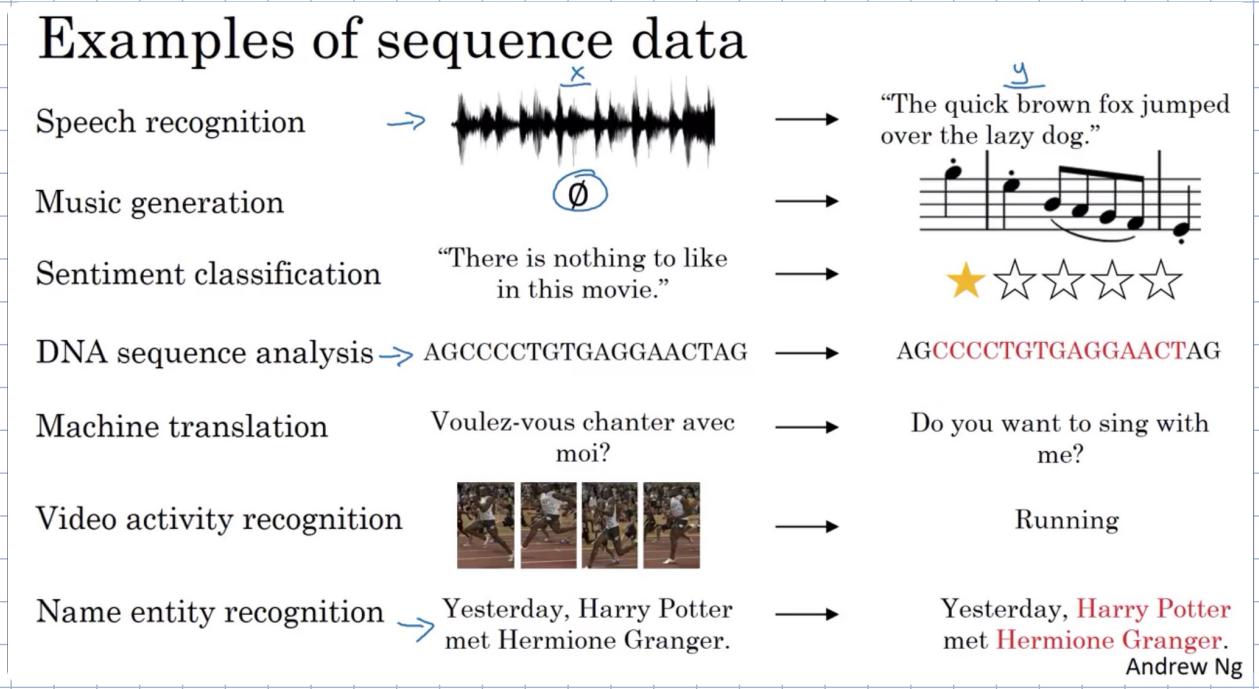
Course 5: Sequence Models

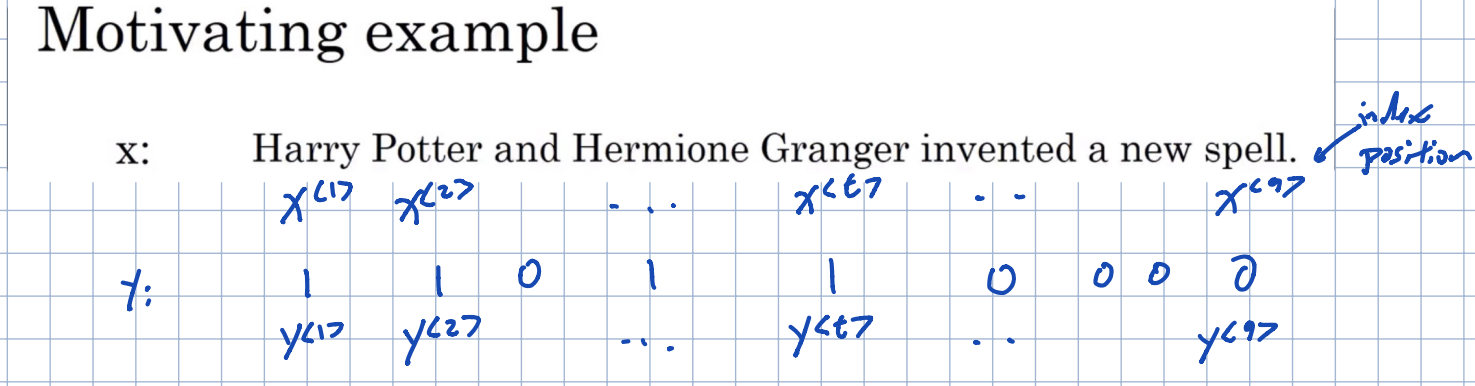
week 1: Recurrent Neural Networks

Why sequence models



Notation

Say we want a sequence model to automatically tell the name in this sentence



T_x = length of input sequence = 9
T_y = length of output sequence = 9
} can be different

$x^{(i)<t>}$: t^{th} element in the sequence of training example i

$T_x^{(i)}$: length of input sequence of training example i

$y^{(i)<t>}$: t^{th} element in output sequence of i^{th} example

$T_y^{(i)}$: length of the output sequence in the i^{th} training example

For NLP, we need to decide how to represent individual words in the sequence.

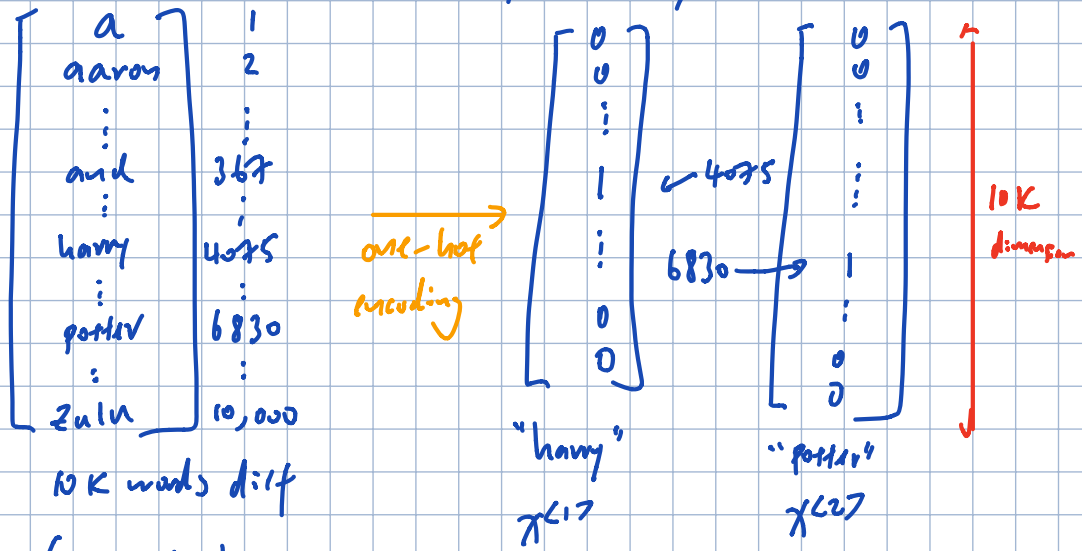
i.e. "Harry" $\Leftrightarrow$ $x^{<1>}$: how to represent

Representing words

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

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

come out with a vocabulary / dictionary, i.e. all the words we will be using



10K words diff

↳ get this by

① choose top 10K words in training set

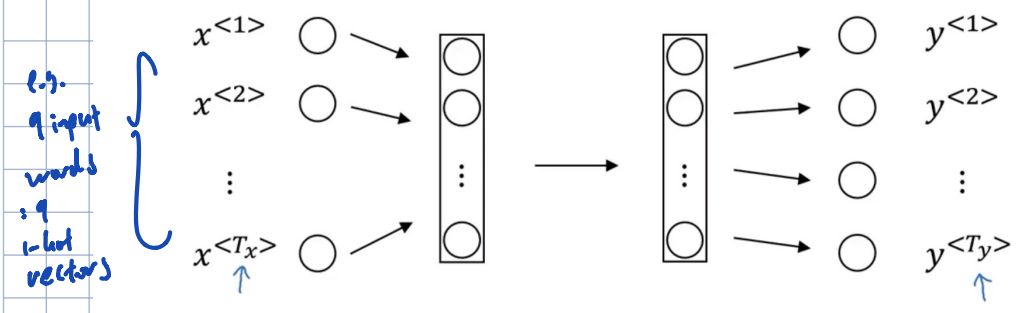
② online

So we have 9 $x^{<t>}$, each a 10K dimension vector

goal: $x \longrightarrow y$

Recurrent Neural Network Model

Why not a standard network?



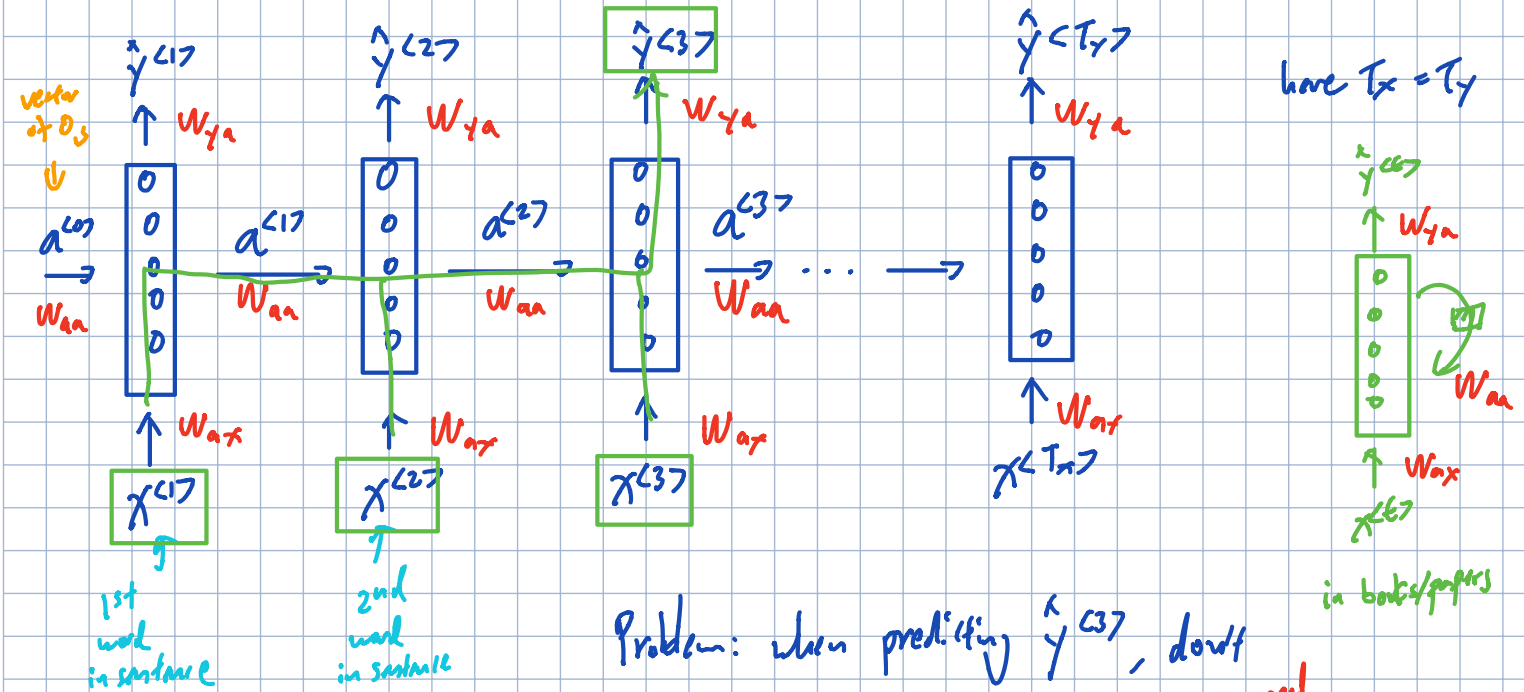
$T_x^{(i)} \neq T_x^{(i+1)}$
$T_y^{(i)} \neq T_y^{(i+1)}$

Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.
  - e.g. "Harry" at position 1: $x^{<1>}$, but can be transferred if "Harry" at position t: $x^{<t>}$.

Want similar effect as CNN, where features learned are generalized quickly to other parts of the image

RNN



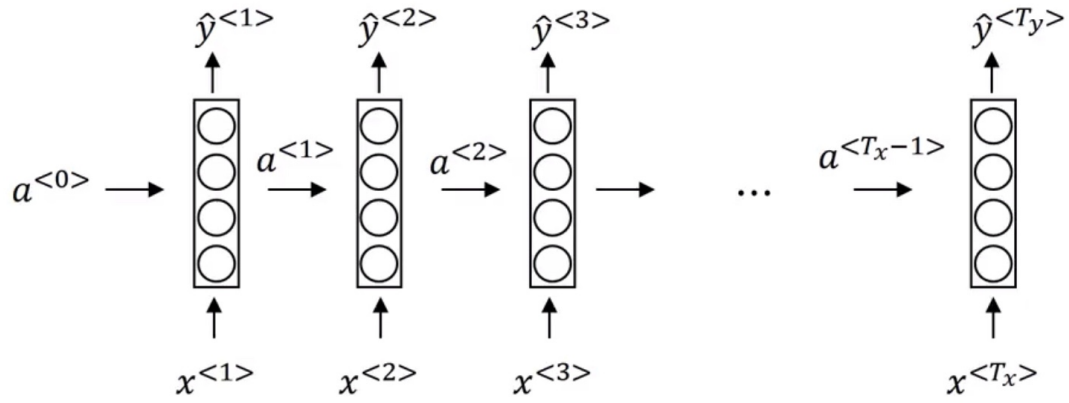
Problem: when predicting $\hat{y}^{<3>}$, don't use $x^{<4>}$, $x^{<5>}$ ...

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

He said, "Teddy bears are on sale!"

