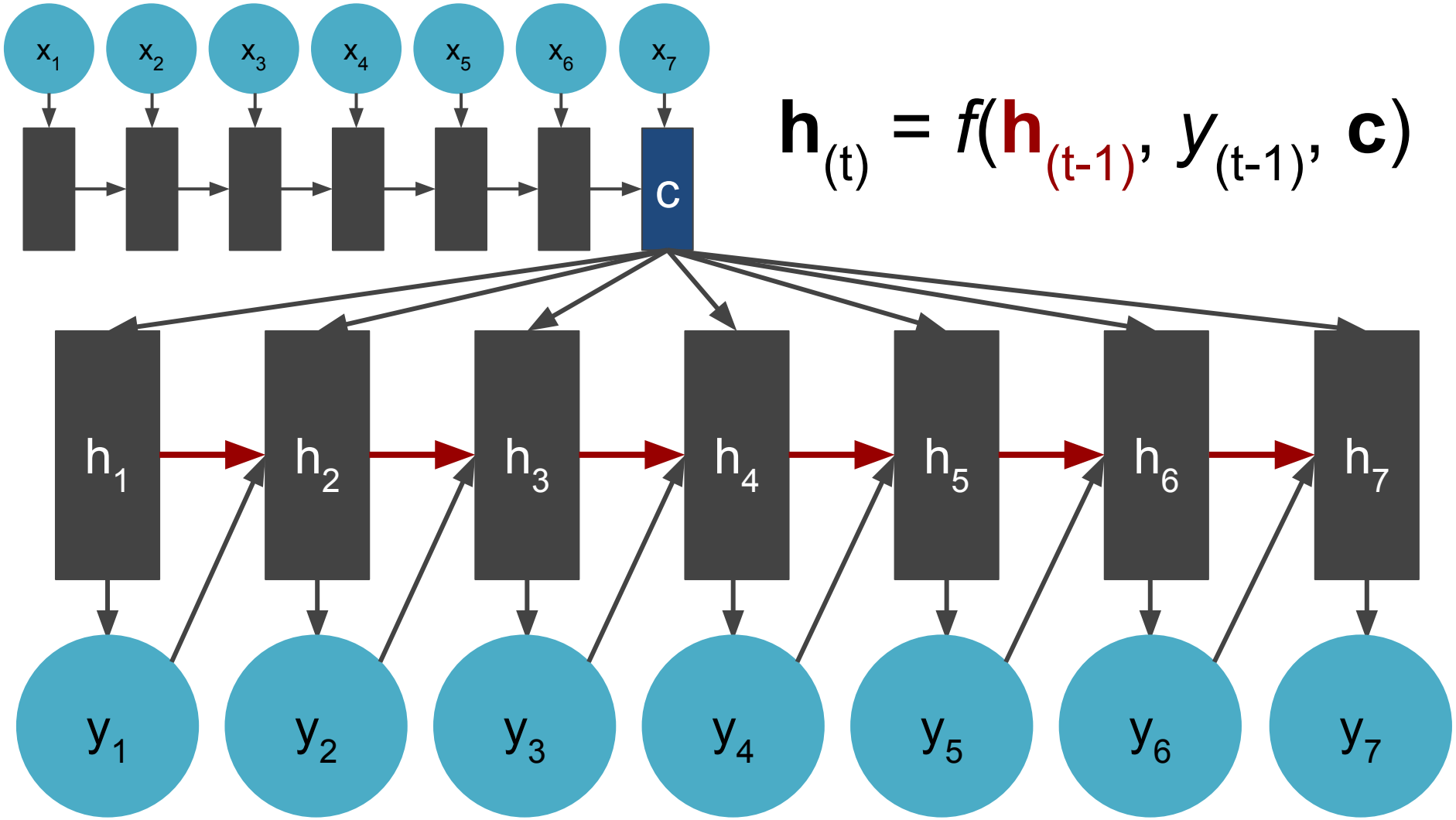


RNN Decoder



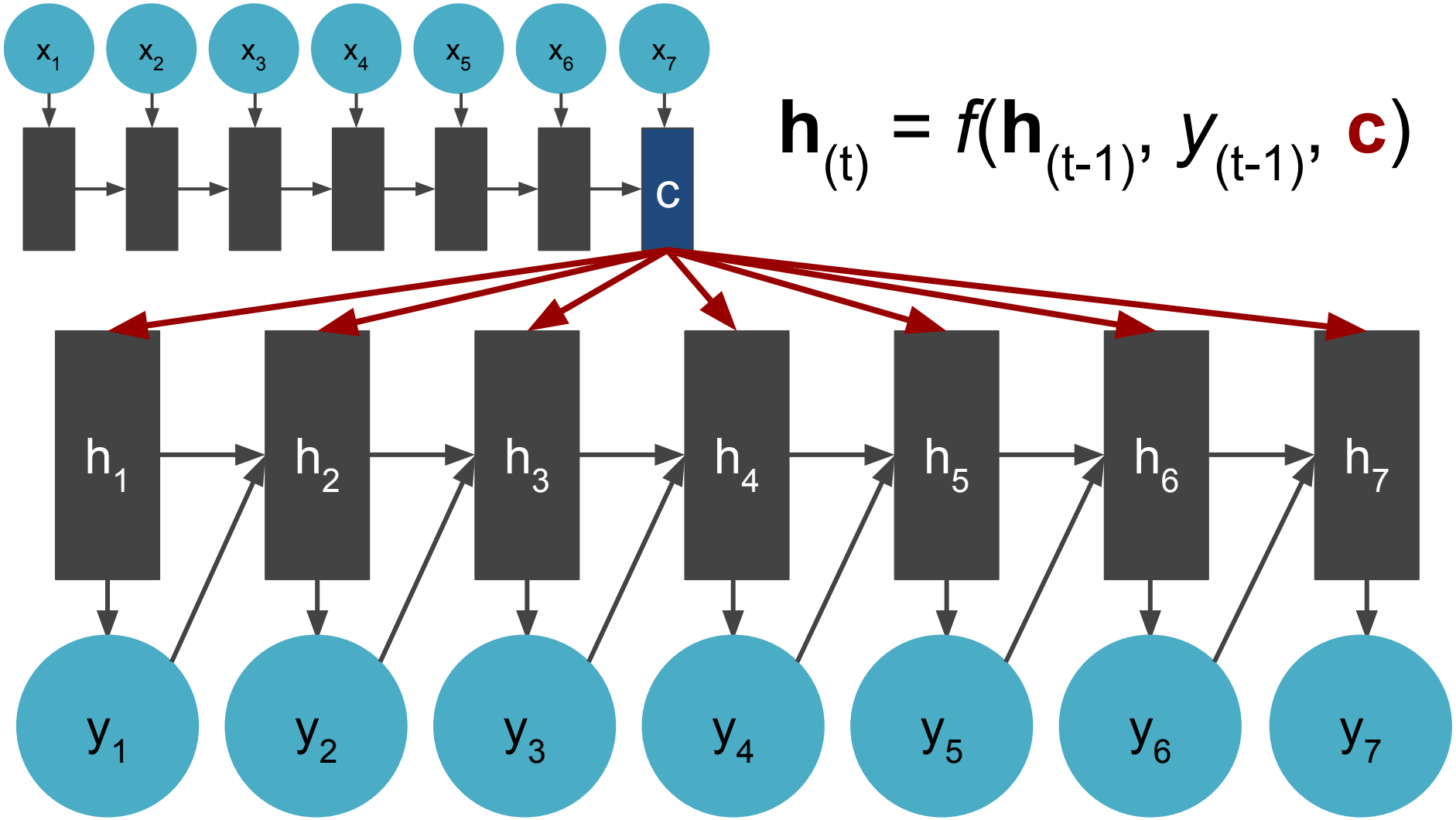
$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$

RNN Decoder



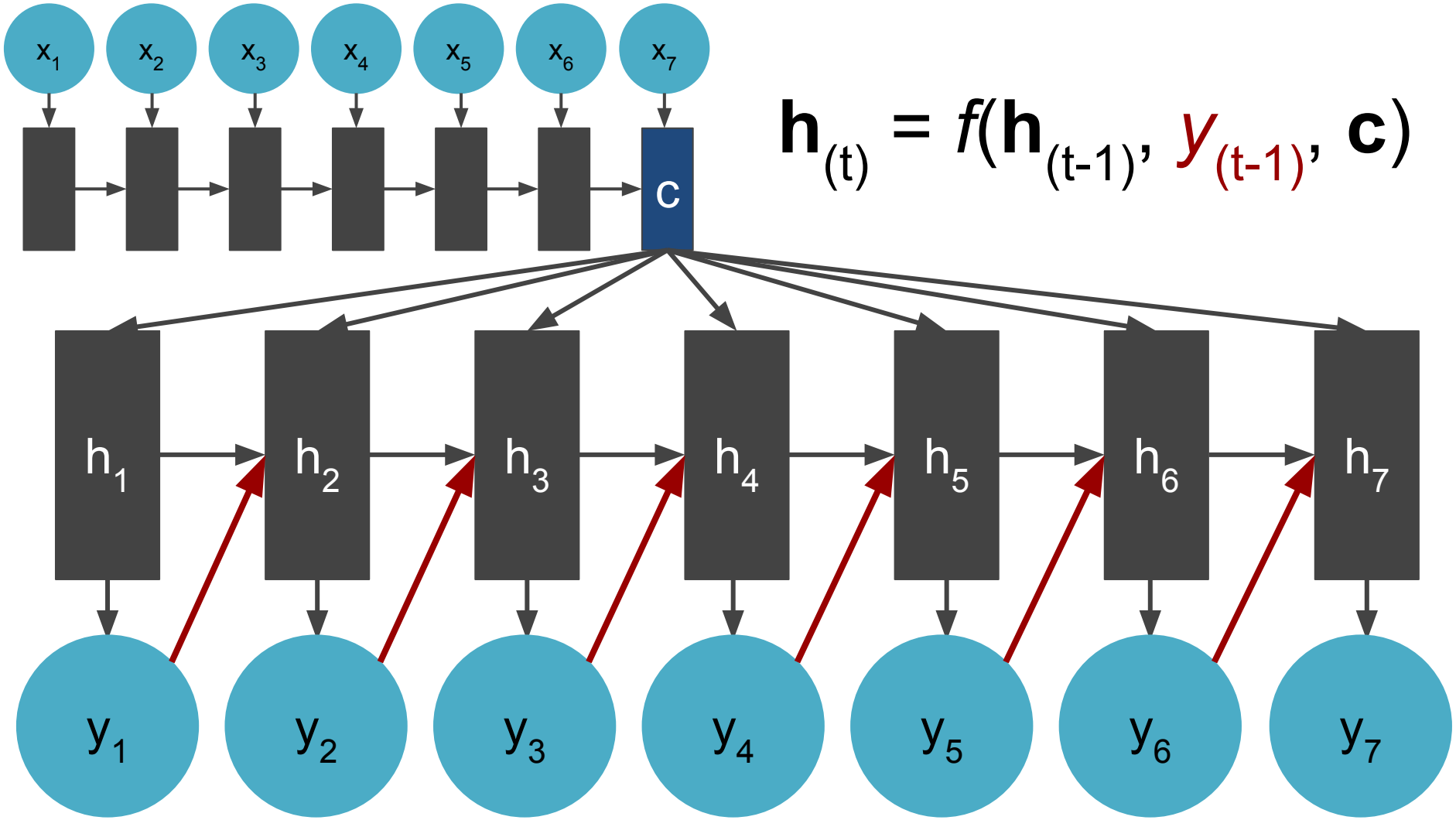
$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$

RNN Decoder



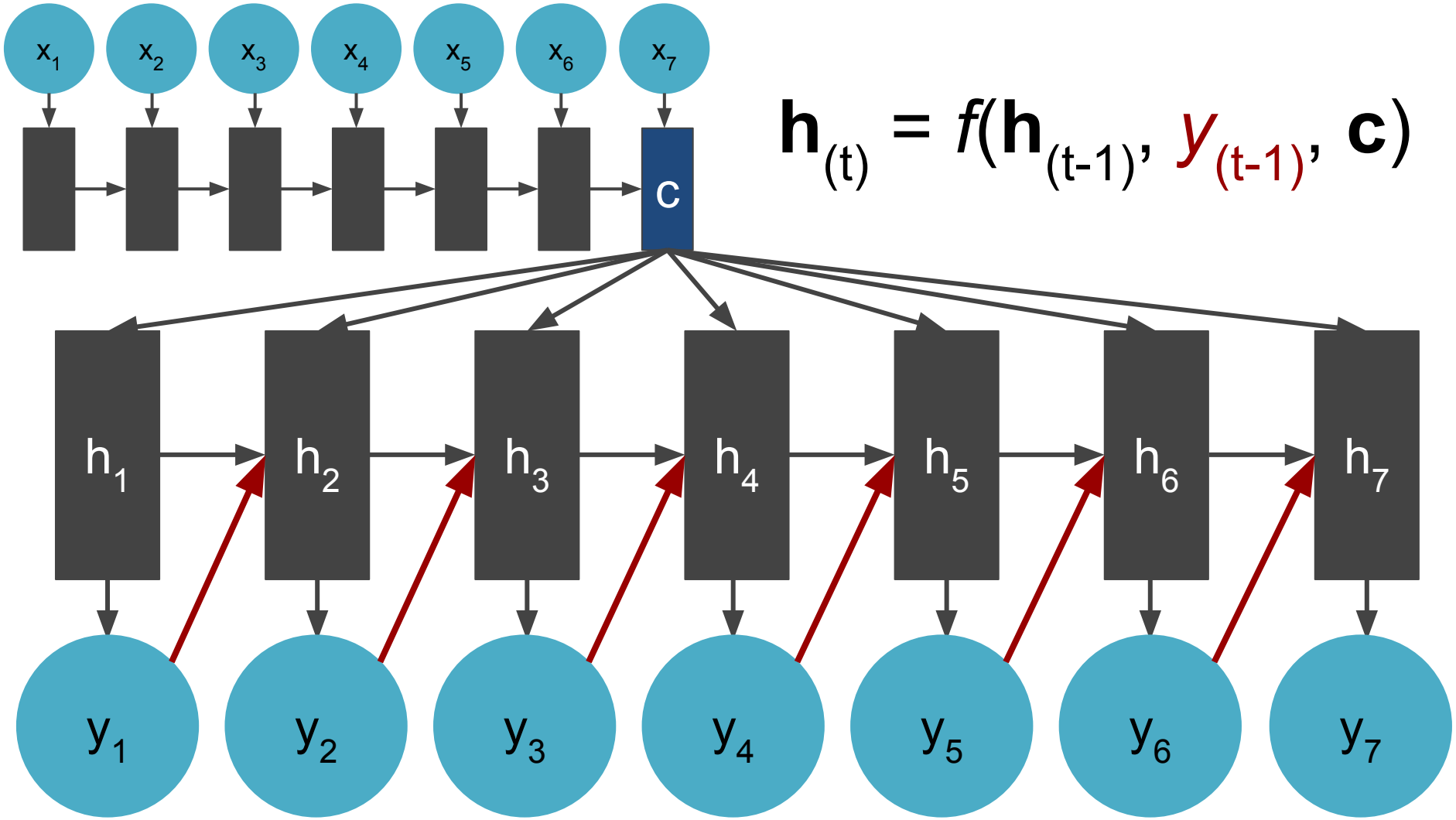
$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$

