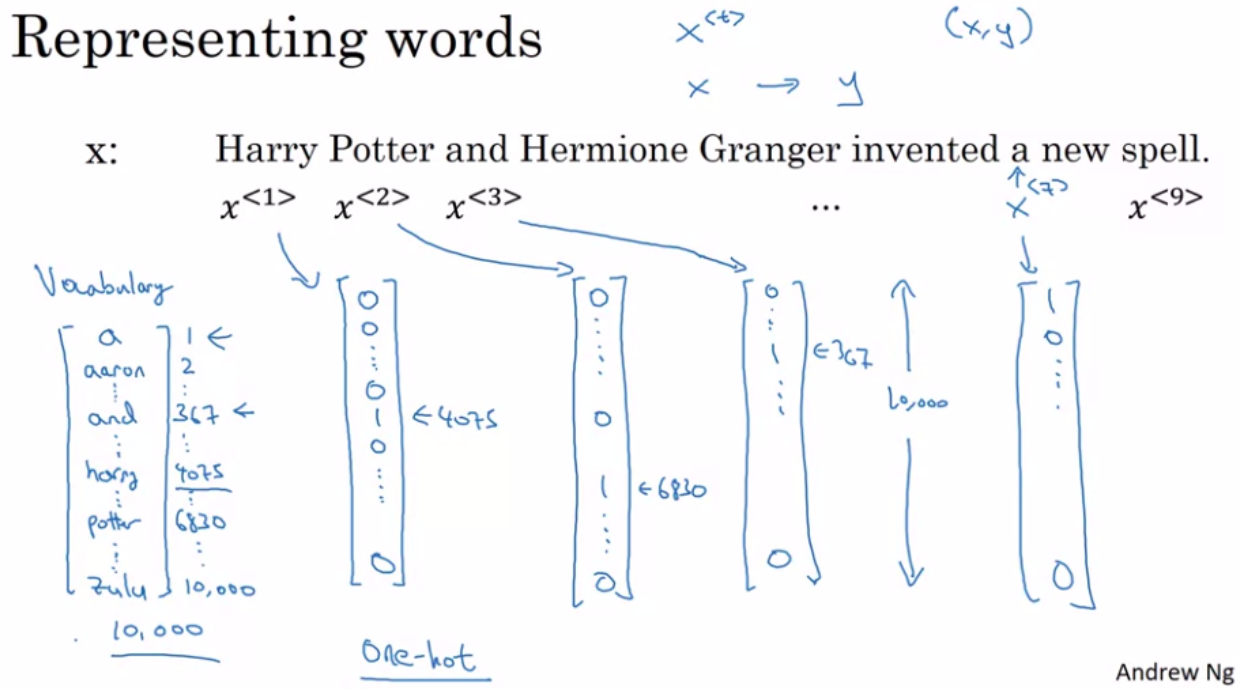
W1L1: Recurrent Neural Networks: why sequence models?

- Examples:
    1. Speech Recognition: Speech to Text
    2. Music Generation: Empty set or music genre as an input to generate music
    3. Sentiment Analysis: textual review to the rating(number of stars out of 5)
    4. DNA sequence analysis: finding the particular type of DNA from the given DNA sequence
    5. Machine translation: linguistic translations
    6. Video activity recognition
    7. Name entity recognition: finding people in the sentences.

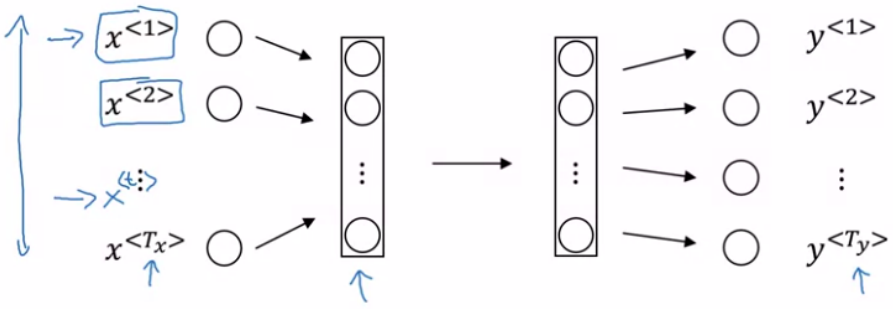
W1L2: Notations

- $X^{(i)<t>}$: i^{th} training sample with the t^{th} sequence element
- $Tx^{(i)}$: input sequence length of the i^{th} training sample. (length can be different for the training examples)
- Similarly for y
- Vocabulary or dictionary in case of words
- One hot representation



W1L3: RNN

- Why standard NN would not work in case of sequence models:

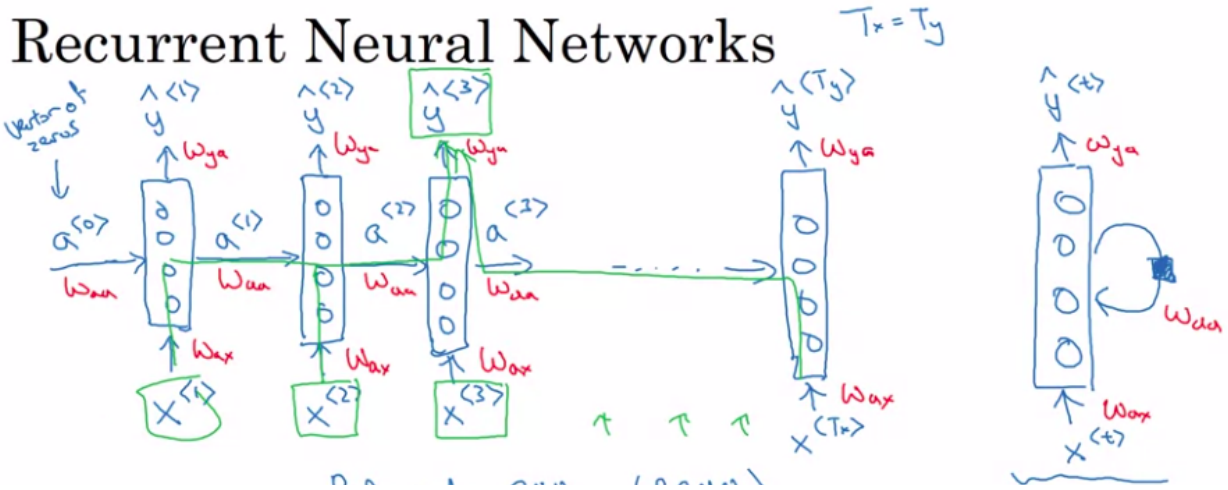


Problems:

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

Andrew Ng

Recurrent Neural Networks

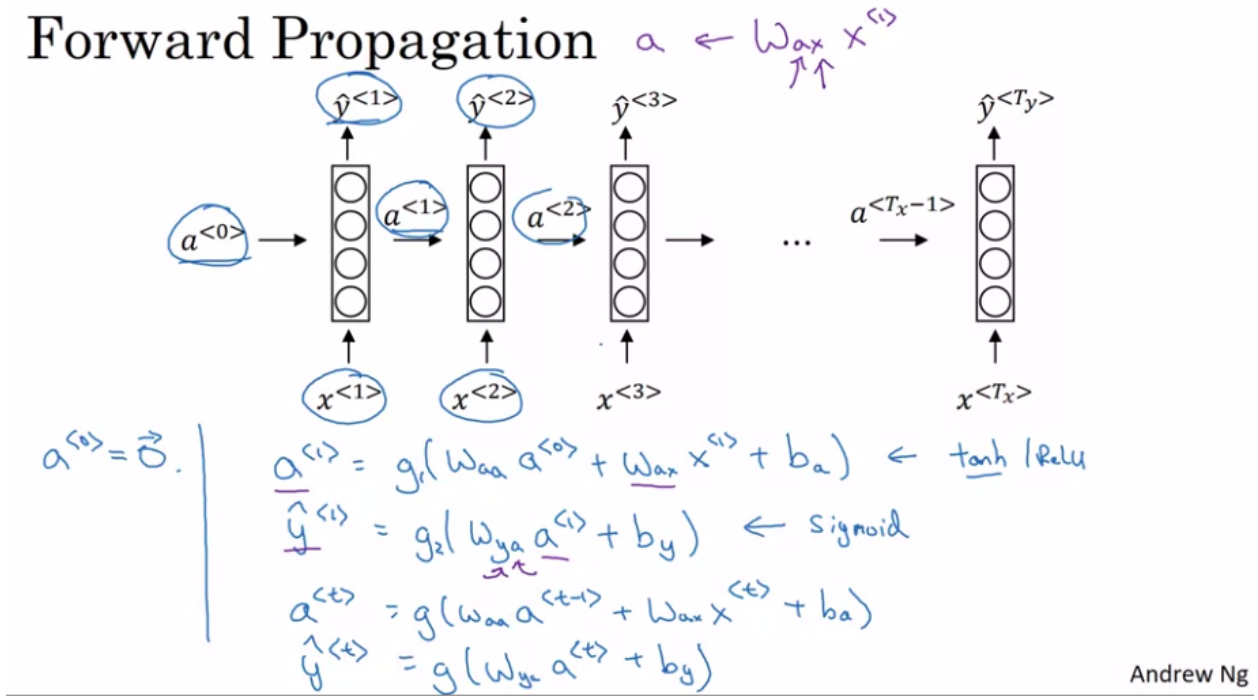


Limitation: only used previous words for the predictions. Due to this, in the second sentence, Teddy will be identified as the name of a person while it's not.

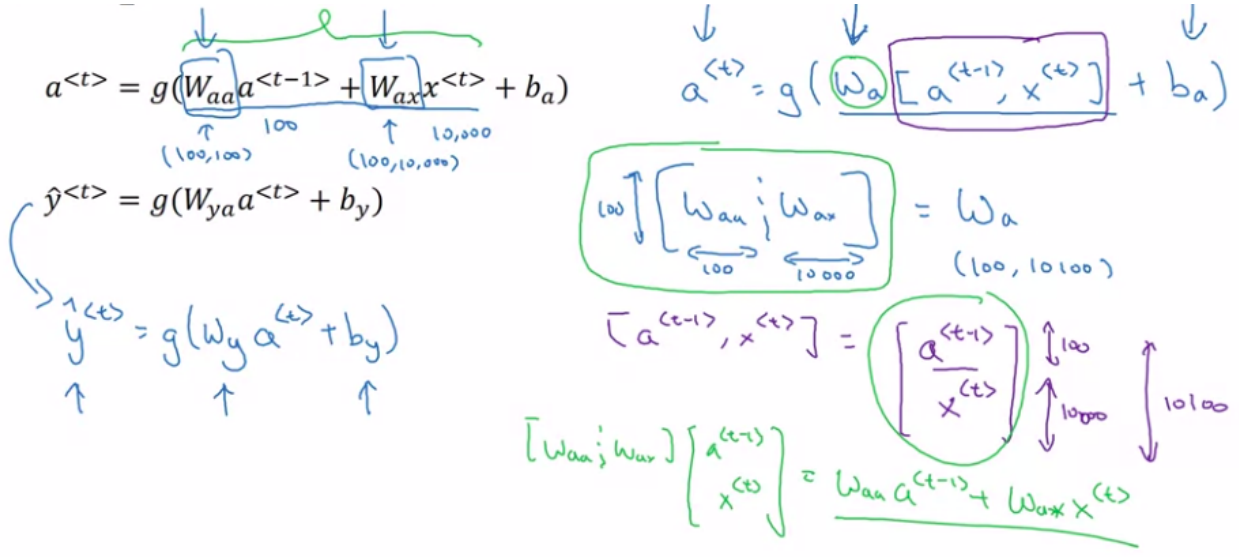
He said, "Teddy Roosevelt was a great President."

He said, "Teddy bears are on sale!"

