NLP1 2020 - Lecture 7

Compositional semantics and sentence representations

Mario Giulianelli

Institute for Logic, Language and Computation

University of Amsterdam

18 November 2020

Outline

Compositional semantics

Compositional distributional semantics

Compositional semantics with neural networks

Outline

Compositional semantics

Compositional distributional semantics

Compositional semantics with neural networks

Compositional semantics

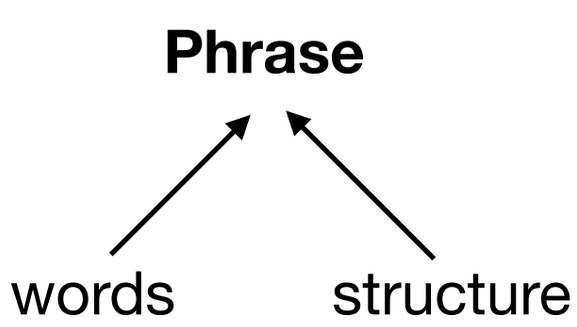
Principle of compositionality

The meaning of a complex expression is determined by the meanings of its constituents and by the rules used to combine them.

Compositional semantics

Principle of compositionality

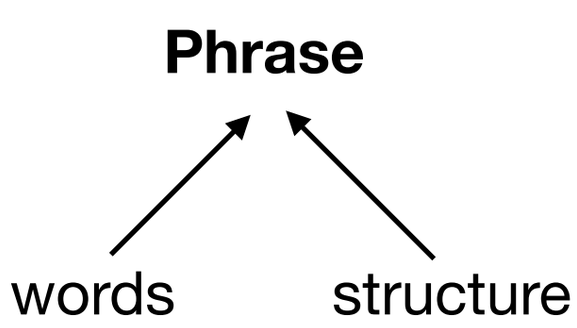
The meaning of a complex expression is determined by the meanings of its constituents and by the rules used to combine them.

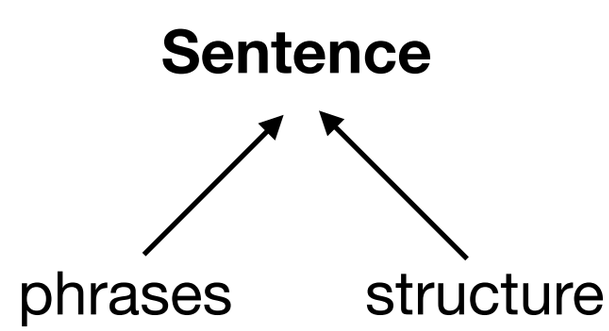


Compositional semantics

Principle of compositionality

The meaning of a complex expression is determined by the meanings of its constituents and by the rules used to combine them.

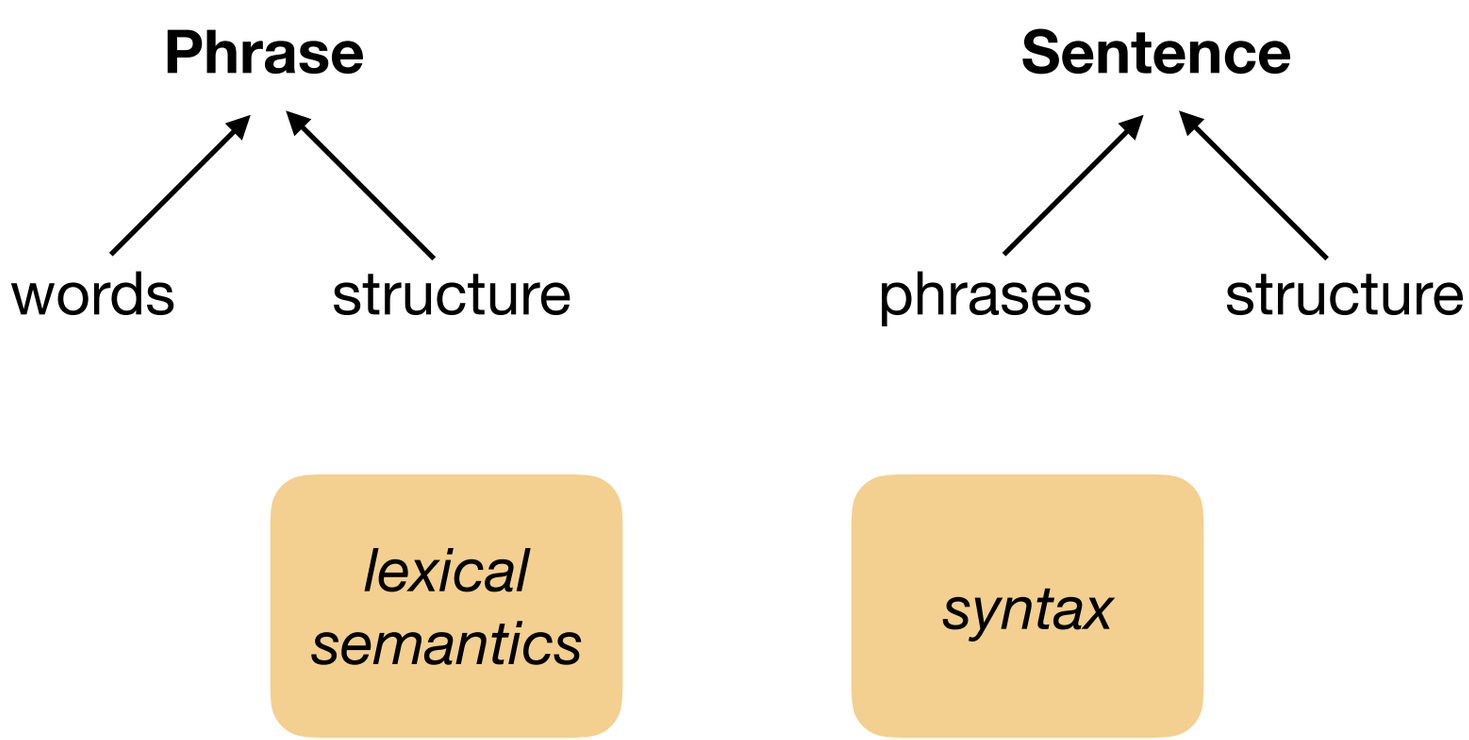




Compositional semantics

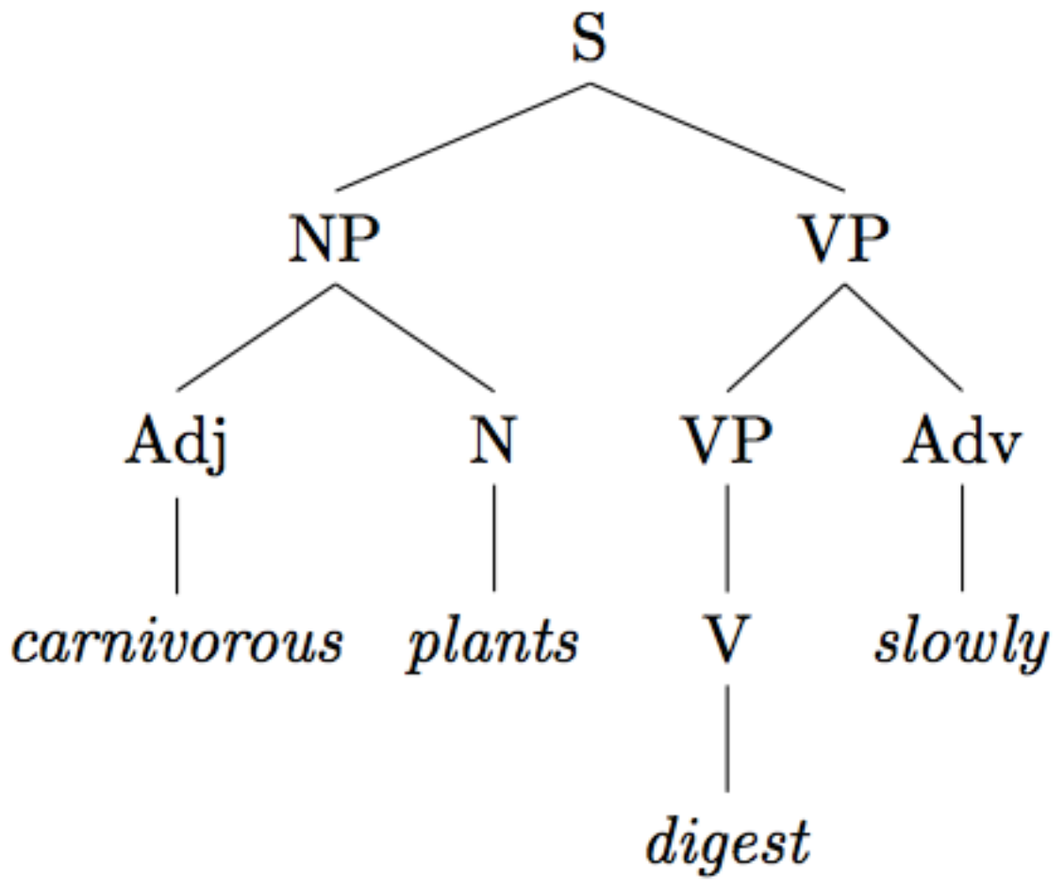
Principle of compositionality

The meaning of a complex expression is determined by the meanings of its constituents and by the rules used to combine them.



Semantic composition

What is the meaning of *"carnivorous plants digest slowly"*?



Issues with semantic composition

- Similar syntactic structures may have different meanings
  - *it runs*
  - *it rains, it snows* (here, *it* is a pleonastic pronoun)

- Different syntactic structures may have the same meaning (e.g., passive constructions)
  - *Eve ate the apple.*
  - *The apple was eaten by Eve.*

- Not all phrases are interpreted compositionally (e.g., idioms)
  - *kick the bucket*
  - *pull someone's leg*

  but the compositional interpretation is still possible.

Issues with semantic composition

- Additional meaning can arise through composition (e.g., logical metonymy)
  - *fast car*
  - *fast algorithm*
  - *begin a book*

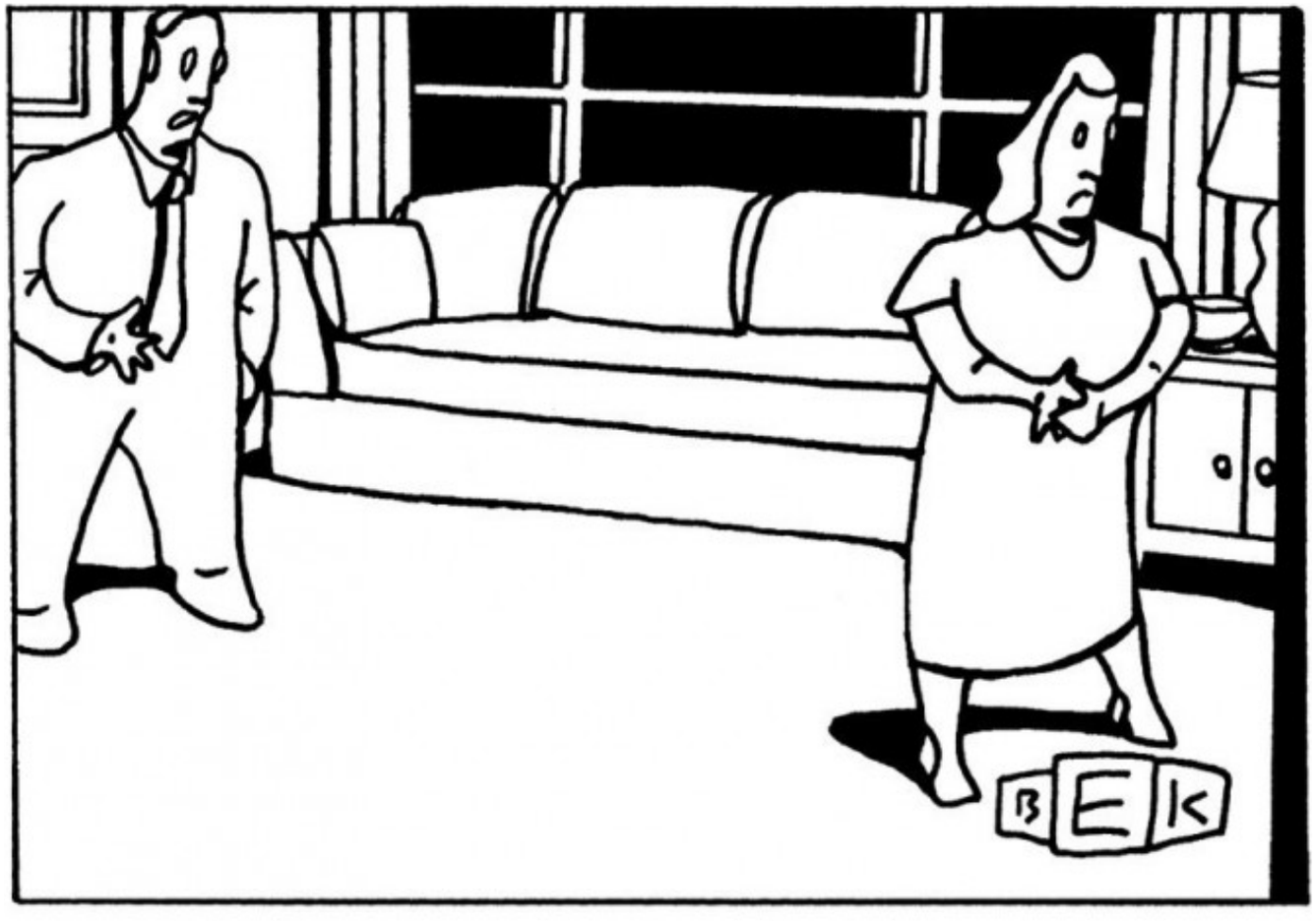
- Meaning transfers (e.g., metaphor)
  - *he put a grape into his mouth and swallowed it whole*
  - *he swallowed her story whole*

- Additional connotations can arise through composition
  - *I can't buy this story*

- Recursive composition

Issues with semantic composition

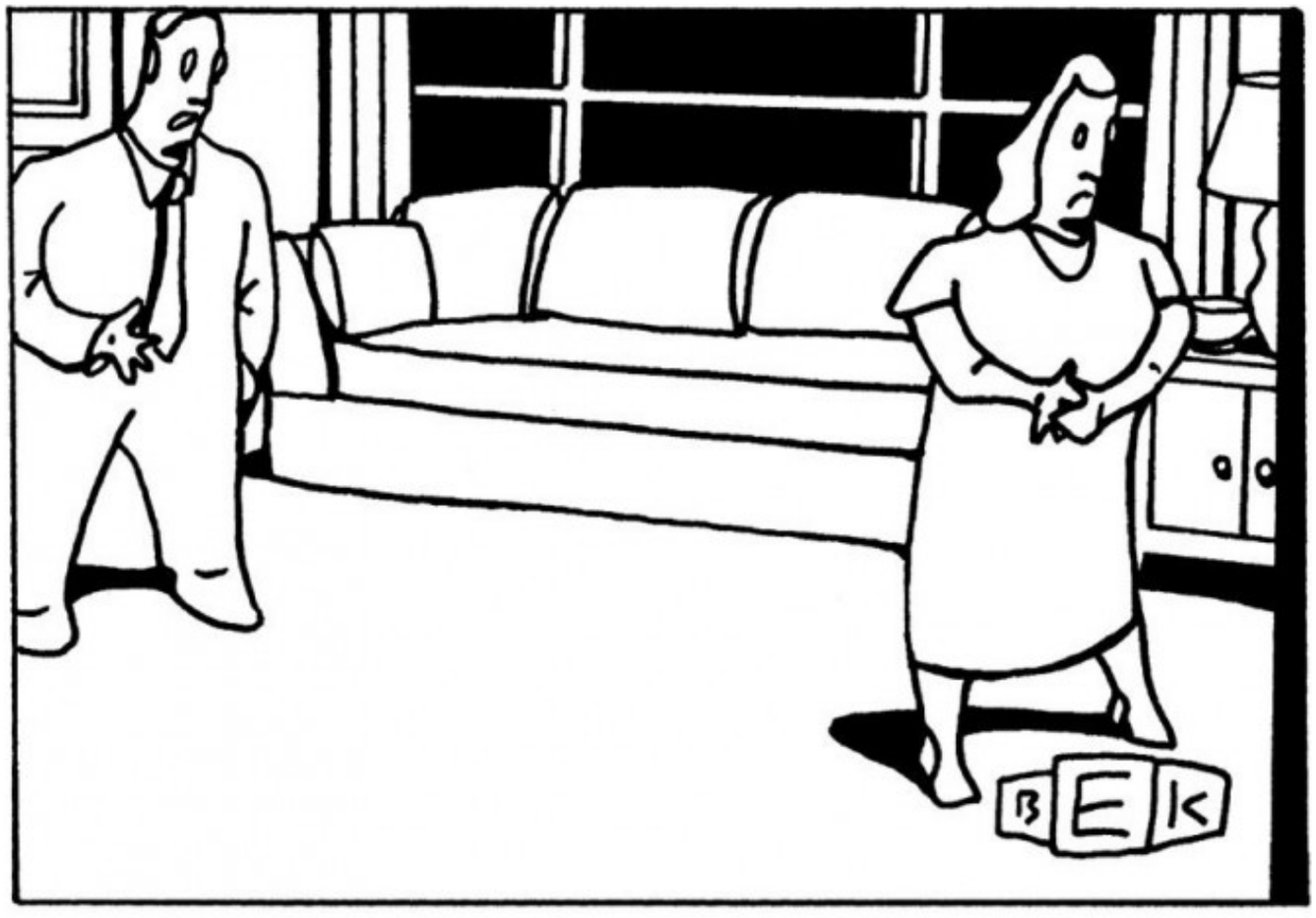
- Recursive composition



"Of course I care about how you imagined I thought you perceived I wanted you to feel."

Issues with semantic composition

- Recursive composition



"Of course I care about how you imagined I thought you perceived I wanted you to feel."

A defining property of natural languages is **productivity**:
they license a theoretically infinite set of possible expressions.

Compositionality and recursion allow for productivity.

Cautionary notes

- The meaning of the whole is constructed from its parts, **and the meaning of the parts is derived from the whole**.

- Compositionality is a matter of **degree** rather than a binary notion.

carnivorous plants *take advantage* *kick the bucket*

fully compositional **non-compositional**

Modelling compositional semantics

1. Compositional **distributional semantics**
   - composition is modelled in a vector space
   - unsupervised
   - general purpose representations

2. Compositional semantics with **neural networks**
   - (typically) supervised
   - (typically) task-specific representations

Outline

Compositional semantics

Compositional distributional semantics

Compositional semantics with neural networks

Compositional distributional semantics

it was authentic scrumpy, rather sharp and very strong
we could taste a famous local product — scrumpy
spending hours in the pub drinking scrumpy

Compositional distributional semantics

it was authentic scrumpy, rather sharp and very strong
we could taste a famous local product — scrumpy
spending hours in the pub drinking scrumpy

Can distributional semantics can be extended to account for the meaning of phrases and sentences?

Compositional distributional semantics

it was authentic scrumpy, rather sharp and very strong
we could taste a famous local product — scrumpy
spending hours in the pub drinking scrumpy

Can distributional semantics can be extended to account for the meaning of phrases and sentences?

her old dog is turning 14 this year!
you see your old dog lumber slowly to the food bowl
in an old dog, behaviour changes can appear suddenly

Compositional distributional semantics

Can distributional semantics can be extended to account for the meaning of phrases and sentences?

- Given a finite vocabulary, natural languages licence an infinite amount of sentences.
- So it is impossible to learn vector representations for all sentences.

Compositional distributional semantics

Can distributional semantics can be extended to account for the meaning of phrases and sentences?

- Given a finite vocabulary, natural languages licence an infinite amount of sentences.
- So it is impossible to learn vector representations for all sentences.

➡ But we can still use distributional word representations and learn to perform **semantic composition in distributional space**.

Compositional distributional semantics

Can distributional semantics can be extended to account for the meaning of phrases and sentences?

- Given a finite vocabulary, natural languages licence an infinite amount of sentences.
- So it is impossible to learn vector representations for all sentences.

➡ But we can still use distributional word representations and learn to perform **semantic composition in distributional space**.

vector mixture models

lexical function models

Vector mixture models

Mitchell and Lapata. (2010). Composition in distributional models of semantics. *Cognitive science.*

Words are represented as vectors, which combine to produce new vectors.

$$\mathbf{p} = f\,(\,\mathbf{u}, \mathbf{v}, R, K\,)$$

Vector mixture models

Mitchell and Lapata. (2010). Composition in distributional models of semantics. *Cognitive science.*

Words are represented as vectors, which combine to produce new vectors.

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K)$$

Vector mixture models

Mitchell and Lapata. (2010). Composition in distributional models of semantics. *Cognitive science.*

Words are represented as vectors, which combine to produce new vectors.

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K)$$

Vector mixture models

Mitchell and Lapata. (2010). Composition in distributional models of semantics. *Cognitive science.*

Words are represented as vectors, which combine to produce new vectors.

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K)$$

Constraint: **p** lies in the same n-dimensional space as **u** and **v**.
Assumption: all syntactic types are similar enough to have the same dimensionality.