RNN Decoder



$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$

RNN Decoder

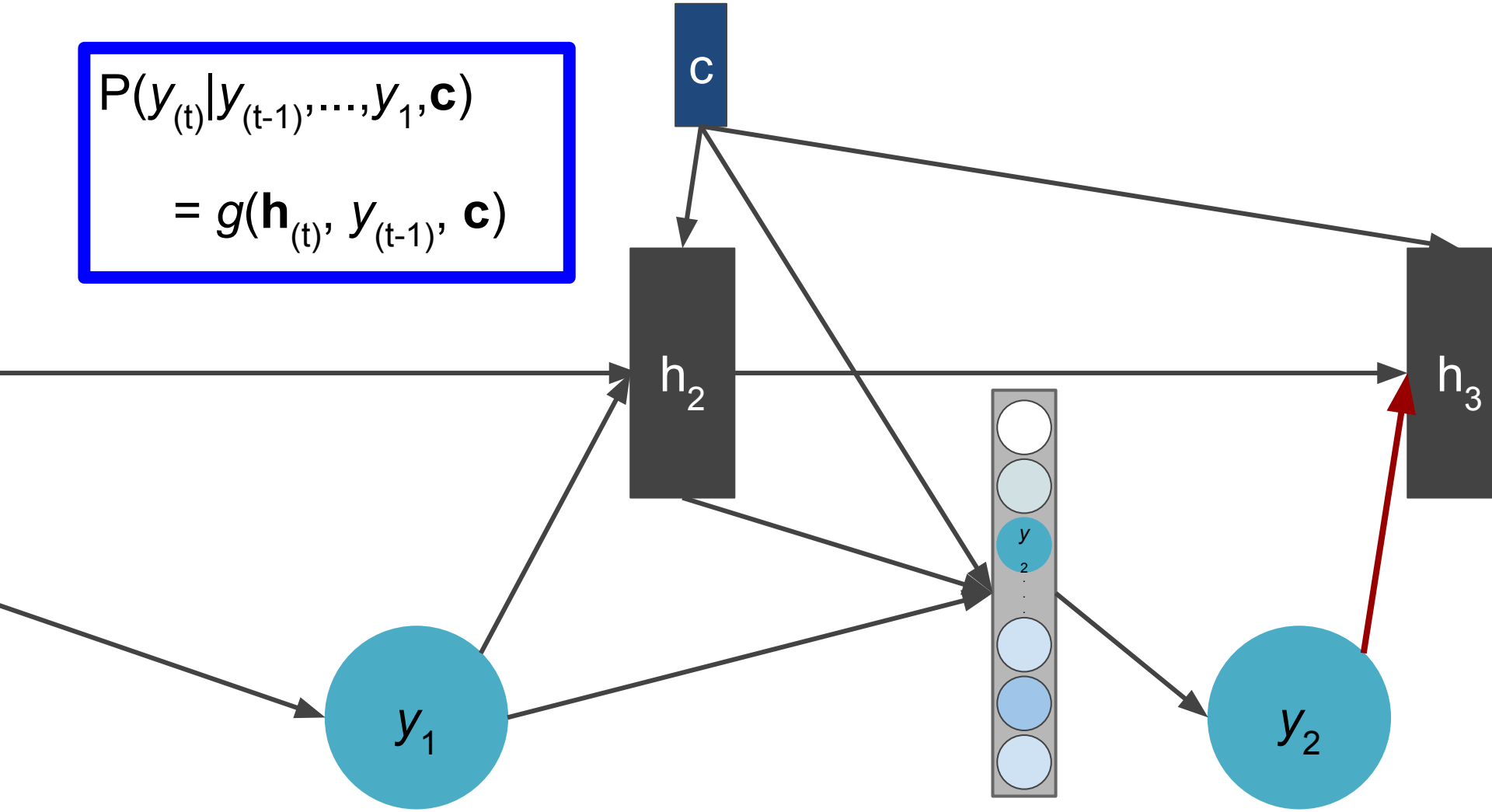


$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$

RNN Decoder: Zoomed In

$$P(y_{(t)}|y_{(t-1)},\ldots,y_1,\mathbf{c})$$

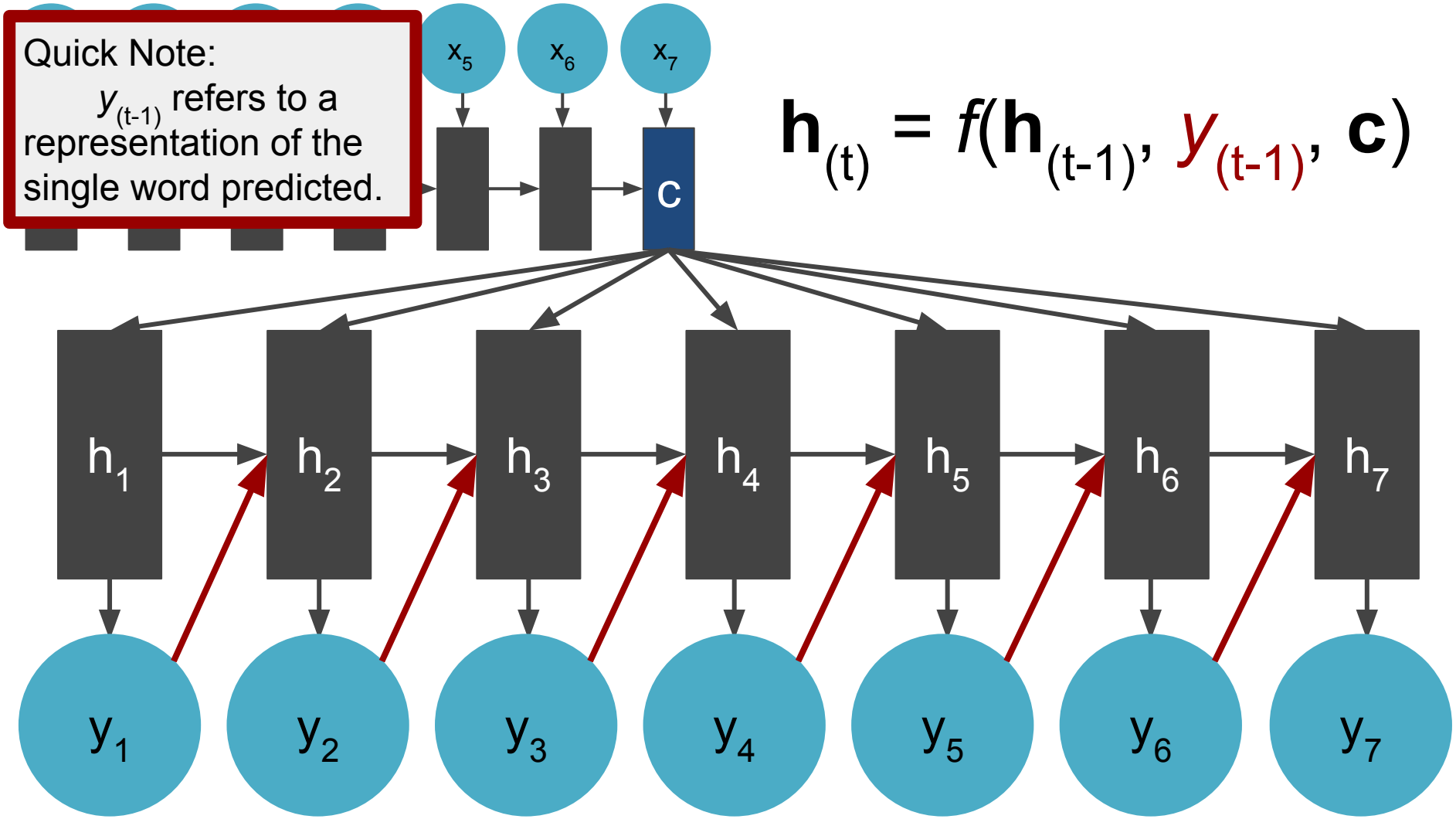
$$= g(\mathbf{h}_{(t)}, y_{(t-1)}, \mathbf{c})$$



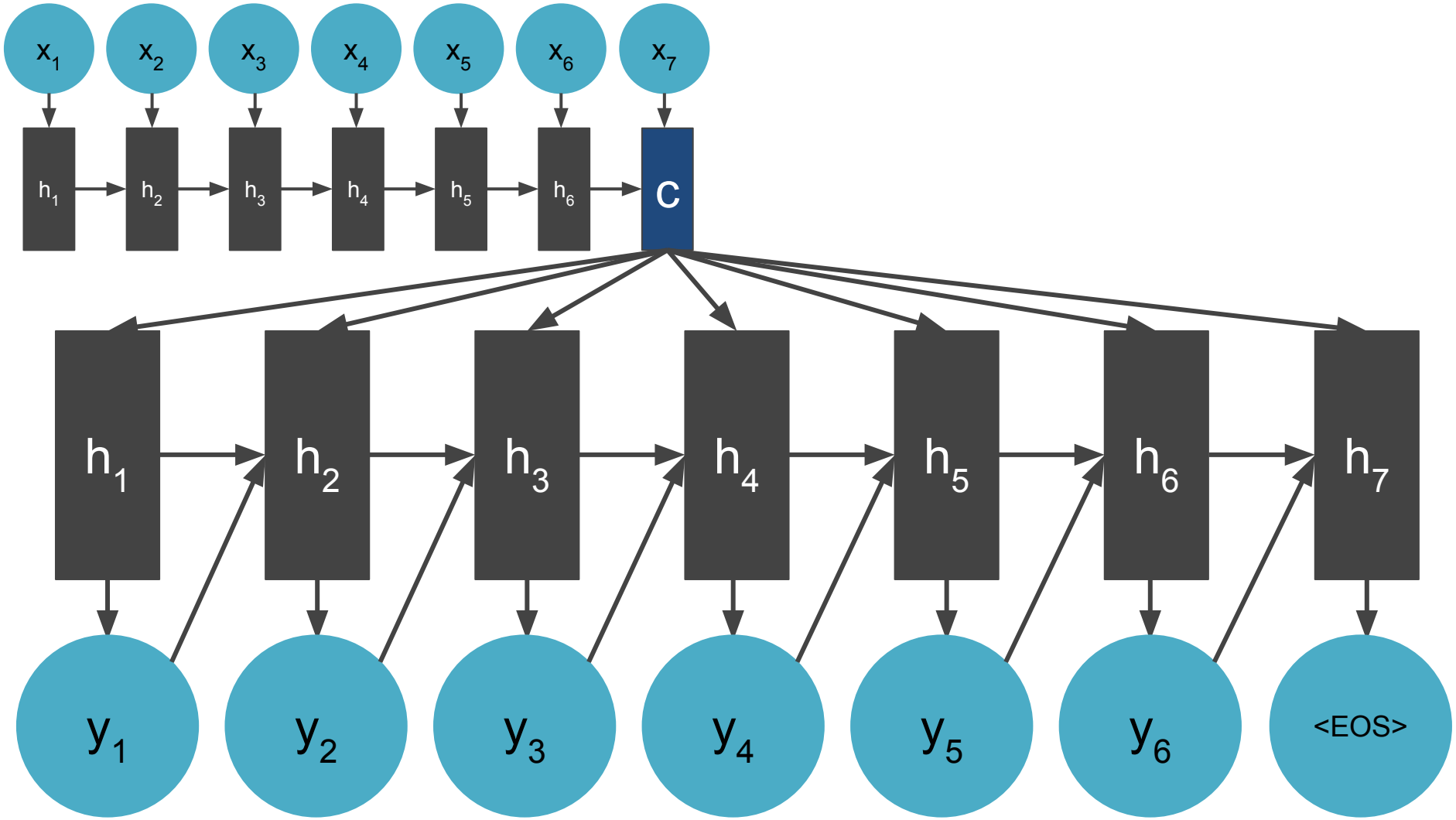
RNN Decoder

Quick Note:
$y_{(t-1)}$ refers to a representation of the single word predicted.

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, y_{(t-1)}, \mathbf{c})$$



RNN Encoder-Decoder



RNN Encoder-Decoder Math

Input: $\mathbf{x} = (x_1,\ldots,x_T)$

Encoder:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}\ ,\ x_t)$$

$$\mathbf{c} = \mathbf{h}_{(T)}$$

Decoder:

$$\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)},\ y_{(t-1)},\ \mathbf{c})$$

$$P(y_{(t)}|y_{(t-1)},\ldots,y_1,\mathbf{c}) = g(\mathbf{h}_{(t)},\ y_{(t-1)},\ \mathbf{c})$$

Output: $\mathbf{y} = (y_1,\ldots,y_{T'})$

Training

Train to maximize the log-likelihood of the corpus:

$$\max_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{y}_n \mid \mathbf{x}_n)$$

A Few Final Details...

