Recurrent Neural Networks

CS 6956: Deep Learning for NLP

THE UNIVERSITY OF UTAH

Overview

1. Modeling sequences
2. Recurrent neural networks: An abstraction
3. Usage patterns for RNNs
4. BiDirectional RNNs
5. A concrete example: The Elman RNN
6. The vanishing gradient problem
7. Long short-term memory units

Overview

1. Modeling sequences
2. *Recurrent neural networks: An abstraction*
3. Usage patterns for RNNs
4. BiDirectional RNNs
5. A concrete example: The Elman RNN
6. The vanishing gradient problem
7. Long short-term memory units

Recurrent neural networks

- First introduced by Elman 1990

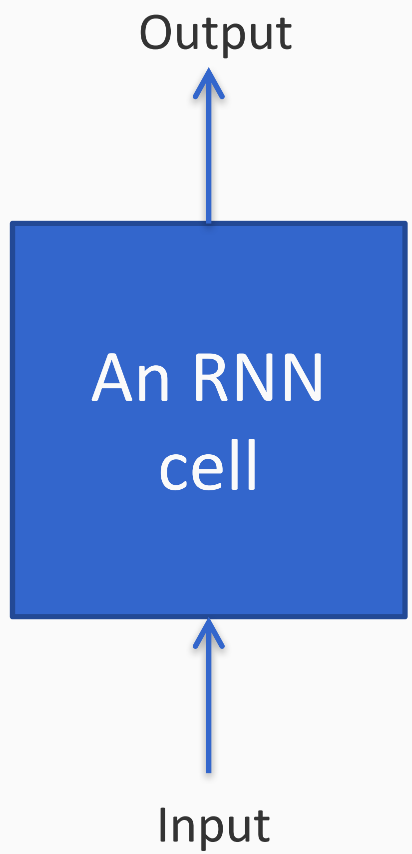
- Provides a mechanism for representing sequences of arbitrary length into vectors that encode the sequential information

- Currently, perhaps one of the most commonly used tool in the deep learning toolkit for NLP applications

The RNN abstraction

A high level overview that doesn't go into details

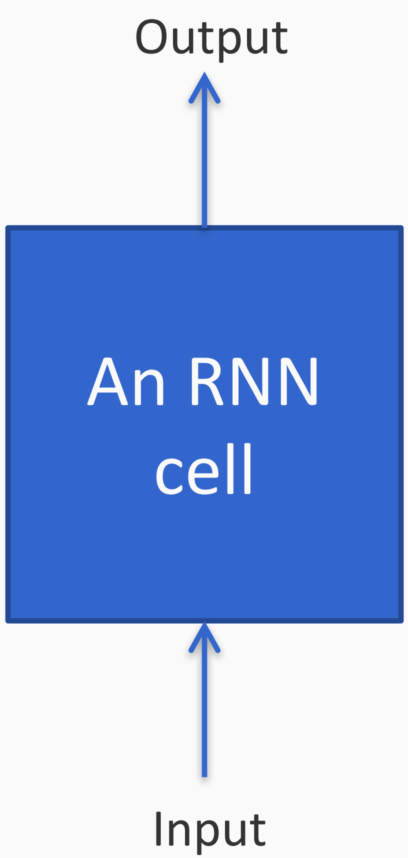
An RNN cell is a unit of differentiable compute that maps inputs to outputs