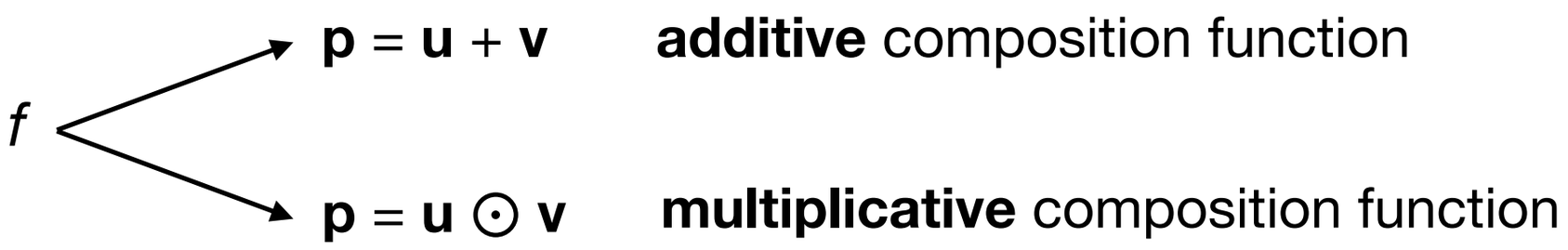
Vector mixture models

Mitchell and Lapata. (2010). Composition in distributional models of semantics. *Cognitive science.*

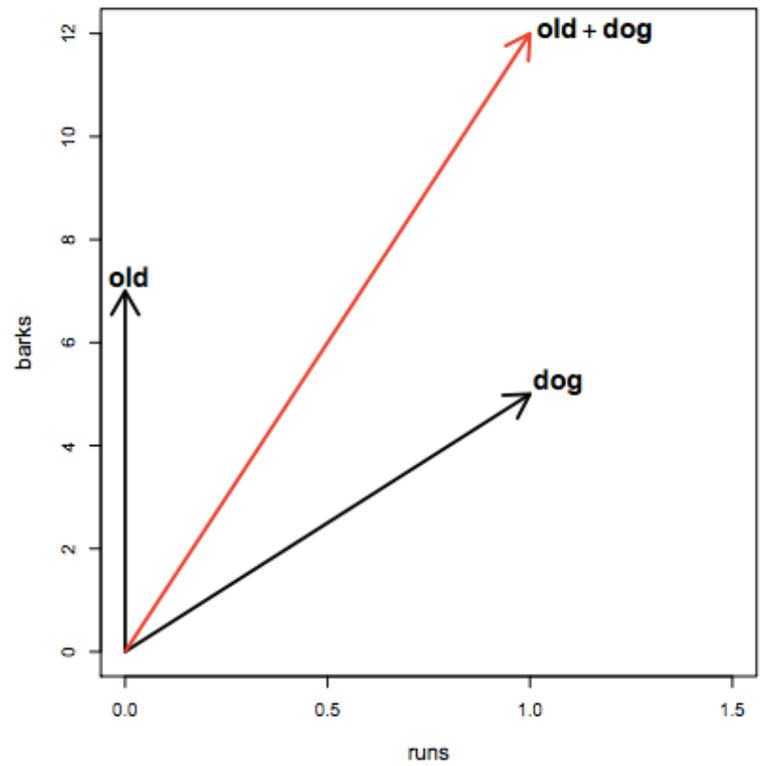
Words are represented as vectors, which combine to produce new vectors.

$$\mathbf{p} = f(\mathbf{u}, \mathbf{v}, R, K)$$

Constraint: **p** lies in the same n-dimensional space as **u** and **v**.
Assumption: all syntactic types are similar enough to have the same dimensionality.



Additive and multiplicative models



				additive		multiplicative	
	dog	**cat**	**old**	**old + dog**	**old + cat**	**old ⊙ dog**	**old ⊙ cat**
runs	1	4	0	1	4	0	0
barks	5	0	7	12	7	35	0

Additive and multiplicative models

- The additive and the multiplicative model are **symmetric** (commutative): they do not take word order or syntax into account.
  - ➡ *John hit the ball* = *The ball hit John*

- Correlate with human similarity judgments about adjective-noun, noun-noun, verb-noun and noun-verb pairs

- More suitable for modelling **content words**, would not apply well to function words:
  - ➡ *some dogs, lice and dogs, lice on dogs*

Lexical function models

~~Assumption: all syntactic types are similar enough to have the same dimensionality.~~

$$\mathbf{p} = f(\mathbf{U}, \mathbf{v}, R, K)$$

Lexical function models

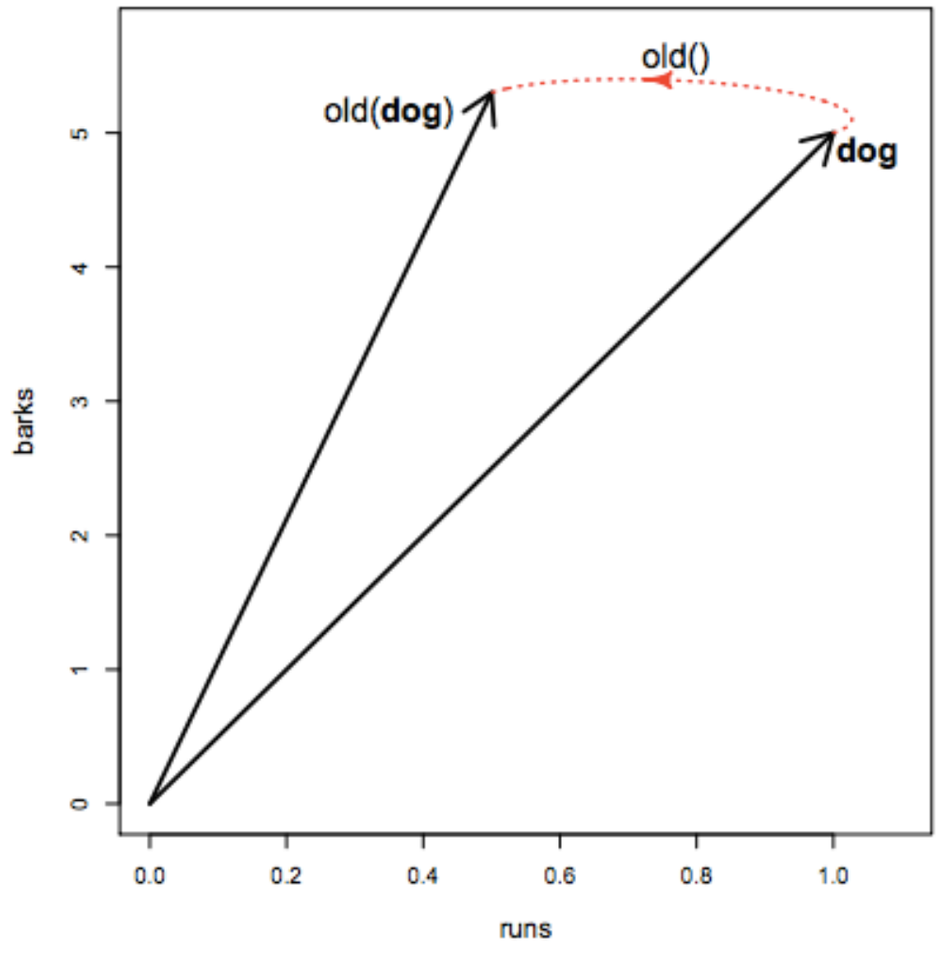
~~Assumption: all syntactic types are similar enough to have the same dimensionality.~~

$$\mathbf{p} = f(\mathbf{U}, \mathbf{v}, R, K)$$

Distinguish between:

- words whose meaning is directly determined by their distributional profile, e.g. nouns
- words that act as functions **transforming** the distributional profile of other words, e.g., adjectives, adverbs

$$\mathbf{p} = f(\mathbf{U}, \mathbf{v}, ADJ) = \mathbf{Uv}$$



Lexical function models

Baroni and Zamparelli. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP.*

Adjectives modelled as lexical functions that are applied to nouns: *old dog* = *old*(*dog*)

- Adjectives are parameter matrices (**A***old*, **A***furry*, etc.)
- Nouns are vectors (**house**, **dog**, etc.)
- Composition is a linear transformation: **old dog** = **A***old* $\times$ **dog.**

Learning adjective matrices

For each adjective, learn a parameter matrix that allows to predict adjective-noun phrase vectors.

	X	Y
Training set	**house**	**old house**
	dog	**old dog**
	car	**old car**
	cat	**old cat**
	toy	**old toy**
	…	…
Test set	**elephant**	**old elephant**
	mercedes	**old mercedes**

Learning adjective matrices

1. Obtain a distributional vector $\mathbf{n}_j$ for each noun n_j in the vocabulary using a conventional DSM.

2. Collective all adjective-noun pairs (a_i, n_j) from the corpus.

3. Obtain a distributional phrase vector $\mathbf{p}_{ij}$ for each pair (a_i, n_j) from the same corpus using a conventional DSM—treating the phrase $a_i\ n_j$ as a single word.

4. The set of tuples $\{(\mathbf{n}_j, \mathbf{p}_{ij})\}_j$ represents a dataset $\mathscr{D}(a_i)$ for the adjective a_i.

5. Learn matrix $\mathbf{A}_i$ from $\mathscr{D}(a_i)$ using linear regression. Minimise the squared error loss:

$$L(\mathbf{A}_i) = \sum_{j \in \mathscr{D}(a_i)} \|\mathbf{p}_{ij} - \mathbf{A}_i \mathbf{n}_j\|^2$$

Verbs as lexical functions

Verbs too can modelled as lexical functions that are applied to their arguments.

They are represented as tensors whose order is determined by the subcategorisation frame of the verb (i.e., how many and what type of arguments the verb takes).

- **Intransitive** verbs take a subject as their only argument

  dogs bark $\mathbf{V}_{bark} \times \mathbf{dogs}$

  modelled as a matrix (second-order tensor)

- **Transitive** verbs take a subject and an object

  dogs eat meat $(\mathbf{V}_{eat} \times \mathbf{meat}) \times \mathbf{dogs}$

  modelled as a third-order tensor

Modelling compositional semantics

1. Compositional **distributional semantics**
   - composition is modelled in a vector space
   - unsupervised learning
   - general purpose representations

2. Compositional semantics with **neural networks**
   - (typically) supervised learning
   - (typically) task-specific representations

Modelling compositional semantics

1. Compositional **distributional semantics**
   - composition is modelled in a vector space
   - unsupervised learning
   - general purpose representations

2. Compositional semantics with **neural networks**
   - (typically) supervised learning
   - (typically) task-specific representations

Task: sentiment classification (Practical 2)

Outline

Compositional semantics

Compositional distributional semantics

Compositional semantics with neural networks

Compositional semantics with NNs

1. Learn sentence (or phrase) representations
2. Learn to make task-specific predictions based on the sentence (or phrase) representation

Compositional semantics with NNs

1. Learn sentence (or phrase) representations
2. Learn to make task-specific predictions based on the sentence (or phrase) representation

Task: Sentiment classification

Darkly funny and frequently insightful

0. very negative
1. negative
2. neutral
3. positive
4. very positive

Task: Sentiment classification

Darkly *funny and frequently insightful*

0. very negative
1. negative
2. neutral
3. positive
4. very positive

Task: Sentiment classification

*Darkly **funny** and frequently insightful*

0. very negative
1. negative
2. neutral
3. positive
4. very positive

Task: Sentiment classification

Darkly funny *and frequently insightful*

0. very negative
1. negative
2. neutral
3. positive
4. very positive

Task: Sentiment classification

*Darkly funny **and** frequently insightful*

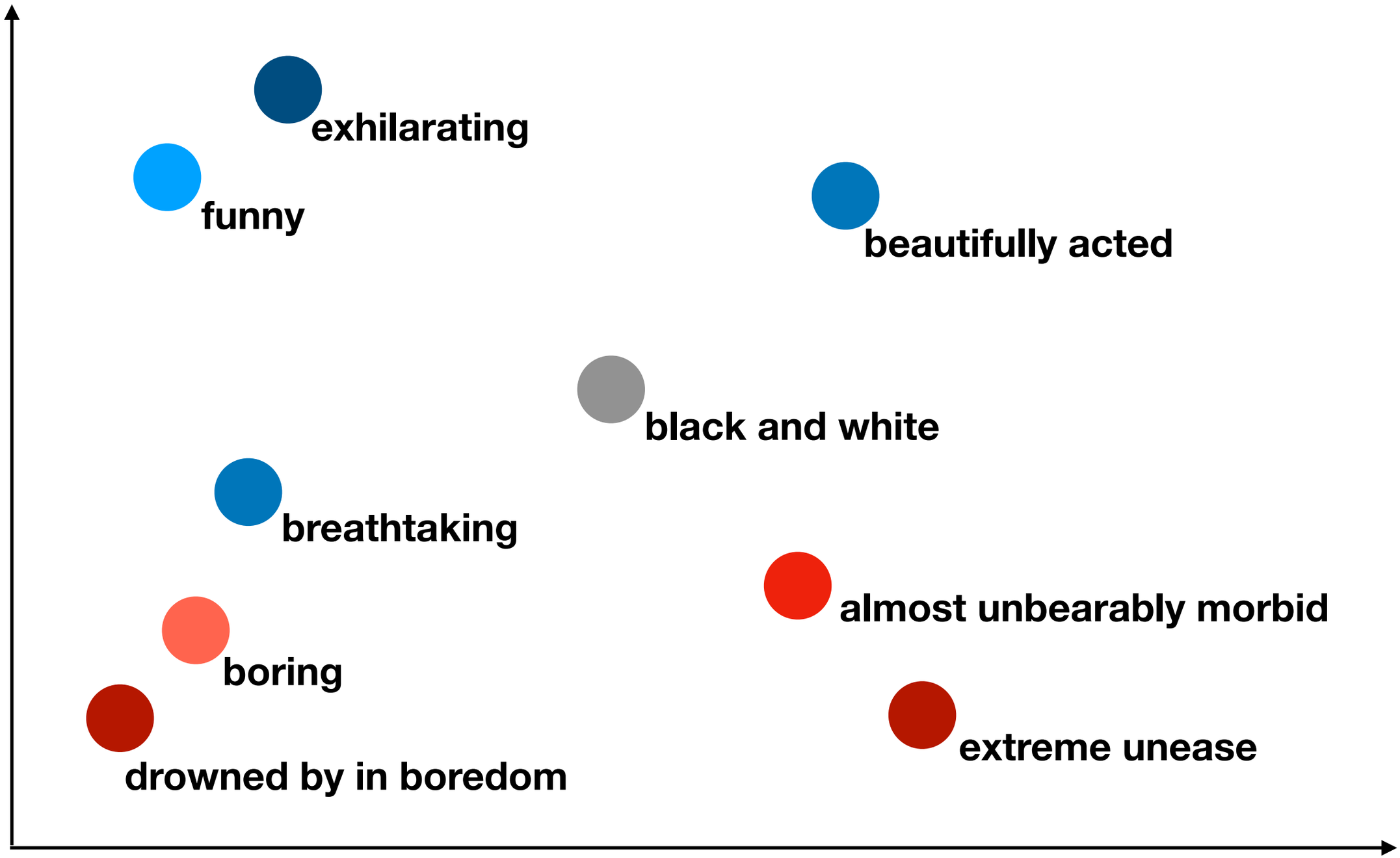
0. very negative
1. negative
2. neutral
3. positive
4. very positive

Task: Sentiment classification

Darkly funny and frequently insightful

0. very negative
1. negative
2. neutral
3. positive
4. very positive

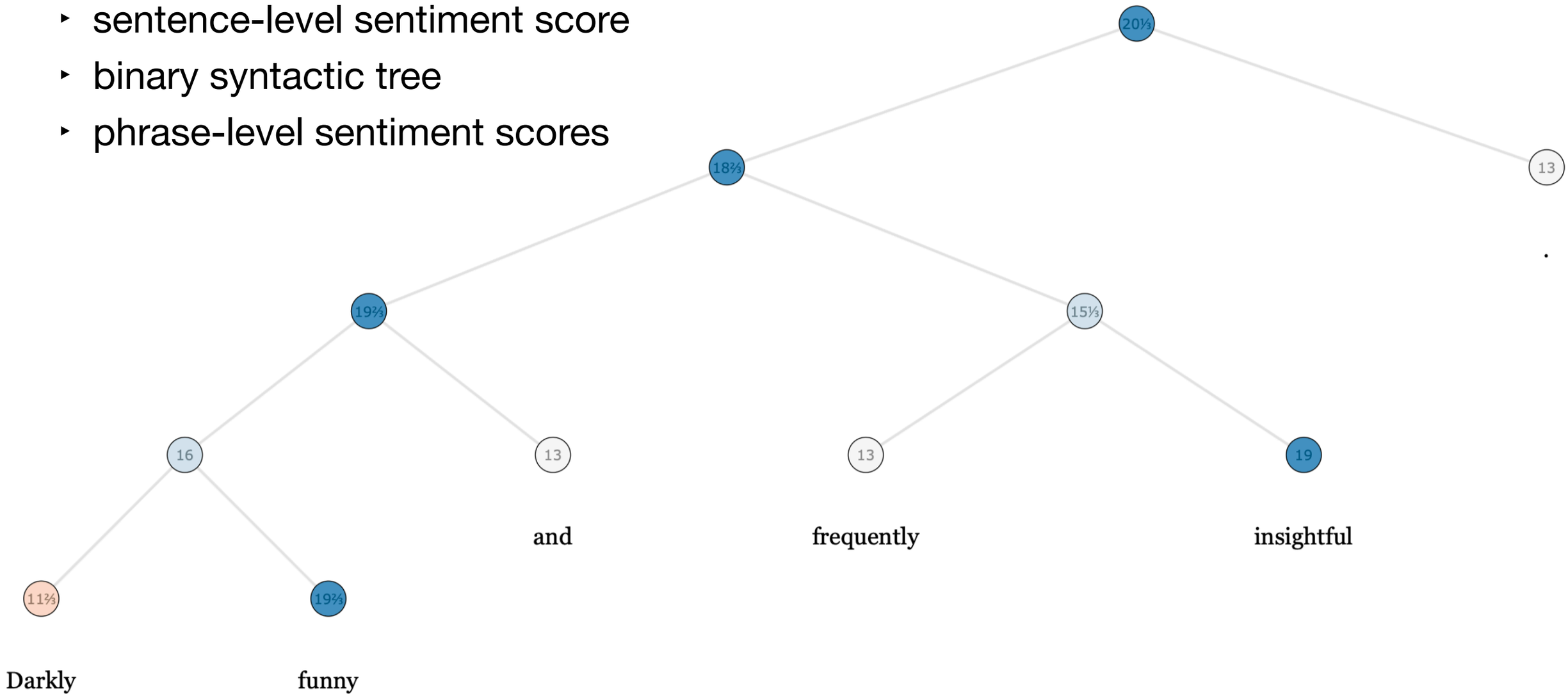
Task-specific representations



Dataset: Stanford Sentiment Treebank

12K movie reviews

- one sentence per review
- sentence-level sentiment score
- binary syntactic tree
- phrase-level sentiment scores



Compositional semantics with NNs

1. Learn sentence (or phrase) representations
2. Learn to make task-specific predictions based on the sentence (or phrase) representation

Compositional semantics with NNs

Models

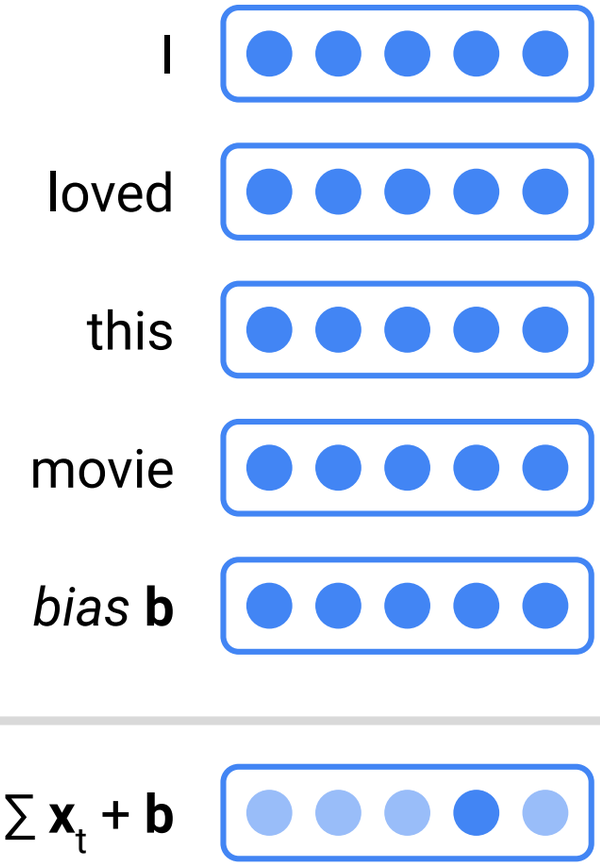
1. Bag of Words (BOW)
2. Continuous Bag of Words (CBOW)
3. Deep Continuous Bag of Words (Deep CBOW)
4. Deep CBOW with pre-trained word embeddings
5. LSTM
6. Tree-LSTM

