Recurrent Neural Networks and Language Models

Ivan Nasonov
Telegram: @naconov
GitHub: NasonovIvan

22 November, 2023

Tokens and RNNs generating

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccc} S & \longrightarrow & \\ \downarrow & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} \\ & & \uparrow \\ & & \\ \text{gor}_s & & \end{array}$$

$$\begin{array}{ccc} & = \alpha' & \longrightarrow \\ & \updownarrow & \\ = \alpha' & \longrightarrow & \alpha \end{array} \qquad \begin{array}{c} X \\ \downarrow \\ \text{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G}) \end{array}$$

$$\text{Spec}(K_\psi) \qquad \text{Mor}_{Sets}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type $f_\bullet$. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \text{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{s}})$$

is an isomorphism of covering of $\mathcal{O}_{X_t}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Source - The Unreasonable Effectiveness of Recurrent Neural Networks

Contex and Bag-of-Words

Вчера на матче ЦСКА-Спартак был забит красивый ...
(гол/барабан/формула?)

Document D1	*The child makes the dog happy* the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	*The dog makes the child happy* the: 2, child: 1, makes: 1, dog: 1, happy: 1

	child	dog	happy	makes	the	BoW Vector representations
D1	1	1	1	1	2	[1,1,1,1,2]
D2	1	1	1	1	2	[1,1,1,1,2]

RNN intuitions

- Initial step: $h_0 = 0$
- i-step:

$$h_{i-1} + x_i \longrightarrow \blacksquare \longrightarrow h_i$$

Size of h_{i-1} = size of h_i
Shapes:

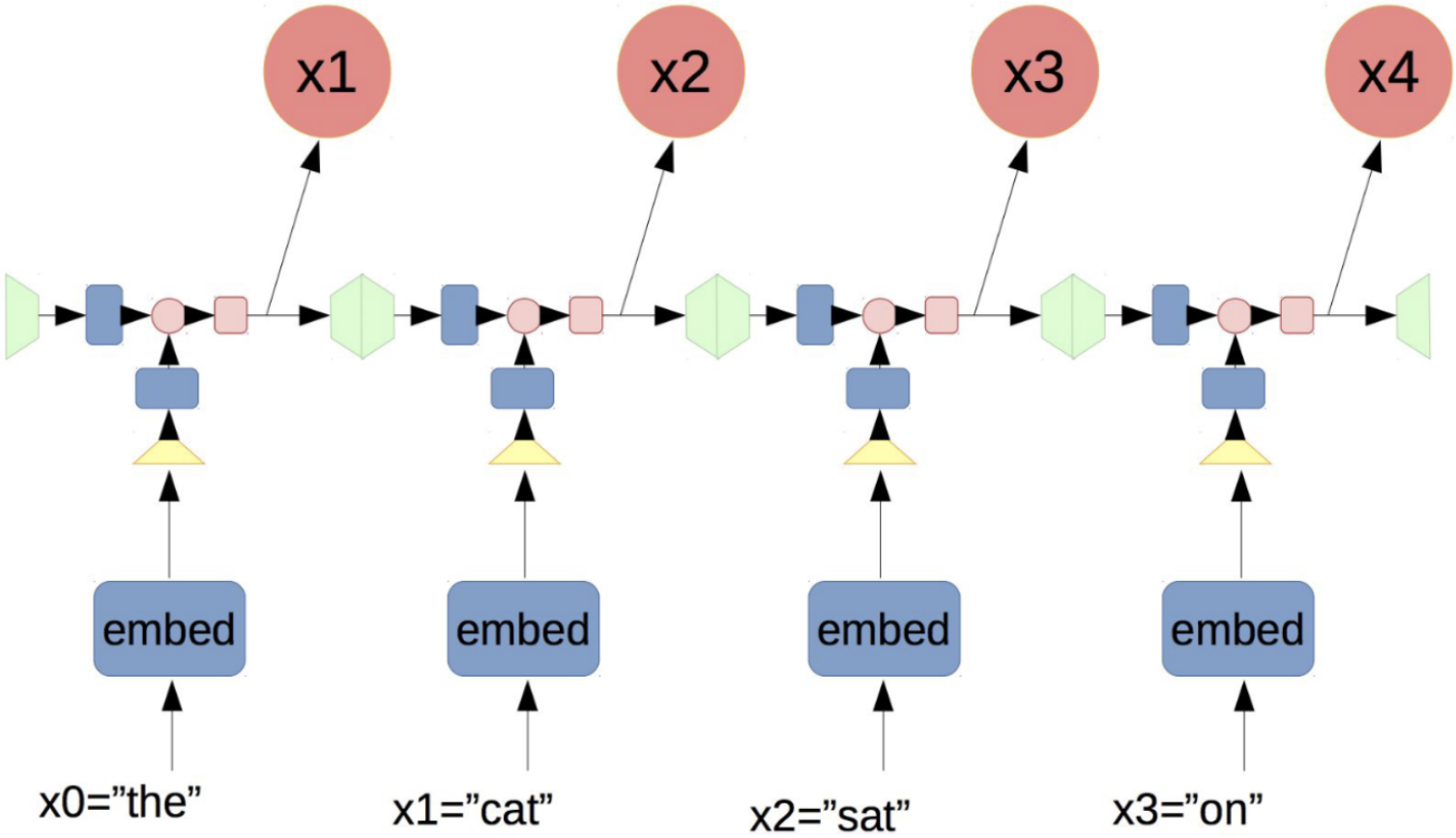
- $x_i = [l]$
- $h_{i-1} = [k]$
- $W = [k + l \times k]$

Simple RNN:

