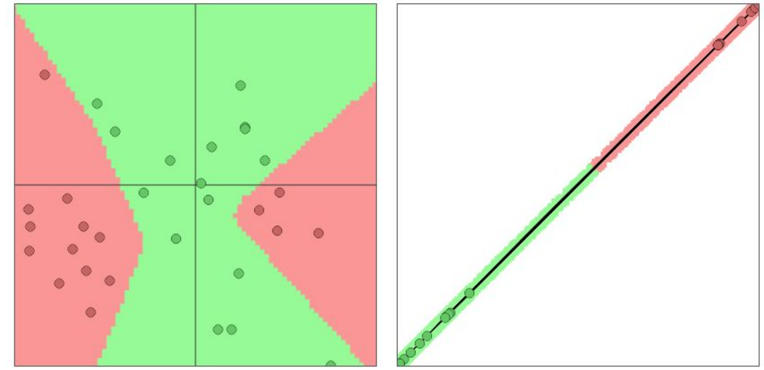
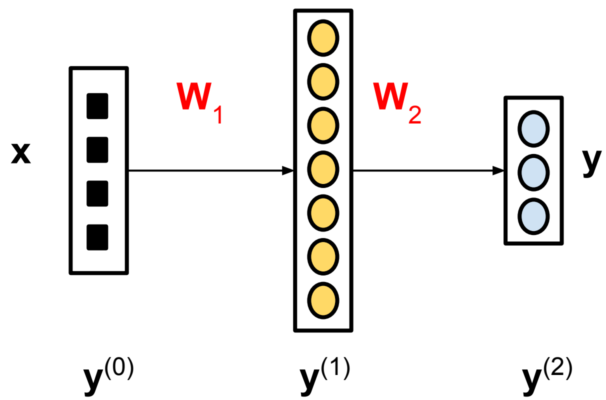
Representing & Processing Sequences

Viktor Schlegel

Recap: Artificial Neural Networks (ANNs)





Matrix multiplication

Non linear activation function

$$h_t = g(W x_t)$$

high-dimensional

Putting it all together

Distributional representation + neural networks + SGD + backpropagation = Deep Learning

Luckily, most of it is already implemented and ready to use with modern DL frameworks

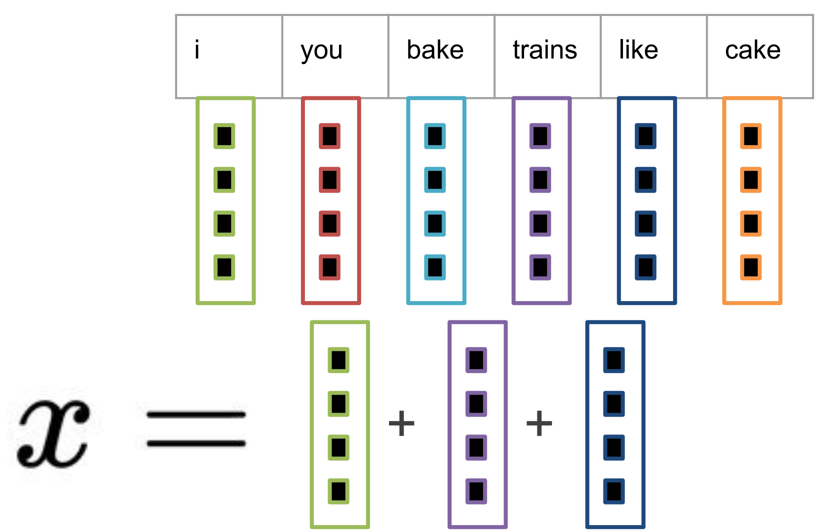
→ Torch, Tensorflow, and so on

From words to numbers

How do we obtain vector **x** as input to the neural network?

Naive solution: Bag of Words vector

$\mathbf{E}$



I like trains.
⇔ indices: [0,4,3]
⇔ BoW: v = [1,0,0,1,1,0]

$$\tilde{x} = \mathbf{E} \times v$$

There must be order!

Bag of Words representation loses the *order* of words.

Dog bites man.
Man bites dog.

Who bites whom?

