

Recurrent Neural Networks

CS 114B
March 28, 2023

Slides thanks to Prof. Srikumar

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

Sequences abound in NLP

S a l t L a k e C i t y

Words are sequences of characters

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

Sentences are sequences of words

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

Paragraphs are sequences of sentences

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

And so on... inputs are naturally sequences at different levels

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

Outputs can also be sequences

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

John lives in Salt Lake City

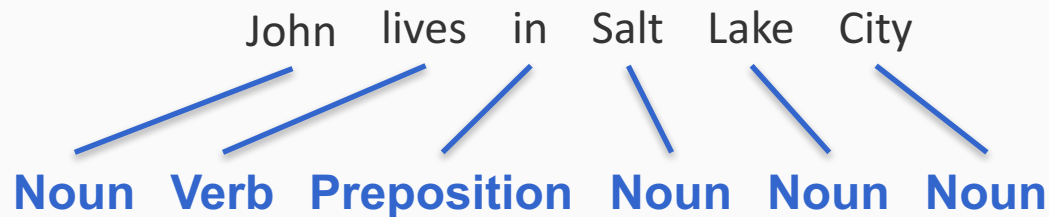
Part-of-speech tags form a sequence

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.



Part-of-speech tags form a sequence

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

Noun Verb Preposition Noun Noun Noun



Even things that don't look like a sequence can be made to look like one

Example: Named entity tags

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

Noun Verb Preposition Noun Noun Noun

John lives in Salt Lake City
/ / / / \ \
B-PER O O B-LOC I-LOC I-LOC

Even things that don't look like a sequence can be made to look like one

Example: Named entity tags

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

John lives in Salt Lake City. He enjoys hiking with his dog. His cat hates hiking.

Noun Verb Preposition Noun Noun Noun

B-PER O O B-LOC I-LOC I-LOC

And we can get very creative with such encodings

Example: We can encode parse trees as a sequence
of decisions needed to construct the tree

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

Natural question: How do we model sequential inputs and outputs?

John lives in Salt Lake City. — He enjoys hiking with his dog. — His cat hates hiking.

Noun Verb Preposition Noun Noun Noun

B-PER O O B-LOC I-LOC I-LOC

And we can get very creative with such encodings

Example: We can encode parse trees as a sequence of decisions needed to construct the tree

Sequences abound in NLP

S a l t L a k e C i t y

John lives in Salt Lake City

Natural question: How do we model sequential inputs and outputs?

More concretely, we need a mechanism that allows us to

1. Capture sequential dependencies between inputs
2. Model uncertainty over sequential outputs

And we can get very creative with such encodings

Example: We can encode parse trees as a sequence of decisions needed to construct the tree

Modeling sequences: The problem

Suppose we want to build a language model that computes the probability of sentences

We can write the probability as

$$P(x_1, x_2, x_3, \dots, x_n) = \prod_i P(x_i \mid x_1, x_2, \dots, x_{i-1})$$

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following "It"

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following "It"

$P(\text{a}|\text{It was}) \times$  Probability of a word following "It was"

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following “It”

$P(\text{a}|\text{It was}) \times$  Probability of a word following “It was”

$P(\text{bright}|\text{It was a}) \times$  Probability of a word following “It was a”

Example: A Language model

It was a bright cold day in April.

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following “It”

$P(\text{a}|\text{It was}) \times$  Probability of a word following “It was”

$P(\text{bright}|\text{It was a}) \times$  Probability of a word following “It was a”

$P(\text{cold}|\text{It was a bright}) \times$

$P(\text{day}|\text{It was a bright cold}) \times \dots$

A history-based model

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | x_1, x_2, \dots, x_{i-1})$$

- Each token is dependent on all the tokens that came before it
 - Simple conditioning
 - Each $P(x_i | \dots)$ is a multinomial probability distribution over the tokens
- What is the problem here?

A history-based model

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | x_1, x_2, \dots, x_{i-1})$$

- Each token is dependent on all the tokens that came before it
 - Simple conditioning
 - Each $P(x_i | \dots)$ is a multinomial probability distribution over the tokens
- What is the problem here?
 - How many parameters do we have?
 - Grows with the size of the sequence!

The traditional solution: Lose the history

Make a modeling assumption

Example: The **first order Markov model** assumes that

$$P(x_i | x_1, x_2, \dots, x_{i-1}) = P(x_i | x_{i-1})$$

The traditional solution: Lose the history

Make a modeling assumption

Example: The **first order Markov model** assumes that

$$P(x_i | x_1, x_2, \dots, x_{i-1}) = P(x_i | x_{i-1})$$

This allows us to simplify

$$P(x_1, x_2, x_3, \dots, x_n) = \prod_i P(x_i | x_1, x_2 \dots, x_{i-1})$$

These dependencies are ignored

The traditional solution: Lose the history

Make a modeling assumption

Example: The **first order Markov model** assumes that

$$P(x_i \mid x_1, x_2, \dots, x_{i-1}) = P(x_i \mid x_{i-1})$$

This allows us to simplify

$$P(x_1, x_2, x_3, \dots, x_n) = \prod_i P(x_i \mid x_{i-1})$$

Example: Another language model

It was a bright cold day in April

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following “It”

$P(\text{a}|\text{was}) \times$  Probability of a word following “was”

$P(\text{bright}|\text{a}) \times$  Probability of a word following “a”

$P(\text{cold}|\text{bright}) \times$

$P(\text{day}|\text{cold}) \times \dots$

Example: Another language model

It was a bright cold day in April

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following “It”

$P(\text{a}|\text{was}) \times$  Probability of a word following “was”

$P(\text{bright}|\text{a}) \times$  Probability of a word following “a”

$P(\text{cold}|\text{bright}) \times$

$P(\text{day}|\text{cold}) \times \dots$

If there are K tokens/states, how many parameters do we need?

Example: Another language model

It was a bright cold day in April

$P(\text{It was a bright cold day in April}) =$

$P(\text{It}) \times$  Probability of a word starting a sentence

$P(\text{was}|\text{It}) \times$  Probability of a word following “It”

$P(\text{a}|\text{was}) \times$  Probability of a word following “was”

$P(\text{bright}|\text{a}) \times$  Probability of a word following “a”

$P(\text{cold}|\text{bright}) \times$

$P(\text{day}|\text{cold}) \times \dots$

If there are K tokens/states, how many parameters
do we need? $O(K^2)$

Can we do better?

- Can we capture the meaning of the entire history without arbitrarily growing the number of parameters?
- Or equivalently, can we discard the Markov assumption?

Can we do better?

- Can we capture the meaning of the entire history without arbitrarily growing the number of parameters?
- Or equivalently, can we discard the Markov assumption?
- Can we represent arbitrarily long sequences as fixed sized vectors?
 - Perhaps to provide features for subsequent classification

Can we do better?

- Can we capture the meaning of the entire history without arbitrarily growing the number of parameters?
- Or equivalently, can we discard the Markov assumption?
- Can we represent arbitrarily long sequences as fixed sized vectors?
 - Perhaps to provide features for subsequent classification
- Answer: Recurrent neural networks (RNNs)

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

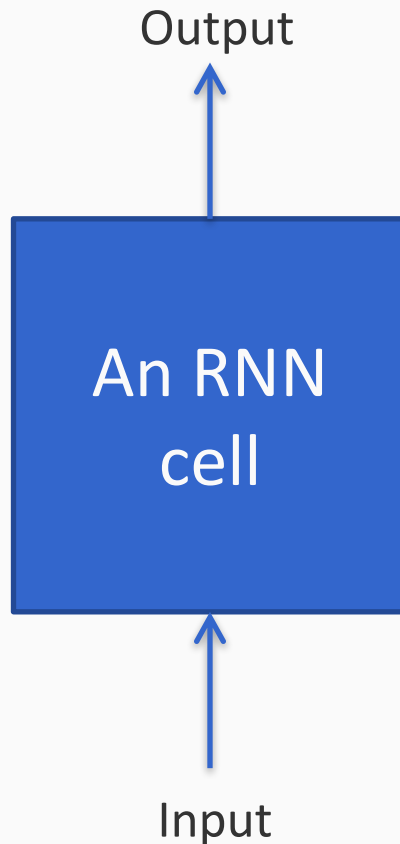
Recurrent neural networks

- 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

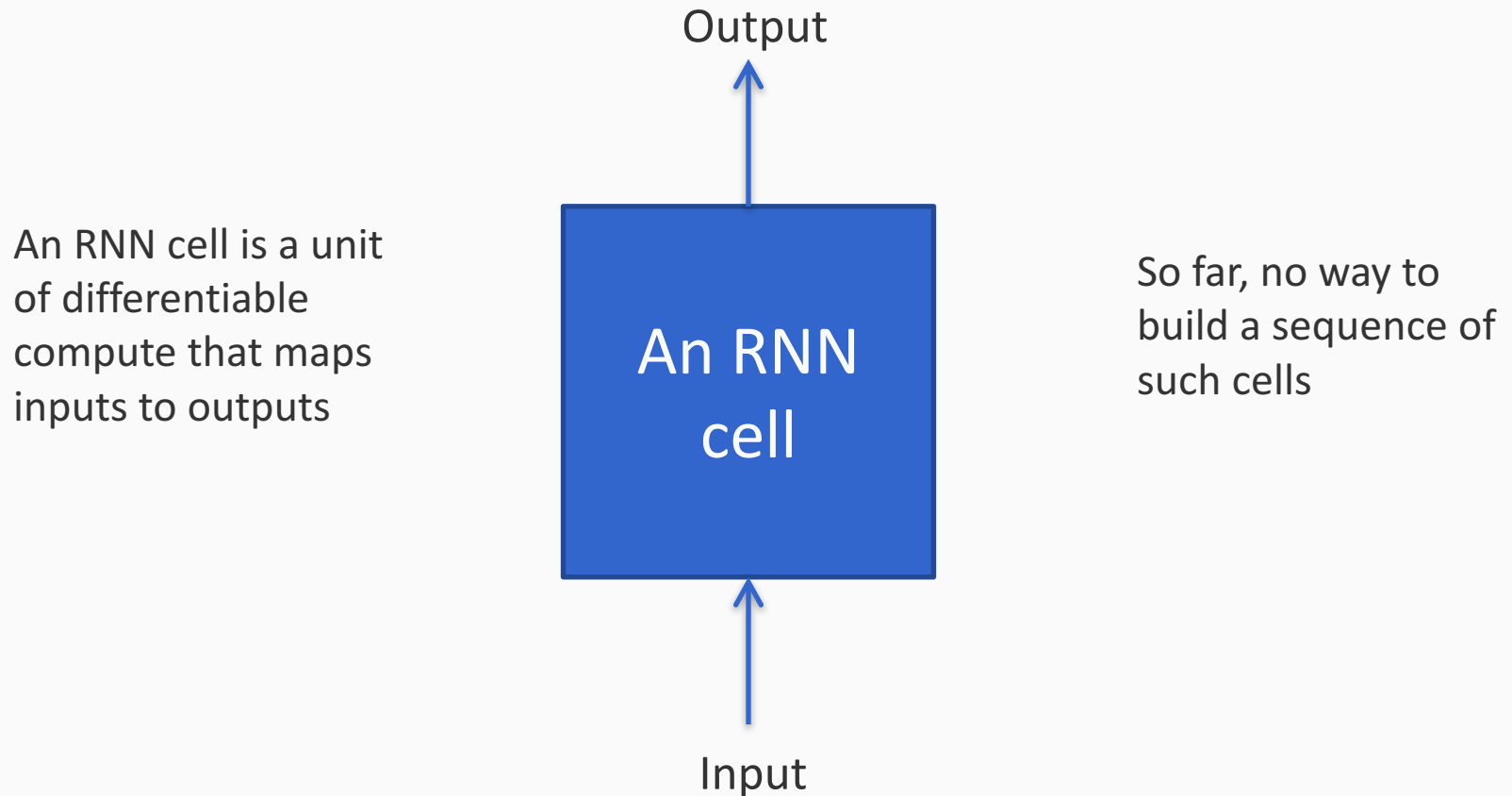
A high level overview that doesn't go into details

