Deep Learning for NLP: Recurrent Networks

COMP90042

Natural Language Processing

Lecture 8

THE UNIVERSITY OF MELBOURNE

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*

N-gram Language Models

- Can be implemented using counts (with smoothing)
- Can be implemented using feed-forward neural networks
- Generates sentences like (trigram model):
  - *I saw a table is round and about*
- Problem: limited **context**

Recurrent Neural Network (RNN)

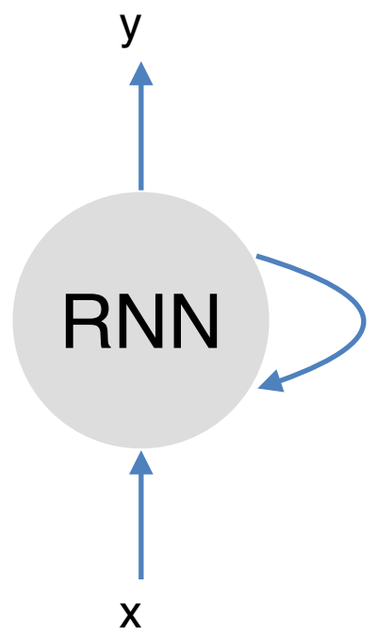
- RNNs allow representing arbitrarily sized inputs
- Core Idea: processes the input sequence one at a time, by applying a recurrence formula
- Uses a state vector to represent contexts that have been previously processed

Recurrent Neural Network (RNN)

function with parameters

$$s_i = f(s_{i-1}, x_i)$$

new state | previous state | input



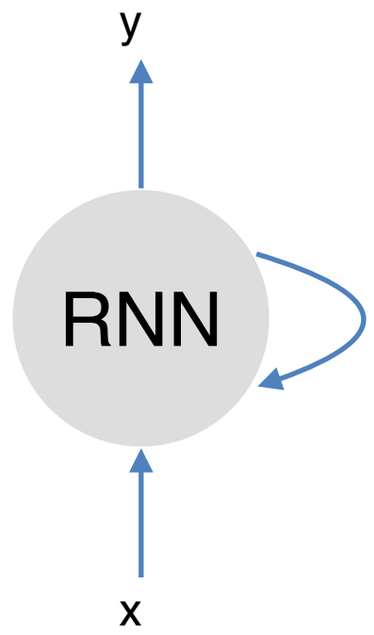
Recurrent Neural Network (RNN)

$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$

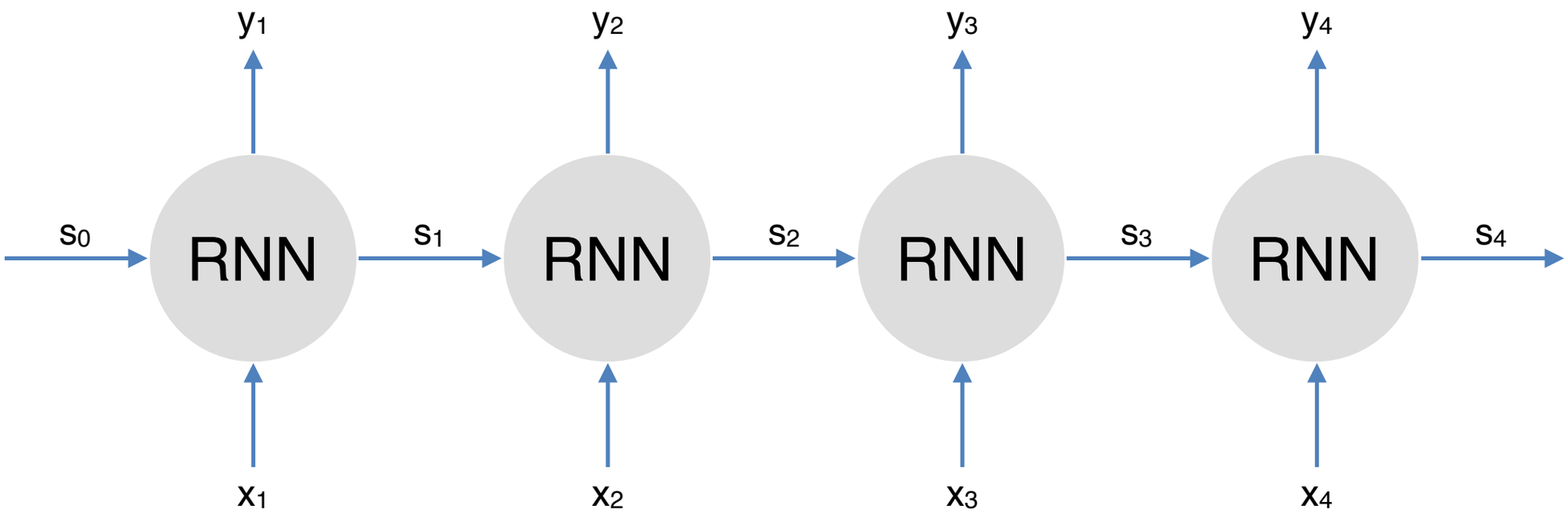
new state

previous state

input



RNN Unrolled



$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$
$$y_i = \sigma(W_y s_i)$$

“Simple RNN”

- Same parameters are shared across all time steps

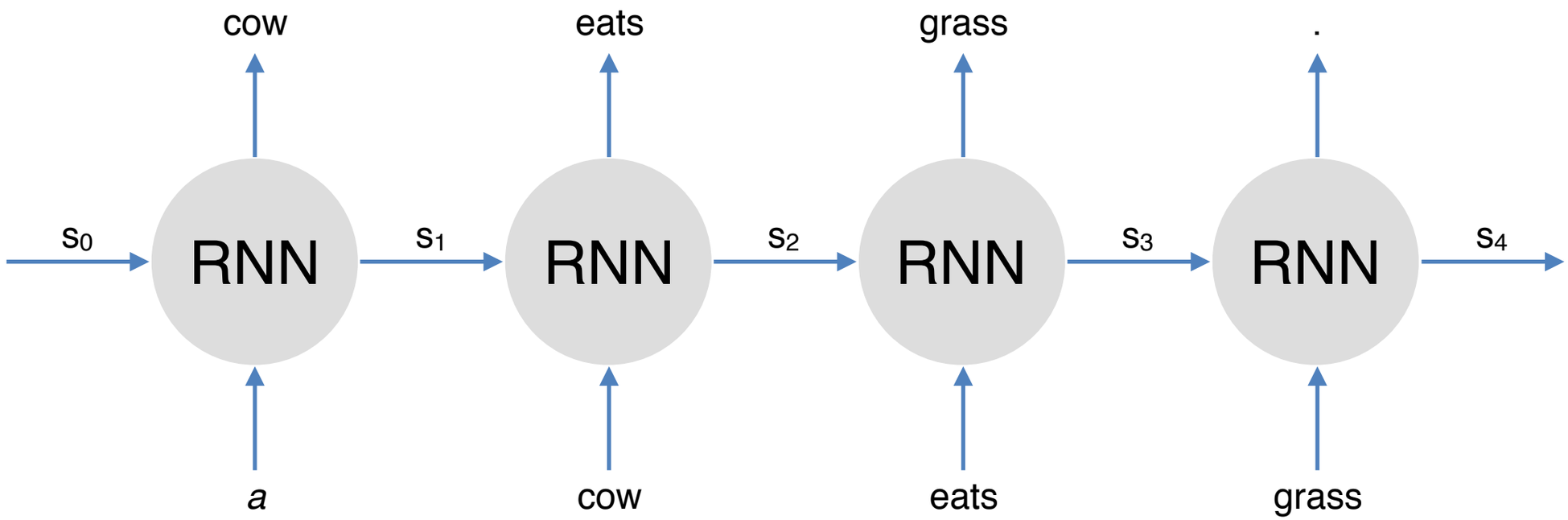
RNN Training

- An unrolled RNN is just a very deep neural network
- But same parameters are shared across many time steps

