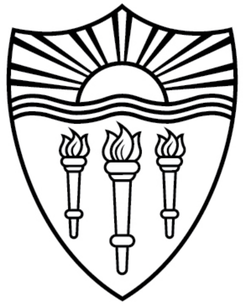
CSCI 544: Applied Natural Language Processing

# Deep Neural Networks for Structured Prediction

Xuezhe Ma (Max)



USC University of Southern California

# Logistical Points: Team Projects

- **Goal**
  - An opportunity to apply NLP knowledge on an in-depth application and expand mastery of using NLP in a focused data
- **Two types of projects**
  - Research oriented: select an interesting problem and try to solve it using NLP
    - Selecting an existing technique and come up with an improvement
    - Selecting an existing technique and try to used it in unexplored languages or domains
    - Selecting an existing technique in a non-NLP domain, e.g. computer vision, and try to adopt it to NLP
    - Selecting a classic research problem in NLP and try to propose new method to address it
  - Implementation oriented: select an NLP paper and try to re-implement the proposed method
    - Reproducing all the reported results in the paper
    - Exploring different hyper-parameters and/or modules.
- **A good proposal**
  - Motivation: a clear description of the problem/task you want to solve
  - Literature Review: a comprehensive survey of related works
  - Proposed methods & experiments: experimental setup
  - Plan: the milestones to check the progress

# Recap: Structured Prediction

- **Tasks**
  - Sequence Labeling: sequential structures
  - Syntactic Parsing: tree structures
- **Models**
  - Sequence Labeling:
    - HMMs
    - MEMMs
    - CRFs
  - Syntactic Parsing:
    - PCFGs
    - Graph-based Dependency Parsing
    - Transition-based Dependency Parsing

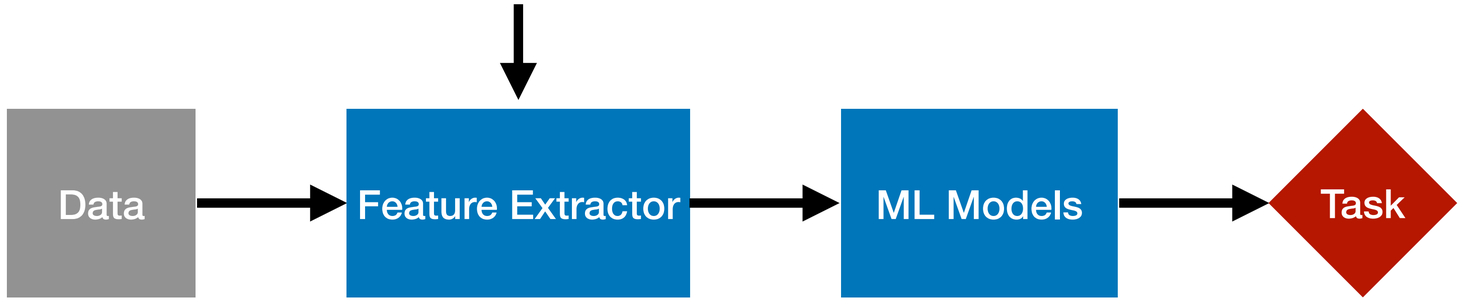
# Traditional Machine Learning: Framework

Feature Engineering

Domain Knowledge

$$f_3 = \begin{cases} 1, & \text{if } x_{i-1} = \text{USC}, x_i = \text{is}, x_{i+1} = \text{in}, s_{i-1} = \text{NNP}, \text{ and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$

$$f_4 = \begin{cases} 1, & \text{if } x_i \text{ ends in } ing, s_{i-1} = \text{NNP}, \text{ and } s_i = \text{VBZ} \\ 0, & \text{otherwise} \end{cases}$$




POS tagging

NNP VBZ IN NNP

**USC is in California**

**CRF:**

$$p(s_1, \ldots, s_m \mid x_1, \ldots, x_m) = \frac{\prod_{j-1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s_{j-1}, s_j))}{\sum_{s'_1, \ldots, s'_m \in \mathbb{S}} \prod_{j=1}^{m} \exp(v \cdot f(x_1, \ldots, x_m, i, s'_{j-1}, s'_j))}$$

# Traditional Machine Learning: Features vs. Models

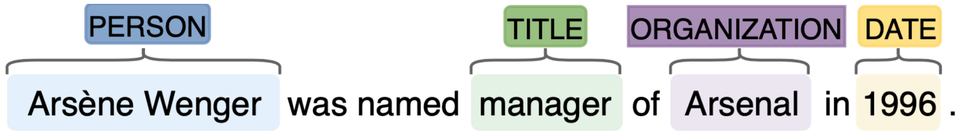
- **Features and Models are highly correlated**
  - More flexible features requires more complex models
  - Graph-based Dependency Parsing
    - 1st-order model: $O(n^3)$
    - Higher-order models: $O(n^5)$ for 4th-order model
- **Two directions of research**
  - Designing more informative features
    - Non-binary features?
    - External resources: wordnet, wikipedia, …
  - Reducing model complexity
    - Approximated learning?
    - Approximated decoding?
    - Reward-augmented learning?
    - …

# Traditional Machine Learning: Problems

- **Model performance stops increasing**
  - POS Tagging: 97.3% accuracy
  - Named Entity Recognition: 91.2% F1
  - Dependency Parsing: 93.4% UAS
  - Coreference Resolution: < 70%
  - …
- **Feature engineering is painful**
  - Variations of data
  - Sparsity
  - Model efficiency
  - …

# Problems of Hand-Crafted Feature Engineering

## Tasks


## Languages

**Max is working at USC**

**麦克斯在南加大工作**

## Sparsity

**I saw a girl with a telescope**

4-gram of words: $|V|^4$

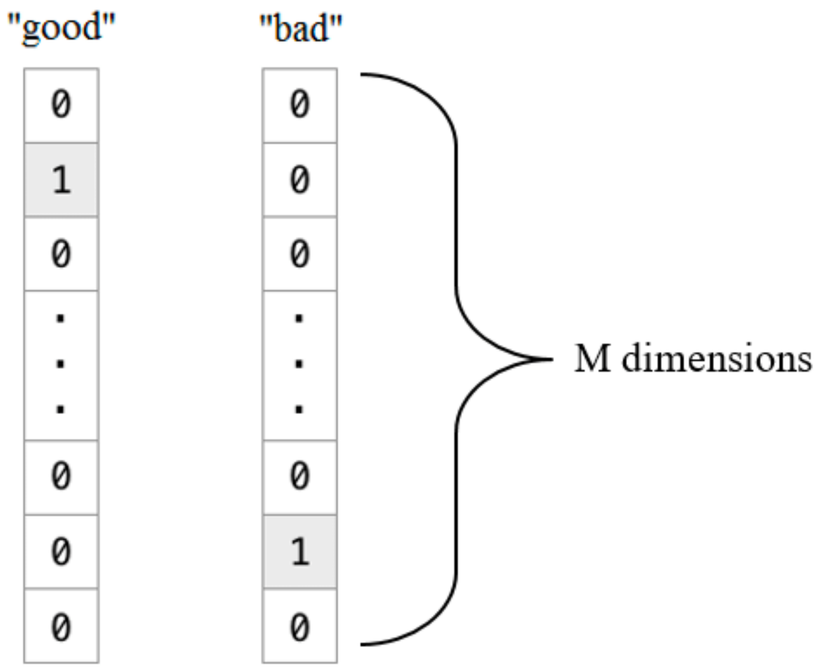
## Model Efficiency

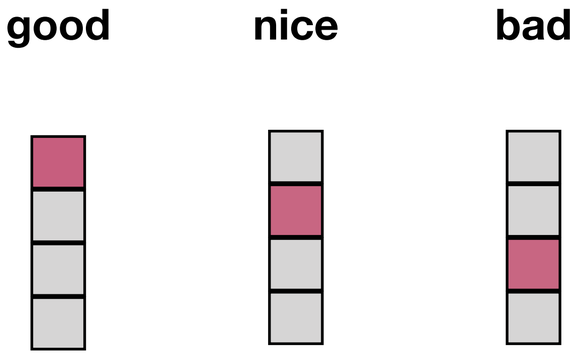
For dependency parser, more than 90% time spent on feature extraction (Chen et at., 2014)

# Problems of Binary Features

## One-hot binary vectors

A vocabulary of $M$ words

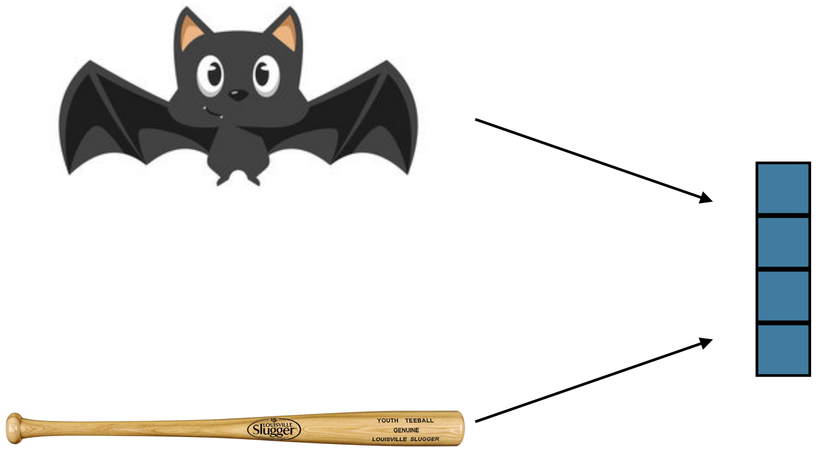



$$\text{dist}(\textbf{good}, \textbf{nice}) = \text{dist}(\textbf{good}, \textbf{bad})$$

# Compositionality

**Bat**



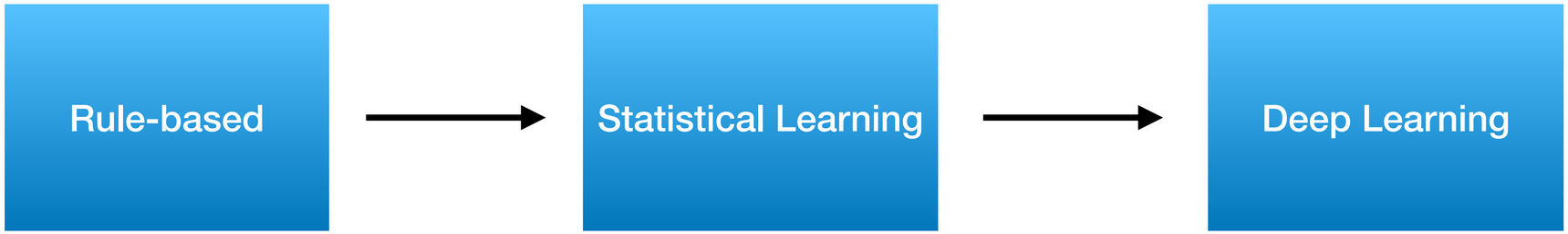

**hit with bat**

semantic compositionality is NOT linear combination

# Feature Representations: What vs. How

What should we do?