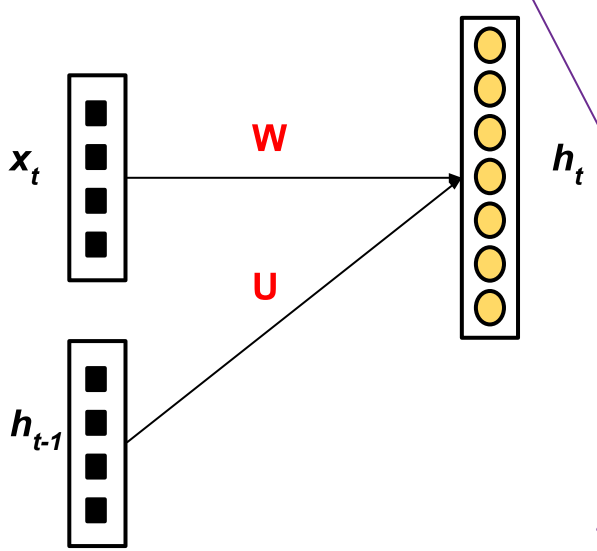
How to do… anything beyond sequence classification?

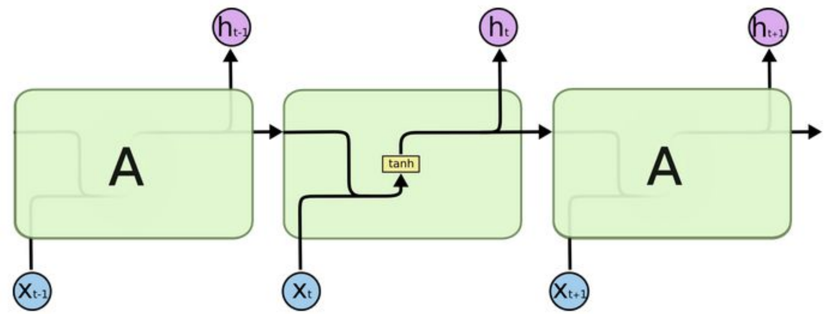
Recurrent neural networks

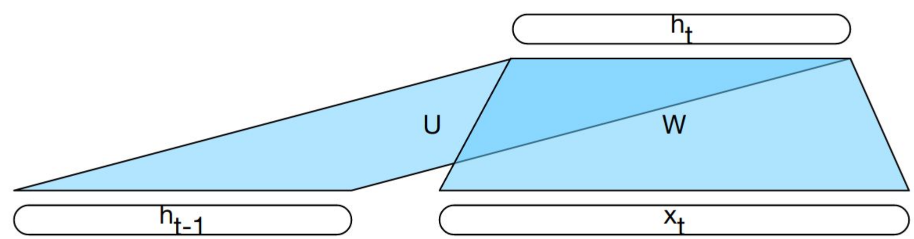
Sketch:

- Feed network token by token
- Use output of previous token as additional input when processing the current token

From NN to RNN





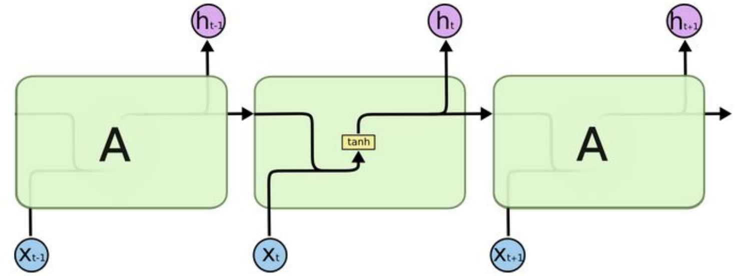


"recurrent"