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

- To train RNN, we just need to create the unrolled computation graph given an input sequence
- And use backpropagation algorithm to compute gradients as usual
- This procedure is called **backpropagation through time**

(Simple) RNN for Language Model

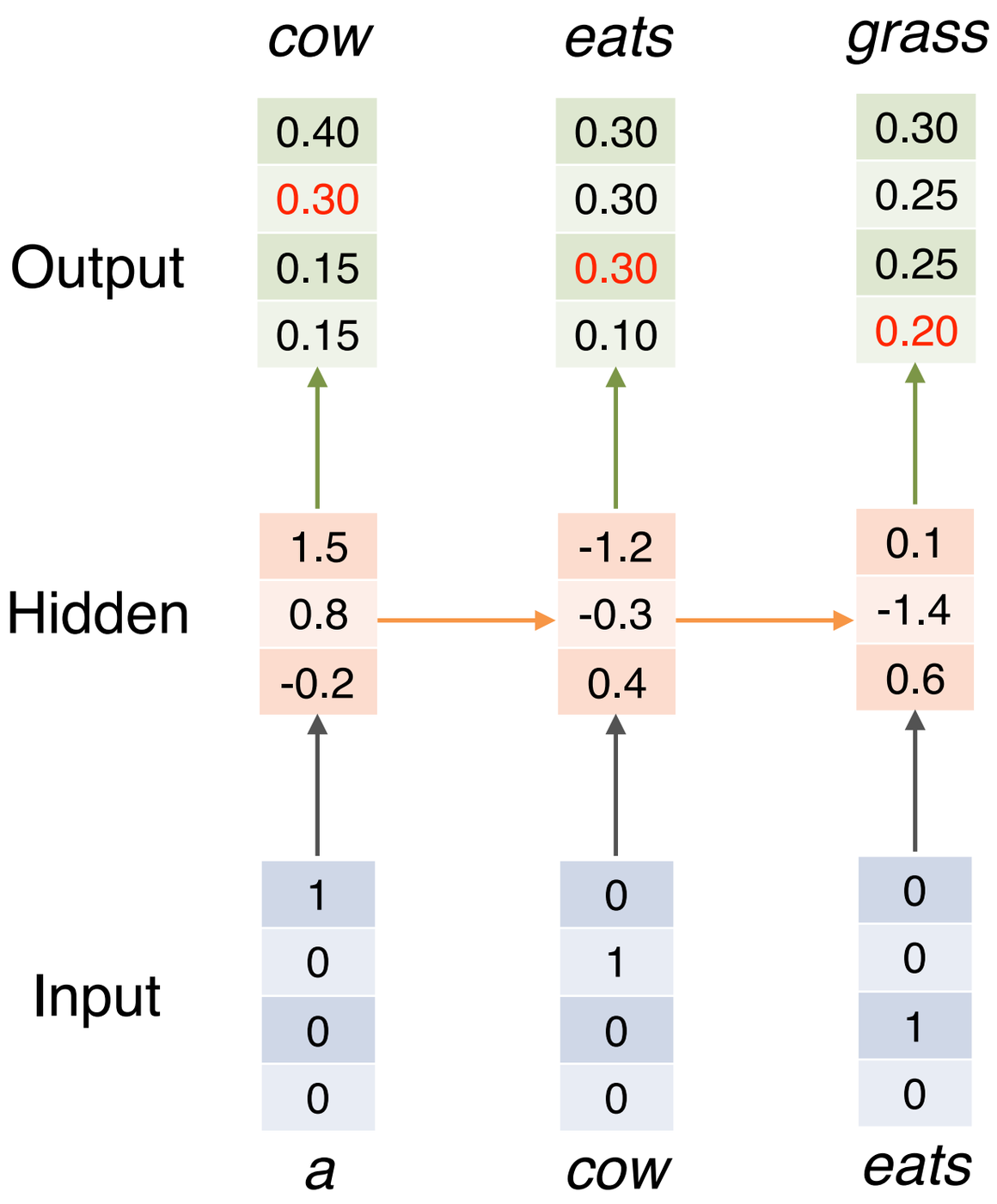