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



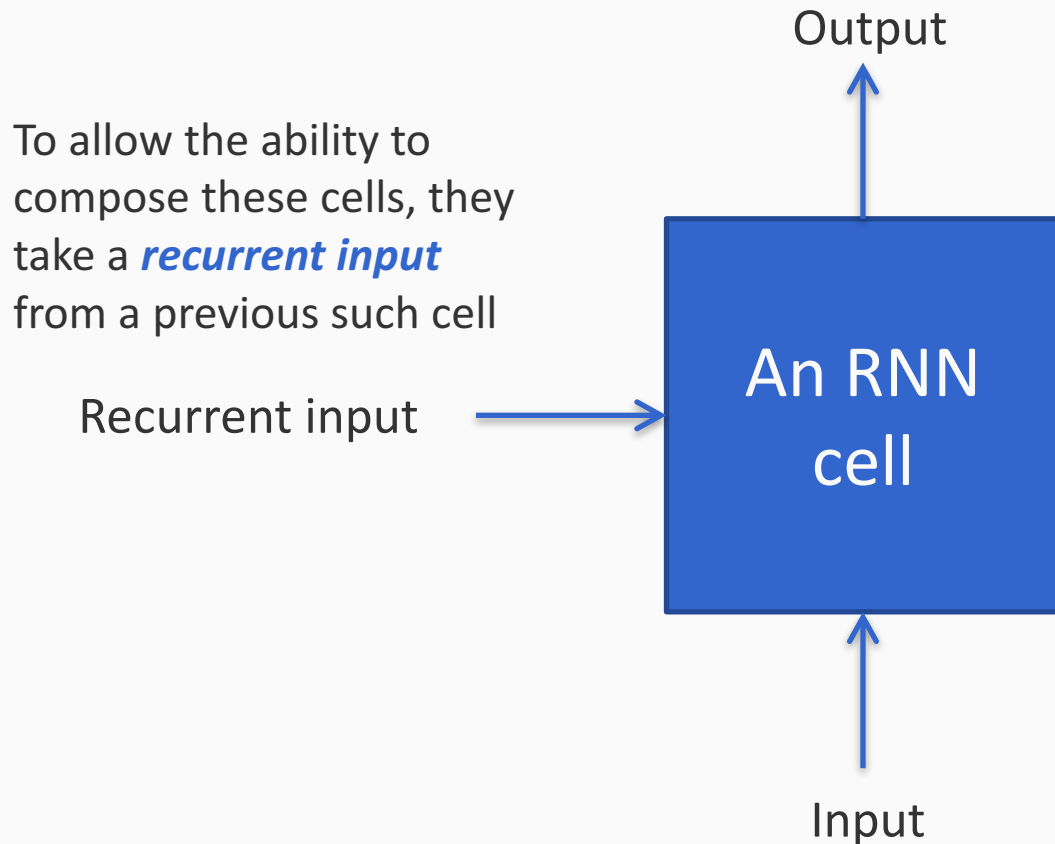
The RNN abstraction

A high level overview that doesn't go into details



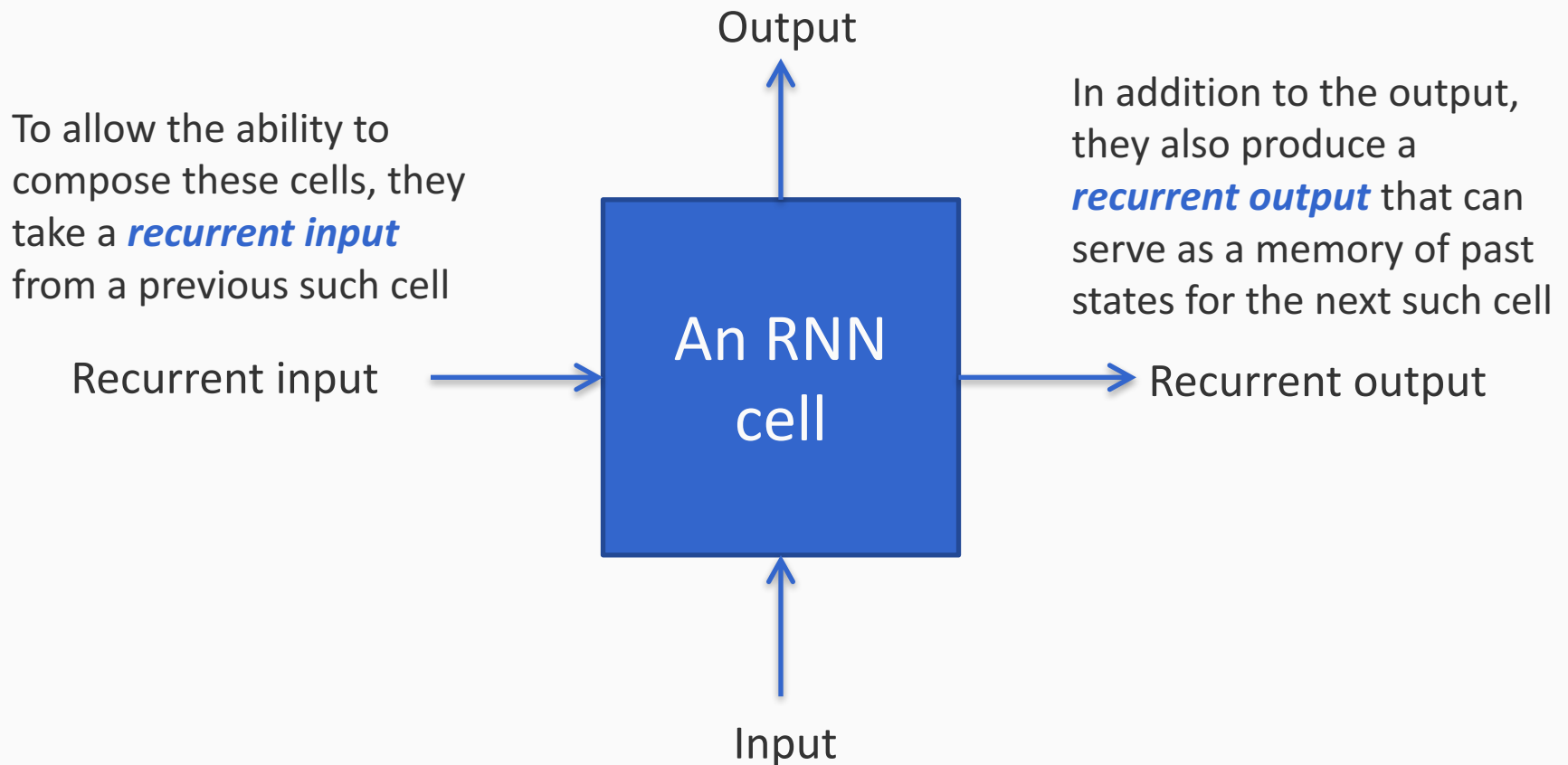
The RNN abstraction

A high level overview that doesn't go into details



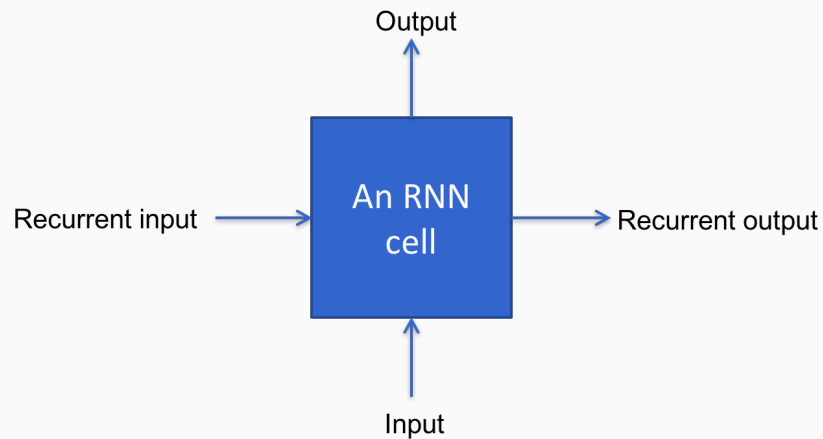
The RNN abstraction

A high level overview that doesn't go into details



The RNN abstraction

A high level overview that doesn't go into details



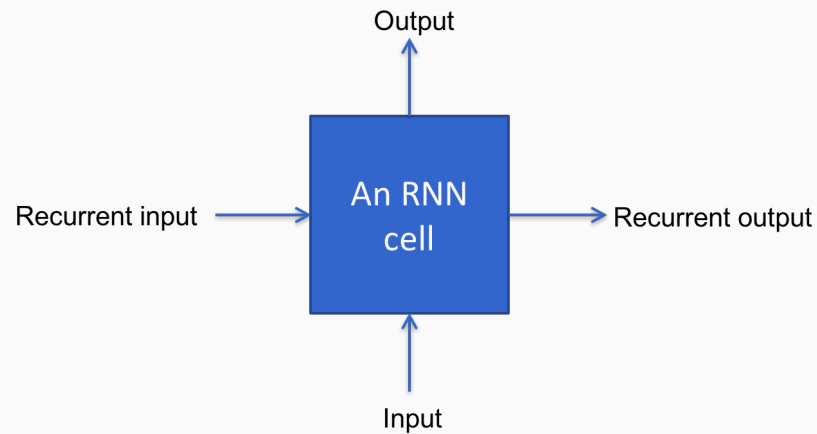
Conceptually two operations

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

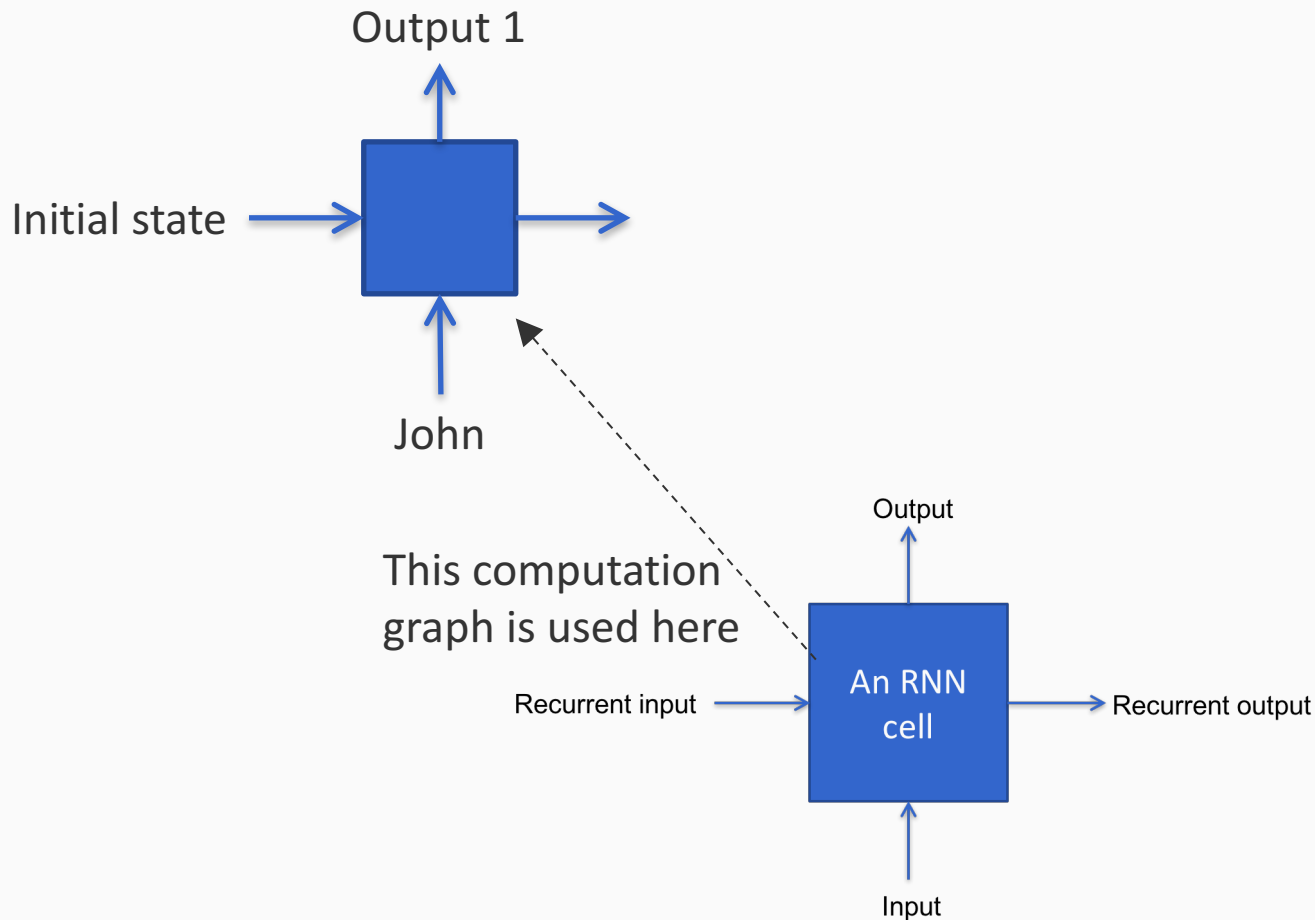
John lives in Salt Lake City



This template is **unrolled** for each input

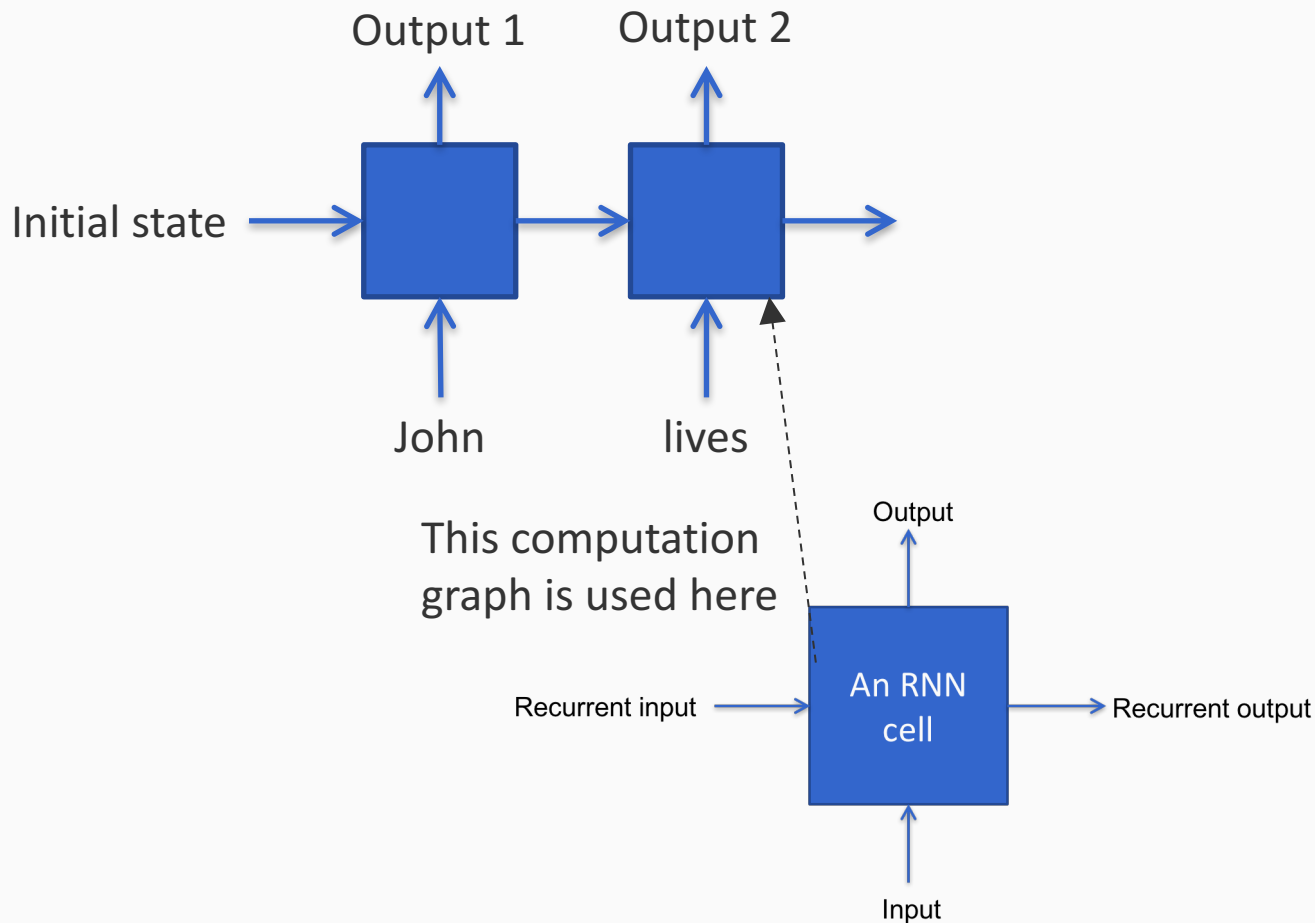
The RNN abstraction: A simple example

John lives in Salt Lake City



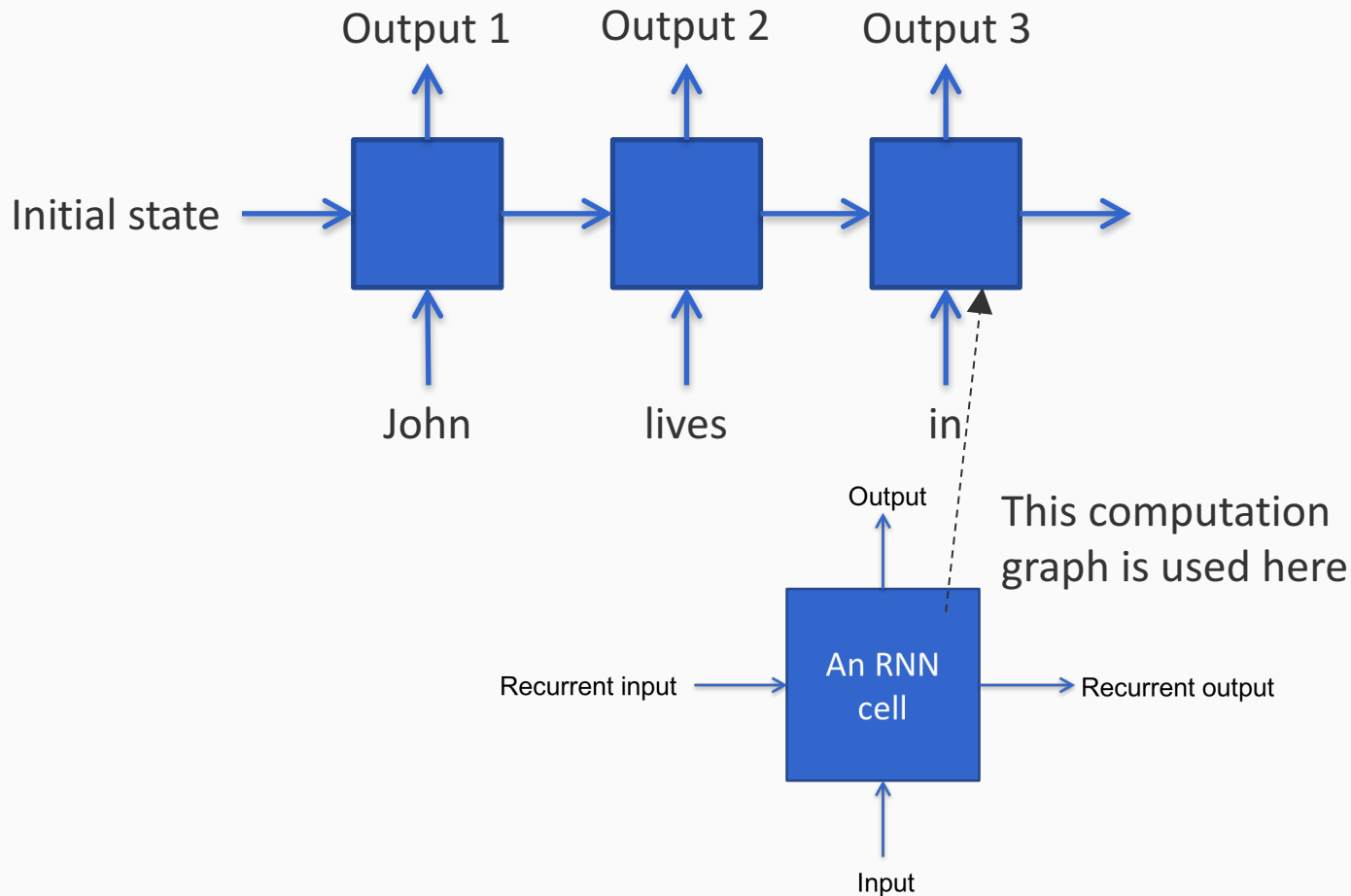
The RNN abstraction: A simple example

John lives in Salt Lake City



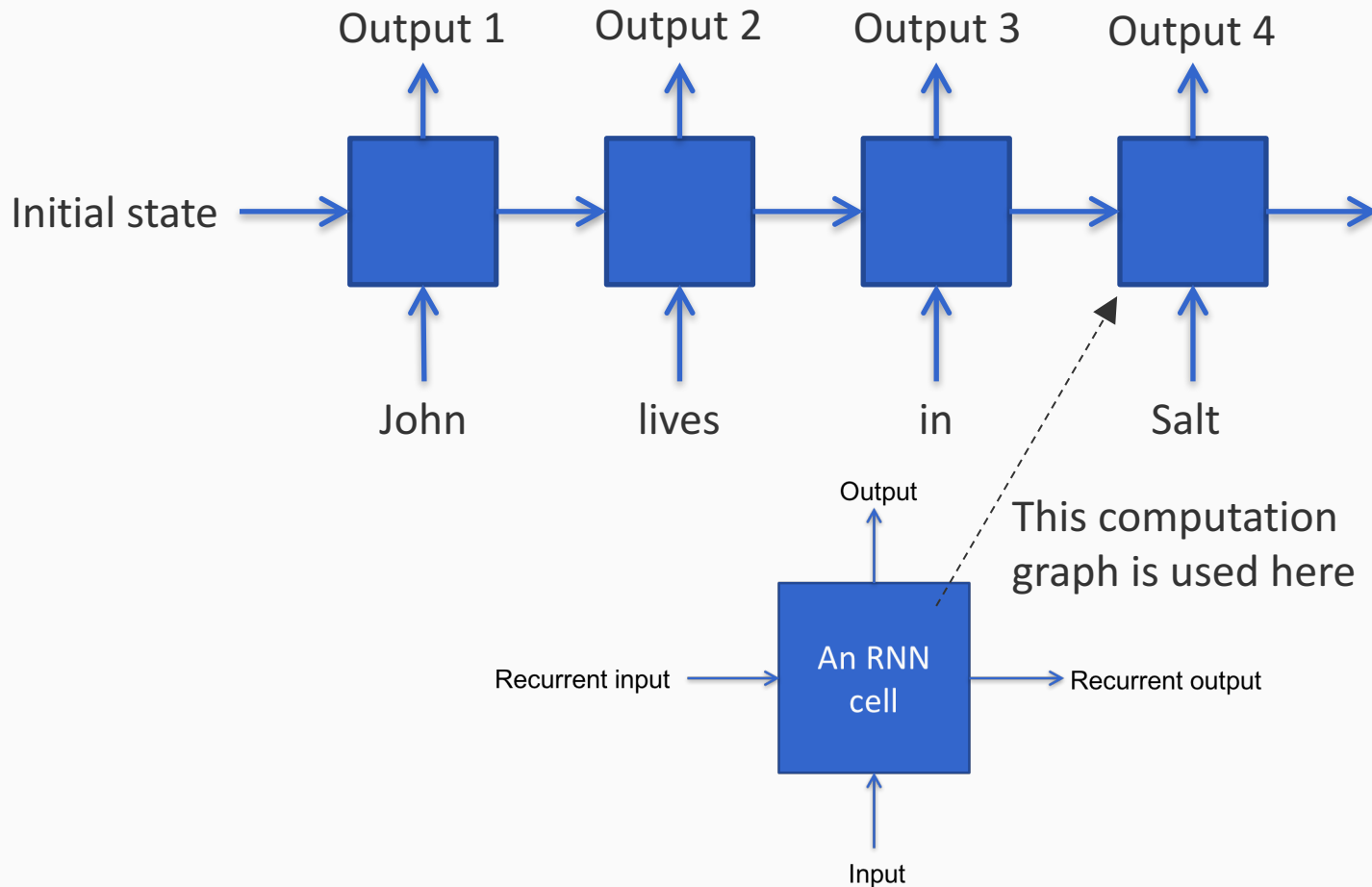
The RNN abstraction: A simple example

John lives in Salt Lake City



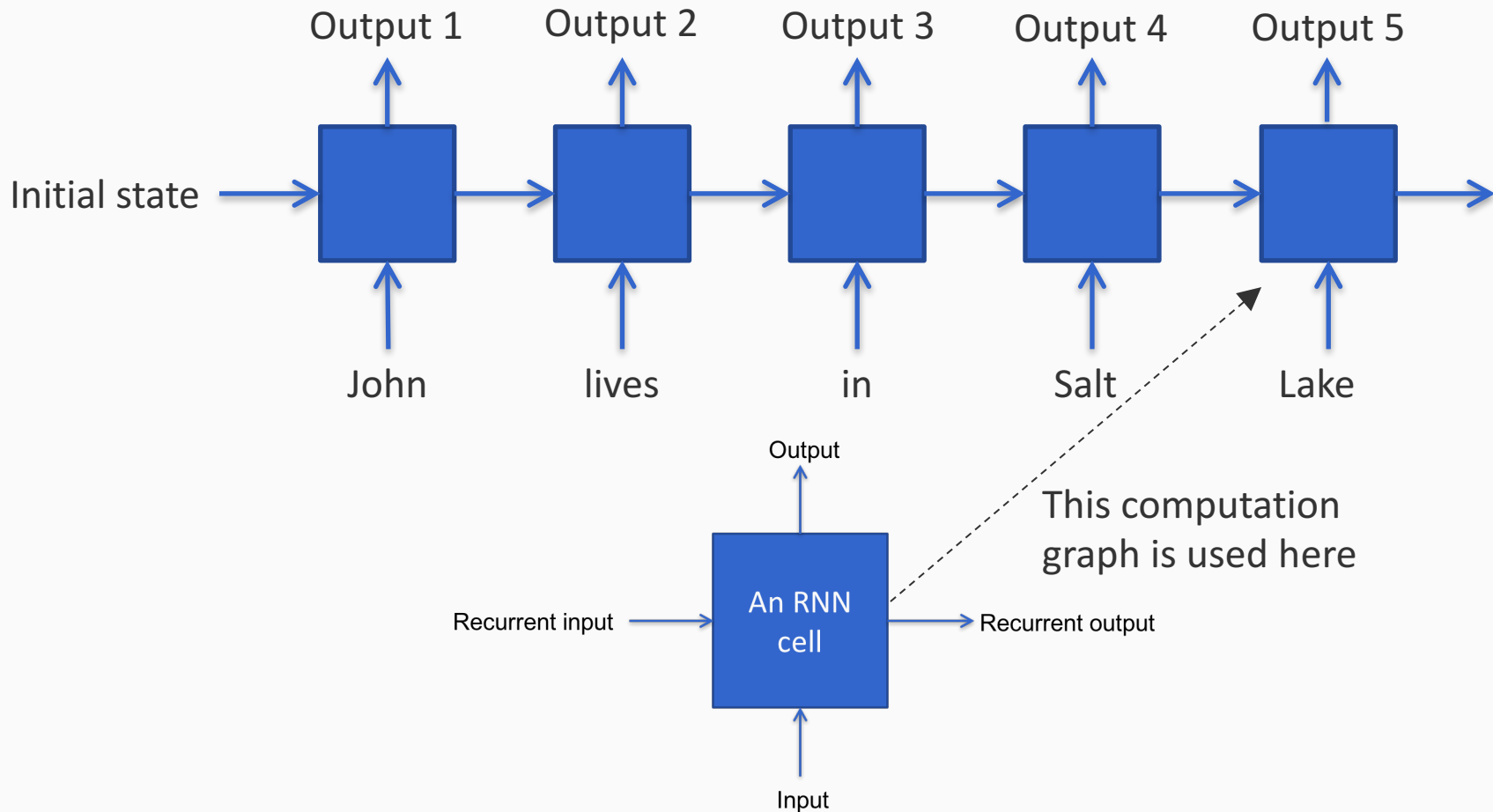
The RNN abstraction: A simple example

John lives in Salt Lake City



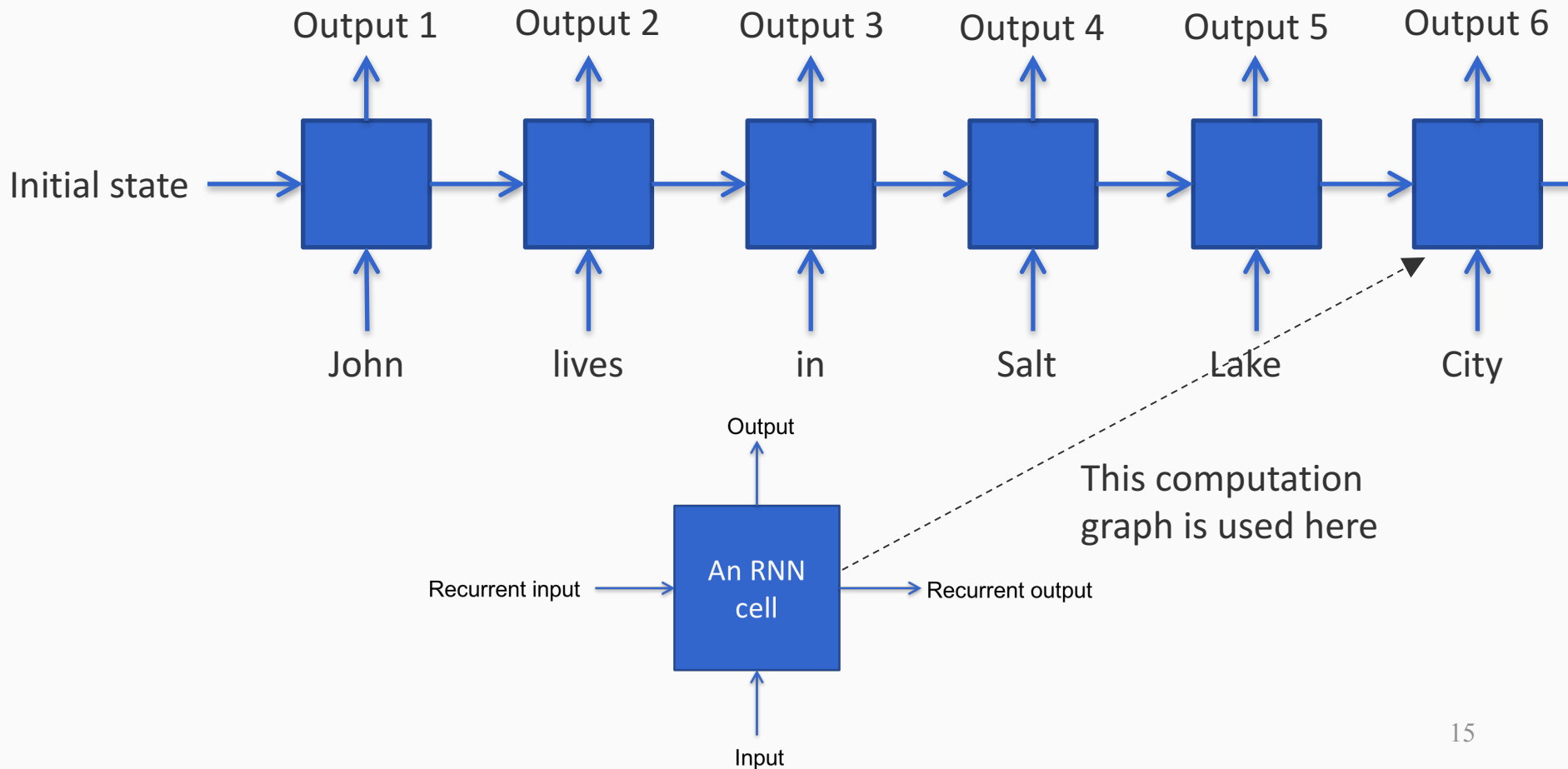
The RNN abstraction: A simple example

John lives in Salt Lake City



The RNN abstraction: A simple example

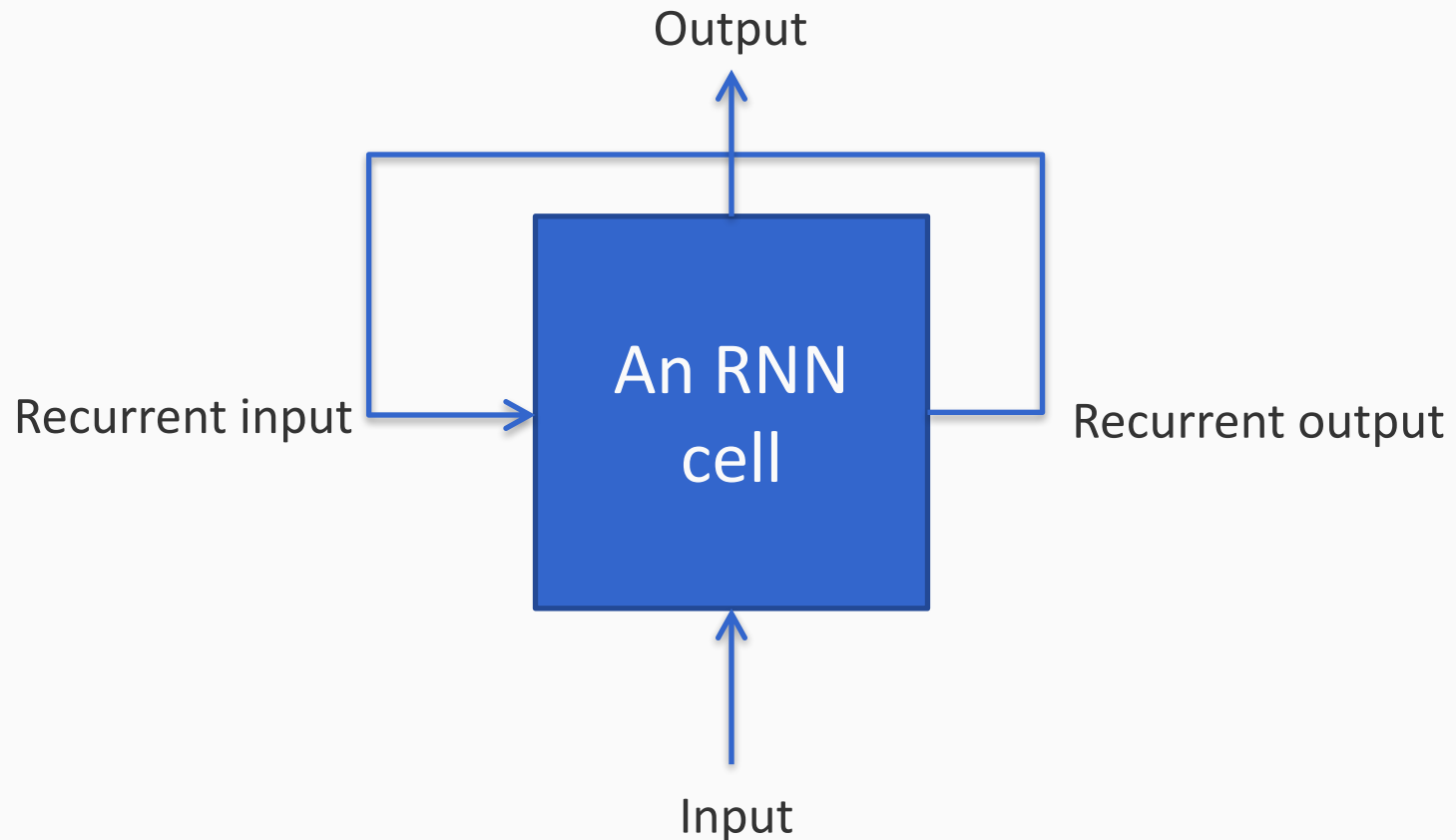
John lives in Salt Lake City



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: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$

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: $\mathbf{s}_t = \mathbf{R}(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = \mathbf{O}(\mathbf{s}_t)$

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: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
- We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$

What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
- We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$

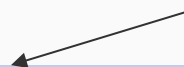
What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
- We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$ ← Encodes the sequence upto t=2 into a single vector

What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
- We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
 - $\mathbf{s}_3 = R(\mathbf{s}_2, \mathbf{x}_3) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$


What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
 - We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
 - $\mathbf{s}_3 = R(\mathbf{s}_2, \mathbf{x}_3) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
