5 Sequence models

Andrew Ng

Note by:
Hossein Mamaghanian
amirhm@gmail.com

"Sequence Models"

"Notations"

Imagine an example of input sequence (a sentence here), where we want to find the "names"

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

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

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ ... $y^{<9>}$ $T_y = 9$

$<\cdot>$ to refer the index in sequence,

$X^{(i)<t>}$: index t in training example i

T_x: length of sequence, $T_x^{(i)}$: length of i^{th} training example.

here T_x and T_y are equal but not necessarily always same

"Representing words"

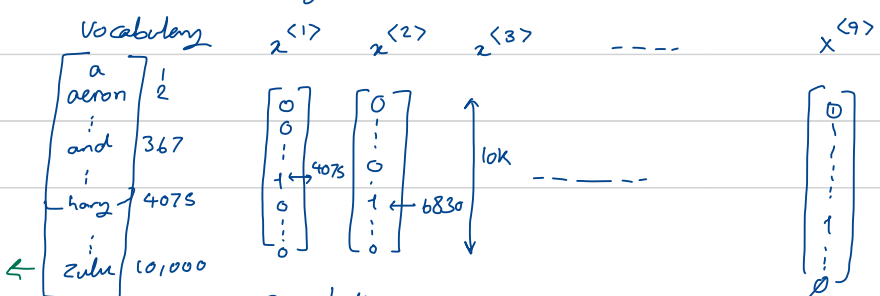
we use a dictionary

x: harry Potter and Hermione ...

in commercial NLP

Dic. size 30k~50k

Maybe even 1M



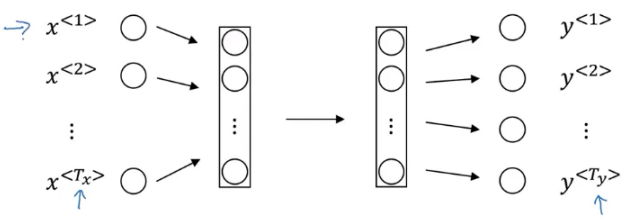
One-hot : because only one index is non zero

it is solved as a supervised $X \longrightarrow y$ learning , input $X^{(i)} \in R^{10^k \times 9}$

if a word doesn't exist in our vocabulary we could assign by a word unv ,

→ how to lean such mapping ?

Why not a standard network?



such naive net. does n't work well

Problems:

- Inputs, outputs can be different lengths in different examples. → even Padding... is not a good idea
- Doesn't share features learned across different positions of text.

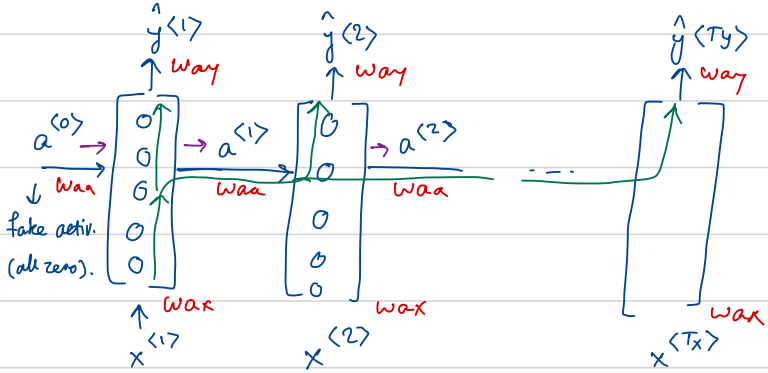
↳ the main Problem, with such model for sequence models.

→ for example in such Nets , it is not easy to generalize that some word in other positions also could appear, (similar to Conv. Nets,).

→ Input size also huge , weights for layers could be enormous.

→ To overcome these problems, we use an architecture which is called "RNN" Recurrent neural networks"

"Recurrent Neural Networks"



→ where inputs are given in sequential order.

→ This is for case $T_x = T_y$ slightly different if not.

→ every output $\hat{y}^{<t>}$ is calculated based on info from previous sequence. /activation are feed to steps after.

weight are same for all layers.

→ This architecture does not take into accounts the words after which is important (for this we could have BRNN : Bi-directional RNN).

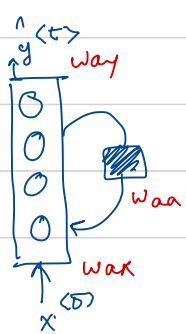
for example:

he said 'Teddy Roscvelt was a great President

" "Teddy Bears are on sale"

To decide if Teddy is part of name we need info from later

Sometime RNN are represented as:



"Forward Propagation"

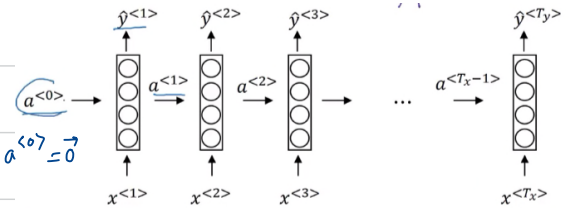
$$a^{\langle 1 \rangle} = g_1(W_{aa}\, a^{\langle 0 \rangle} + W_{ax}\, x^{\langle 1 \rangle} + b_a)$$

$$\hat{y}^{\langle 1 \rangle} = g_2(W_{ya}\, a^{\langle 1 \rangle} + b_y)$$

$$\vdots$$

$$a^{\langle t \rangle} = g(W_{aa}\, a^{\langle t-1 \rangle} + W_{ax}\, x^{\langle t \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = g(W_{ya}\, a^{\langle t \rangle} + b_y)$$



→ activation function Tanh / Relu (More common)

→ sigmoid (if binary)

softmax ...

in case of more common Tanh other methods are taken to overcome problems like vanishing gradients.

