

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



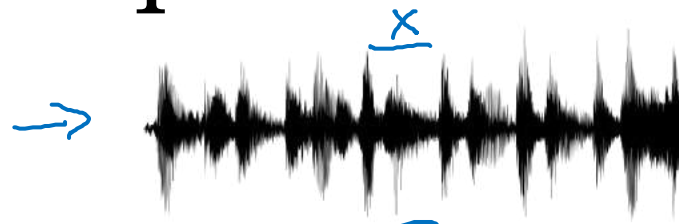
deeplearning.ai

Recurrent Neural Networks

Why sequence
models?

Examples of sequence data

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



AG**CCCCTGTGAGGAACTAG**

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**.

Andrew Ng



deeplearning.ai

Recurrent Neural Networks

Notation

Motivating example

NLP

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

$\rightarrow x^{(1)} \quad x^{(2)} \quad x^{(3)} \quad \dots \quad x^{(t)} \quad \dots \quad x^{(9)}$

$$T_x = 9$$

$\rightarrow y:$

$y^{(1)} \quad y^{(2)} \quad y^{(3)} \quad \dots \quad y^{(9)}$

Stylized example - where
0/1 represent name or not
for each word

$$T_y = 9$$

$x^{(i)(t)}$

$$T_x^{(i)} = 9$$

15

$y^{(i)(t)}$
 \uparrow

$T_y^{(i)}$

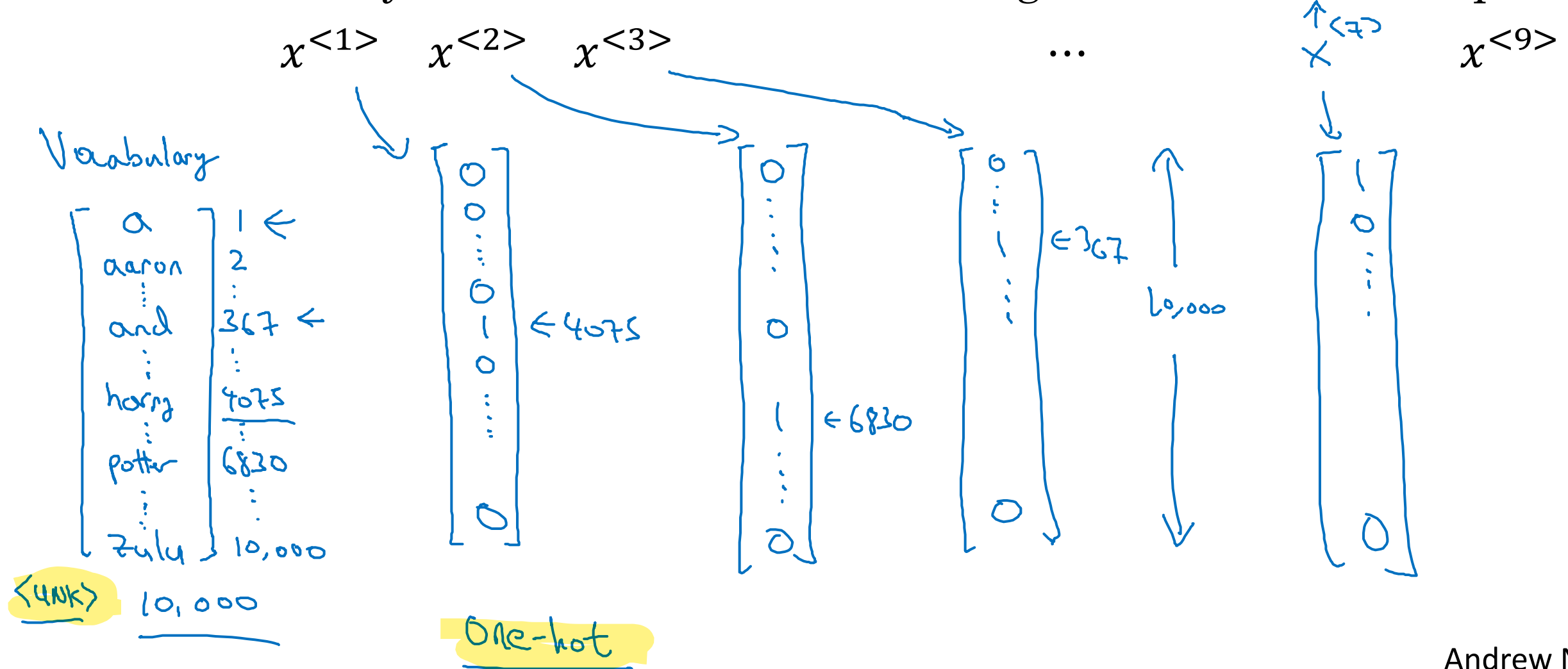
T = length

Representing words

Commercial applications may use dictionary of size 30-50k,
some internet companies may use 1mn+ dictionaries

$x^{(t)}$ (x,y)
 $x \rightarrow y$

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



Representing words

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

$$x^{<1>} \quad x^{<2>} \quad x^{<3>} \quad \dots \quad x^{<9>}$$

And = 367

Invented = 4700

$$A = 1$$

New = 5976

Spell = 8376

Harry = 4075

Potter = 6830

Hermione = 4200

Gran... = 4000

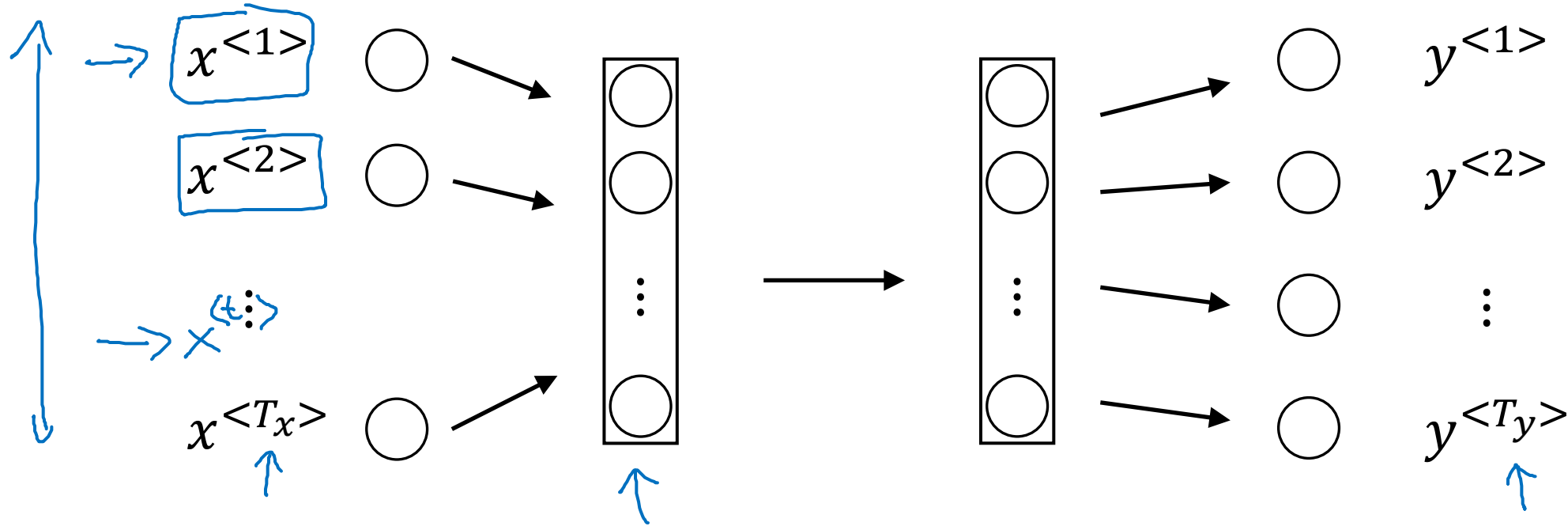


deeplearning.ai

Recurrent Neural Networks

Recurrent Neural Network Model

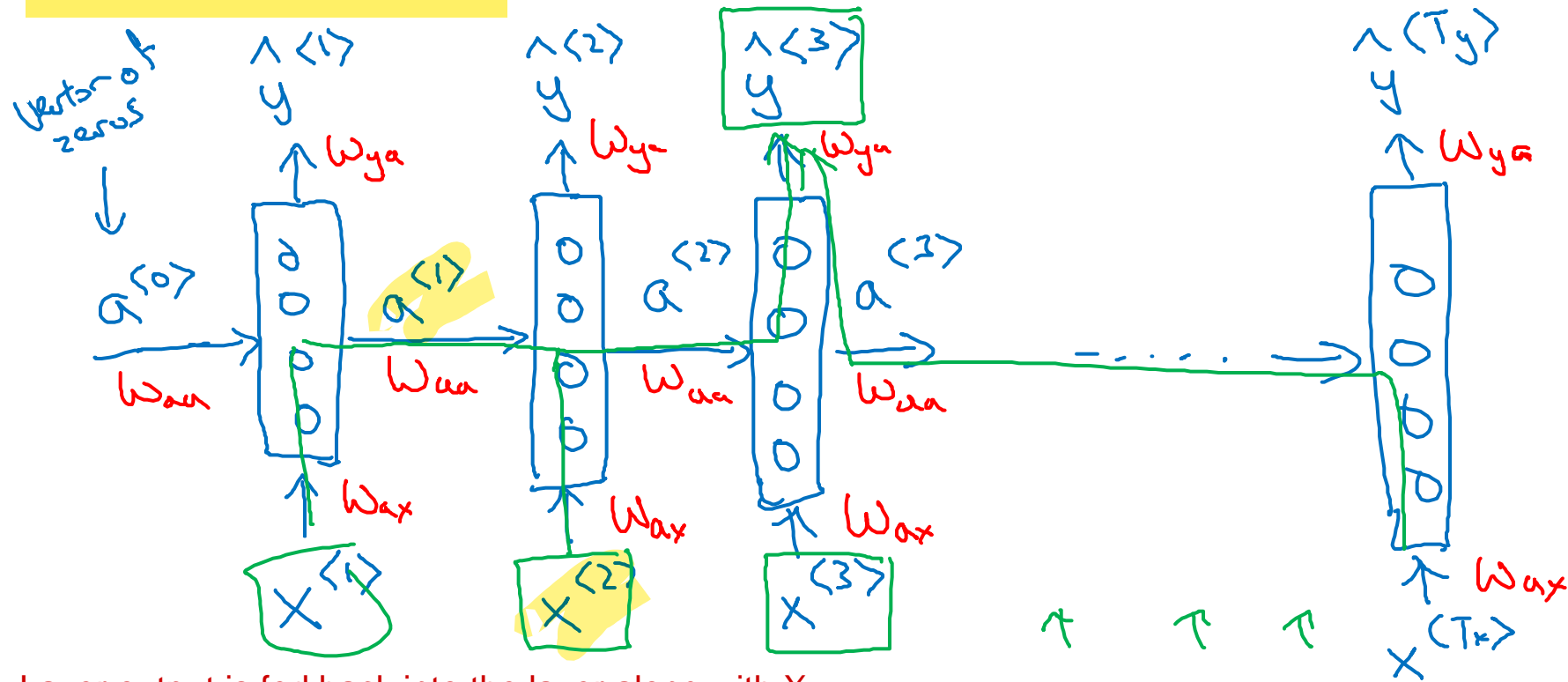
Why not a standard network?



Problems:

- - Inputs, outputs can be **different lengths** in different examples. can be managed through padding
- - **Doesn't share features learned across different positions of text.**

Recurrent Neural Networks



$$T_x = T_y$$

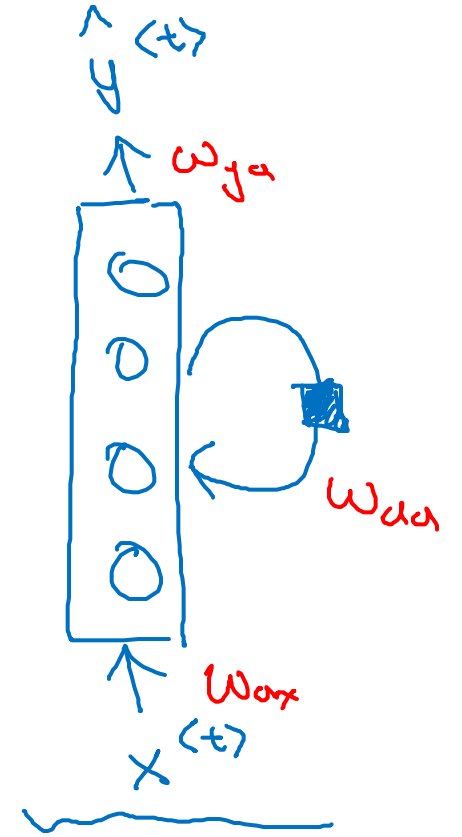
Layer output is fed back into the layer along with x

W_x and W_a parameters are shared

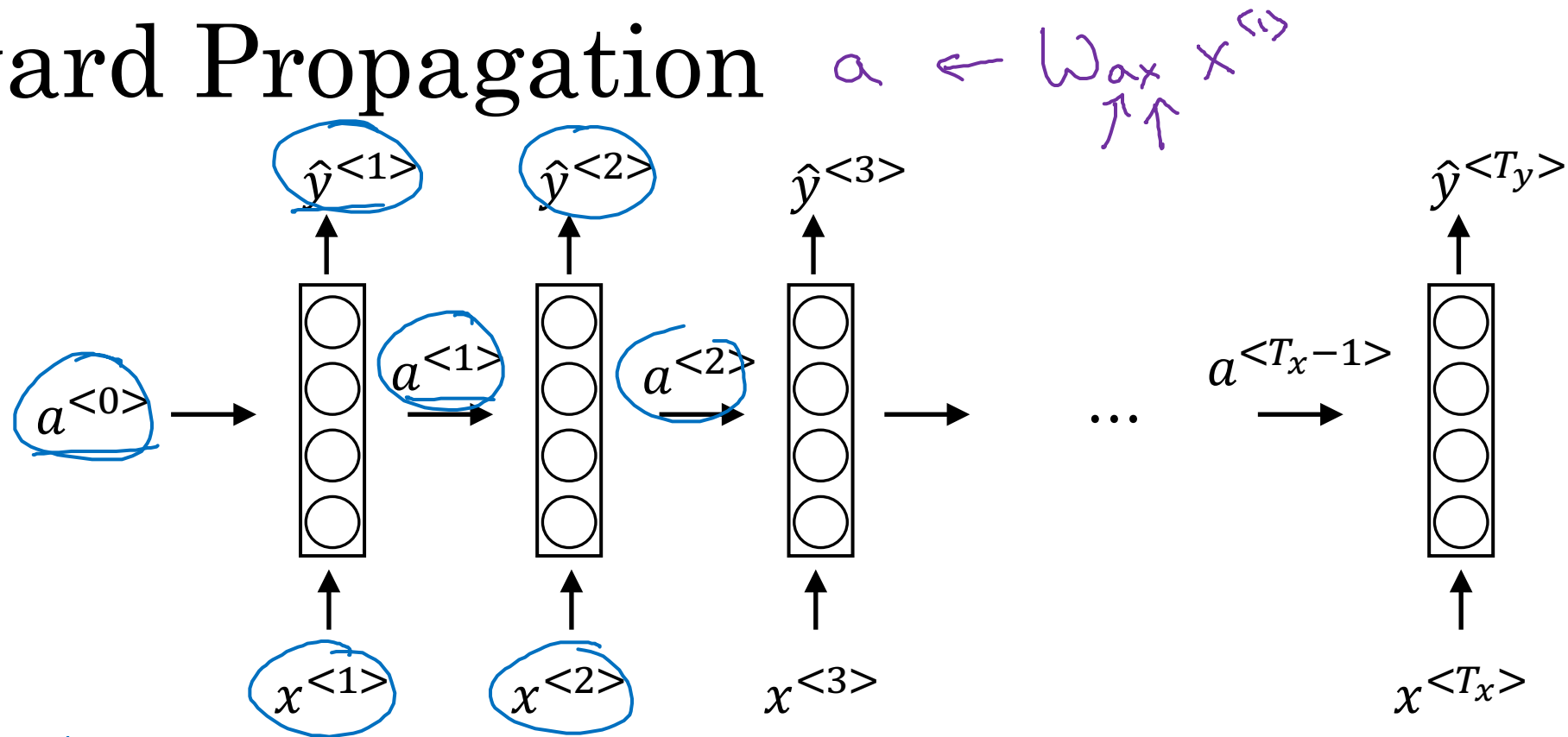
Weakness: only prior words can be used not following words

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

He said, "Teddy bears are on sale!"



Forward Propagation



$$a^{<0>} = \vec{0}$$

$$\underline{a}^{<1>} = g_1(W_{aa}a^{<0>} + \underline{W_{ax}}x^{<1>} + b_a) \leftarrow \text{tanh / ReLU}$$

$$\underline{\hat{y}}^{<1>} = g_2(W_{ya}\underline{a}^{<1>} + b_y) \leftarrow \text{Sigmoid}$$

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

Note: 2 different activation functions are used
 - usually, a and y occur in different layers of a NN
 - but here same layer provides both a and y , therefore uses both activations simultaneously

Simplified RNN notation

parameters for W_{aa} and W_{ax} can be combined into one large W_a matrix

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

Annotations: W_{aa} is 100×100 (labeled $(100, 100)$), W_{ax} is $100 \times 10,000$ (labeled $(100, 10,000)$).

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

$$y^{<t>} = g(W_y a^{<t>} + b_y)$$

$$a^{<t>} = g(W_a [a^{<t-1>}, x^{<t>}] + b_a)$$

$$\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} = W_a$$

Dimensions: 100×100 and $100 \times 10,000$ combine to 100×10100 (labeled $(100, 10100)$).

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$$

Dimensions: 100 for $a^{<t-1>}$, 10000 for $x^{<t>}$, total 10100 .

$$\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$$

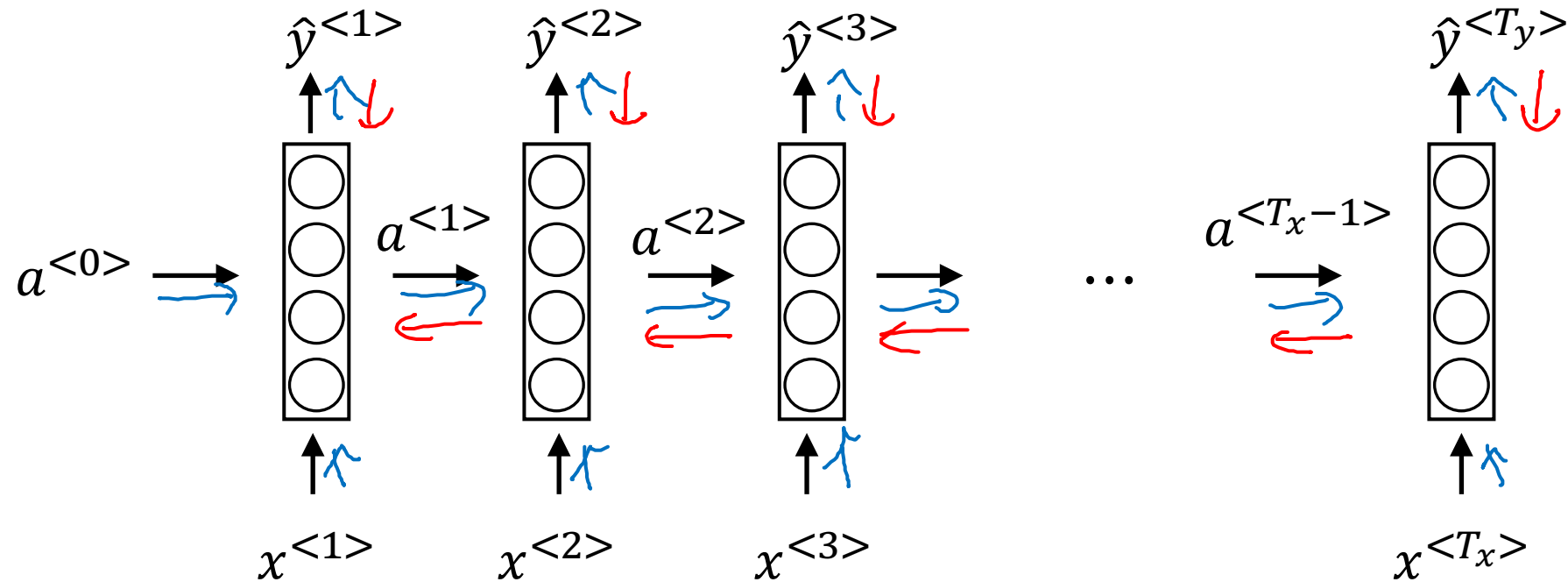


deeplearning.ai

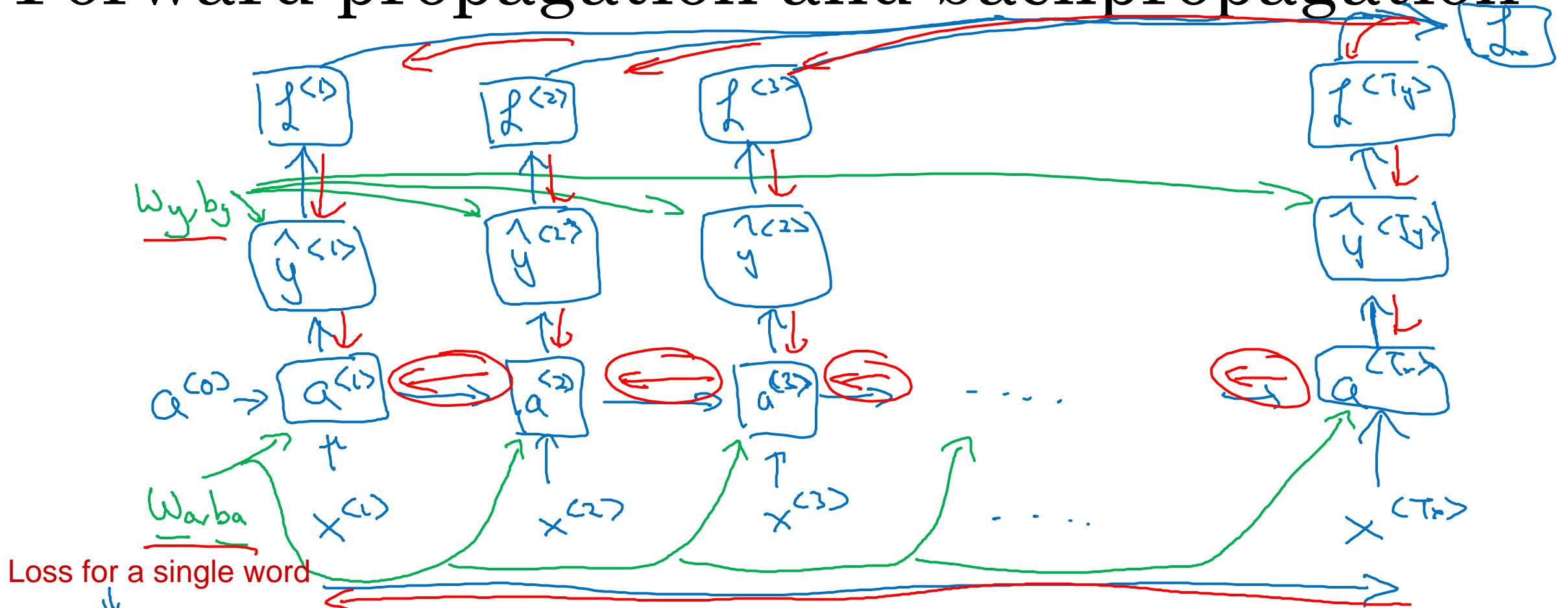
Recurrent Neural Networks

Backpropagation
through time

Forward propagation and backpropagation



Forward propagation and backpropagation



Loss for a single word

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

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

Backpropagation through time

Sum of losses from each step or each word



deeplearning.ai

Recurrent Neural Networks

Different types of RNNs

$$T_x \quad T_y \quad y$$

Input length = 0 or 1 (\emptyset)

“There is nothing to like
in this movie.”

AGCCCCTGTGAGGAAGTAG

Voulez-vous chanter avec moi?



Yesterday, Harry Potter
met Hermione Granger.

Output length 1 integer

AGCCCCTGTGAGGAAGTAG

Do you want to sing with
me?

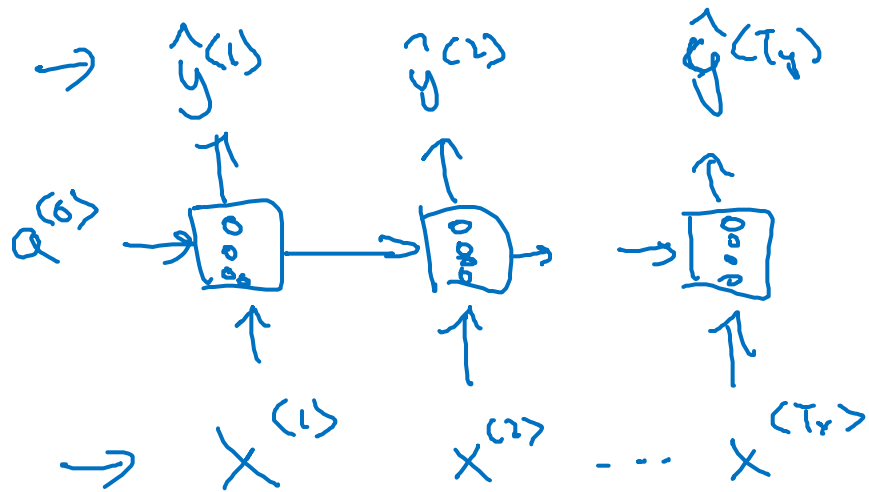
Input length \neq output length

Running

Yesterday, Harry Potter
met Hermione Granger.