- Encodes the sequence upto $t=3$ into a single vector
- 

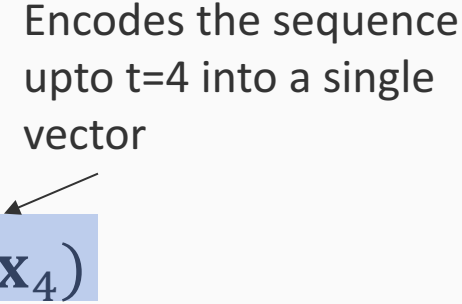
What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
- We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
 - $\mathbf{s}_3 = R(\mathbf{s}_2, \mathbf{x}_3) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
 - $\mathbf{s}_4 = R(\mathbf{s}_3, \mathbf{x}_4) = R(R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$

What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
 - We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
 - $\mathbf{s}_3 = R(\mathbf{s}_2, \mathbf{x}_3) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
 - $\mathbf{s}_4 = R(\mathbf{s}_3, \mathbf{x}_4) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$
- Encodes the sequence upto t=4 into a single vector
- 

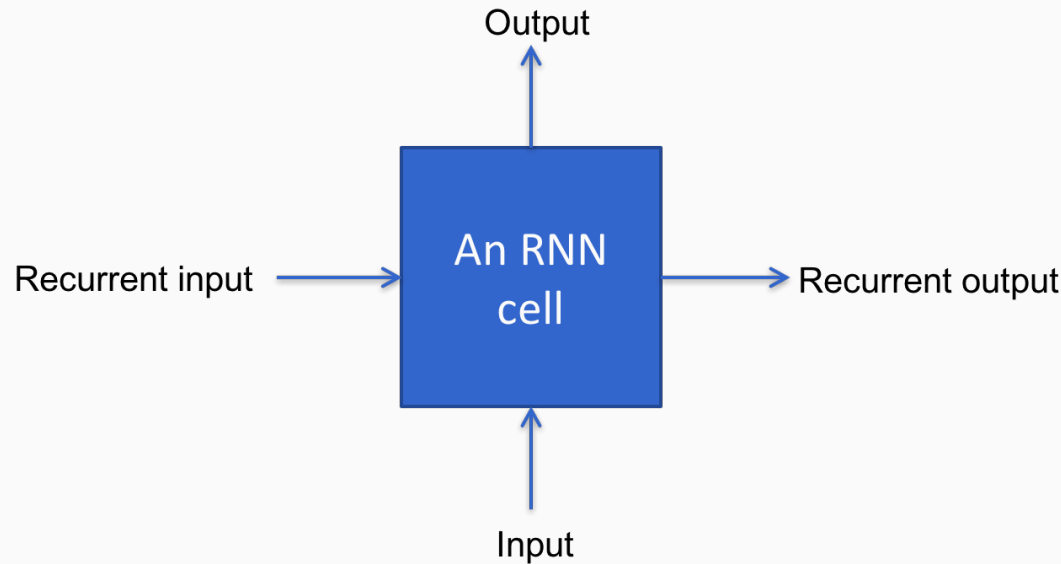
What does unrolling the RNN do?

- At each step:
 - Compute the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
 - Compute the output: $\mathbf{y}_t = O(\mathbf{s}_t)$
 - We can write this as:
 - $\mathbf{s}_1 = R(\mathbf{s}_0, \mathbf{x}_1)$
 - $\mathbf{s}_2 = R(\mathbf{s}_1, \mathbf{x}_2) = R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2)$
 - $\mathbf{s}_3 = R(\mathbf{s}_2, \mathbf{x}_3) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3)$
 - $\mathbf{s}_4 = R(\mathbf{s}_3, \mathbf{x}_4) = R(R(R(\mathbf{s}_0, \mathbf{x}_1), \mathbf{x}_2), \mathbf{x}_3), \mathbf{x}_4)$
 - ... and so on
- Encodes the sequence upto $t=4$ into a single vector
- 

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

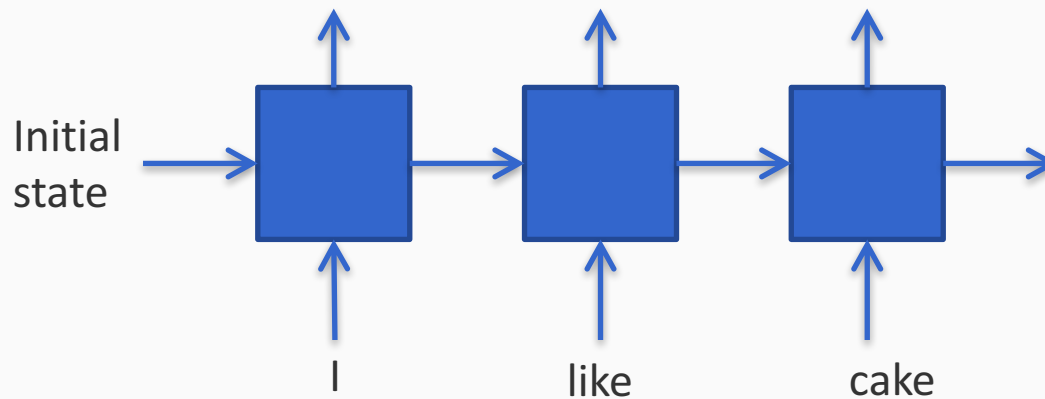
What can we do with such an abstraction?



1. The encoder: Convert a sequence into a feature vector for subsequent classification
2. A generator: Produce a sequence using an initial state
3. A transducer: Convert a sequence into another sequence
4. A conditioned generator (or an encoder-decoder): Combine 1 and 2

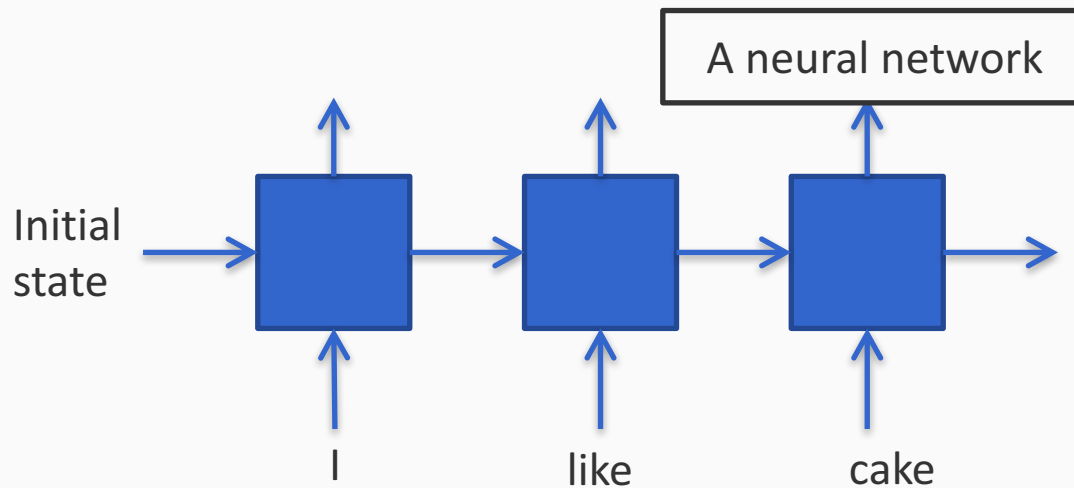
1. An Encoder

Convert a sequence into a feature vector for subsequent classification



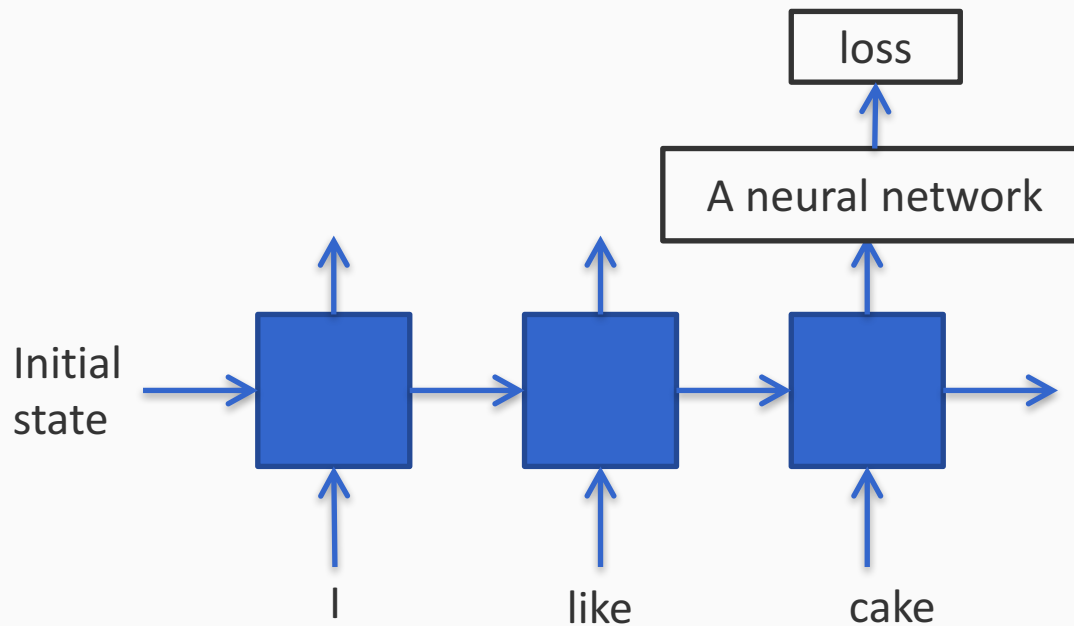
1. An Encoder

Convert a sequence into a feature vector for subsequent classification



1. An Encoder

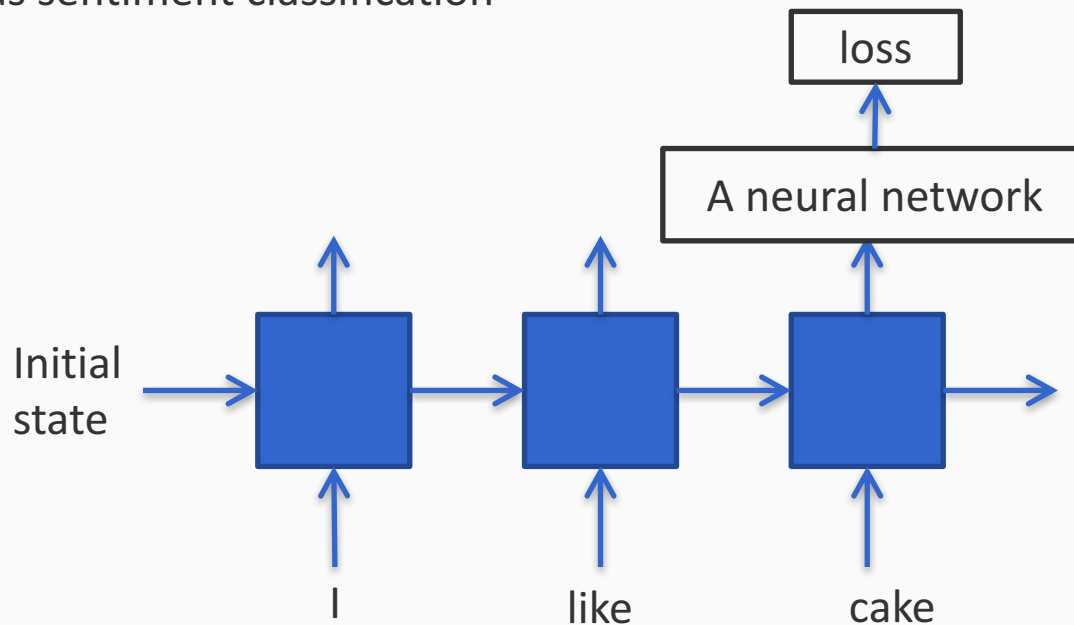
Convert a sequence into a feature vector for subsequent classification



1. An Encoder

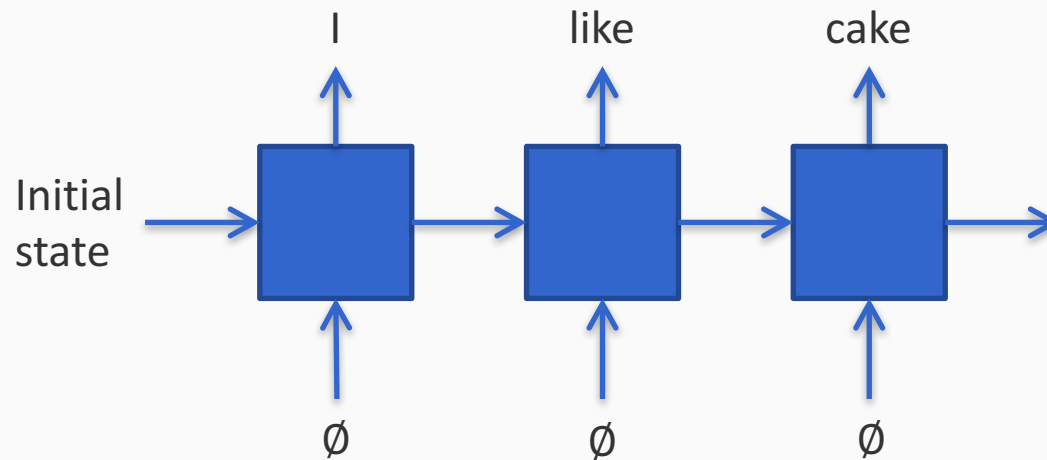
Convert a sequence into a feature vector for subsequent classification

Example: Encode a sentence or a phrase into a feature vector for a classification task such as sentiment classification