$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$

$$y_i = \text{softmax}(W_y s_i)$$

- Input words mapped to an embedding
- Output = next word

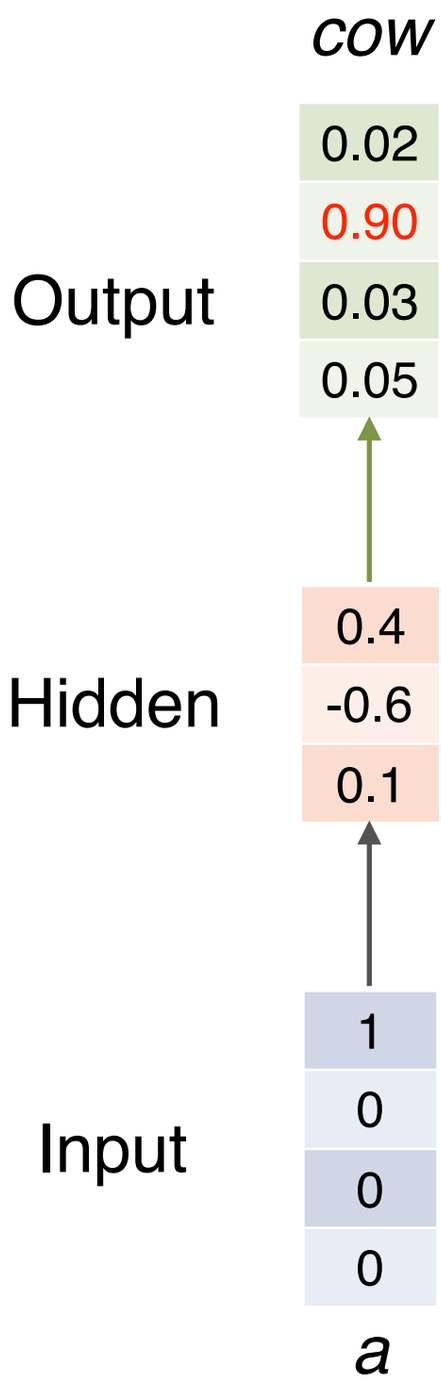
RNN Language Model - Training



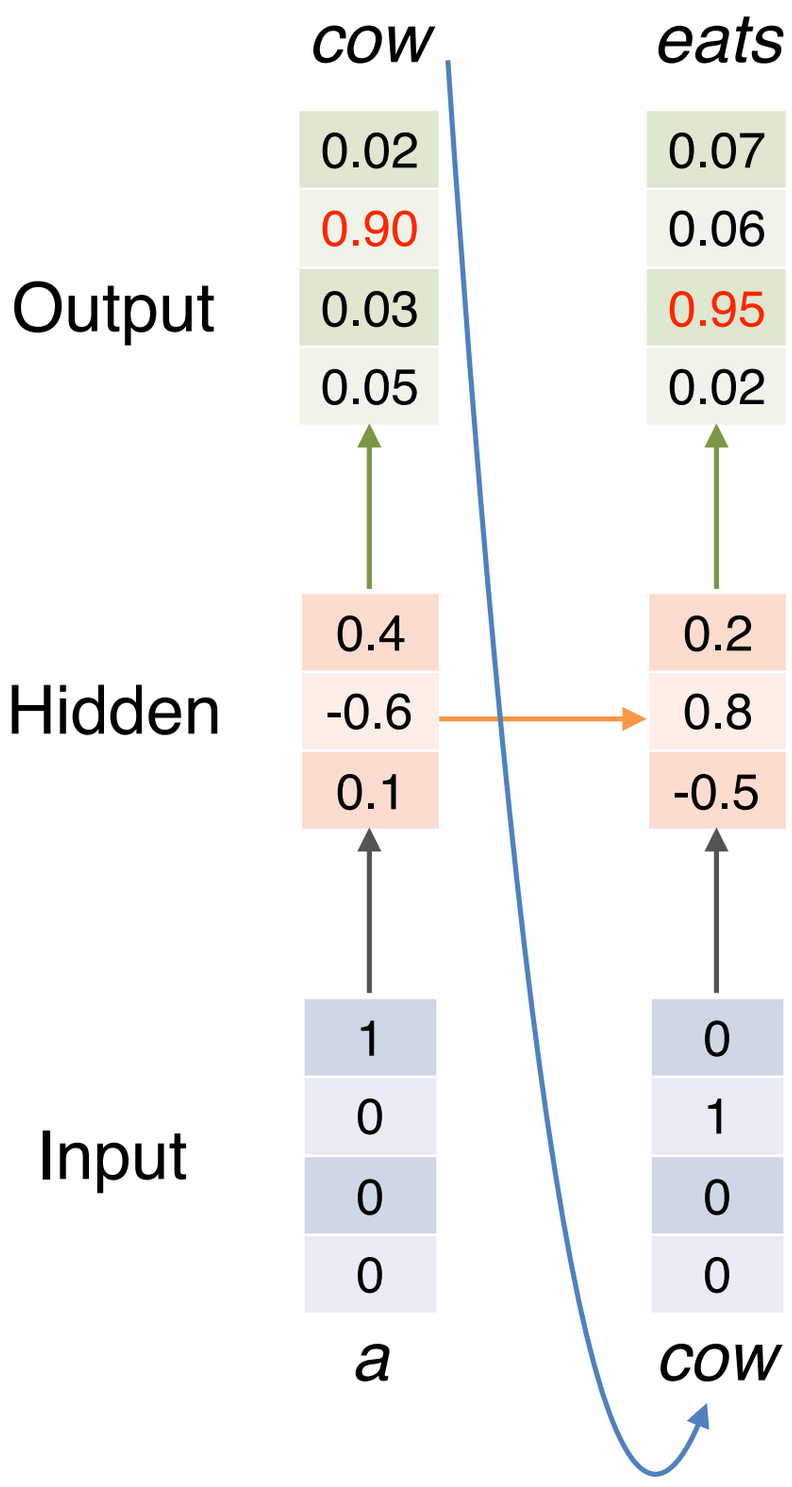
- Vocabulary: [*a*, *cow*, *eats*, *grass*]
- Training Example: *a cow eats grass*