Notations: $a \leftarrow W_a \; x$

"Simplified Notations"

More common representation.

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

$$\hat{y}^{\langle t \rangle} = g(W_{ya}\, a^{\langle t \rangle} + b_y) \qquad \hat{y}^{\langle t \rangle} = g(W_y\, a^{\langle t \rangle} + b_y)$$

→ W_a is the concatenation of W_{aa}, W_{ax}

$$W_a = \left[\begin{array}{c|c} W_{aa} & W_{ax} \end{array}\right]$$

if a size is 100 → W_{aa} : 100 × 100

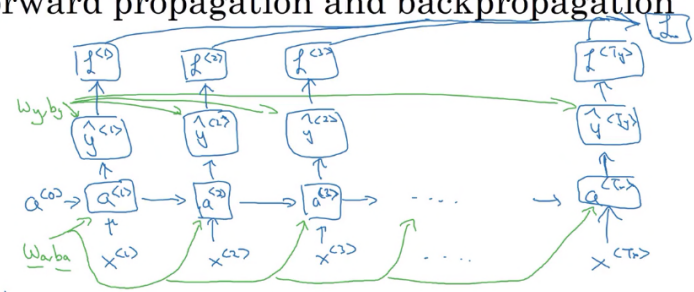
W_{ax} : 100 × 10,000

$$[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] = \begin{bmatrix} a^{\langle t-1 \rangle} \\ \hline x^{\langle t \rangle} \end{bmatrix}$$

"Backpropogation"

logistic-loss or cross entropy loss

Forward propagation and backpropagation



$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{(t)} \log \hat{y}^{(t)} - (1-y^{<t>}) \log (1-\hat{y}^{(t)})$ ← logistic loss (cross entropy loss)

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

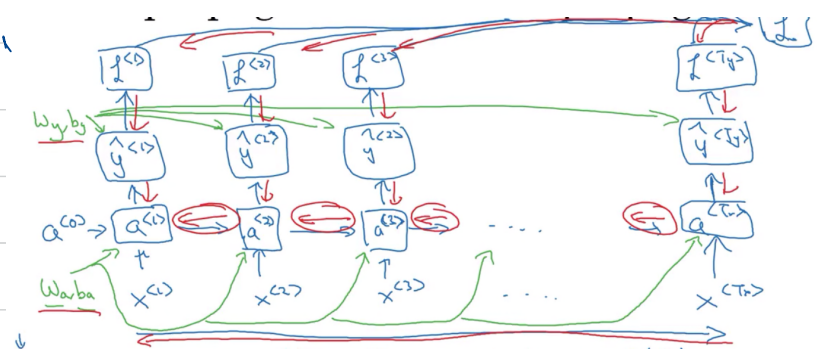
Andrew N

in forward computation for every timestamp t, we want to calculate

$a^{<t>}, x^{<t>} \xrightarrow{\text{calc}} a^{<t>}$ → using W_a, b_a

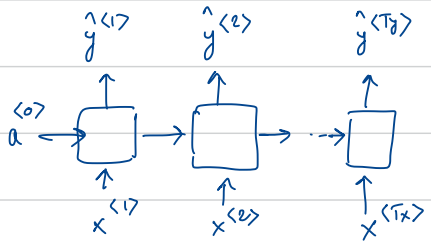
and $a^{<t>} \xrightarrow{\text{calc}} y^{<t>}$ using W_y, b_y

In Backpropagation time, error is propagated backward, since it is in the reverse direction of timestamp, it is called "B.P. through time"



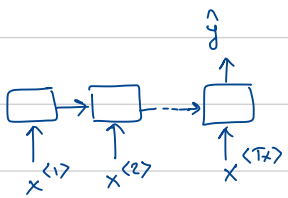
"RNN types"

Many-to-Many



$T_x = T_y$

Many-to-One



ex. sentiment classification

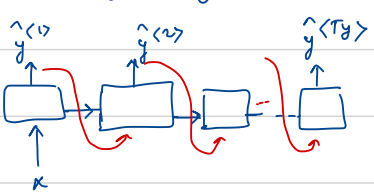
x = text

y = 0/1 1--5

One-to-One

One-to-many

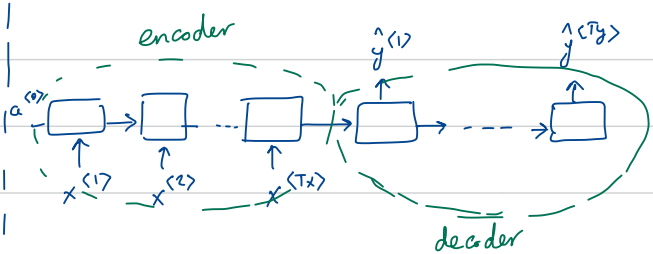
$x \rightarrow y^{\langle 1 \rangle} y^{\langle 2 \rangle} \cdots y^{\langle T_y \rangle}$



in reality, we use output of previous as input for next.

Many-to Many (2nd version $T_x \neq T_y$)

ex: Machine translation



encoder reads the input and decoder generate the output.

"language Model"

→ Model of language which gives a probability for each sentence.

{ The apple and pair salad — $P(\_\_.) \leq 3.2 \times 10^{-13}$

" " pear salad ✓ (More Probable) $P(\dots) \leq 5.7 \times 10^{-10}$

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

language modelling with an RNN.

→ Training set: large corpus of english texts.

→ Step 1: Tokenize.