- Context Vector $h_i = \sigma\big(W \times [h_{i-1}, x_i] + b\big)$

Source - Girafe.AI

Simple RNN Example



Source - Girafe.AI

RNN with formulas

$$h_0 = \bar{0}$$

$$h_1 = \sigma\big(\langle W_{\text{hid}}[h_0, x_0]\rangle + b\big)$$

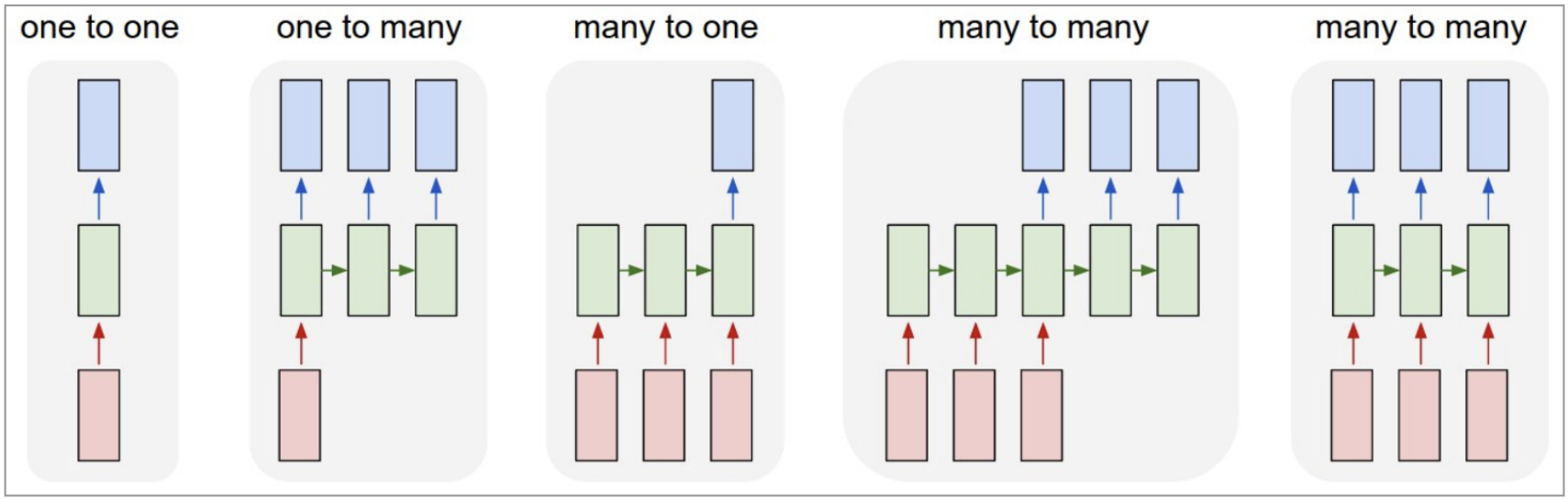
$$h_2 = \sigma\big(\langle W_{\text{hid}}[h_1, x_1]\rangle + b\big) = \sigma\big(\langle W_{\text{hid}}[\sigma\big(\langle W_{\text{hid}}[h_0, x_0]\rangle + b, x_1]\rangle + b\big)$$

$$h_{i+1} = \sigma\big(\langle W_{\text{hid}}[h_i, x_i]\rangle + b\big)$$

$$P(x_{i+1}) = \text{softmax}\big(\langle W_{\text{out}}, h_i\rangle + b_{\text{out}}\big)$$