$$s_i = \tanh(W_s s_{i-1} + W_x x_i + b)$$
$$y_i = \text{softmax}(W_y s_i)$$

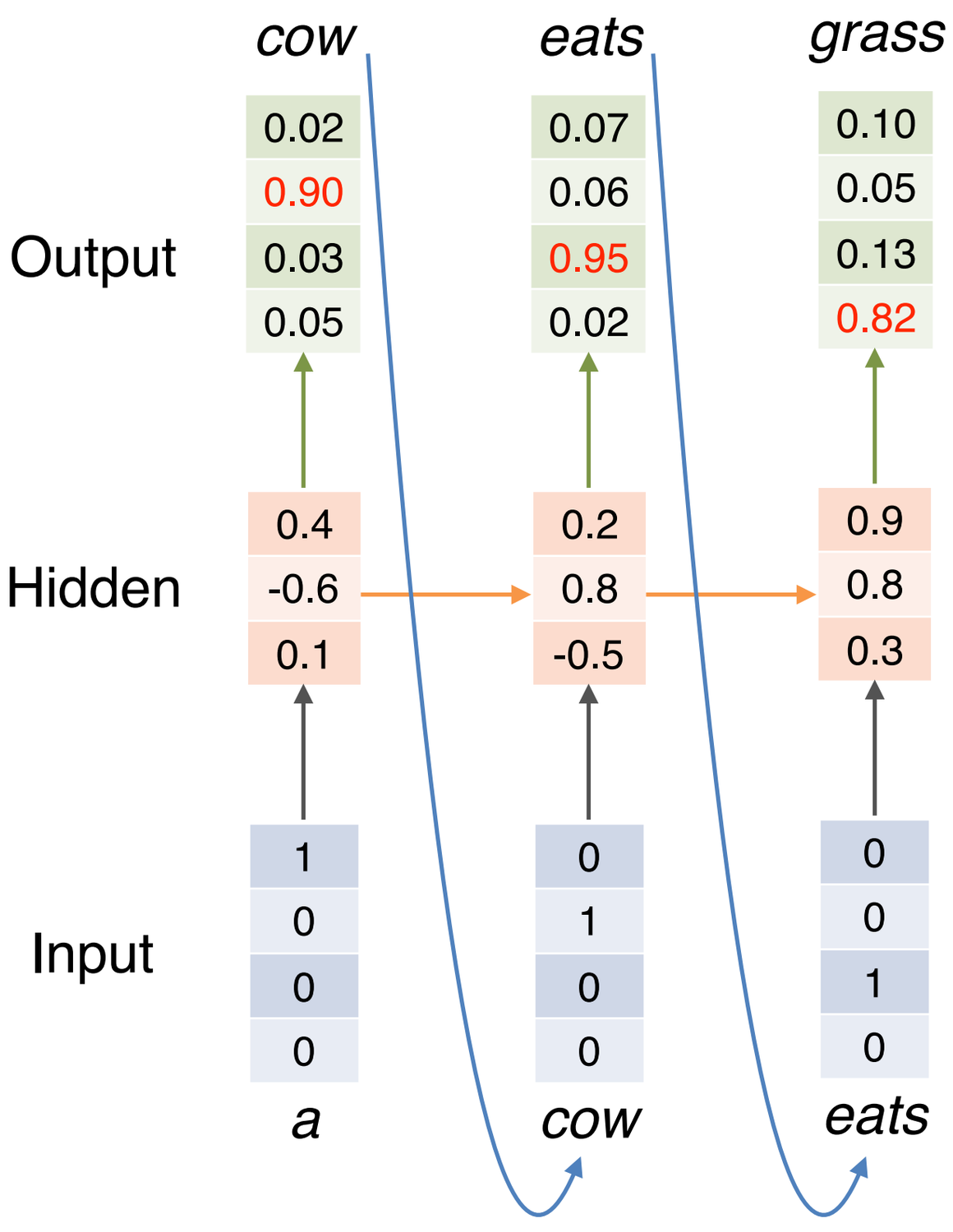
RNN Language Model - Generation



RNN Language Model - Generation



RNN Language Model - Generation



Language Model… Solved?

- RNN has the capability to model infinite context
- But can it actually capture long-range dependencies in practice?

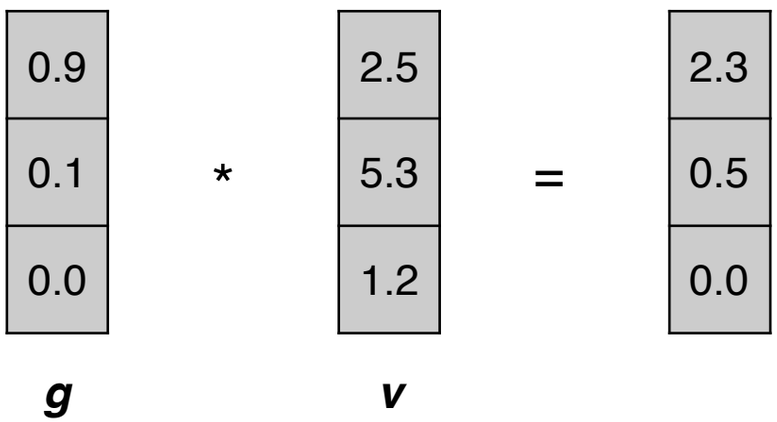
- No… due to “**vanishing gradients**”
- Gradients in later steps diminish quickly during backpropagation
- Earlier inputs do not get much update

Long Short-term Memory (LSTM)

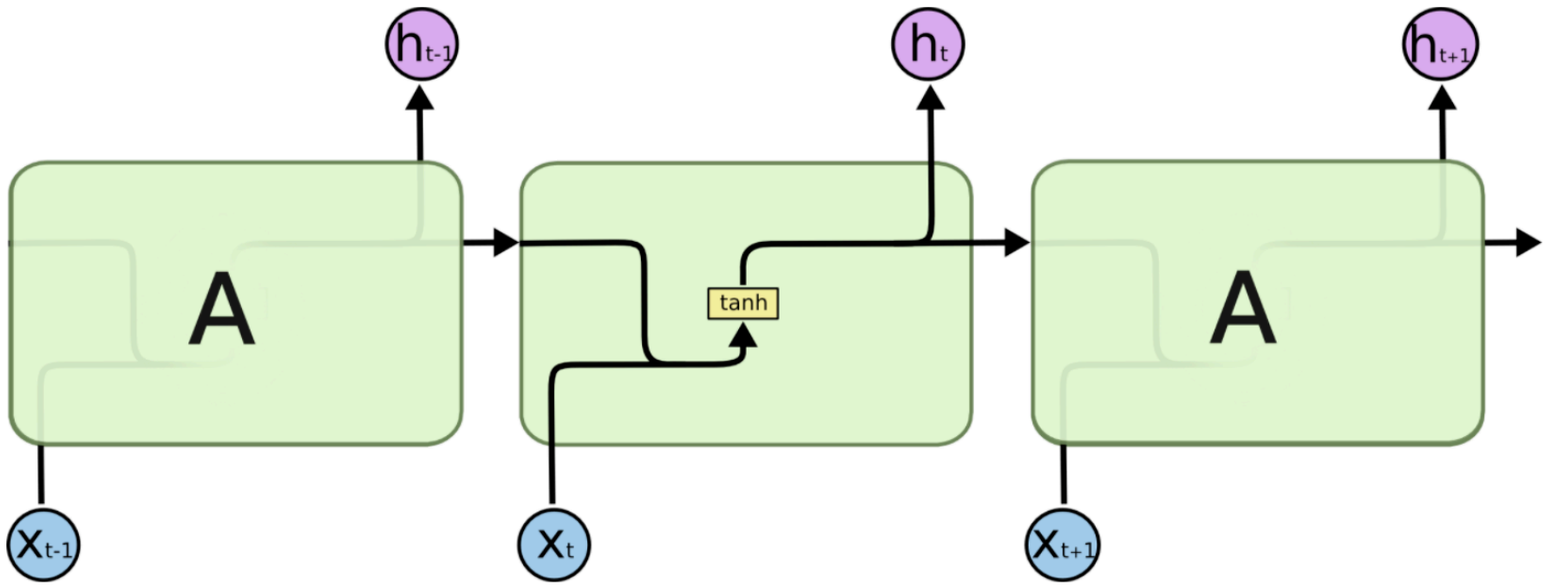
- LSTM is introduced to solve vanishing gradients
- Core idea: have “memory cells” that preserve gradients across time
- Access to the memory cells is controlled by “gates”
- For each input, a gate decides:
  - how much the new input should be written to the memory cell
  - and how much content of the current memory cell should be forgotten

Long Short-term Memory (LSTM)

