732A78 Deep Learning (2023)

# Recurrent neural networks

Marcel Bollmann
Department of Computer and Information Science (IDA)

LINKÖPING UNIVERSITY

Slides by Marco Kuhlmann

# Watch Marco's lectures from last year

youtube.com/playlist?list=PLvWwkcdbWwLWIyTnP0G0k7U8f82OCtgwx

(Main difference: Lab content!)

# Prelude: Word embeddings

# Sentiment analysis

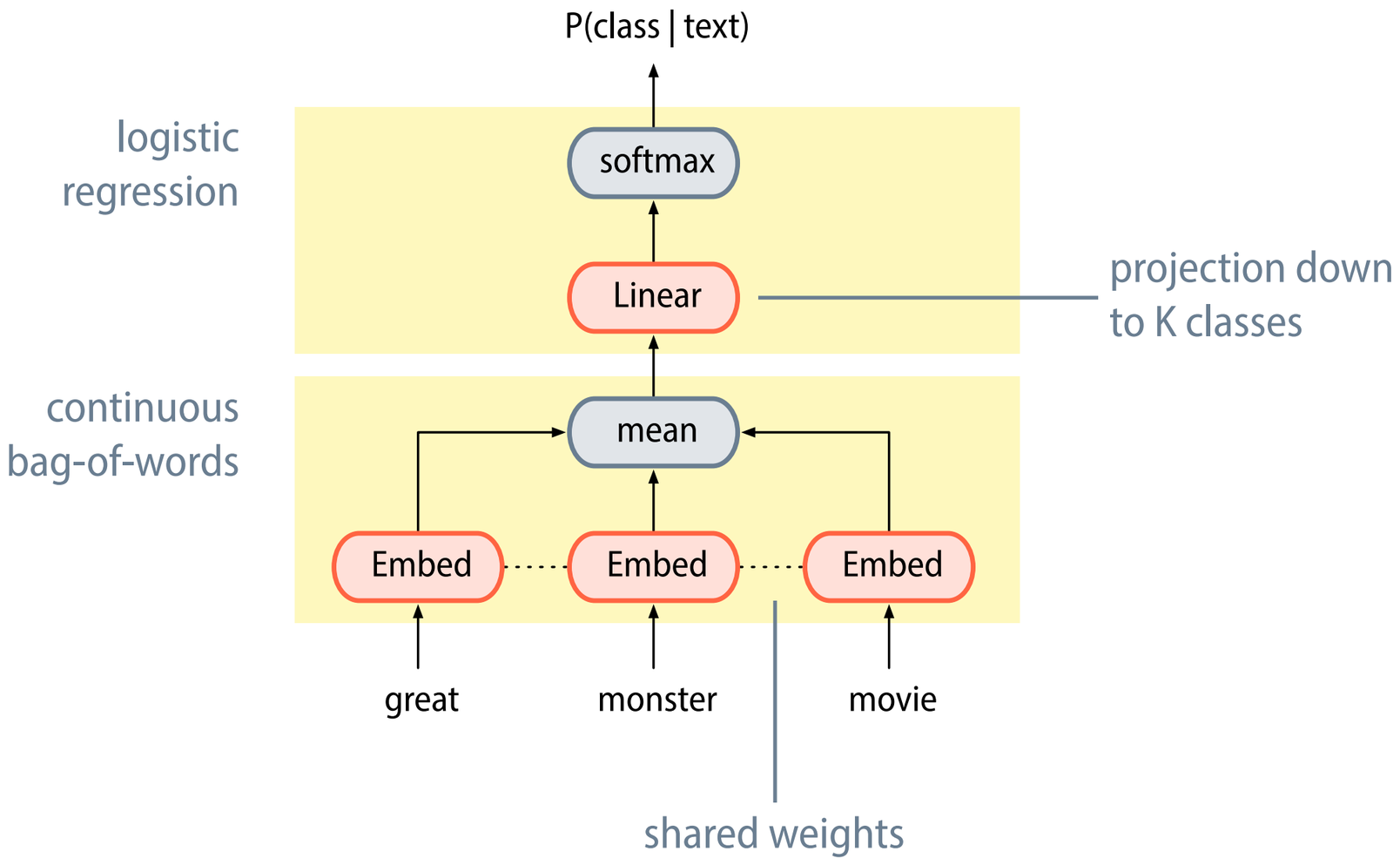
The gorgeously elaborate continuation of “The Lord of the Rings” trilogy is so huge that a column of words cannot adequately describe co-writer/director Peter Jackson’s expanded vision of J.R.R. Tolkien’s Middle-earth

positive

… is a sour little movie at its core; an exploration of the emptiness that underlay the relentless gaiety of the 1920’s, as if to stop would hasten the economic and global political turmoil that was to come.

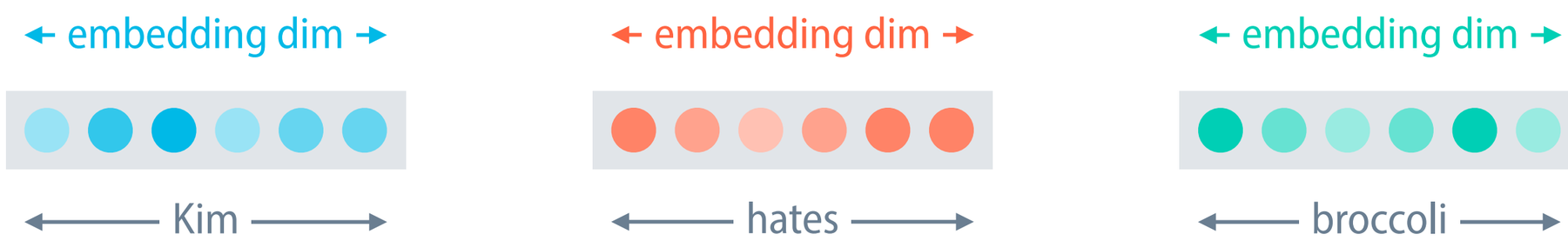
negative

# The continuous bag-of-words (CBOW) classifier



# Word embeddings

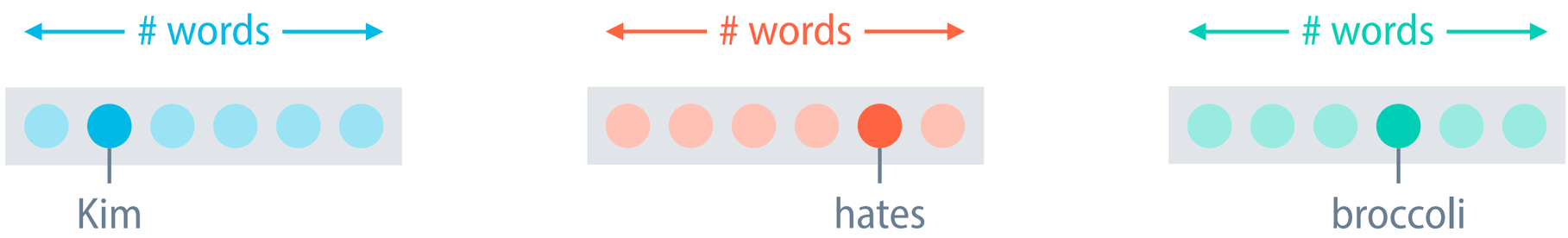
- A **word embedding** is a mapping from a finite set of words $V$ to a $d$-dimensional vector space, where $d \ll |V|$.




- Embedding vectors can be initialised with random numbers and trained alongside the weights of a network.
- Training tunes the embedding vectors to the task at hand.

# Embedding layers are linear transformations

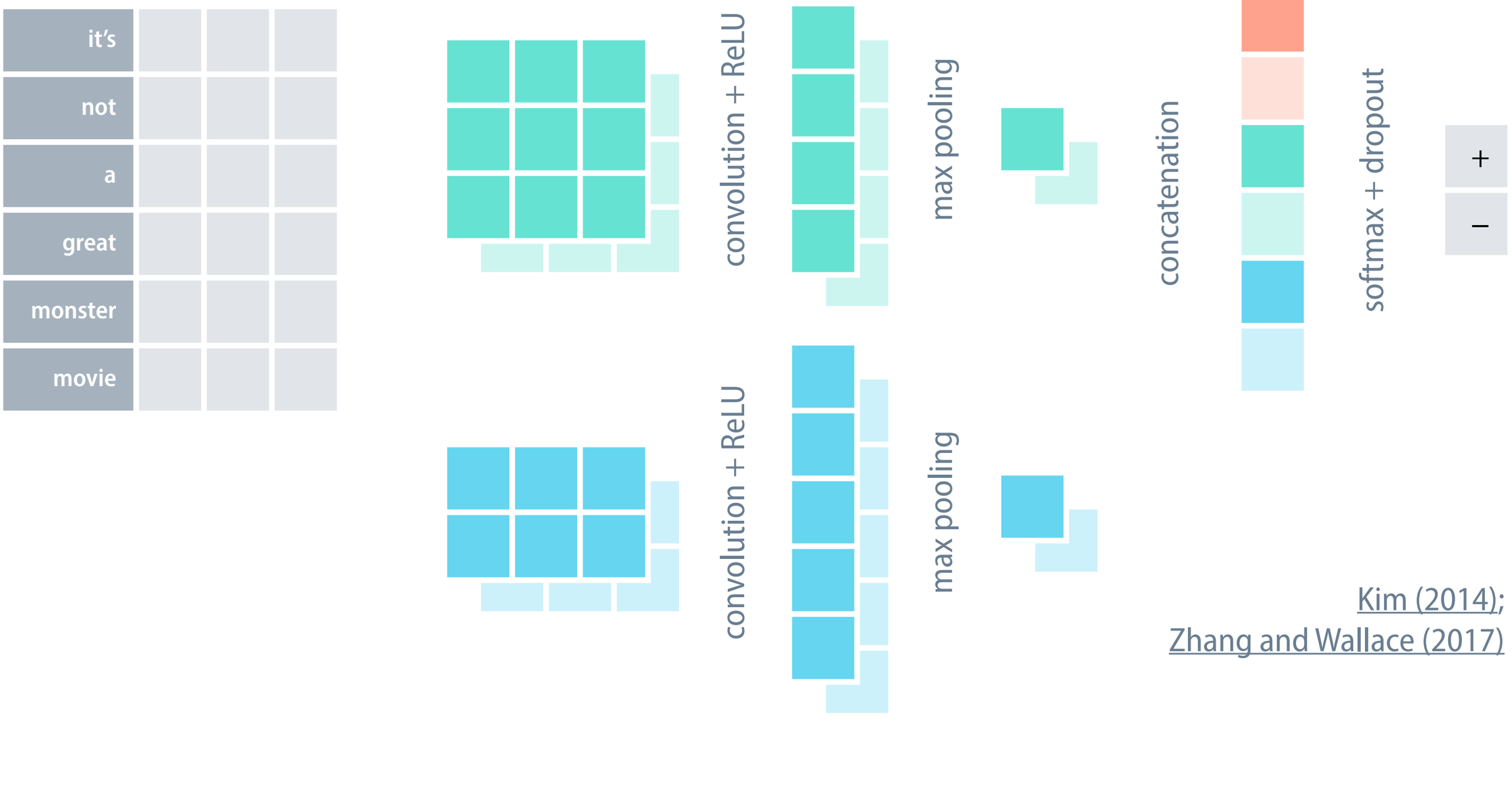
- A **one-hot vector** is a vector where one component has value 1, and all other components have value 0.




- A word embedding can be viewed as a linear transformation from one-hot vectors into the $d$-dimensional embedding space.

  values of the embedding vector = weights for the non-zero component

# CNN architecture for sentence classification




Kim (2014);
Zhang and Wallace (2017)

# Structure of this unit

- Prelude:  Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1:  Part-of-speech tagging
- Use case 2:  Machine translation
- Attention

# Introduction to recurrent neural networks

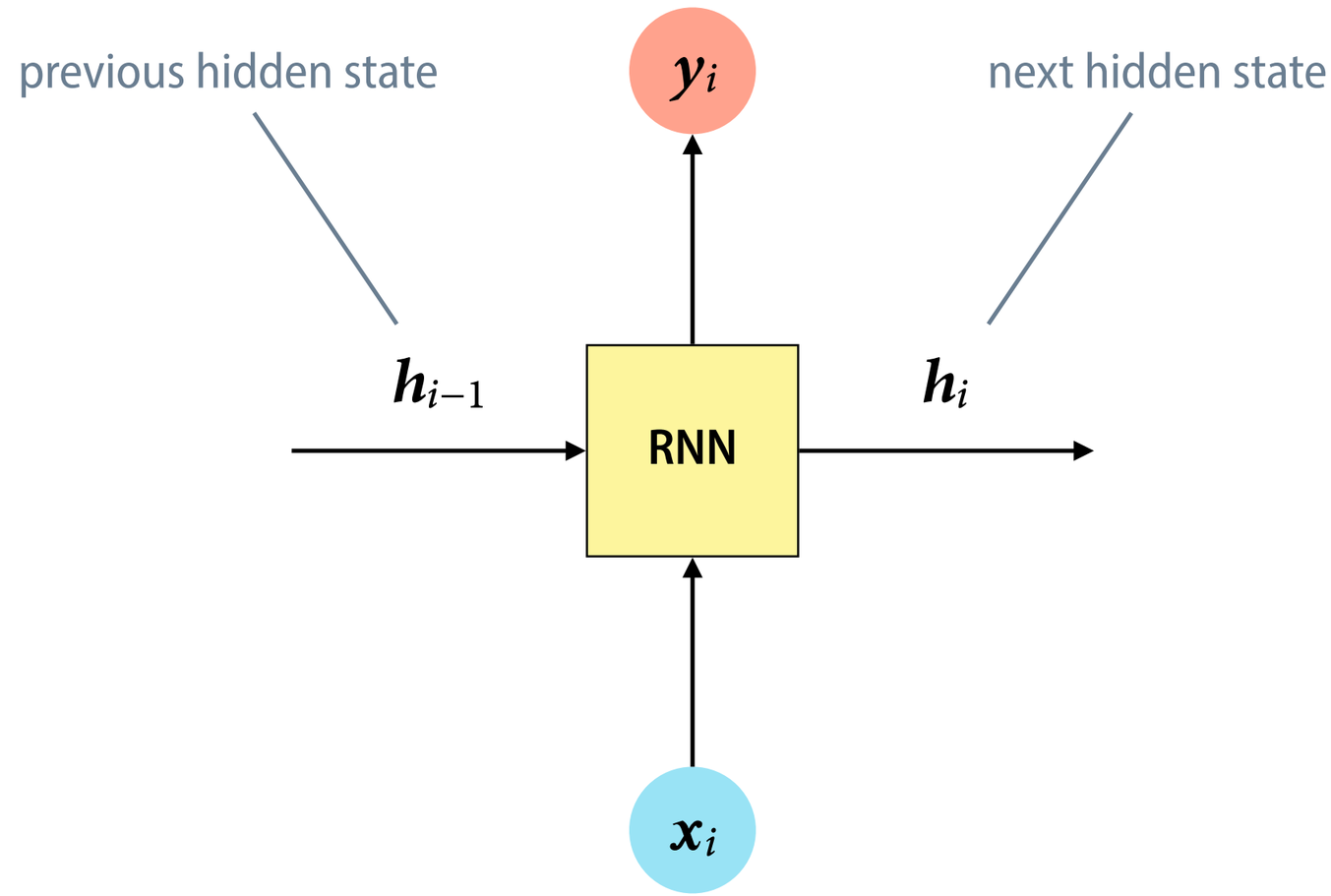
# Recurrent neural networks

- **Recurrent neural networks (RNNs)** can process variable length sequences of inputs, such as sequences of letters or words.
- For any input sequence, a recurrent neural network is ‘unrolled’ into a deep feedforward network.

  Depth is proportional to the length of the sequence.

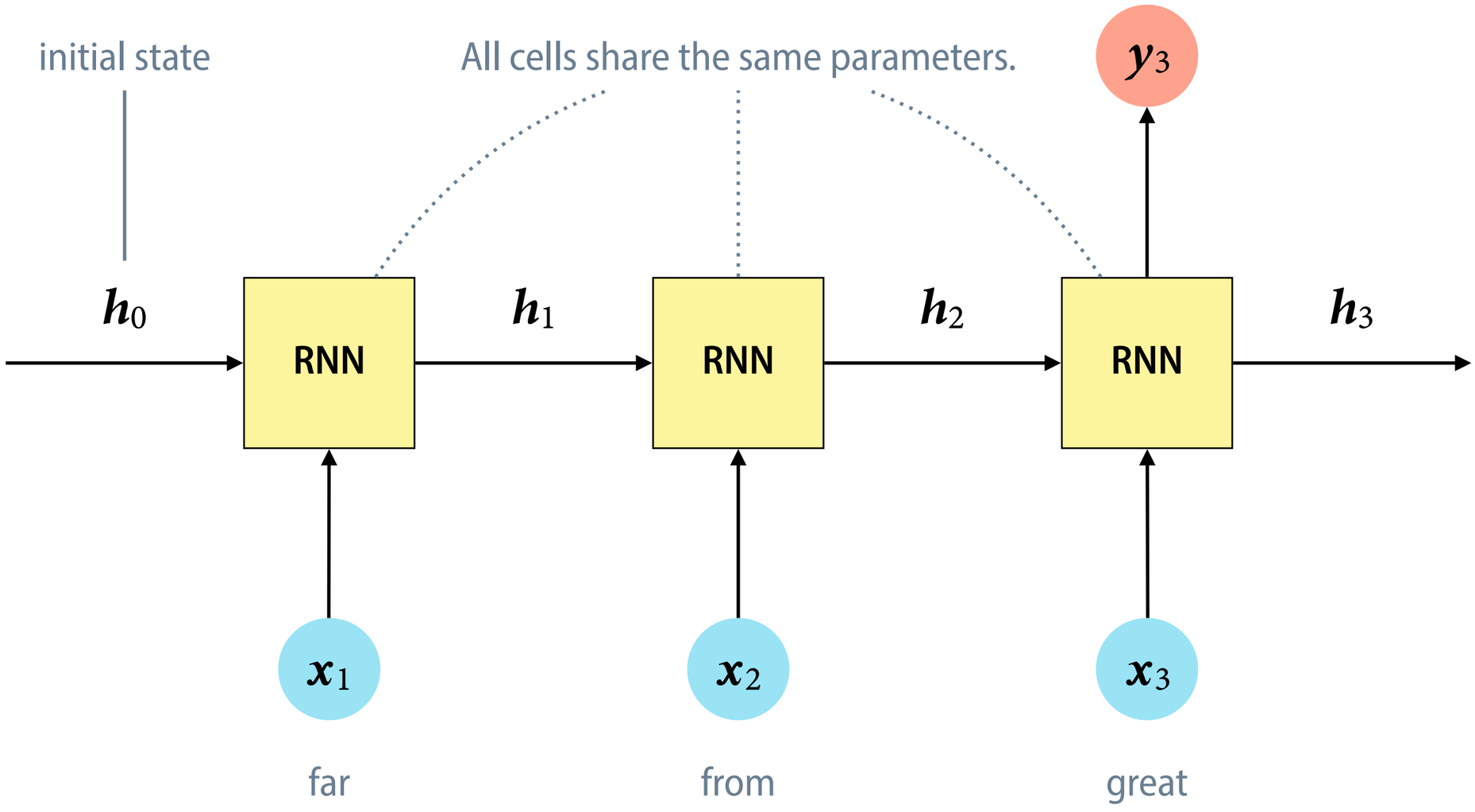
- In contrast to the situation with deep feedforward networks, all parameters are shared across all positions of the sequence.