# Evolution of NLP




**1950** **1990** **2010**

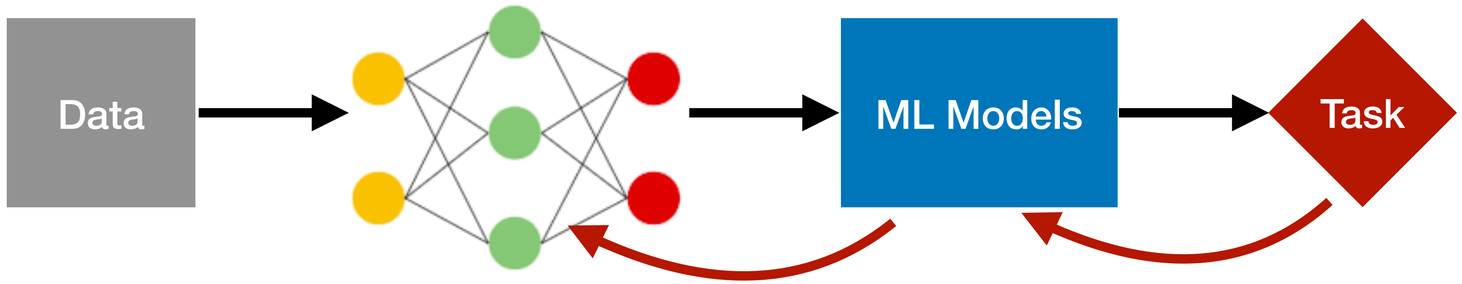
manually design rules

- manually design features
- learn rules

- manually design neural networks
- learn features

more learning, less human knowledge

# Deep Learning System



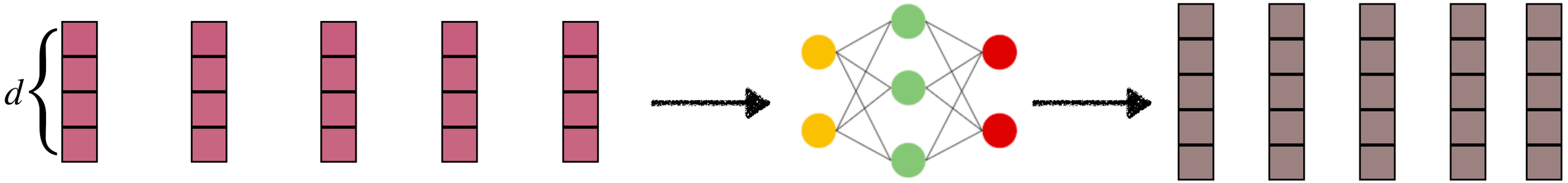

Automatically learning feature representations for the end task

Deep Learning a.k.a Representation Learning

# Our Goal: Sentence Representations

- **One hidden vector for each word**

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



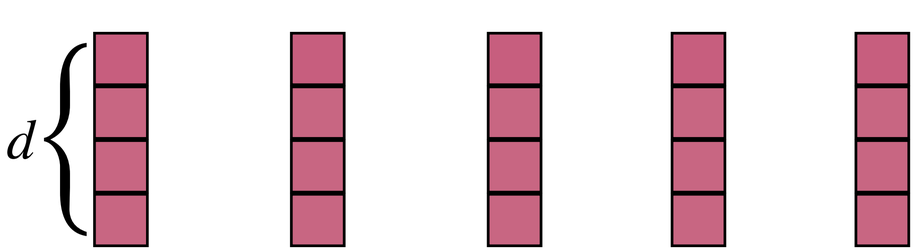

$Y \in \mathbb{R}^{n \times d_2}$

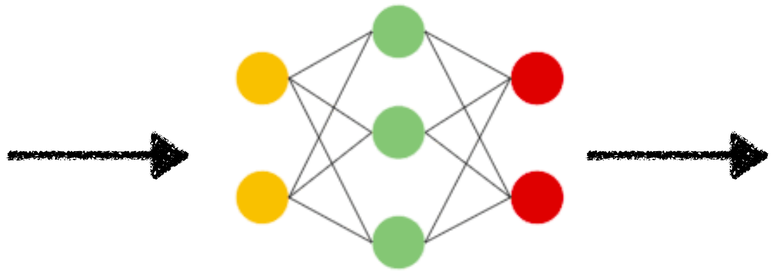
**Max is working at USC**

# Our Goal: Sentence Representations

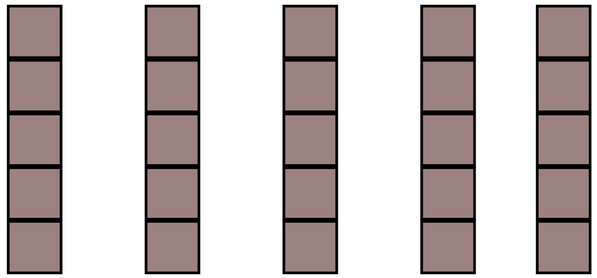
- **One hidden vector for each word**

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



$Y \in \mathbb{R}^{n \times d_2}$



**Max is working at USC**

why not a single vector?

"You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"
— Ray Mooney

# Outline

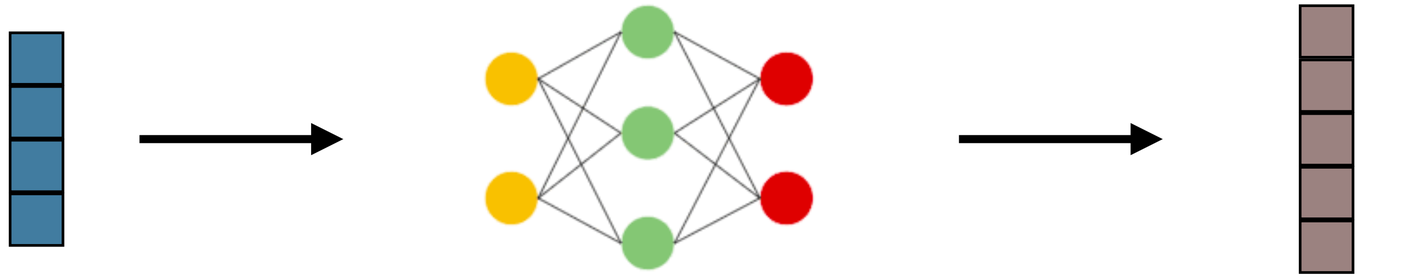
- **A Brief Review of Some Neural Networks**
  - Feedforward Neural Networks
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)
- **Neural Networks for Structured Prediction**
  - Sequence Labeling
  - Dependency Parsing

# Neural Networks

# Feedforward Neural Networks

- **One layer of Feedforward neural network**

$$f : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$$



$X \in \mathbb{R}^{d_1}$

$Y \in \mathbb{R}^{d_2}$

$$Y = f(X) = \sigma(W \cdot X + b)$$

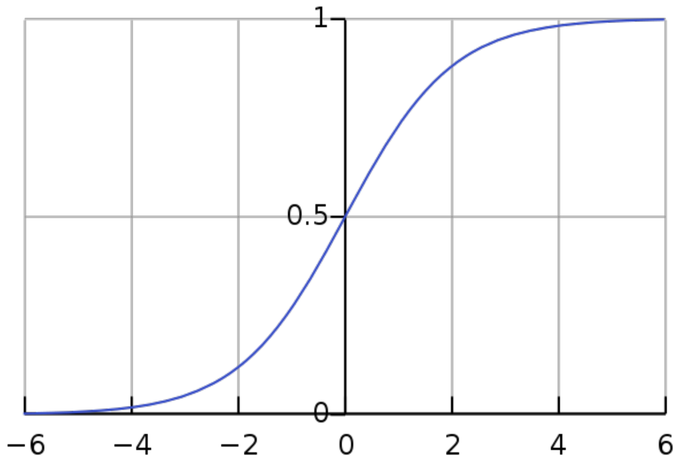
non-linear function

parameters

# Non-linear Functions

- **A.k.a Activation functions**

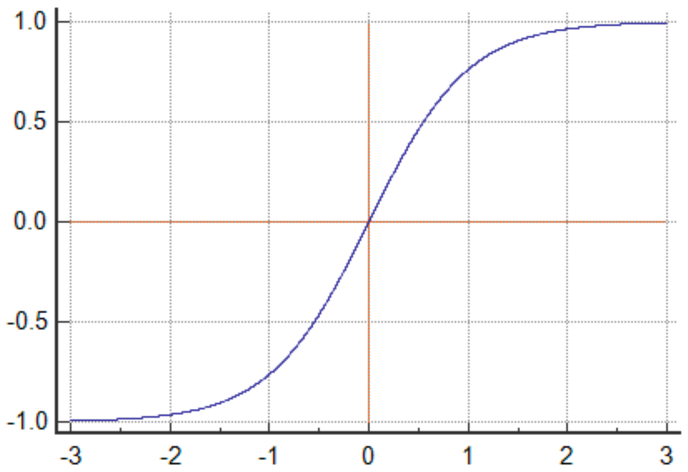
Sigmoid function




$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

**range** $(0,1)$

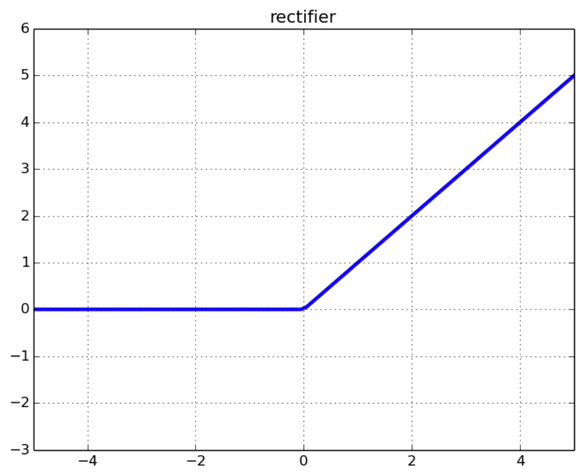
tanh function




$$\sigma(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

**range** $(-1,1)$

ReLU function




$$x = \max(0,\ x)$$

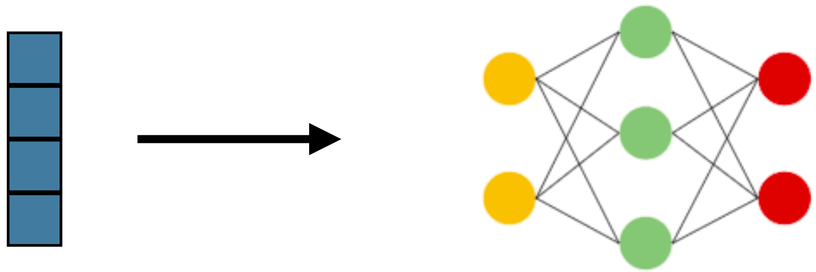
**range** $(0,1)$

# Feedforward Neural Networks

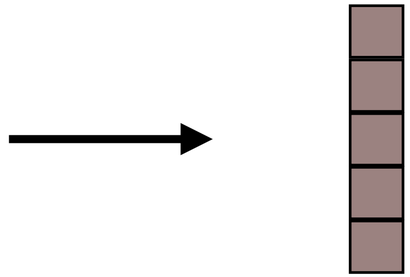
- **One layer of Feedforward neural network**