The RNN abstraction

A high level overview that doesn't go into details

An RNN cell is a unit of differentiable compute that maps inputs to outputs

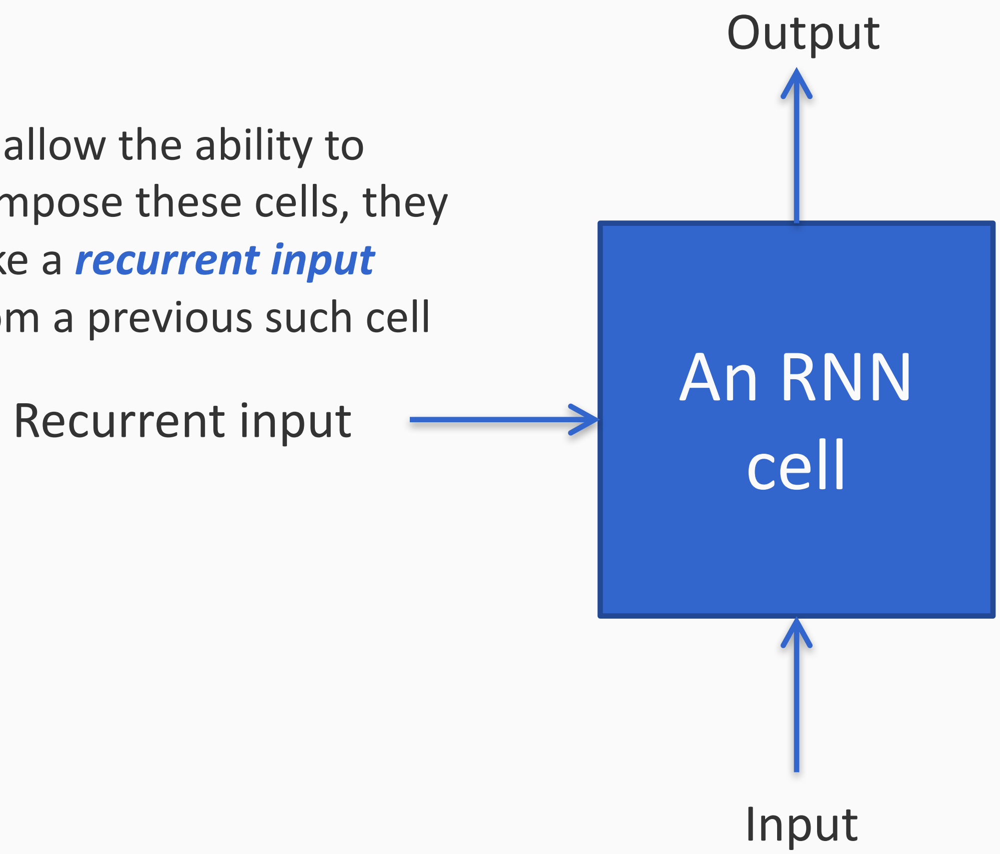


So far, no way to build a sequence of such cells

The RNN abstraction

A high level overview that doesn't go into details

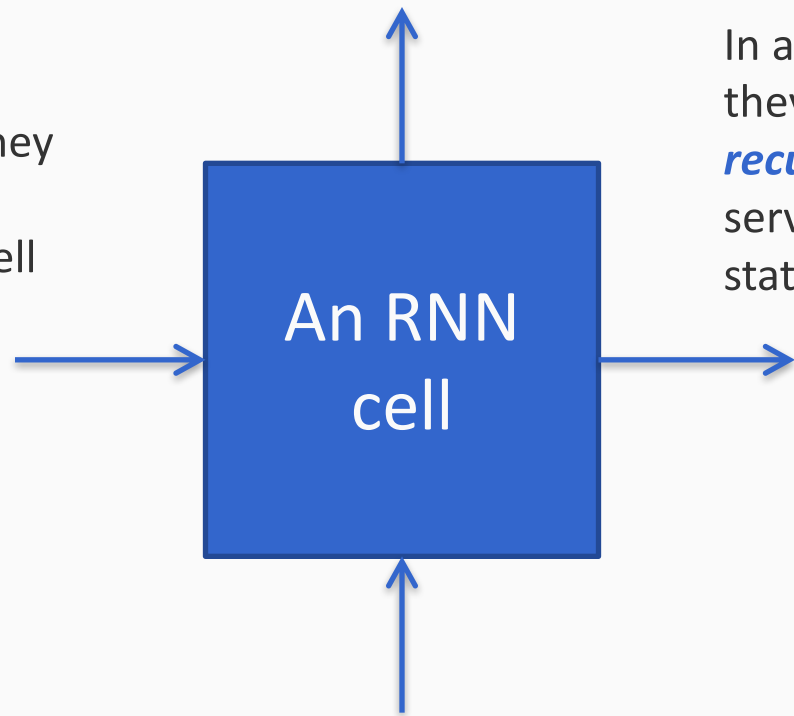
To allow the ability to compose these cells, they take a ***recurrent input*** from a previous such cell



The RNN abstraction

A high level overview that doesn't go into details

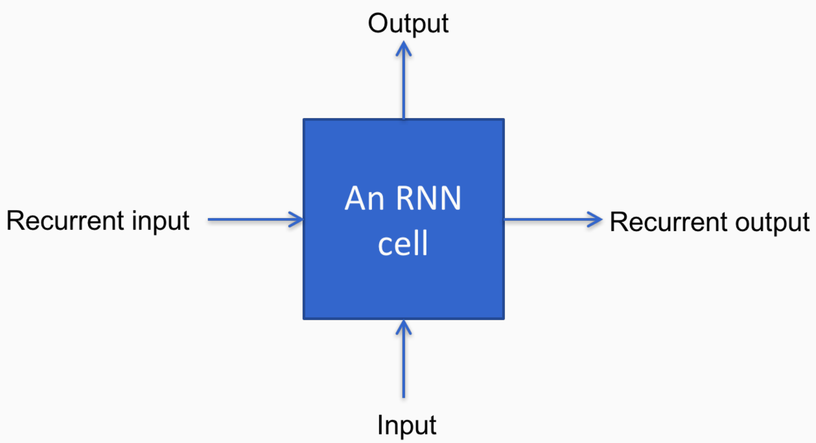
To allow the ability to compose these cells, they take a ***recurrent input*** from a previous such cell



In addition to the output, they also produce a ***recurrent output*** that can serve as a memory of past states for the next such cell

The RNN abstraction

A high level overview that doesn't go into details



Conceptually two operations

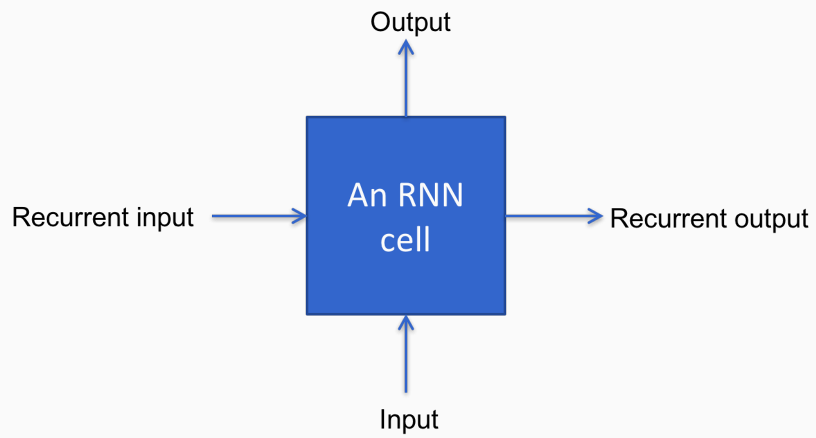
Using the input and the recurrent input (also called the previous cell state), compute