→ need Bi-directional RNNs (BRNNs)

Forward Propagation



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

$a^{<1>} = g_1(W_{aa} a^{<0>} + W_{ax} x^{<1>} + b_a)$ ← tanh / ReLu

vanishing gradient
↳ have ways to fix

activation

$y^{<1>} = g_2(W_{ya} a^{<1>} + b_y)$ ← Sigmoid / Softmax
binary multiple

activation (can be diff)

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

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

stack horizontally

$[W_{aa} : W_{ax}] = W_a$ (100, 10100)
100 | 10,000

Simplified RNN notation

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

(100, 100) 100 (100, 10000) 10,000

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

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

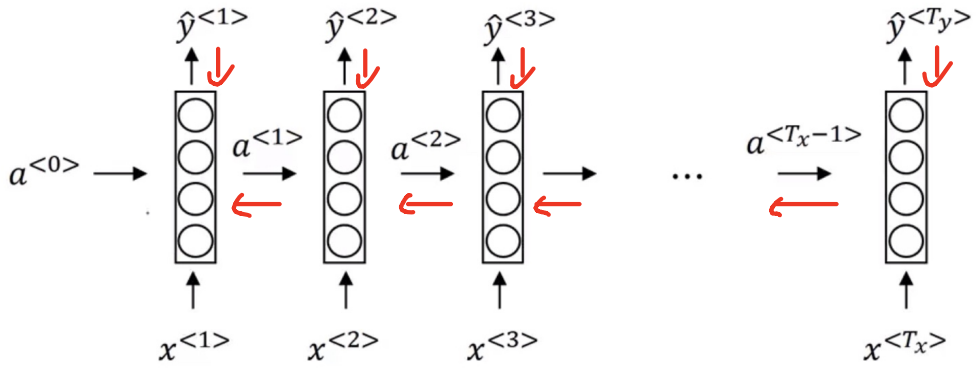
$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$ ↕ 100, 10000

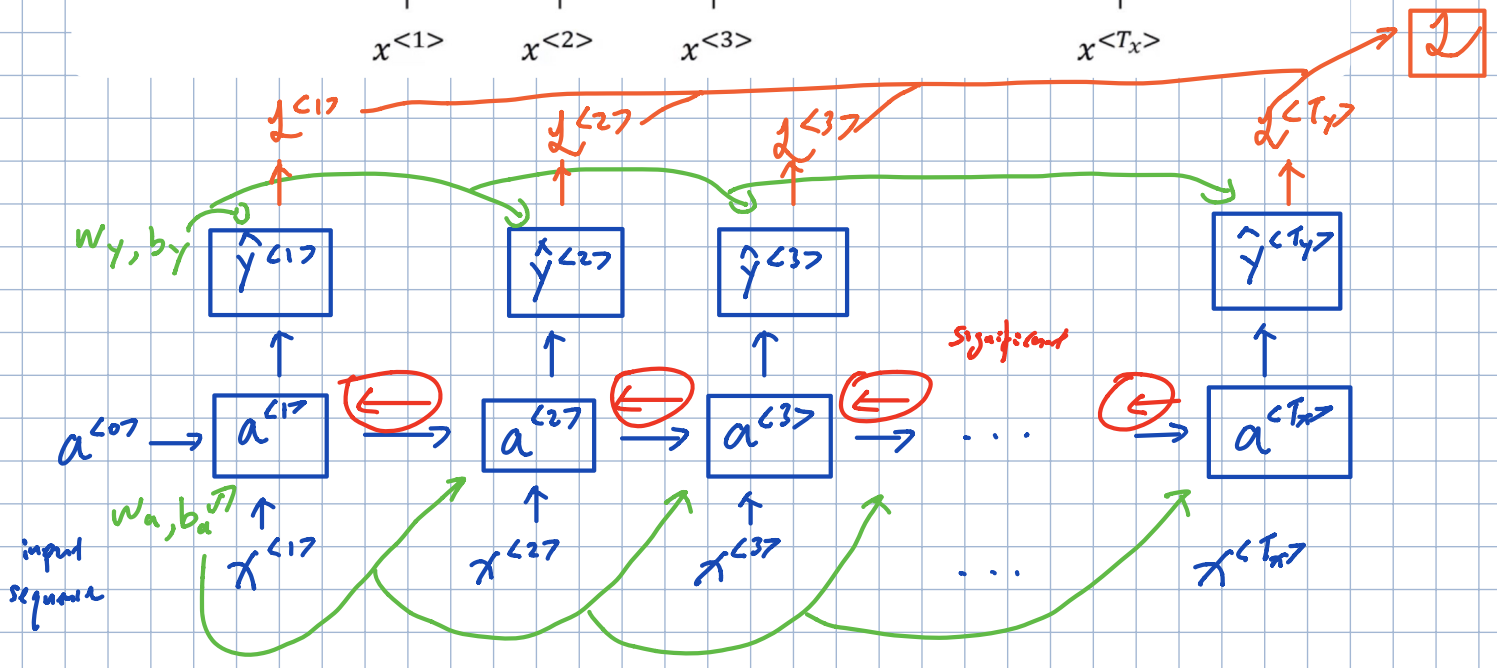
$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$

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

Back propagation through time

Forward propagation and backpropagation





To get back prop, need loss function. We define an element-wise loss first, which is supposed for a certain word in the sequence.

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

$\hat{y}^{<t>}$: 0.1 ; $y^{<t>}$: 0 or 1

standard logistic regression loss / cross-entropy loss

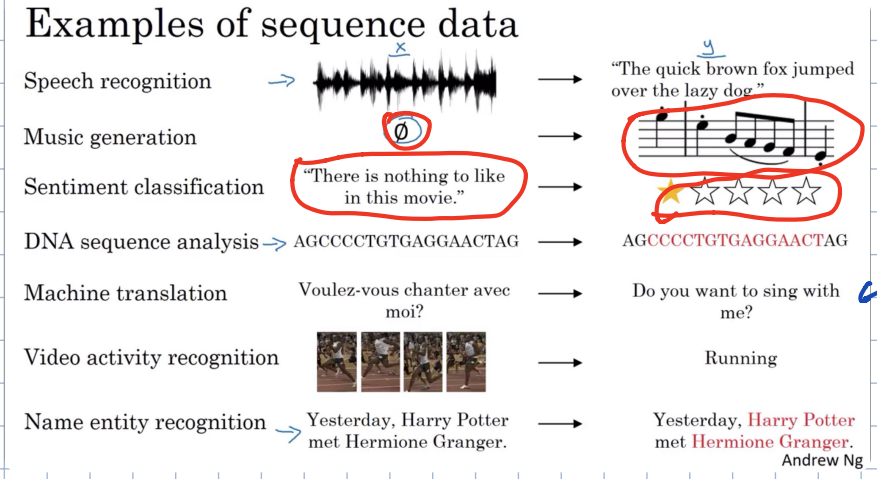
This is the loss associated with a single prediction at a single position or at a single time set.

Overall loss for the entire sequence:

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

Different types of RNNs

So far we have seen RNN architecture where # of inputs (T_x) = # of outputs (T_y). For some applications, T_x may not equal to T_y.

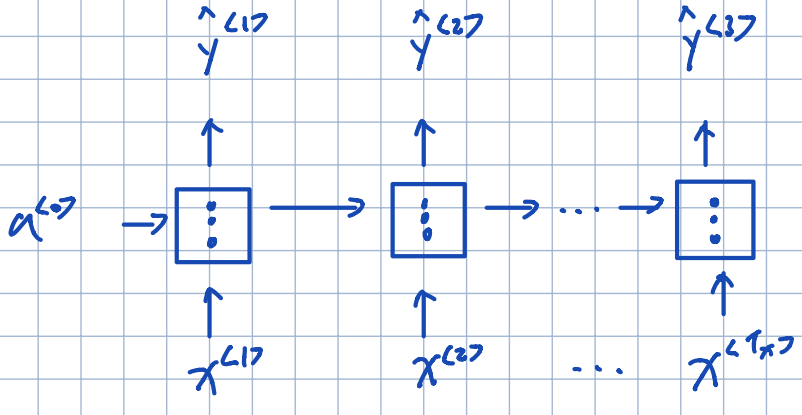


not the same as input!

same type sequence, different lengths

Examples of RNN architectures

$T_x = T_y$



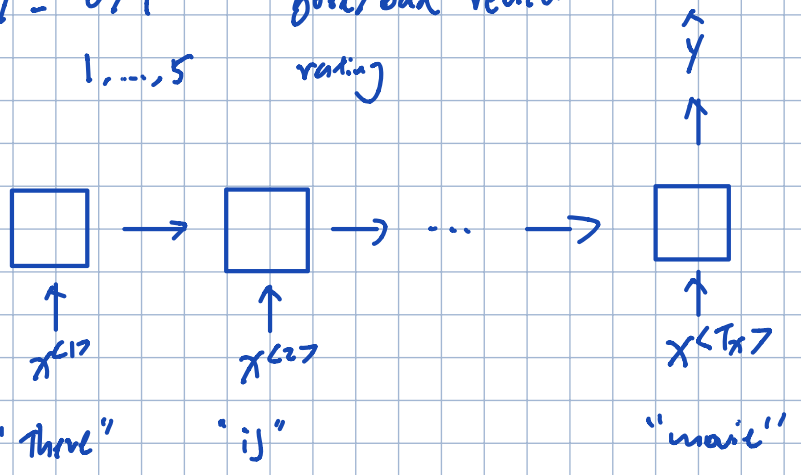
Many-to-Many (same input/output lengths)

Sentiments classification

x = text e.g. movie review

y = 0/1 good/bad review

1, ..., 5 rating

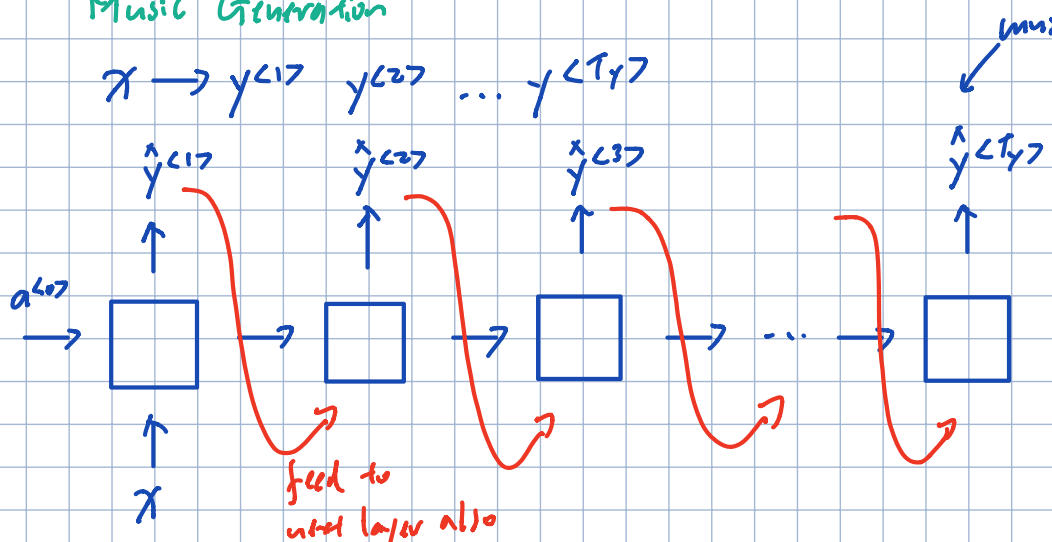


many-to-one

also have one-to-one

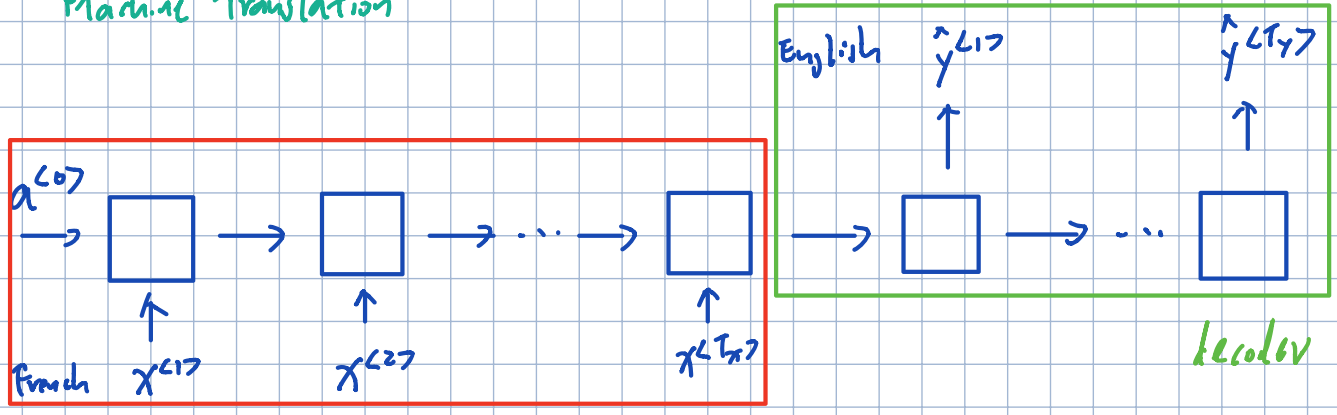
↳ just the usual NN

Music Generation



one-to-many

Machine Translation



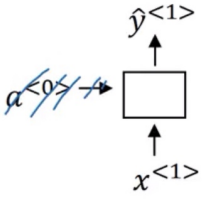
encoder

decoder

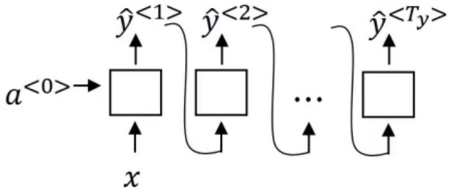
many-to-many

(different input/output lengths)

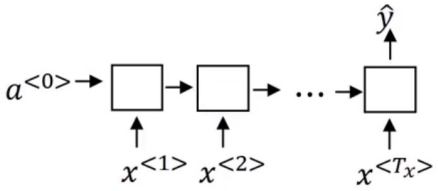
Summary of RNN types



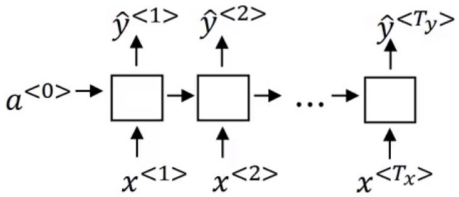
One to one



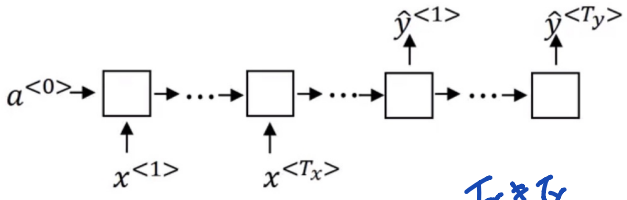
One to many



Many to one



Many to many $T_x = T_y$



Many to many $T_x \neq T_y$

Andrew Ng

Language model and sequence generation

What is language modelling?

Speech recognition

The apple and pair salad. } sound the same

→ The apple and pear salad. }

model output

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

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

Given any sentence, language modelling tells us the probability of sentence

i.e. $P(\text{sentence}) = ?$

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

use y rather than x

output input

Language modelling with an RNN

Training set: large corpus of english text

e.g. a sentence in training set

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

end-of-sentence

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ...

$y^{<9>}$ tokenize

↳ we ignore punctuation

what if words are not in our vocab?