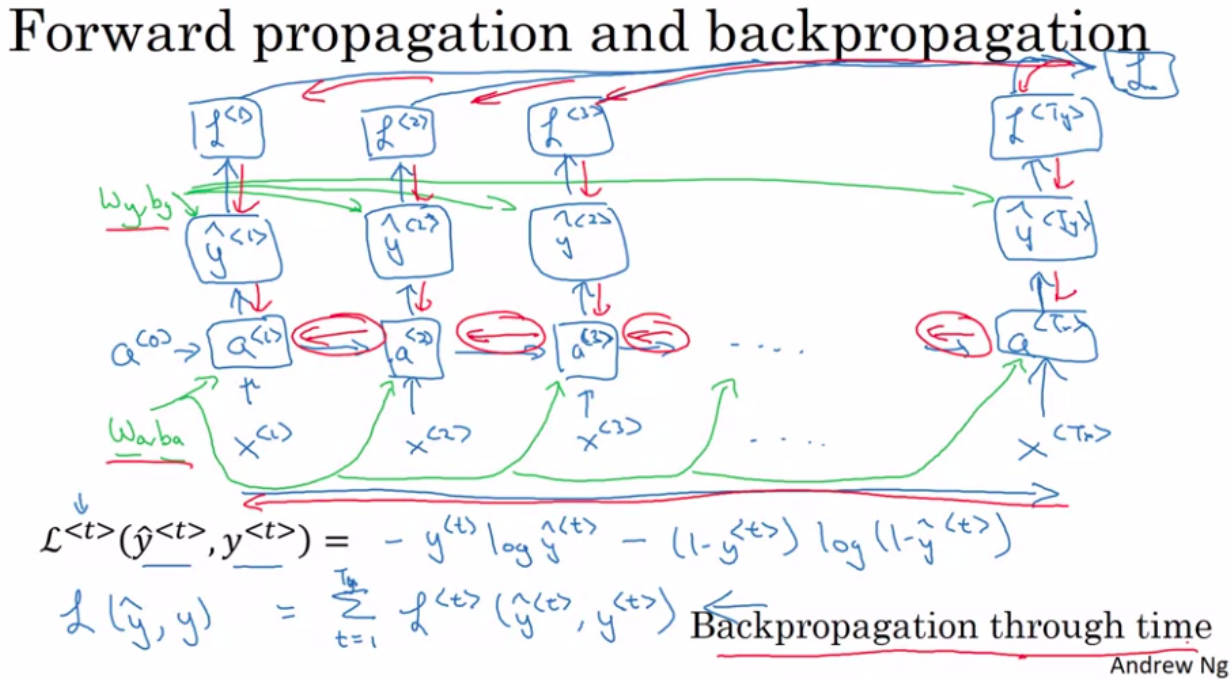
- Forward Propagation:



- Simplified RNN notations

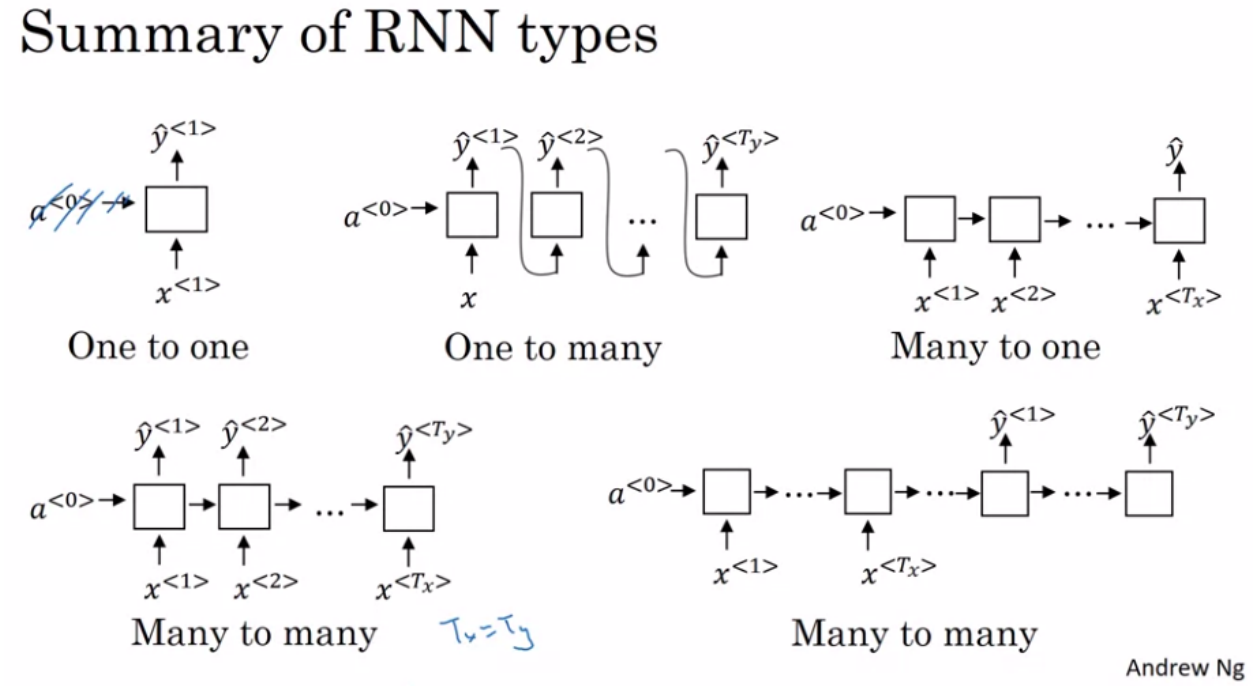


W1L4: Backpropagation: similar to Backprop in NN



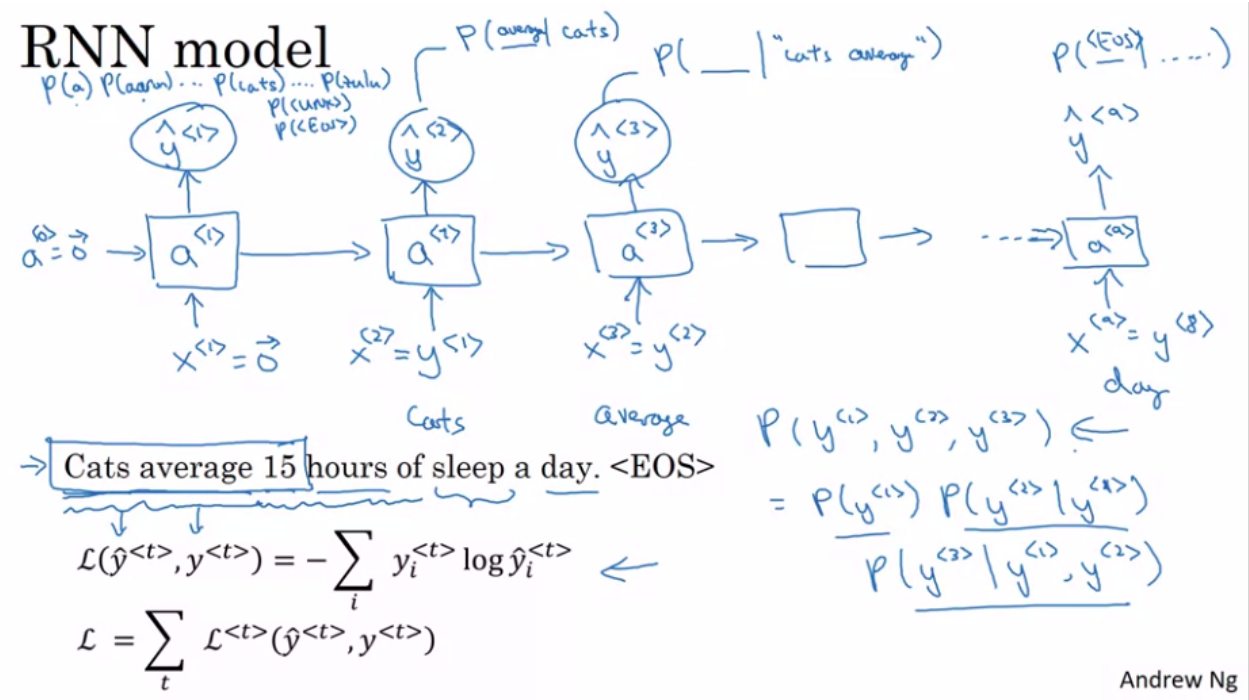
W1L5: types of RNNs

- One to one: Tx = Ty
- One to many: music generation(Tx is the empty set)
- Many to many: machine translation (Tx not equal to Ty)
- Many to one: movie review/ sentiment classification



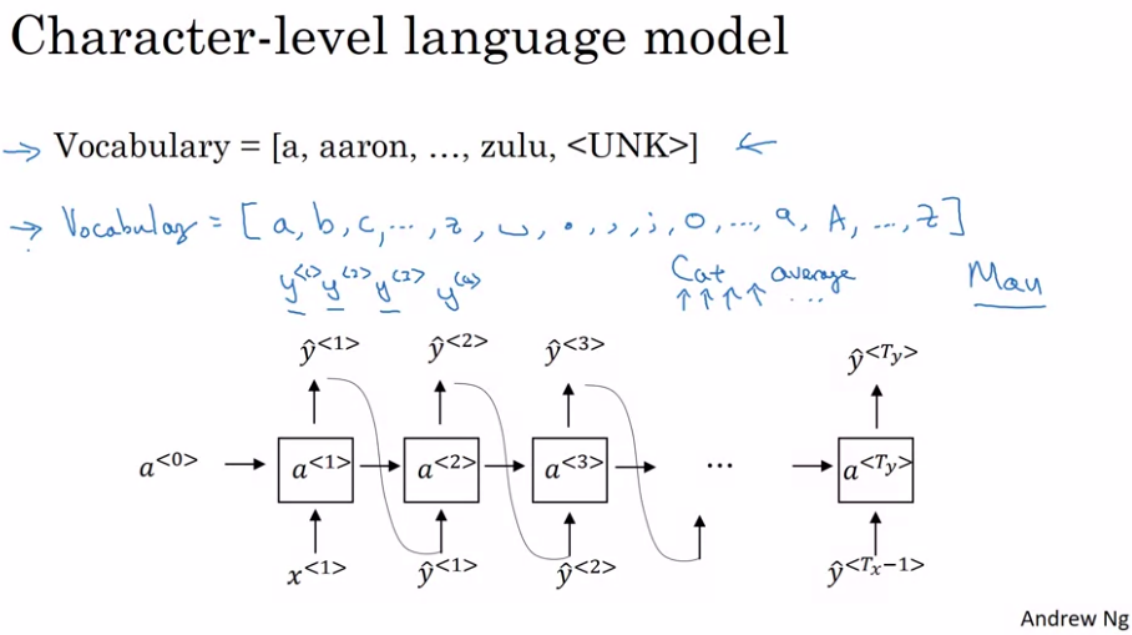
W1L6: Language models and sequence generation

- Predicting the probability of occurrence of the next words in the sentence.