# RNN, recursive view




$$h_i = H(h_{i-1}, x_i) \qquad y_i = O(h_{i-1}, x_i)$$

# RNN, unrolled view



# Properties of recurrent neural networks

- The parameters of the model are shared across all positions.

  The number of parameters does not grow with the sequence length.

- The output can be influenced by the entire input seen so far.

  Contrast this with the locality constraint of CNNs.

- The hidden state can be a ‘lossy summary’ of the input sequence.

  Hopefully, it will encode useful information for the task at hand.

# Training recurrent neural networks

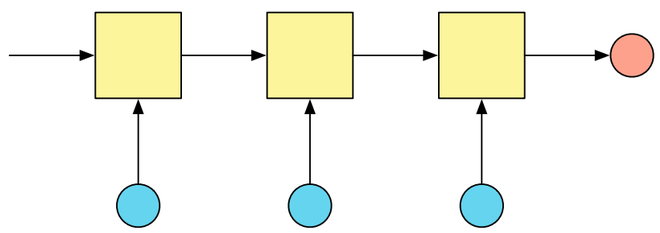
- Unrolled recurrent neural networks are just feedforward networks, and can therefore be trained using backpropagation.

  No specialised algorithm necessary!

- This way of training recurrent neural networks is called **backpropagation through time**.

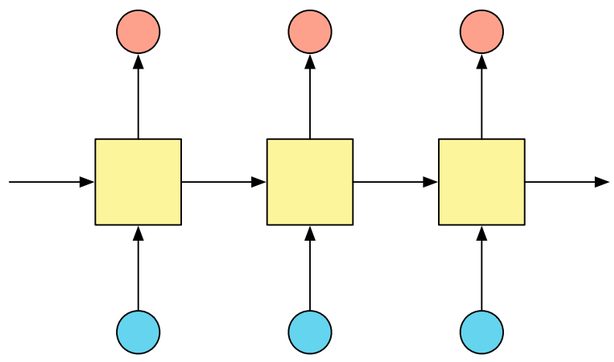
- Shared weights are updated by summing over the gradients computed for each position.

# Common usage patterns for RNNs



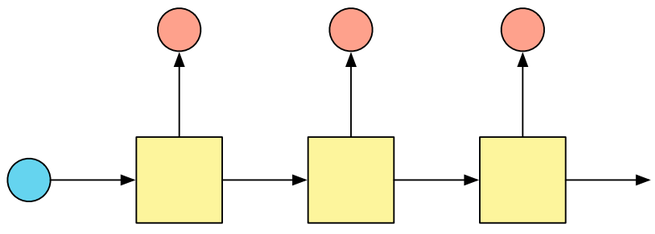
## encoder

example: text classification



## transducer

example: part-of-speech tagging

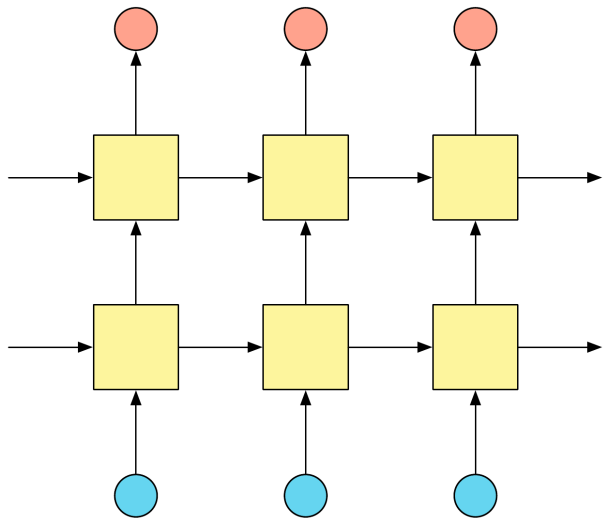


## decoder

example: text generation

# Stacked RNNs

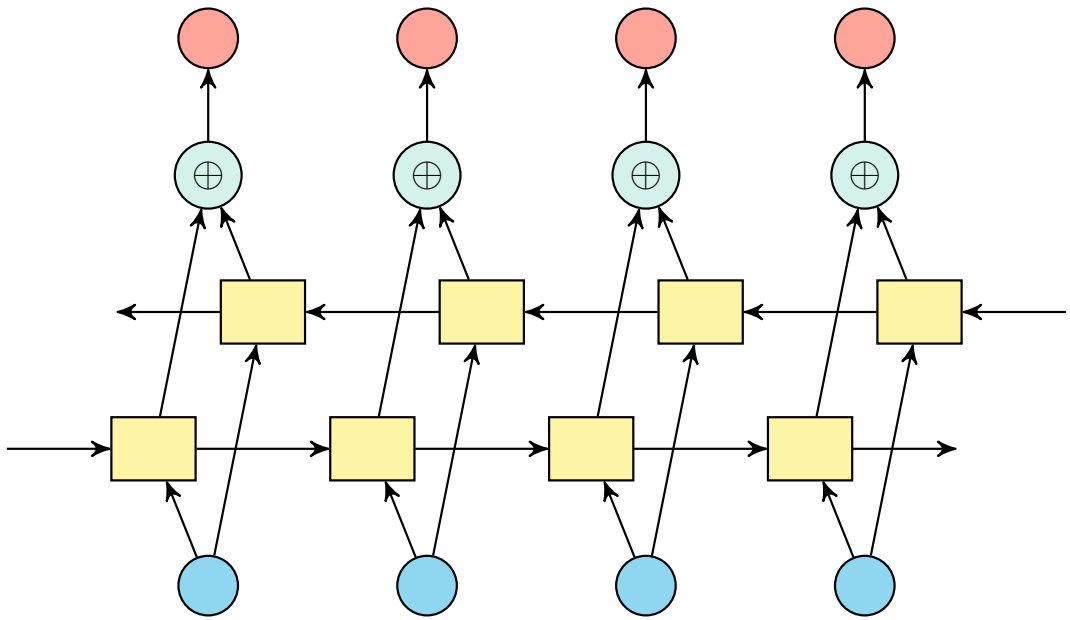
- RNNs with several layers, where the outputs of one layer become the inputs of the next.



# Bidirectional RNNs

- Combine one RNN that moves forward through the input with another RNN that moves backward.

  outputs at each position are concatenated

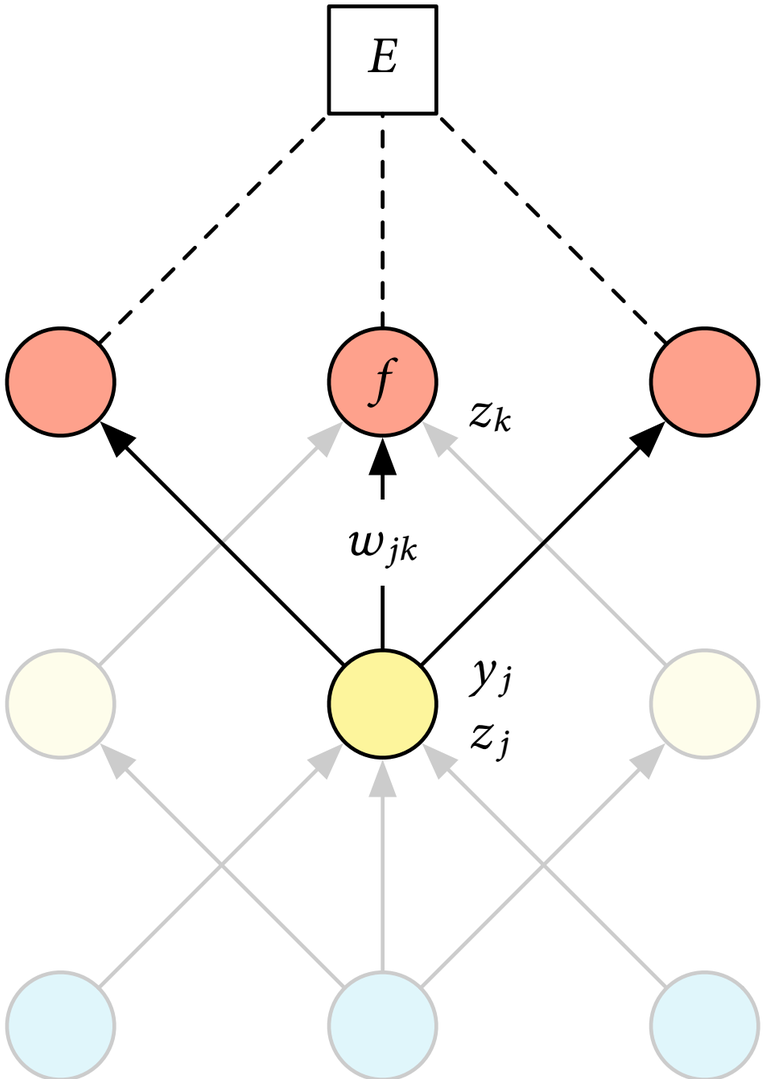


# The LSTM architecture

# Challenges with recurrent neural networks

- In principle, recurrent neural networks are capable of learning long-distance dependencies in input sequences.
- In practice, training recurrent neural networks is challenging due to the large depth of the unrolled networks.

# Vanishing and exploding gradients




$$\delta_k = \frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k} = \frac{\partial E}{\partial y_k} f'(z_k)$$

$$\delta_j = \frac{\partial E}{\partial z_j} = \frac{\partial y_j}{\partial z_j}\sum_k \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial y_j} = f'(z_j)\sum_k \delta_k w_{jk}$$

# Vanishing and exploding gradients

- In backpropagation there is a risk of gradients either vanishing or exploding, depending on the magnitude of the weights.
- This problem is exacerbated in recurrent networks, whose unrolled computation graphs can be very deep.
- Research on recurrent networks has proposed various methods to mitigate this problem.

  weight scaling and clipping, specialised architectures

# Long Short-Term Memory (LSTM)

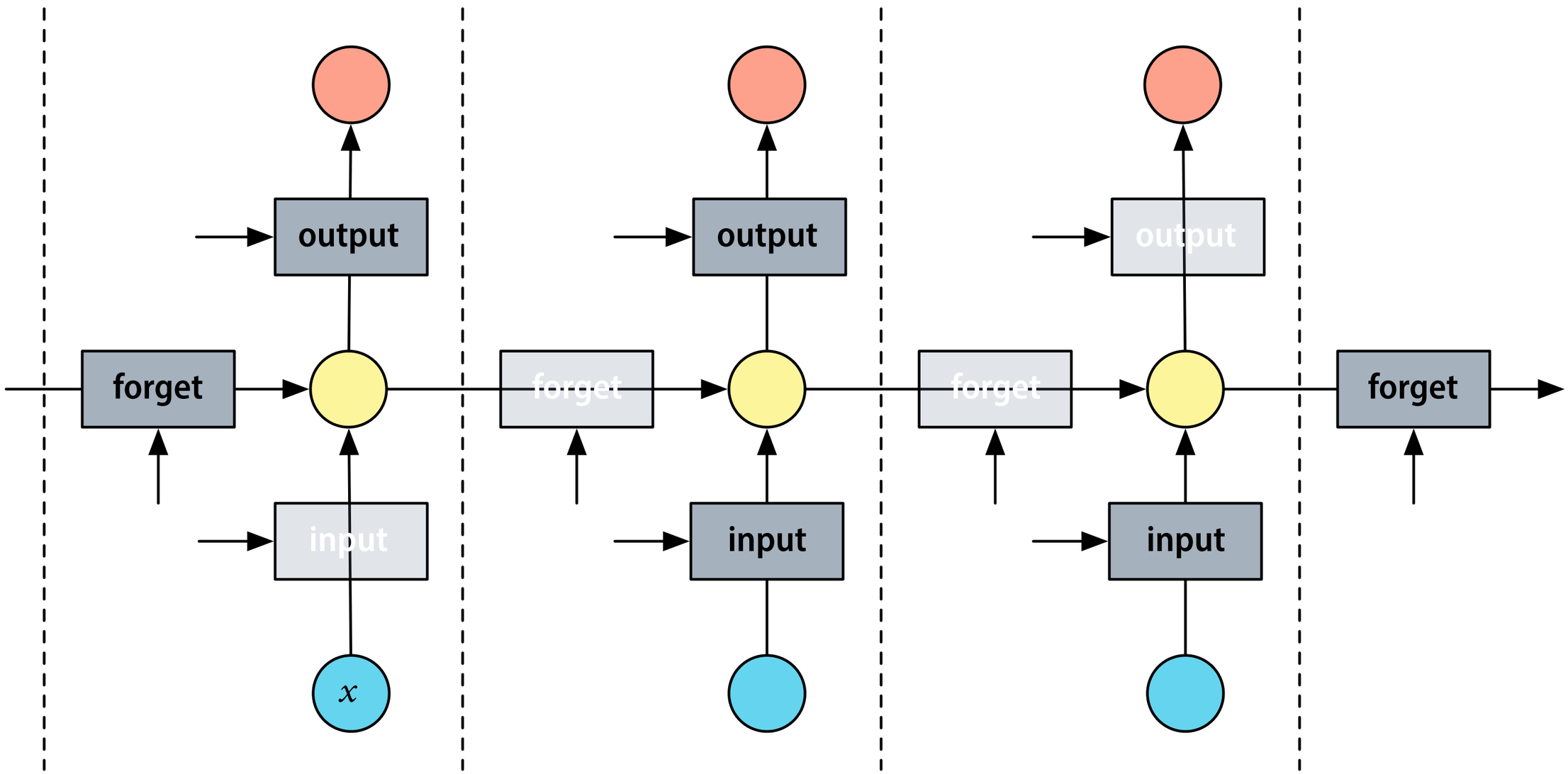
- The **Long Short-Term Memory (LSTM)** architecture was specifically designed to adress the vanishing gradients problem.
  - Metaphor: The hidden state of the neural network can be considered as a short-term memory.
  - The LSTM architecture tries to make this short-term memory last as long as possible by preventing vanishing gradients.

# Memory cell and gating mechanism

The crucial innovation in an LSTM is the design of its memory cell.

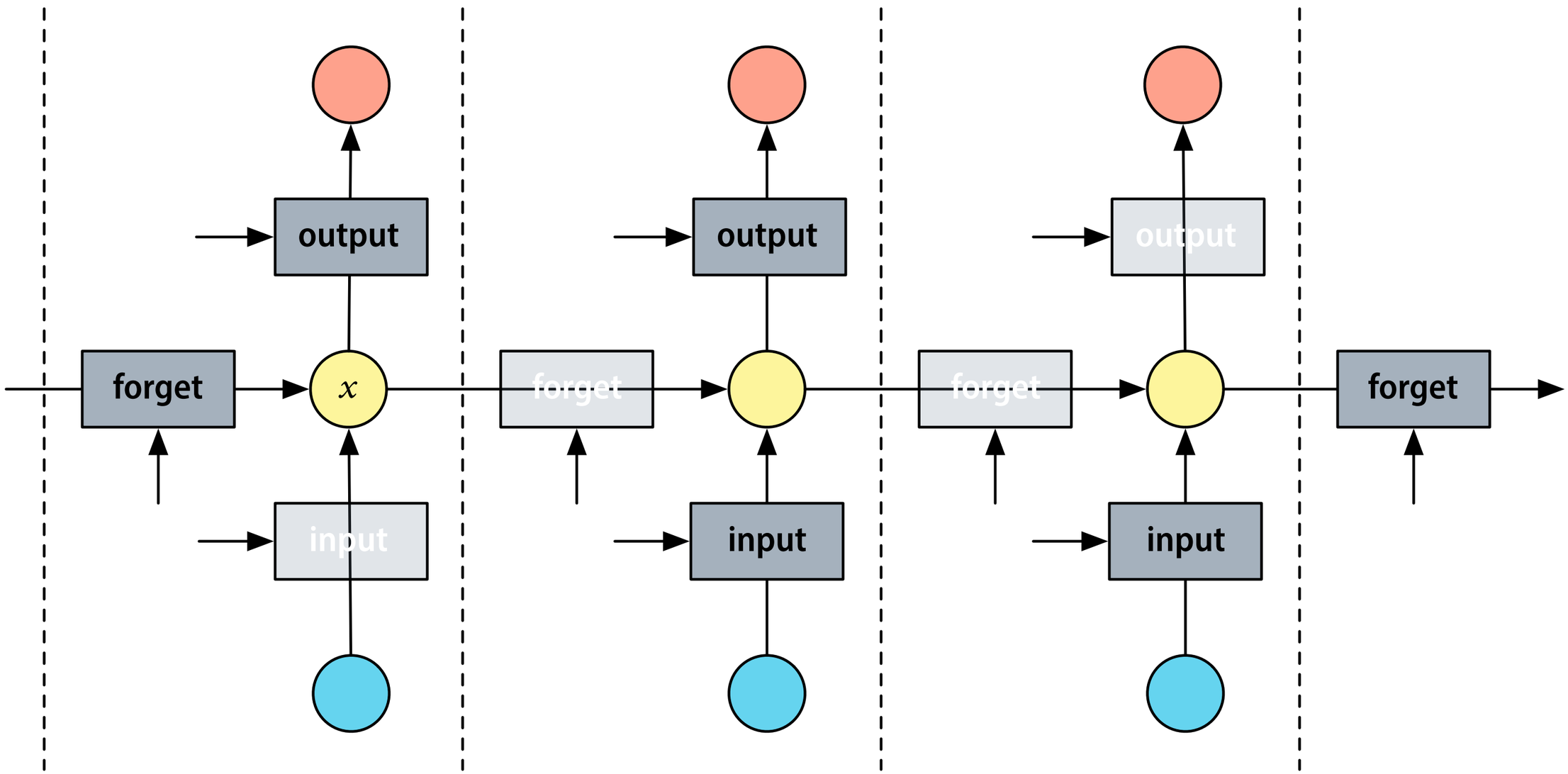
- Information is written into the cell if its INPUT gate is open.
- Information stays in the cell as long as its FORGET gate is closed.
- Information is read from the cell if its READ gate is open.

