Recurrent Neural Networks (part 1)

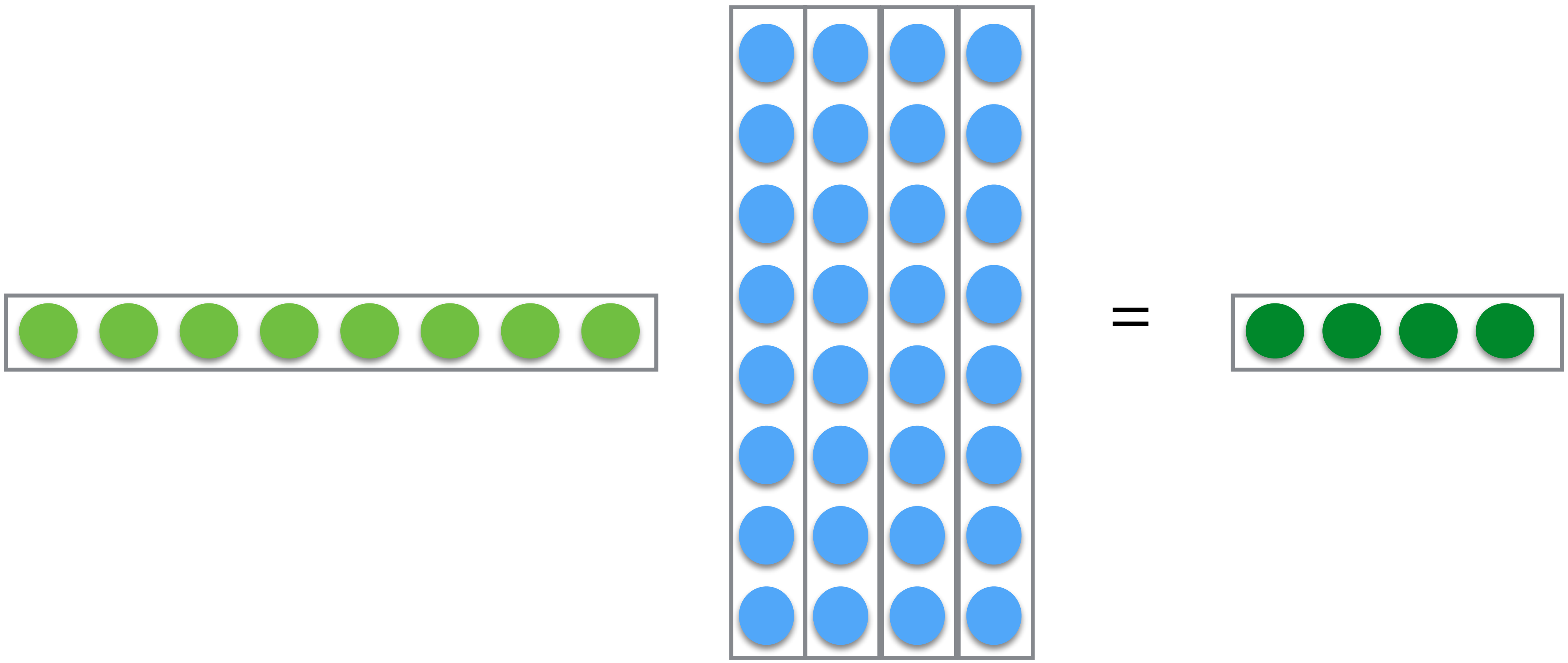
Yoav Goldberg

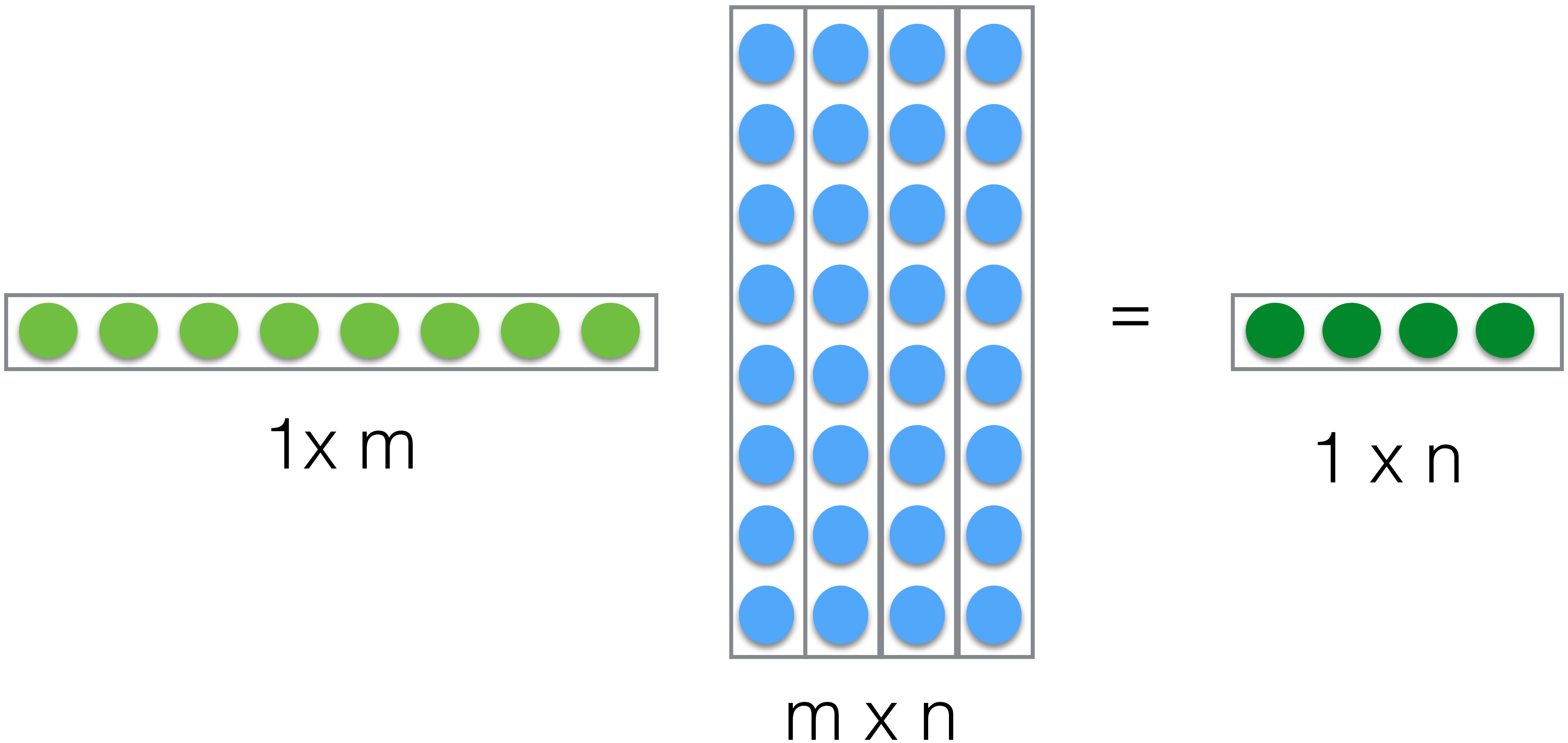
Previously:

- Feed forward networks (multi-layer perceptrons).
- Word/feature embeddings.
- Convolution Networks (n-gram extractors)
- The **computation graph**. Software toolkits.

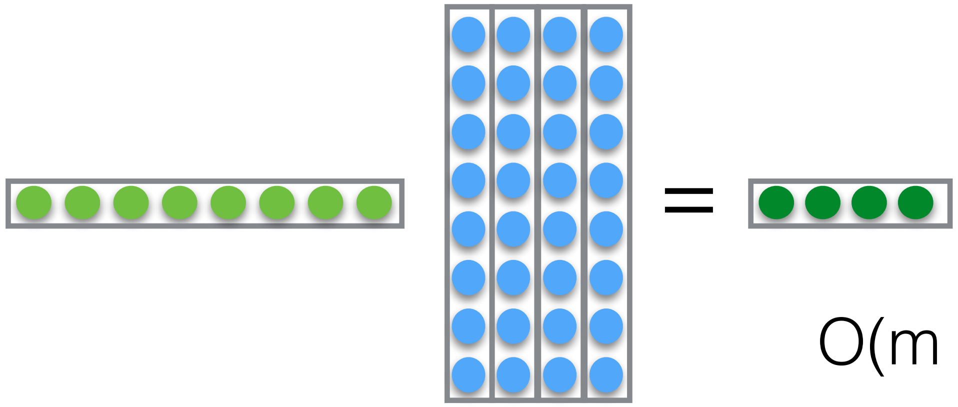
(not about RNNs)

batching for efficiency



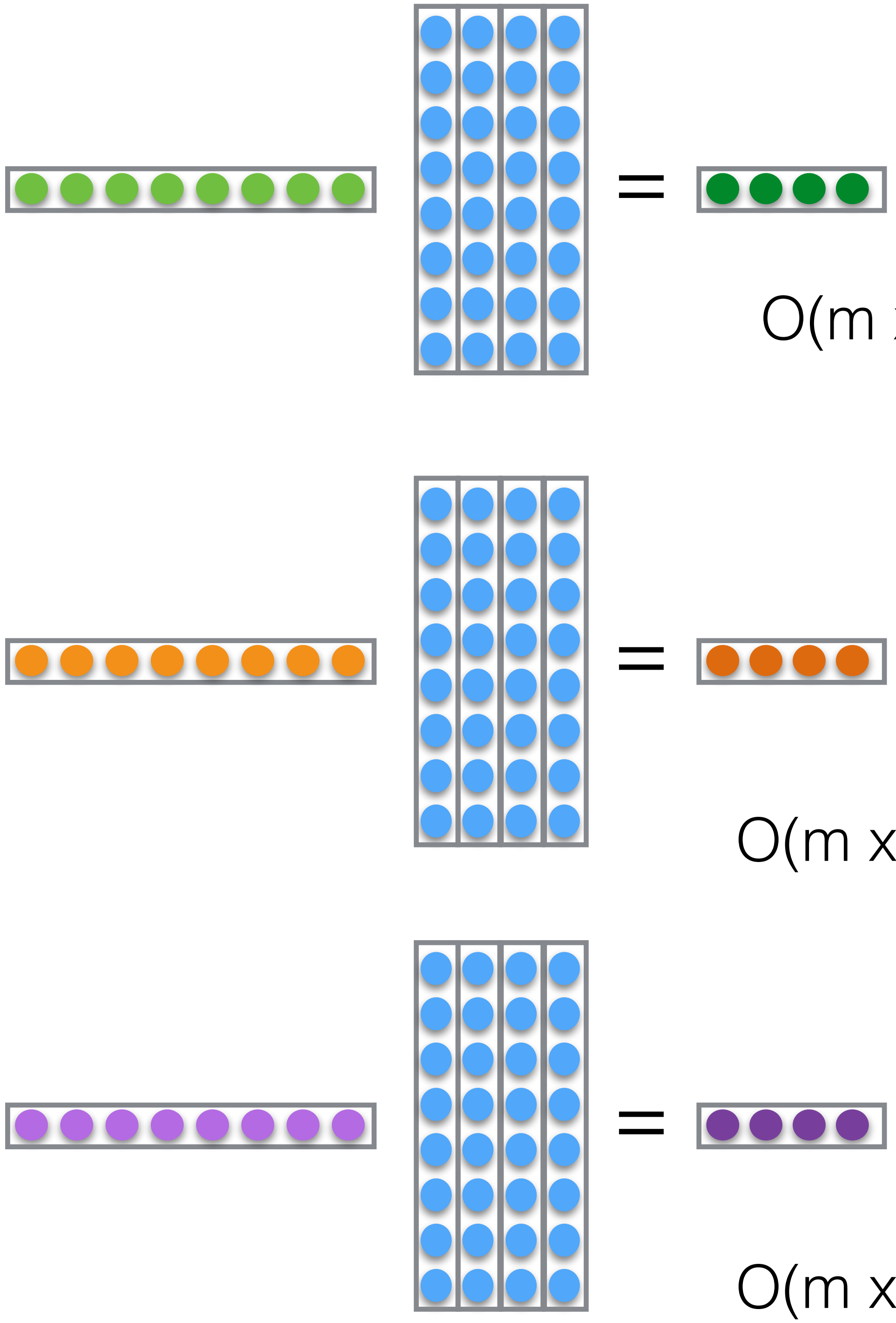


O(m x n)



O(m x n)

k vector-matrix multiplications



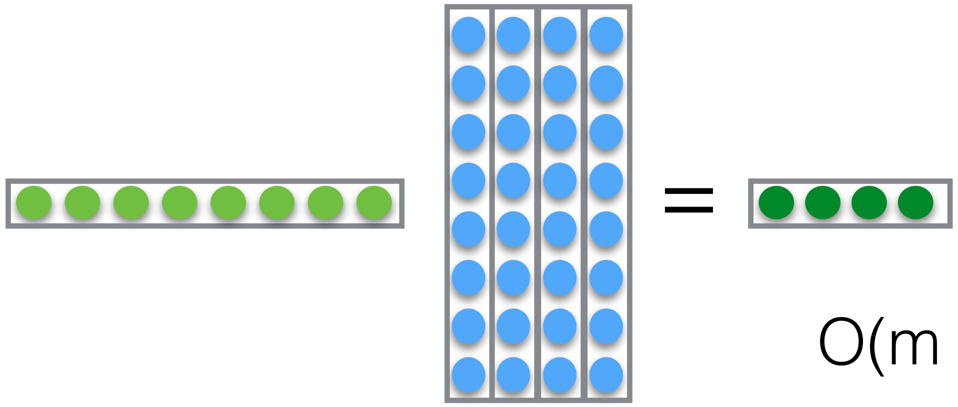
O(m x n)

O(m x n)

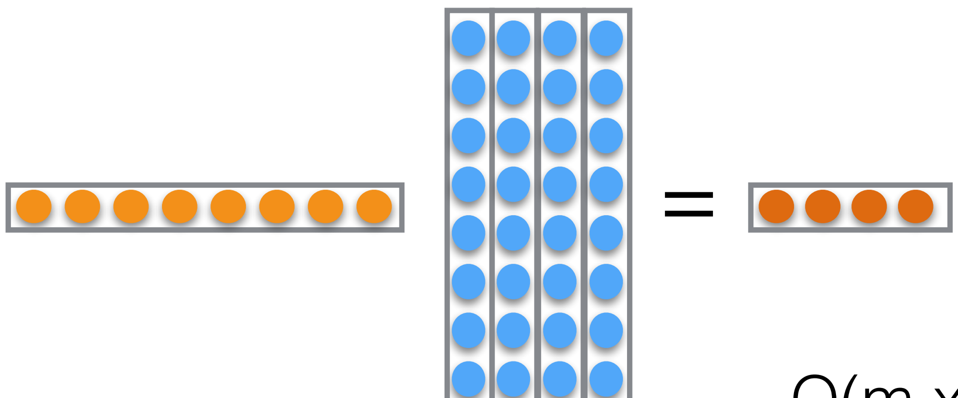
O(m x n)

O(k x m x n)

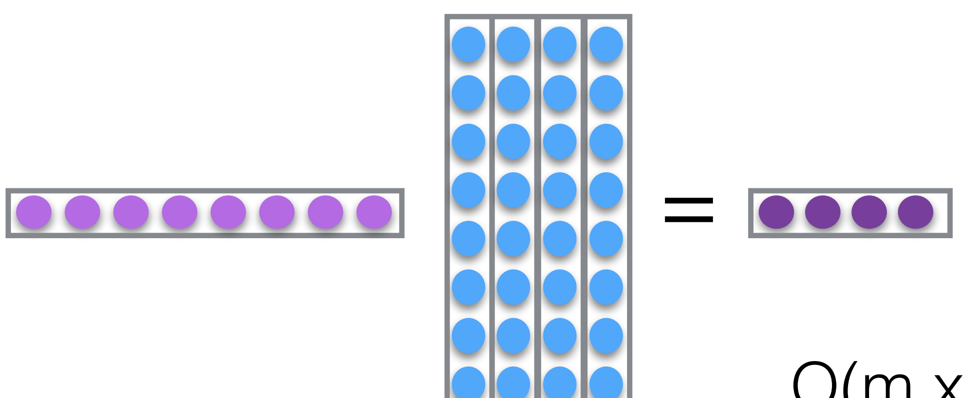
k vector-matrix multiplications



O(m x n)



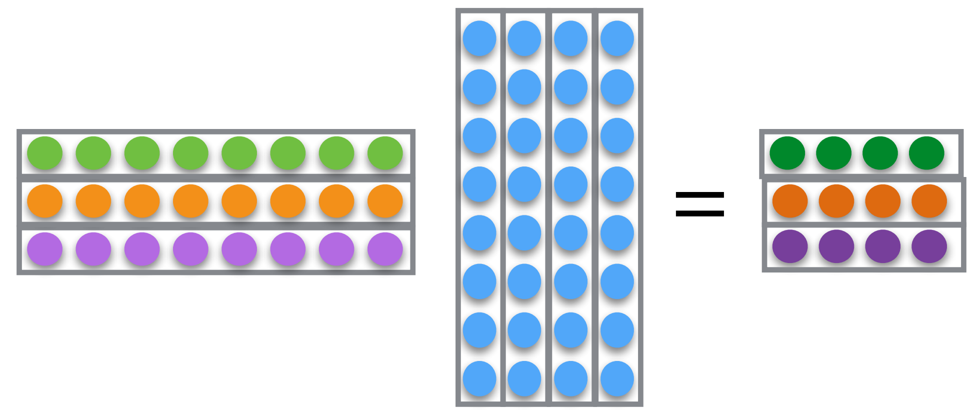
O(m x n)



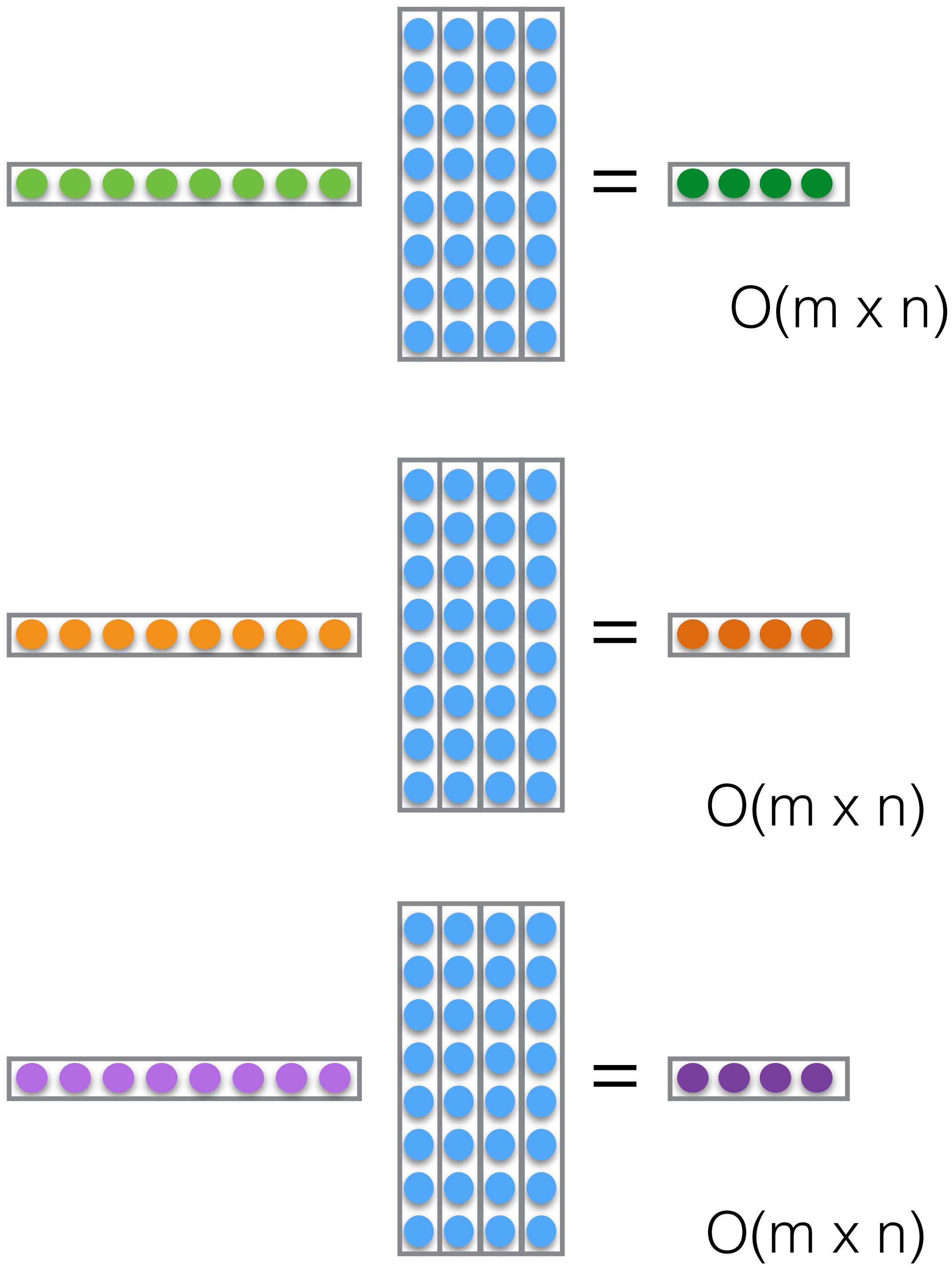
O(m x n)