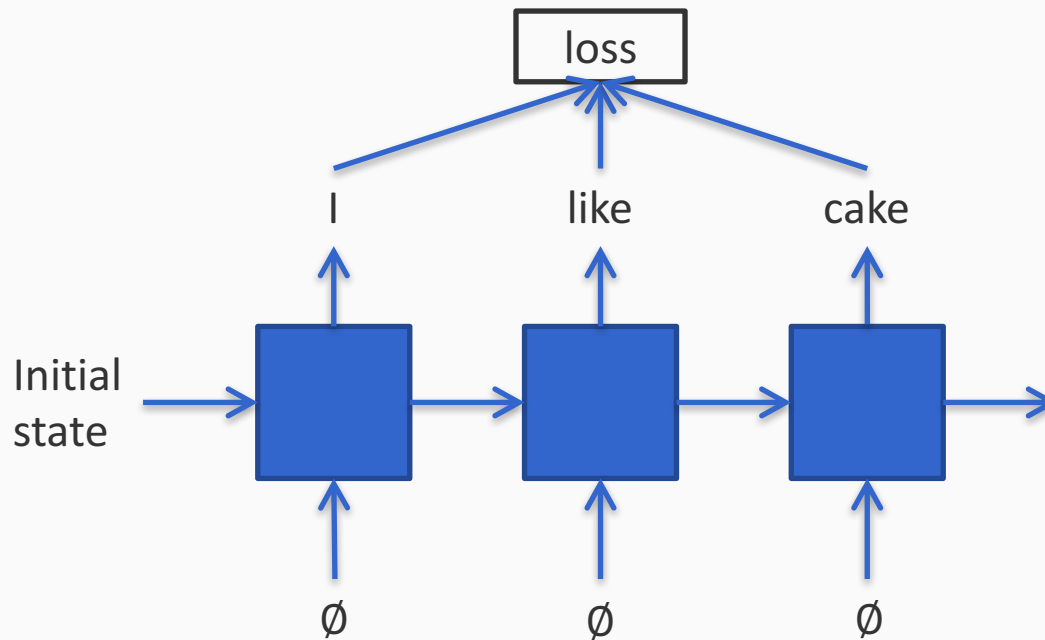
2. A Generator

Produce a sequence using an initial state



2. A Generator

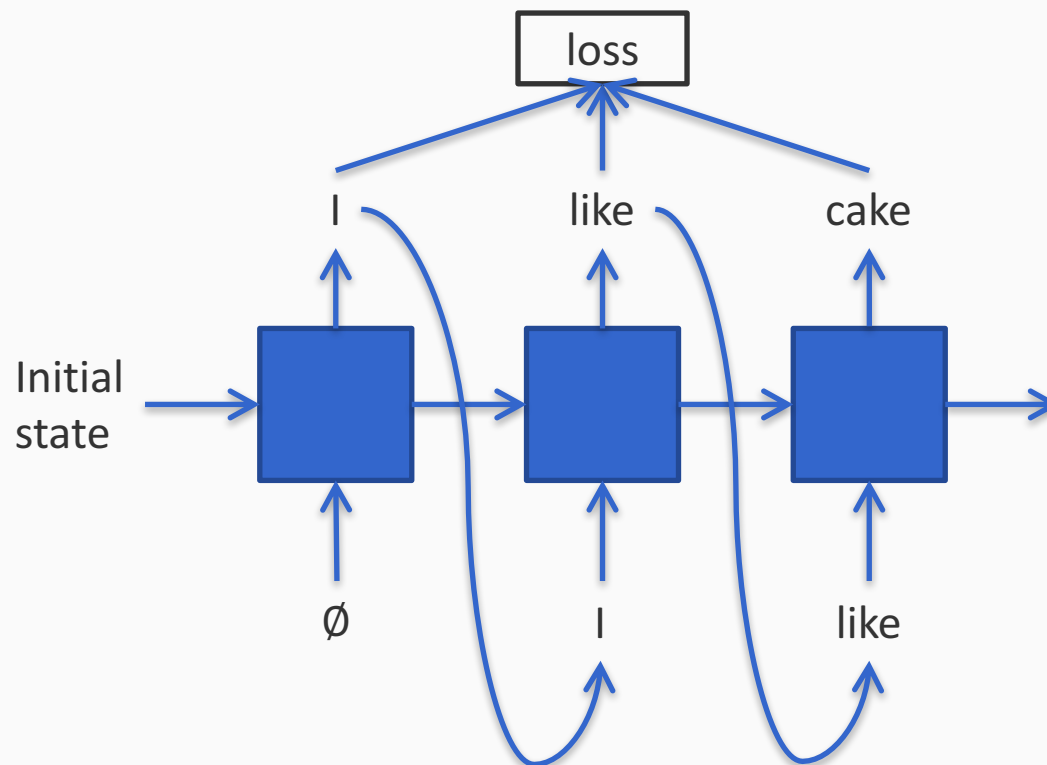
Produce a sequence using an initial state



2. A Generator

Produce a sequence using an initial state

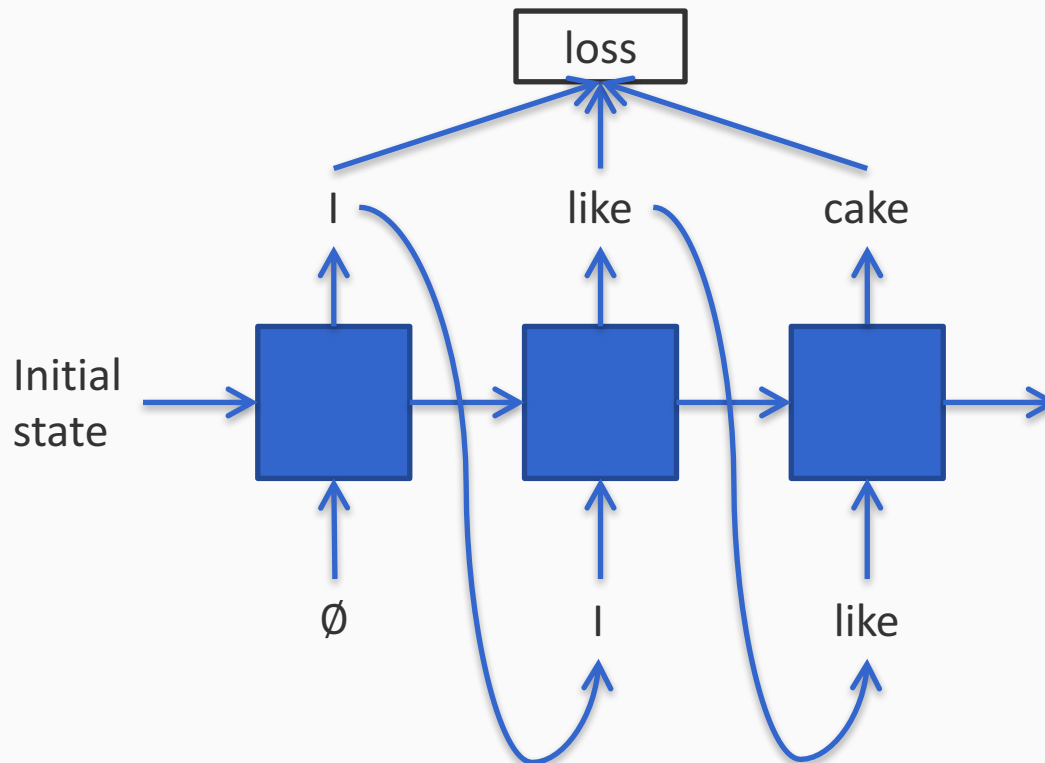
Maybe the previous output becomes the current input



2. A Generator

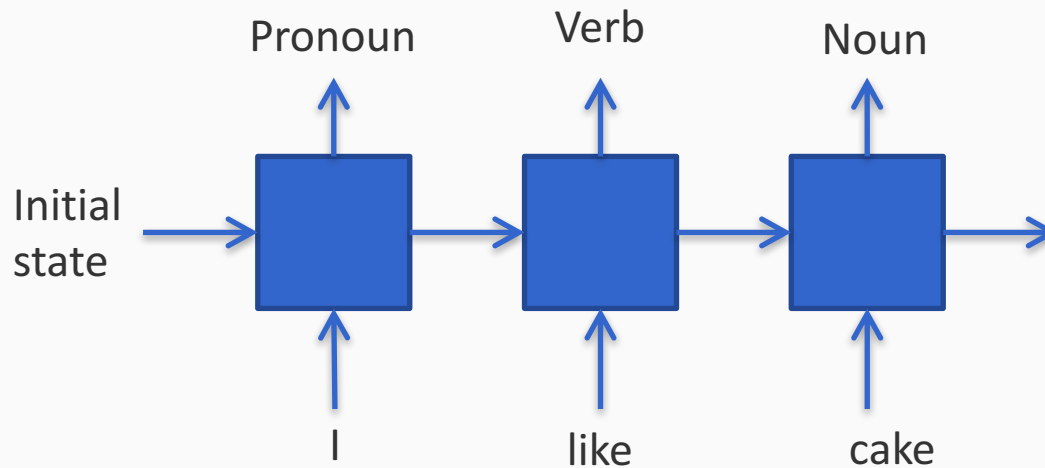
Produce a sequence using an initial state

Examples: Text generation tasks



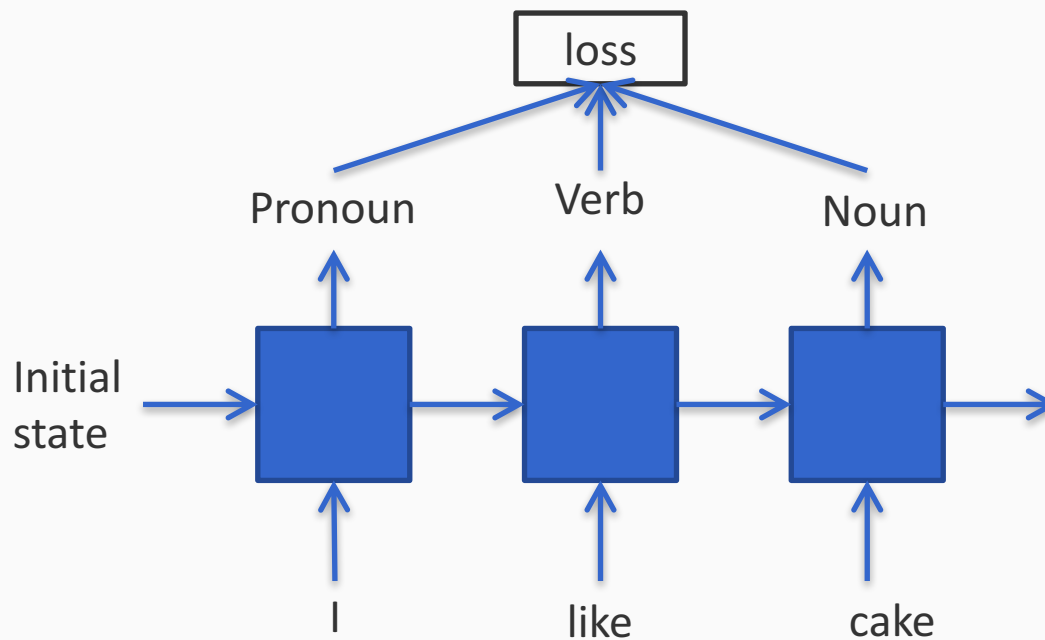
3. A Transducer

Convert a sequence into another sequence



3. A Transducer

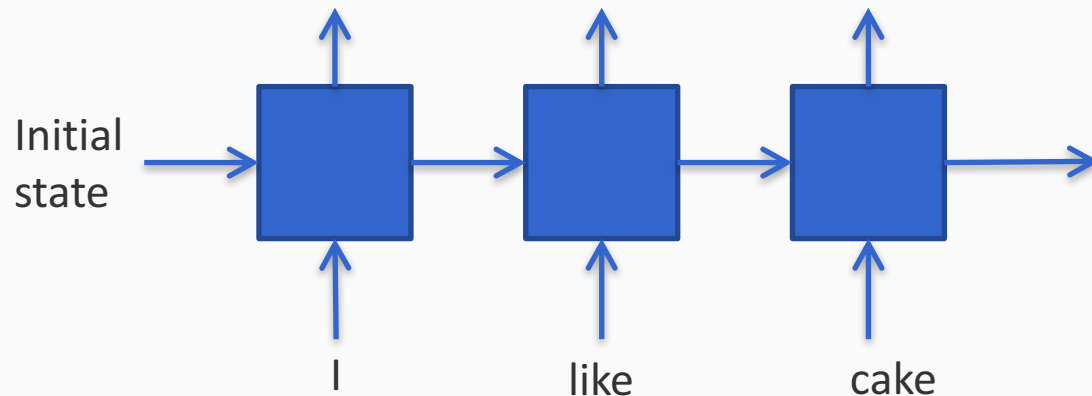
Convert a sequence into another sequence



4. Conditioned generator

Or an encoder-decoder: First encode a sequence, then generate another one

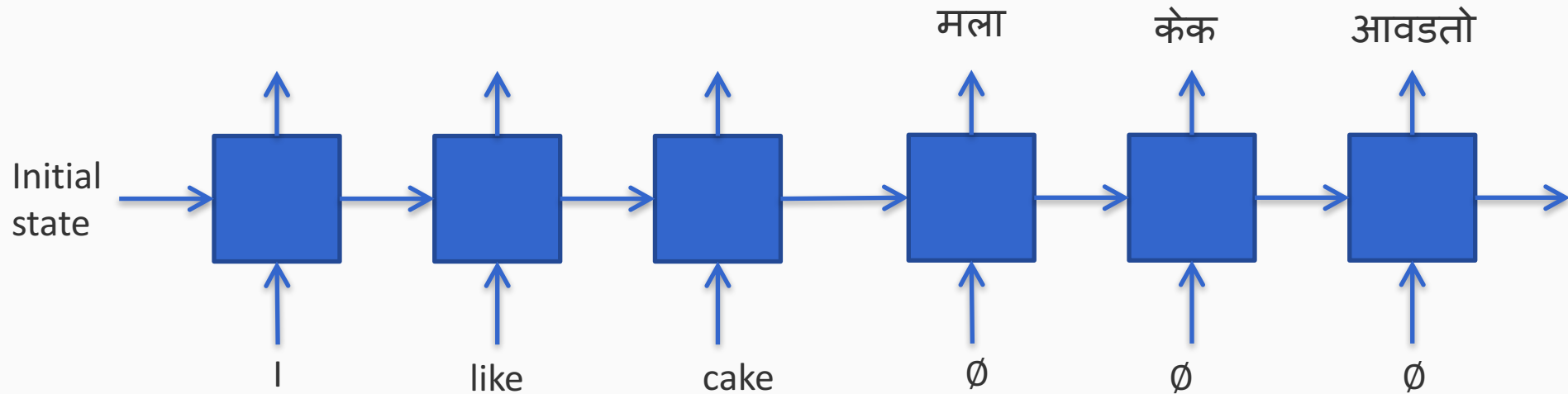
First encode a sequence



4. Conditioned generator

Or an encoder-decoder: First encode a sequence, then generate another one

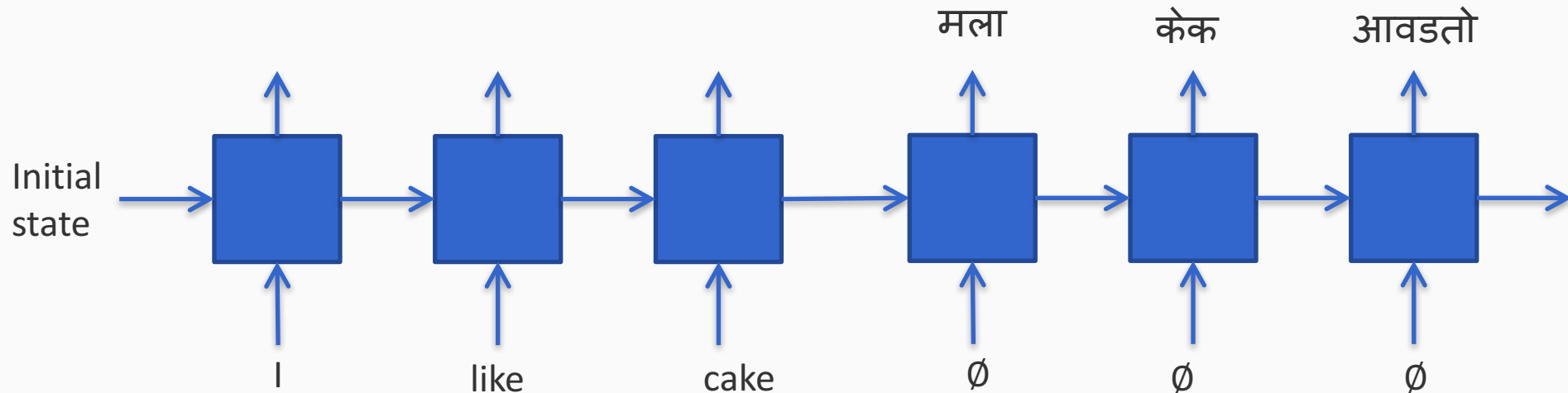
Then decode it to produce a different sequence



4. Conditioned generator

Or an encoder-decoder: First encode a sequence, then generate another one

Example: A building block for neural machine translation



Stacking RNNs

- A commonly seen usage pattern
- An RNN takes an input sequence and produces an output sequence
- The input to an RNN can itself be the output of an RNN – **stacked RNNs**, also called **deep RNNs**
- Two or more layers often seems to improve prediction performance

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

Why left to right?

Everything we saw so far models sequences (e.g. words) from left to right

Implicit assumption: If we want to represent a word in a sentence, the words before are useful

Is this right?

Why left to right?

Everything we saw so far models sequences (e.g. words) from left to right

Implicit assumption: If we want to represent a word in a sentence, the words before are useful

Is this right? **Not really**

For example: For a sequence labeling task, the words after a target word may also be useful in deciding its label

How do we address this?

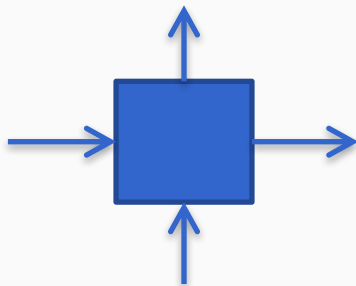
Bidirectional RNNs

[Schuster and Paliwal 1997]

One answer (currently the most popular one): Maintain two separate RNNs – one forward and one reverse

BiRNN: A simple example *Forward*

John ate cake

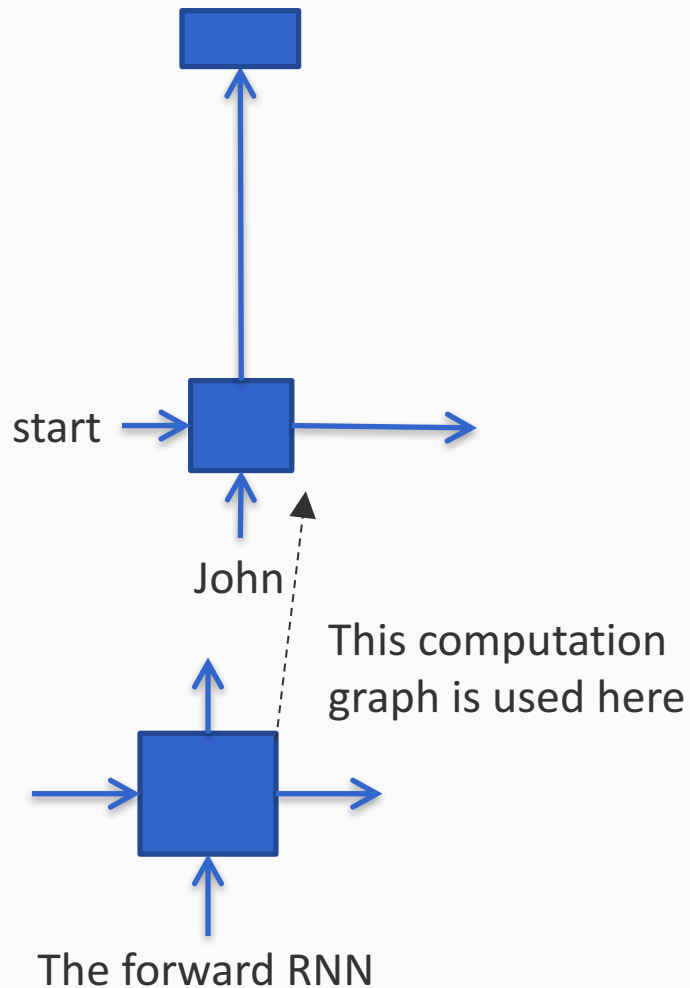


The forward RNN

First, the forward case. We have seen this before.

