CSCI 544: Applied Natural Language Processing

Recurrent Neural Networks for Sequence Labeling

Xuezhe Ma (Max)

USC University of Southern California

Course Organization

NLP Tasks

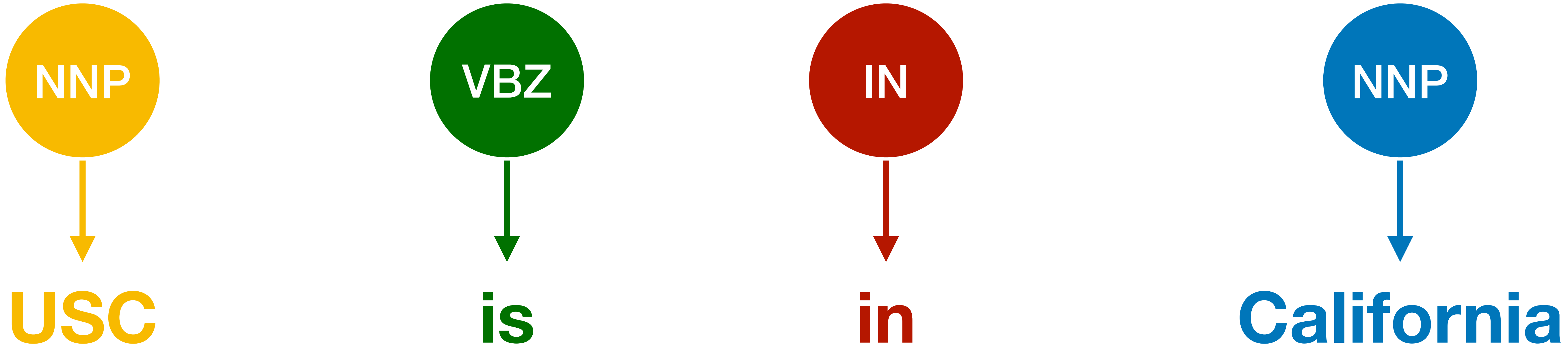
Models	Text Classification	Sequence Labeling	Syntactic Parsing	Seq2seq & MT	Language Modeling & Generation
	Naive Bayes Logistic Regression SVM …	HMMs MEMMs CRFs	Tree-structured Models/Algorithms	……	……
	Word Embedding & Neural Networks	RNN LSTM	Neural Structured Networks	Attention & Transformers	Generative Models & LLMs

Recap: What is Sequence Labeling?

A type of structured prediction tasks

$$Y = < y_i, y_2, \ldots, y_n >$$

$$X = < x_i, x_2, \ldots, x_n >$$



Assigning each token of X, e.g. x_i a corresponding label y_i

Recap: Classical Models for Sequence Labeling

- **Structured Models**
  - HMMs
  - MEMMs
  - CRFs
- **Decoding Algorithms**
  - Greedy decoding
  - Viterbi decoding

Motivation: modeling dependencies between multiple labels/tags

An Essential Question

- Do we need structured models if the feature representations of the input sentence is perfect

VBD

The cat sat on a mat

$$P(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j | x_1, \ldots, x_n)$$

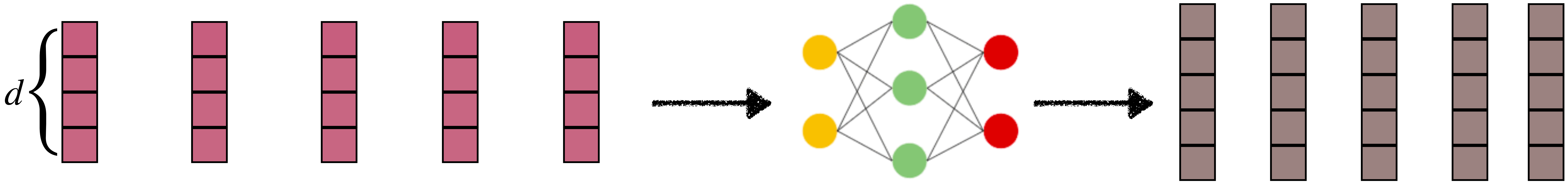
?

Our Goal: Sentence Representations

- One feature vector for each word

$X \in \mathbb{R}^{n \times d}$ Word Embeddings

$Y \in \mathbb{R}^{n \times d'}$



Max is working at USC

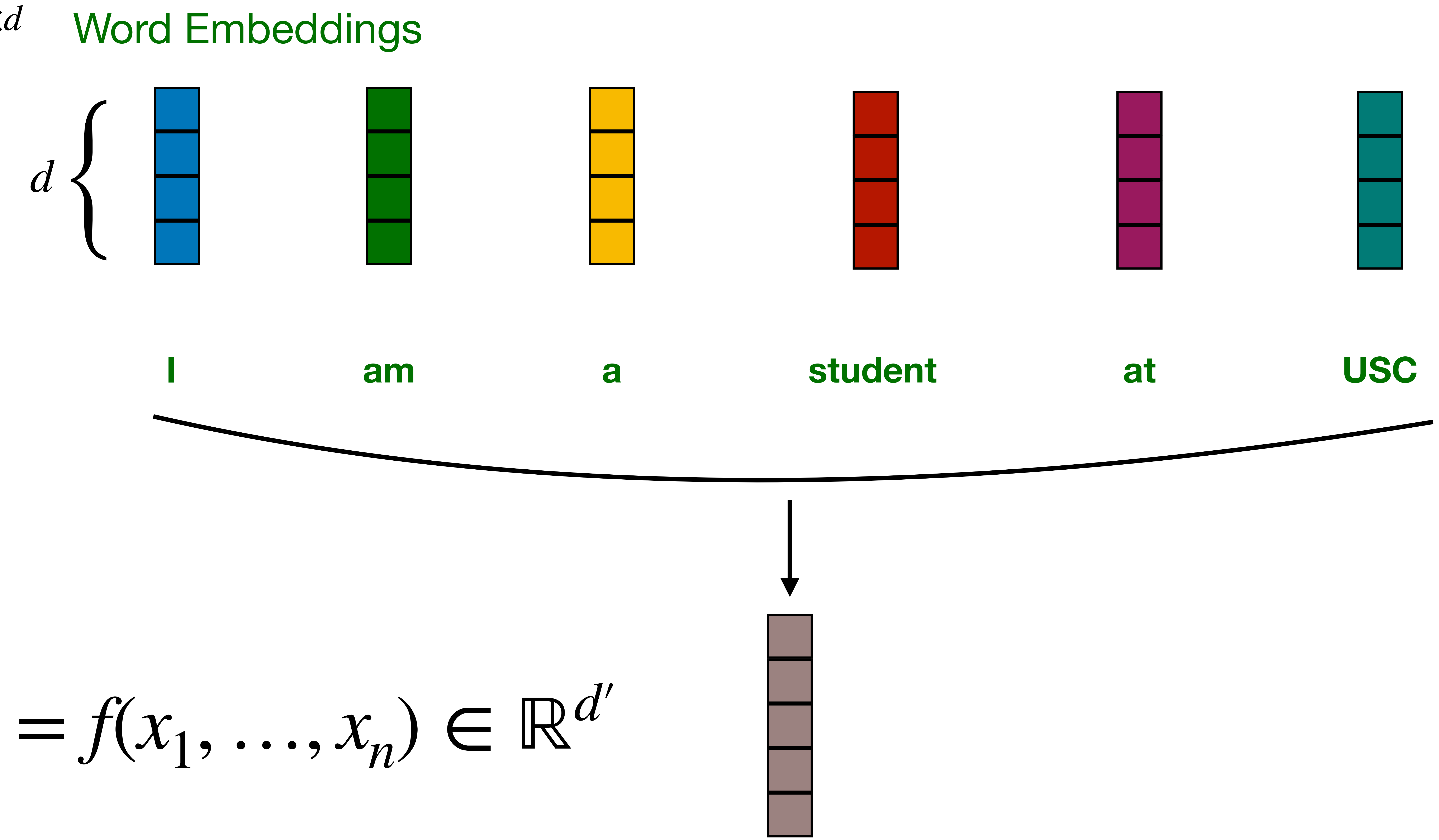
Contextual information in y!

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \mid x_1, \ldots, x_n)$$

Recap: One Vector to Represent a Document

- **Classification**
  - We need a single feature vector to feed into ML classifiers