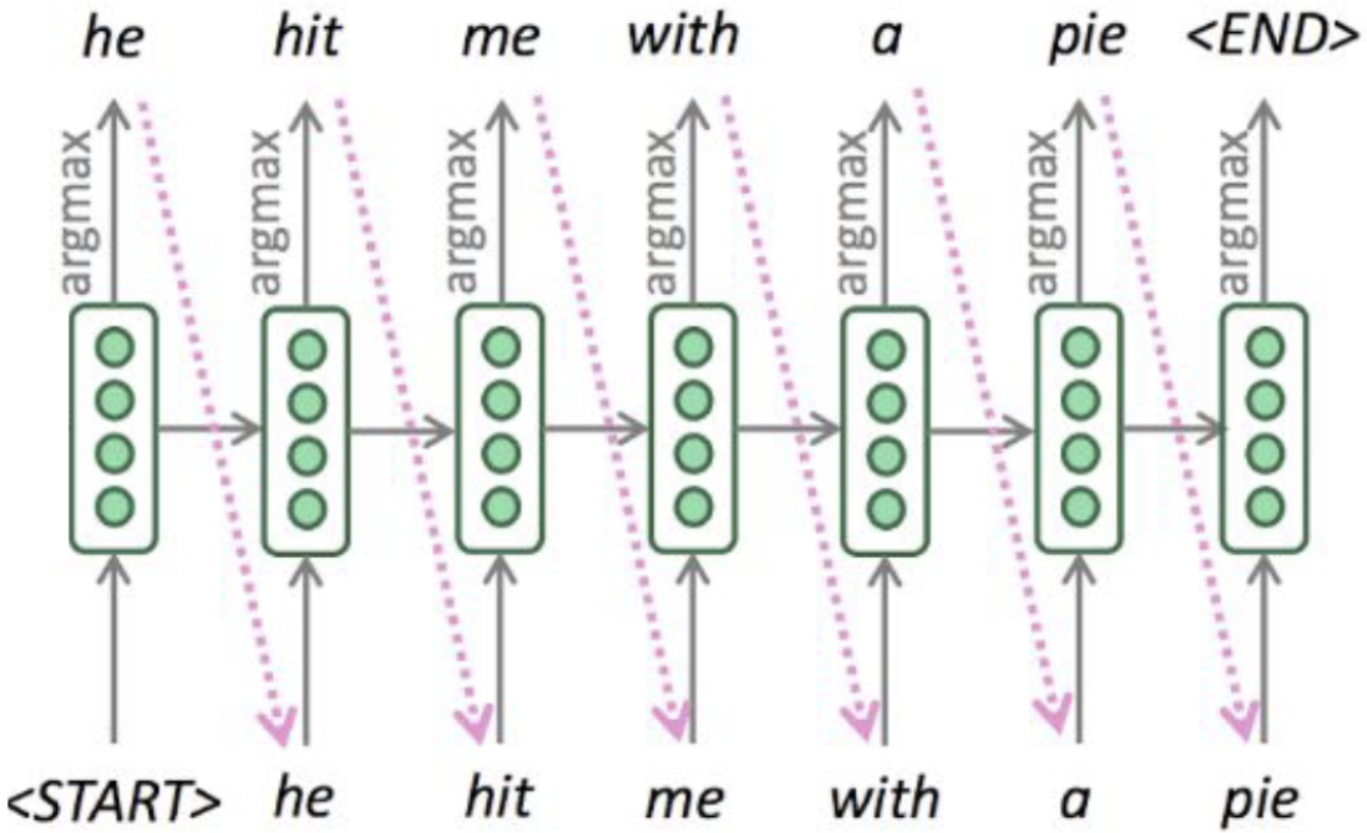
Source - Girafe.AI

RNN's tasks



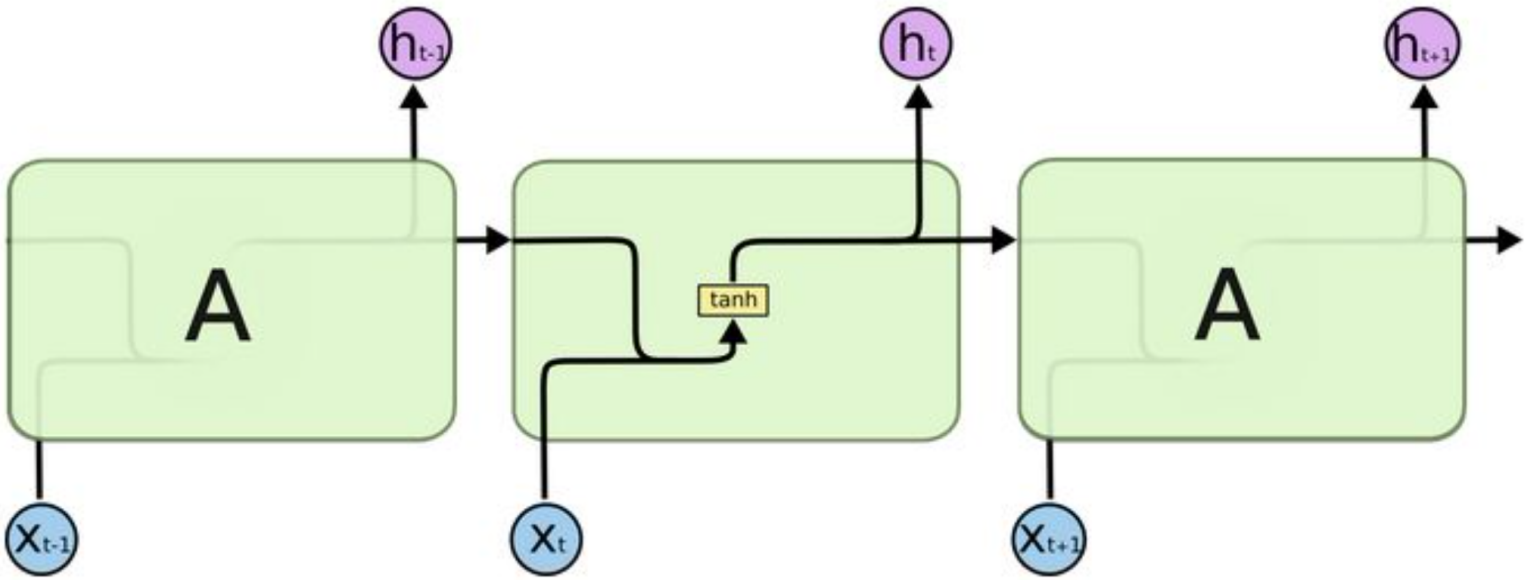
Source - The Unreasonable Effectiveness of Recurrent Neural Networks