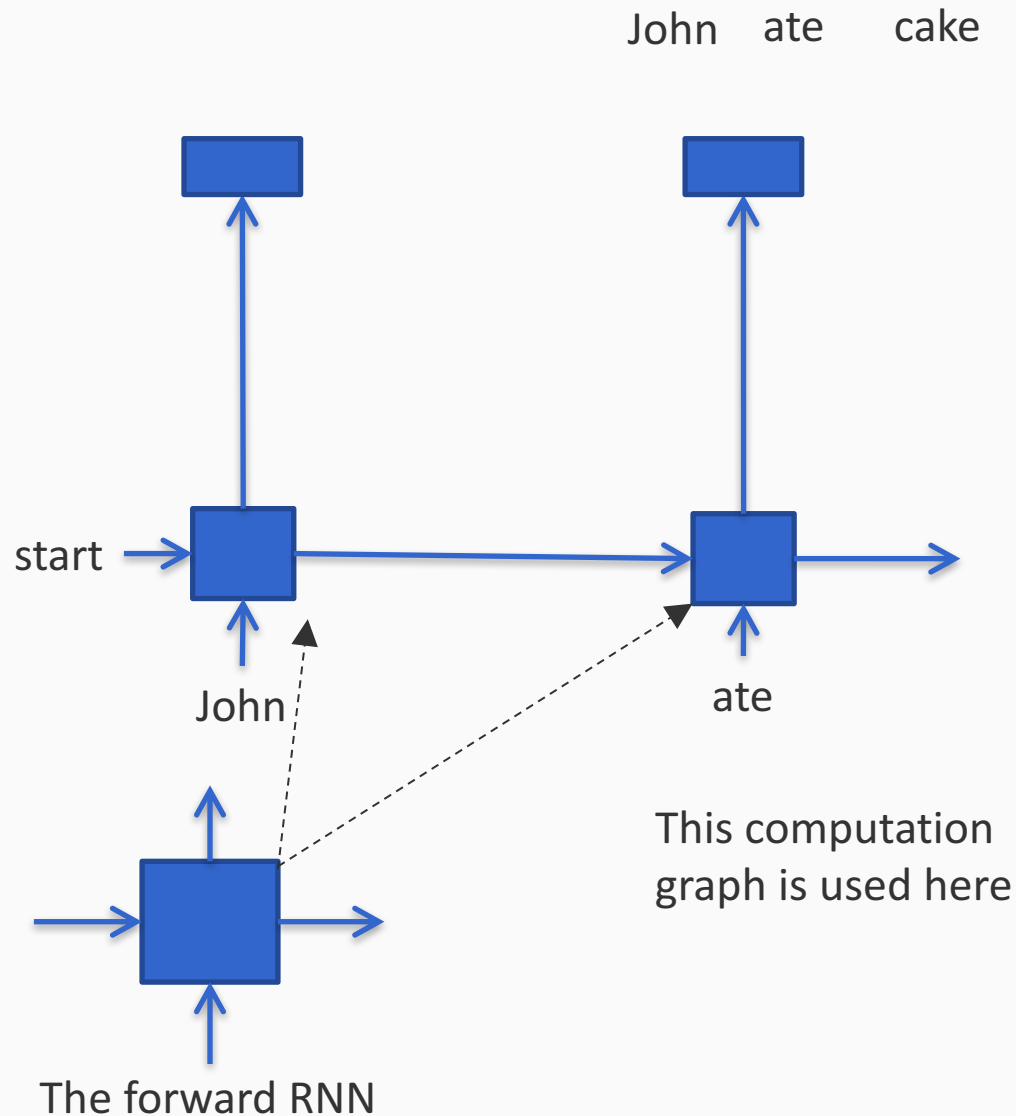
BiRNN: A simple example *Forward*

John ate cake



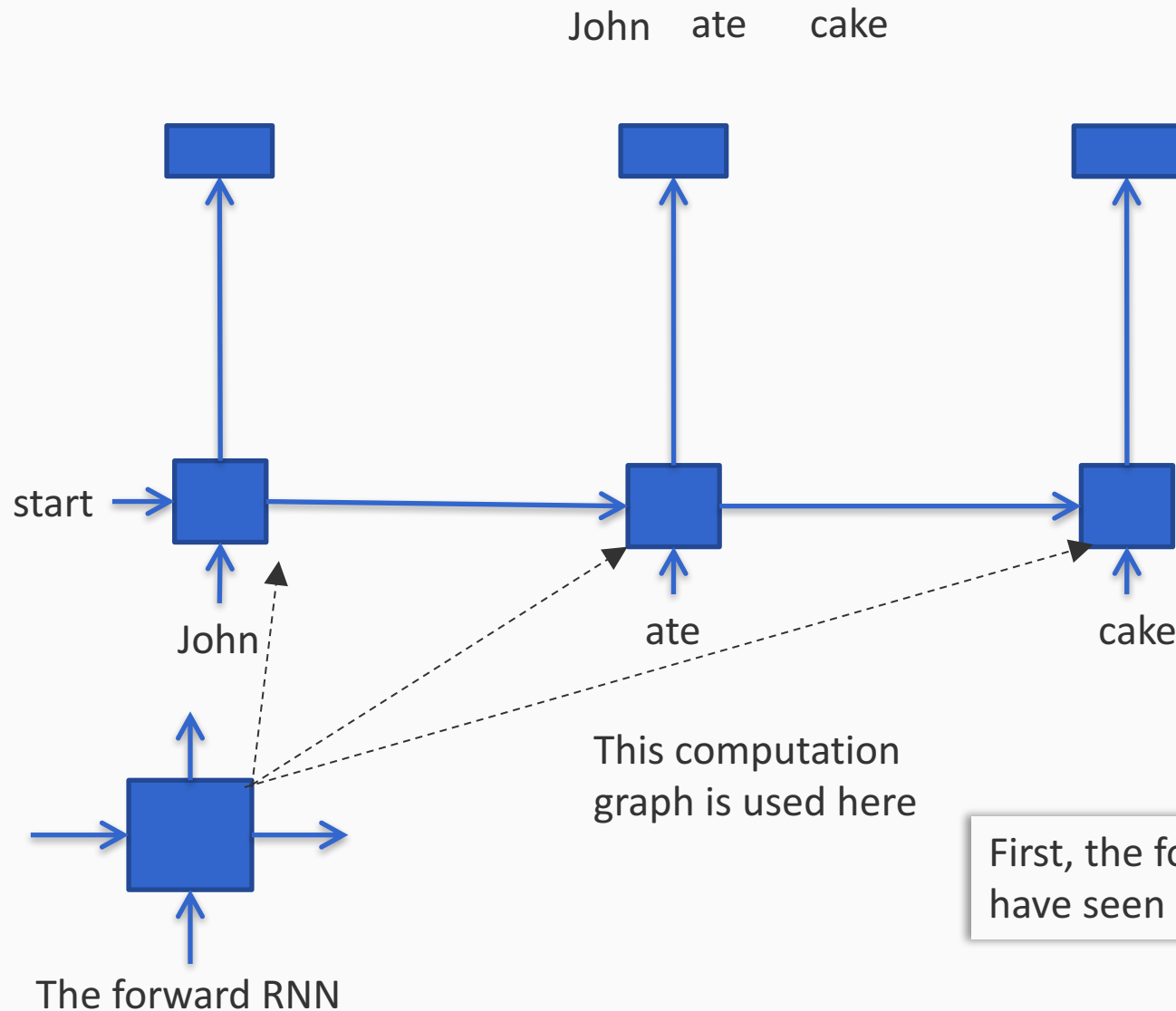
First, the forward case. We have seen this before.

BiRNN: A simple example *Forward*

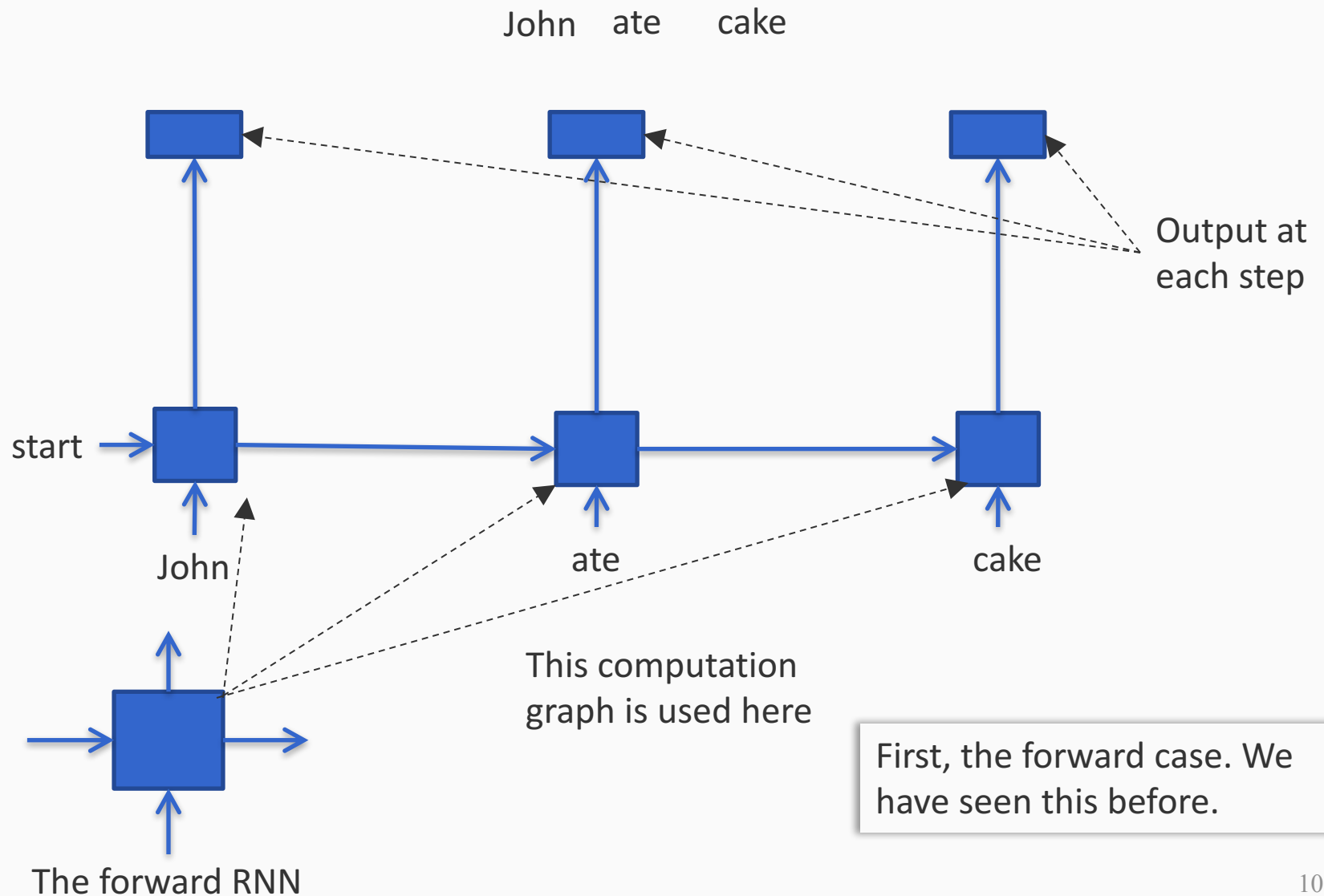


First, the forward case. We have seen this before.