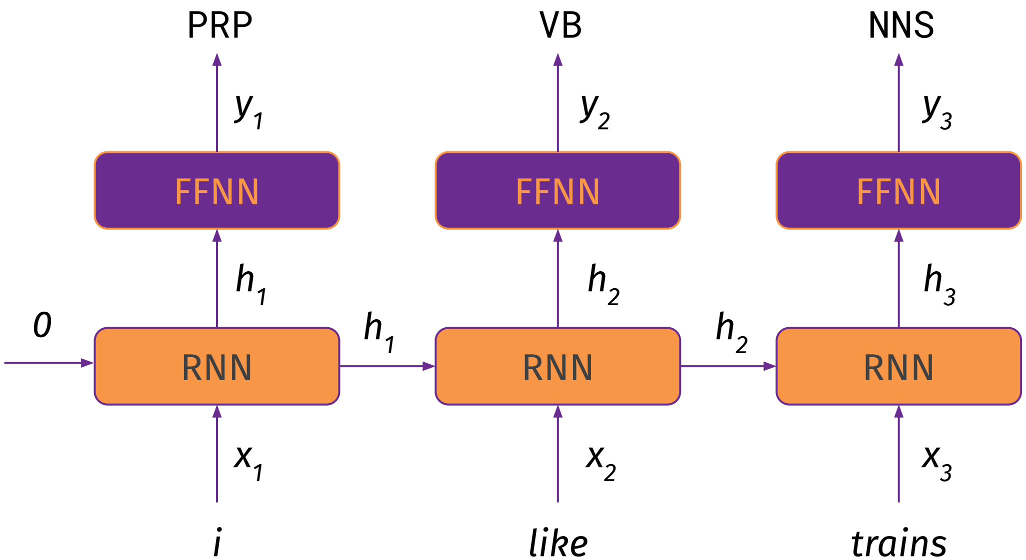
$$h_t = g(Wx_t)$$

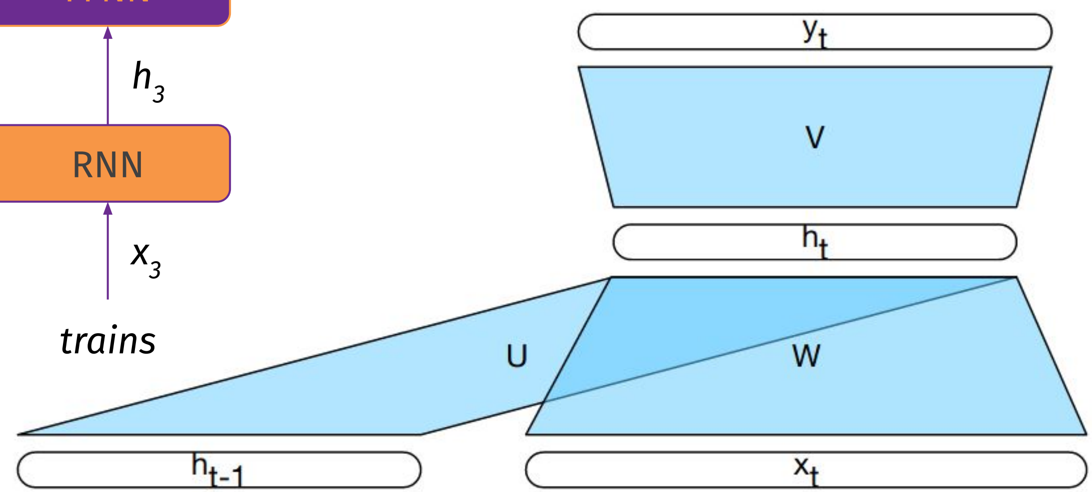
$$h_t = g(\boxed{Uh_{t-1}} + Wx_t)$$

The University of Manchester



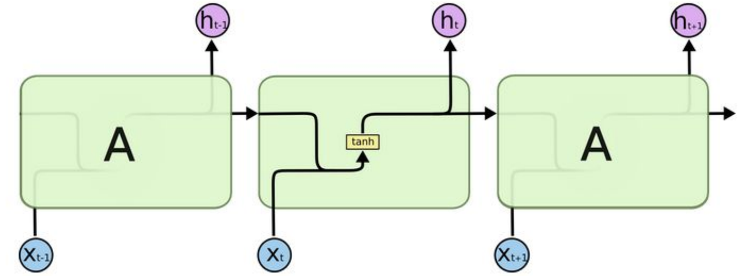
RNN: Forward run





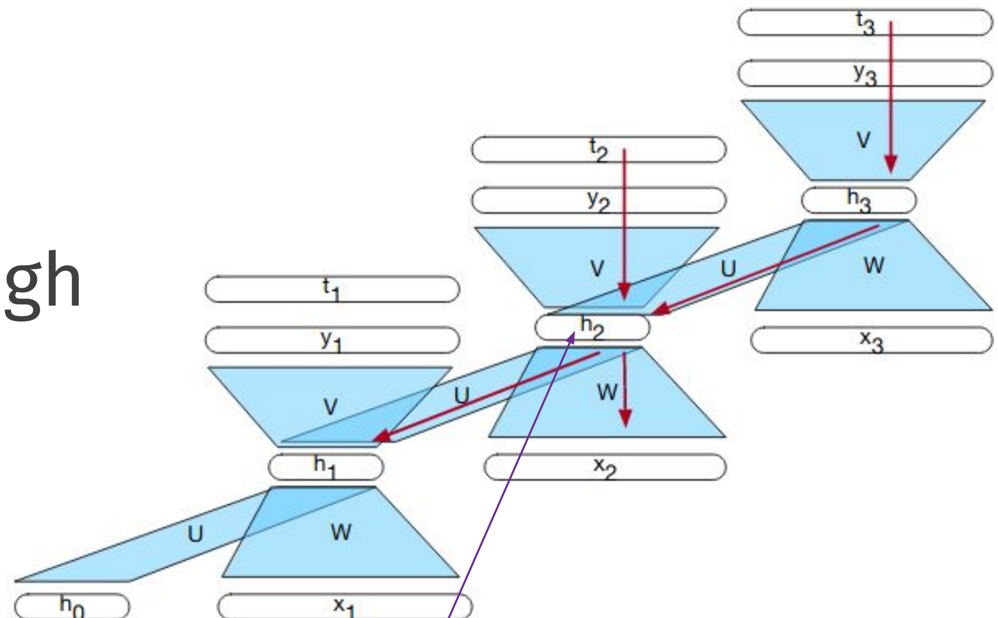
$$h_0 = 0$$
$$h_t = g(\boxed{U h_{t-1}} + W x_t + b)$$
$$y_t = f(V h_t + b_z)$$