O(k x m x n)

single matrix-matrix mult

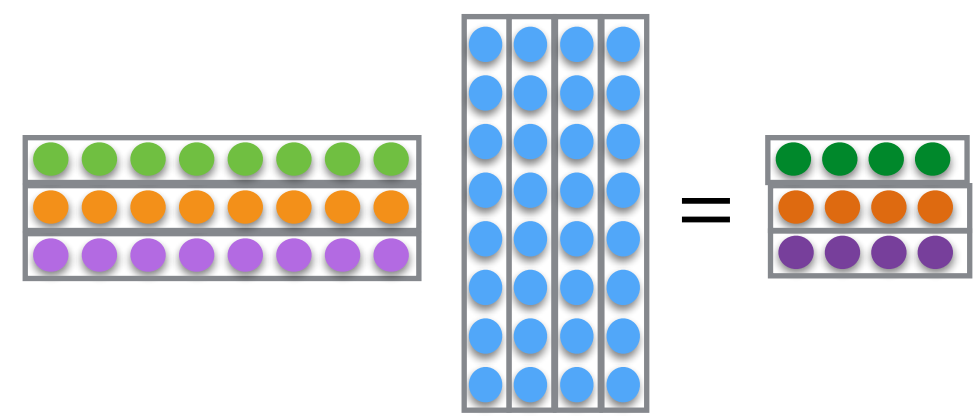


k vector-matrix multiplications



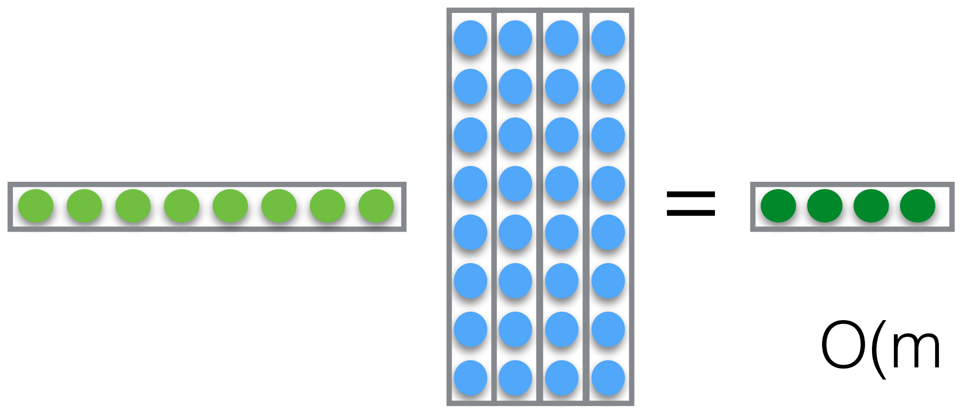
O(k x m x n)

single matrix-matrix mult

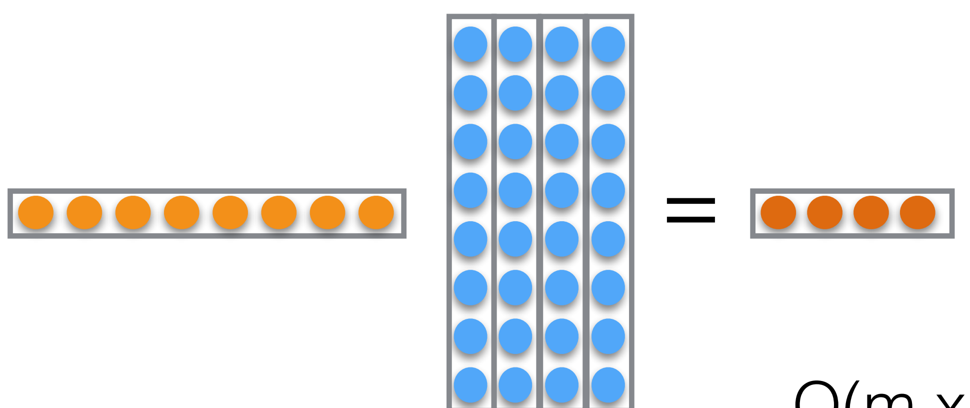


O(k x m x n)

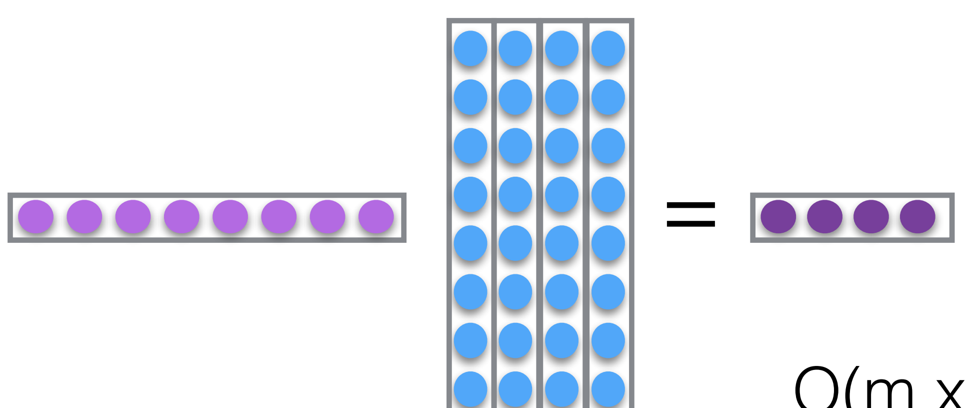
k vector-matrix multiplications



O(m x n)



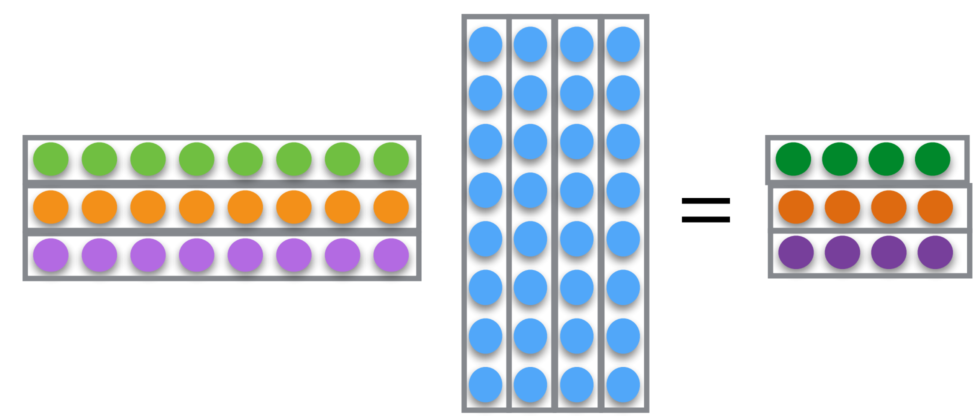
O(m x n)



O(m x n)

O(k x m x n)

single matrix-matrix mult



this is much faster in practice

esp on GPU

O(k x m x n)

batching

- Instead of a k vector-matrix operations,
  call a single matrix-matrix operation.
- You need to order your data it to be efficient.
  - Note: memory copies also cost some.
- Batching can be very effective, need to be controlled manually.

RNNS

Dealing with Sequences

- For an input sequence **x1**,...,**xn**, we can:
  - If *n* is **fixed**: *concatenate* and feed into an MLP.
  - *sum* the vectors (*CBOW*) and feed into an MLP.
  - Break the sequence into *windows* (i.e., for tagging). Each window is fixed size, concatenate into an MLP.
  - Find good ngrams using ConvNet, using *pooling* (either sum/avg or max) to combine to a single vector.

Dealing with Sequences

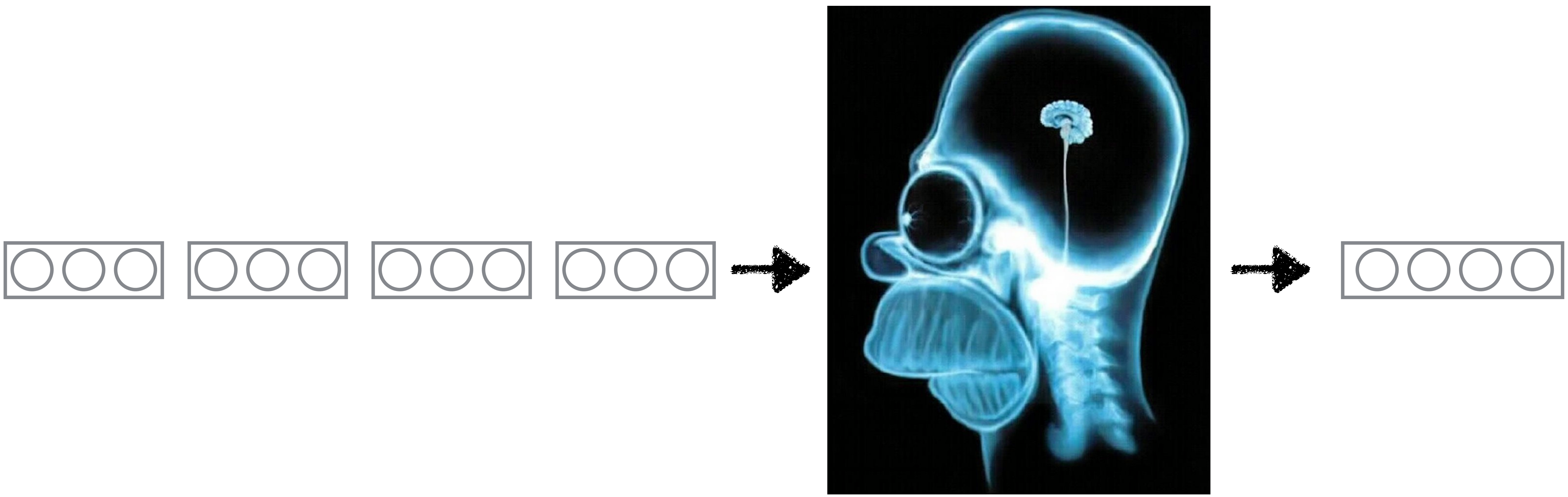
- For an input sequence **x1**,...,**xn**, we can:

Some of these approaches
consider **local** word order (which ones?).

How can we consider **global** word order?

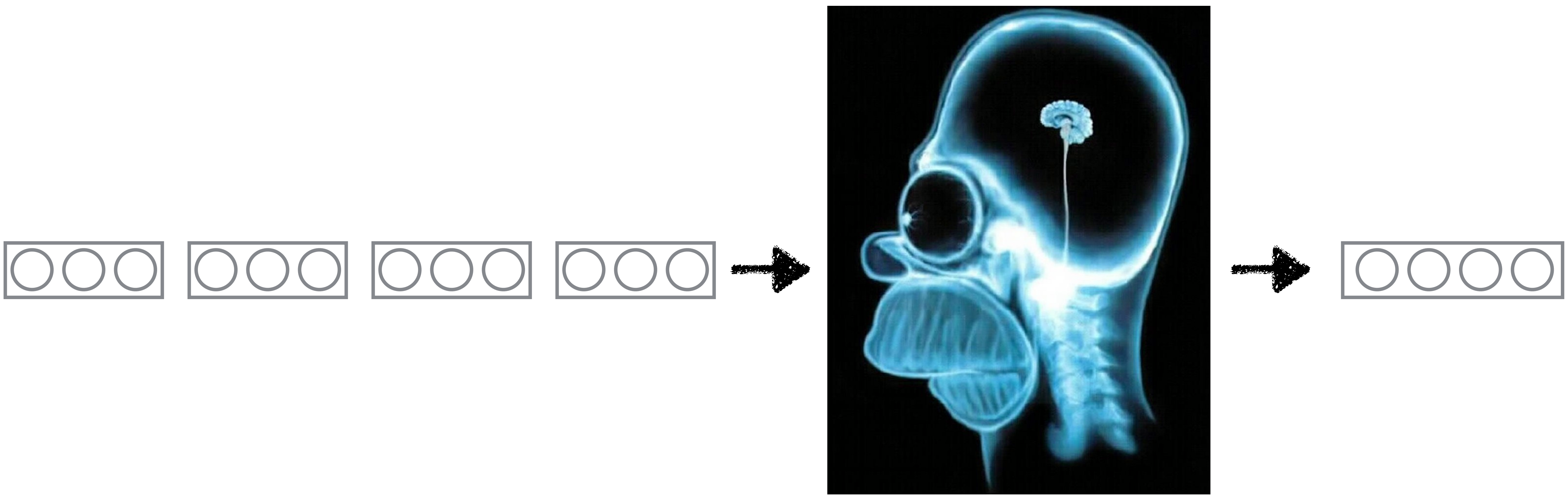
  - Find good ngrams using ConvNet, using *pooling* (either sum/avg or max) to combine to a single vector.

Recurrent Neural Networks



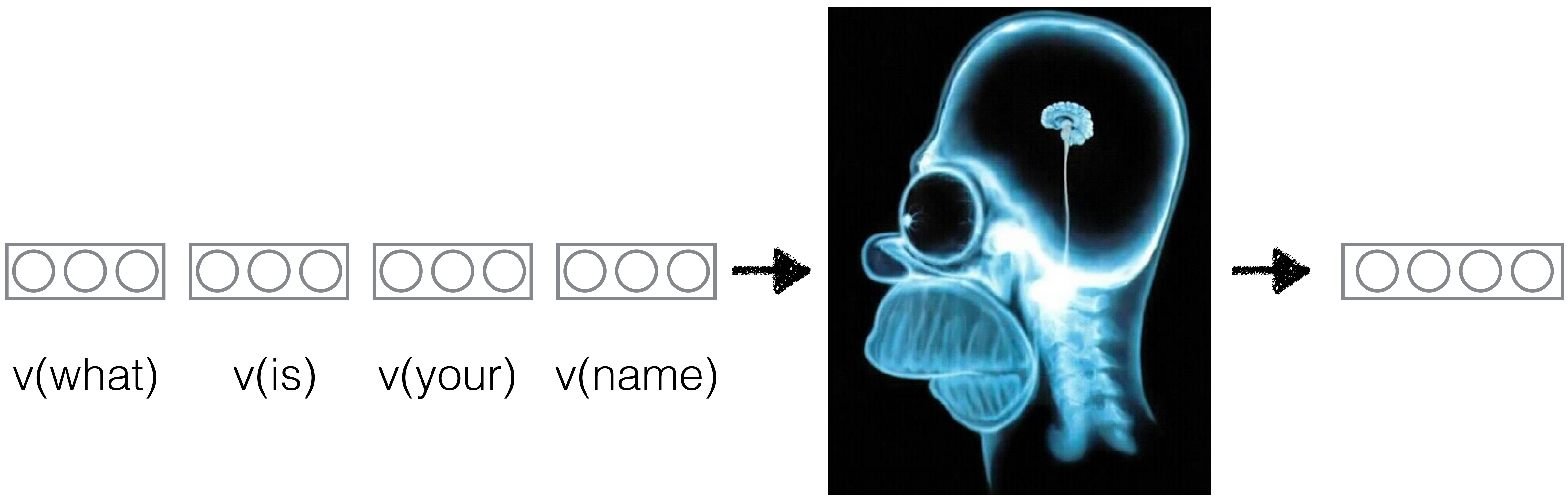
- **Very strong models of sequential data.**
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



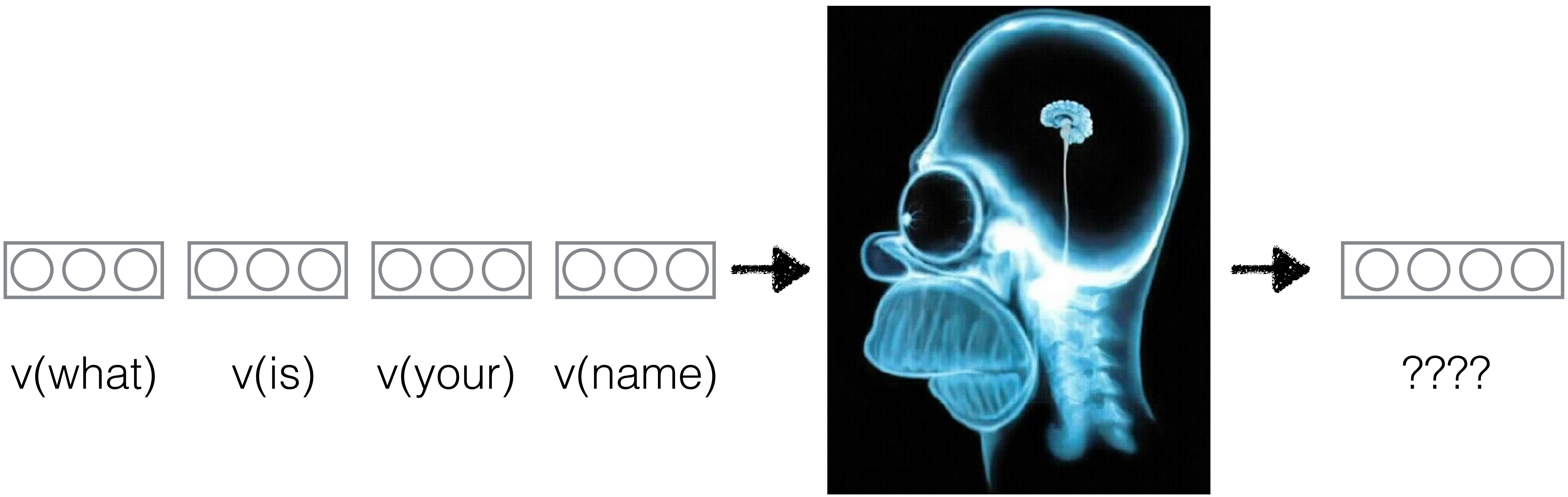
- Very strong models of sequential data.
- **Function from *n* vectors to a single vector.**

Recurrent Neural Networks



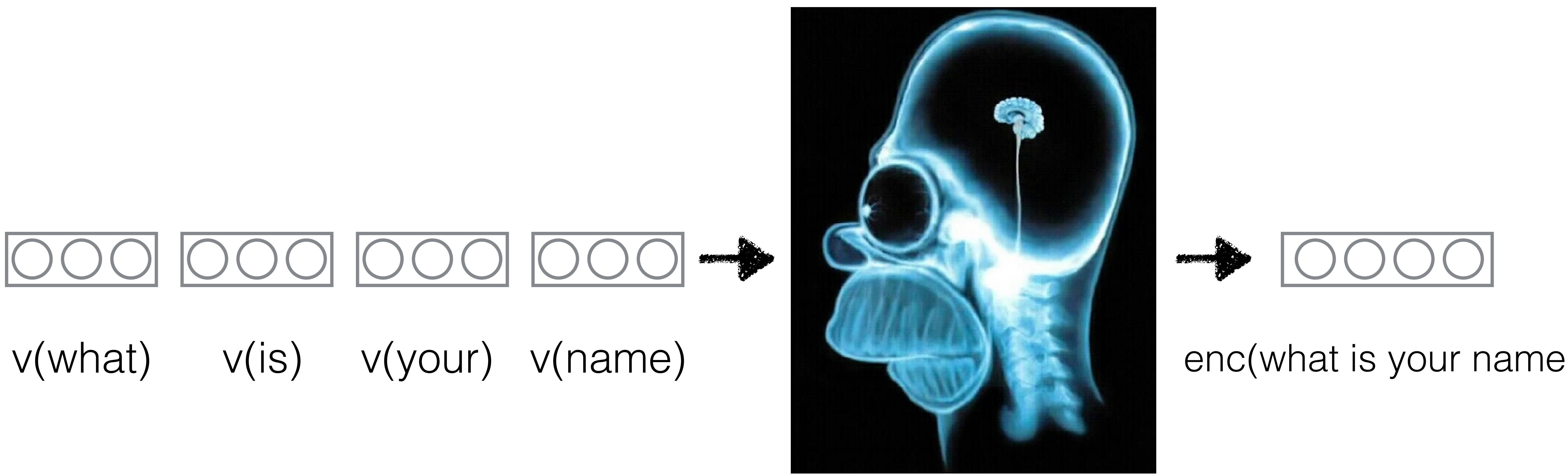
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