Bag of Words

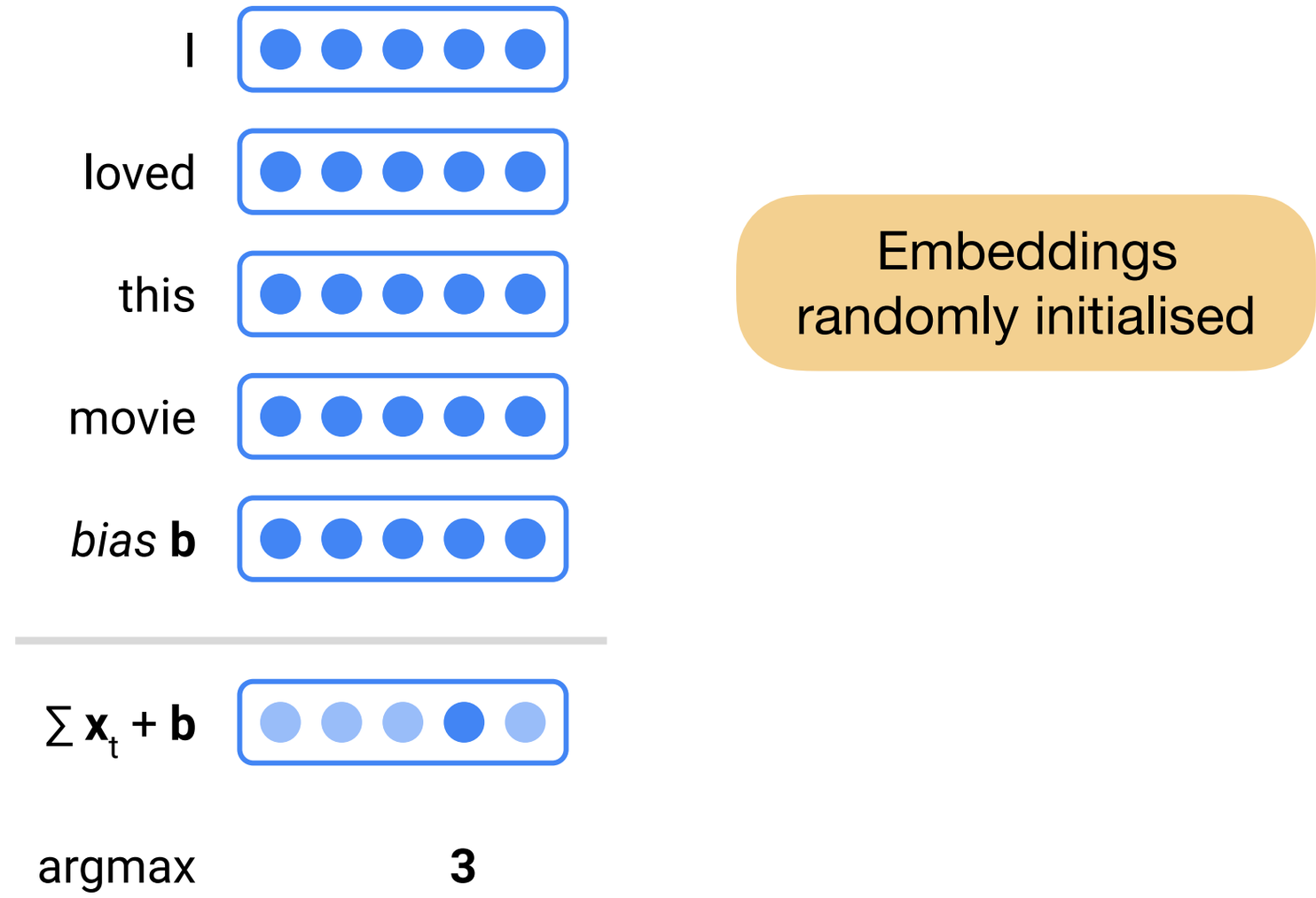
- Additive model: does not take word order or syntax into account
- Task-specific word representations with **fixed dimensionality** ($d = 5$)
- Dimensions of vector space are explicit, **interpretable**

Bag of Words

Sum word embeddings, add bias

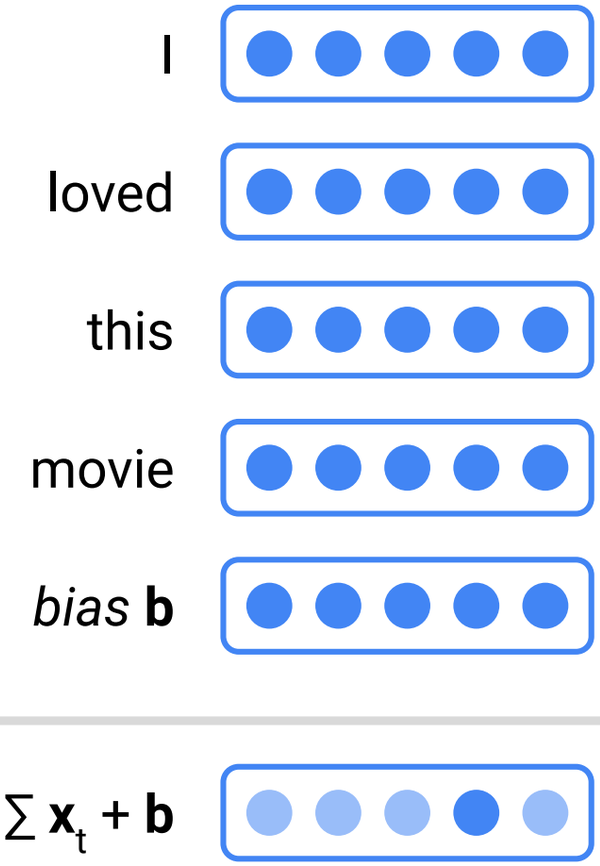


Bag of Words


Sum word embeddings, add bias
I
loved
this
movie
bias **b**
$\sum \mathbf{x}_t + \mathbf{b}$
argmax
3
Embeddings randomly initialised

Bag of Words

Sum word embeddings, add bias



What will be the role of the bias vector?

Bag of Words

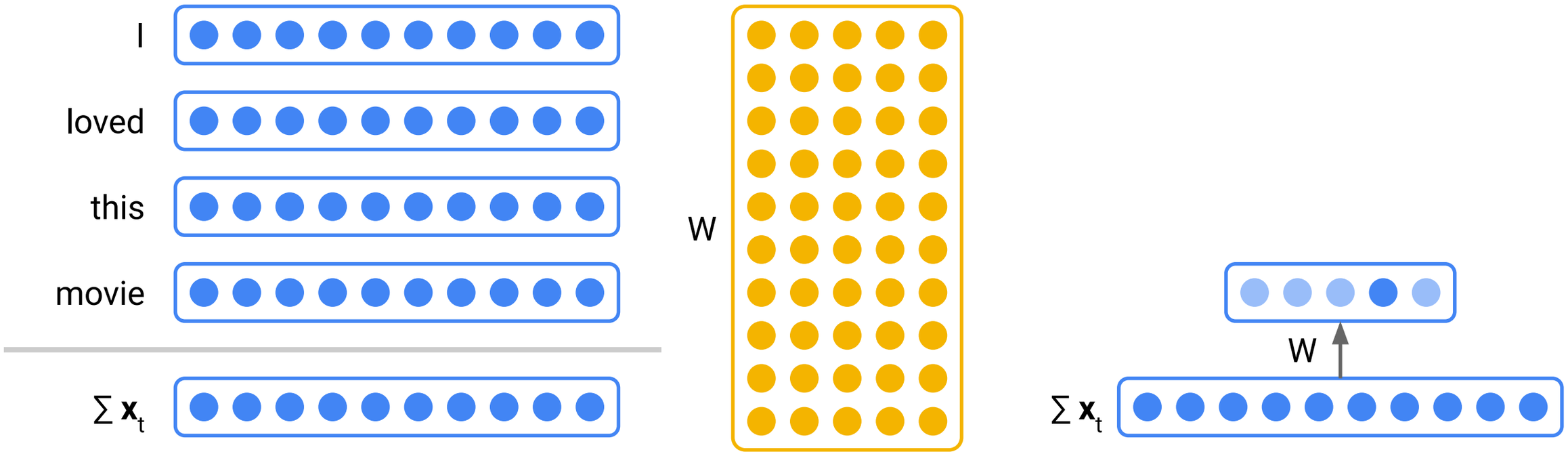
```
this    [0.0, 0.1, 0.1, 0.1, 0.0]
movie   [0.0, 0.1, 0.1, 0.2, 0.1]
is      [0.0, 0.1, 0.0, 0.0, 0.0]
stupid  [0.9, 0.5, 0.1, 0.0, 0.0]

bias    [0.0, 0.0, 0.0, 0.0, 0.0]
--------------------------------
sum     [0.9, 0.8, 0.3, 0.3, 0.1]

argmax: 0 (very negative)
```

Continuous Bag of Words

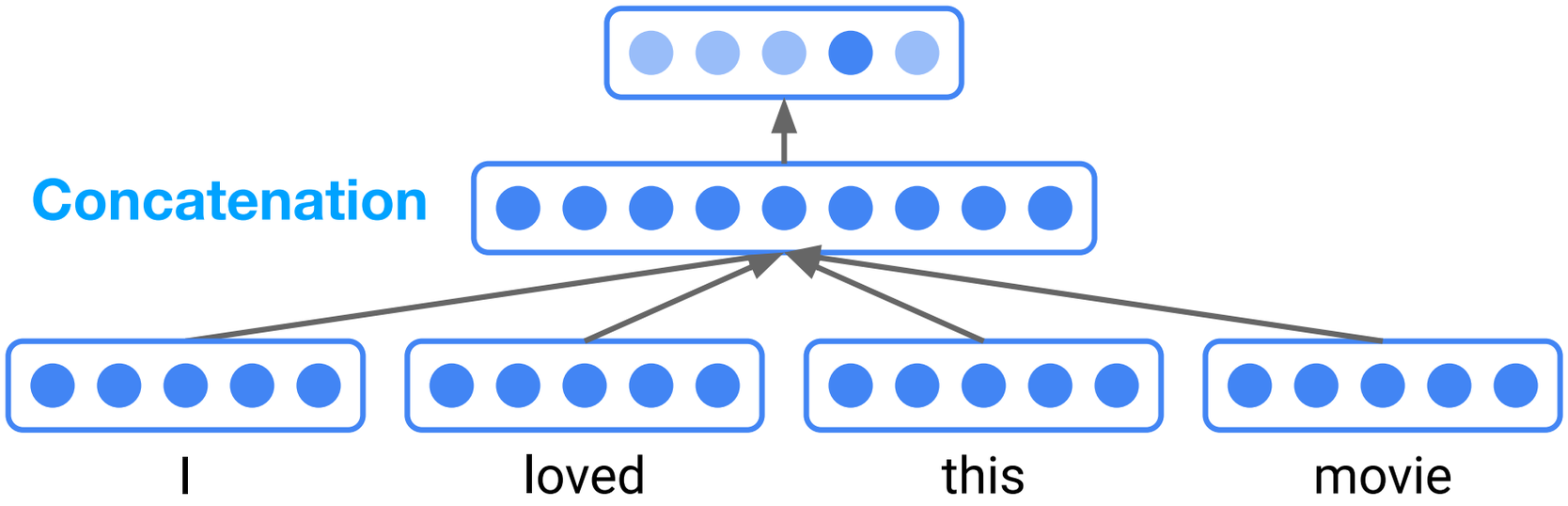
- Additive model: does not take word order or syntax into account
- Task-specific word representations of **arbitrary dimensionality**
- Dimensions of vector space are **not interpretable**
- Prediction can be traced back to the sentence vector dimensions

Continuous Bag of Words

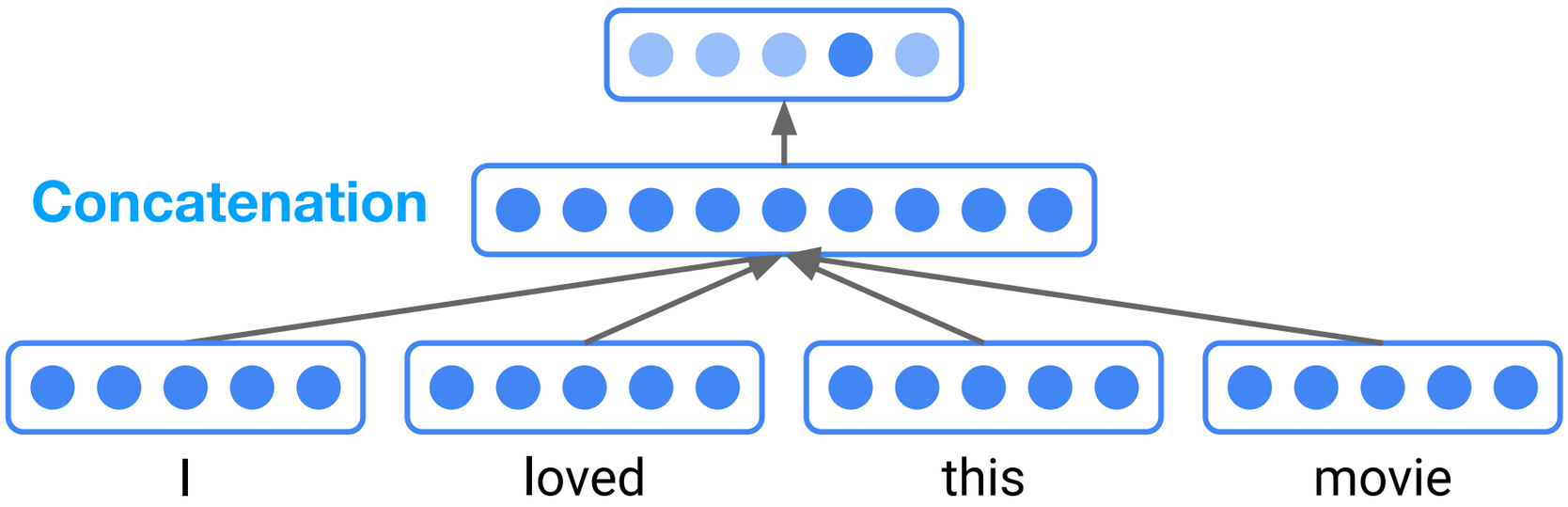
Sum word embeddings, project to 5D using W, add bias: W $(\sum \mathbf{x}_t)$ + **b**



Continuous Bag of Words



Continuous Bag of Words



Variable sentence vector size, dependent on sentence length

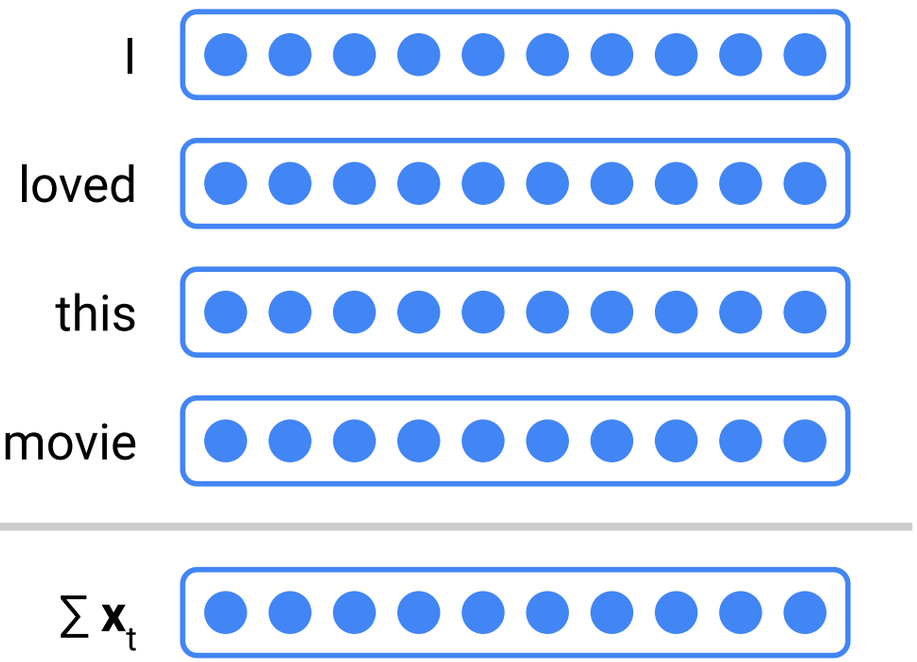
- Not very sensible conceptually
  - ➡ sentences in a different vector space than words
  - ➡ one vector space for each sentence length in the dataset
- Hardly practicable
  - ➡ what size should the transformation matrix be?
  - ➡ vector size can grow very large

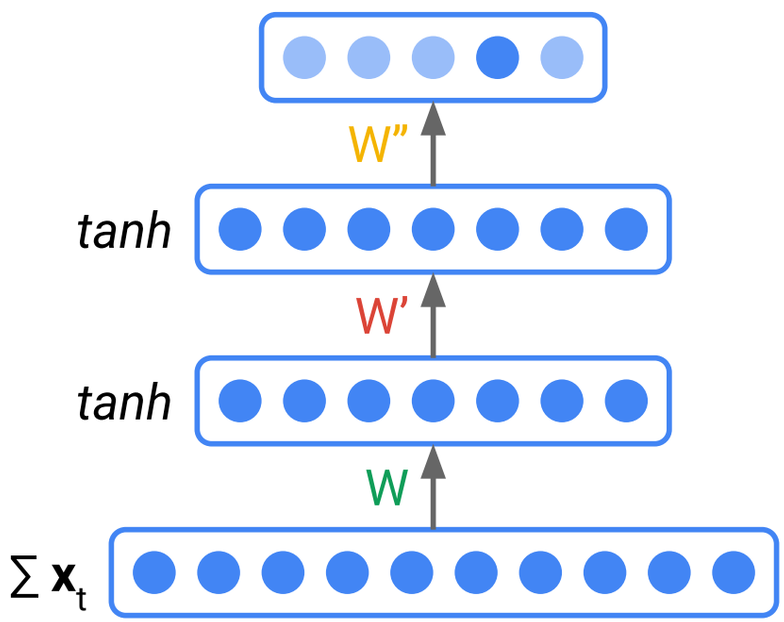
Deep Continuous Bag of Words

- Additive model: does not take word order or syntax into account
- Task-specific word representations of **arbitrary dimensionality**
- Dimensions of vector space are **not interpretable**
- **More layers and non-linear transformations**: prediction cannot be easily traced back

Deep Continuous Bag of Words

$$W'' \tanh(W' \tanh(W (\sum \mathbf{x}_t) + \mathbf{b}) + \mathbf{b}') + \mathbf{b}''$$





Deep CBOW + pre-trained embeddings

- Additive model: does not take word order or syntax into account

- Pre-trained **general-purpose** word representations (e.g., Skip-gram, GloVe)
  - ➡ **frozen**: not updated during training
  - ➡ **fine-tuned**: updated with task-specific learning signal (*specialised*)

- Dimensions of vector space are not interpretable

- Multiple layers and non-linear transformations: prediction cannot be easily traced back

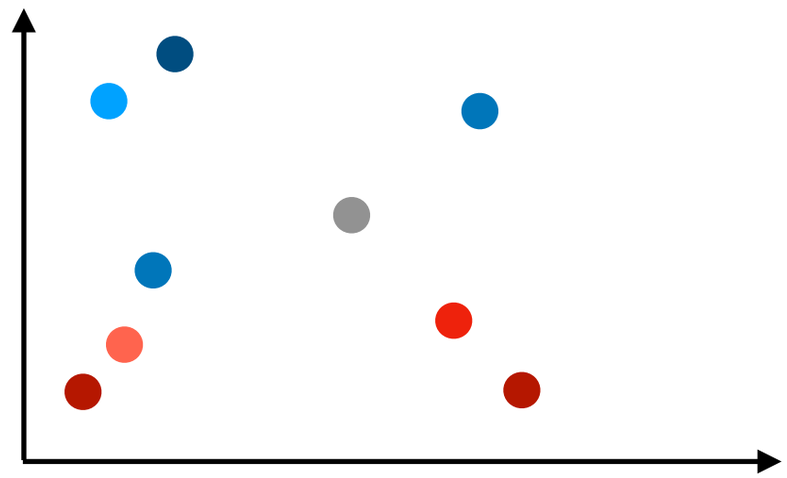
Deep CBOW + pre-trained embeddings

- Additive model: does not take word order or syntax into account

- Pre-trained **general-purpose** word representations (e.g., Skip-gram, GloVe)
  - ➡ **frozen**: not updated during training
  - ➡ **fine-tuned**: updated with task-specific learning signal (*specialised*)

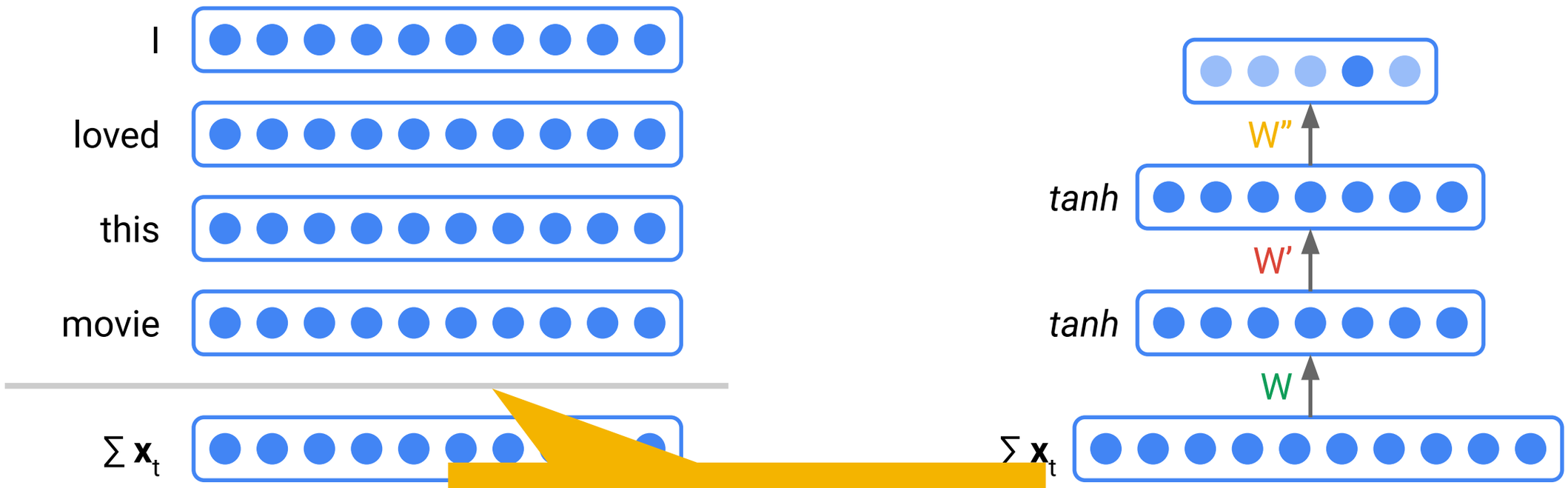
- Dimensions of vector space are not interpretable