$$f : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$$

$d_1$ and $d_2$ are pre-defined hyper-parameters



$$X \in \mathbb{R}^{d_1}$$



$$Y \in \mathbb{R}^{d_2}$$

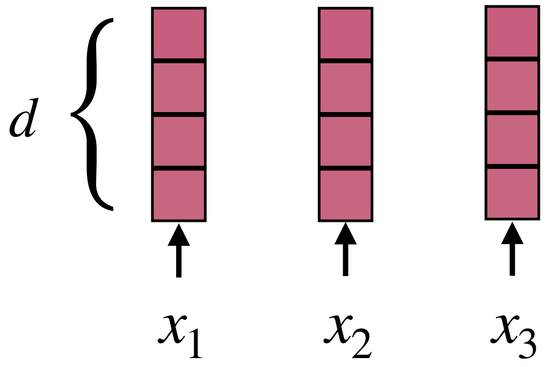
$$Y = f(X) = \sigma(W \cdot X + b)$$

non-linear function

parameters

# Modeling Sentences with Various Lengths

- **Feedforward Neural Networks cannot model sentences with various lengths.**

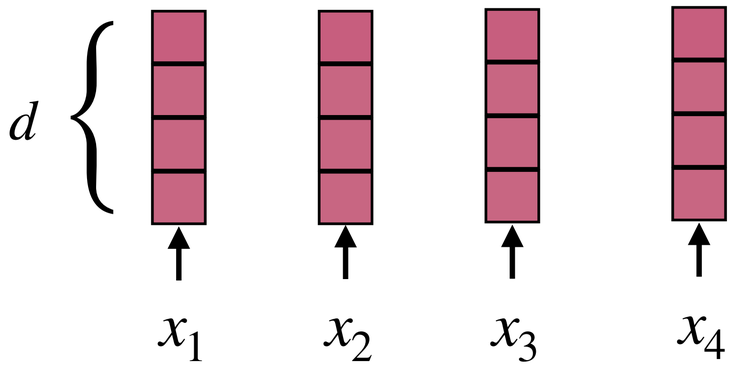



$$f : \mathbb{R}^{3\times d} \rightarrow \mathbb{R}^{d_2}$$

$$X \in \mathbb{R}^{3\times d}$$

# Modeling Sentences with Various Lengths

- **Feedforward Neural Networks cannot model sentences with various lengths.**




$$f : \mathbb{R}^{3\times d} \rightarrow \mathbb{R}^{d_2}$$

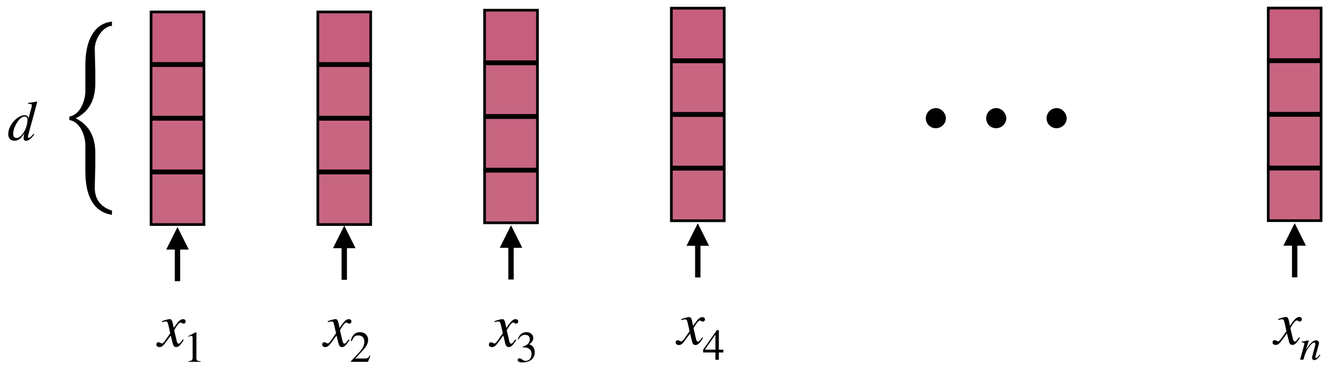
$$X \in \mathbb{R}^{4\times d}$$

# Convolutional Neural Networks (CNNs)

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

$$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$$

$k$ is the window size




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

# Convolutional Neural Networks (CNNs)

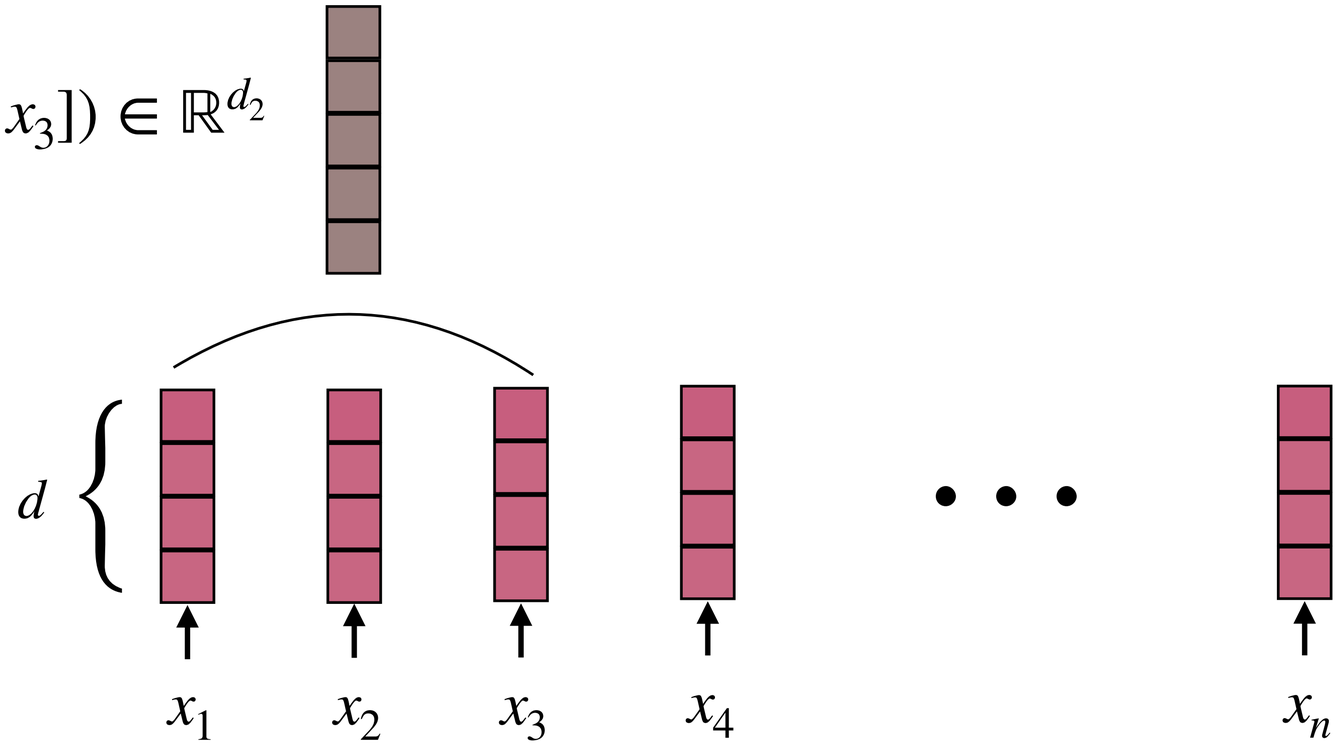
- **Basic Idea:** only model a segment of input with fixed window-size

$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$

$k$ is the window size

$k = 3$

$y_1 = f([x_1, x_2, x_3]) \in \mathbb{R}^{d_2}$




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

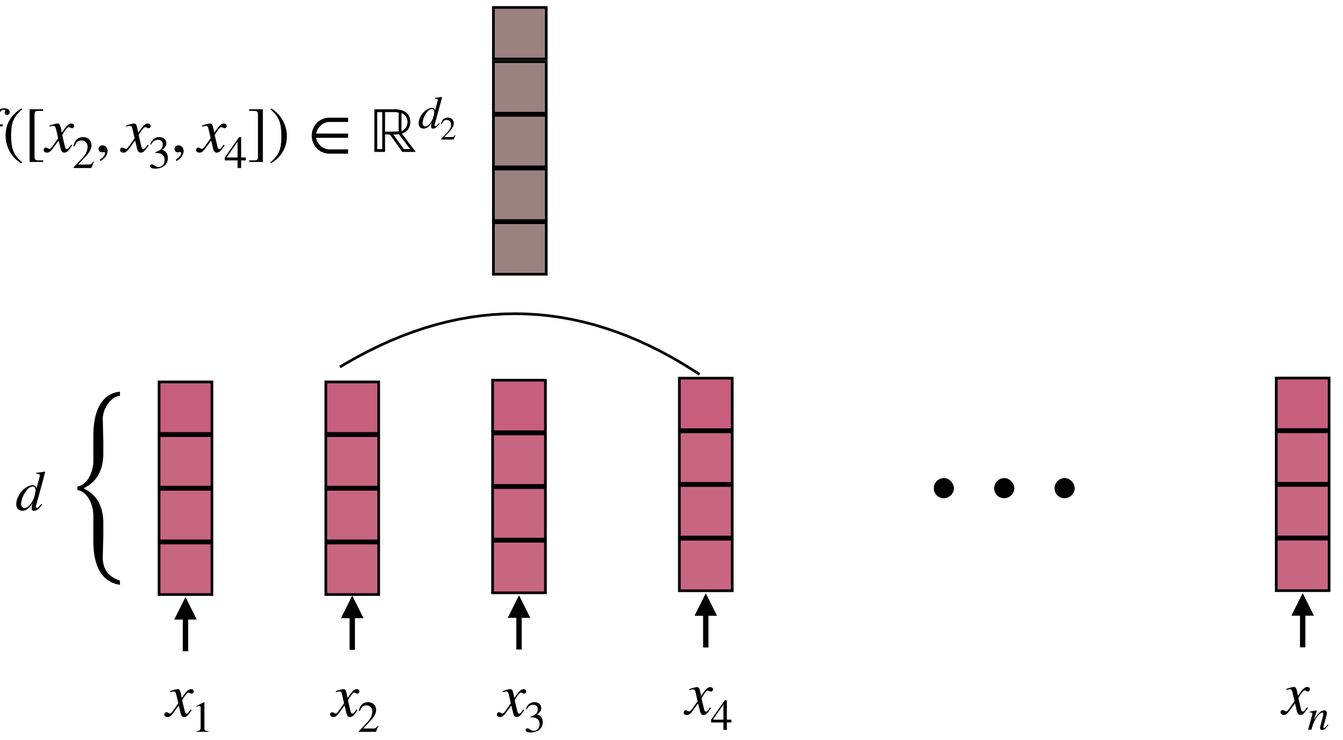
# Convolutional Neural Networks (CNNs)

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

$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$ $k$ is the window size

$k = 3$

$y_2 = f([x_2, x_3, x_4]) \in \mathbb{R}^{d_2}$




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

# Convolutional Neural Networks (CNNs)

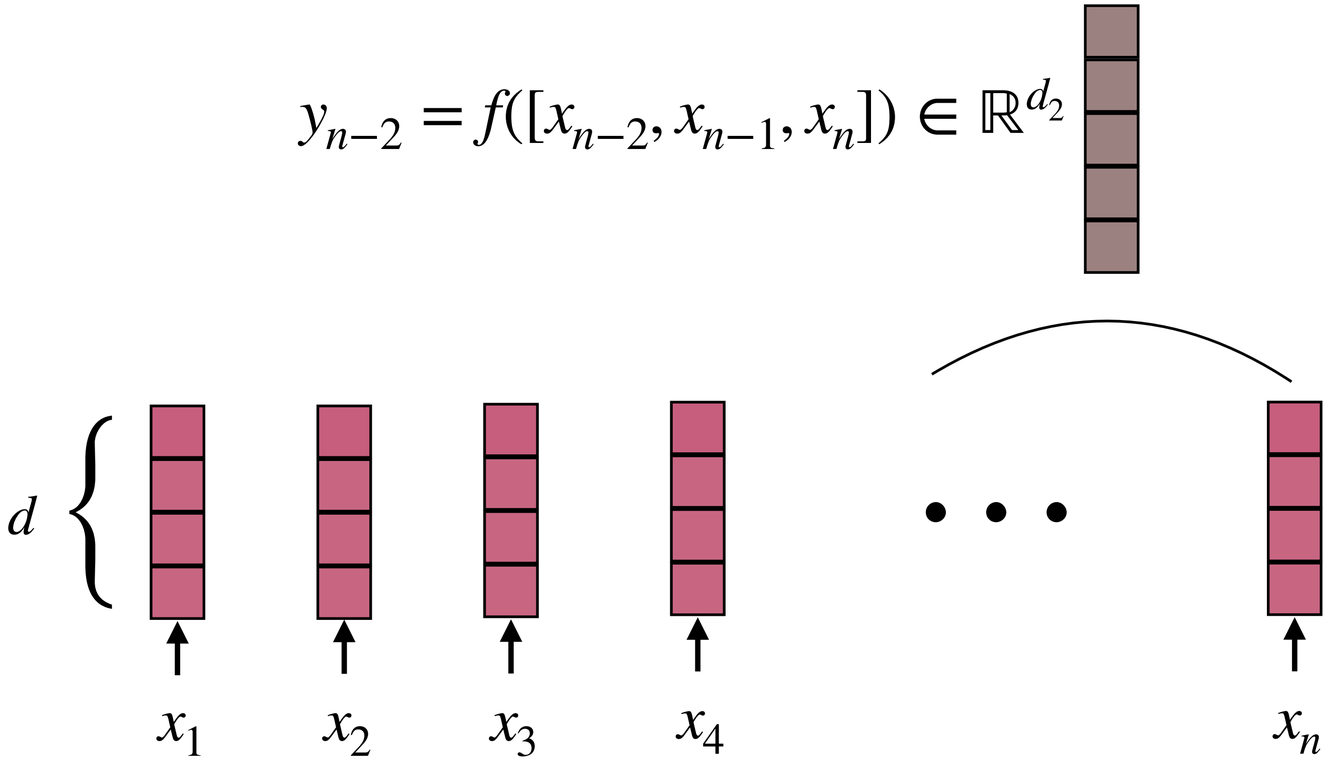
- **Basic Idea:** only model a segment of input with fixed window-size