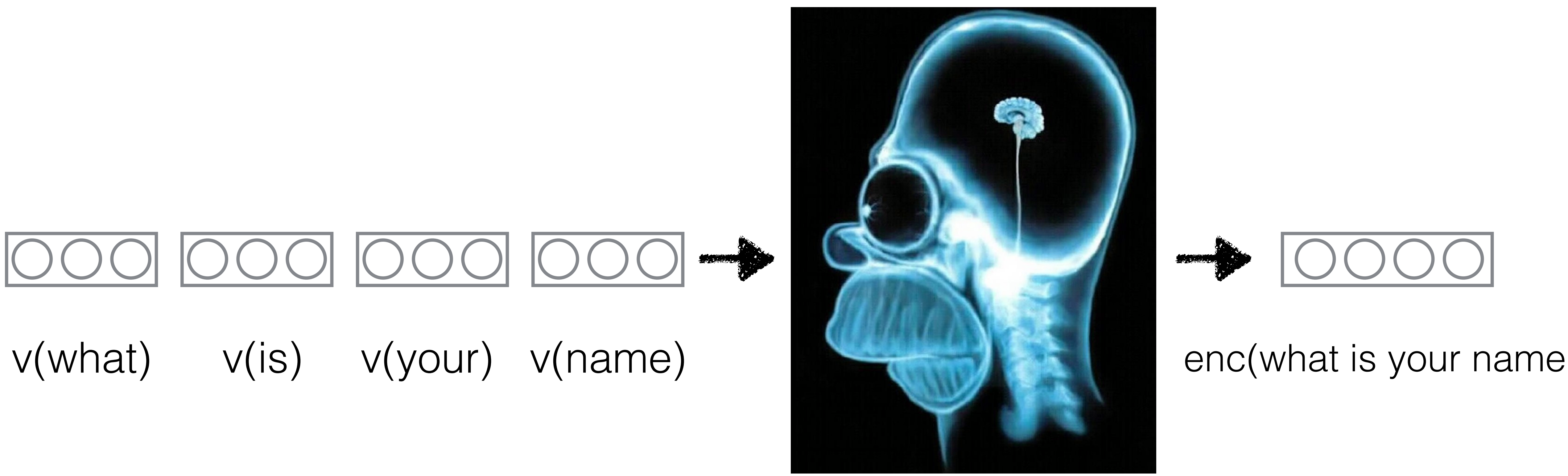
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



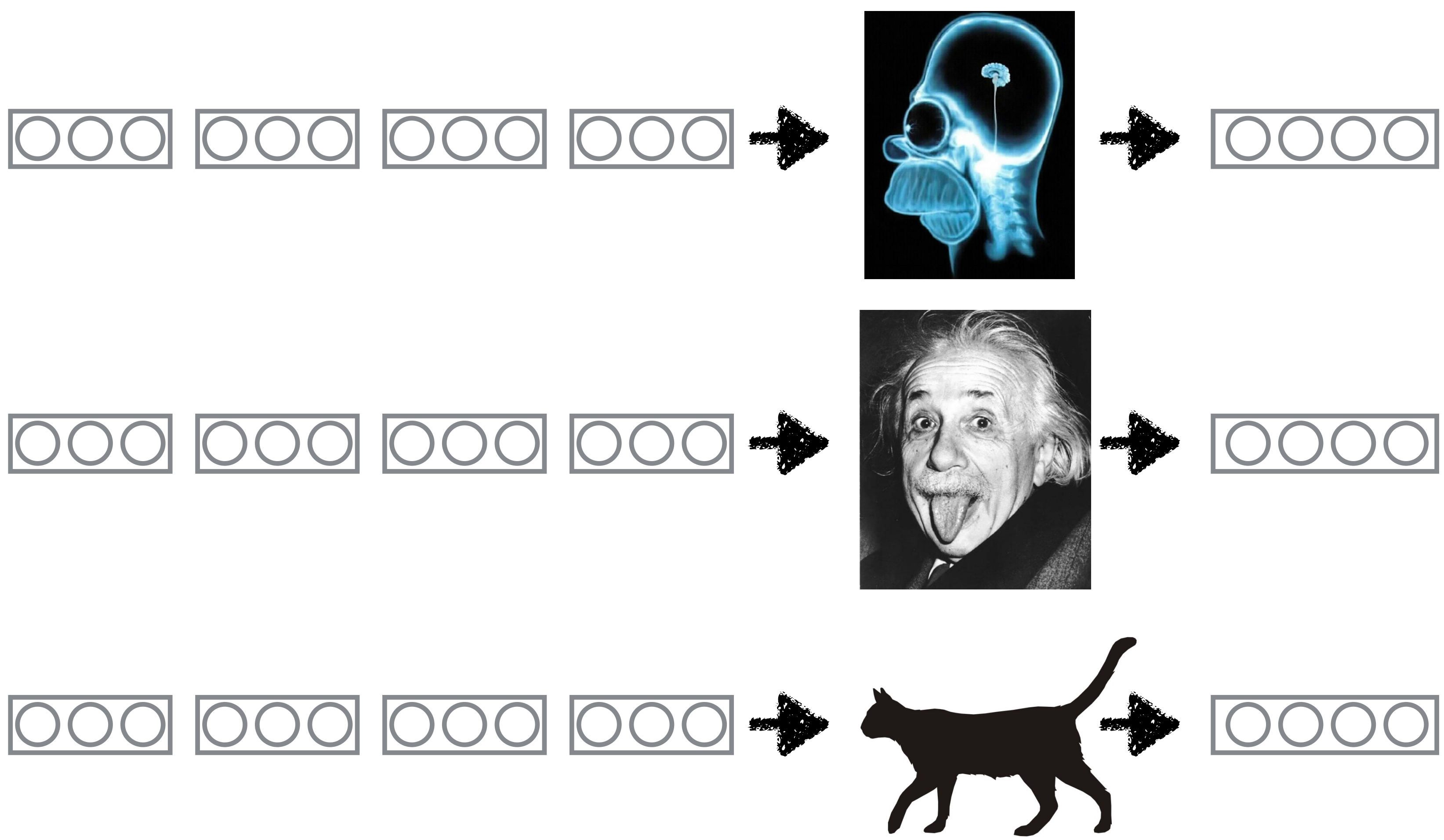
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single vector.

Recurrent Neural Networks



- There are different variants (implementations).
- We'll start by focusing (mostly) on the interface level.

Recurrent Neural Networks

$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single* vector.

Recurrent Neural Networks

$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

*this one is internal. we only care about the **y**

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}}, \ \mathbf{y_i} \in \mathbb{R}^{d_{out}}, \ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single* vector.

Recurrent Neural Networks

$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$
$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$
$$\mathbf{y_i} = O(\mathbf{s_i})$$

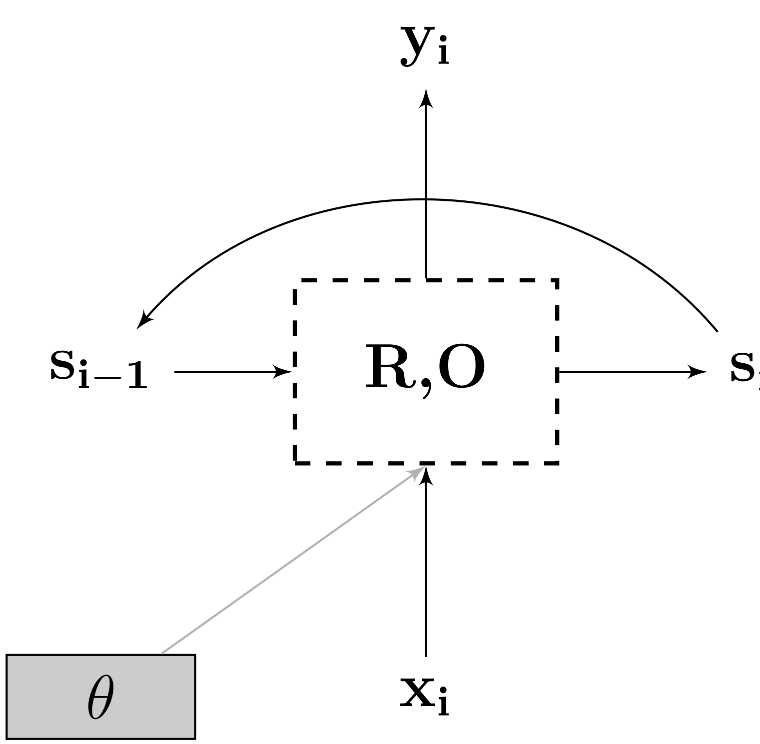
$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- **Recursively defined.**
- There's a vector $\mathbf{y_i}$ for every prefix $\mathbf{x_{1:i}}$

Recurrent Neural Networks

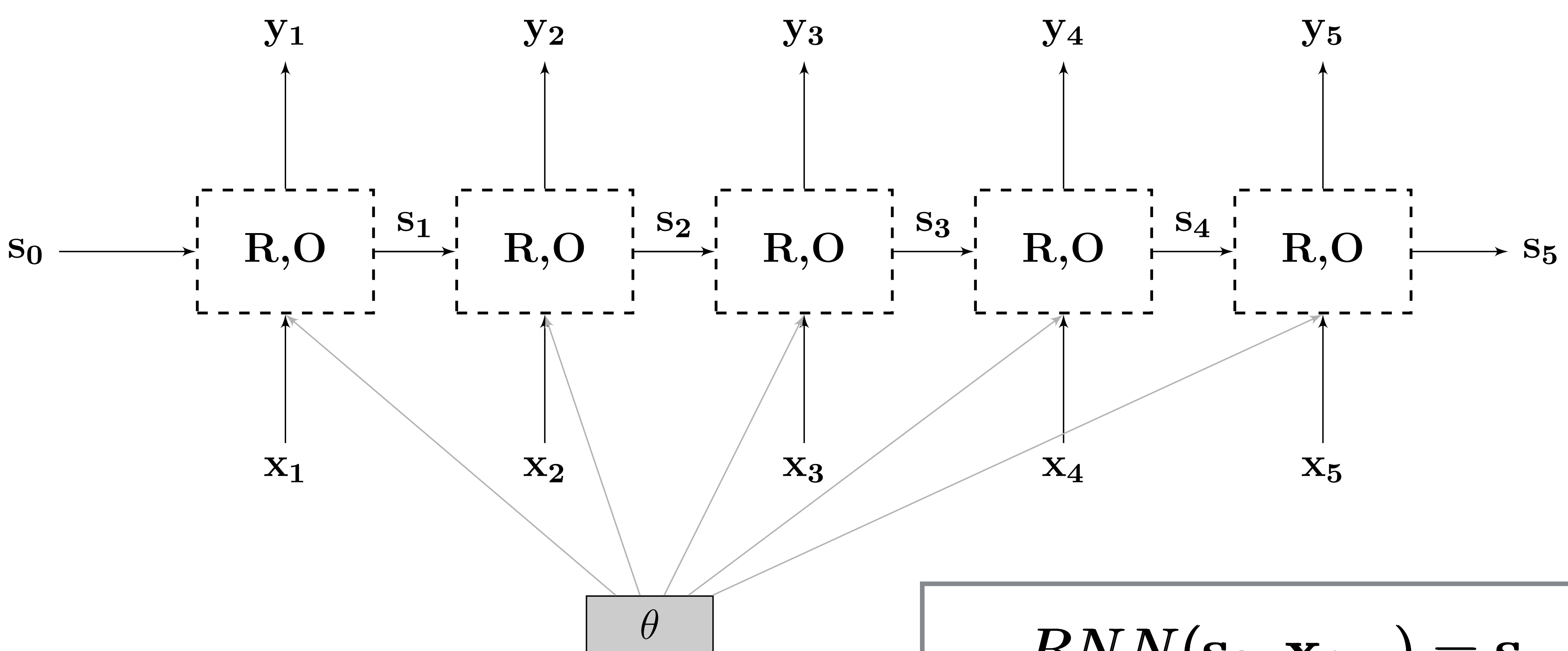
$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$
$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$
$$\mathbf{y_i} = O(\mathbf{s_i})$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$



- **Recursively defined**.
- There's a vector $\mathbf{y_i}$ for every prefix $\mathbf{x_{1:i}}$

Recurrent Neural Networks



$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$

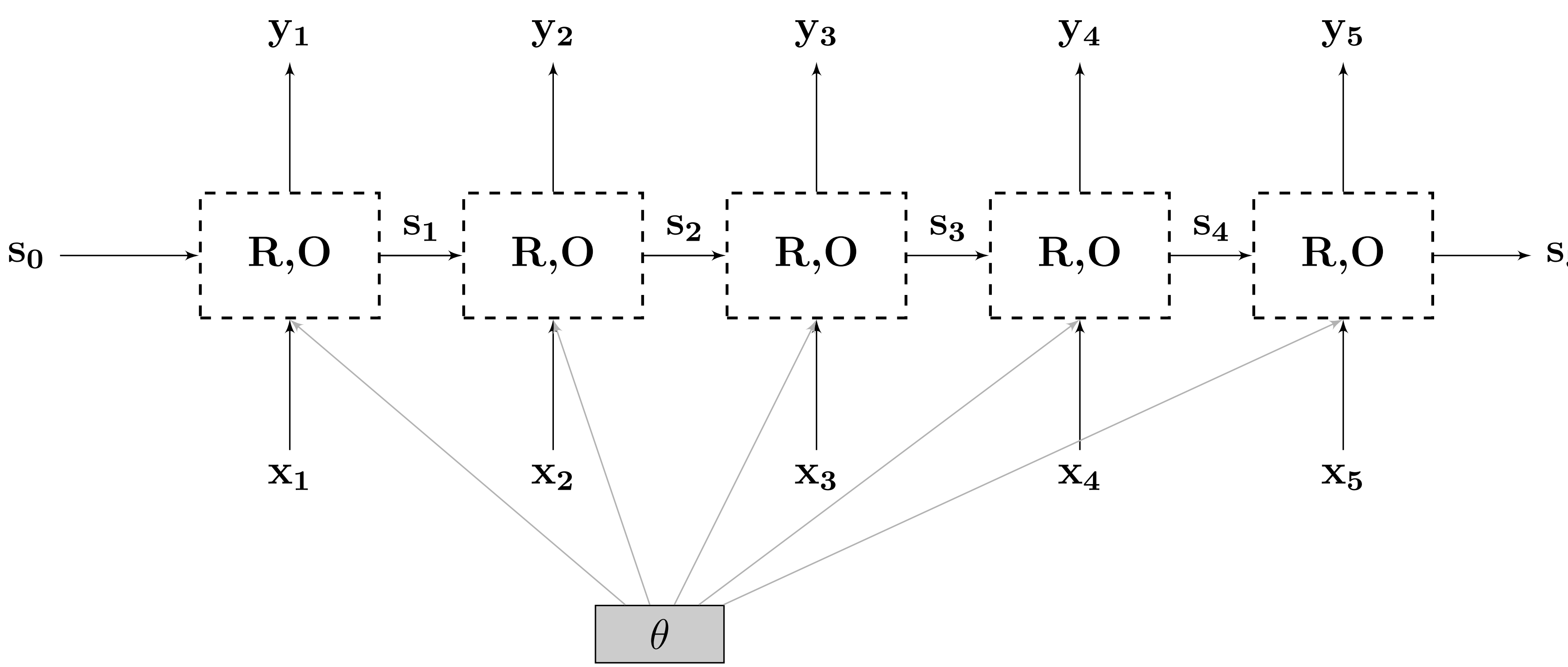
$$\mathbf{y_i} = O(\mathbf{s_i})$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}}, \ \mathbf{y_i} \in \mathbb{R}^{d_{out}}, \ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

for every finite input sequence,
can unroll the recursion.

- Recursively defined.
- There's a vector $\mathbf{y_i}$ for every prefix $\mathbf{x_{1:i}}$

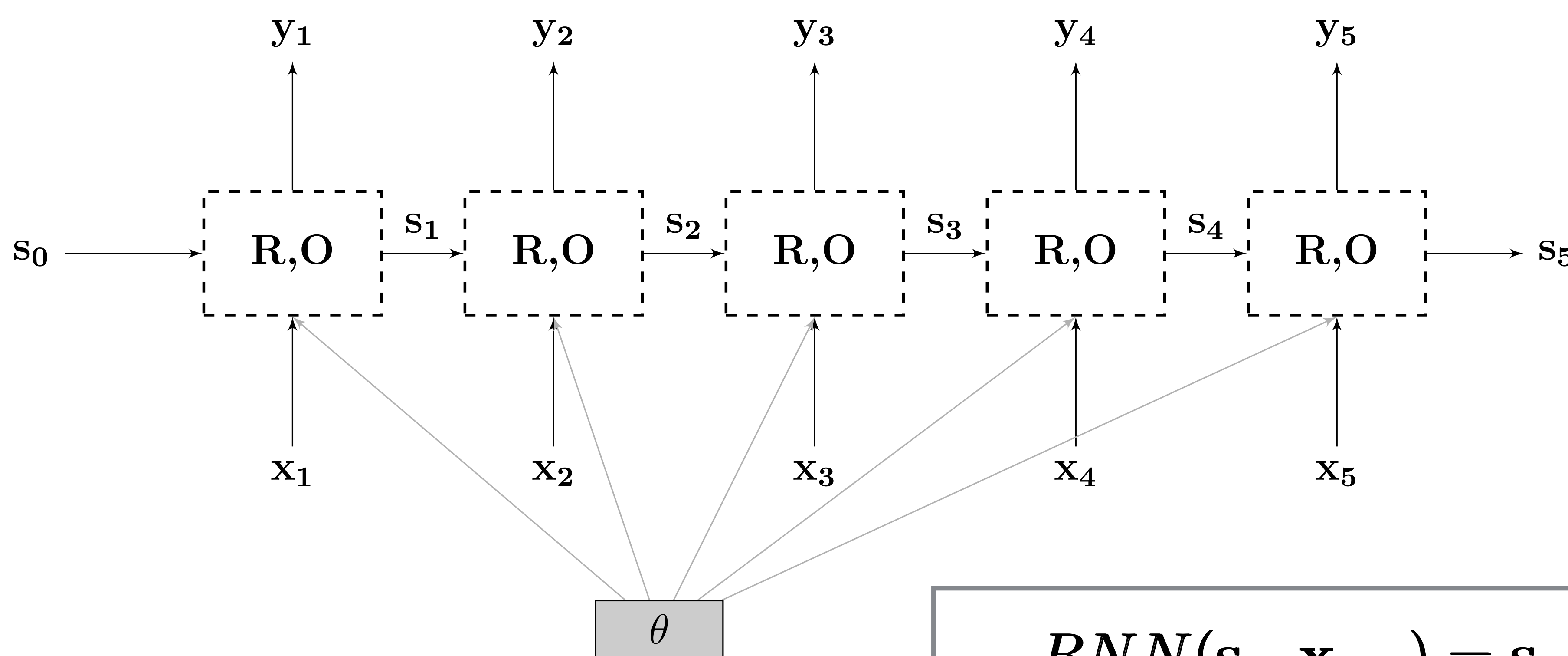
Recurrent Neural Networks



for every finite input sequence,
can unroll the recursion.

An unrolled RNN is just a very deep Feed Forward Network
with shared parameters across the layers,
and a new input at each layer.

Recurrent Neural Networks



$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

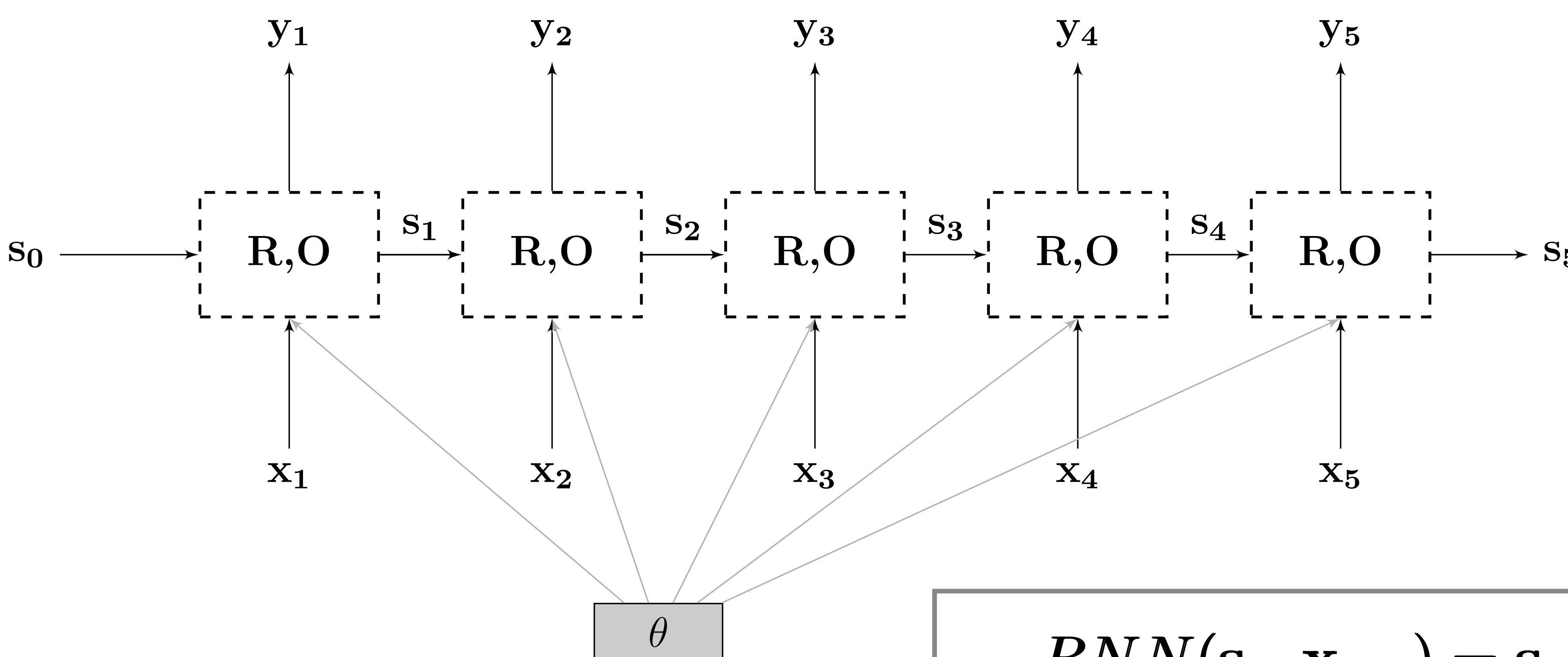
$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$

$$\mathbf{y_i} = O(\mathbf{s_i})$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}}, \ \mathbf{y_i} \in \mathbb{R}^{d_{out}}, \ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- Think of $\mathbf{s_i}$ as "memory".