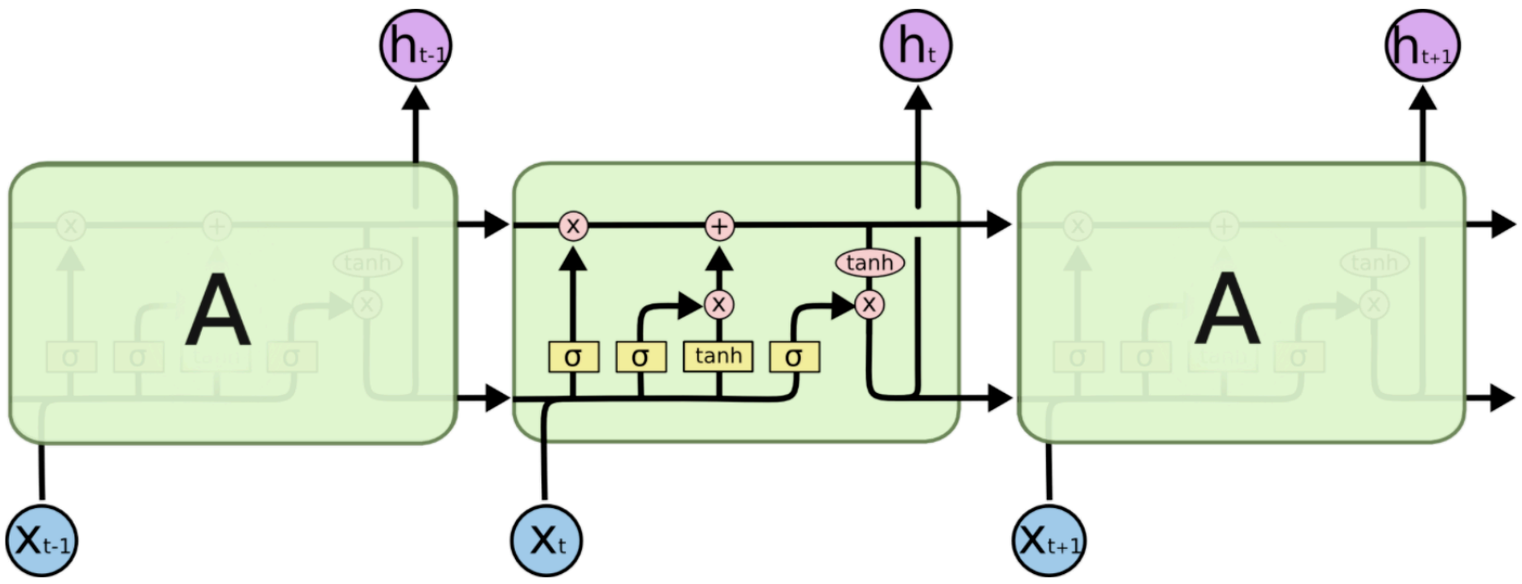
- A gate ***g*** is a vector
  - each element has values between 0 to 1
- ***g*** is multiplied component-wise with vector ***v***, to determine how much information to keep for ***v***
- Use sigmoid function to keep values of ***g*** close to either 0 or 1



LSTM vs. Simple RNN



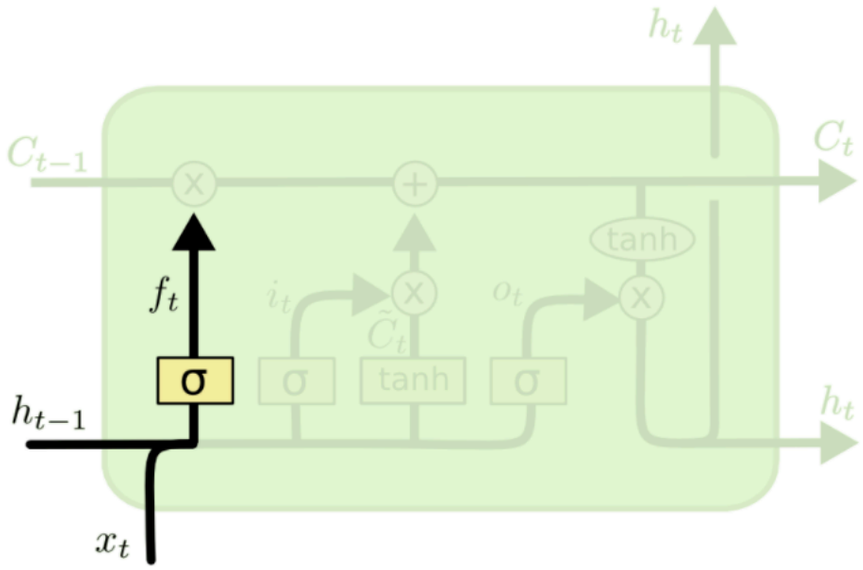
The repeating module in a standard RNN contains a single layer.



The repeating module in an LSTM contains four interacting layers.

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM: Forget Gate

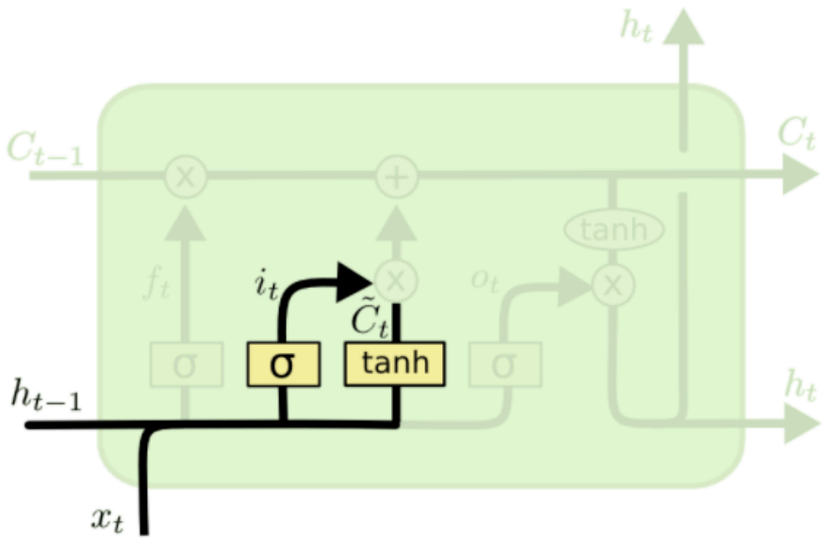


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

previous state current input

- Controls how much information to “forget” in the memory cell (C_{t-1})
- *The cats that the **boy** likes*
- Memory cell was storing pronoun information (*cats*)
- The cell should now forget *cats* and store *boy* to correctly predict the singular verb *likes*

LSTM: Input Gate

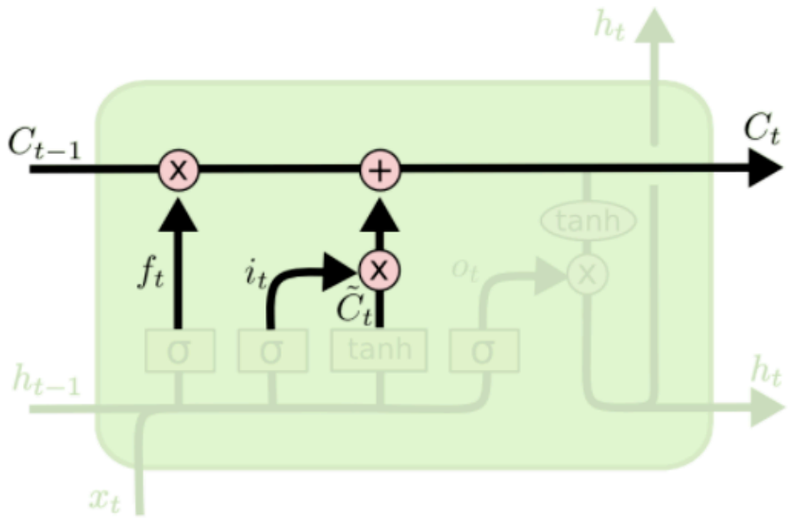


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- Input gate controls how much new information to put to memory cell
- $\tilde{C}_t$ = new distilled information to be added
  - e.g. information about *boy*