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

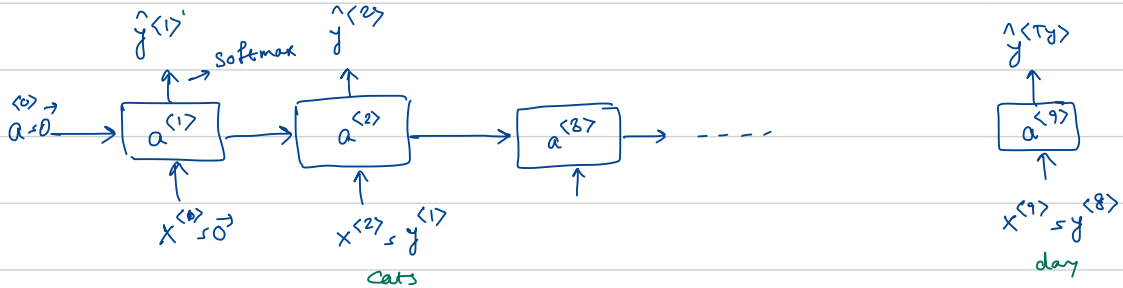
from dic. → $y^{<1>}$ $y^{<2>}$ ------

The egyptian Mau is a breed of cat <EOS>

→ if word does not exist in vocabulary, replaced by <UNK>

→ another word <EOS> is sometimes used to mark end of sentence.

"RNN model"



$\hat{y}^{<1>}$ is actually the probability of every word in dic to be first word

P(a) P(aaron) ---- P(cats) -- P(zulu) P(<EOS>)

$\hat{y}^{<2>}$ P(____ / cats) : average

$\hat{y}^{<3>}$ P(____ | "cats average") : 15

⋮

→ This RNN in every timestep learn to guess the next word.

Cost function (softmax cost)

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

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

→

if network correctly learned, given previous words could predict the next words.

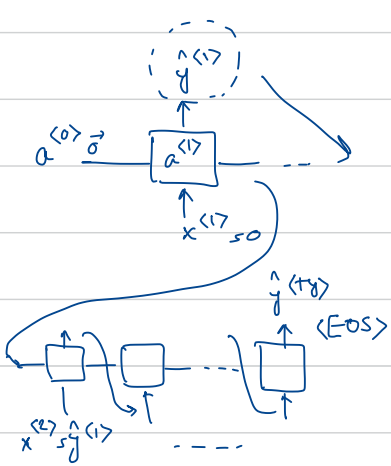
$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) P(y^{<2>} | y^{<1>}) P(y^{<3>} | y^{<1>}, y^{<2>})$$

1st softmax × 2nd softmax × 3rd Timestamp softmax

Sampling a sequence from a trained RNN

→ "To verify or check the model"

for sampling



first step $\hat{y}^{\langle 1 \rangle}$ is the distribution of every word as first word in sentence.

then we can randomly sample based on this distribution. (np.random.choice)

Then this word, will be given to next time-stamp and so on to generate sentence. until we see <EOS>.

Character-level language model

→ Similarly, it is possible to have character-level language model.

→ Vocabulary: [a, b, ---, ␣, ., ,, ;, 0, ---, A, --- Z], $y^{\langle 1 \rangle}$, $y^{\langle 1 \rangle}$ --- are characters!

cons
- → word-level is superior due to better capturing the relation
- -ship between word in the sentence.
- - computationally heavy

pros.
- No unknown word
- Not wide-use.

example generated texts

Sequence generation

News	Shakespeare
President enrique peña nieto, announced sench's sulk former coming football langston paring.	The mortal moon hath her eclipse in love.
"I was not at all surprised," said hich langston.	And subject of this thou art another this fold.
"Concussion epidemic", to be examined.	When besser be my love to me see sabl's.
The gray football the told some and this has on the uefa icon, should money as.	For whose are ruse of mine eyes heaves.

"Vanishing gradient with RNN"

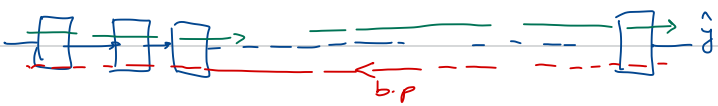
Take example of sentence in NLP, sometimes the words has very long relashanship.

→ The cat, which already ate ------, was full

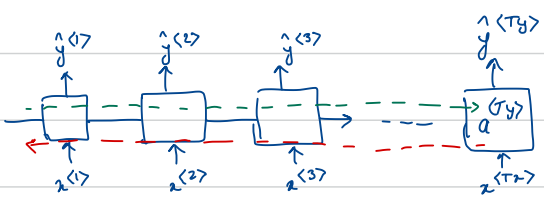
The cats, " " " ", were full

The basic RNN structure that seen so for is not very good at capturing these for dependancies,

Similar to problem in very deep neural networks, vanishing gradient is problem here



similar in RNNs.



The gradient in b.p has very hard time to affect earlier layers.

The main weakness of basic RNNs: The effect of each output will be very limited to close neighbours. and will not travel much for to affect earlier layers.

exploding gradients: another problem which could exist in RNNs, similarly but this is not the main challenges in training RNNs.

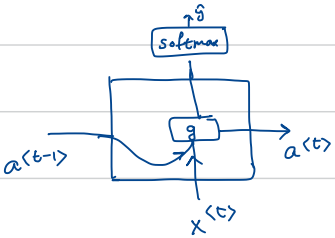
→ in this case we see "NaN" in computations.

we could use gradient clipping: put some threshold on gradient if bigger, rescale. or clip. (relatively robust).

"gated recurrent unit" (simplified)

→ modification to RNN hidden unit that make it much better in capturing long range relationship and help with "vanishing gradient problem"

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



c = memory cell.