Recurrent Neural Networks



$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$

$$\mathbf{y_i} = O(\mathbf{s_i})$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- Think of $\mathbf{s_i}$ as "memory".
- The output vector $\mathbf{y_i}$ depends on **all** inputs $\mathbf{x_{1:i}}$

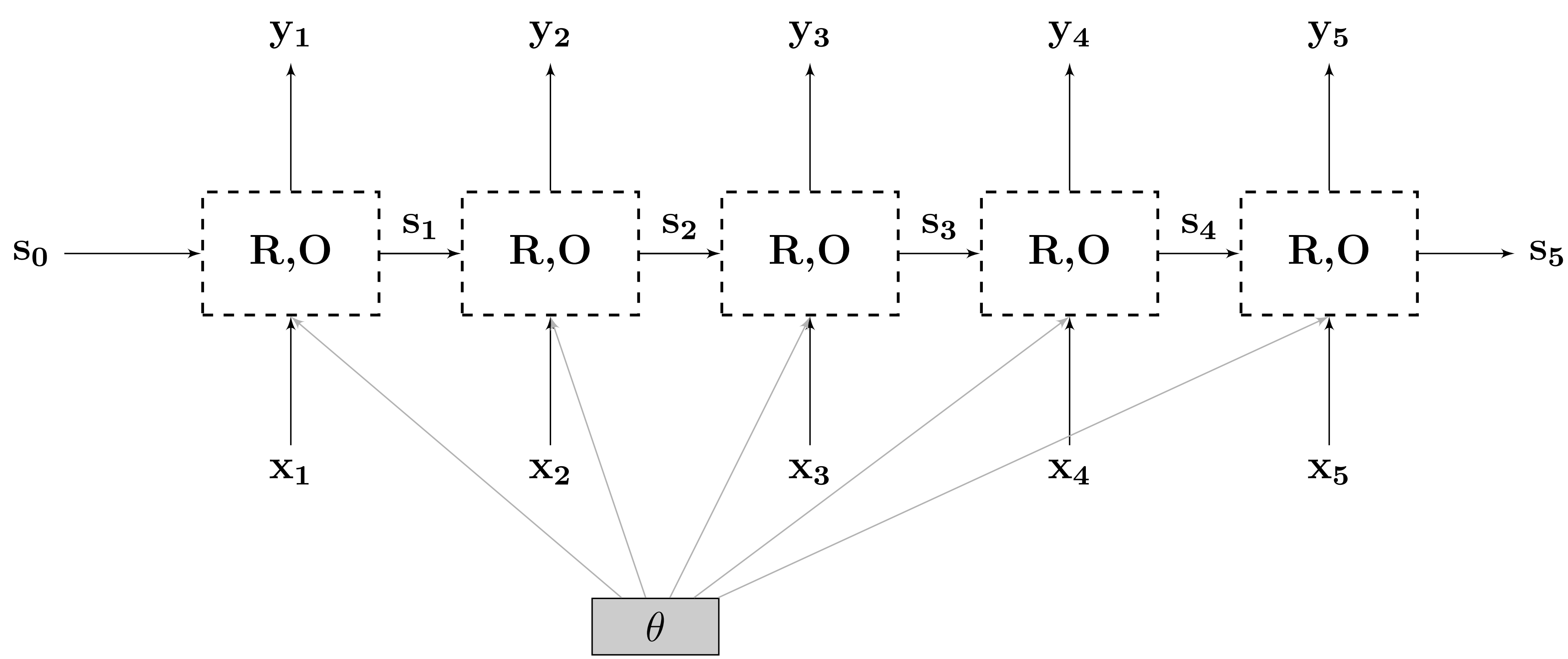
Recurrent Neural Networks

$$
\begin{aligned}
\mathbf{y_4} &= O(\mathbf{s_4}) \\
\mathbf{s_4} &= R(\mathbf{s_3}, \mathbf{x_4}) \\
&= R(\overbrace{R(\mathbf{s_2}, \mathbf{x_3})}^{\mathbf{s_3}}, \mathbf{x_4}) \\
&= R(R(\overbrace{R(\mathbf{s_1}, \mathbf{x_2})}^{\mathbf{s_2}}, \mathbf{x_3}), \mathbf{x_4}) \\
&= R(R(R(\overbrace{R(\mathbf{s_0}, \mathbf{x_1})}^{\mathbf{s_1}}, \mathbf{x_2}), \mathbf{x_3}), \mathbf{x_4})
\end{aligned}
$$

- The output vector $\mathbf{y_i}$ depends on **all** inputs $\mathbf{x_{1:i}}$

Recurrent Neural Networks

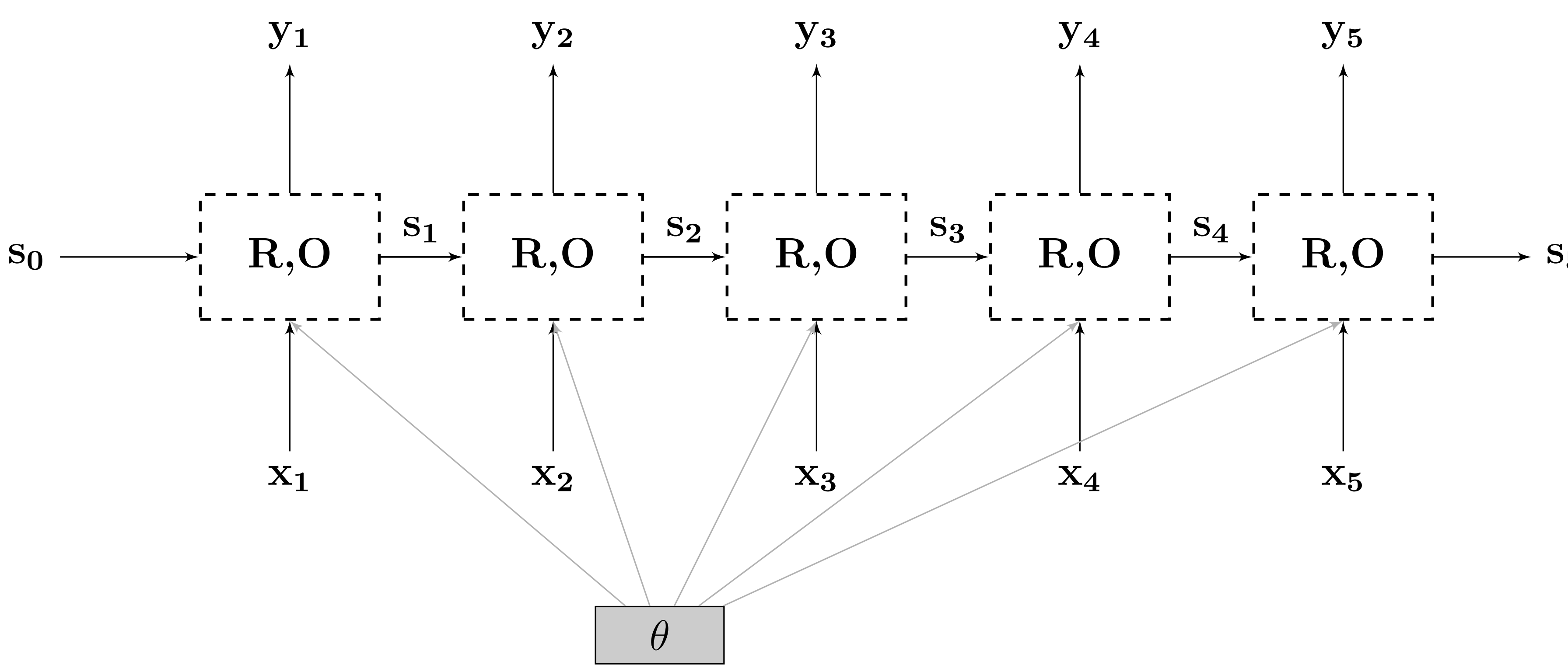
- What are the vectors $\mathbf{y_i}$ good for?



- On their own? **nothing**.

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?

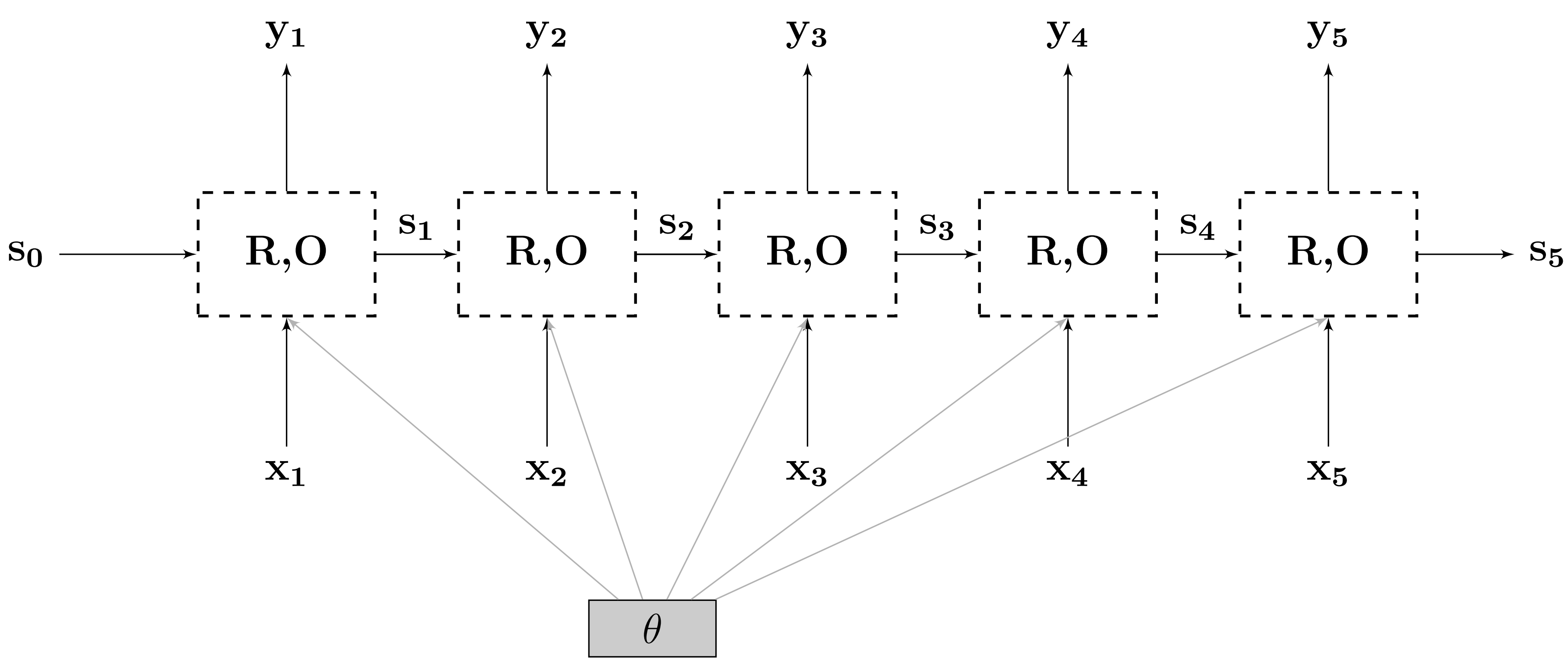


- On their own? **nothing**.

- **But we can train them.**

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



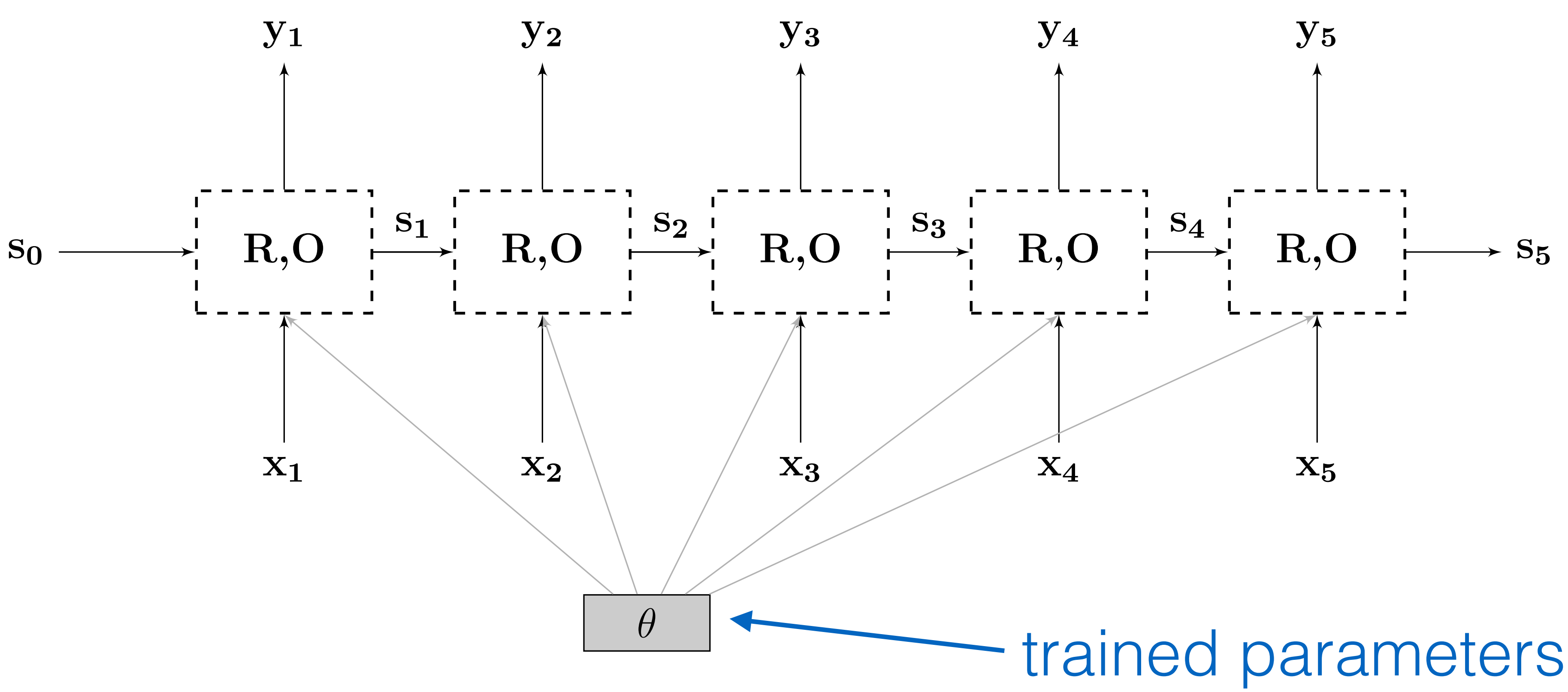
- On their own? **nothing**.
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



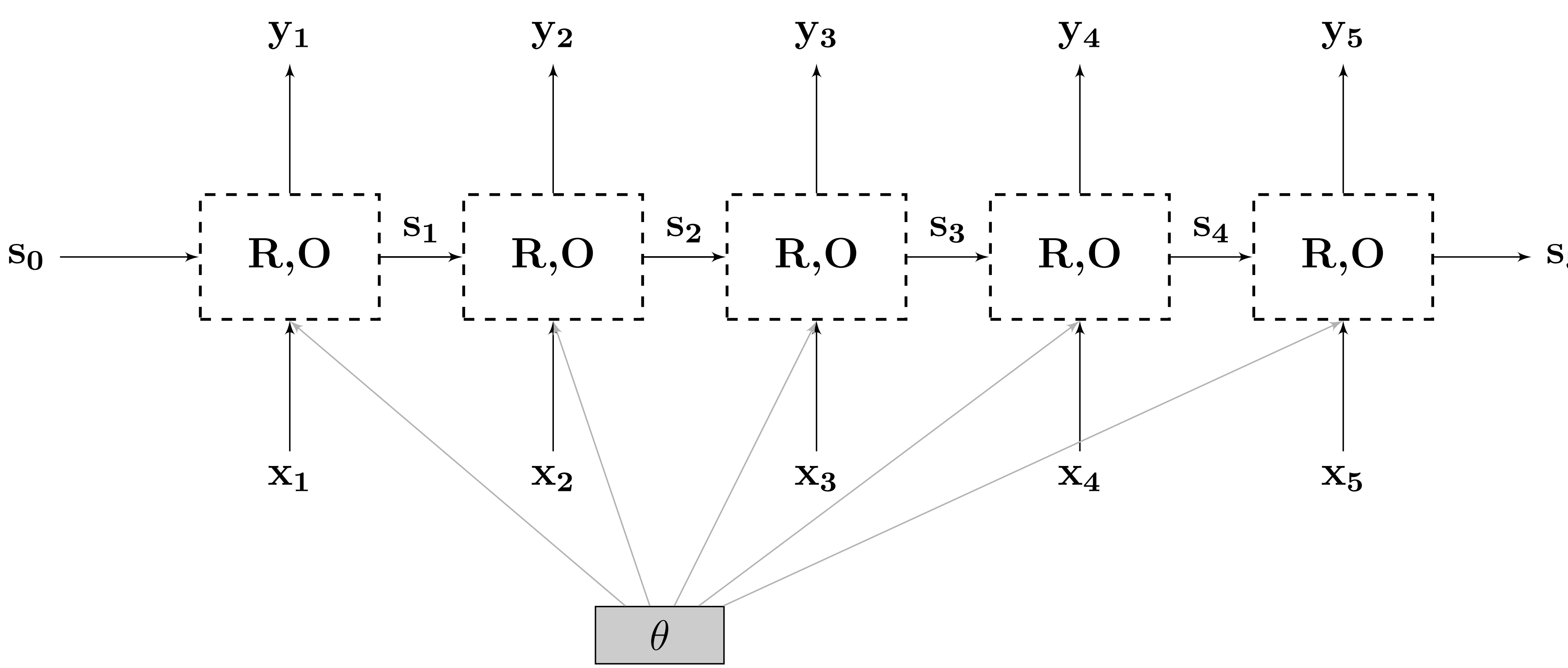
- On their own? **nothing**.
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



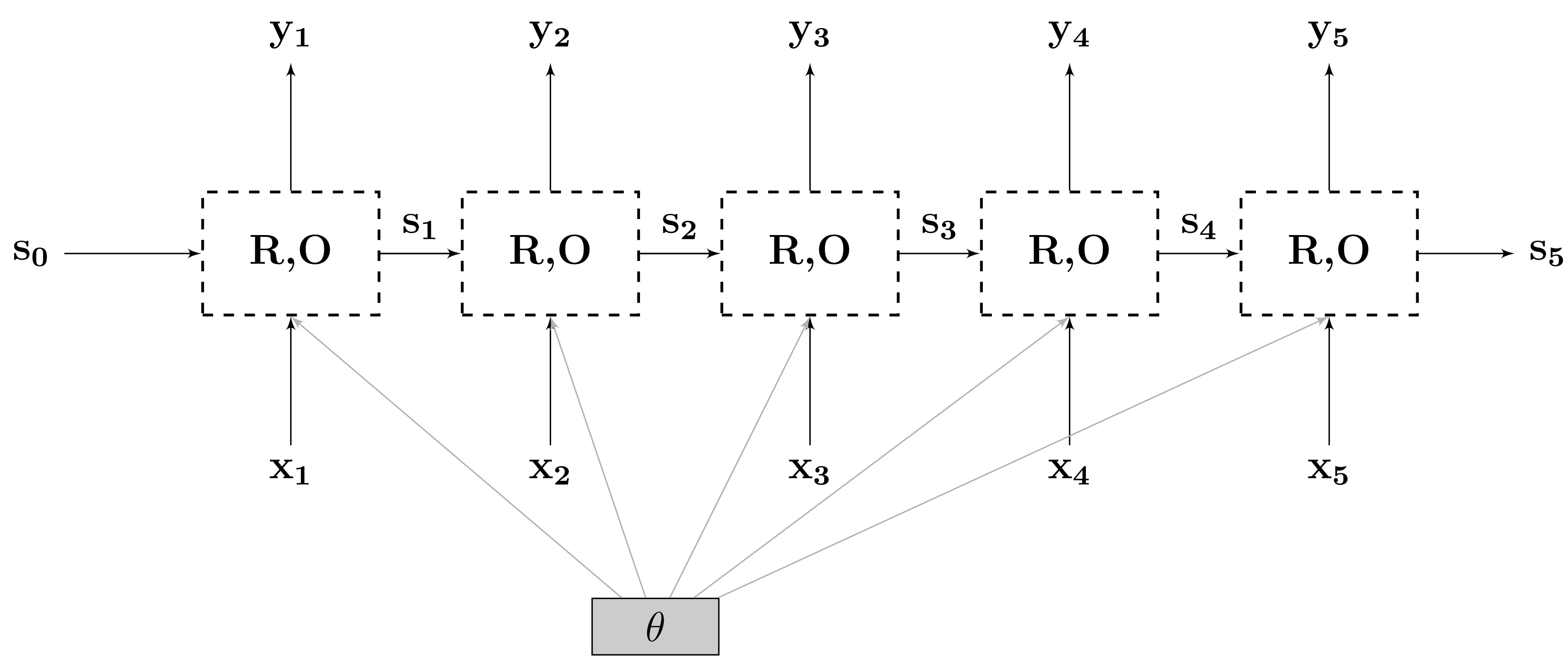
- On their own? **nothing**.
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



- On their own? **nothing**.