$X \in \mathbb{R}^{n \times d}$

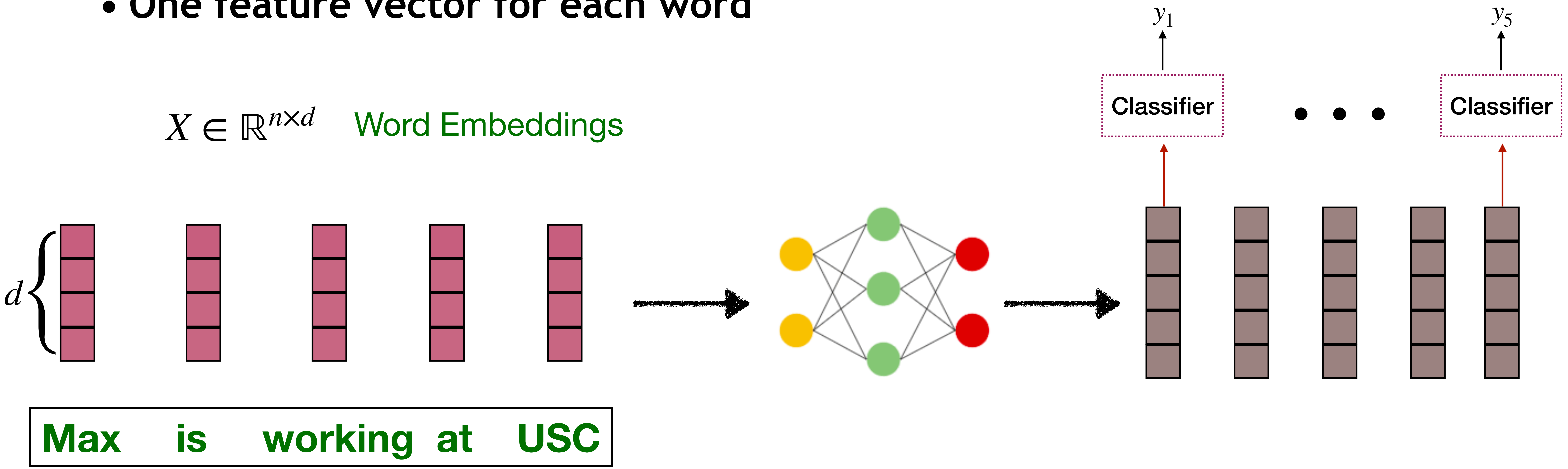


$$y = f(x_1, \ldots, x_n) \in \mathbb{R}^{d'}$$

Our Goal: Sentence Representations

- One feature vector for each word

$X \in \mathbb{R}^{n \times d}$ Word Embeddings

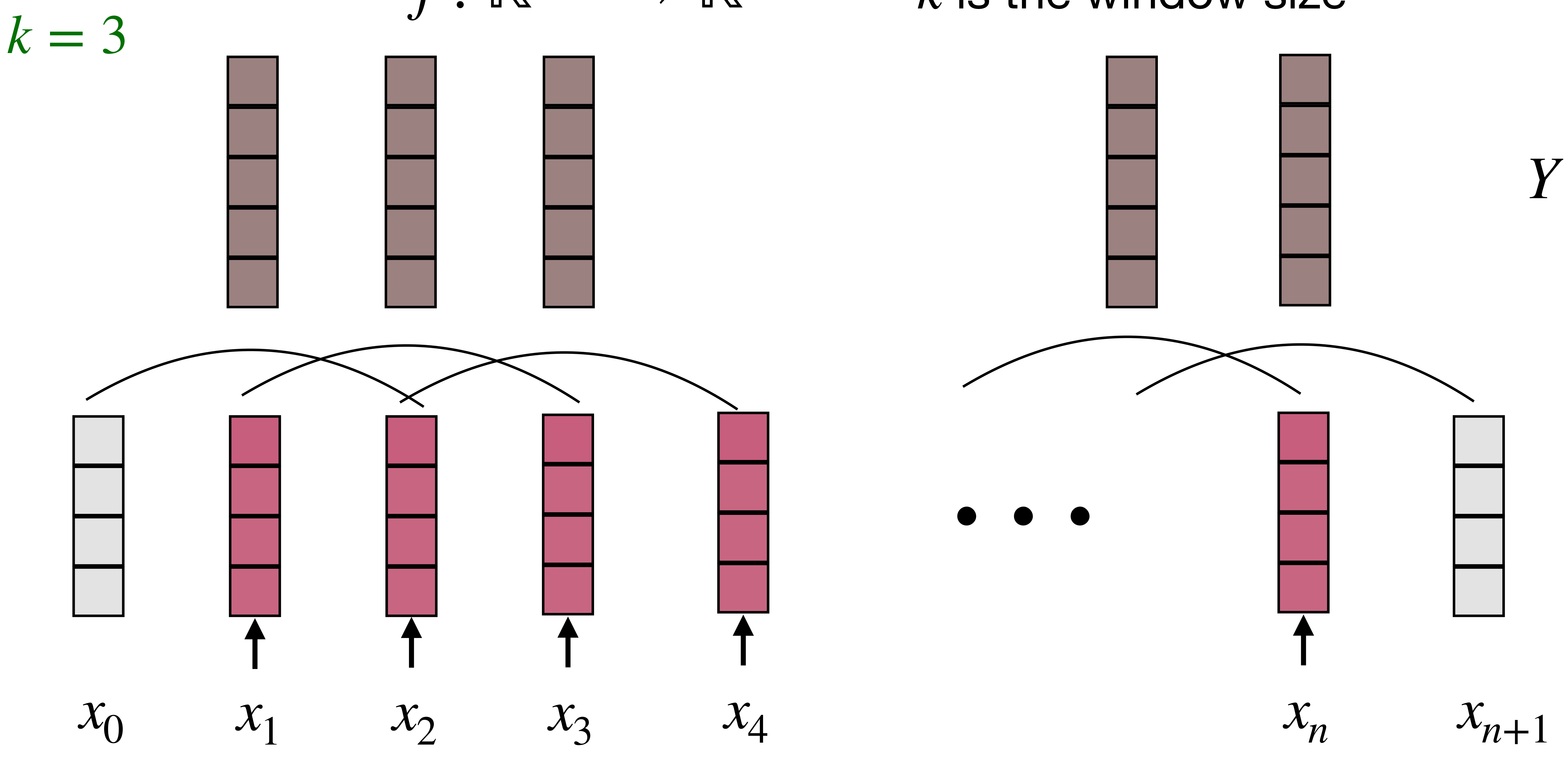


$$P(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j | x_1, \ldots, x_n)$$

Convolutional Neural Networks (CNNs)

- **Basic Idea:** only model a segment of input with fixed window-size

$f: \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d'}$ k is the window size



$$Y = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^{n \times d'}$$

$$X \in \mathbb{R}^{n \times d}$$

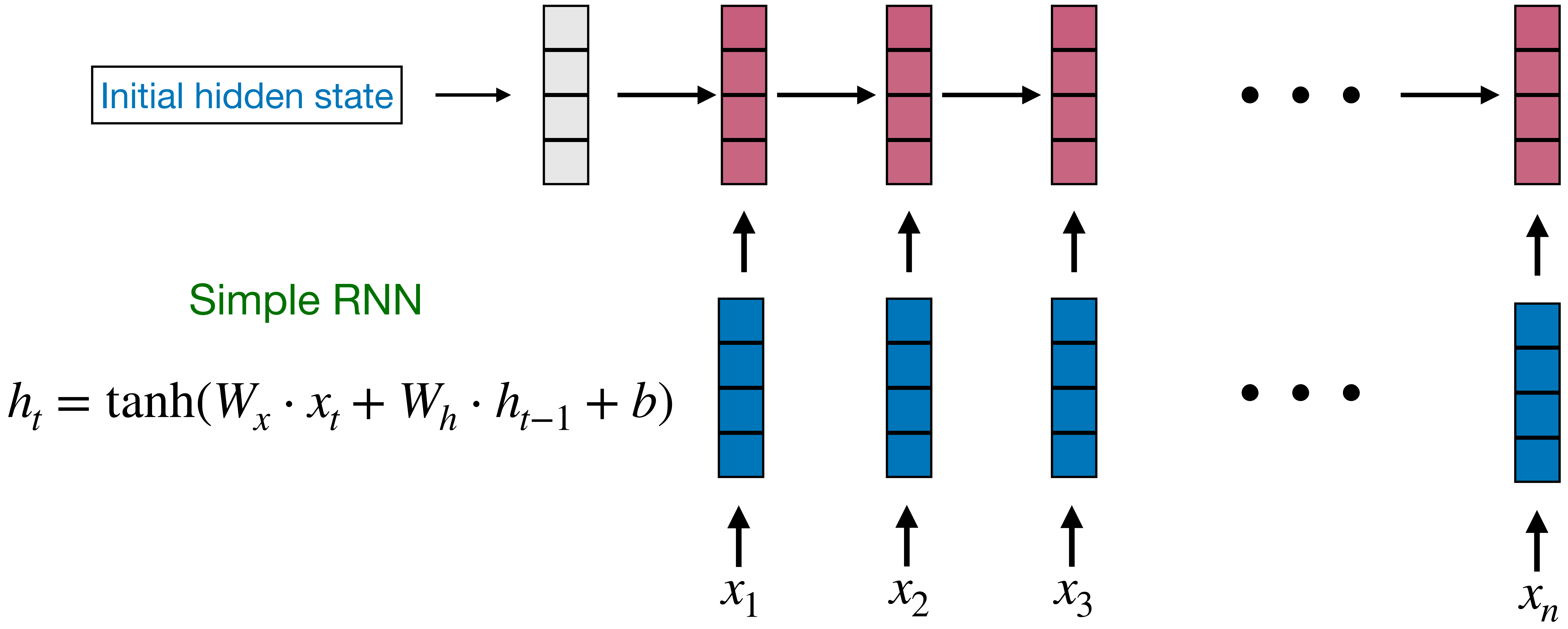
Pros:
- Simple architecture
- Parallel computation

Cons:
- Small context window
- Not good at modeling long dependencies

Recurrent Neural Networks

Neural Networks for Sentence Representations

- **Recurrent Neural Networks (RNNs)**
  - Re-using one feed forward network in a recurrent way

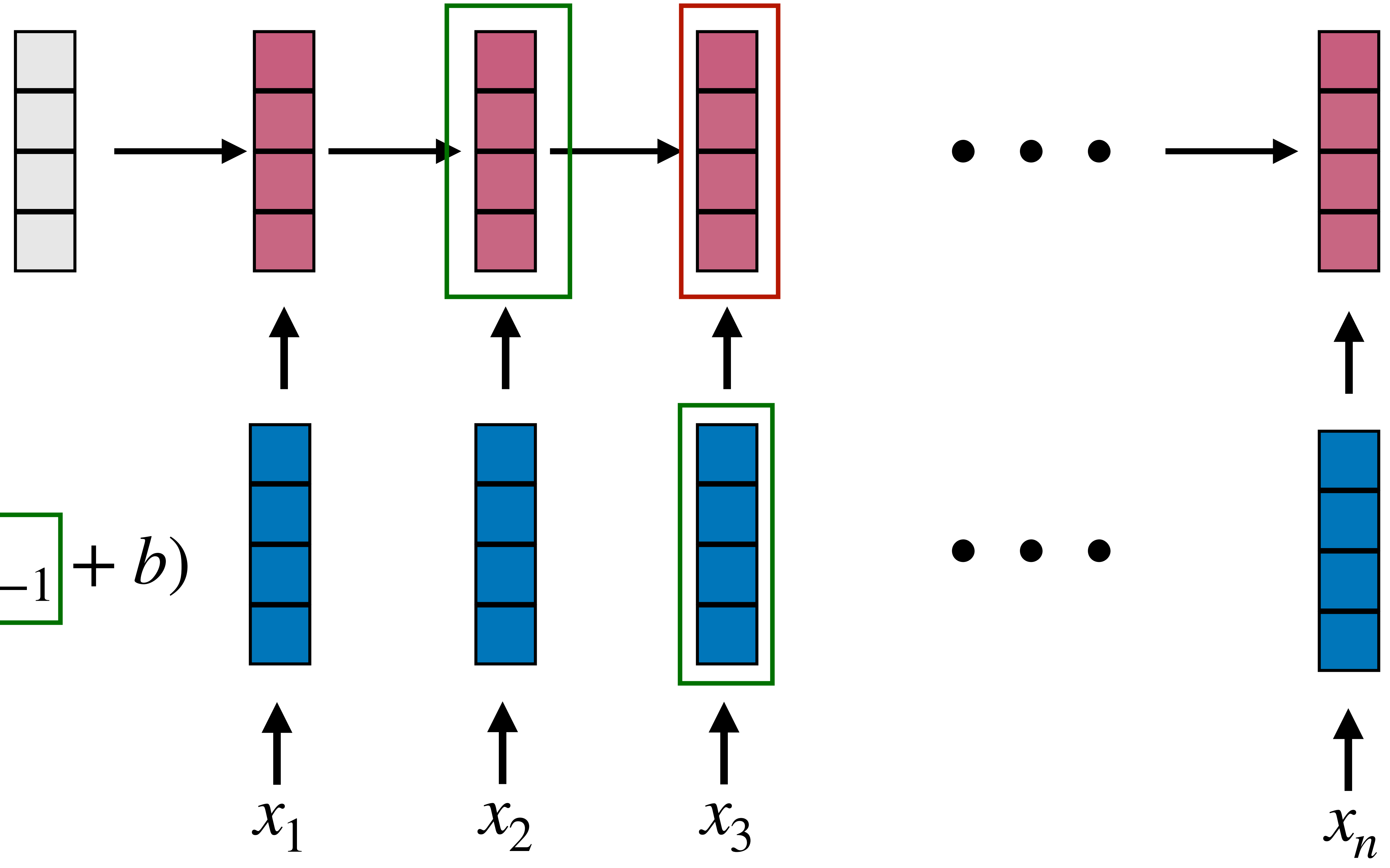


Neural Networks for Sentence Representations

- **Recurrent Neural Networks (RNNs)**
  - Re-using one feed forward network in a recurrent way