Examples of RNN architectures

$$T_x = T_y$$

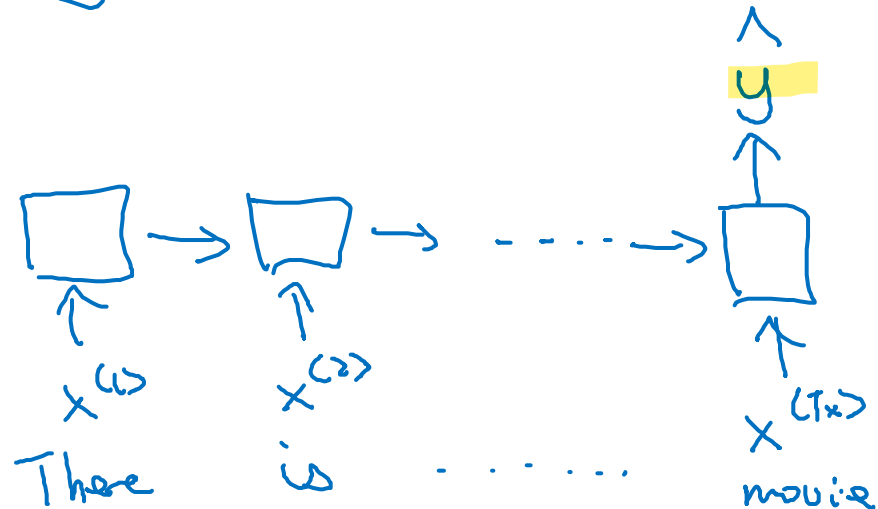


Many-to-many

Sentiment classification

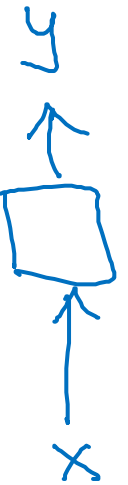
$x = \text{text}$

$y = 0/1 \quad 1 \dots 5$



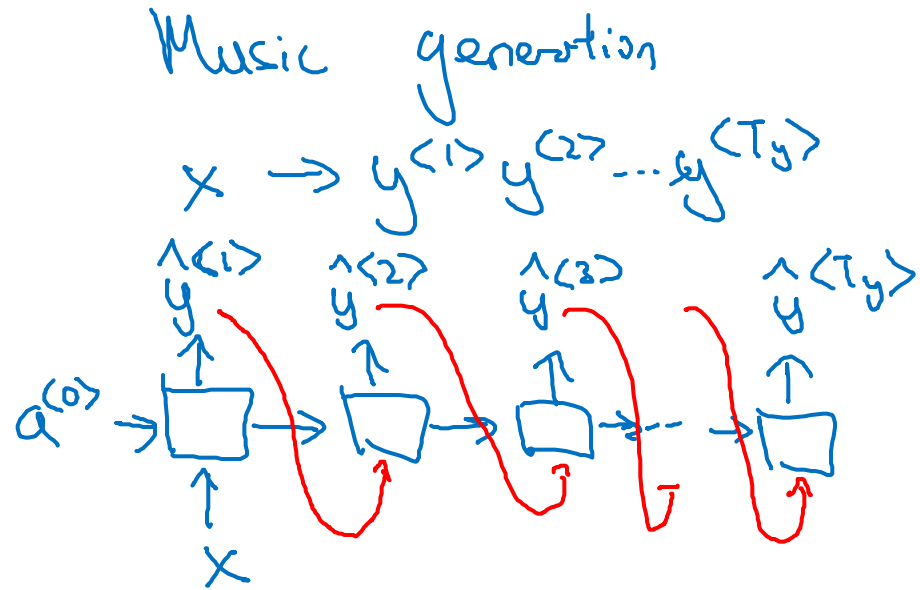
Many-to-one

Trivial /
earlier
courses



One-to-one

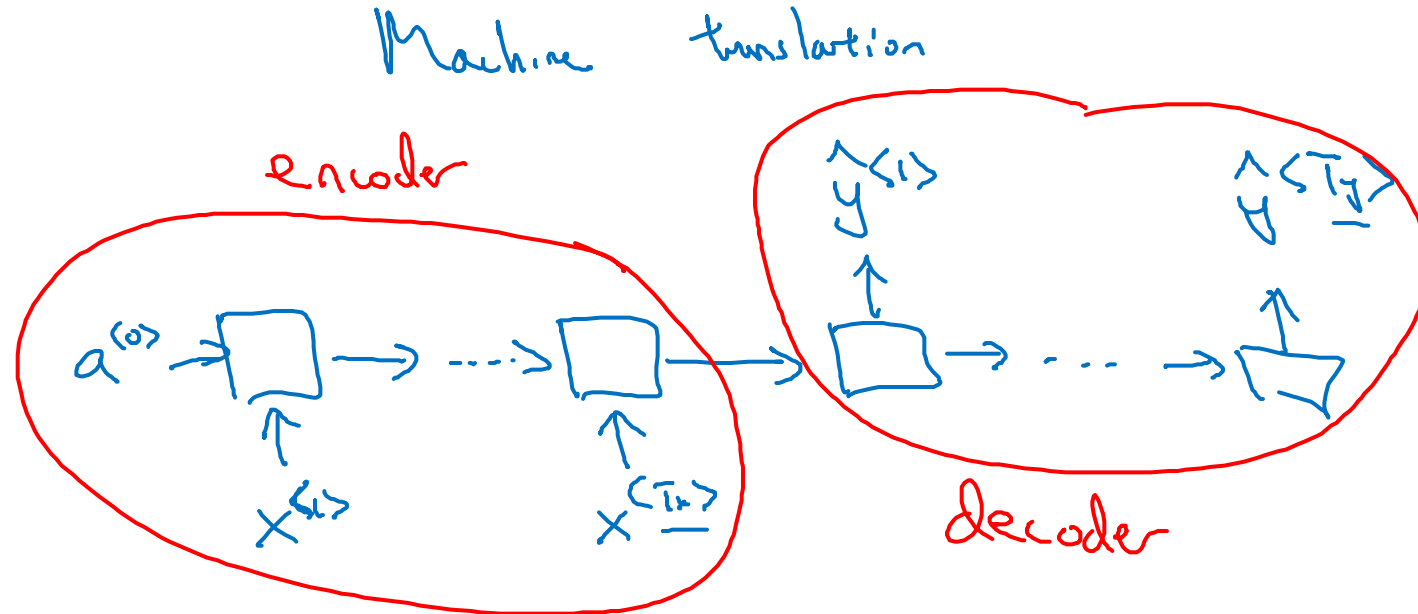
Examples of RNN architectures



One-to-many

$$x = \phi$$

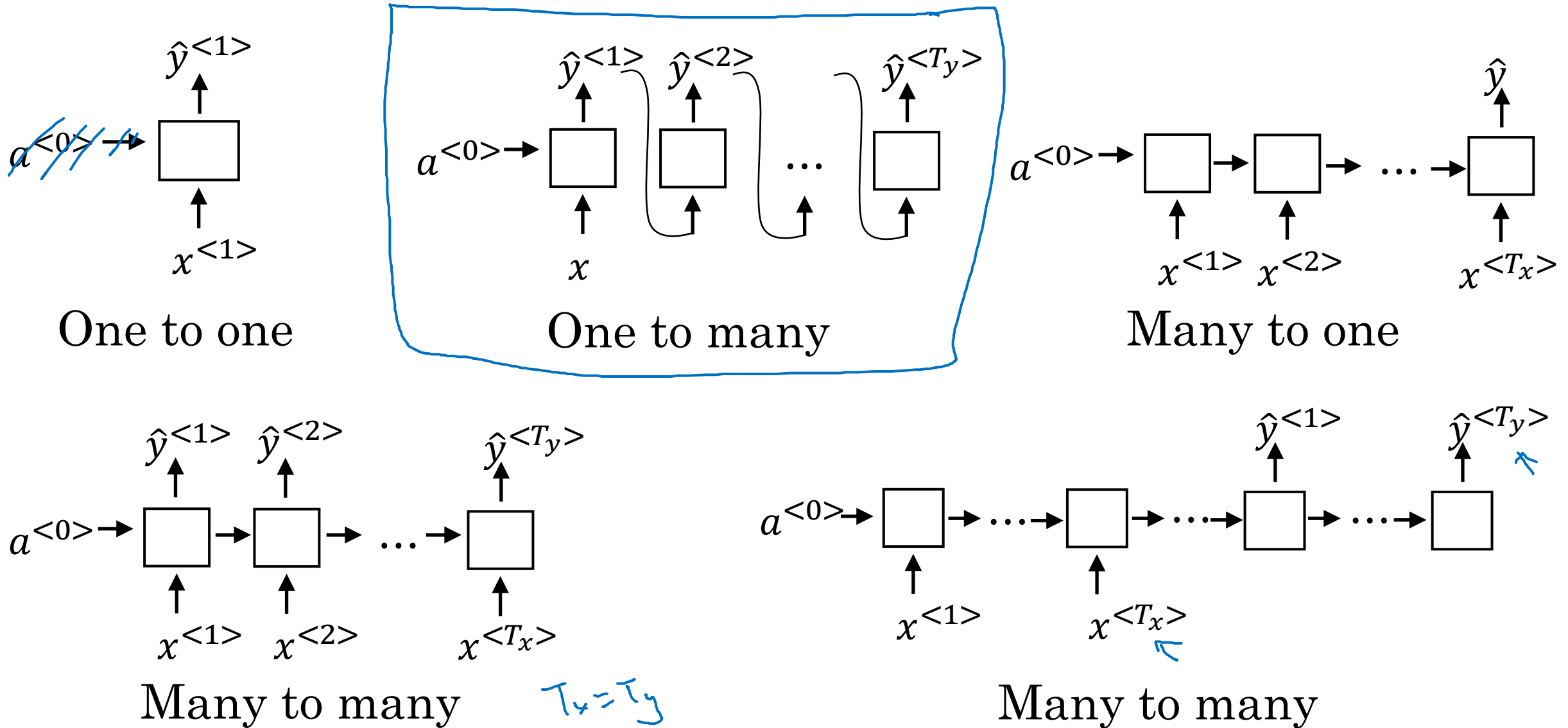
Interesting both 'a' and 'y' feed to next layer



Many-to-many

Translation

Summary of RNN types





deeplearning.ai

Recurrent Neural Networks

Language model and
sequence generation

What is language modelling?

Language model provides probability for a sentence

Speech recognition

The apple and pair salad.

→ The apple and pear salad.

$$P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$$

$$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$$

$$P(\text{Sentence}) = ?$$

$$P(y^{(1)}, y^{(2)}, \dots, y^{(T)})$$

Speech recognition system will pick the sentence with higher probability. Probability will come from Language model.

Language model assigns higher probability to second sentence. Even though both sound same.

Language modelling with an RNN

Training set: large corpus of english text.

collection

Tokenize

add an 'end of sentence' token;
comes in handy

Cats average 15 hours of sleep a day. \downarrow $\langle \text{EOS} \rangle$

One-Hot
vectors

$y^{(1)}$

$y^{(2)}$

$y^{(3)}$

...

$y^{(8)}$

$y^{(9)}$

$x^{(t)} = y^{(t-1)}$

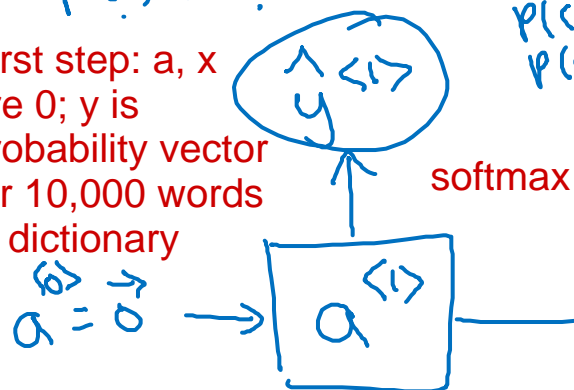
The Egyptian ~~Mau~~ is a breed of cat. $\langle \text{EOS} \rangle$

$\langle \text{UNK} \rangle$

10,000

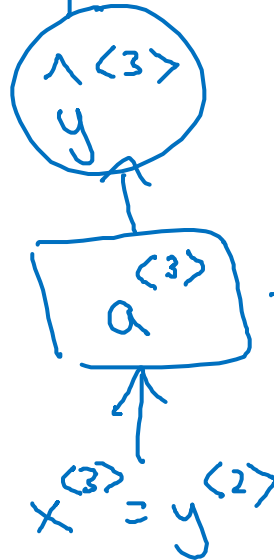
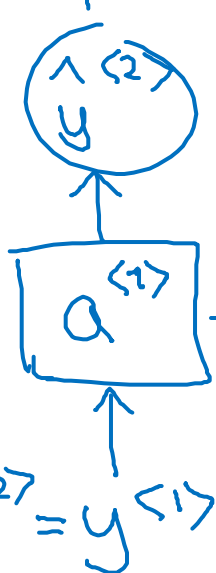
RNN model

First step: a , x are 0; y is probability vector for 10,000 words in dictionary

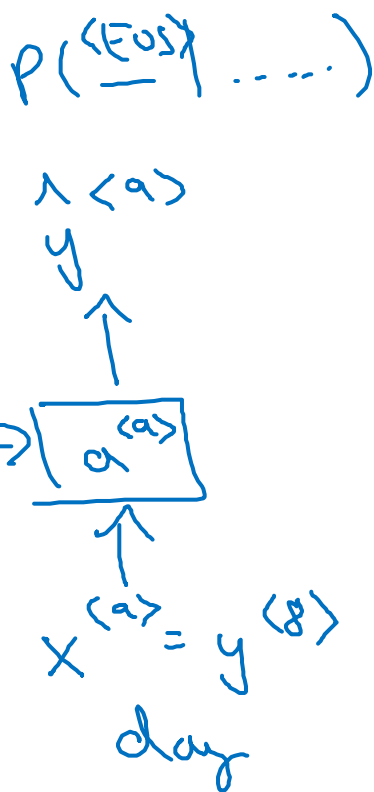


meaning - what is the probability for any of those 10,000 words to be the 1st word

at second step, we tell it correct first word Cats



Average



→ Cats average 15 hours of sleep a day. <EOS>

$$P(y^{(1)}, y^{(2)}, y^{(3)}) \leftarrow$$

$$= \frac{P(y^{(1)}) P(y^{(2)} | y^{(1)})}{P(y^{(3)} | y^{(1)}, y^{(2)})}$$

For speech recognition:
Prob(sentence) = multi. of conditional probabilities

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = - \sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

trained by reducing this loss.

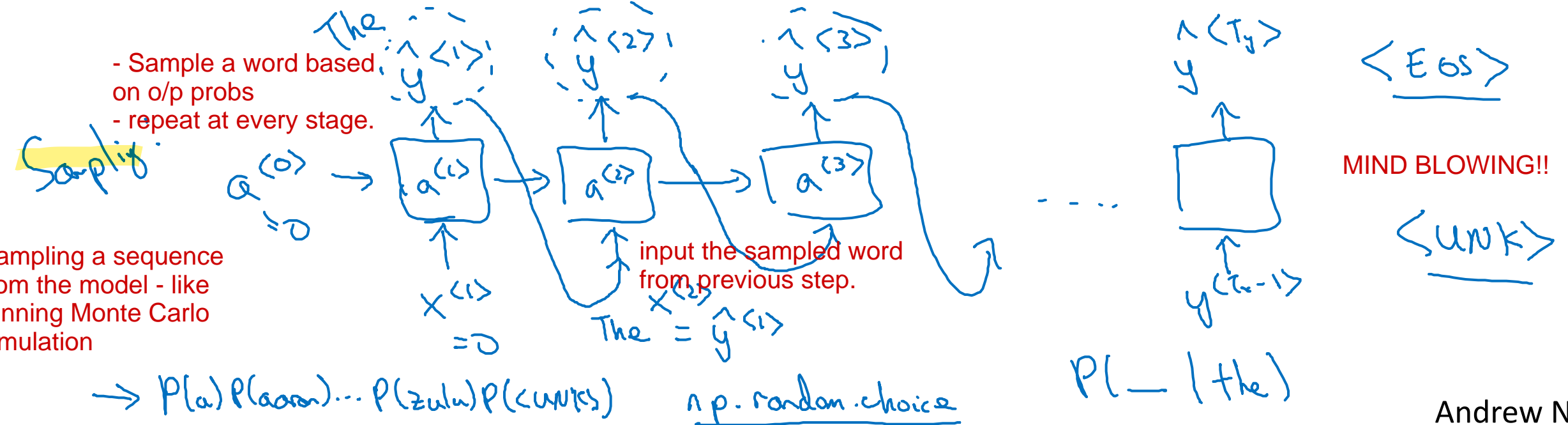
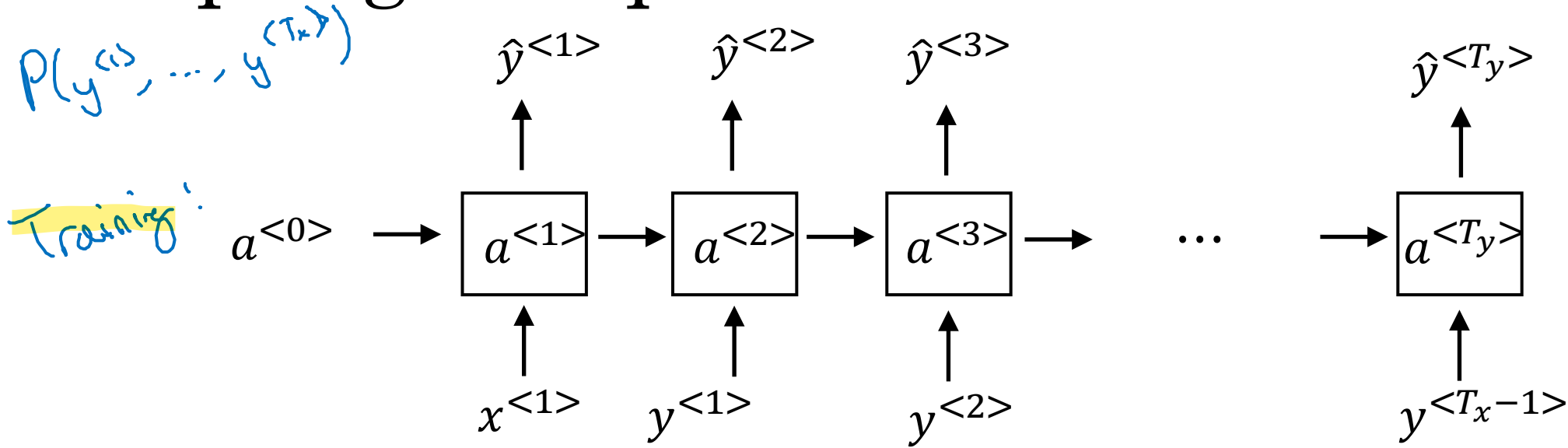


deeplearning.ai

Recurrent Neural Networks

Sampling novel
sequences

Sampling a sequence from a trained RNN



Character-level language model

→ Vocabulary = [a, aaron, ..., zulu, <UNK>] ←

→ Vocabulary = [a, b, c, ..., z, $_$, ., , , ;, 0, ..., 9, A, ..., Z]

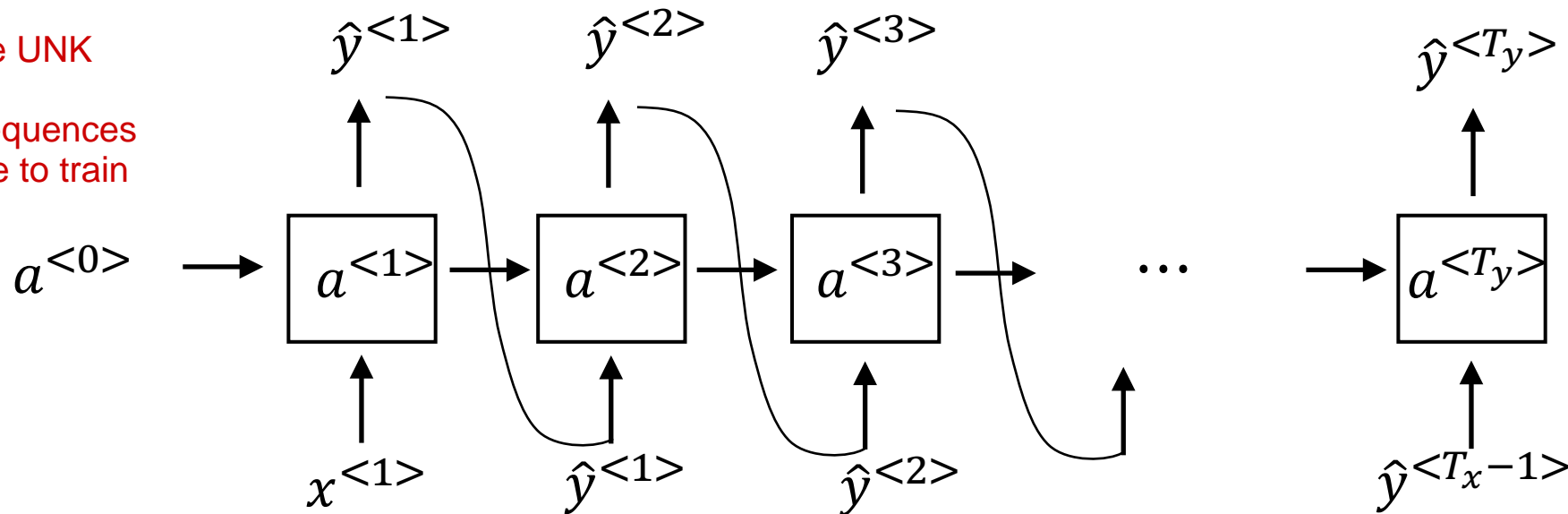
Dictionary can also be at
CHARACTER level -
azAZ09,.,; etc.

$y^{(1)}$ $y^{(2)}$ $y^{(3)}$ $y^{(4)}$

Cat
↑ ↑ ↑ ↑
average

Man

- + never generate UNK token
- much longer sequences
- more expensive to train



Sequence generation

News

President enrique peña nieto, announced
sench's sulk former coming football langston
paring.

"I was not at all surprised," said hich langston.

"Concussion epidemic", to be examined. ←

The gray football the told some and this has on
the uefa icon, should money as.

Shakespeare

The mortal moon hath her eclipse in love.

And subject of this thou art another this fold.

When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.



deeplearning.ai

Recurrent Neural Networks

Vanishing gradients with RNNs

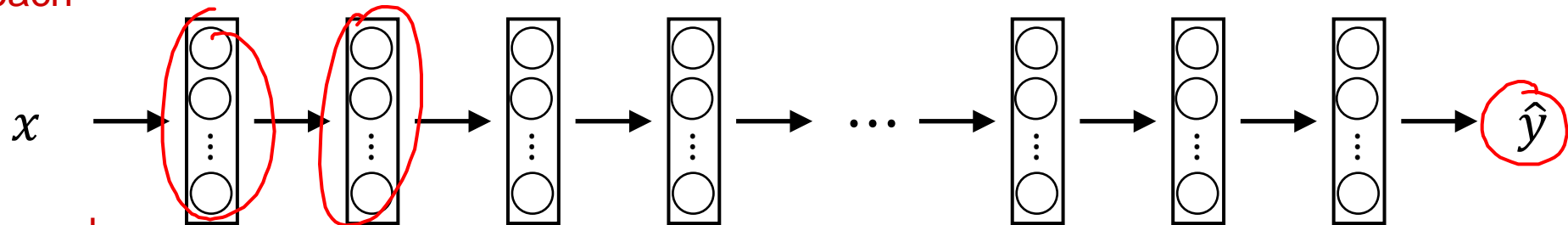
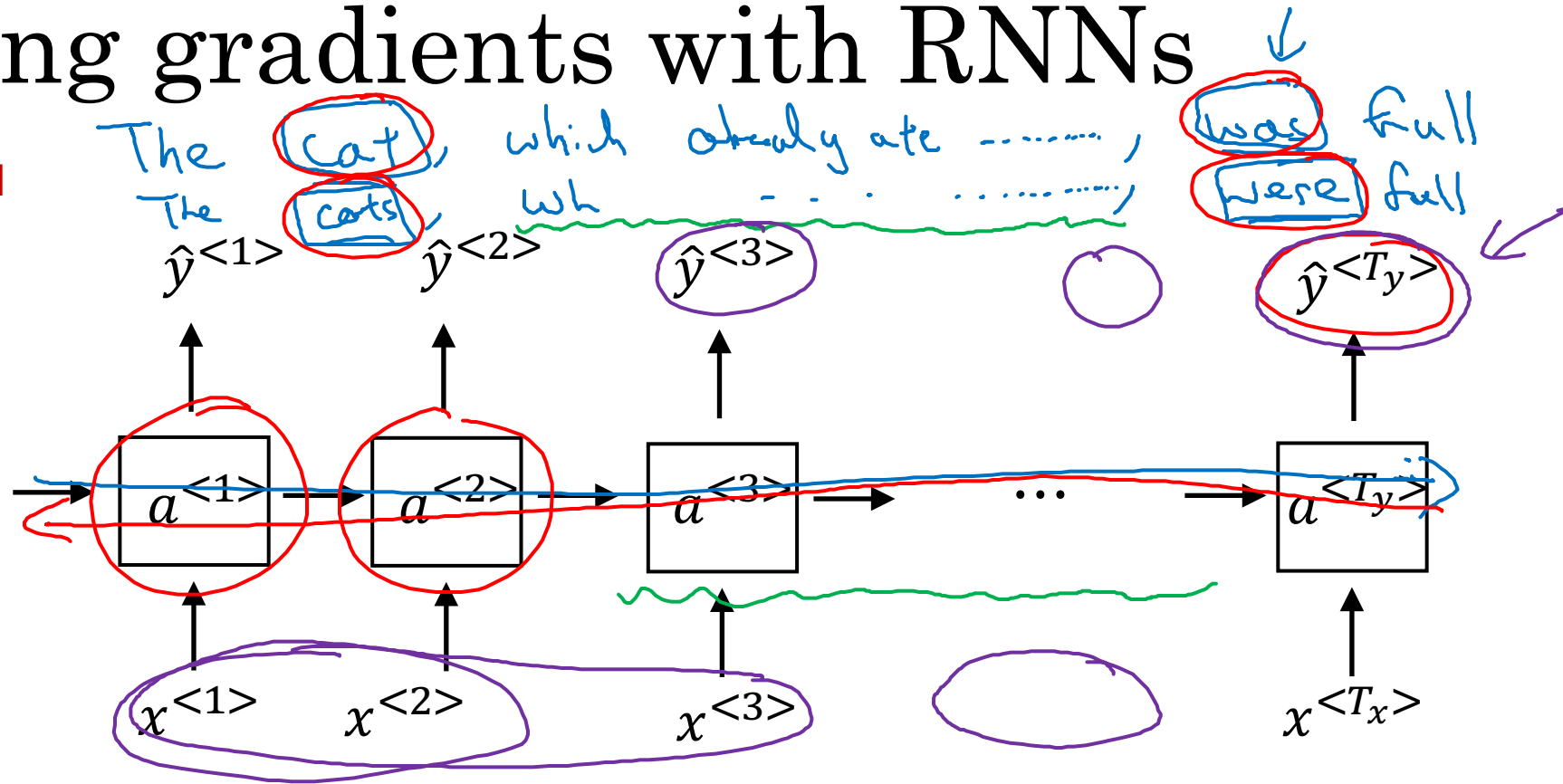
Vanishing gradients with RNNs

Gradient has a hard time propagating back for a deep network. Hard for o/p at a later layer to influence parameters at earlier layer $a^{<0>}$

Basic RNN - can only model 'local' influences, i.e. close by terms impacting each other

gradients may also explode and require clipping

Exploding gradients.



