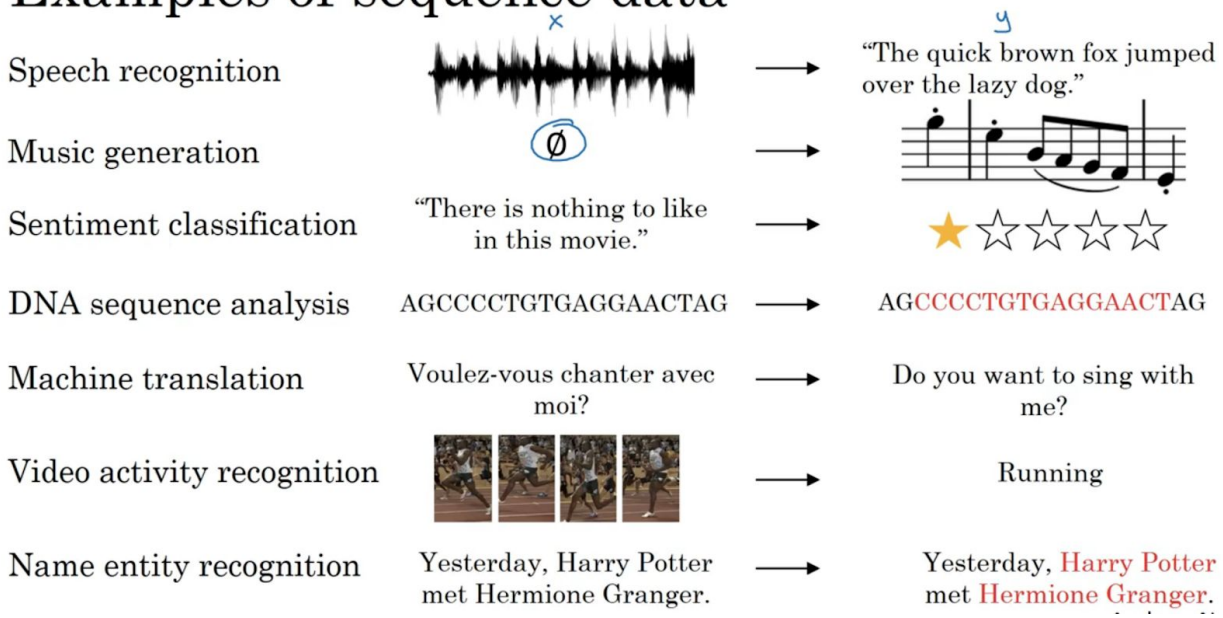
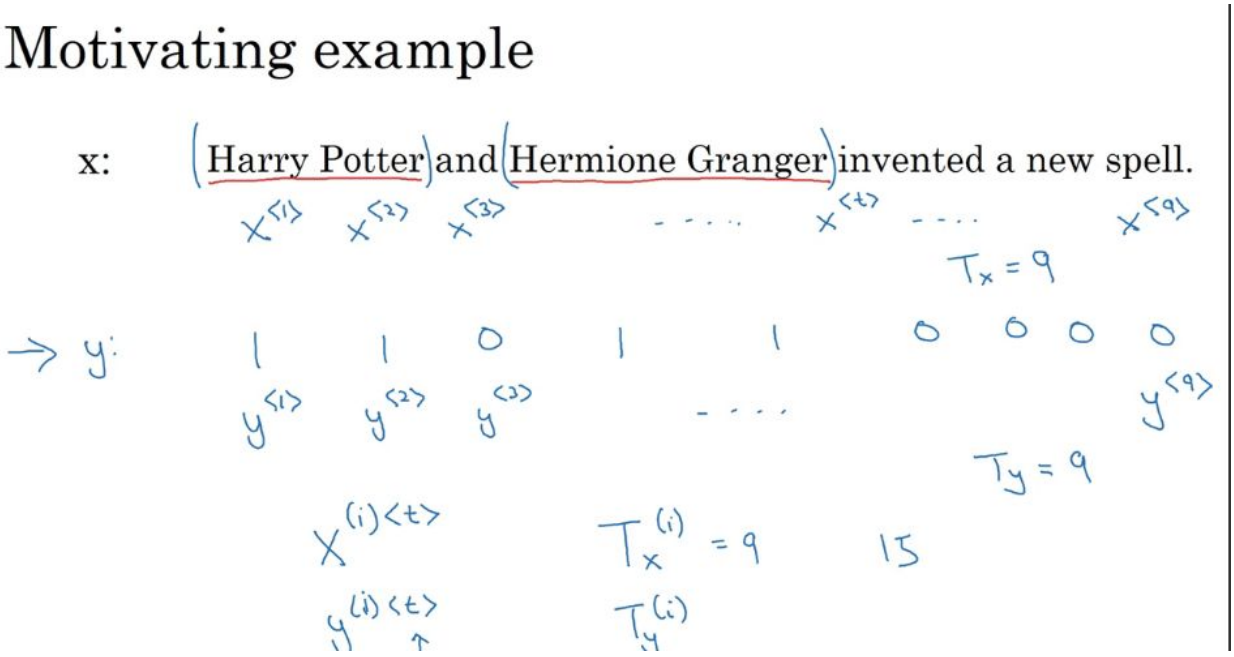
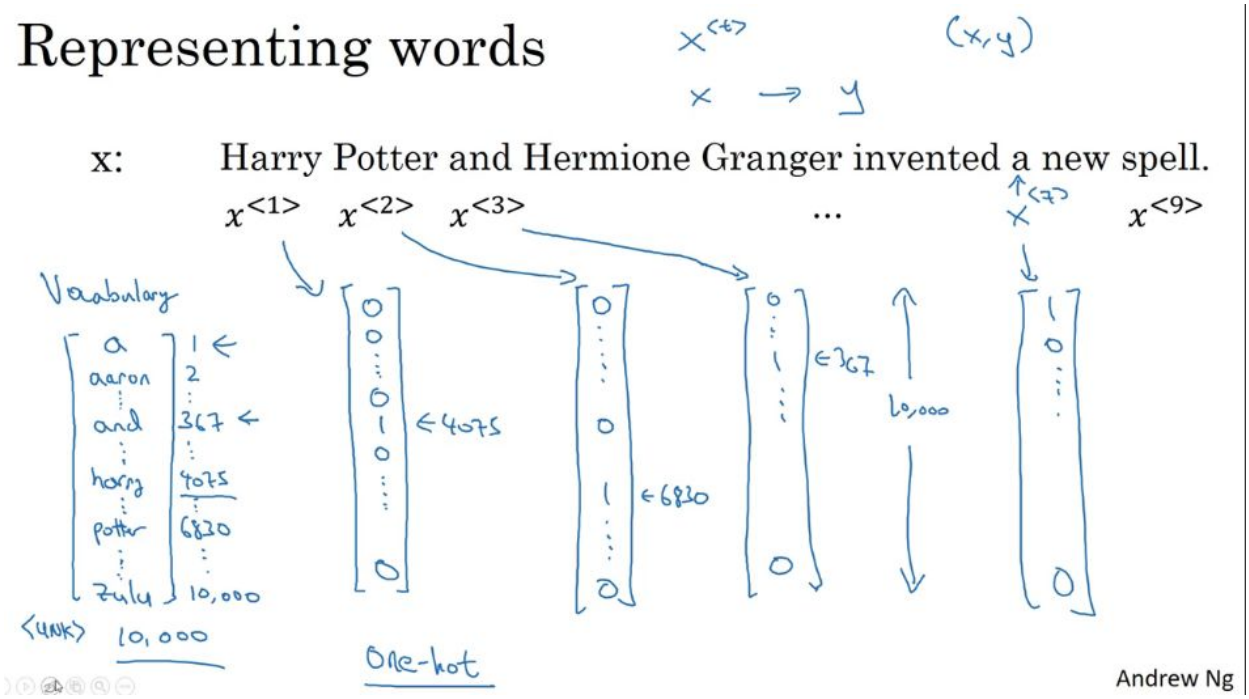




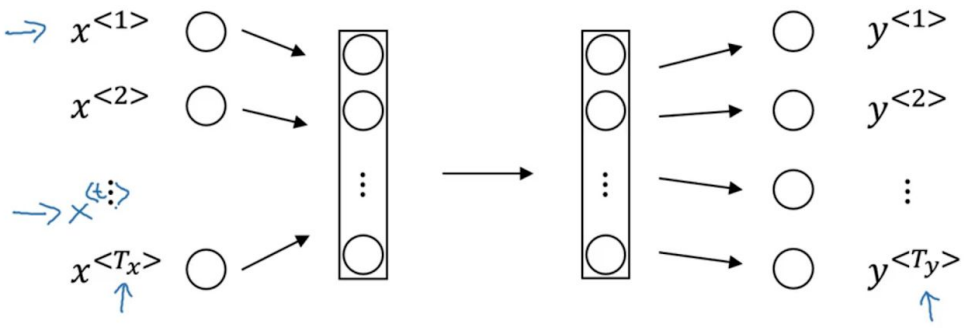
We want to do named entity recognition to identify where the names are in a sentence -- Training data is sentence mapped to 1s and 0s.

ONE HOT VECTOR

Representing words



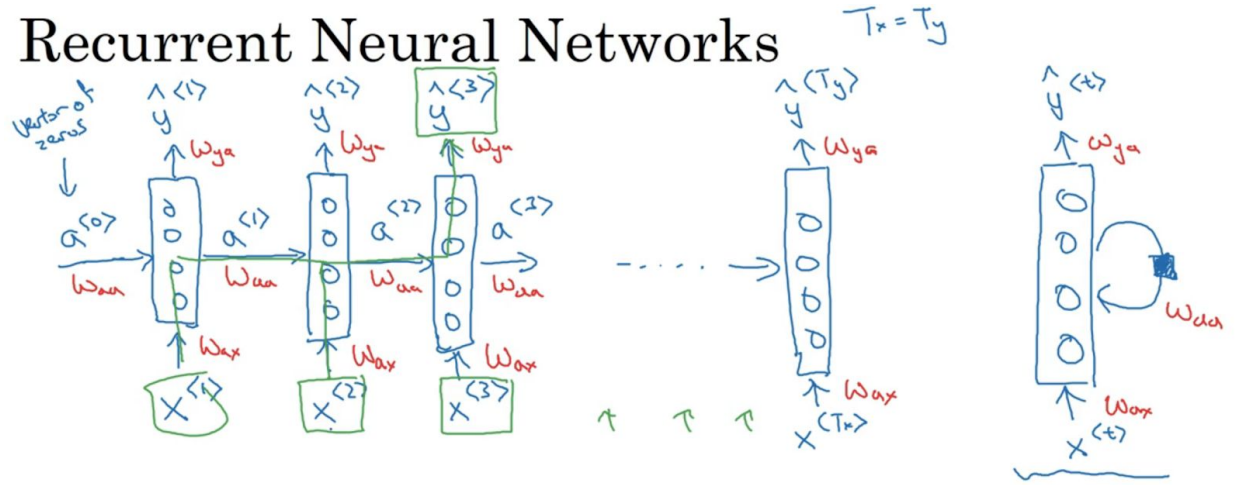
Why not a standard network?



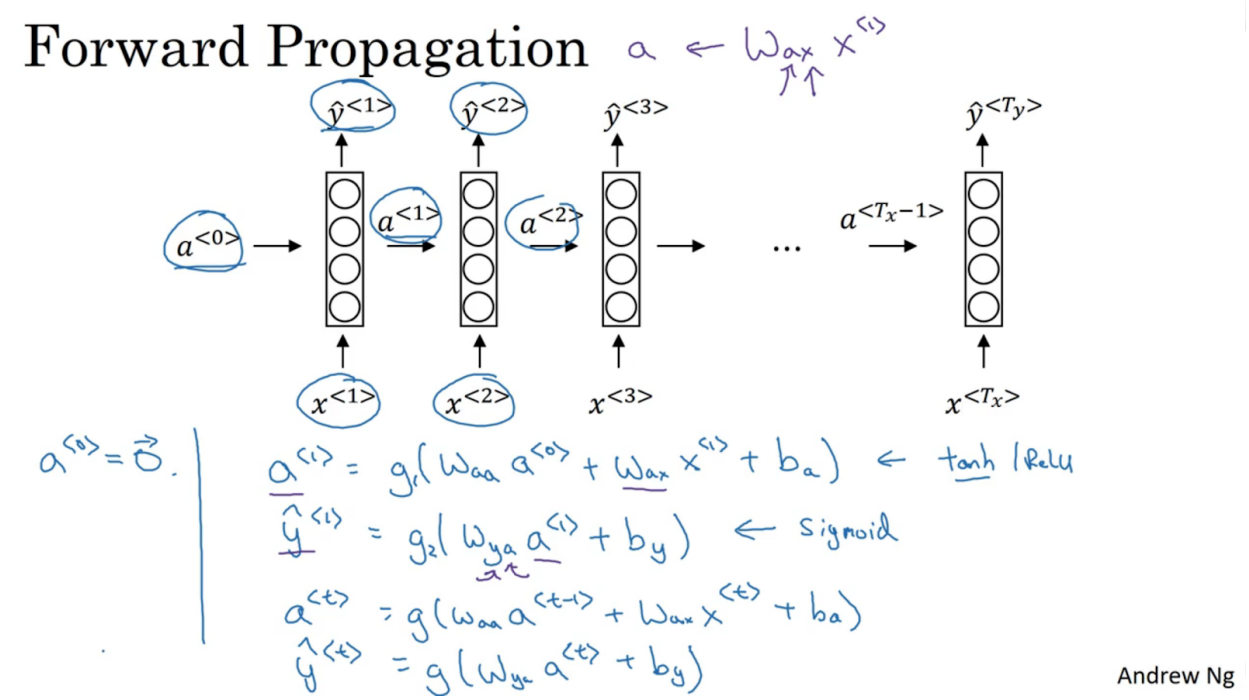
Problems:

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

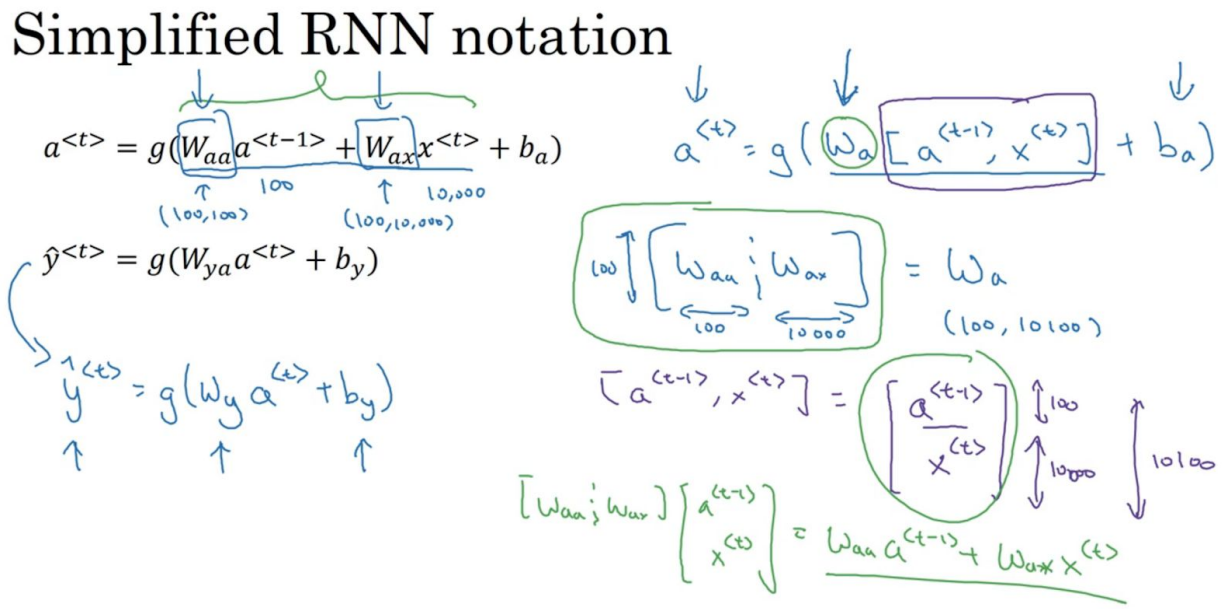
Need a better way to generalize for knowledge learnt in one part to be used elsewhere. Like CNNs for images to generalize the angles etc.