# Information flow in an LSTM



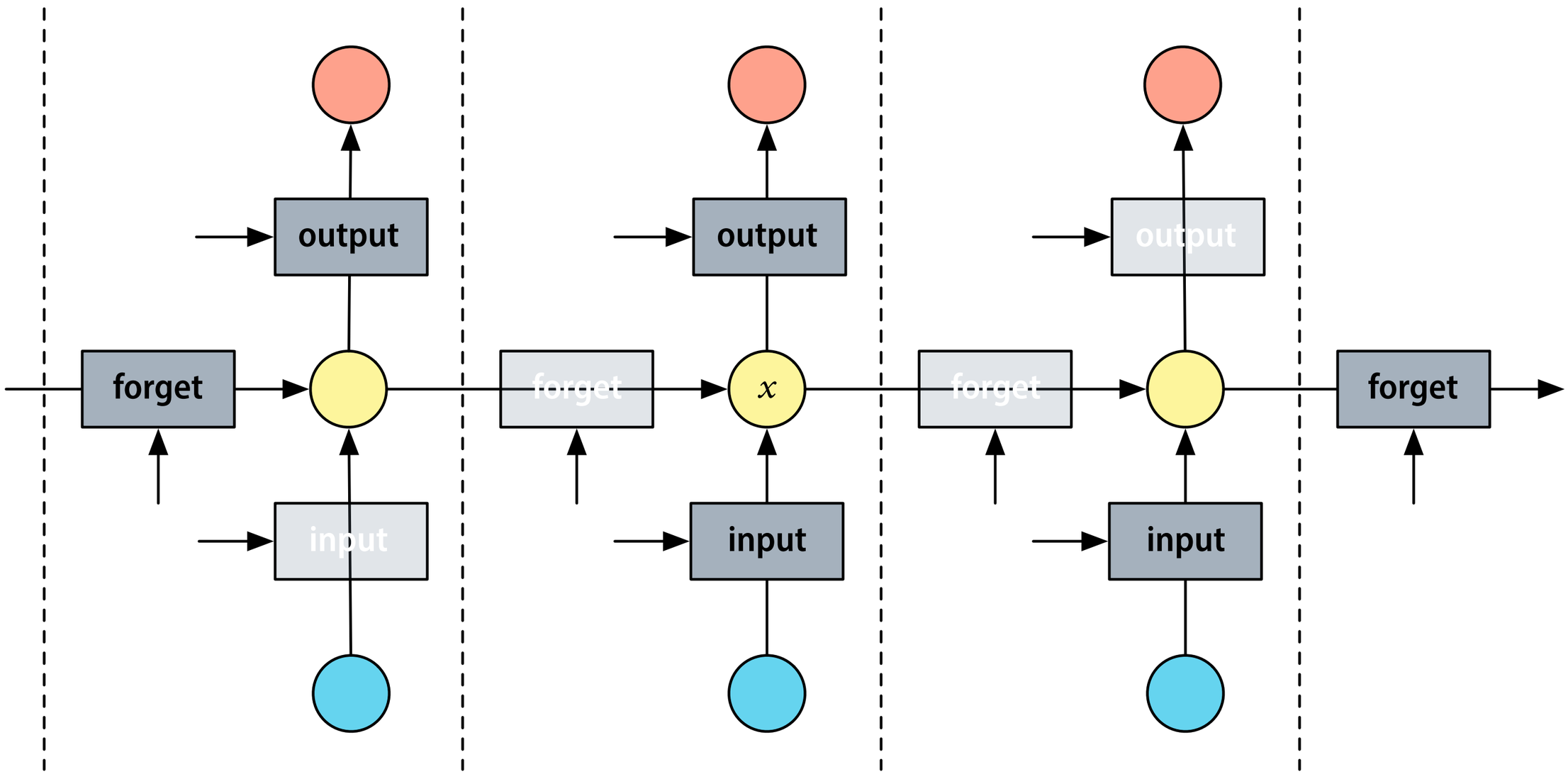

Attribution: Geoffrey Hinton

# Information flow in an LSTM



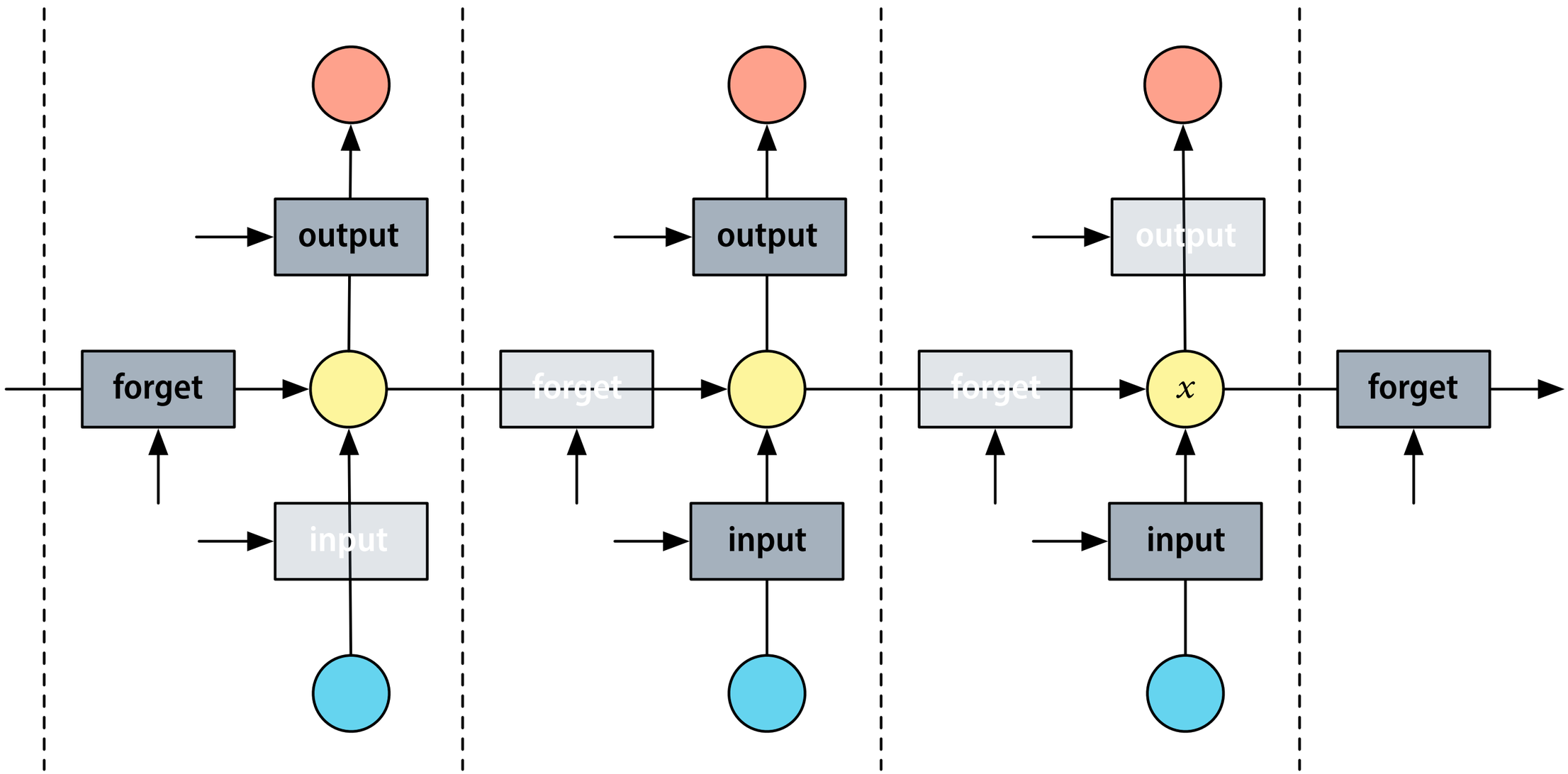

Attribution: Geoffrey Hinton

# Information flow in an LSTM



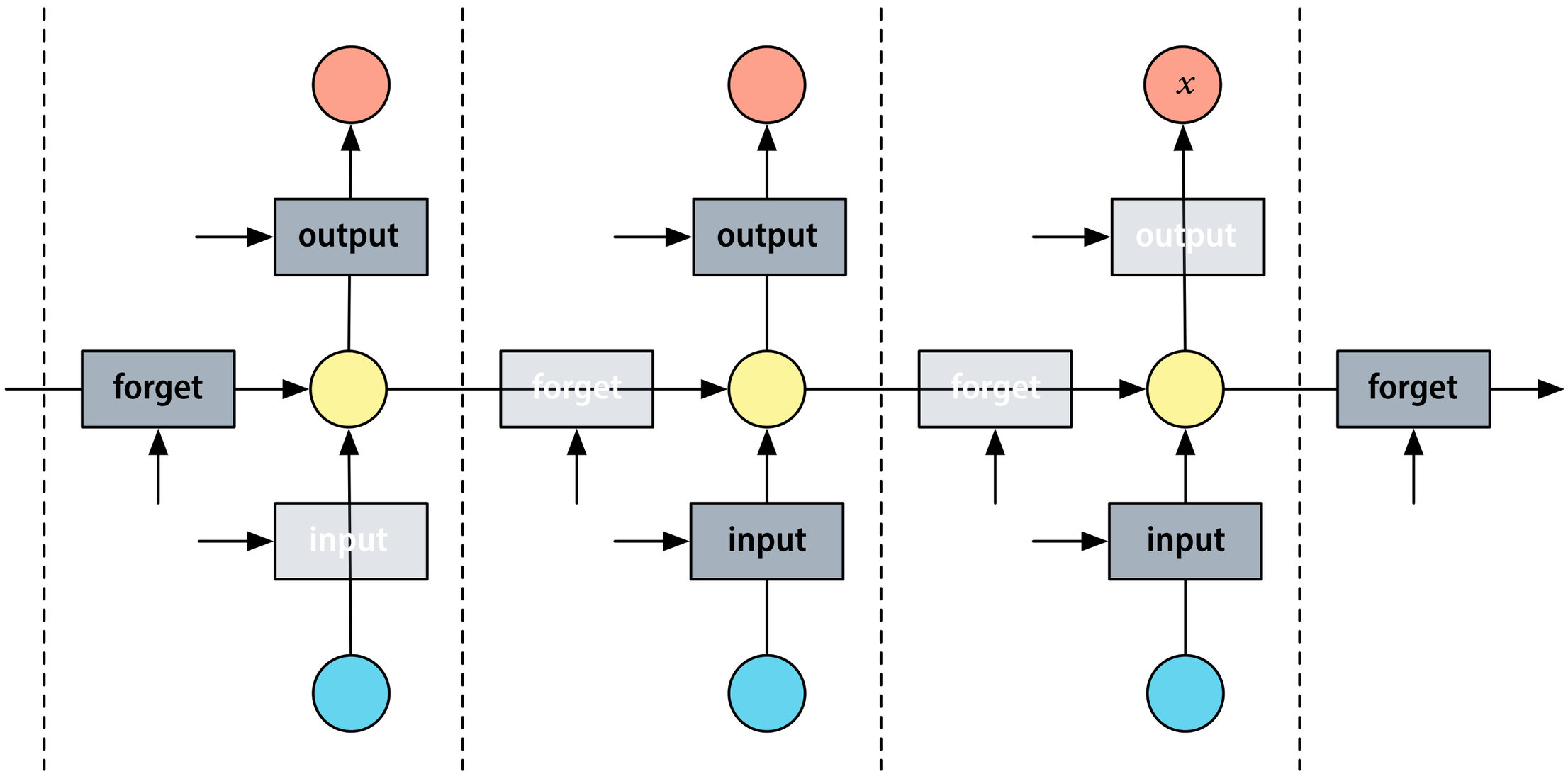

Attribution: Geoffrey Hinton

# Information flow in an LSTM




Attribution: Geoffrey Hinton

# Information flow in an LSTM




Attribution: Geoffrey Hinton

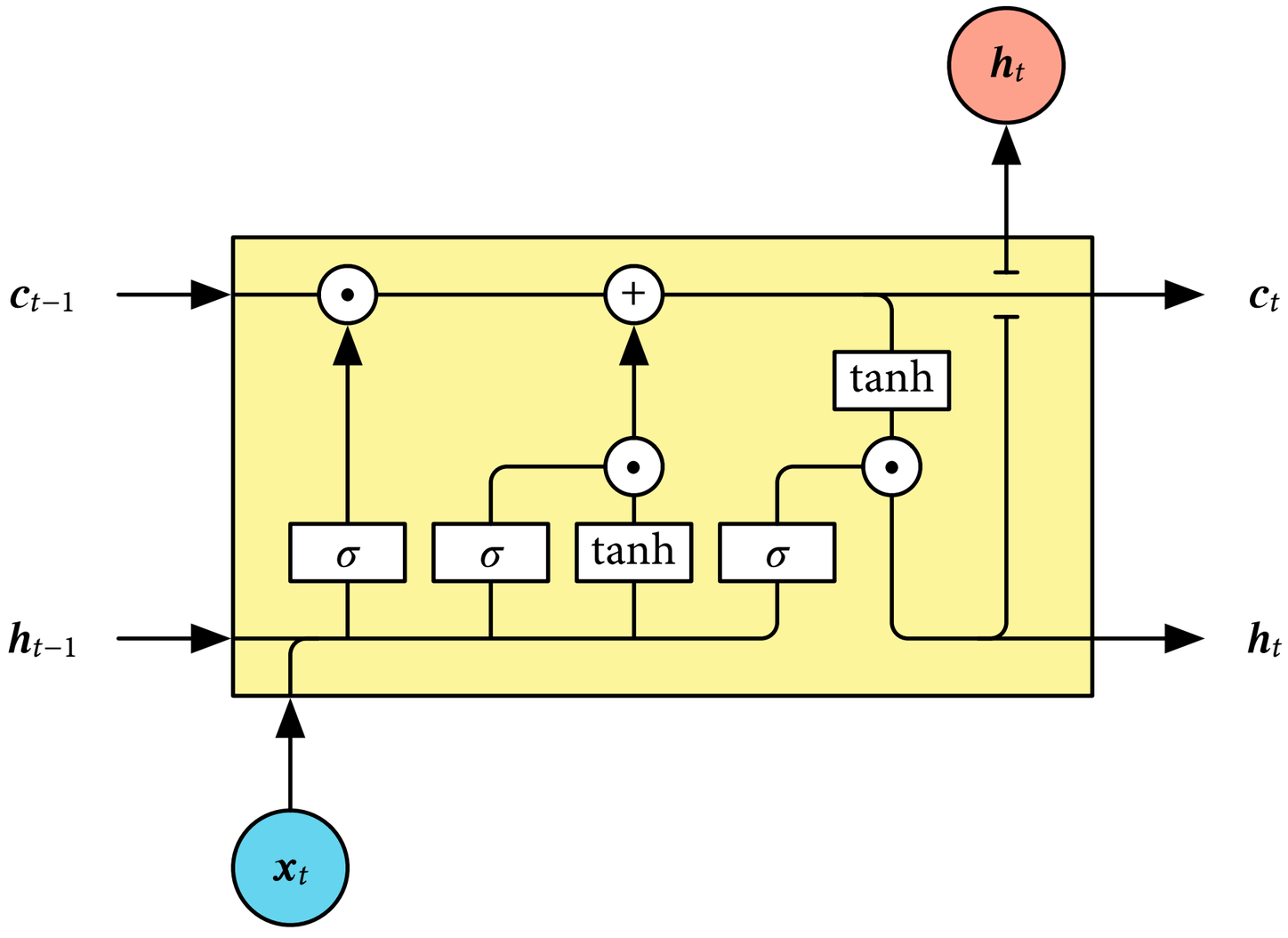
# Gating mechanism

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} \odot \begin{bmatrix}0\\1\\1\\0\end{bmatrix} + \begin{bmatrix}5\\6\\7\\8\end{bmatrix} \odot \begin{bmatrix}1\\0\\0\\1\end{bmatrix} = \begin{bmatrix}5\\2\\3\\8\end{bmatrix}$$

$\boldsymbol{h}_{t-1}$ $\boldsymbol{g}$ $\boldsymbol{x}_t$ $1-\boldsymbol{g}$ $\boldsymbol{h}_t$

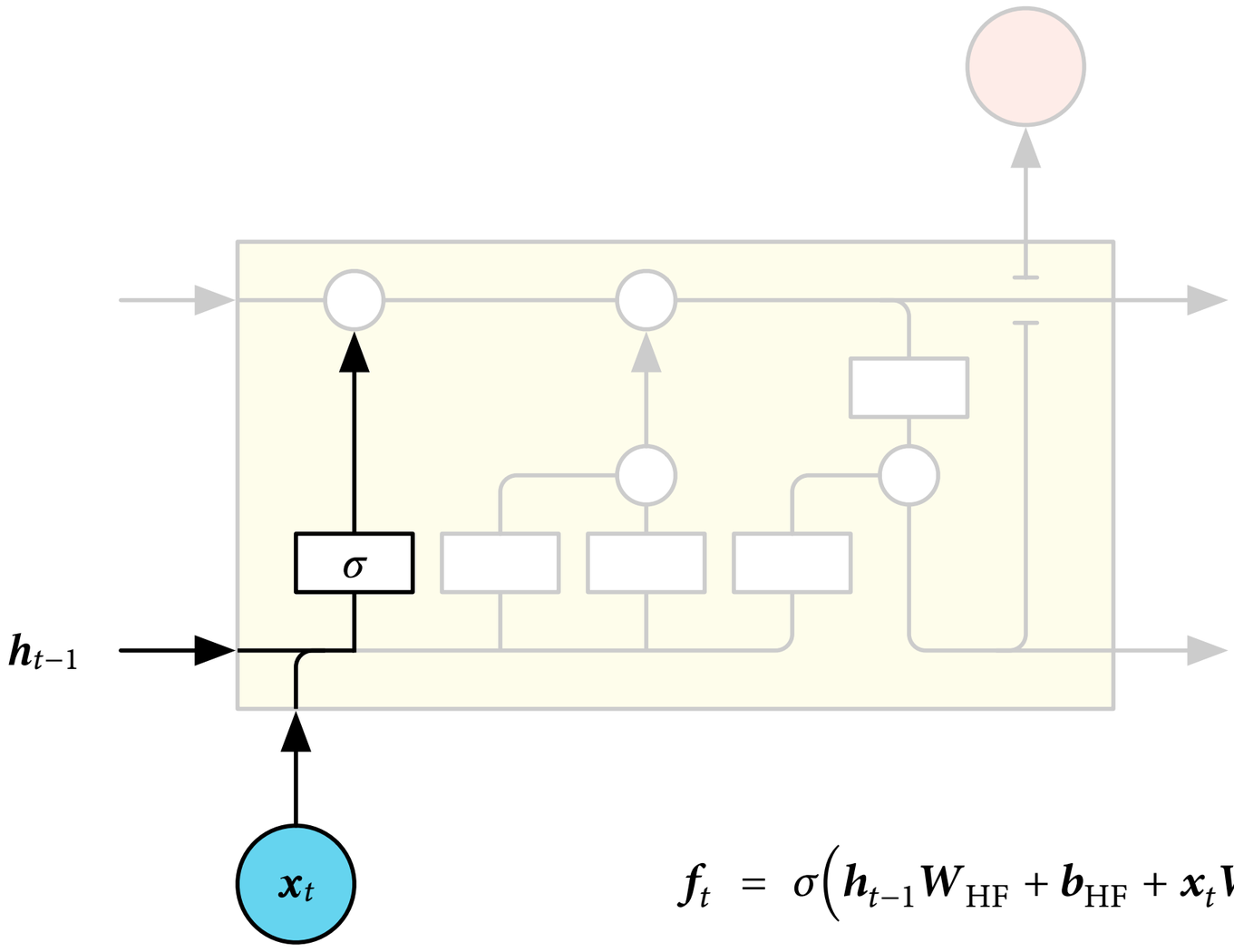
The gating masks $\boldsymbol{g}$ are learned values between 0 and 1.