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

↳ for gru, c is output a value, this notation is used more for later describing lstm

candidate for update $c^{\langle t \rangle}$

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

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

↓ gate unit

↳ gate unit will decide weather we update $\tilde{c}^{\langle t \rangle}$ or not.

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

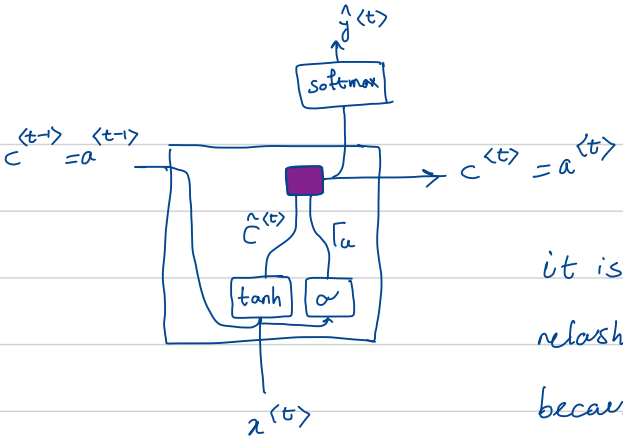
↑ element wise.

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

$c^{\langle t \rangle} = 1$ ⇒ ↓ since memorize $c^{\langle t \rangle} = 1$, then select "was"

↑

The cat, which _ _ _ _ _ , was full.



it is very good at memorizing long range relashionships.

because Γ_u is very easy to set to zero

then $c^{\langle t \rangle} = c^{\langle t-1 \rangle}$,

and since Γ_u is very close to zero, it does not suffer from vanishing gradients.

$c^{\langle t \rangle}$ is vector $\mathbb{R}^N$

$\tilde{c}^{\langle t \rangle} \in \mathbb{R}^N$

$\Gamma_u \in \mathbb{R}^N$, intuition is that every element in Γ_u memorize part of information in the sentence.

Full GRU:

$\tilde{h}$ $\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$, Γ_r weights are added to simplified version to decide which elements of $c^{\langle t-1 \rangle}$ are relevant for calculating the $\tilde{c}^{\langle t \rangle}$

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

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

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

some literature use this notation.

LSTM (long, short term memory)

Paper: Hochreiter & Schmidhuber 1997, long short-term memory

other type of units which allows to learn longer dependencies one LSTM units. (even more power full than gru).

Full GRU:

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

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

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

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

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

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

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

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

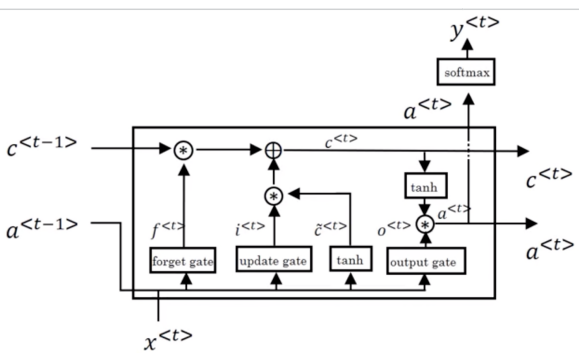
output $\Gamma_o = \sigma\left(W_o\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right)$

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

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

↙ is no longer directly $= c^{\langle t \rangle}$

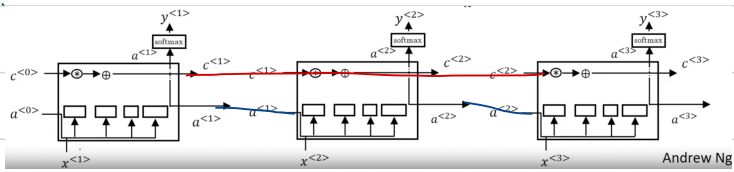
in LSTM, we have 3 different gates compared to 2 in Gru.



if we connect these units, we build our net

by setting right Γ_u, Γ_f correct value, it is easy to keep $c^{\langle 3 \rangle} = c^{\langle 1 \rangle}$ (memorize for)

true for gru as well.



Sometime gate value could made to also on previous memory cell values, ($c^{\langle t-1 \rangle}$)

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

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

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

↑ peephole connection

this is called peephole connection.

→ the relationship between $c^{\langle t-1 \rangle}$, Γ would be one-to-one, (element i^{th} only affect i^{th} element of Γ)

→ LSTM is more strong, much complex. more powerfull. historically more proven choice.

gru is more simpler and easier to build bigger models, train faster. (may be just as well).

→ Gru is more recent than lSTM.

Bidirectional RNN

forwad/Uni directional RNNs has a problem that don't take into account the next words

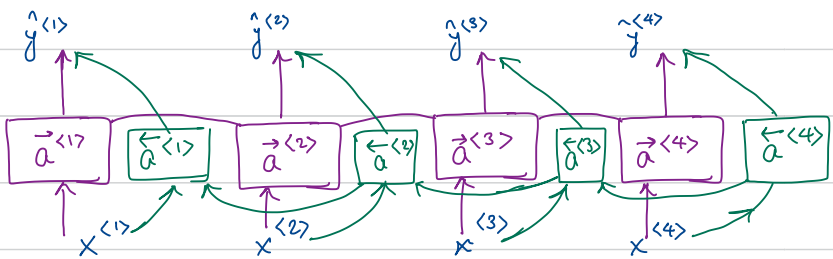
he said, "Teddy Bear are on sale"

u o , "Teddy roseveltc ..."