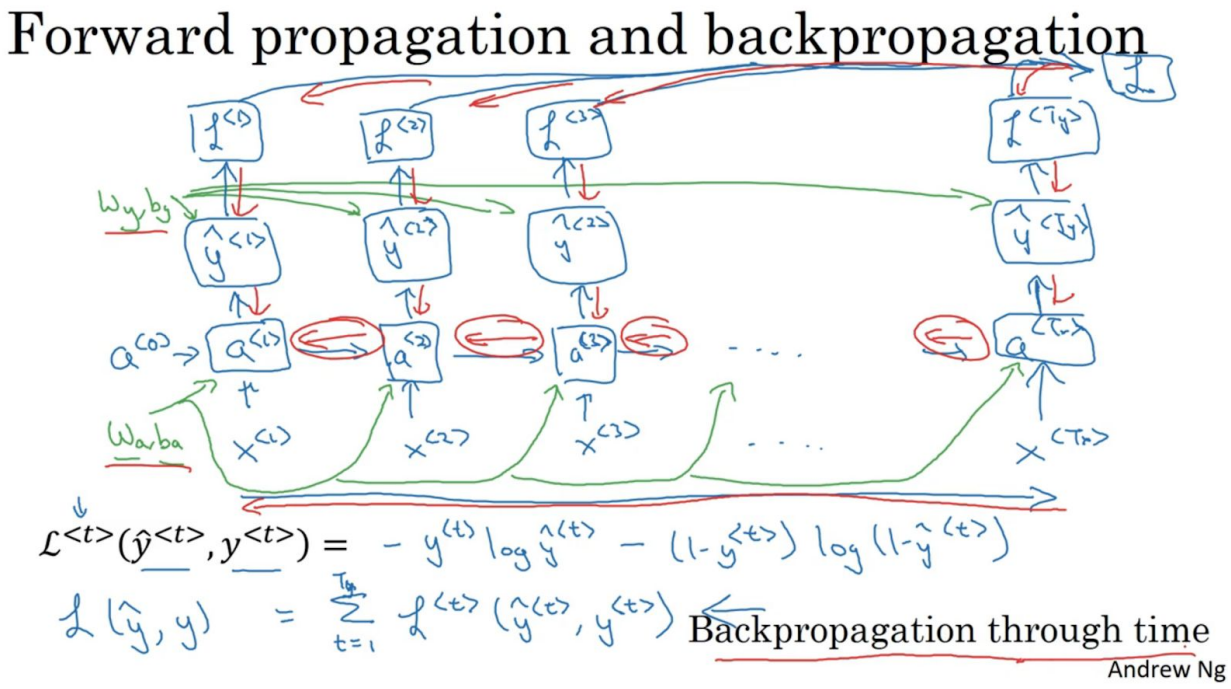
Only uses past to predict future words. Bi directional RNNs are better for this.



Activations or rather RNN has 100 elements in it, there are 10000 words each word having its own weight vector- -- this results in current activation which goes into next activations.

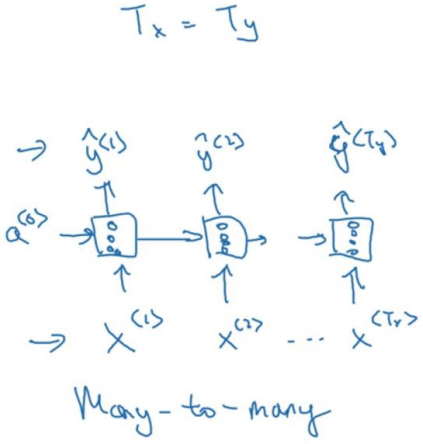


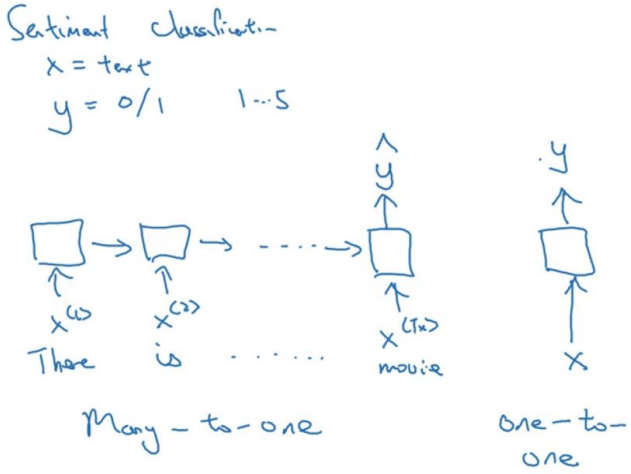
LOSS : Compute loss at every time step and combine for one single loss. Basically at every time step, there is a sigmoid 0 or 1. So you use -ylog(yhat) at every point and try to minimize it so that you can best predict where Harry Potter is in the sentence.



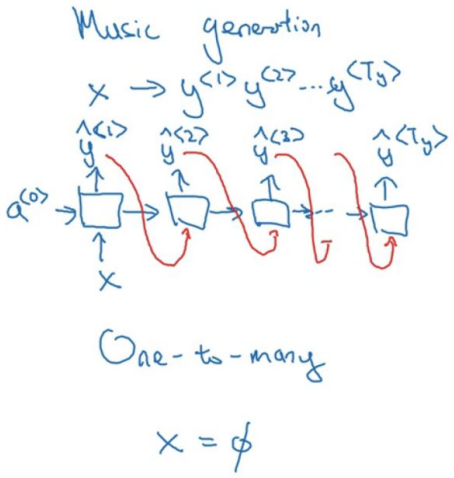
ONE to ONE is vanilla neural network. asd

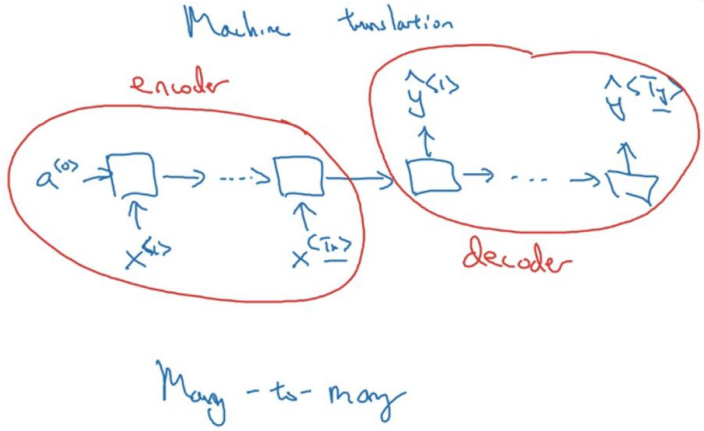
Examples of RNN architectures



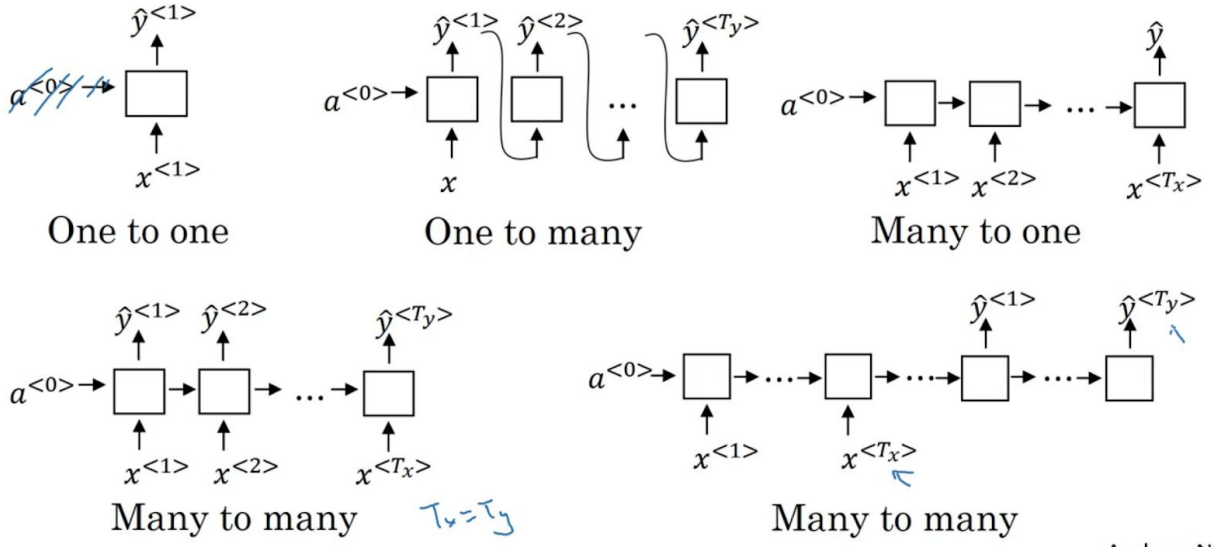


Examples of RNN architectures





Summary of RNN types



LANGUAGE MODELLING:
Identical sentences, which one is more likely?

What is language modelling?

Speech recognition

The apple and pair salad.

→ The apple and pear salad.

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

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

$P(\text{Sentence}) = ?$ $\quad P(y^{<1>}, y^{<2>}, \ldots, y^{<T_y>})$

Language modelling with an RNN

Training set: large corpus of english text.

Tokenize

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

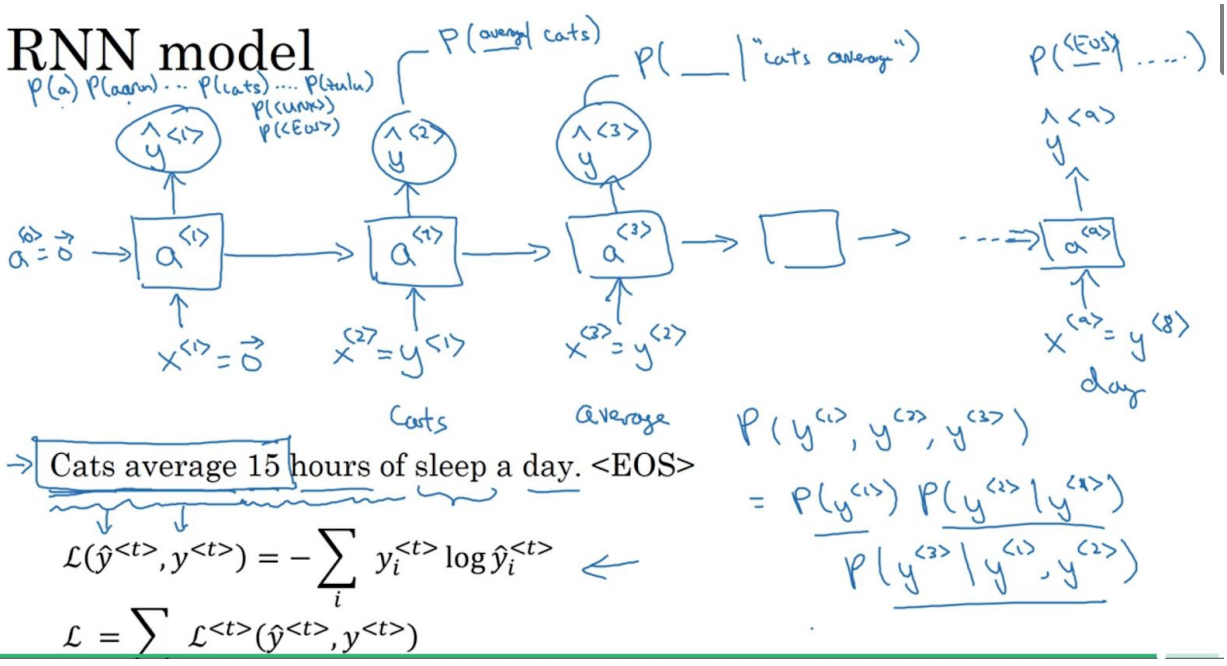
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{(8)}$ $y^{(9)}$

The Egyptian ~~Mau~~ is a bread of cat. <EOS>

<UNK>

10,000

RNN model

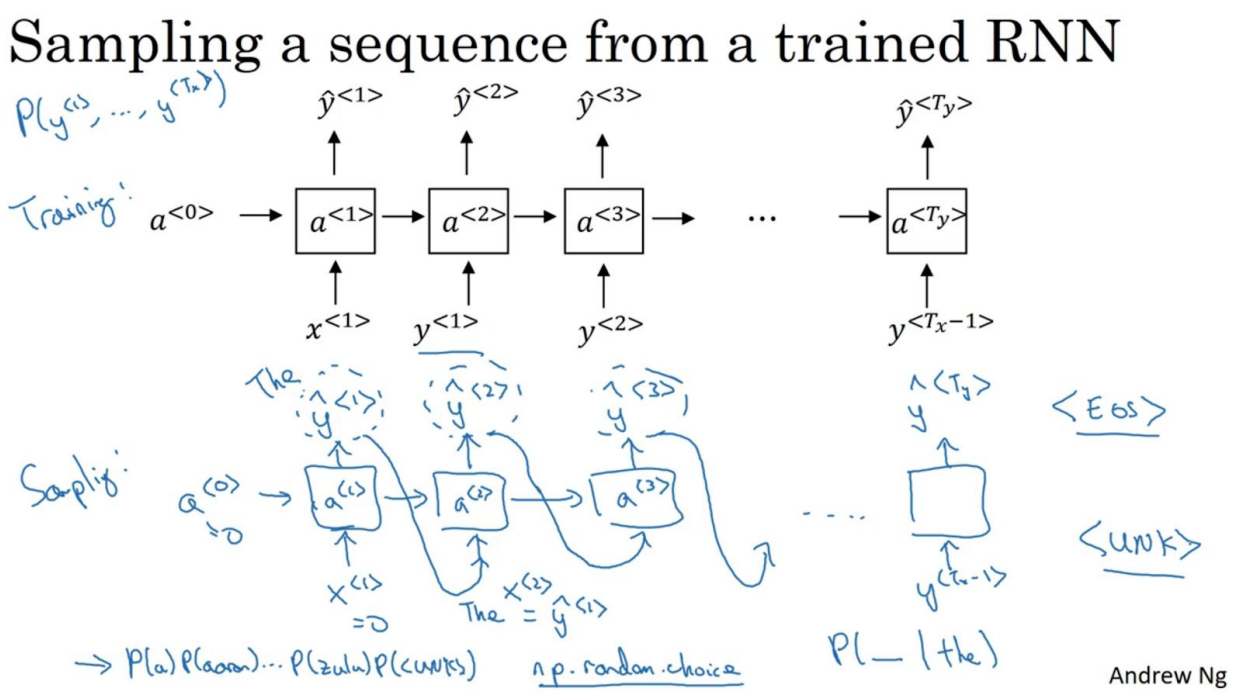


$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

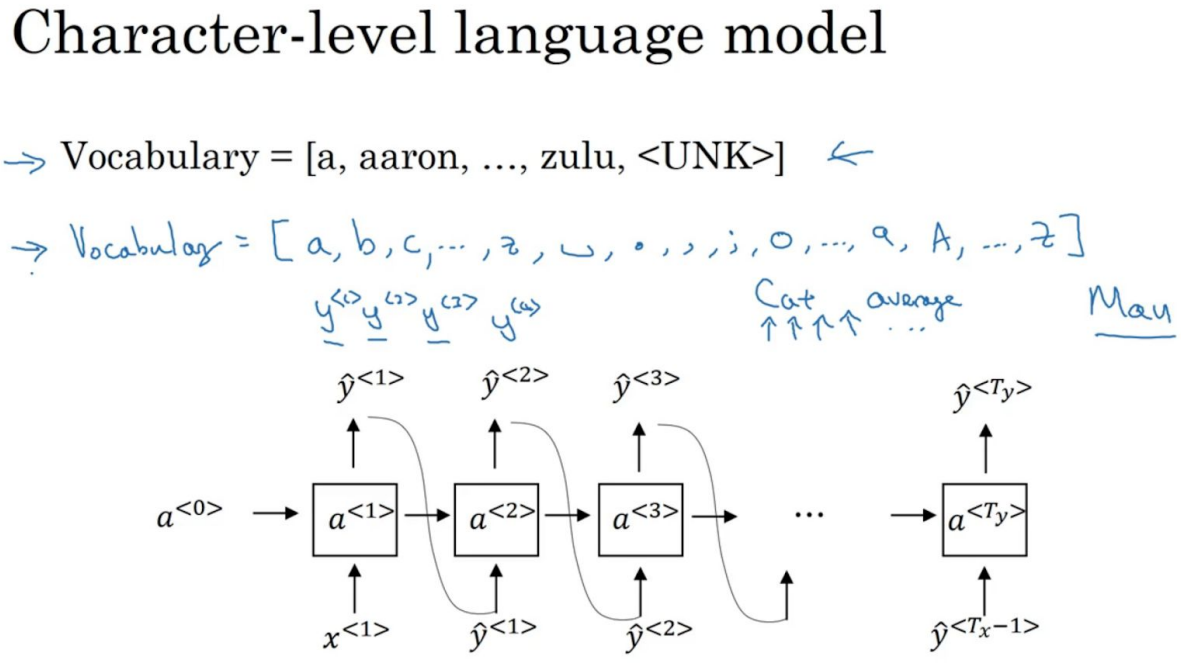
$$\mathcal{L} = \sum \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>})\, P(y^{<2>} \mid y^{<1>})\, P(y^{<3>} \mid y^{<1>}, y^{<2>})$$