Inherent Markov Property

Initial hidden state

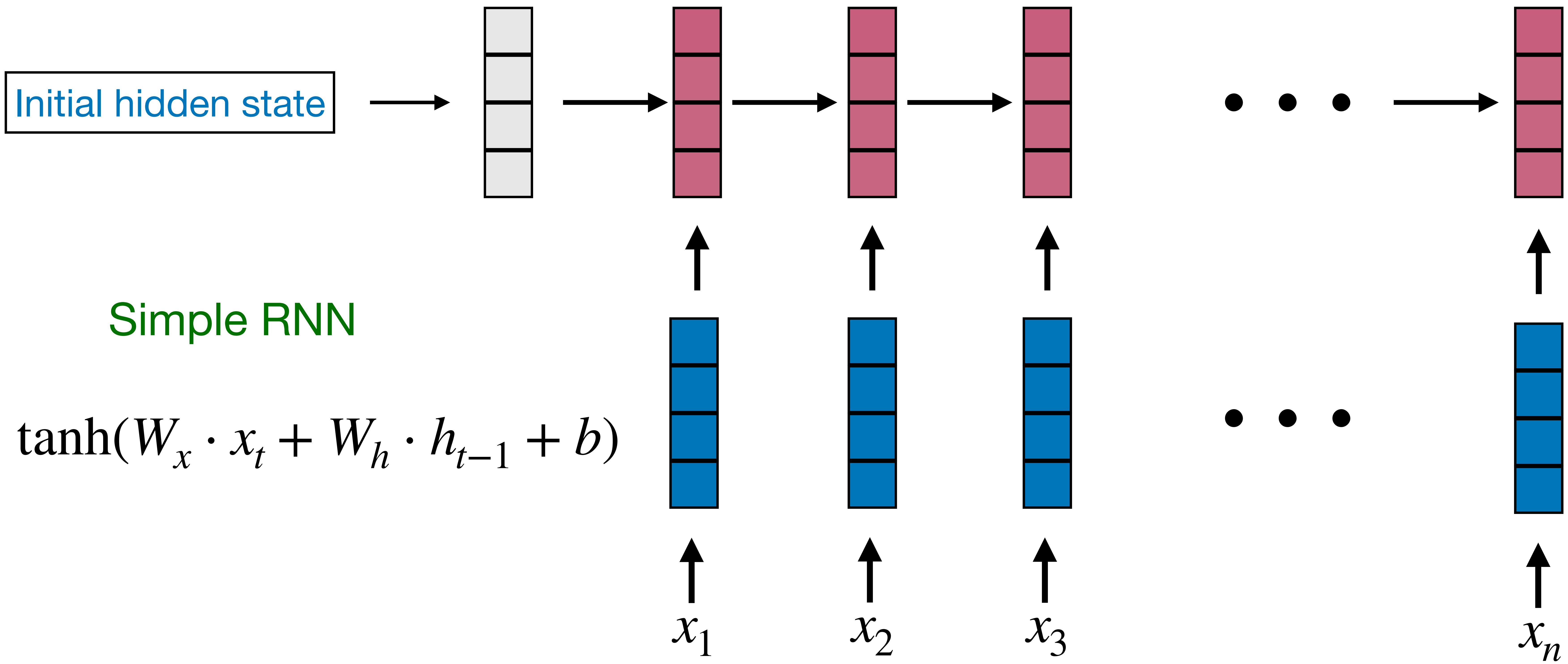


Simple RNN

$$\boxed{h_t} = \tanh(W_x \cdot \boxed{x_t} + W_h \cdot \boxed{h_{t-1}} + b)$$

Problems of Simple RNN

- **No future contexts**
- **Hard to train**
  - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation
- **Limited size of hidden states**
  - Memory cost

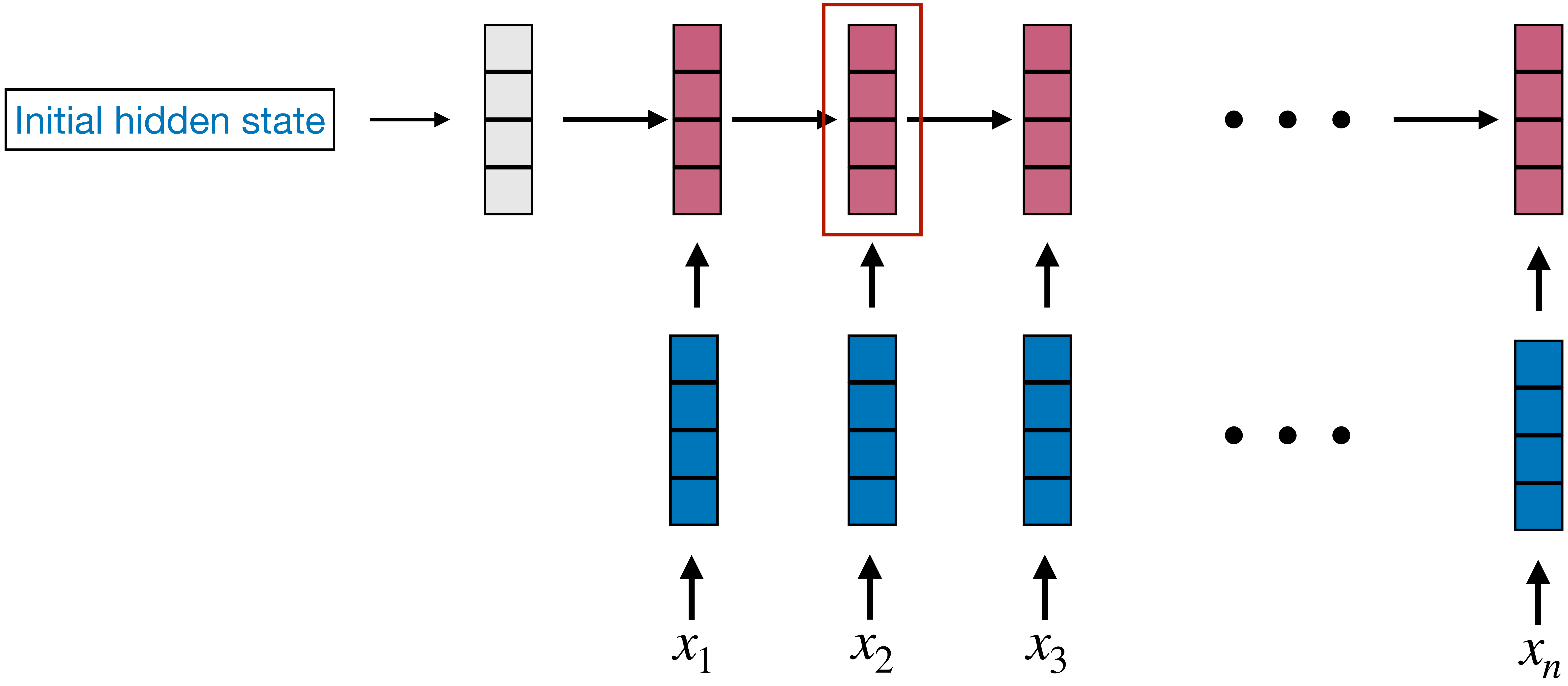


$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

Bidirectional RNNs

- **RNN cannot model future information**

h_2 cannot access the information in $x_3, \ldots x_n$



Bidirectional RNNs

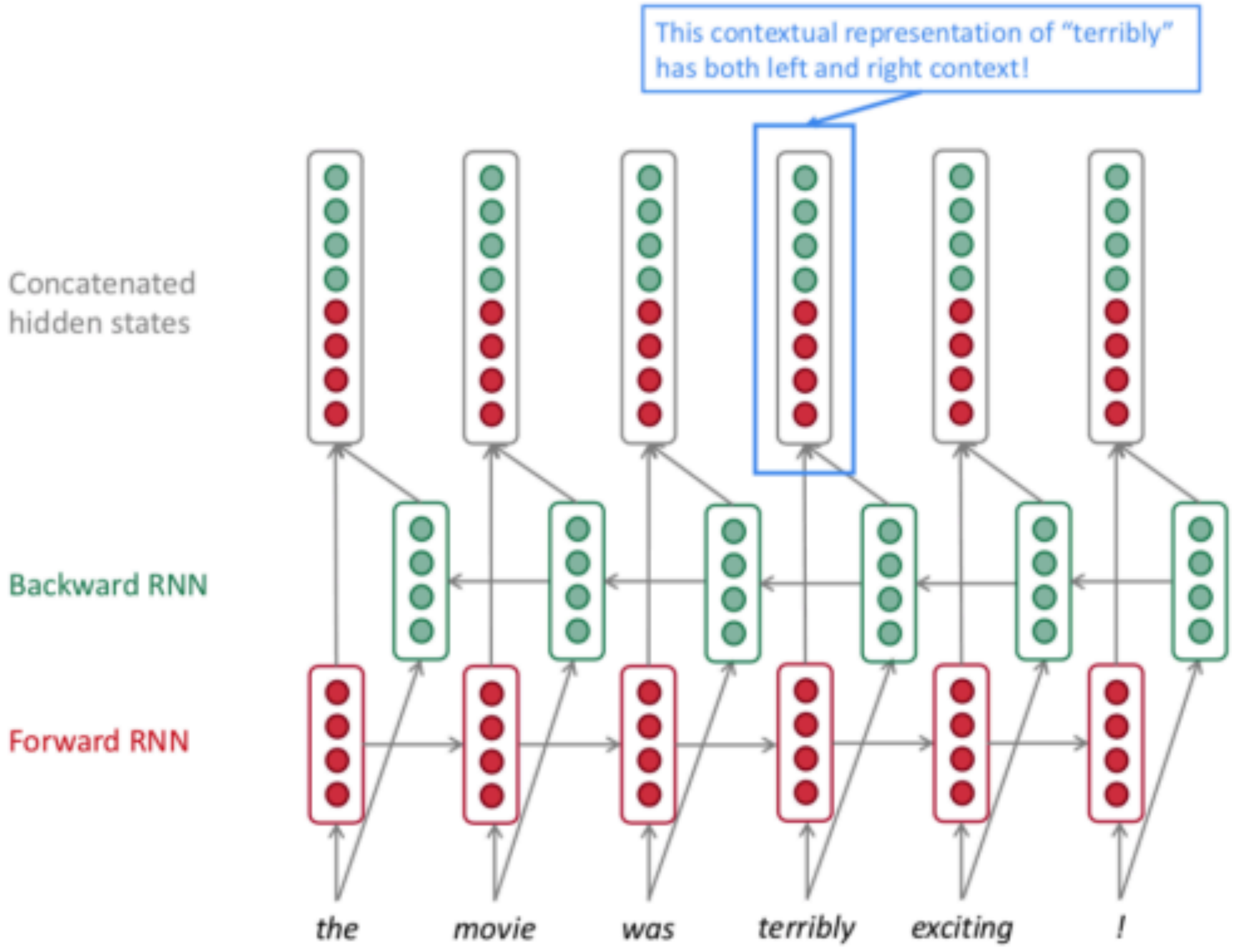
- **Future information is important for sequence labeling tasks!**