1. The next cell state

2. The output

The RNN abstraction: A simple example

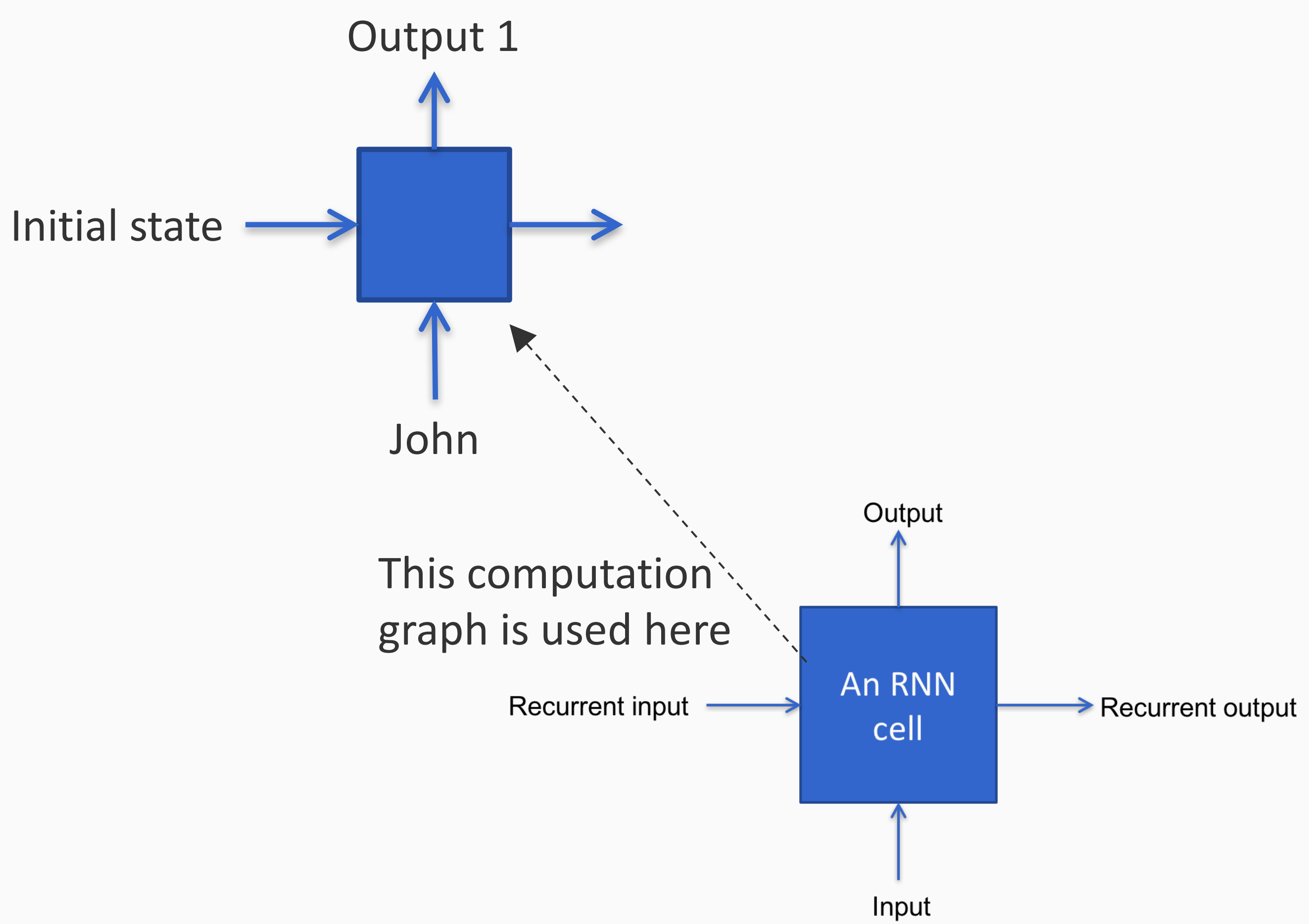
John lives in Salt Lake City



This template is **unrolled** for each input

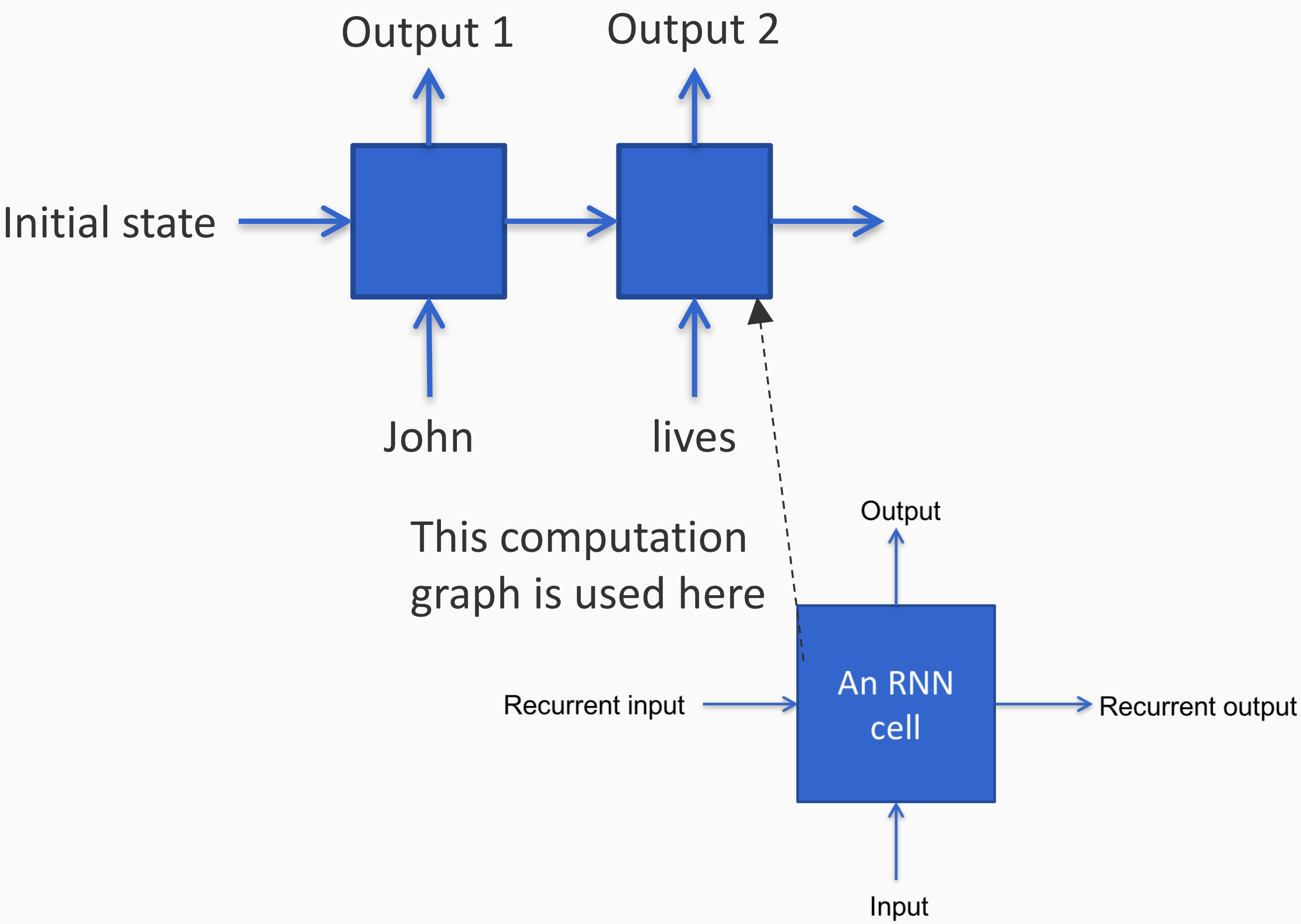
The RNN abstraction: A simple example

John lives in Salt Lake City



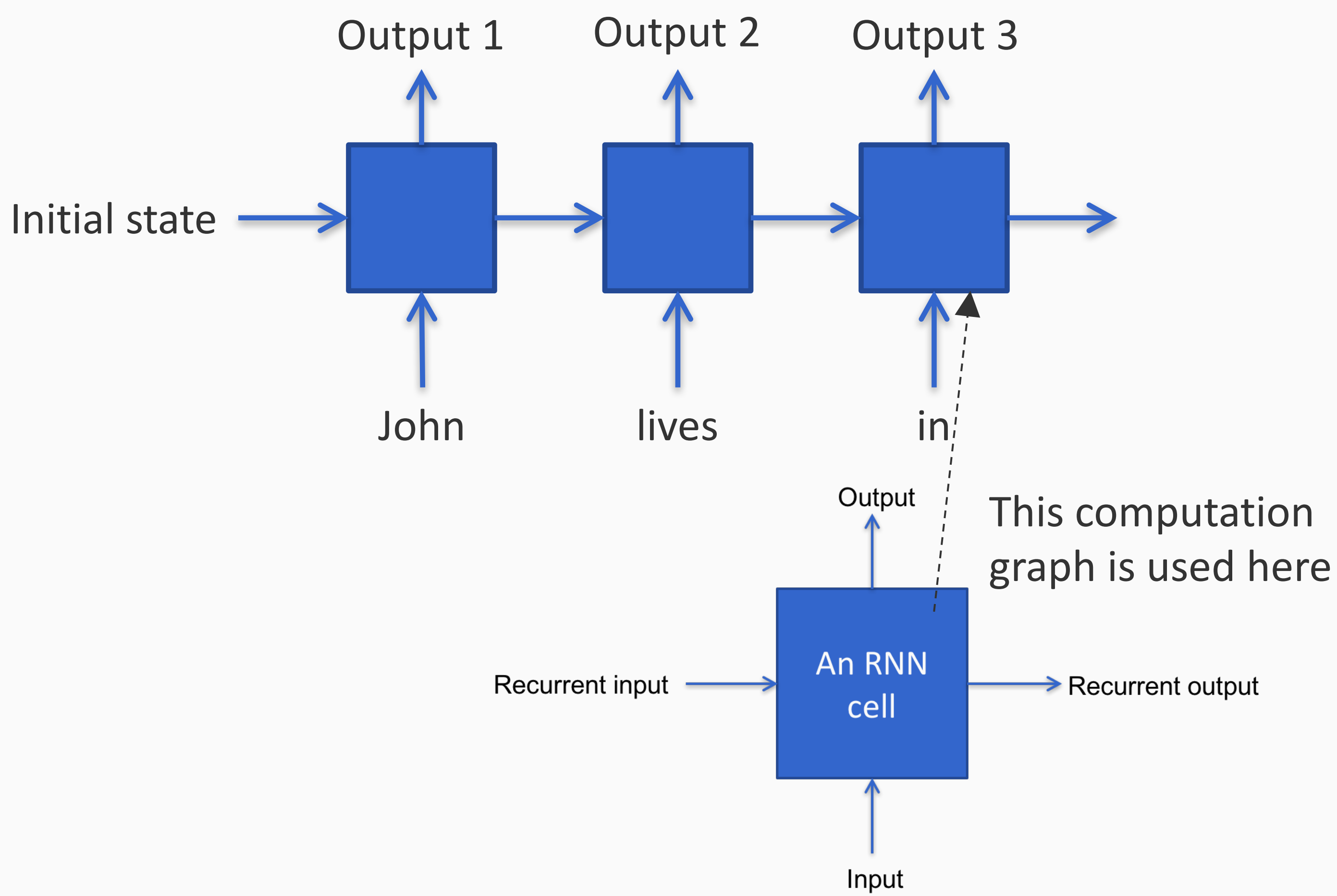
The RNN abstraction: A simple example

John lives in Salt Lake City



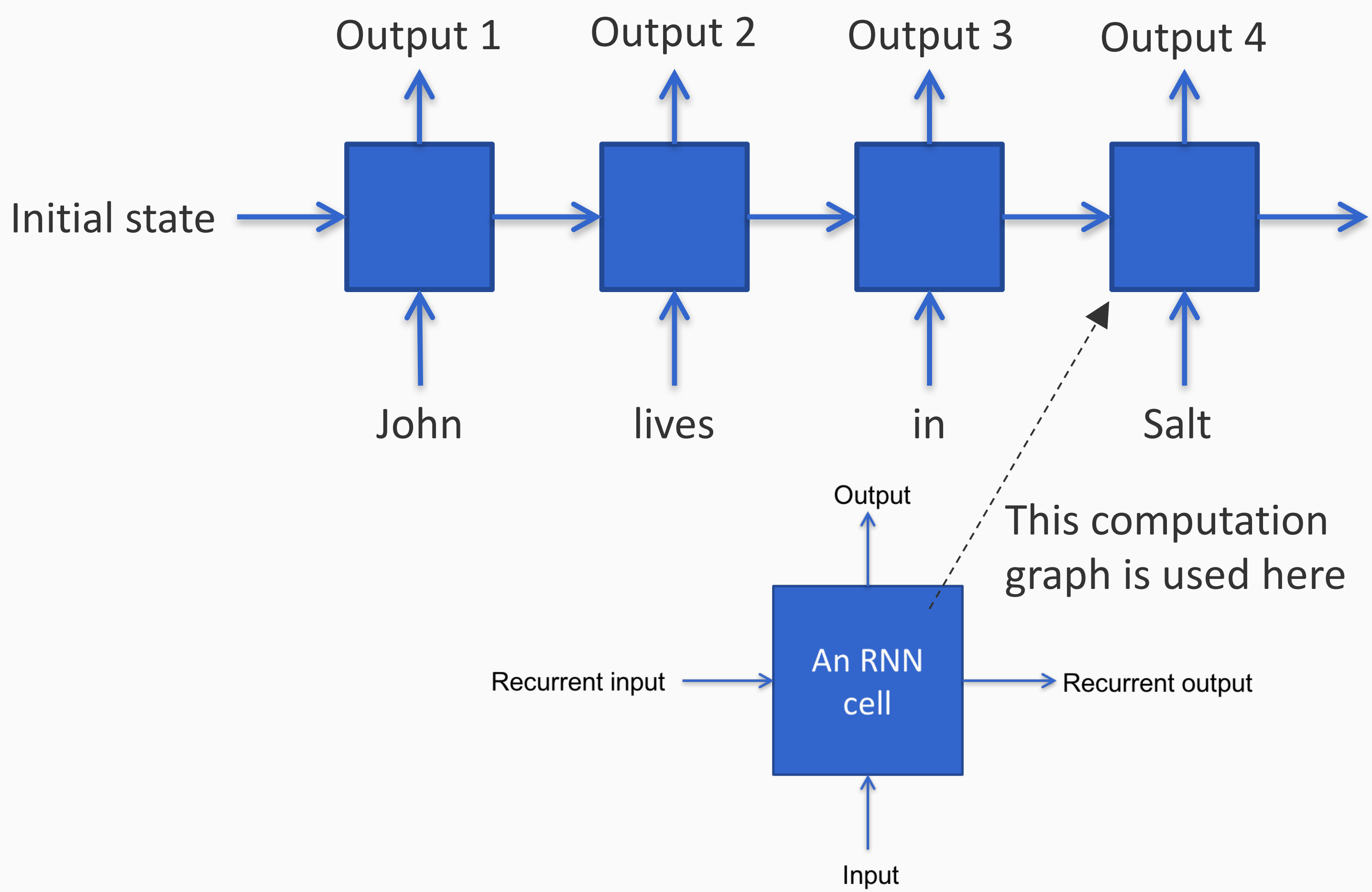
The RNN abstraction: A simple example

John lives in Salt Lake City

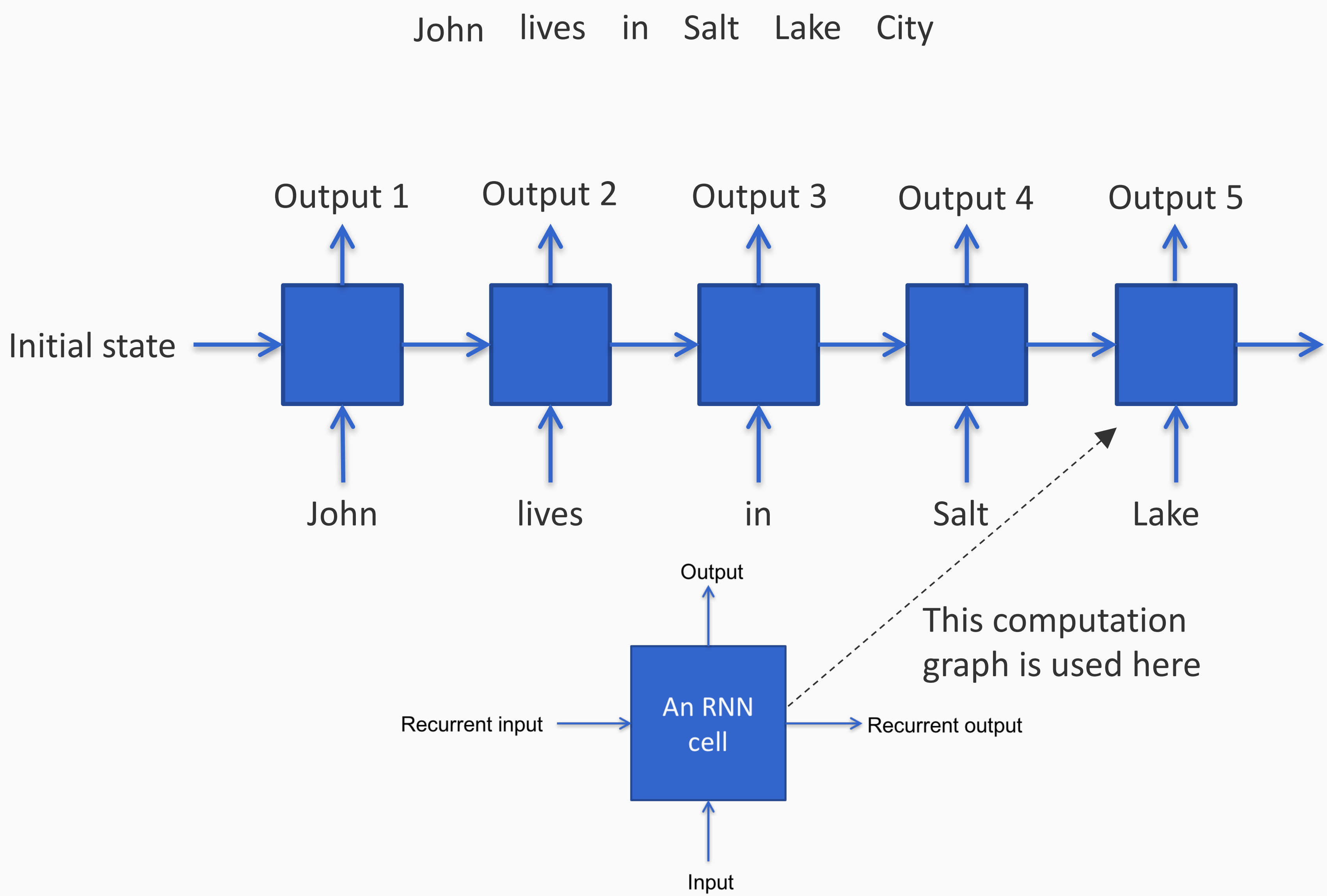


The RNN abstraction: A simple example

John lives in Salt Lake City

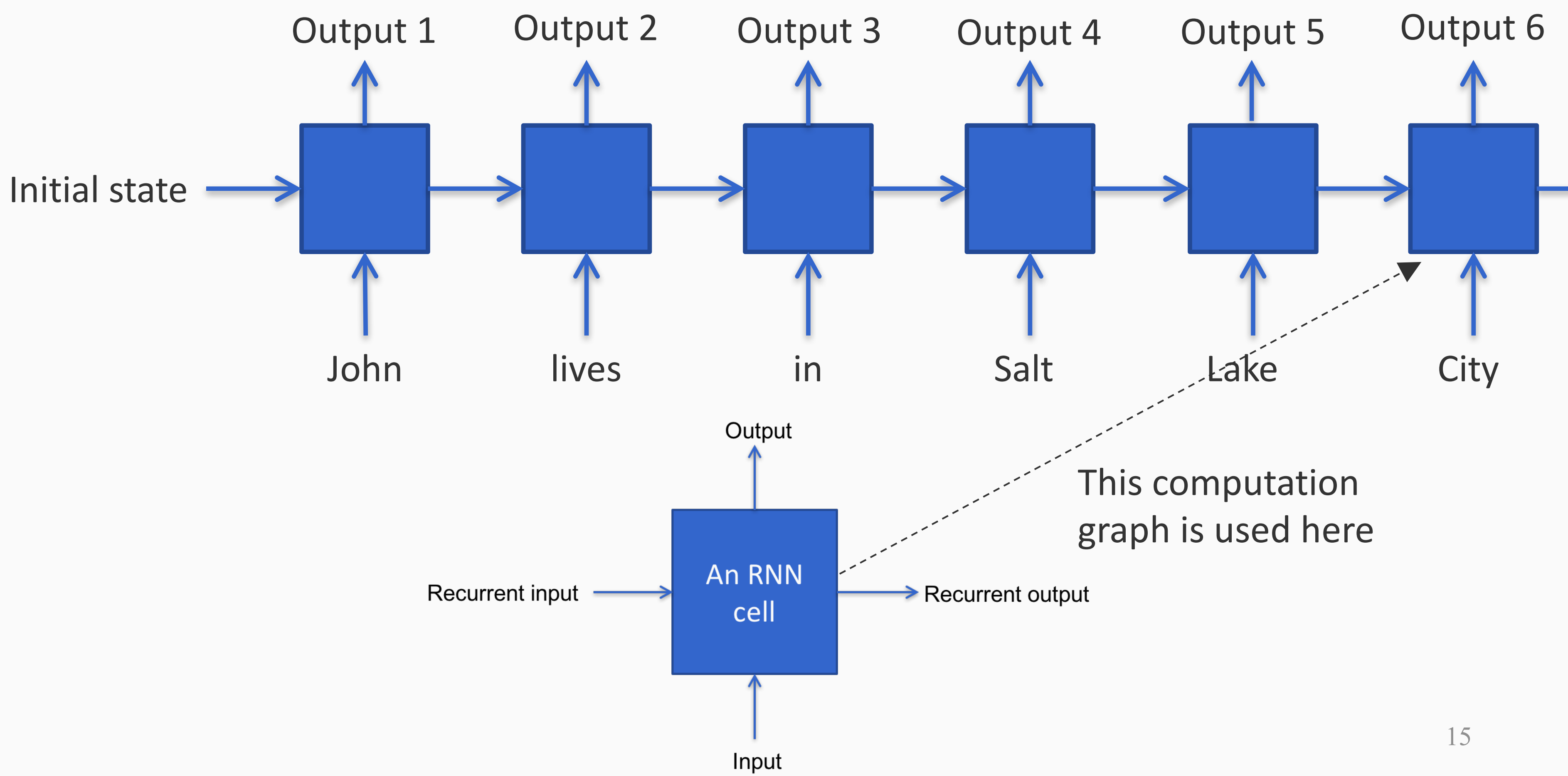


The RNN abstraction: A simple example



The RNN abstraction: A simple example