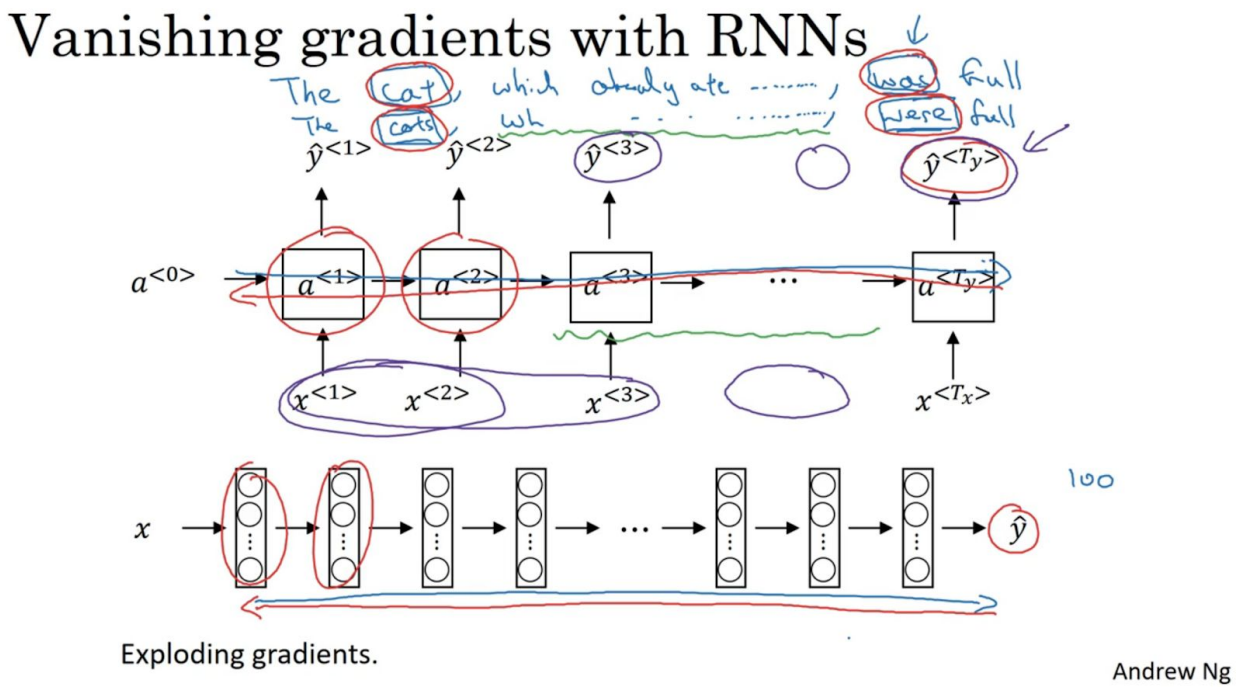
GENERATING A RANDOM SENTENCE FROM RNN:



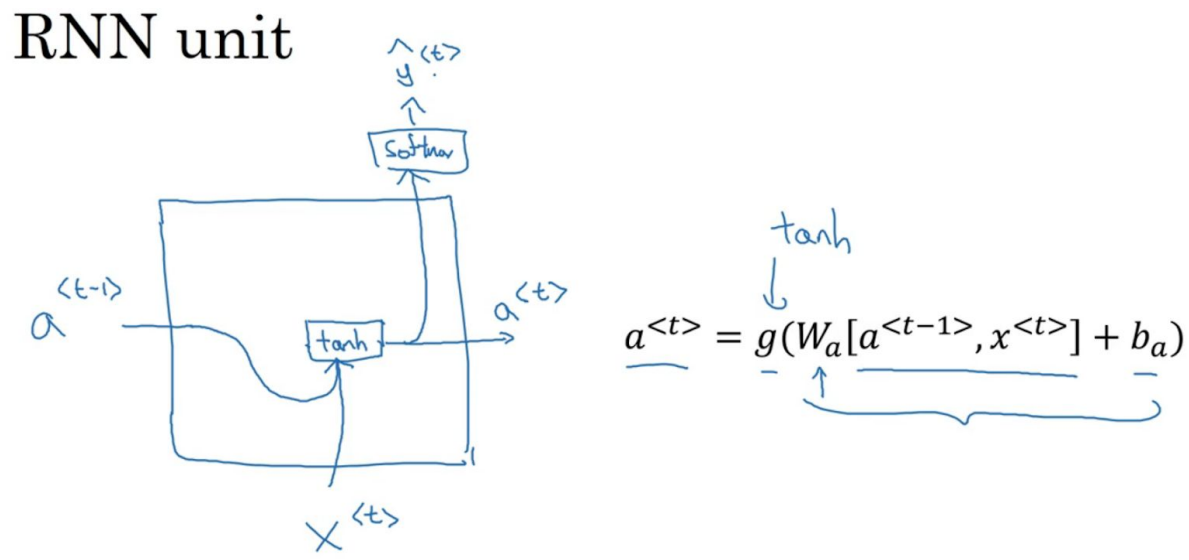
More expensive and challenging as 2 dozen or so characters are common for a few words. So not so common.



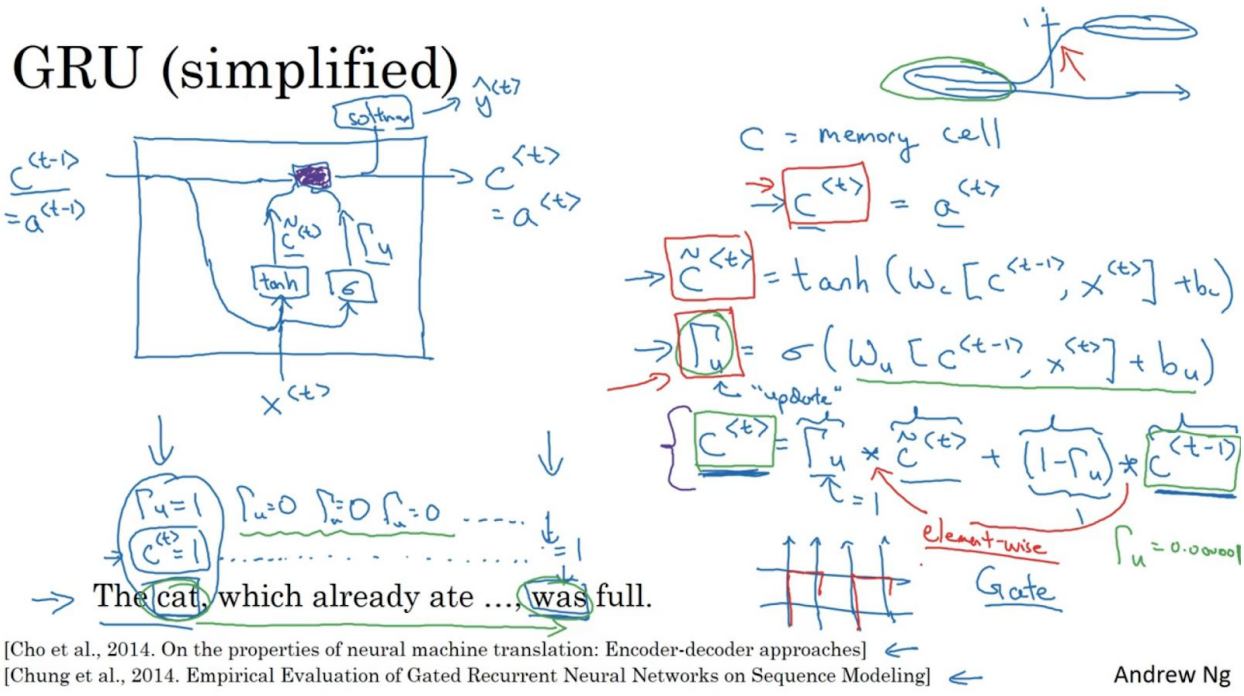
Gradient clipping can solve for gradient exploding. But vanishing gradient is harder to solve.



RNN Unit so far:



Basic difference is that you now create a candidate which gets either “chosen” or “ignored” depending on another parameter the Gate which also gets trained separately.
Gate. candidate + (1-Gate)* Previous candidate is used to decide the final cell state. This basically means that how much you should ignore this value compared to memorizing and continuing previous value.



Because the gate and cells can be multiple dimensions, you can keep the cat in one dimension, food in another dimension etc.

In GRU the vanishing gradient problem does not exist because the gate can be very very small like 0.000001 basically just storing the previous value of cell state -- so no gradient vanishing!

Full GRU

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

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

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

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

Gamma R (r stands for relevance) is how much important is the current set of inputs and previous inputs at all. Its a very iterative approach to come up with these architectures - no fixed set. Its more empirical!

LSTM (more general form of GRU):

GRU and LSTM

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

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

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

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

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

c<t-1> can also be used as peephole connection.

LSTM in pictures

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

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

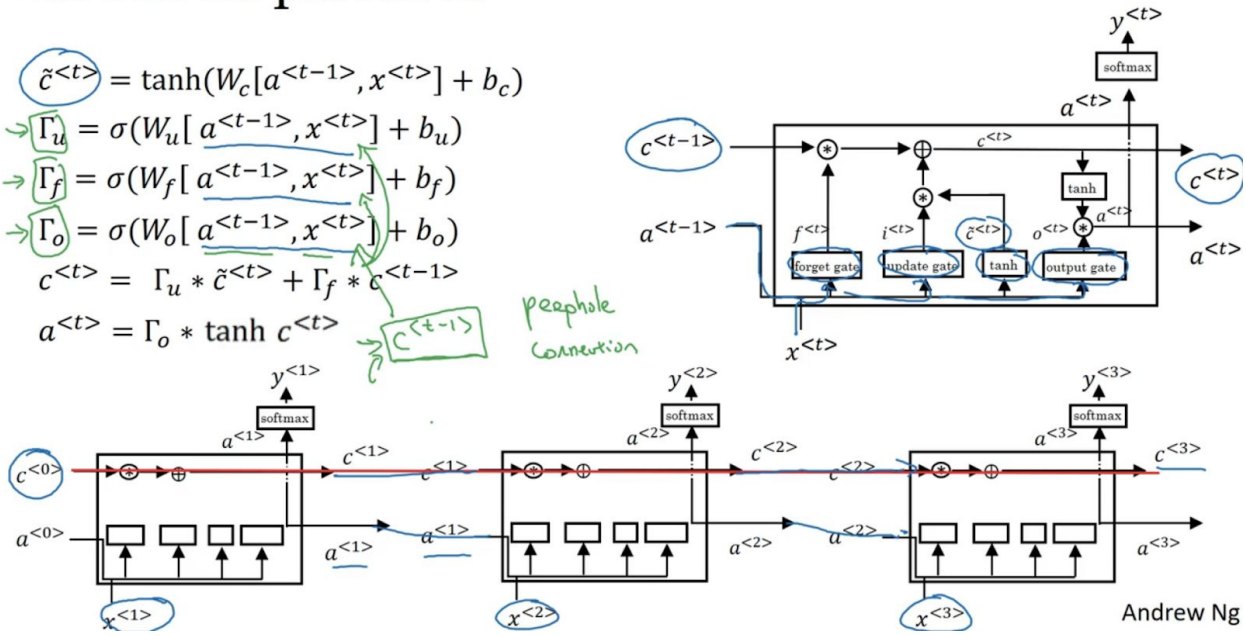
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

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

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

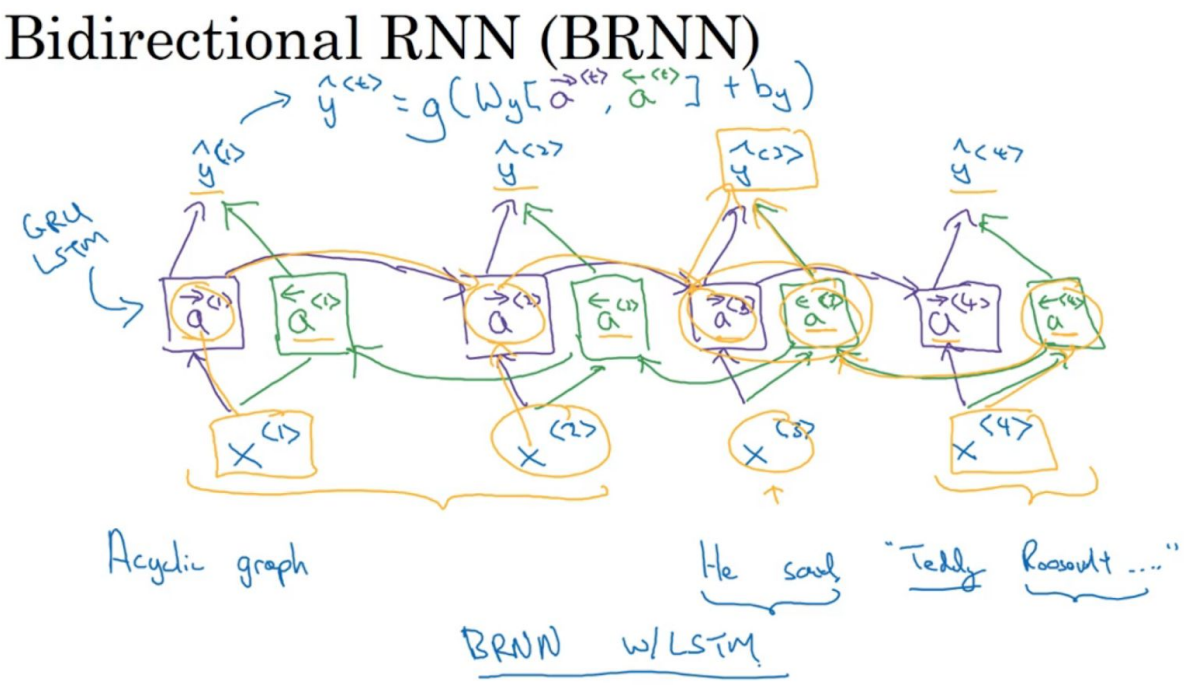
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

peephole connection



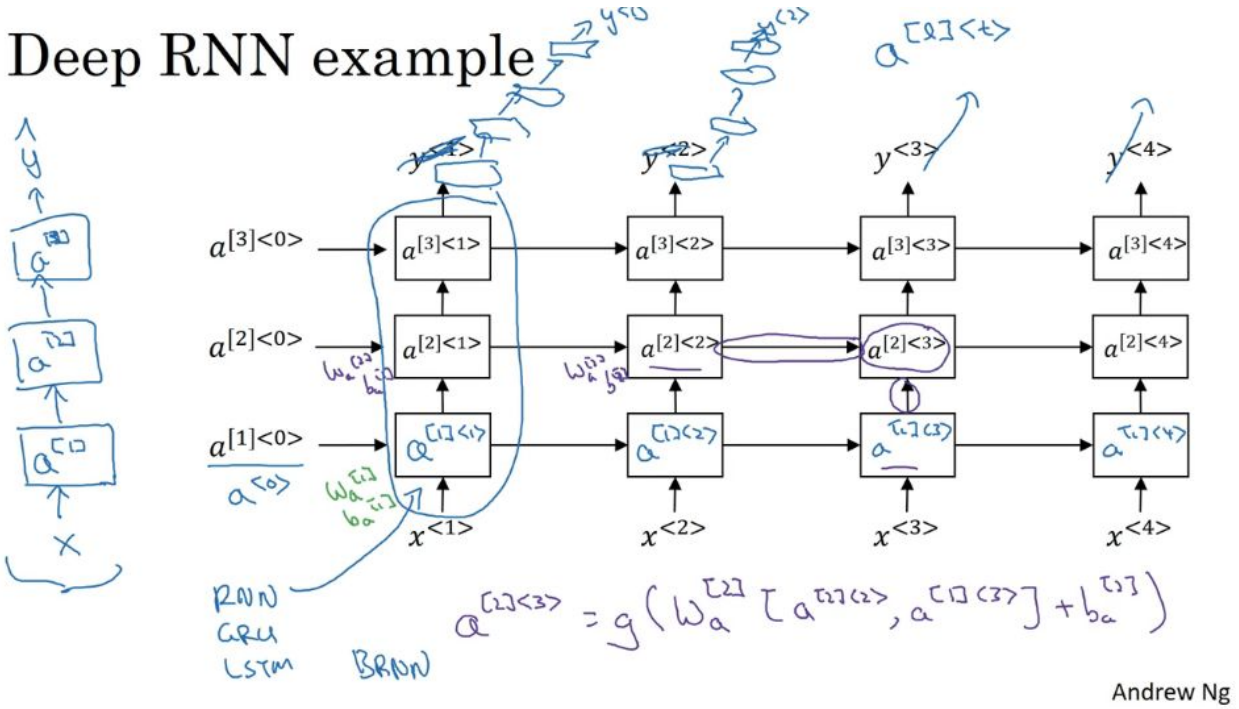
Cell state flows through -- so there is no vanishing gradient!

LSTM is usually the first choice, but GRUs more and more often these days as there are only 2 gates compared to 3 gates so computationally more efficient.



NLP tasks BRNN with LSTM is very common! Disadvantage is that you need entire sequence, you need to wait for person to stop talking before making a prediction. For complex speech recognition systems, you need more complex models.
For tasks where you can get entire sentence or sequence, BRNN is quite powerful and good!

DEEP RNN you can stack LSTM or GRU or RNN or BRNN and then hook them up in the end with a generic long deep network which does not have any temporal info.



