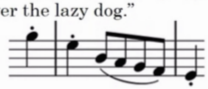
SEQUENCE MODELS

COURSE I

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

NOTATION

x: Harry Potter and Hermione Granger invented a new spell.

$x^{\langle 1\rangle}$ $x^{\langle 2\rangle}$ $x^{\langle 3\rangle}$ $x^{\langle 4\rangle}$ ... $x^{\langle 9\rangle}$

$T_x = 9$ → LENGTH OF INPUT SEQUENCE

y: 1 1 0 1 1 0 0 0 0

$y^{\langle 1\rangle}$ $y^{\langle 2\rangle}$ $y^{\langle 3\rangle}$ $y^{\langle 4\rangle}$ ... $y^{\langle 9\rangle}$

- $X^{(i)\langle t\rangle}$ → t^{th} element in the input sequence of training example i.
- T_x → length of a sequence
- $T_x^{(i)}$ → input sequence length for training example i.
- $y^{(i)\langle t\rangle}$ → t^{th} element in the output sequence of training example i.
- $T_y^{(i)}$ → output sequence length for training example i.

* for representing words you first need to come up with an array of all words that will be used in your model.

x: Harry Potter and Hermione Granger invented a new spell.

$x^{\langle 1\rangle}$ $x^{\langle 2\rangle}$

vocabulary = [a, aaron, ..., and, ..., harry, ..., zulu] → [1, 2, ..., 367, ..., 4,075, ..., 10,000]

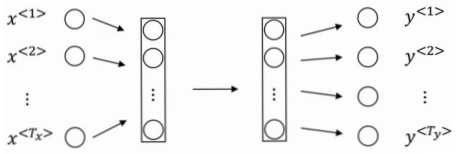
ONE-HOT ENCODED VECTOR

$x^{\langle 1\rangle}$: [0, 0, ..., 0, 1, 0] ← 4,075

$x^{\langle 2\rangle}$: [0, 0, ..., 1, 0, ..., 0] ← 367

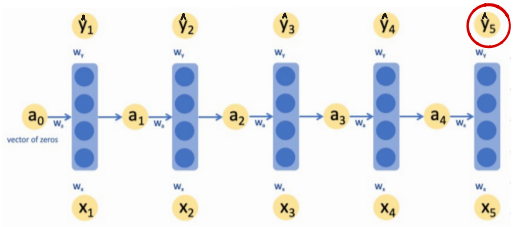
RECURRENT NEURAL NETWORK

* why not a standard (NN)?



(#1). inputs and outputs can have (≠) lengths in (≠) examples.
(#2). doesn't share features learned across (≠) positions in the text.

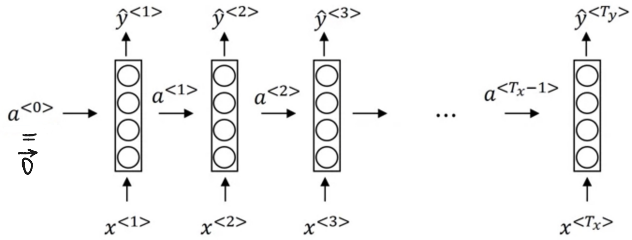
* RNN architecture:



WE MAKE THE PREDICTION FOR $\hat{y}^{<5>}$ NOT ONLY FROM THE INFO OF $x^{<5>}$ BUT ALSO FROM THE INFO OF $x^{<1>}$, $x^{<2>}$, $x^{<3>}$, AND $x^{<4>}$.

* weakness of this (NN) ⟶ it only uses past info to make a prediction.

* forward propagation:



$a^{<1>} = g(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$ ⟶ activation functions: tanh or ReLU

$\hat{y}^{<1>} = g(W_{ya}\, a^{<1>} + b_y)$ ⟶ sigmoid (binary)
softmax (multiclass)

$$a^{<t>} = g(W_{aa}\, a^{<t-1>} + W_{ax}\, x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g(W_{ya}\, a^{<t-1>} + b_y)$$

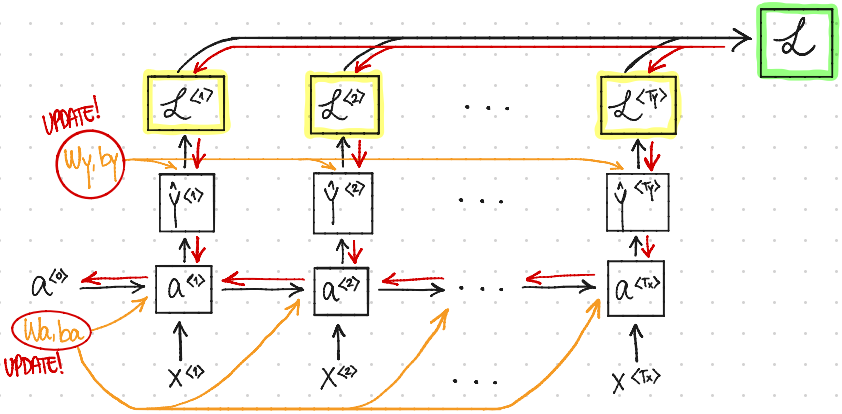
$$a^{\langle t \rangle} = g\left(W_a \left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_a\right)$$