NaN

Gradient clipping

100

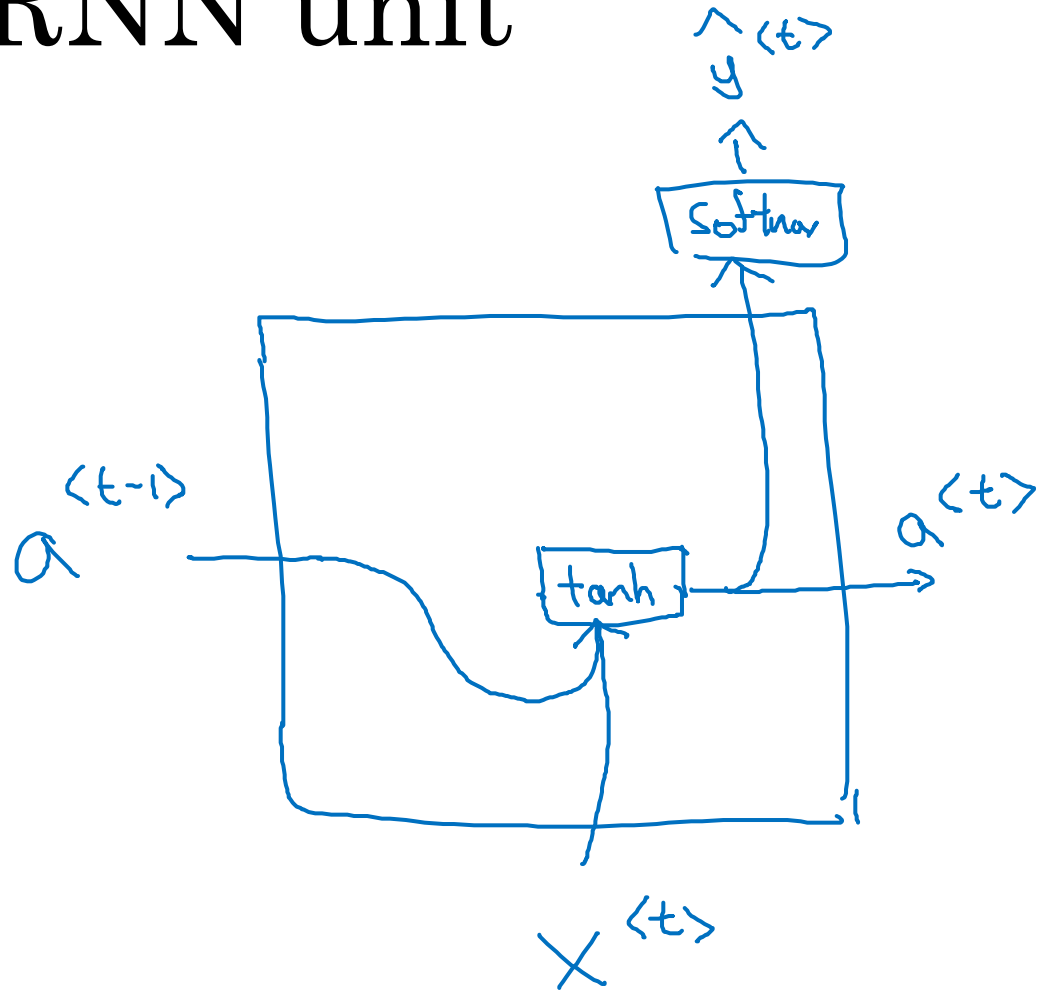


deeplearning.ai

Recurrent Neural Networks

Gated Recurrent Unit (GRU)

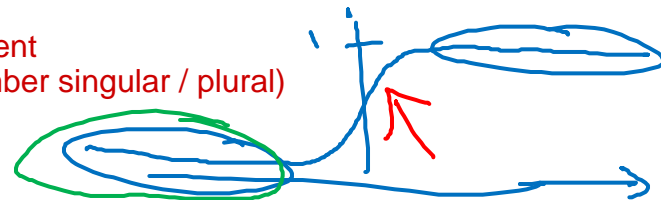
RNN unit



$$\underline{a^{<t>}} = \overset{\substack{\text{tanh} \\ \downarrow}}{g}(\underbrace{W_a[a^{<t-1>}, x^{<t>}]}_{\uparrow} + b_a)$$

GRU (simplified)

Key Idea: $C(t)$ vector is a candidate to replace $C(t)$
 Gate (gamma-u) vector either $\{0,1\}$ (sigmoid) - controls the assignment
 Remember this are all vectors! (maybe one bit is assigned to remember singular / plural)



$C =$ memory cell

$$\rightarrow \boxed{C^{(t)}} = \underline{a}^{(t)}$$

$$\rightarrow \boxed{\tilde{C}^{(t)}} = \tanh(W_c [C^{(t-1)}, x^{(t)}] + b_c)$$

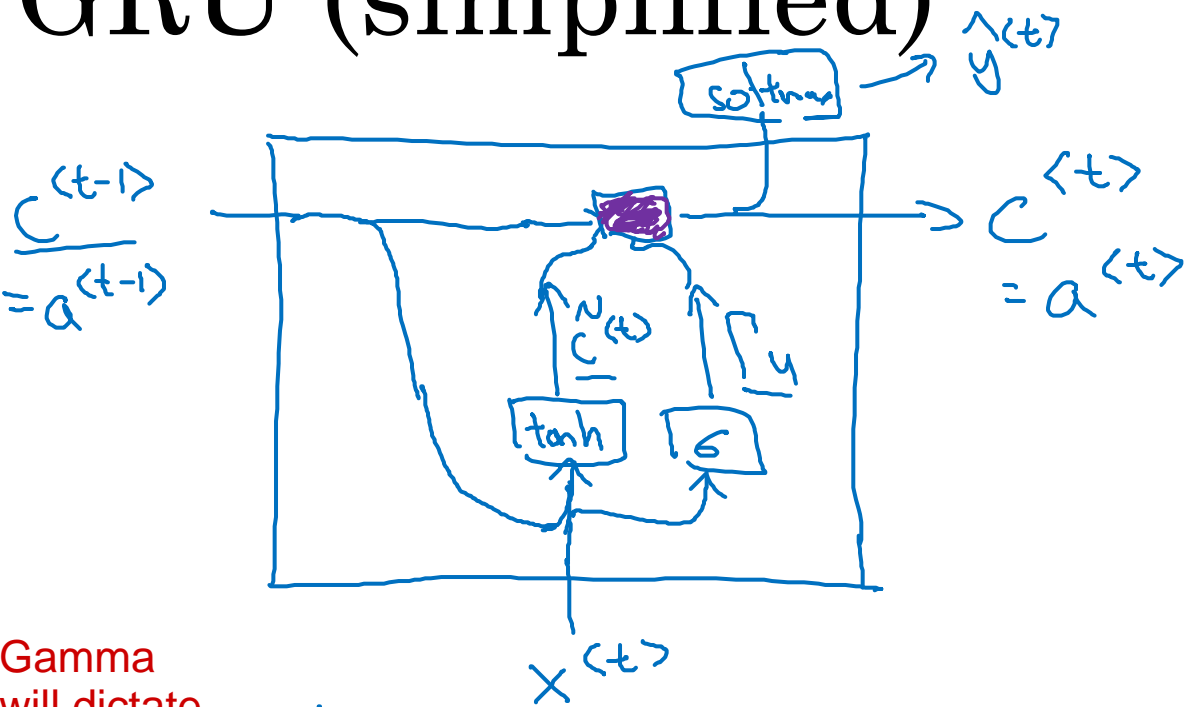
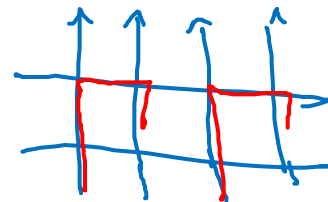
$$\rightarrow \boxed{\Gamma_u} = \sigma(W_u [C^{(t-1)}, x^{(t)}] + b_u)$$

$$\boxed{C^{(t)}} = \underbrace{\Gamma_u}_{\text{"update"}} * \tilde{C}^{(t)} + (1 - \Gamma_u) * \boxed{C^{(t-1)}}$$

element-wise

Gate

$$\Gamma_u = 0.000001$$



Gamma will dictate when to update that memory cell

$$\Gamma_u = 1 \quad \Gamma_u = 0 \quad \Gamma_u = 0 \quad \Gamma_u = 0 \quad \dots$$

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

[Cho et al., 2014. On the properties of neural machine translation: Encoder-decoder approaches]

[Chung et al., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling]

Andrew Ng

Full GRU

$$\tilde{c}^{<t>} = \tanh(W_c [\tilde{c}^{<t-1>}, x^{<t>}] + b_c)$$

$$\begin{cases} \Gamma_u = \sigma(W_u [c^{<t-1>}, x^{<t>}] + b_u) \\ \Gamma_r = \sigma(W_r [c^{<t-1>}, x^{<t>}] + b_r) \end{cases}$$

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

The cat, which ate already, was full.

for LSTM we make one modification:

- add 'Gamma(r)' to first equation
- C(tilda) equation becomes: $W_c [\Gamma_r * c^{<t-1>}, x^{<t>}]$
- Gamma(r) represents relevance - very similar to Gamma(u)

Why use Gamma(r)

- based on research ideas to ensure gradients don't vanish and long-term relevance is captured

LSTM

LSTM is another common version

KEY INSIGHT: This Gate vector can become 1 for a particular index, and replace C(tilda) with C(t-1) for that index of C(t). This takes away the impact of 'W' parameters and activation function and avoid the problem of vanishing or exploding gradients.



deeplearning.ai

Recurrent Neural Networks

LSTM (long short
term memory) unit

GRU and LSTM

$a \neq c$

two gamma gates for $c(t)$ update - 'update' + 'forget'
one gamma gate for 'output'

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * \underline{c}^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\underline{c}^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\underline{c}^{<t-1>}, x^{<t>}] + b_r)$$

$$\underline{c}^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * \underline{c}^{<t-1>}$$

$a^{<t>} = c^{<t>}$

Γ_f

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

(update) $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

(forget) $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

(output) $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

$$\underline{c}^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * \underline{c}^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \cancel{c^{<t>}} \quad \tanh(\underline{c}^{<t>})$$

LSTM units

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

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

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \overset{\text{tanh}}{(c^{<t>})}$$

LSTM in pictures

3 outputs from the box: $c(t)$, $a(t)$
and $y(t) = \text{softmax}(a(t))$

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

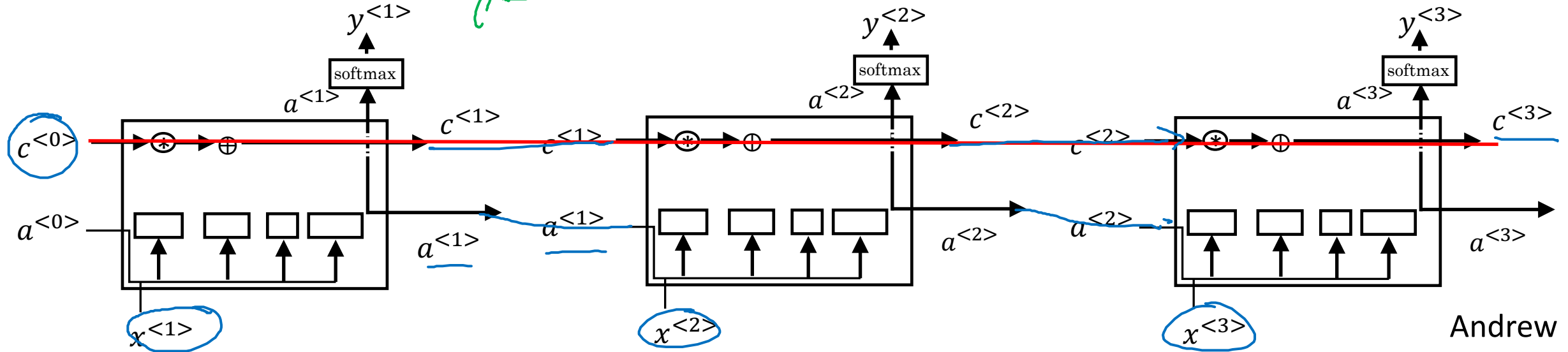
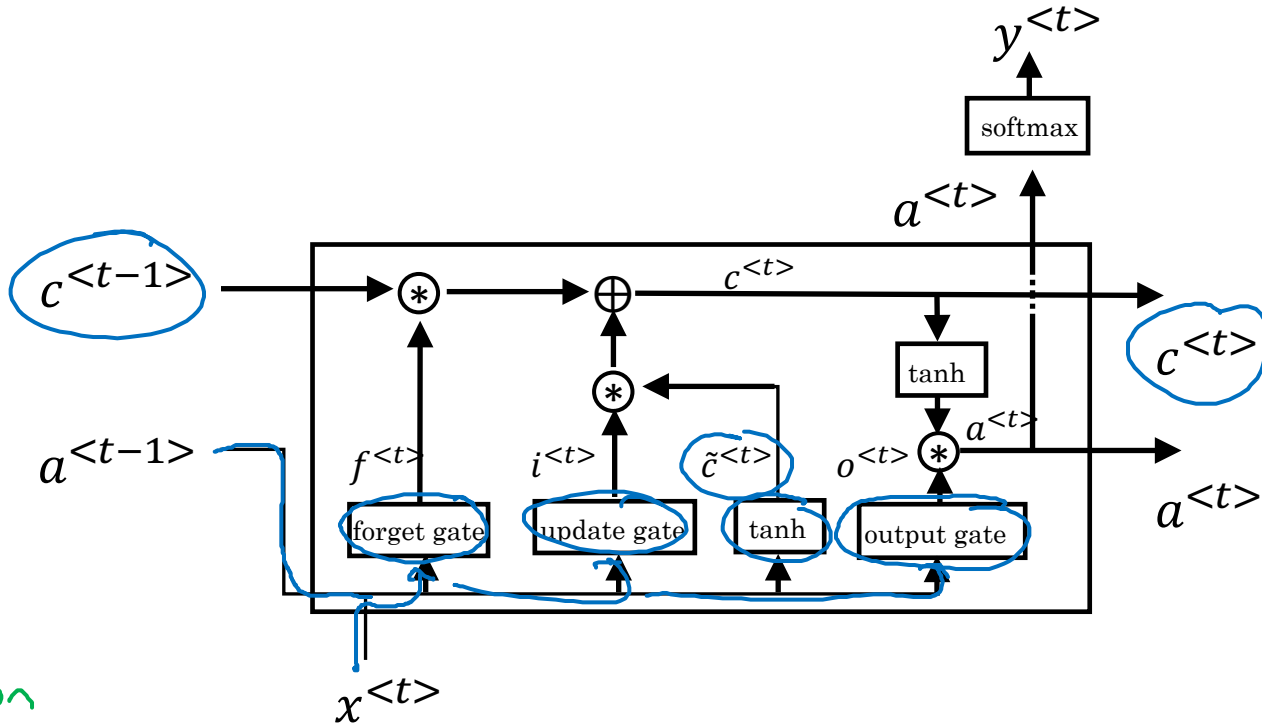
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh(c^{<t>})$$

peephole
connection





deeplearning.ai

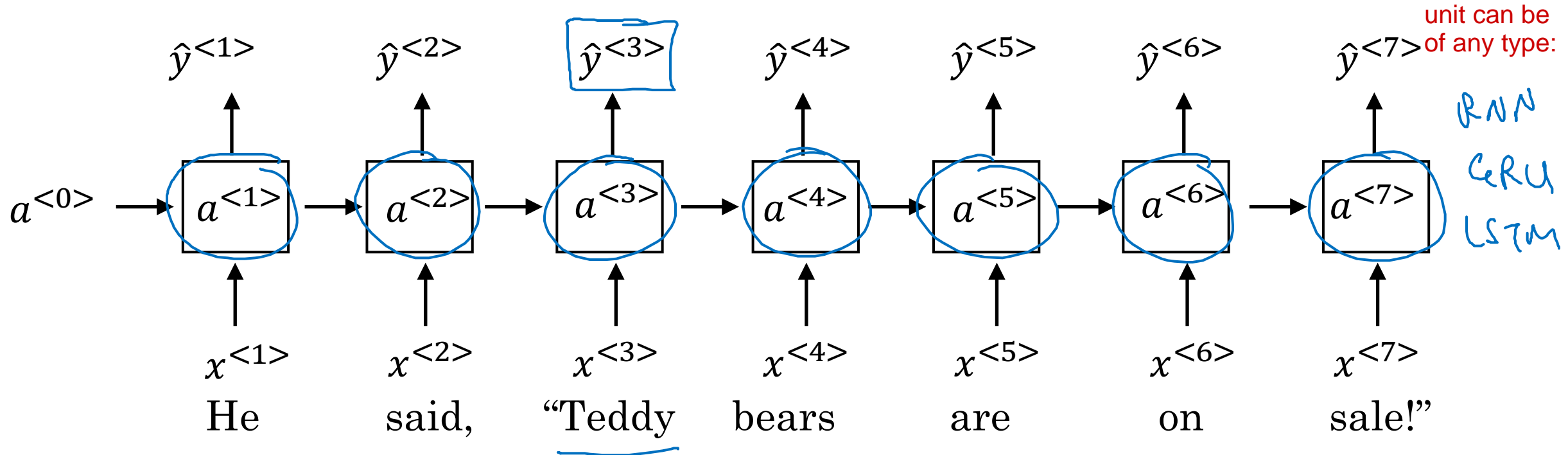
Recurrent Neural Networks

Bidirectional RNN

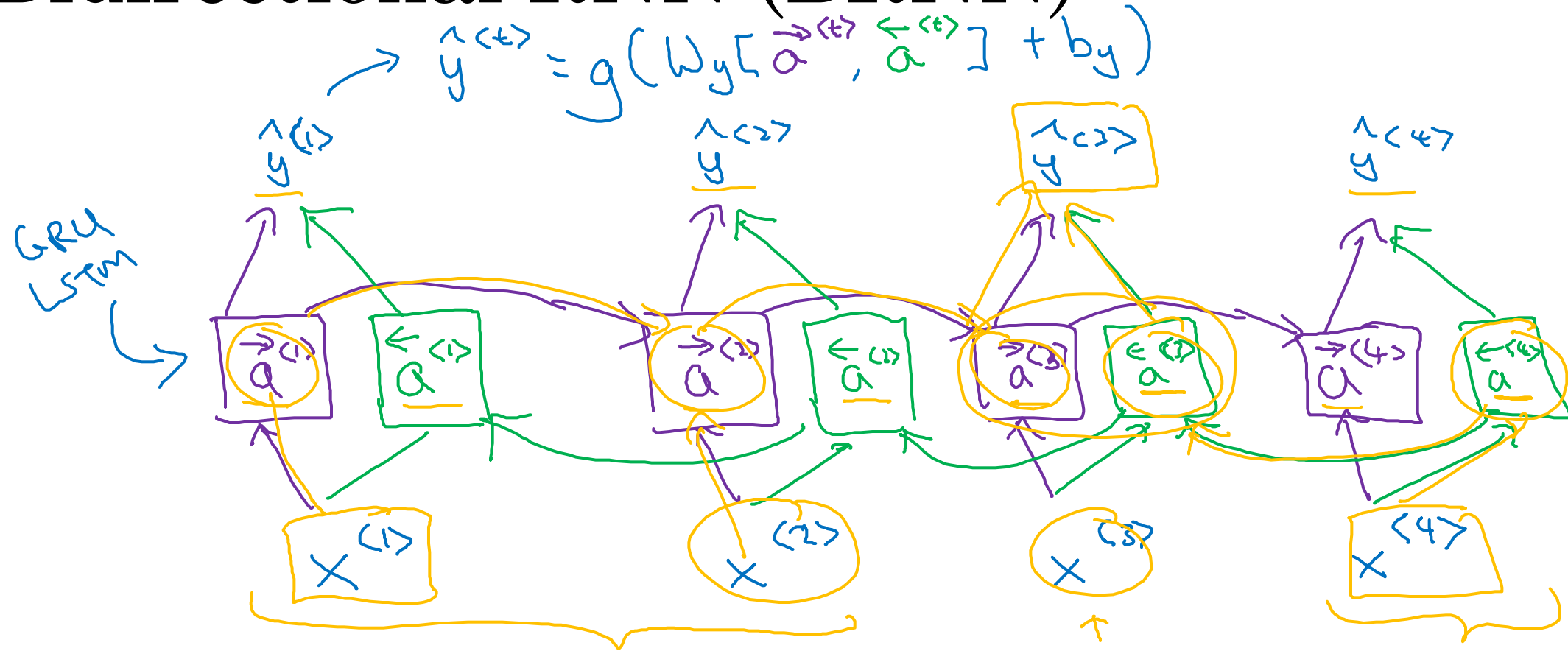
Getting information from the future

He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



Bidirectional RNN (BRNN)



Acyclic graph

He said "Teddy Roosevelt ..."

BRNN w/ LSTM

Common choice for NLP is to use BRNN + LSTM
- need entire sequence of data to make prediction
- e.g. won't work for translation, until the person stops speaking



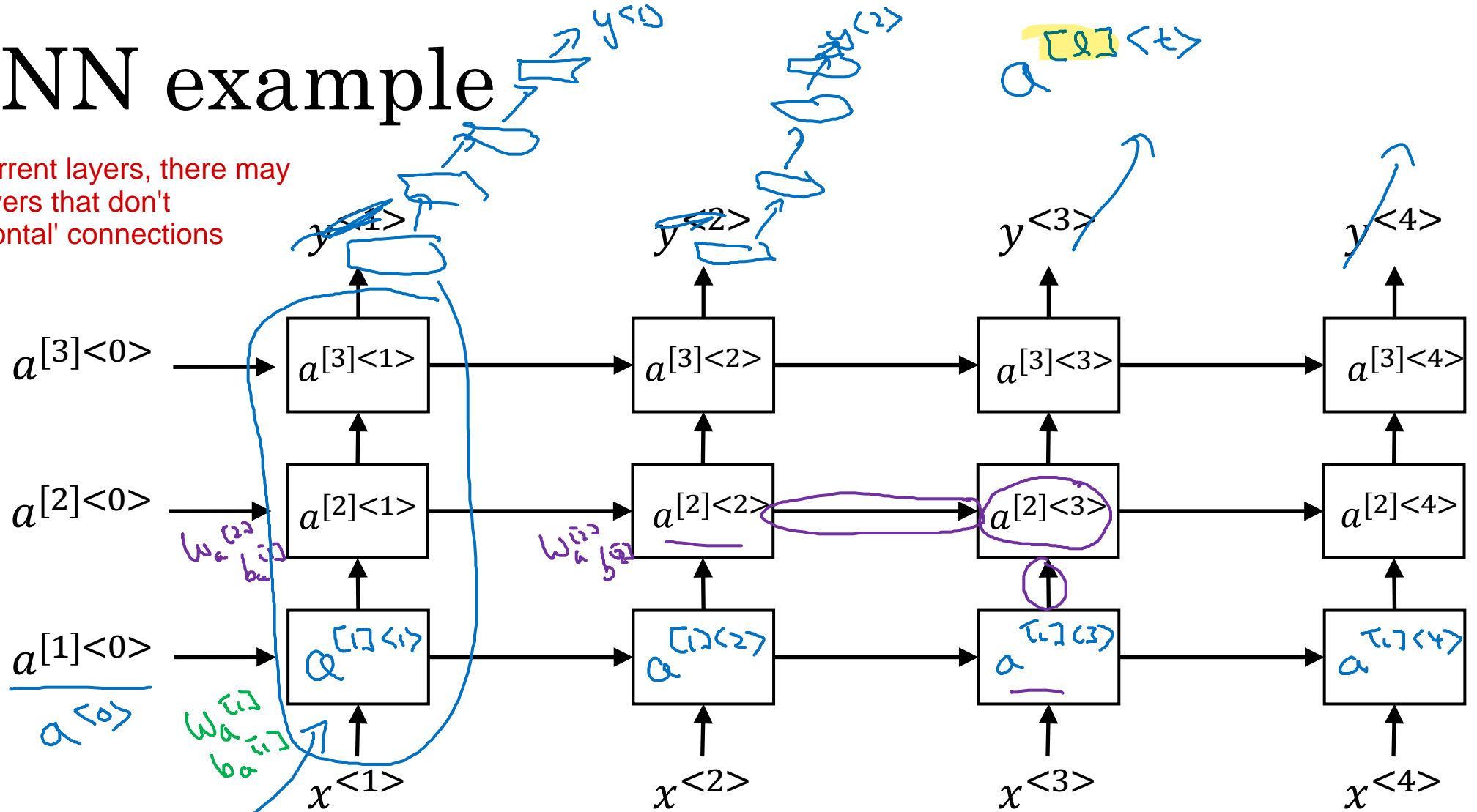
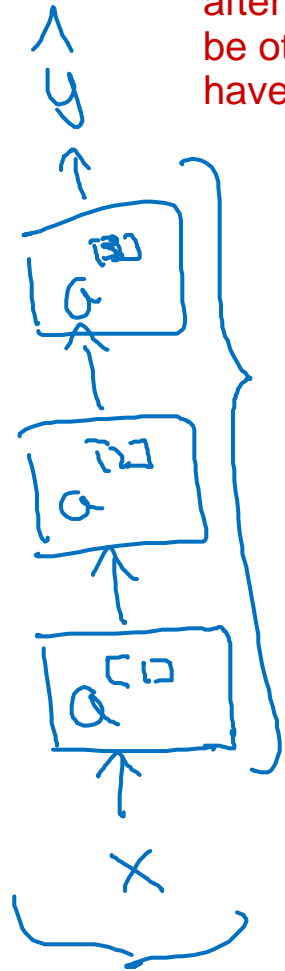
deeplearning.ai

Recurrent Neural Networks

Deep RNNs

Deep RNN example

after 3 recurrent layers, there may be other layers that don't have 'horizontal' connections



RNN
GRU
LSTM

BRNN

$$a^{(2)} \langle 3 \rangle = g(W_a^{(2)} [a^{(2)} \langle 2 \rangle, a^{(1)} \langle 3 \rangle] + b_a^{(2)})$$

Each layer has its own W, b parameters - same as before. (Same within layer, diff. across layers) Andrew Ng

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

NLP and Word Embeddings

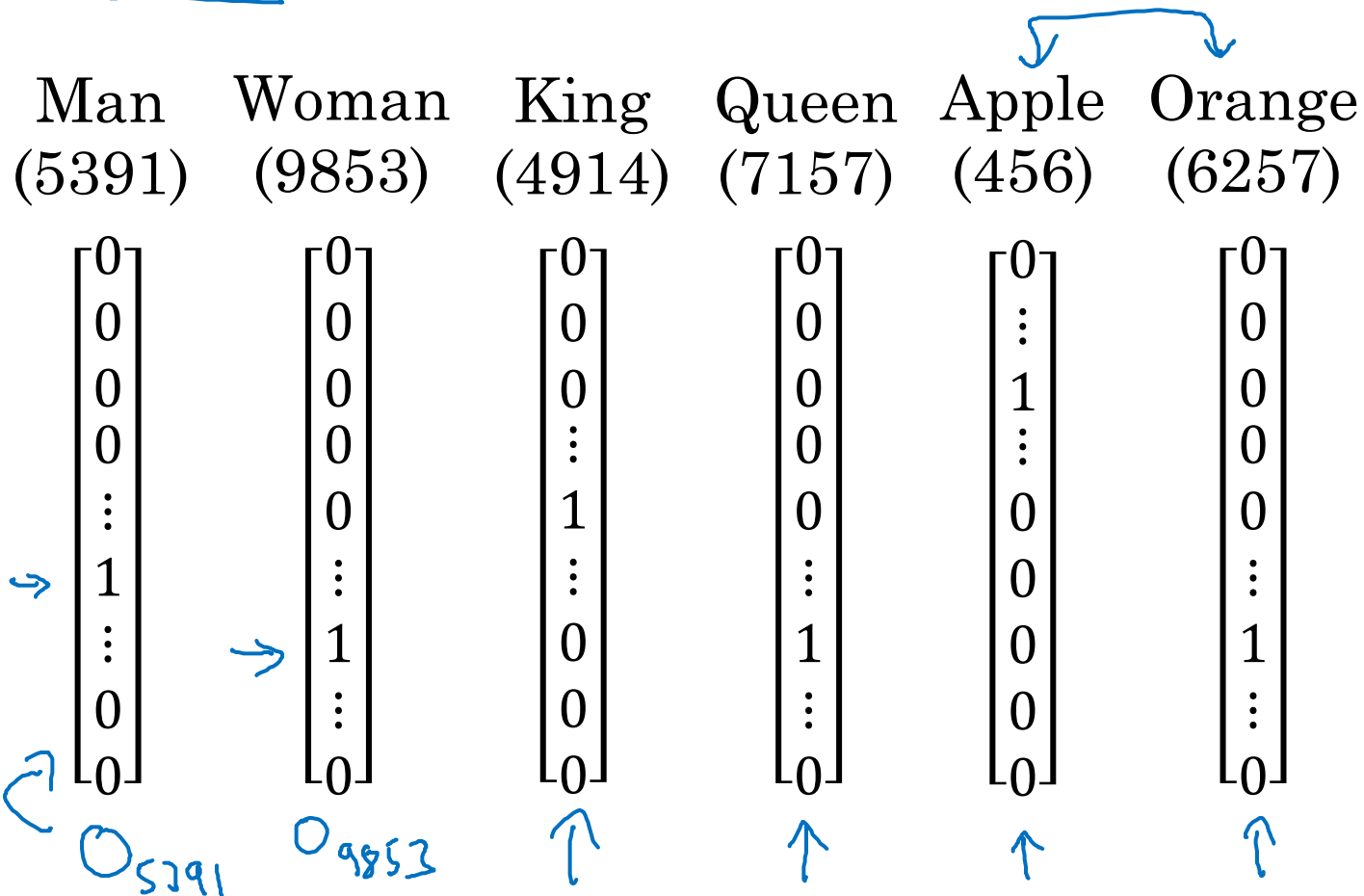
Word representation

Word representation

$V = [a, aaron, \dots, zulu, <UNK>]$

$|V| = 10,000$

1-hot representation



I want a glass of orange juice.