→ This is true wheather standard RNN / GRU / LSTM units,.

could be unit of

RNN
GRU
LSTH →

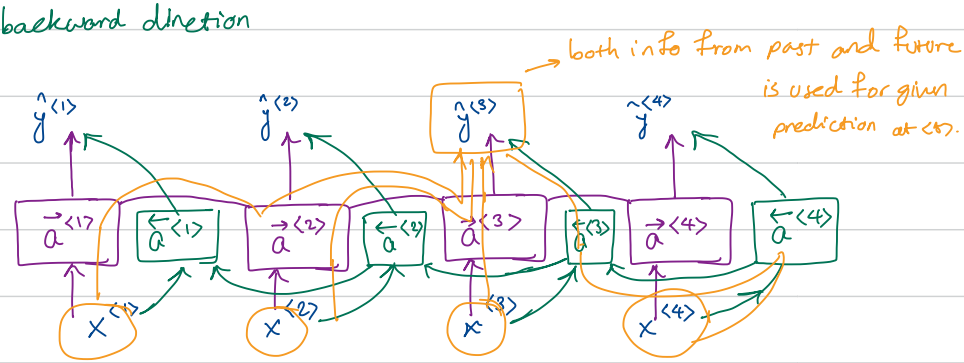


Acydic graph

first → forward direction

second → baekward direction

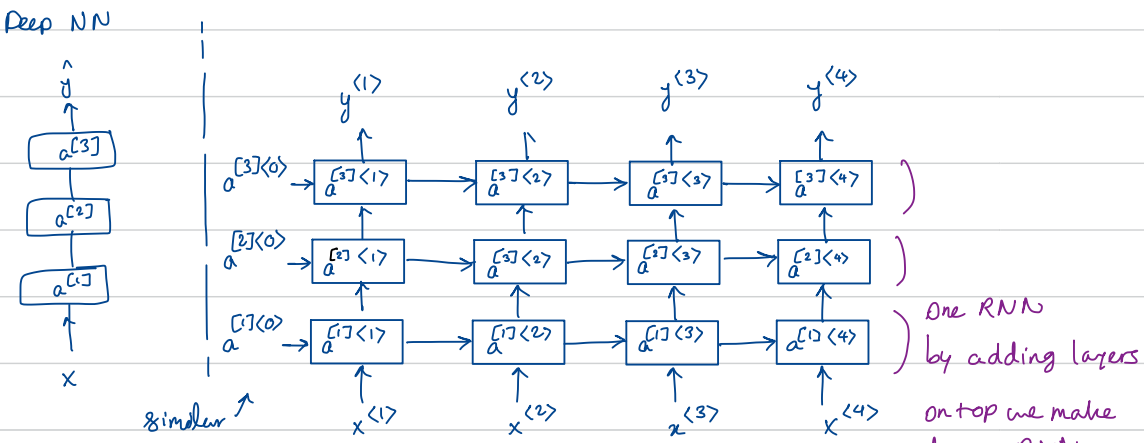
$$\hat{y}^{<t>} = g\left(W_y \left[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}\right] + b_y\right)$$



in text (NLP), BRNN W/ LSTM block reasonable choice.

→ disadvantage: entire sequence is needed. for example in speech, we have to wait for full sentence to start predicting. (realtime is not possible).

"Deep RNN"



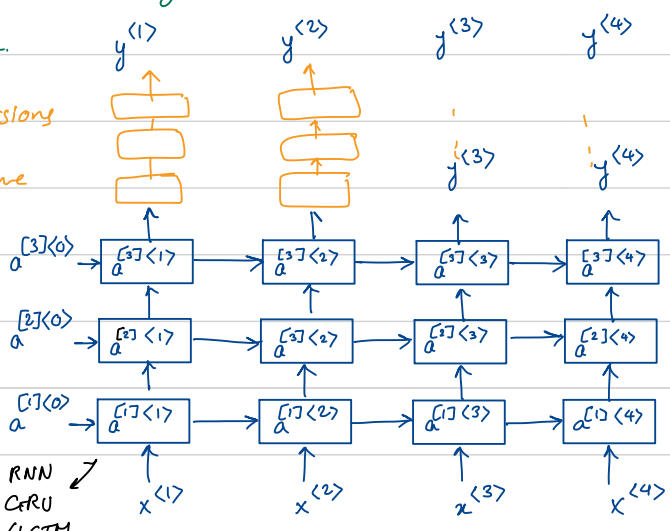
$$a^{[2]\langle 3\rangle} = g\left(W_a^{[2]}\left[a^{[2]\langle 2\rangle}, a^{[1]\langle 3\rangle}\right] + b_a^{[2]}\right)$$

Unlike NNs, usually for RNN, having 3 layers is quite a lot and usually deeper versions are not used.

→ instead we may see these versions which after RNN layers we have DNN layers, which are not connected horizontally.

→ blocks as well could be GRU, or LSTM blocks,

→ also we could build Deep BRNN as well.



"Back propagation"

NLP and word embedding.

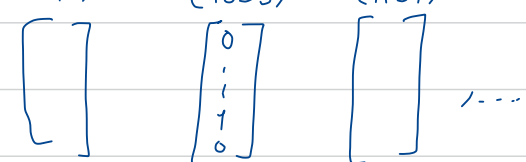
→ "word embedding"

let's model to learn analogies like: Men ~ Women is King ~ queen.

weakness of one-hot representation is that treat each word until itself, and it is difficult to generalize.