- Start with a <BOS> token (not shown before).
- Translate until the <EOS> token is generated (or a max length is reached).
- This lacks:
  - Explicit alignments (to be discussed next)
  - Coverage vector

Basic RNN Encoder-Decoder

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder– Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar. Association for Computational Linguistics. (http://aclweb.org/anthology/D/D14/D14-1179.pdf)

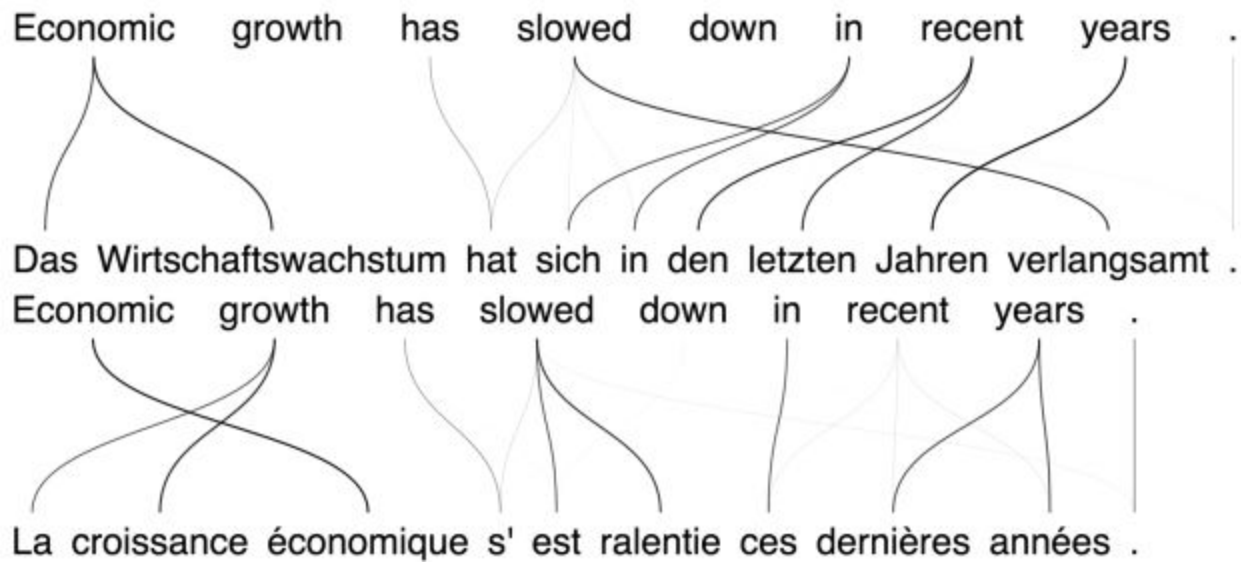
Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS 2014). (https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf)

Attention

Intuition

- Summary vector is a bottleneck, especially for long sentences
- Want to introduce some form of alignment

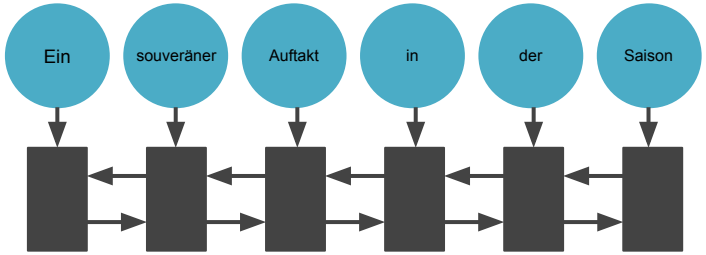
Attention

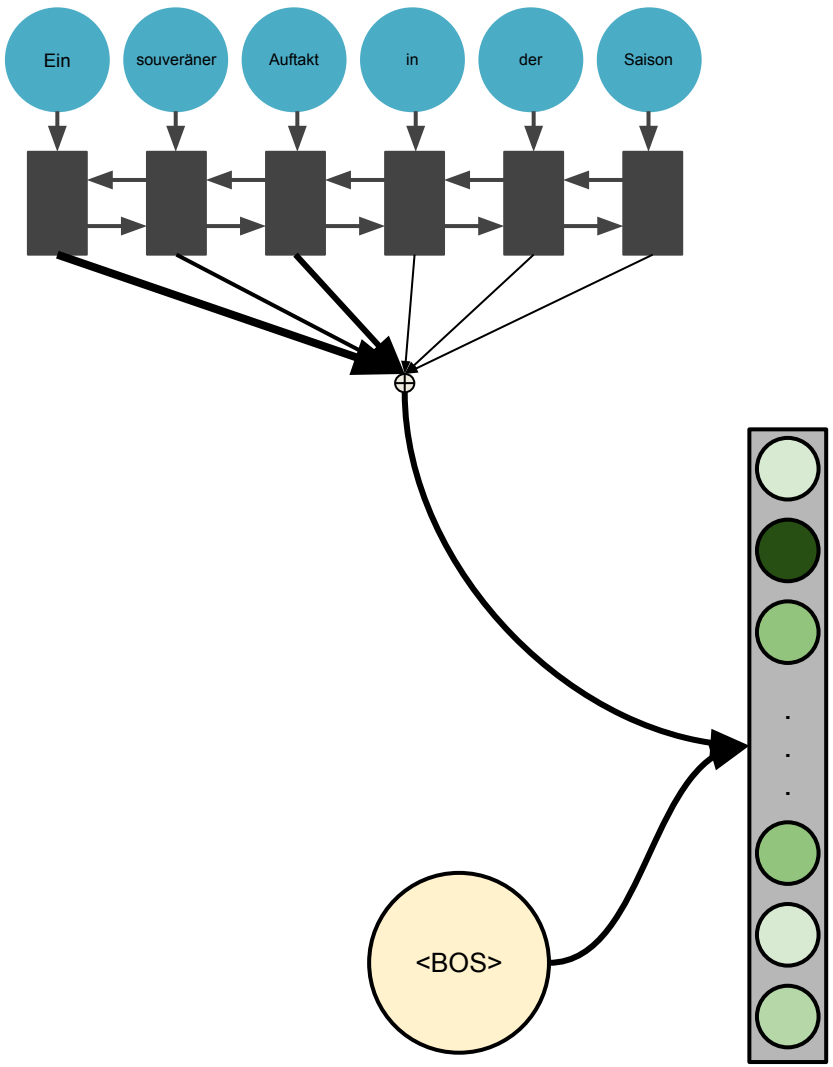


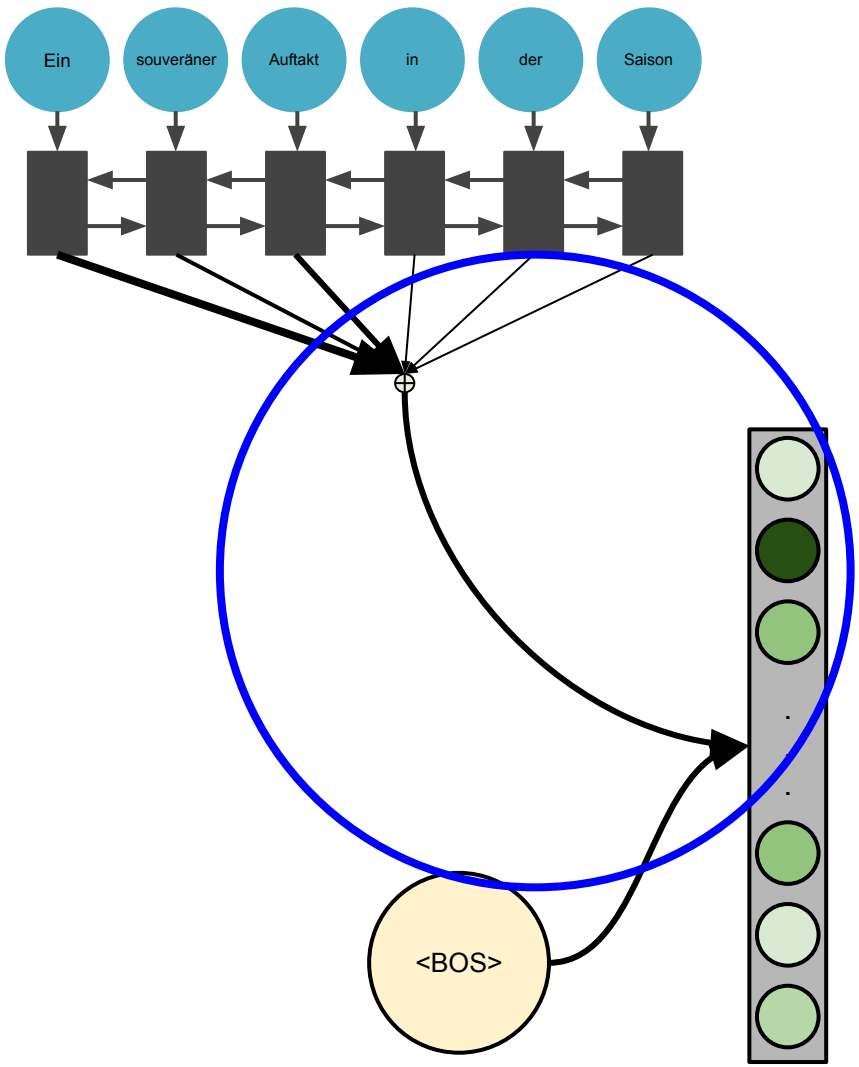
Soft attention, visualized.