W1L7: Sampling Novel Sequences

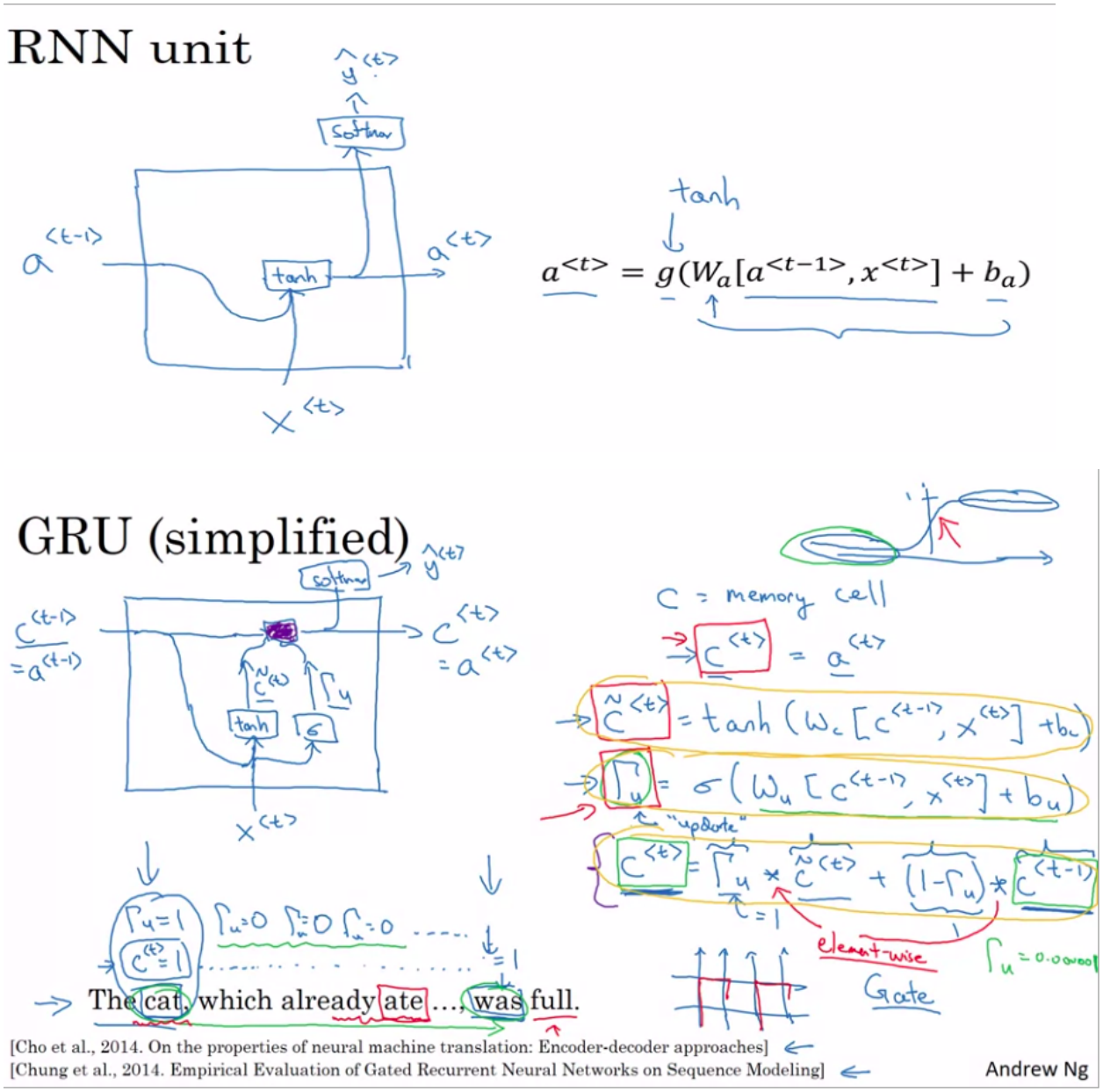
- Character level: computationally expensive but accurate
- Word level models are simple and most widely used.



W1L8: Vanishing gradients with RNN

- For an RNN, the output is dependent on the inputs close to it, and not much on the earlier inputs, i.e. it fails to backpropagate all the way to start for computing the output towards the end. So, RNNs are not very good at capturing long-range dependencies. This is an issue with the RNNs.
- The sentences might be long and last words could be dependent on the starting words, this can't be captured by the RNNs.
- To take care of this: GRUs.

W1L9: Gated Recurrent Unit



Full GRU

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

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

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

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

Correction in "Gated Recurrent Unit (GRU)" (14:04):

The last line should use an element-wise multiplication "*" instead of a plus sign "+".

W1L10: LSTM

GRU and LSTM

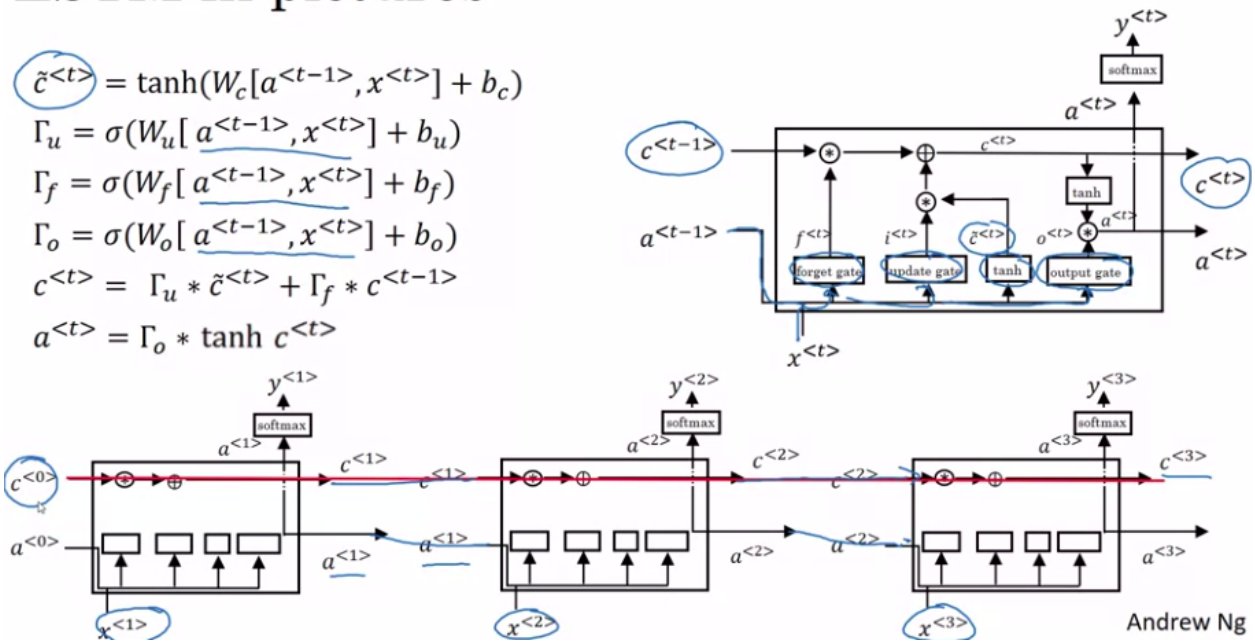
GRU	LSTM
$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$	$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$
$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$	(update) $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$
$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$	(forget) $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$
	(output) $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$	$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$
$a^{<t>} = c^{<t>}$	$a^{<t>} = \Gamma_o * c^{<t>}$

- Hochreiter & Schmidhuber 1997. Long short-term memory

Andrew Ng

LSTM in pictures

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

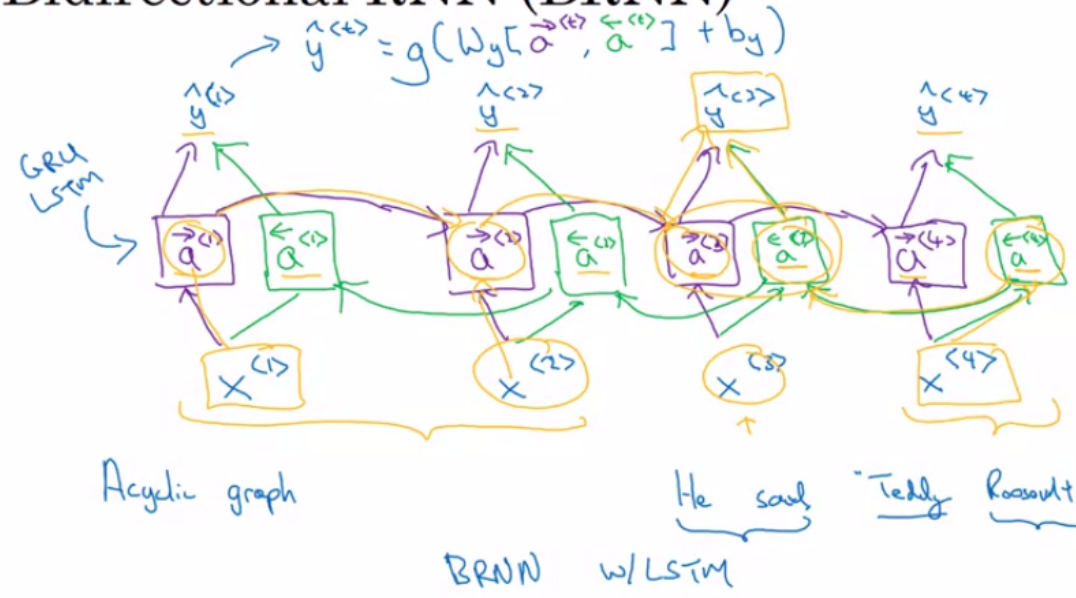


Andrew Ng

W1L11: Bidirectional RNN

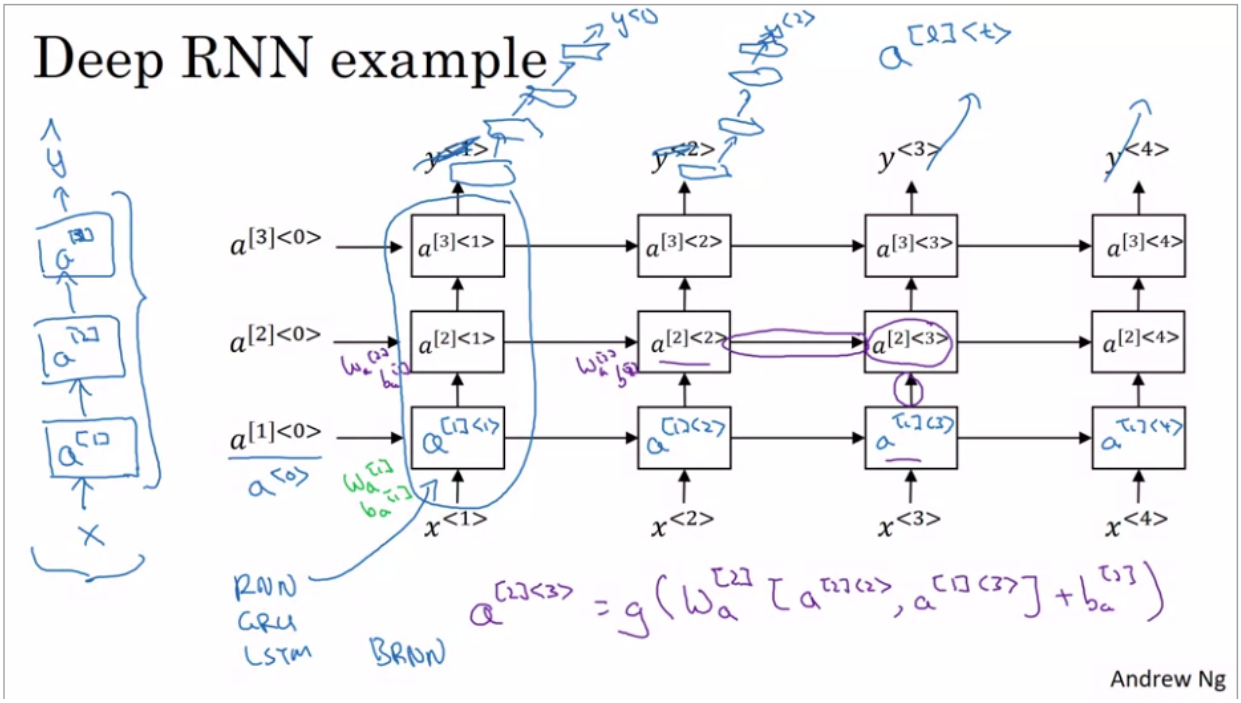
- BRNN with LSTM is a popular model.