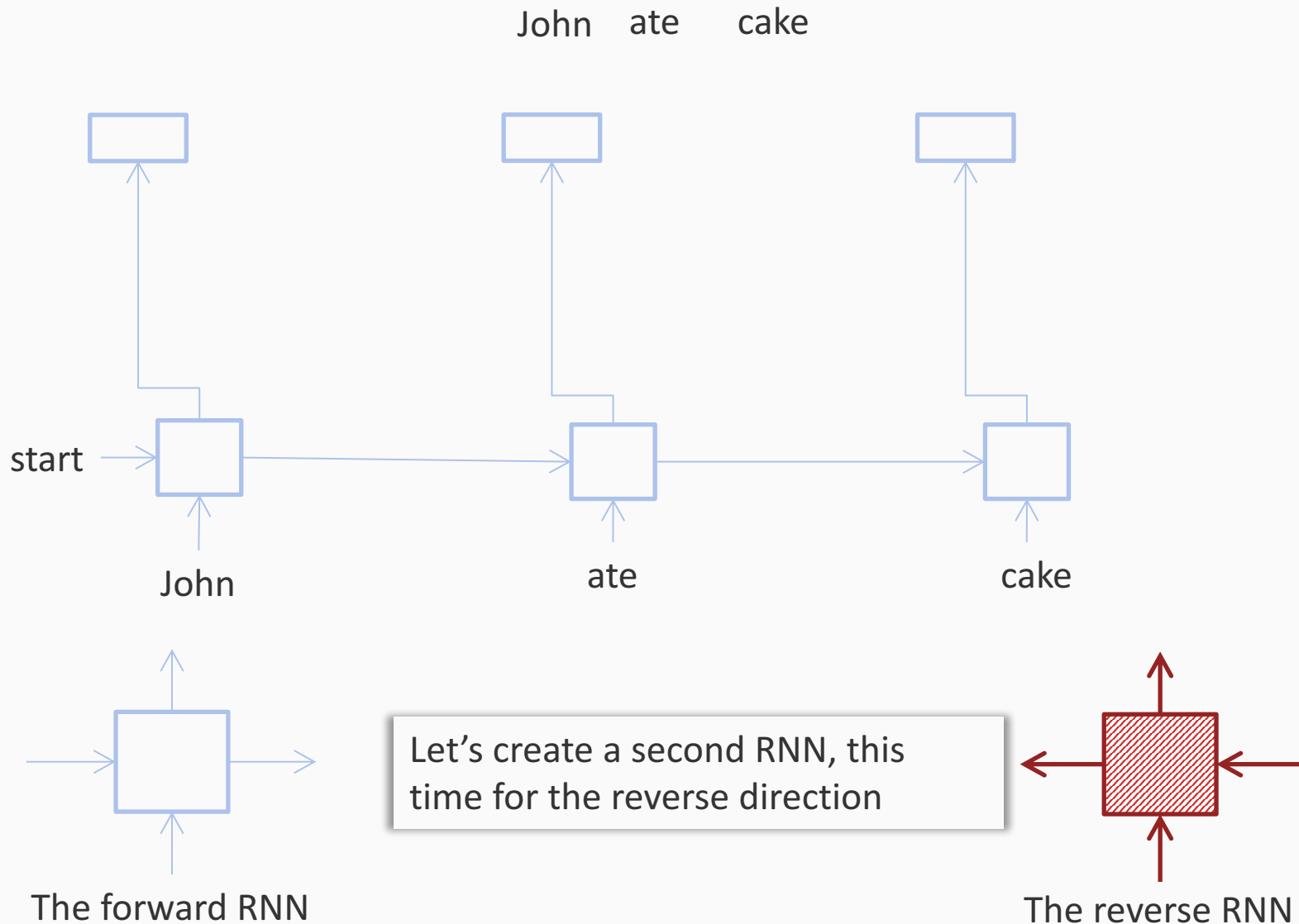
BiRNN: A simple example *Forward*



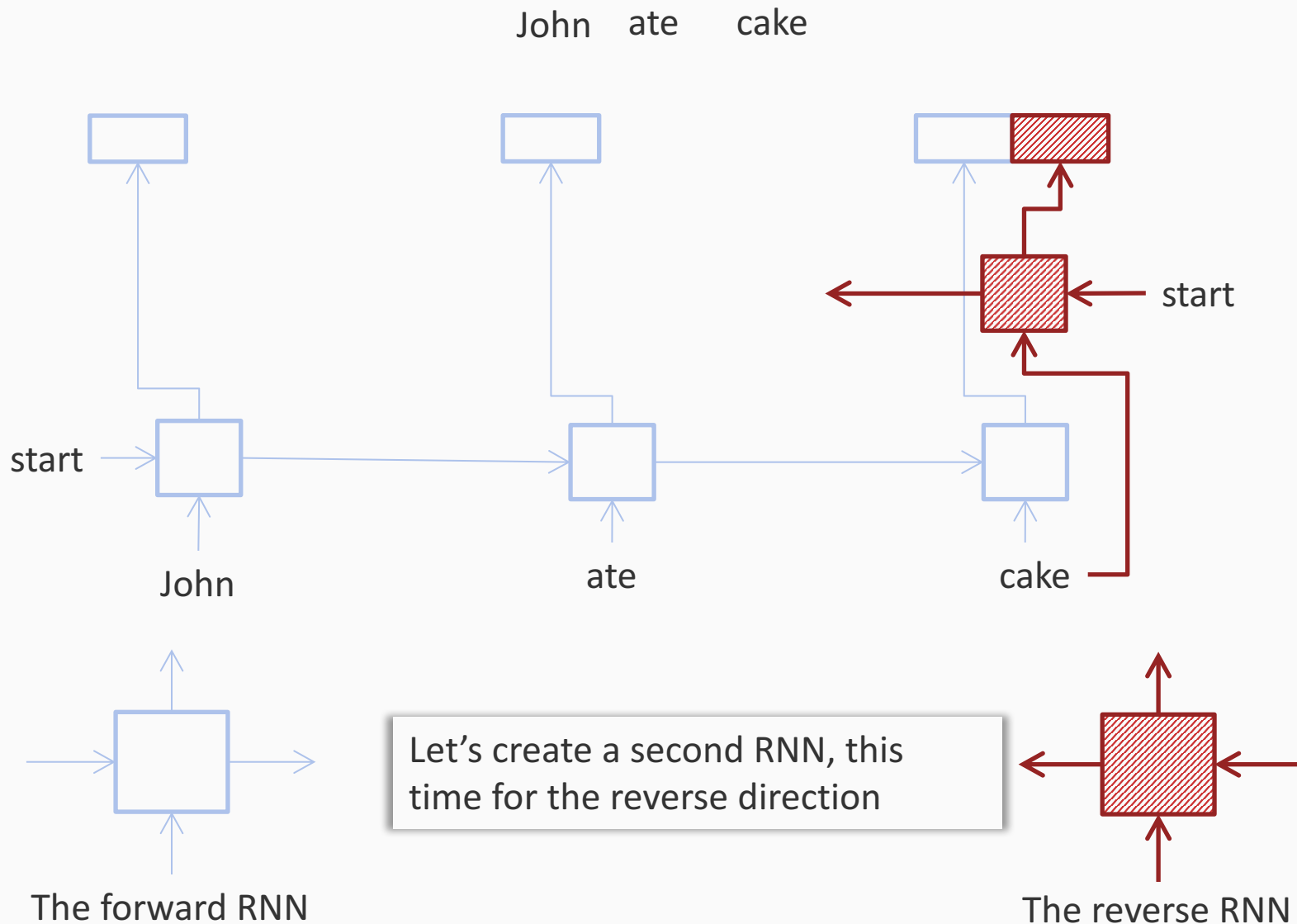
BiRNN: A simple example *Forward*



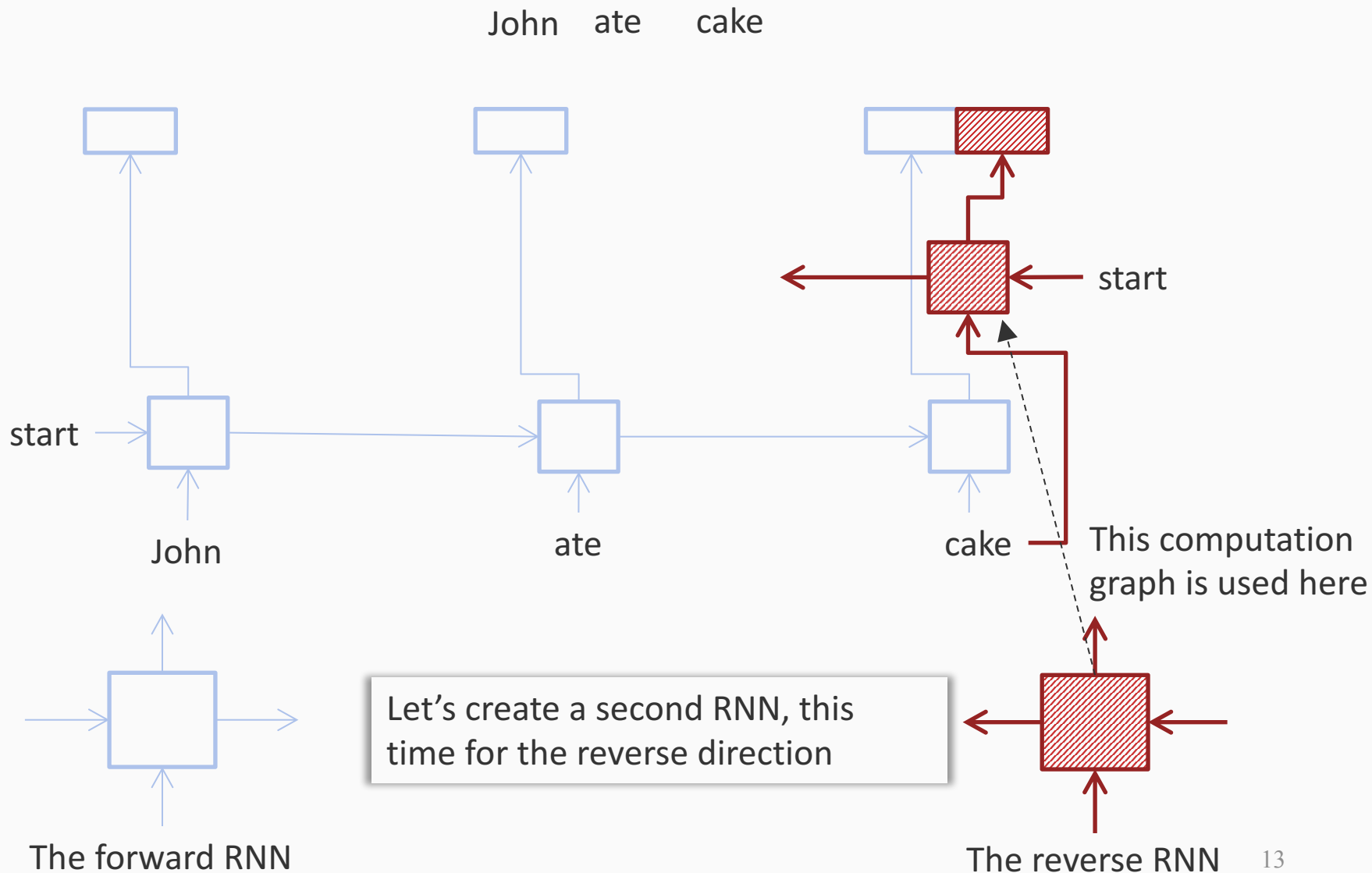
BiRNN: A simple example *Reverse*



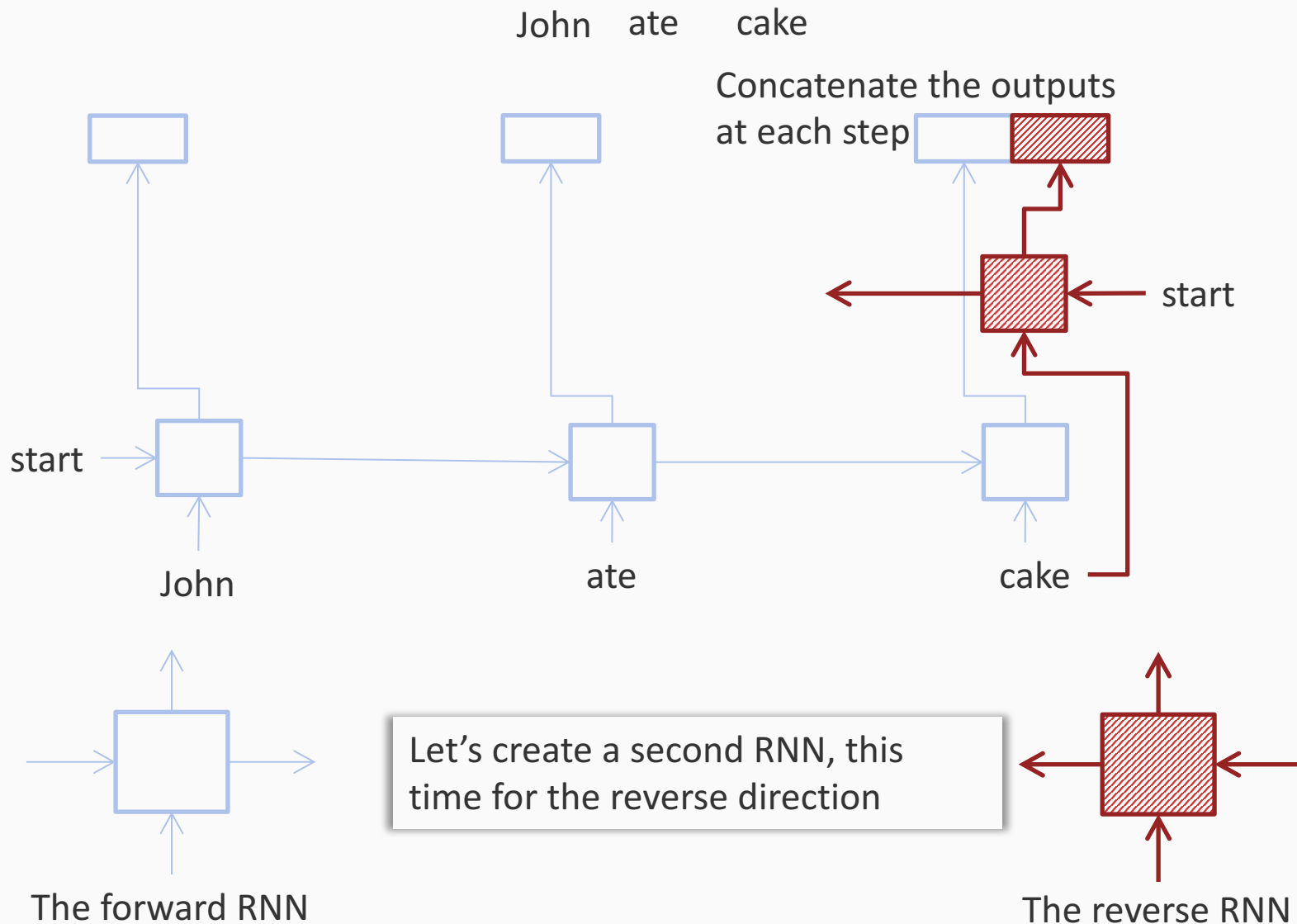
BiRNN: A simple example *Reverse*



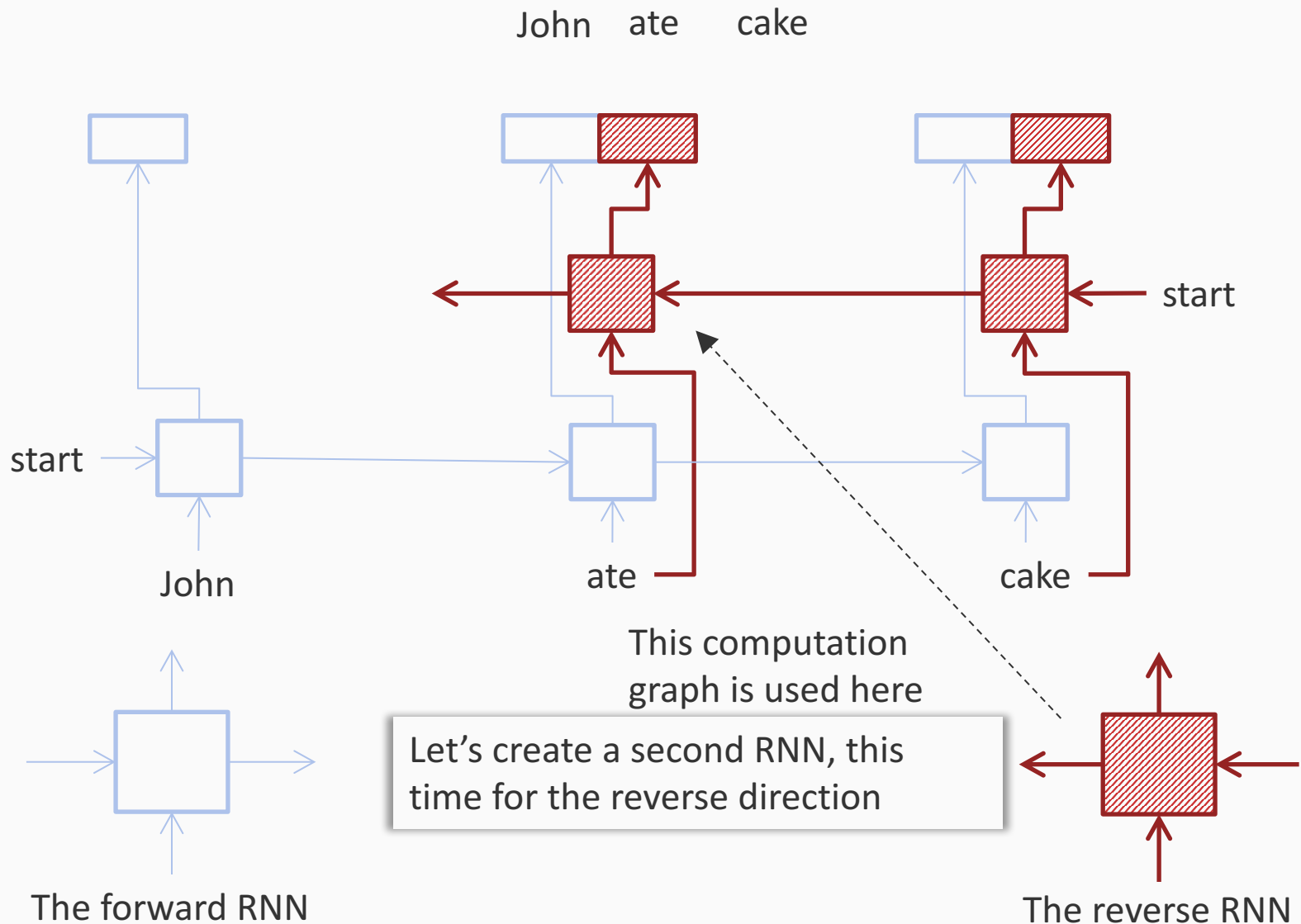
BiRNN: A simple example *Reverse*



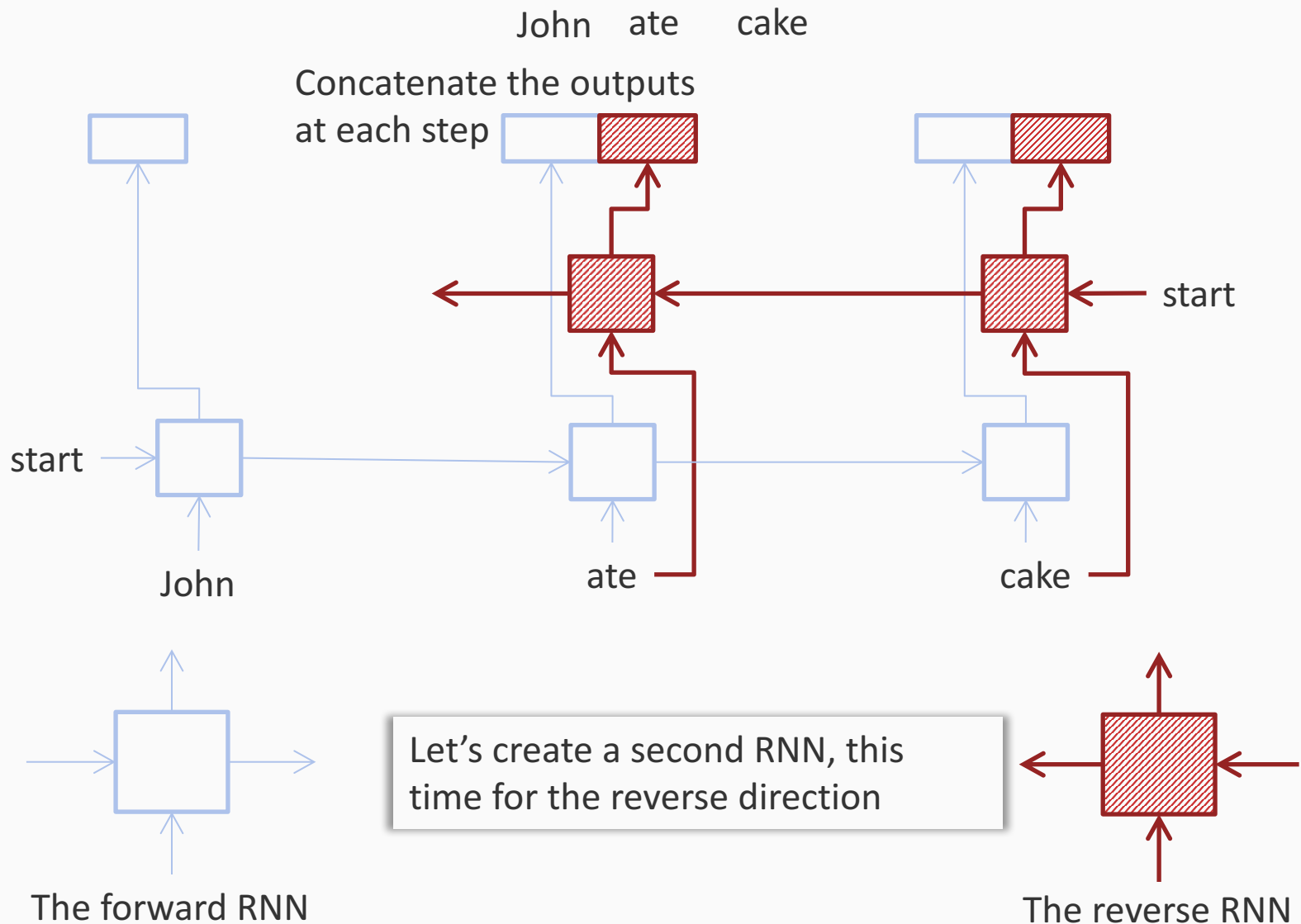
BiRNN: A simple example *Reverse*



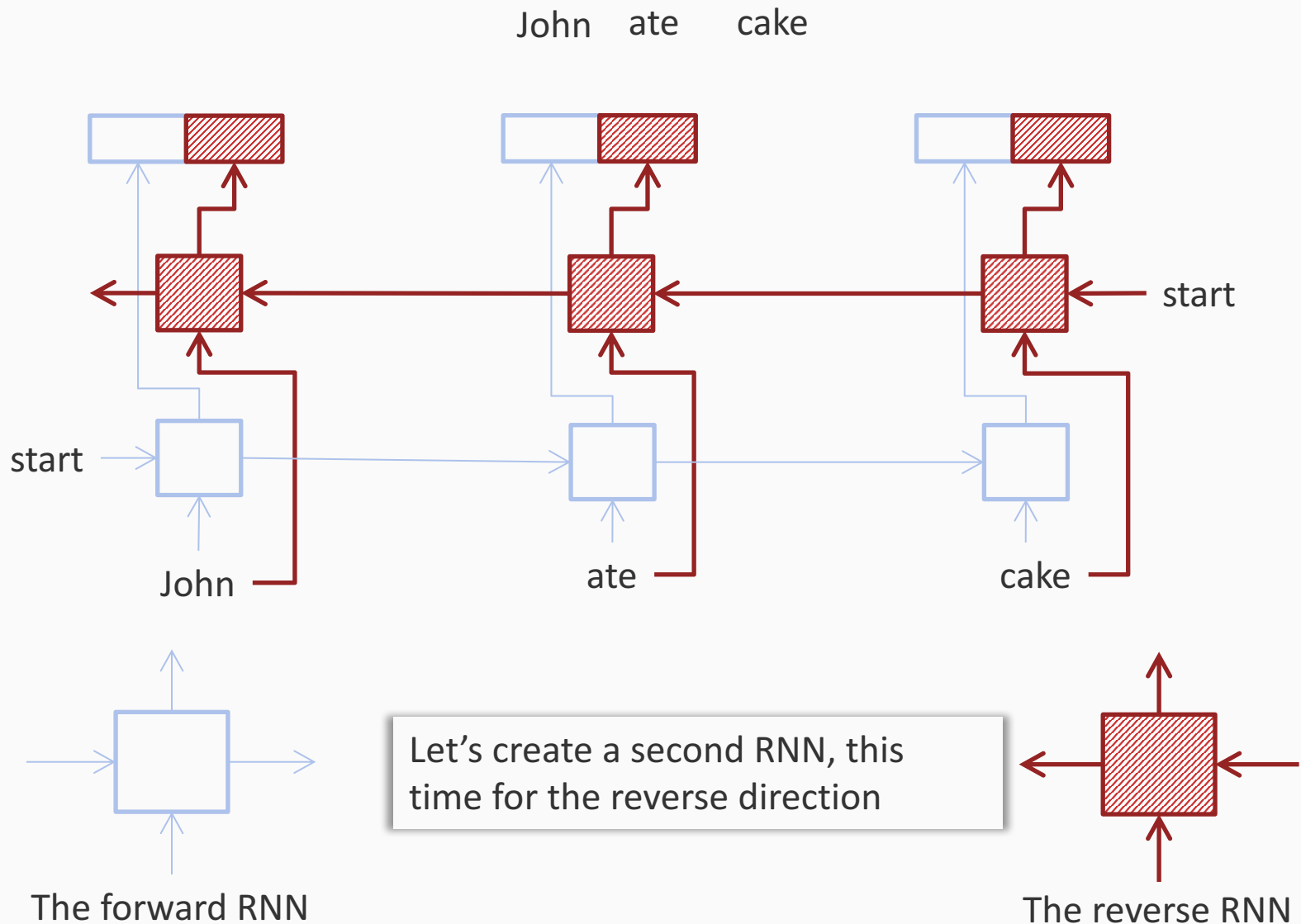
BiRNN: A simple example *Reverse*



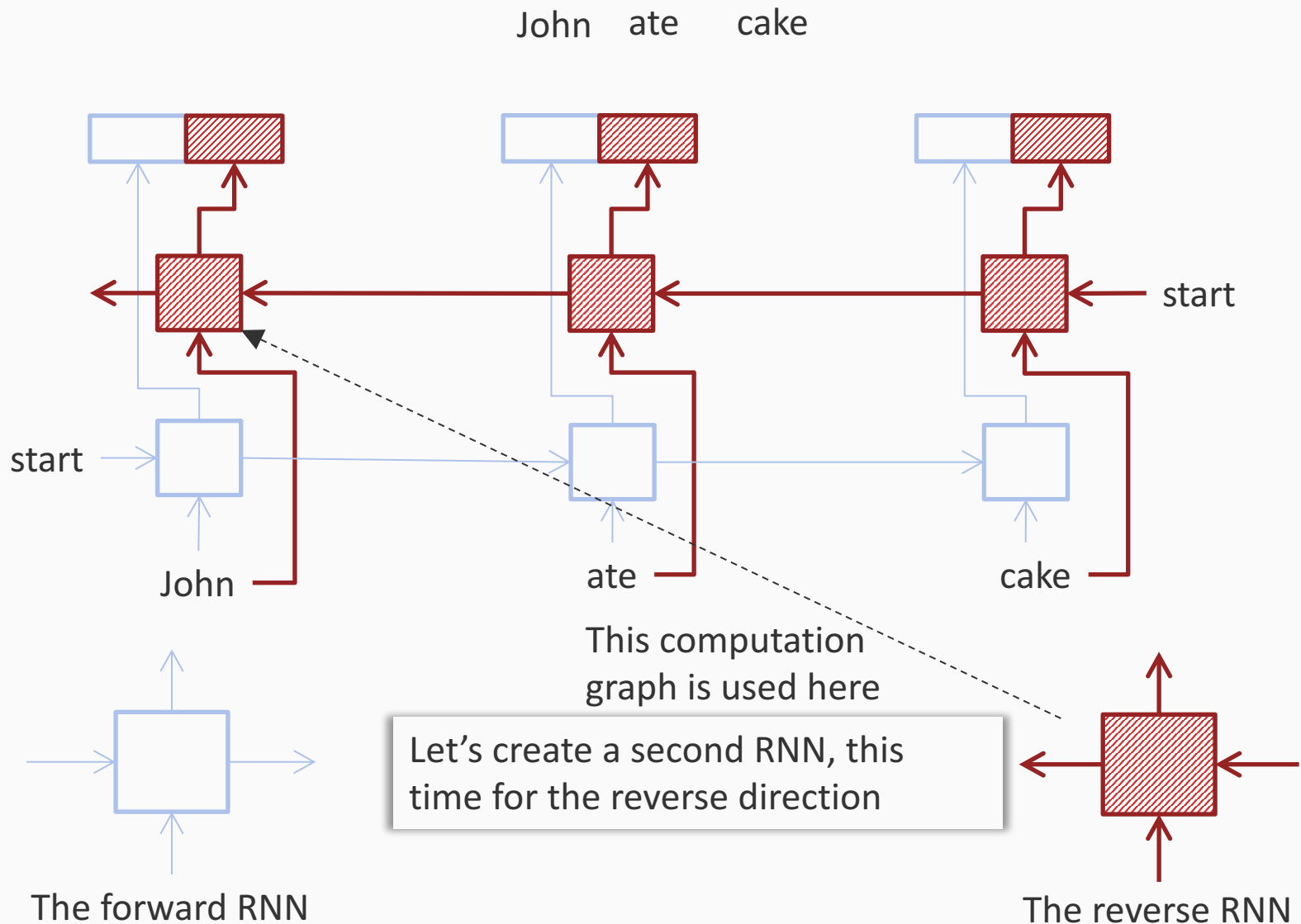
BiRNN: A simple example *Reverse*



BiRNN: A simple example *Reverse*

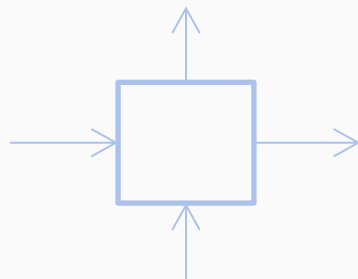
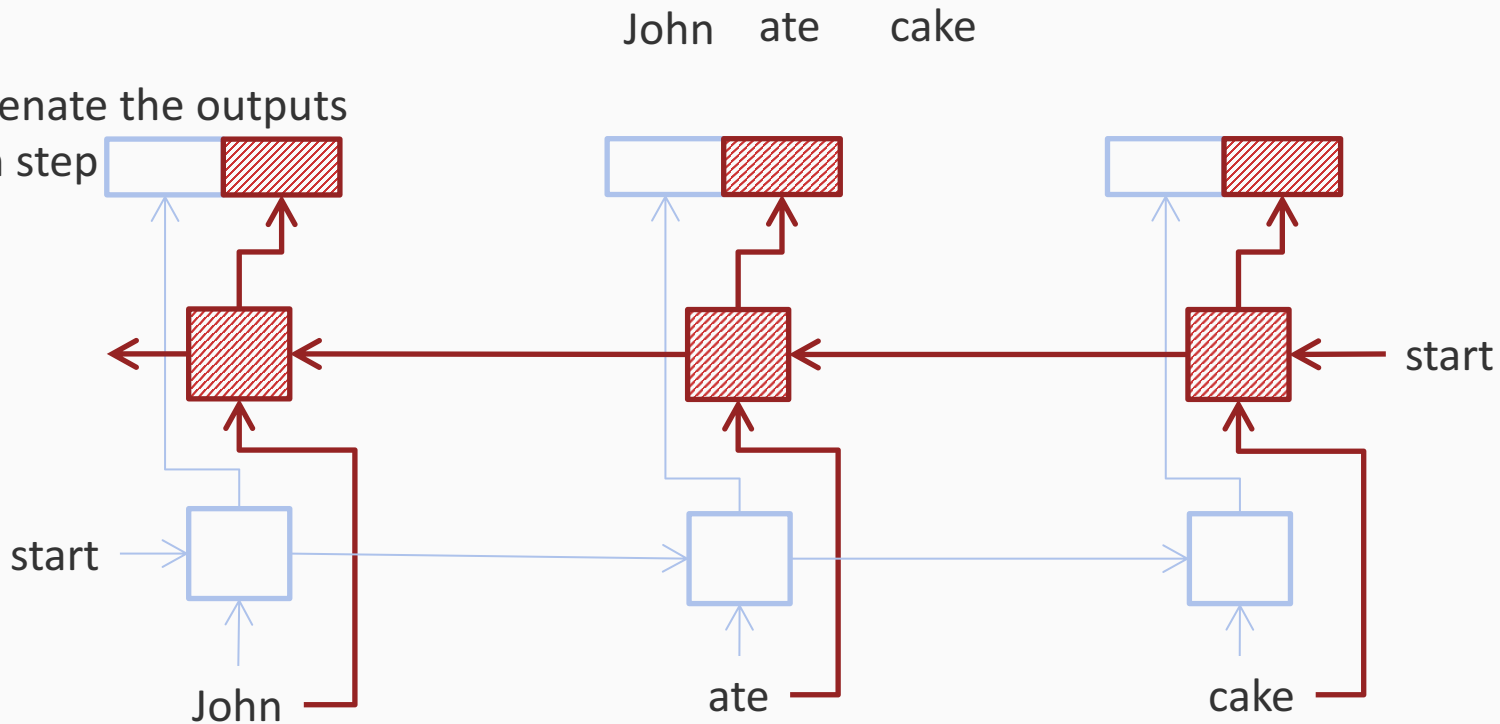