# A look inside an LSTM cell




Attribution: Chris Olah

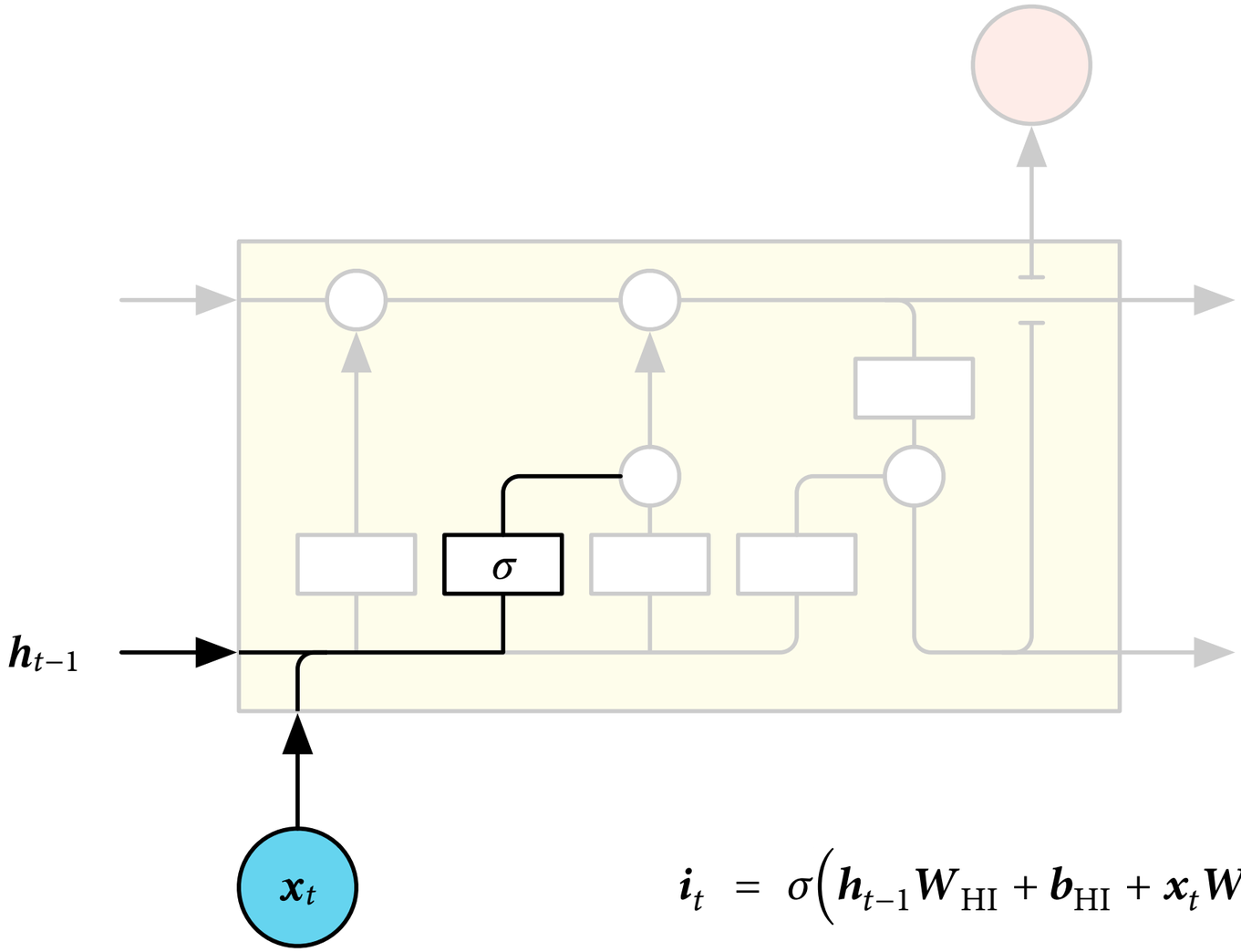
# Forget gate




$$f_t = \sigma\left(h_{t-1}W_{\mathrm{HF}} + b_{\mathrm{HF}} + x_t W_{\mathrm{XF}} + b_{\mathrm{XF}}\right)$$

Attribution: Chris Olah

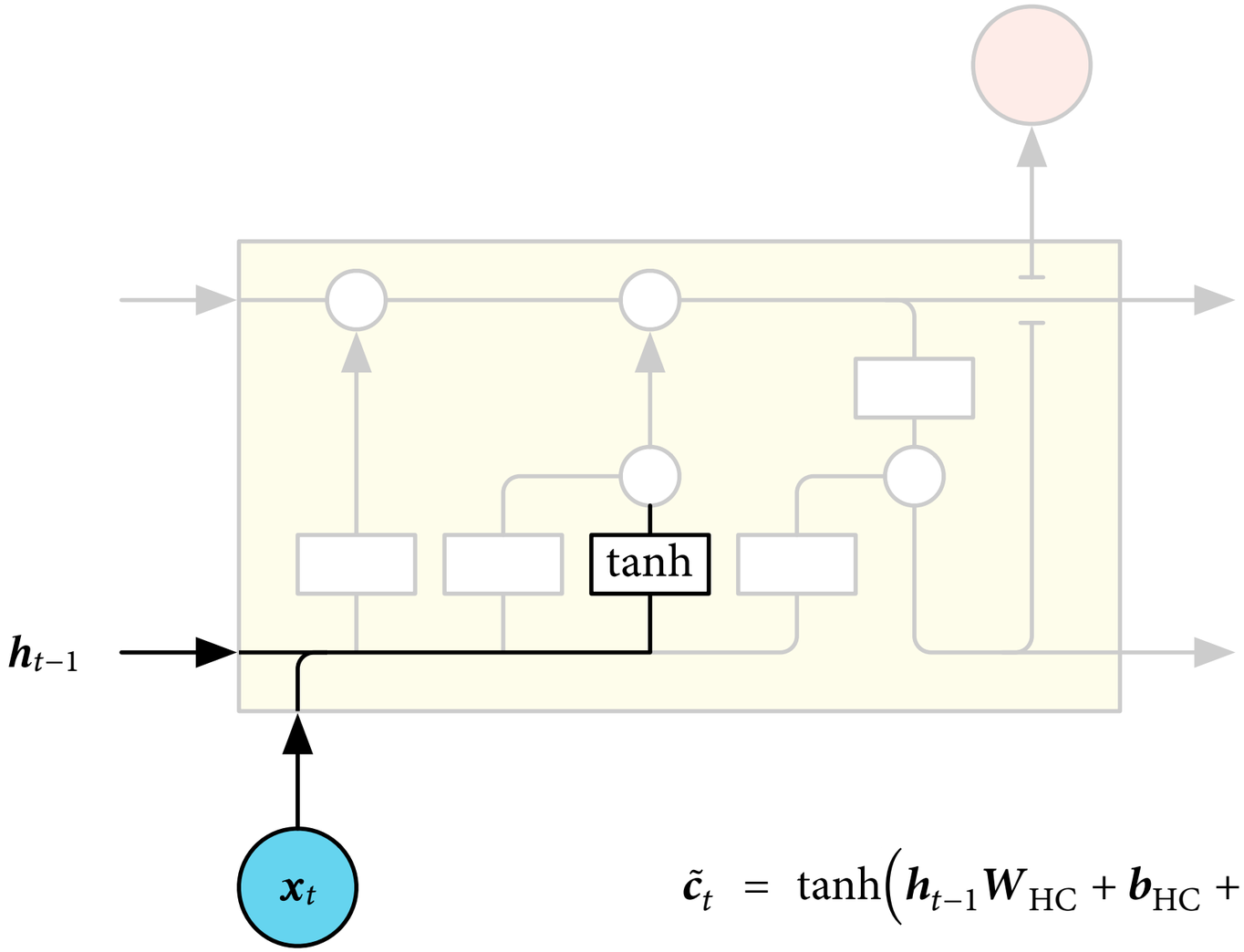
# Input gate




$$i_t = \sigma\left(h_{t-1}W_{\mathrm{HI}} + b_{\mathrm{HI}} + x_t W_{\mathrm{XI}} + b_{\mathrm{XI}}\right)$$

Attribution: Chris Olah

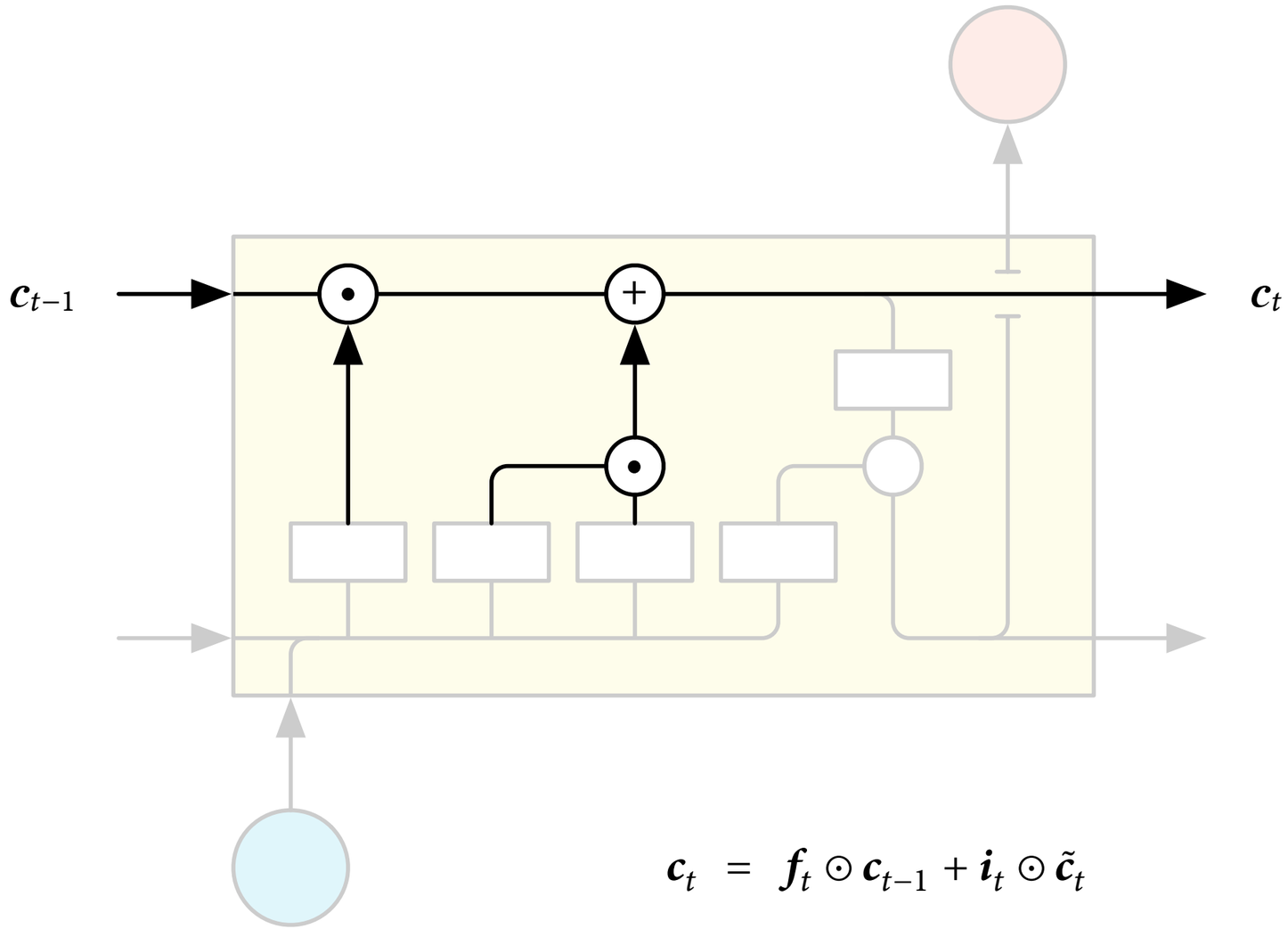
# Update candidate




$$\tilde{c}_t = \tanh\left(h_{t-1}W_{\mathrm{HC}} + b_{\mathrm{HC}} + x_t W_{\mathrm{XC}} + b_{\mathrm{XC}}\right)$$

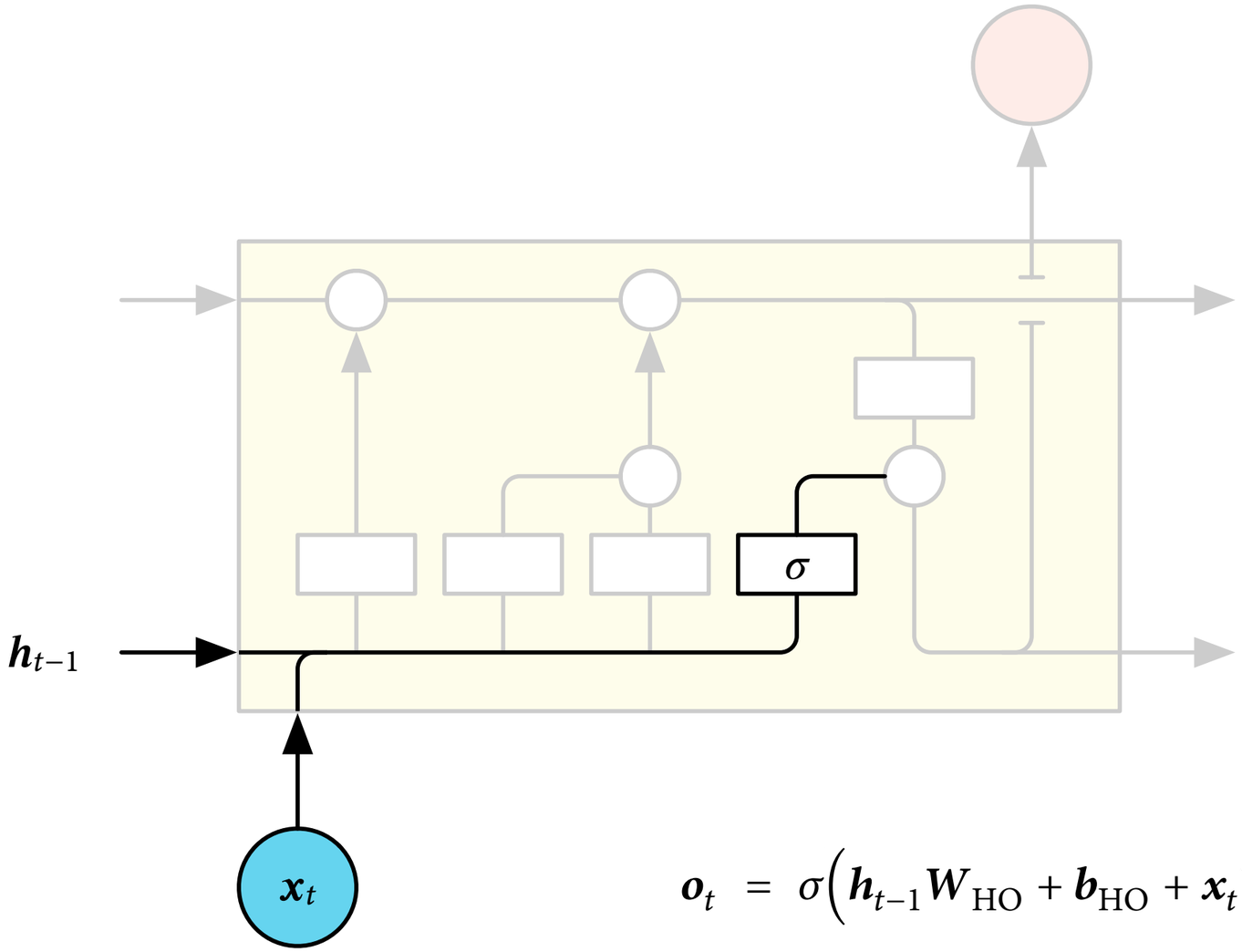
Attribution: Chris Olah

# Memory cell update




Attribution: Chris Olah

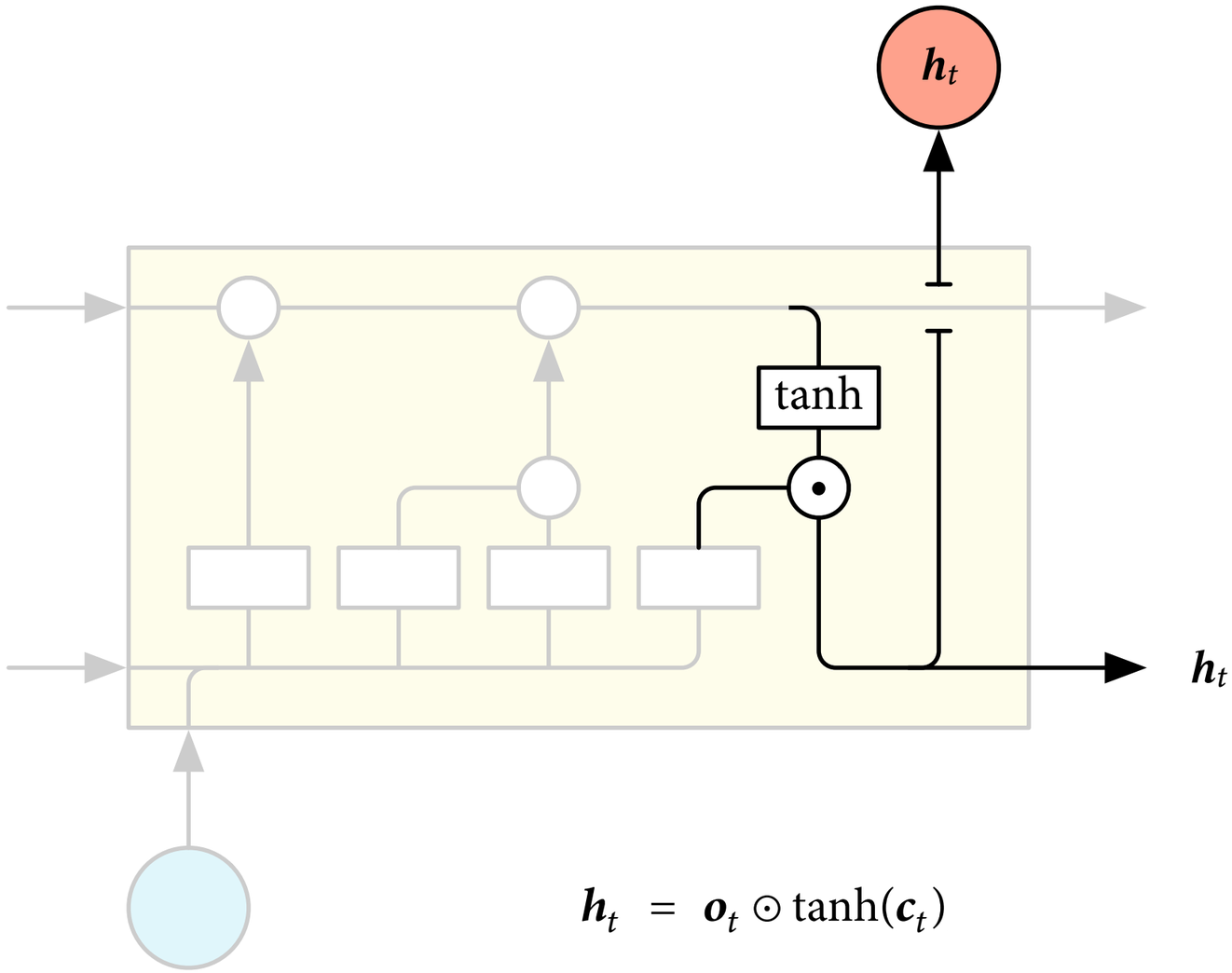
# Output gate




$$o_t = \sigma\left(h_{t-1}W_{\mathrm{HO}} + b_{\mathrm{HO}} + x_t W_{\mathrm{XO}} + b_{\mathrm{XO}}\right)$$