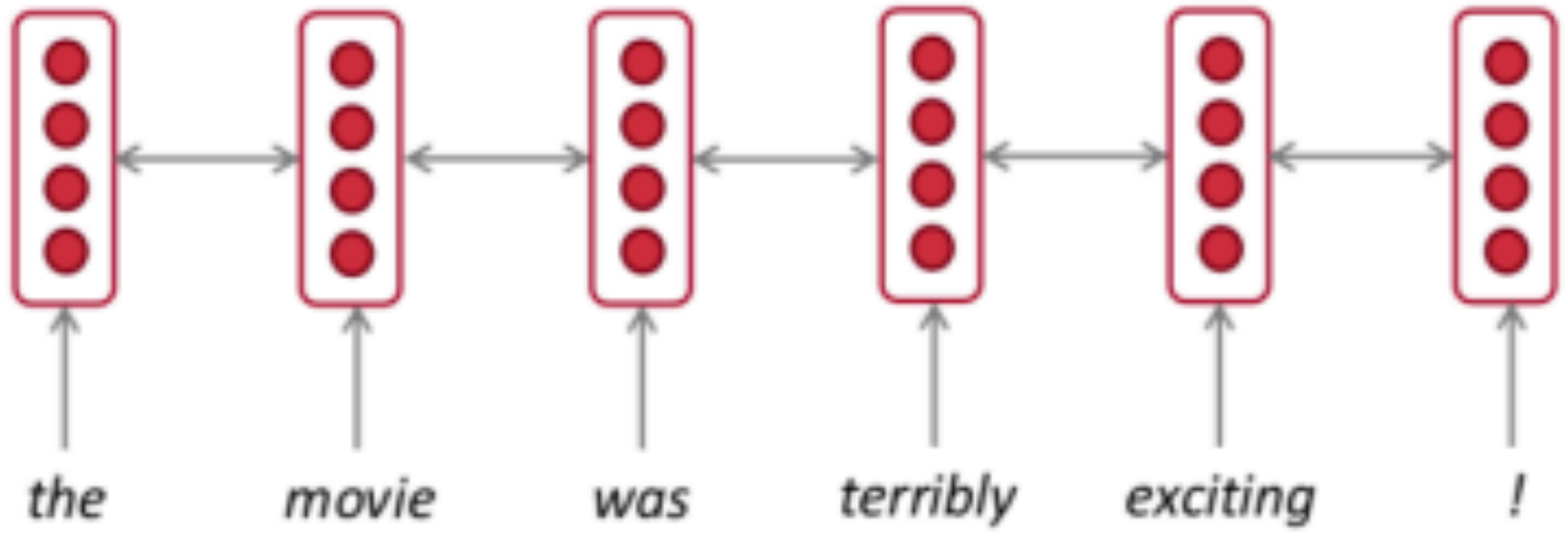
VBD

The cat sat on a mat

$$P(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j | x_1, \ldots, x_n)$$

Bidirectional RNNs





$$\boxed{\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t) \in \mathbb{R}^h}$$

$$\overrightarrow{\mathbf{h}}_t = f_1(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), t = 1, 2, \ldots n$$

$$\overleftarrow{\mathbf{h}}_t = f_2(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{x}_t), t = n, n-1, \ldots 1$$

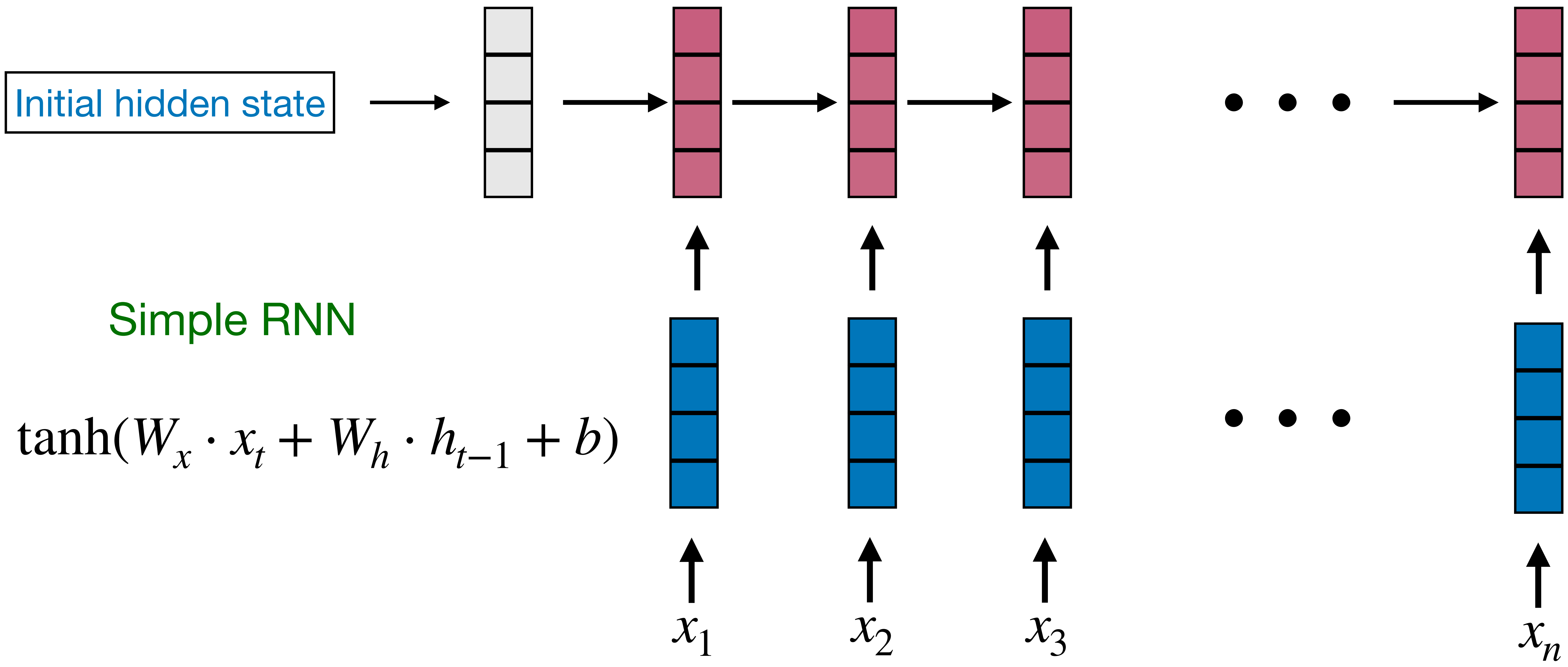
$$\mathbf{h}_t = [\overleftarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{h}}_t] \in \mathbb{R}^{2h}$$

Bidirectional RNNs

- Bidirectional RNNs are only applicable if we have access to the entire input sequence
- If we do have entire input sequence, bidirectionality is powerful (and should be the default choice)
- A very common choice for sentence/document encoding: multi-layer bidirectional RNNs

Problems of Simple RNN

- No future contexts
- Hard to train
  - Gradient varnishing/exploding
- Inefficient
  - Sequential computation
- Limited size of hidden states
  - Memory cost