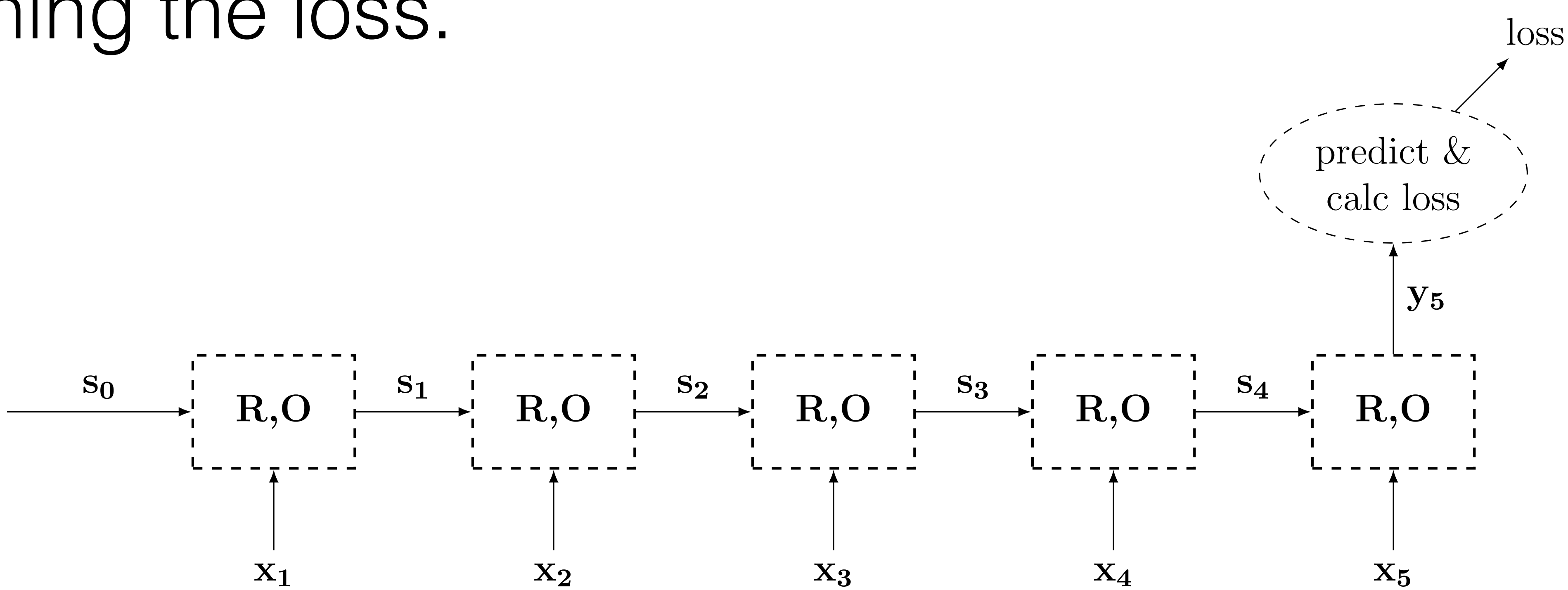
- **But we can train them.**

define function form

define loss

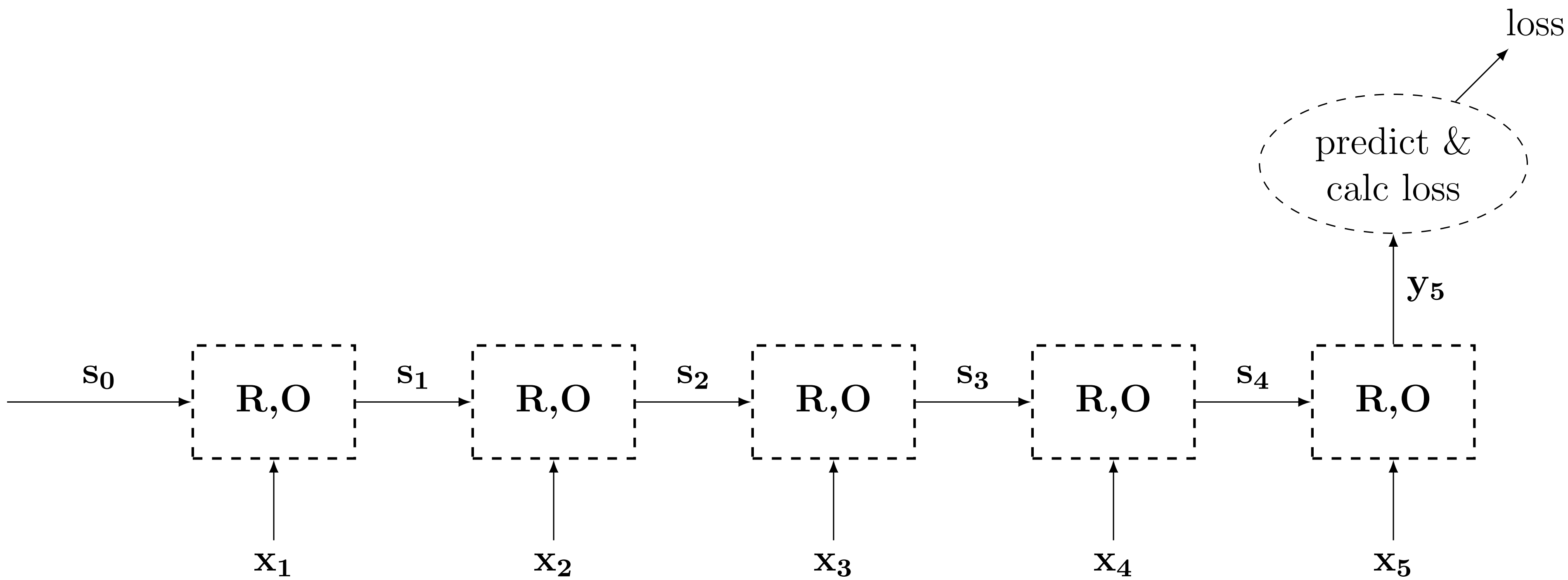
Recurrent Neural Networks

Defining the loss.



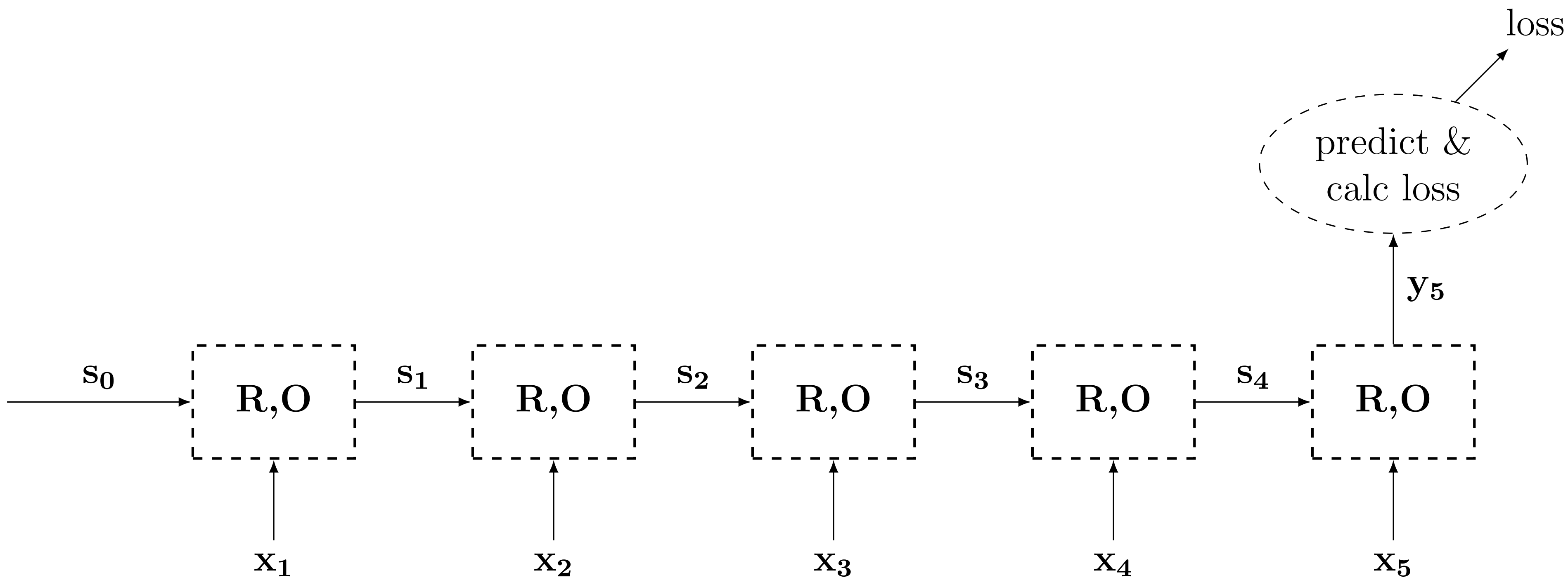
Acceptor: predict something from end state.
Backprop the error all the way back.
Train the network to capture meaningful information

Acceptor Examples



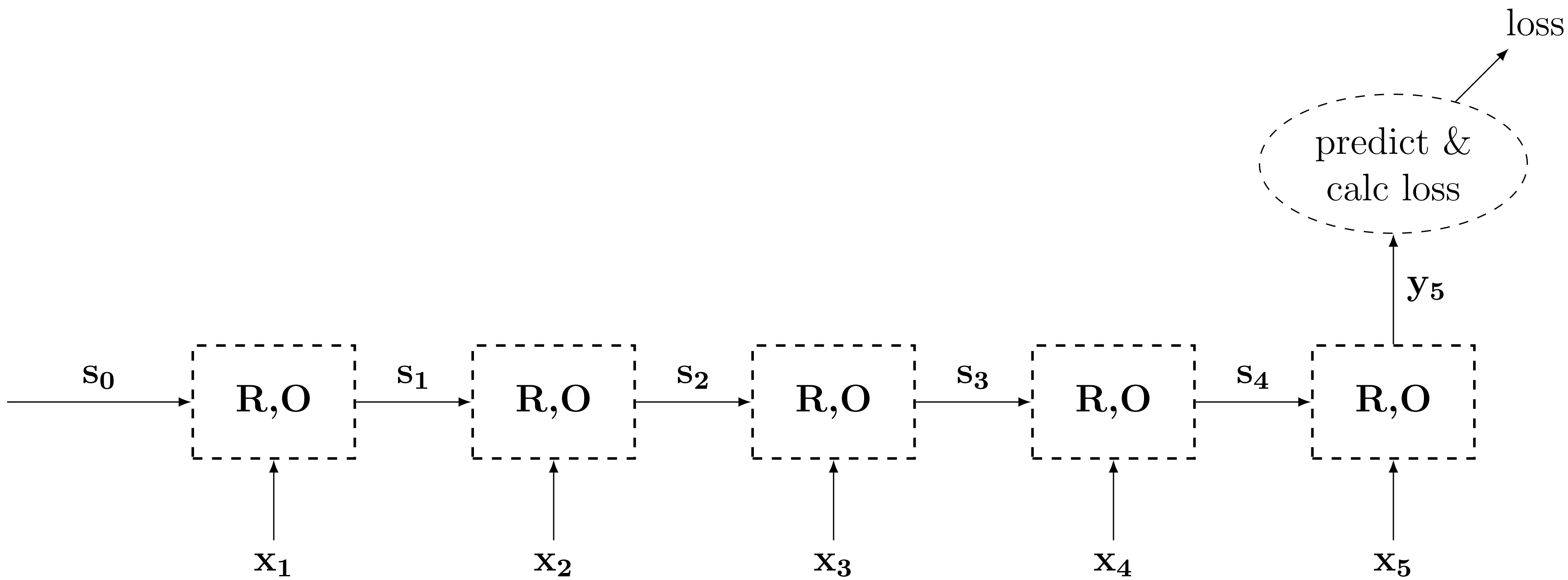
- Predict **sentiment** based on sentence words.

Acceptor Examples



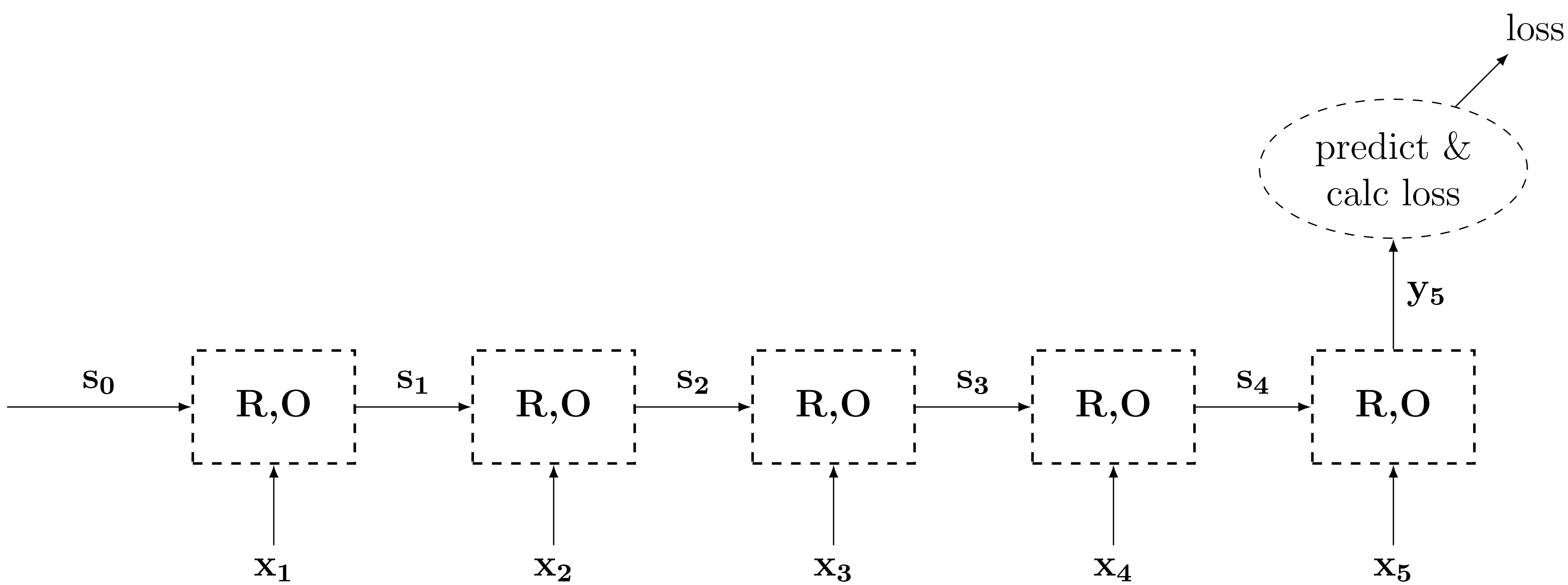
- Predict **POS** based on word's letters sequence.

Acceptor Examples



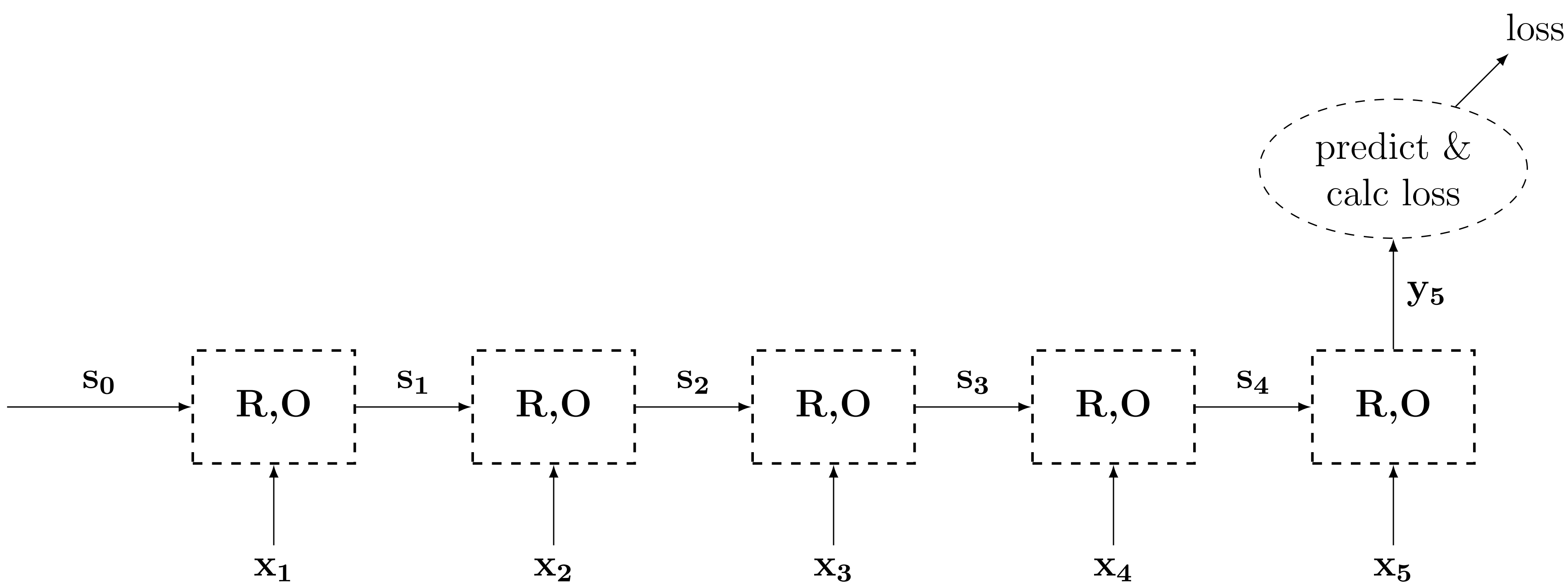
- More examples?

Acceptor Examples



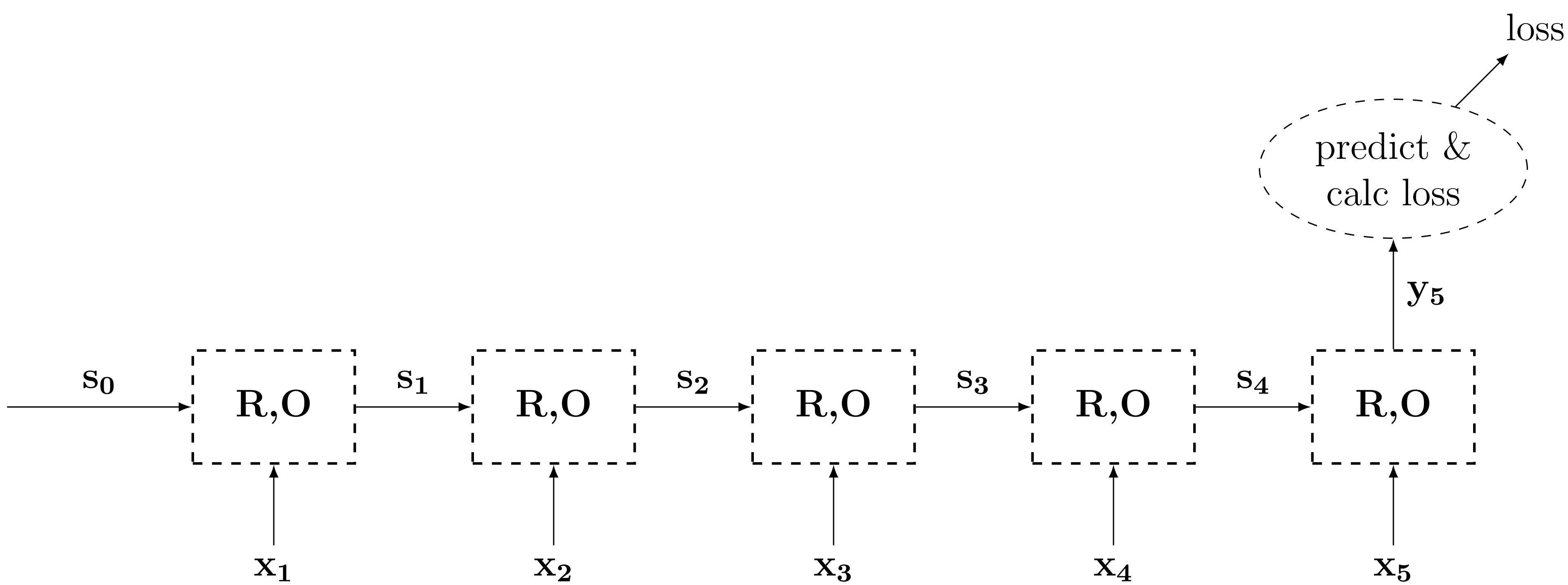
- Will a customer hang up, based on a sequence of call-menu items.