Attribution: Chris Olah

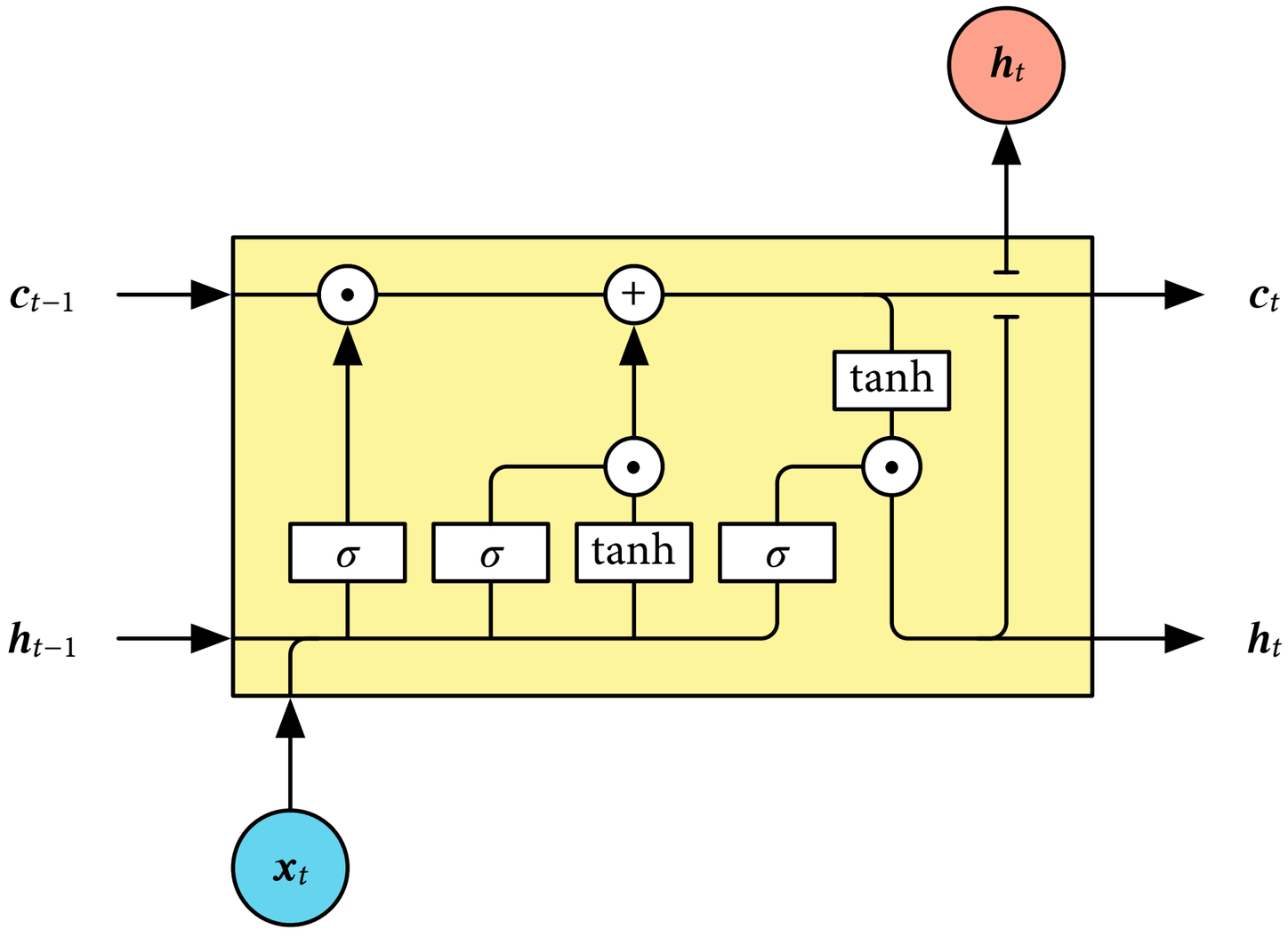
# Output




$$h_t = o_t \odot \tanh(c_t)$$

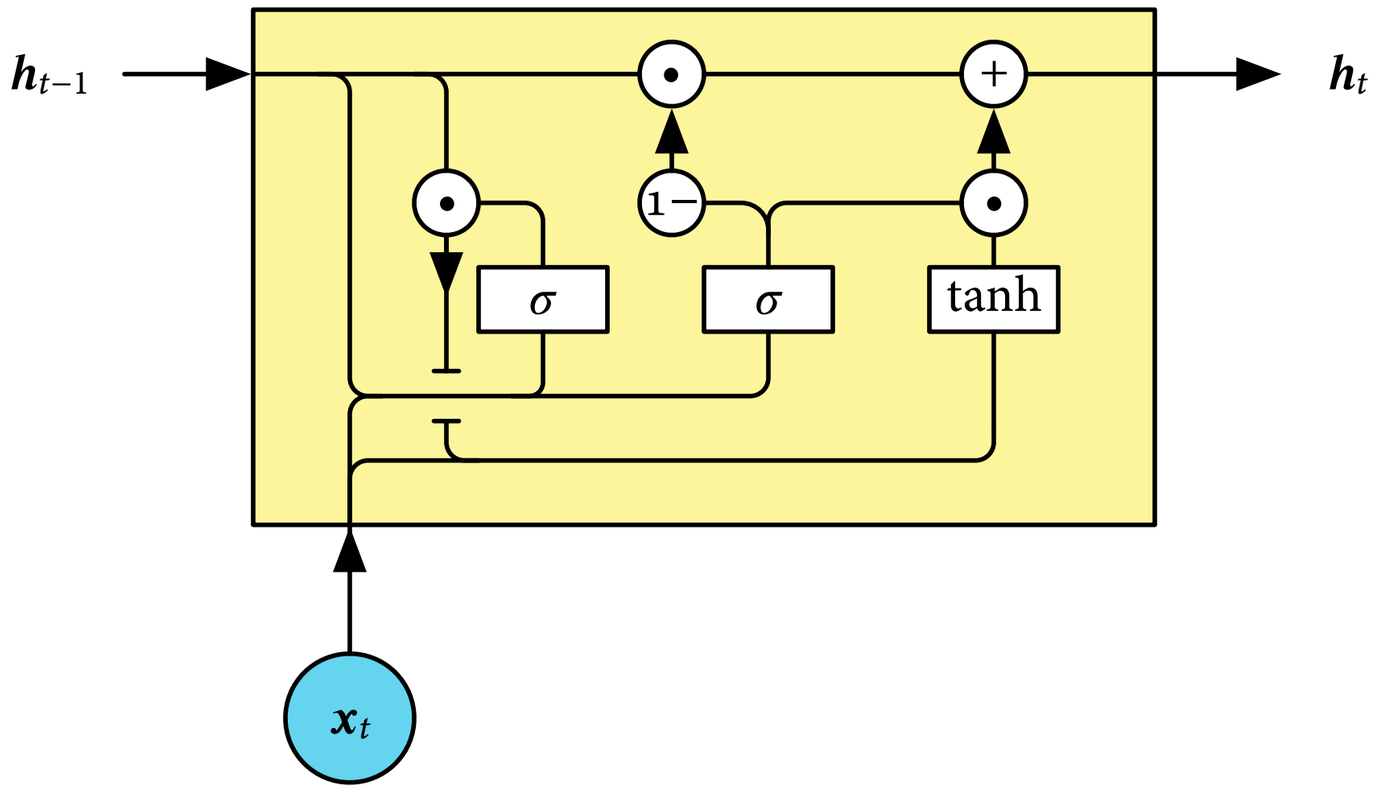
Attribution: Chris Olah

# A look inside an LSTM cell




Attribution: Chris Olah

# Gated Recurrent Unit (GRU)



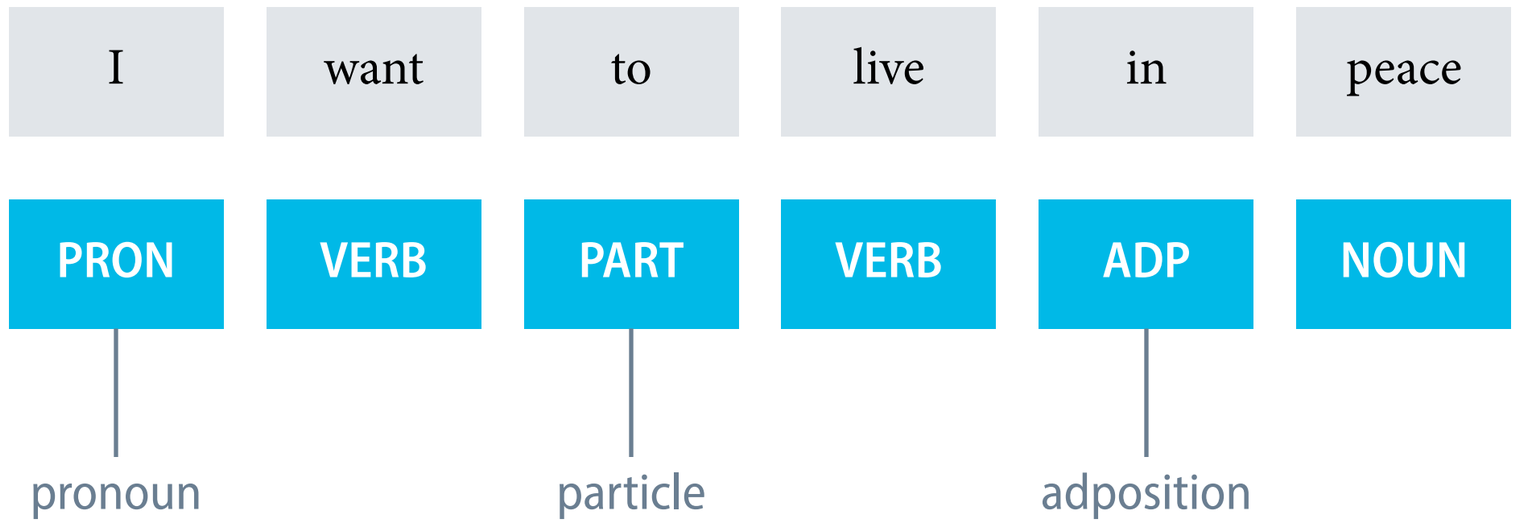

Attribution: Chris Olah

# Use case 1:  Part-of-speech tagging

# Parts of speech

- A **part of speech** is a category of words that play similar roles within the syntactic structure of a sentence.
- Commonly listed English parts of speech include noun, verb, adjective, adverb, pronoun, preposition, and conjunction.
- Determining the parts of speech of a sentence is one of the first steps in the traditional NLP analysis pipeline.

# Part-of-speech tagging, example




‘I only want to live in peace, plant potatoes, and dream!’ – Moomin

# Universal part-of-speech tags

| Tag | Category | Examples |
|---|---|---|
| ADJ | adjective | *big, old* |
| ADV | adverb | *very, well* |
| INTJ | interjection | *ouch!* |
| NOUN | noun | *girl, cat, tree* |
| VERB | verb | *run, eat* |
| PROPN | proper noun | *Mary, John* |

| Tag | Category | Examples |
|---|---|---|
| ADP | adposition | *in, to, during* |
| AUX | auxiliary verb | *has, was* |
| CCONJ | conjunction | *and, or, but* |
| DET | determiner | *a, my, this* |
| NUM | cardinal numbers | *o, one* |
| PRON | pronoun | *I, myself, this* |

Missing: PART, SCONJ, PUNCT, SYM, X

Source: Universal Dependencies Project

# Part-of-speech tagging

- A **part-of-speech tagger** is a computer program that tags each word in a sentence with its part of speech.

- Part-of-speech tagging can be approached as a supervised machine learning problem. This requires training data.

  linguistic data sets with gold-standard part-of-speech annotation

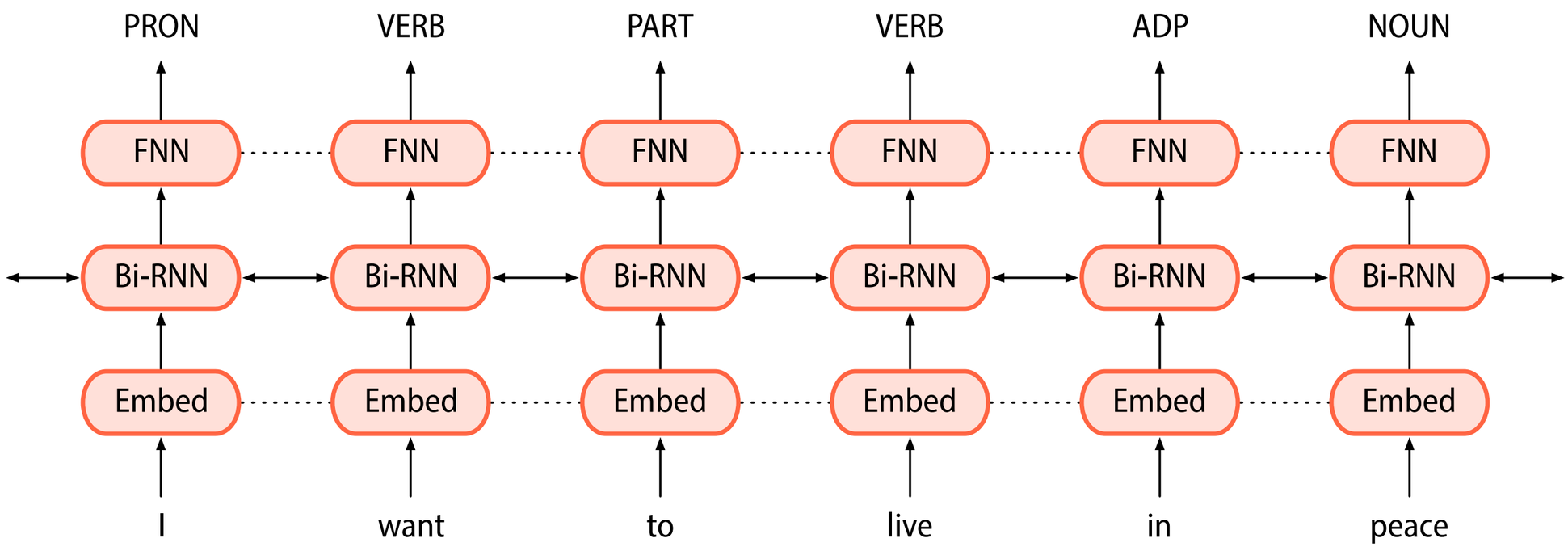
- Part-of-speech taggers are commonly evaluated using accuracy, precision, and recall.

# Ambiguity

| I | want | to | live | in | peace |
|---|---|---|---|---|---|
| PRON | VERB | PART | VERB | ADP | NOUN |
| NOUN | NOUN | ADP | ADJ | ADV | VERB |
| | | ADV | ADV | ADJ | |
| | | | | NOUN | |

‘I only want to live in peace, plant potatoes, and dream!’ – Moomin

# Bidirectional RNN model




loss = sum or mean of the token-specific losses

# Beyond part-of-speech tagging

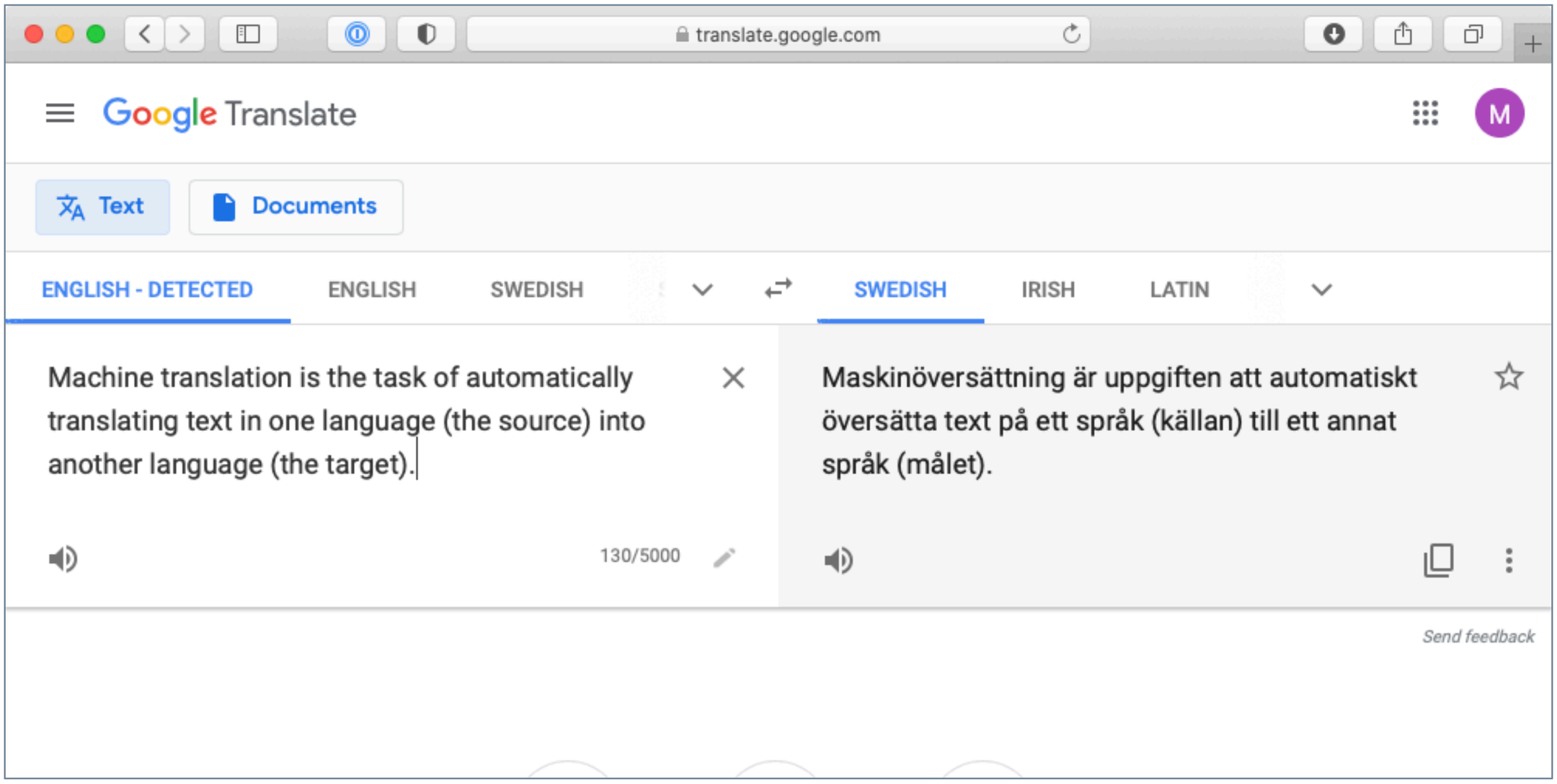
- Many other NLP tasks can be framed as **sequence labelling**, e.g. named entity recognition, event extraction.

  "span-level" annotations can be transformed into BIO notation

| | ORG | | | | LOC | | | |
|---|---|---|---|---|---|---|---|---|
| *Taco Bell* | *was* | *founded* | *in* | *Irvine* | *,* | *California* | *.* | |
| B-ORG | I-ORG | O | O | O | B-LOC | I-LOC | I-LOC | O |

# Use case 2: Machine translation

# Machine translation



# A timeline of machine translation

rule-based machine translation

statistical machine translation

1950 1980 1990 2015 2020

example-based machine translation

neural machine translation

# Neural Machine Translation (NMT)

- **Neural machine translation (NMT)** models the translation task through a single artificial neural network.

- The first systems for NMT were based on recurrent neural networks; more recent systems typically use Transformers.

- Many practical implementations are based on the OpenNMT ecosystem for neural machine translation.