Andrew Ng

WORD EMBEDDINGS (WEEK 2)

Word representation

V = [a, aaron, …, zulu, <UNK>]

$|V| = 10{,}000$

1-hot representation

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

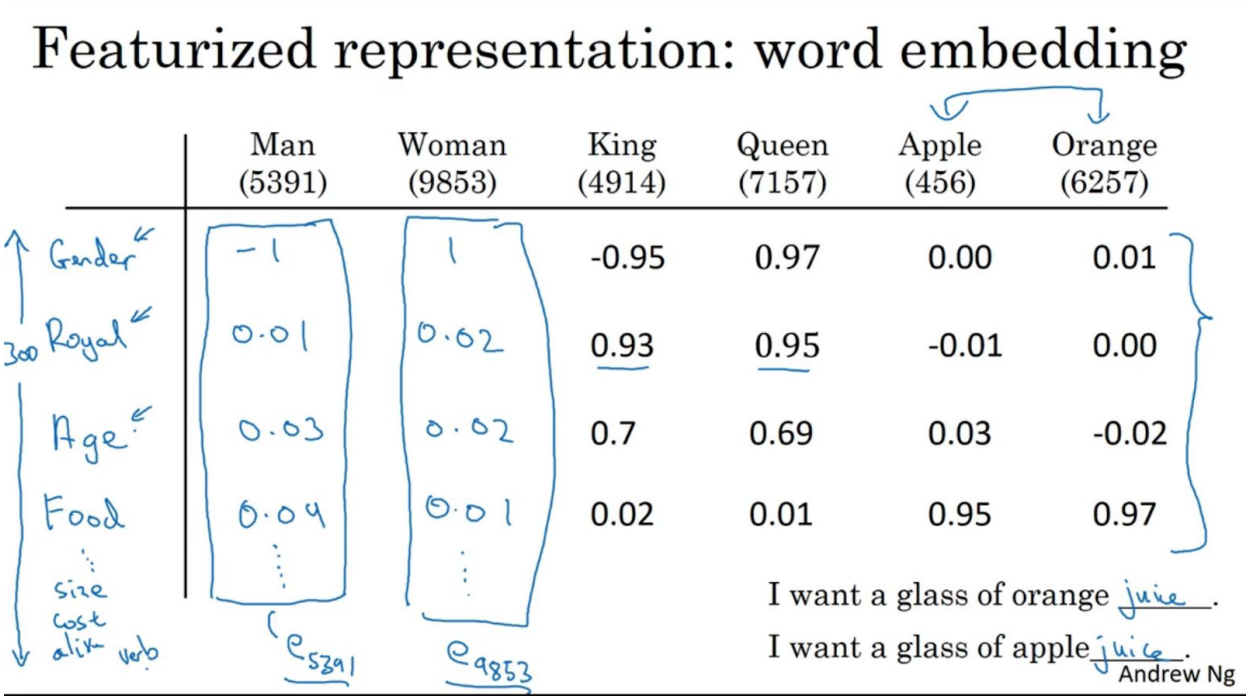
O_{5391} O_{9853}

I want a glass of orange juice.

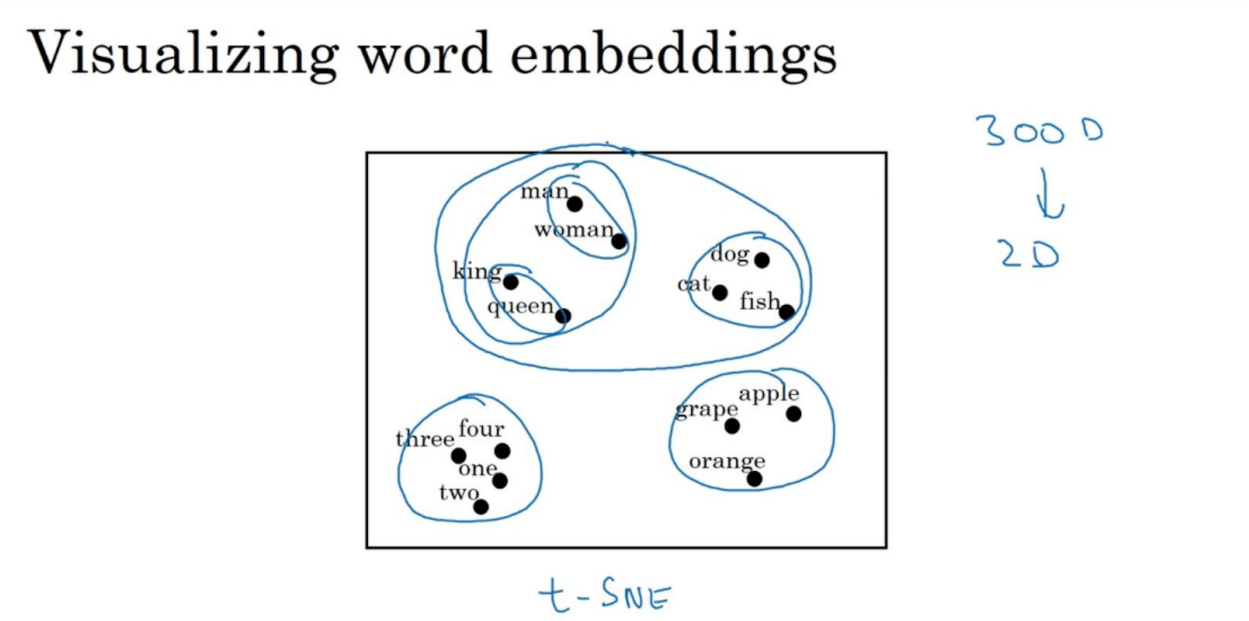
I want a glass of apple ____.

Andrew Ng

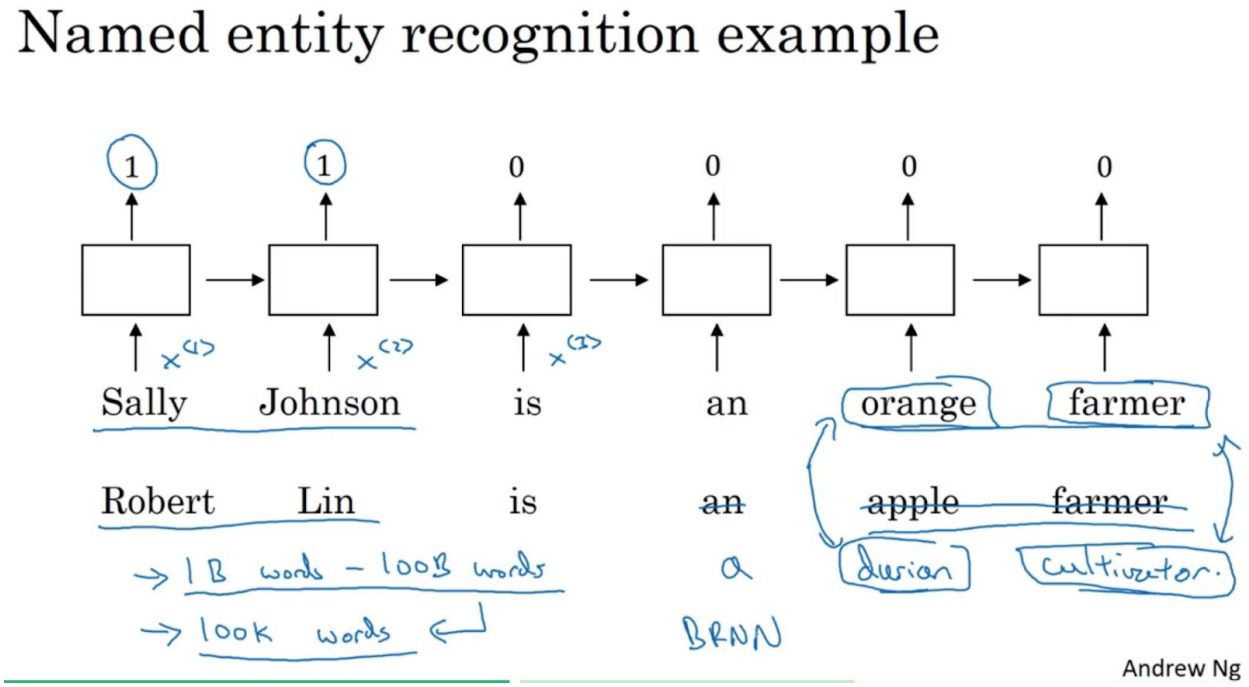
Can’t transfer learning from orange juice to also apple juice with One Hot Encoding. Every word is different. Another disadvantage is that the word vectors are really very sparse.



Above instead of one hot -- you are using features. So 100 dimensional vector is basically 100 features like gender, age, food,size, color etc. and a dot product of the two (OR some other analogy function like subtraction _ more later_) can give “nearness”.
So apple and juice while they differ in colors, might have mostly similar features and hence can transfer learning about orange juice into apple juice.



300 D to 2D so orthogonal dimension in same space



TRANSFER LEARNING in RNNs is about using someone else's word embeddings from 1B or 100B words and use it for your scenario of 100k or so words. In real life you should use BRNN to get full context.

Transfer learning and word embeddings

1. Learn word embeddings from large text corpus. (1-100B words)

   (Or download pre-trained embedding online.)

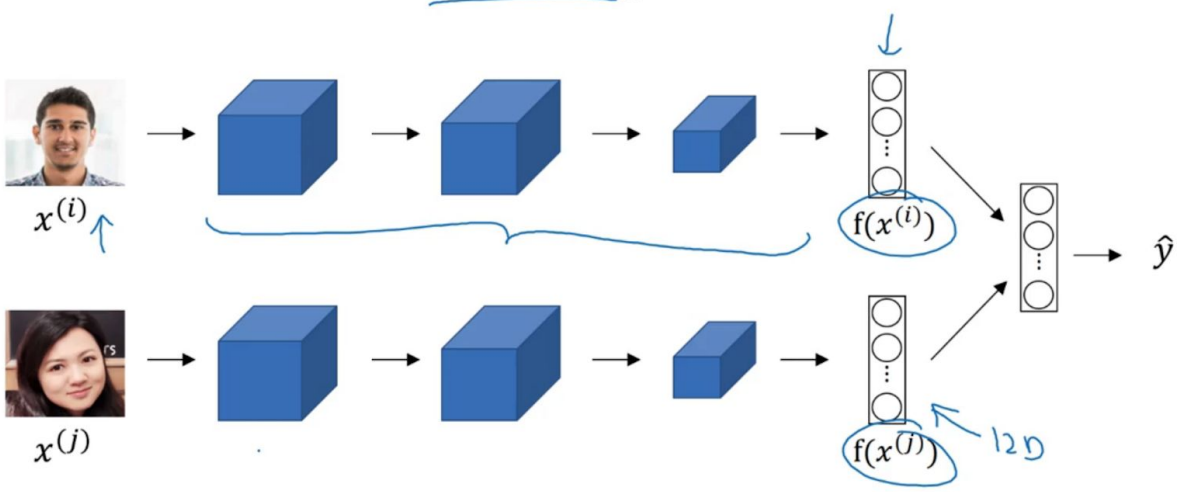
2. Transfer embedding to new task with smaller training set. (say, 100k words) → 10,000 → 300

3. Optional: Continue to finetune the word embeddings with new data.

If Step 2 has many many training data, then can do Step 3.

Word embeddings is useful for NLP tasks like Named Entity Recog, text summarization, coreference resolution, parsing etc. But not very useful for language modeling, machine translation (esp. if there is a lot of data)

Relation to face encoding (embedding)



IN CNN literature, you can compare if the two photos are exactly the same. In faces, you can continue to do this for unseen images. But for text its a fixed vocabulary, everything else is just UNKnown word.

Analogies

	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_{5391} e_{man} e_{woman}

Man → Woman as King → ? Queen

$e_{man} - e_{woman} \approx e_{king} - e_?$ $e_?$ ← queen

$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

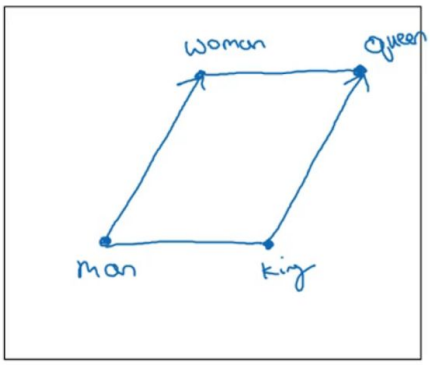
$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Mikolov et. al., 2013, Linguistic regularities in continuous space word representations]

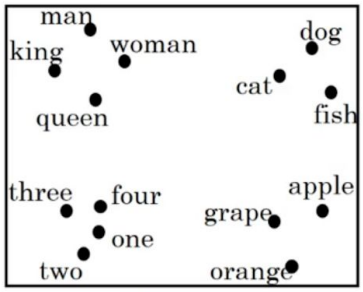
Andrew Ng

Really influential paper!

Analogies using word vectors



300D



3000 → 2D

t-SNE

$e_{man} - e_{woman} \approx e_{king} - e_?$ e_w

Find word w: $\arg\max_w sim(e_w, e_{king} - e_{man} + e_{woman})$

30 - 75%

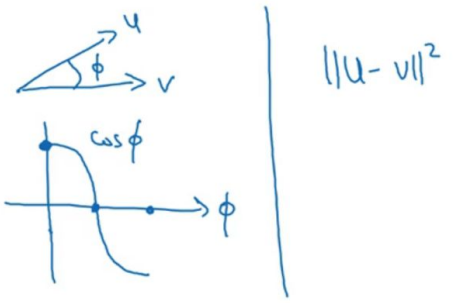
Andrew Ng

Please -- the parallelogram is more common in 300D Space! Not iN TSNE!! Refer to original d dimensional space for cosine similarity

Cosine similarity

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

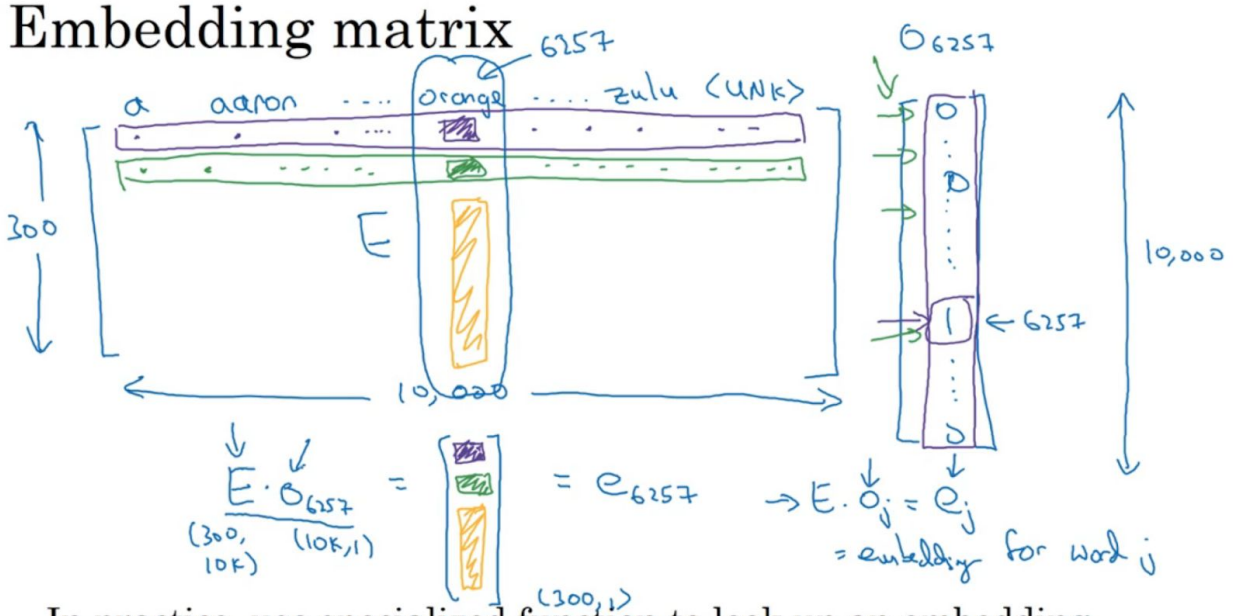
$$Sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$