No need to remember the math in detail…



RNN: Backward run

Backpropagation through time



$$\frac{\delta \mathcal{L}_{t+1}}{\delta U} = \frac{\delta \mathcal{L}_{t+1}}{\delta y_{t+1}} \frac{\delta y_{t+1}}{\boxed{\delta h_{t+1}}} \frac{\delta h_{t+1}}{\delta U}$$

$$\frac{\delta \mathcal{L}_{t+1}}{\delta U} = \frac{\delta \mathcal{L}_{t+1}}{\delta y_{t+1}} \frac{\delta y_{t+1}}{\delta h_{t+1}} \boxed{\frac{\delta h_{t+1}}{\delta h_t}} \frac{\delta h_t}{\delta U}$$

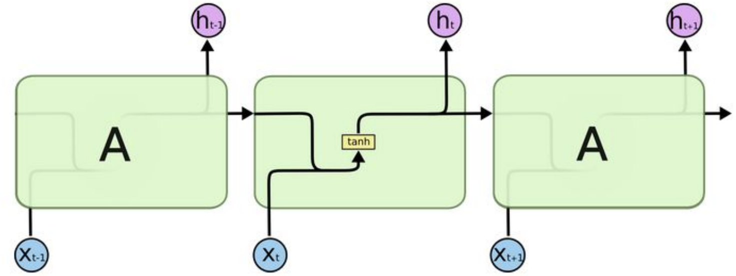
$$\frac{\delta \mathcal{L}}{\delta U} = \sum_t \boxed{\sum_{k=1}^{t+1}} \frac{\delta \mathcal{L}_{t+1}}{\delta y_{t+1}} \frac{\delta y_{t+1}}{\delta h_{t+1}} \boxed{\frac{\delta h_{t+1}}{\delta h_k}} \frac{\delta h_k}{\delta U}$$

$$\frac{\delta \mathcal{L}}{\delta W} = \sum_t \boxed{\sum_{k=1}^{t+1}} \frac{\delta \mathcal{L}_{t+1}}{\delta y_{t+1}} \frac{\delta y_{t+1}}{\delta h_{t+1}} \boxed{\frac{\delta h_{t+1}}{\delta h_k}} \frac{\delta h_k}{\delta W}$$

h_t depends on h_{t-1}
Unroll computation graph and do standard backpropagation, because t is not infinite.

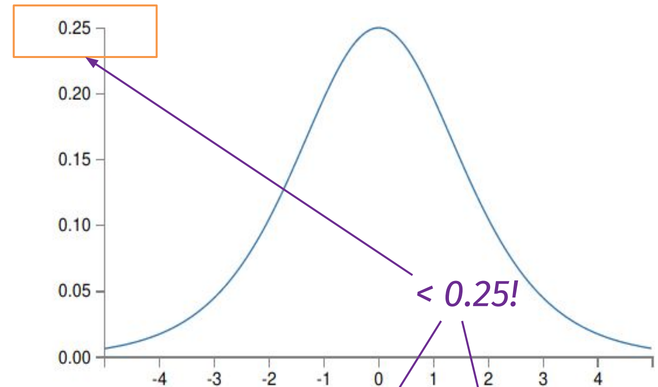
$$\frac{\delta \mathcal{L}}{\delta V} = \sum_t \frac{\delta \mathcal{L}}{\delta y_t} \frac{\delta y_t}{\delta V}$$

The University of Manchester



RNN: vanishing gradients

$$\frac{\delta \mathcal{L}}{\delta W} = \sum_t \sum_{k=1}^{t+1} \frac{\delta \mathcal{L}_{t+1}}{\delta y_{t+1}} \frac{\delta y_{t+1}}{\delta h_{t+1}} \boxed{\frac{\delta h_{t+1}}{\delta h_k}} \frac{\delta h_k}{\delta W}$$