$k = 3$

$$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$$

$k$ is the window size

$$y_{n-2} = f([x_{n-2}, x_{n-1}, x_n]) \in \mathbb{R}^{d_2}$$




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

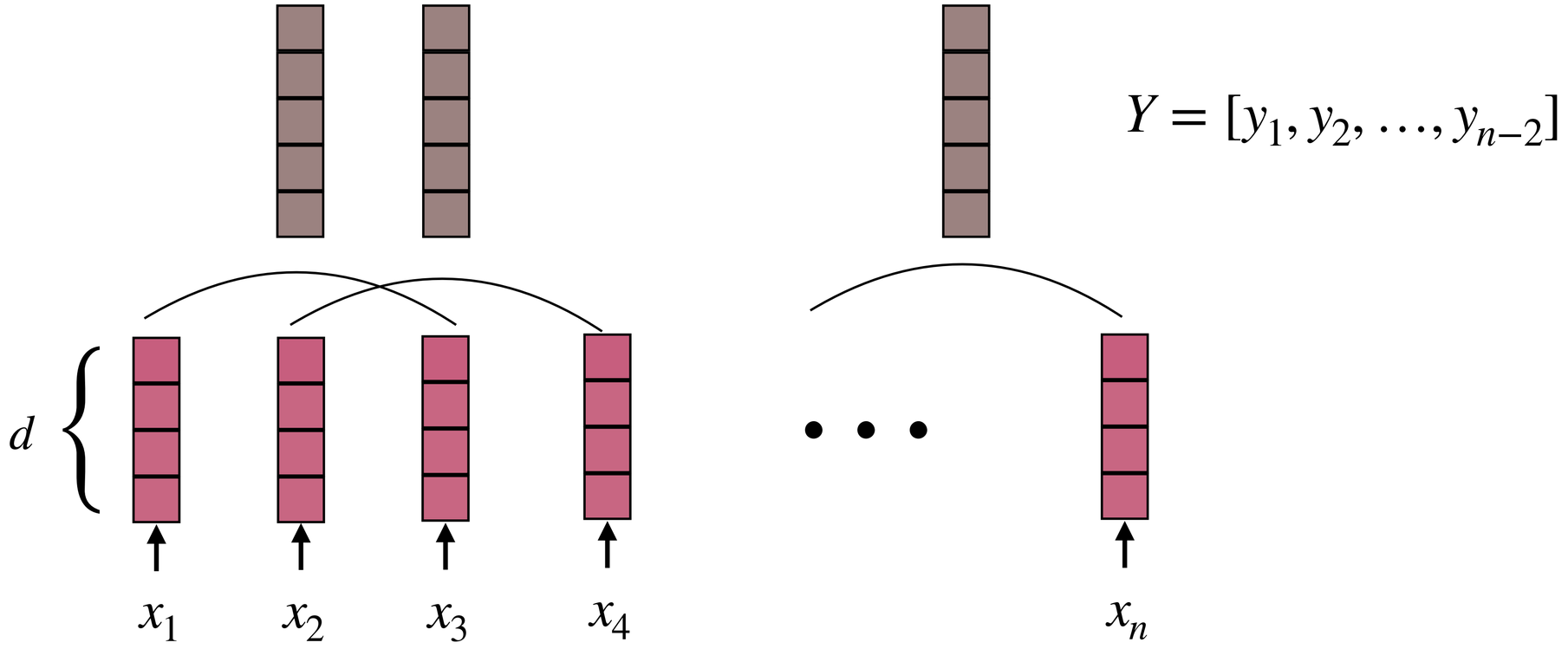
# Convolutional Neural Networks (CNNs)

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

$k = 3$

$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$

$k$ is the window size




$Y = [y_1, y_2, \ldots, y_{n-2}]$

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

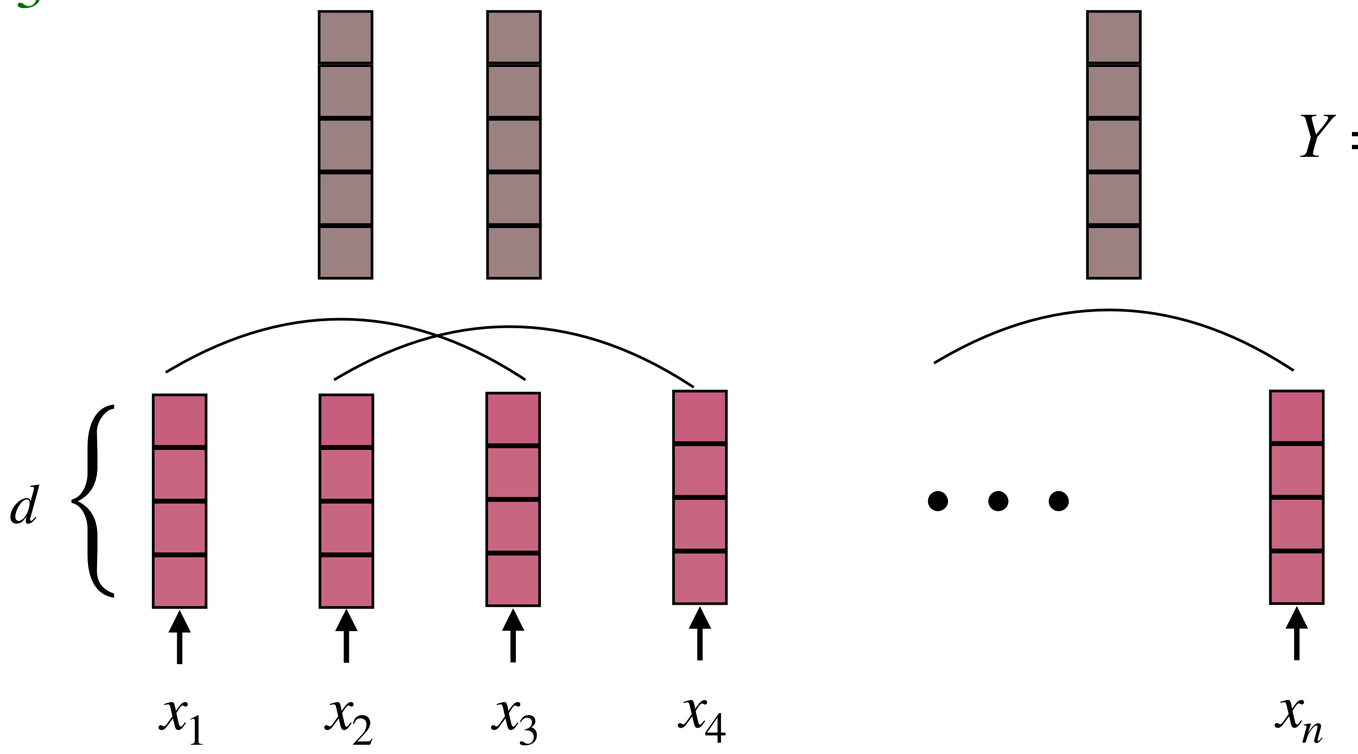
# Convolutional Neural Networks (CNNs)

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

$k = 3$

$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$

$k$ is the window size




$Y = [y_1, y_2, \ldots, y_{n-2}]$

The input X contains $n$ words
but only gets $n - 2$ output vectors

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

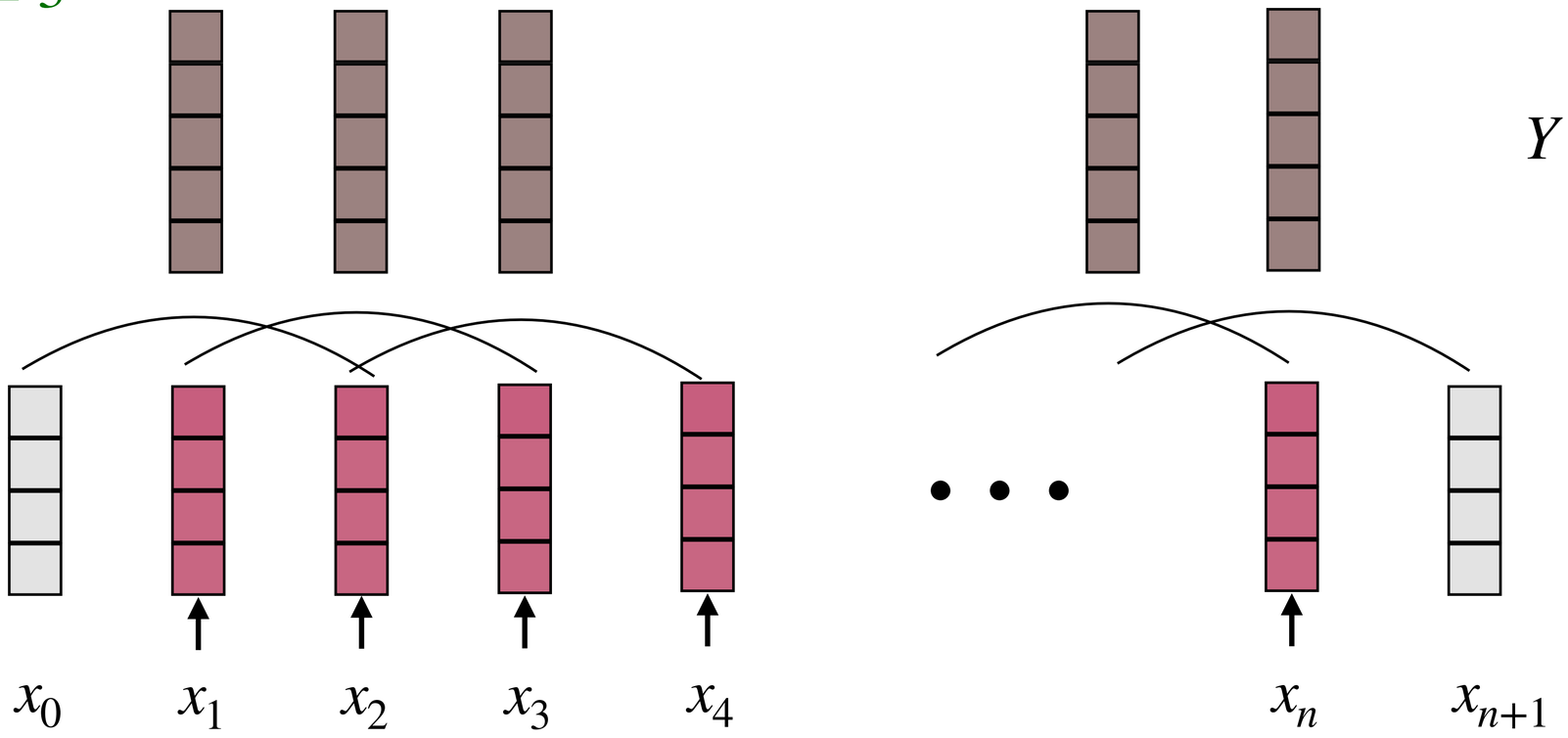
# Convolutional Neural Networks (CNNs)