I want a glass of apple ?.

OHE doesn't make it easy generalize relationships
- because inner product of any two different OHE is 0
- even if the algorithm learns it should be 'juice' after orange, it cannot translate that to apple, and will have to relearn that for apple

Featurized representation: word embedding

Makes generalization easier

Lateral Features	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	<u>0.93</u>	<u>0.95</u>	-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
Size	⋮	⋮				
Cost						
alive						
verb						

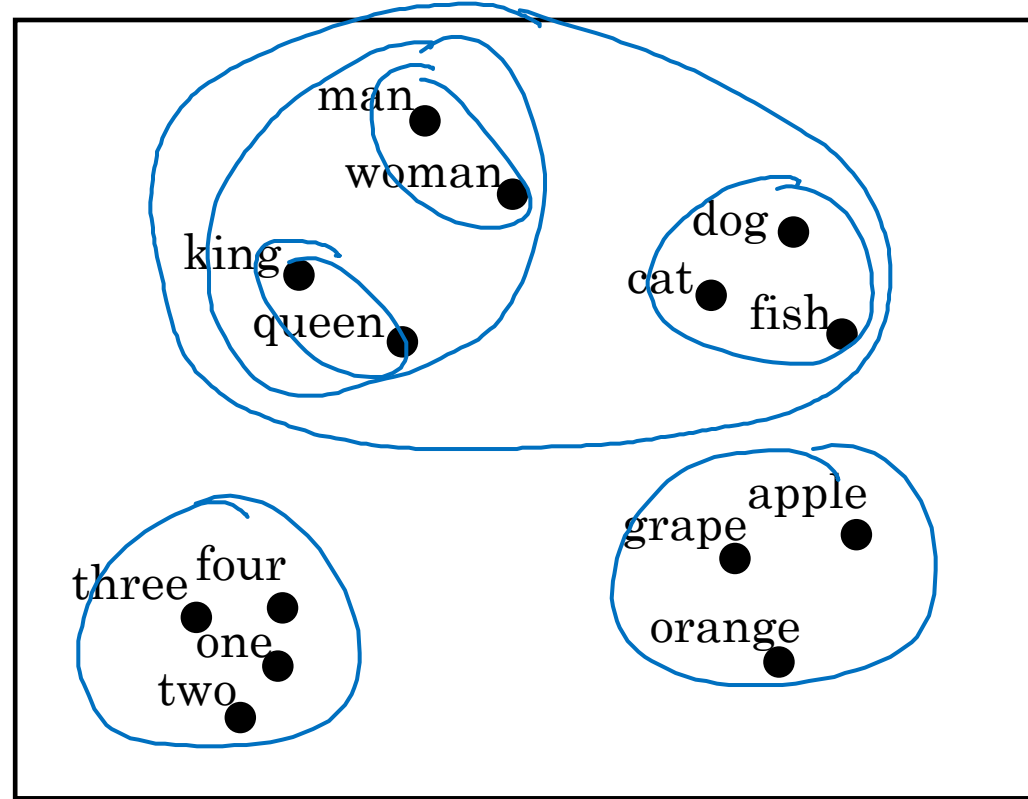
I want a glass of orange juice.

I want a glass of apple juice.

Andrew Ng

Visualizing word embeddings

t-SNE algorithm to visualize
300D feature space in 2D graph

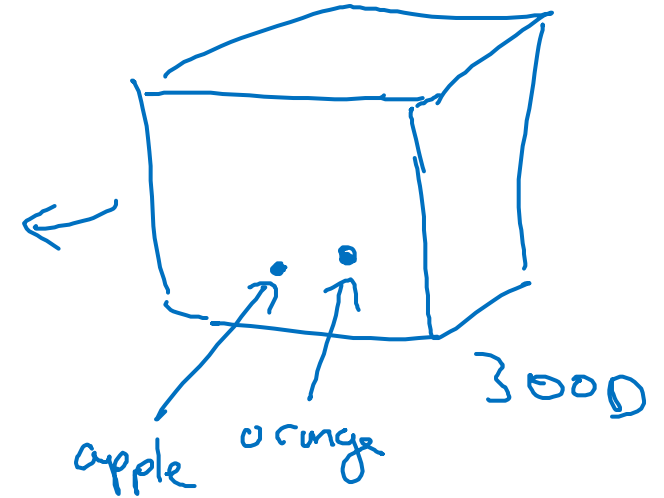


t-SNE

→ 300 D



2D





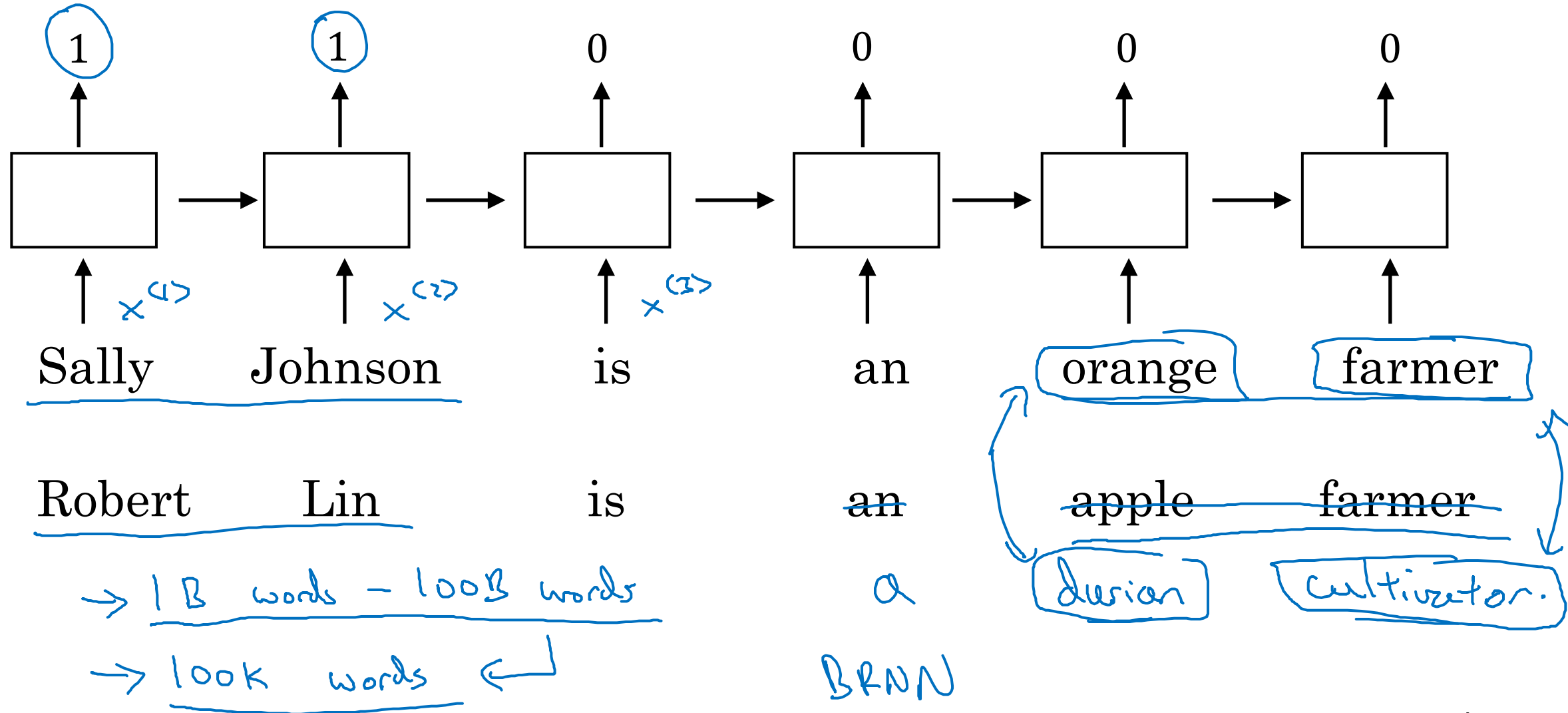
deeplearning.ai

NLP and Word Embeddings


Using word embeddings

Named entity recognition example

Word embeddings algorithm may train on large corpus of unlabeled text - to learn similarities



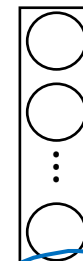
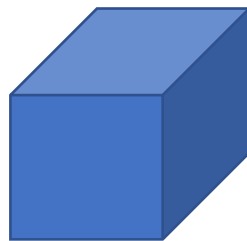
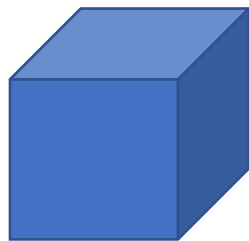
Transfer learning and word embeddings

- 
1. Learn word embeddings from large text corpus. (1-100B words)
(Or download pre-trained embedding online.)
 2. Transfer embedding to new task with smaller training set.
(say, 100k words) → 10,000 → 300
 3. Optional: Continue to finetune the word embeddings with new data.

Relation to face encoding (embedding) 128D



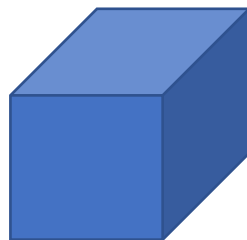
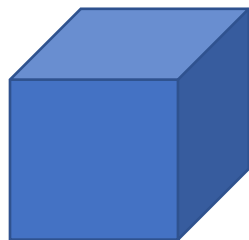
$x^{(i)}$



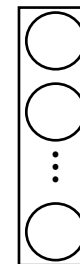
$f(x^{(i)})$



$x^{(j)}$



$f(x^{(j)})$



\hat{y}

$|V| = 10,000$

$e_1, \dots, e_{10,000}$

CNN can provide encoding for 'unseen' images also
Word Embedding can only provide embedding for the seen words / fixed vocabulary



deeplearning.ai

NLP and Word Embeddings

Properties of word embeddings

Analogy

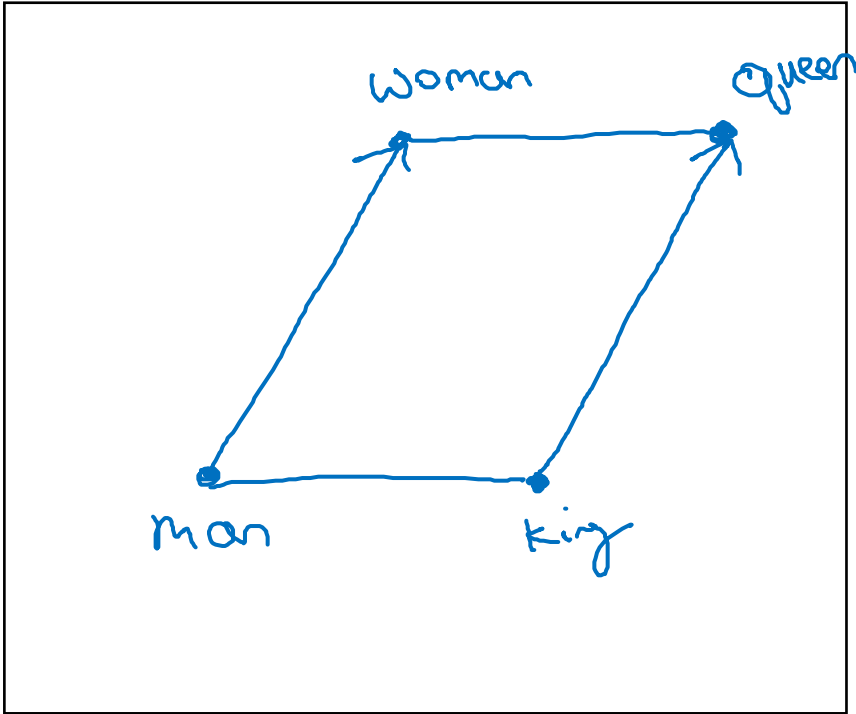
	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

$$\underbrace{e_{\text{Man}}}_{5391} - \underbrace{e_{\text{Woman}}}_{9853} \approx \underbrace{e_{\text{King}}}_{4914} - \underbrace{e_{\text{Queen}}}_{7157}$$

$$\underbrace{e_{\text{Man}}}_{5391} - \underbrace{e_{\text{Woman}}}_{9853} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\underbrace{e_{\text{King}}}_{4914} - \underbrace{e_{\text{Queen}}}_{7157} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

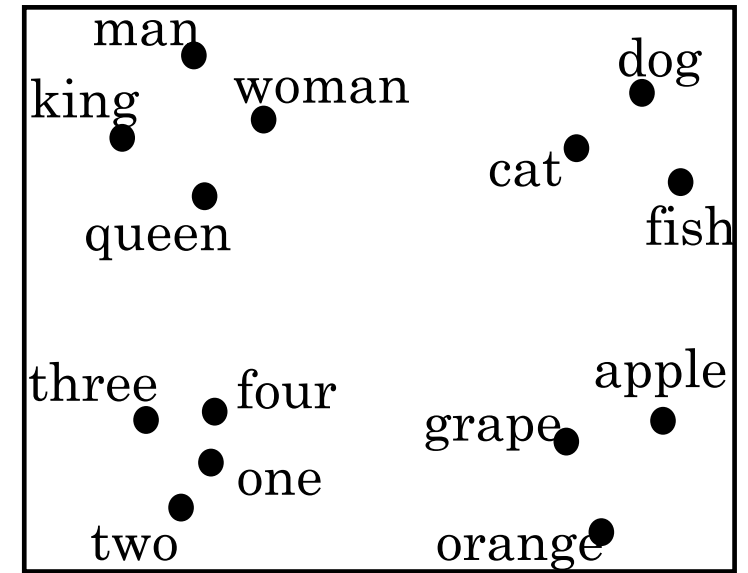
Analogies using word vectors



300 D

Find word w : $\arg \max_w$

3000 \rightarrow 20
↑



t-SNE

$$e_{man} - e_{woman} \approx e_{king} - \cancel{e_w} \quad e_w$$

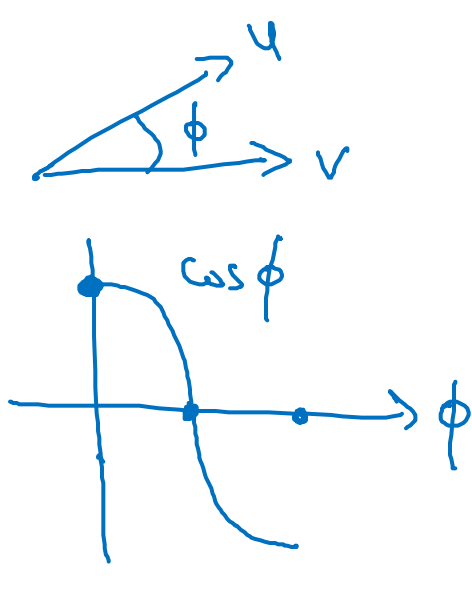
$$\text{Sim}(e_w, e_{king} - e_{man} + e_{woman})$$

30 - 75%

Cosine similarity

$$\rightarrow \text{sim}(e_w, e_{king} - e_{man} + e_{woman})$$

$$\text{sim}(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$



$$\|u - v\|^2$$

Man:Woman as Boy:Girl

Ottawa:Canada as Nairobi:Kenya

Big:Bigger as Tall:Taller

Yen:Japan as Ruble:Russia

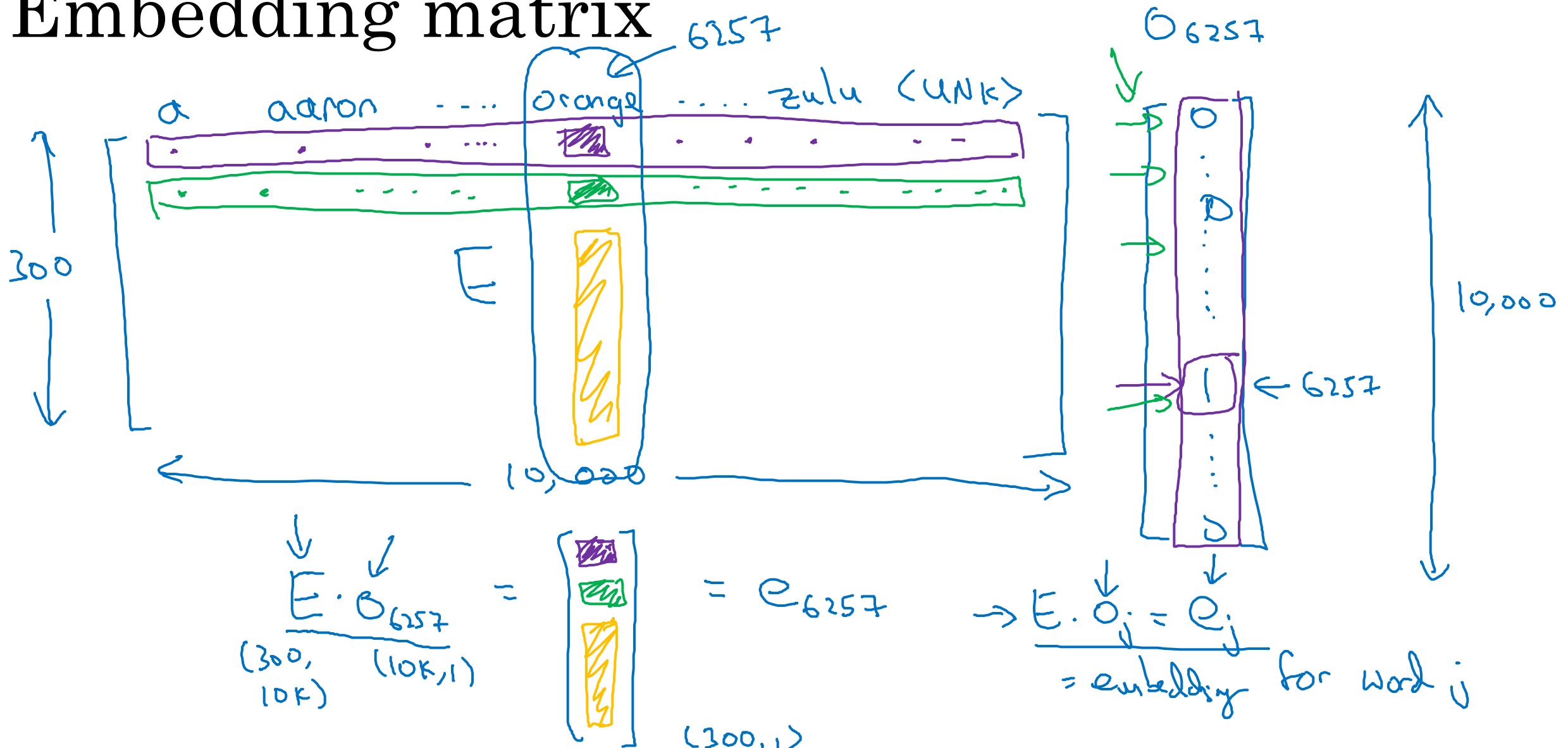


deeplearning.ai

NLP and Word Embeddings

Embedding matrix

Embedding matrix



In practice, use specialized function to look up an embedding.

\rightarrow Embedding



deeplearning.ai

NLP and Word Embeddings

Learning word
embeddings

Neural language model

An earlier approach to learn word embedding matrix E , by training a language model.

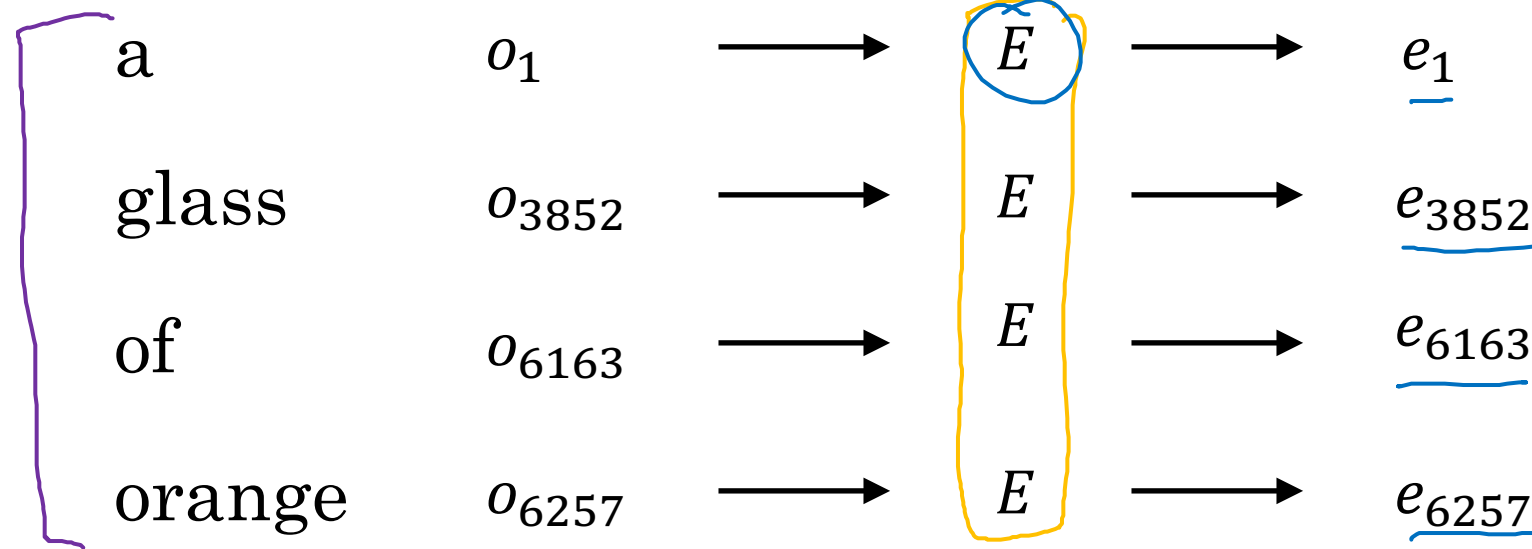
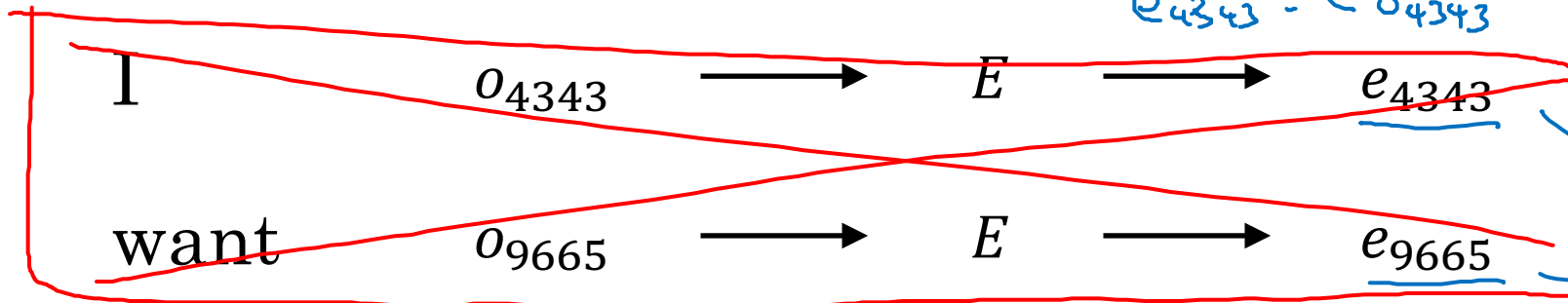
I want a glass of orange juice.