Cho, https://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/

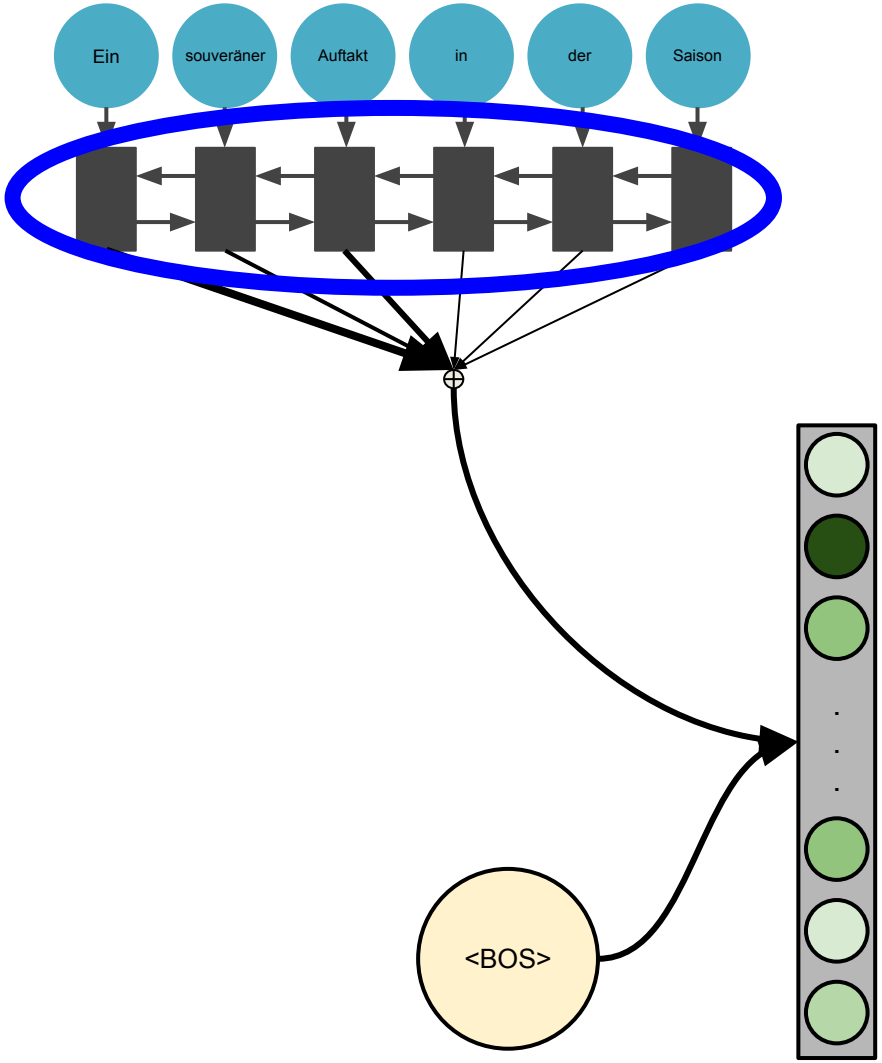
High-Level Walk-Through


Ein
souveräner
Auftakt
in
der
Saison

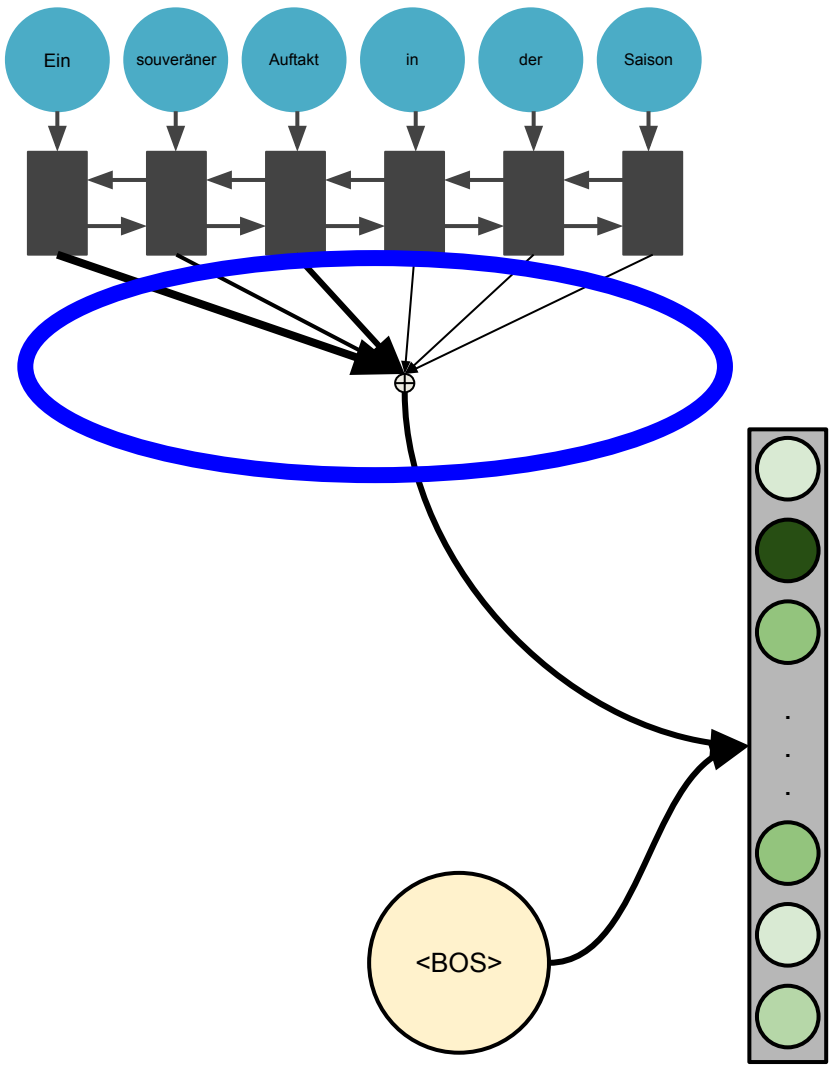

Ein
souveräner
Auftakt
in
der
Saison
<BOS>



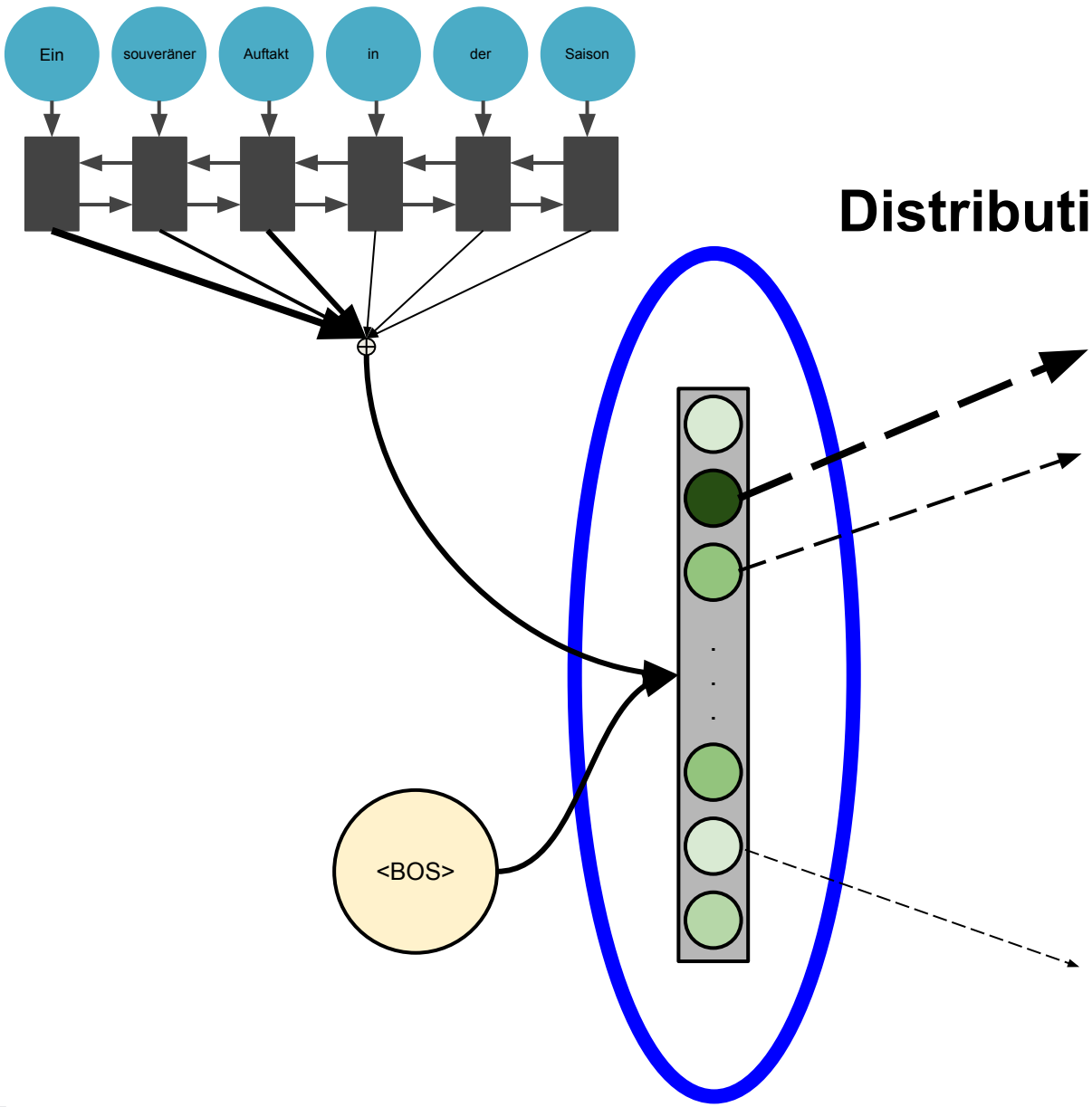
(Quite a bit of detail between the input and the probability distribution over the vocabulary is omitted from this drawing, but will be discussed in the following slides.)



Bidirectional RNN

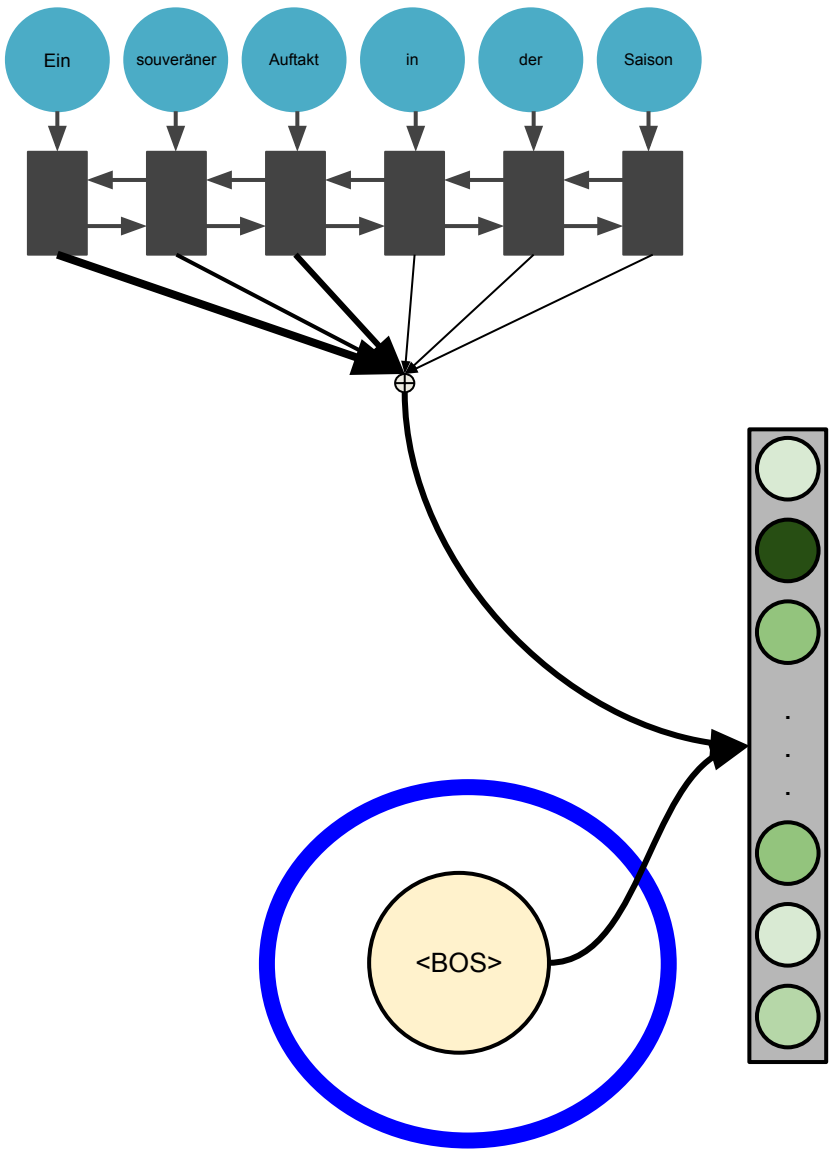


Attention Mechanism

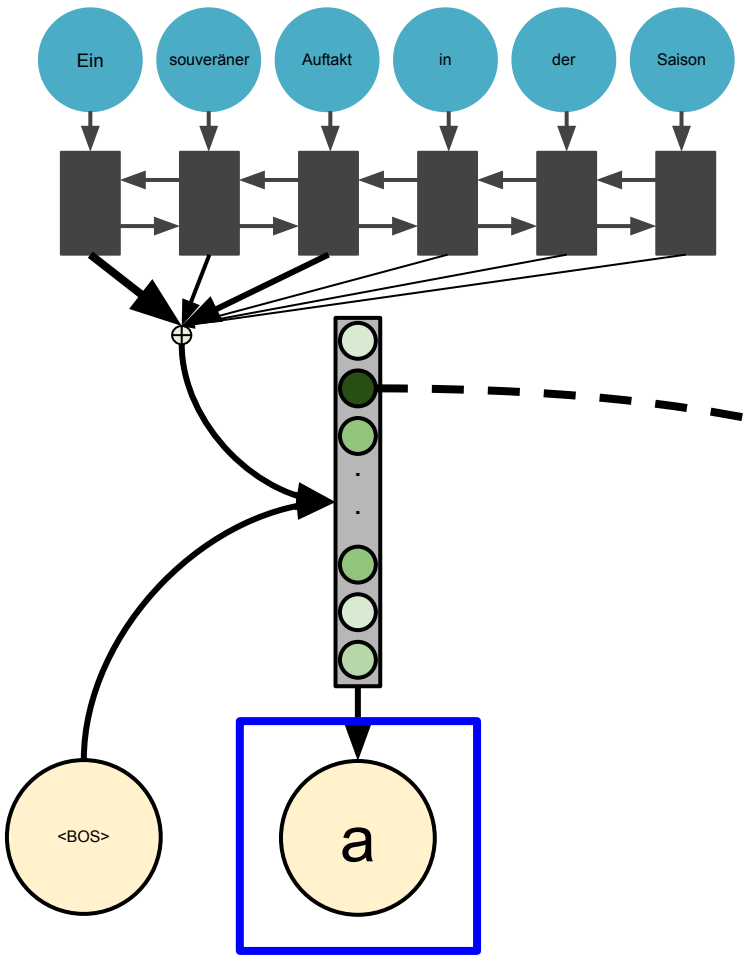


Distribution Over Vocabulary

0.6	**a**
0.3	an
0.05	the
0.01	that
...	...
1e-9	halloween
1e-10	trilobite

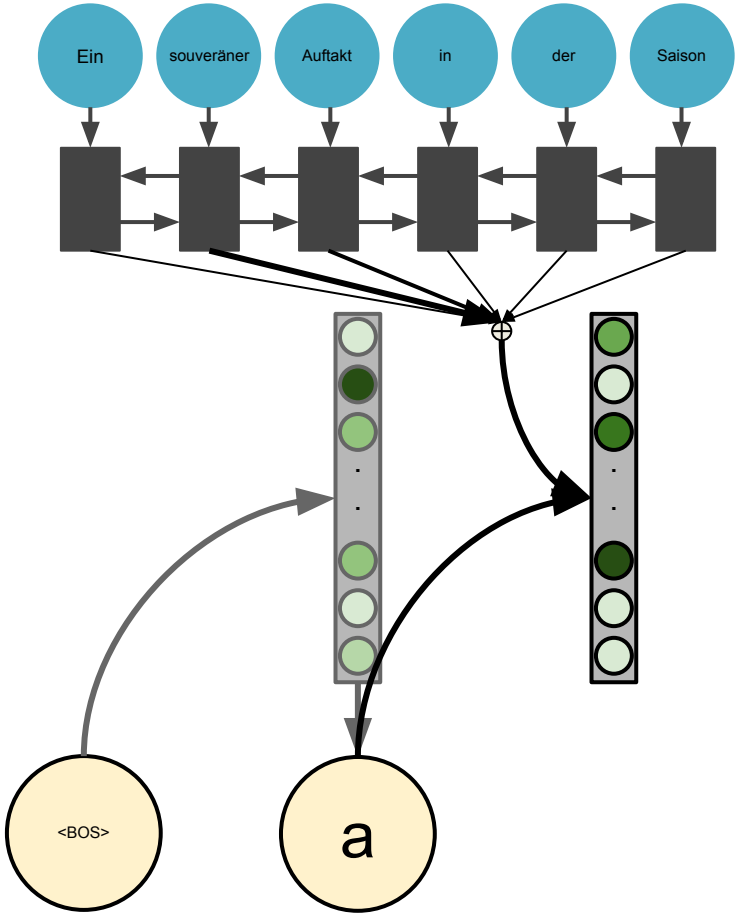


Condition on Previous Word

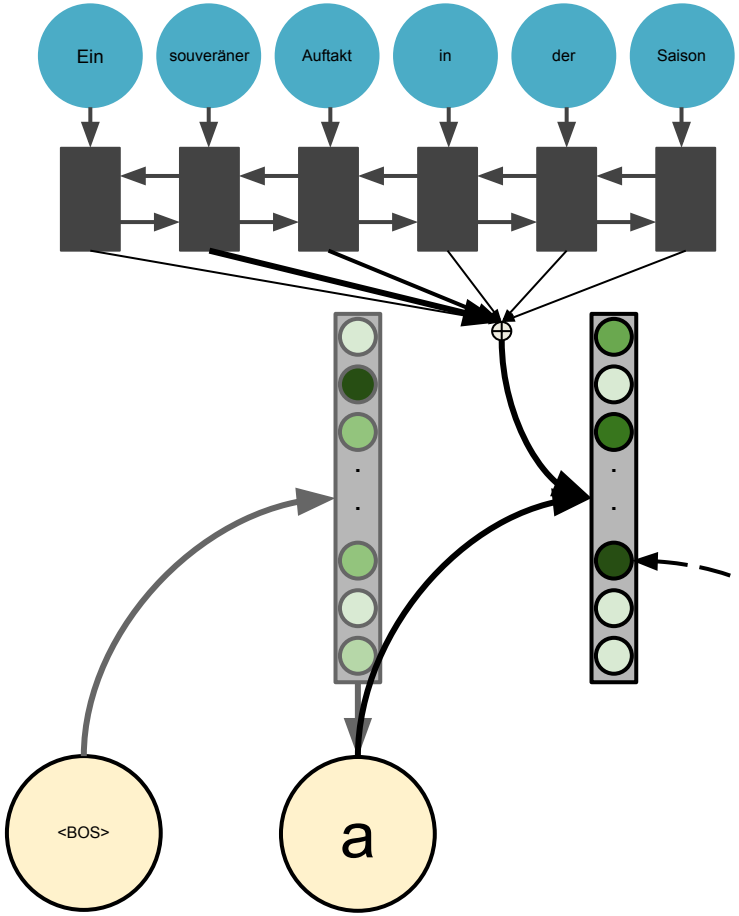


Select Most Probable Word

0.6	**a**
0.3	an
0.05	the
0.01	that
...	...
1e-9	halloween
1e-10	trilobite



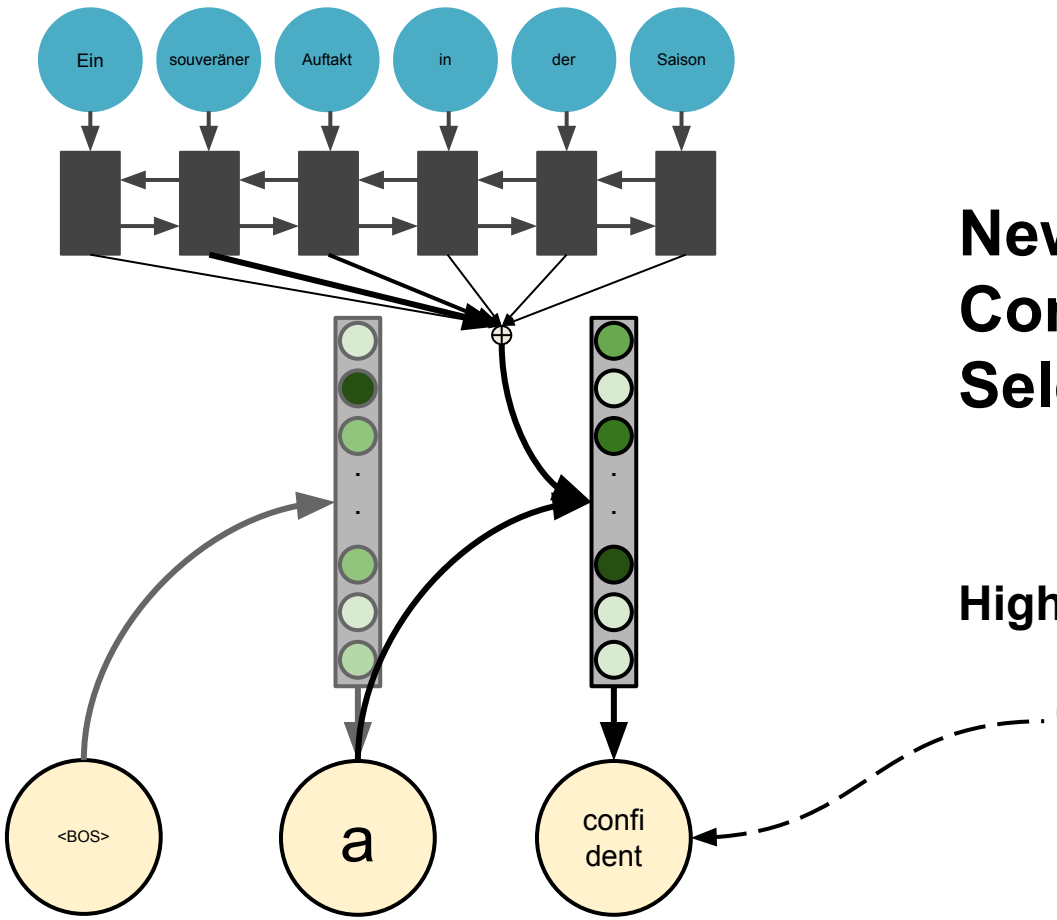
New Distribution, Conditioned on Previous Selection