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

$k = 3$

$f : \mathbb{R}^{k \times d} \rightarrow \mathbb{R}^{d_2}$

$k$ is the window size

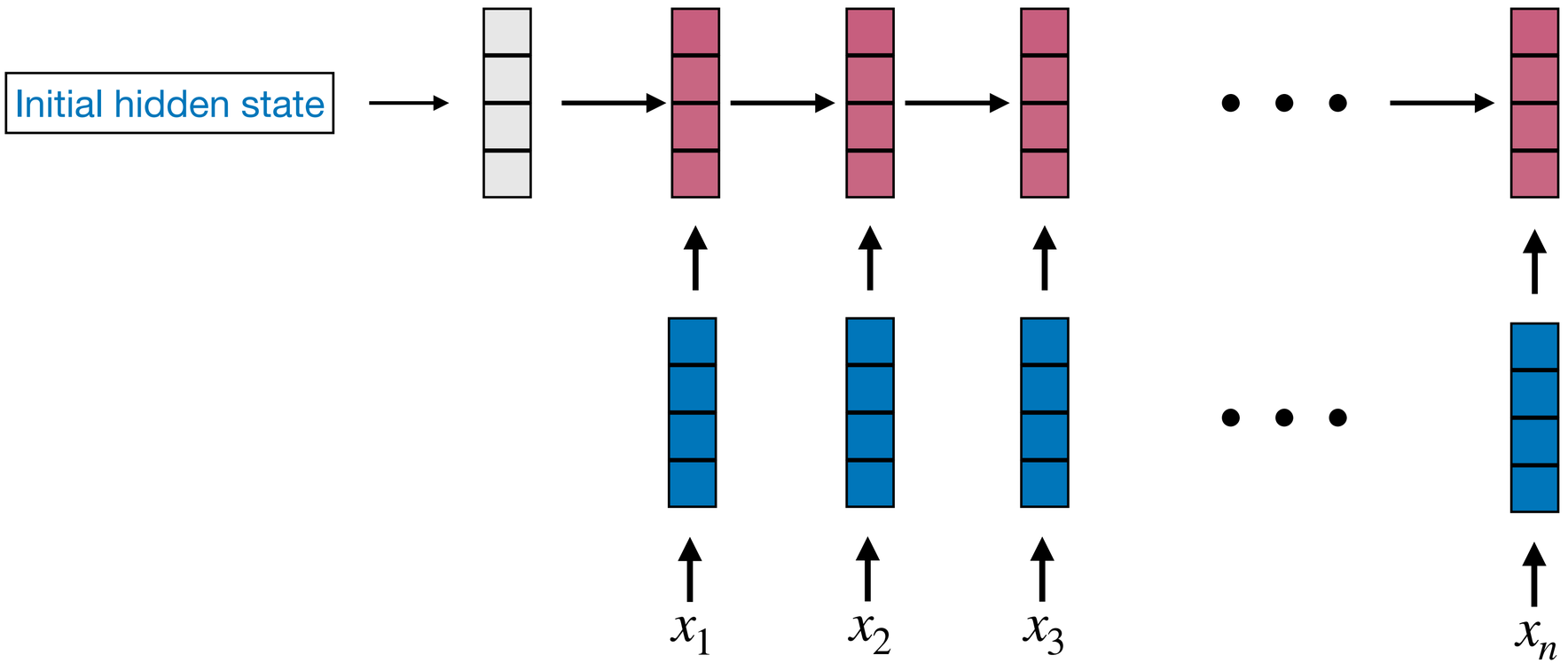



$Y = [y_1, y_2, \ldots, y_n] \in \mathbb{R}^{n \times d_2}$

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

# Neural Networks for Sentence Representations

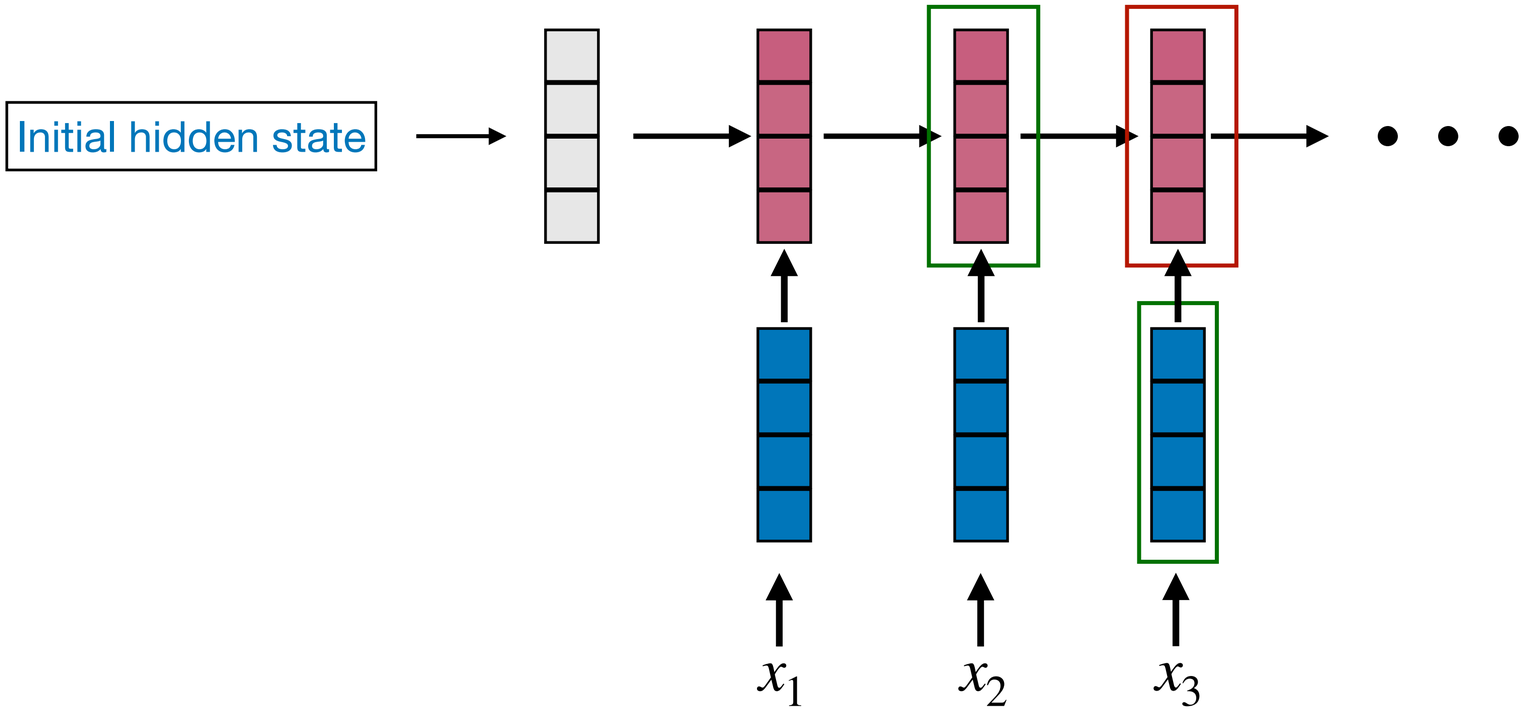
- **Recurrent Neural Networks (RNNs)**



# Neural Networks for Sentence Representations

- **Recurrent Neural Networks (RNNs)**

**Inherent Markov Property**



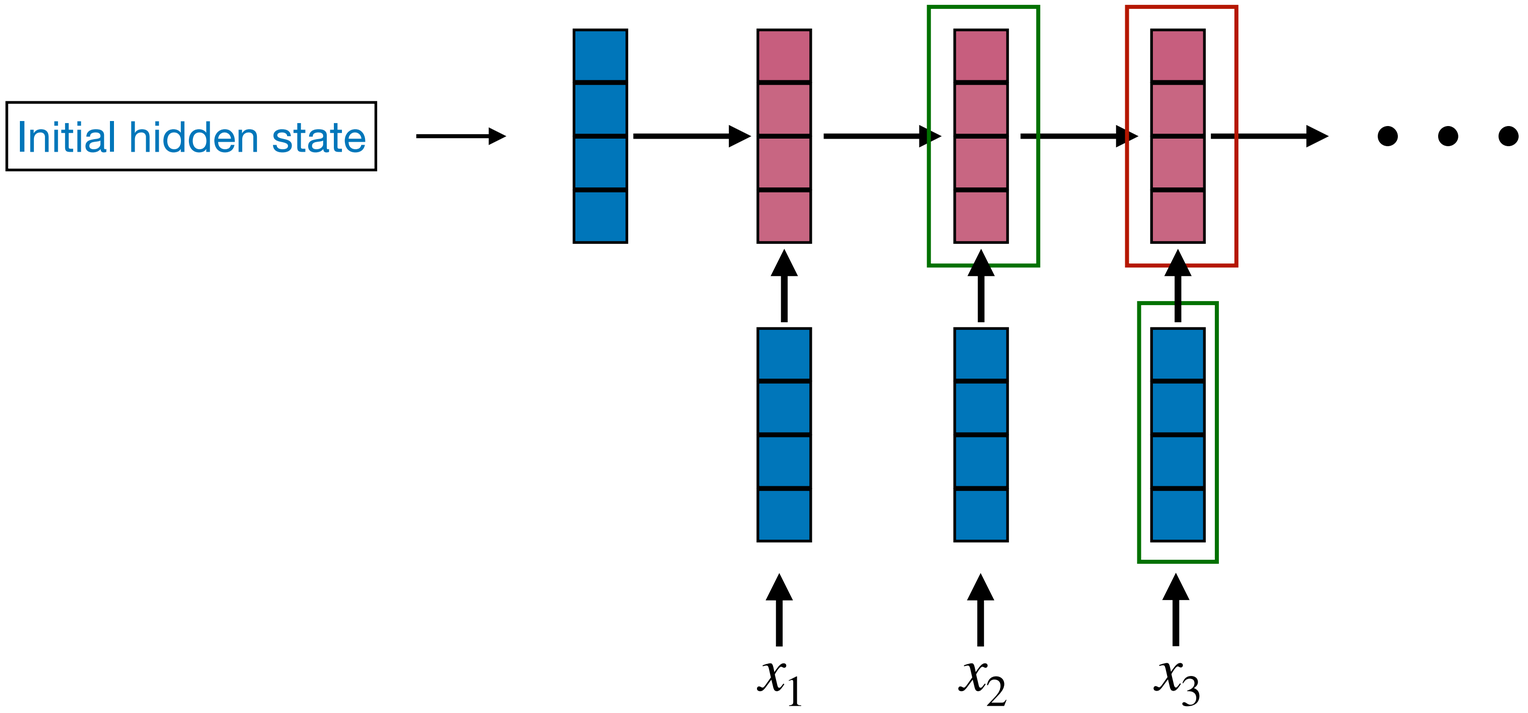
# Neural Networks for Sentence Representations

- **Recurrent Neural Networks (RNNs)**

**Inherent Markov Property**

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

Simple RNN



# 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**

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

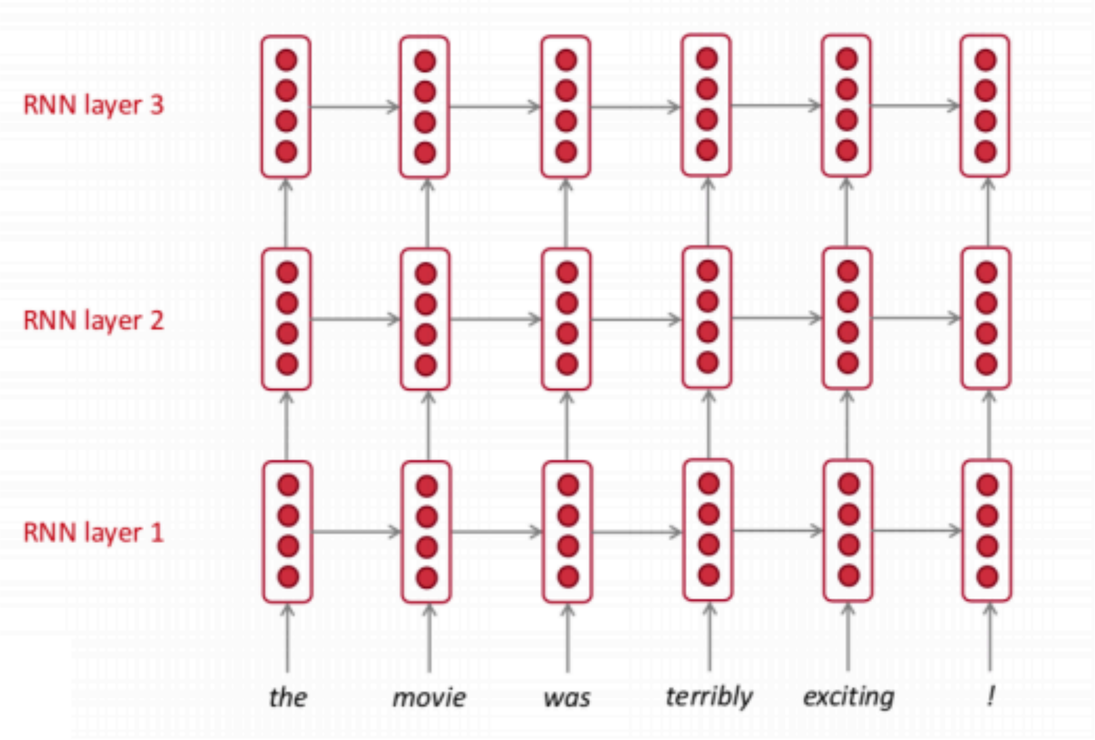
$$\mathbf{z}_t = \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z)$$