- Multiple layers and non-linear transformations: prediction cannot be easily traced back



Deep CBOW + pre-trained embeddings

$$W'' \tanh(W' \tanh(W (\sum \mathbf{x}_t) + \mathbf{b}) + \mathbf{b}') + \mathbf{b}''$$



Training feedforward models

Train networks using Stochastic Gradient Descent (SGD):

1. **Sample** a training example
2. **Forward pass**: compute network activations and obtain an output vector
3. **Loss**: compare output vector with ground-truth label
4. **Compute gradient** of loss w.r.t (trainable) network parameters
5. **Backpropagation**: update parameters taking a step in the opposite direction of the gradient

Model predictions

output vector (*logits*)

$o = [-0.1,\ 0.1,\ 0.1,\ 2.4,\ 0.2]$

Model predictions

output vector (*logits*)

$o = [-0.1, 0.1, 0.1, 2.4, 0.2]$

$$argmax(o) \longrightarrow$$

class prediction

3

Model predictions

output vector (*logits*)

$o = [-0.1, 0.1, 0.1, 2.4, 0.2]$

$\xrightarrow{argmax(o)}$

class prediction

3

$$softmax(o)_i = \frac{exp(o_i)}{\sum_j exp(o_j))}$$

$\hat{y} = [0.06, 0.07, 0.07, 0.72, 0.08]$

Model predictions

output vector (*logits*)

$o = [-0.1, 0.1, 0.1, 2.4, 0.2]$

$argmax(o)$

class prediction

3

$$softmax(o)_i = \frac{exp(o_i)}{\sum_j exp(o_j))}$$

$\hat{y} = [0.06, 0.07, 0.07, 0.72, 0.08]$

$y = [0, \quad 0, \quad 0, \quad 1, \quad 0]$

Model predictions

output vector (*logits*)

$o = [-0.1, 0.1, 0.1, 2.4, 0.2]$

$\xrightarrow{argmax(o)}$

class prediction

3

$$softmax(o)_i = \frac{exp(o_i)}{\sum_j exp(o_j))}$$

$\hat{y} = [0.06, 0.07, 0.07, 0.72, 0.08]$

$y = [0, \quad 0, \quad 0, \quad 1, \quad 0]$

Cross Entropy loss

$$-\sum_i y_i \log \hat{y}_i = -\log \hat{y}_{argmax(o)}$$

Recurrent neural networks

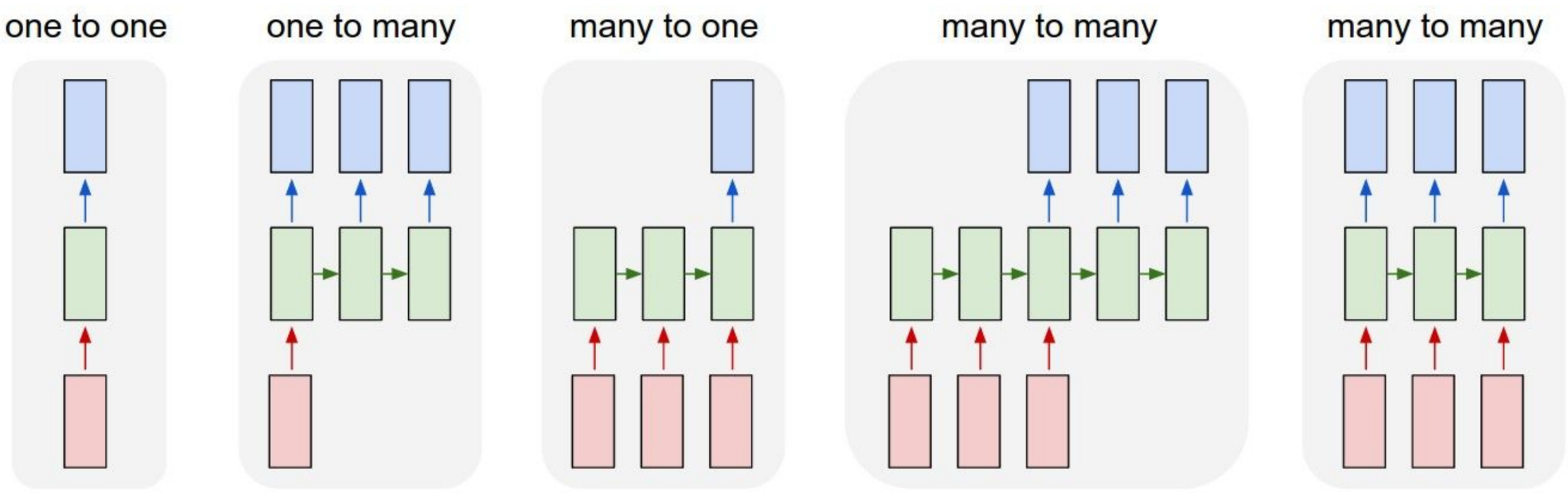
Recurrent neural networks

- Networks that contain a cycle within their connections
  - ➡ The value of a network unit is dependent on earlier outputs

- Naturally handle **sequential input**: process one element at a time

- Drop history independence assumption
  - ➡ Capture and exploit the **temporal nature of language**
  - ➡ Capture word order (and syntactic structure)

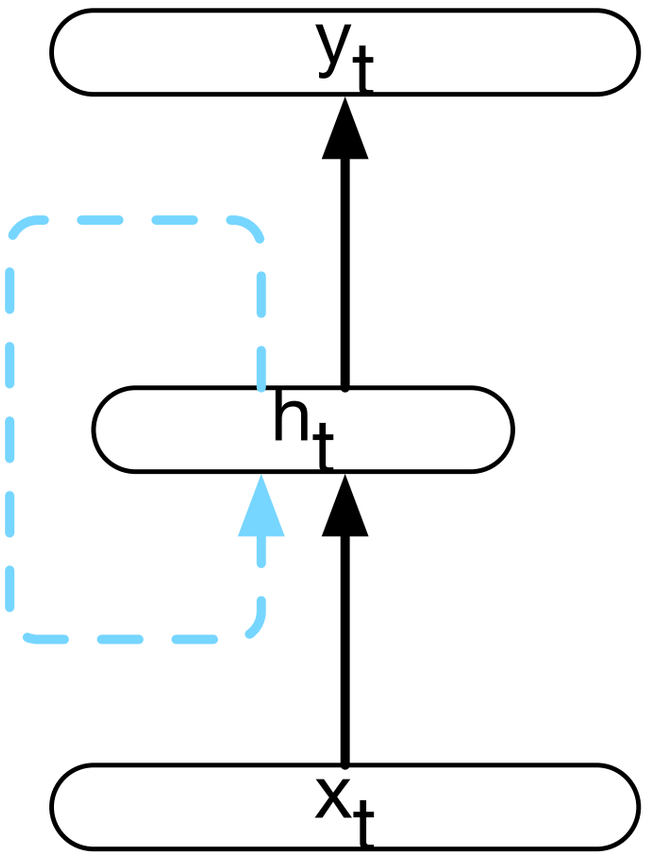
Recurrent neural networks



Credits: Fei-Fei Li & Justin Johnson & Serena Yeung (Stanford CS231 2017)

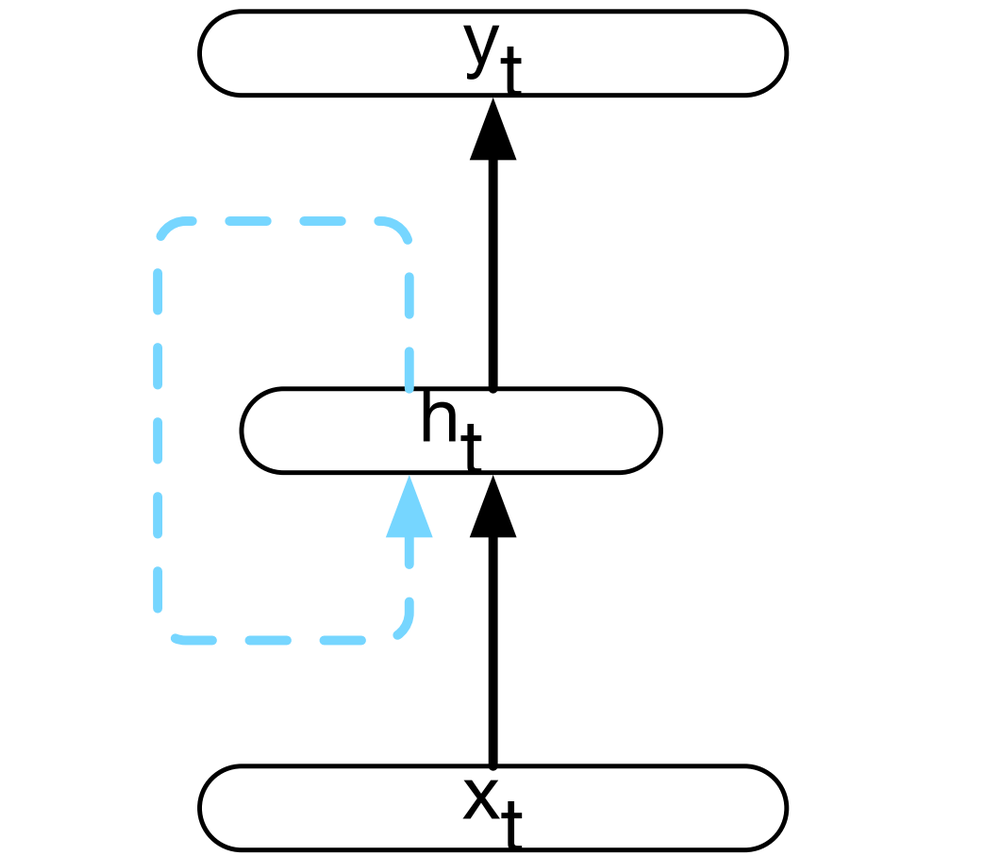
Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*



Simple recurrent neural networks

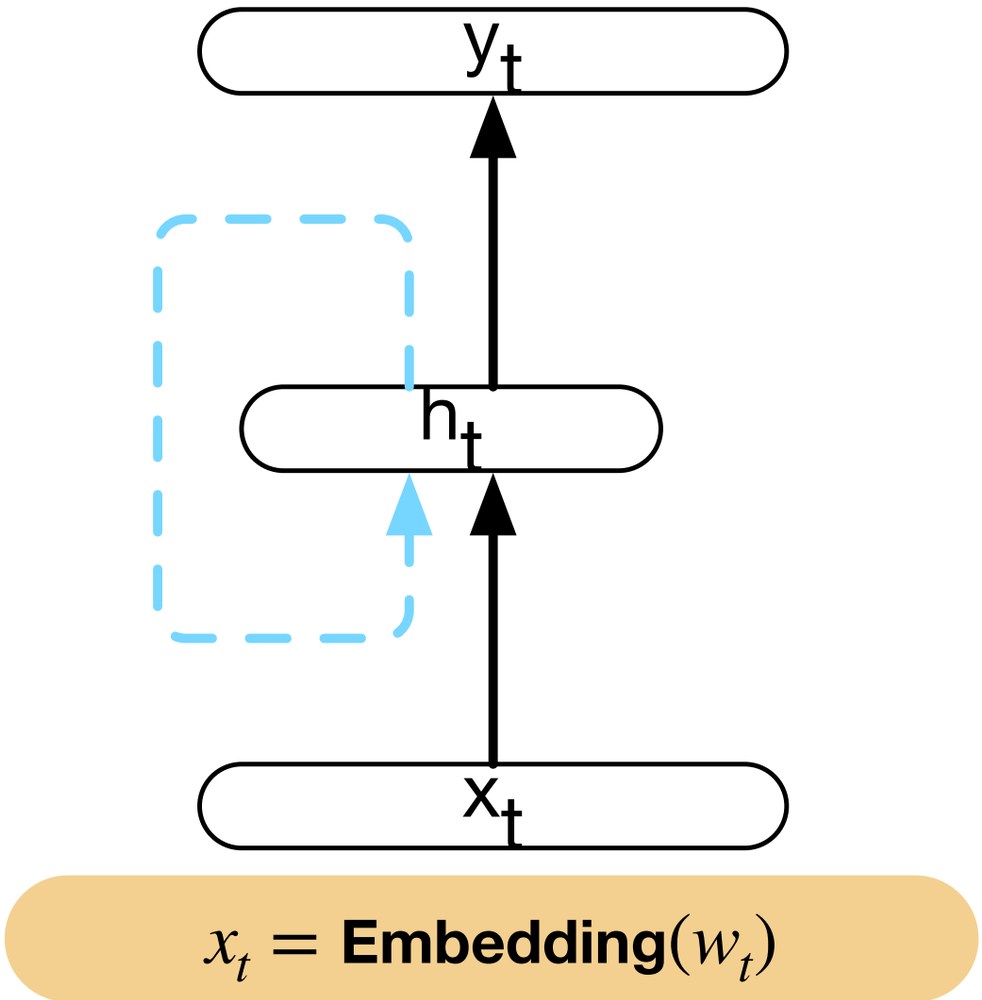
Elman, J. L. (1990). Finding structure in time. *Cognitive science.*



$$x_t = \textbf{Embedding}(w_t)$$

Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*



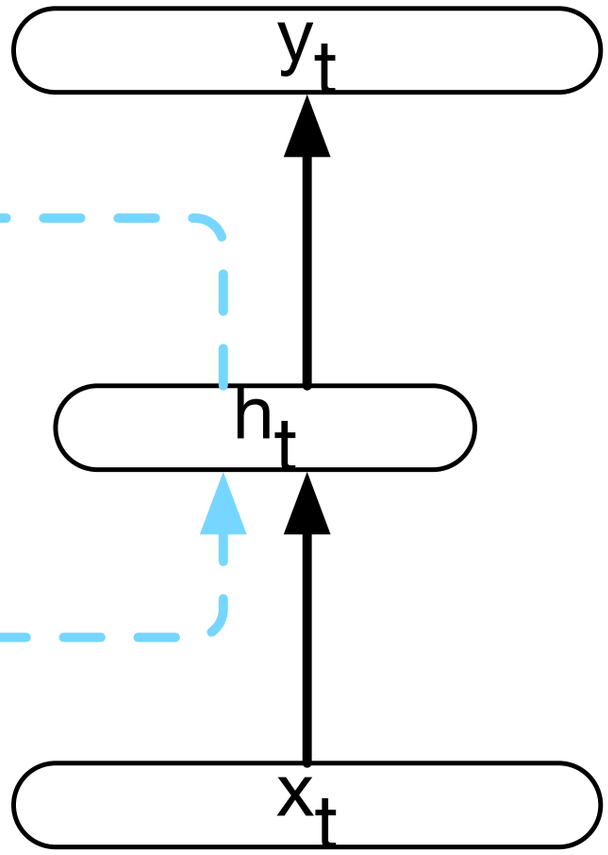
Darkly funny and frequently insightful

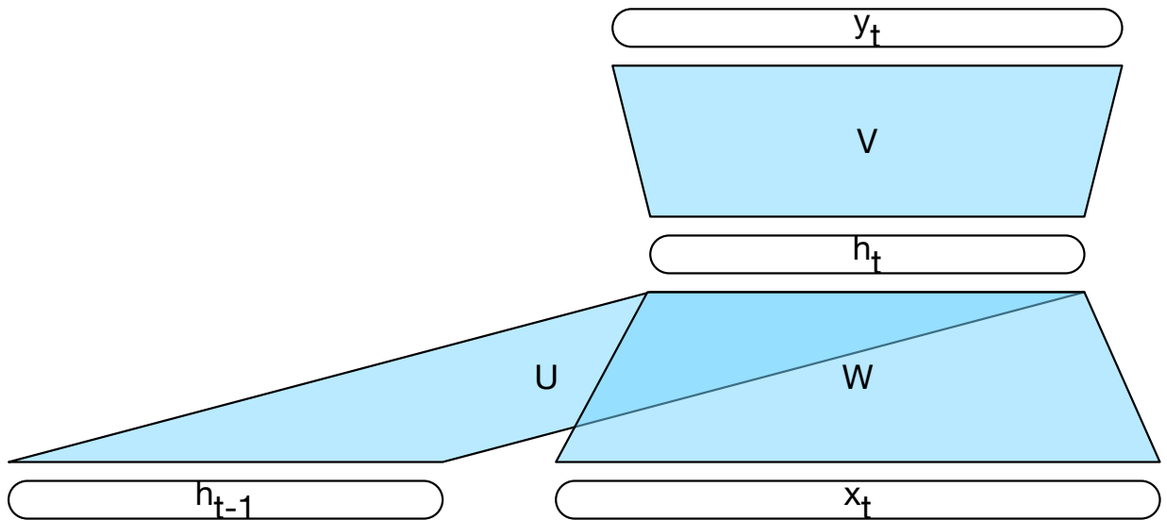
x_1 x_2 x_3 x_4 x_5

$$h_1 = f(x_1, h_0)$$
$$h_2 = f(x_2, f(x_1, h_0)) = f(x_2, h_1)$$
$$h_3 = f(x_3, f(x_2, f(x_1, h_0))) = f(x_3, h_2)$$
$$\dots$$

Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

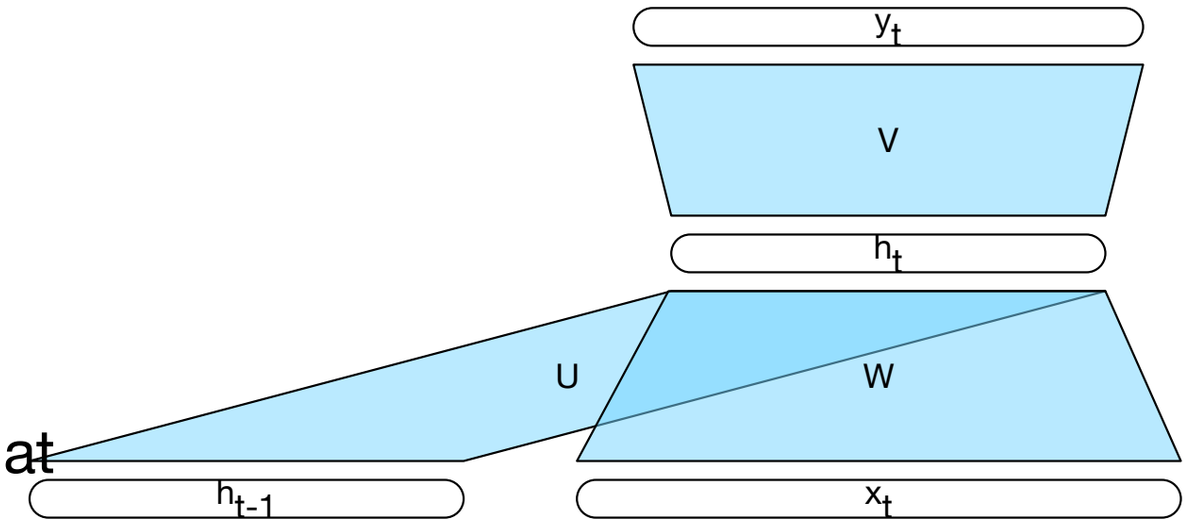




Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

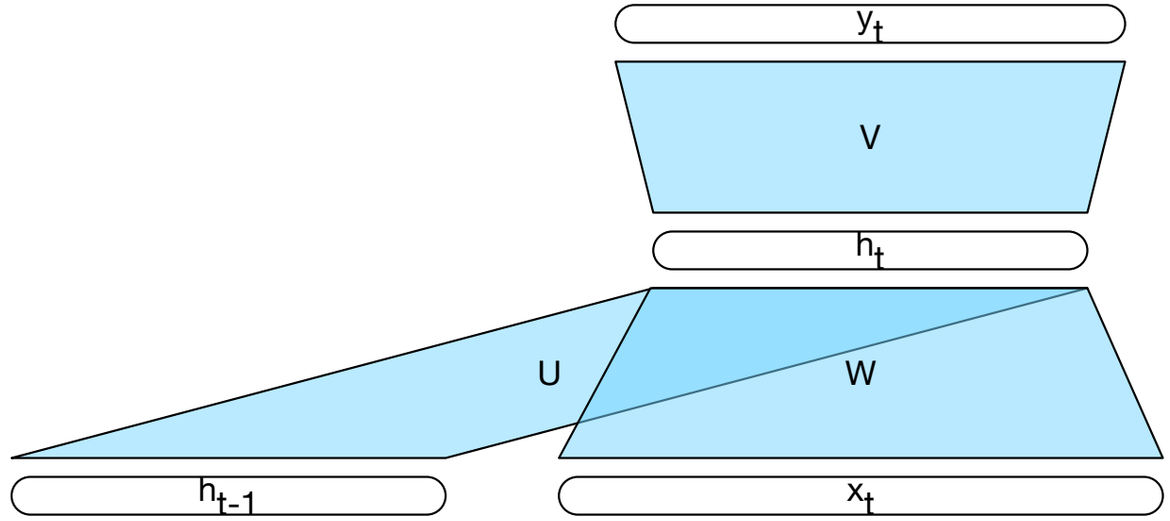
- Consist of a **single cell**, the “RNN cell”, that is fed input elements one at a time (one for each timestep)
- Hidden layer from previous timestep provides a form of **memory of preceding context**
  - ➡ encodes earlier processing
  - ➡ informs the decisions to be made at later points in time
- No fixed limit on the length of the preceding context



Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

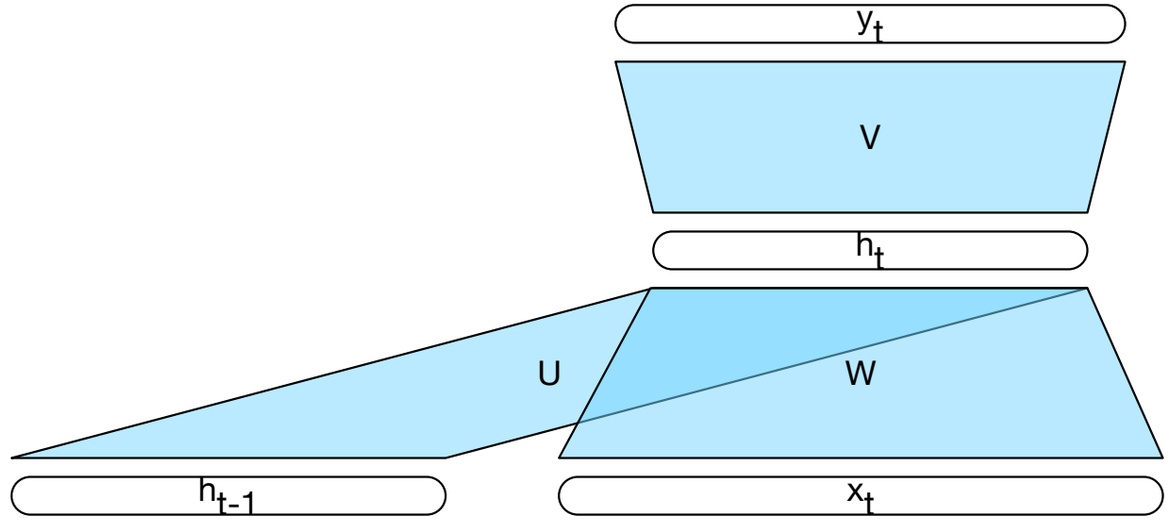
$$h_t = f(Uh_{t-1} + Wx_t)$$



Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

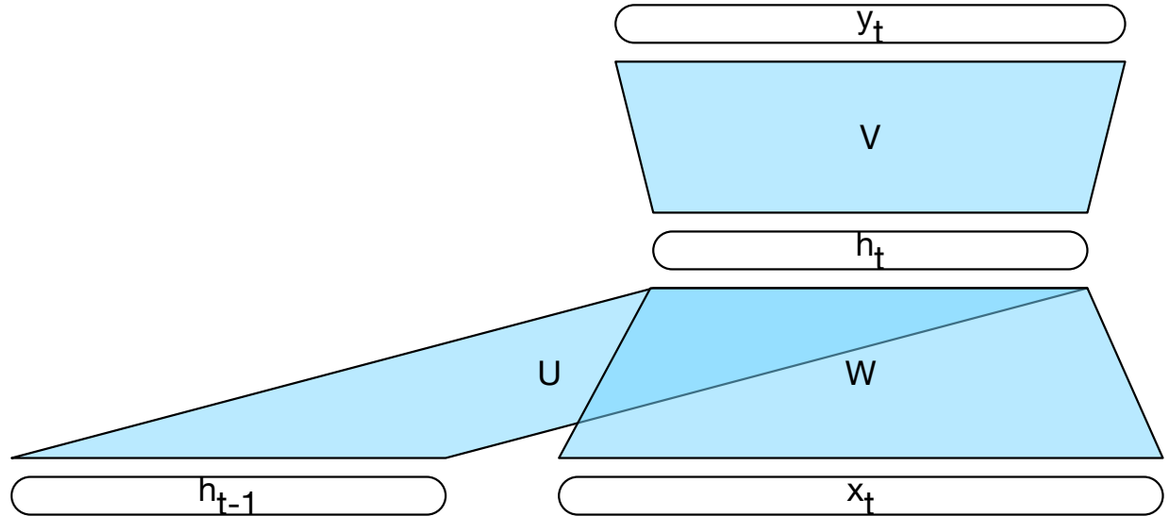
$$h_t = f(Uh_{t-1} + Wx_t)$$
$$y_t = g(Vh_t)$$



Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

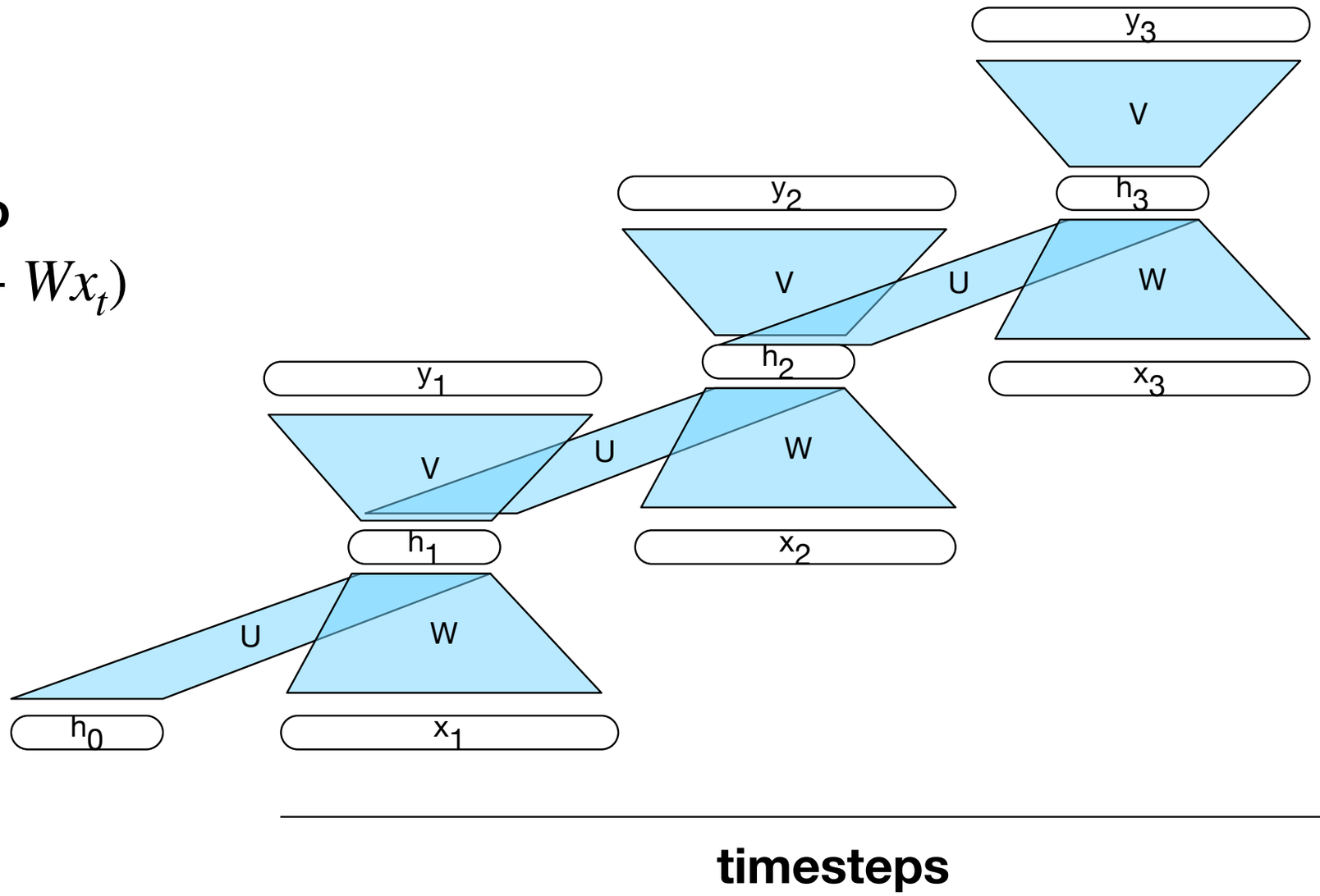
$x \leftarrow (x_1, \ldots, x_n)$
$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$h_t = f(Uh_{t-1} + Wx_t)$
$y_t = g(Vh_t)$
return y



Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

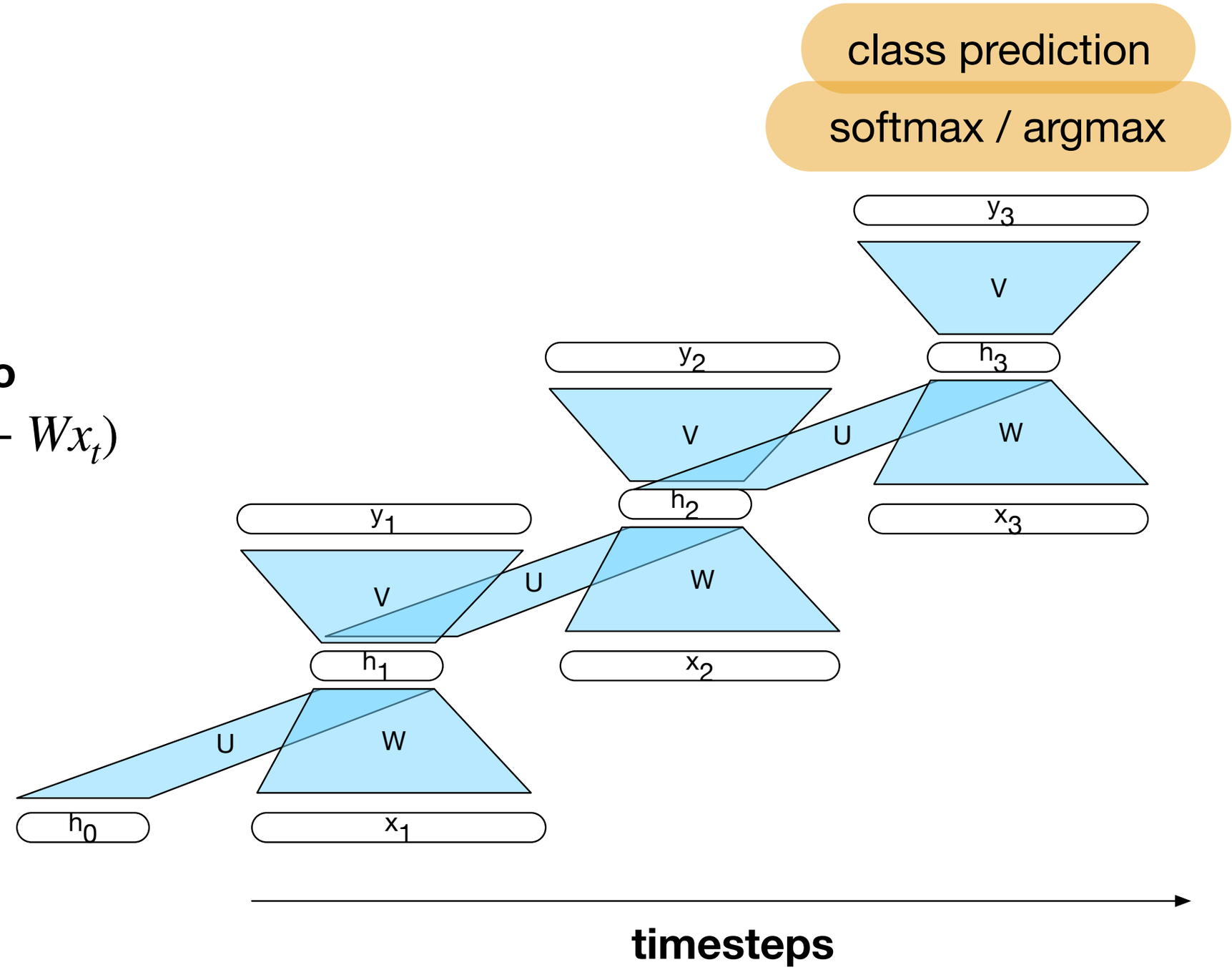
$x \leftarrow (x_1, \ldots, x_n)$
$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$h_t = f(Uh_{t-1} + Wx_t)$
$y_t = g(Vh_t)$
return y



Simple recurrent neural networks

Elman, J. L. (1990). Finding structure in time. *Cognitive science.*

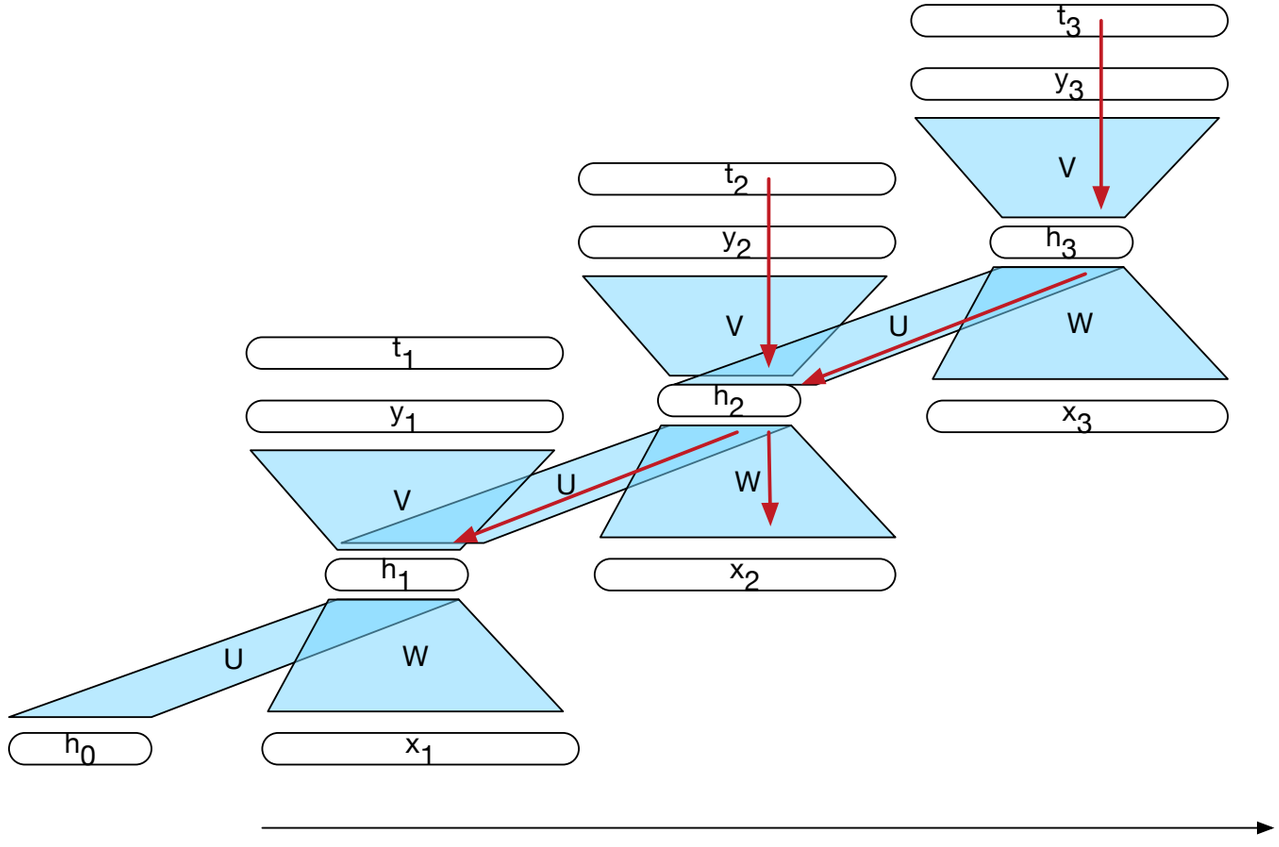
$x \leftarrow (x_1, \ldots, x_n)$
$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$h_t = f(Uh_{t-1} + Wx_t)$
$y_t = g(Vh_t)$
return y



Simple RNN: vanishing gradient

Simple RNNs are hard to train because of the vanishing gradient problem:

- during backpropagation, gradients can quickly become small
- as they repeatedly go through multiplications and non-linear functions (e.g. *sigmoid* or *tanh*)



LSTM: long short-term memory

Hochreiter and Schmidhuber (1997). Long short-term memory. *Neural Computation*.

- Designed to maintain relevant context over time
  - ➡ **learn to forget** information that is no longer needed
  - ➡ **learn to remember** information required for future decisions

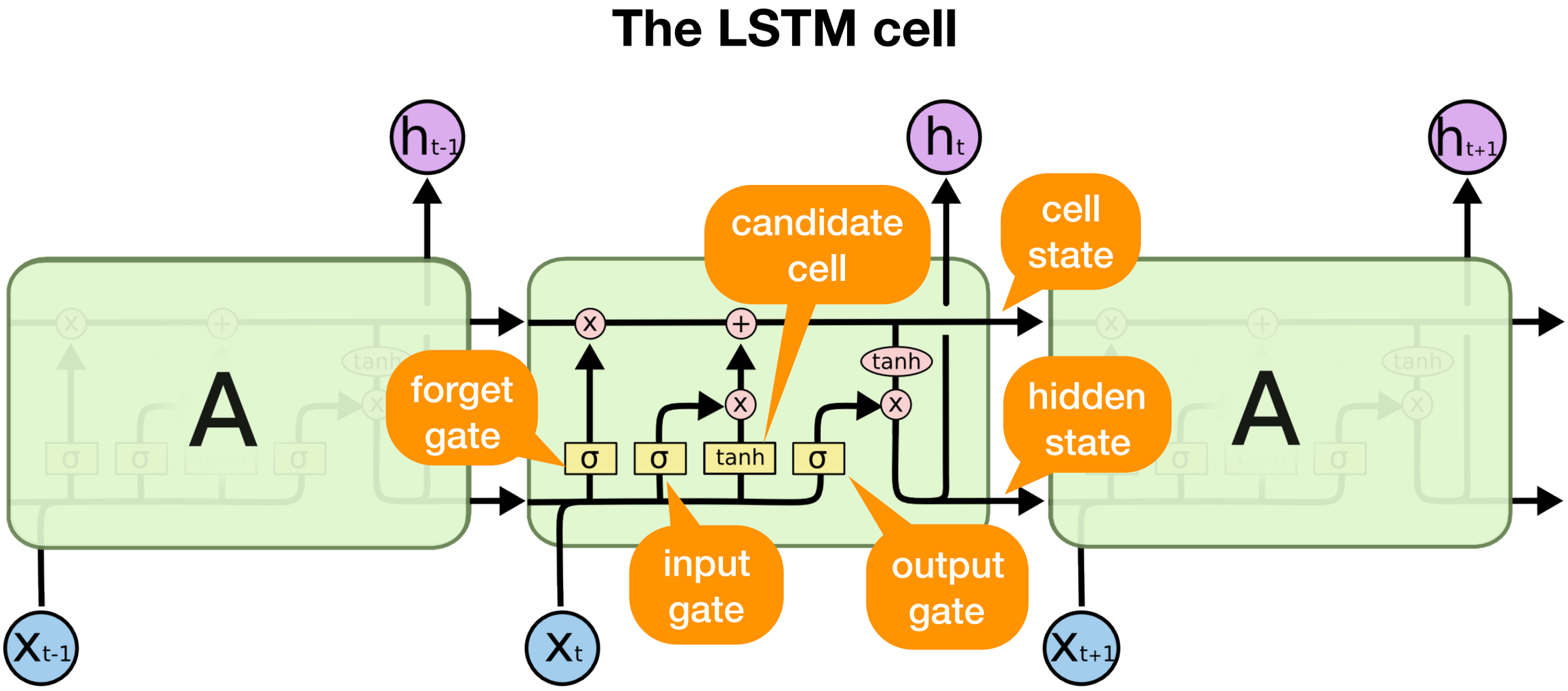
- Deals well with **long-term dependencies** in the input sequence

- With respect to a Simple RNN
  - ➡ add an explicit context layer (*cell state*) to the architecture
  - ➡ make use of **gates** to control the flow of information

- Consists of a **single cell**, the “LSTM cell”, that is repeatedly applied to the input elements

LSTM: long short-term memory

Hochreiter and Schmidhuber (1997). Long short-term memory. *Neural Computation*.