BiRNN: A simple example *Reverse*



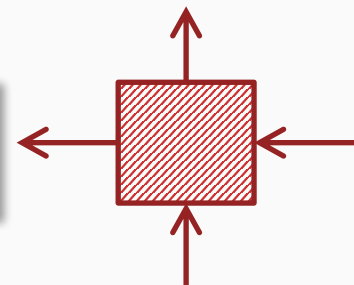
BiRNN: A simple example *Reverse*

Concatenate the outputs
at each step



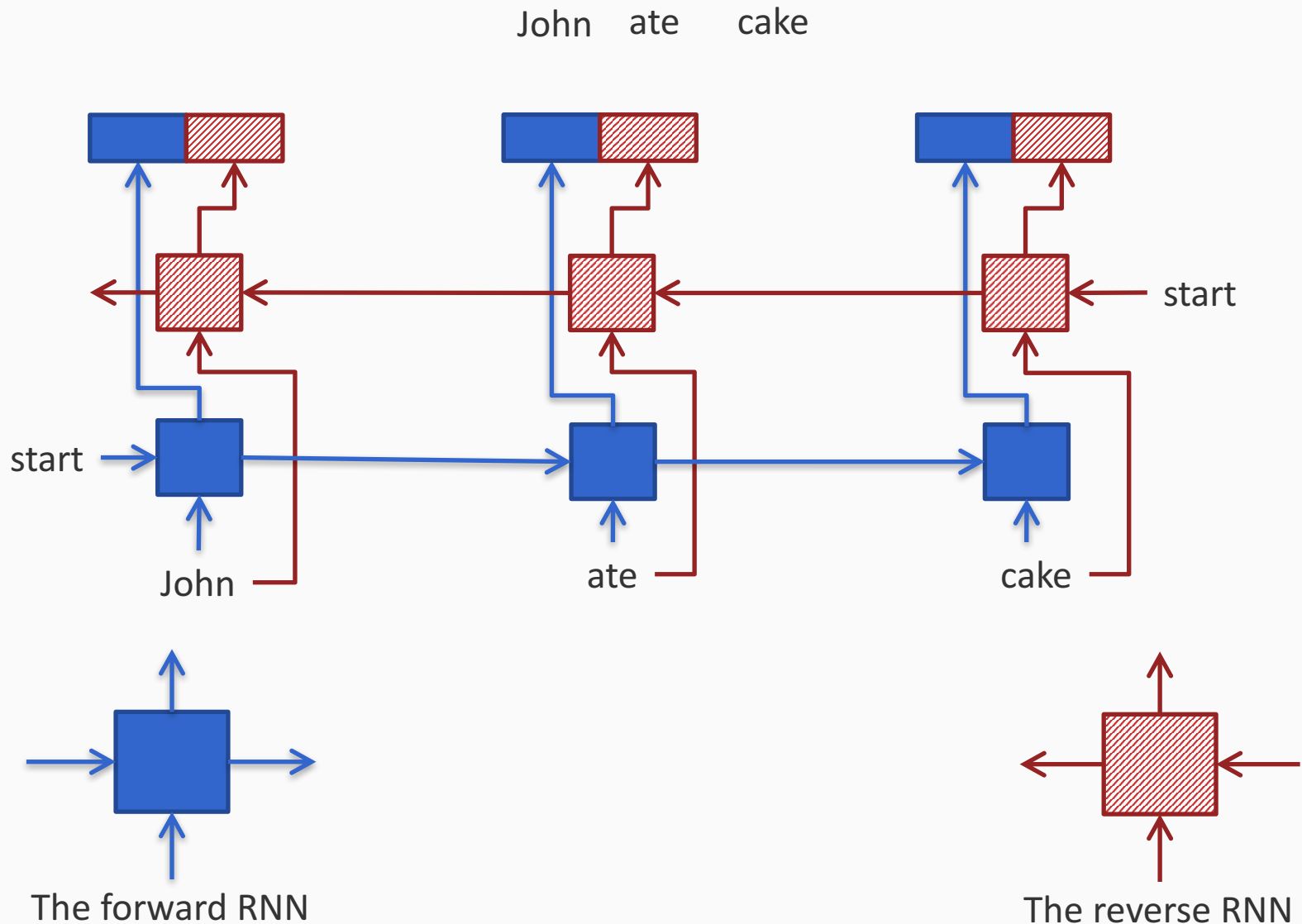
The forward RNN

Let's create a second RNN, this
time for the reverse direction



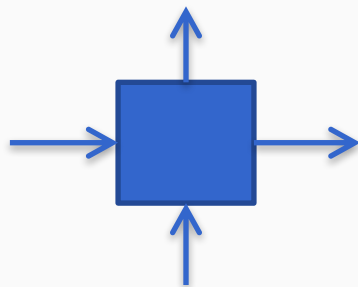
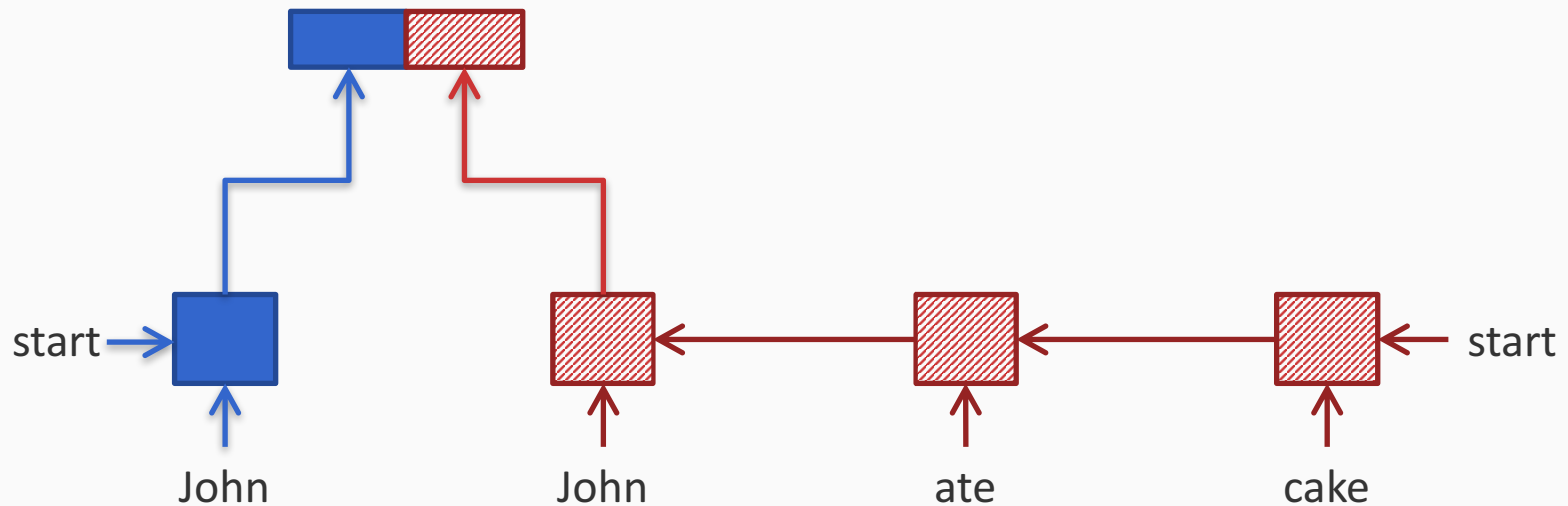
The reverse RNN

BiRNN: Putting both parts together

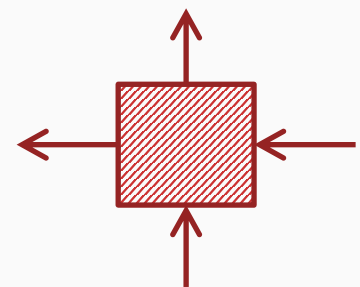


Another way of seeing this

Concatenate to get the representation for the word *John* that accounts for both left and right contexts



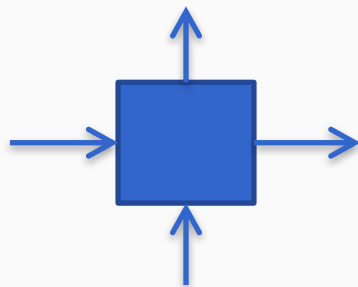
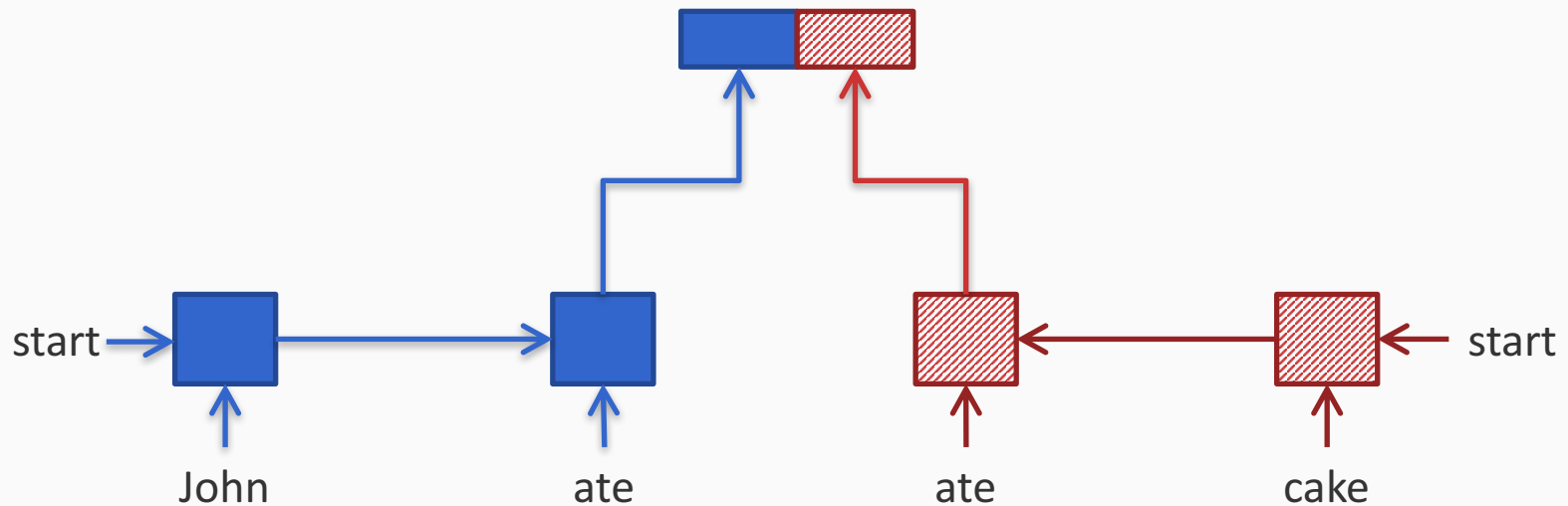
The forward RNN



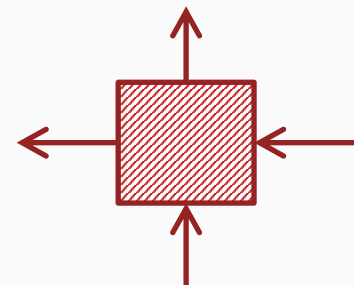
The reverse RNN

Another way of seeing this

Concatenate to get the representation for the word *ate* that accounts for both left and right contexts



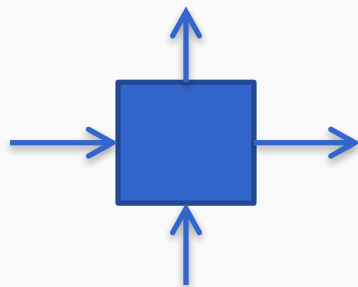
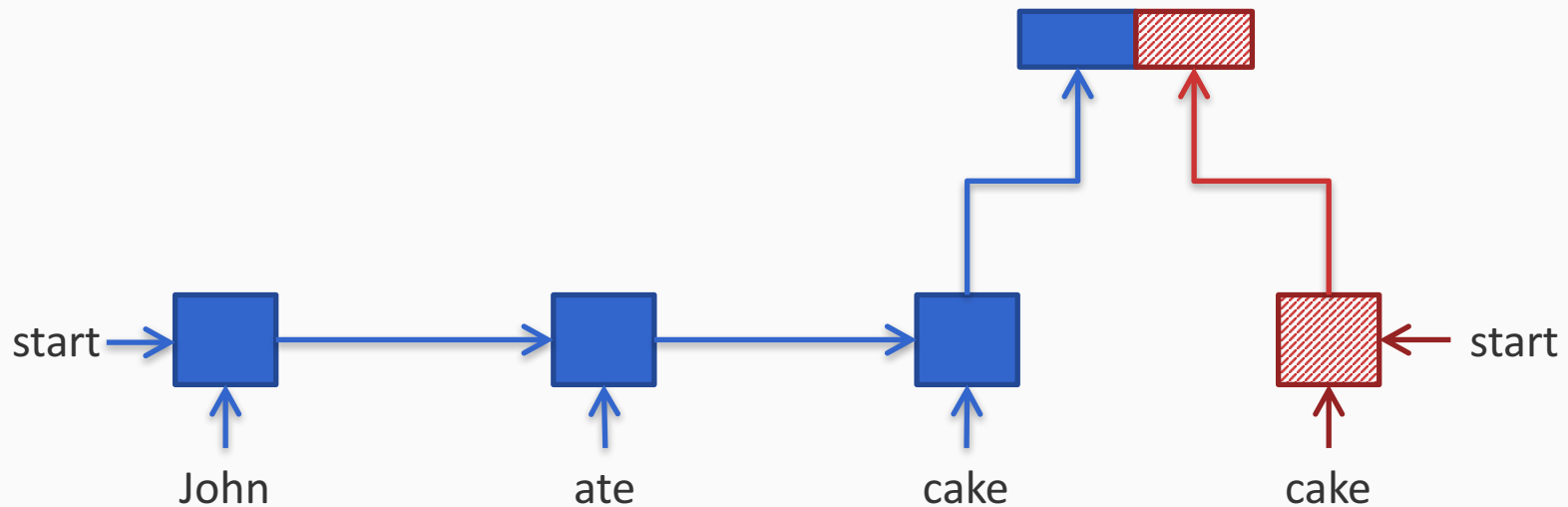
The forward RNN



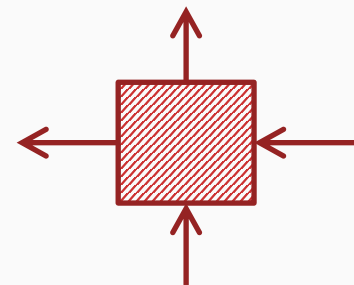
The reverse RNN

Another way of seeing this

Concatenate to get the representation for the word *cake* that accounts for both left and right contexts



The forward RNN



The reverse RNN

A Bidirectional RNN

- Two RNNs
 - **Forward**, defined by functions $R^f(\mathbf{s}_{t-1}^f, \mathbf{x}_t)$ and $O^f(\mathbf{s}_t)$
 - **Backward**, defined by functions $R^b(\mathbf{s}_{t+1}^b, \mathbf{x}_t)$ and $O^b(\mathbf{s}_t)$

A Bidirectional RNN

- Two RNNs
 - **Forward**, defined by functions $R^f(\mathbf{s}_{t-1}^f, \mathbf{x}_t)$ and $O^f(\mathbf{s}_t)$
 - **Backward**, defined by functions $R^b(\mathbf{s}_{t+1}^b, \mathbf{x}_t)$ and $O^b(\mathbf{s}_t)$
- The i^{th} output is defined by
$$\mathbf{y}_i = [O^f(\mathbf{s}_t^f), O^b(\mathbf{s}_t^b)]$$

A Bidirectional RNN

- Two RNNs
 - **Forward**, defined by functions $R^f(\mathbf{s}_{t-1}^f, \mathbf{x}_t)$ and $O^f(\mathbf{s}_t)$
 - **Backward**, defined by functions $R^b(\mathbf{s}_{t+1}^b, \mathbf{x}_t)$ and $O^b(\mathbf{s}_t)$
- The i^{th} output is defined by
$$\mathbf{y}_i = [O^f(\mathbf{s}_t^f), O^b(\mathbf{s}_t^b)]$$
- Another way to write this
$$\text{biRNN}(\mathbf{x}_{1:n}, t) = [\text{RNN}^f(\mathbf{x}_{1:t}), \text{RNN}^b(\mathbf{x}_{n:t})]$$

BiRNNs: Summary

- Allows capturing both left and right contexts
- Commonly used today as a base encoding layer for a variety of NLP tasks
 - Often stacked
- Specific versions of RNNs give us different BiRNNs
 - BiLSTMs or BiGRUs are typically used

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

A simple RNN

- What we saw so far is just a template for a recurrent neural network
 - Did not specify what the functions inside it are
- Let's look at a simple instantiation, first introduced by Elman 1990

A simple RNN

At each step, an RNN:

- Computes the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
- Computes the output: $\mathbf{y}_t = O(\mathbf{s}_t)$

Need to specify two functions:

1. How to generate the current state using the previous state and the current input?
2. How to generate the current output using the current state?

A simple RNN

At each step, an RNN:

- Computes the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
- Computes the output: $\mathbf{y}_t = O(\mathbf{s}_t)$

Need to specify two functions:

1. How to generate the current state using the previous state and the current input?
2. How to generate the current output using the current state?