Bidirectional RNN (BRNN)



Andrew Ng

W1L12: Deep RNNs



Featurized representation: word embedding

	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

size, cost, alive, verb

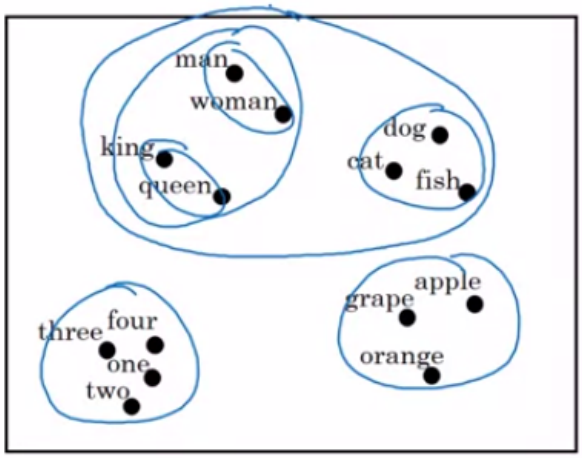
e_{5391} e_{9853}

I want a glass of orange juice.

I want a glass of apple juice.

Andrew Ng

Visualizing word embeddings

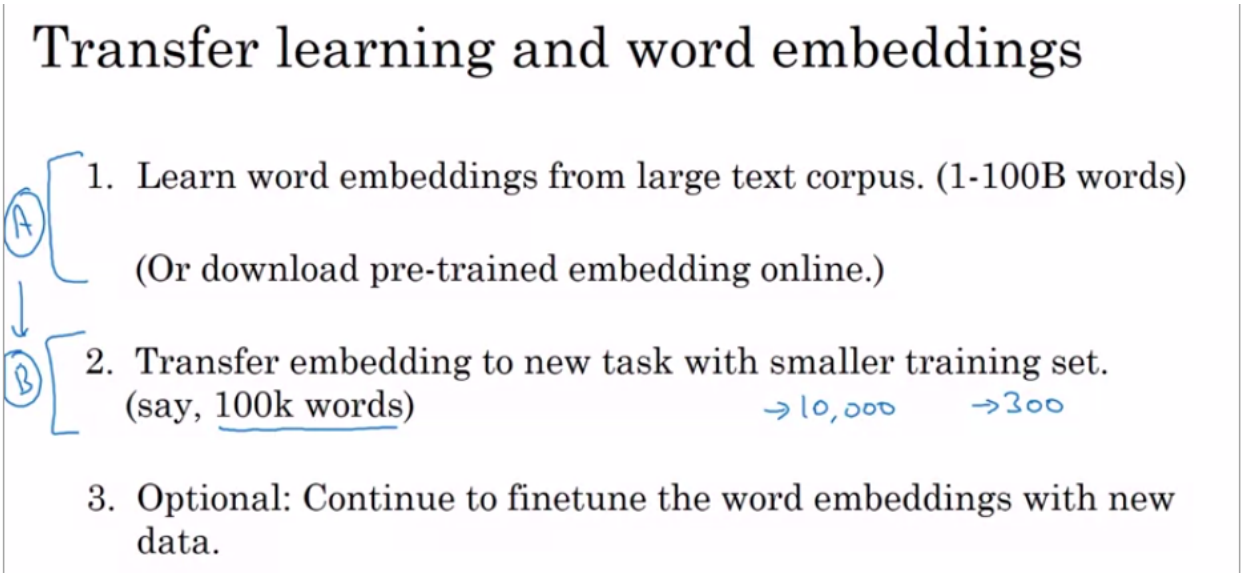


300D → 2D

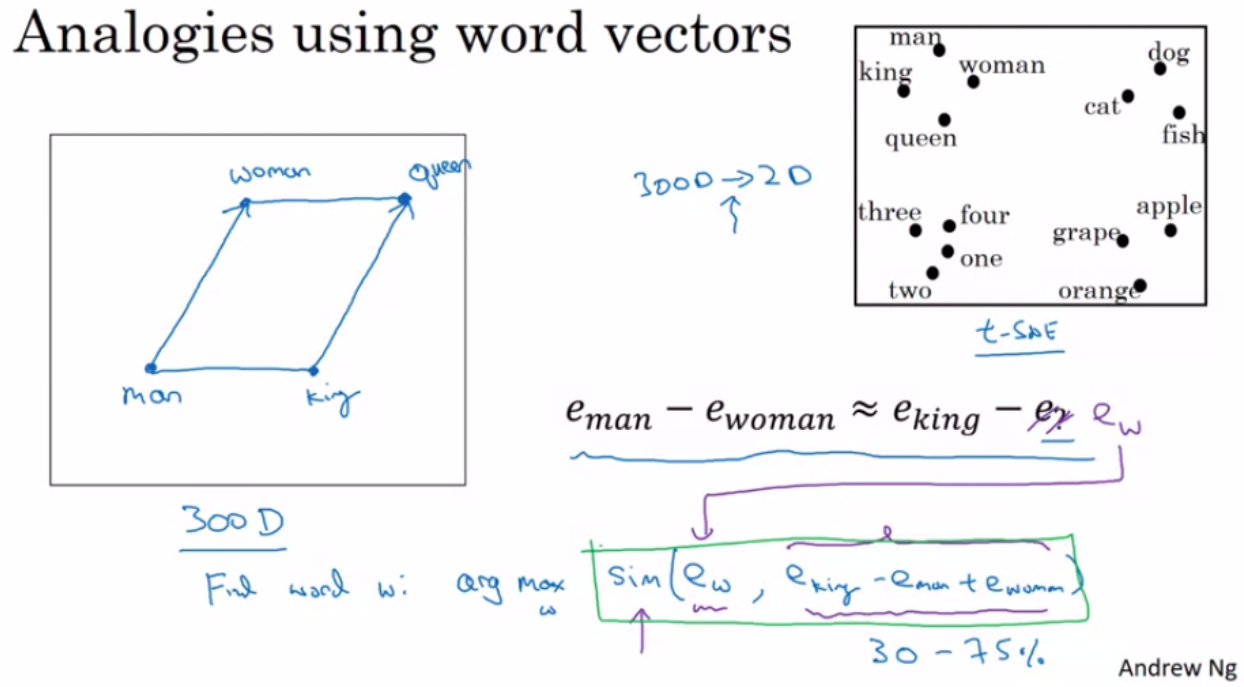
t-SNE

- t-SNE algorithm for the visualization

W2L2:



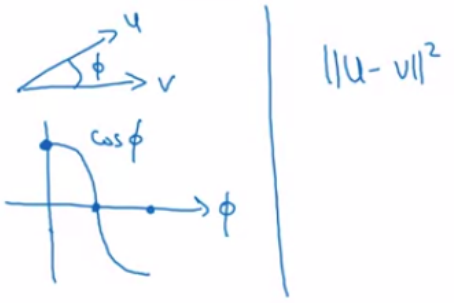
W2L3: Properties of Word Embeddings: similarity



Cosine similarity

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

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



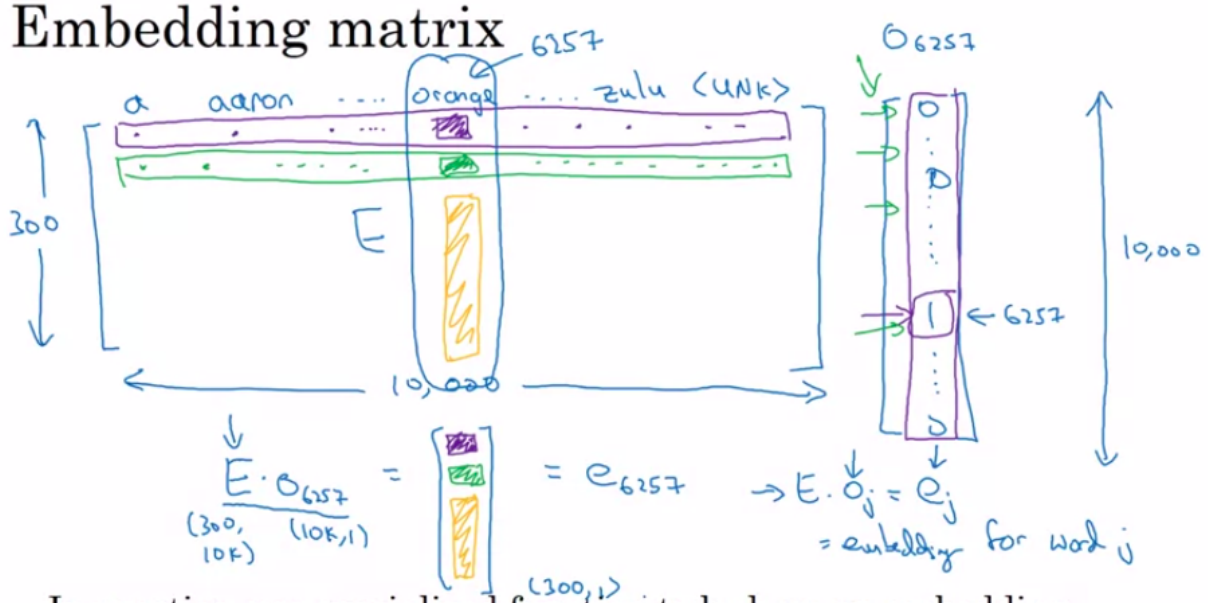
$\|u - v\|^2$

Man:Woman as Boy:Girl
Ottawa:Canada as Nairobi:Kenya
Big:Bigger as Tall:Taller
Yen:Japan as Ruble:Russia