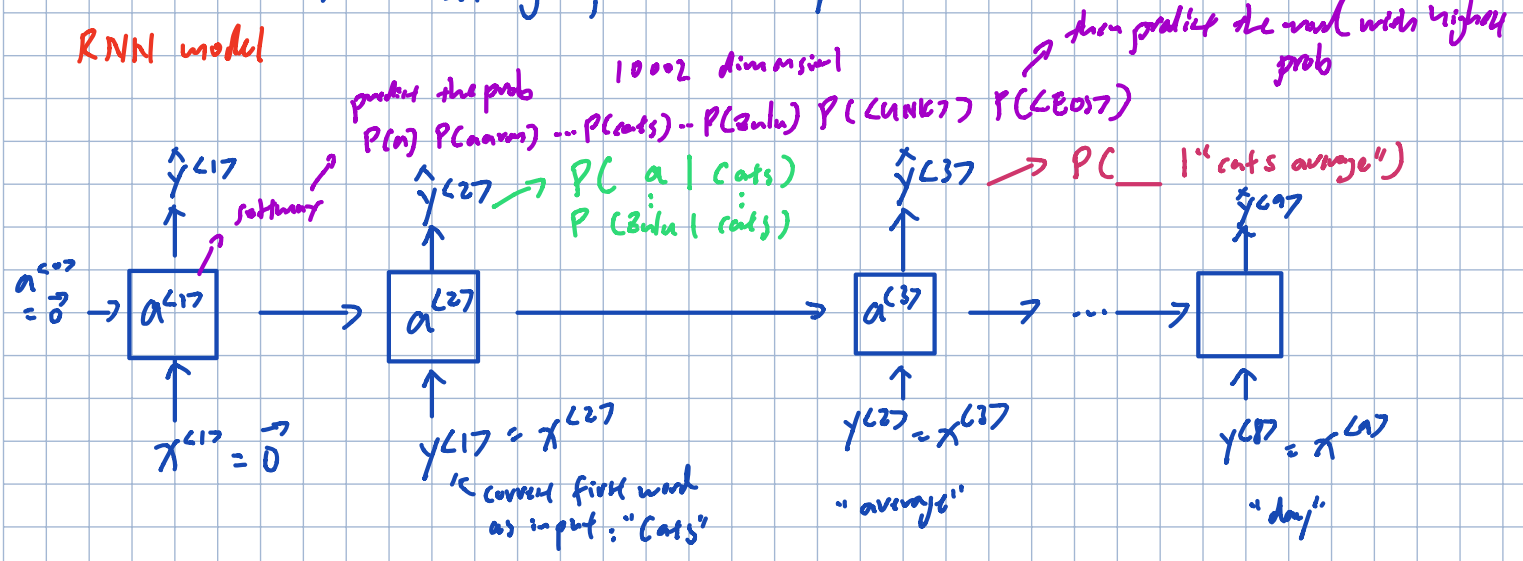
The Egyptian ~~Mau~~ is a breed of cat. <EOS>

<UNK> for unknown words

↳ model the chance of unknown words instead of the specific word

In RNN model, we setting input $x^{<t>} = y^{<t-1>}$

RNN model



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

Each step in the RNN look at the set of preceding words, and ask what is the distribution over the next word?

↳ predict 1 word at a time, left to right.

To train the RNN, need loss function:

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

overall loss:

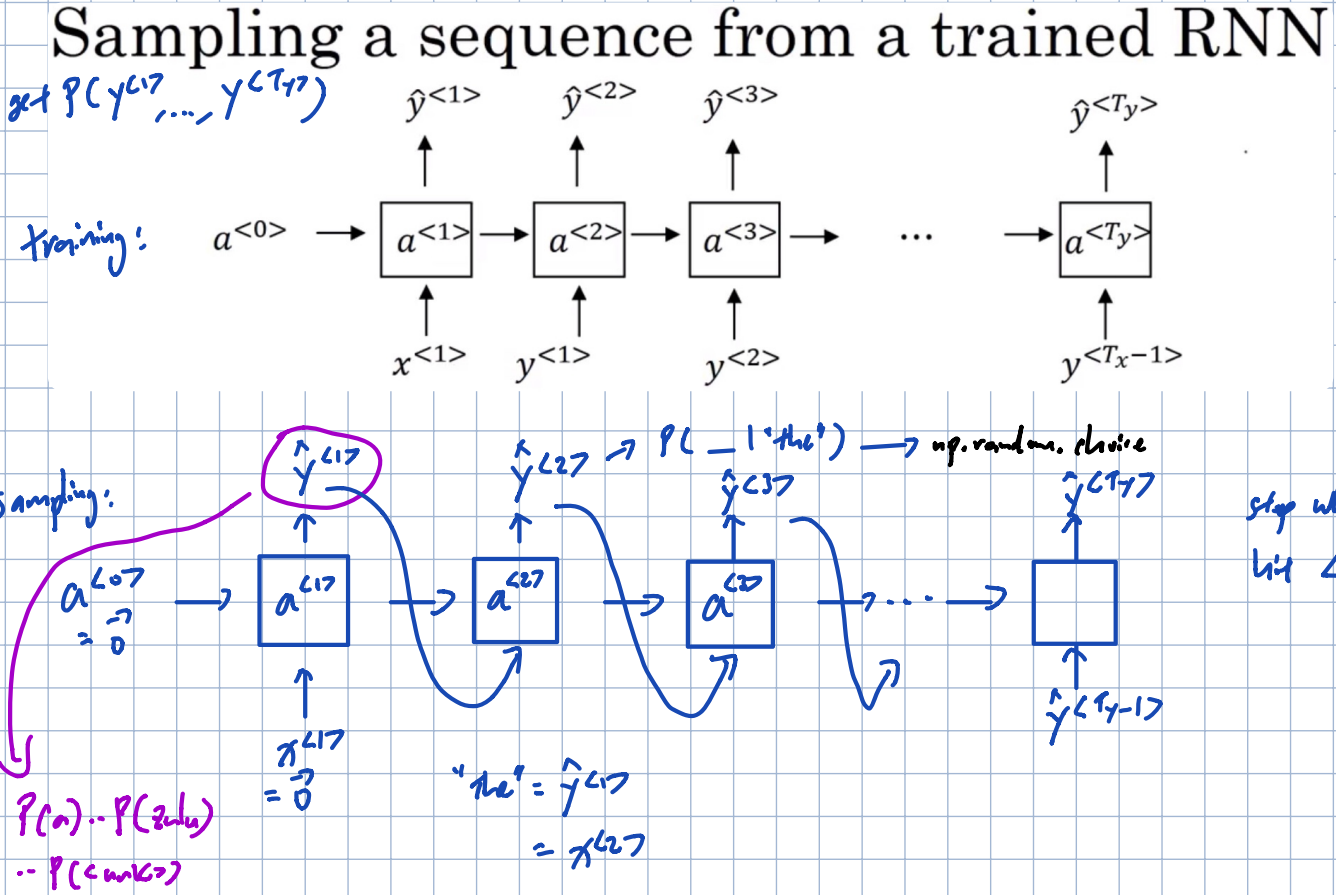
$$\mathcal{L} = \sum_t \mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$

e.g. given a new sentence with 3 words:

can compute $P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, y^{\langle 3 \rangle})$ → probability of the 3-word sentence

$= P(y^{\langle 1 \rangle}) P(y^{\langle 2 \rangle} | y^{\langle 1 \rangle}) P(y^{\langle 3 \rangle} | y^{\langle 1 \rangle} y^{\langle 2 \rangle})$

Sampling novel Sequences

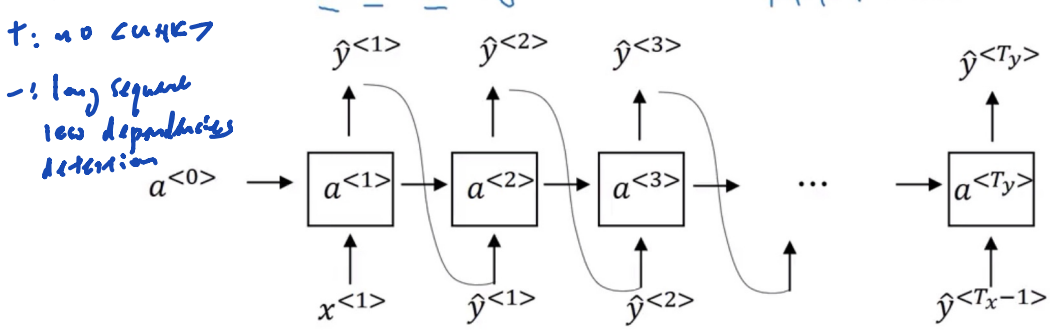


this is how we generate a randomly chosen sentence from a trained RNN language model.

→ this is word-level RNN, also can have character-level RNN

Character-level language model

Vocabulary = [a, aaron, …, zulu, <UNK>]



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.

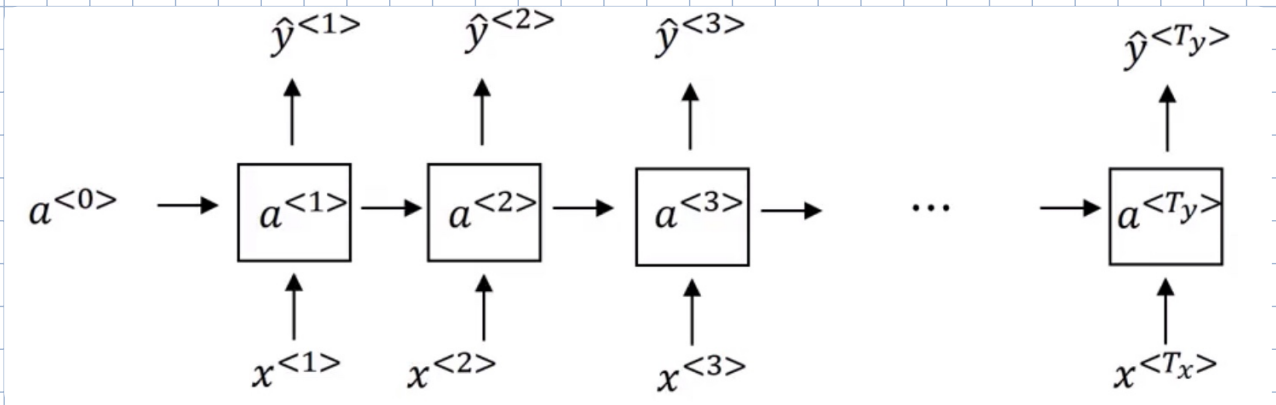
And subject of this thou art another this fold.

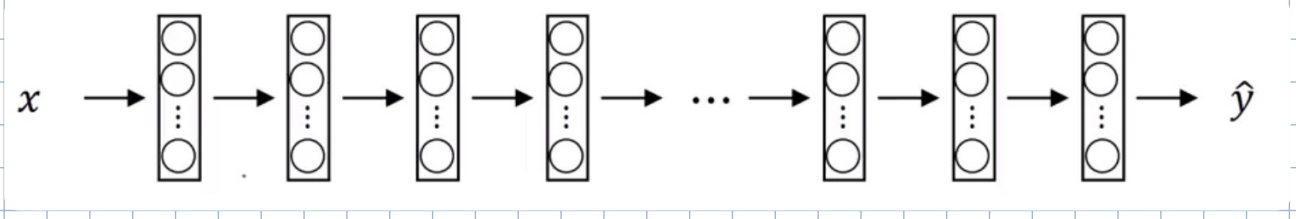
When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

Vanishing gradients with RNNs

e.g. The cat, which ate ..., was full.
The cats, which ate ..., were full. } language has long-term dependency





back prop are hard for deep NN, to affect the weights of earlier layers.
In RNN there are a lot of local influences, i.e. $\hat{y}^{<3>}$ is mainly influence by $x^{<1>}$, $x^{<2>}$, $x^{<3>}$ (ie. values that are close to $\hat{y}^{<3>}$)

Exploding gradients also happens in RNN, but it is less common.
↳ apply gradient clipping

Gated Recurrent Unit (GRU)

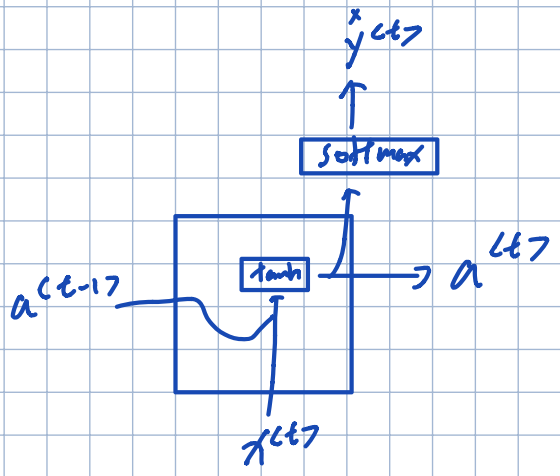
GRU is a modification to the RNN hidden layer, makes it better to capture long range connections and helps with the vanishing gradient.

RNN unit:

activations at time t of RNN

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

↖ tanh



GRU (simplified)

e.g. the cat, which already ate ..., was full.

as we read the sentence, left to right, there is a new variable,

C = memory cell → provides memory if cat is singular / plural

at time t, $C^{\langle t \rangle}$. the GRU will output $a^{\langle t \rangle} = C^{\langle t \rangle}$ (for GRU, they are same, but for LSTM, they are different)

at every time step, we consider $\tilde{C}^{\langle t \rangle}$, i.e. a candidate for replacing $C^{\langle t \rangle}$

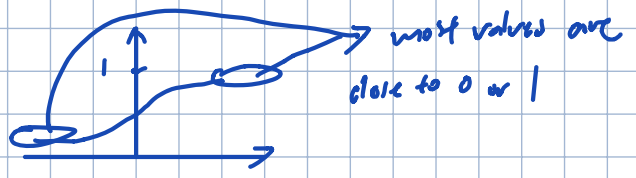
$$\tilde{C}^{\langle t \rangle} = \tanh(W_c [C^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

Important! GRU have a gate: Γ_u, "u" for "update", value between 0 and 1.

we can think of Γ_u always 0 or 1 for intuition (most of the time)

In practice, $\Gamma_u = \sigma(W_u [C^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

↑ sigmoid



update $C^{\langle t \rangle}$ / don't update $C^{\langle t \rangle}$

the $\tilde{C}^{\langle t \rangle}$ is a candidate, Γ_u will decide if we actually update it.

$\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$

$C^{\langle t \rangle} = 1$ - - - - - - - - - - - - $= 1$ ← remembers!

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

e.g. $C^{\langle t \rangle}$ is set 0 or 1, depend on whether the word are singular/plural.

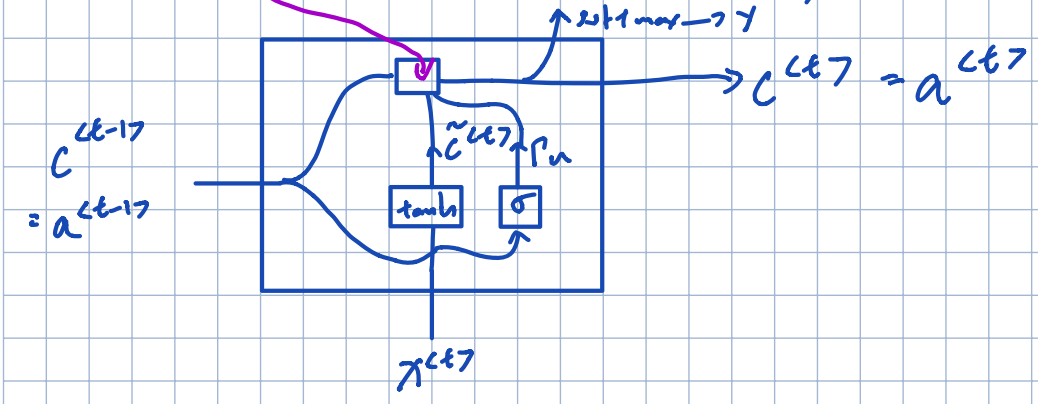
GRU will memorise the values of the $C^{\langle t \rangle}$ all the way until "was"

Γ_u decides when do we update this value

e.g. update @ "cat", and carry till "was". then decide job done and no need to memorise anymore.

↳ $C^{\langle t \rangle} = \Gamma_u * \tilde{C}^{\langle t \rangle} + (1 - \Gamma_u) * C^{\langle t-1 \rangle}$ — element-wise

if =1, set $C^{\langle t \rangle} := \tilde{C}^{\langle t \rangle}$



∴ Γ_u is quite easy to set to 0

∴ $C^{\langle t \rangle}$ is good at maintaining the value for the cell

↳ even across many time steps

↳ helps with vanishing gradient problems.