LSTM: Update Memory Cell

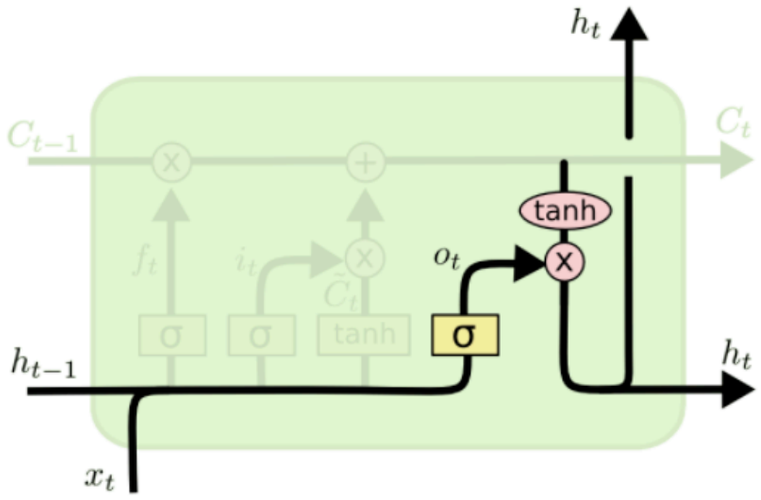


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

forget gate input gate

- Use the forget and input gates to update memory cell

LSTM: Output Gate

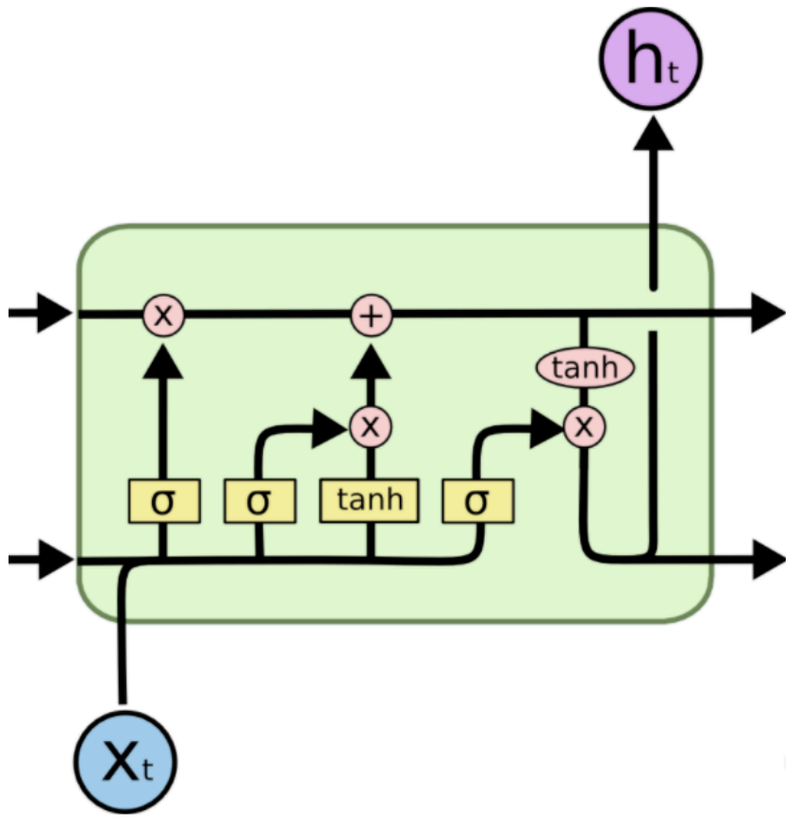


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- Output gate controls how much to distill the content of the memory cell to create the next state (h_t)

LSTM: Summary



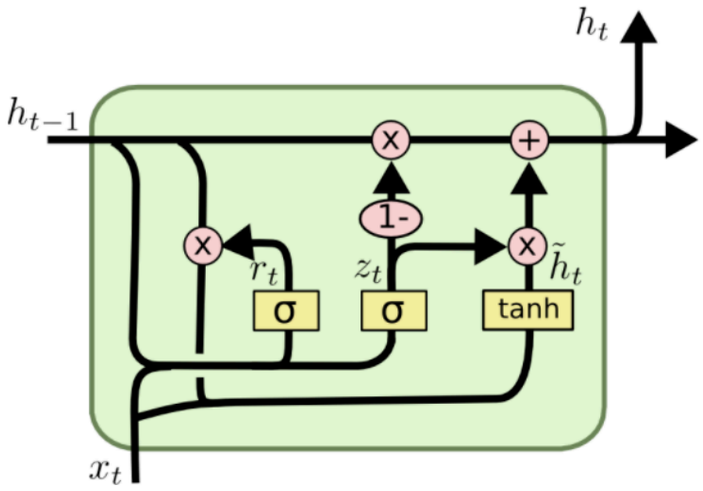
$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

Variants

- Peephole connections
  - Allow gates to look at cell state

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$

- Gated recurrent unit (GRU)
  - Simplified variant with only 2 gates



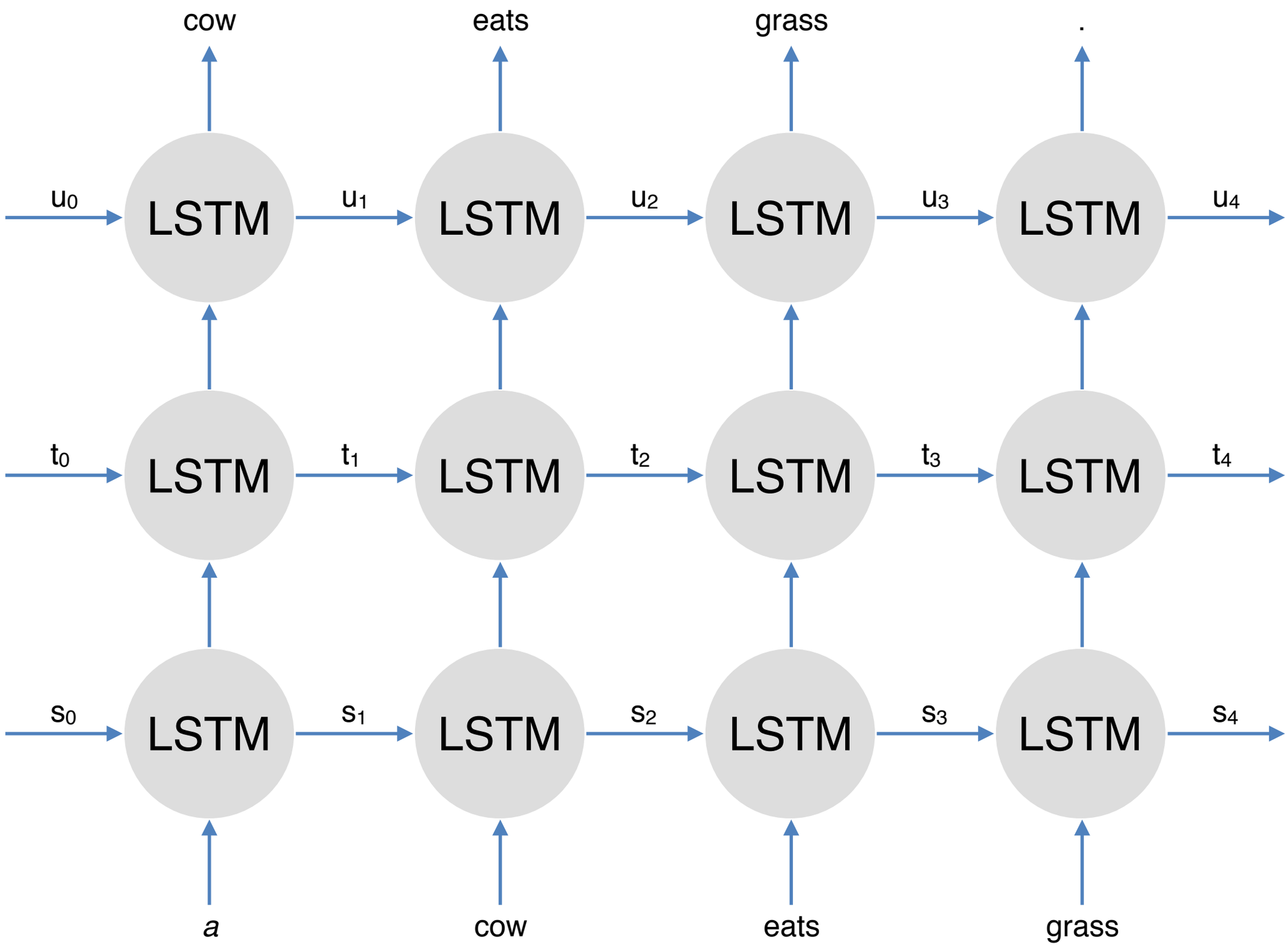
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