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

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

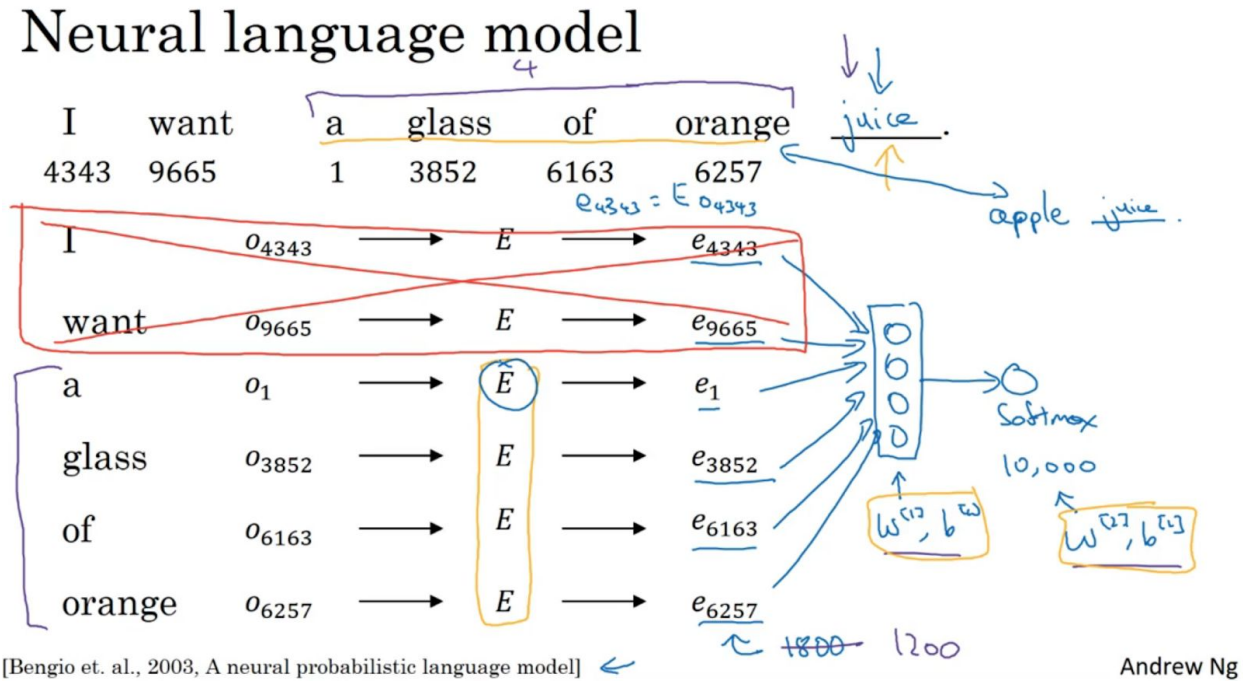
Embedding matrix



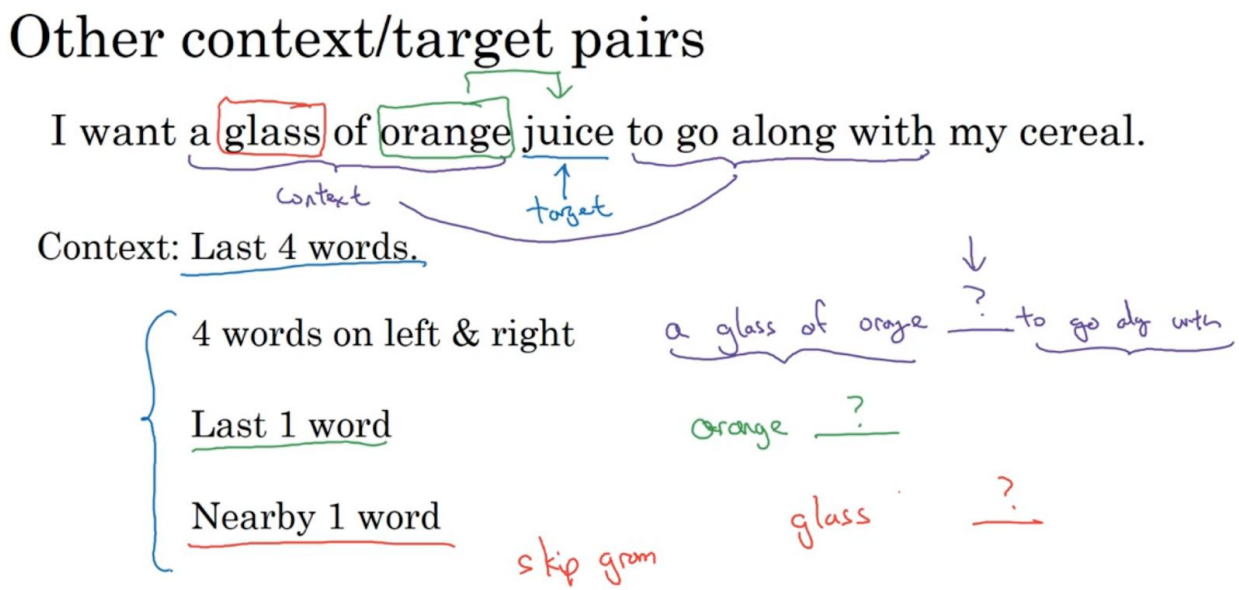
In practice, use specialized function to look up an embedding.

Andrew Ng

IN reality, you use Embedding layer in Keras which just calls the column you want instead of doing this so expensive calculation.

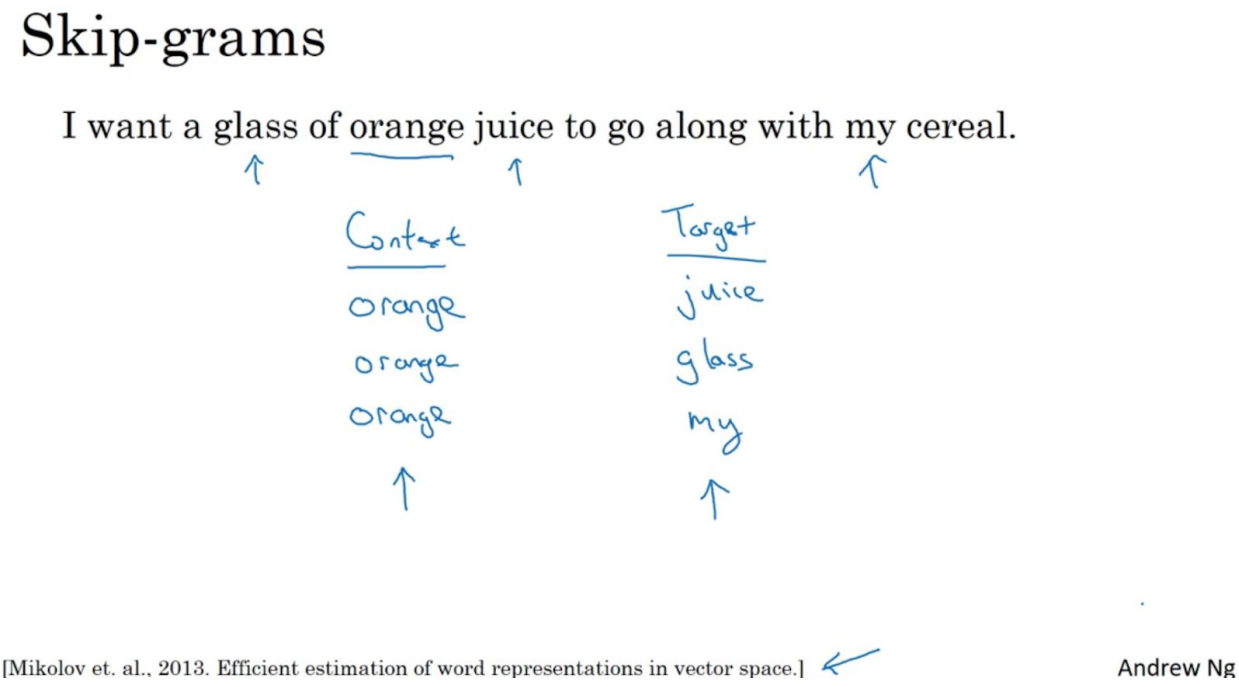


Use backprop to learn E and the weights. Stack all the e vectors - pass to deep layer and then use softmax to predict probabilities of next word.



Language Modelling - last 4 words is best! Like Probability of a sentence, pear or pair salad.

But just for word embeddings all the rest of the context vectors are also good enough. {NEED TO LEARN MORE HERE, SOME DOUBTS, why does softmax layer also need weights? Is this above method good enough to train E for first time or for refinement?}

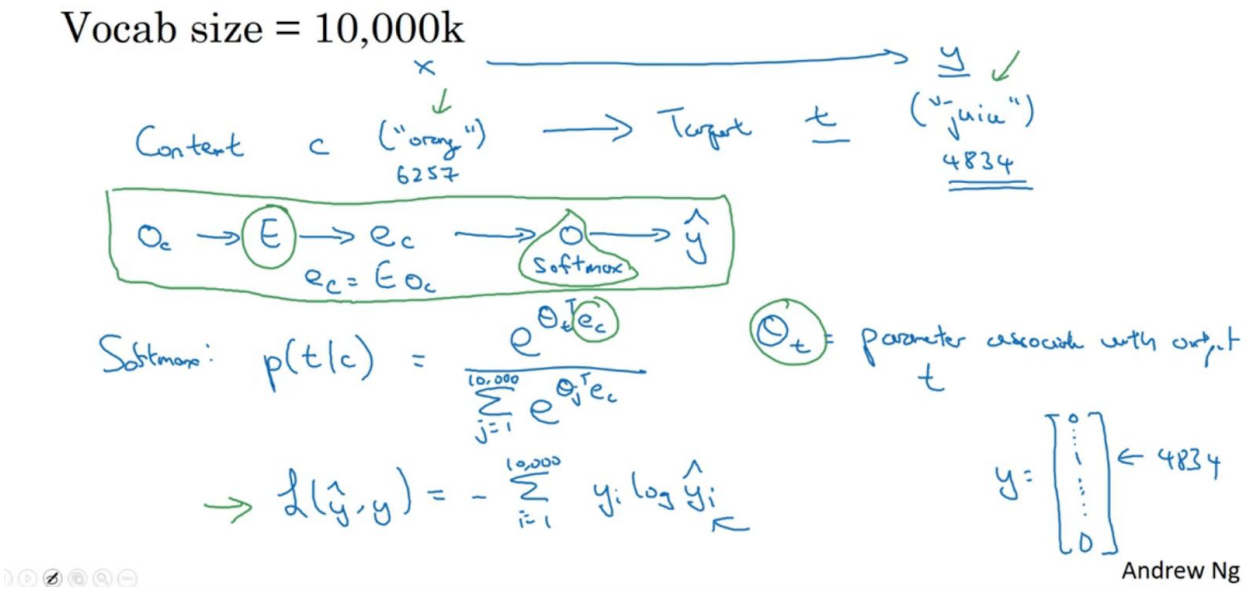


Randomly chosen context word in a window and a randomly chosen target word again in a window

SKIP GRAM MODEL:

In stanford class CS224n, THETA there was v, and word emdeddings was u. Here its theta, and e. Every word has actually 2 vectors associated with them, one as context and one as target. When its the context, thats the word embeddings.

Model



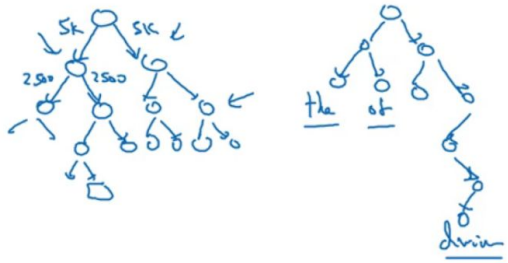
Andrew Ng

Problems with softmax classification

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

Hierarchical softmax.

$\log |V|$



How to sample the context c?

→ the, of, a, and, to, ...

→ orange, apple, durian

e_{durian}

t

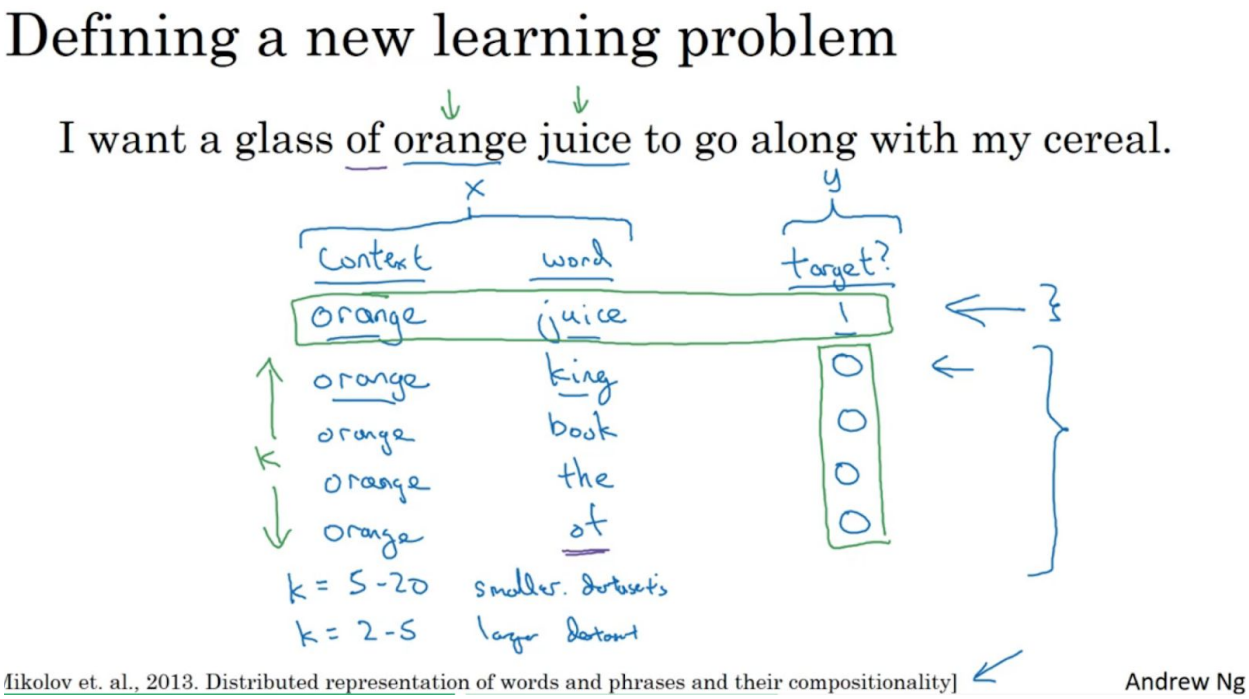
$c \rightarrow t$

$P(c)$

Hierarchical softmax doesn't use softmax but just sigmoid logistic regression stuff to predict whether the target word is in left half or right half of vocab and so on. And hence log|v| complexity. In reality, more common words like the, of etc. are on the left and uncommon words like durian buried deep.

In terms of sampling context word c, dont want too many too, the, of, a etc. so there are ways to sample such that you spend some time on orange, apple etc. too.

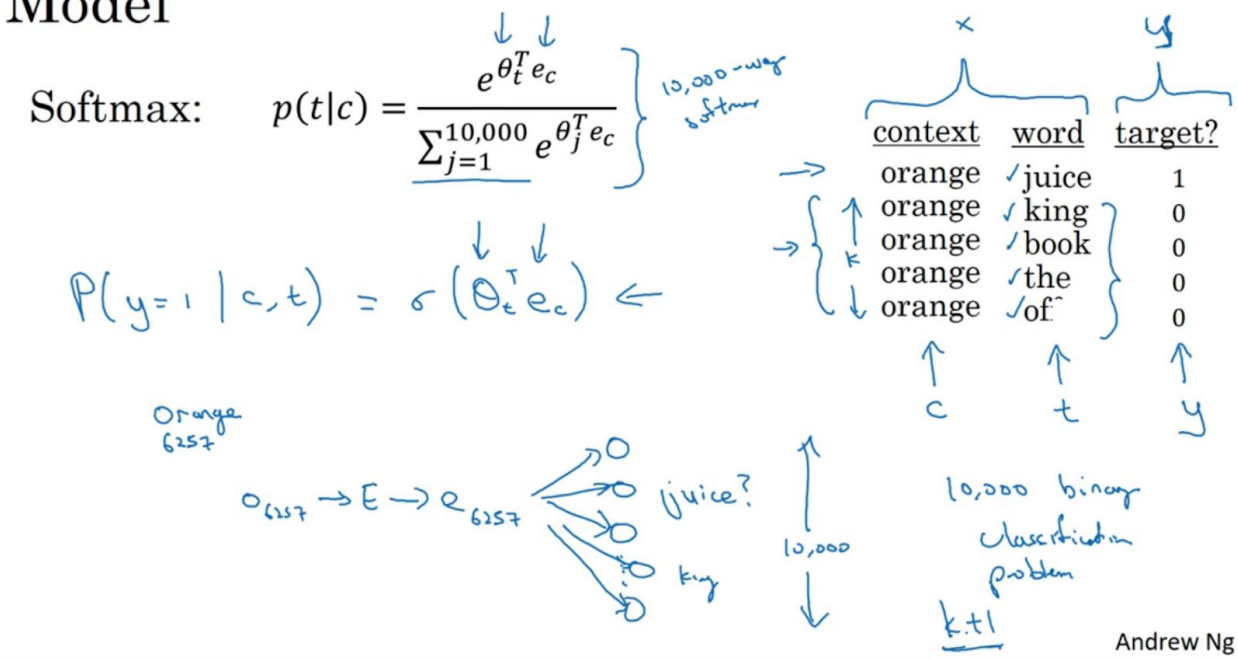
NEGATIVE SAMPLING



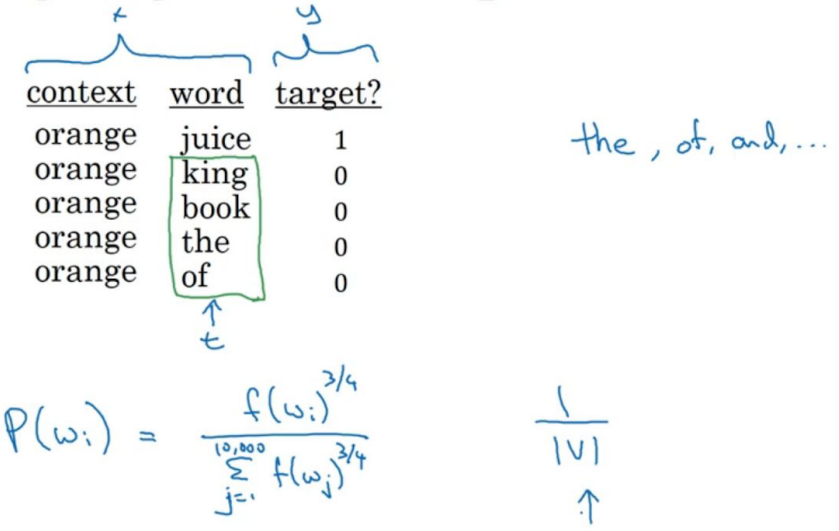
Orange from the window, and one word from window for target 1 , and some negative samples with one word from window and the rest from randomly chosen words in vocabulary.

Instead of 10000 softmax algorithm, you train k + 1 logistic regression models, so its much computationally easier.