$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

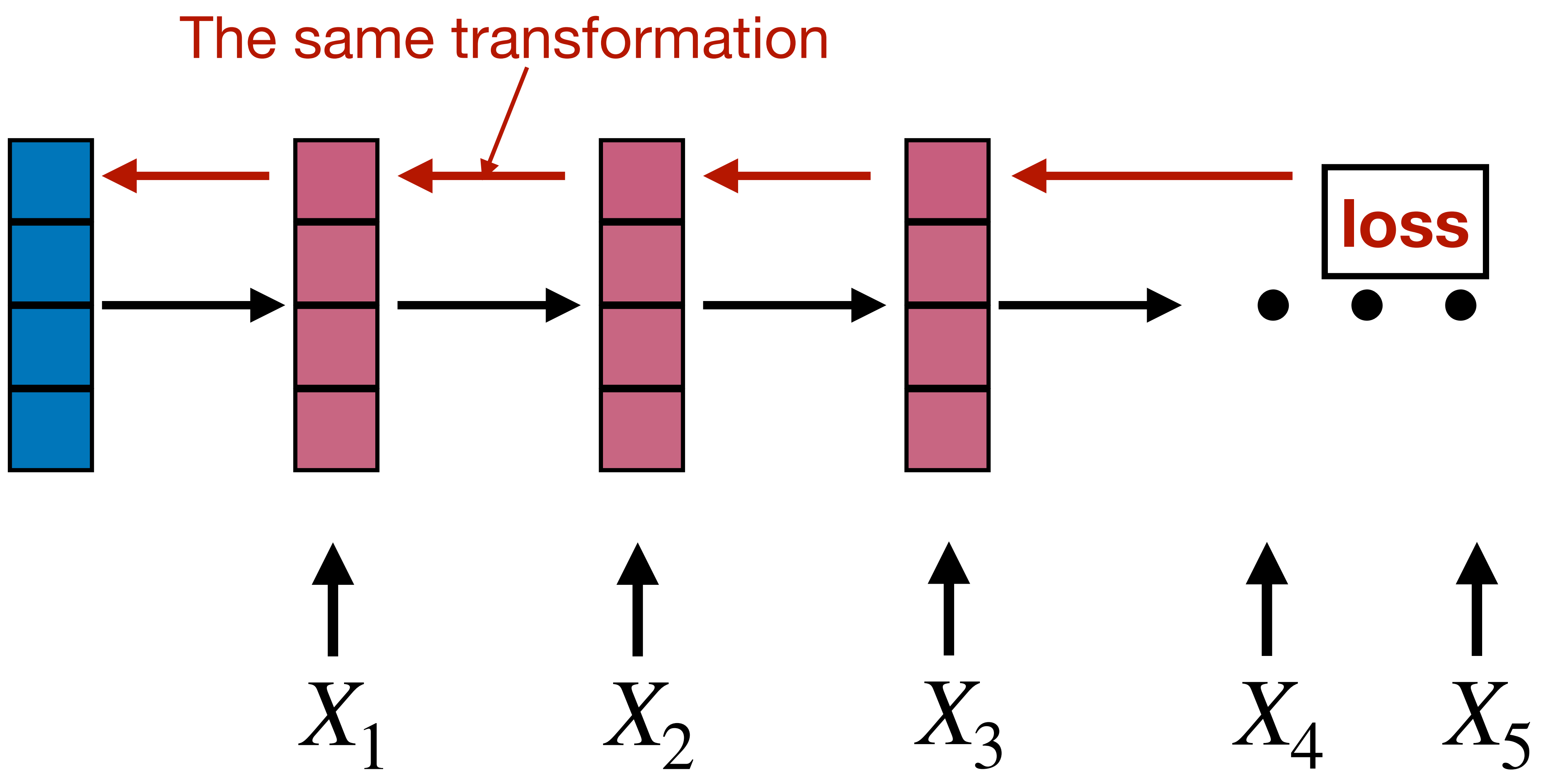
Simple RNN is hard to train

- **Hard to capture long-distance information: vanishing/exploding gradients**

$$h_t = \tanh(W_x \cdot x_t + W_h \cdot h_{t-1} + b)$$

$$\|\frac{\partial h_t}{\partial h_{t-1}}\| \sim \|W_h\|$$

$$\|\frac{\partial h_t}{\partial h_{t-m}}\| \sim \|W_h\|^m$$



Why is this not a serious problem for multi-layer FFN/CNN?

Advanced RNN Variants

- **Long-short Term Memory (LSTMs)**
- **Gated Recurrent Units (GRUs)**

LSTMs

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{U}^i \mathbf{x}_t + \mathbf{b}^i)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{U}^f \mathbf{x}_t + \mathbf{b}^f)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{U}^o \mathbf{x}_t + \mathbf{b}^o)$$

$$\mathbf{g}_t = \tanh(\mathbf{W}^g \mathbf{h}_{t-1} + \mathbf{U}^g \mathbf{x}_t + \mathbf{b}^g)$$

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{g}_t \odot \mathbf{i}_t$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

GRUs

$$\mathbf{r}_t = \sigma(\mathbf{W}^r \mathbf{h}_{t-1} + \mathbf{U}^r \mathbf{x}_t + \mathbf{b}^r)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z)$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{U}\mathbf{x}_t + \mathbf{b})$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

LSTM is more an art than a science

LSTM: Long Short-Term Memory

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{h}_{t-1} + \mathbf{U}^i \mathbf{x}_t + \mathbf{b}^i)$$

Input gate: $i_t \in (0,1)$

$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{h}_{t-1} + \mathbf{U}^f \mathbf{x}_t + \mathbf{b}^f)$$

Forget gate: $f_t \in (0,1)$

$$\mathbf{o}_t = \sigma(\mathbf{W}^o \mathbf{h}_{t-1} + \mathbf{U}^o \mathbf{x}_t + \mathbf{b}^o)$$

Output gate: $o_t \in (0,1)$

$$\mathbf{g}_t = \tanh(\mathbf{W}^g \mathbf{h}_{t-1} + \mathbf{U}^g \mathbf{x}_t + \mathbf{b}^g)$$

Simple RNN

$$\mathbf{c}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t + \mathbf{g}_t \odot \mathbf{i}_t$$

Cell state vector (internal memory)

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t$$

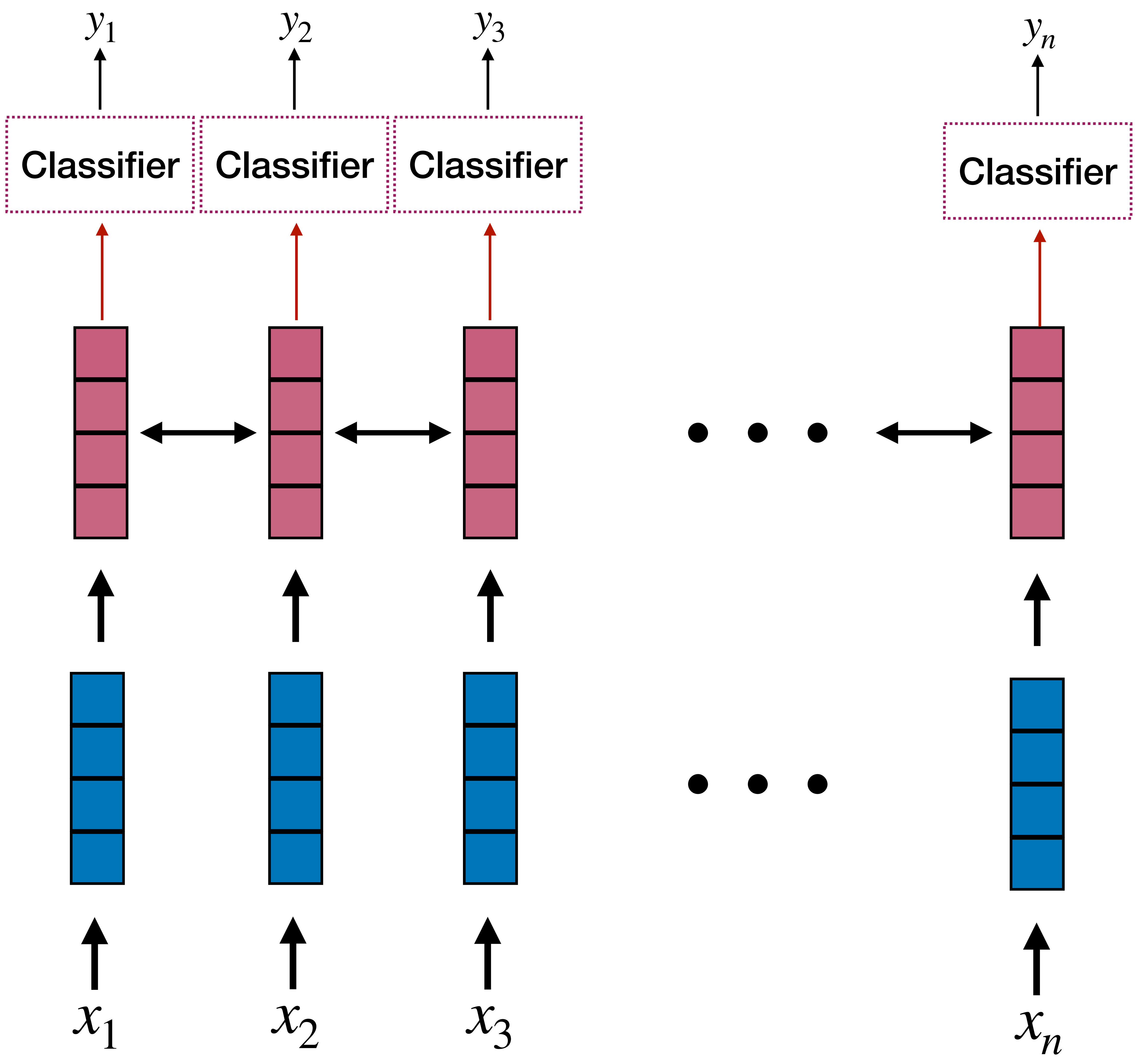
Final hidden vector

All the components are directly learned during training

LSTMs for Sequence Labeling

LSTM for Sequence Labeling

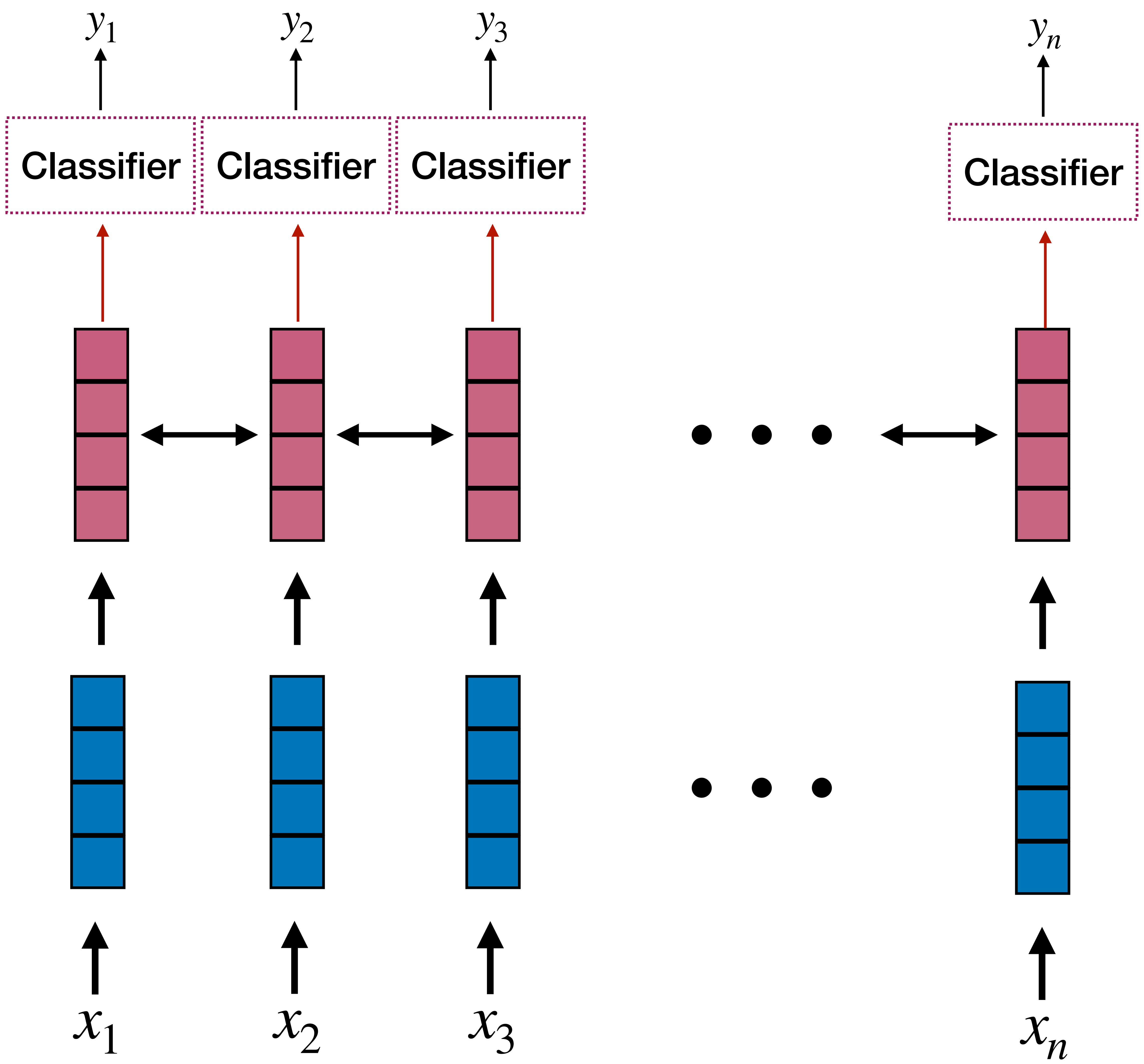
- A simple bidirectional LSTM model



No structured models to consider dependencies between Y

Sequence Labeling

- How to initialize parameters?