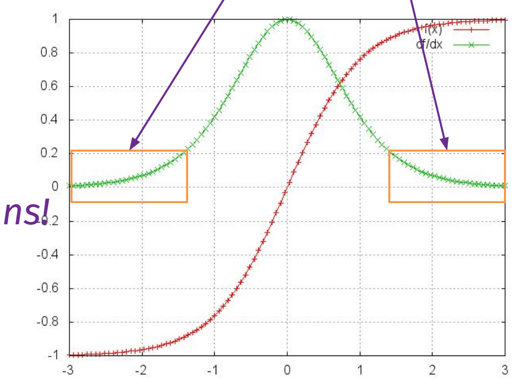


< 0.25!

$$g = \sigma, tanh \quad \frac{\delta \sigma}{\delta x}$$

$$\frac{\delta h_t}{\delta h_k} = \prod_{t \geq i > k} \frac{\delta h_i}{\delta h_{i-1}} = \prod_{t \geq i > k} \boxed{W}^T diag\left(\boxed{\frac{\delta g}{\delta h_{i-1}}}\right)$$

Lots of multiplications!



$$\frac{\delta tanh}{\delta x}$$

There once was a man from Peru
who dreamed he was eating his shoe.
Waking up in a fright
in the dark of the night
he found it was perfectly true.

The University of Manchester

LSTM

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

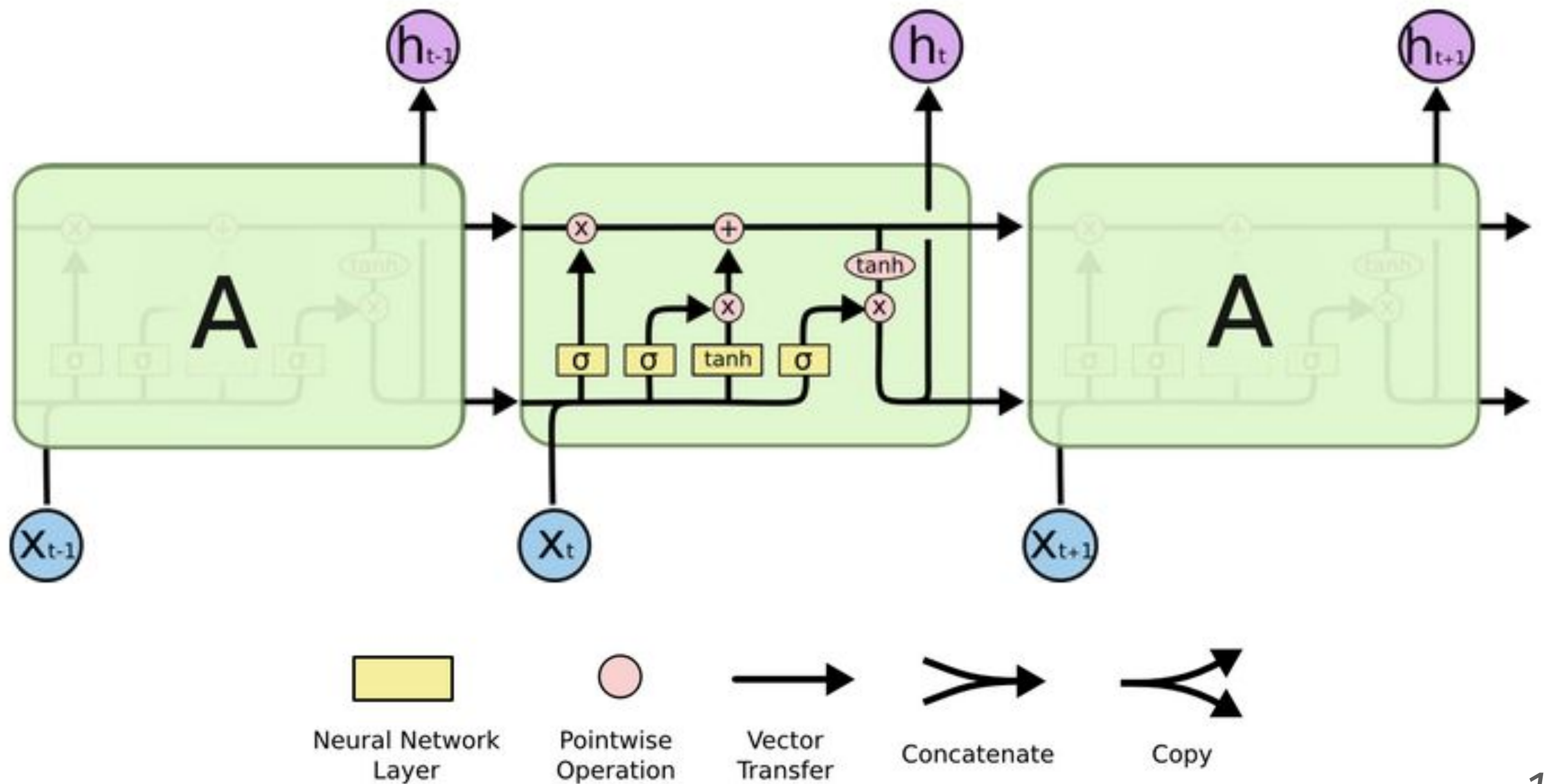
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