Vs[a, aeron, ___ zulu, <UNK>]

Man (5391) Women (9853) King (7157)



O_{5391} O_{9853} O_{7157}

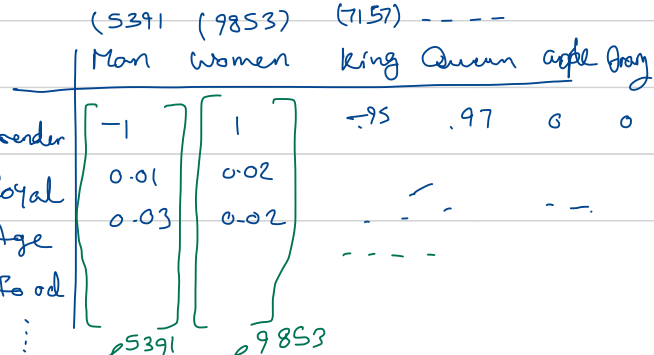
I want a glass of apple juice

orang juice

inner-product of each of these vectors are zero. then really difficult to generalize that apple/orang, king/queen, ---

→ another alternative to one-hot representation is to learn with Feature-representation.

→ For example 300 dimentional vector for representing each word. $(e^{5391}, e^{9853}, \ldots \in R^{300})$

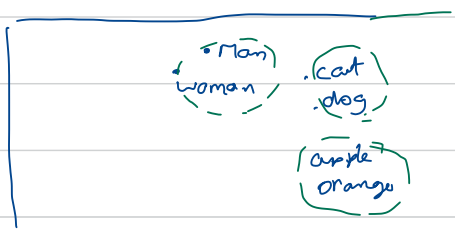


if we use such embeddings, then representation for apple and orang are close thing, then it would be easier to generalize.

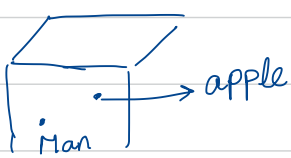
In practice we learn these featurized representation, and because of this is not easy to really interpret each feature

→ To visualize

t-SNE → 300 ~ 2D



↳ These vectors are called embeddings.



"Using word embeddings."

→ "Named entity recognition example"

if learns one, it is easy to learn that Robert also should be name. (bc of apple/orange relashionship).

→ how about this more rare words?

(Normally for such task BRNN should be used)

if we have small training set, we may have not seen "durian cultivator", but if we use word embeddings, we should be able to recognize that it is "fruit farmer."

→ idea → using huge data set for word embeddings. tB ~ 100B word

→ later just smaller data set for the task of name entity will be enough

"Transfer learning"

1. learn word embeddings from large text corpus. (or just download)
2. Transfer embeddings to new task with smaller set is available. (One Pros is that now instead of one-hot vector size of 10^k ~> 300

3. Optional : fine-tune embeddings with new data.

generally Transfer learning make sense when for the given task, relatively smaller data is available.

NLP Task which transfer learning could be useful:

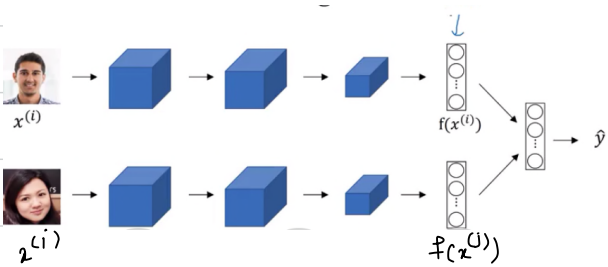
→ Name entity / Text parsing / Co-reference recognition.

less useful: language modeling, Machine Translation.

(specially when we have a lot of data dedicated to our Task)

→ Similar concept to face encoding.

→ when we trained a seasame network : face $\xrightarrow{\text{Map}}$ 1280 vector



- encoding / embeddings are similar concepts.

Difference is that in seamese net we could encode for new faces not seen before.

but in word embedding, we have fixed vocab set. $|V| \leq 10{,}000$. $e_1, \dots e_{10000}$

Properties of word embedding.

linguistic analogies: Man → Women as king ~ ?

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

e_{man} e_{women} ----

(e_{5391}) (e_{9853})

Man → Women

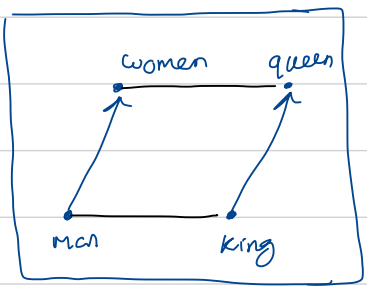
$e_{man} - e_{women} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

King → ?

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

↳ queen

Analogies using word vectors:



300D

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

→ find word w:

→ $\arg\max_w \text{sim}(e_w, e_{king} - e_{man} + e_{women})$

as a performance check for embeddings.

for example 30–75% on analogy reasoning task.

This parallel path is true if visualization done on original 300D space not in 2D map (ea. by using t-SNE).

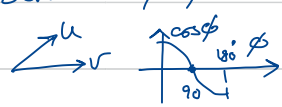
Types of similarity functions could be used:

Cosine Similarity:

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

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

(called cosine, bc it is really the cos angle between u, v)



→ less popular:

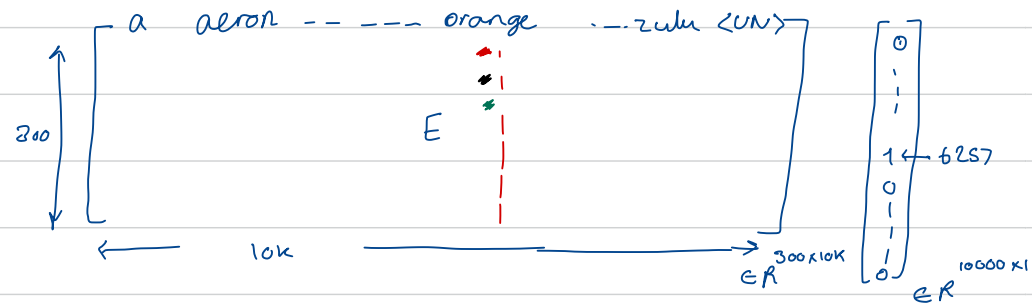
euclidian distance: $\|u - v\|^2$

Problems of word embeddings

Notation for one-hot rep.