LSTM: **Unif**$(-\sqrt{1/d}, \sqrt{1/d})$

Embedding: Word2Vec or GloVe

Sequence Labeling

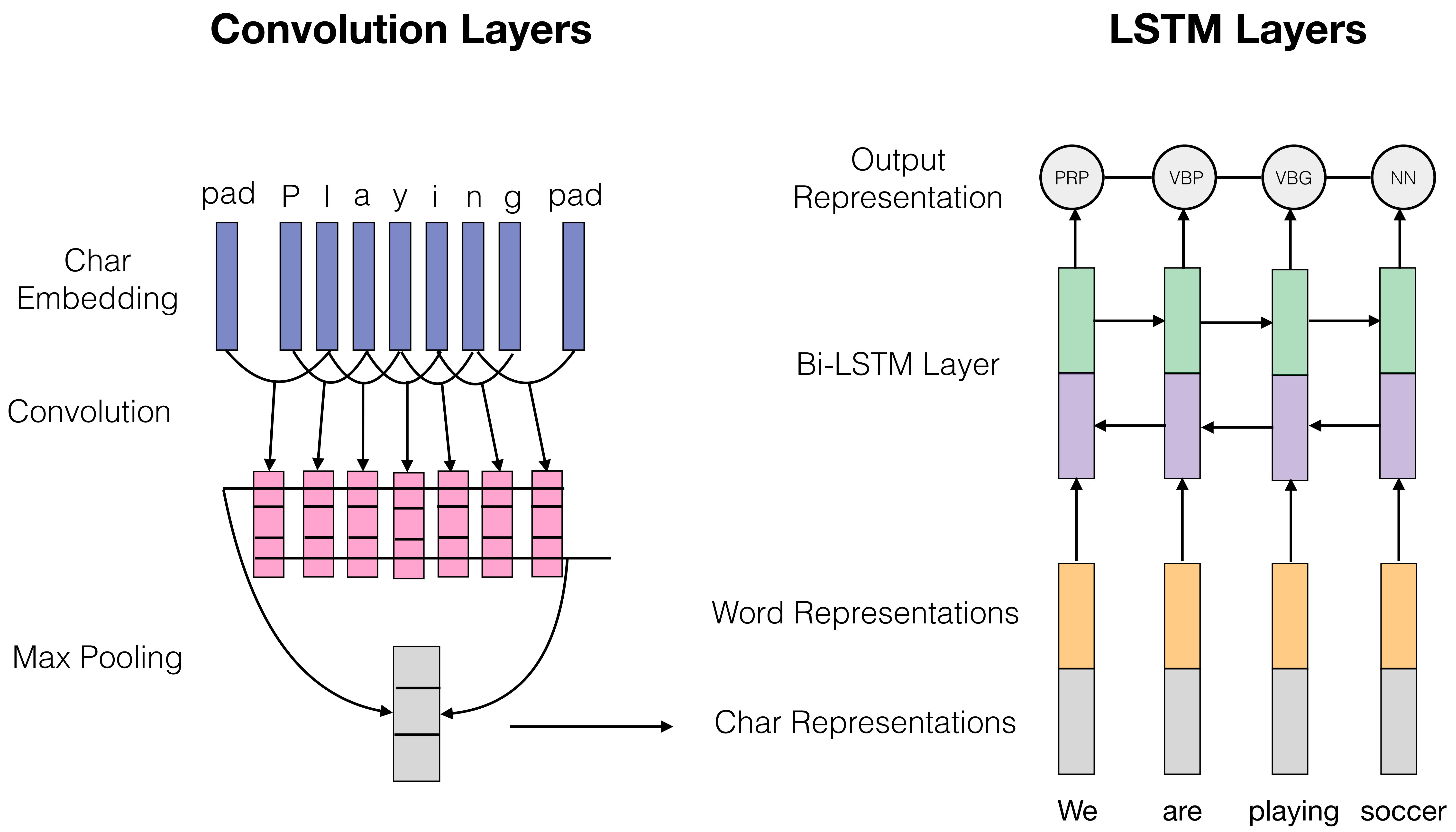
- A simple bidirectional LSTM model

	POS Tagging	NER
HMM	96.4%	75.3%
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM-Random	96.1%	80.7%
BLSTM-GloVe	96.9%	87.0%

BLSTM is not good enough!

Bidirectional LSTMs + CNNs

- **Bidirectional LSTM only encodes word-level information**
- **Spelling features are important**
  - Using CNN to model character-level information



Sequence Labeling

- **BLSTM-CNN**

	POS Tagging	NER
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM	96.9%	87.0%
BLSTM-CNN	97.3%	89.4%

BLSTM-CNN is better than CRF!

An Essential Question

- Do we need structured models if the feature representations of the input sentence is perfect

VBD

The **cat** **sat** **on** **a** **mat**

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n) = \prod_{j=1}^{n} P(y_j \mid x_1, \ldots, x_n) \quad ?$$

BiLSTM + Char-level CNN is not strong enough…

Structured models are still useful?

Combining structured models with LSTM?

Recap: Log-Linear Models

- **MEMMs**

$$p(y_j \mid y_{j-1}, x_1, \ldots, x_m) = \frac{\exp(\boxed{v \cdot f(x_1, \ldots, x_m, j, y_{j-1}, y_j)})}{\sum_{y'_j \in \mathbb{Y}} \exp(v \cdot f(x_1, \ldots, x_m, j, y_{j-1}, y'_j))}$$

- **CRFs**

$$p(y_1, \ldots, y_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(\boxed{v \cdot f(x_1, \ldots, x_m, j, y_{j-1}, y_j)})}{\sum_{y'_1, \ldots, y'_m \in \mathbb{Y}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, j, y'_{j-1}, y'_j))}$$

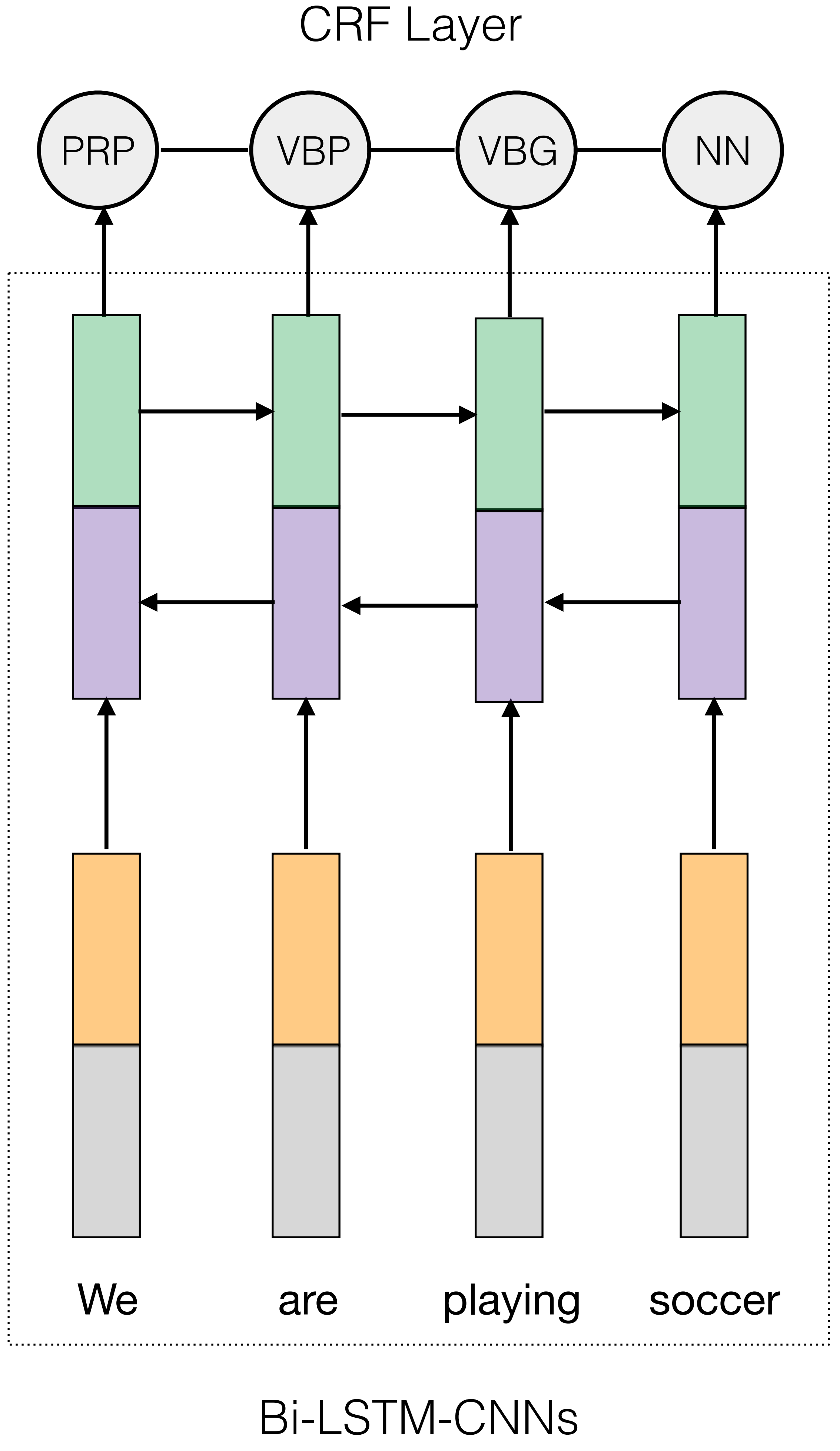
Sequence Labeling: BLSTM-CNNs-CRF

$$v \cdot f(x_1, \ldots, x_m, j, y_{j-1}, y_j) = \mathbf{W}^T_{y_{j-1},y_j} \mathbf{h}_j$$

Parameters w.r.t y_{j-1}, y_j

LSTM hidden state at j

$$p(\mathbf{y} \,|\, \mathbf{x}; \mathbf{W}) = \frac{\prod_{j=1}^{n} \exp\left(\mathbf{W}_{y_{j-1},y_j} \mathbf{h}_j\right)}{\sum_{y' \in \mathcal{Y}} \prod_{j=1}^{n} \exp\left(\mathbf{W}_{y'_{j-1},y_j} \mathbf{h}_j\right)}$$