W_a $(100 \times 10{,}100)$ $\rightarrow$ $\left[W_{aa}\ (100 \times 100) \mid W_{ax}\ (100 \times 10{,}000)\right]$

$\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] = \begin{bmatrix} a^{\langle t-1 \rangle} \\ x^{\langle t \rangle} \end{bmatrix}$ (100 + 10,000 = 10,100)

$$\left[W_{aa} \mid W_{ax}\right] \begin{bmatrix} a^{\langle t-1 \rangle} \\ x^{\langle t \rangle} \end{bmatrix} = W_{aa}\, a^{\langle t-1 \rangle} + W_{ax}\, x^{\langle t \rangle}$$

FORWARD AND BACKPROP THROUGH TIME



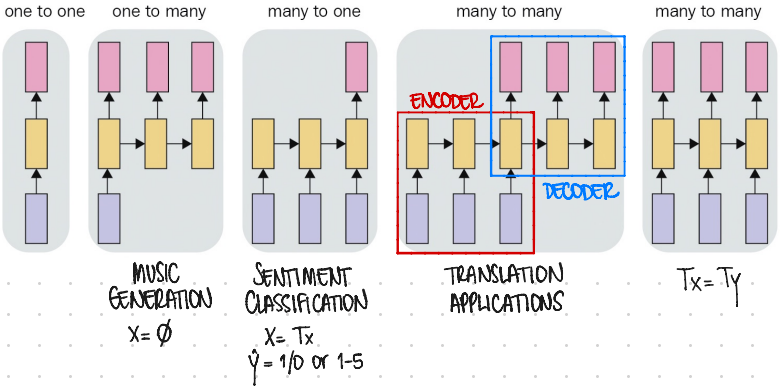
$$\mathcal{L}^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - \left(1 - y^{\langle t \rangle}\right) \log\left(1 - \hat{y}^{\langle t \rangle}\right)$$

LOSS OF A SINGLE PREDICTION

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right)$$

LOSS OF ENTIRE SEQUENCE

DIFFERENT TYPES OF RNNs



LANGUAGE MODEL AND SEQUENCE GENERATION

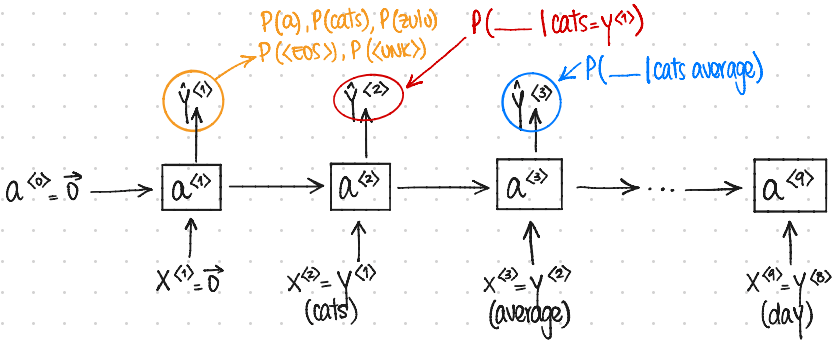
what a language model does is it tells you what the probability of a sentence is.

it inputs a sentence and estimates its probability

$P(y^{<1>}, y^{<2>}, \dots, y^{<T_y>})$

#1 get a large corpus of text (training set)
#2 tokenize the text
#3 map each token to a one-hot vector
#4 build the RNN model

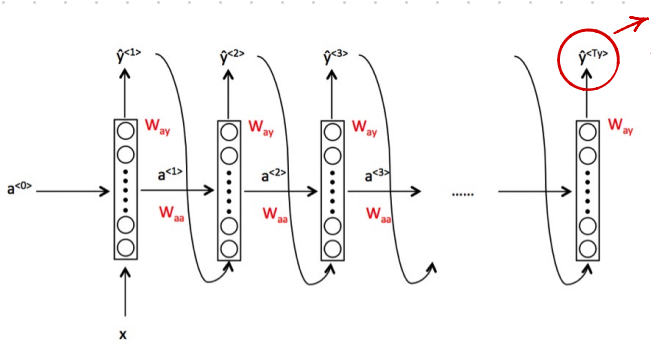
Cats average 15 hours of sleep a day.



VANISHING GRADIENTS WITH RNNS

* language can have very long-term dependencies, where a word can affect what needs to come much later in a sentence.