New Distribution, Conditioned on Previous Selection

Highest probability word:

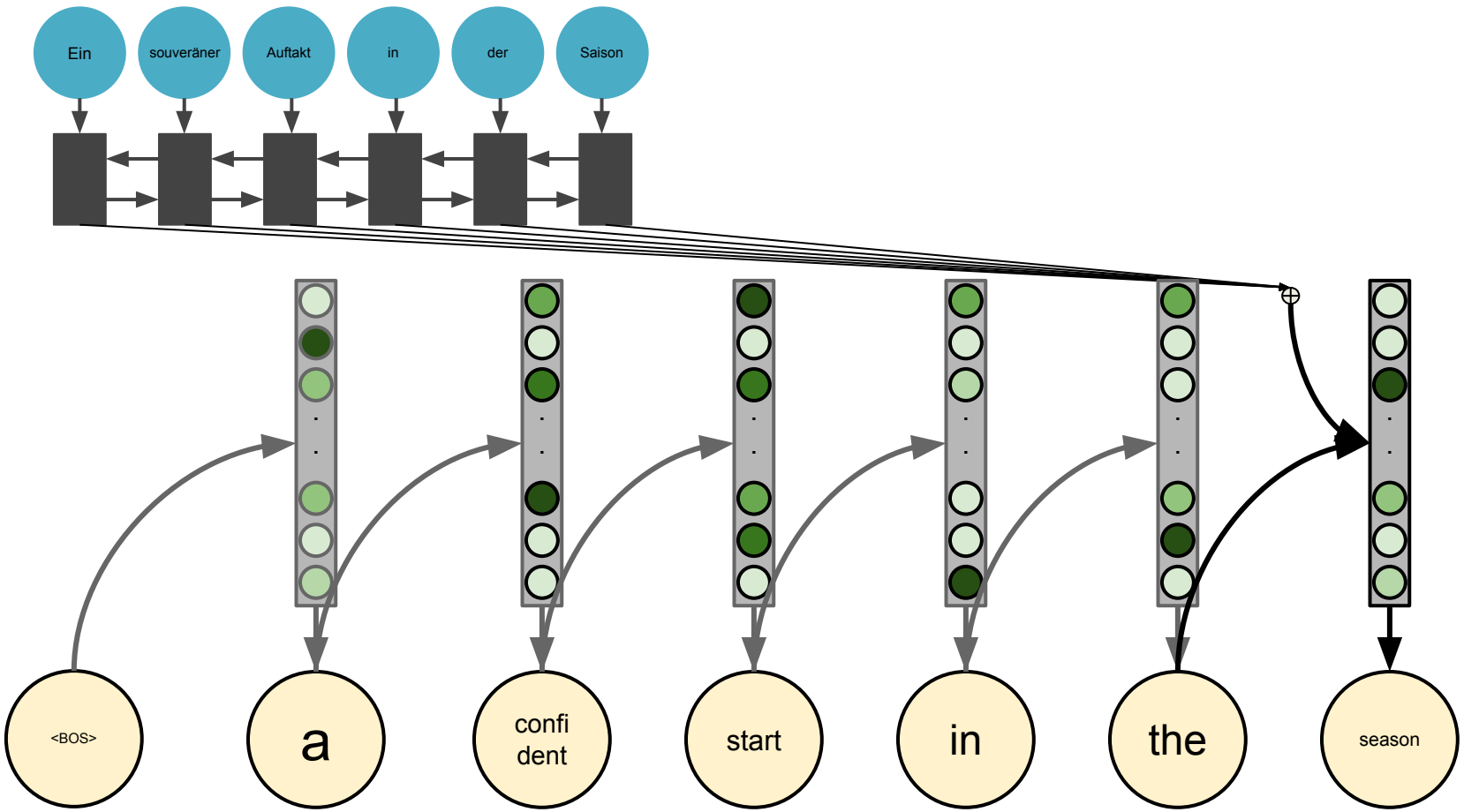
“confident”



New Distribution, Conditioned on Previous Selection

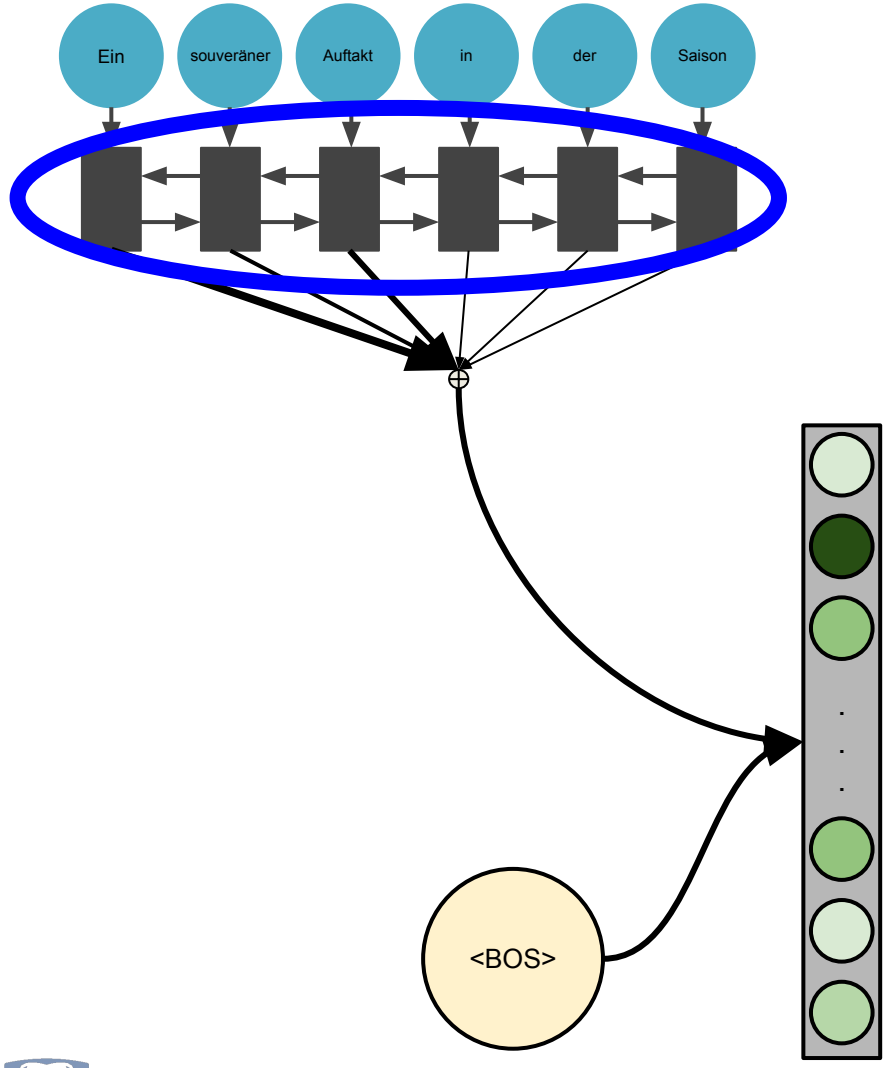
Highest probability word:

“confident”



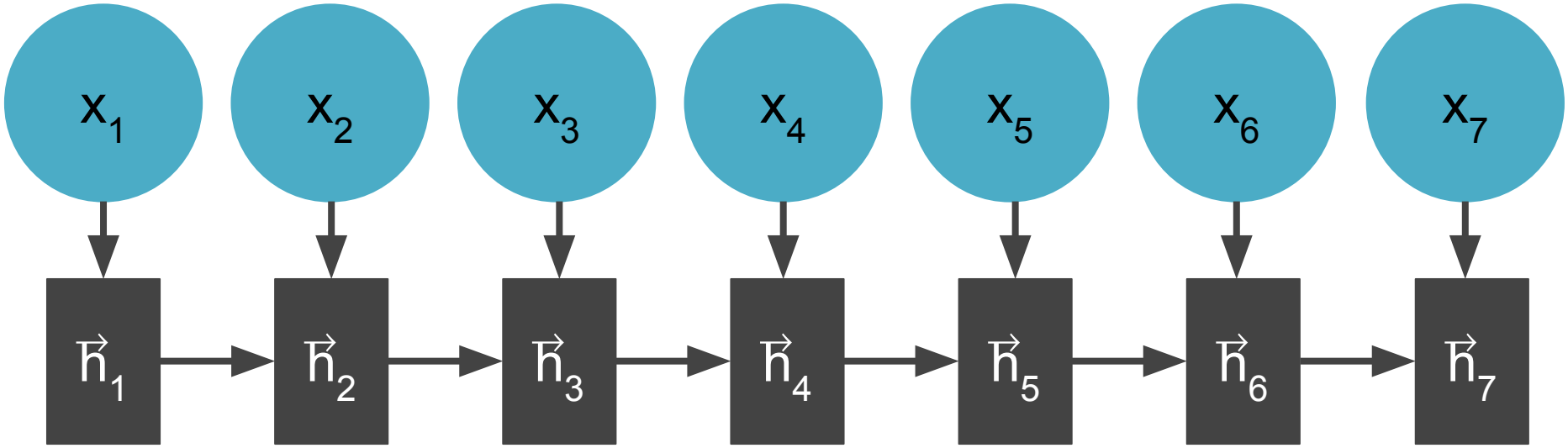
A confident start in the season

Attention: Bidirectional RNN



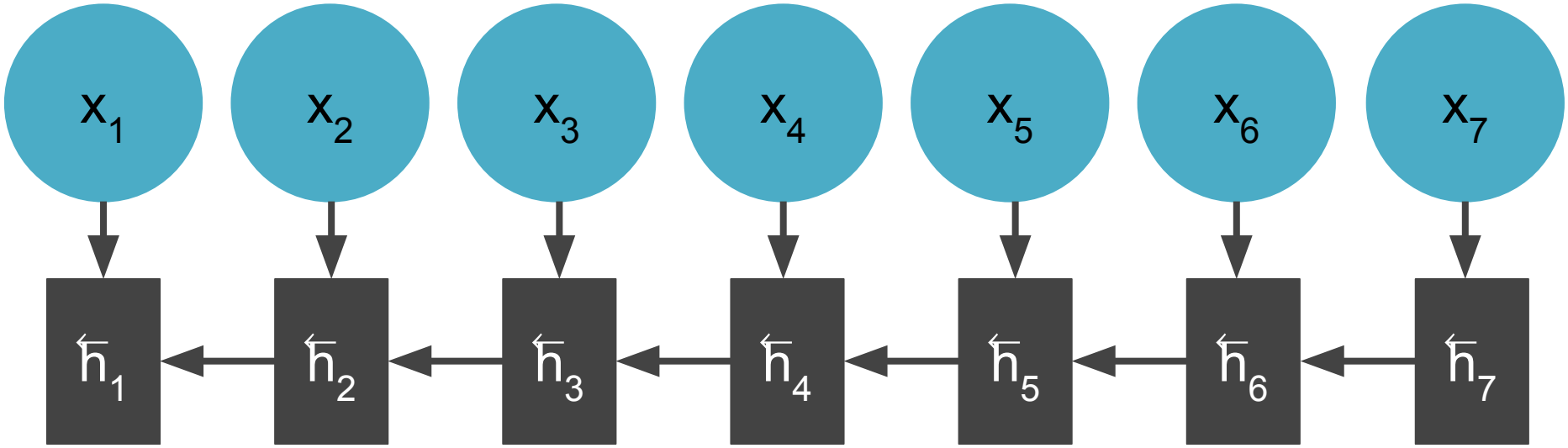
- Let hidden states contain more location-sensitive information
- Maintain information about the full sentence in the hidden states