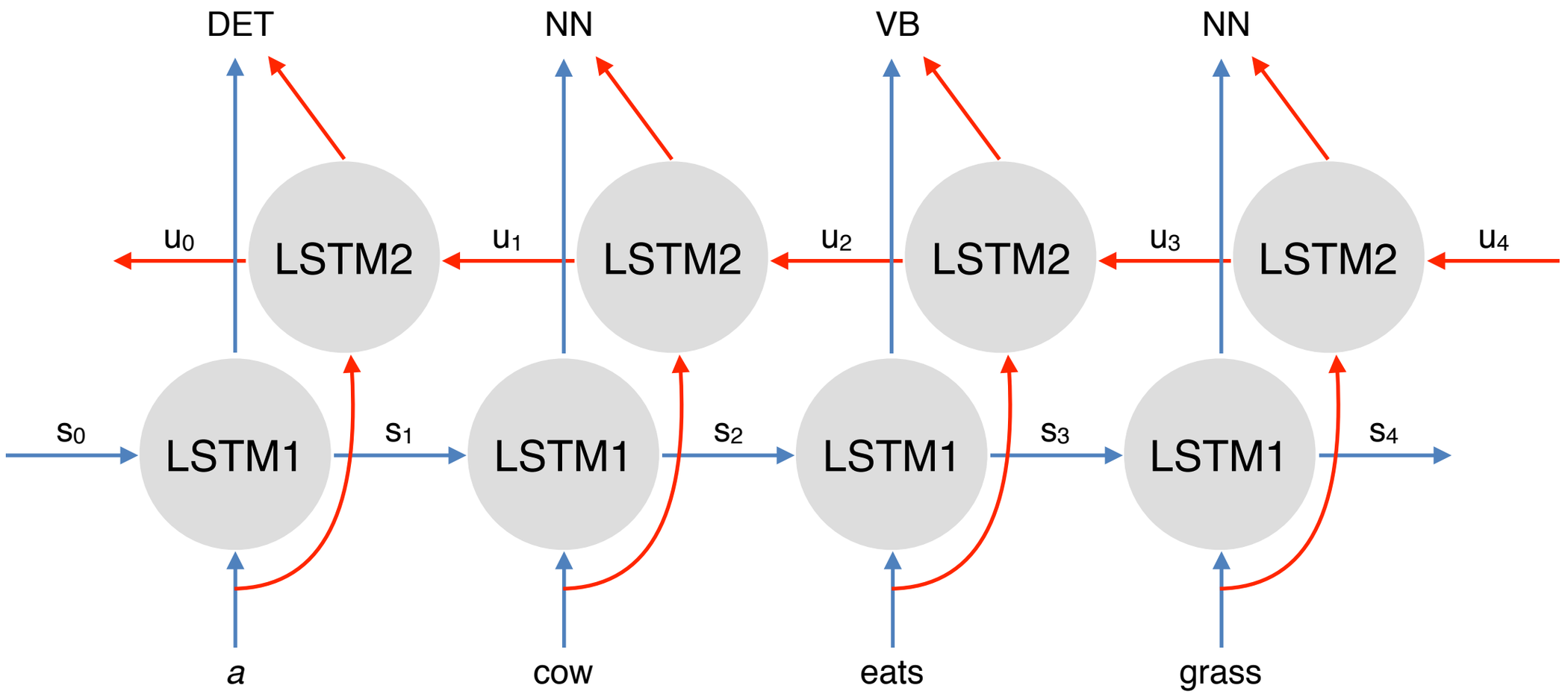
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Multi-layer LSTM



Bidirectional LSTM



$$y_i = \text{softmax}(W_s[s_i, u_i])$$

Shakespeare Generator

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

- Training data = all works of Shakespeare
- Model: 3-layer **character** RNN, hidden dimension = 512

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Wikipedia Generator

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

- Training data = 100MB of Wikipedia raw data

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Code Generator

```
/*
 * 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;
}
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Deep-Speare

- Generates Shakespearean sonnets

```
python sonnet_gen.py -m trained_model/ -d 1

Temperature = 0.6 - 0.8
  01  [0.43]  with joyous gambols gay and still array
  02  [0.44]  no longer when he twas, while in his day
  03  [0.00]  at first to pass in all delightful ways
  04  [0.40]  around him, charming and of all his days