> The cat, which already ate ..., was full.
> The cats, which already ate ..., were full.



* it is difficult to get the (RNN) to realize that it needs to memorize or to just see a singular noun or plural noun, and then later in the sequence generate the corresponding verb form that goes with the noun it previously saw.

GATED RECURRENT UNIT (GRU)

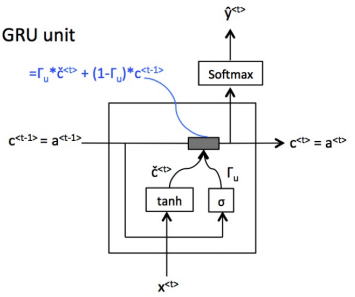
* it has a unit (c) called memory cell
  - it remembers whether "cat" is singular or plural.
  - $c^{\langle t \rangle} = a^{\langle t \rangle}$ → $c^{\langle t \rangle}$ will output an activation value
  - at every time step, we will consider over-writing the memory cell with a value $\tilde{c}^{\langle t \rangle}$.
  - $\tilde{c}^{\langle t \rangle} = \tanh(w_c[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$ → $\tilde{c}^{\langle t \rangle}$ will be computed using a tanh activation function
  - $\Gamma_u = \sigma(w_u[c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$ → gate of the GRU (it's either 0 or 1)
    - u = UPDATE

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

The Cat, which already ate .... , was full.

$c^{\langle t \rangle}=1$, $\Gamma_u=1$ $\quad \Gamma_u=0 \quad \Gamma_u=0 \quad \Gamma_u=0$

you memorize "cat" was singular



* $c^{\langle t \rangle}$, $\tilde{c}^{\langle t \rangle}$, Γ_u will all have the same dimension

* for a full GRU you add Γ_r which tells you how relevant $c^{\langle t-1 \rangle}$ is to compute the next candidate for $c^{\langle t \rangle}$.

$$\tilde{c}^{\langle t \rangle} = \tanh\left(w_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$
$$\Gamma_u = \sigma\left(w_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$
$$\Gamma_r = \sigma\left(w_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$$
$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) + c^{\langle t-1 \rangle}$$

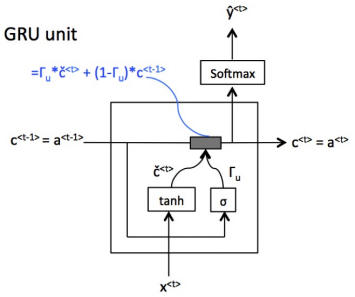
FULL GRU

LONG SHORT TERM MEMORY

* more powerful than GRU, and more general, too.

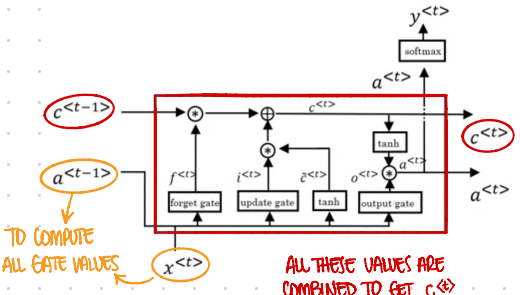
GRU

$$\tilde{c}^{\langle t \rangle} = \tanh\left(w_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$
$$\Gamma_u = \sigma\left(w_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$
$$\Gamma_r = \sigma\left(w_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$$
$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) + c^{\langle t-1 \rangle}$$
$$a^{\langle t \rangle} = c^{\langle t \rangle}$$



LSTM

$$\tilde{c}^{\langle t \rangle} = \tanh\left(w_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$
$$\Gamma_u = \sigma\left(w_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right) \quad \text{UPDATE GATE}$$
$$\Gamma_f = \sigma\left(w_f\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right) \quad \text{FORGET GATE}$$
$$\Gamma_o = \sigma\left(w_o\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right) \quad \text{OUTPUT GATE}$$
$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$$
$$a^{\langle t \rangle} = \Gamma_o * \tanh\left(c^{\langle t \rangle}\right)$$

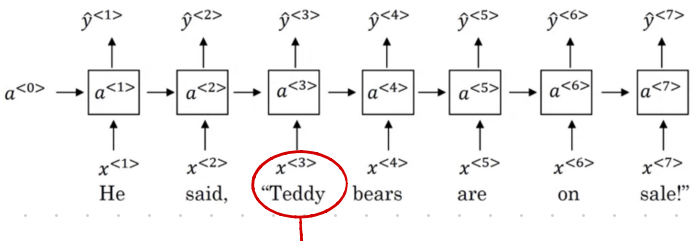


* when to use each?
  - GRUs are simpler and easier to build a much bigger (NN). Computationally, they even run faster.
  - LSTMs are more powerful and flexible. Historically, more proven choice.