O_{6257}

embedding matrix



$E \cdot O_{6257} = \begin{bmatrix} \\ \\ \end{bmatrix} = e_{6257}$

$\rightarrow E \cdot O_j = e_j$ embedding for word j

In practice we really do not use matrix multiplication for this, we use dedicated functions to select the jth column (keras.embedding).

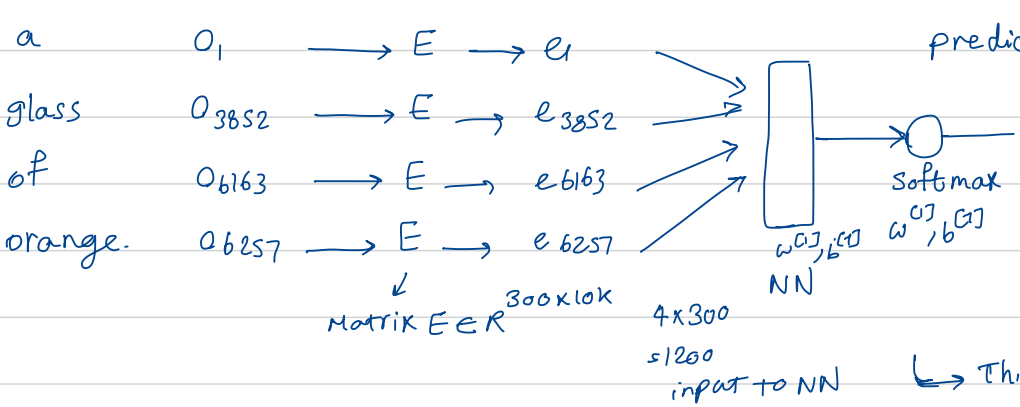
"learning word embeddings"

'Natural language model' → (Paper: Bengio et al 2003, a neural probabilistic language model)

consider ex. I want a glass of orange Juice

4343 9696 1 3852 6163 6257

we build problem like this. that we want to predict the word for example from last 4 word.



↳ This will end up learning E.

other context/target pairs.

→ last 4 words.

I want a glass of orange juice to go along.

- 4 word on left & right : a glass of orange --- to go along with
- last 1 word : orange
- Nearby 1 word → (skip gram) ex. for orange: glass (skip of)

word2vec

skip-grams.

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

Context	→	Target
orange		juice
orange		glass
orange		my
⋮		⋮

(supervised learning).

randomly we select the target word from sentence.

→ obviously this would be difficult supervised learning to really predict target, but our goal is just to learn word embeddings.

(No NN between, only E → softmax).

$O_c \longrightarrow E \longrightarrow e_c \longrightarrow (\text{softmax}) \longrightarrow \hat{y}$

context

θ_t: parameter associated with output t (parameter of softmax)

$$\text{Softmax}: \; p(t/c) = \frac{e^{\theta_t^T \cdot e_c}}{\sum_{j=1}^{10K} e^{\theta_j^T \cdot e_c}}$$

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

$$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$ one_hot rep.

Problems with soft max classification

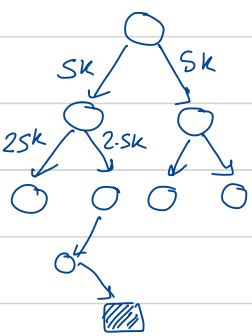
→

$$P(t|c) = \frac{e^{\theta_t^T \cdot e_c}}{\sum e^{\theta_j^T \cdot \theta_c}}$$

(1) every time when calculating, we need to sum over all vocab in list (ex. 10k here).

→ we could use "hinearchial softmax"

→ computation complexity would be $\log|V|$ for softmax = $|V|$



→ In practice hirearchial softmax does not use perfectly balanced tree. (More common words be in top, ——)

(heuristical approach to build tree...)

how to sample context c?

if done randomly, you see a lot of the, of, a, ---- more frequently.

- use heuristically developed algorithms to sample in balance to cover all words

* Another variation CBOW: Take context from surronding middle word and try to predict middle word.

Negative sampling.

→ Algorithms like w2v has the problem of complex computation on softmax layer (due to large size of the vocabulary set).

Negative sampling is a method, tries to solve a much reduced version of supervised learning problem.

→ Imagine a sentence: I want a glass of orange juice to go along ....

Training Data are organized like this.

context	word	target	
orange	juice	1	one positive example and "K" example for negative which randomly found.
orange	king	0	
"	book	0	
"	of	0	→ even though maybe the randomly selected word "of" may really happen on sentence. (ok)
c	x t	y	

↳ "supervised learning problem"

K = 5–20	small datasets
K = 2–5	larger datasets

called negative sampling, since we have one possitive example and then randomly sample negative examples.

Model:

Softmax $P(t/c) = \frac{e^{\theta_t^T \cdot \theta_c}}{\sum_{j=1}^{10k} e^{\theta_j^T \theta_c}}$

to model $P(y=1 / c,t)$ we use logistic regression.