The output is the state. That is, $\mathbf{y}_t = \mathbf{s}_t$

Computing the value of a state

1. How to generate the current state using the previous state and the current input?

\mathbf{s}_{t-1}

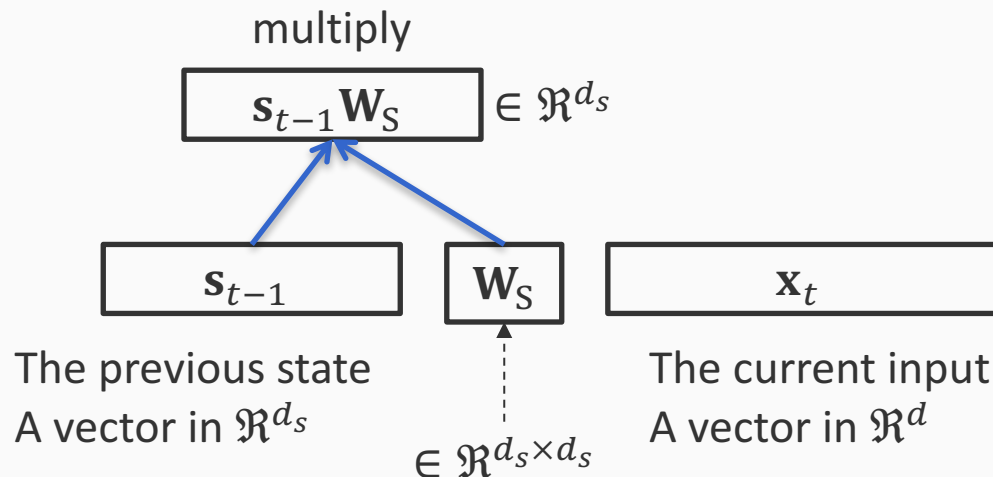
The previous state
A vector in \mathfrak{R}^{d_s}

\mathbf{x}_t

The current input
A vector in \mathfrak{R}^d

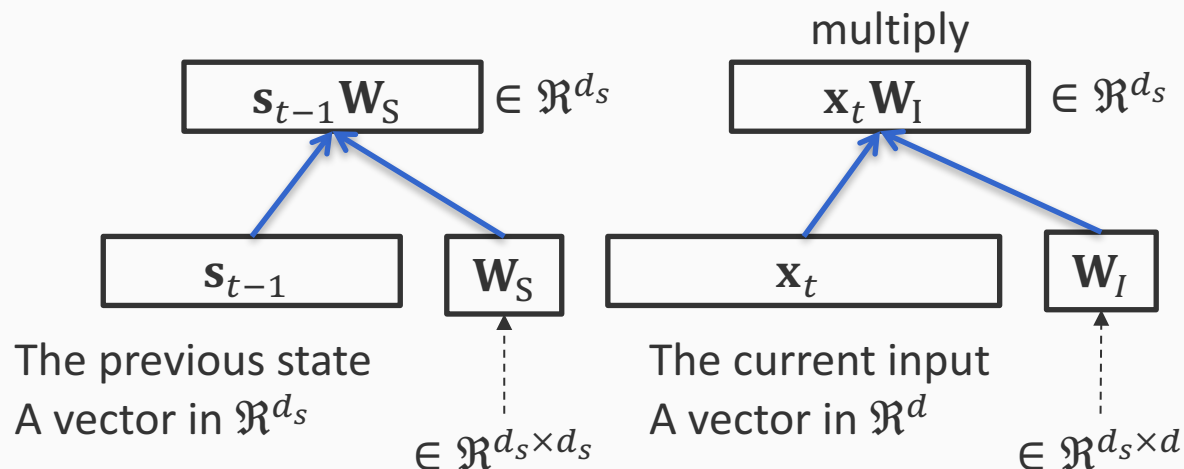
Computing the value of a state

1. How to generate the current state using the previous state and the current input?



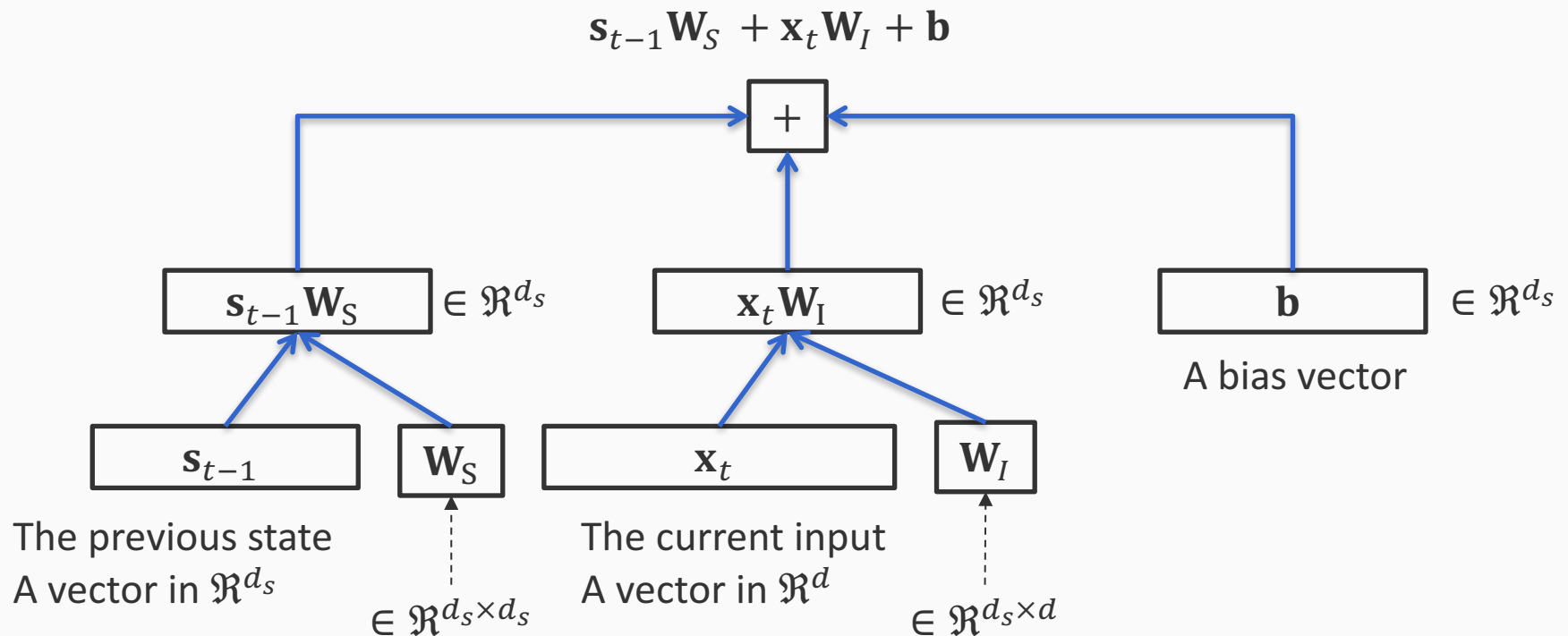
Computing the value of a state

1. How to generate the current state using the previous state and the current input?



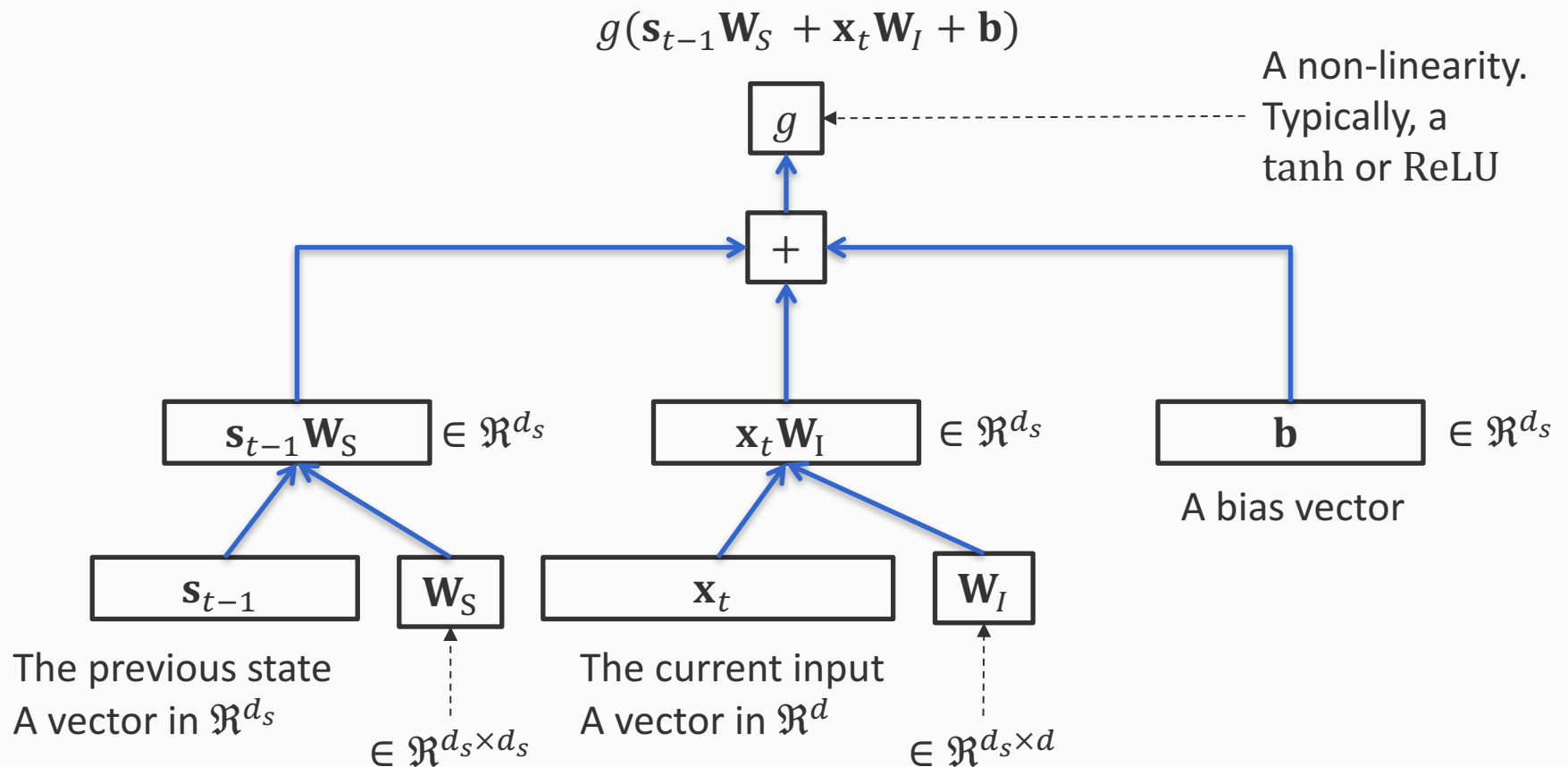
Computing the value of a state

1. How to generate the current state using the previous state and the current input?



Computing the value of a state

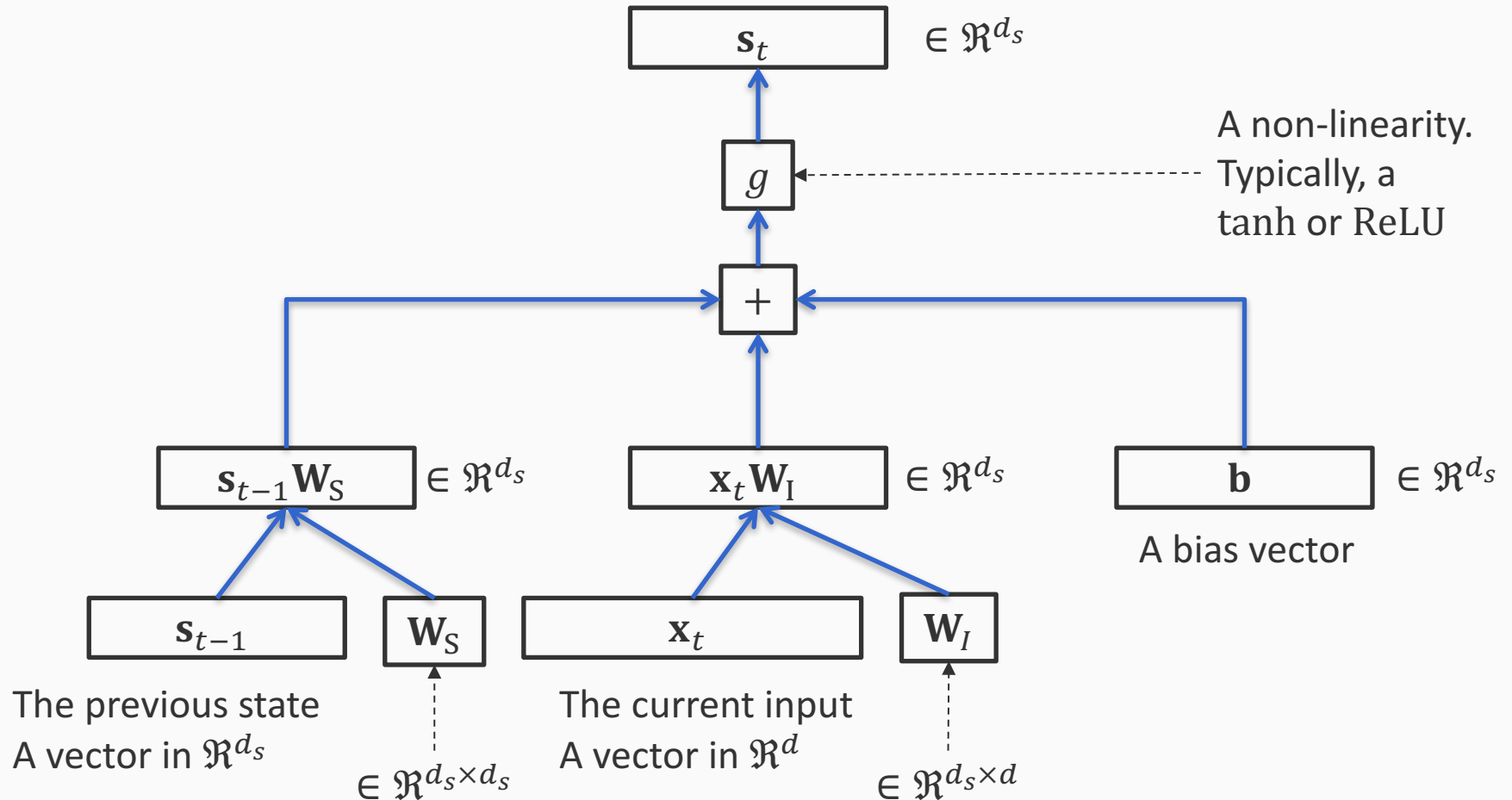
1. How to generate the current state using the previous state and the current input?



Computing the value of a state

1. How to generate the current state using the previous state and the current input?

$$\text{Next state } \mathbf{s}_t = g(\mathbf{s}_{t-1} \mathbf{W}_S + \mathbf{x}_t \mathbf{W}_I + \mathbf{b})$$



A simple RNN

At each step, an RNN:

- Computes the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
- Computes the output: $\mathbf{y}_t = O(\mathbf{s}_t)$

Need to specify two functions:

1. How to generate the current state using the previous state and the current input?
2. How to generate the current output using the current state?

The output is the state. That is, $\mathbf{y}_t = \mathbf{s}_t$

A simple RNN

At each step, an RNN:

- Computes the next cell state: $\mathbf{s}_t = R(\mathbf{s}_{t-1}, \mathbf{x}_t)$
- Computes the output: $\mathbf{y}_t = O(\mathbf{s}_t)$

Need to specify two functions:

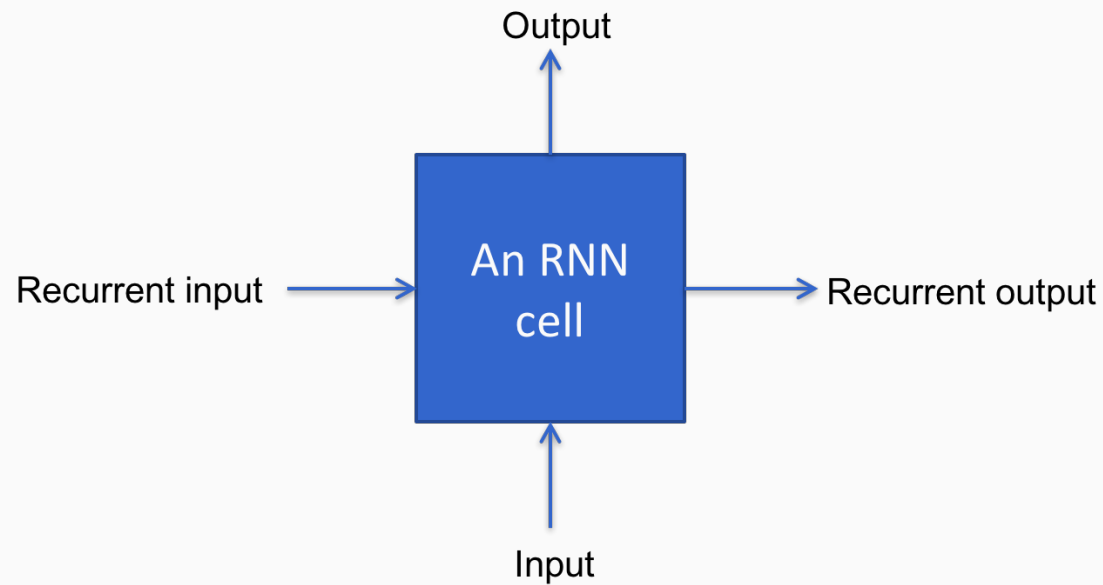
1. How to generate the current state using the previous state and the current input?

Next state $\mathbf{s}_t = g(\mathbf{s}_{t-1}\mathbf{W}_S + \mathbf{x}_t\mathbf{W}_I + \mathbf{b})$

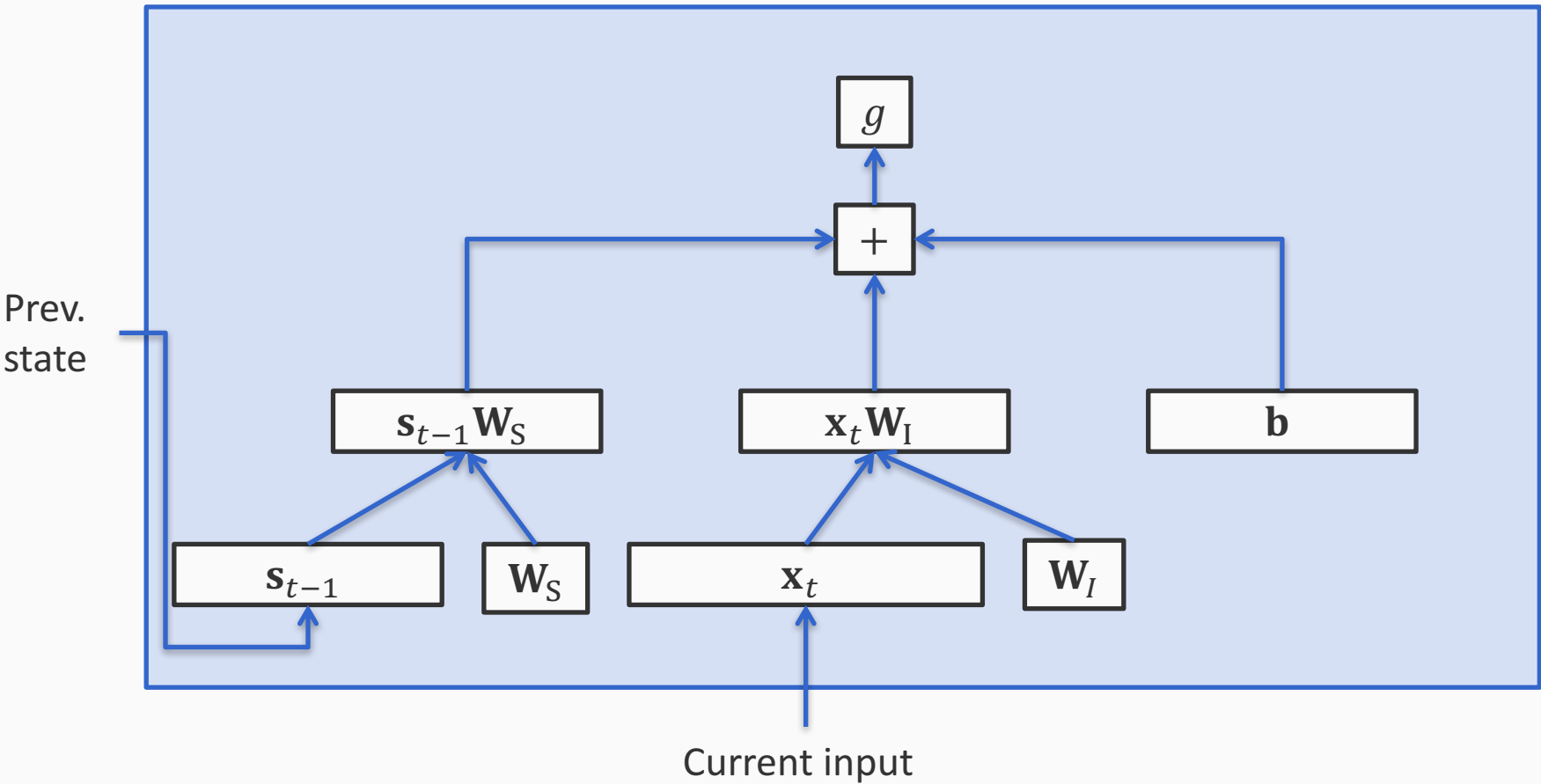
2. How to generate the current output using the current state?

The output is the state. That is, $\mathbf{y}_t = \mathbf{s}_t$

The Elman RNN



The Elman RNN



The Elman RNN