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

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

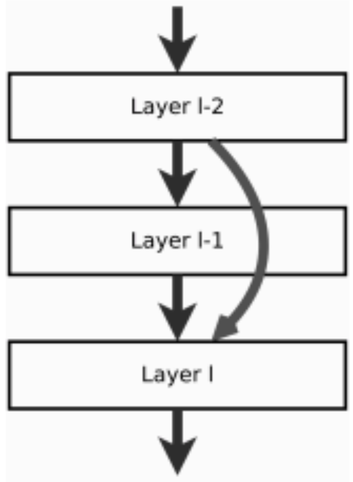
LSTM is more an art than a science

# Multi-layer RNNs




The hidden states from RNN layer $i$ are the inputs to RNN layer $i + 1$

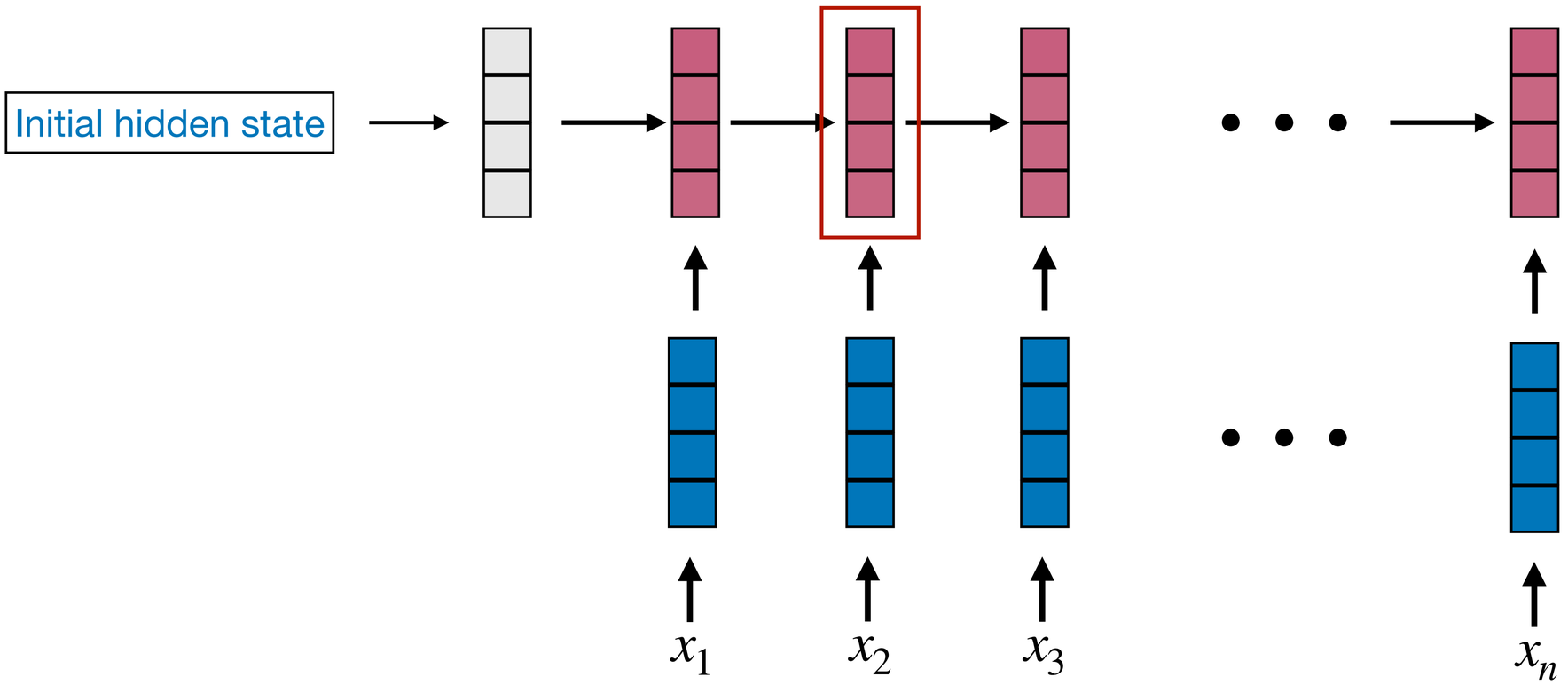
skip-connections



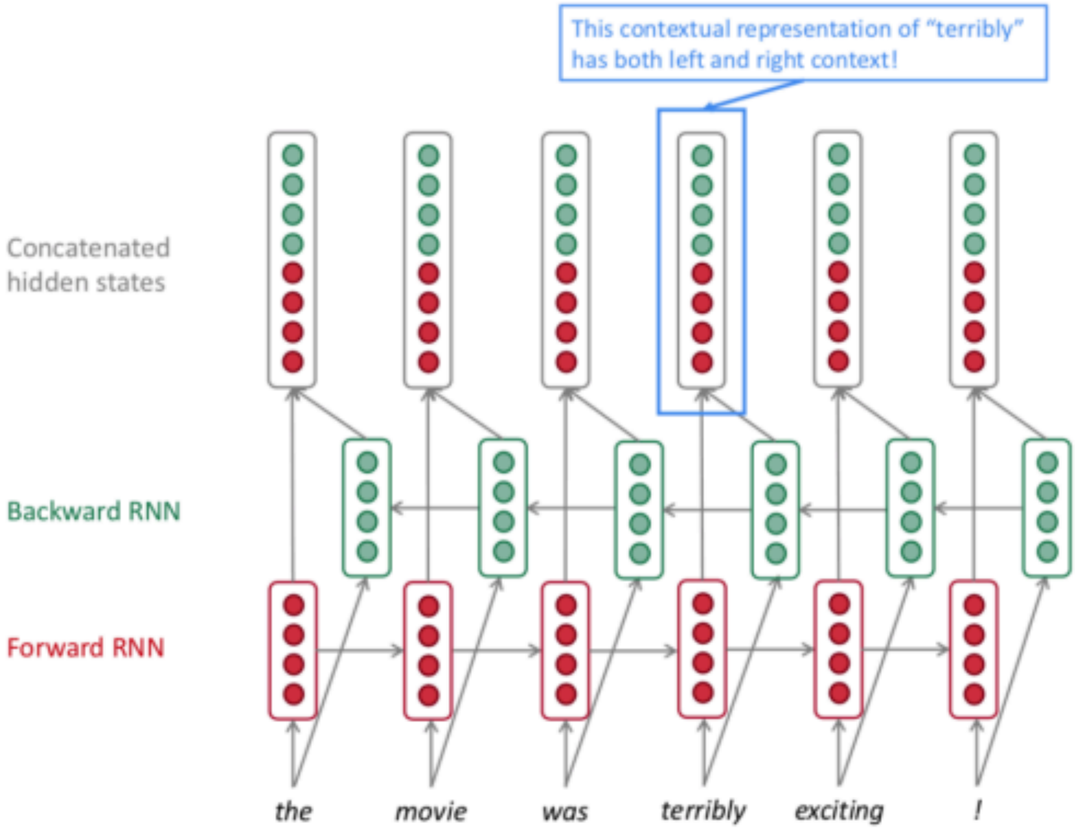

- In practice, using 2 to 4 layers is common (usually better than 1 layer)
- When the number of layers is large, we need to use skip-connections

# Bidirectional RNNs

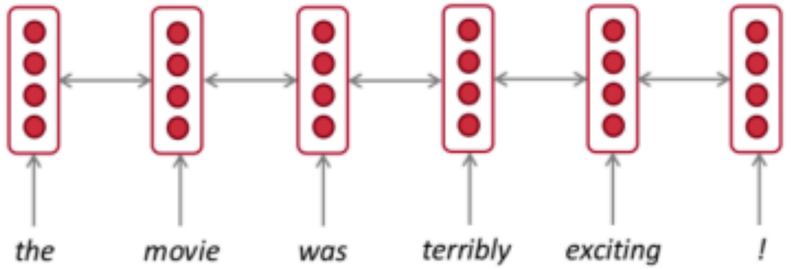
- **RNN cannot model future information**



# Bidirectional RNNs




=




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

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

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

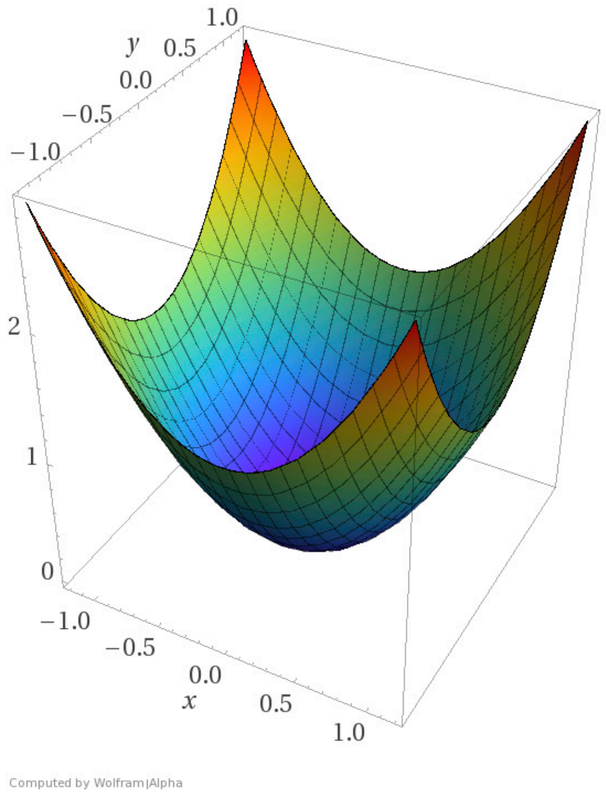
$$\mathbf{h}_t = [\overleftarrow{\mathbf{h}}_t, \overrightarrow{\mathbf{h}}_t] \in \mathbb{R}^{2h}$$