4343 9665 1 3852 6163 6257

$e_{4343} = E_{04343}$

juice.

apple juice.



Softmax

10,000

$W^{(1)}, b^{(1)}$

$W^{(2)}, b^{(2)}$

1800 1200

6 words, 300 length each.. or we may only select "4" word history

Other context/target pairs

For language model, context may be few words preceding the "target"
For word embedding in general, using simpler/ different context works as well.

I want a glass of orange juice to go along with my cereal.

The diagram illustrates context and target pairs for the word "juice". A purple bracket labeled "context" spans the words "a glass of orange". A blue bracket labeled "target" is positioned under the word "juice". A green arrow points from the "orange" box to the "juice" target, and another green arrow points from the "glass" box to the "juice" target.

Context: Last 4 words.

- 4 words on left & right
- Last 1 word
- Nearby 1 word

a glass of orange ? to go along with

orange ?

glass ?

skip gram
skip gram model



deeplearning.ai

NLP and Word Embeddings

Word2Vec

Skip-grams

Randomly pick a context word, and then pick a target word within some range, say +/- 5 words window

I want a glass of orange juice to go along with my cereal.



Context

orange

orange

orange



Target

juice

glass

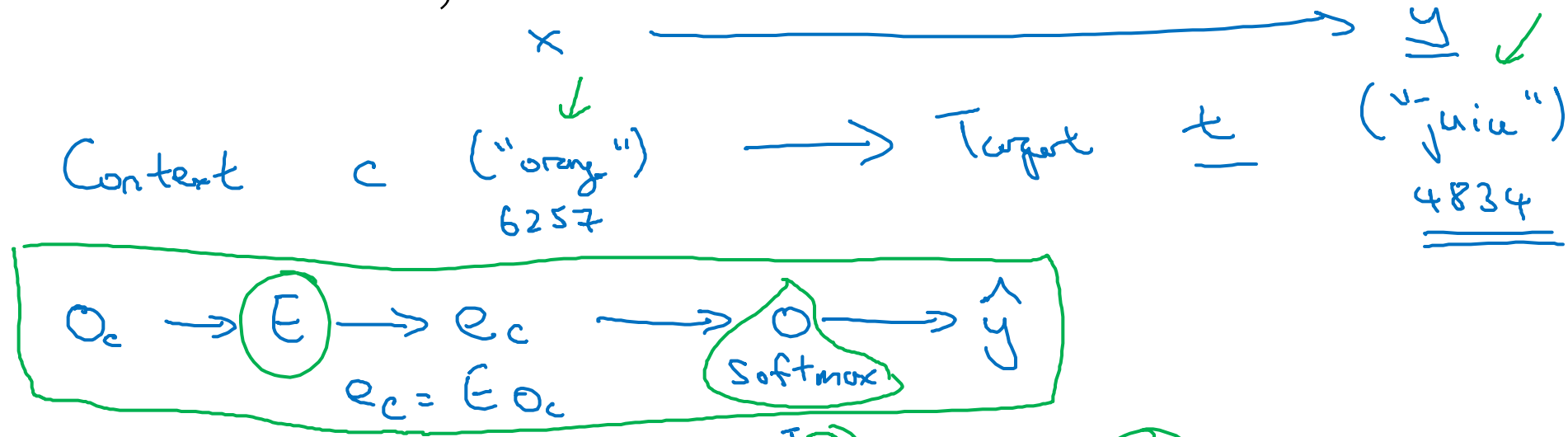
my



Model

One-hot vector (1,10000) x E matrix (10000, 300) => (1,300) Embedding (e_c)
 => $\Theta_t = [W, b]$ (bias can be ignored) (300,10000)
 => Softmax (1, 10000) = prob for each 'target' word given 'context' word

Vocab size = 10,000k



Softmax: $p(t|c) = \frac{e^{\Theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\Theta_j^T e_c}}$

Θ_t = parameter associated with output t

Loss function: $\mathcal{L}(\hat{y}, y) = - \sum_{i=1}^{10,000} y_i \log \hat{y}_i$

Output vector y (one-hot):

$$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \leftarrow 4834$$

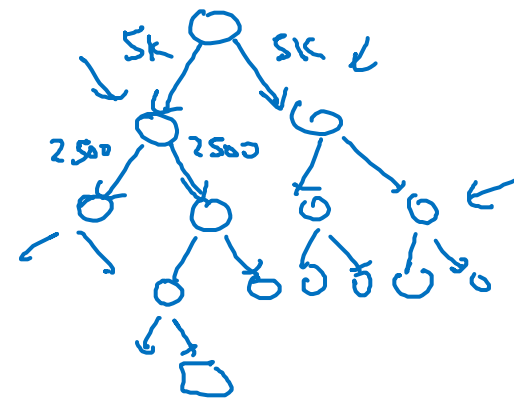
Problems with softmax classification

computationally expensive +
makes it harder to scale $O(n)$

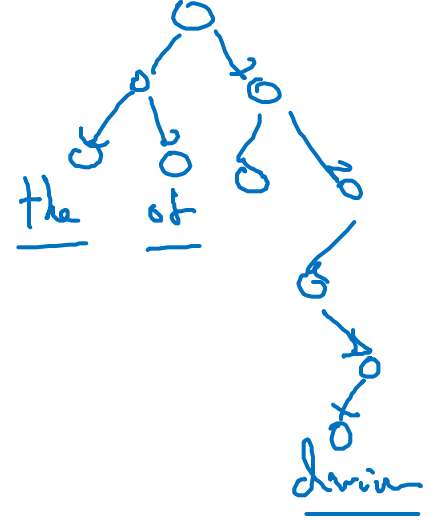
computationally expensive +
makes it harder to scale $O(n)$

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

Hierarchal software.

 $\log |v|$ 

making a tree of classifiers $O(\log n)$



How to sample the context c ?

→ the, of, a, and, to, ...

remove such
common words

$$C \rightarrow t$$

→ orange, apple, durian

$$P(c)$$

Q. Durian



deeplearning.ai

NLP and Word Embeddings

Negative sampling

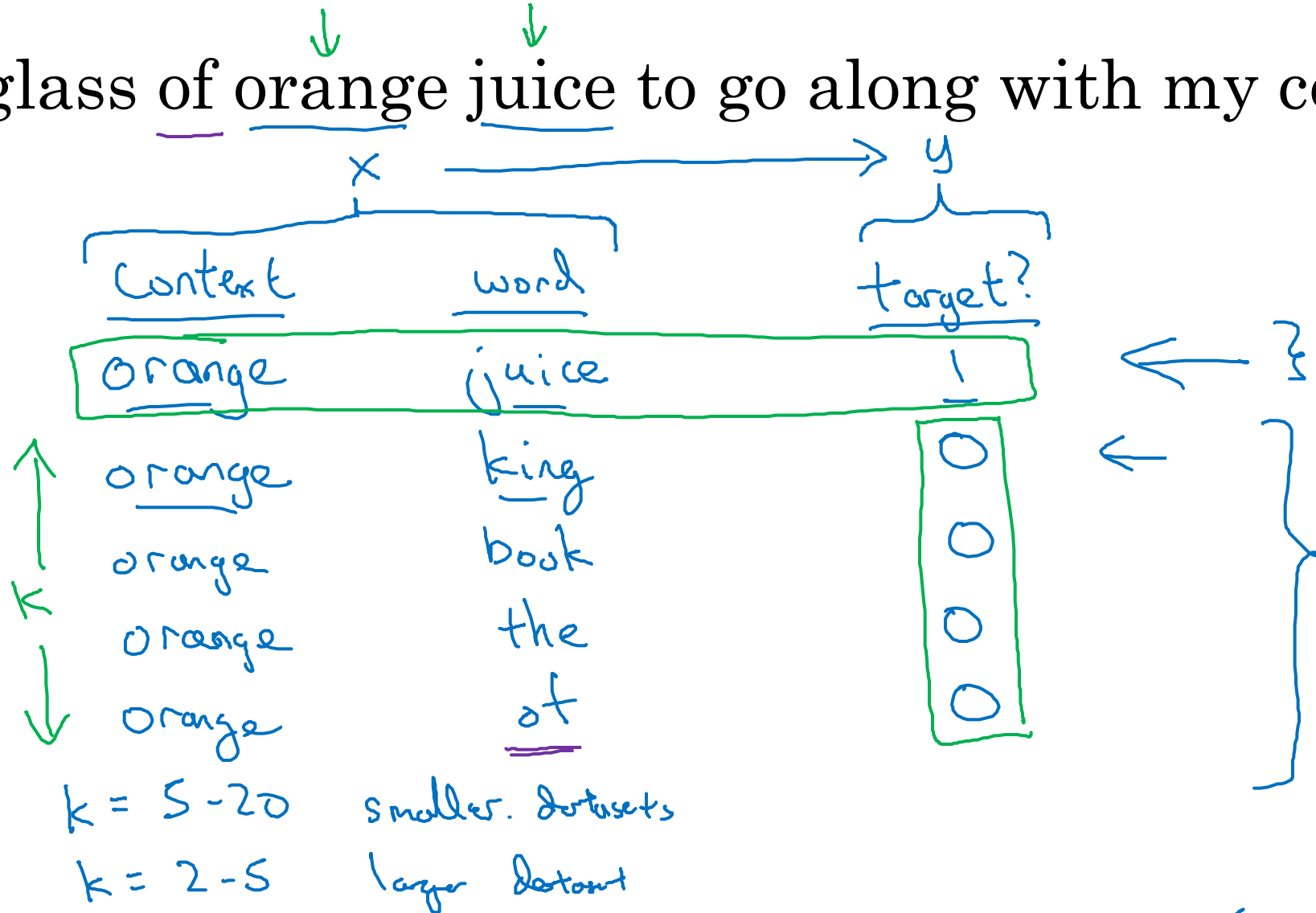
Defining a new learning problem

I want a glass of orange juice to go along with my cereal.

Create a new supervised problem of Positive and Negative examples

Positive sample:
sample context word +
sample target word
from a 5/10 word
window

Negative sample:
sample target word
randomly from
dictionary



Instead of 1 softmax, learn a bunch of binary classifiers

Model

Softmax:
$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

10,000-way softmax

$$P(y=1 | c, t) = \sigma(\theta_t^T e_c) \leftarrow$$

context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

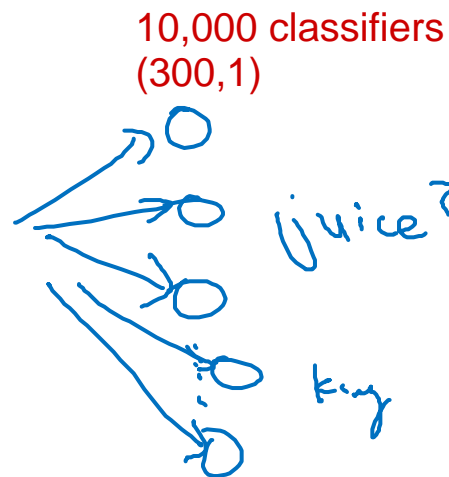
Orange
6257

10000, 300

e_{6257}
1, 10000

$E \rightarrow$

e_{6257}
1, 300



instead of training all 10,000 on each iteration, we only train 'k+1' on each iteration. Scalable.

10,000 binary classification problem

$k+1$

Selecting negative examples

<u>context</u>	<u>word</u>	<u>target?</u>
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

the, of, and, ...

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$$

$$\frac{1}{|V|}$$

How to sample negative words?

(1) using natural frequency: (-ve) a, the, an, of etc.

(2) uniform sampling: (-ve) non-representative of english language

Empirical solution - pick somewhere in between, using a power of 3/4.



deeplearning.ai

NLP and Word Embeddings

GloVe word vectors

GloVe (global vectors for word representation)

I want a glass of orange juice to go along with my cereal.

c, t

X_{ij} = # times i appears in context of j .

$\uparrow \quad \uparrow$
 $c \quad t$

$$X_{ij} = X_{ji}$$

similar if we use word window
not same if we use 'immediately preceding' as context

Model

minimize

$$\sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(x_{ij}) \left(\underbrace{\Theta_i^T e_j}_{\substack{t \quad c \\ \text{"}\Theta_t^T e_c\text{"}}} + b_i + b_j' - \log x_{ij} \right)^2$$

ignore cases where $x_{ij} = 0$
use $0 \log(0) = 0$

weighting term

$$f(x_{ij}) = 0 \text{ if } x_{ij} = 0.$$

$$0 \log 0 = 0$$

reduces weight for frequent words, increases with infrequent words

this, is, of, a, ...
derivation

Θ_i, e_j are symmetric

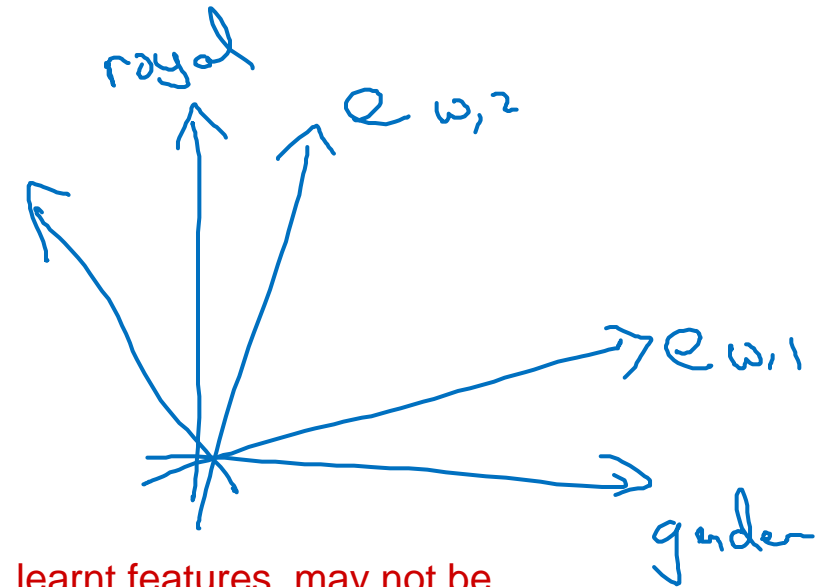
$$e_w^{(final)} = \frac{e_w + \Theta_w}{2}$$

unlike before, in this formulate, theta and e pay symmetric role, therefore after training we average them.

Andrew Ng

A note on the featurization view of word embeddings

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	
Gender	-1	1	-0.95	0.97	←
Royal	0.01	0.02	0.93	0.95	←
Age	0.03	0.02	0.70	0.69	←
Food	0.09	0.01	0.02	0.01	←



learnt features, may not be orthogonal or interpretable

$$\text{minimize } \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij}) (\underbrace{\theta_i^T e_j}_{\text{handwritten}} + b_i - b'_j - \log X_{ij})^2$$

$$\underbrace{(A \theta_i)^T (A^{-T} e_j)}_{\text{handwritten}} = \theta_i^T \cancel{A^T A} e_j$$



deeplearning.ai

NLP and Word Embeddings

Sentiment classification

Sentiment classification problem



The dessert is excellent.



Service was quite slow.



Good for a quick meal, but nothing special.



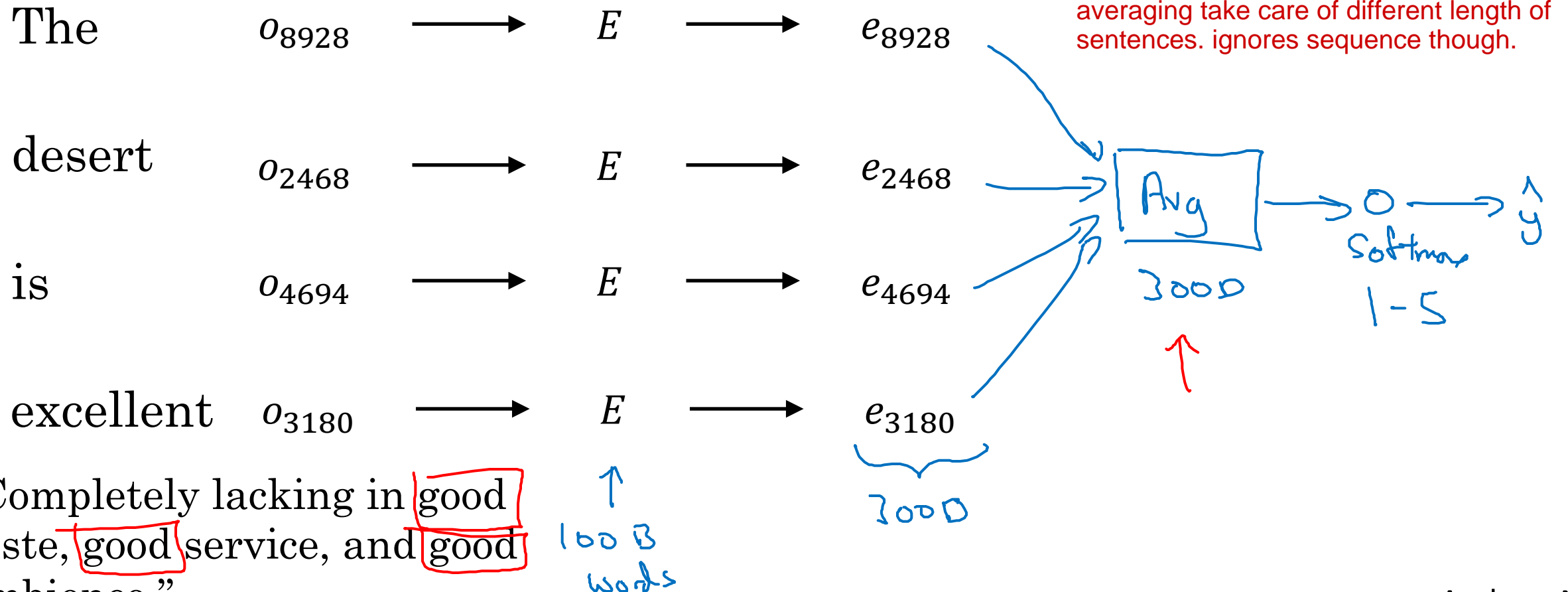
Completely lacking in good taste, good service, and good ambience.



10,000  100,000 words

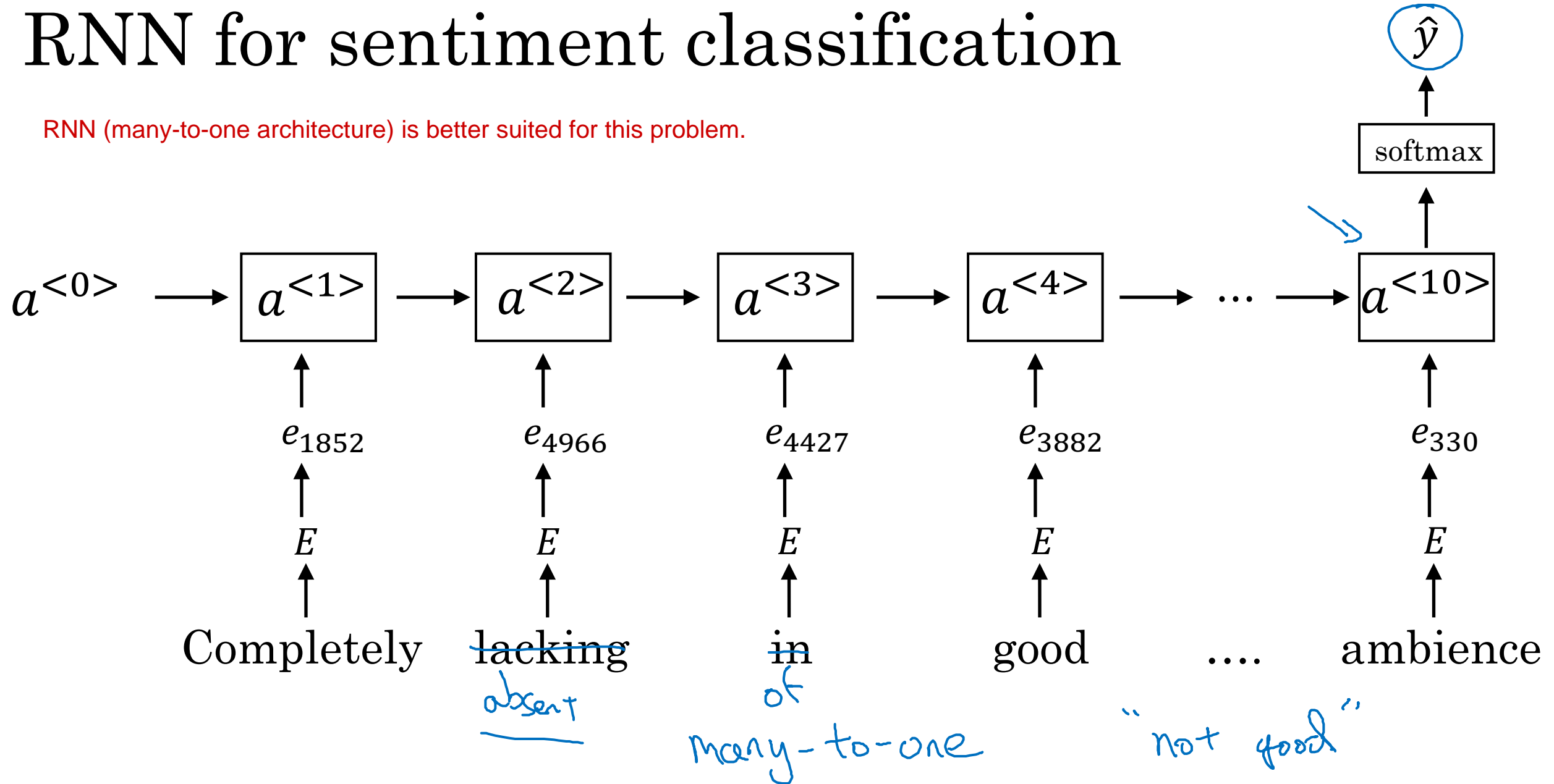
Simple sentiment classification model

The dessert is excellent
8928 2468 4694 3180



RNN for sentiment classification

RNN (many-to-one architecture) is better suited for this problem.





deeplearning.ai

NLP and Word Embeddings

Debiasing word embeddings

The problem of bias in word embeddings

Man:Woman as King:Queen

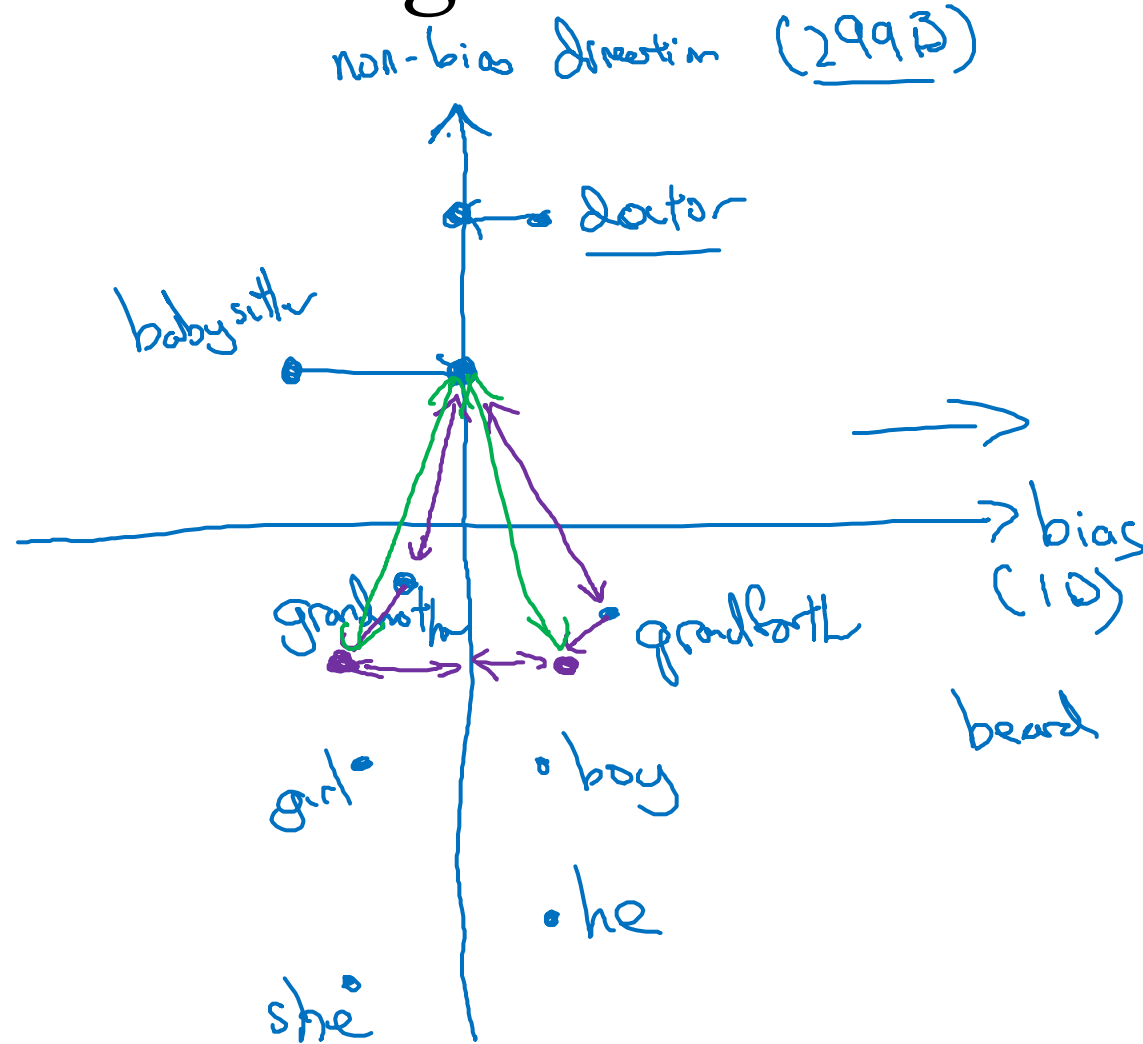
Man:Computer_Programmer as Woman:Homemaker X

Father:Doctor as Mother:Nurse X

Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model.