RNN Encoder: Forward



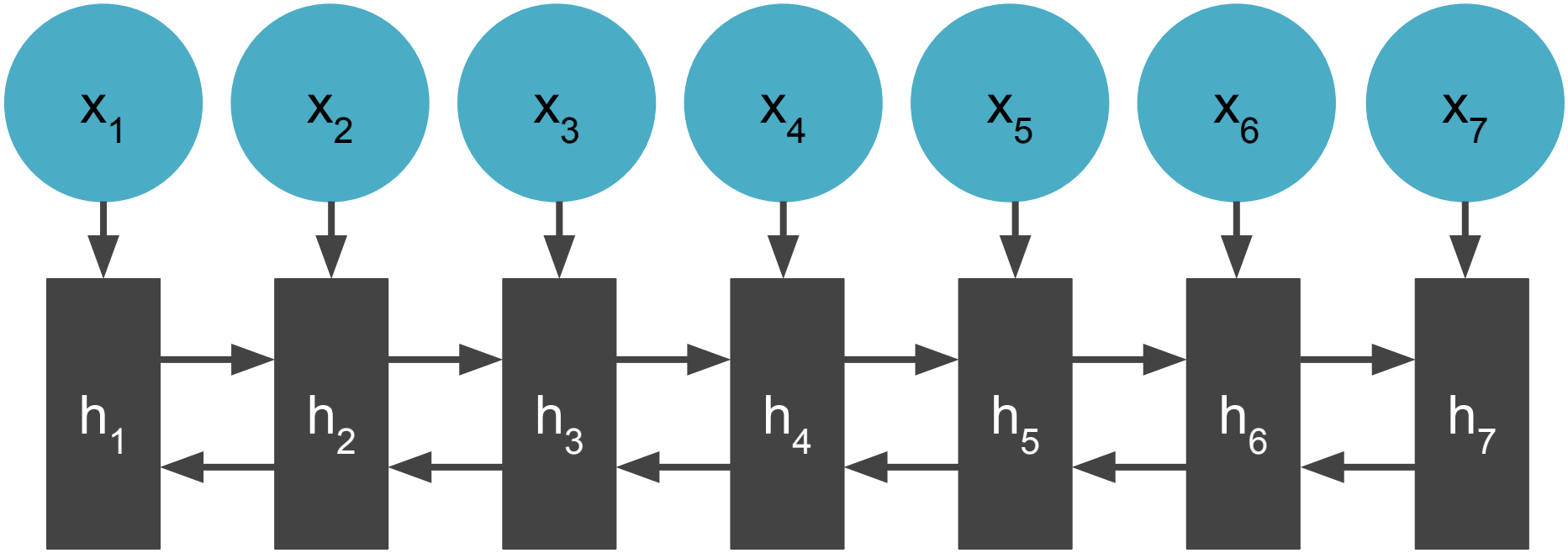
$$\vec{\mathbf{h}}_{(i)} = f(\vec{\mathbf{h}}_{(i-1)}, x_i)$$

RNN Encoder: Backward



$$\overleftarrow{\mathbf{h}}_{(i-1)} = f(\overleftarrow{\mathbf{h}}_{(i)}, x_i)$$

Concatenated

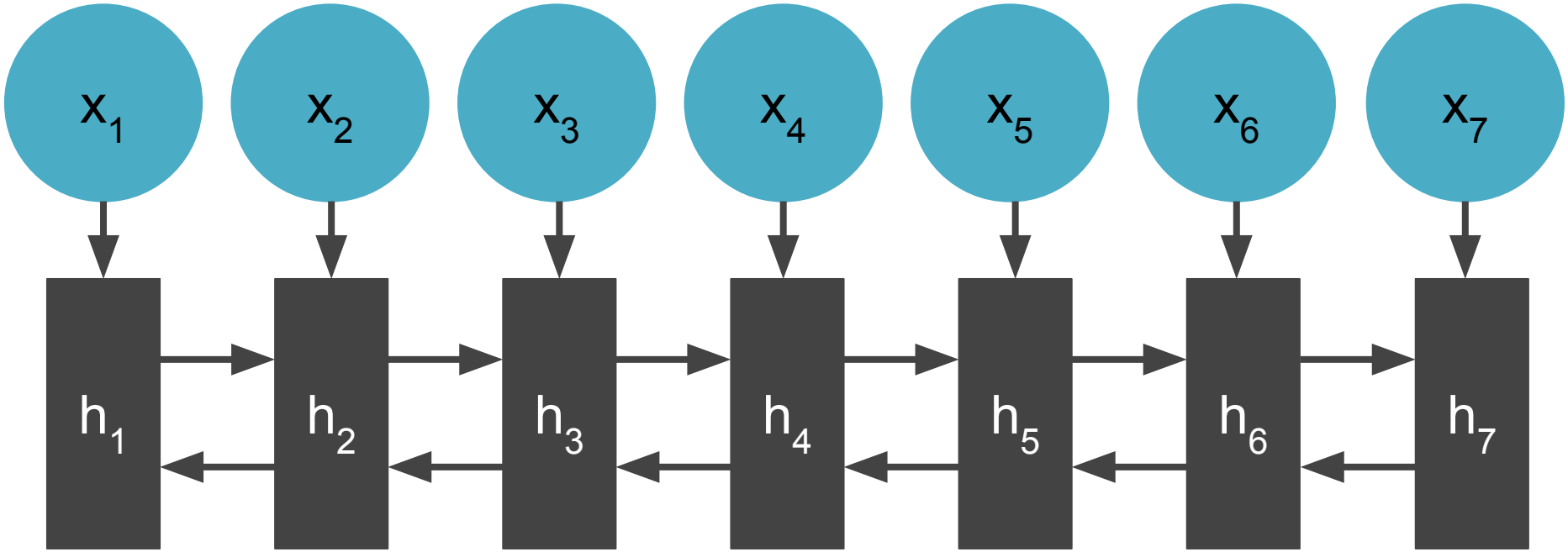


$$\mathbf{h}_{(i)} = \mathit{concat}(\overrightarrow{\mathbf{h}}_{(i)}, \overleftarrow{\mathbf{h}}_{(i)})$$

RNN Encoder: Concatenated

- By concatenating a forward RNN and a backward RNN, we have a bidirectional RNN
- f is a nonlinear function, likely a GRU or LSTM
- Each $\mathbf{h}_i$ contains information about the current word x_i, and its context (in both directions)
- Intuitively, we may expect that $\mathbf{h}_i$ will be particularly useful when translating x_i, but it may be useful for translating other words too

Concatenated

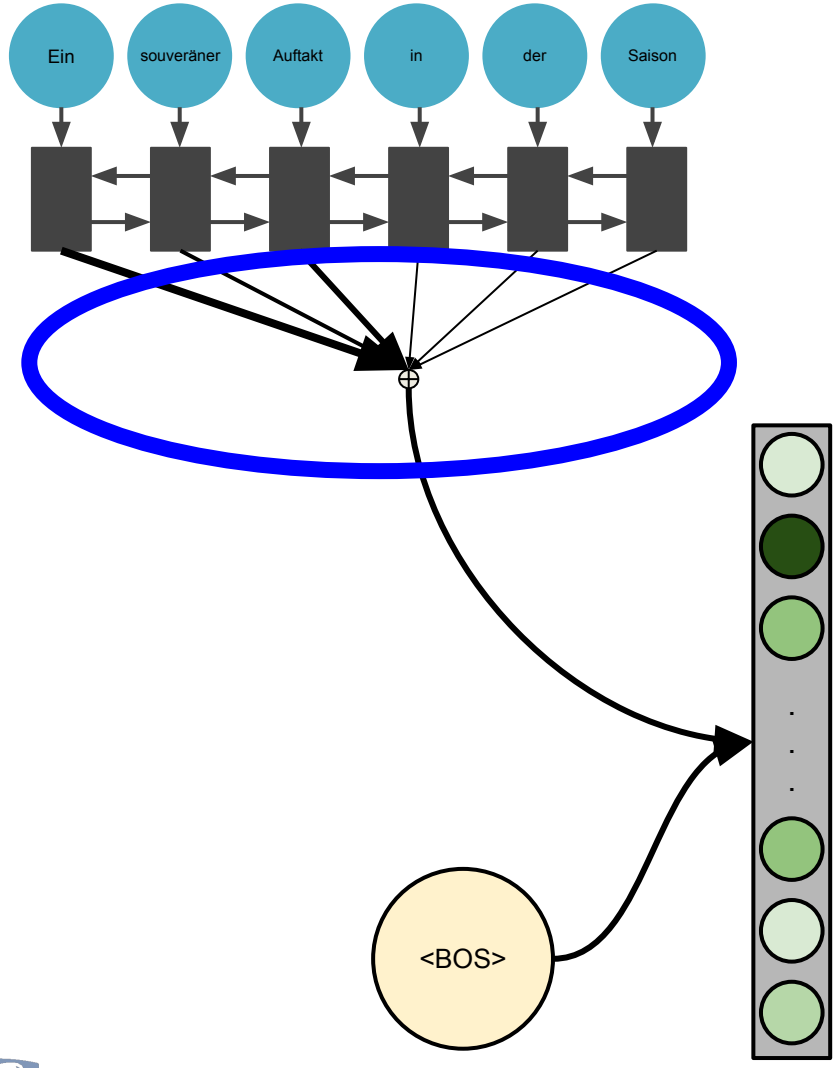


The “attention” or “soft alignments” will determine how much each hidden state (“annotation”) influences the generation of each word in the translation.

Attention Mechanism

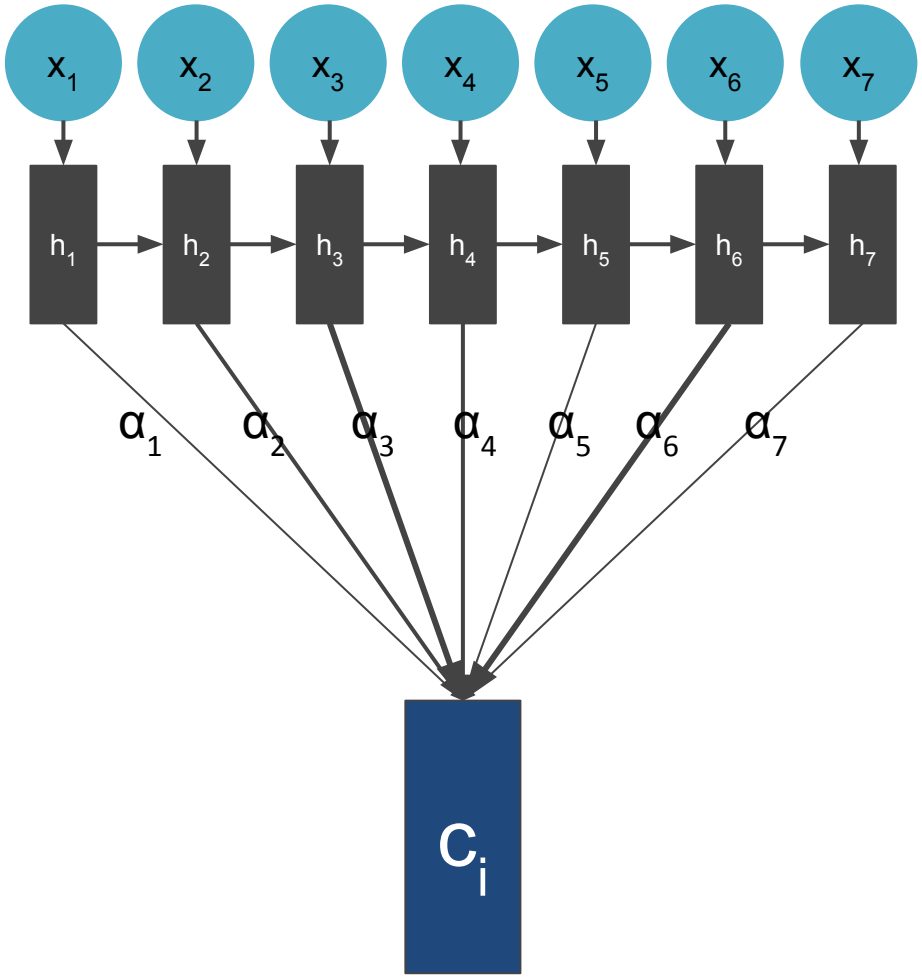
Equations from Bahdanau et al
(https://arxiv.org/pdf/1409.0473v7.pdf)

Attention Mechanism



- Context vectors
- Hidden states
- α

Context Vectors

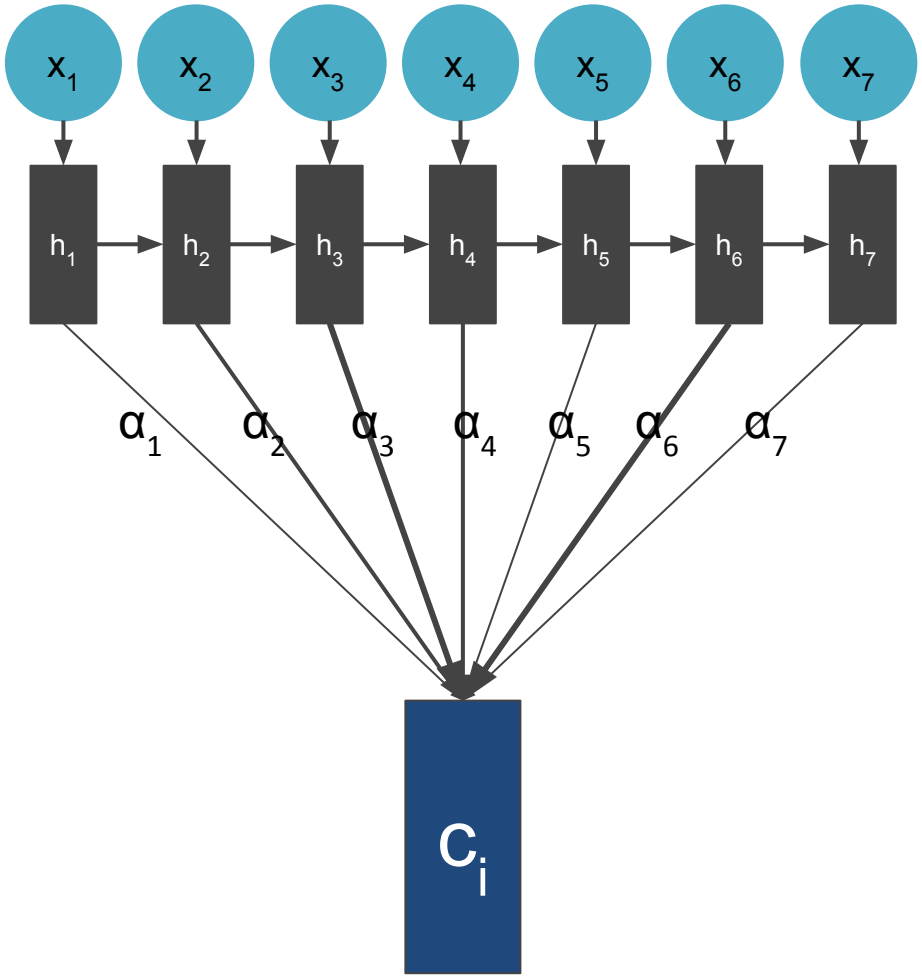


- Each time we generate a word y_i, we'll create a context vector, $\mathbf{c}_i$
- $\mathbf{c}_i$ is a weighted sum of the hidden states $\mathbf{h}_j$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