Andrew Ng

W2L4: Embedding Matrix

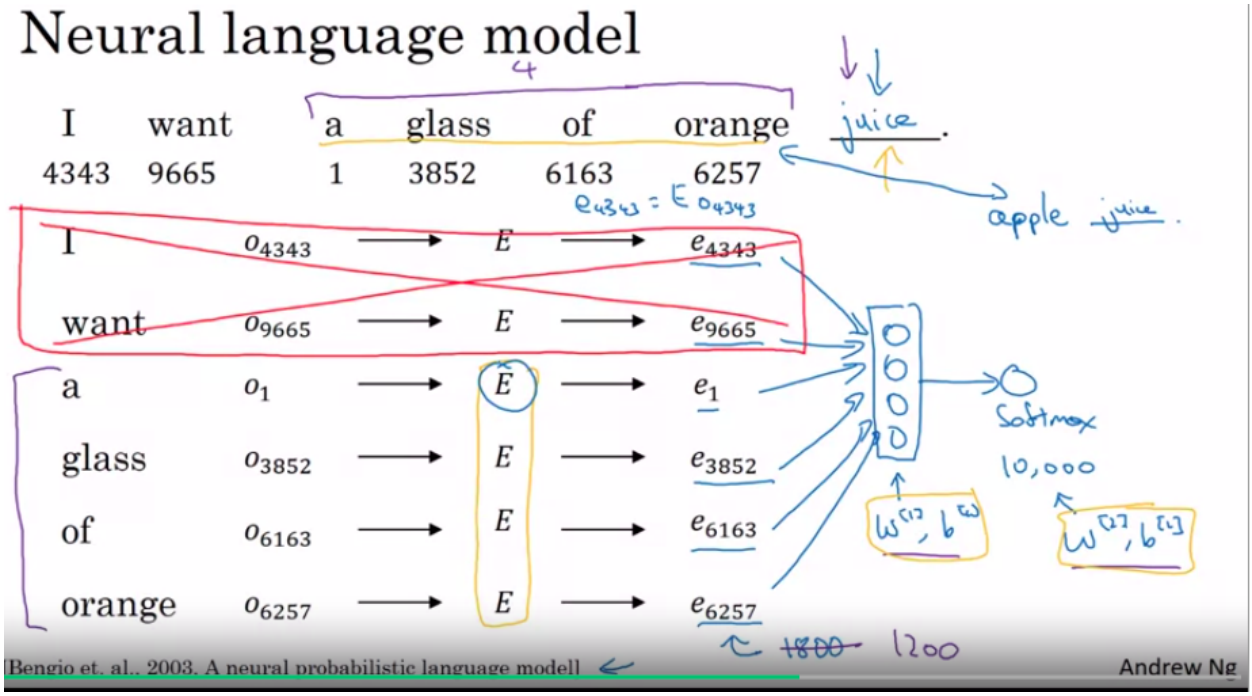
Embedding matrix



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

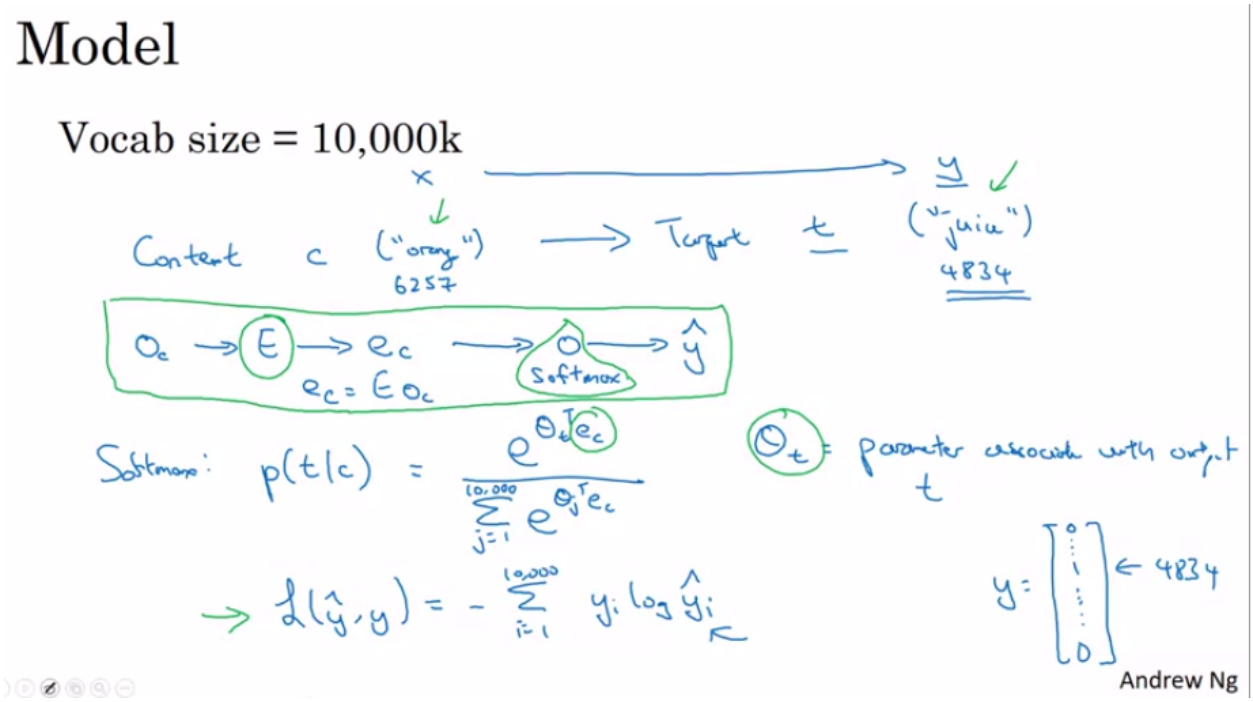
Andrew Ng

W2L5: Learning word embeddings



- The window of context can be variable and be also decided to use left and right 4 words or just one previous word and so on.

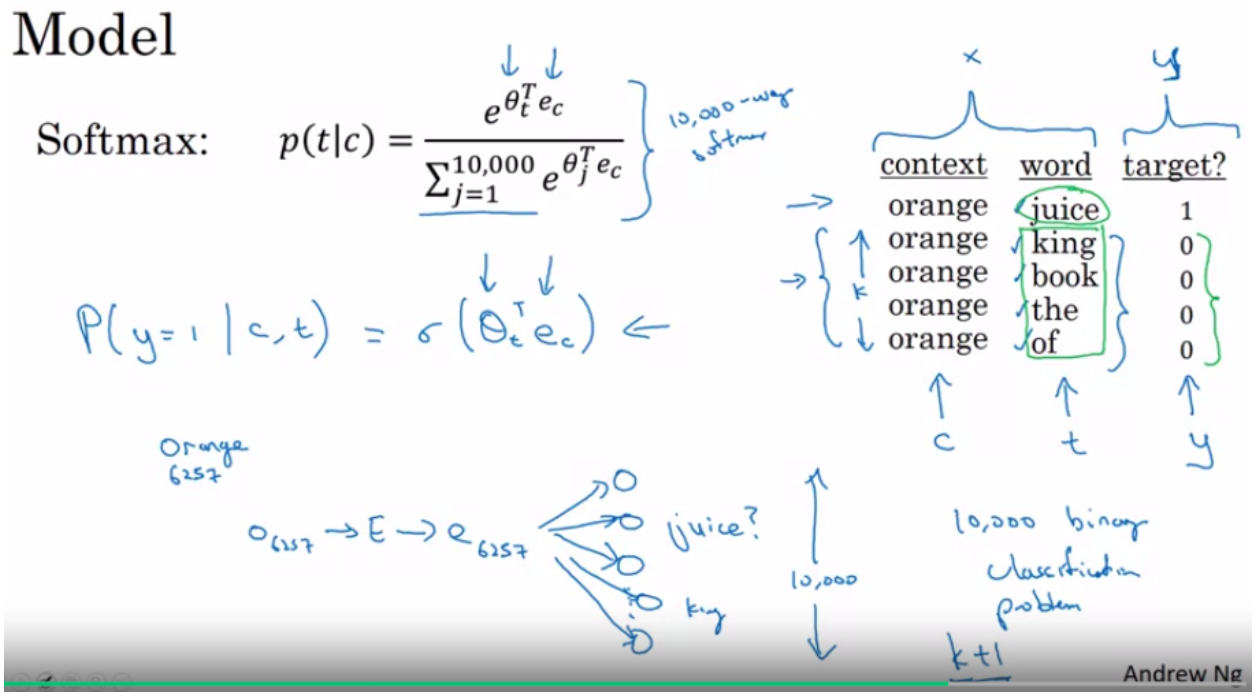
W2L6: Word2Vec



- To sample context c: uniformly random probability is used.
- The issue with this model: The softmax calculation is computationally expensive.

- Solution:
    1. Hierarchical softmax: using the tree data structure. Generally, the trees are unbalanced as the most commonly used words are in upper nodes.
    2. Negative sampling

W2L7: Negative Sampling

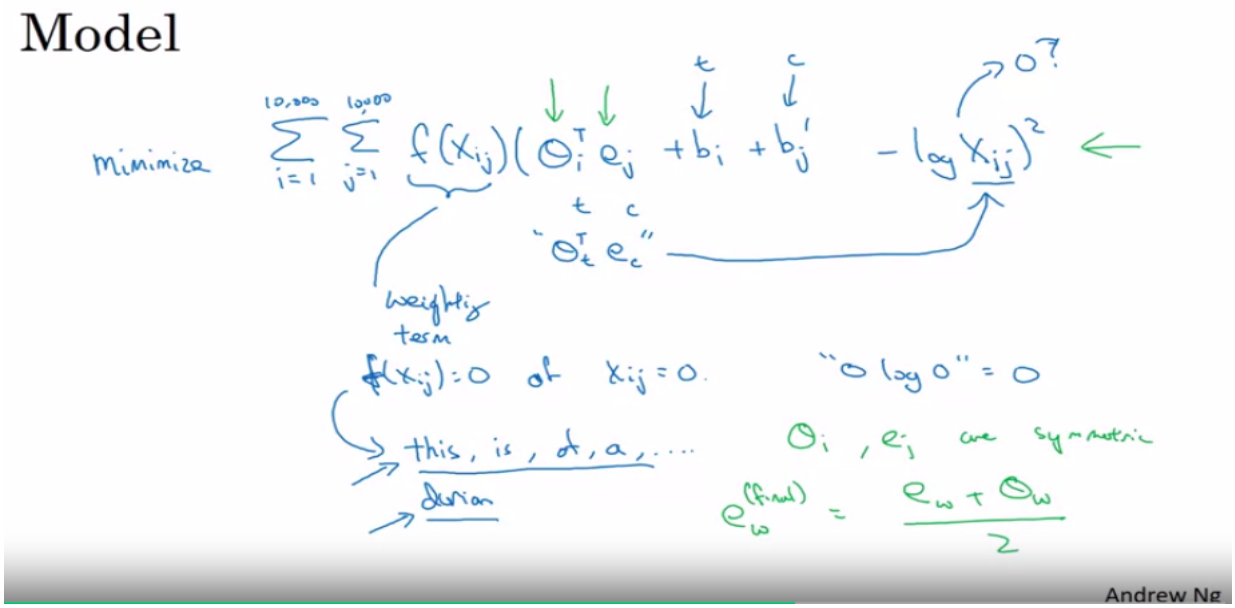


- Choosing negative samples:

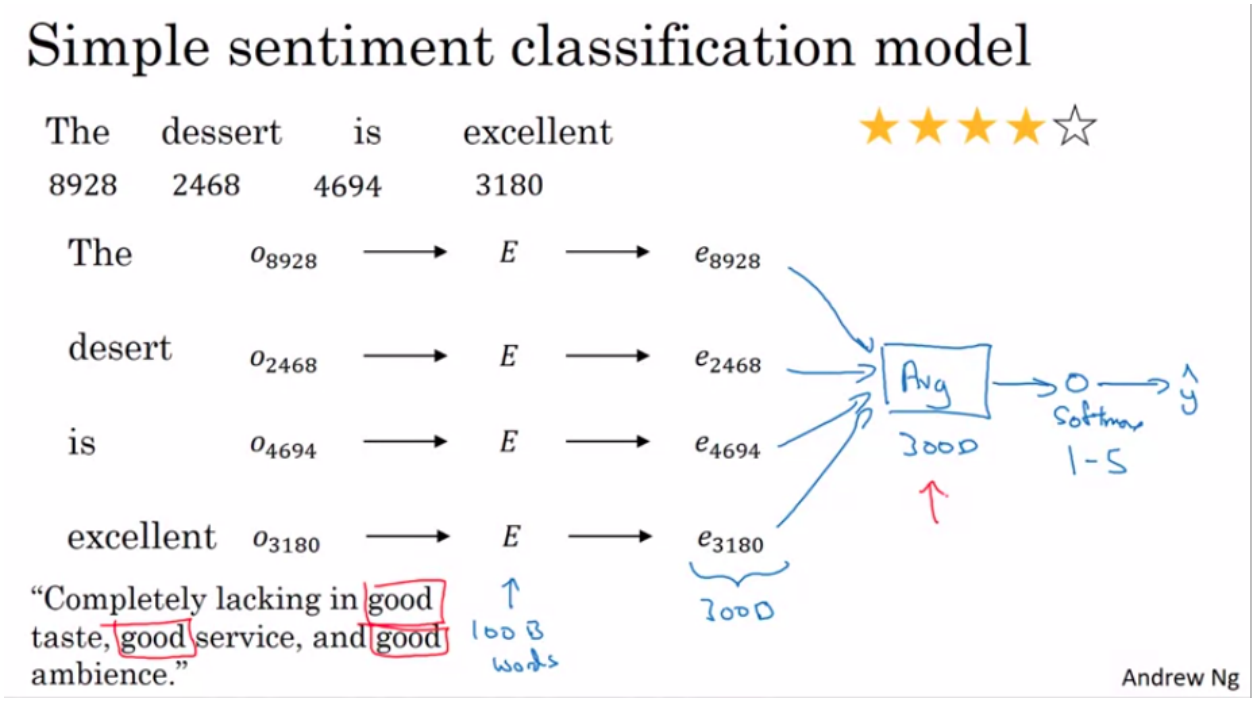
$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$$