RNN text generation



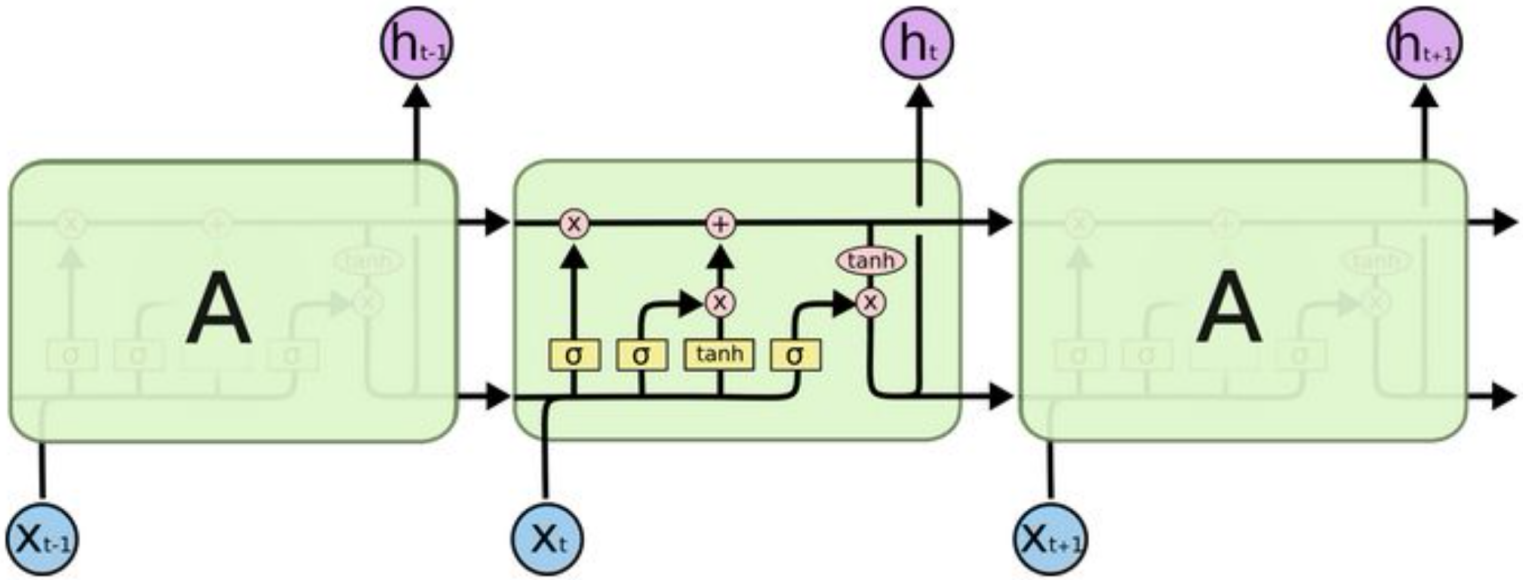
Source - Girafe.AI

Vanilla RNN



Source - Girafe.AI

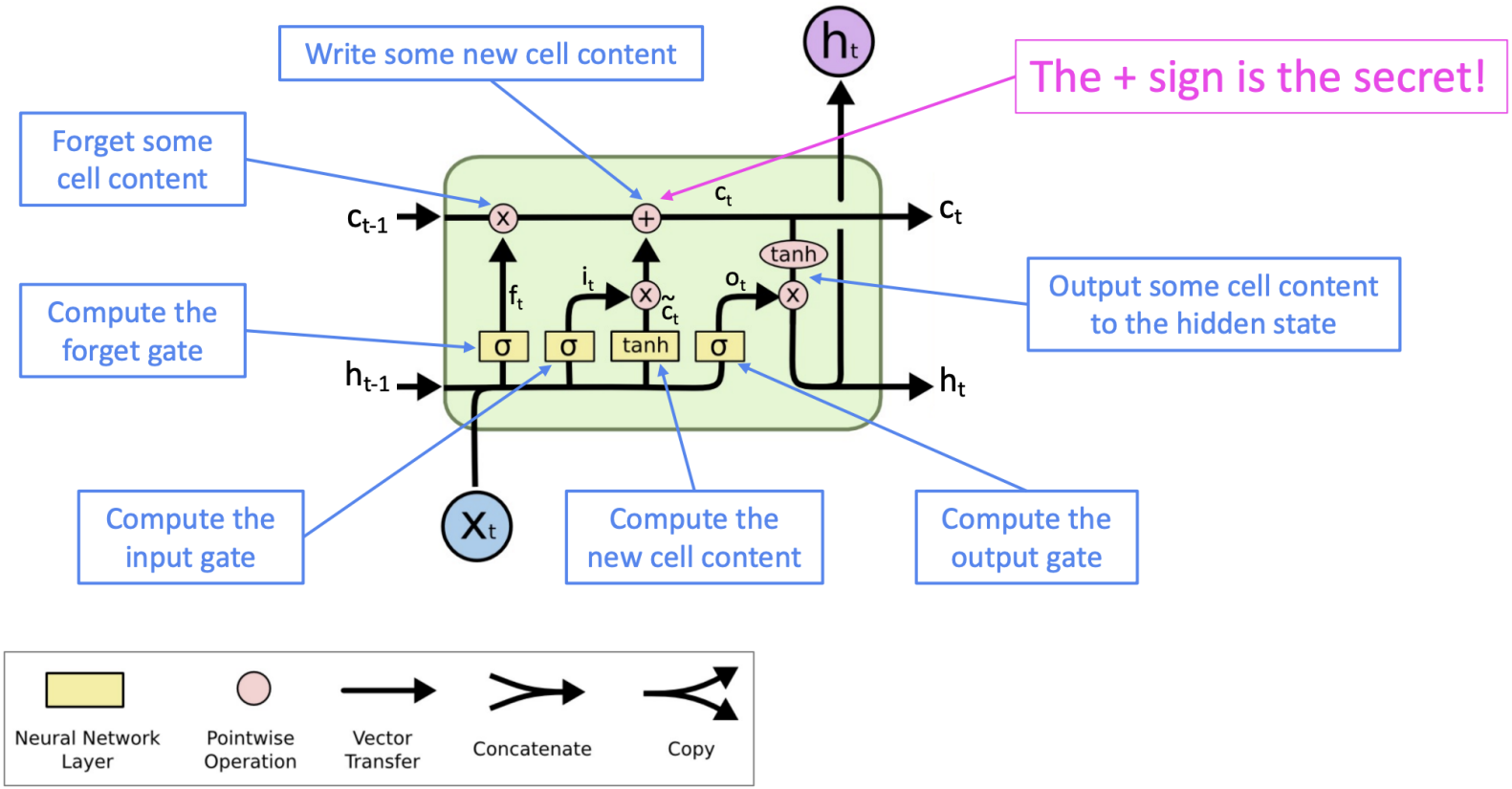
Long short-term memory (LSTM)