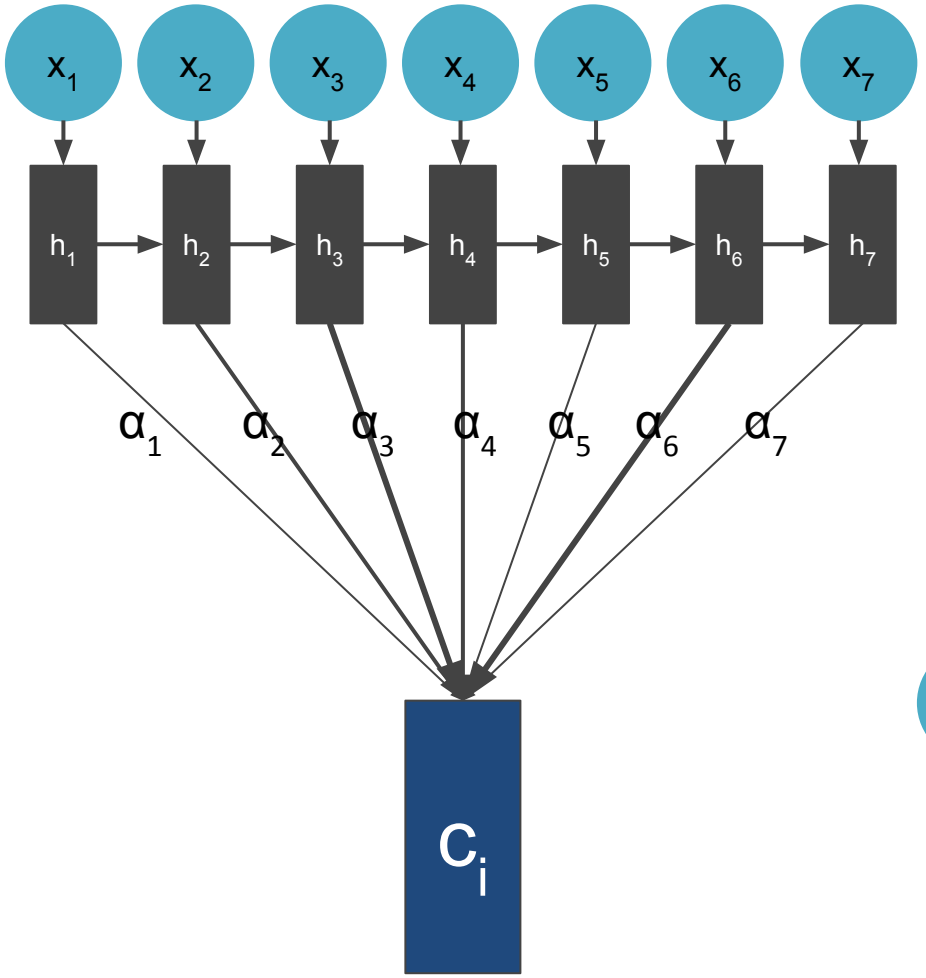
- (we'll look at how to compute α shortly)

Context Vectors

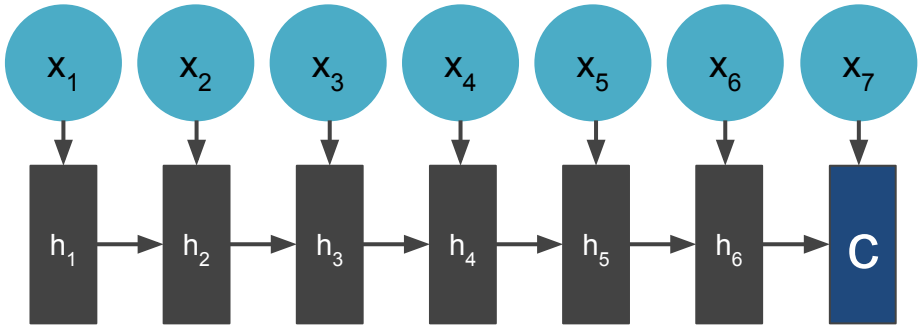


- Initialization: context vector $\mathbf{c}_0$ is the unweighted average of $\mathbf{h}_j$ for all j

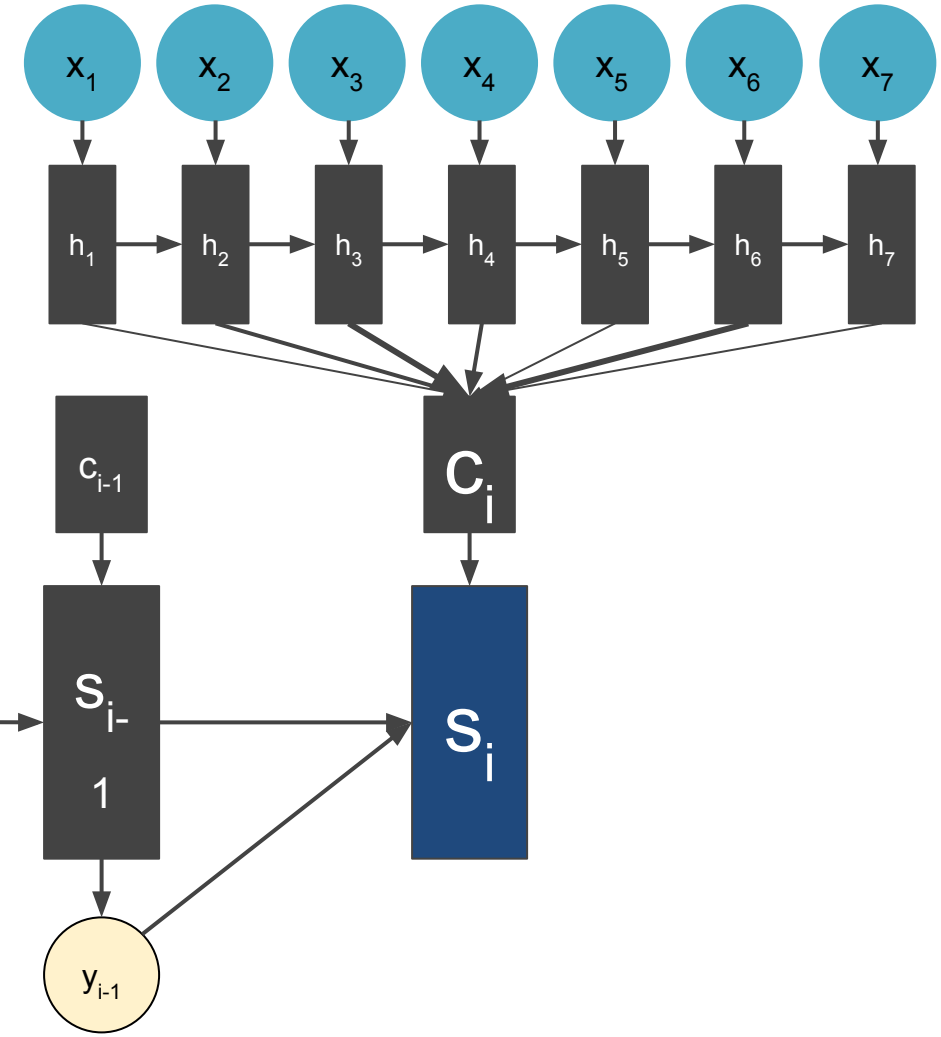
Context Vectors



- This differs from the earlier version we defined, which had a single **c** for the whole input sentence



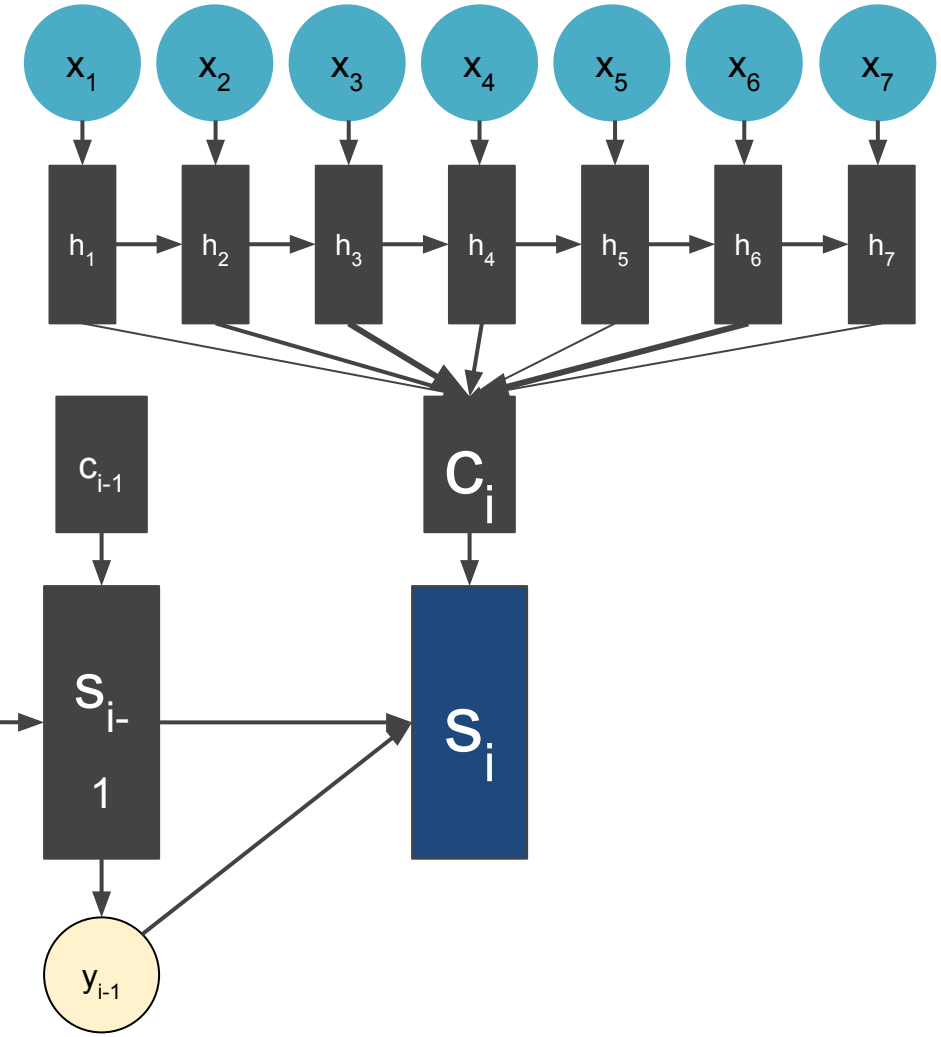
Hidden States



$$\mathbf{s}_i = f(\mathbf{s}_{(i-1)}, y_{(i-1)}, \mathbf{c}_i)$$

$\mathbf{s}_i$ is an RNN hidden state for time i

α

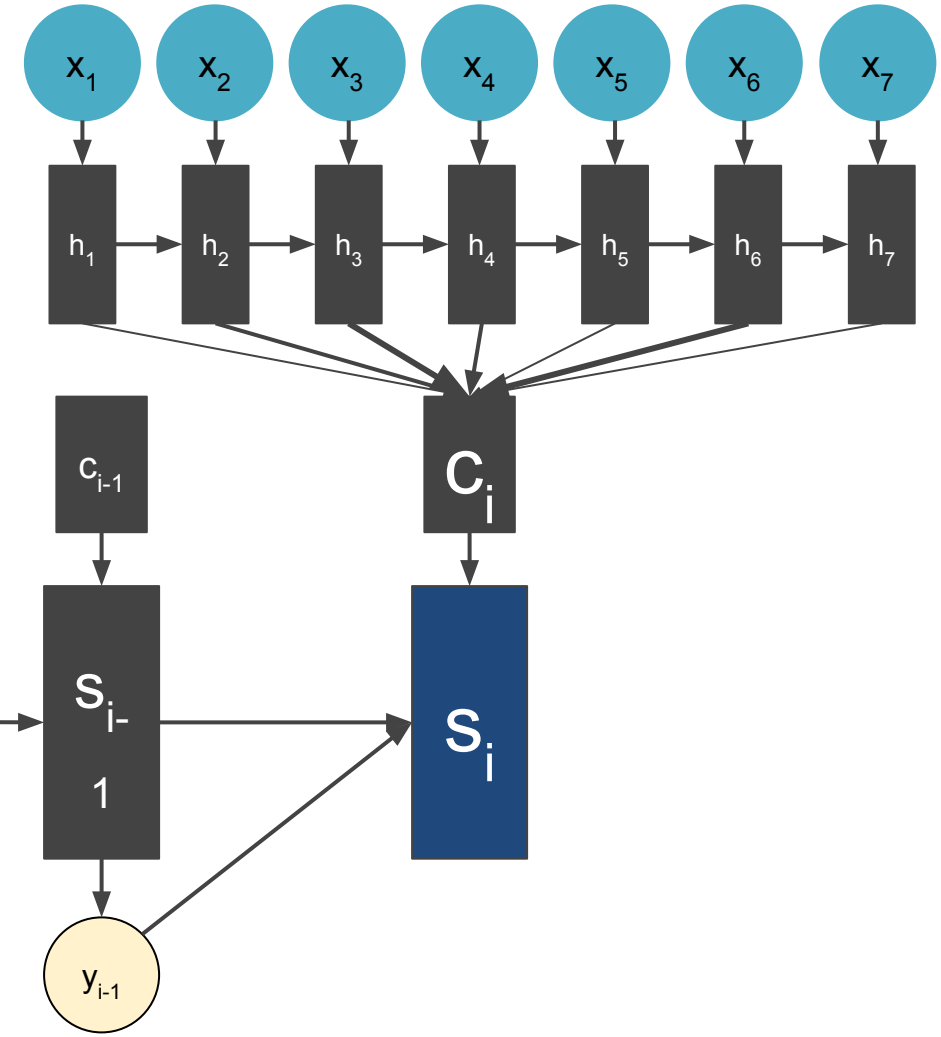


$$\mathbf{e}_{ij} = a(\mathbf{s}_{(i-1)}, \mathbf{h}_j)$$

a is a feed-forward neural net, our alignment model

Think of it as checking how similar inputs (around j) are to outputs (around i).