python sonnet_gen.py -m trained_model/ -d 2

Temperature = 0.6 - 0.8
  01  [0.44]  shall i behold him in his cloudy state
  02  [0.00]  for just but tempteth me to stop and pray
  03  [0.00]  a cry: if it will drag me, find no way
  04  [0.40]  from pardon to him, who will stand and wait
```

THE TIMES Log in Subscribe

Computers produce poetry by the metre

Mark Bridge, Technology Correspondent

Tuesday July 31 2018, 12.01am BST, The Times

Shakespeare's 12th sonnet, illustrated by John Gilbert
CULTURE CLUB/GETTY IMAGES

Sonnets written by a computer are superior to Shakespeare's in some respects — and the general public is unable to tell the difference between the two, researchers say.

Computer scientists have trained a "neural network" using 2,600 sonnets taken from a free online database and put the system to work

DIGITAL TRENDS

Best Movies on Netflix | Best Shows on Netflix | Best Movies on Hulu | Best Shows on Hulu

Emerging Tech

Move over, Shakespeare: This sonnet-writing A.I. is the poet we need

By Luke Dormehl
July 30, 2018

Daily Mail AUSTRALIA

Home | U.K. | U.S. | News | World News | Sport | TV&Showbiz | Femail | Health | Science | W

Latest Headlines | Facebook | YouTube | Google | eBay

Can YOU spot the real Shakespearean sonnet? The AI learning how write its own poetry

- **Researchers at IBM used 2600 sonnets to train their AI**
- **Researchers asked workers from a crowdsourcing website to rate them**
- **Found they were unable to distinguish them from the real deal**
- **However, experts were more critical and said they still lacked real emotion**

By MARK PRIGG FOR DAILYMAIL.COM
PUBLISHED: 05:25 AEDT, 28 July 2018 | **UPDATED:** 05:56 AEDT, 28 July 2018

Share 50 shares 16 View comments

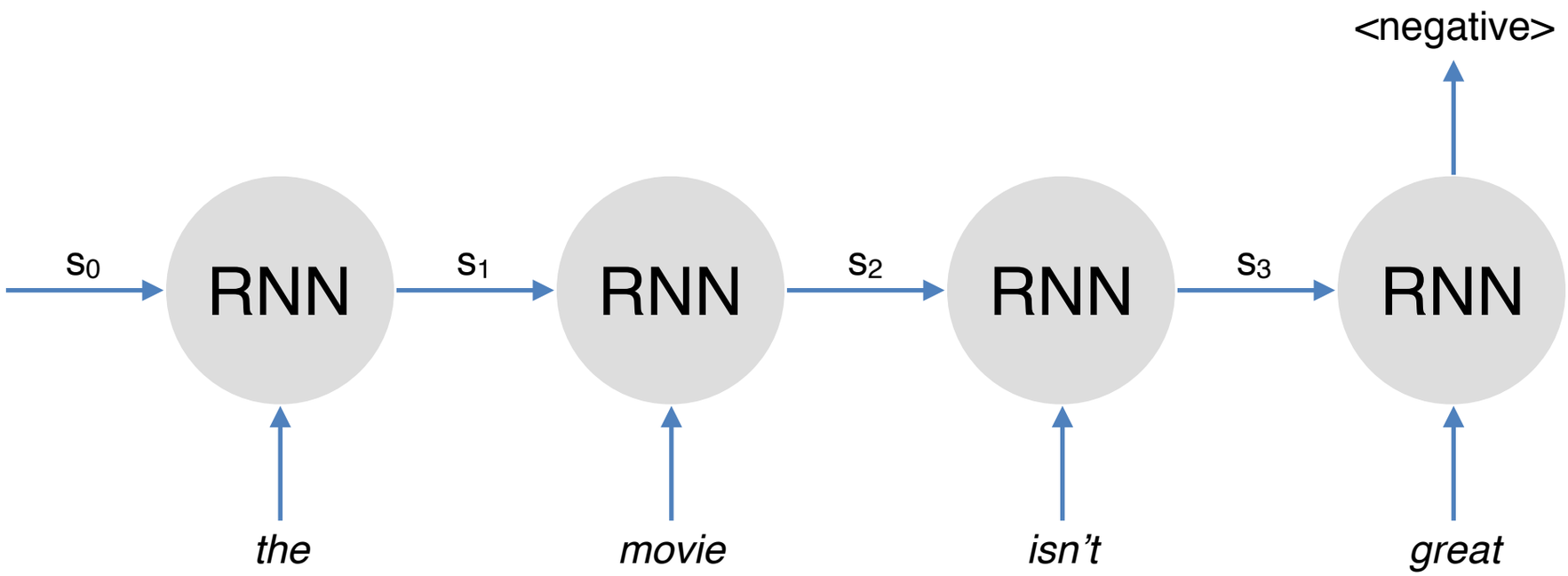
Researchers have revealed an AI that they hope will one day write a perfect Shakespearean sonnet.

Researchers at IBM used created a sonnet writing AI.

They trained it using 2600 sonnets taken from a free online database of out-of-copyright books called Project Gutenberg.

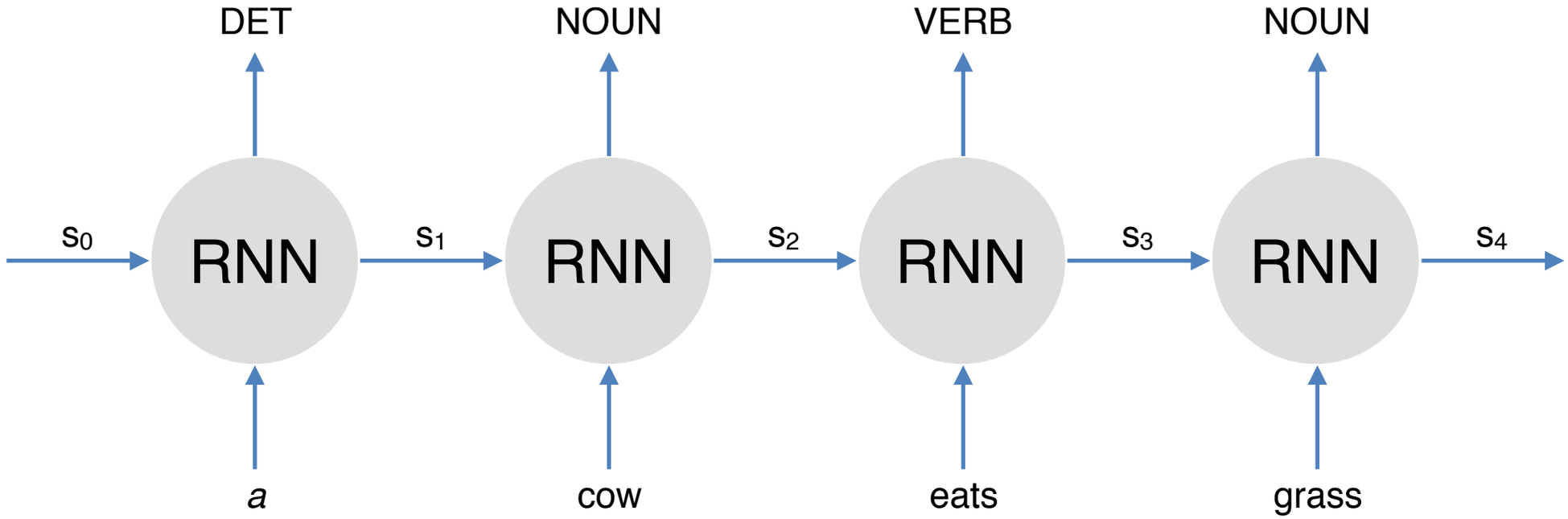
Text Classification

- Recurrent networks can be used in a variety of NLP tasks
- Particularly suited for tasks where order of words matter, e.g. sentiment classification



Sequence Labeling

- RNNs work particularly for sequence labelling problems, e.g. POS tagging



Final Words

- Pros
  - Has the ability to capture long range contexts
  - Excellent generalisation
  - Just like feedforward networks: flexible, so it can be used for all sorts of tasks
  - Common component in a number of NLP tasks
- Cons
  - Slower than feedforward networks due to sequential processing
  - In practice still doesn’t capture long range dependency very well (evident when generating long text)

Readings

- G15, Section 10 & 11