Acceptor Examples



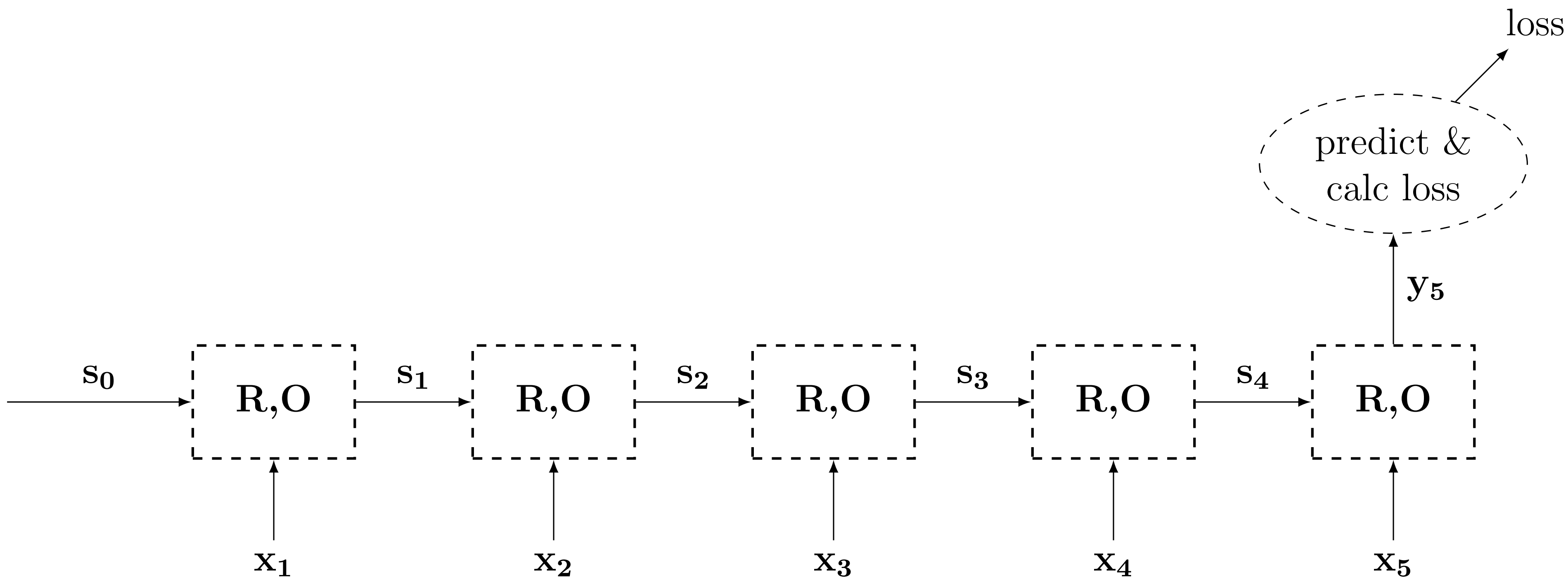
- What next movie am I going to watch, after watching a sequence of previous movies.

Acceptor Examples



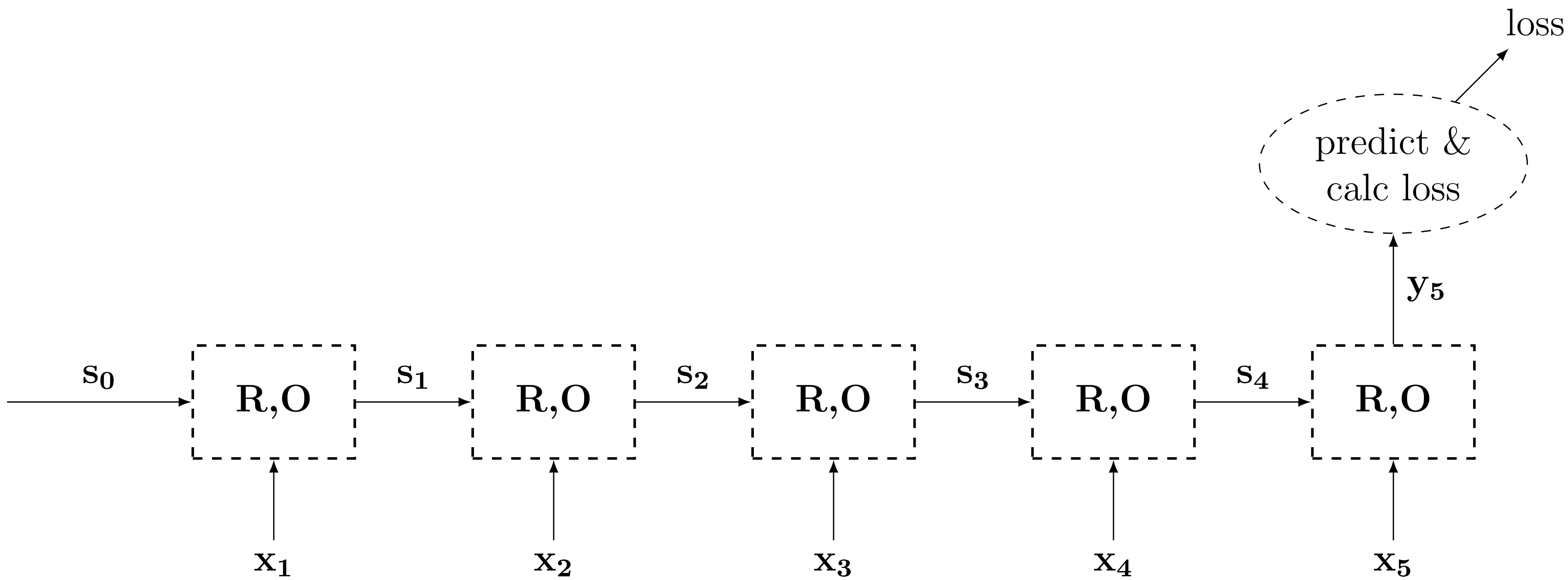
- How about the language identification task from assignment 1? (discuss)

Acceptor Examples



- Predict **word i** based on **words 1,...,i-1**

Acceptor Examples

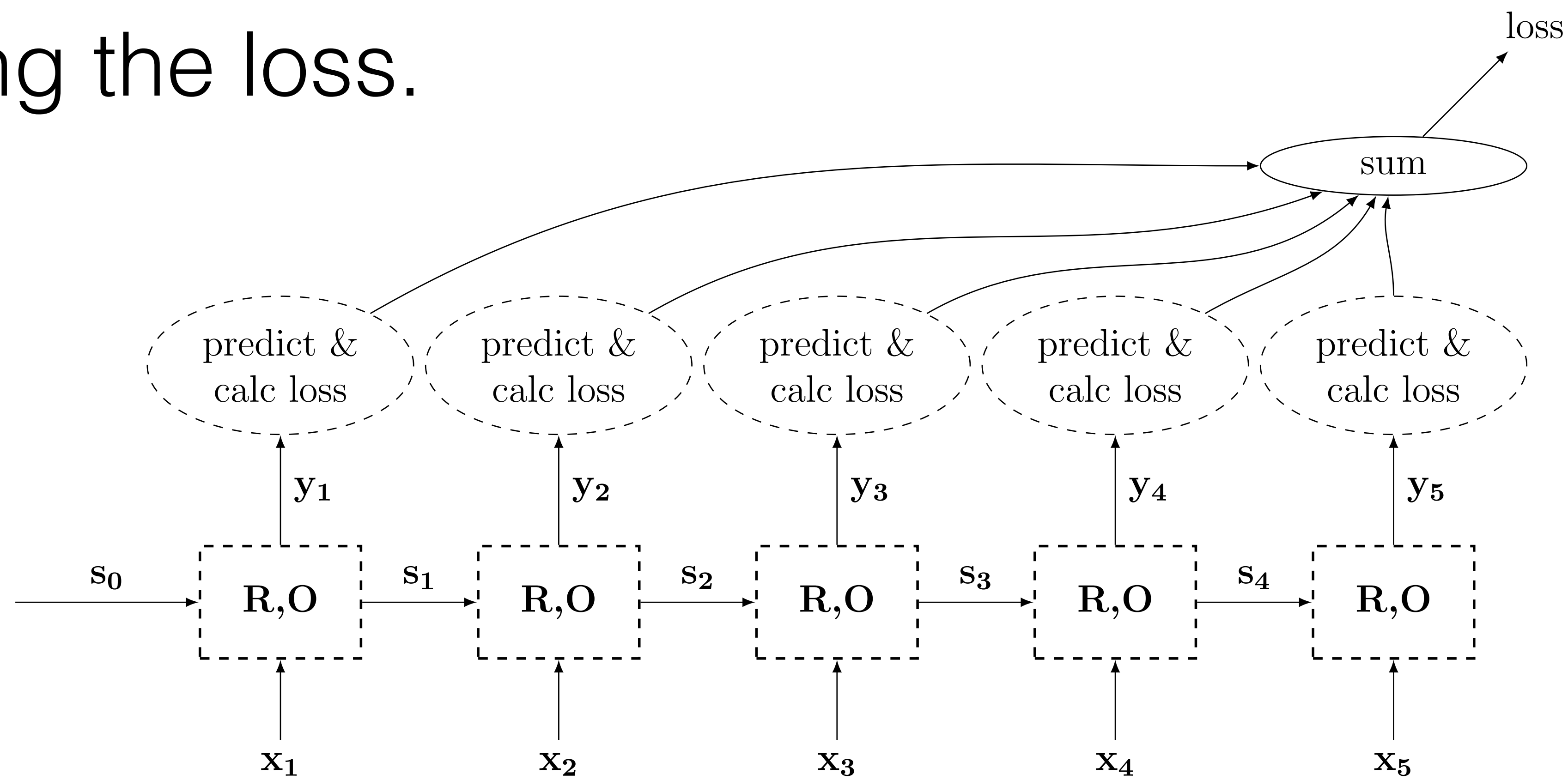


- Predict **word i** based on **words 1,...,i-1**

This is a **language model** with **infinite history**!

Transducers

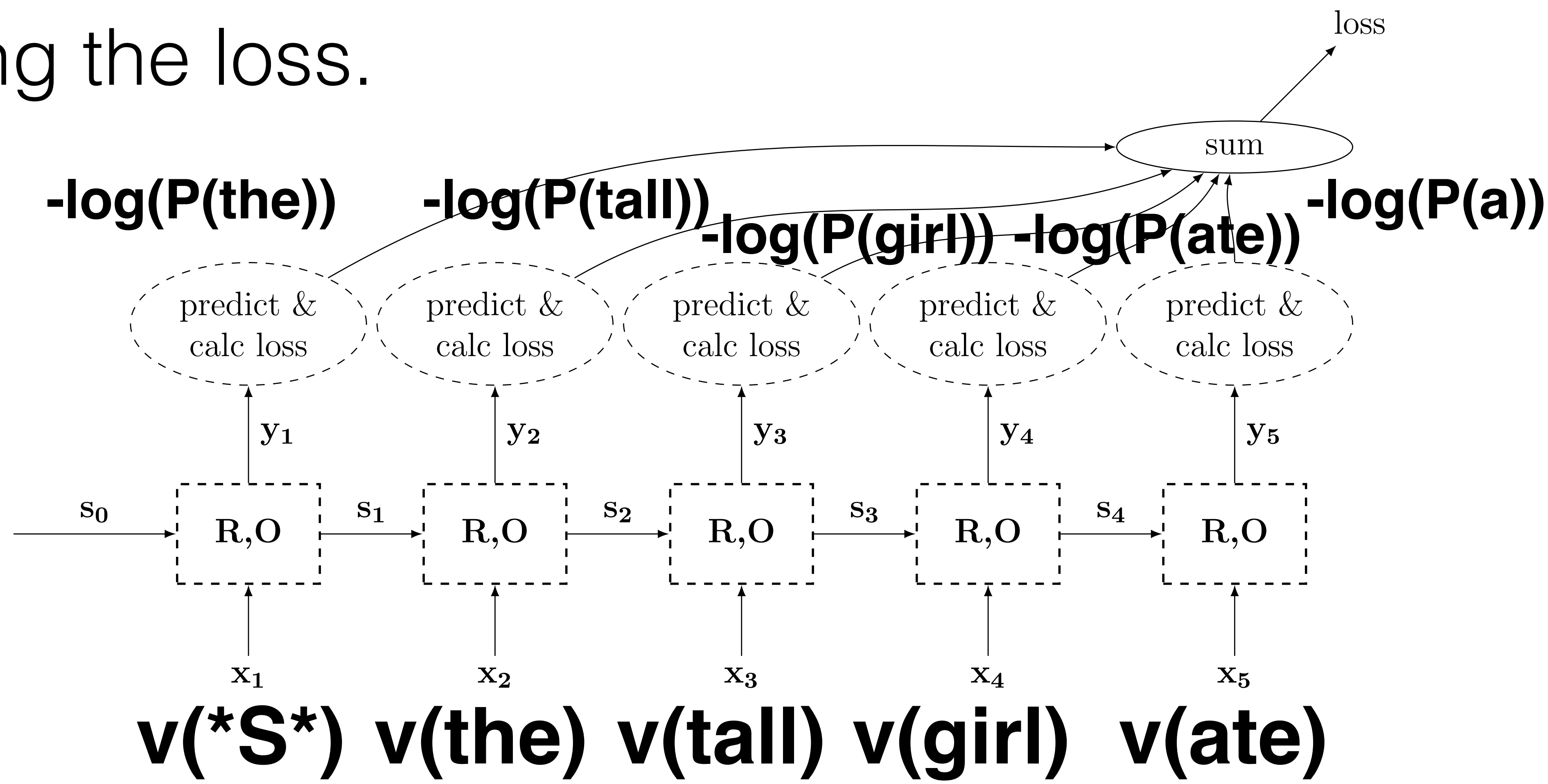
Defining the loss.



Transducer: predict something from each state.
Backprop the sum of errors all the way back.
Train the network to capture meaningful information

Transducers

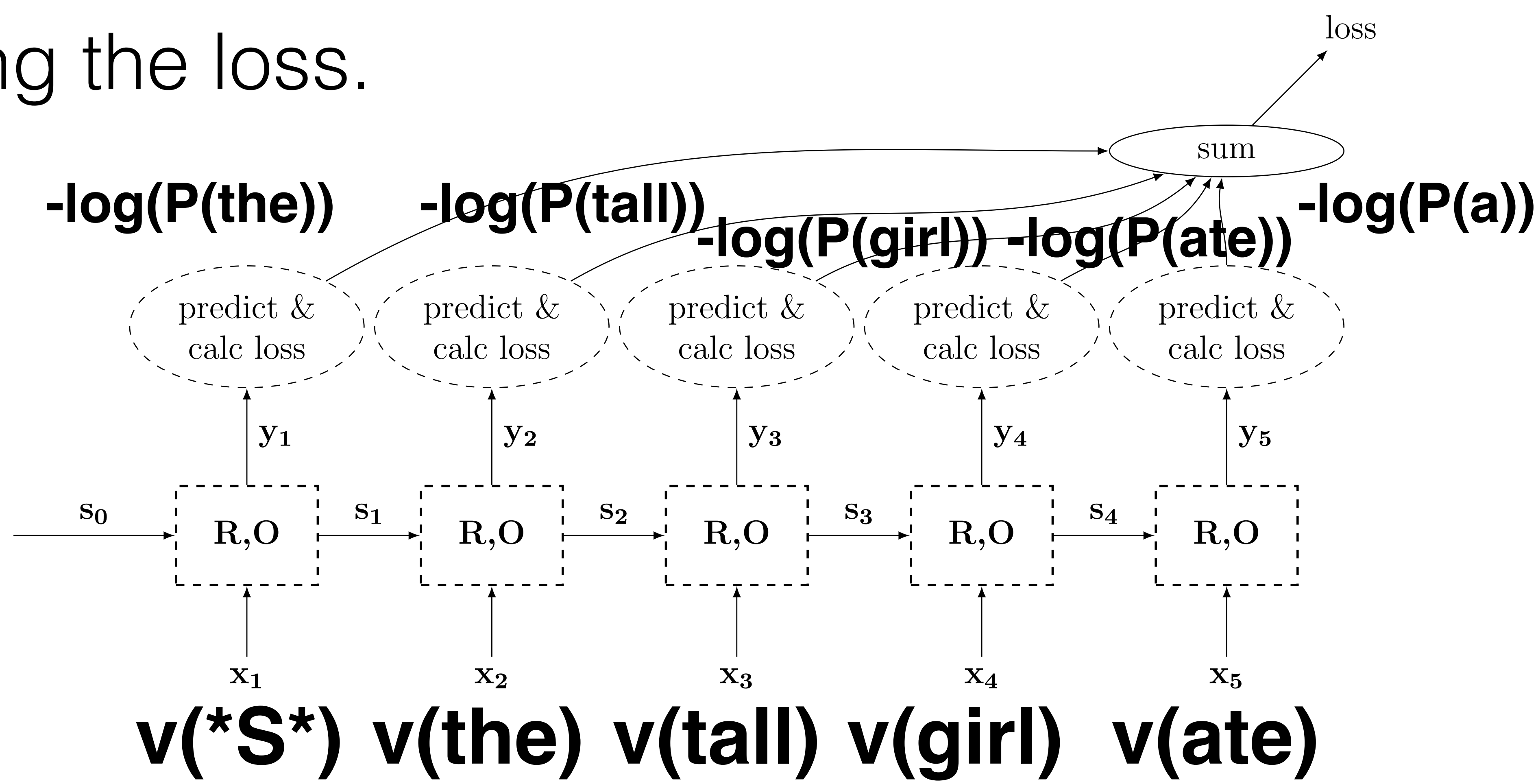
Defining the loss.



Transducer: predict something from each state.
Backprop the sum of errors all the way back.
Train the network to capture meaningful information

Transducers

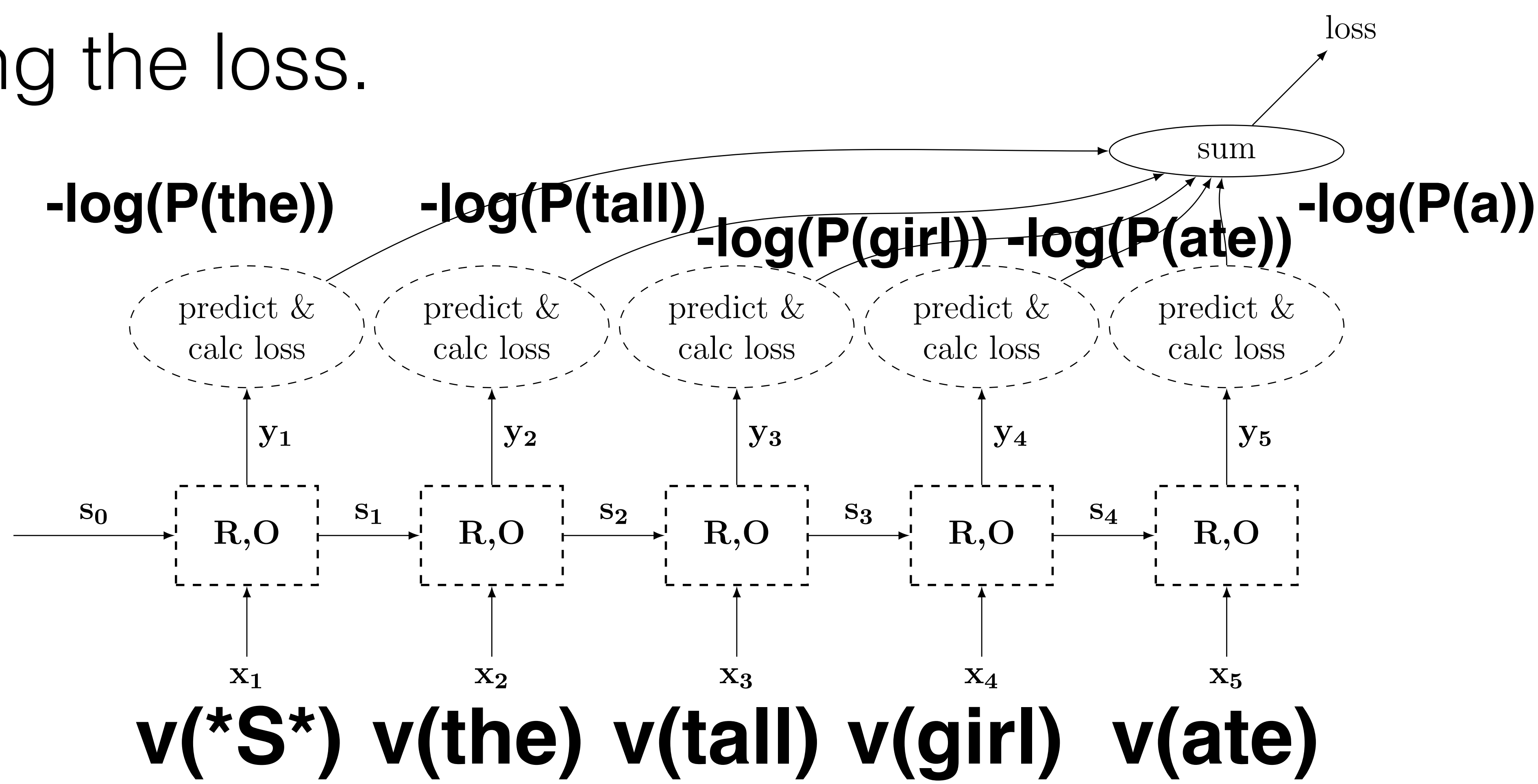
Defining the loss.



p(the|∗s∗)p(tall|∗s∗,the)p(girl|∗s∗,the,tall)p(ate|∗s∗,the,tall,girl)...