↳ long range dependencies

$C^{\langle t \rangle}$ and $\tilde{C}^{\langle t \rangle}$ have the same dimension (i.e. a vector of hidden activation units)

Γ_u also have the same dimension

Full GRU

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

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

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

- another gate, "r" stands for "relevance"
  - how relevant is $c^{\langle t-1 \rangle}$ to compute next candidate for $c^{\langle t \rangle}$

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

- relevance symbol in literature

Long Short Term Memory (LSTM) unit

GRU and LSTM

GRU

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

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

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

} 2 gates

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

$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

gives the option of keeping the old values and add new value in the memory cell →

more powerful / general

LSTM

→ use $a^{\langle t-1 \rangle}$ explicitly instead of $c^{\langle t-1 \rangle}$
→ no Γ_r

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

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

new for LSTM

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

- instead of $(1-\Gamma_u)$, we have "forget gate"

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

- output gate

} 3 gates

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

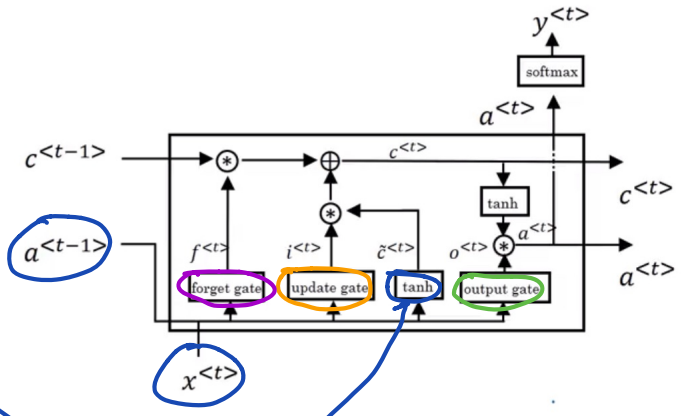
- $*$: element-wise vector × vector

$$a^{\langle t \rangle} = \Gamma_o * \tanh(c^{\langle t \rangle})$$

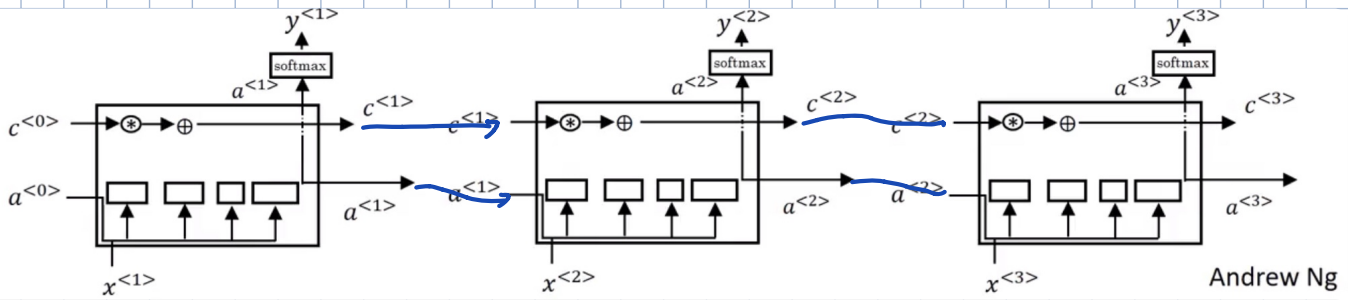
LSTM in pictures

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

add $c^{<t-1>}$ peephole connection



If we have them in parallel



$c^{<0>}$'s values can be passed down easily, for some forget and update gate

e.g. $c^{<3>} = c^{<0>}$

Bidirectional RNN (BRNN)

Getting information from the future

He said, "Teddy bears are on sale!"

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

unidirectional RNN (forward direction only)

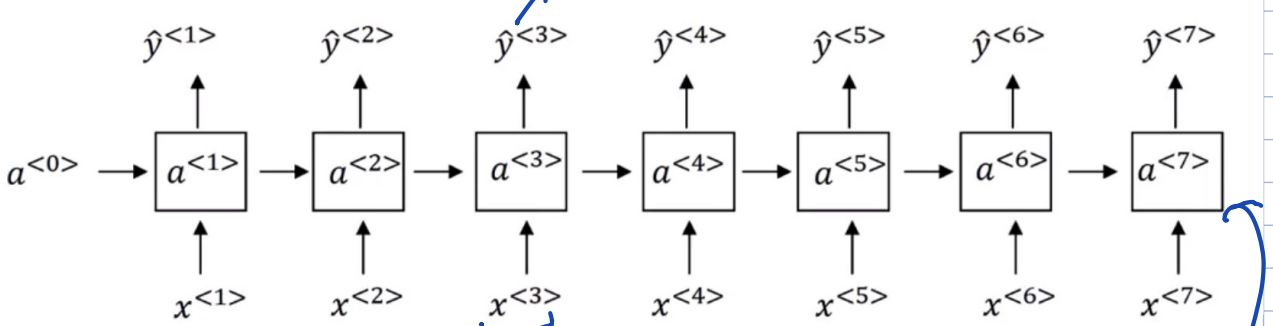
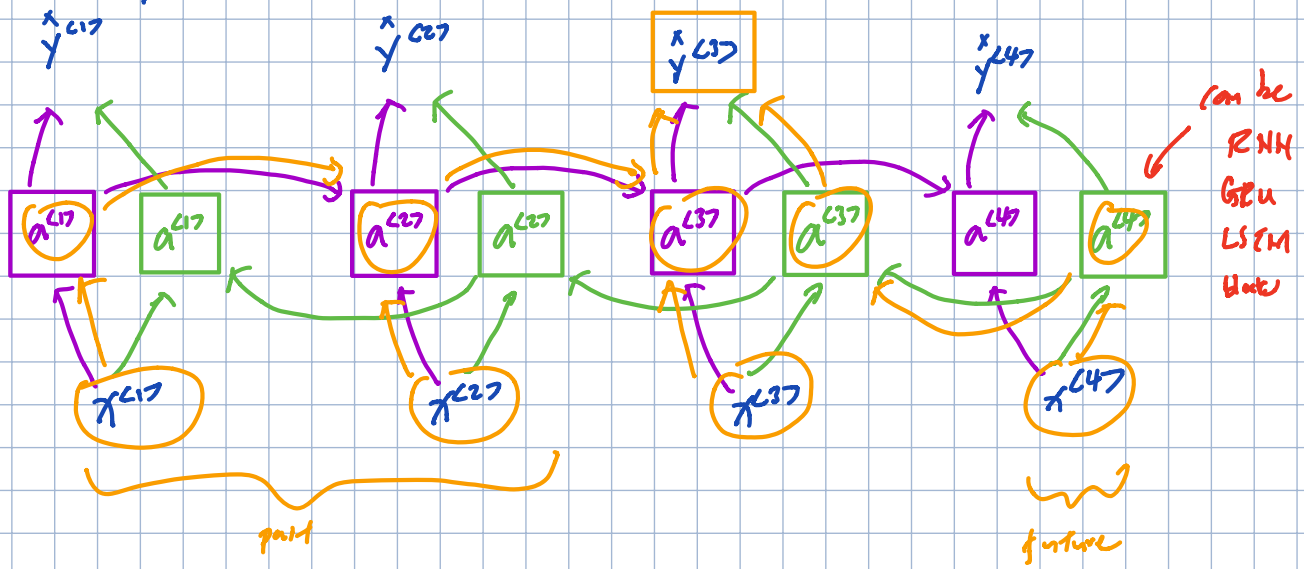


figure out if 3rd word "Teddy" is part of the person's name

regardless RNN GRU LSTM looks all can't tell

BRNN

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



forward recurrent layer: start from $a^{<1>}$

backward recurrent layer: start from $a^{<4>}$

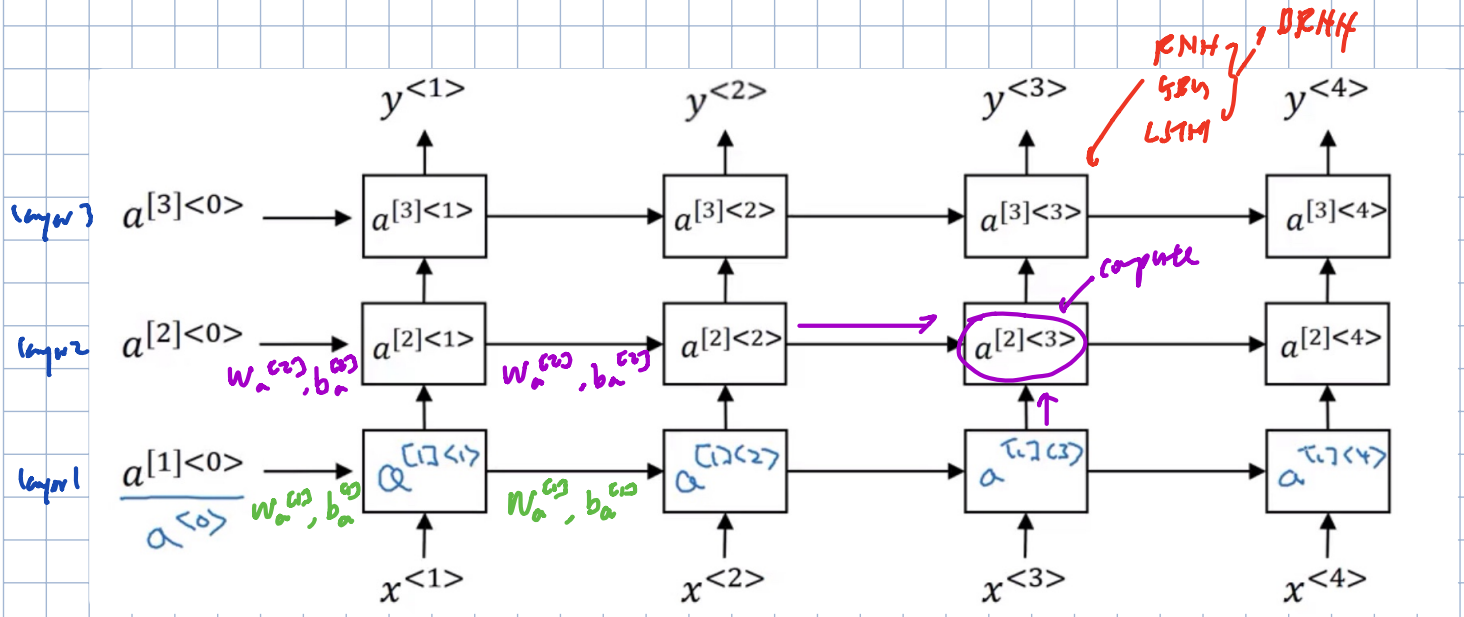
Acydic graph

* both are forward prop, just different direction in time!

* Disadvantage of BRNN: need entire sequence before prediction.

Deep RNN

$a^{[l]<t>}$ (layer, time)



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

Week 2 : Natural Language Processing & Word Embeddings

Word Representation

Word representation

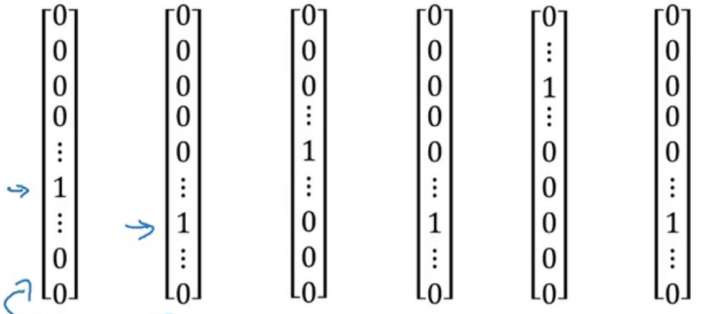
V = [a, aaron, …, zulu, <UNK>] ← vocab

|V| = 10,000

1-hot representation

same closeness as other pair of words →

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



O_{5391} O_{9853}

what's the next word ?

I want a glass of orange juice.

I want a glass of apple _____.

Hard for algorithm to generalize from "orange juice" to "apple juice"

Andrew Ng

"O" stands for one-hot

bag of words treats each word as a thing unto itself → weakness

Inner product of any 1-hot vector = 0

↳ can't distance any pair of these vectors, same distance

↳ algorithm doesn't know "apple" and "orange" are much more similar than "King" or "Queen" - "orange".

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	0.00	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
⋮						
Size						
Cost						
Alive						

300 features

300 dimension vector for "man" → e_{5391}

It would be nice for each word in the dictionary to learn some features (with associated values)

I want a glass of orange _____.

I want a glass of apple _____.

Some of their features are similar

↳ increase the odds the learning algorithm has figure out orange juice is a thing

↳ quick figure out apple juice is also a thing.

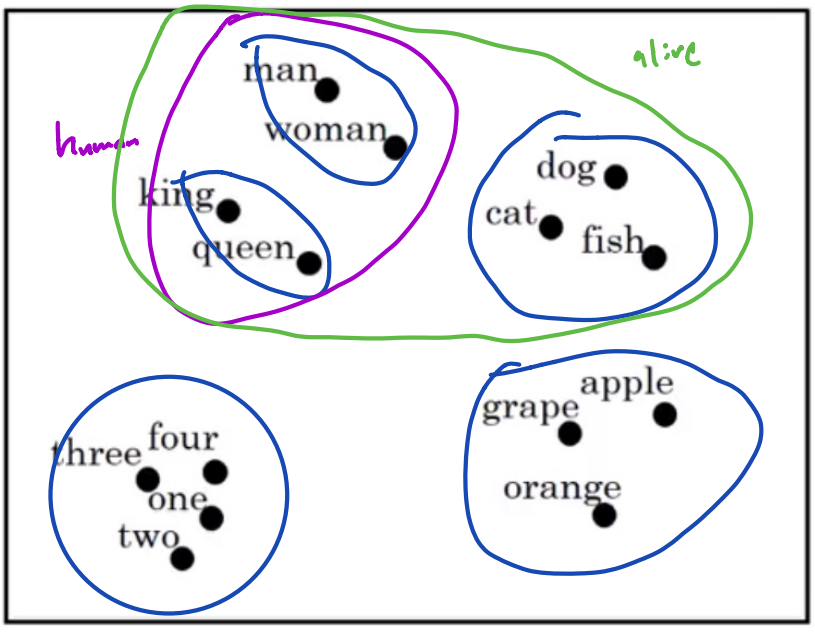
visualising word embeddings

From 300 dimensions (for each word)

↓

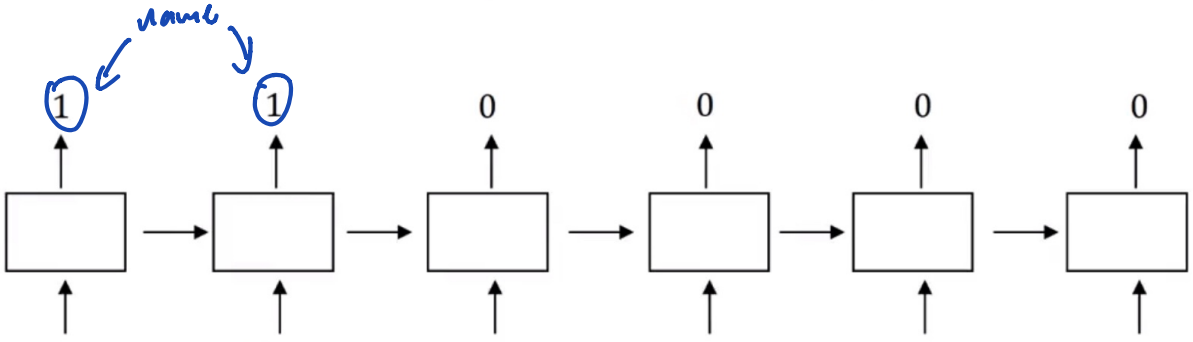
2-Dimensional

t-SNE



Using word embeddings

Named entity recognition example



↳ "orange farmer" and "apple farmer" are similar
↳ helps to identify "Robert Lin" is also a name
↳ how about "durian cultivator"?
↳ need to learn "durian" is a fruit

transfer learning:

1B – 100B words . unlabeled data
↳ figure out "durian" is a fruit
↳ apply 100K training set → for name recognition

Transfer learning and word embeddings

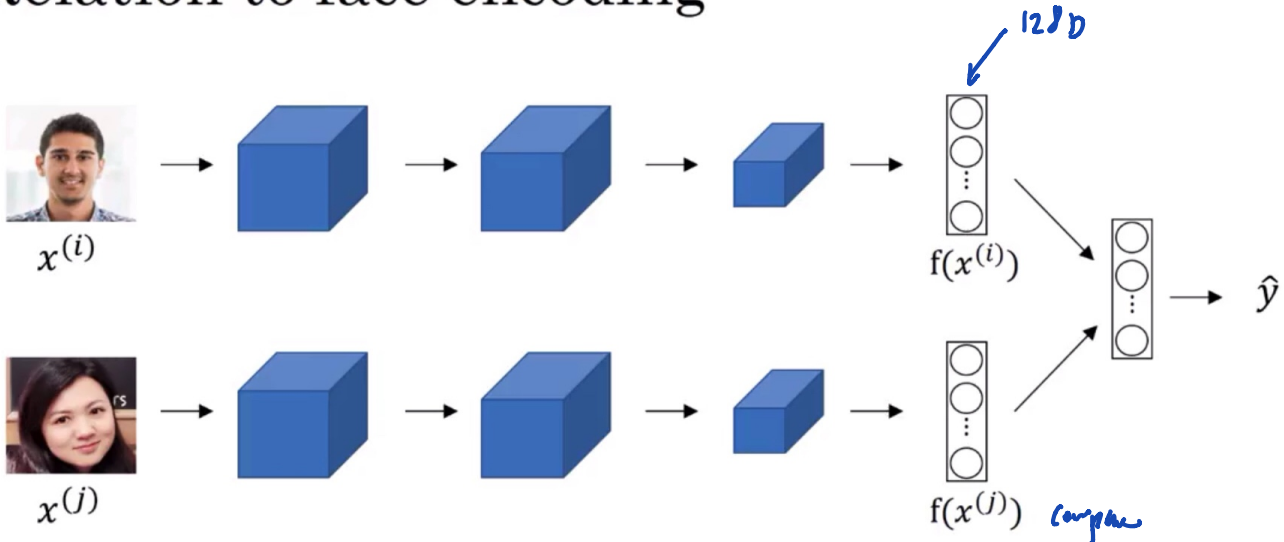
1. Learn word embeddings from large text corpus. (1-100B words)

   (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



compare to see if same face

Properties of word embeddings

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}

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

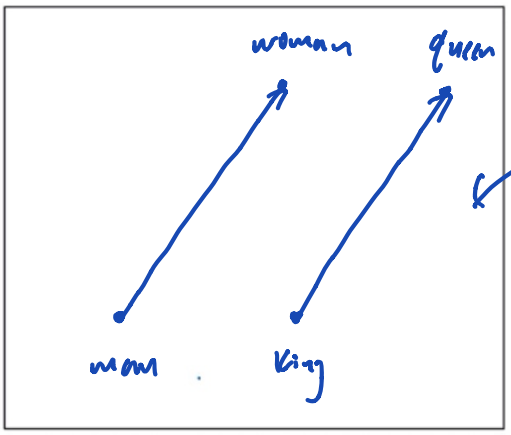
$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

Q:

Man → Woman as King → ?

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

Analogies using word vectors



HOT t-SNE
↳ highly non-linear projection
mostly original feature space

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

300 dimensional

Soln:

Find word w so that: $\underset{w}{\text{argmax}}$ sim $(e_w, e_{king} - e_{man} + e_{woman})$ ← some similarity function

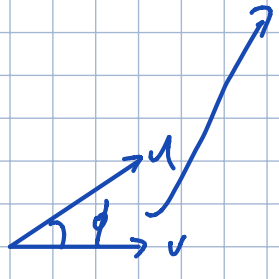
Most commonly used similarity function: Cosine similarity

Cosine Similarity

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

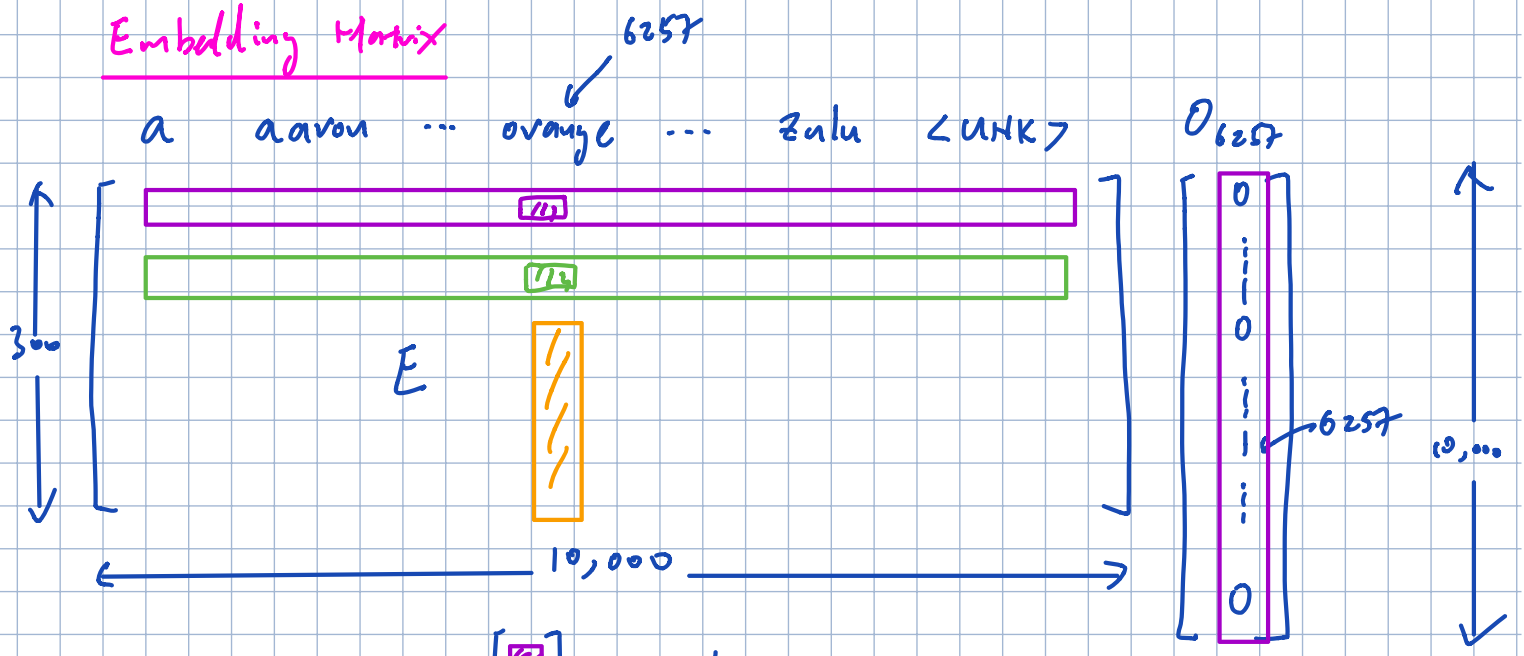
↳ $Sim(u, v) = \dfrac{u^T v}{\|u\|_2 \|v\|_2}$ ← inner product of u and v

$\|u\|_2$, $\|v\|_2$ ← euclidean length



can also use $\|u - v\|$

Embedding Matrix



$$E \cdot O_{6557} = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \leftarrow \text{select out "orange"'s embeddings}$$

$(300 \times 10k)\ (10k \times 1)$ $\quad 300 \times 1$

$$= e_{6257}$$

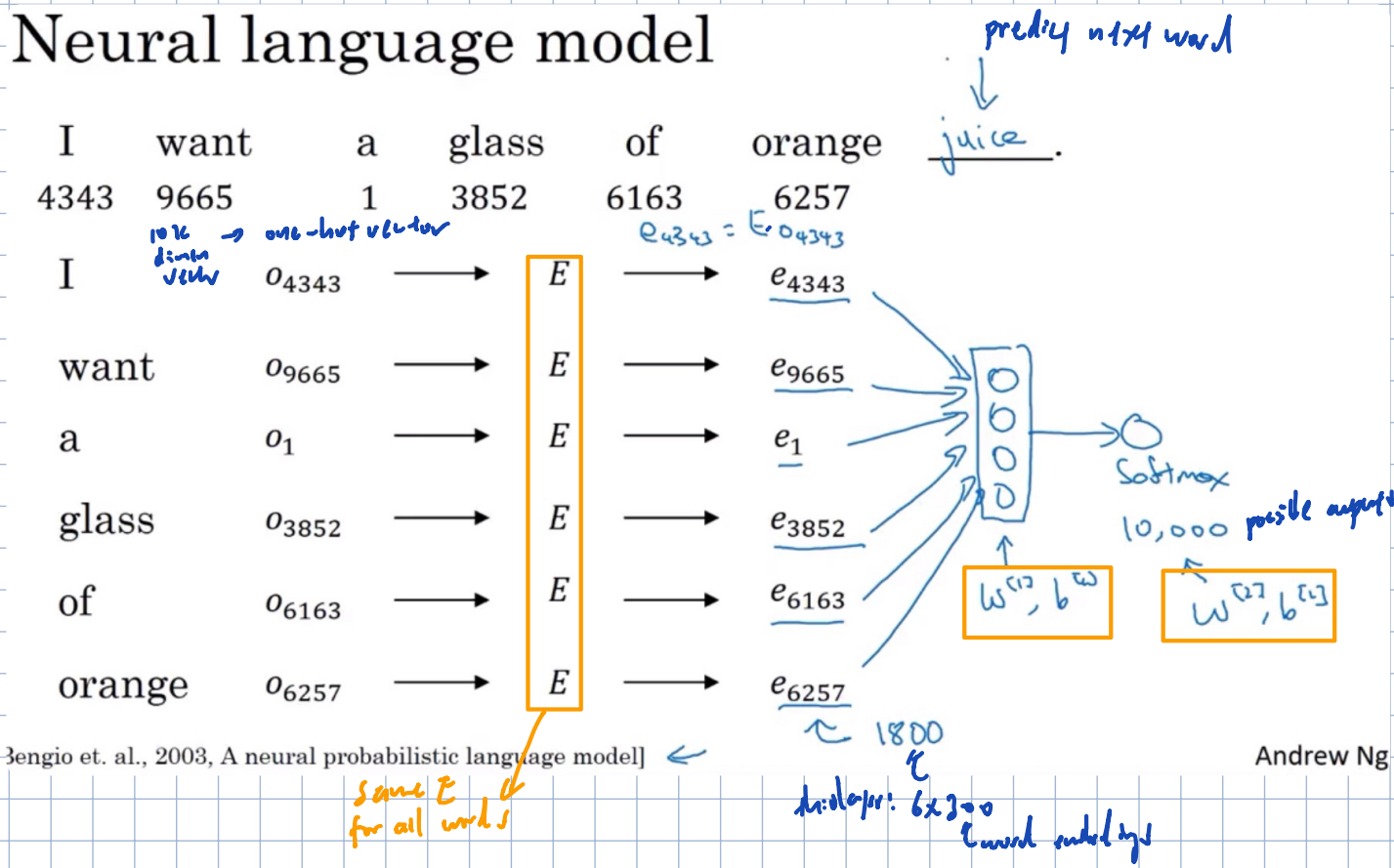
↳ e_w = embeddings for word w

↳ $E \cdot O_j = e_j$

In practice, use specialised function to look up embedding

↳ : most are O_j, not efficient

Learning word embeddings



We commonly use a fixed historical window: e.g. always want to predict the next word given the 4 previous words.

↳ can deal with arbitrary long sentences.

Other context/target pairs

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

context

target

Context: Last 4 words

4 words on left & right

Last 1 word

Nearby 1 word

Word 2 Vec

Skip-grams

come up with a few context to target errors to create our supervised learning problem

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

context	target
orange	juice
orange	glass
orange	my

① randomly pick a word to be context word

② randomly pick another word, within some window, to be target word.

③ Supervised problem: given context, predict what is this randomly chosen words within the window.

Model

Vocab size = 10,000 k

$x \longrightarrow y$

Context c ("orange") 6257 $\longrightarrow$ Target t ("juice") 4834

$O_c \longrightarrow E \longrightarrow e_c = E \cdot O_c \longrightarrow$ O (softmax) $\longrightarrow \hat{y}$

E: parameters

e_c: (embedding vectors for input context)

O (softmax): parameters

θ_t = parameters associated with output t

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

Loss: $J(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$ (one hot)

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

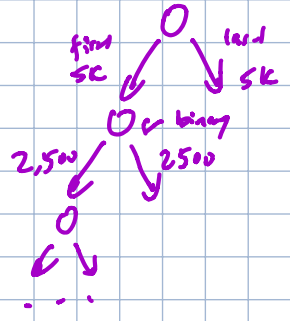
Problems with softmax classification → computational speed

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

every time evaluate this probability, need to sum over 10K.

Solu:

Hierarchical Softmax



classifier.
if target in first 5K or last 5K in the vocab

scale with $\log |V|$

In practice, the tree is not balanced, i.e. common words on top, rare words are deep below.

How to sample context C?

(once context is sampled, target can be sampled within a window)

→ sample uniformly, randomly.

↳ the, of, a ... appear extremely frequently

Negative Sampling

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

context	word	target?	
orange	juice	1	← +ve example
orange	king	0	
orange	book	0	

k

x — k picks randomly from dictionary

y

Supervised problem: given pair of words, is it the target?

problem formulation:
given a pair of words like "orange" and "juice", predict if this a context – target pair?

To generate dataset, pick a context word and then target word → +ve example

↳ for k time:
pick some random target word
pick a random word from dictionary

Model for supervised learning $x \to y$

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

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

define logistic regression model:

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

c e y

orange 6257

$O_{6257} \to E \to e_{6257}$ → 10K binary logistic reg - classifier (0 juice?, 0 king?, ...)

if k=4, then 5 classifier

So instead of having 1 giant 10K-way softmax, (expensive to compute)
we have 10K binary classification problems. (cheap to compute)

[negative sampling: have 1 positive example, then deliberately generate -ve examples.

Selecting negative examples

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

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

① sample according to how often different words appears.
↳ high "the", "and", ...