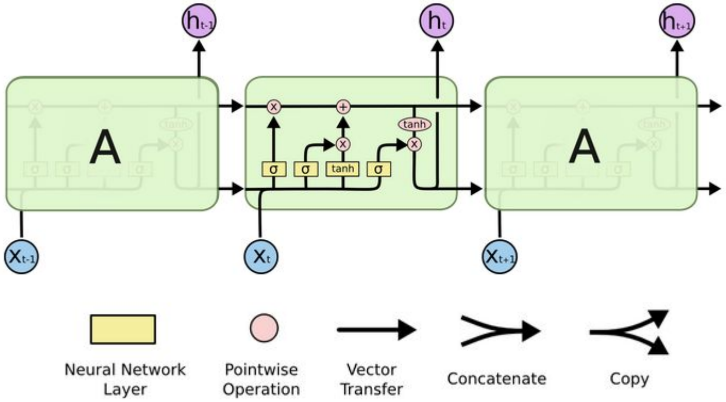
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

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

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

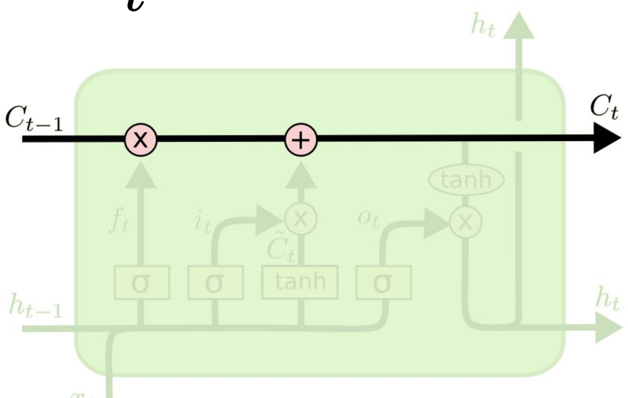
$$h_t = o_t * \tanh\left(C_t\right)$$





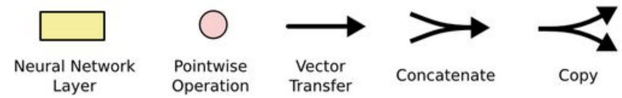
LSTM: Context vector

- Context, or memory vector C_t in addition to h_t
- Context information from “the past” calculations
- At any step t, LSTM learns how much of h_t is to be “added” to C_t and how much of C_{t-1} is “kept”

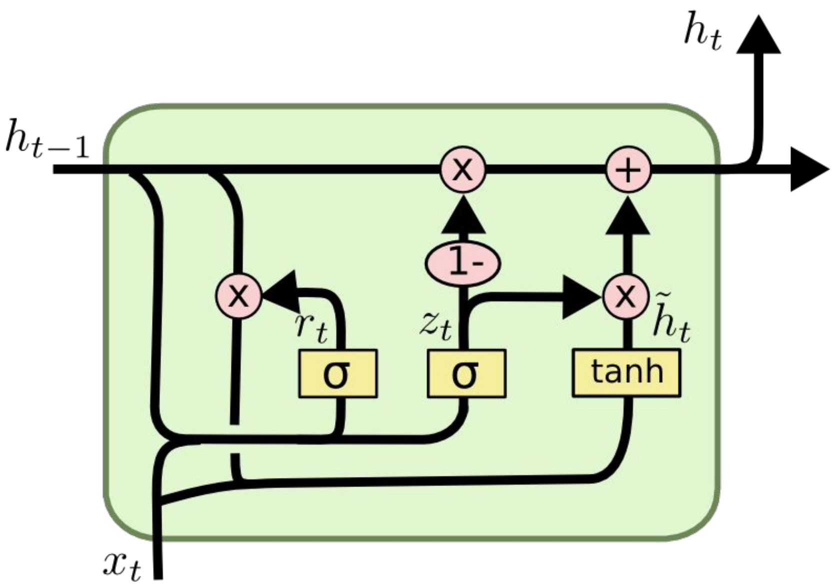


For example:
Information about grammatical gender of subject

GRU



- Similar idea but less parameters
- Similar performance to LSTM



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

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

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

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

RNN: Recap

What can we do with RNN now?