Model



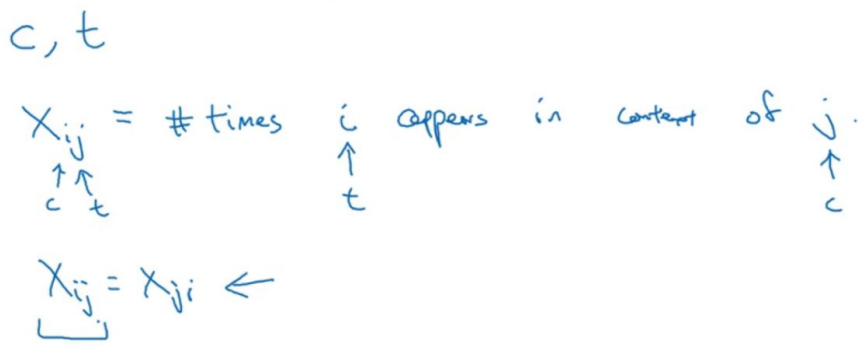
Selecting negative examples



Neither most common negative words and nor least occuring, somewhere in between is better! thats why 3/4th the freq of words

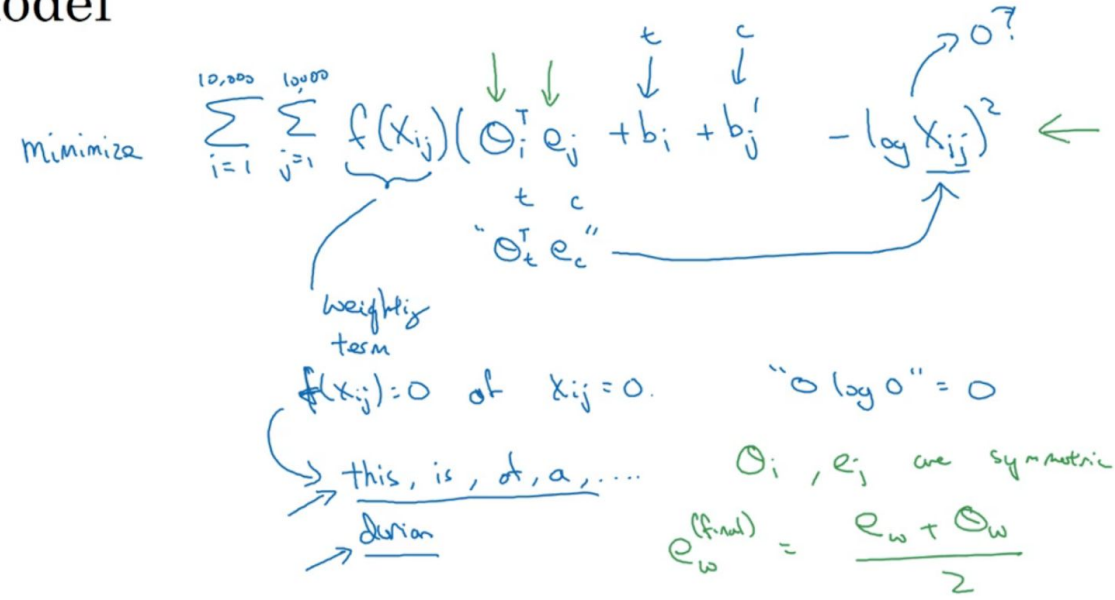
GloVe (global vectors for word representation)

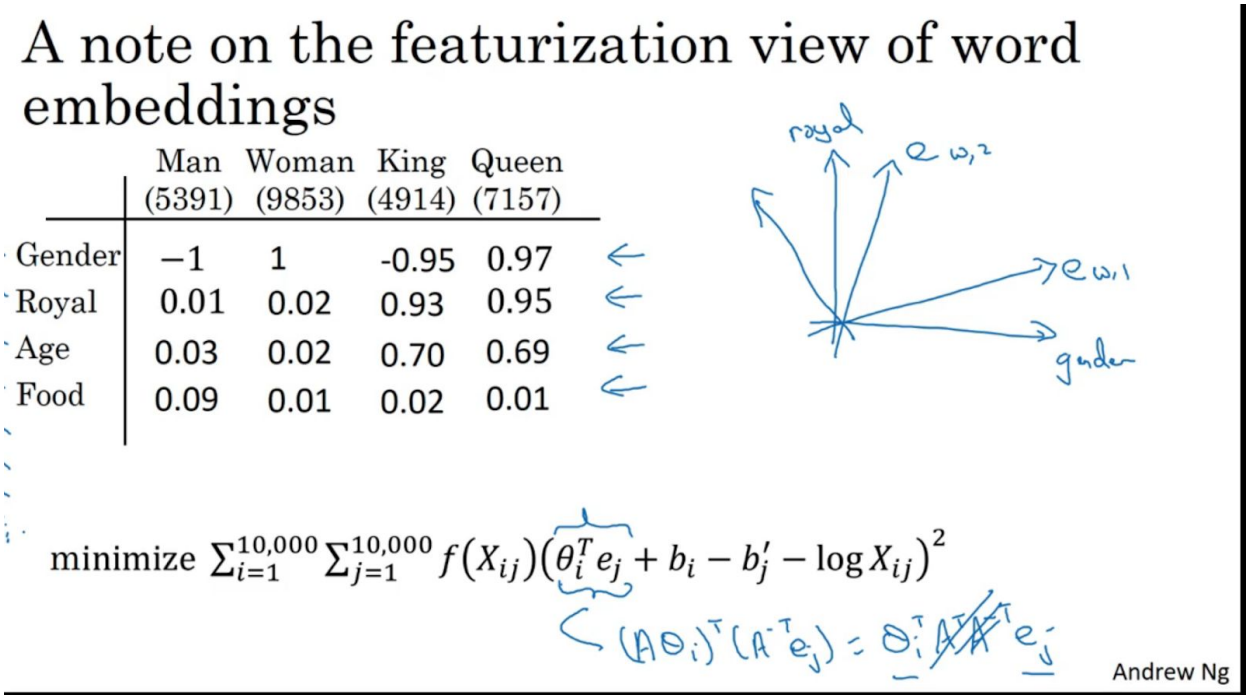
I want a glass of orange juice to go along with my cereal.



GLOVE uses counts of number of times a word appears in context of word j in vocab.

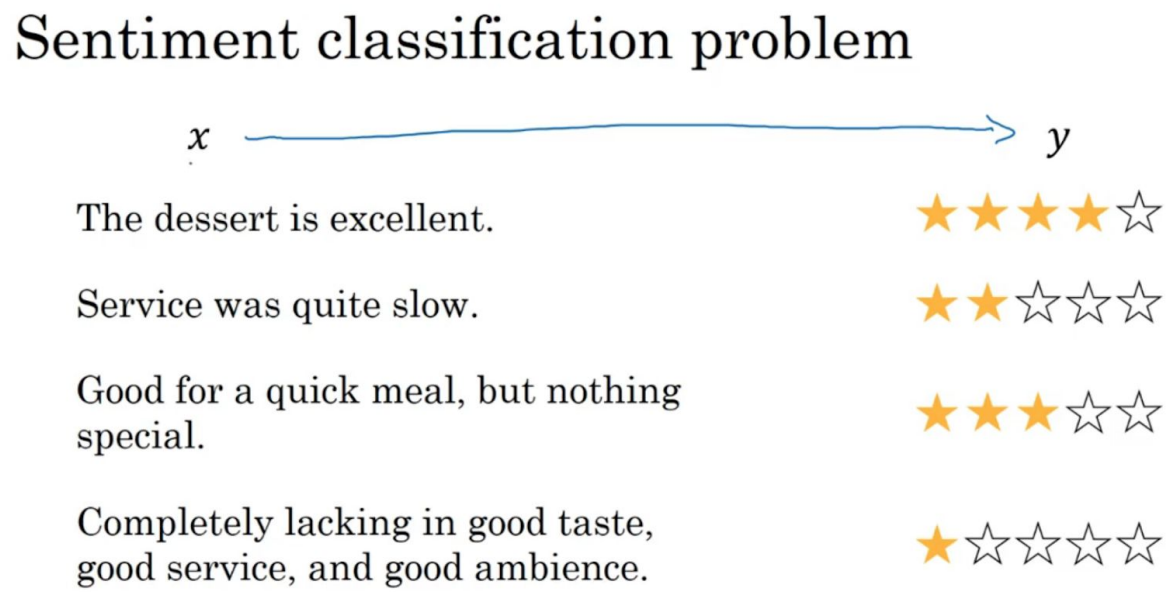
Model



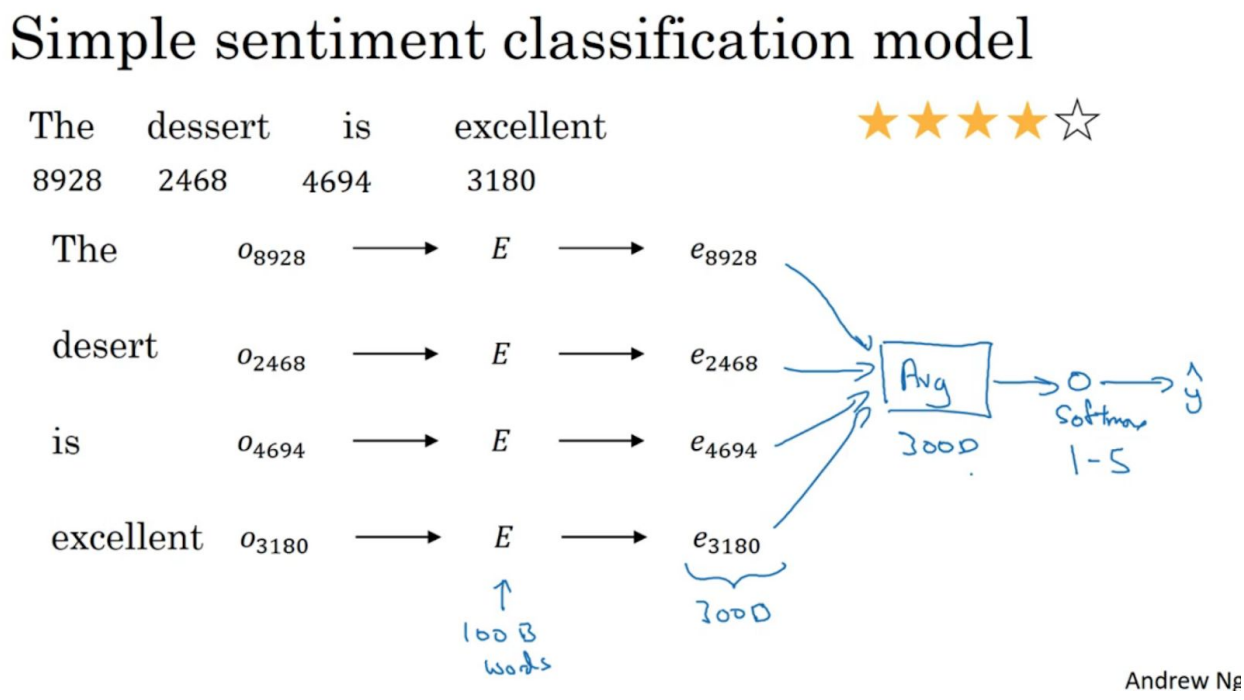


Not always human interpretable, but vector parallelogram math still works!

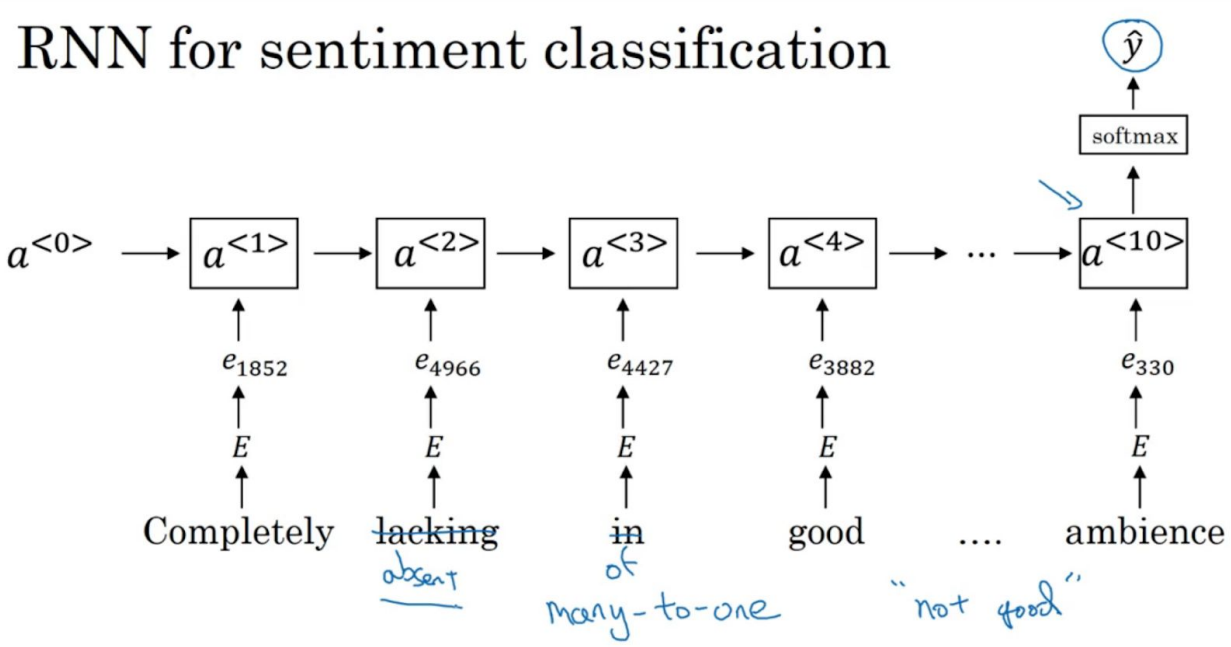
APPLICATIONS OF WORD EMBEDDINGS



You might not have a lot of training examples. Word embeddings can help if your training set is low.



AVerage is good because of 100 or 1000 word reviews, it just averages them and you can softmax the output. But there is no context!



This is better for context!

The problem of bias in word embeddings

Man:Woman as King:Queen

Man:Computer_Programmer as Woman:Homemaker ✗

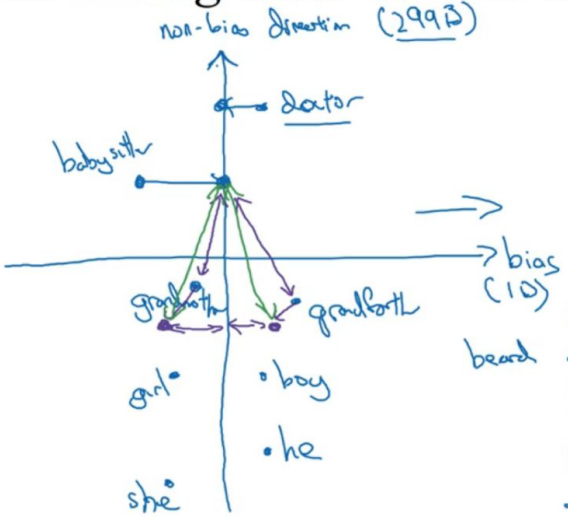
Father:Doctor as Mother:Nurse ✗

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

[Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings]

Andrew Ng

Addressing bias in word embeddings



1. Identify bias direction.

$e_{he} - e_{she}$

$e_{male} - e_{female}$

average

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

3. Equalize pairs.

→ grandmother − grandfather

girl boy

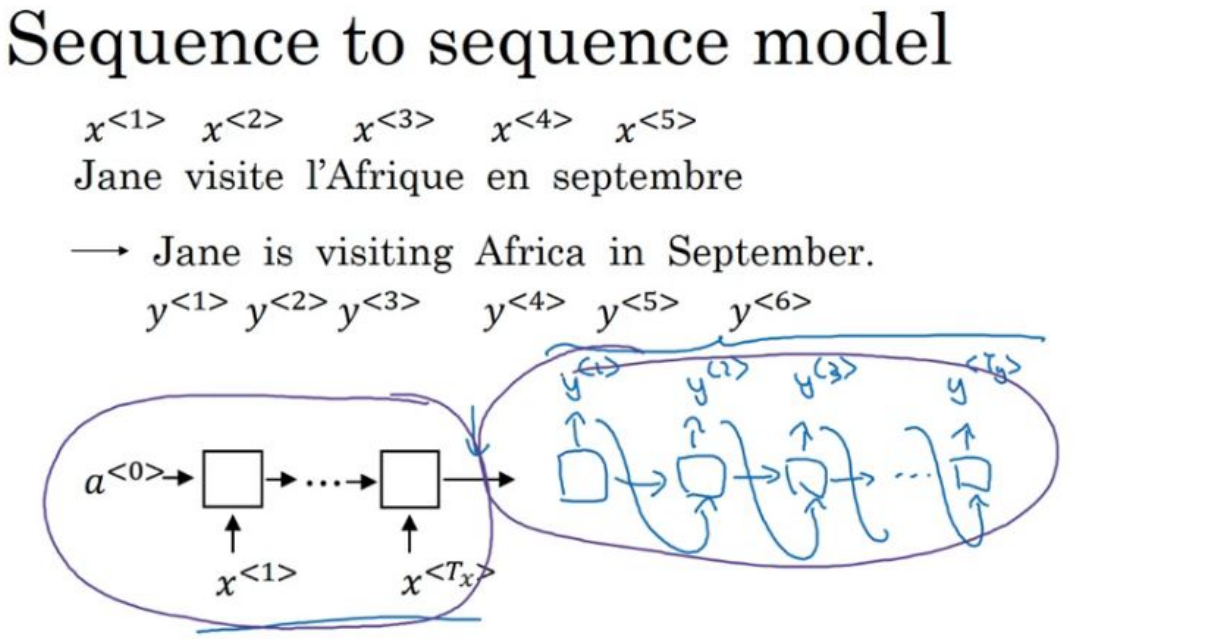
[Bolukbasi et. al., 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings]