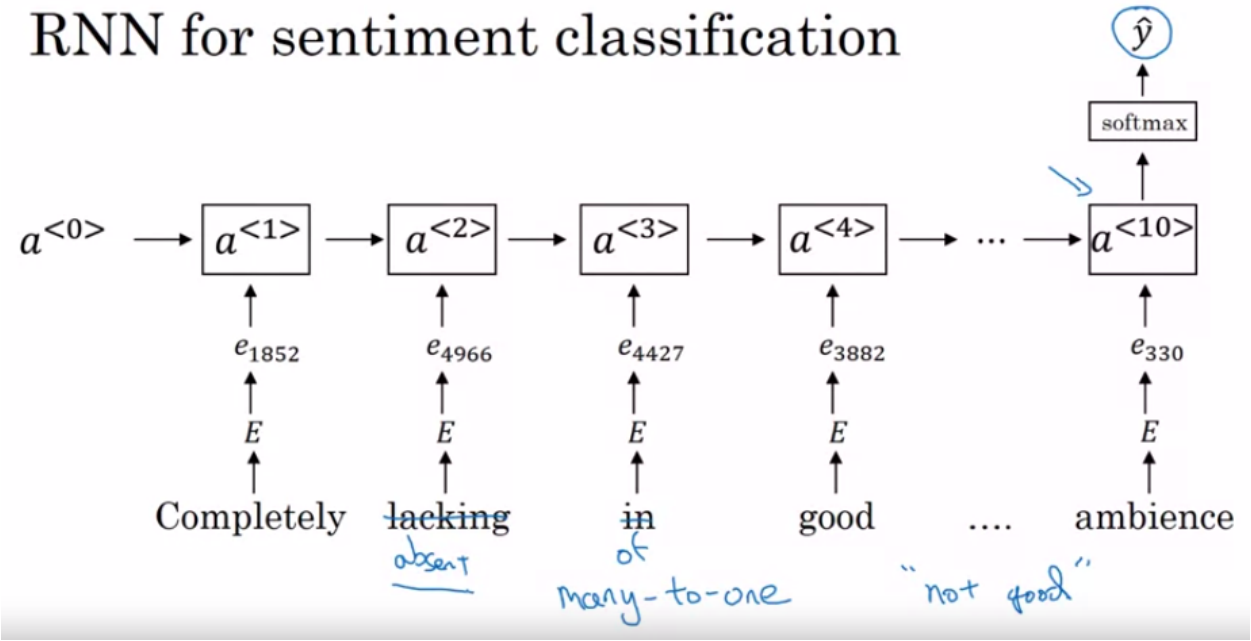
W2L8: Global vectors for word representations(GloVe):

- Xij: number of times the target word(t) has occured with context(c)

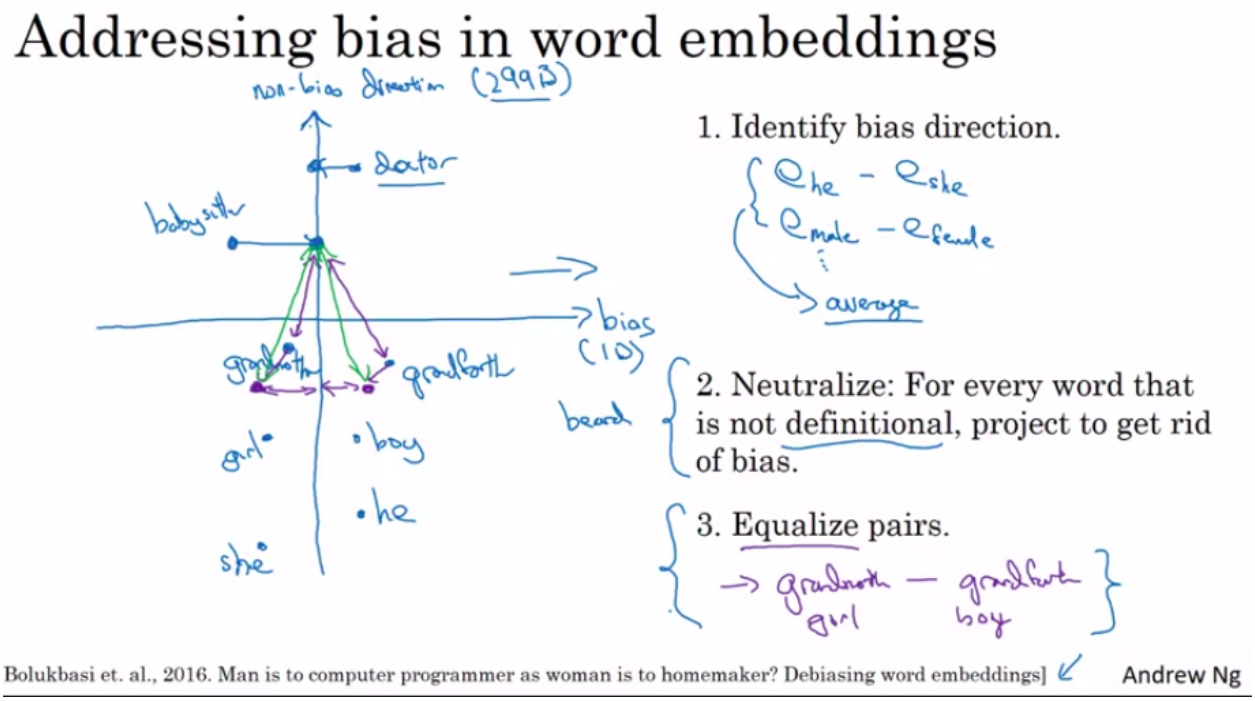


W2L9: Application: Sentiment Classification





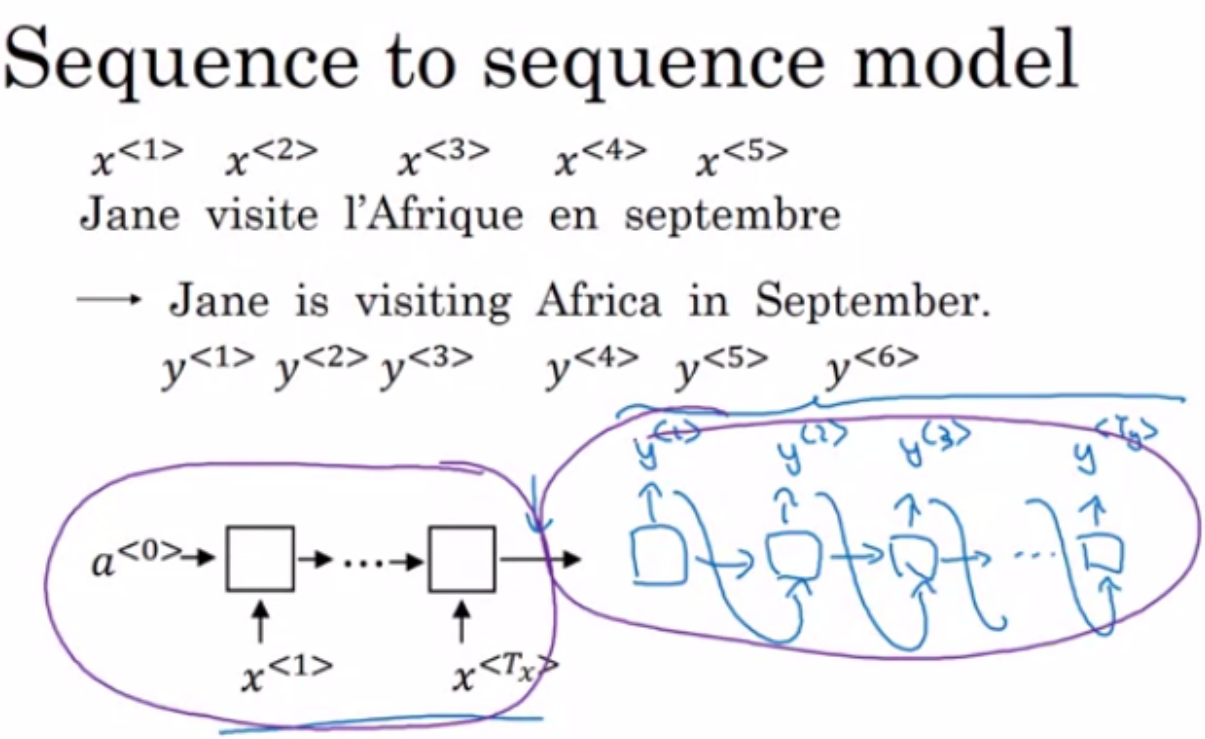
W2L10: Debiasing



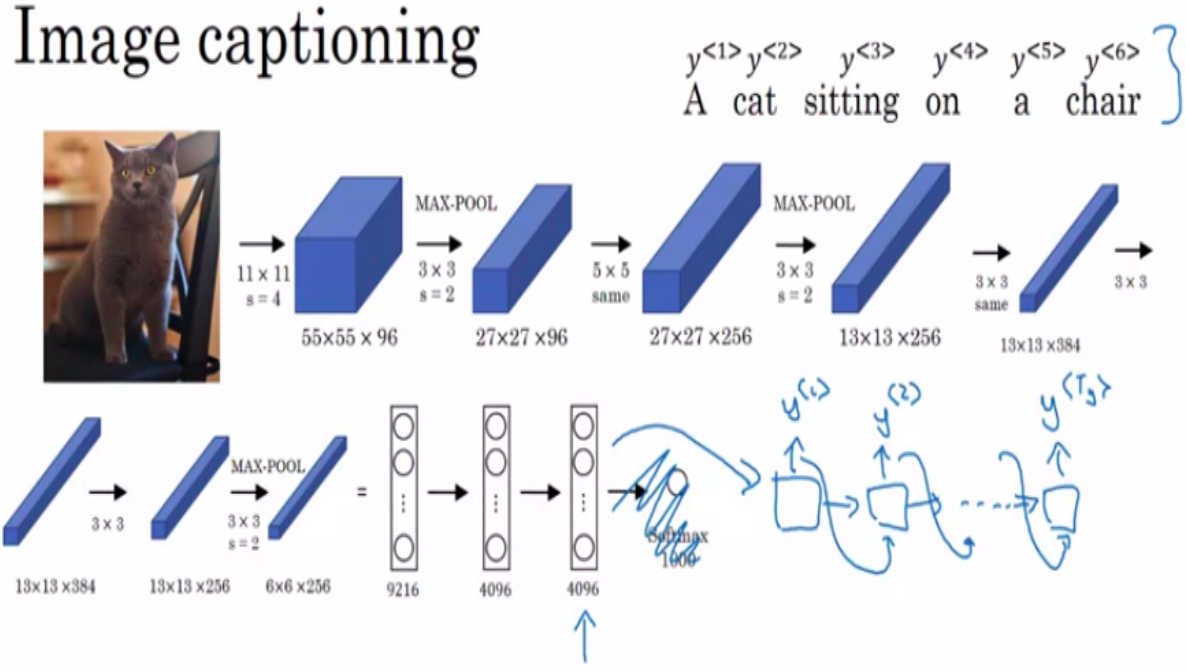
Week 3: Sequence to sequence architecture