Source - Girafe.AI

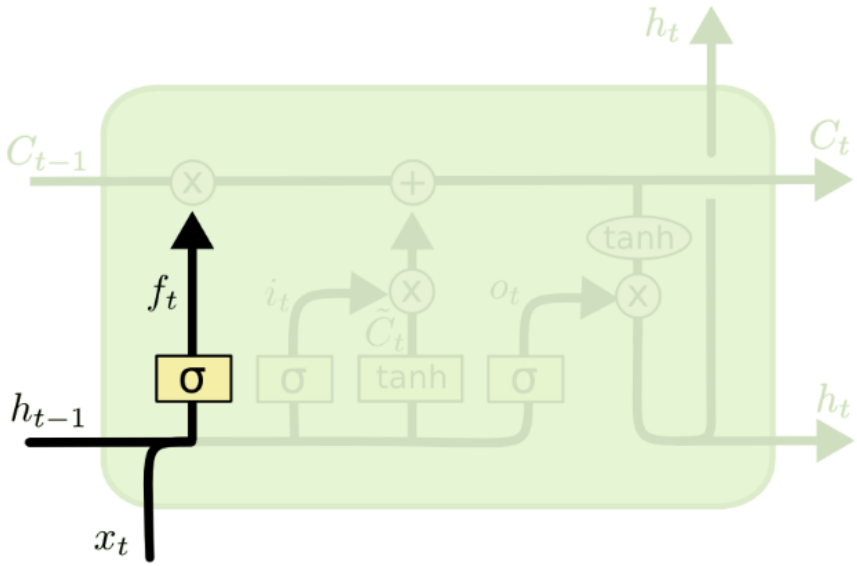
LSTM Overview

You can think of the LSTM equations visually like this:



Source - Stanford CS224n

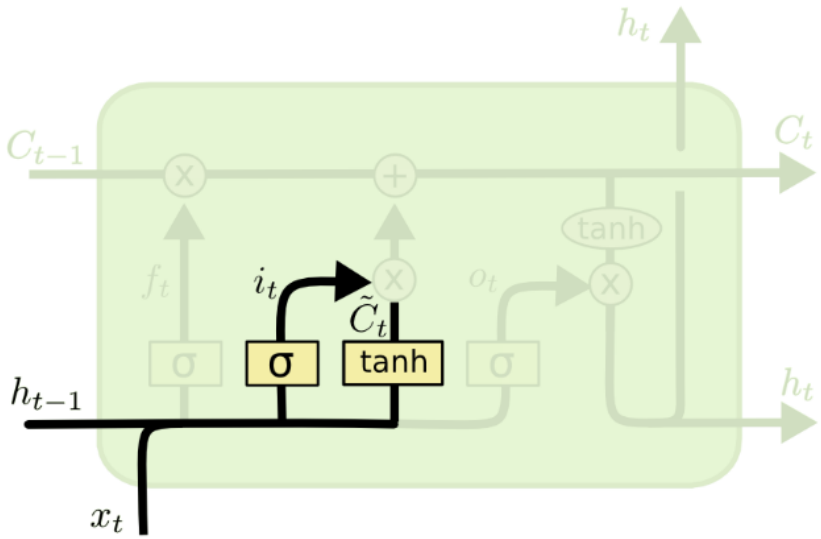
LSTM Overview. Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Source - Understanding LSTM Networks

LSTM Overview. Input Gate and New Cell Content

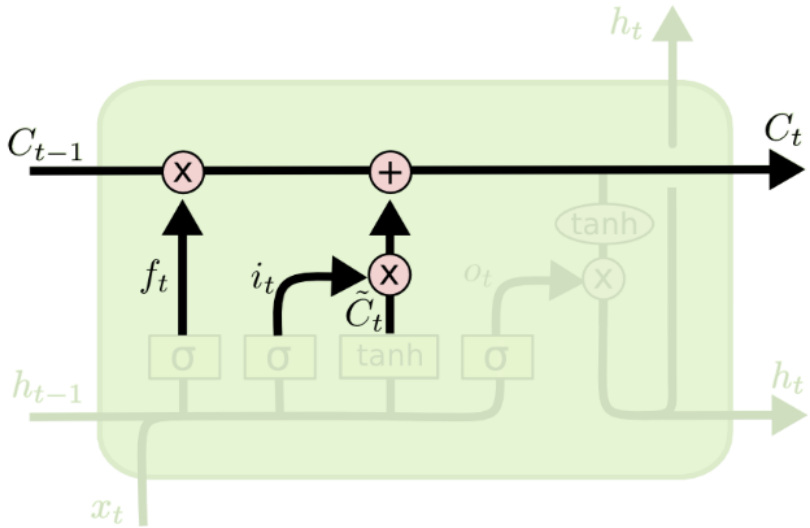


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Source - Understanding LSTM Networks

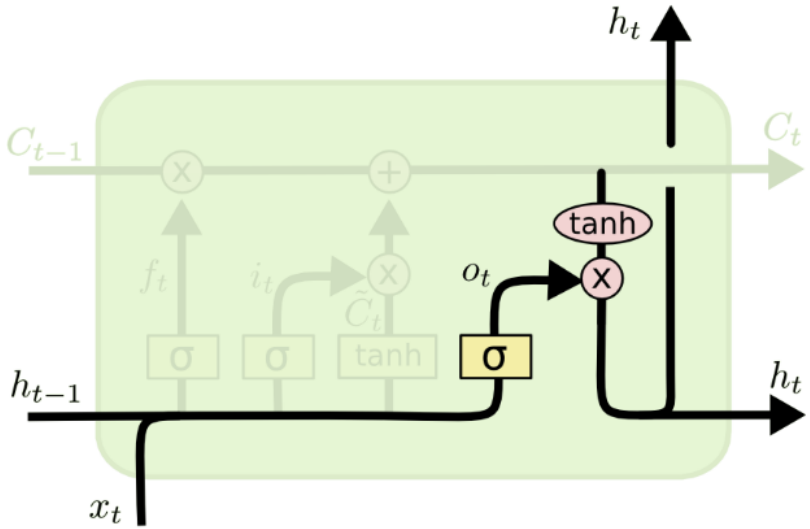
LSTM Overview. Cell State



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source - Understanding LSTM Networks