Sequence Labeling

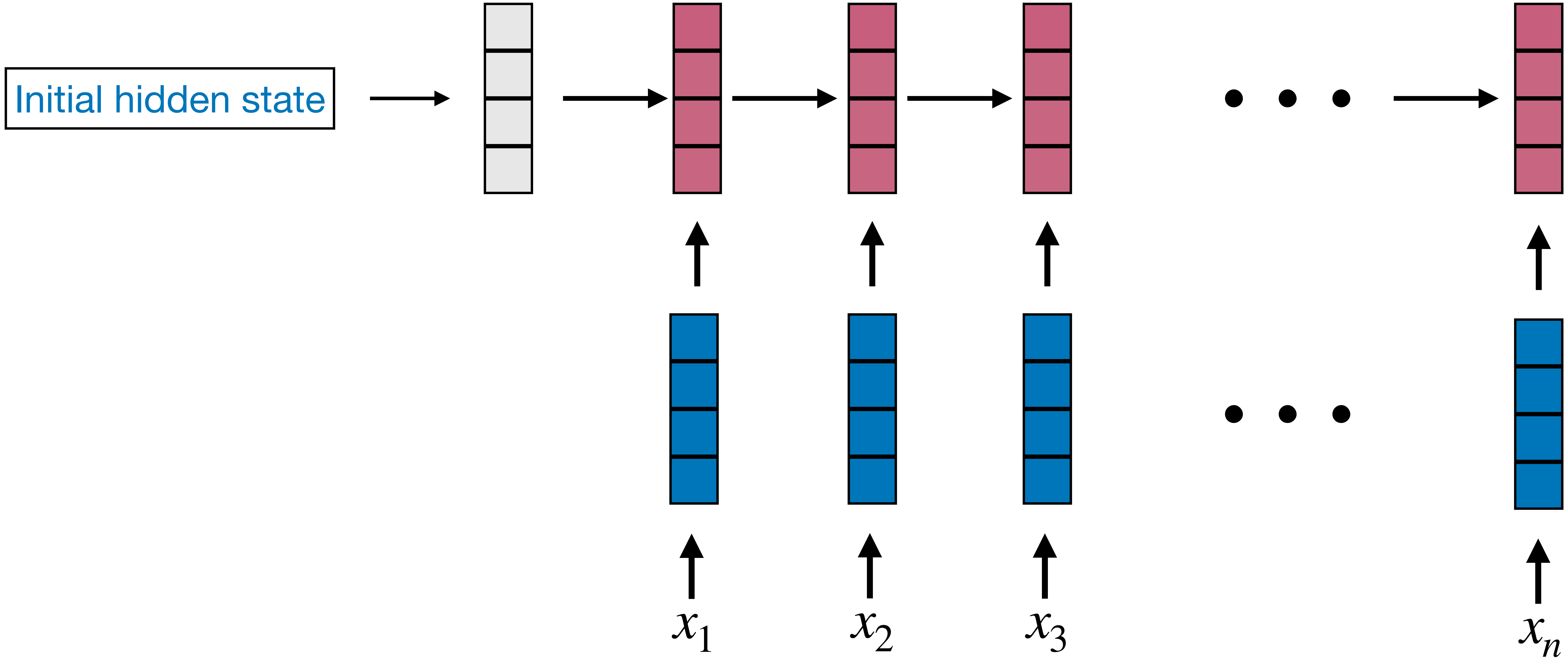
- **BLSTM-CNN-CRF**

	POS Tagging	NER
CRF	97.0%	88.7%
CRF+external resources	97.3%	91.2%
BLSTM	96.9%	87.0%
BLSTM-CNN	97.3%	89.4%
BLSTM-CNN-CRF	97.6%	91.2%

Structured models are still useful!

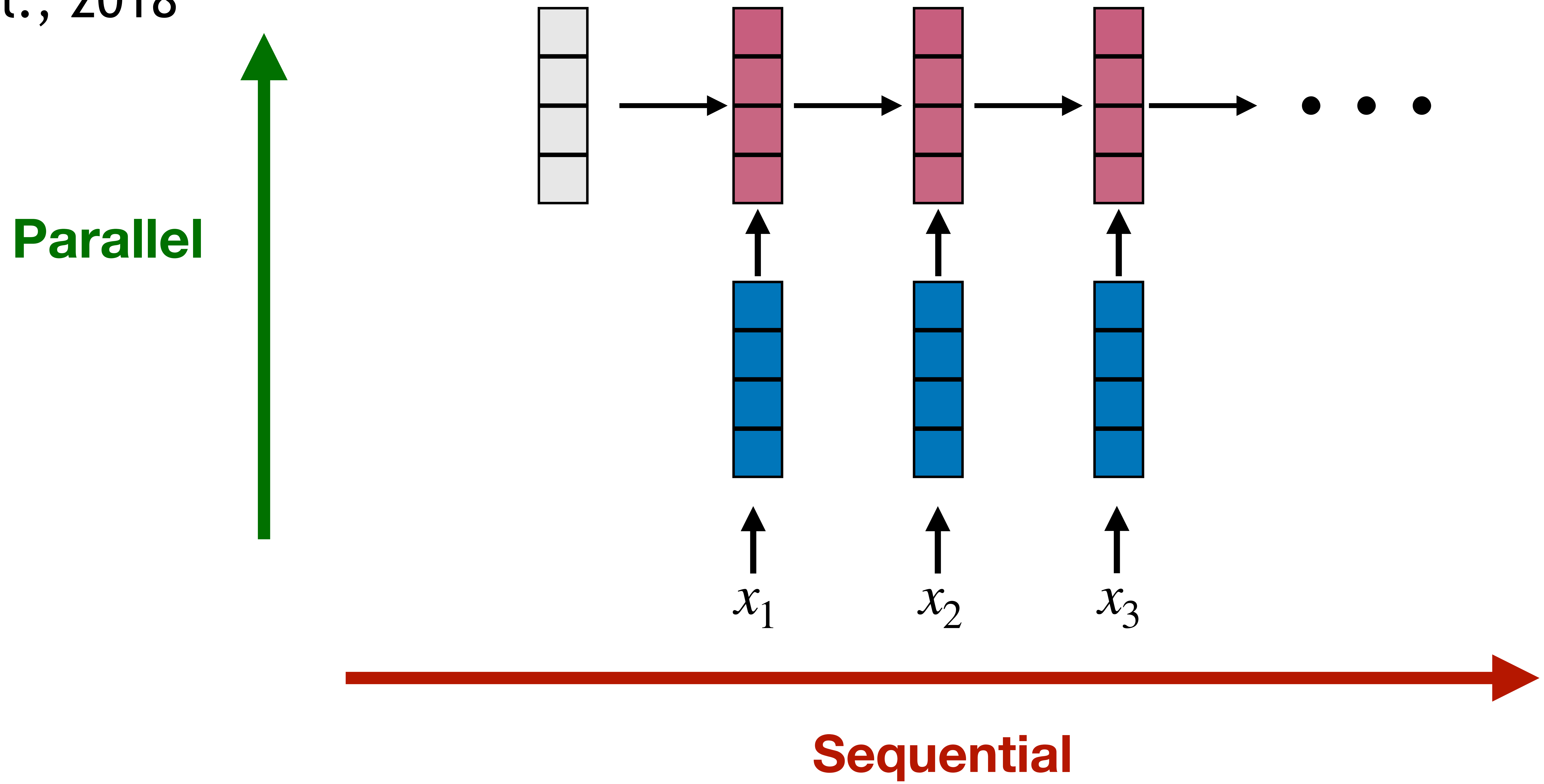
Problems of Simple RNN

- **No future contexts**
- **Hard to train**
  - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation
- **Limited size of hidden states**
  - Memory cost



Improving RNN efficiency

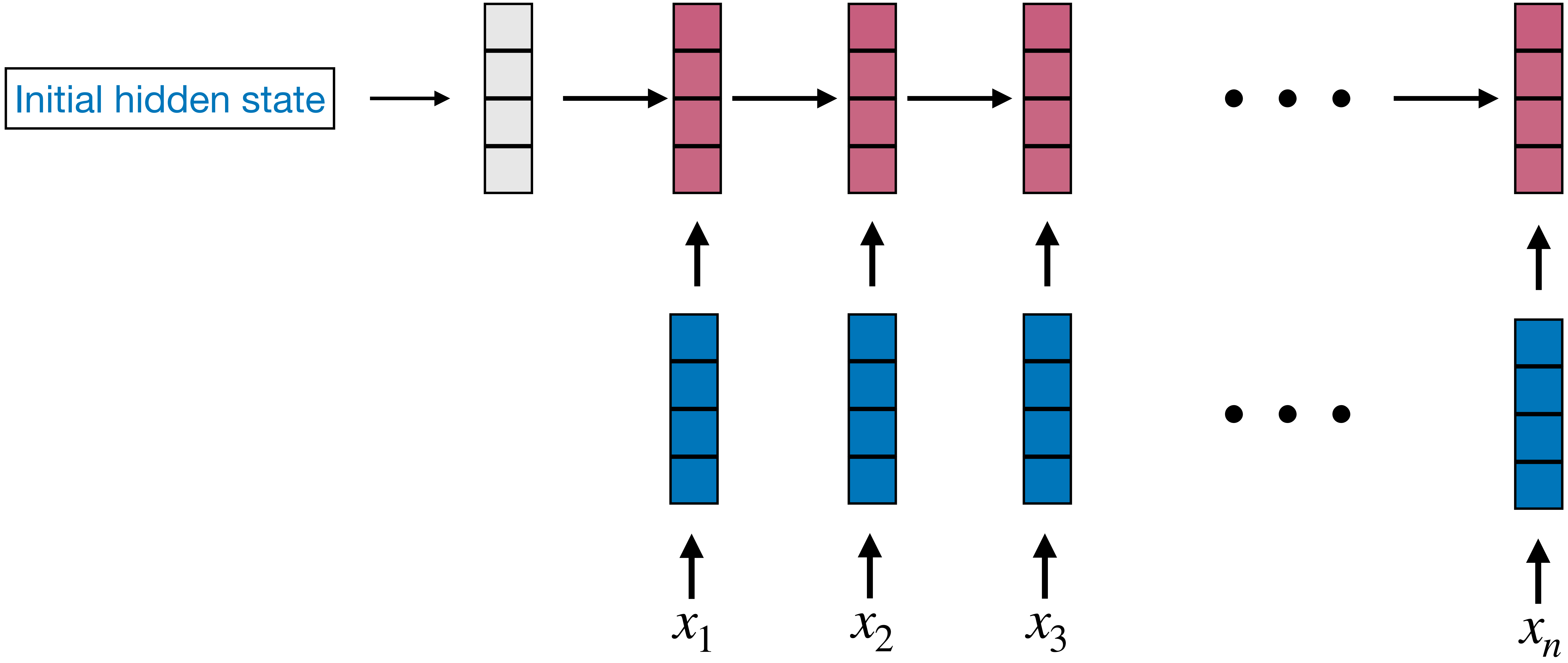
- **Simple Recurrent Unit**
  - Lei et al., 2018



$$h_t = \tanh(W_x \cdot x_t + \boxed{W_h \cdot h_{t-1}} + b) \longrightarrow V_h \odot h_{t-1}$$