  Link to OpenNMT

# The sequence-to-sequence model (seq2seq)

The sequence-to-sequence model consists of two components:

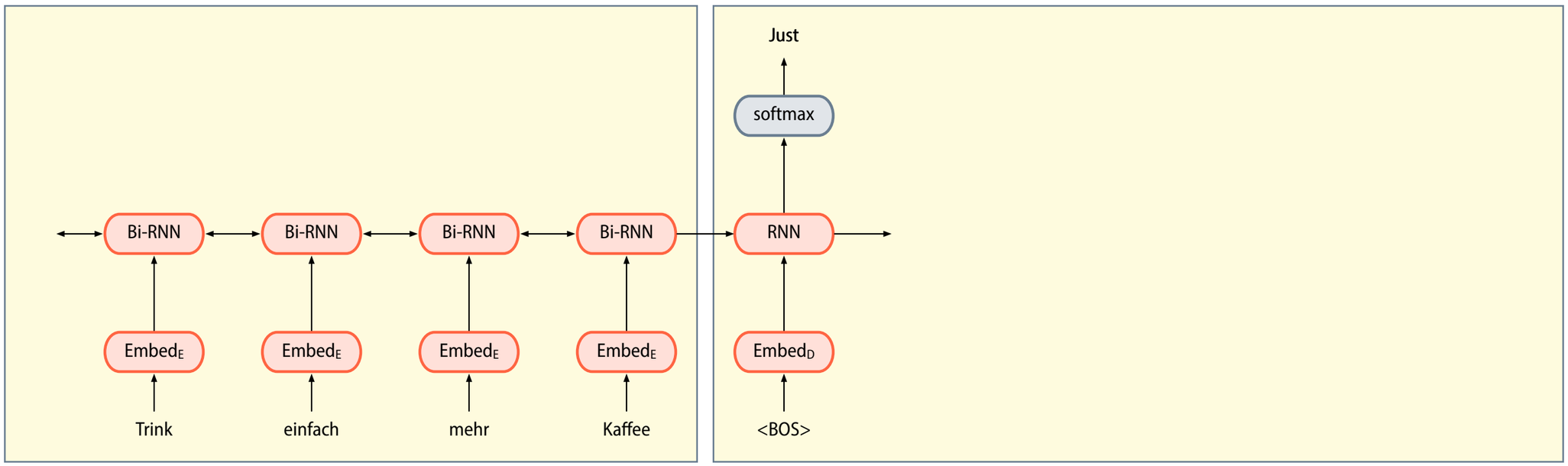
- The **encoder** is a neural network that produces a representation of the source sentence.

  typically implemented as a bidirectional recurrent neural network

- The **decoder** is an autoregressive language model that generates the target sentence, conditioned on the output of the encoder.

  autoregressive = takes its own outputs as new inputs

# Standard seq2seq architecture

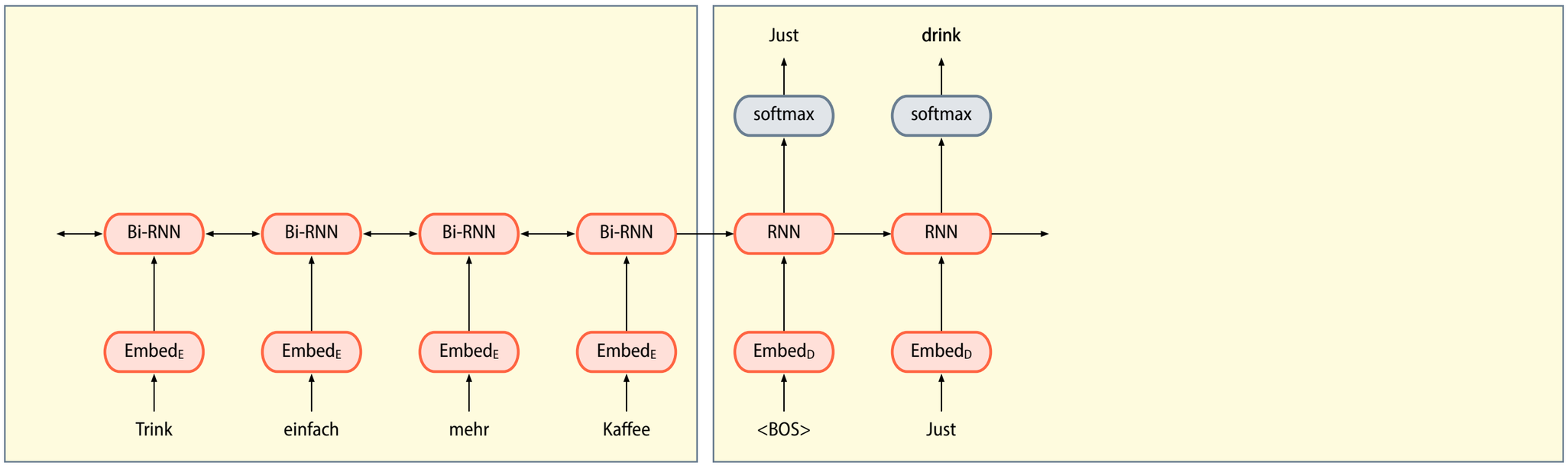



encoder

decoder

Sutskever et al. (2014)

# Standard seq2seq architecture

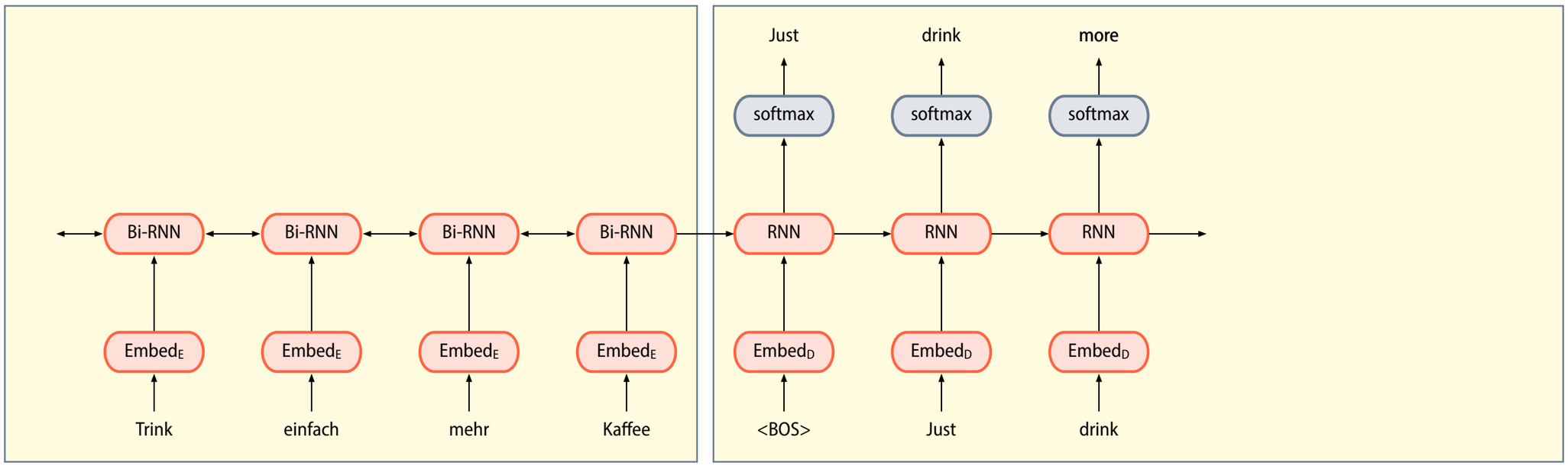



encoder

decoder

[Sutskever et al. (2014)]

# Standard seq2seq architecture

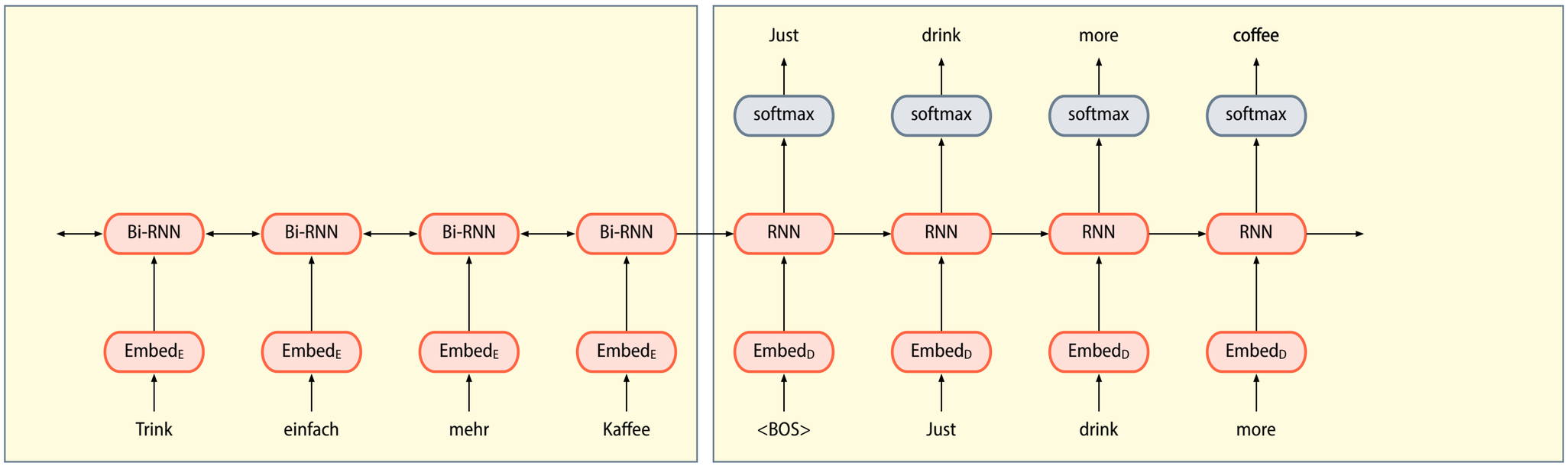



encoder

decoder

Sutskever et al. (2014)

# Standard seq2seq architecture

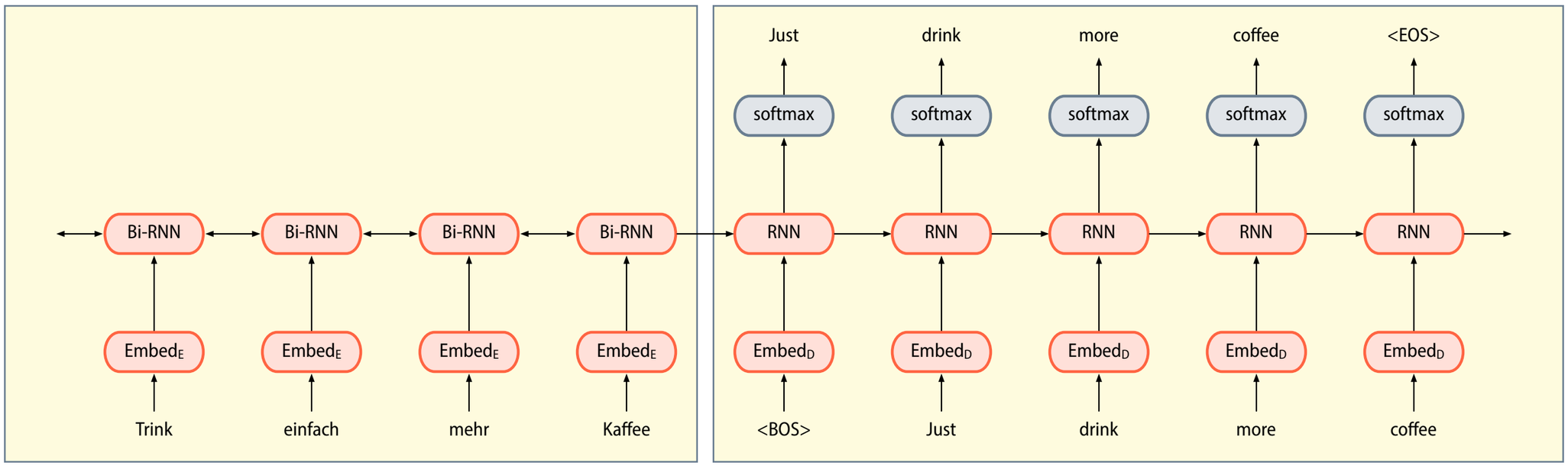



encoder

decoder

[Sutskever et al. (2014)]

# Standard seq2seq architecture




encoder

decoder

[Sutskever et al. (2014)]

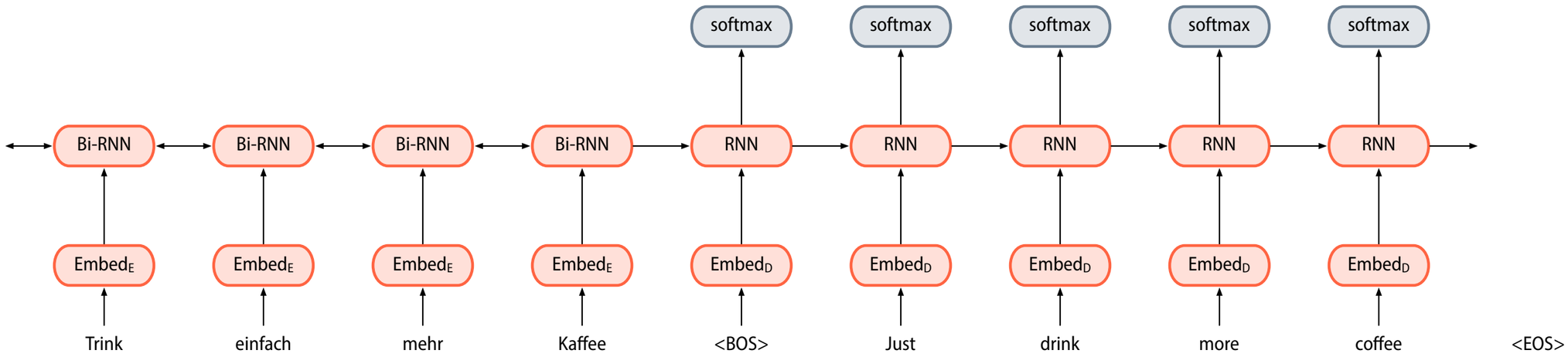
# Properties of the seq2seq model

- The seq2seq model directly learns and uses $P(\boldsymbol{y}|\boldsymbol{x})$, rather than decomposing it into $P(\boldsymbol{x}|\boldsymbol{y})$ and $P(\boldsymbol{y})$ as in SMT.
- The model can be trained trained end-to-end using backpropagation, without alignments or auxiliary models.

  only needs parallel data

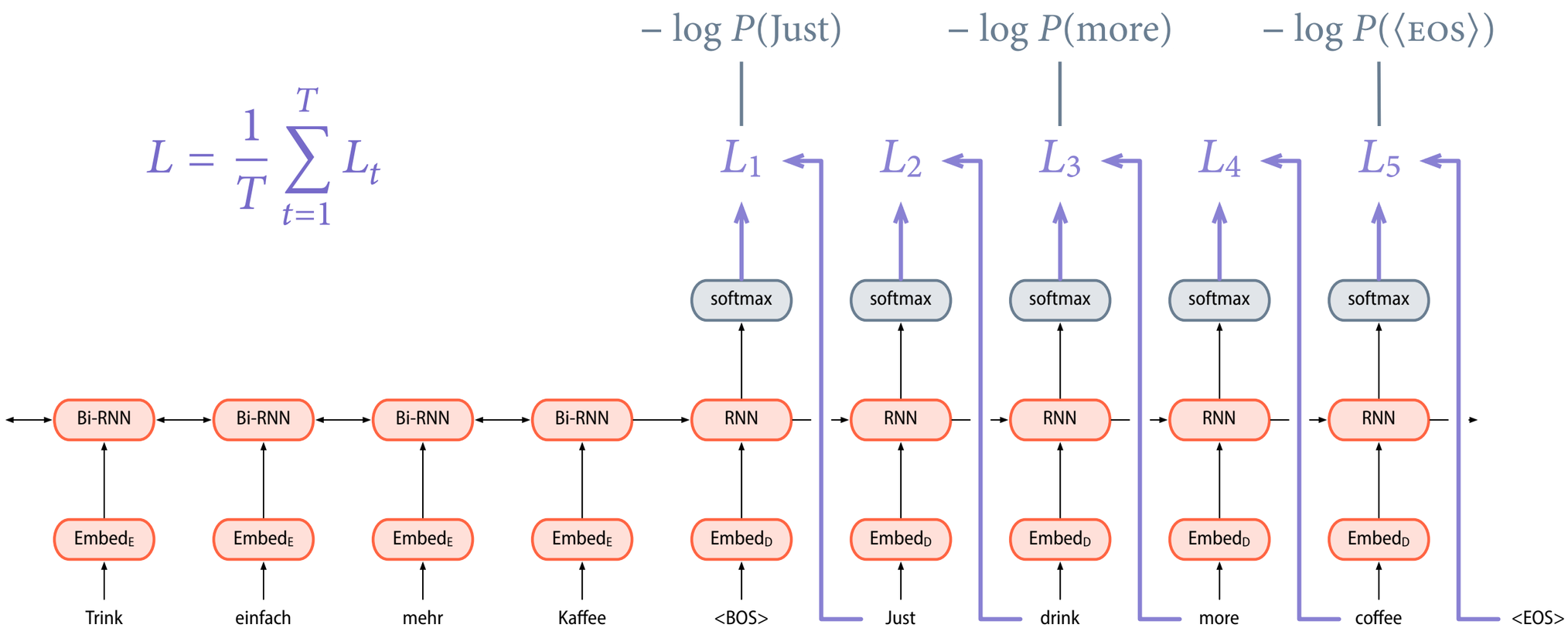
- The seq2seq model is useful for a range of other tasks, including text summarisation, dialogue, and code generation.

# Training an encoder–decoder model



# Training an encoder–decoder model

$$L = \frac{1}{T}\sum_{t=1}^{T} L_t$$



# Decoding algorithms

- **Greedy decoding**

  At each step, predict the highest-probability word. Stop when the end-of-sentence marker is predicted.

- **Beam search**

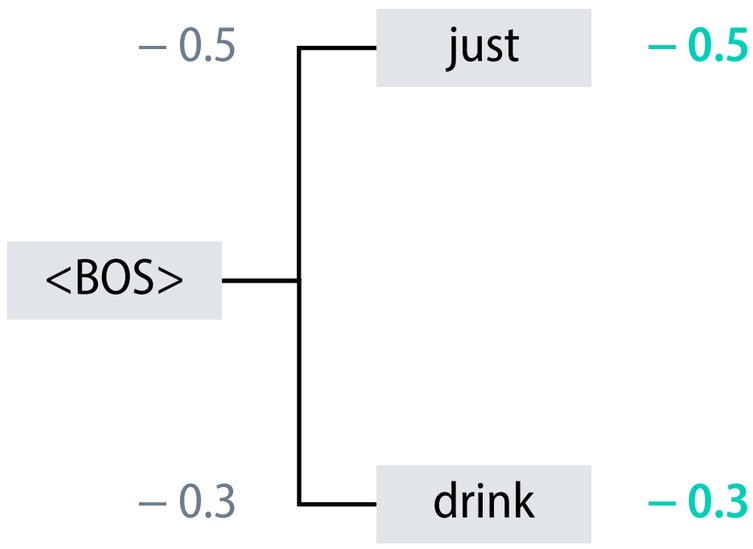
  Keep a limited number of highest-scoring partial translations. Expand the items on the beam, score the new items, and prune.

  Typical beam widths are between 2 and 16.