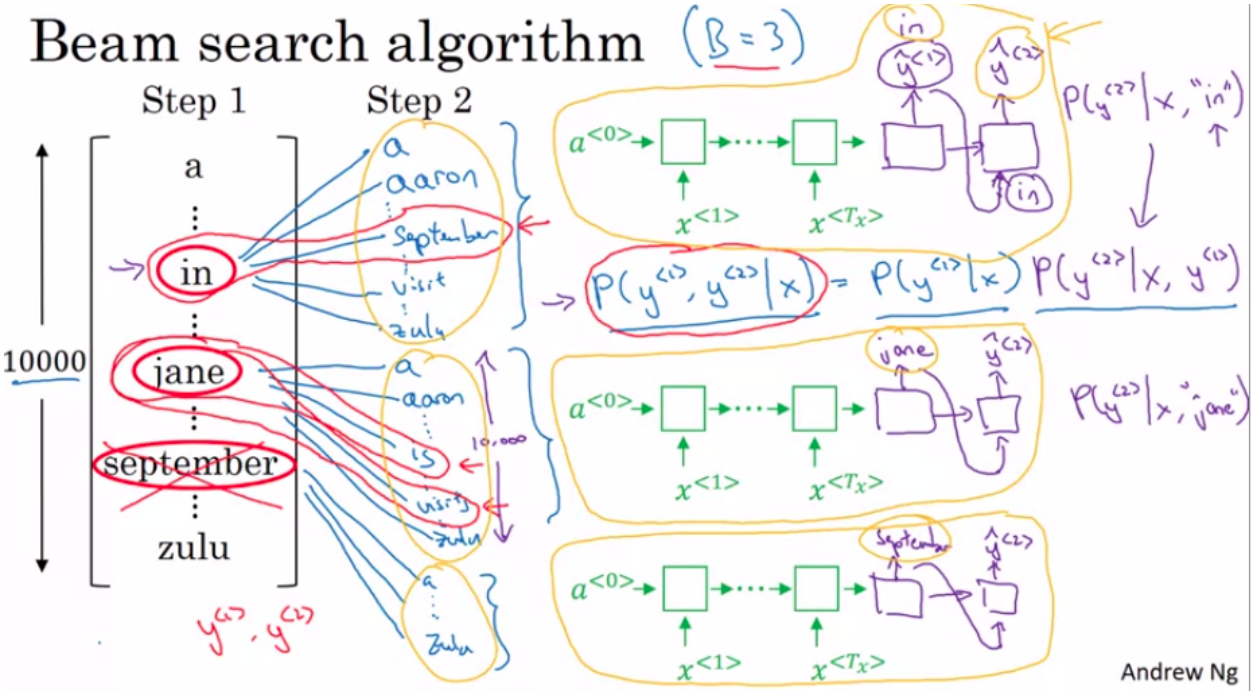
W3L1: Basic models

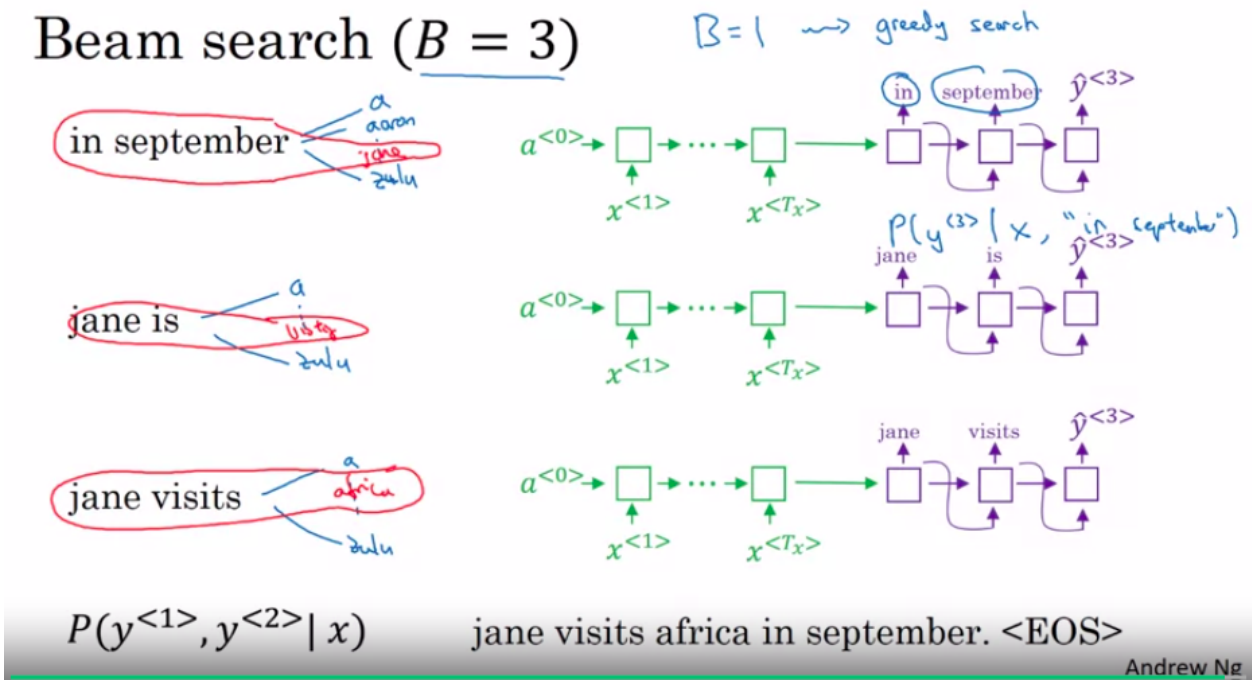
- Machine translation model:



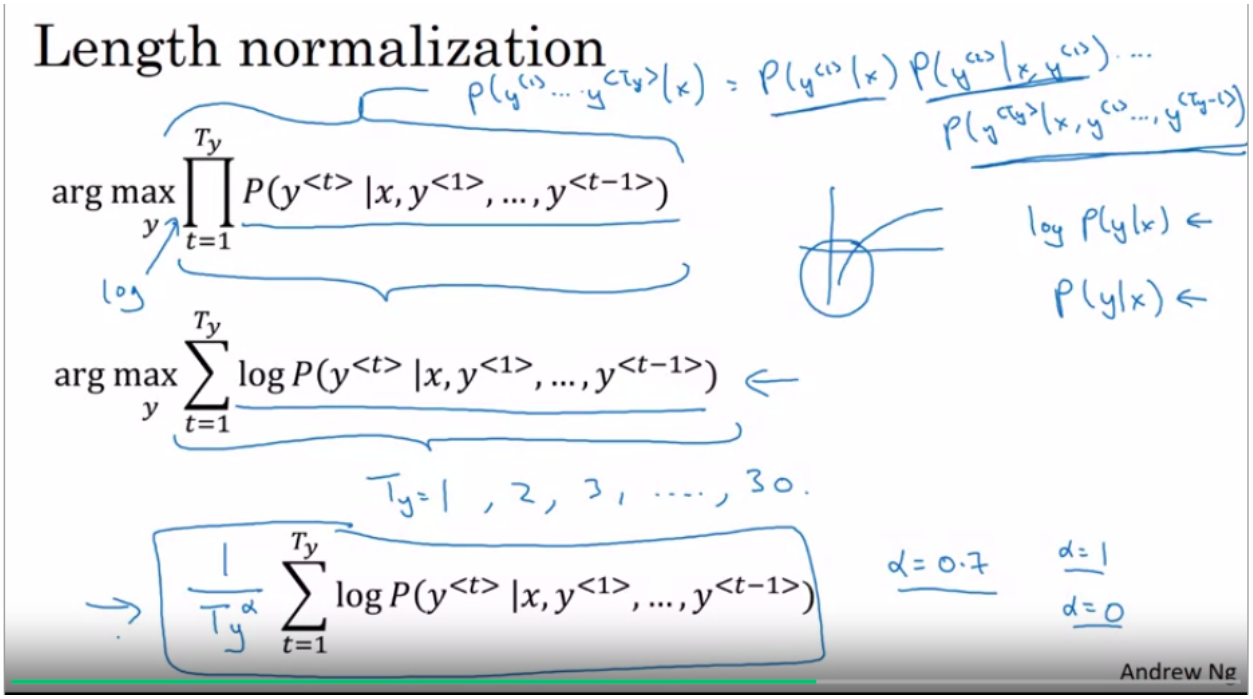
- Image captioning:




Beam search algorithm
(B = 3)
Step 1
Step 2
10000
a
in
jane
september
zulu
aaron
September
visit
Andrew Ng

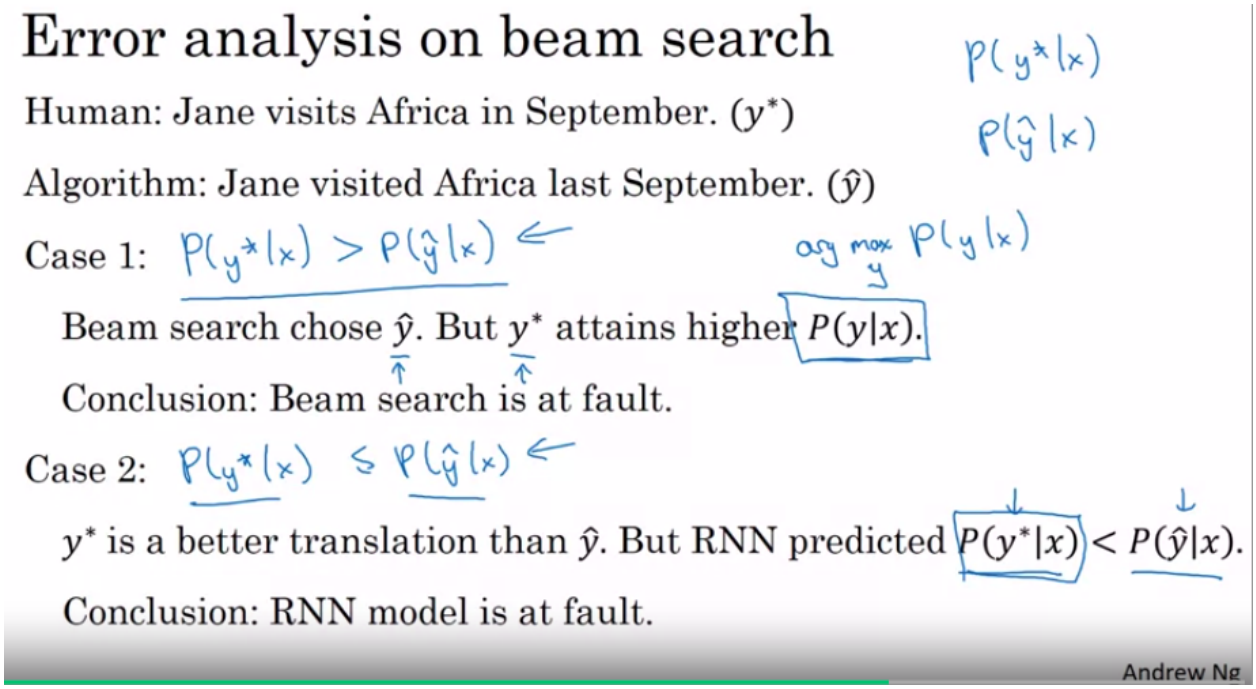

Beam search (B = 3)
B = 1 ⇝ greedy search
in september
jane is
jane visits
P(y<1>, y<2> | x)
jane visits africa in september. <EOS>
Andrew Ng

W3L4: Refinements in the beam search



- Choosing beam width(B) large: better results but slower
- Beam width small: worst results but faster
- In production: 10-100 is mostly used.

W3L5: Error analysis in beam search



Error analysis process

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	B
				R
				B
				R
				R

Figures out what faction of errors are "due to" beam search vs. RNN model

Andrew Ng

- Modify the one causing maximum errors.
-

W3L5: Bleu Score

- There might be more than 1 correct translations of any sentence. In that case, the bleu score evaluates the correctness of the machine-translated output.

Evaluating machine translation

French: Le chat est sur le tapis.

Bleu
bilingual evaluation understudy

→ Reference 1: The cat is on the mat. (2 appearance)

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

→ MT output: the the the the the the the.