Addressing bias in word embeddings



1. Identify bias direction.

$$\begin{cases} e_{he} - e_{she} \\ e_{male} - e_{female} \\ \vdots \end{cases} \rightarrow \text{average}$$

2. Neutralize: For every word that is not definitional, project to get rid of bias.

3. Equalize pairs.

$$\rightarrow \begin{cases} \text{grandmother} - \text{grandfather} \\ \text{girl} - \text{boy} \end{cases}$$

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

Sequence to sequence models

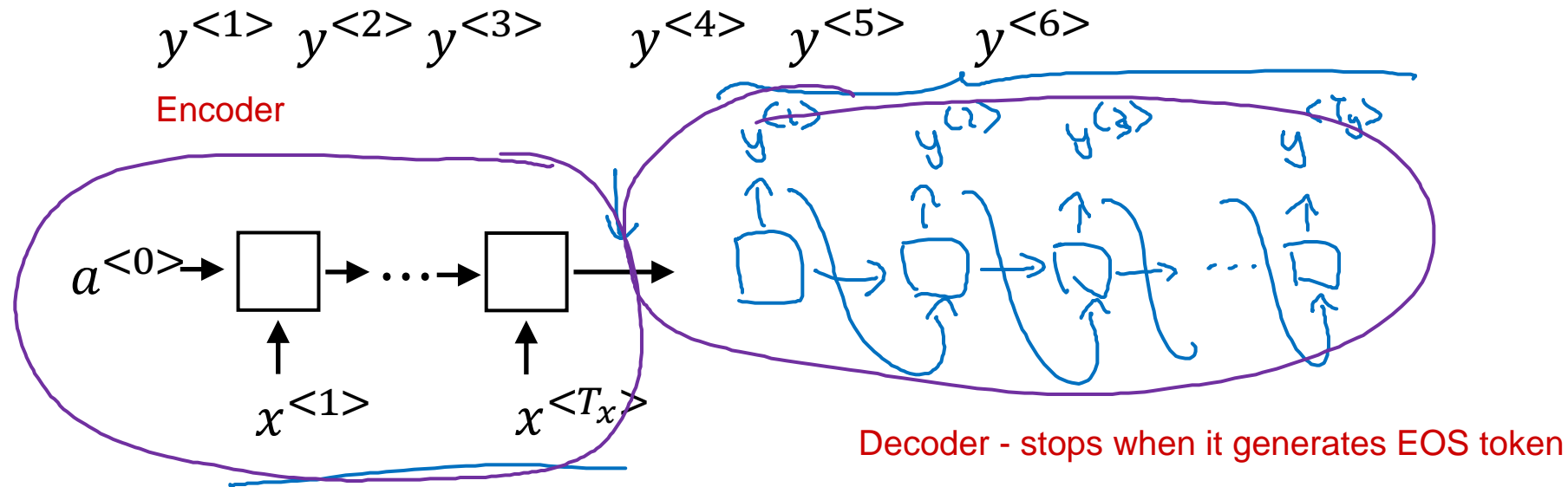
Basic models

Sequence to sequence model

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$
Jane visite l'Afrique en septembre

Remarkable thing is - this model works
when enough training data is applied

→ Jane is visiting Africa in September.

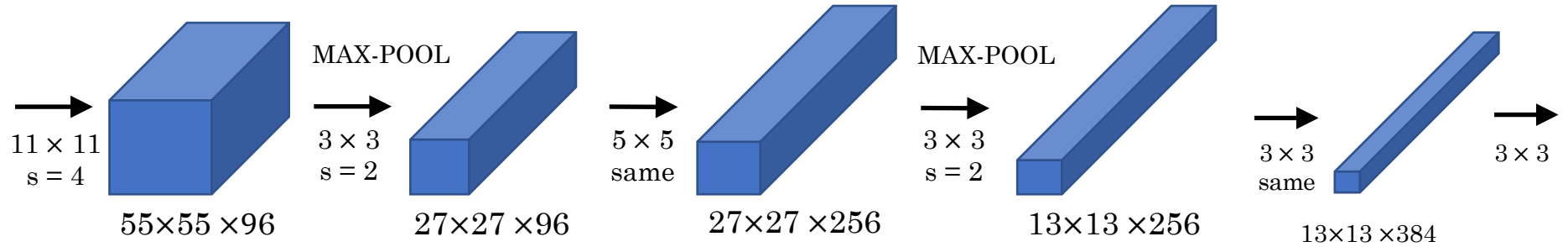
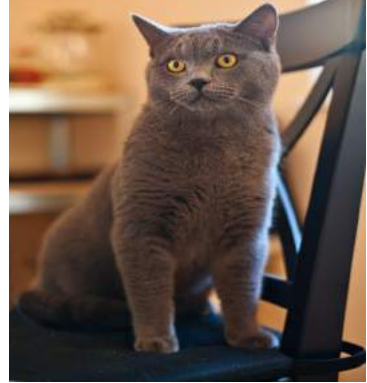


[Sutskever et al., 2014. Sequence to sequence learning with neural networks] ←

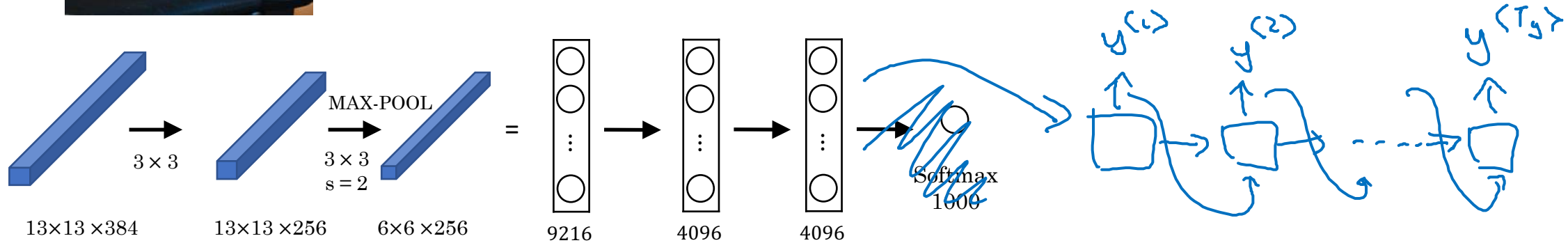
[Cho et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation] ←

Andrew Ng

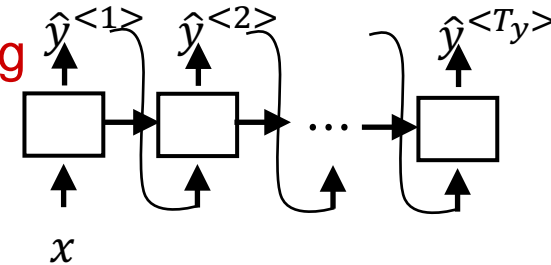
Image captioning



$y^{<1>} y^{<2>} y^{<3>} y^{<4>} y^{<5>} y^{<6>}$
 A cat sitting on a chair }



similar architecture to last slide also works for image captioning



[Mao et. al., 2014. Deep captioning with multimodal recurrent neural networks]

[Vinyals et. al., 2014. Show and tell: Neural image caption generator]

[Karpathy and Li, 2015. Deep visual-semantic alignments for generating image descriptions]



deeplearning.ai

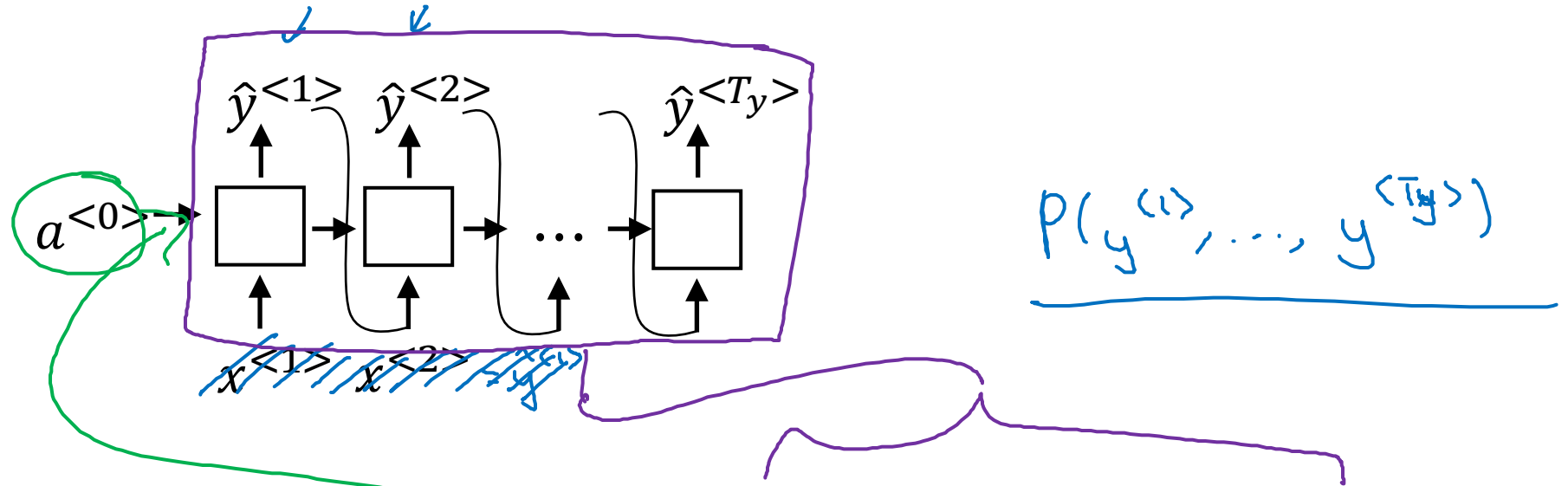
Sequence to sequence models

How to generate most likely sequence, not just a random sequence

Picking the most likely sentence

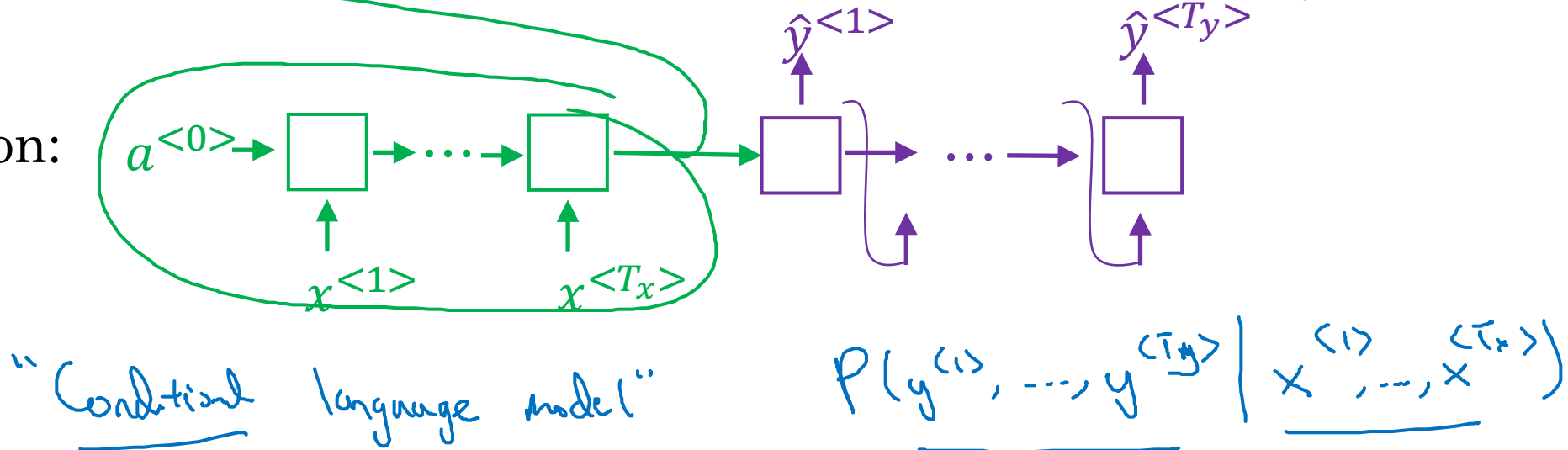
Machine translation as building a conditional language model

Language model:



Machine translation:

decoder network is similar to 'conditional language model' - conditional on input French sentence



Finding the most likely translation

Jane visite l'Afrique en septembre.

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

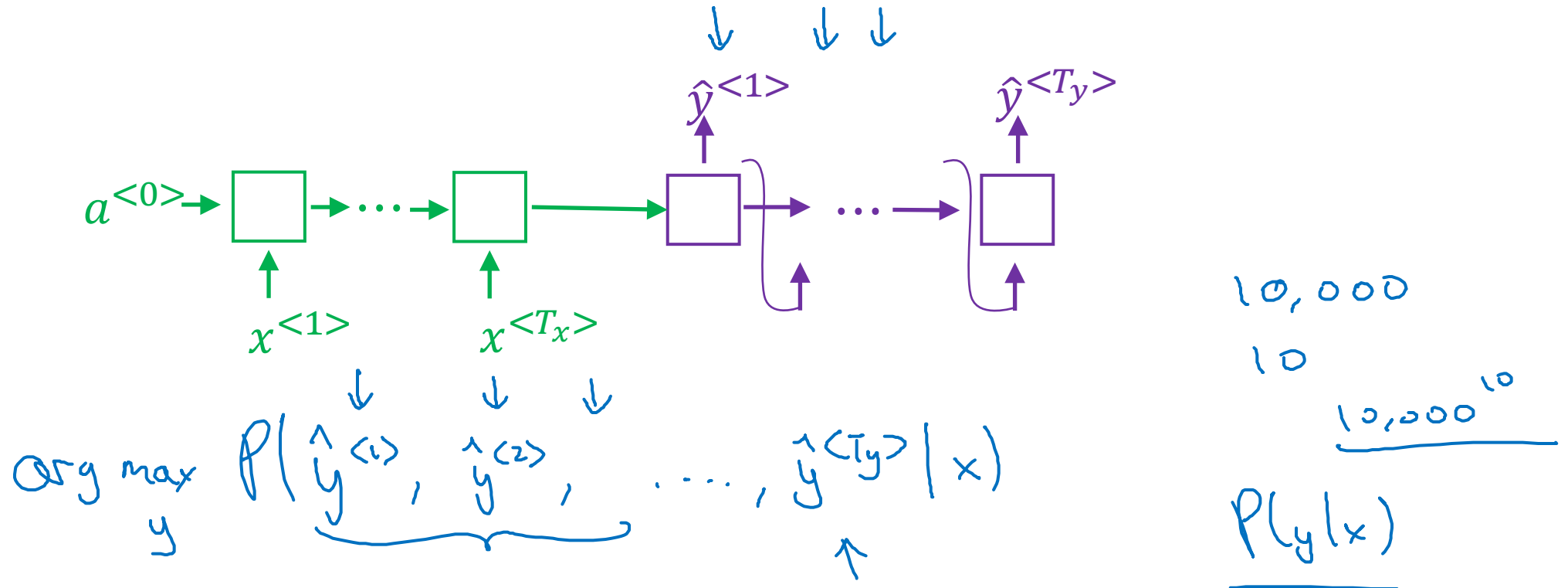
- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.
- In September, Jane will visit Africa.
- Her African friend welcomed Jane in September.

$$\arg \max_{y^{<1>}, \dots, y^{<T_y>}} \underbrace{P(y^{<1>}, \dots, y^{<T_y>} | x)}$$

How to maximize this OVERALL prob.??

- greedy search - pick the most likely word at each stage
- beam search

Why not a greedy search?



→ Jane is visiting Africa in September.

→ Jane is going to be visiting Africa in September.

$$P(\text{Jane is going} | x) > P(\text{Jane is visiting} | x)$$



deeplearning.ai

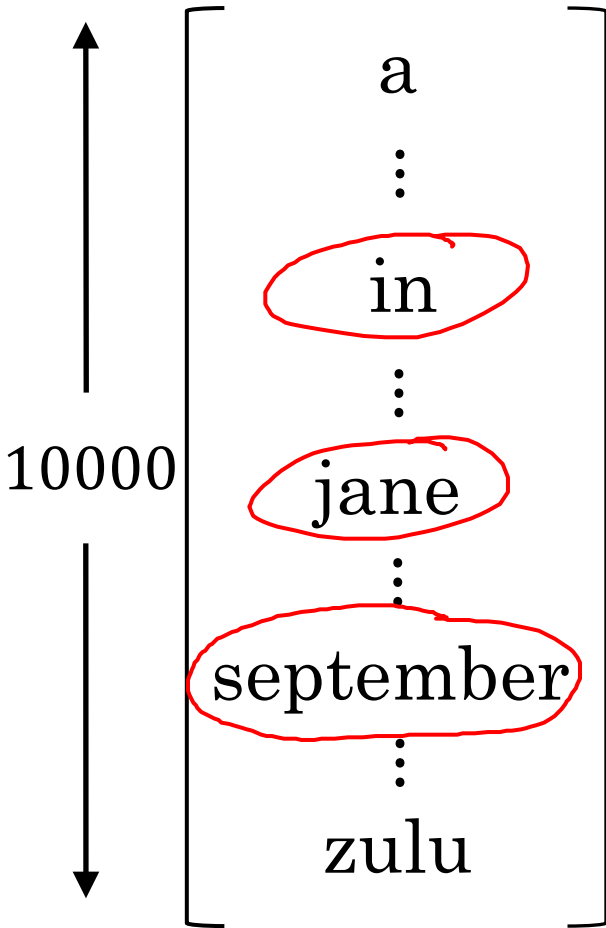
Sequence to sequence models

Beam search

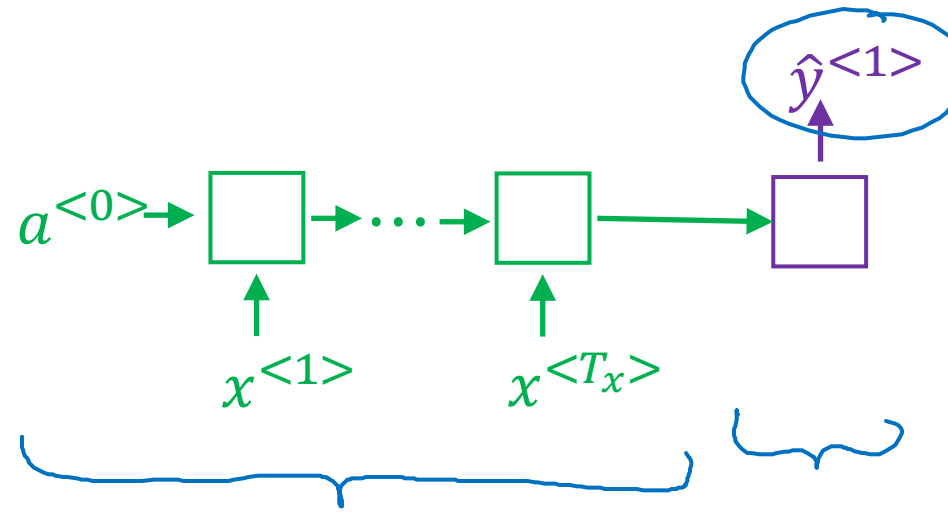
Beam search algorithm

$B = 3$ (beam width)

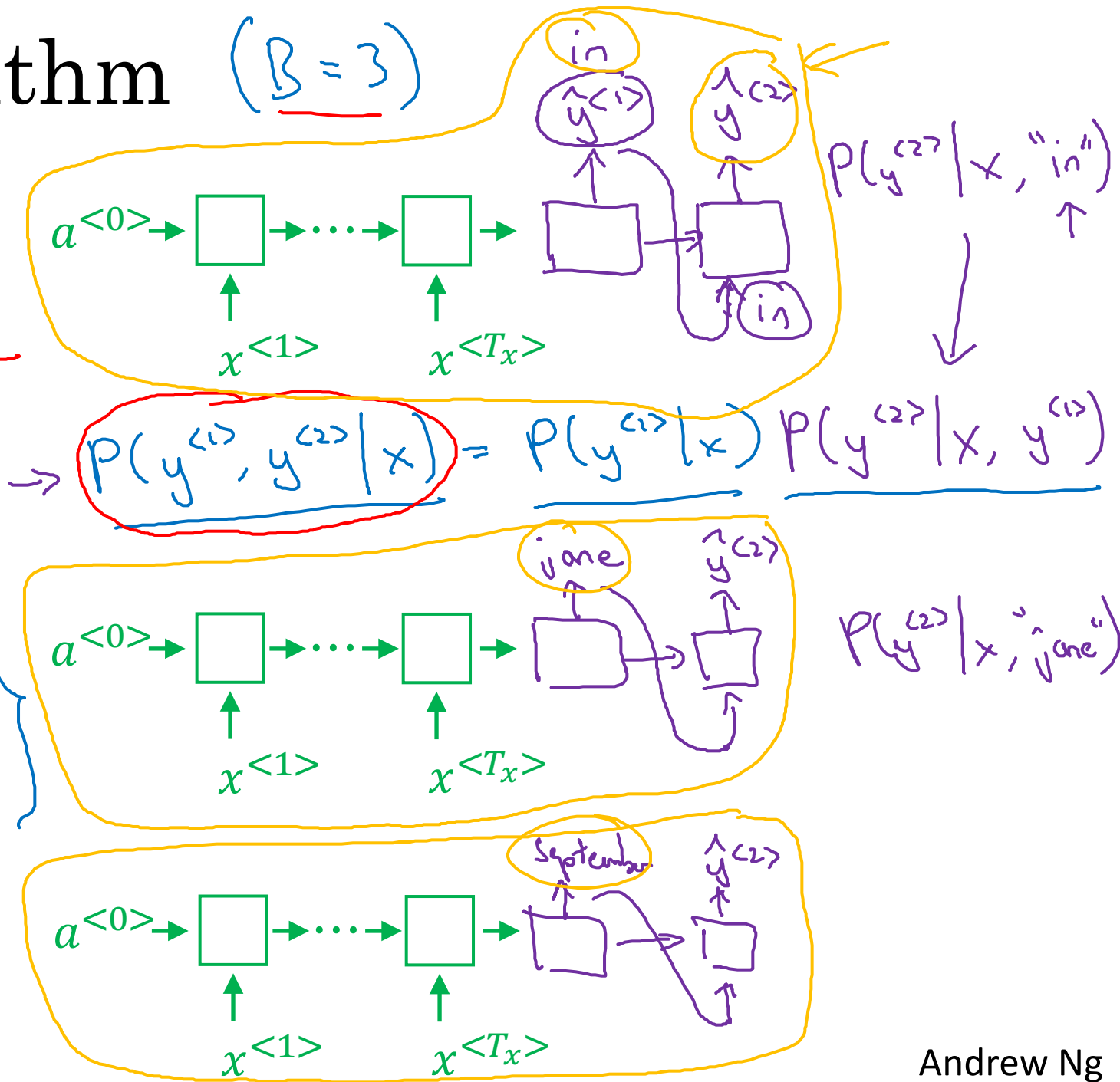
Step 1



$$\rightarrow \underline{P(y^{<1>} | x)}$$



- 3 best options are selected at each stage
- create 3 copies of the network at each stage

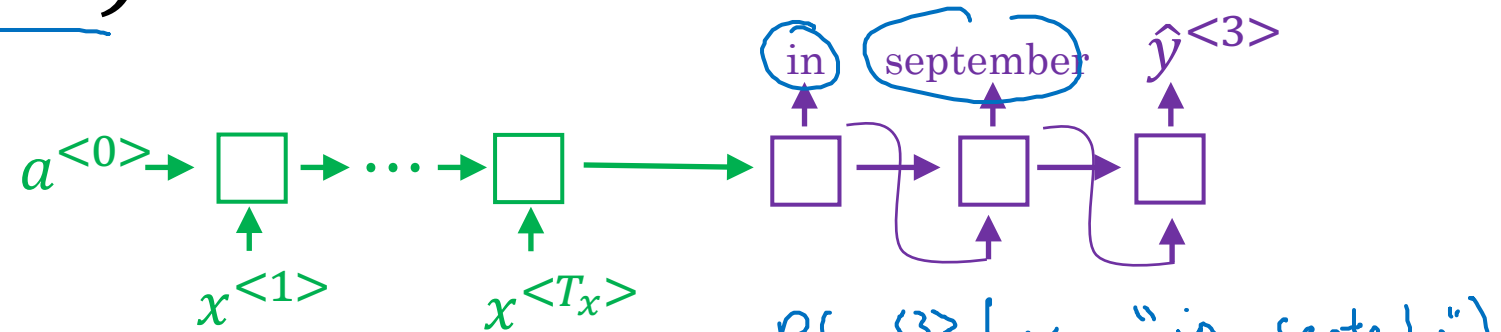
$$(B = 3)$$


Beam search ($B = 3$)

$B=1 \rightsquigarrow$ greedy search

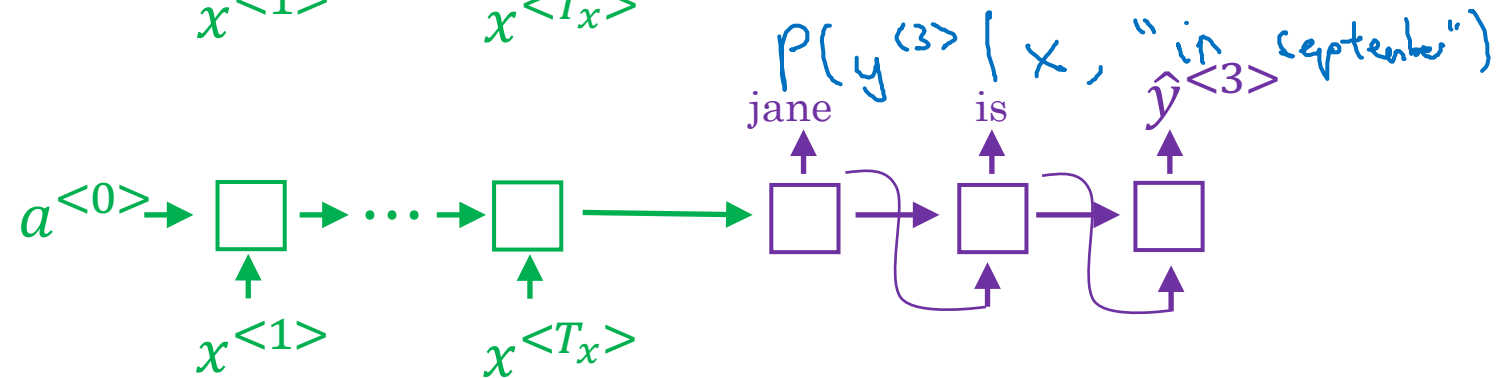
in september

a
aaron
jane
zulu



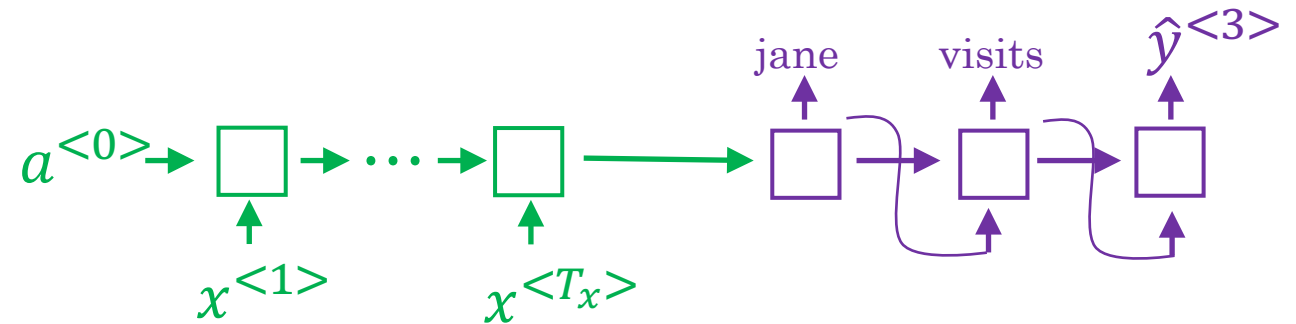
jane is

a
visits
zulu



jane visits

a
africa
zulu



$$P(y^{<1>}, y^{<2>} | x)$$

jane visits africa in september. <EOS>



deeplearning.ai

Sequence to sequence models

Refinements to beam search

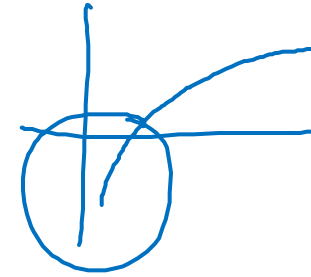
Length normalization

$$P(y^{(1)} \dots y^{(T_y)} | x) = \frac{P(y^{(1)} | x) P(y^{(2)} | x, y^{(1)}) \dots}{P(y^{(T_y)} | x, y^{(1)}, \dots, y^{(T_y-1)})}$$

$$\arg \max_y \prod_{t=1}^{T_y} P(y^{(t)} | x, y^{(1)}, \dots, y^{(t-1)})$$

log

$$\arg \max_y \sum_{t=1}^{T_y} \log P(y^{(t)} | x, y^{(1)}, \dots, y^{(t-1)})$$



$$\log P(y|x) \leftarrow$$

$$P(y|x) \leftarrow$$

Changes:

probabilities are small numbers
(1) Log changes it to addition, and reduces error from numerical precision

joint probabilities will reduce as more words are added

(2) normalize by # of words

$$T_y = 1, 2, 3, \dots, 30.$$

$$\rightarrow \frac{1}{T_y} \sum_{t=1}^{T_y} \log P(y^{(t)} | x, y^{(1)}, \dots, y^{(t-1)})$$

$$\underline{\alpha = 0.7}$$

$$\underline{\alpha = 1}$$

$$\underline{\alpha = 0}$$

Beam search discussion

Beam width B?