LSTM Overview. Output Gate and Hidden State



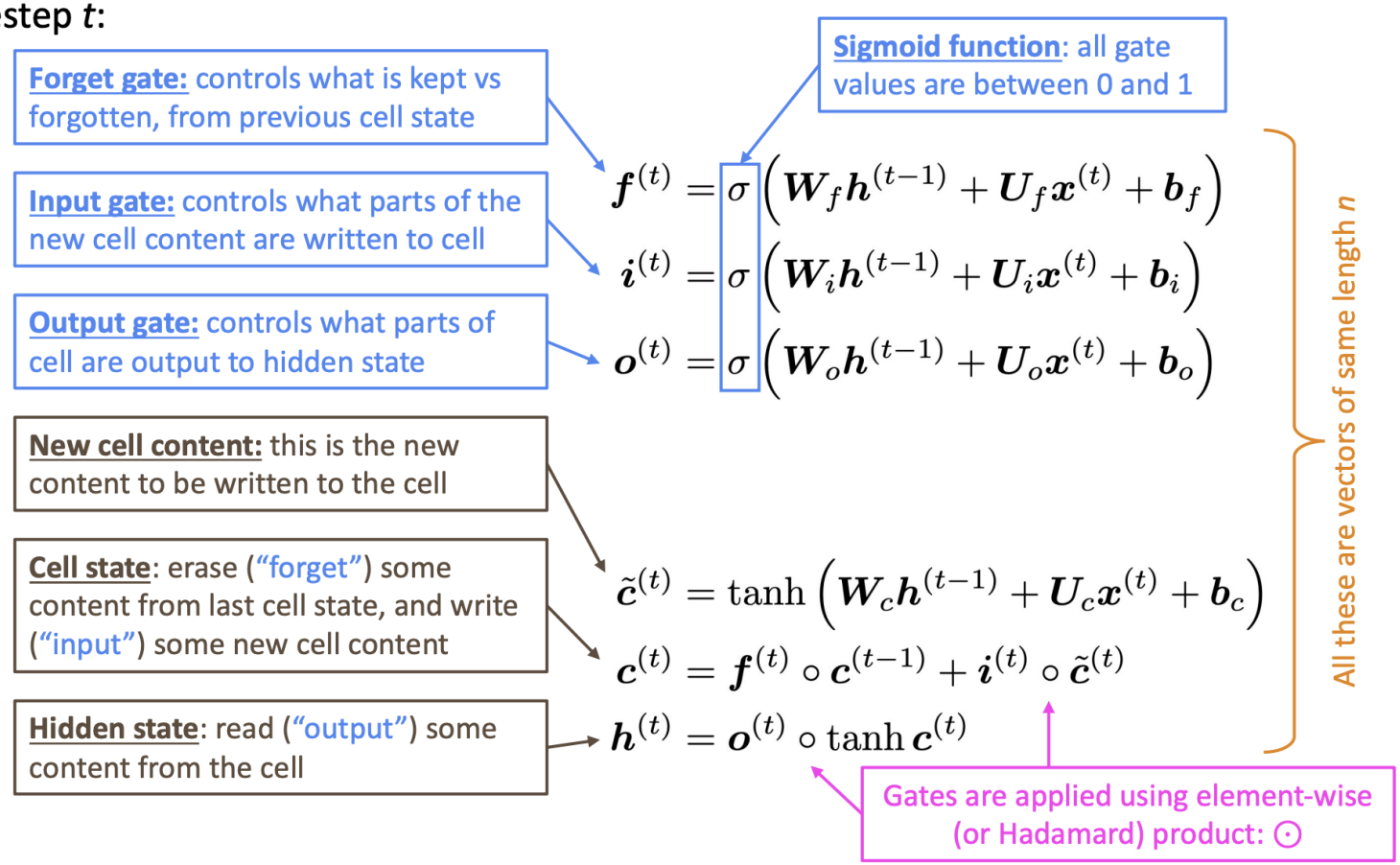
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source - Understanding LSTM Networks

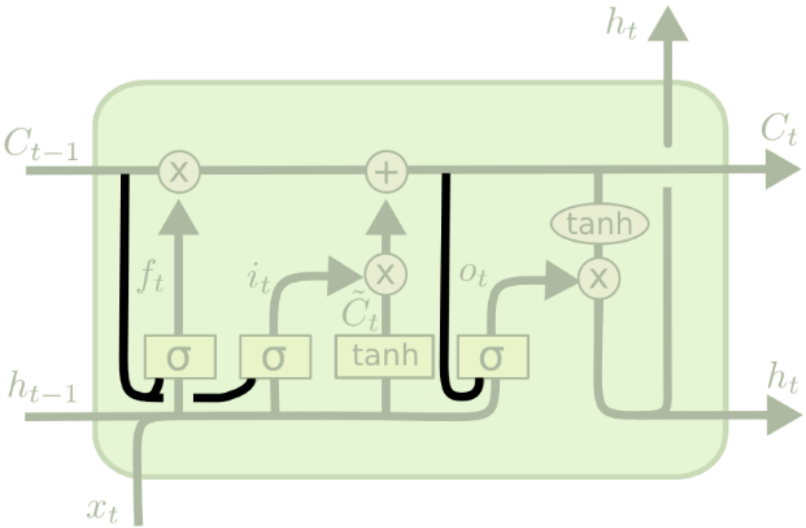
LSTM formulas

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:



Source - Stanford CS224n

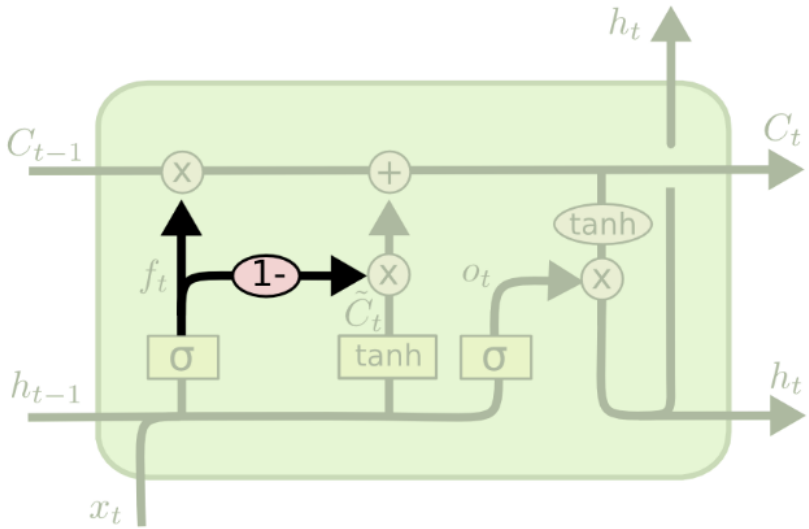
LSTM Variants. Peephole Connections



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Source - Understanding LSTM Networks

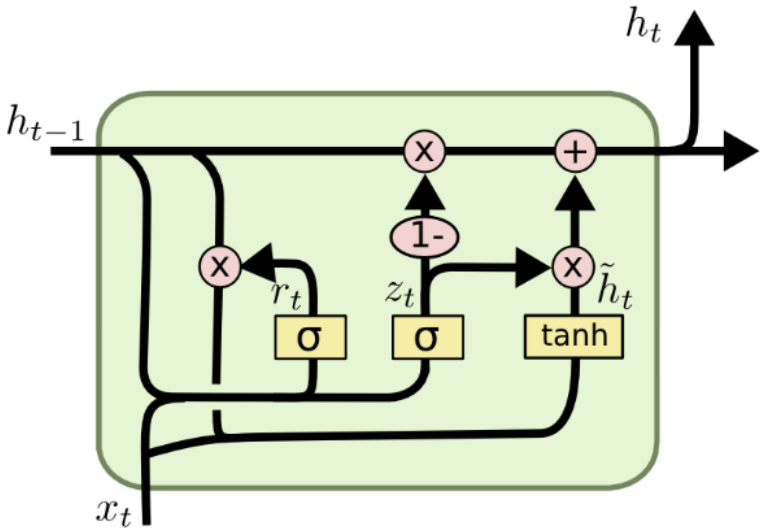
LSTM Variants. Coupled Forget and Input Gates



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Source - Understanding LSTM Networks

Gated Recurrent Unit (GRU)



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

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

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

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

Source - Understanding LSTM Networks

GRU formulas

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep t we have input $\boldsymbol{x}^{(t)}$ and hidden state $\boldsymbol{h}^{(t)}$ (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_u \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u \boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

Reset gate: controls what parts of previous hidden state are used to compute new content

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r \boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_h(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h \boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$$

How does this solve vanishing gradient?
Like LSTM, GRU makes it easier to retain info long-term (e.g., by setting update gate to 0)

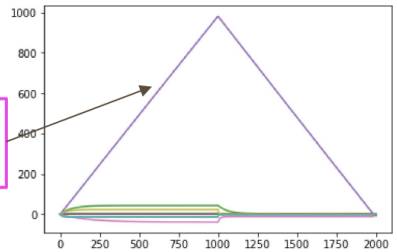
Stanford CS224n

LSTM vs GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- Rule of thumb: LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data); Switch to GRUs for speed and fewer parameters.
- **Note**: LSTMs can store unboundedly* large values in memory cell dimensions, and relatively easily learn to count. (Unlike GRUs.)

LSTM

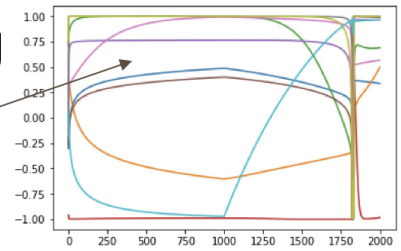
Single dimension used as counter



(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$

GRU

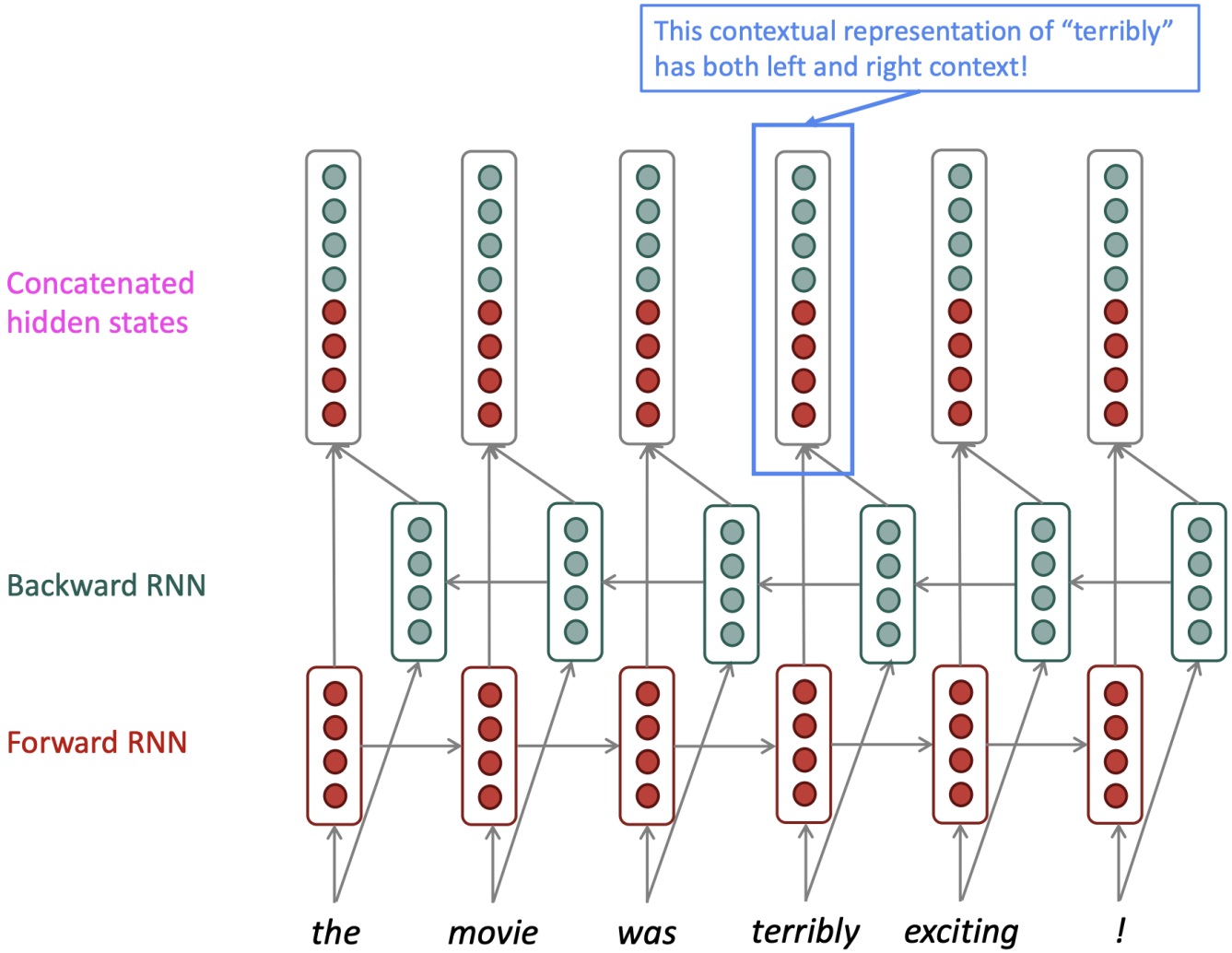
???