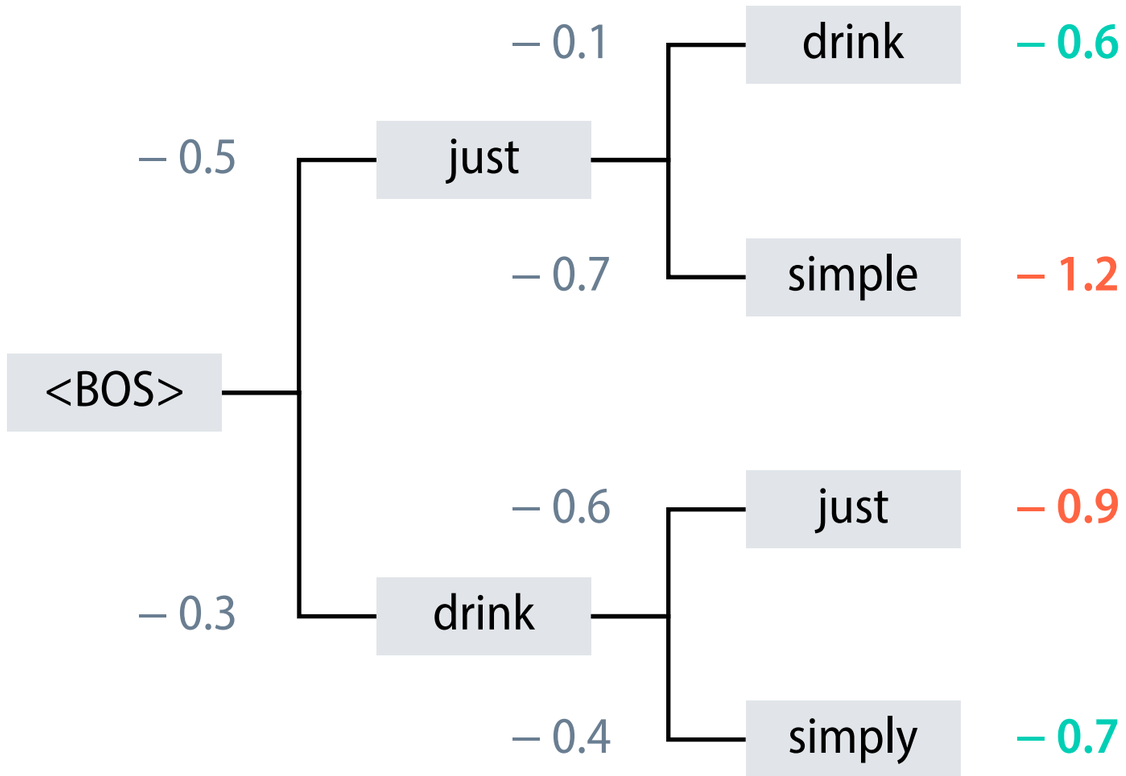
# Beam search example

<BOS>

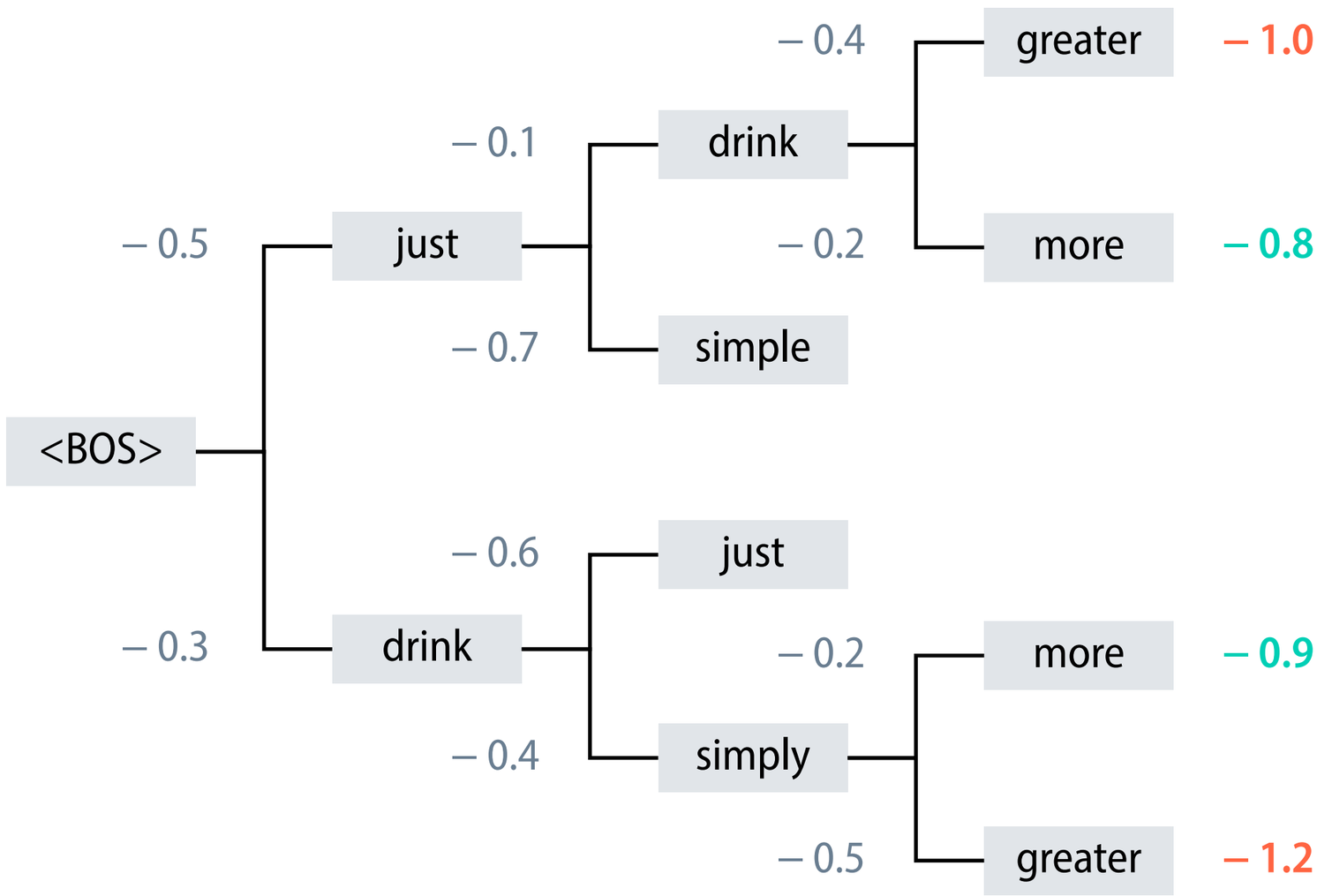
# Beam search example



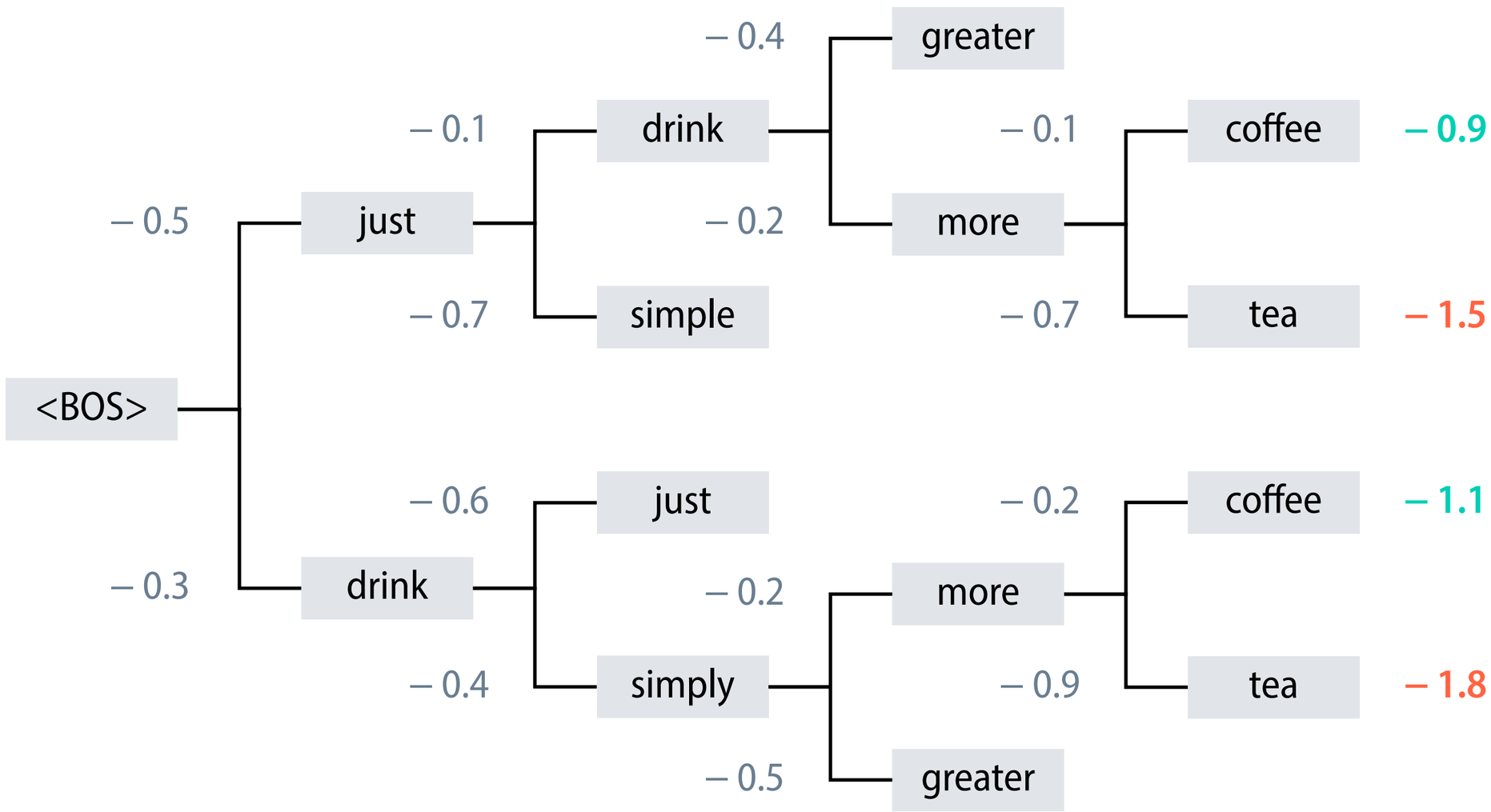
# Beam search example



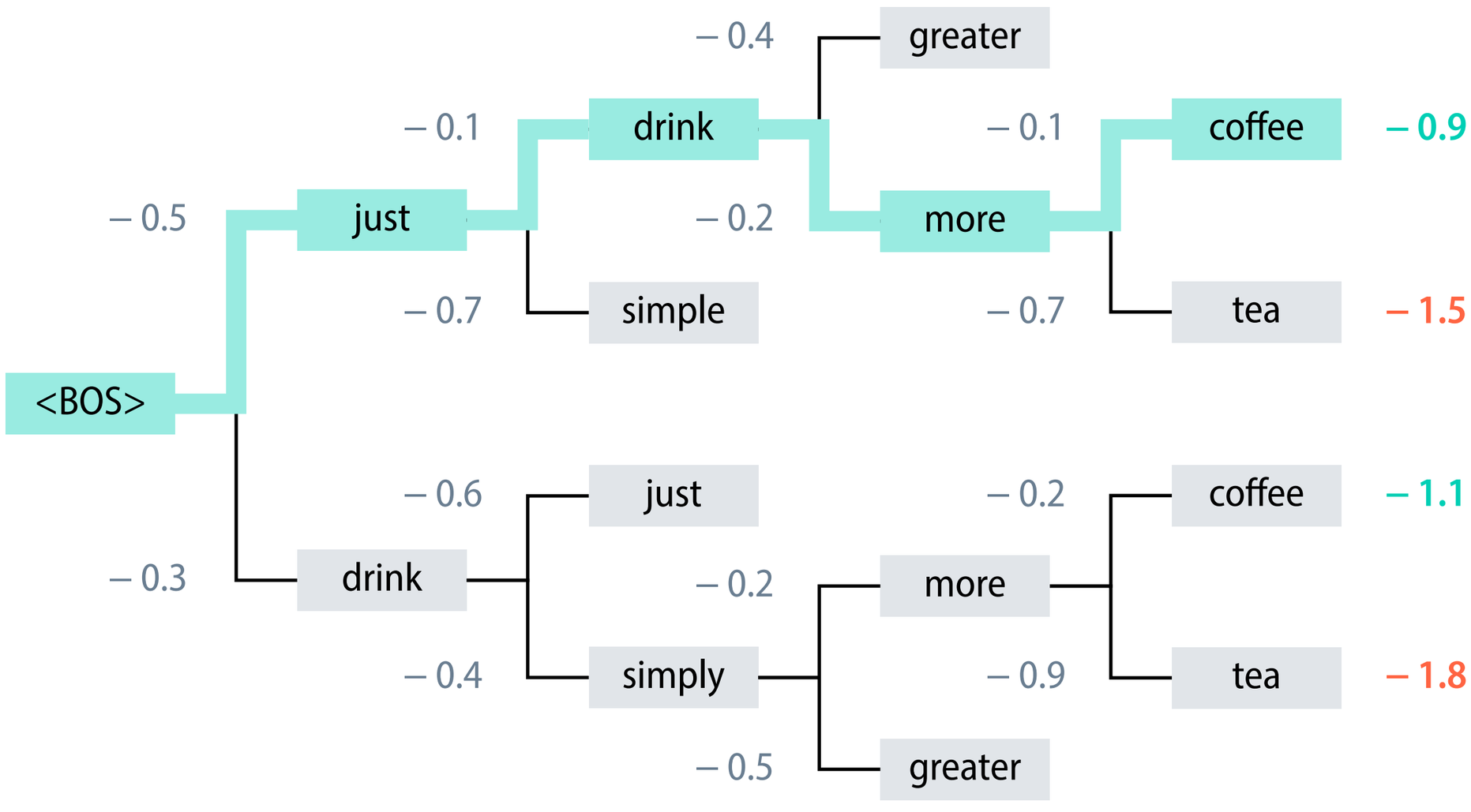
# Beam search example



# Beam search example

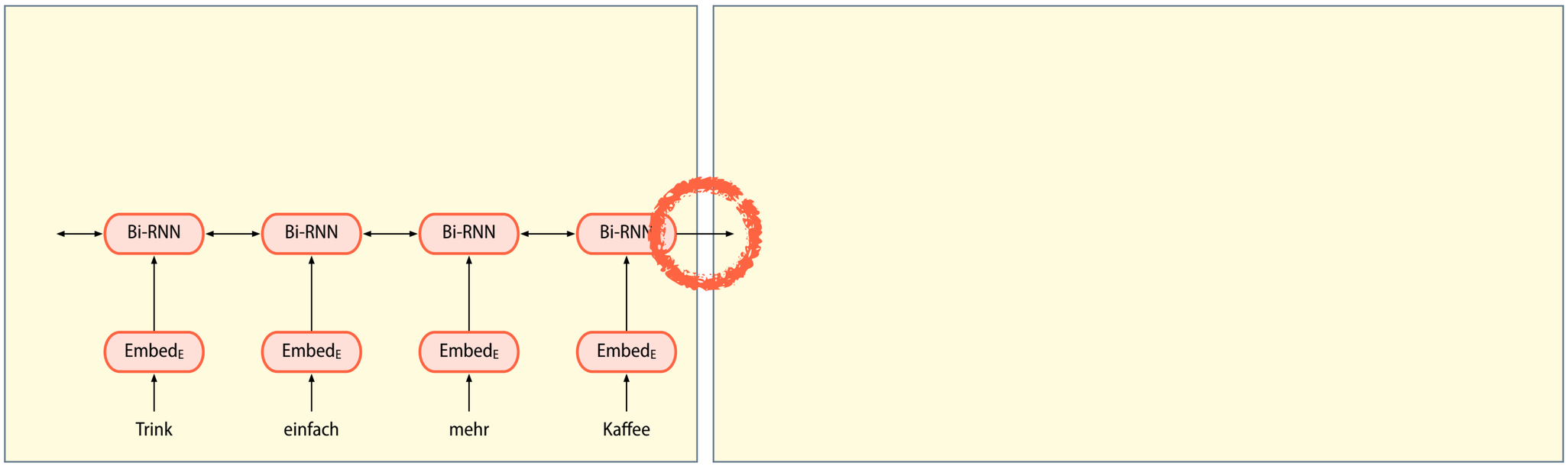


# Beam search example



# Attention

# Recency bias in recurrent neural networks

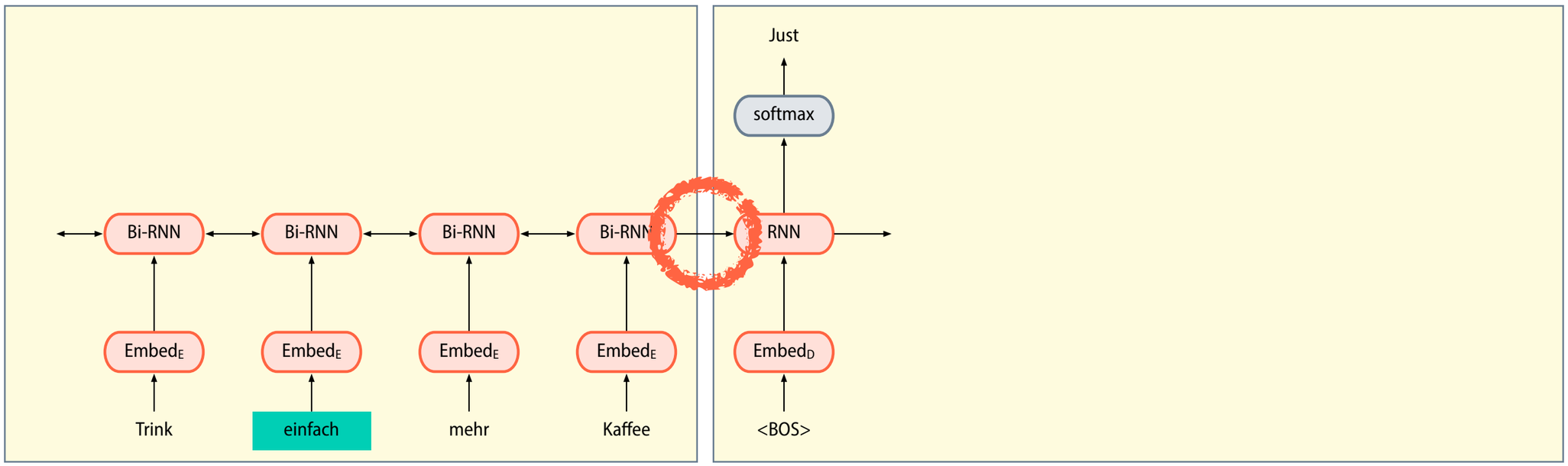



encoder

decoder

Sutskever et al. (2014)