$1 \rightarrow 3 \rightarrow 10, \quad 100, \quad 1000 \rightarrow 3000$

large B: better result, slower
small B: worse result, faster

Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find exact maximum for $\arg \max_y P(y|x)$.



deeplearning.ai

Sequence to sequence models

Error analysis on beam search

Example

Jane visite l'Afrique en septembre.

→ RNN

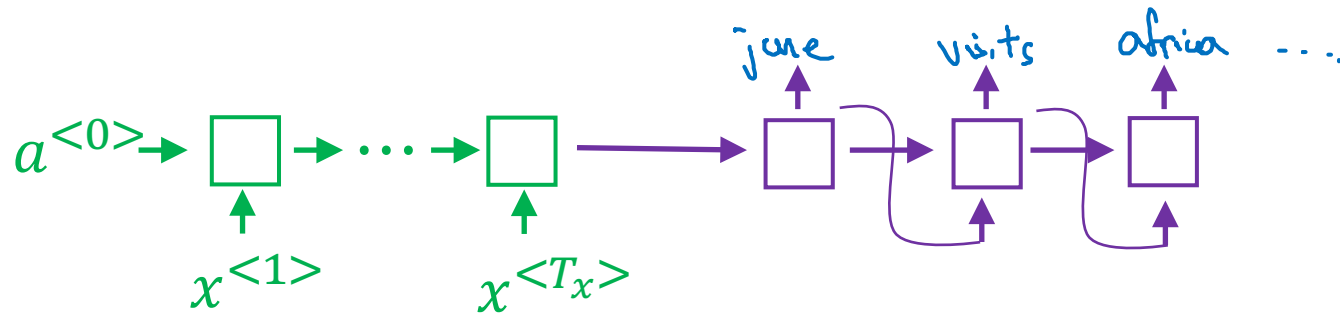
→ Beam Search

BT

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. (\hat{y}) ←

RNN computes $P(y^*|x) > P(\hat{y}|x)$



Error analysis on beam search

Human: Jane visits Africa in September. (y^*)

$$P(y^*|x)$$

Algorithm: Jane visited Africa last September. (\hat{y})

$$P(\hat{y}|x)$$

Case 1: $P(y^*|x) > P(\hat{y}|x)$ \leftarrow

$$\arg \max_y P(y|x)$$

Beam search chose \hat{y} . But y^* attains higher $P(y|x)$.

Conclusion: Beam search is at fault.

Same idea in ML course - calculate cost for model estimated theta and theta from another source - to see if cost function is faulty or optimization algo

Case 2: $P(y^*|x) \leq P(\hat{y}|x)$ \leftarrow

y^* is a better translation than \hat{y} . But RNN predicted $P(y^*|x) < P(\hat{y}|x)$.

Conclusion: RNN model is at fault.

Error analysis process

Human	Algorithm	$P(y^* x)$	$P(\hat{y} x)$	At fault?
Jane visits Africa in September. - - - ...	Jane visited Africa last September. - - - ...	$\frac{2 \times 10^{-10}}{\text{---}}$ ---	$\frac{1 \times 10^{-10}}{\text{---}}$ ---	<div>B</div> <div>R</div> <div>R</div> <div>R</div> <div>R</div> <div>...</div>

Figures out what fraction of errors are “due to” beam search vs. RNN model



deeplearning.ai

Sequence to sequence models

Bleu score (optional)

Evaluating machine translation

French: Le chat est sur le tapis.

Reference 1: The cat is on the mat. ←

Reference 2: There is a cat on the mat. ←

MT output: the the the the the the the.

Precision: 7 / 7
each word in o/p
appears in the
reference

Modified precision: 2/7
it appears 2 times in first word in
reference

Bleu
bilingual evaluation understudy
bilingual evaluation understudy

MT - machine translation

Bleu score on bigrams

instead of looking at isolated words - we look at pair of words (bigram / trigram etc.)

Example: Reference 1: The cat is on the mat. ←

Reference 2: There is a cat on the mat. ←

MT output: The cat the cat on the mat. ←

	Count	Count _{clip}	
the cat	2 ←	1 ←	
cat the	1 ←	0	4
cat on	1 ←	1 ←	<hr/>
on the	1 ←	1 ←	6
the mat	1 ←	1 ←	
	↑		

Bleu score on unigrams

Example: Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

→ MT output: The cat the cat on the mat. (\hat{y})

$$P_1, P_2 = 1.0$$

every P_n will be 1, if translation is perfect

$$p_1 = \frac{\sum_{unigram \in \hat{y}} \text{count}_{clip}(unigram)}{\sum_{unigram \in \hat{y}} \text{count}(unigram)}$$

unigram

2/7 from 2 slides back

$$p_n = \frac{\sum_{ngram \in \hat{y}} \text{count}_{clip}(ngram)}{\sum_{ngram \in \hat{y}} \text{count}(ngram)}$$

Bleu details

p_n = Bleu score on n-grams only

p_1, p_2, p_3, p_4

Combined Bleu score: $BP \exp\left(\frac{1}{4} \sum_{n=1}^4 p_n\right)$

BP = brevity penalty

brevity penalty - to penalize too short of a translation (which inherently can score better, hence the penalty)

$$BP = \begin{cases} 1 & \text{if } \text{MT_output_length} > \text{reference_output_length} \\ \exp(1 - \text{MT_output_length} / \text{reference_output_length}) & \text{otherwise} \end{cases}$$

$\exp(1 - \text{reference_output_length} / \text{MT_output_length})$

Bleu scored established an industry standard performance metric for these models - and allowed rapid advancements in this field.



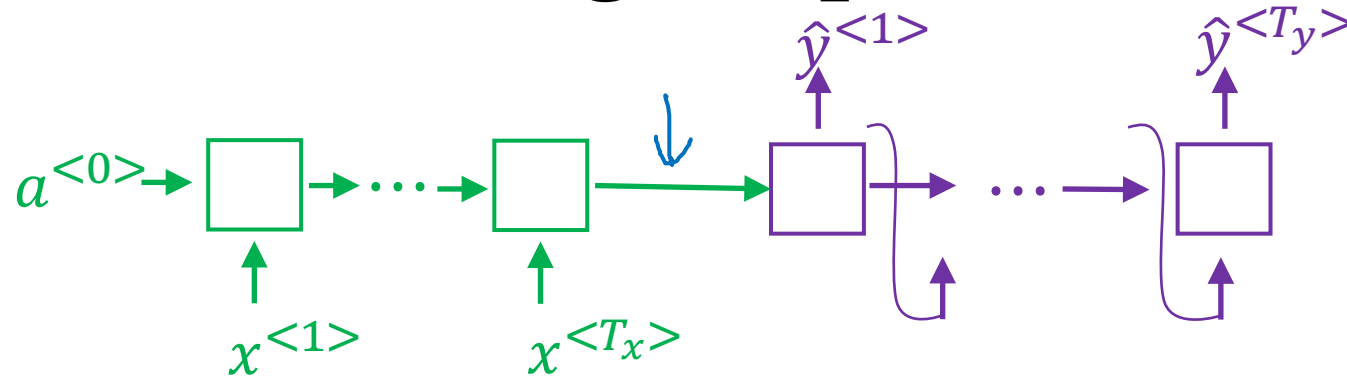
deeplearning.ai

Sequence to sequence models

One of the most influential ideas

Attention model intuition

The problem of long sequences

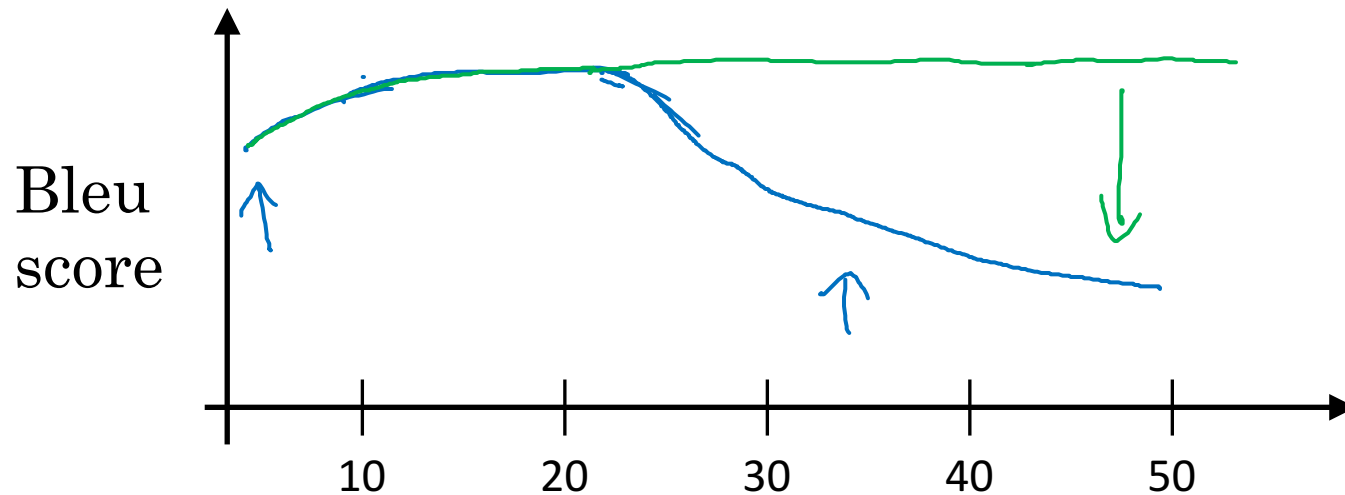


we're asking model to read in whole sentence and remember and store it in activations - so decoder can translate

not how human does it!
human would do it part by part

Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



conventional methods does poorly on longer sentences

attention model can give better performance (green line)

Sentence length

Attention model intuition

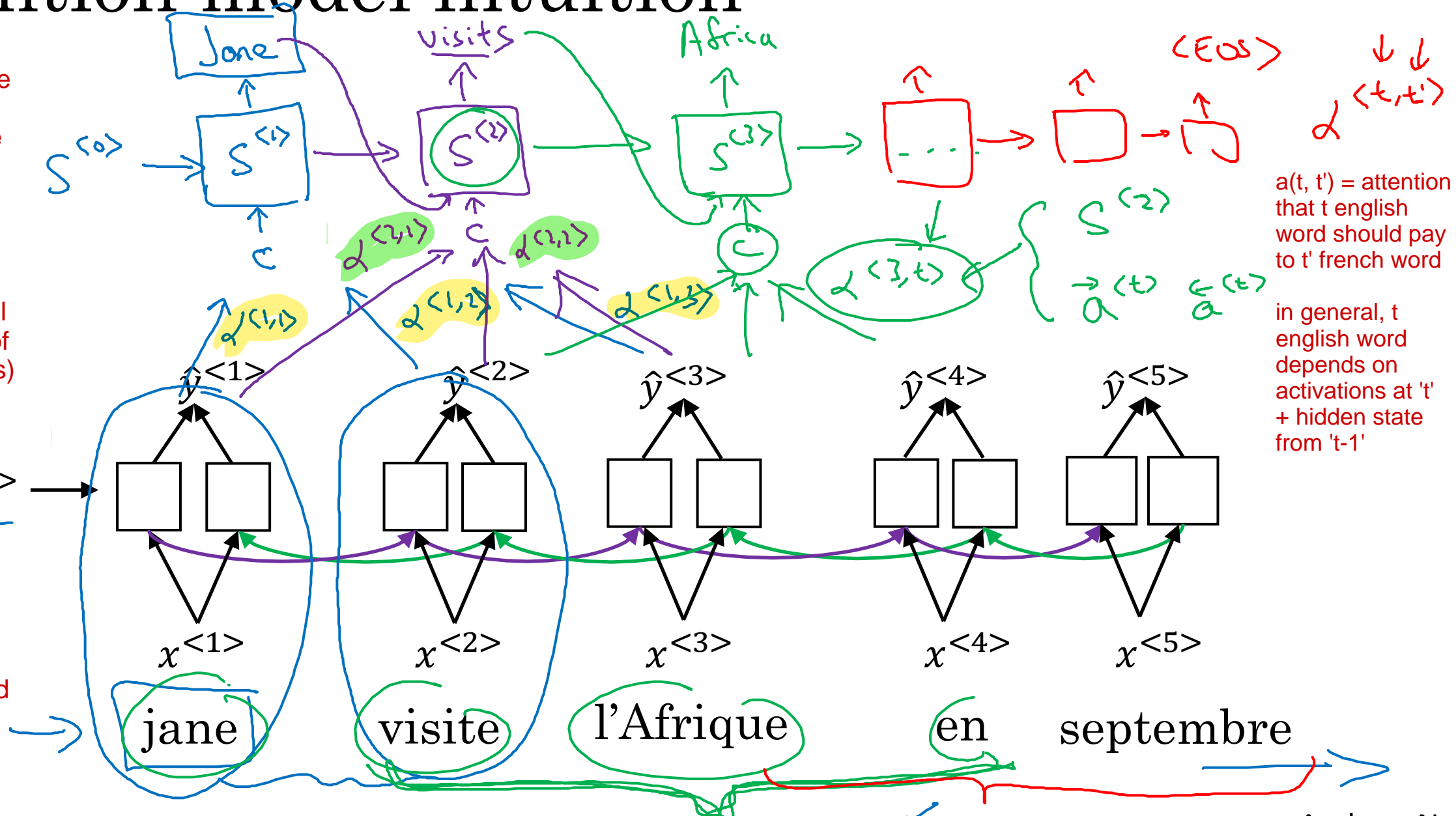
Pre-attention RNN + Post-attention RNN

For 1st word of translation to be Jane, we don't need the whole French sequence just the first few words.

Attention model computes set of weights (alphas)

next o/p word will have different weights and so on.

c denotes context that is input to o/p RNN (explained later)



$a(t, t')$ = attention that t english word should pay to t' french word

in general, t english word depends on activations at t' + hidden state from $t-1$



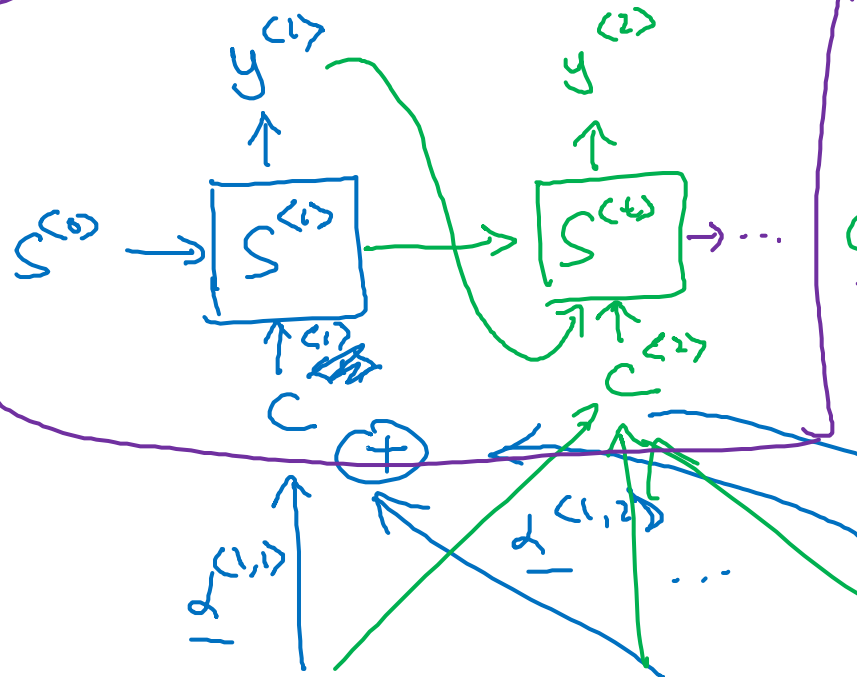
deeplearning.ai

Sequence to sequence models

Attention model

Attention model

single-dir
ection



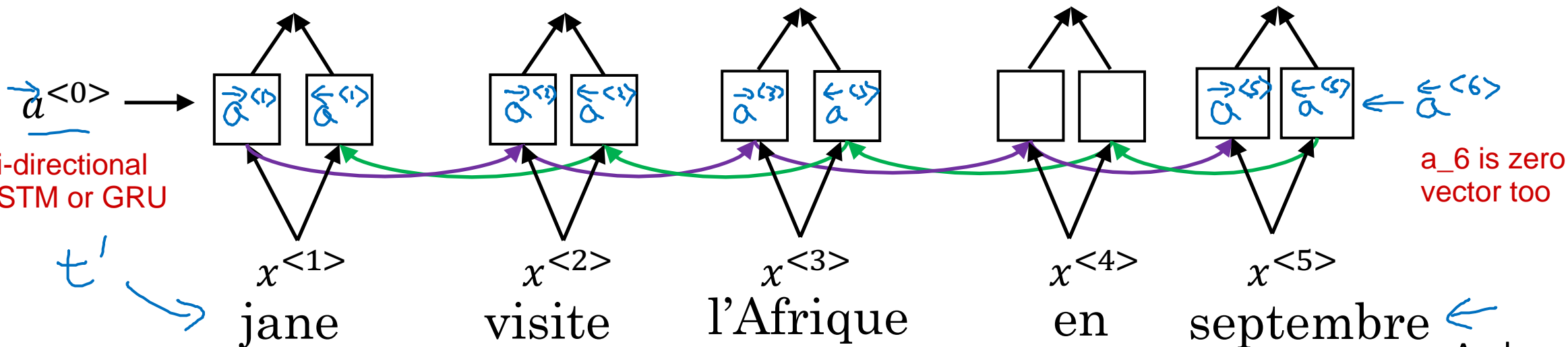
$\alpha^{(t,t')}$ = amount of 'attention' $y^{(t)}$ should pay to $a^{(t')}$.

$$c^{(2)} = \sum_{t'} \alpha^{(2,t')} a^{(t')}$$

$$a^{(t')} = (\vec{a}^{(t')}, \leftarrow{a}^{(t')})$$

$$\sum_{t'} \alpha^{(1,t')} = 1 \quad \sim \text{weighted average of attentions}$$

$$c^{(1)} = \sum_{t'} \alpha^{(1,t')} a^{(t')}$$



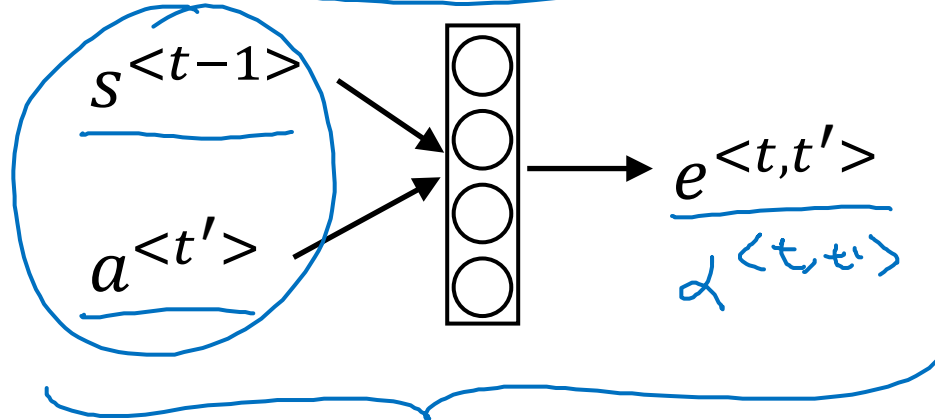
Computing attention $\alpha^{<t,t'>}$

T_x T_y

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

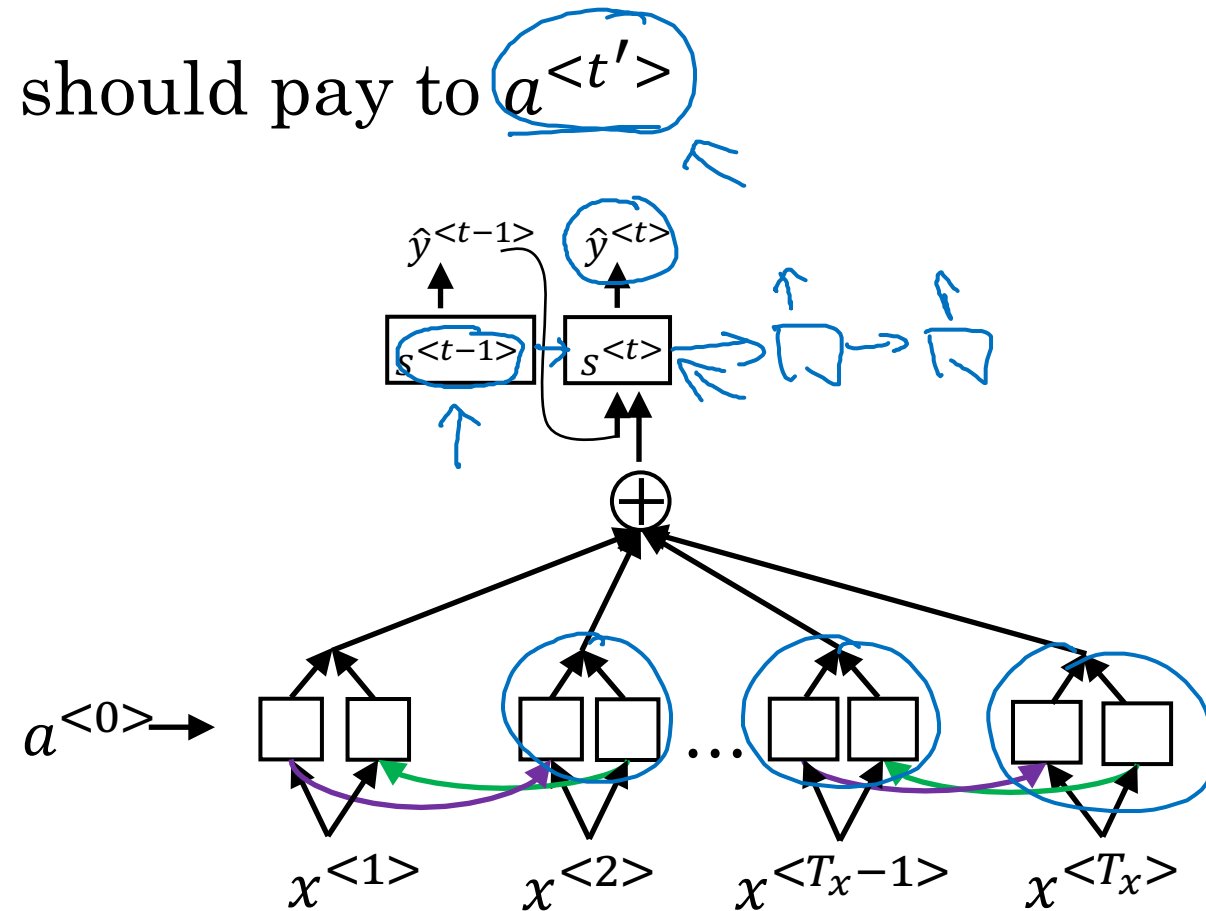
softmax config so all weights add to 1

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$



how to determine function 'e'

- we train a very small / 1-layer network for this
- train every (T_x, T_y) pair - quadratic computation time



[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

[Xu et. al., 2015. Show, attend and tell: Neural image caption generation with visual attention]