Transducers

Defining the loss.

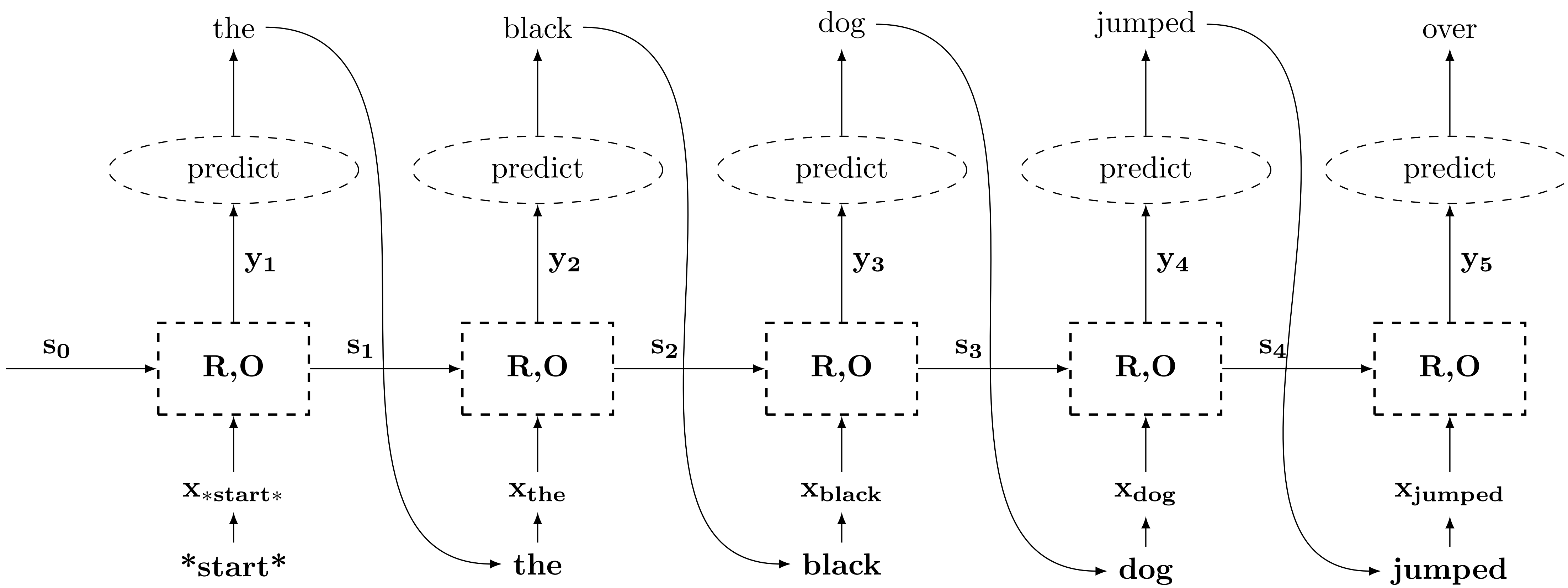


Transducer: predict something from each state.
Backprop the sum of errors all the way back.
Train the network to capture meaningful information

RNN Language Models

- *Training*: an RNN Tranducer.

- *Generation*: the output of step i is input to step i+1.



Andrej Karpathy blog

About   Hacker's guide to Neural Networks

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

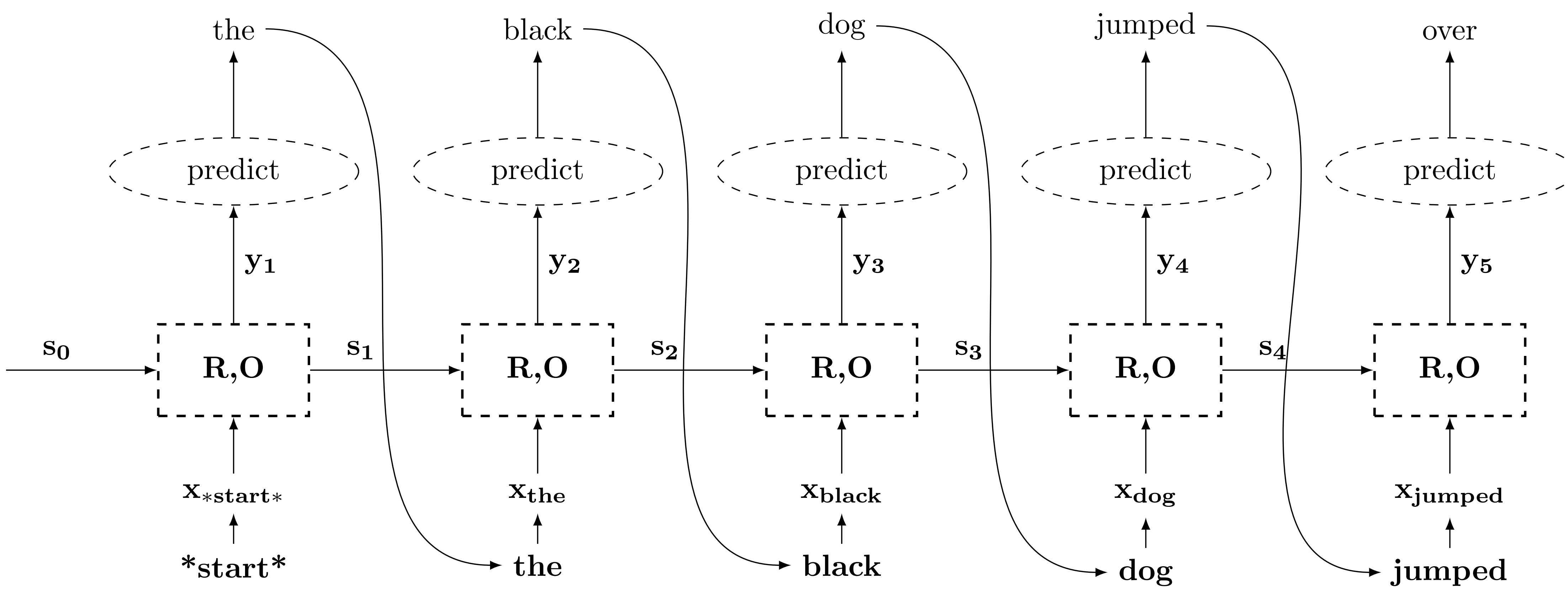
Train and generate from a character-level RNN (LSTM)

Generating Text with Recurrent Neural Networks

Ilya Sutskever ILYA@CS.UTORONTO.CA
James Martens JMARTENS@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, ON M5S 3G4 CANADA

RNN Language Models

- *Generation*: the output of step i is input to step i+1.



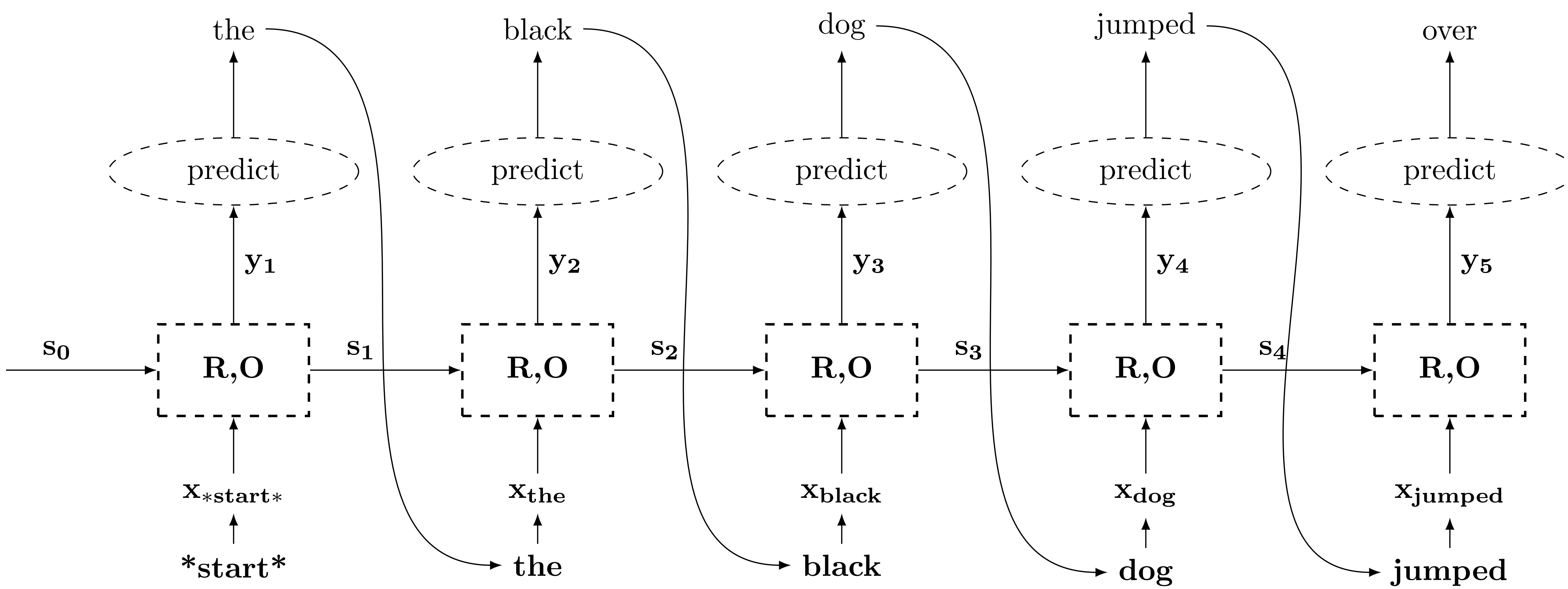
- **How to generate?**
  - Sampling.
  - Beam search.

Generated C Code (character level)

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

RNN Language Models

- *Generation*: the output of step i is input to step i+1.



- **How to generate?**
  - Sampling.
  - Beam search.

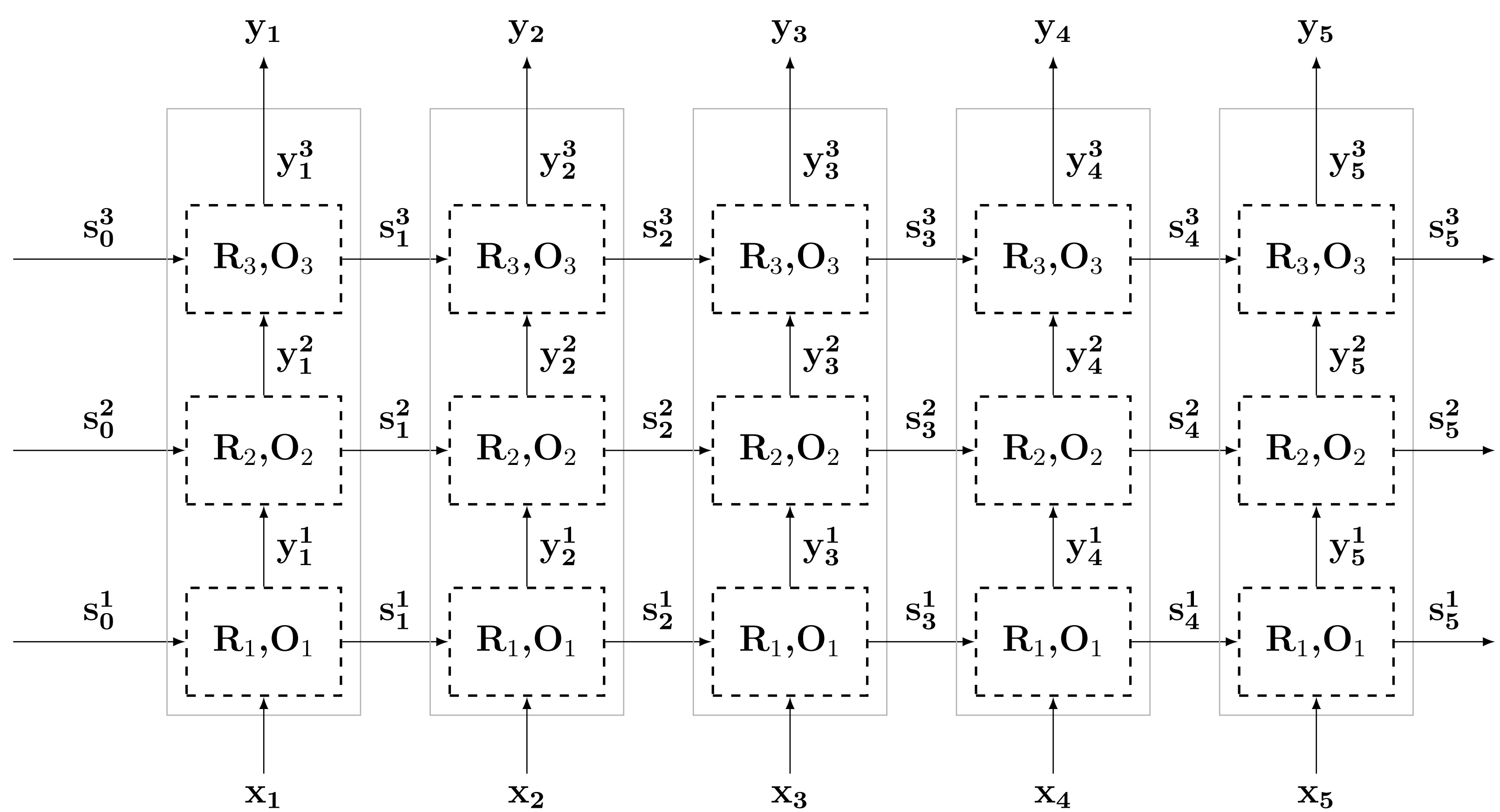
RNN Language Models

- ***Classification***: reminder: we can classify based on trained LMs.

$$predict(\mathbf{x_{1:n}}) = \arg\max_i P(\mathbf{x_{1:n}}|RNN_i)$$

$$P(\mathbf{x_{1:n}}|RNN) = \prod_{j=1}^{n} softmax(MLP(RNN(\mathbf{x_{1:j}})))_{[x_j]}$$

"deep RNNs"



RNN can be stacked
deeper is better!
(better how?)

RNNs and Hierarchical Structures

Andrej Karpathy's Char-LM

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Generated C Code (character level)

Andrej Karpathy's Char-LM

- This example is fascinating, and shows a lot of power.
- Definitely learning something hierarchical.
- Can we do a more controlled experiment?
- ... on a language which is less rigid?

Showing an example of modeling complex interactions

Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies

Tal Linzen[1,2] **Emmanuel Dupoux**[1]
LSCP[1] & IJN[2], CNRS,
EHESS and ENS, PSL Research University
{tal.linzen,
emmanuel.dupoux}@ens.fr

Yoav Goldberg
Computer Science Department
Bar Ilan University
yoav.goldberg@gmail.com

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.
- For example, subject verb agreement.

the boy kicks the ball

the boys kick the ball

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.
- For example, subject verb agreement.

the boy with the white shirt with the blue collar kicks the ball

the boys with the white shirts with the blue collars kick the ball

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.

- For example, subject verb agreement.

the boy (with the white shirt (with the blue collar)) kicks the ball

he boys (with the white shirts (with the blue collars)) kick the ball

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral rationalism are plato and immanuel kant .

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

replace rare words with their POS

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

choose a verb with a subject

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

cut the sentence at the verb

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

binary prediction task

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

↑

plural or singular?

in order to answer:

Need to learn the concept of number.

Need to identify the **subject** (ignoring irrelevant words)

Somewhat Harder Task

Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

choose a verb with a subject

Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

some prominent figures in the history of philosophy who have defended moral NN is plato and immanuel kant .

choose a verb with a subject
and flip its number.

Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant . V

some prominent figures in the history of philosophy who have defended moral NN is plato and immanuel kant . X

can the LSTM learn to distinguish good from bad sentences?

Can a sequence LSTM learn agreement?

$\boxed{\text{LSTMs learn agreement remarkably well.}}$

predicts number with **99**% accuracy.

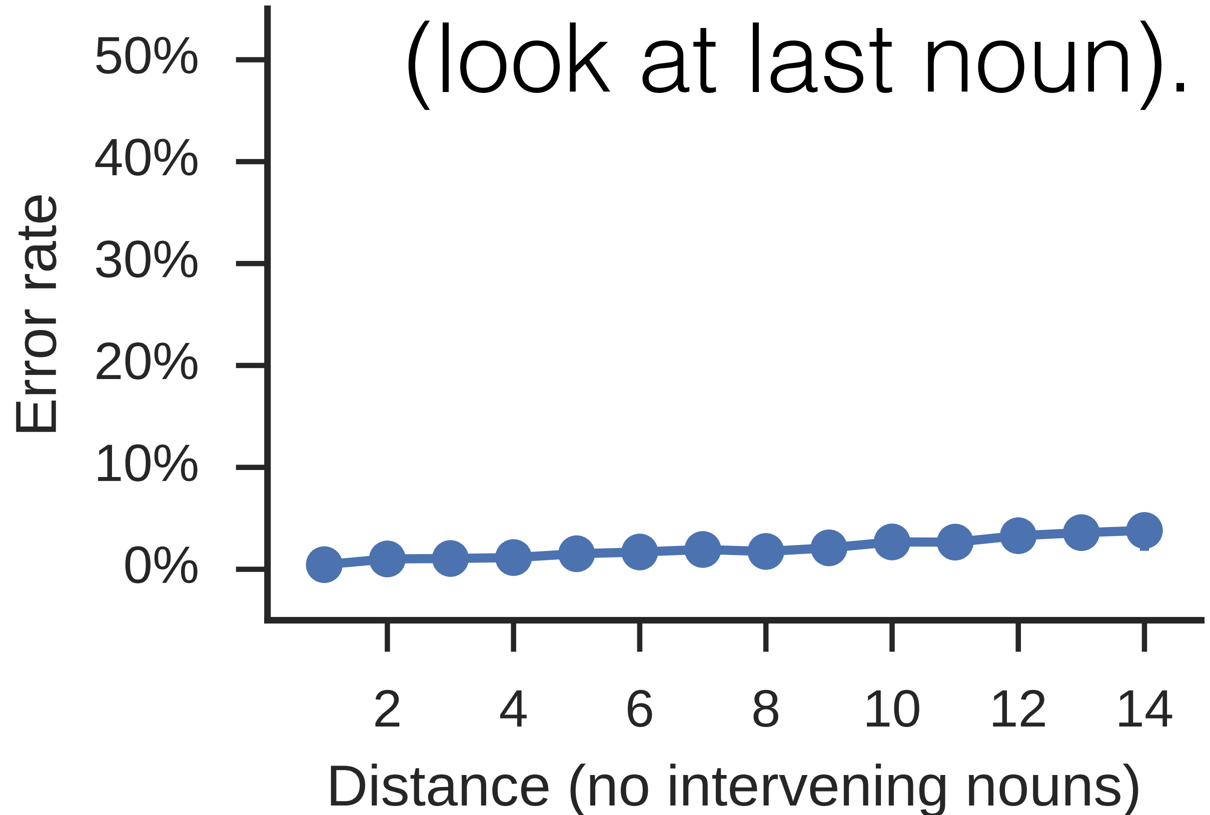
...but most examples are very easy
(look at last noun).

Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

predicts number with **99**% accuracy.

...but most examples are very easy
(look at last noun).



Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

predicts number with **99**% accuracy.

...but most examples are very easy
(look at last noun).

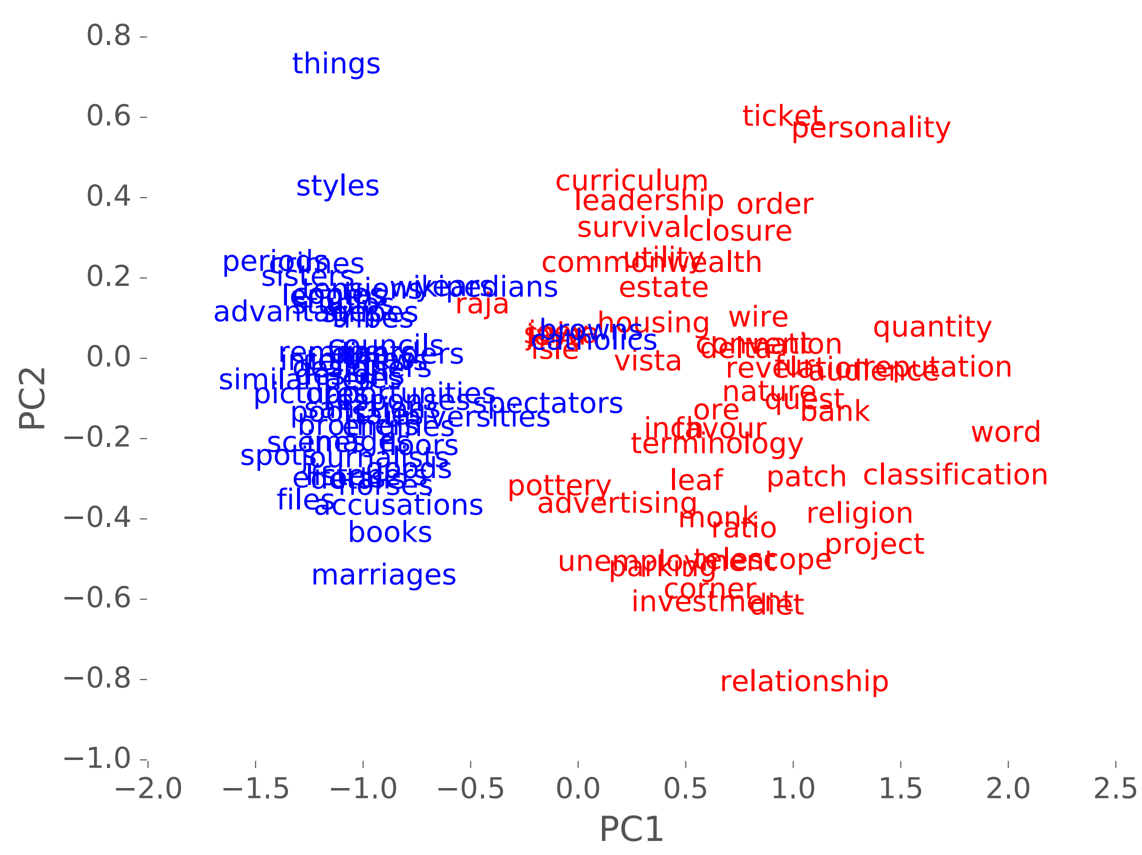
when restricted to cases
of at least one intervening noun:

97% accuracy

Can a sequence LSTM learn agreement?

$\boxed{\text{LSTMs learn agreement remarkably well.}}$

learns number of nouns

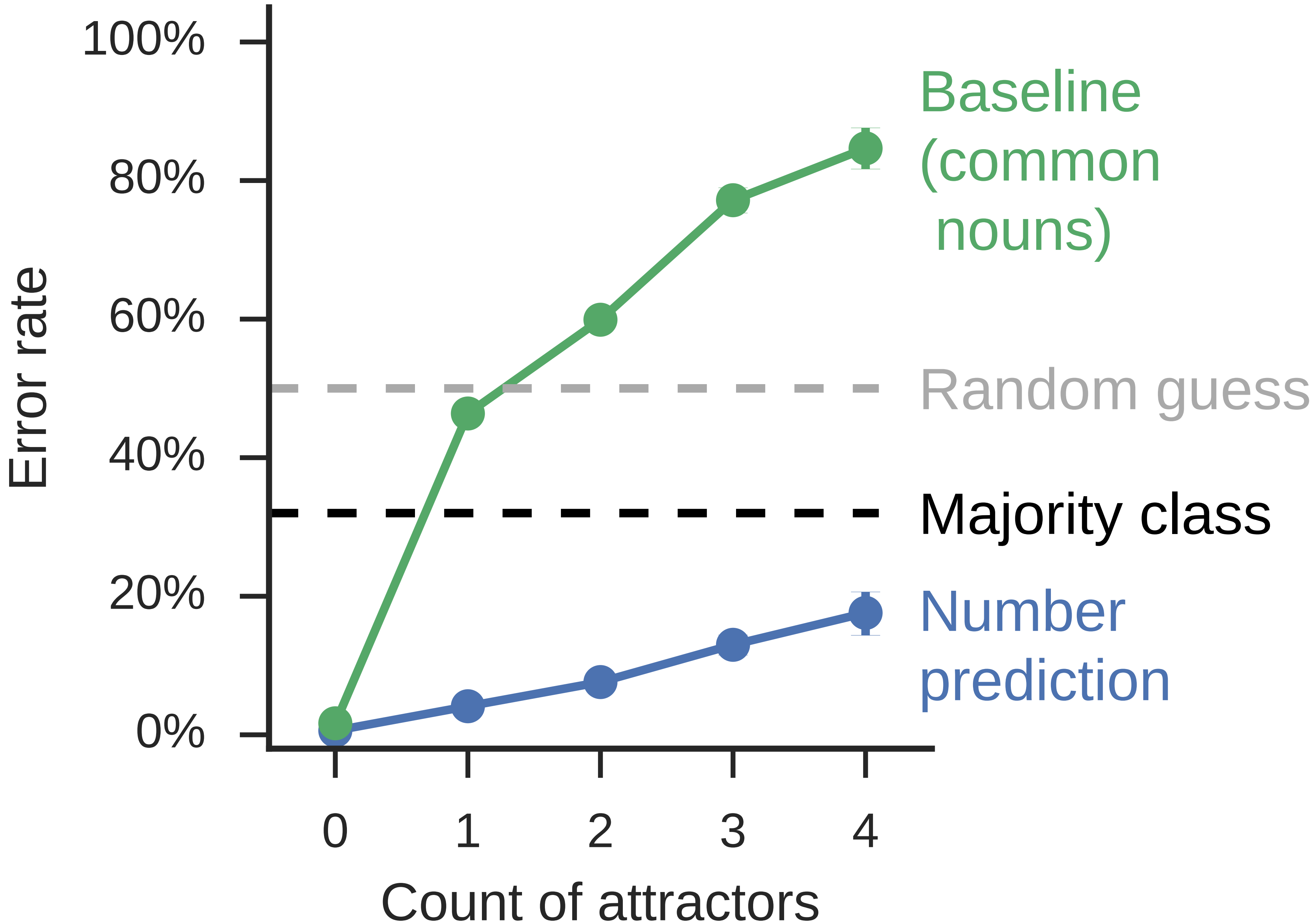


Can a sequence LSTM learn agreement?

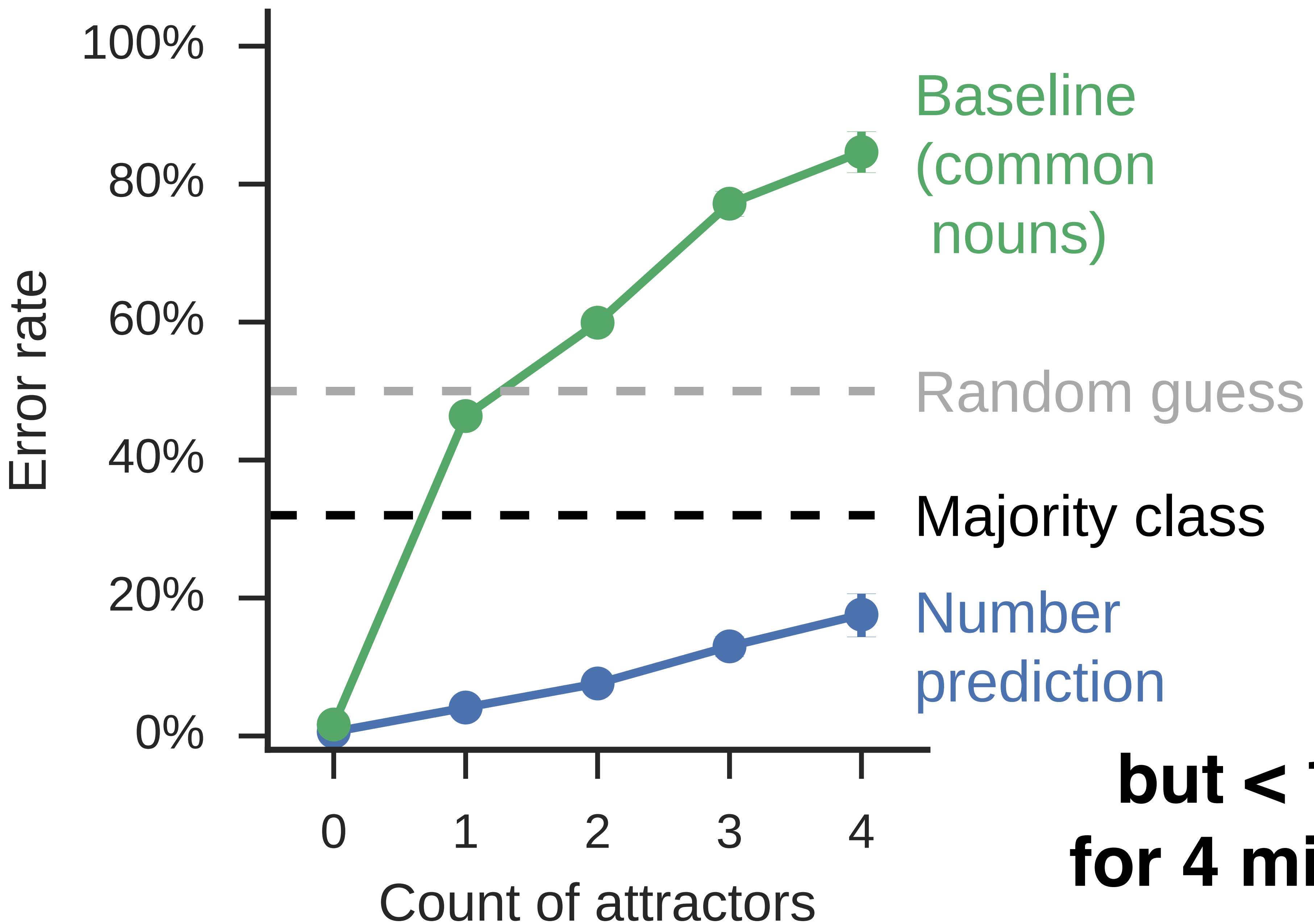
LSTMs learn agreement remarkably well.

more errors as the number of intervening nouns of opposite number increases

Can a sequence LSTM learn agreement?



Can a sequence LSTM learn agreement?



but < 17% err for 4 misleading nouns...

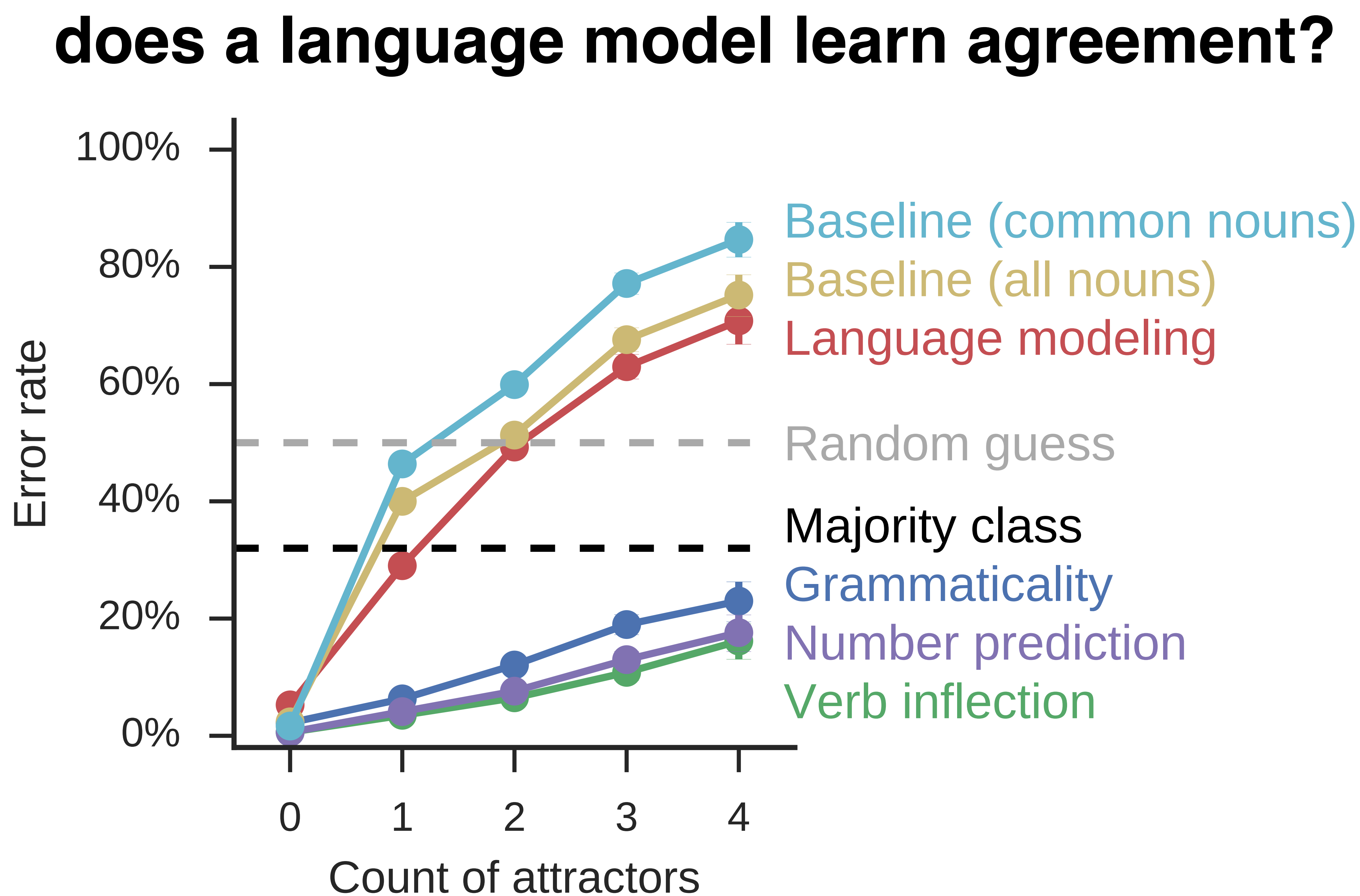
Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

but we trained it on the agreement task.

does a language model learn agreement?

Can a sequence LSTM learn agreement?



Can a sequence LSTM learn agreement?

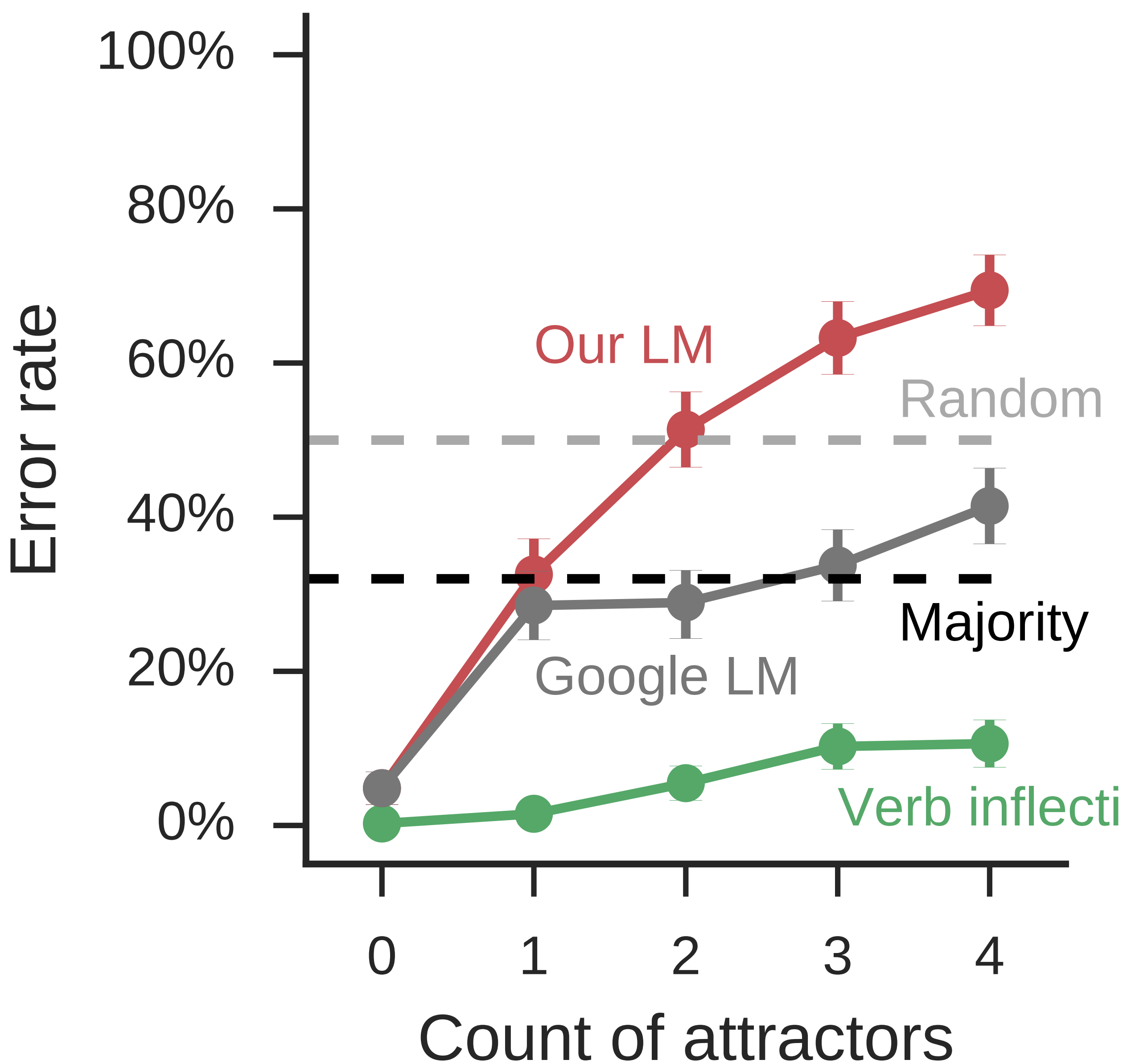
does a language model learn agreement?

what if we used the **best* LM in the world?**

*at the time

Can a sequence LSTM learn agreement?

does a language model learn agreement?



Google's beast LM does better than ours but still struggles considerably.

Can a sequence LSTM learn agreement?

does a language model learn agreement?

LSTMs can learn agreement very well.

But LSTM-LM **does not** learn agreement.

Explicit error signal is required.

Can a sequence LSTM learn agreement?

does a language model learn agreement?

LSTMs can learn agreement very well.

~~But LSTM-LM **does not** learn agreement.~~

~~**Explicit error signal is required.**~~

Later works: LSTM-LM **does** learn agreement.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

in many and diverse cases.

but we did manage to find some common trends.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

noun compounds can be tricky

Conservation **refugees live** in a world colored in shades of gray; limbo.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

Relative clauses are hard.

The **landmarks** *that* this <u>article</u> lists here **are** also run-of-the-mill and not notable.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

Reduced relative clauses are harder.

The **landmarks** this <u>article</u> lists here **are** also run-of-the-mill and not notable.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

	Error
No relative clause	3.2%
Overt relative clause	9.9%
Reduced Relative clause	**25%**

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

	Error
No relative clause	3.2%
Overt relative clause	9.9%
Reduced Relative clause	**25%**

humans also fail much more on reduced relatives.

The agreement experiment: recap

- We wanted to show LSTMs can't learn hierarchy.
    - --> We sort-of failed.
- LSTMs learn to cope with natural-language patterns that exhibit hierarchy, based on minimal and indirect supervision.
- But some sort of relevant supervision is required.

RNNs recap (for now)

- Representing a **variable-length** sequence of vectors as a single vector.
- Capturing the **global order** of the elements.
- Using a **recursively defined** trainable function.
- We saw the following configurations:
    - Acceptors
    - Transducers
    - Deep
- RNN Language Models.
- Generation from a language model.