α

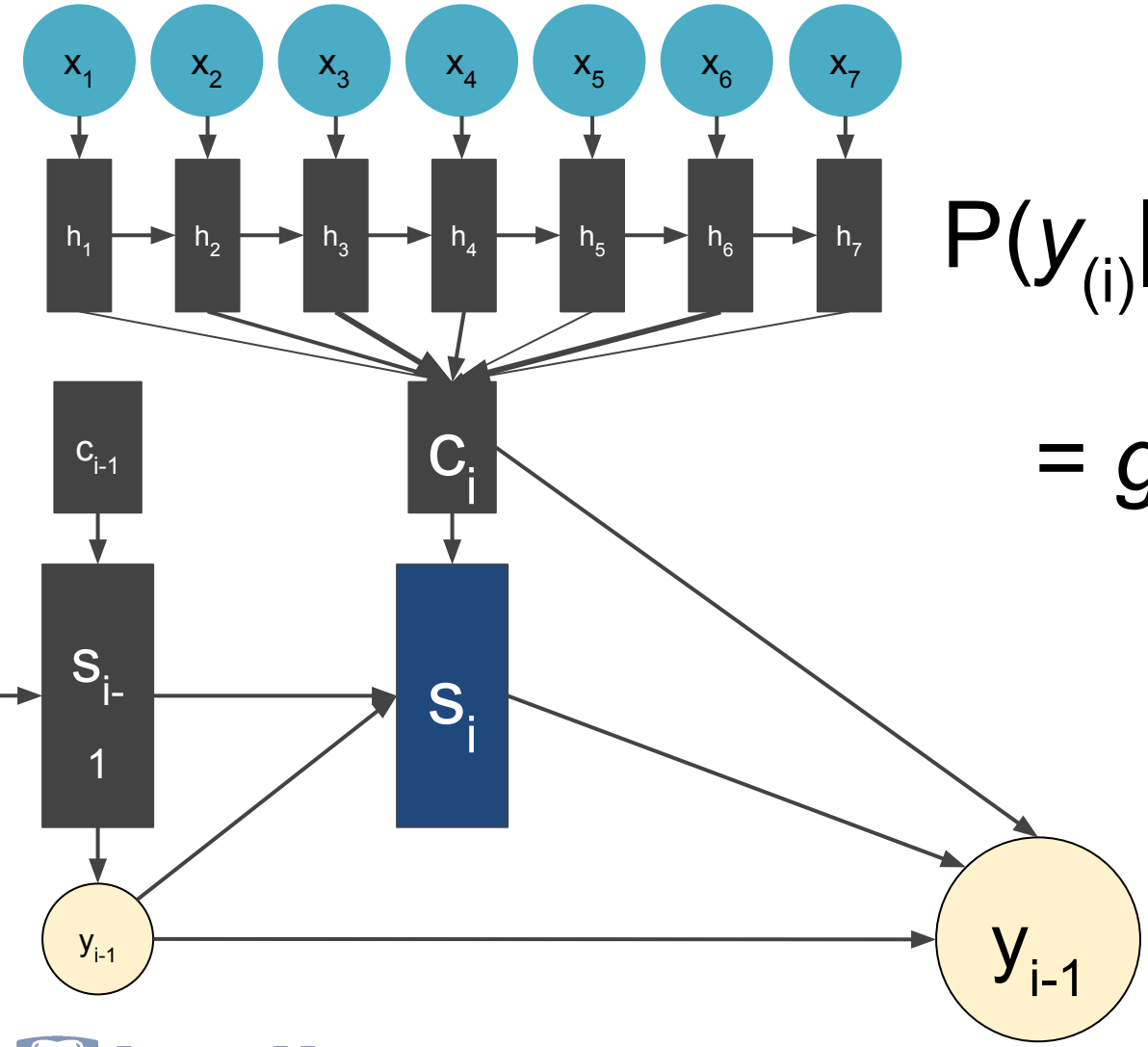


$$\mathbf{e}_{ij} = a(\mathbf{s}_{(i-1)}, \mathbf{h}_j)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

(softmax)

Generating y_i



$$P(y_{(i)}|y_{(i-1)},\ldots,y_1,\mathbf{x})$$

$$= g(y_{(i-1)}, \mathbf{s}_{(i)}, \mathbf{c}_i)$$

Attention Model Math

Encoding:

$$\overrightarrow{\mathbf{h}}_{(i)} = f(\overrightarrow{\mathbf{h}}_{(i-1)}, x_i)$$

$$\overleftarrow{\mathbf{h}}_{(i-1)} = f(\overleftarrow{\mathbf{h}}_{(i)}, x_i)$$

$$\mathbf{h}_{(i)} = concat(\overrightarrow{\mathbf{h}}_{(i)}, \overleftarrow{\mathbf{h}}_{(i)})$$

Attention Model Math

Attention: $\mathbf{e}_{ij} = a(\mathbf{s}_{(i-1)}, \mathbf{h}_j)$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

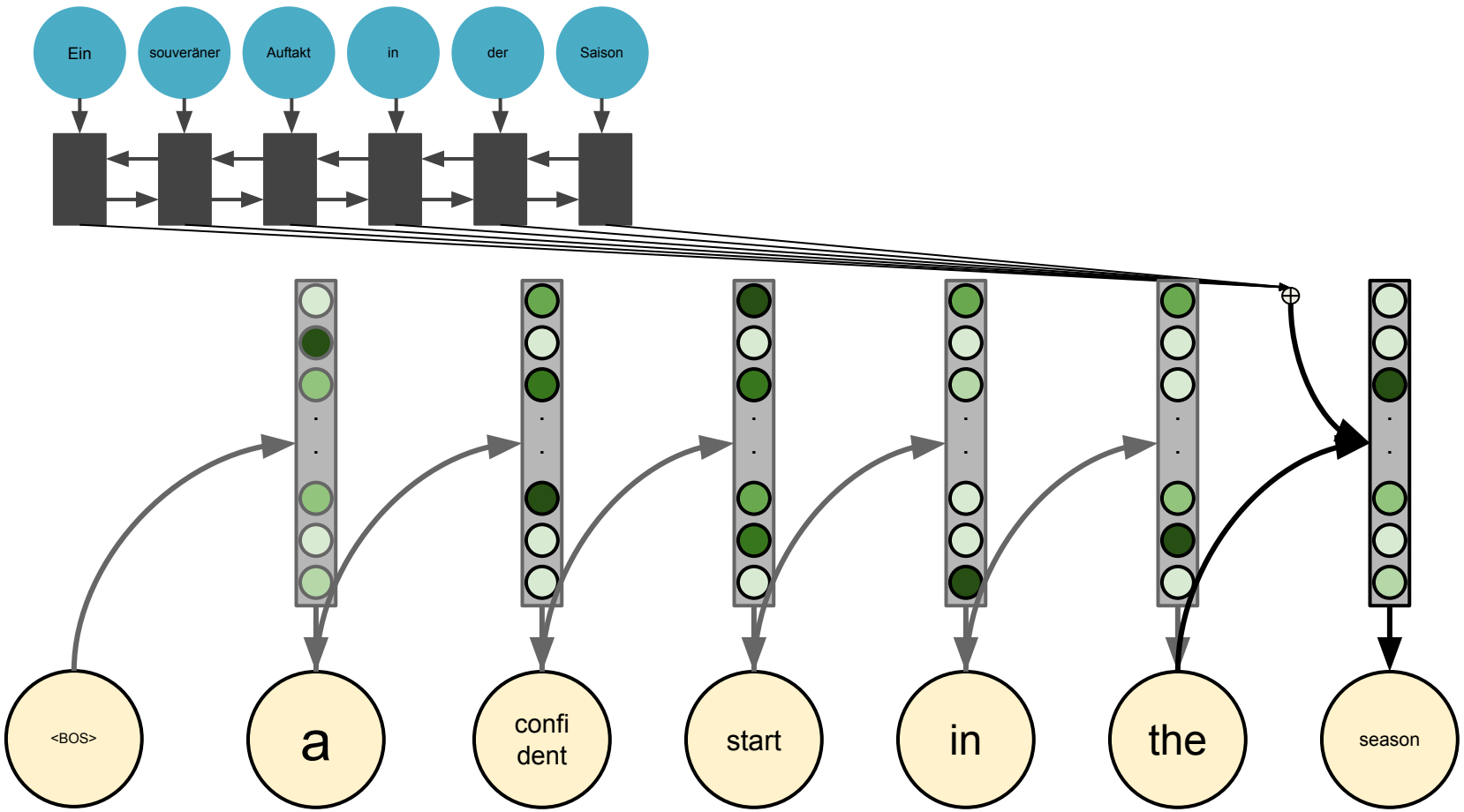
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\mathbf{s}_i = f(\mathbf{s}_{(i-1)}, y_{(i-1)}, \mathbf{c}_i)$$

Attention Model Math

Decoding:

$$P(y_{(i)}|y_{(i-1)},\ldots,y_1,\mathbf{x}) = g(y_{(i-1)}, \mathbf{s}_{(i)}, \mathbf{c_i})$$



A confident start in the season

Attention Code

Implemented in *gru_cond*

nematus/layers.py, lines 171-239, 242-377 (https://github.com/rsennrich/nematus/blob/9b08048e278db46ede288bcca928d176811ee2c2/nematus/layers.py)

Attention

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the International Conference on Learning Representations (ICLR). (https://arxiv.org/pdf/1409.0473v7.pdf)

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany. (https://www.aclweb.org/anthology/P/P16/P16-1162.pdf)

Rico Sennrich; Barry Haddow; Alexandra Birch (2016). Edinburgh Neural Machine Translation Systems for WMT 16. Proceedings of the First Conference on Machine Translation, Volume 2: Shared Task Papers. Berlin, Germany, pp. 368-373. (http://www.statmt.org/wmt16/pdf/W16-2323.pdf)

Other Useful Things To Know

Search

- In the examples, I've shown beam size of 1
  - Either sample from the probability distribution, or greedily choose the 1 best
  - Continue until reaching <eos> (or some max)
- Can also do beam search
  - Keep K hypotheses with highest probability
  - Expand each one at each step

Search

- For neural models, small beam sizes
- Beam size ~12 seems to work best, but:
  - most improvement seems to come even with just beam size ~2
  - it's a bit slower

Ensembling

- Average the probability distributions of different models in order to generate words in translation
- Models must have same output vocab and factorization of **y**
- Can ensemble different training runs, different initializations, or ensemble using checkpoints from one run

Synthetic Training Data

- Dummy sentences
  - Given large monolingual target-language corpus.
  - "Translate" empty source sentences into target monolingual sentences.
  - Freeze encoder/attention parameters.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany. (https://arxiv.org/abs/1511.06709)

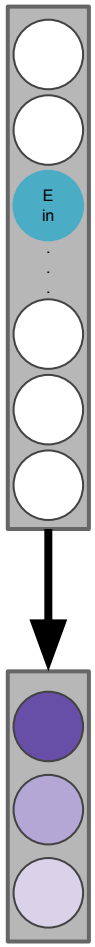
Synthetic Training Data

- Back-translation
  - Given a large target-language monolingual corpus, use an MT system to "back-translate" it into the source language.
  - Train on both human-translated parallel text and this MT-generated parallel text.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving Neural Machine Translation Models with Monolingual Data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany. (https://arxiv.org/abs/1511.06709)

Additional Linguistic Features

- Add additional linguistic features to input
- Recall that vocab items are represented as lower-dimensional vectors of continuous values; do the same with additional features ("factored NMT")



Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany. (https://www.aclweb.org/anthology/W/W16/W16-2209.pdf)

Problem: Vocabulary

- Neural models tend to have vocabulary sizes <50,000
- Want to be able to handle rare words
- May want to be able to generate novel words (for example, in morphologically complex languages)

Byte-Pair Encoding (BPE)

word	freq
‘l o w </w>’	5
‘l o w e r </w>’	2
‘n e w e s t </w>’	6
‘w i d e s t </w>’	3

- Treat each word as a sequence of character tokens (plus word-end token)
- Collect into dictionary (efficiency)
- Given dictionary and desired number of merges, merge most frequent pairs into symbols
- Return new character + symbol vocabulary
- Example from Sennrich, Birch, Junczys-Dowmunt (http://homepages.inf.ed.ac.uk/rsennric/amta2016-tutorial.pdf)

Byte-Pair Encoding (BPE)

word	freq
‘l o w </w>’	5
‘l o w e r </w>’	2
‘n e w **es** t </w>’	6
‘w i d **es** t </w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	**‘es’**

Byte-Pair Encoding (BPE)

word	freq
‘l o w </w>’	5
‘l o w e r </w>’	2
‘n e w **est** </w>’	6
‘w i d **est** </w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	**‘est’**

Byte-Pair Encoding (BPE)

word	freq
‘l o w </w>’	5
‘l o w e r </w>’	2
‘n e w **est</w>**’	6
‘w i d **est</w>**’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	**‘est</w>’**

Byte-Pair Encoding (BPE)

word	freq
‘**lo** w </w>’	5
‘**lo** w e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	**‘lo’**

Byte-Pair Encoding (BPE)

word	freq
‘**low** </w>’	5
‘**low** e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	**‘low’**
...			

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	‘low’
...			

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	**(‘e’, ‘s’)**	**→**	**‘es’**
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	‘low’
...			

‘l o w e s t </w>’ →

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	**(‘e’, ‘s’)**	→	**‘es’**
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	‘low’
...			

‘l o w e s t </w>’ → ‘l o w **es** t </w>’

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	**(‘es’,’t’)**	→	**‘est’**
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	‘low’
...			

‘l o w es t </w>’ → ‘l o w **est** </w>’

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	**(‘est’,’</w>’)**	**→**	**‘est</w>’**
7	(‘l’,’o’)	→	‘lo’
7	(‘lo’,’w’)	→	‘low’
...			

‘l o w est </w>’ → ‘l o w **est**</w>’

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	**(‘l’,’o’)**	**→**	**‘lo’**
7	(‘lo’,’w’)	→	‘low’
...			

‘l o w est</w>’ → ‘**lo** w est</w>’

Byte-Pair Encoding (BPE)

word	freq
‘low </w>’	5
‘low e r </w>’	2
‘n e w est</w>’	6
‘w i d est</w>’	3

freq	symbol pair		new symbol
9	(‘e’, ‘s’)	→	‘es’
9	(‘es’,’t’)	→	‘est’
9	(‘est’,’</w>’)	→	‘est</w>’
7	(‘l’,’o’)	→	‘lo’
7	**(‘lo’,’w’)**	**→**	**‘low’**
...			

‘lo w est</w>’ → ‘**low** est</w>’

Alternatives

- Character-Level Models
  - Costa-Jussà and Fonollosa, ACL 2016 (https://aclweb.org/anthology/P/P16/P16-2058.pdf)
- Hybrid Word-Character Models
  - Luong and Manning, ACL 2016 (https://www.aclweb.org/anthology/P/P16/P16-1100.pdf)
- Other segmentations

Byte-Pair Encoding (BPE)

Philip Gage. 1994. A New Algorithm for Data Compression. C Users J., 12(2):23–38, February.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany. (https://www.aclweb.org/anthology/P/P16/P16-1162.pdf)

Code: https://github.com/rsennrich/subword-nmt

Example:

http://homepages.inf.ed.ac.uk/rsennric/amta2016-tutorial.pdf

Thank You

Contact:

Rebecca Knowles

rknowles@jhu.edu

www.cs.jhu.edu/~rknowles/

This work was supported, in part, by a National Science Foundation Graduate Research Fellowship under Grant No. DGE-1232825. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect those of the sponsors.