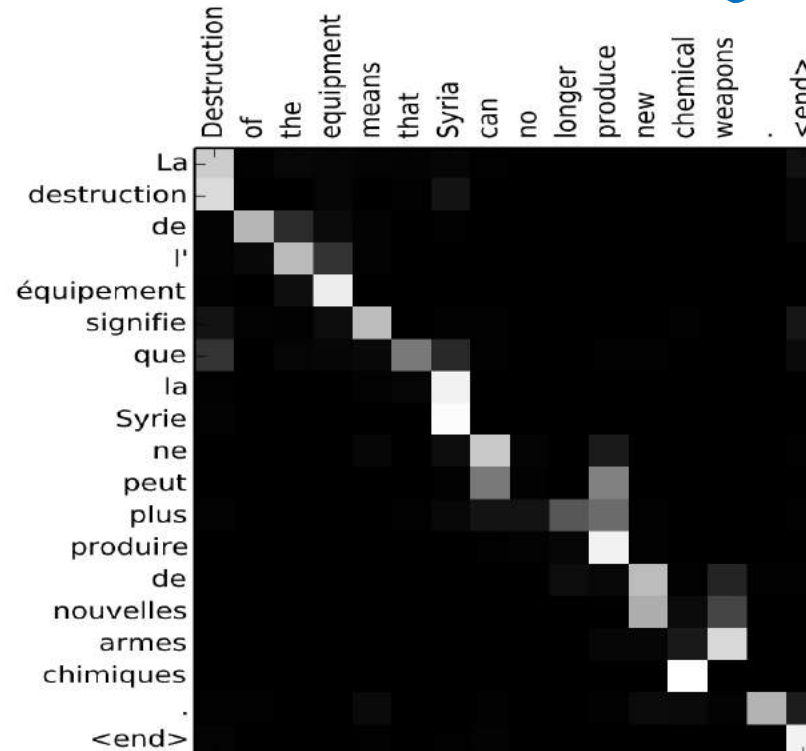
Andrew Ng

Attention examples

July 20th 1969 → 1969 – 07 – 20

23 April, 1564 → 1564 – 04 – 23

Visualization of $\alpha^{<t,t'>}$:





deeplearning.ai

Audio data

Speech recognition

Speech recognition problem

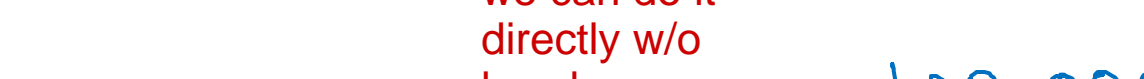
x

audio clip



y

transcript



previously,
modeled as
phonemes -
hand engineered

with end-to-end
deep learning
we can do it
directly w/o
hand
engineering

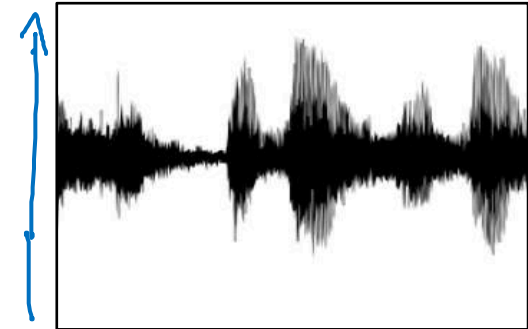
“the quick brown fox”

phonemes: de kwi br au n

300h

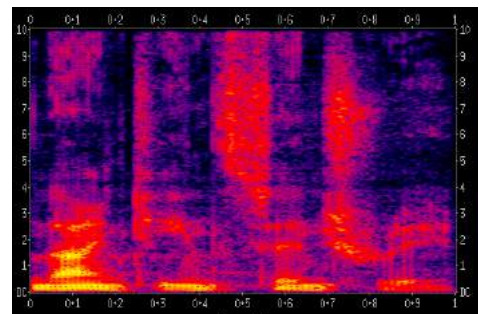
3000h

100,000h



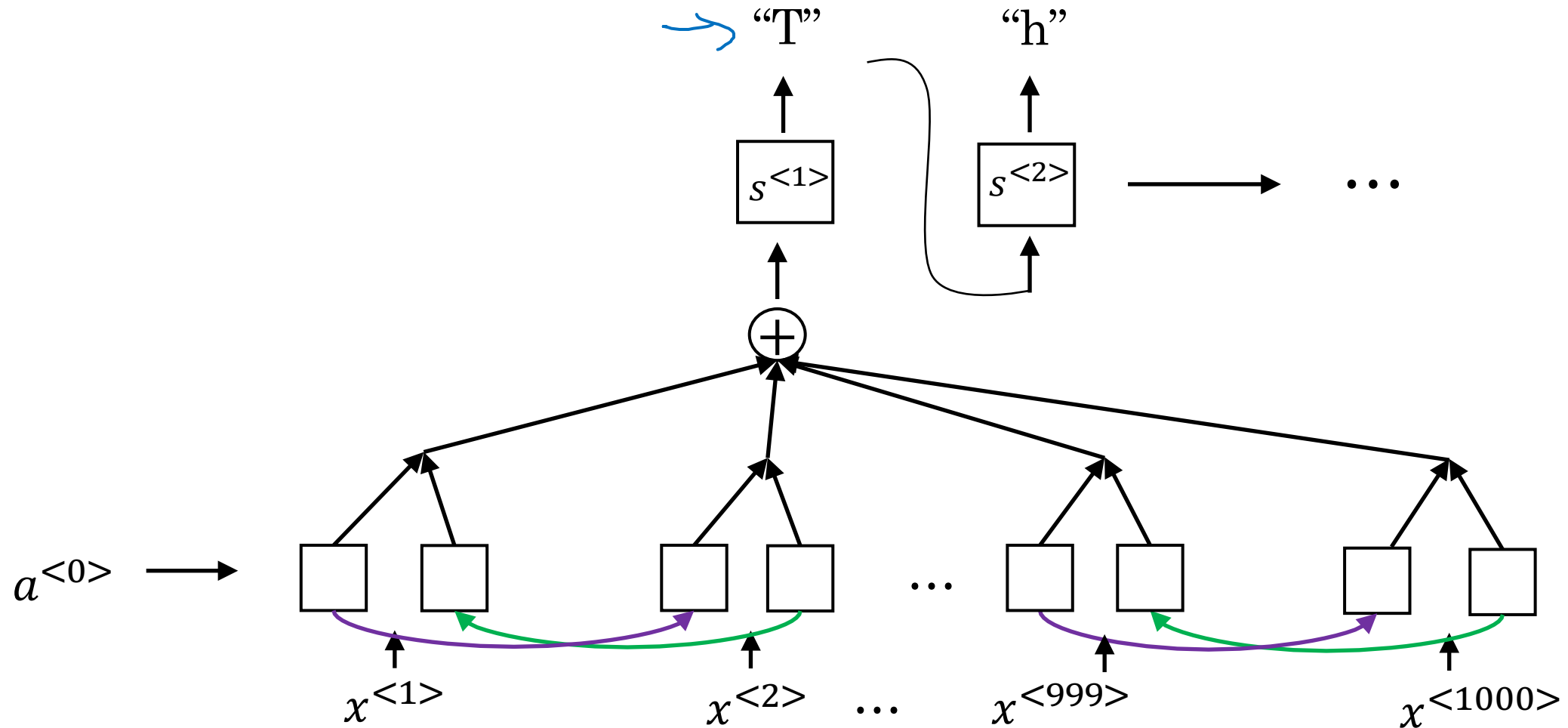
Spectrogram - first preprocessing step

color = how loud



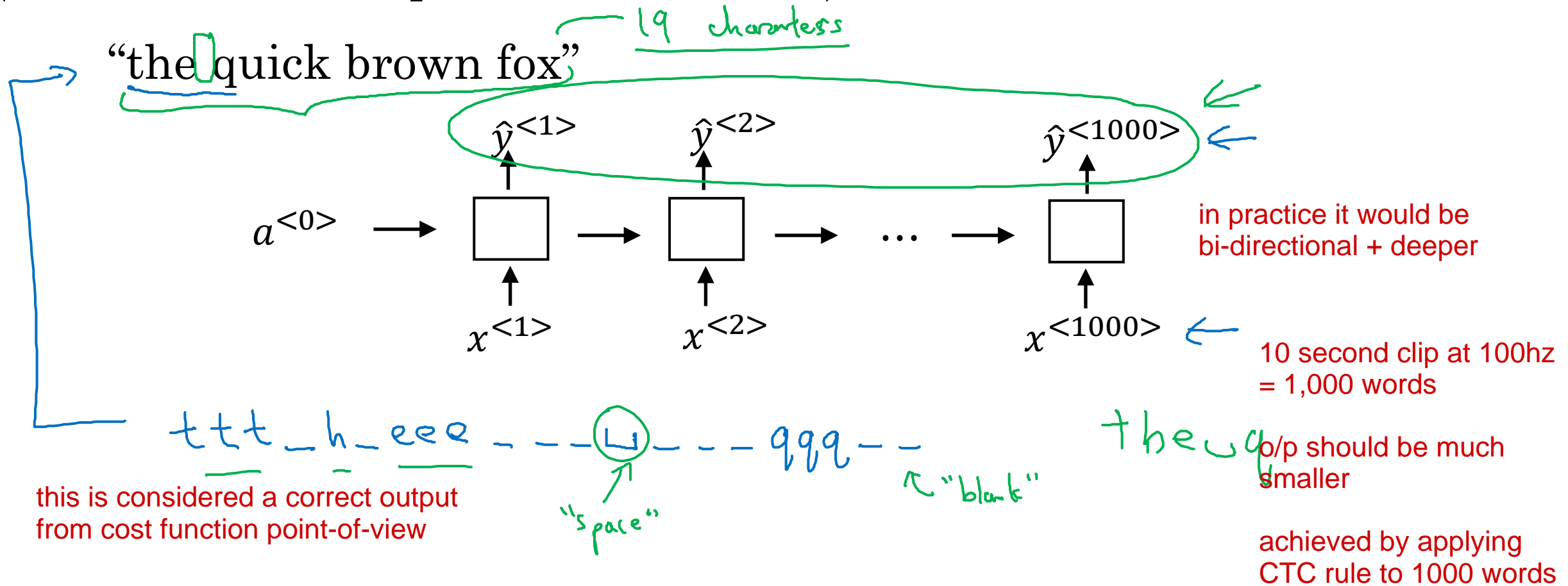
time

Attention model for speech recognition



CTC cost for speech recognition

(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by “blank”



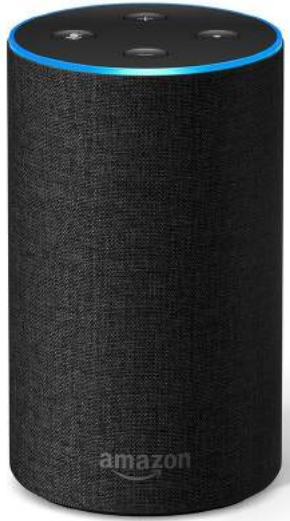
deeplearning.ai

Audio data

Trigger word
detection

What is trigger word detection?

research on trigger word detection
is still evolving and no consensus
on best approach



Amazon Echo
(Alexa)



Baidu DuerOS
(xiaodunihao)



Apple Siri
(Hey Siri)



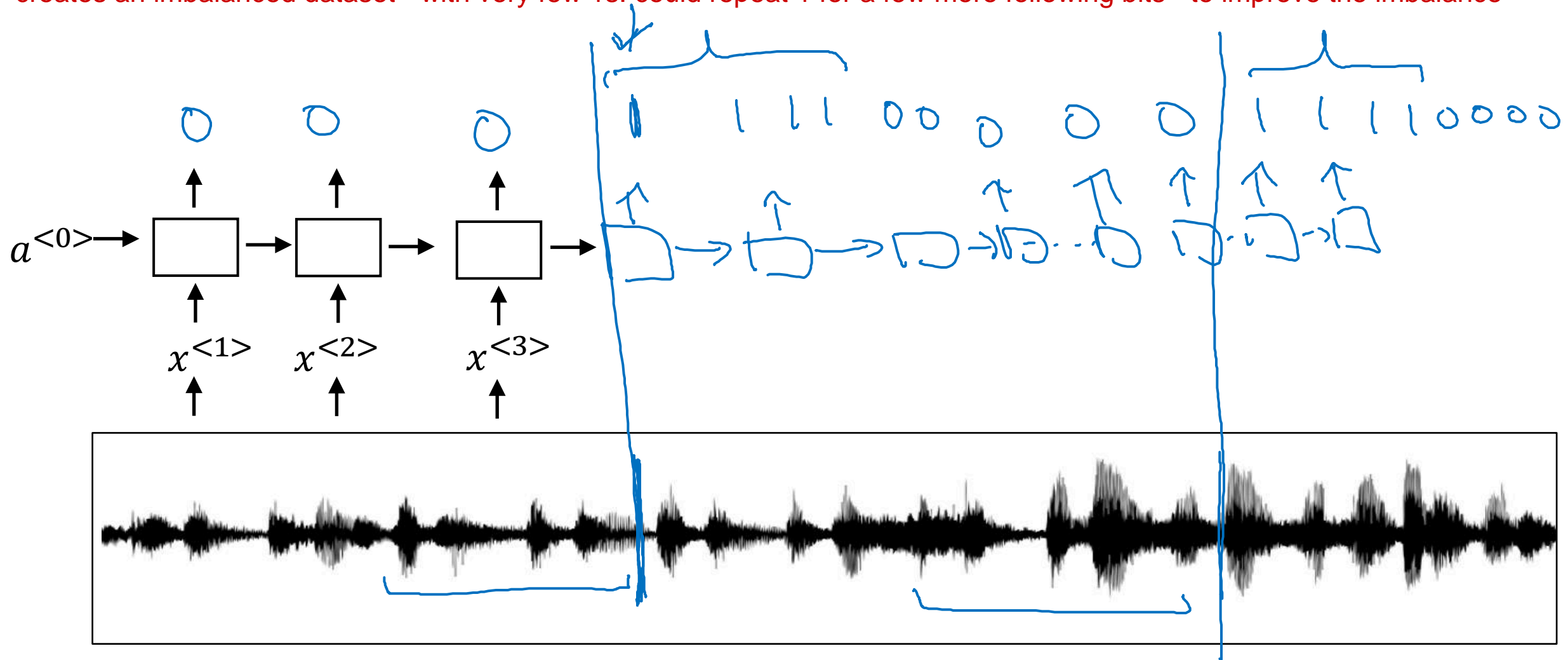
Google Home
(Okay Google)

Trigger word detection algorithm

simple approach:

use a RNN as before. training set: has 1 - right after the user finished saying the trigger word

creates an imbalanced dataset - with very few 1s. could repeat 1 for a few more following bits - to improve the imbalance





deeplearning.ai

Conclusion

Summary and thank you

Specialization outline

1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring Machine Learning Projects
4. Convolutional Neural Networks
5. Sequence Models

Deep learning is a super power

Please buy this
from shutterstock
and replace in
final video.



www.shutterstock.com · 331201091

Thank you.

- Andrew Ng

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

Sequence to sequence models

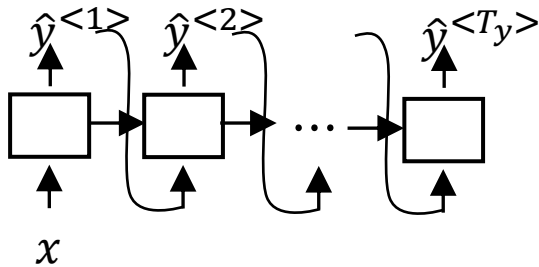
Transformers Intuition

Transformers Motivation

all these models are still 'sequential'
because all units had to be read in 1-by-1

Increased complexity,
sequential

RNN

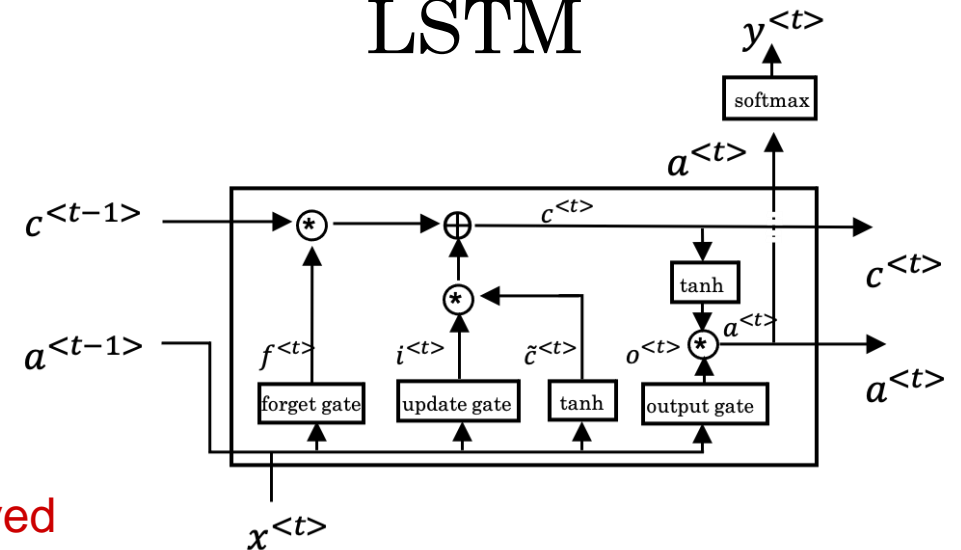


RNN had issues with training
for long term dependencies
due to vanishing gradients

GRU

GRU / LSTM solved
this by adding gates
and memory cell

LSTM

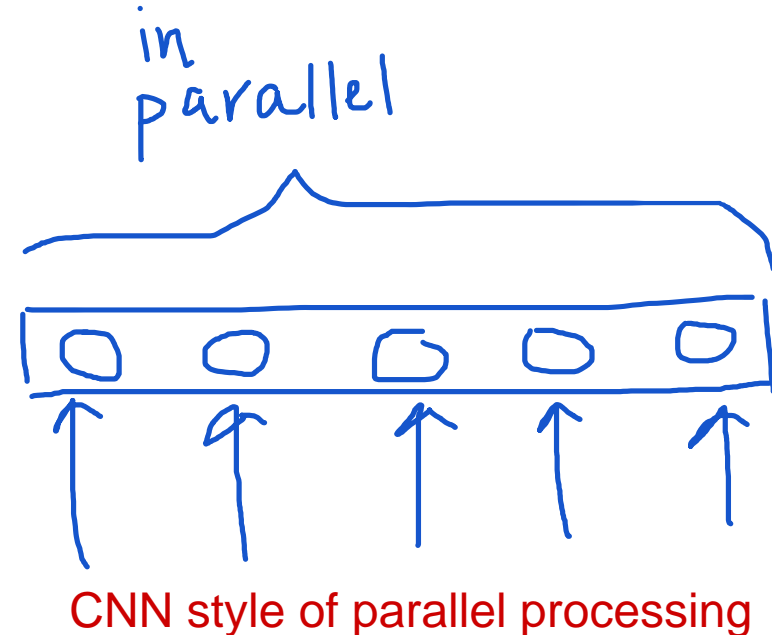
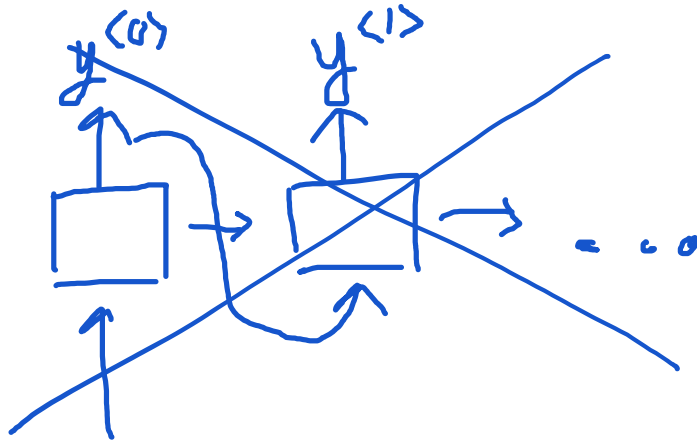


Transformers Intuition

- Attention + CNN

- Self-Attention if our sentence has 5 words - we compute 5 representations for these 5 words:
 $A_{<1>}$ to $A_{<5>}$
- Multi-Head Attention ~ FOR Loop over self-attention

sequential way of processing tokens





deeplearning.ai

Sequence to sequence models

Self-Attention

Self-Attention Intuition

$A(q, K, V)$ = attention-based vector representation of a word

↪ calculate for each word

Q-query, K-key, V-value

RNN Attention

$$\alpha^{<t, t'>} = \frac{\exp(e^{<t, t'>})}{\sum_{t'=1}^T x \exp(e^{<t, t'>})}$$

say we are calculating $A^{<3>}$ for l'Afrique

- previously we would just use a straight word embedding
- now we'll also include context - are we thinking of Africa as a holiday destination, or a place of historical significance, or second largest continent? The model will look at words around it, and use those to come up with a vector

$x^{<1>}$

Jane

$x^{<2>}$

visite

$x^{<3>}$

l'Afrique

$x^{<4>}$

en

$x^{<5>}$

septembre

Transformers Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{<q \cdot k^{<i>}>})}{\sum_j \exp(e^{<q \cdot k^{<j>}>})} v^{<i>}$$

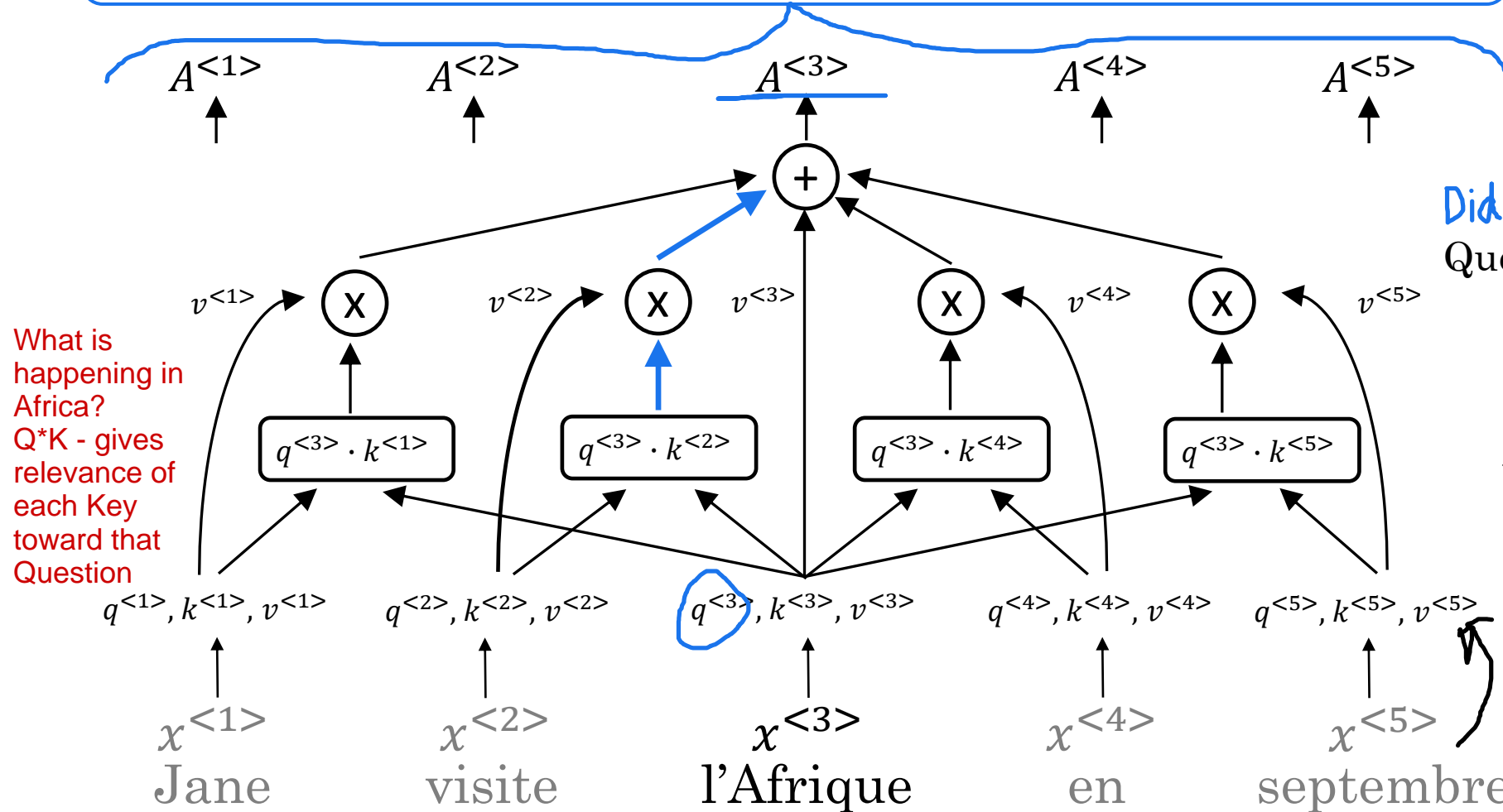
Self-Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{< q \cdot k^{<i>} >})}{\sum_j \exp(e^{< q \cdot k^{<j>} >})} v^{<i>}$$

softmax

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

vectorized representation
denominator is a scaling
variable





deeplearning.ai

Sequence to sequence models

Multi-Head Attention

Multi-Head Attention

Each self-attention for a sequence is called 'head'
Essentially a big FOR-LOOP over the self-attention mechanism

another W_0 here!

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Go through the loop 1st time -
same as last slide - "What is
going on there" - 'visit' may give
highest value

Then repeat it again for total 8
times - WHY??
- second time the network asks a
different question, e.g. "When is
it happening" - 'september' may
give highest value

- third question - "WHO" - 'jane'
may have highest value

Each head is a different "feature"

$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1 \text{head}_2 \dots \text{head}_n) W_0$$

$$\text{head}_i = \text{Attention}(W_i^Q Q, W_i^K K, W_i^V V)$$

concat

$$\text{Attention}(W_i^Q Q, W_i^K K, W_i^V V)$$

Multi-Head
Attention

Q K V

$W_3^Q, W_3^K, W_3^V, \text{Who?}$

$W_2^Q, W_2^K, W_2^V, \text{When?}$

$W_1^Q, W_1^K, W_1^V, \text{Did what?}$

W^Q, W^K, W^V

$q^{<1>}, k^{<1>}, v^{<1>}$

$x^{<1>}$

Jane

$q^{<2>}, k^{<2>}, v^{<2>}$

$x^{<2>}$

visite

$q^{<3>}, k^{<3>}, v^{<3>}$

$x^{<3>}$

l'Afrique

$q^{<4>}, k^{<4>}, v^{<4>}$

$x^{<4>}$

en

$q^{<5>}, k^{<5>}, v^{<5>}$

$x^{<5>}$

septembre



deeplearning.ai

Sequence to sequence models

Transformers

Transformer Details

<SOS> Jane visits Africa in September <EOS>

Encoder

Decoder

Why is this block repeated?