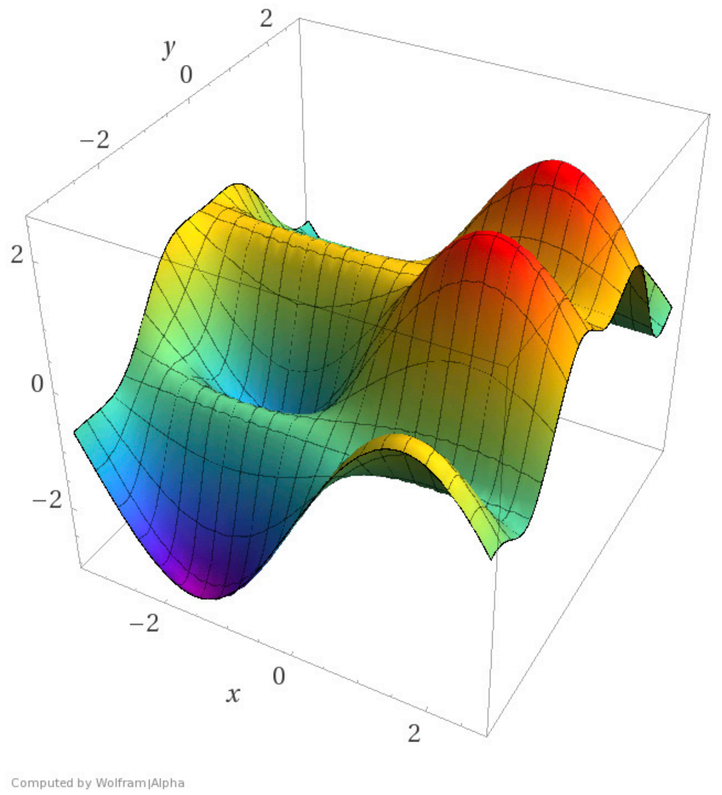
# Bidirectional RNNs

- 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

# Before Building Your DNNs

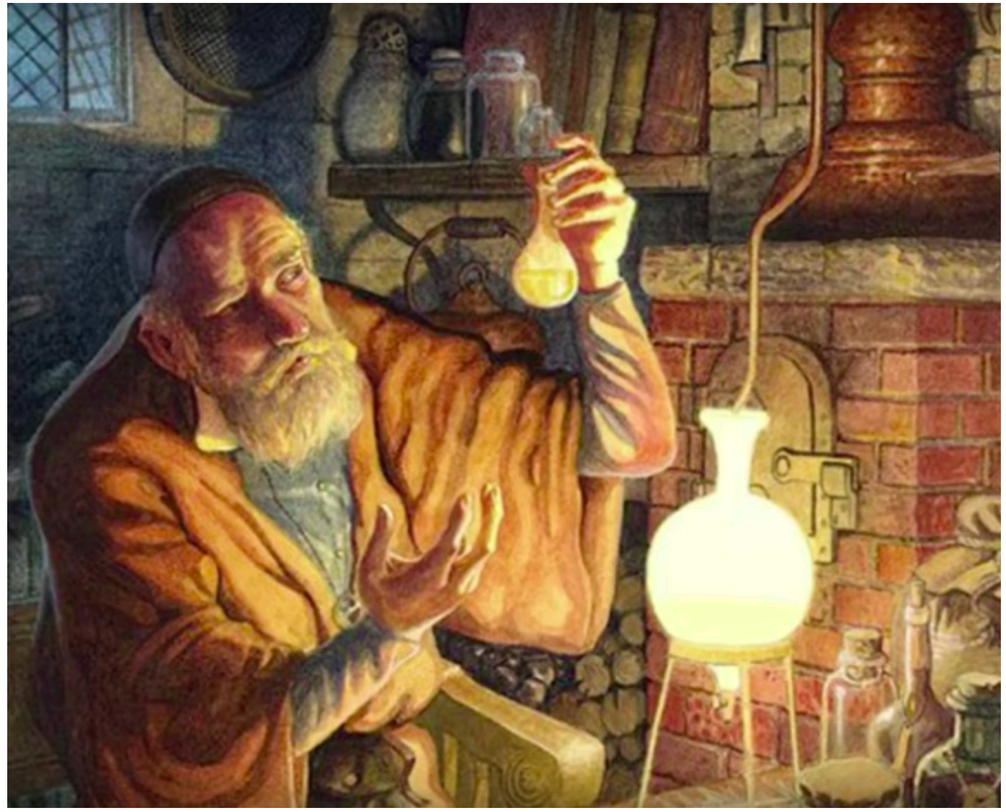
- **DNNs are non-convex**



# Before Building Your DNNs

- DNNs are non-convex
- No rigorous theoretical supports, but only empirical evidences



Rahimi@NIPS2017:
Has Deep Learning Become Alchemy?

炼丹

# Neural Networks for Sequence Labeling

# An Essential Question

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

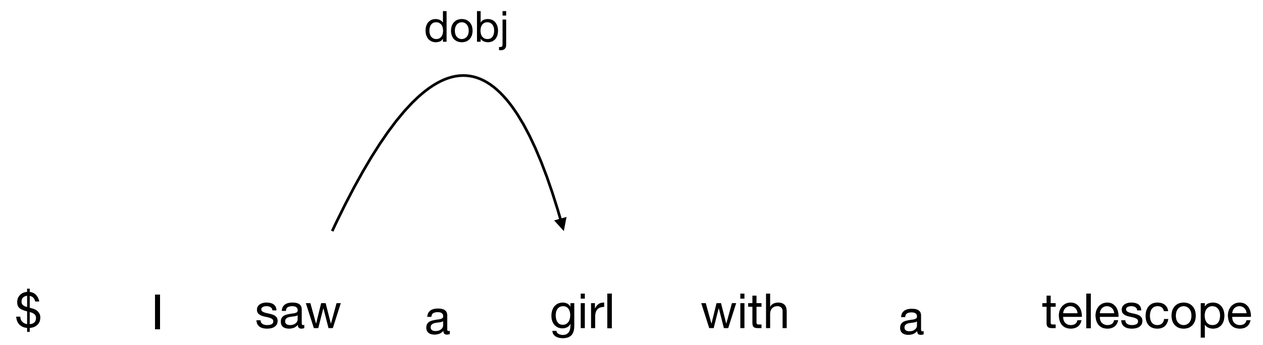

The cat sat on a mat

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

?

# An Essential Question

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




$$P(y \mid x_1, \ldots, x_n) = \prod_{e \in y} P(e \mid x_1, \ldots, x_n)$$

# An Essential Question

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

我 不 知道

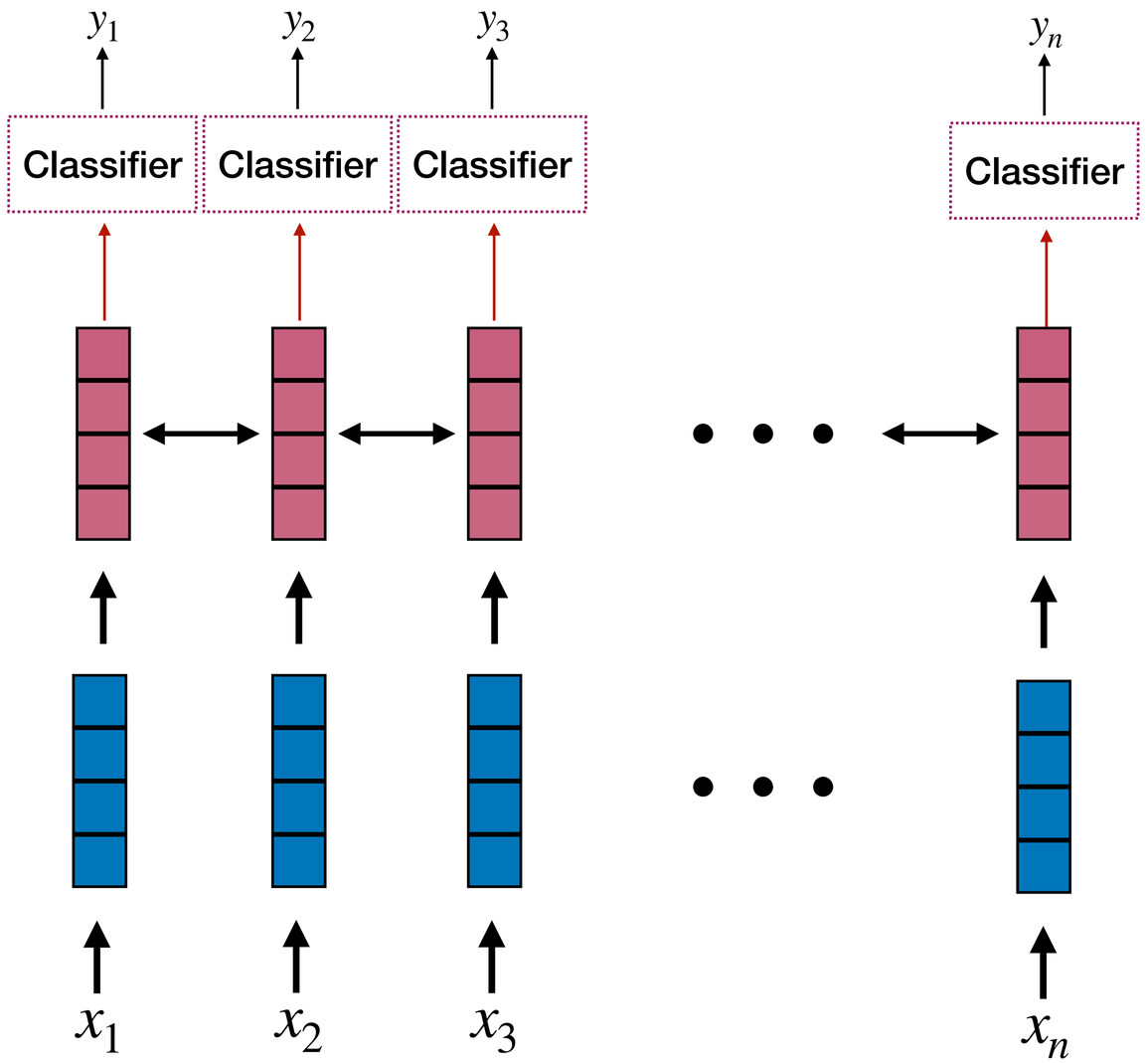
I don't know

I do not know

I have no idea

# Sequence Labeling

- **A simple bidirectional RNN model**




No structured models to consider dependencies between Y

# Sequence Labeling

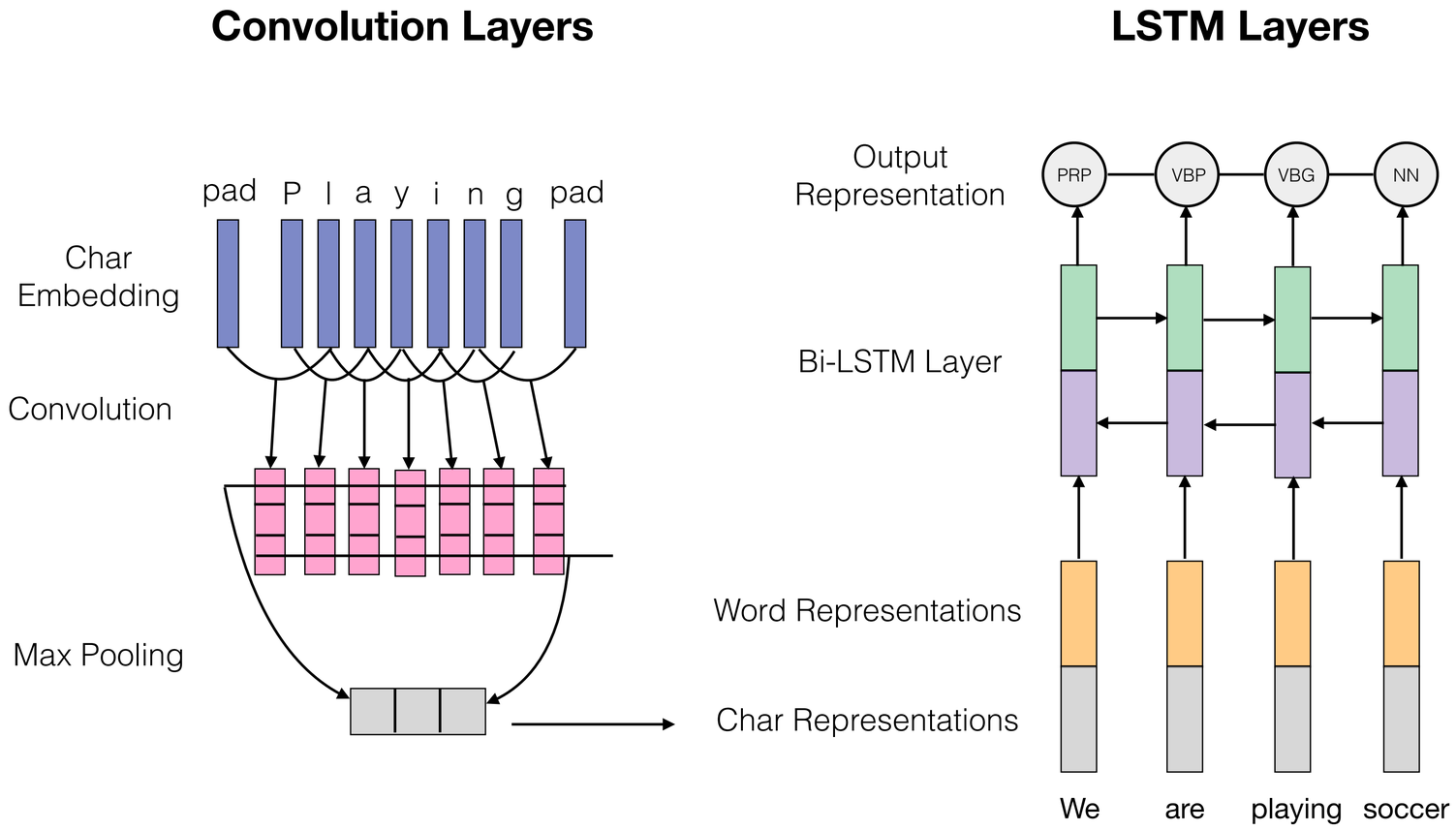
- **A simple bidirectional RNN model**

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

BLSTM is not good enough!