Have vector representation of words that incorporate information from “the past” (left)

From now on: RNN = RNN/LSTM/GRU

Forward References

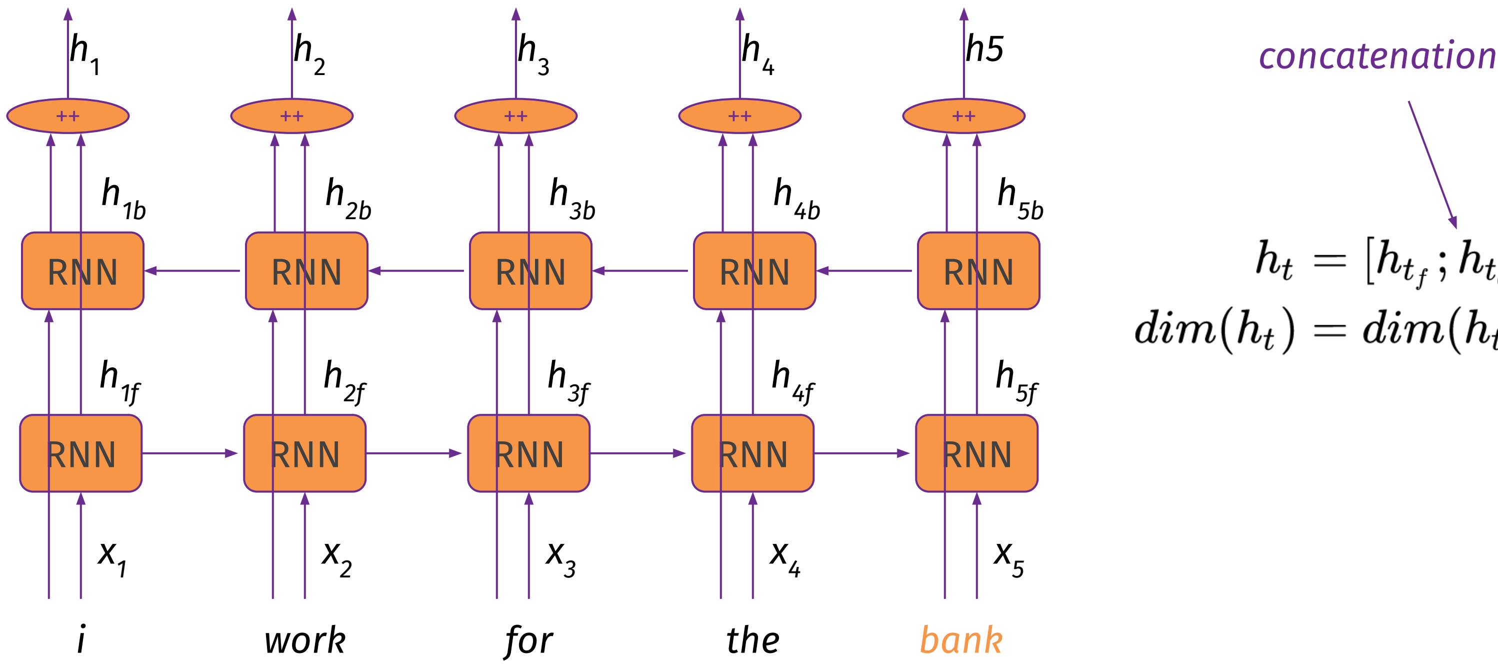
Does (and can) past history help us to get meaningful information about e.g. cataphora?

"In their free time, the kids listen to music".