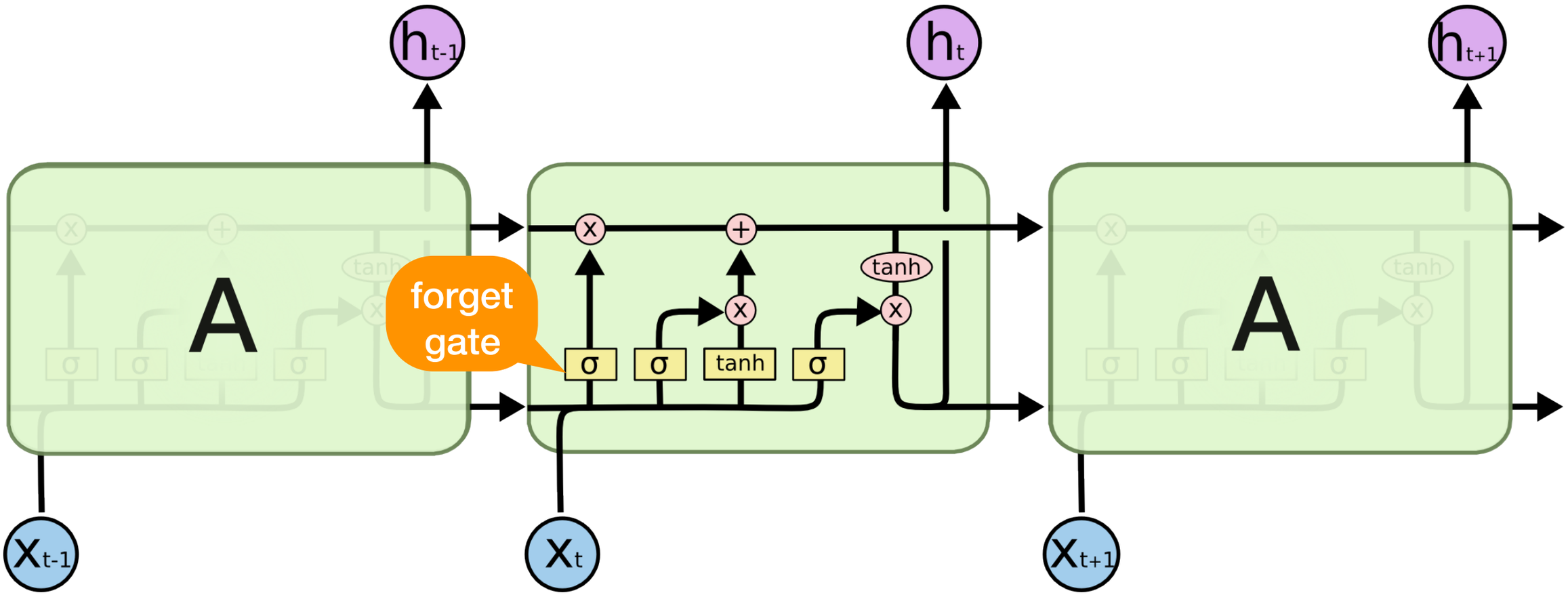


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: forget gate

Delete information from the cell state that is no longer needed.

$$f_t = \sigma(U^f h_{t-1} + W^f x_t)$$

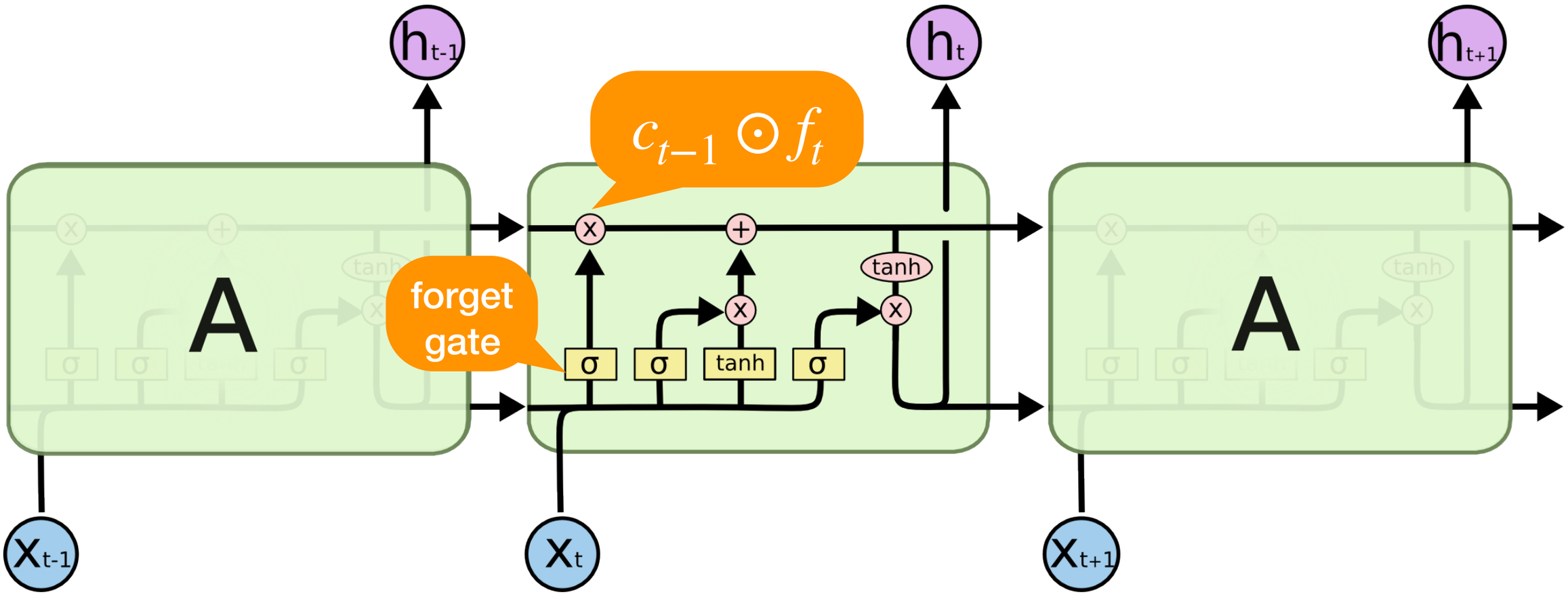


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: forget gate

Delete information from the cell state that is no longer needed.

$$f_t = \sigma(U^f h_{t-1} + W^f x_t)$$

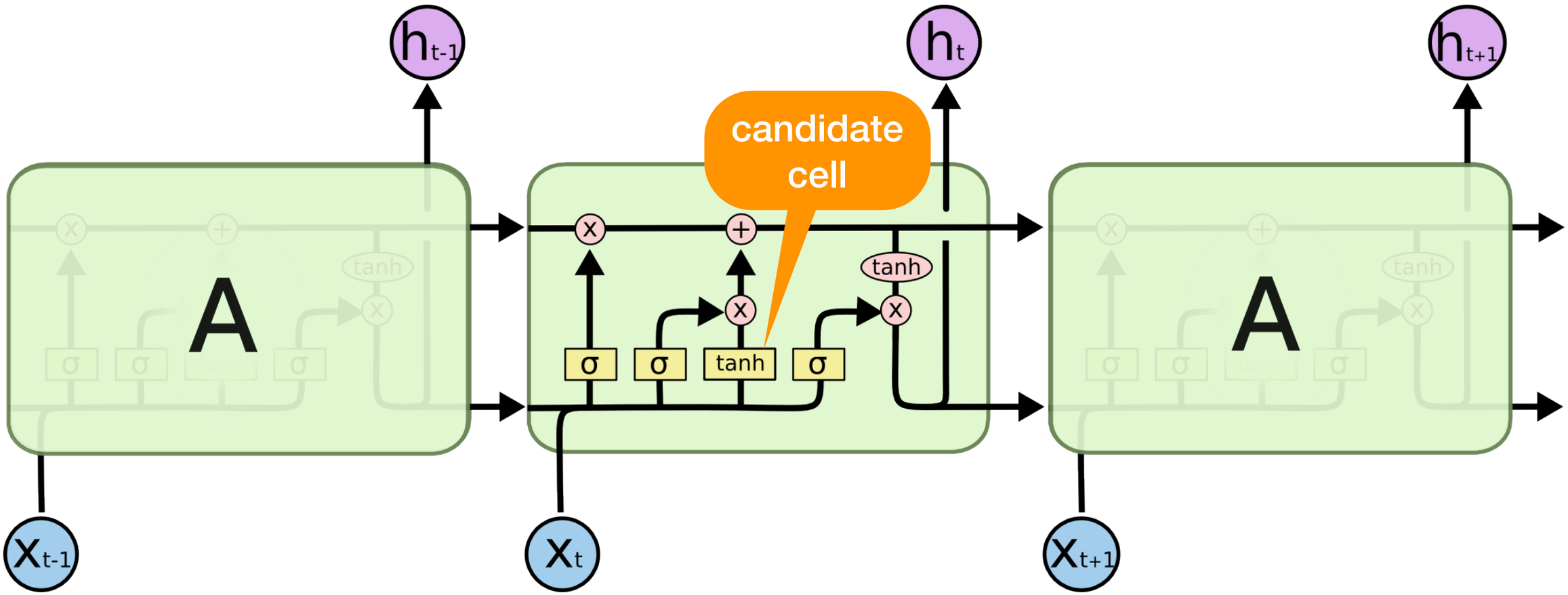


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: candidate cell

Extract information from the previous hidden state and the current input.

$$\tilde{c}_t = \tanh(U^c h_{t-1} + W^c x_t)$$

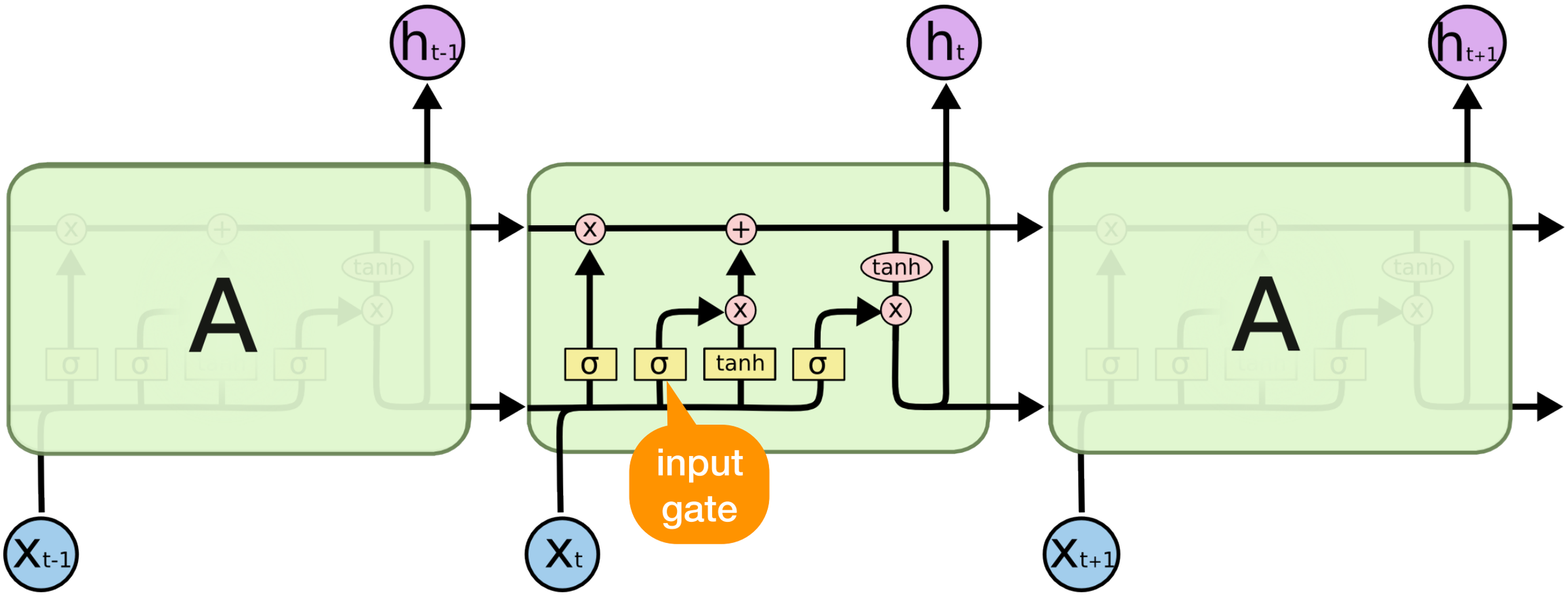


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: input gate

Select the information to add to the new cell state.

$$i_t = \sigma(U^i h_{t-1} + W^i x_t)$$

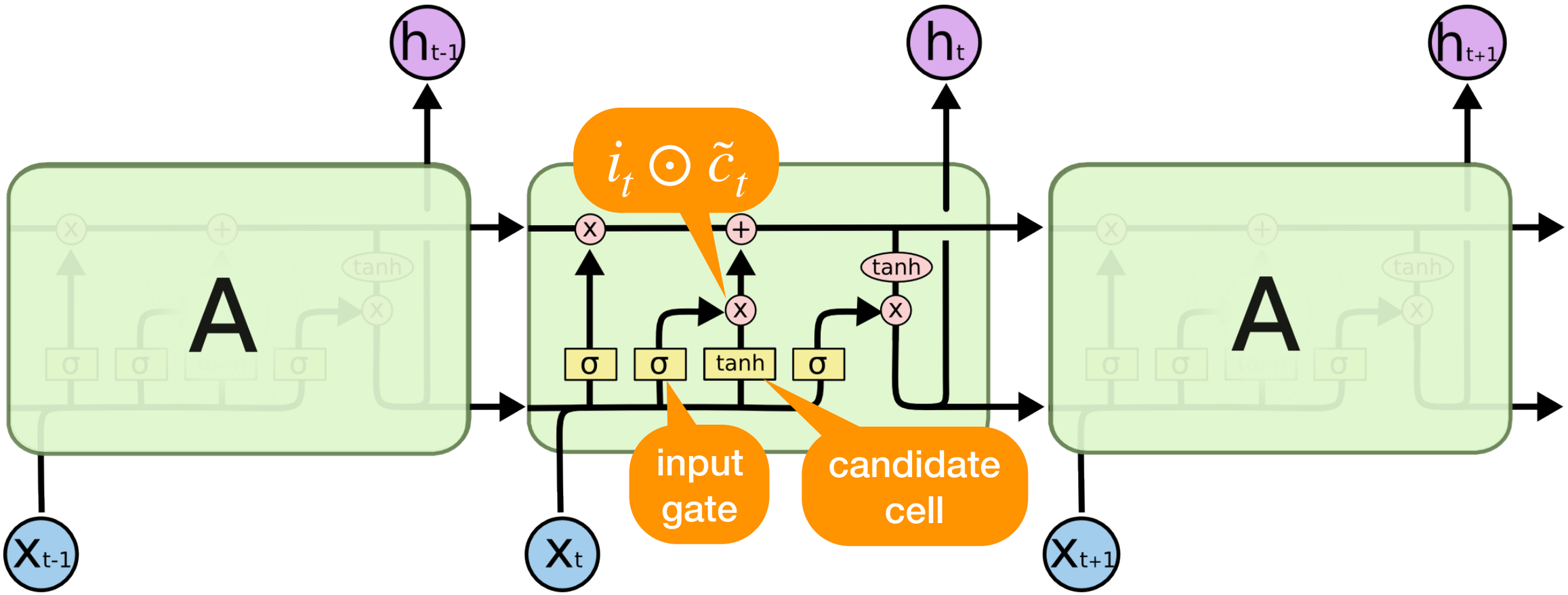


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: input gate

Select the information to add to the new cell state.

$$i_t = \sigma(U^i h_{t-1} + W^i x_t)$$

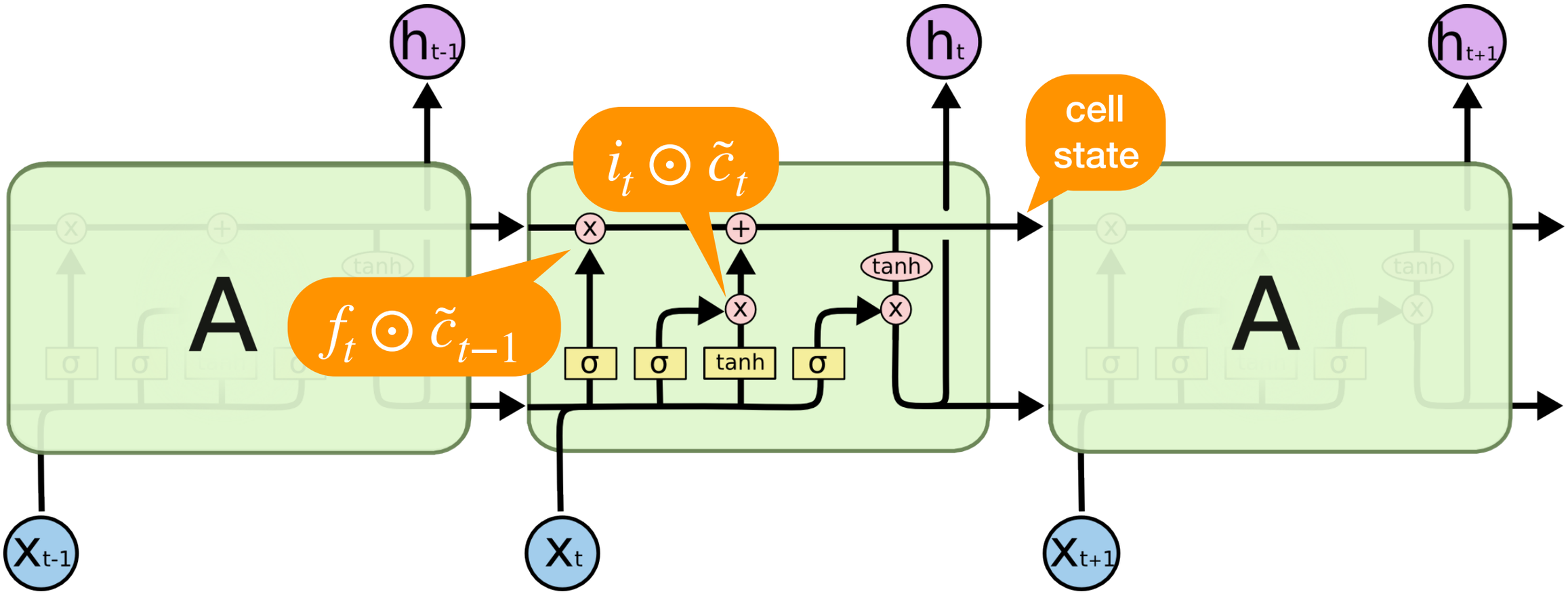


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: cell state

Update the cell state.

$$c_t = f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c}_t$$

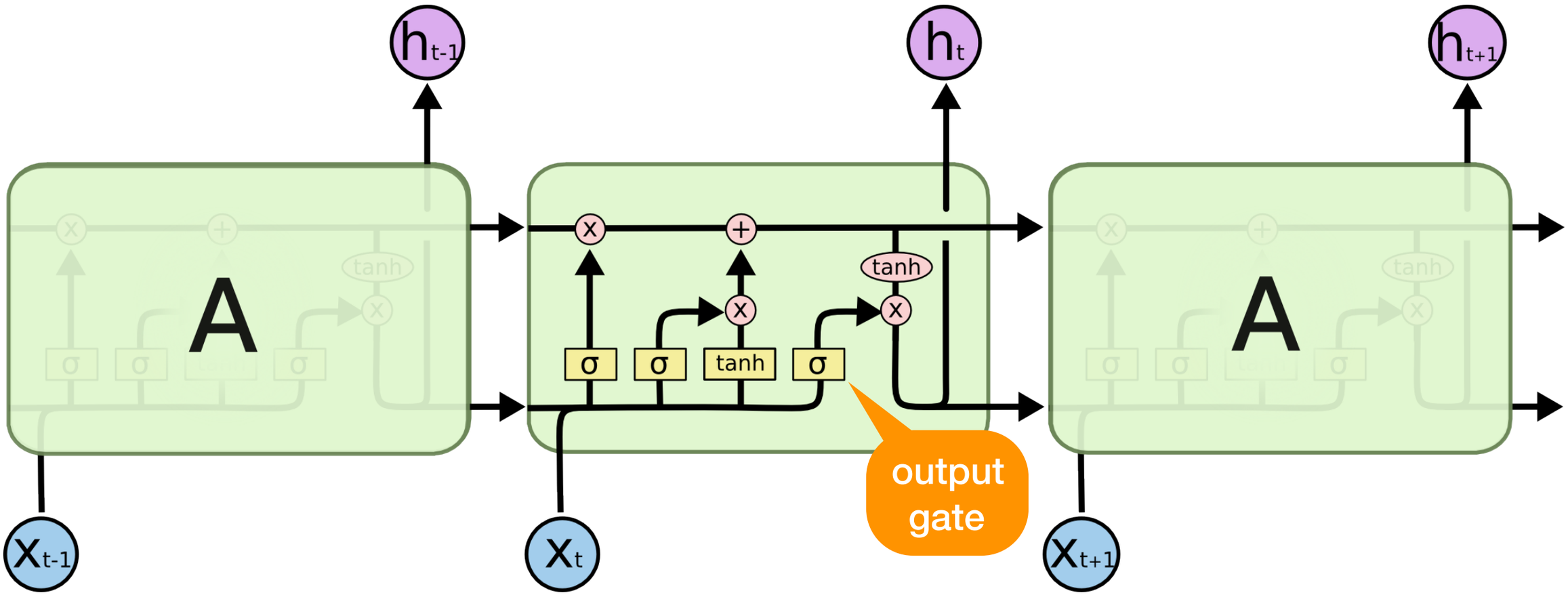


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: output gate

Select what information is required for the current hidden state (as opposed to what information needs to be preserved for future decisions).

$$o_t = \sigma(U^o h_{t-1} + W^o x_t)$$

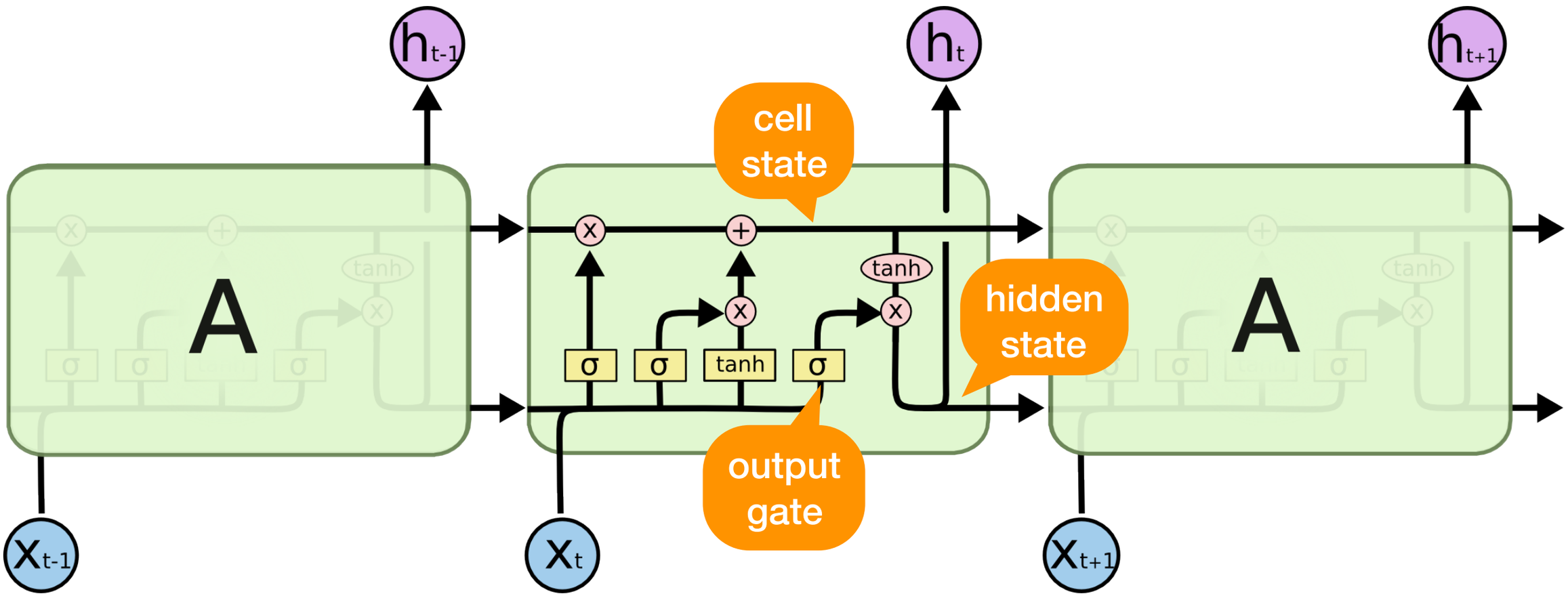


Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: hidden state

Update the hidden state.