# Bidirectional LSTMs + CNNs

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



# Sequence Labeling

- **BLSTM-CNN**

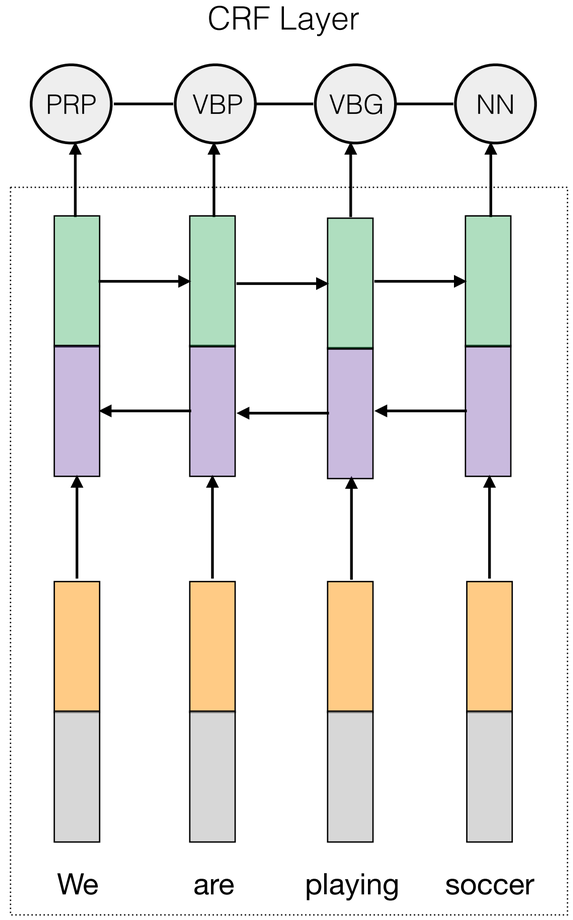
| | **POS Tagging** | **NER** |
|---|---|---|
| **CRF** | 97.0% | 88.7% |
| **CRF+**external resources | 97.3% | 91.2% |
| **BLSTM** | 96.9% | 87.0% |
| **BLSTM-CNN** | 97.3% | 89.4% |

BLSTM-CNN is better than CRF!
Structured models are no longer useful?

# Sequence Labeling: BLSTM-CNNs-CRF

$$v \cdot f(x_1, \ldots, x_m, i, y_{j-1}, y_j) = \mathbf{W}^T_{y_{i-1}, y_i} \mathbf{z}_i + b_{y_{i-1}, y_i}$$

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{W}, \mathbf{b}) = \frac{\prod_{i=1}^{n} \exp\left(\mathbf{W}_{y_{i-1}, y_i} \mathbf{z}_i + b_{y_{i-1}, y_i}\right)}{\sum_{y' \in \mathcal{Y}(\mathbf{z})} \prod_{i=1}^{n} \exp\left(\mathbf{W}_{y', y} \mathbf{z}_i + b_{y', y}\right)}$$




Bi-LSTM-CNNs

# 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!

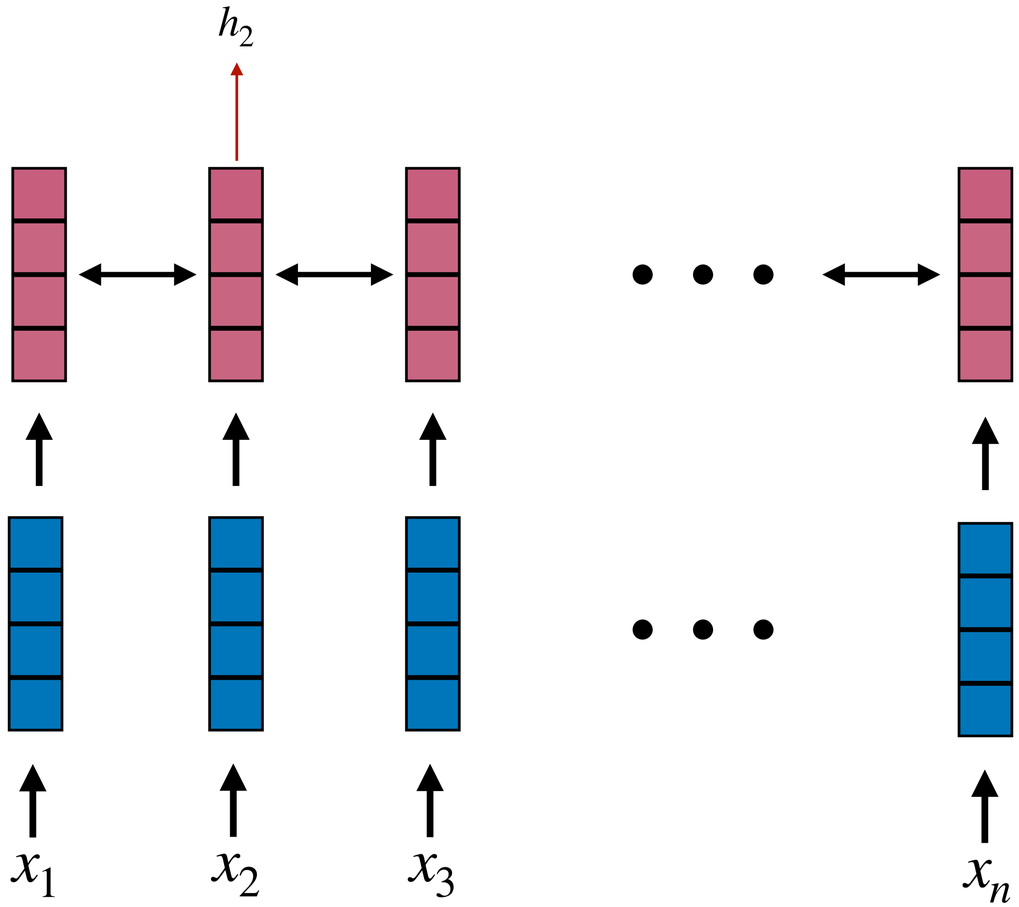
# Neural Networks for Dependency Parsing

# Dependency Parsing

- **An Unstructured Model**
  - Deep BiAffine Parser (Dozat, 2017)

$$P(y \mid x) = \prod_{i=1}^{n} P(i \rightarrow h_i \mid x)$$

$$P(i \rightarrow h_i \mid X) = \frac{\exp(z_i^T W z_{h_i})}{\sum_{h=0}^{n} \exp(z_i^T W z_h)}$$



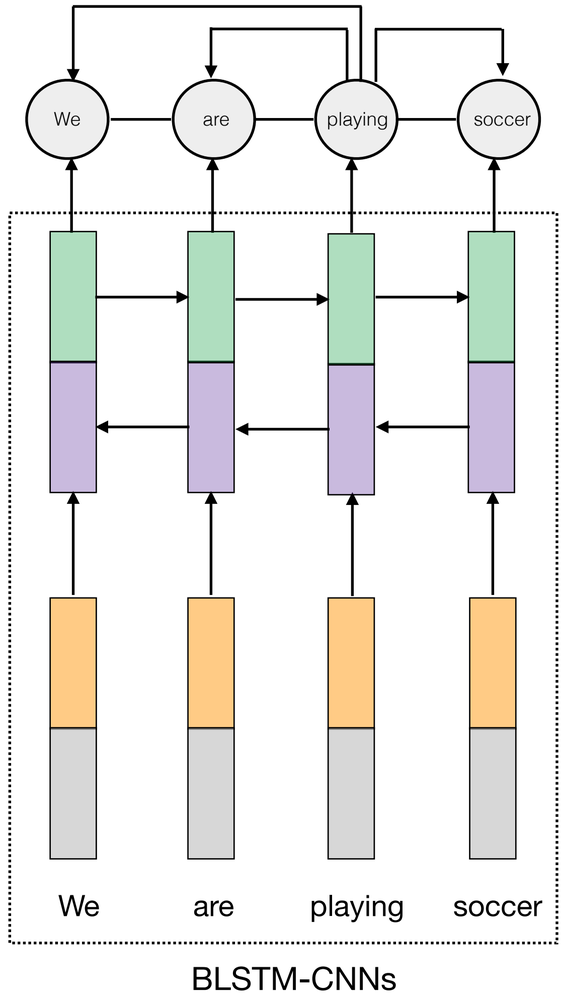
# Dependency Parsing

- **DeepBiAffine Parser**

| | English | German |
|---|---|---|
| **4th-proj** | 93.4% | 89.3% |
| **BiAffine** | 94.1% | 91.6% |
| **BiAffine+CNNs** | 94.9% | 93.4% |

# Dependency Parsing: NeuroMST Parser

- We stack a first-order graph-based model on top of a BLSTM-CNN encoder



# Dependency Parsing

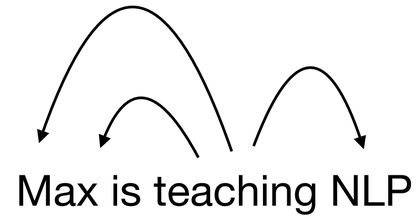
- **DeepBiAffine Parser**

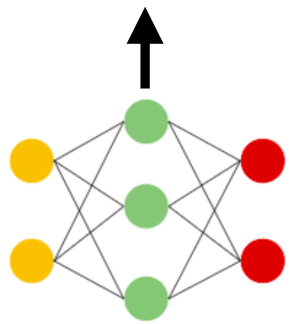
| | English | German |
|---|---|---|
| **4th-proj** | 93.4% | 89.3% |
| **BiAffine** | 94.1% | 91.6% |
| **BiAffine+CNNs** | 94.9% | 93.4% |
| **NeuroMST** | 95.8% | 93.8% |

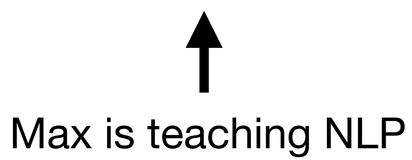
# An Interesting Observation

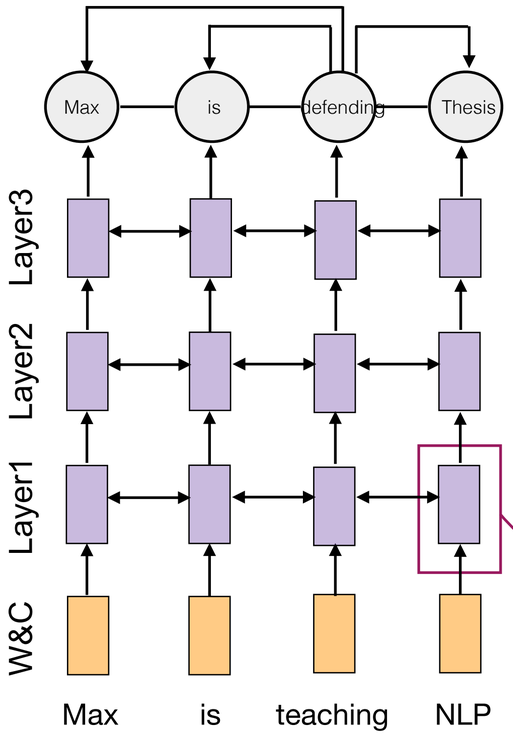
- **A Probing Experiment**


Main task: Dependency Parsing



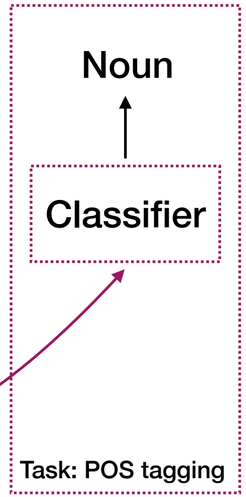






Probing task:

- POS tagging



# An Interesting Observation

| | POS Tagging |
|---|---|
| **BLSTM-CNN-CRF** | 97.6% |
| **LSTM1 + SVM** | 97.7% |
| **LSTM2 + SVM** | 97.8% |

# Reading Materials

- **Relavant Papers**
  - BLSTM-CNNs-CRF
  - Deep BiAffine Parser
  - NeuroMST Parser
  - Stack-Pointer Parser