② $1/|V|$

GloVe word vectors

GloVe (global vectors for word representation)

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

picking c, t

X_{ij} = # time word j (t) occurs in the context of word i (c)

define context and target as whether 2 words appear in close proximity

$X_{ij} = X_{ji}$

X_{ij} is a count that captures how often do words i and j appear with each other

Model

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

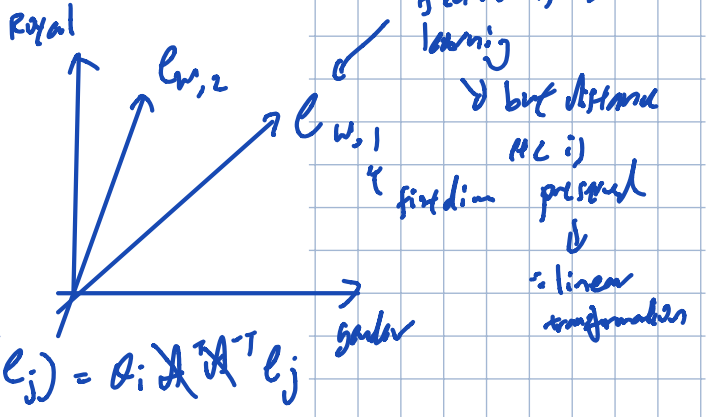
- θ_i: t, e_j: c
- $\theta_t^T e_c$
- weighting
- $f(X_{ij}) = 0$ if $X_{ij} = 0$ → 0? → 0 by convention
- "this" "is" ... → stop words
- "durian" .. → rare words
- } adjusted weights

θ_i, e_j are symmetric

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

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	←



$(A\theta_i)^T(A^{-T}e_j) = \theta_i^T A^T A^{-T} e_j$

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

You cannot guarantee that individual components of the embeddings are interpretable

Sentiment Classification

It is the task of looking at a piece of text and telling if someone likes or dislikes the thing they're talking about.

Sentiment classification problem

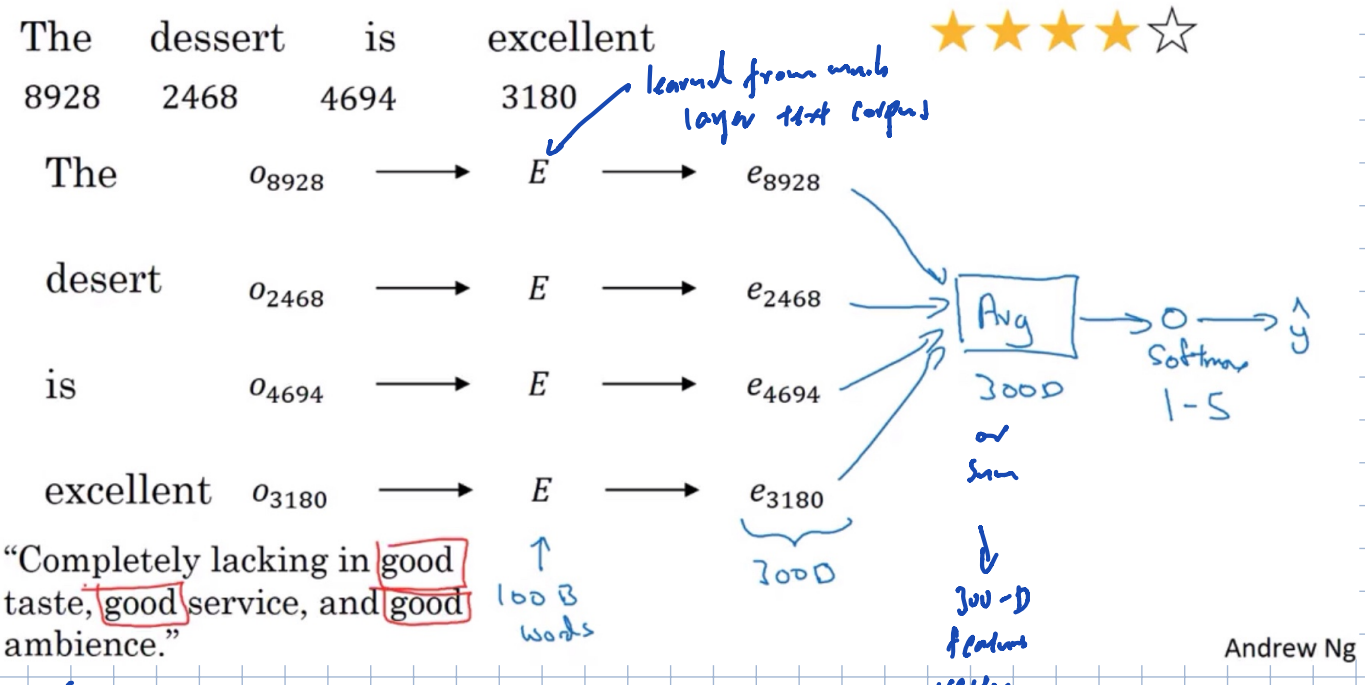
x → y

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.	★☆☆☆☆

training set: 10,000 → 100,000 words

might not have a huge labeled data set

Simple sentiment classification model

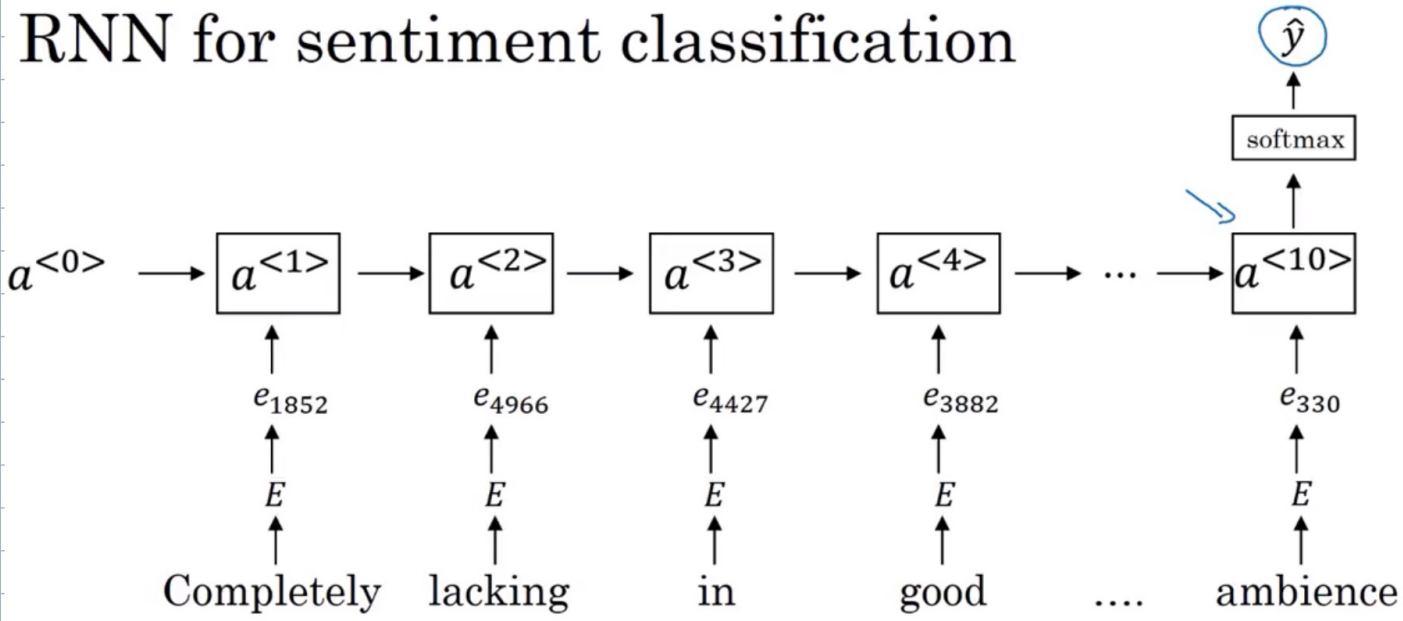


Andrew Ng

Problem: ignore word orders

Use RNN

RNN for sentiment classification



many-to-one

↳ take care of word order

Debiasing word embeddings

The problem of bias in word embeddings

Man:Woman as King:Queen

gender stereotype

Man:Computer_Programmer as Woman:Homemaker ✗

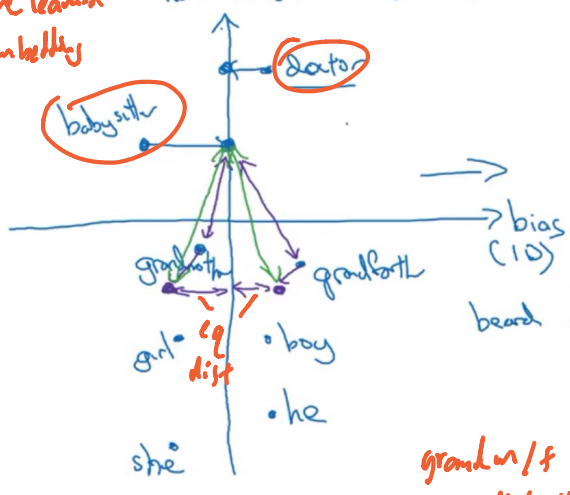
Father:Doctor as Mother:Nurse ✗

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

Addressing bias in word embeddings

say we learned an embedding

non-bias direction (2993)



1. Identify bias direction. (say gender)
   - $e_{he} - e_{she}$
   - $e_{male} - e_{female}$
   - ⋮
   - → average

2. Neutralize: For every word that is not definitional, project to get rid of bias.
   - → Not grandmother etc → inherently F/M
   - e.g. doctor etc

3. Equalize pairs.
   - grandmother/f same distance to babysitter
   - → grandmother − grandfather
     - girl
     - boy

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

Week 3: Sequence model & Attention mechanism

Basic models

Sequence to sequence model

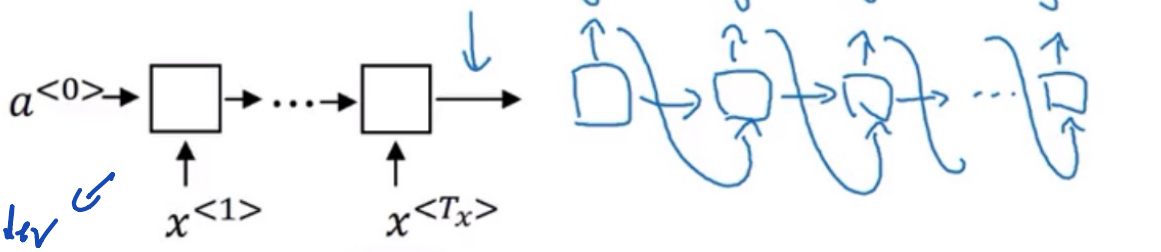
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in September.

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

decoder



encoder

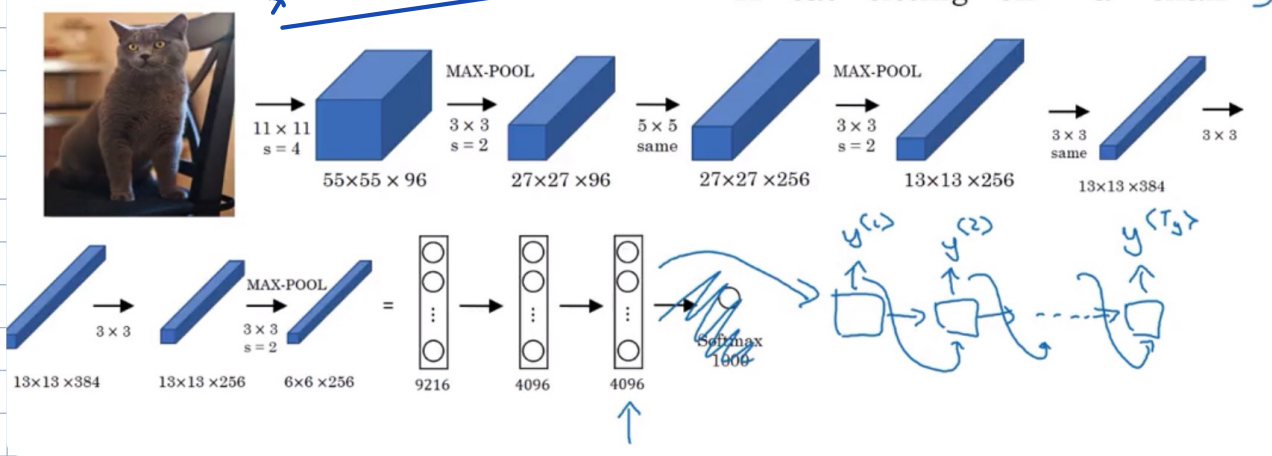
RNN / GRU or LSTM

Image captioning

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

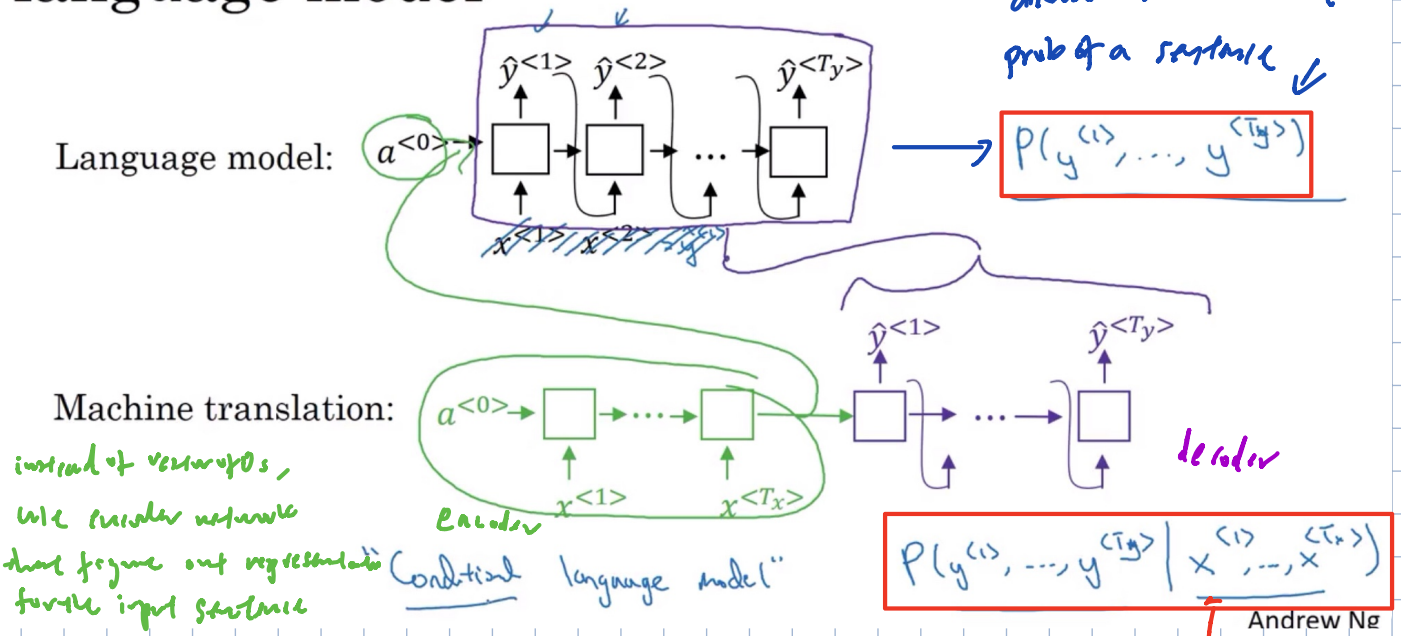
A cat sitting on a chair

x



Picking the most likely sentence

Machine translation as building a conditional language model



allows us to estimate prob of a sentence

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

instead of zeros, we consider network that figure out representation for the input sentence

"Conditional language model"

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

1). French sentence

Machine translation as building a conditional language model.