BIDIRECTIONAL RNN

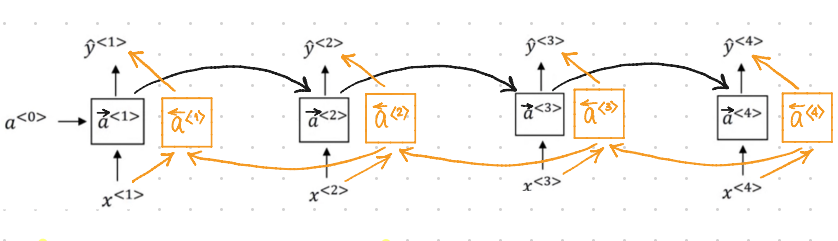
He said, "Teddy bears are on sale!"

He said, "Teddy Roosevelt was a great President!"



REGULAR RNN

the first three words don't tell you if you are talking about teddy bears or Roosevelt.

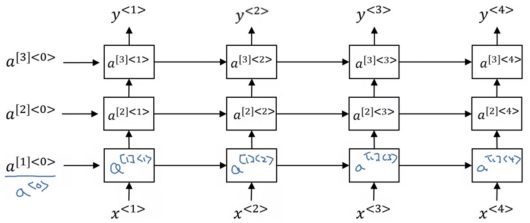


BIDIRECTIONAL RNN (acyclic graph)

$$\hat{y}^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

* disadvantage → you need the full sequence of data before you start making predictions.

DEEP RNNs



$$a^{[2]<3>} = g(W_a^{[2]}[a^{[2]<2>}, a^{[1]<3>}] + b_a^{[2]})$$

WORD REPRESENTATION

* a way of representing words.

* we represent words through a vocabulary vector $|V| = 10{,}000$. And generate a one-hot representation for each of the words in $|V|$.

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮
0	1	0	1	0	1
0	0	0	0	0	0

* weakness of this representation: it treats each word as a thing unto itself, and it doesn't allow an algorithm to generalize the cross-words.

APPLE AND ORANGE ARE MUCH MORE SIMILAR AND COULD GO WELL WITH OTHER WORDS, LIKE "JUICE"

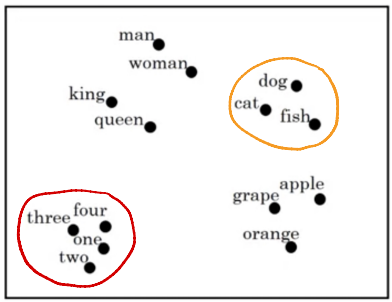
* to solve that issue we use featurized representation where we can learn some features from each of them.

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
GENDER	-1	+1	-0.95	0.97	0.00	0.01
ROYAL	0.01	0.02	0.93	0.95	-0.01	0.00
AGE	0.03	0.02	0.7	0.69	0.03	-0.02
FOOD	0.04	0.01	0.02	0.01	0.95	0.97

300 FEATURES = 300-D VECTOR

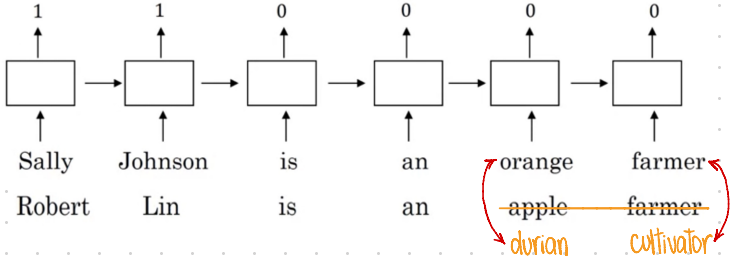
e_{456} — THE 300-D VECTOR

* visualizing word embeddings → you take the 300-D vector and visualize it in a 2-D space using t-SNE.



USING WORD EMBEDDINGS

* named entity recognition example:



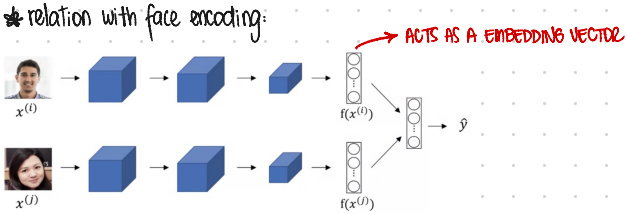
* because of word embedding your algorithm can learn that both apple and durian are fruits, and that farmer and cultivator are similar, too.

(#1). Learn word embeddings from a large text corpus (1 - 10 B words)

(#2). Transfer embedding to new task with smaller training set (100k words)

(#3) Optional: continue to finetune the word embeddings with new data.

* relation with face encoding:



PROPERTIES OF WORD EMBEDDINGS

* they can also help with analogy reasoning

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

e_{man} e_{woman}

* can we define an algorithm that understands that man → woman
king → queen ?