(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$

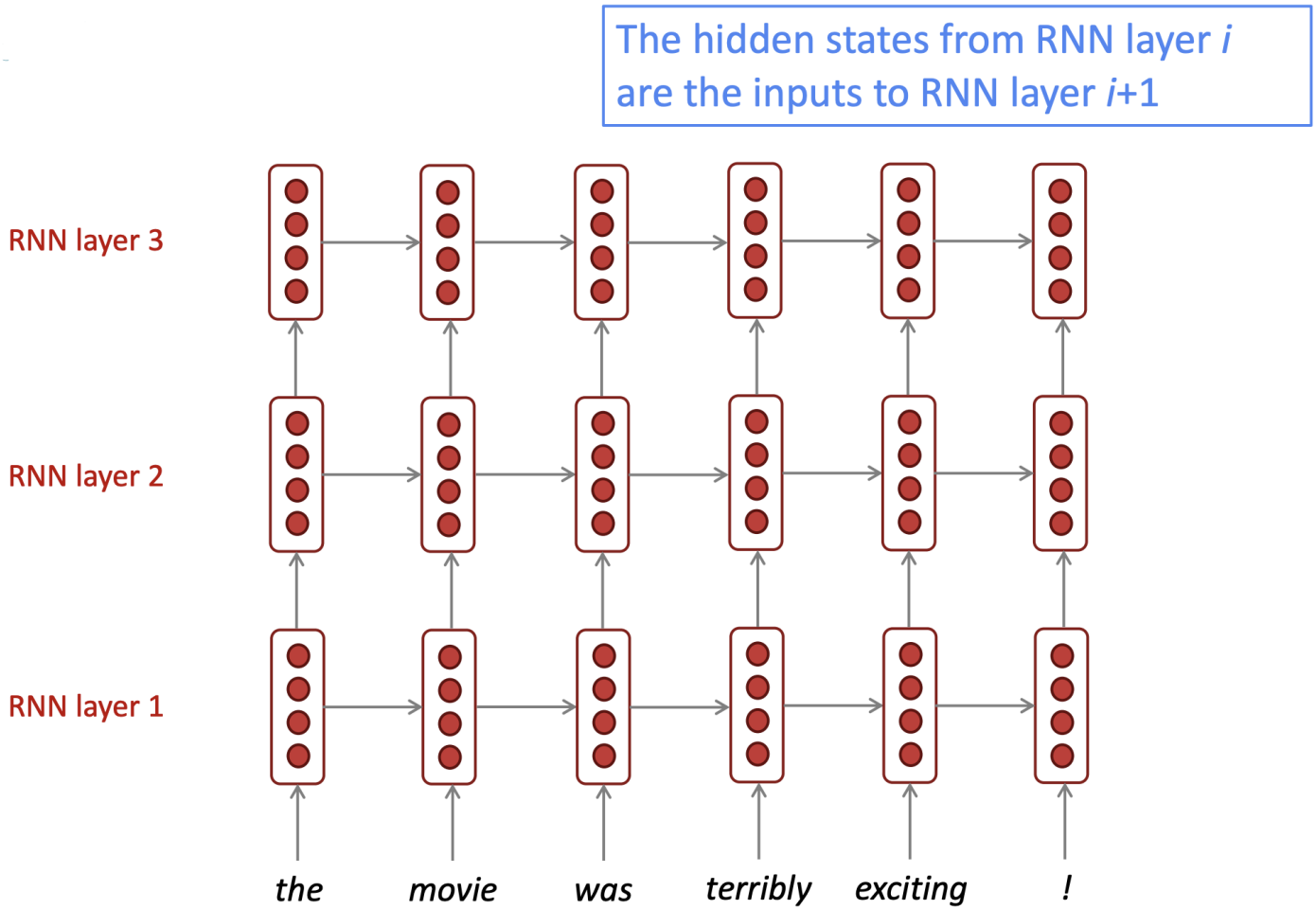
Stanford CS224n

Bidirectional RNNs



Stanford CS224n

Multi-layer RNNs



Stanford CS224n