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



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

LSTM: all equations

$$f_t = \sigma(U^f h_{t-1} + W^f x_t)$$

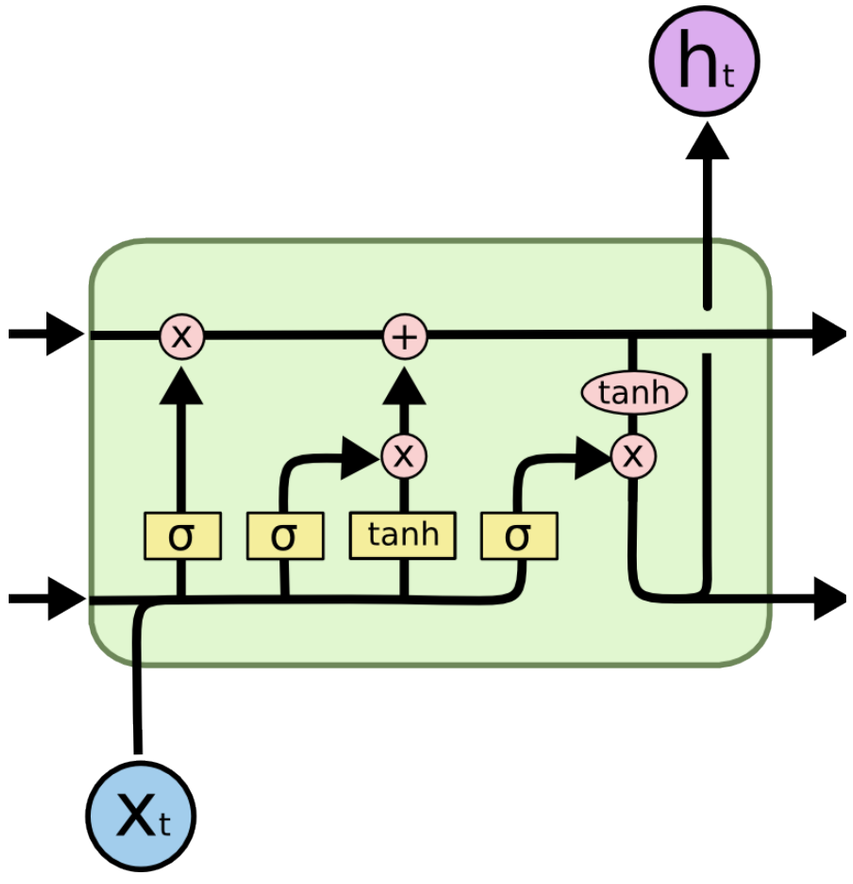
$$i_t = \sigma(U^i h_{t-1} + W^i x_t)$$

$$o_t = \sigma(U^o h_{t-1} + W^o x_t)$$

$$\tilde{c}_t = \tanh(U^c h_{t-1} + W^c x_t)$$

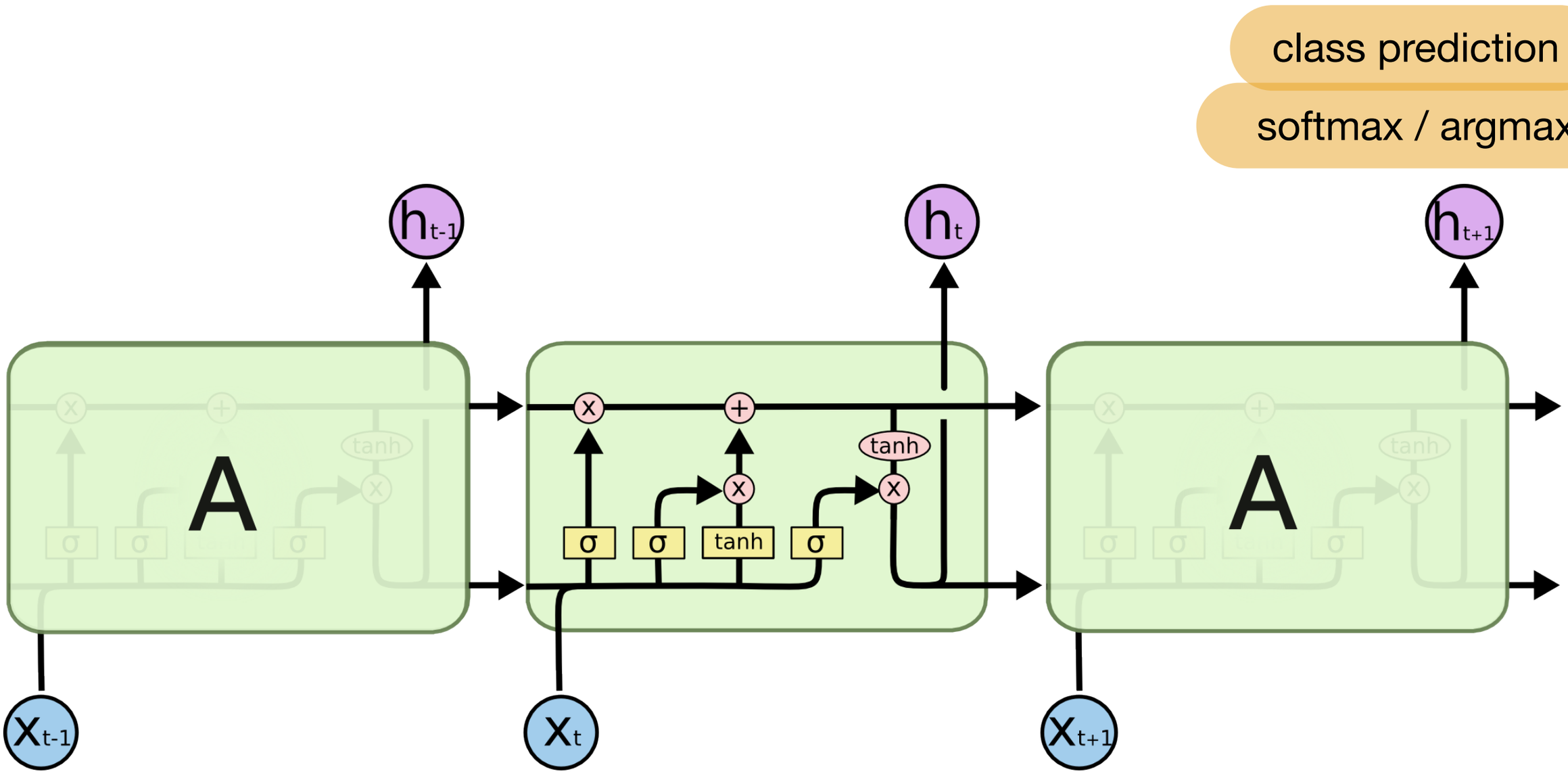
$$c_t = f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c}_t$$

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



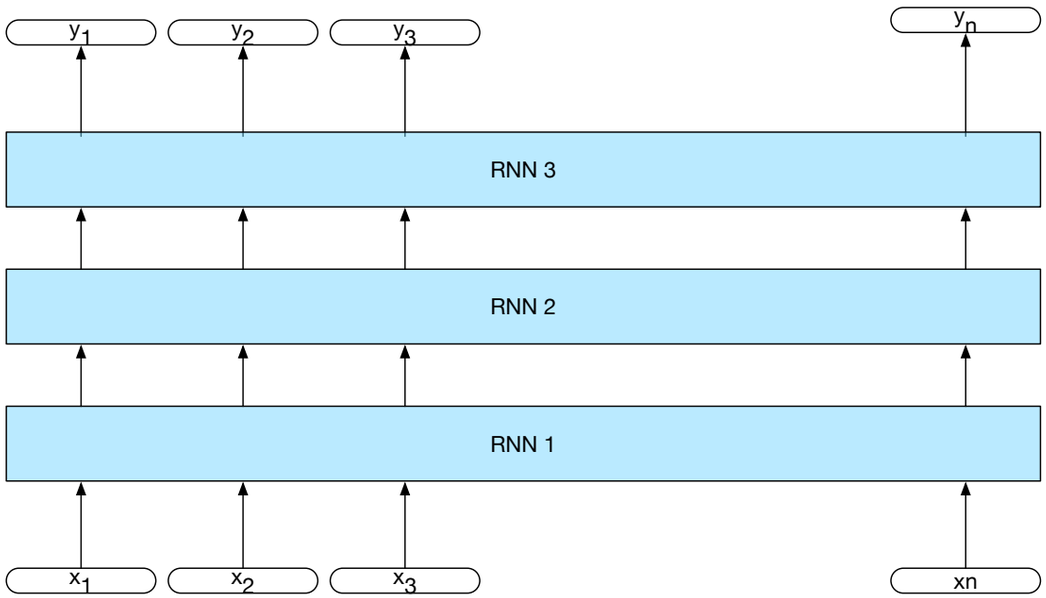
Credits: http://colah.github.io/posts/2015-08-Understanding-LSTMs

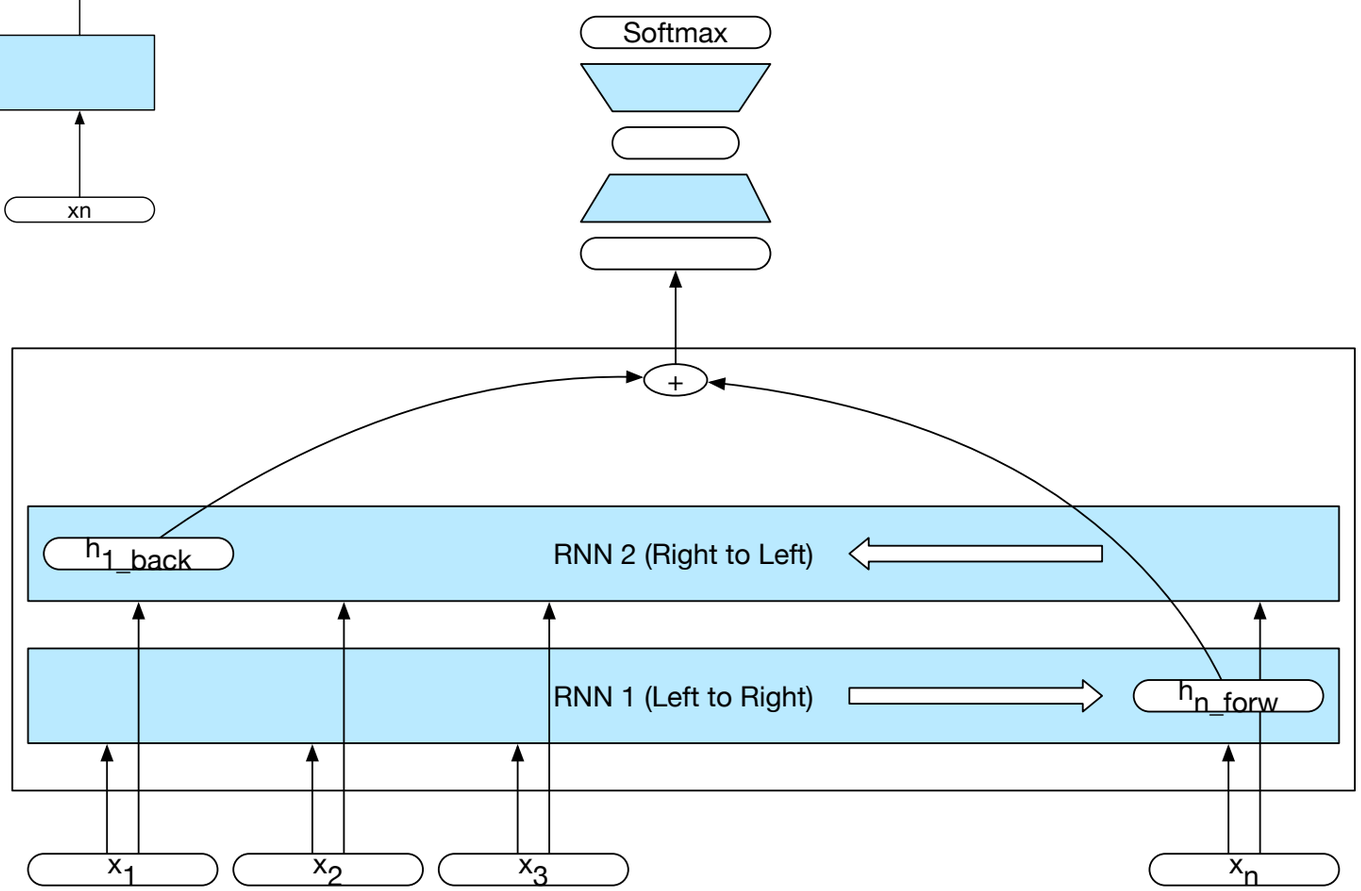
LSTM: predictions



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

LSTM: stacked and bidirectional





LSTM: applications

- **Language modelling** (Mikolov et al., 2010; Sundermeyer et al., 2012)
- **Sequence labelling** (e.g., POS tagging, Named Entity Recognition)
- **Parsing** (Vinyals et al., 2015; Kiperwasser and Goldberg, 2016; Dyer et al., 2016)
- **Machine translation** (Bahdanau et al., 2015)
- Auto-regressive **generation** (e.g., image captioning (Bernardi et al., 2016))
- **Sequence classification** (e.g., sentiment analysis, fake news detection)
- …

Sentence representations with NNs

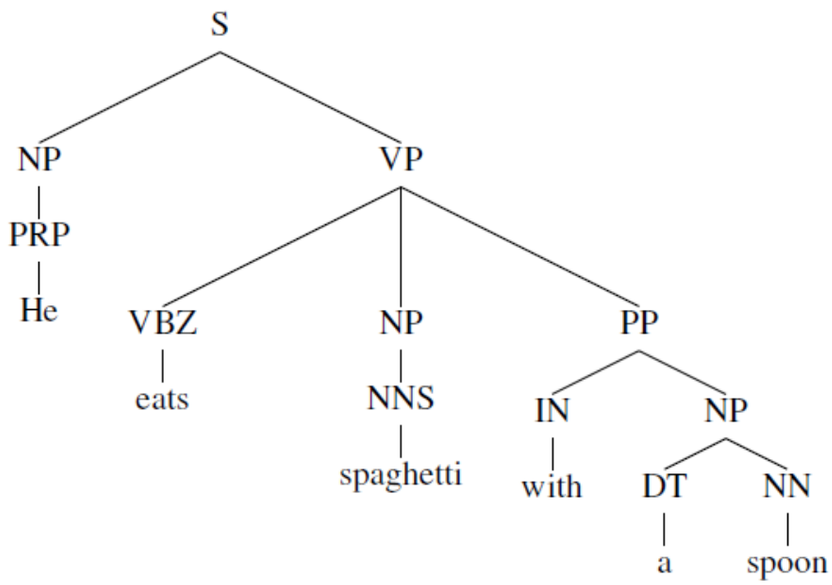
- **Bag of Words models**
  - ➡ sentence representations are **order-independent** function of the word representations

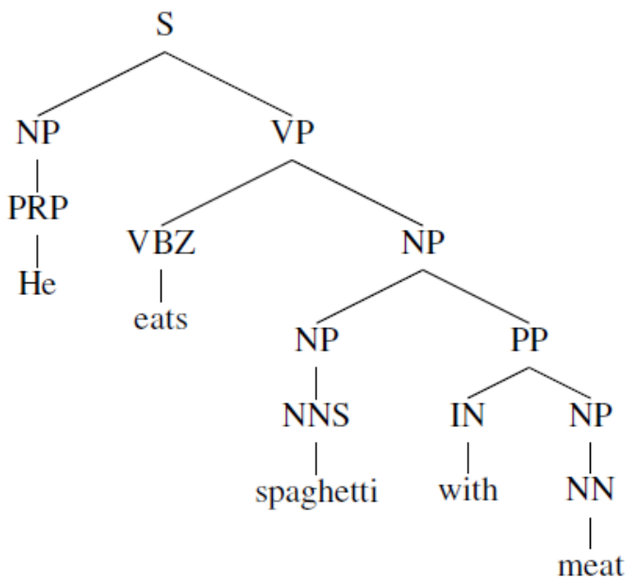
- **Sequence models**
  - ➡ sentence representations are an **order-sensitive** function of a sequence of word representations (surface form)
  - ➡ can they still capture the underlying syntactic structure? (e.g., Linzen et al., 2016, Giulianelli et al., 2018)

- **Tree-structured models**
  - ➡ sentence representations are a function of the word representations, **sensitive to the syntactic structure** of the sentence

Tree-structured models

- More faithful operationalisation of the principle of compositionality
- Helpful in disambiguation: similar surface form, different underlying structure
- Requires syntactic parse trees (constituency or dependency)





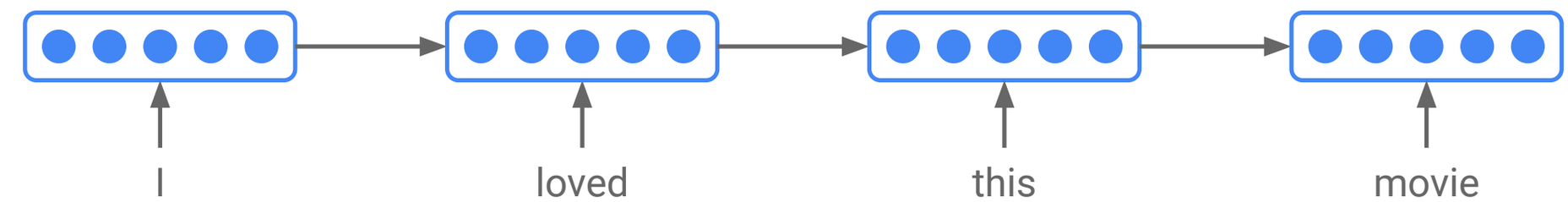
Credits: Richard Socher (Stanford CS224d 2016, Lecture 9)

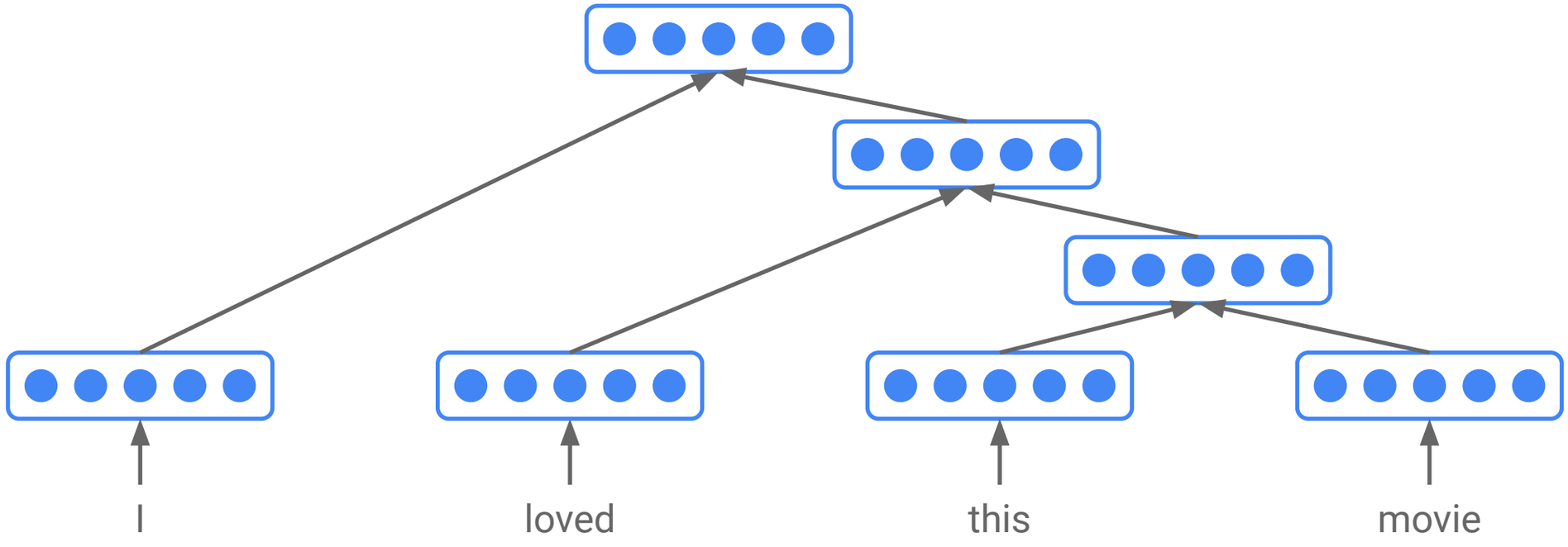
Tree-LSTM

A generalisation of the LSTM to tree-structured input.

- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *ACL 2015*.
  - ➡ Child-Sum Tree-LSTM
  - ➡ N-ary Tree-LSTM
- Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. **SEM 2015*.
- Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. *ICML 2015*.

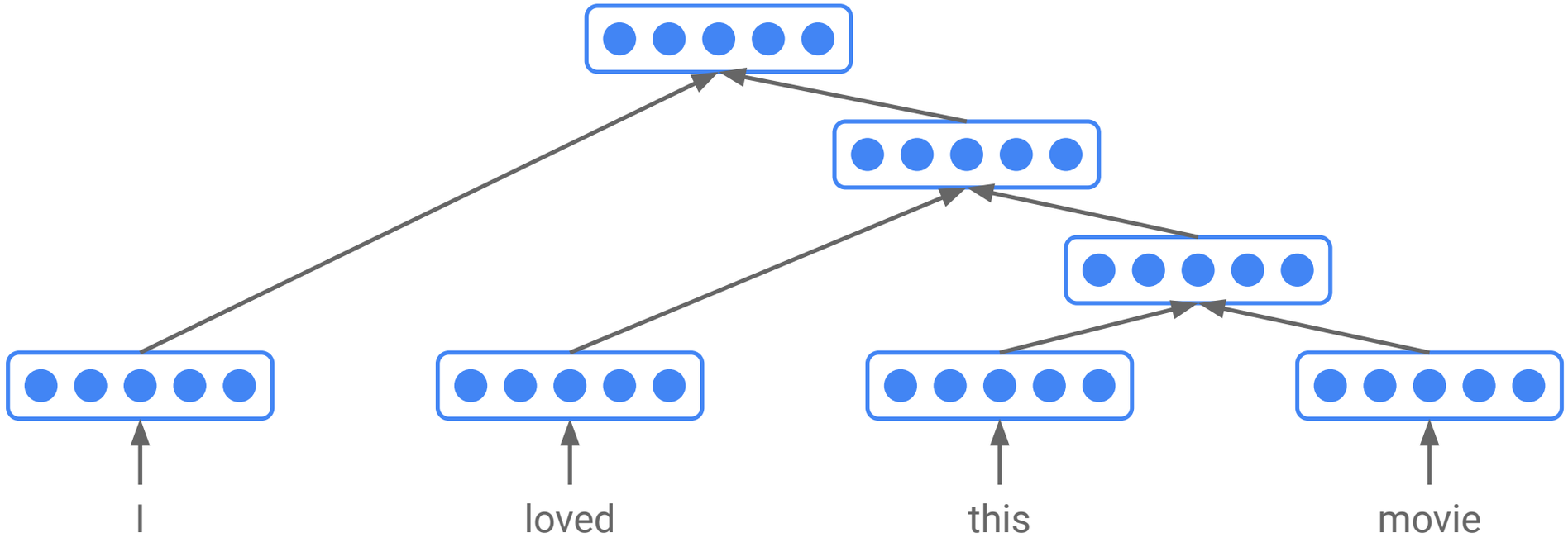
Tree-LSTM





Tree-LSTM

- Gates and memory cell updates are dependent on the states of a node's children, rather than on the states of the previous words.
- Instead of a single forget gate, Tree-LSTM unit contains **one forget gate for each child** to selectively incorporate information from each child.



Tree-LSTM variants

- **Child-Sum Tree-LSTM**
  - sum over the hidden representations of all children of a node (**no children order**)
  - can be used for a **variable** number of children
  - **shares parameters** between children
  - suitable for dependency trees

- **N-ary Tree-LSTM**
  - discriminates between **children node positions** (weighted sum)
  - **fixed** maximum branching factor: can be used with N children at most
  - **different parameters** for each child
  - suitable for constituency trees

Child-Sum Tree-LSTM

$$\tilde{h}_j = \sum_{k \in C(j)} h_k$$

$$f_{jk} = \sigma(U^f h_k + W^f x_j)$$

$$i_j = \sigma(U^i \tilde{h}_j + W^i x_j)$$

$$o_j = \sigma(U^o \tilde{h}_j + W^o x_j)$$

$$\tilde{c}_j = \tanh(U^c \tilde{h}_j + W^c x_j)$$

$$c_j = i_j \odot \tilde{c}_j + \sum_{k \in C(j)} f_{jk} \odot c_k$$

$$h_j = o_j \odot \tanh(c_j)$$

Child-Sum Tree-LSTM

$$\tilde{h}_j = \sum_{k \in C(j)} h_k$$

$$f_{jk} = \sigma(U^f h_k + W^f x_j)$$

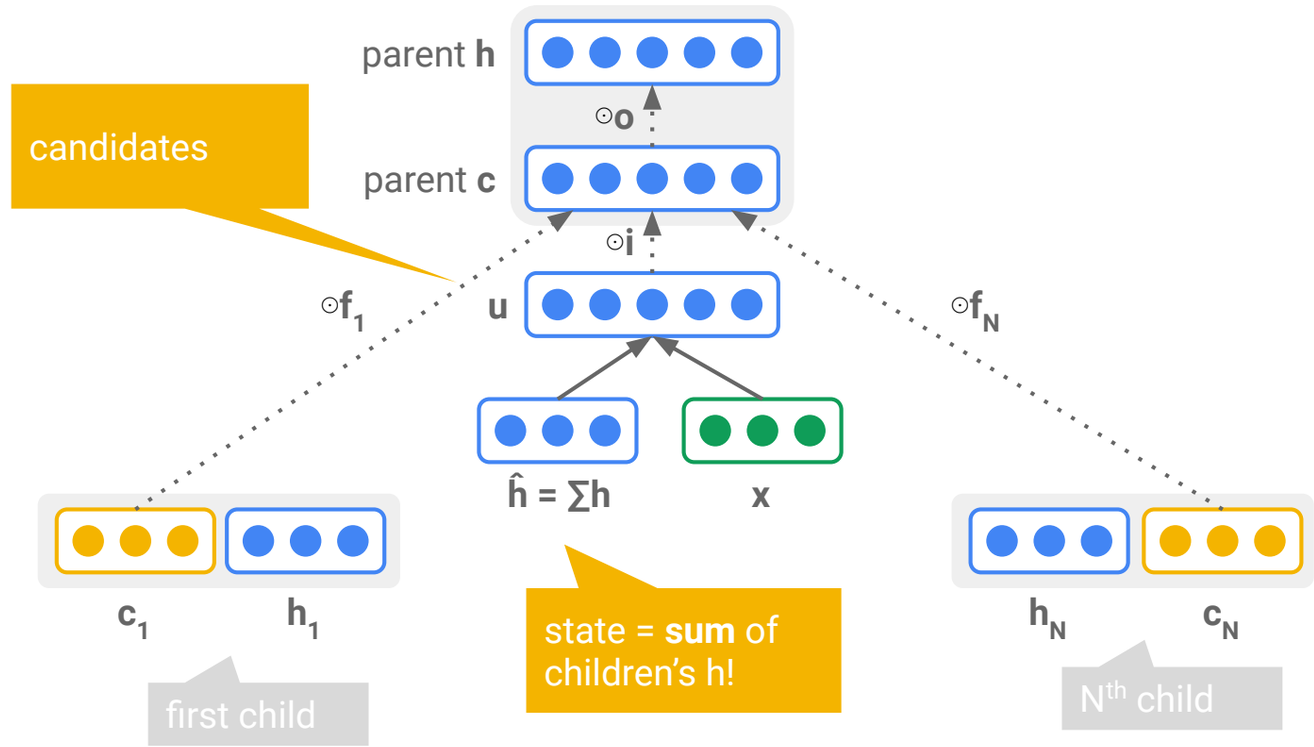
$$i_j = \sigma(U^i \tilde{h}_j + W^i x_j)$$

$$o_j = \sigma(U^o \tilde{h}_j + W^o x_j)$$

$$\tilde{c}_j = \tanh(U^c \tilde{h}_j + W^c x_j)$$

$$c_j = i_j \odot \tilde{c}_j + \sum_{k \in C(j)} f_{jk} \odot c_k$$

$$h_j = o_j \odot \tanh(c_j)$$



8 parameter matrices

N-ary Tree-LSTM

$$f_{jk} = \sigma\left(W^f x_j + \sum_{l=1}^{N} U_{kl}^f \, h_{jl}\right)$$

$$i_j = \sigma\left(W^i x_j + \sum_{l=1}^{N} U_l^i \, h_{jl}\right)$$

$$o_j = \sigma\left(W^o x_j + \sum_{l=1}^{N} U_l^o \, h_{jl}\right)$$

$$\tilde{c}_j = \tanh\left(W^c x_j + \sum_{l=1}^{N} U_l^c \, h_{jl}\right)$$

$$c_j = i_j \odot \tilde{c}_j + \sum_{l=1}^{N} f_{jl} \odot c_l$$

$$h_j = o_j \odot \tanh(c_j)$$

N-ary Tree-LSTM

$$f_{jk} = \sigma\left(W^f x_j + \sum_{l=1}^{N} U_{kl}^f \, h_{jl}\right)$$

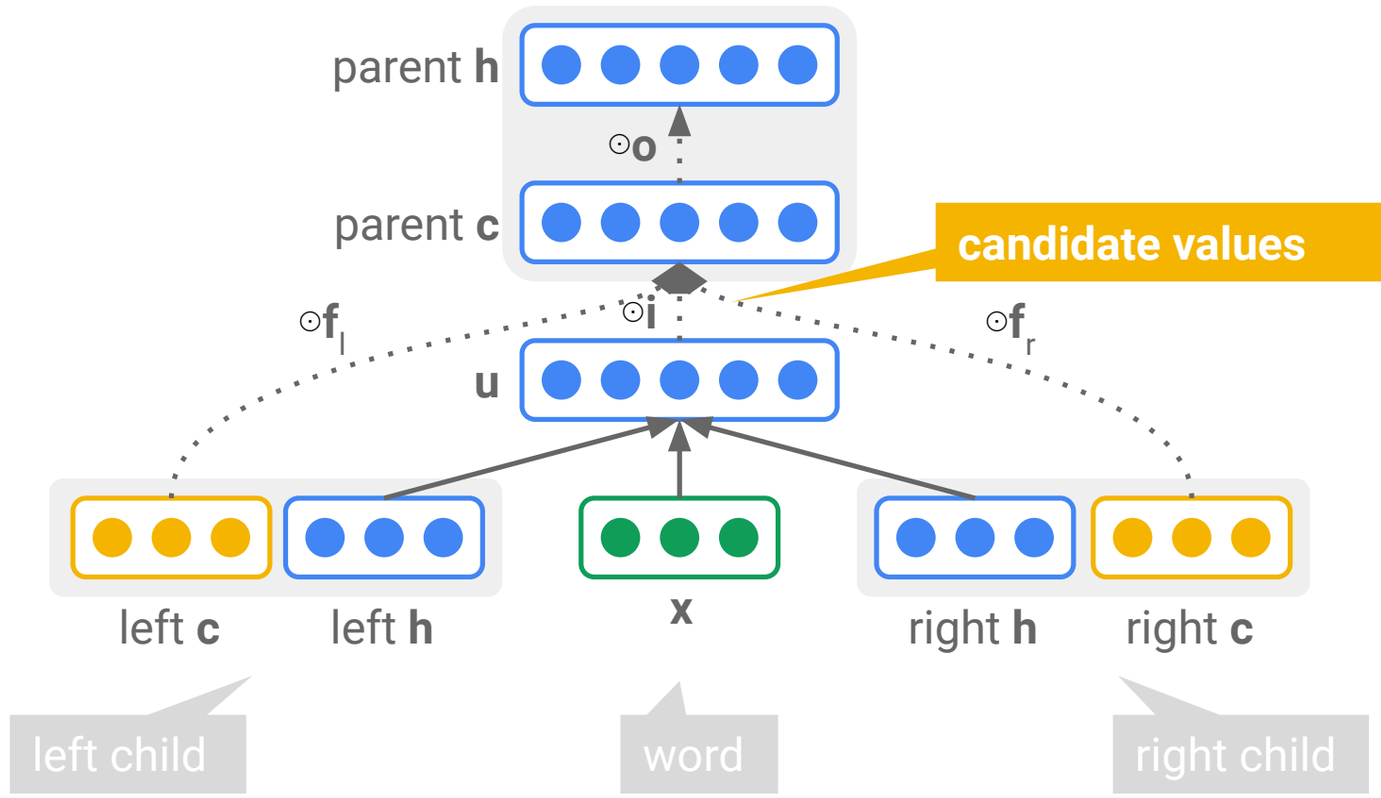
$$i_j = \sigma\left(W^i x_j + \sum_{l=1}^{N} U_l^i \, h_{jl}\right)$$

$$o_j = \sigma\left(W^o x_j + \sum_{l=1}^{N} U_l^o \, h_{jl}\right)$$

$$\tilde{c}_j = \tanh\left(W^c x_j + \sum_{l=1}^{N} U_l^c \, h_{jl}\right)$$

$$c_j = i_j \odot \tilde{c}_j + \sum_{l=1}^{N} f_{jl} \odot c_l$$

$$h_j = o_j \odot \tanh(c_j)$$



$4 + 3N + N^2$ parameter matrices

Tree-LSTM: forget gates

Child-Sum Tree-LSTM

$$f_{j,left} = \sigma(U^f h_{left} + W^f x_j)$$

$$f_{j,right} = \sigma(U^f h_{right} + W^f x_j)$$

1 parameter matrix U^f: symmetric treatment of children

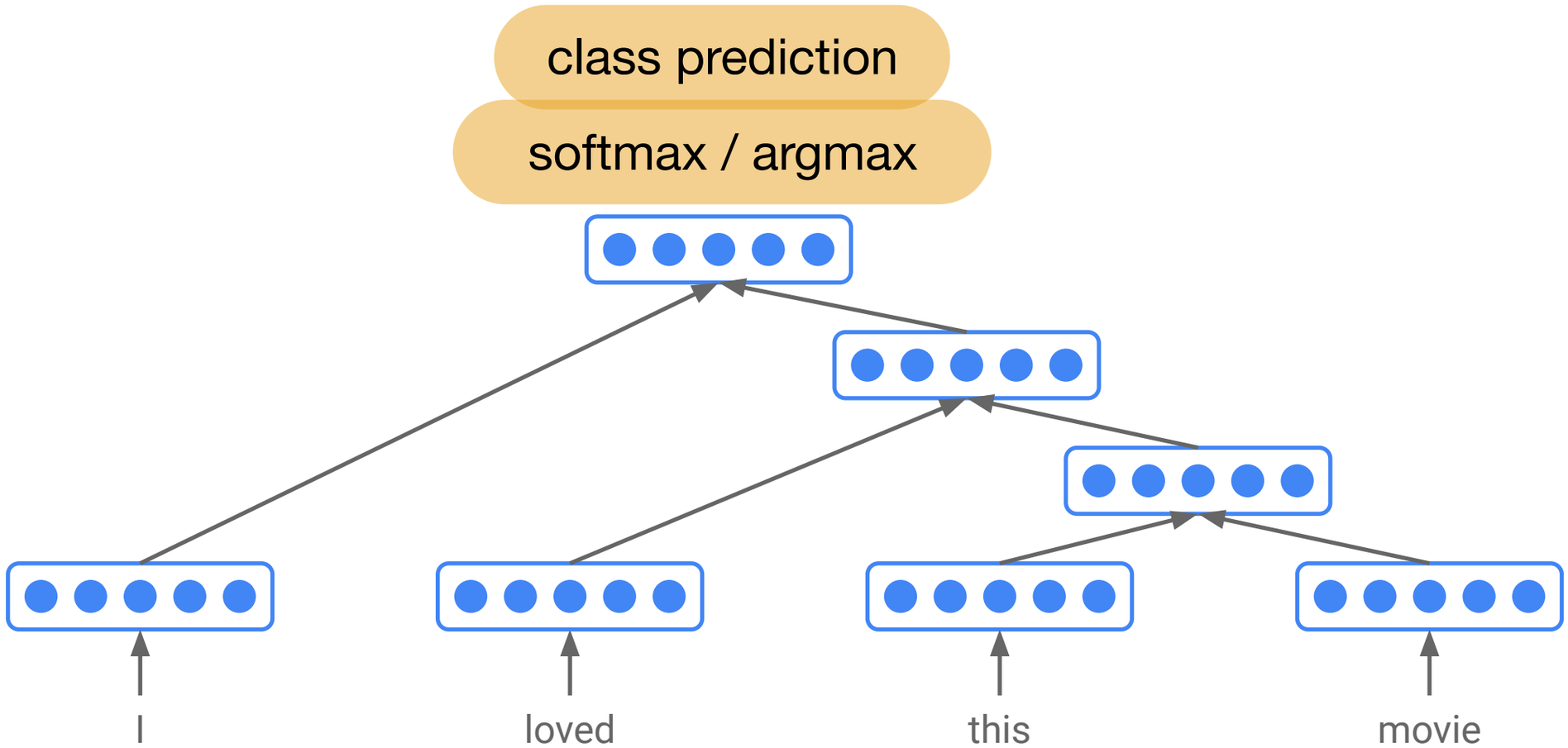
Binary Tree-LSTM

$$f_{j,left} = \sigma\left(W^f x_j + U^f_{left,left}\ h_{j,left} + U^f_{left,right}\ h_{j,right}\right)$$

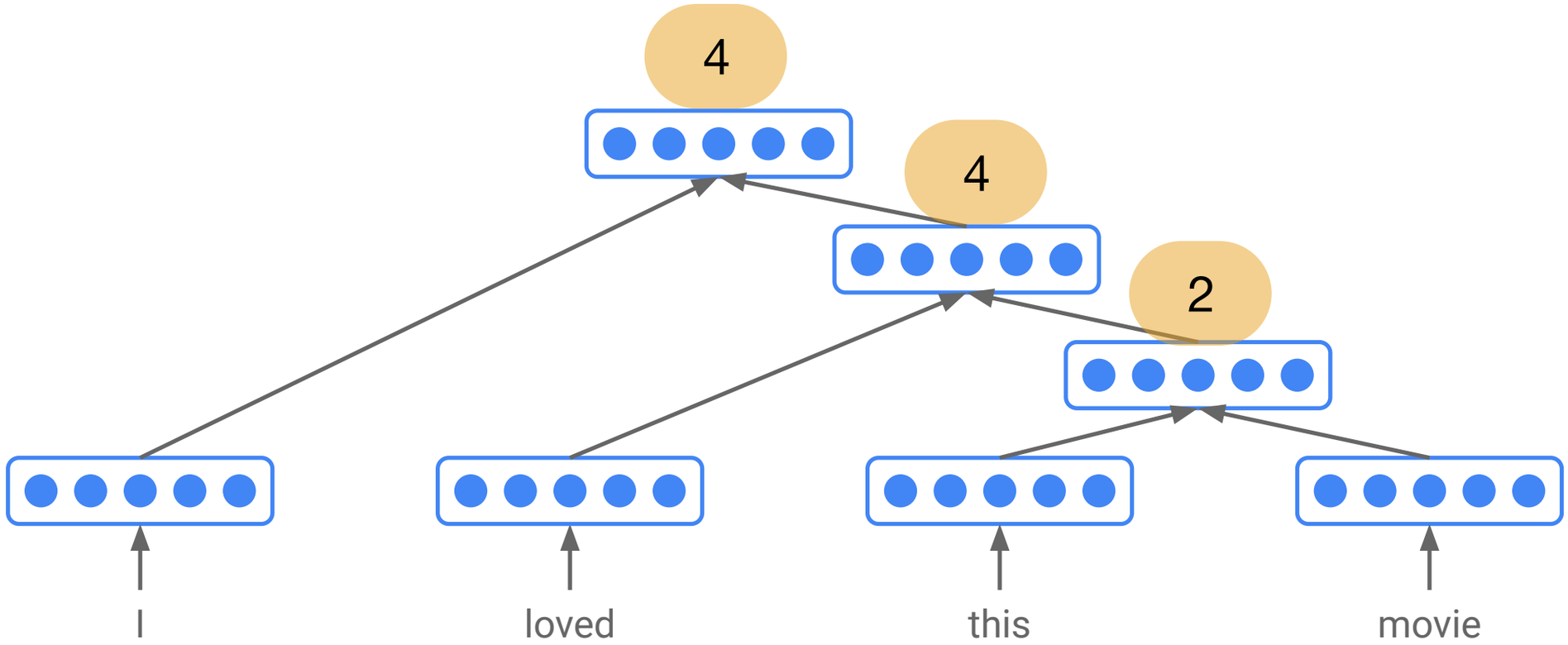
$$f_{j,right} = \sigma\left(W^f x_j + U^f_{right,left}\ h_{j,left} + U^f_{right,right}\ h_{j,right}\right)$$

4 parameter matrices $U^f_{x,y}$: asymmetric treatment of children

Binary Tree-LSTM: sentiment classification



Binary Tree-LSTM: sentiment classification



Compositional semantics with NNs

Models

1. Bag of Words (BOW)
2. Continuous Bag of Words (CBOW)
3. Deep Continuous Bag of Words (Deep CBOW)
4. Deep CBOW with pre-trained word embeddings
5. LSTM
6. Tree-LSTM