in general: forward references

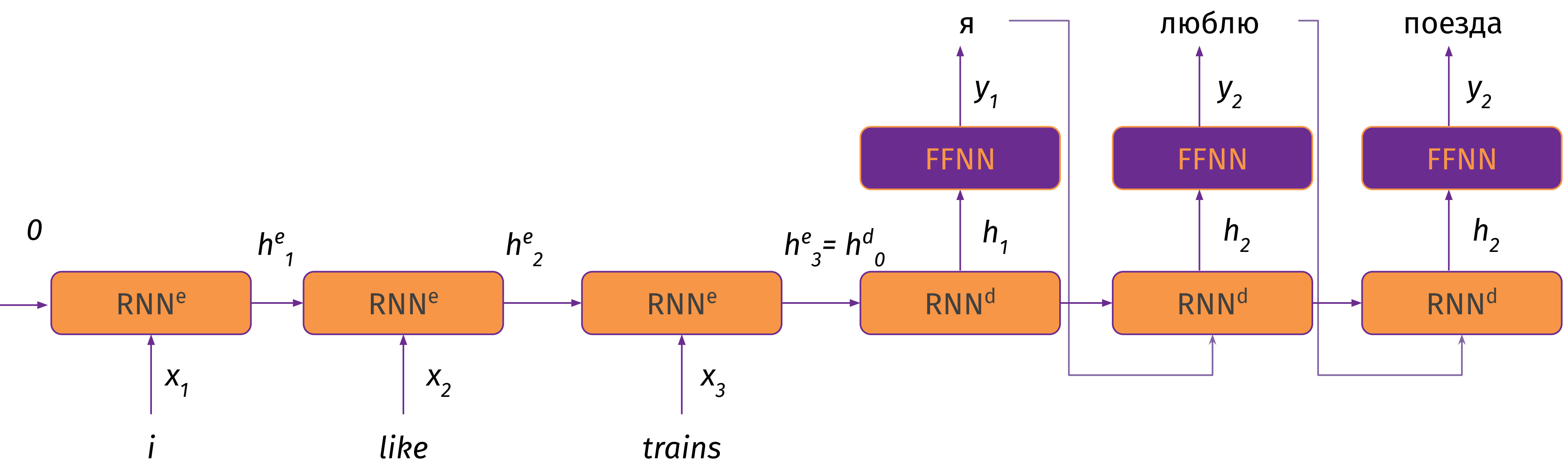
BiRNN

Idea: If we can go from left to right ("past"), why not also just go right to left ("future")



$$h_t = [h_{t_f}; h_{t_b}]$$
$$dim(h_t) = dim(h_{t_f}) + dim(h_{t_b})$$

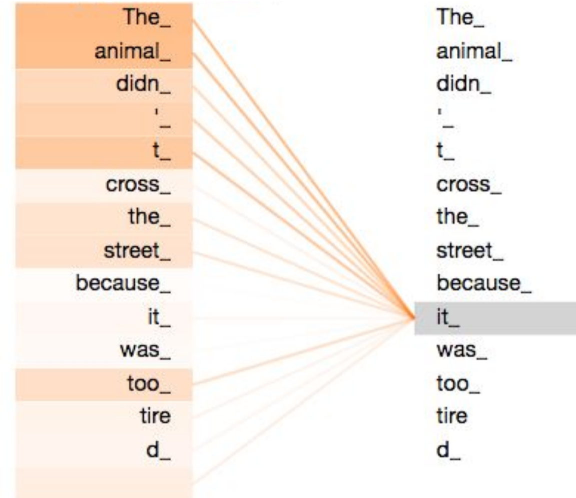
Encoder/Decoder



$$h_0^e = 0$$
$$h_t^e = g(U^e h_{t-1} + W^e x_t + b^e)$$

$$\boxed{h_0^d = h^e_{|T|}}$$
$$h_t^d = g(U^d h^d_{t-1} + W^d y_{t-1} + b^d)$$
$$y_t = f(V^d h_t^d + b_z^d)$$

The University of Manchester



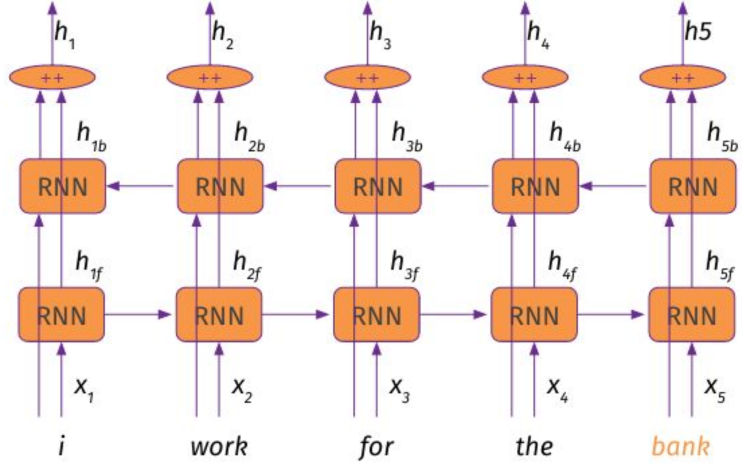
Self-attention

Alternative to RNN encoding/decoding

Intuition:

- For every token in input/output learn the relative "importance" of all tokens in input and output (that was decoded so far)
- Encode position to maintain order of words in sentence

Practice: a bit more complicated, multiple attention "heads", multiple layers, some "wizardry"



Recap: BiRNN

We can learn to represent a word in its forward and backward context of “arbitrary” length

Learning from task-specific dataset

~ 10-100K examples
Glove: 6B, 42B, 840B

Can we do better?

The University of Manchester

Computational graphs

Draw a computational graph of sorts, maybe of an NN or of a simple computation, depending on what makes more sense

The University of Manchester

Loss function

For example: cross entropy

(Stochastic) Gradient Descent

What the intuition behind it is

Backpropagation

From total error to errors in the nodes