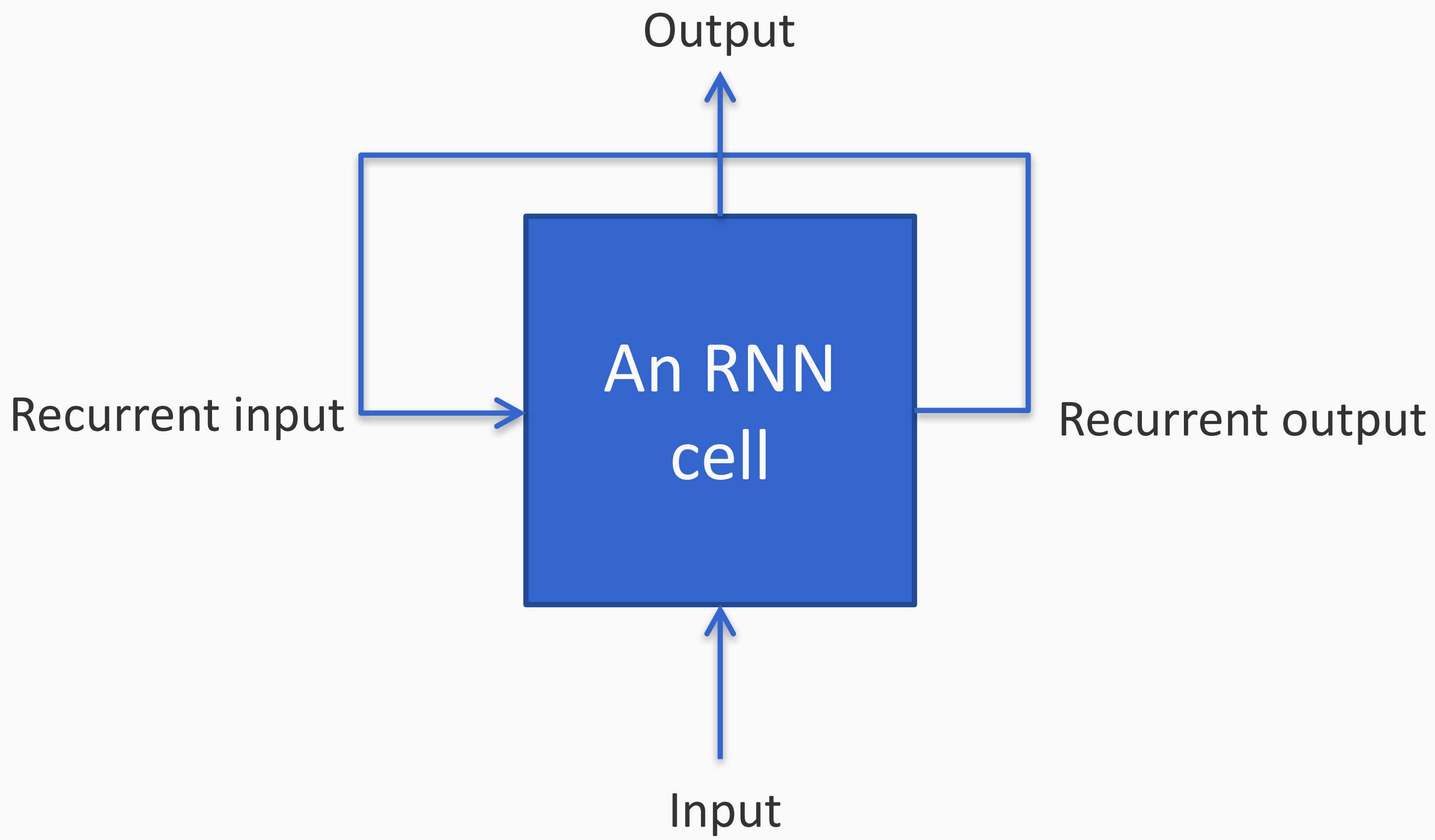
John lives in Salt Lake City



The RNN abstraction

Sometimes this is represented as a "neural network with a loop".

But really, when unrolled, there are no loops. Just a big feedforward network.



An abstract RNN :Notation

- Inputs to cells: $\mathbf{x}_t$at the t^{th} step
  - These are vectors

An abstract RNN :Notation

- Inputs to cells: $\mathbf{x}_t$at the t^{th} step
  - These are vectors
- Cell states (i.e. recurrent inputs and outputs): $\mathbf{s}_t$at the t^{th} step
  - These are also vectors

An abstract RNN :Notation

- Inputs to cells: $\mathbf{x}_t$ at the t^{th} step
  - These are vectors
- Cell states (i.e. recurrent inputs and outputs): $\mathbf{s}_t$ at the t^{th} step