Andrew Ng

SEQUENCE to SEQUENCE MODELS (WEEK 3)

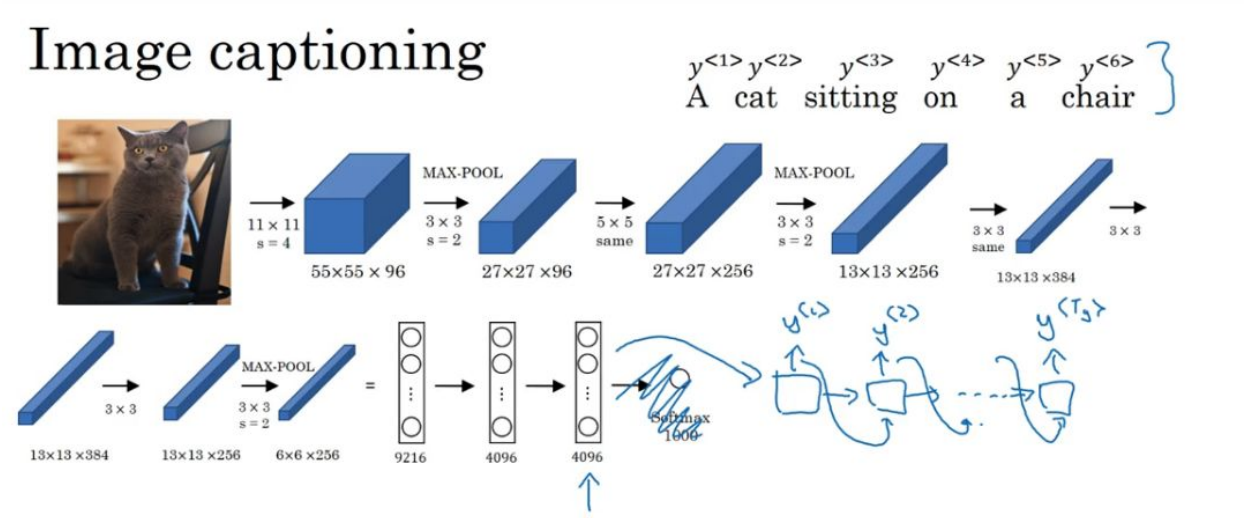
Encoder is a must for many to many (different lengths) RNN.

Encoder to Decoder



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

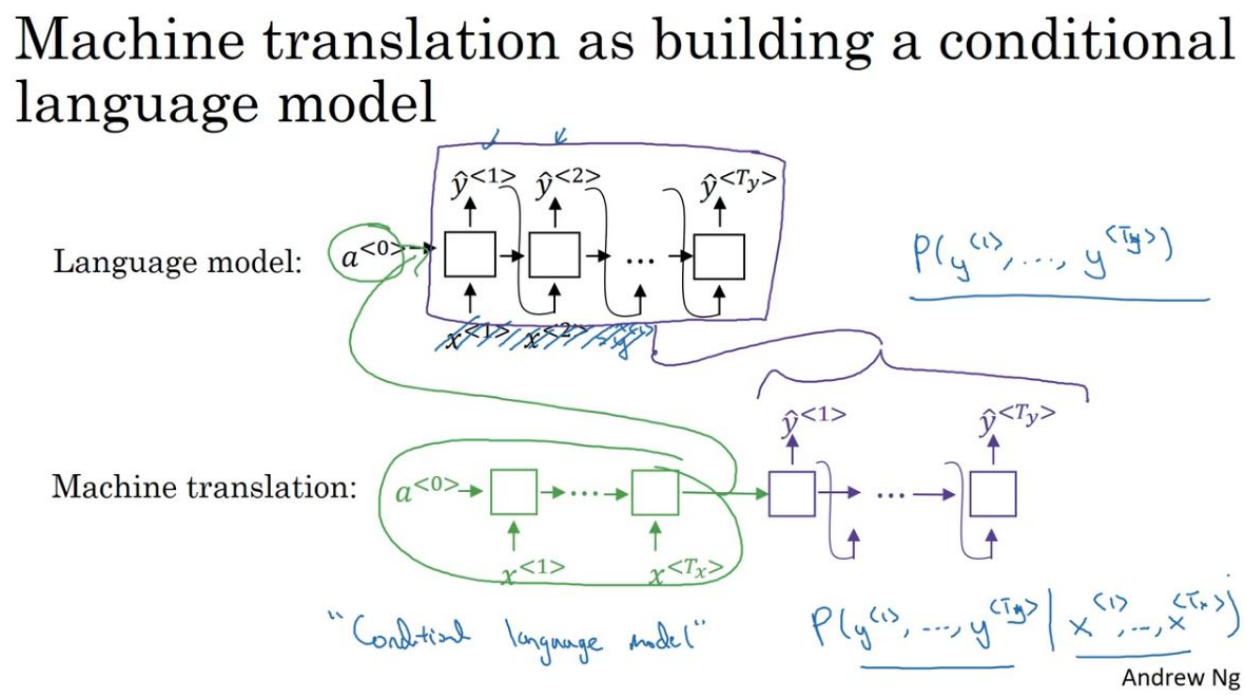
[Cho et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation]



[Mao et. al., 2014. Deep captioning with multimodal recurrent neural networks]
[Vinyals et. al., 2014. Show and tell: Neural image caption generator]
[Karpathy and Fei Fei, 2015. Deep visual-semantic alignments for generating image descriptions]

Andrew Ng

Probability of a general sentence vs conditional probability given a French sentence



Andrew Ng

Finding the most likely translation

Jane visite l'Afrique en septembre.

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

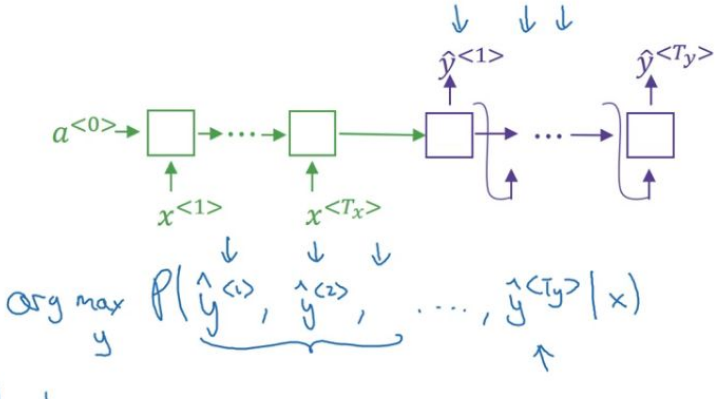
- → Jane is visiting Africa in September.
- → Jane is going to be visiting Africa in September.
- → In September, Jane will visit Africa.
- → Her African friend welcomed Jane in September.

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

CANT SAMPLE AT RANDOM, you want arg max of the probability.

Greedy Search: Pick first word P (y_1|x) and then pick the best 2nd word and then best 3rd word. Does Not work!

Why not a greedy search?



→ Jane is visiting Africa in September.

→ Jane is going to be visiting Africa in September.

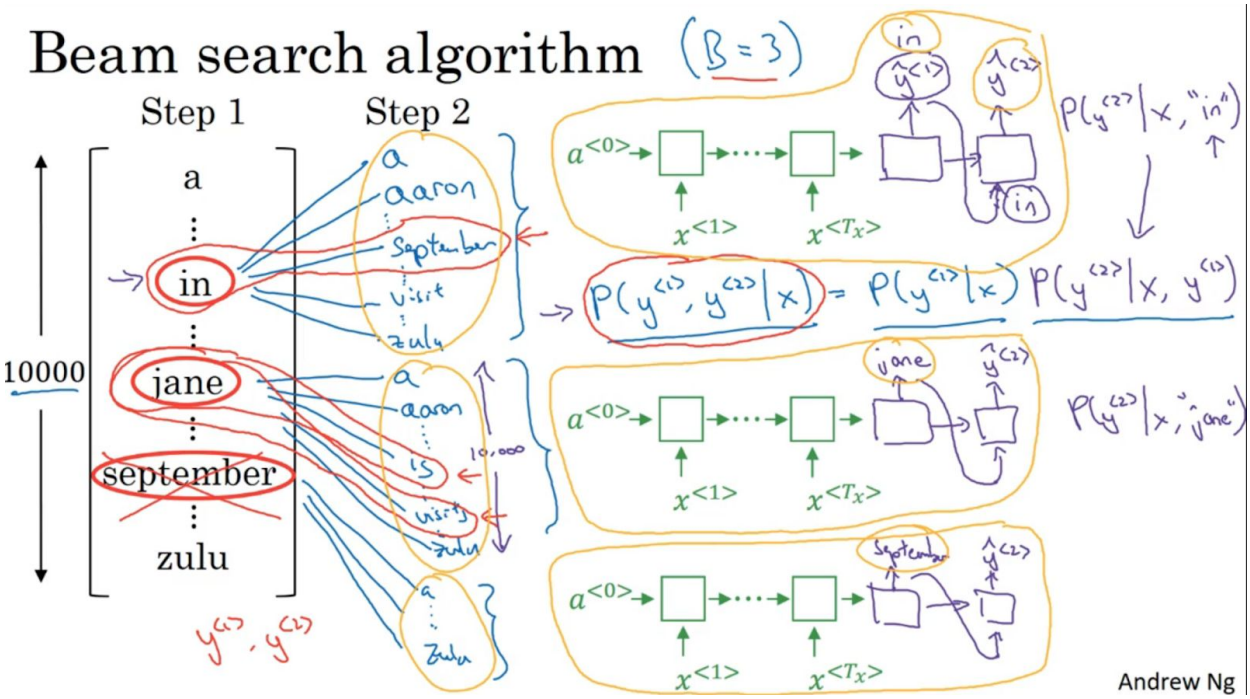
$P(\text{Jane is going} | x) > P(\text{Jane is visiting} | x)$

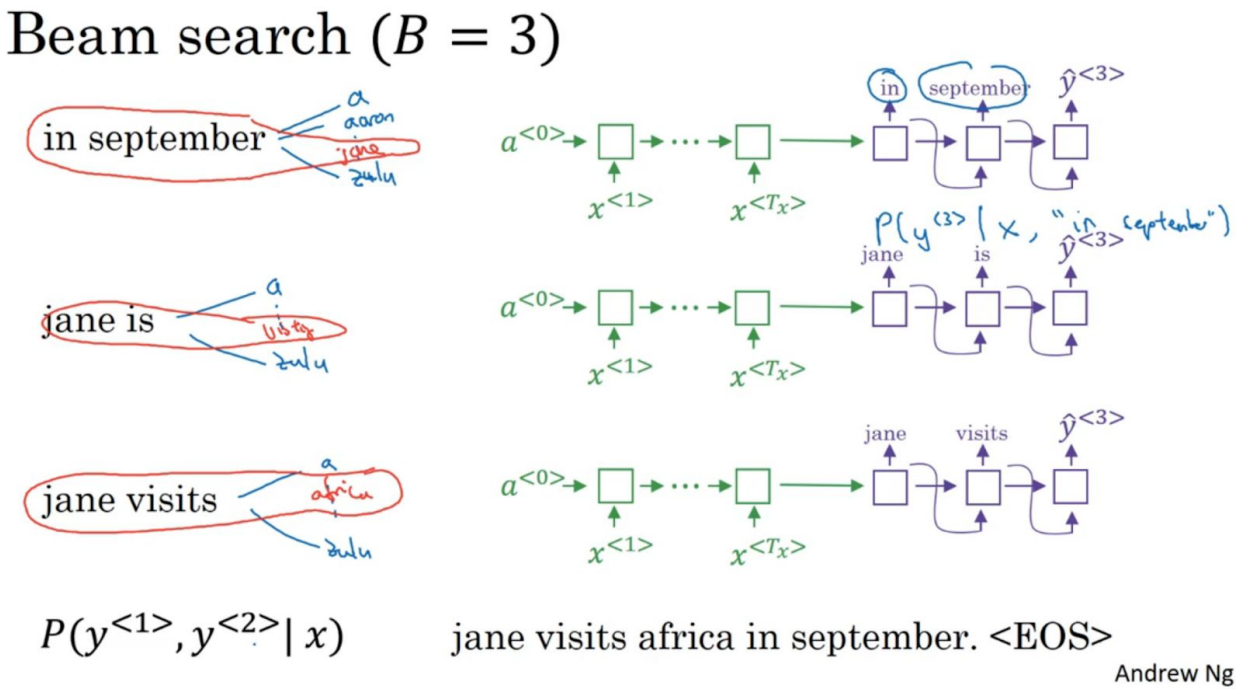
Andrew Ng

Therefore we need beam search that picks max probability of a sentence together!

BEAM SEARCH

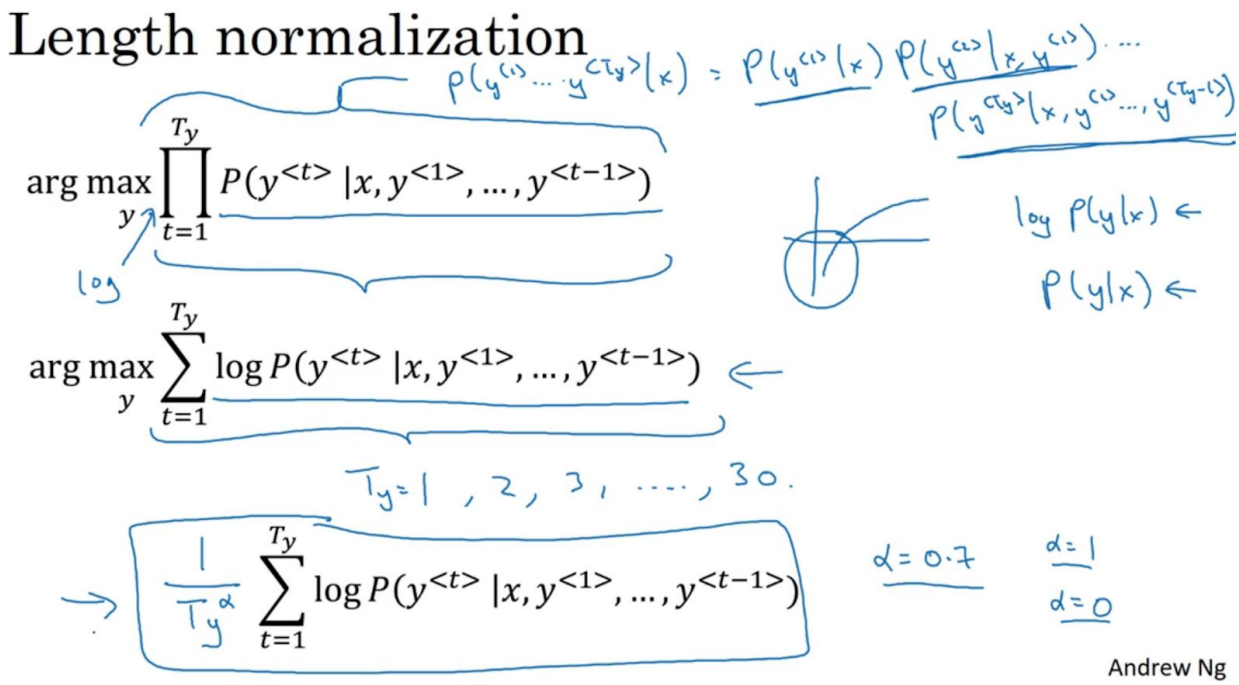
So instead of exploring 10000 ^ 10 possibilities for 10 word sentence, beam search works as follows. It finds top 3 candidates for first word, and then for each word looks at 10k words each for 2nd word. Now it picks top 3 candidates for P(y2,y1|x) by multiplying P(y1|x) from the top 3 list and then new probability. It then picks top 3 pairs again and so on. So the three pairs might be In September, Jane is, Jane goes.





If B=1 then this becomes greedy search algorithm. If B=3 or so it can be better.

Instead of product of probabilities, its better to take sum of p's.
You should also take average of probas as you dont want to penalize longer sentences.



Alpha of 0.7 works pretty well!

Beam search discussion

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

Beam width B?

1 → 3 → 10, 100, 1000, → 3000

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

B of 3 to 10 is common for commercial applications, for research from 100 to 3000.

How to know if RNN or Beam Search is at fault?

Error analysis on beam search

$P(y^*|x)$

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

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

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

Case 1: $P(y^*|x) > P(\hat{y}|x)$ ←

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

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

Conclusion: Beam search is at fault.

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

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.	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

If RNN is at fault, then add training data, regularization or try other architectures.

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")

Only attribute the twice or the maximum number of times it occurs in any one reference.

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 ←

$\frac{4}{6}$

Only credit bigram for the maximum number of times it occurs in the reference sentence.