Finding the most likely translation

Jane visite l'Afrique en septembre. — $P(y^{<1>}, \dots, y^{<T_y>} \mid x)$ (English / French)

we don't want to sample output at random

- 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.

different translation output each time

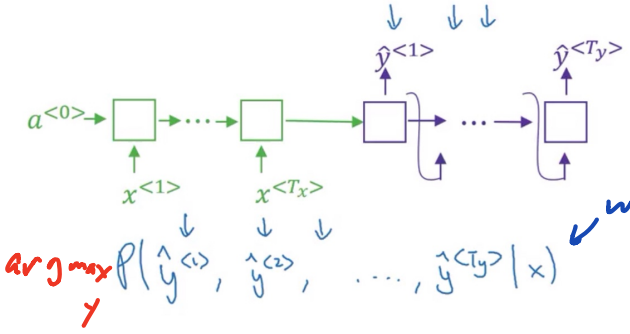
$\arg\max_{y^{<1>}, \dots, y^{<T_y>}} P(y^{<1>}, \dots, y^{<T_y>} \mid x)$

algorithm to maximize conditional prob (i.e. find the right y)
↳ beam search

Why not a greedy search?

$P(\hat{y}^{<1>} \mid x)$

greedy search: pick the most likely word sequentially.



we want this, joint probability maximizing.

$\arg\max_y P(\hat{y}^{<1>}, \hat{y}^{<2>}, \dots, \hat{y}^{<T_y>} \mid x)$

$P(y \mid x)$ is higher

- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.

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

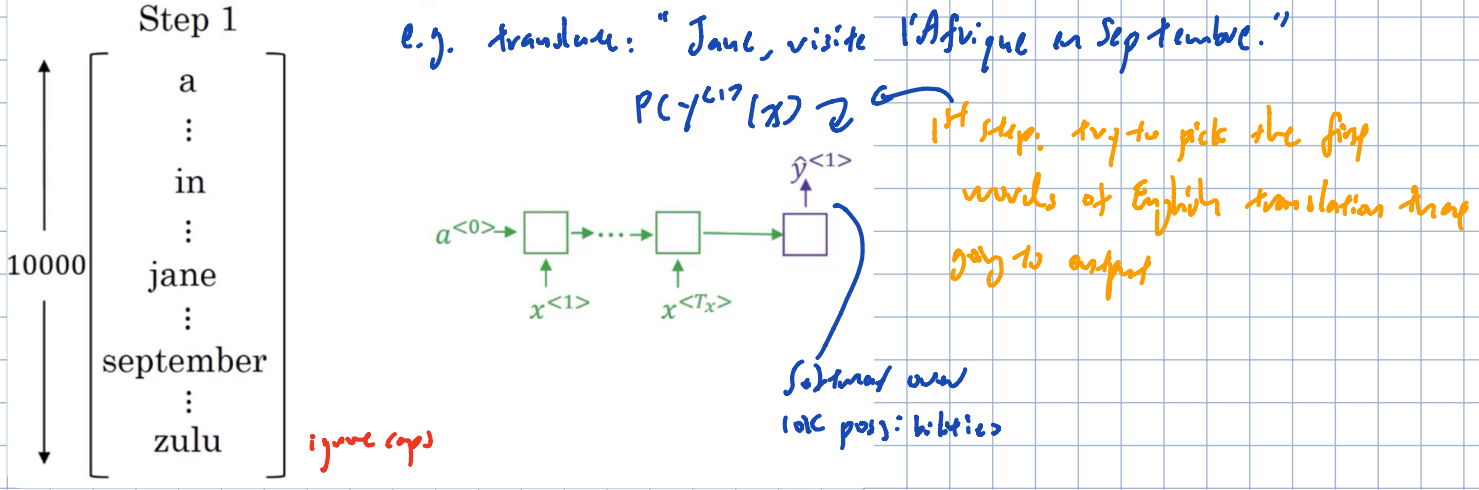
bcz less word sequences

Rather than generate a sentence at random, we want to find the most likely English sentence for machine translation.
But the set of all English sentences of a certain length is too large to exhaustively enumerate ⟶ need an search algorithm.

Beam Search

In machine translation, given a French input, we don't want to output a random English translation, but the best and most likely English translation.

Beam search algorithm



while greedy search will pick only the one most likely words and move on, beam search can consider multiple alternatives.

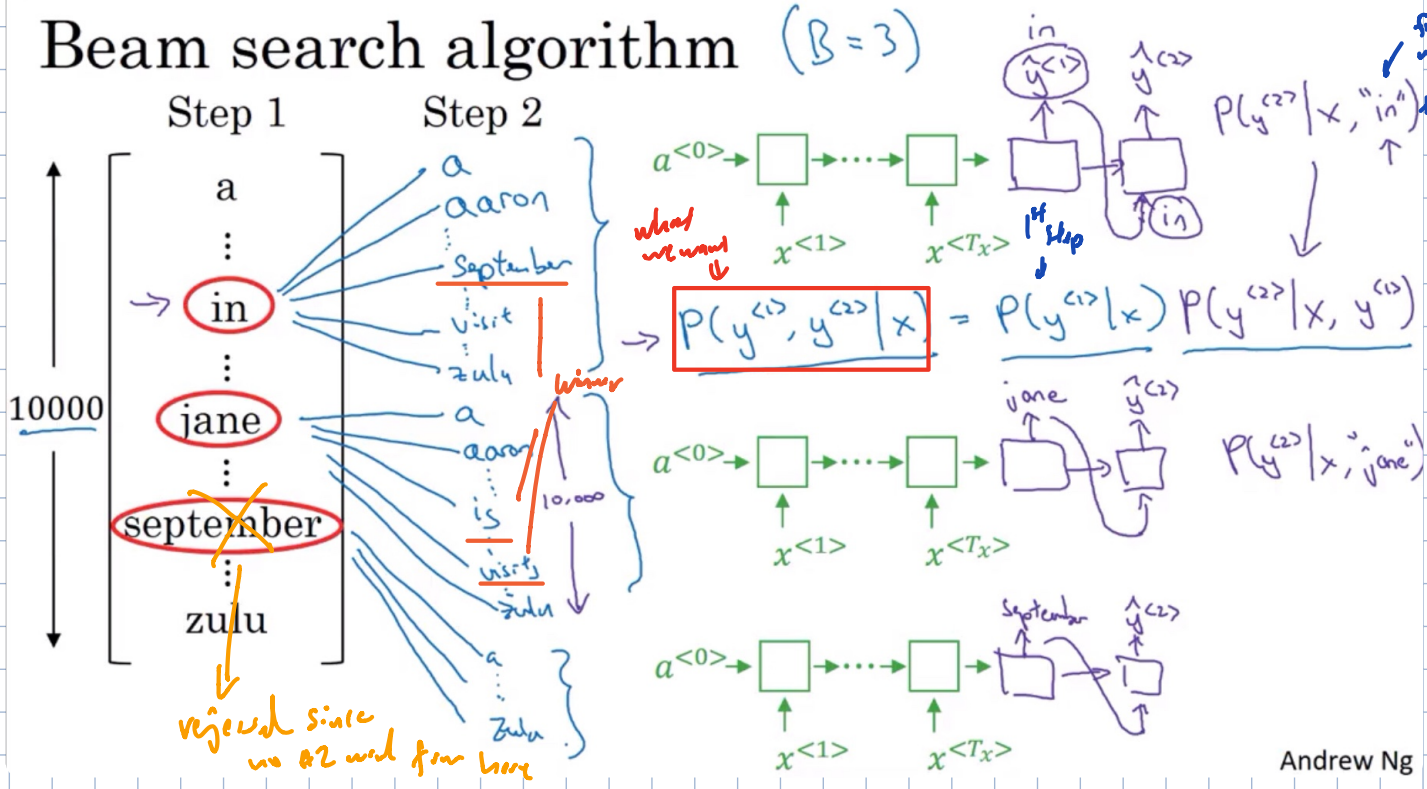
Beam width, $B=3$ (3 possibilities)

e.g. evaluate $P(y^{<1>}|x)$ over different choice of first words, highest

i) "in", "Jane", "September"

then beam search will store in memory that it will try all 3 of these words.

Beam search algorithm (B = 3)



2nd step, for each of these 3 choices, consider what is the 2nd word?

↳ In total, there will be 3 × 10,000 options.

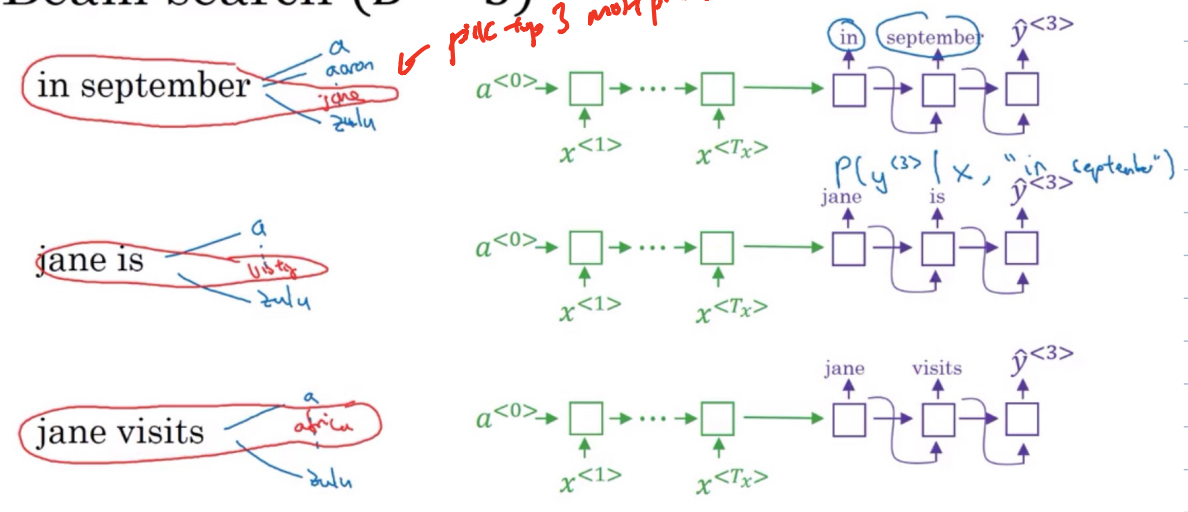
↳ evaluate all these 30K options and pick the top 3.

Since we reject 1 ("September"), we still keep track of 3 pairs $y^{<1>}, y^{<2>}$

(i.e. in September, jane is, jane visits)

↳ ∵ beam width = 3 ∴ every step we instantiate three copies of the network to evaluate these partial sentence fragments and output.

Beam search ($B = 3$)



if B=1 ≡ greedy search

$P(y^{<1>}, y^{<2>} | x)$ jane visits africa in september. <EOS>

Refinements to Beam Search

Length normalisation is a small change to beam search to get much better results.

Length Normalisation

beam search (max this probability)

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

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

all these numbers are very small, multiply them can result in numerical underflow

log

In practice, we take log, (so become sum)

$\max \log P(y|x)$
$\max P(y|x)$ → same y gives max result

② $\arg\max_y \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$

If the sentence is long, then the probability of that sentence is low, since multiply multiple terms together (most of them < 1)
↳ bias towards very short translation
↳ same as the log version

③ $\frac{1}{T_y} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$

↳ normalise by the number of words in the translation

softer approach

$\frac{1}{T_y^{\alpha}}$ $(0 < \alpha < 1)$

↳ complete normalisation by length

beam width = 'a'
↳ always keep track top 'a' sentences
↳ score all of them use this
↳ find maxes

Beam search discussion

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

Beam width B?

1 → 3 → 10, ok 100, large 1000 → 3000 ? research

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

Error analysis on beam search

Beam search is an approximate search algorithm (heuristic search algorithm) ⟶ what if beam search make a mistake?

Example

Jane visite l'Afrique en septembre.

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

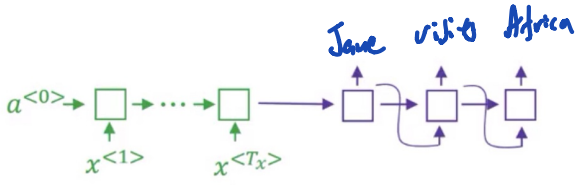
Algorithm: Jane visited Africa last September. ($\hat{y}$)

model has 2 components

① RNN: encoder – decoder

② Beam search

↳ can increase B ↑



RNN computes $P(y^*|x)$ and $P(\hat{y}|x)$, and see which is larger

Error analysis on beam search

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

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

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

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

Bleu Score

Given a French sentence, there can be multiple English translations that are equally good → how to evaluate the machine translation systems?
↳ how to measure accuracy?

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.

Bleu
bilingual evaluation understudy

Any machine translated sentence that is close to reference, gets a high score.
These reference should be provided as part of dev set / test set.

Precision: 7/7

fraction: # of MT words that appear in ref 1/2

Modified precision: 2/7 (count clip "the")

give each word credit up to a max # of time it appears in ref

since "the" appear at most 2 times we give limit =2

We not only want single isolated word appearance, but also pairs of word as well.

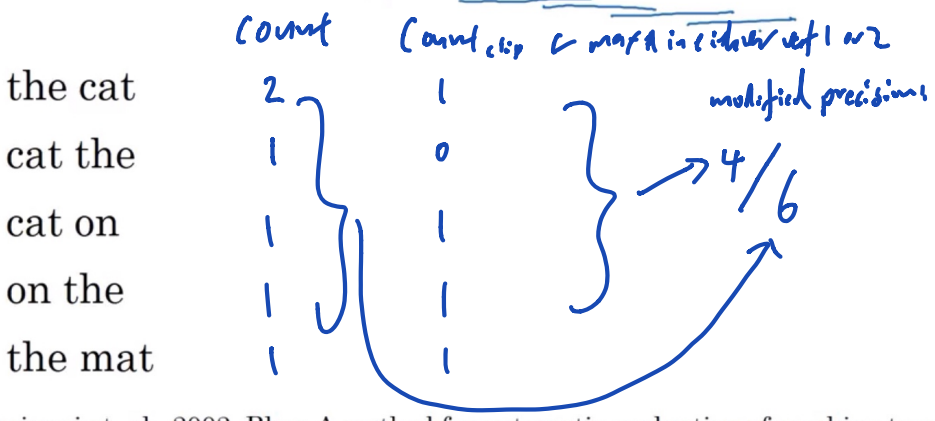
Bigram: pair of words appear next to each other.

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. ←



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

Andrew Ng

Formally:

unigram:

n-gram

$$p_1 = \frac{\sum_{unigram \in \hat{y}} Count_{clip}(unigram)}{\sum_{unigram \in \hat{y}} Count(unigram)} \qquad p_n = \frac{\sum_{ngram \in \hat{y}} Count_{clip}(n\text{-}gram)}{\sum_{ngram \in \hat{y}} Count(n\text{-}gram)}$$

p_1 = modified precision for unigram

If MT output is exactly same as ref 1 or 2, $p_1, p_2 \ldots = 1.0$

Final Bleu score:

p_n = Bleu score on n-grams only ⇒ get $p_1, p_2 \ldots p_4$

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

BP = brevity penalty

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

↓

∴ Favors translation gets easier Bleu score

Bleu usually for MT ∴ multiple good translations

not in speech recognition ∴ usually have ground truth to compare with.