  - These are also vectors
- Outputs: $\mathbf{y}_t$ at the t^{th} step
  - These are also vectors

An abstract RNN :Notation

- Inputs to cells: $\mathbf{x}_t$at the t^{th} step
  - These are vectors
- Cell states (i.e. recurrent inputs and outputs): $\mathbf{s}_t$at the t^{th} step
  - These are also vectors
- Outputs: $\mathbf{y}_t$at the t^{th} step
  - These are also vectors
- At each step:
  - Compute the next cell state: $\mathbf{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

An abstract RNN :Notation

- Inputs to cells: $\mathbf{x}_t$ at the t^{th} step
  - These are vectors
- Cell states (i.e. recurrent inputs and outputs): $\mathbf{s}_t$ at the t^{th} step
  - These are also vectors
- Outputs: $\mathbf{y}_t$ at the t^{th} step
  - These are also vectors
- At each step:
  - Compute the next cell state: $\mathbf{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

Both these functions can be parameterized. That is, they can be neural networks whose parameters are trained.

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\mathbf{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$

Encodes the sequence upto t=2 into a single vector

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
  - $\mathbf{s}_3 = \mathrm{R}(\mathbf{s}_2, \mathbf{x}_3) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\boldsymbol{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\boldsymbol{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
  - $\mathbf{s}_3 = \mathrm{R}(\mathbf{s}_2, \mathbf{x}_3) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$

Encodes the sequence upto t=3 into a single vector

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
  - $\mathbf{s}_3 = \mathrm{R}(\mathbf{s}_2, \mathbf{x}_3) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
  - $\mathbf{s}_4 = \mathrm{R}(\mathbf{s}_3, \mathbf{x}_4) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\boldsymbol{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
  - $\mathbf{s}_3 = \mathrm{R}(\mathbf{s}_2, \mathbf{x}_3) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
  - $\mathbf{s}_4 = \mathrm{R}(\mathbf{s}_3, \mathbf{x}_4) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$

Encodes the sequence upto t=4 into a single vector

What does unrolling the RNN do?

- At each step:
  - Compute the next cell state: $\mathbf{s}_t = \mathrm{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
  - Compute the output: $\boldsymbol{y}_t = \mathrm{O}(\mathbf{s}_t)$

- We can write this as:
  - $\mathbf{s}_1 = \mathrm{R}(\mathbf{s}_0, \mathbf{x}_1)$
  - $\mathbf{s}_2 = \mathrm{R}(\mathbf{s}_1, \mathbf{x}_2) = \mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
  - $\mathbf{s}_3 = \mathrm{R}(\mathbf{s}_2, \mathbf{x}_3) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
  - $\mathbf{s}_4 = \mathrm{R}(\mathbf{s}_3, \mathbf{x}_4) = \mathrm{R}(\mathrm{R}(\mathrm{R}(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$

  … and so on

Encodes the sequence upto t=4 into a single vector