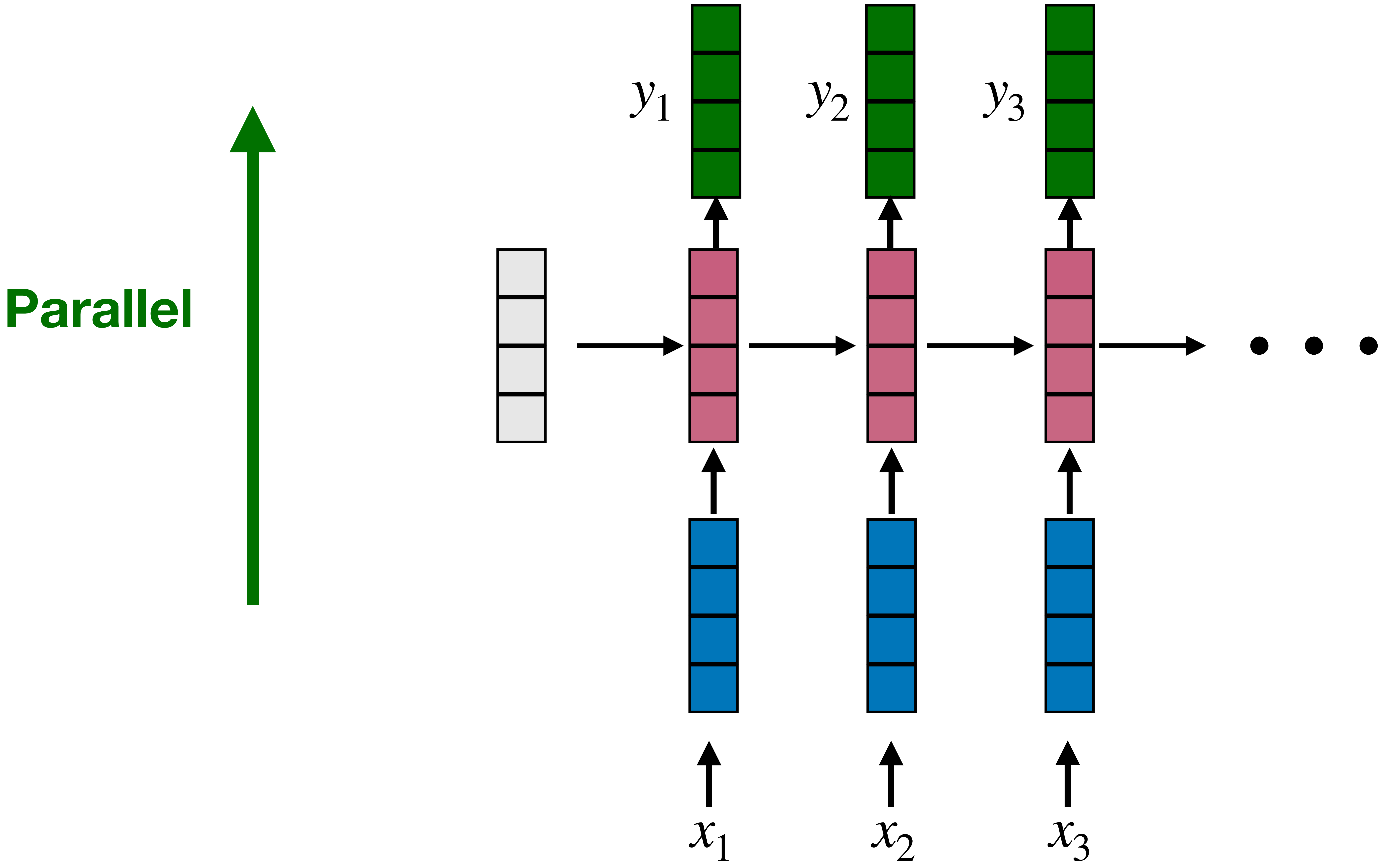
Problems of Simple RNN

- **No future contexts**
- **Hard to train**
  - Gradient varnishing/exploding
- **Inefficient**
  - Sequential computation
- **Limited size of hidden states**
  - Memory cost



Enlarging Hidden Size

- **Linear Recurrent Neural Networks**
  - Directly computing output y_t by skipping the explicit computation of h_t



Linear RNN

- Taking one dimension of x as an example, e.g. $x_t \in \mathbb{R}$

- Expanding x_t to n dimensions

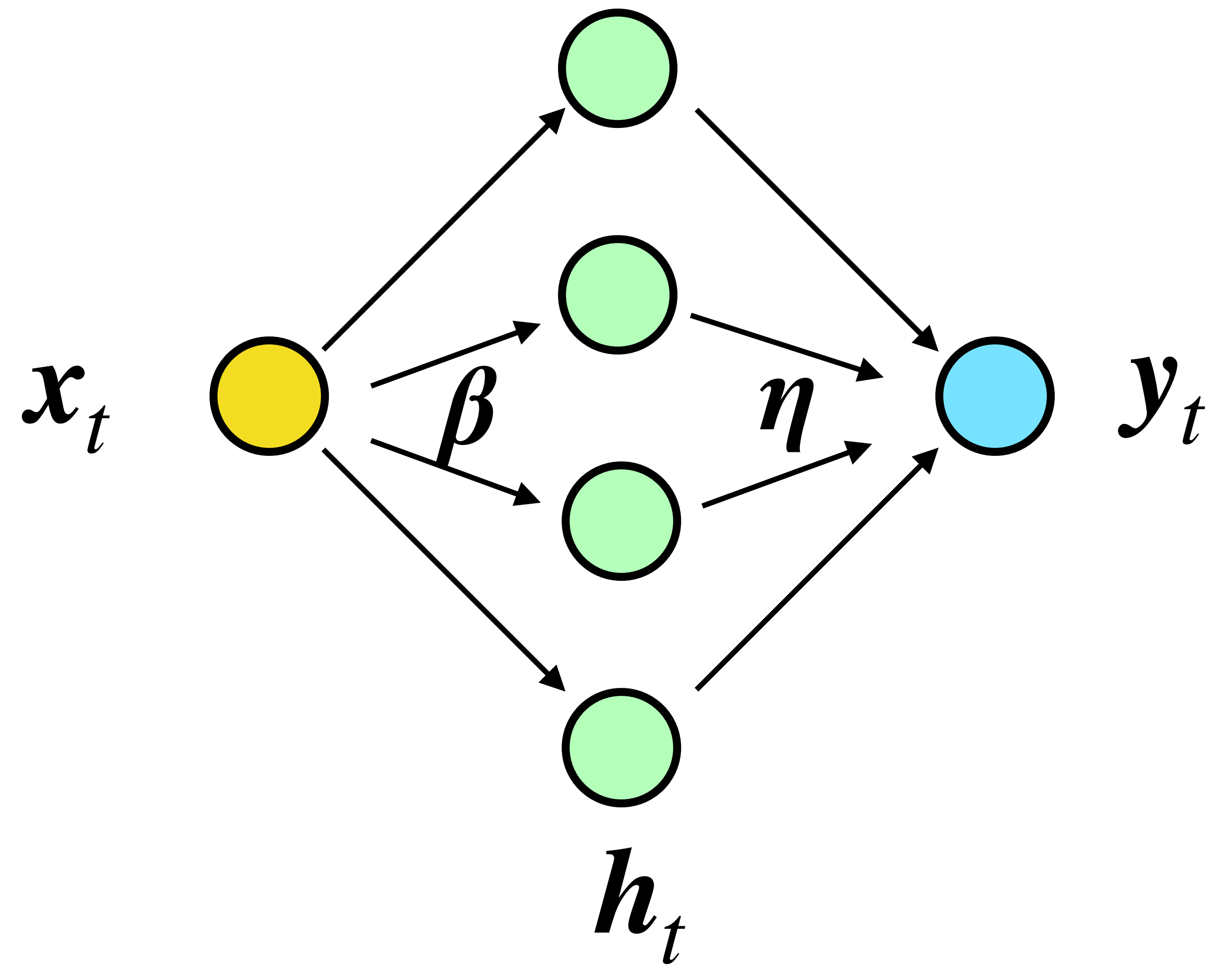
$$\mathbf{u}_t = \boldsymbol{\beta} x_t \in \mathbb{R}^n$$

- Applying linear operation on hidden state

$$\mathbf{h}_t = A\mathbf{h}_{t-1} + \mathbf{u}_t = A\mathbf{h}_{t-1} + \beta x_t \in \mathbb{R}^n$$

- Mapping the h-dimensional vector back to 1 dimension

$$\mathbf{y}_t = \boldsymbol{\eta}^T \mathbf{h}_t \in \mathbb{R}$$



Linear RNN

- **Unrolling the computation of** y_t

$$\mathbf{h}_t = \boldsymbol{A}\mathbf{h}_{t-1} + \mathbf{u}_t = \boldsymbol{A}\mathbf{h}_{t-1} + \beta x_t \in \mathbb{R}^n$$

$$\mathbf{y}_t = \boldsymbol{\eta}^T \mathbf{h}_t \in \mathbb{R}$$

$$\mathbf{h}_1 = \boldsymbol{A}\mathbf{h}_0 + \beta x_1$$

$$\mathbf{y}_1 = \boldsymbol{\eta}^T \mathbf{h}_1 = (\boldsymbol{\eta}^T \boldsymbol{A})\mathbf{h}_0 + (\boldsymbol{\eta}^T \beta)x_1$$

$$\mathbf{h}_2 = \boldsymbol{A}\mathbf{h}_1 + \beta x_2 = \boldsymbol{A}^2\mathbf{h}_0 + \boldsymbol{A}\beta x_1 + \beta x_2$$

$$\begin{aligned}\mathbf{y}_2 &= \boldsymbol{\eta}^T \mathbf{h}_2 \\ &= (\boldsymbol{\eta}^T \boldsymbol{A}^2)\mathbf{h}_0 + (\boldsymbol{\eta}^T \boldsymbol{A}\beta)x_1 + (\boldsymbol{\eta}^T \beta)x_2\end{aligned}$$

$$\vdots$$

Reading Materials

- **Relavant Papers**
  - BLSTM-CNNs-CRF