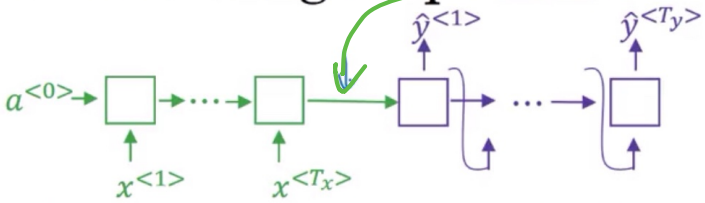
Attention model intuition

Encoder – decoder architecture:

1 RNN reads in sentence, then 1 RNN outputs a sentence

A modification to this is called attention model

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.

what we ask the (green) encoder in our network do is read in the whole sentence and then memorize the whole sentence and store it in the activations here

the decoding network then generate the English translation

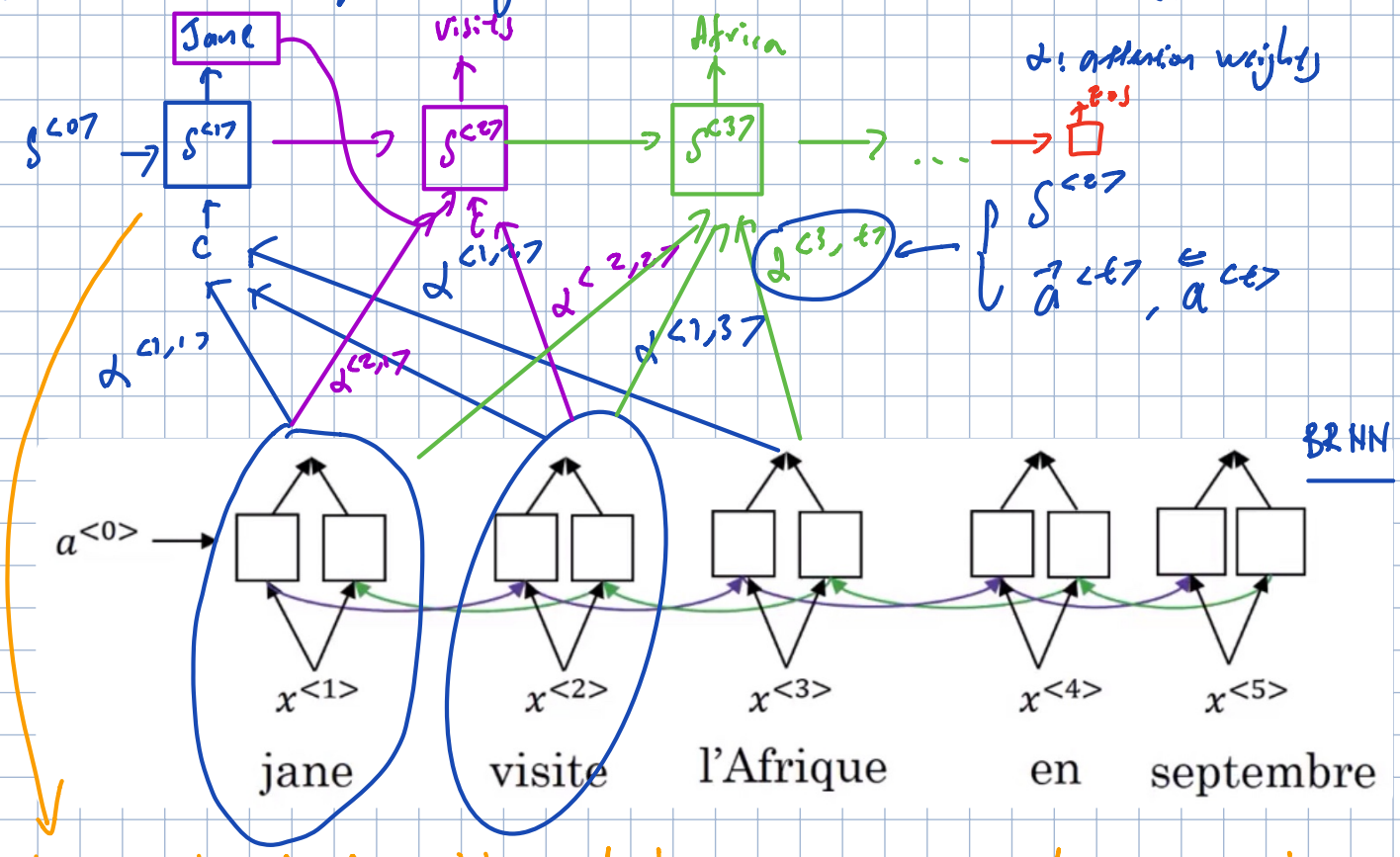
In human translation, we usually do it part by part through the sentence.

Attention model intuition

use another RNN to generate english translations

"s" another notation for hidden unit here.

α: attention weights



when translate the first english word, how much attention should we pay to the first word in the French sentence.

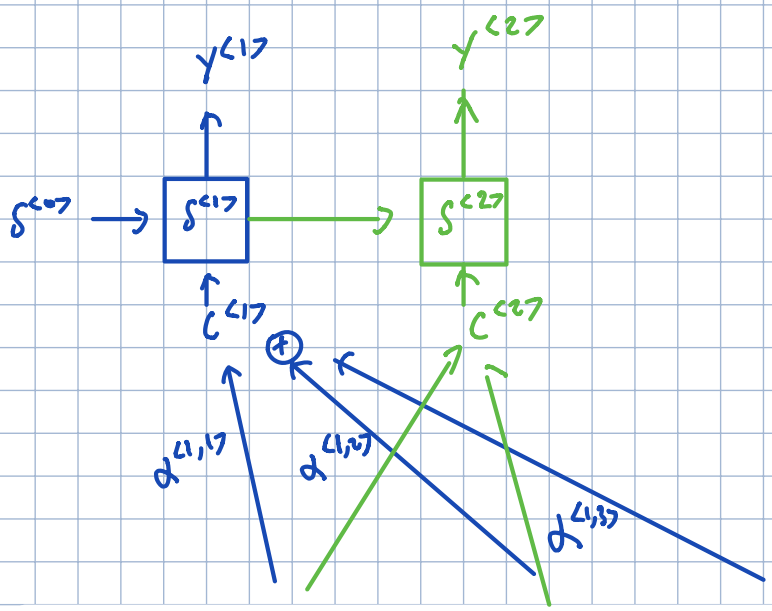
The RNN (above) at every step, there are their attention weights $\alpha^{<t,t'>}$, when we are trying to generate t English word, how much should we pay attention to t' French word.

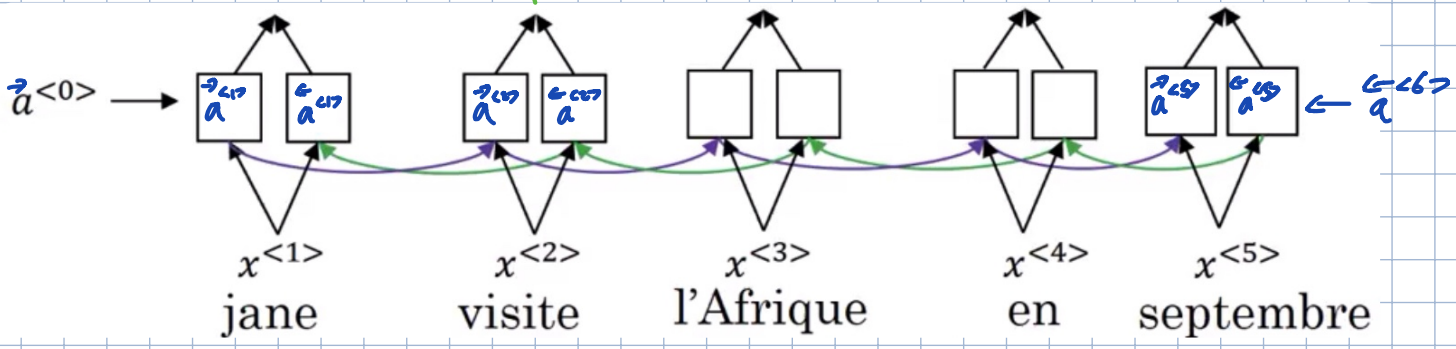
Attention model

$$\sum_{t'} \alpha^{<1,t'>} = 1$$

$$c^{<1>} = \sum_{t'} \alpha^{<1,t'>} a^{<t'>}$$

$$c^{<2>} = \sum_{t'} \alpha^{<2,t'>} a^{<t'>}$$





$a^{<t'>} = (\vec{a}^{<t'>}, \overleftarrow{a}^{<t'>})$

↳ both of the activations concatenate together

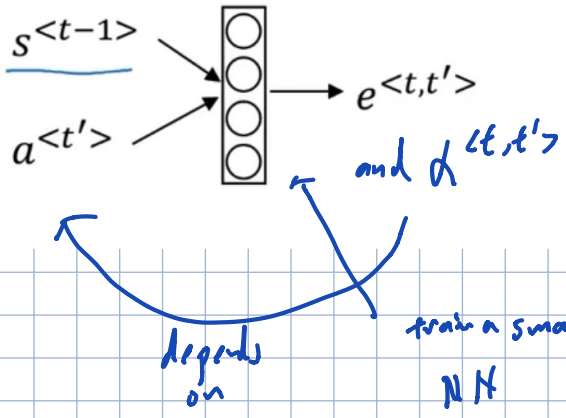
↳ feature vector for time step t

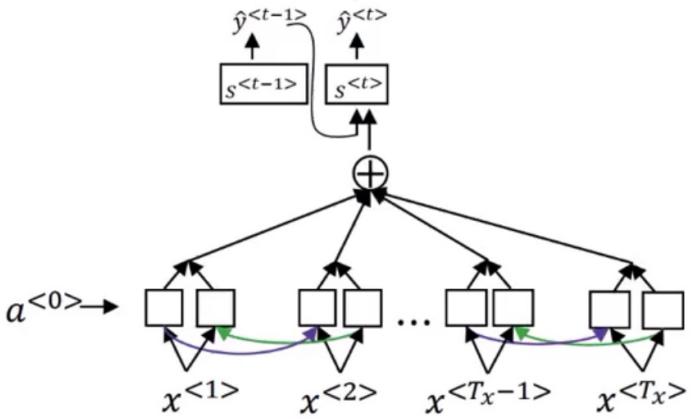
$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

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

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

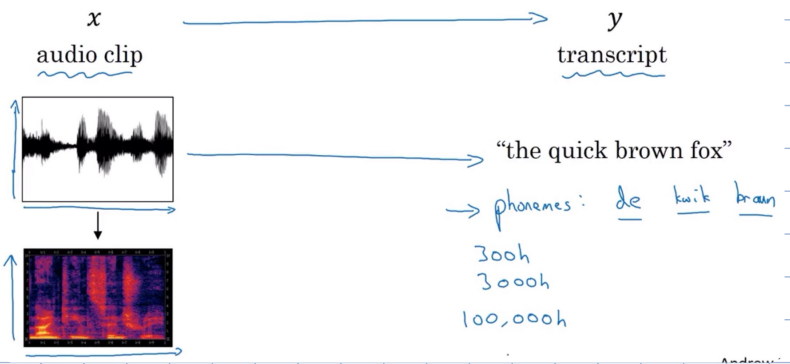
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$



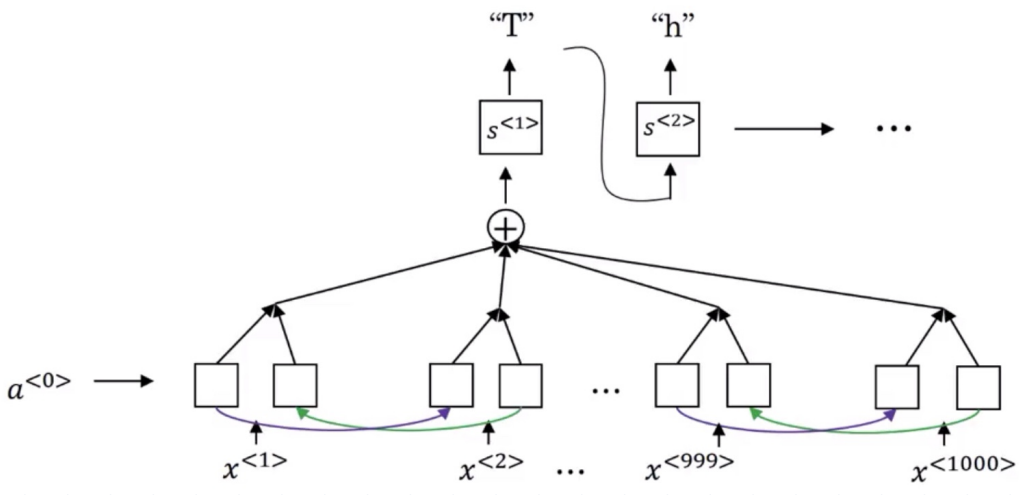


Audio data: Speech Recognition

Speech recognition problem



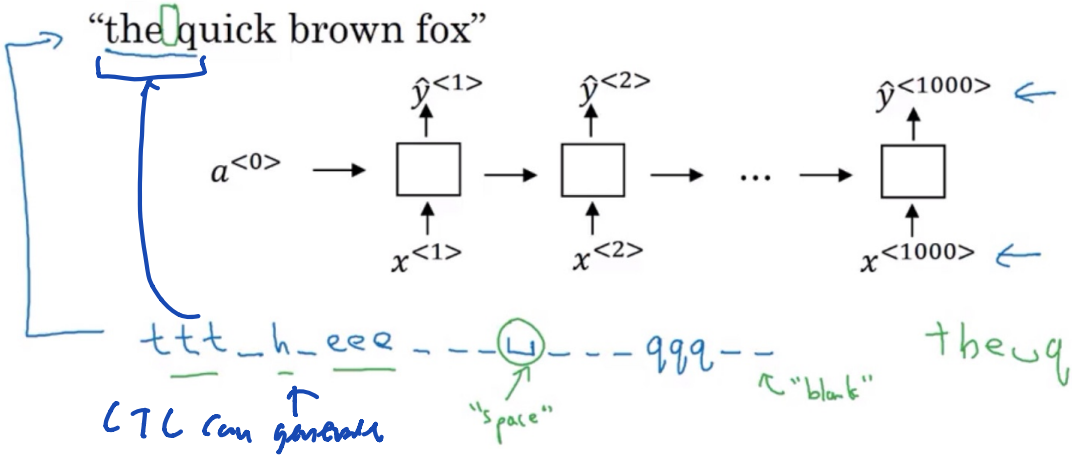
Attention model for speech recognition



CTC cost for speech recognition

(Connectionist temporal classification)

equal no. of x and $\hat{y}$

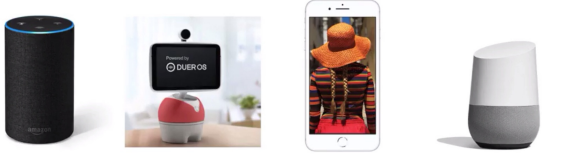


Basic rule: collapse repeated characters not separated by "blank"

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

Trigger Word Detection

What is trigger word detection?



Amazon Echo (Alexa) | Baidu DuerOS (xiaodunihao) | Apple Siri (Hey Siri) | Google Home (Okay Google)

Trigger word detection algorithm