# Recency bias in recurrent neural networks

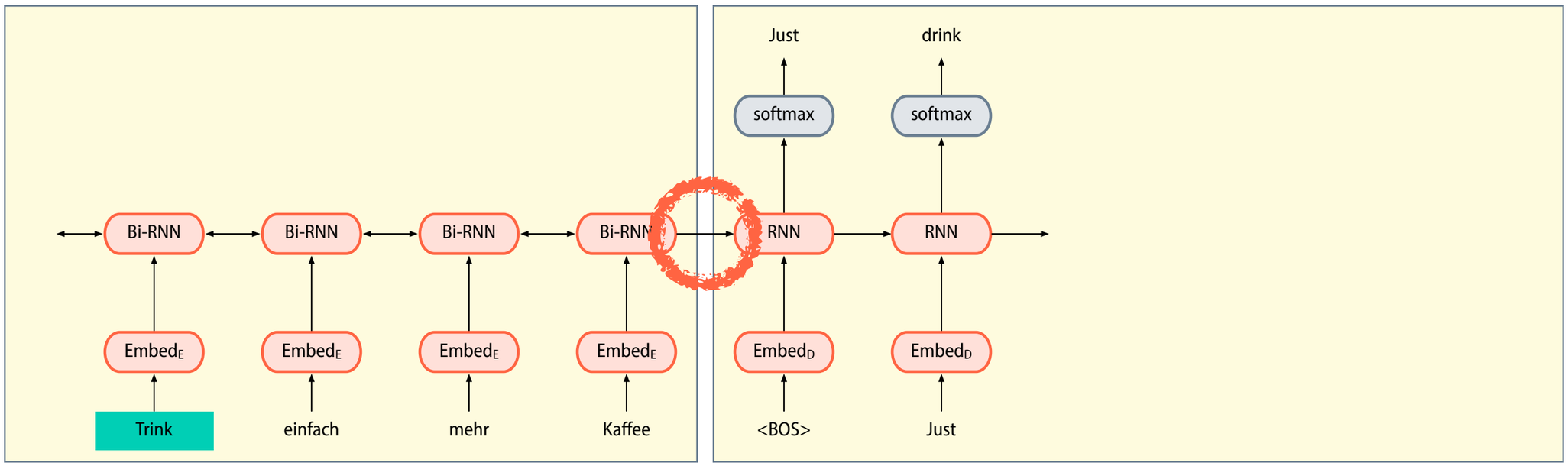



encoder

decoder

Sutskever et al. (2014)

# Recency bias in recurrent neural networks




encoder

decoder

Sutskever et al. (2014)

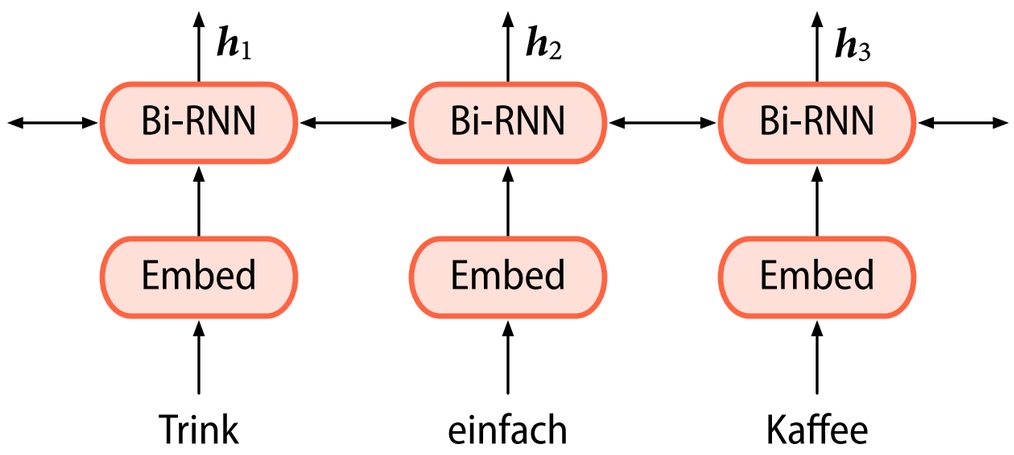
# Attention

- In the context of machine translation, **attention** enables the model to learn 'soft' word alignments.
- Essentially, we compute a set of weights that allow us to score words based on how much the model should 'attend to them'.
- Attention was first proposed in the context of the sequence-to-sequence architecture, but is now used in many architectures.

  Bahdanau et al. (2015)

# Attention for translation

Just drink coffee

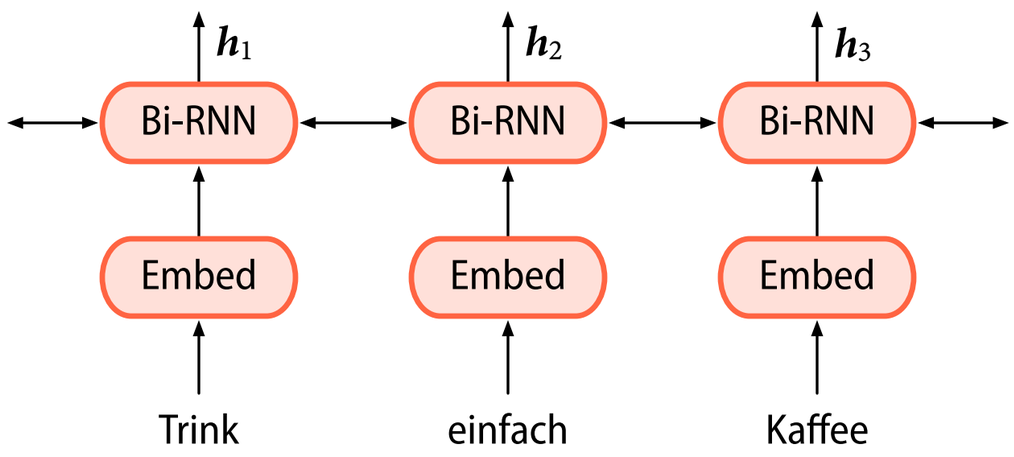



Bahdanau et al. (2015)

# Attention for translation

Just drink coffee

previous hidden state of the decoder — $s$

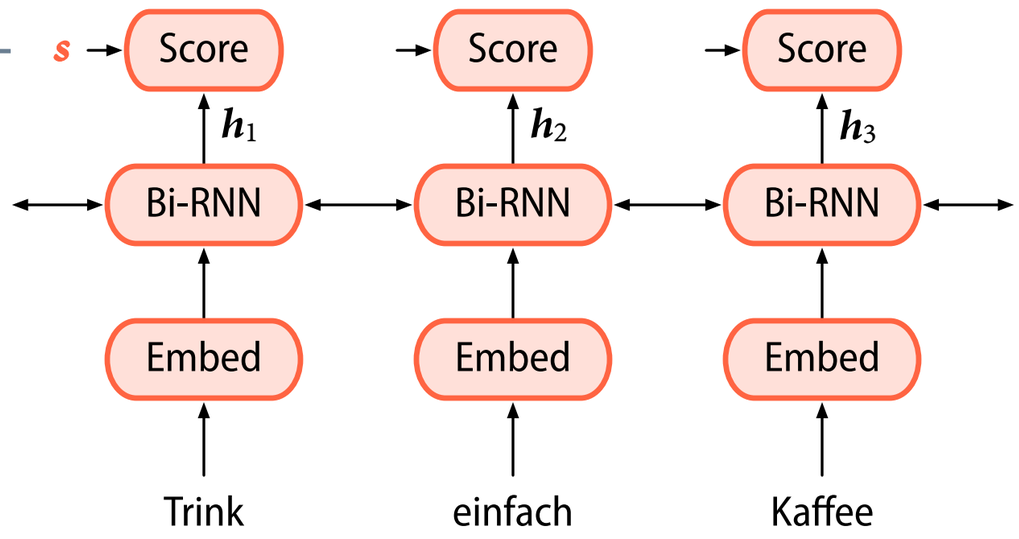



Bahdanau et al. (2015)

# Attention for translation

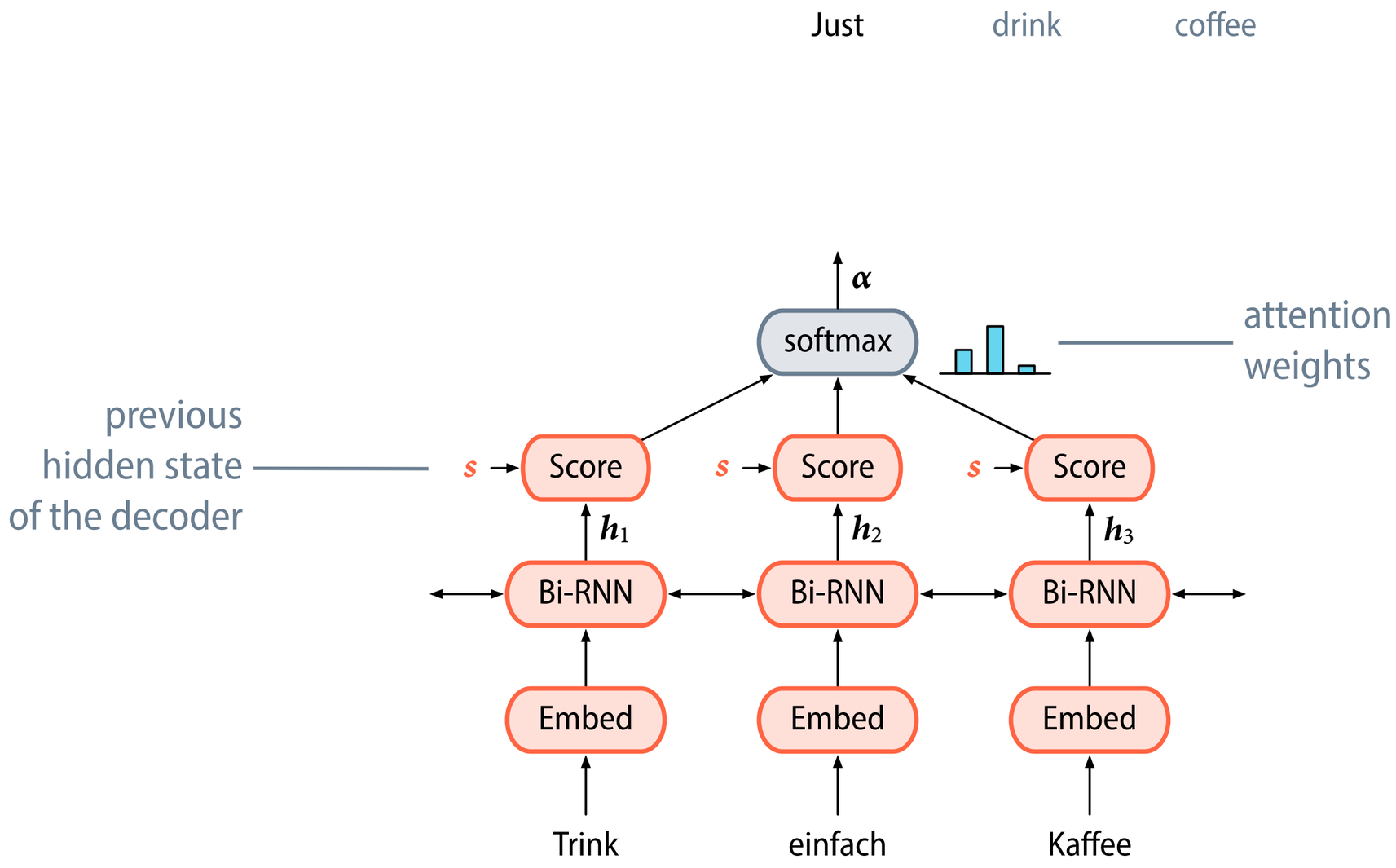
Just drink coffee

previous hidden state of the decoder



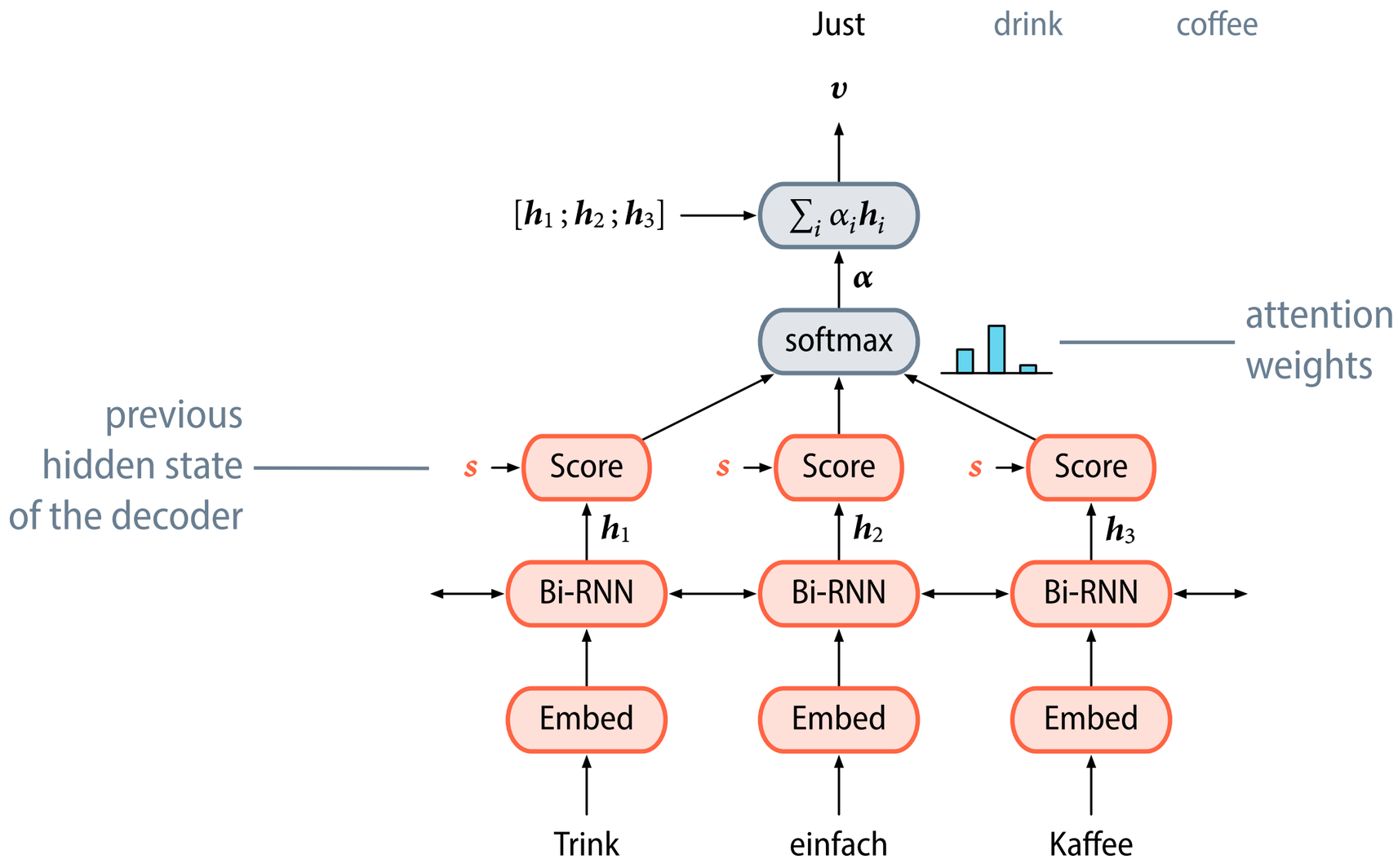

Bahdanau et al. (2015)

# Attention for translation




Bahdanau et al. (2015)

# Attention for translation



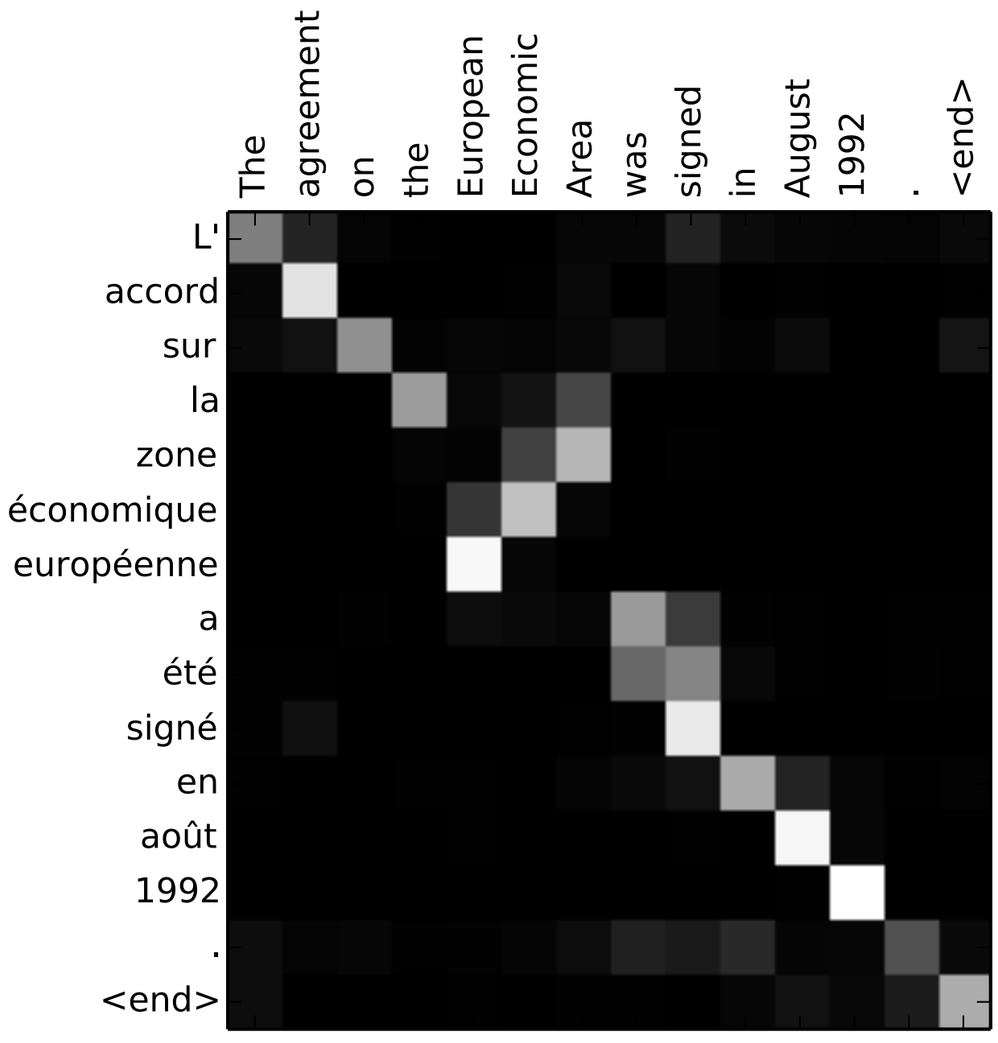

Bahdanau et al. (2015)

# A more general characterisation of attention

- In general, attention can be described as a mapping from a query $\boldsymbol{q}$ and a set of key–value pairs $\boldsymbol{k}_i$, $\boldsymbol{v}_i$ to an output.
- The output is the weighted sum of the $\boldsymbol{v}_i$, where the weight of each $\boldsymbol{v}_i$ is given by the compatibility between $\boldsymbol{q}$ and $\boldsymbol{k}_i$.
- In the translation architecture, the query $\boldsymbol{q}$ corresponds to the hidden state of the decoder; and the keys and values correspond to the hidden states of the encoder, $\boldsymbol{h}_i$.

Vaswani et al. (2017)

# Attention as word alignments




In the context of the encoder–decoder architecture for neural machine translation, attention can be interpreted as word alignments.

Image source: Bahdanau et al. (2015)

# Attention is all you need

- The attention mechanism allows the direct modelling of dependencies between words, without regard to their distance.
- However, recurrent neural networks implement a mode of computation that is essentially sequential.

  precludes parallelisation

- The **Transformer** is an architecture that eschews recurrence and instead relies entirely on an attention mechanism.

Vaswani et al. (2017)

# Structure of this unit

- Prelude:  Word embeddings
- Introduction to recurrent neural networks
- The LSTM architecture
- Use case 1:  Part-of-speech tagging
- Use case 2:  Machine translation
- Attention