Precision: $\frac{7}{7}$

Modified precision: $\frac{2}{7}$ ← Count_{clip}("the") / Count("the")

Bleu score on bigrams

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
cat on	1	1
on the	1	1
the mat	1	1

$\frac{4}{6}$

[Papineni et. al., 2002. Bleu: A method for automatic evaluation of machine translation]

Andrew Ng

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. ($\hat{y}$)

$p_1, p_2, \ldots = 1.0$

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

unigram

n-gram

$$p_n = \frac{\sum_{\text{n-grams} \in \hat{y}} \text{Count}_{clip}(\text{n-gram})}{\sum_{\text{n-grams} \in \hat{y}} \text{Count}(\text{n-gram})}$$

Bleu details

p_n = Bleu score on n-grams only

p_1, p_2, p_3, p_4

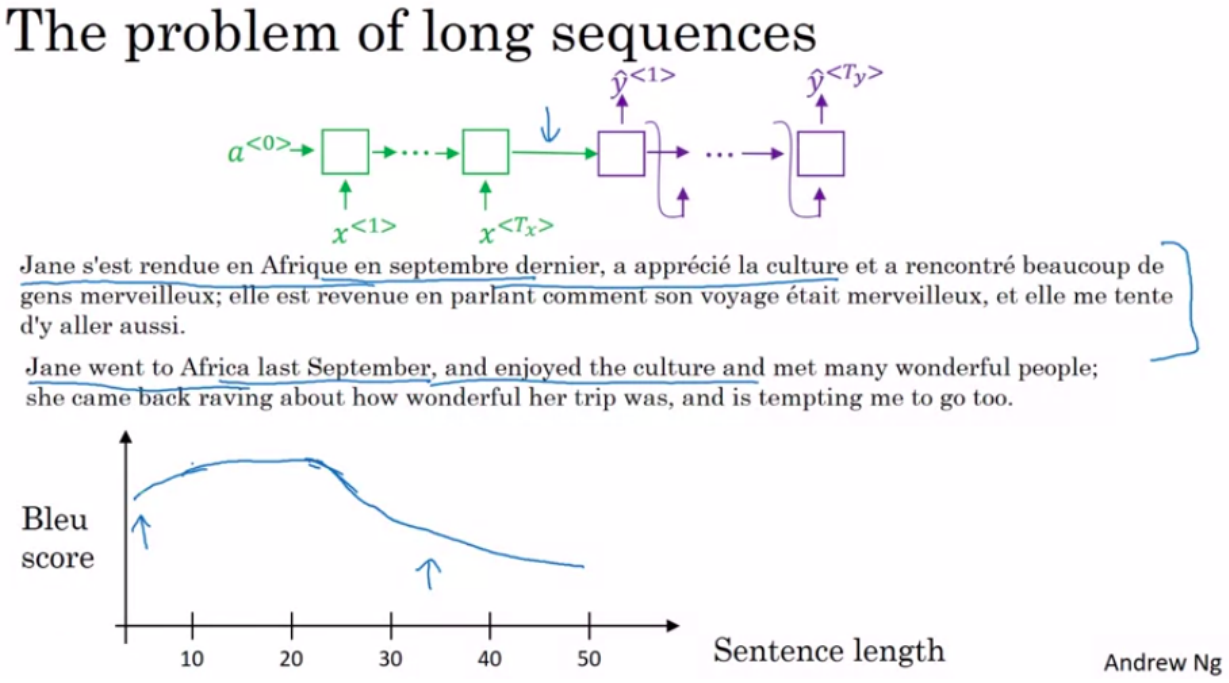
Combined Bleu score: $BP \exp\left(\frac{1}{4}\sum_{n=1}^{4} p_n\right)$

BP = brevity penalty

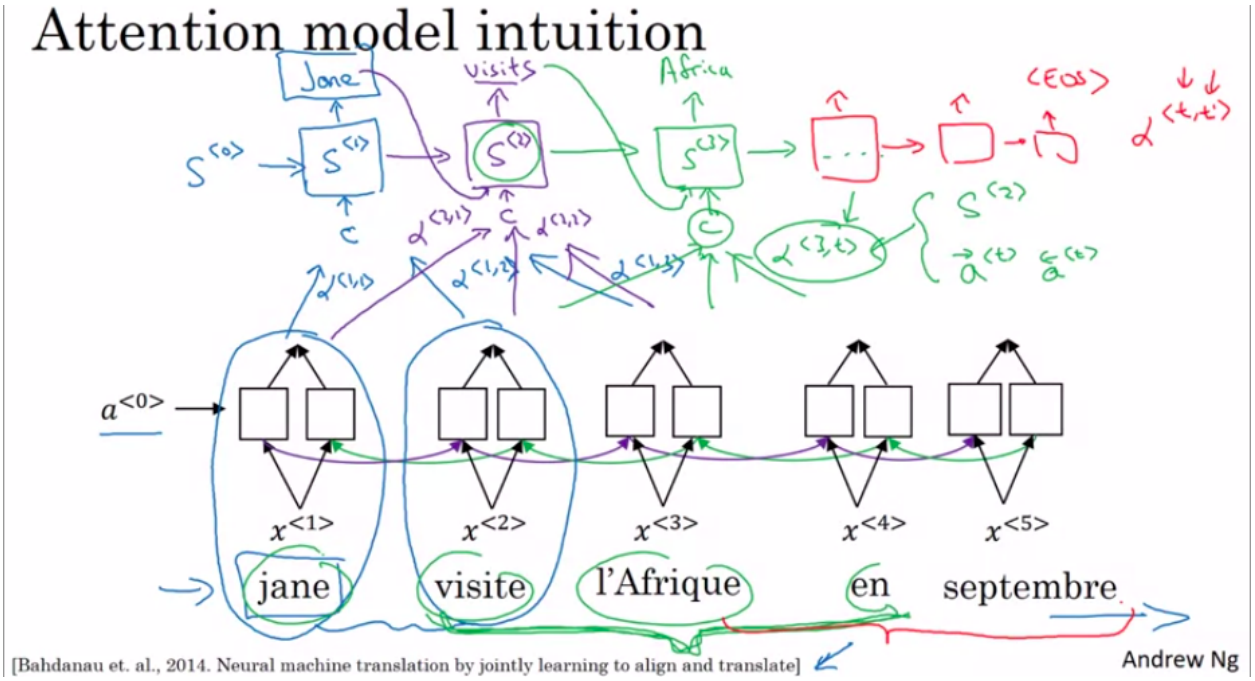
$$BP = \begin{cases} 1 & \text{if MT\_output\_length} > \text{reference\_output\_length} \\ \exp(1 - \text{MT\_output\_length}/\text{reference\_output\_length}) & \text{otherwise} \end{cases}$$

exp(1 − reference_output_length/MT_output_length)

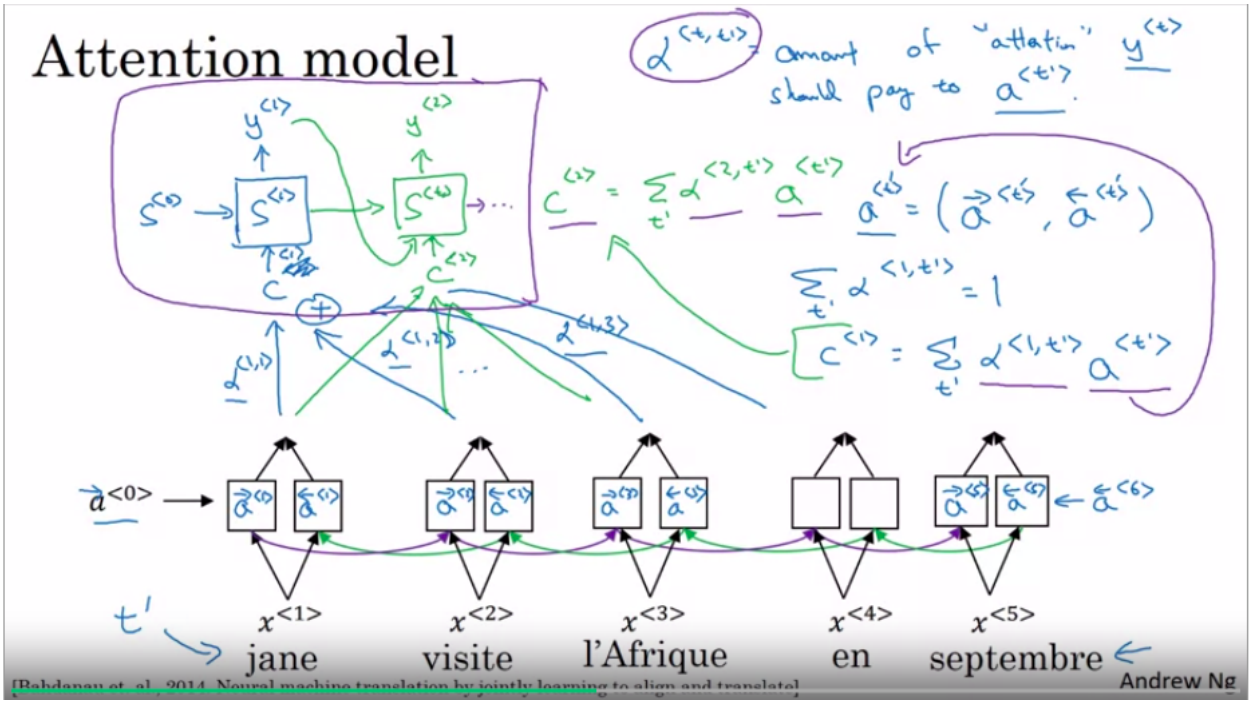
W3L6: Attention model

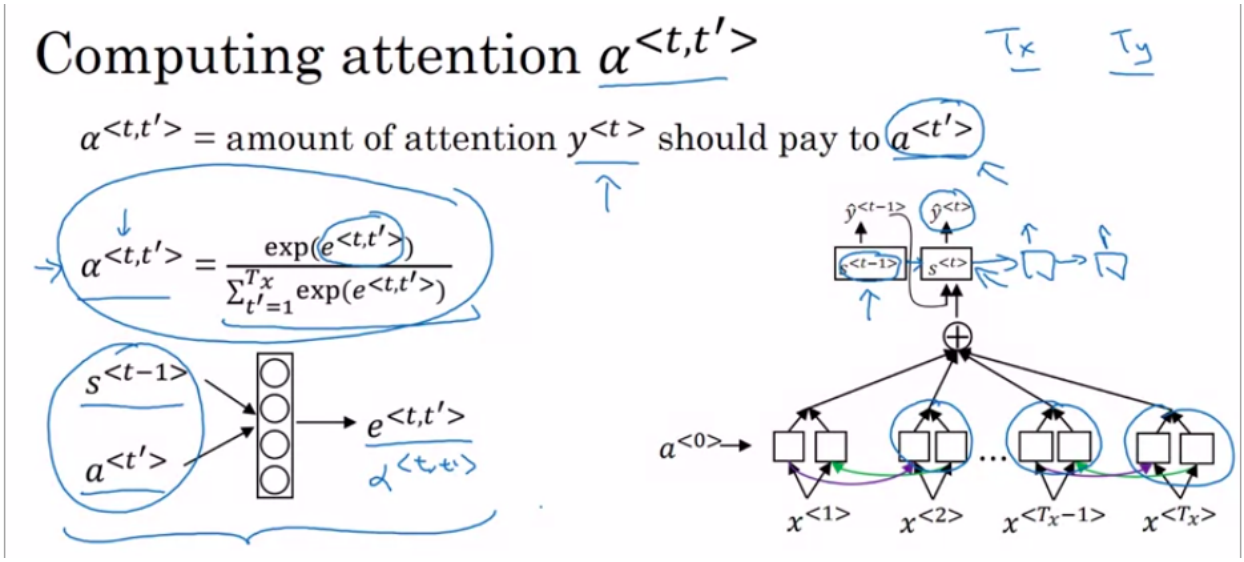


- To avoid this, ATTENTION MODEL



- alpha(m,n): the amount of attention paid to n-th word to predict m-th word


Attention model
jane
visite
l'Afrique
en
septembre
Andrew Ng


Computing attention $\alpha^{<t,t'>}$
$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$
$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$
$s^{<t-1>}$
$a^{<t'>}$
$e^{<t,t'>}$

Trigger word detection algorithm