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. ($\hat{y}$)

$$P_1 = \frac{\sum_{\text{unigram} \in \hat{y}} \text{Count}_{clip}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})}$$

unigram

n-gram

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

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

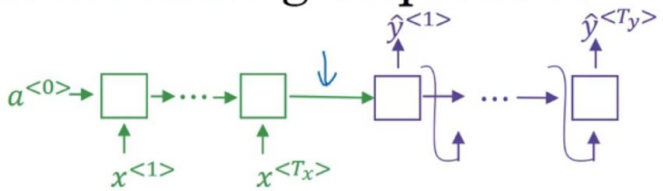
$$BP = \begin{cases} 1 & \text{if MT\_output\_length} > \text{reference\_output\_length} \\ \exp(1 - \text{MT\_output\_length}/\text{reference\_output\_length}) & \text{otherwise} \end{cases}$$

Sometimes RNN can make very small sentences and can get high precision ! SO penalise if short sentences! And not penalize if sentences are bigger than references!

ATTENTION MODEL

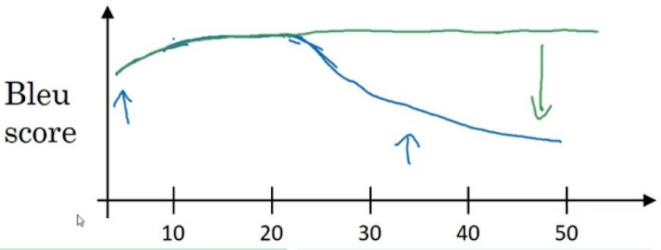
Hard to memorize everything in encoder -- so need something better

The problem of long sequences

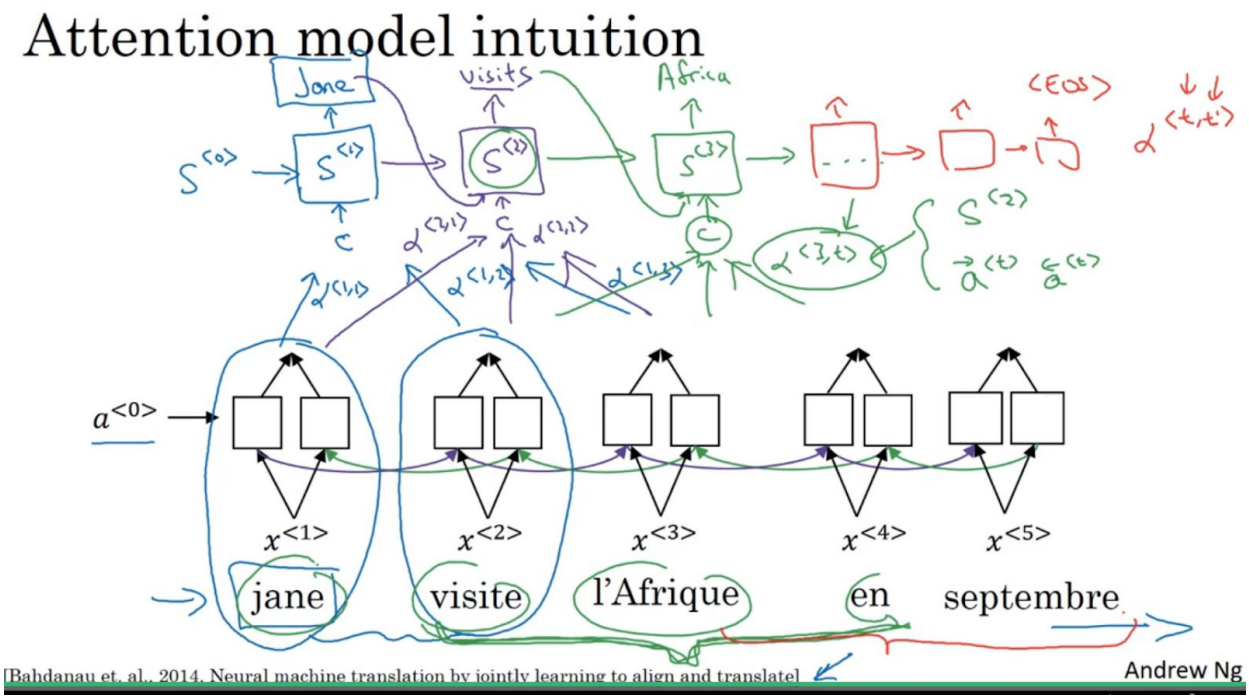


Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

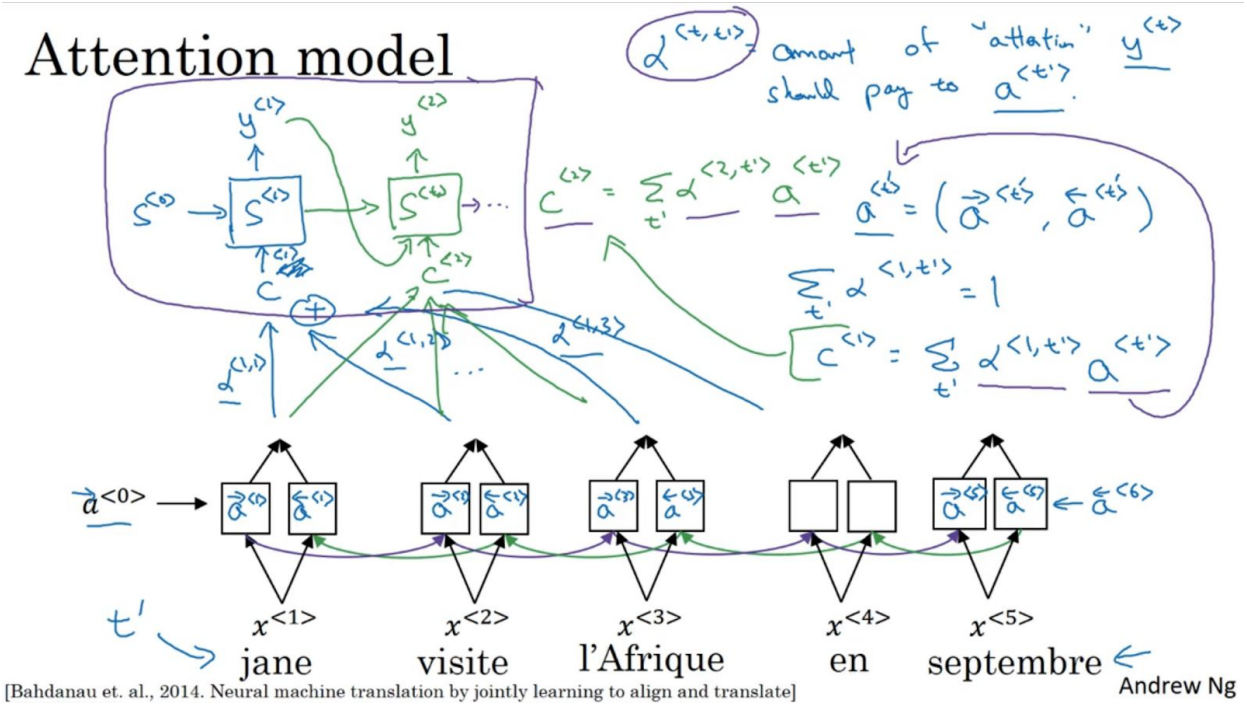
Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



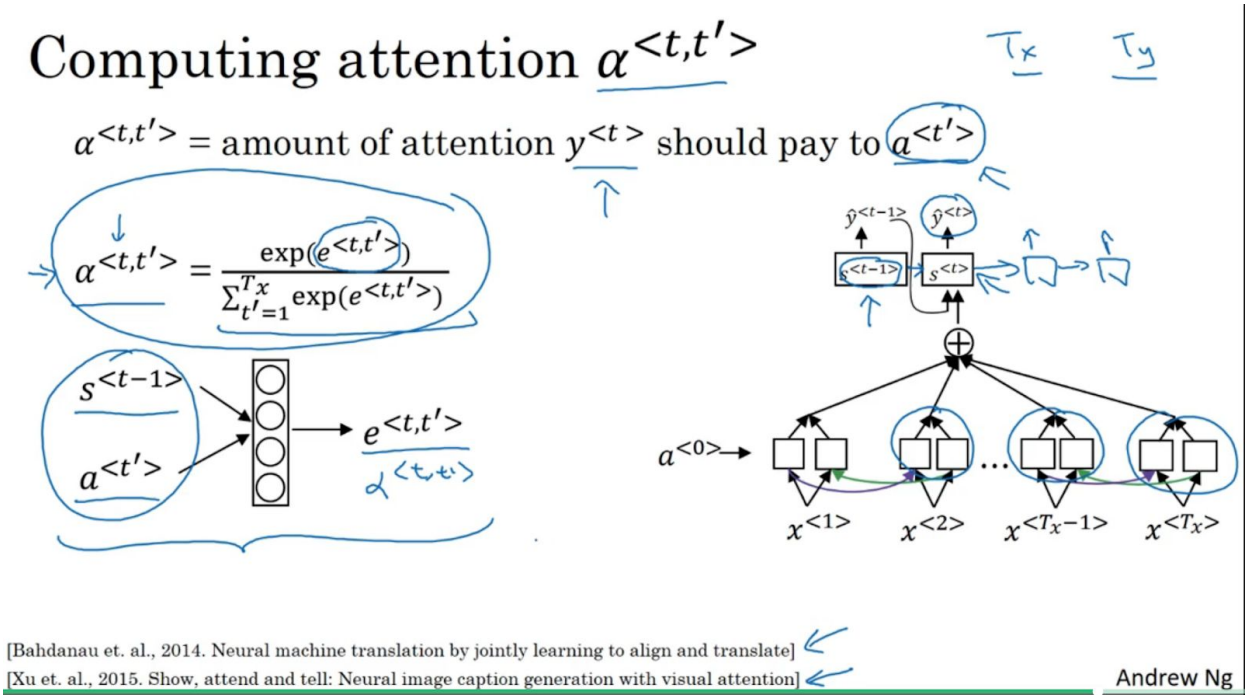
Andrew Ng



Alpha decides the context, which few words to consider while translating the sentence at the current step.



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



[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]

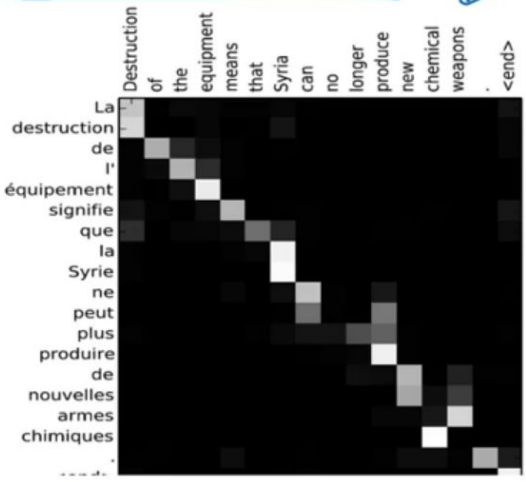
Small neural network is used to train the weights of activations for each step.
One of the papers above show how we can use attention while doing image captioning too -- paying attention to part of images while doing captioning work

Attention examples

July 20th 1969 ⟶ 1969 – 07 – 20

23 April, 1564 ⟶ 1564 – 04 – 23

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

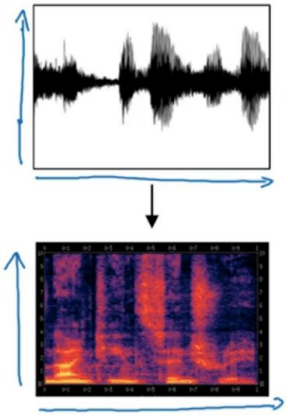


SPEECH RECOGNITION

Speech recognition problem

x ⟶ y

audio clip ⟶ transcript



"the quick brown fox"

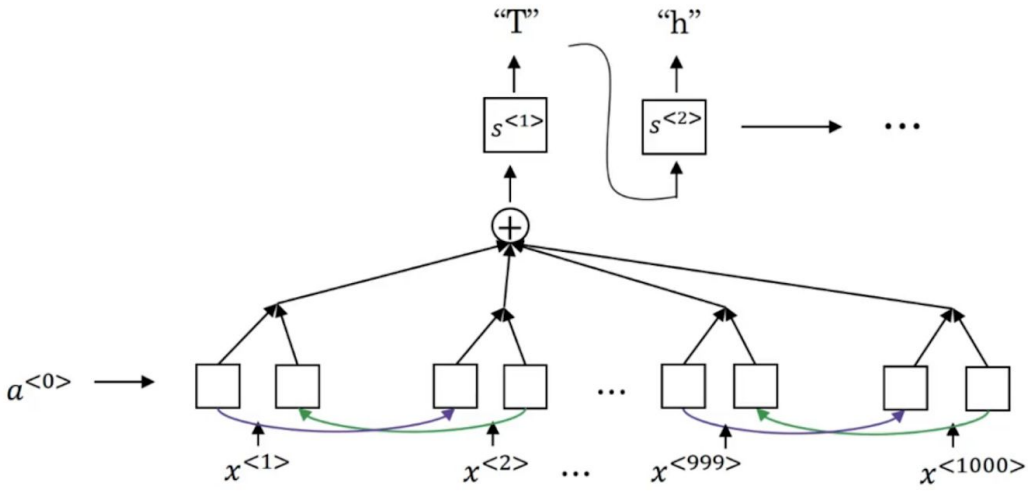
→ phonemes: de kwik braun

300h

3000h

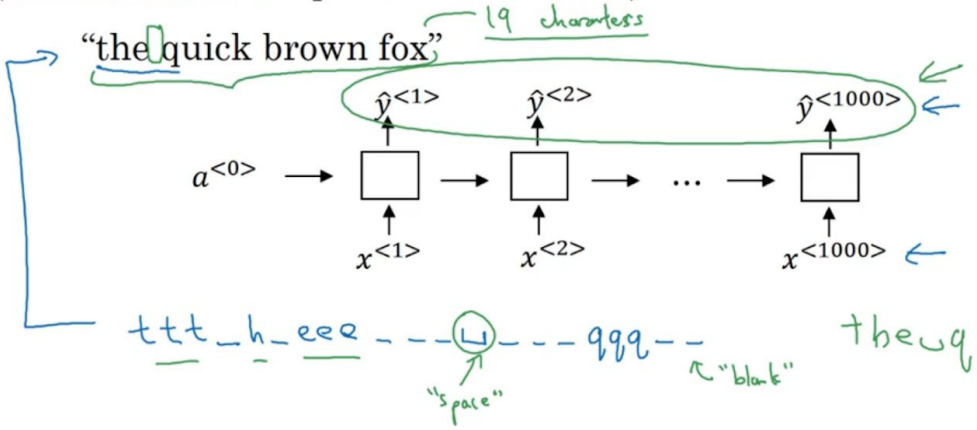
100,000h

Attention model for speech recognition



CTC cost for speech recognition

(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by "blank"

[Graves et al., 2006. Connectionist Temporal Classification: Labeling unsegmented sequence data with recurrent neural networks]

What is trigger word detection?

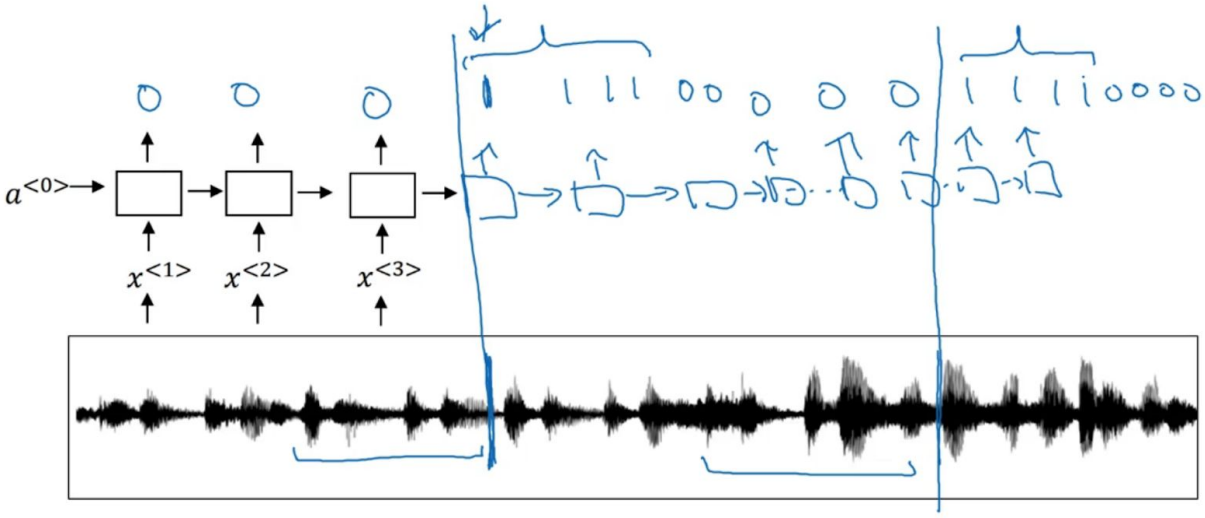
Amazon Echo
(Alexa)

Baidu DuerOS
(xiaodunihao)

Apple Siri
(Hey Siri)

Google Home
(Okay Google)

Trigger word detection algorithm



Andrew Ng