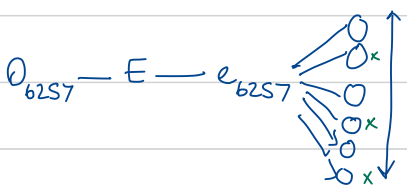
$$P(y=1/c,t) = \sigma(\theta_t^T \cdot \theta_c)$$

if we have k negative example, the ration is k/1 neg/pos example.

to train this model.

flow to train:

example orange 6257



10K possible logistic regression

* but instead of all of them only (k+1) subset is trained

→ training only (k+1) binary logistic regression problem is much cheaper than softmax with 10K units.

→ "selecting negative examples"

Oneway: candidate for negative could be sample based on imperical frequency of word in dic : a lot of the, of, ...

other way: uniformly sample $\frac{1}{|V|}$: not representitive of real significace of words.

emperically suggested formula $P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{i=1}^{10k} f(w_i)^{3/4}}$ $f(w_i)$: observed frequency of w_i in lang.

(kind of trade of b/w two extrem of 1&2)

Glove algorithm (global vectors for word representation)

→ simpler but not as popular as W2V or skipgrams.

Paper: Pennington et.al, 2014. GloVe: Global vectors for word representation).

let X_{ij} : # time i (c, t) appears in context of j (t, c)

→ depends on definition of c, t maybe $X_{ij} \neq X_{ji}$

exp: c, t one the word appear in the proximity of n word $X_{ij} \neq X_{ji}$

: c is the word immediately before t : $X_{ij} \neq X_{ji}$

Model:

$$\text{minimize} \sum_{i=1}^{10K} \sum_{j=1}^{10K} f(X_{ij}) \left(\underbrace{\theta_i^T \theta_j}_{t,c} + b_i^{(t)} + b_j^{(c)} - \log X_{ij} \right)^2$$

↳ similar to "$\theta_t^T \cdot e_c$" as before

Solved wit G.D to minize.

if $X_{ij} = 0$ then $\log X_{ij} = \text{nan}$, then $f(X_{ij}) = 0$ if $X_{ij} = 0$

another point for $f(\cdot)$ is to correct the weigth for

1) very frequent words This, is, of, a, ---

2) less frequent but rather important durion.

→ another point here is that θ_i, e_j one symetric.

→ (both should initialized randomly → G.D → then we can take average

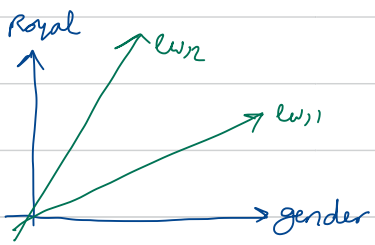
$$e_w^{(final)} = \frac{e_w + \theta_w}{2}$$

GloVE algorithm came later than skip-gram, W2V,... and simplified a lot the learning embeddings.

Important Points.

for word embedding learned by all these algorithms, there is no guarantee to be "human interpretable"

	man	women	king	Queen	...
gender					
Royal					
Age					



$\theta_i^T e_j$

↳ $(A Q_{w_i})^T (A^{-1} \theta_j) = \theta_i^T \theta_j$

→ but parrallelogram for figuring out linguistic analogies holds.

"Sentiment classification"

example The desert is excellent ★★★★

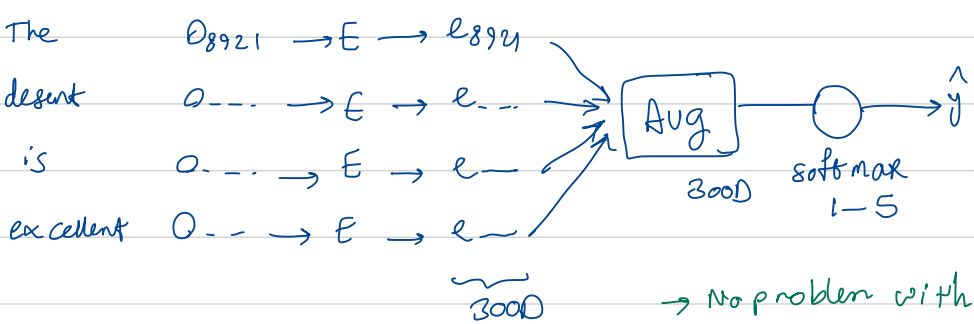
service was slow ★★

completely lacking in good service ★

X Y

challenge: Maybe not of a big training set.

One simple solution, imagine we have word embeddings E



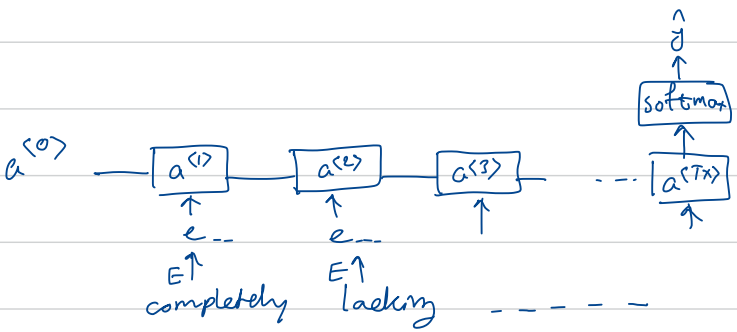
→ No problem with sequence len.

→ ignores word orders!!

ex. completely lacking in good taste, good..., good...

↳ very likely miss this example due to not considering the order.

"RNN for sentiment classification"



many to one architecture

This architecture could learn the orders.

advantages of using pretrained embedding, is that even some word were not in our labed traing set, still would be reognize these and generalize.

De biasing word embeddings.

example: bias in terms of gender, age, sexual social status, orientation, ethnicity, --- which are in texts.

Man : Computer_programer as Women : Homemaker ✳

Father : Doctor as Mother : Nurse. ✳

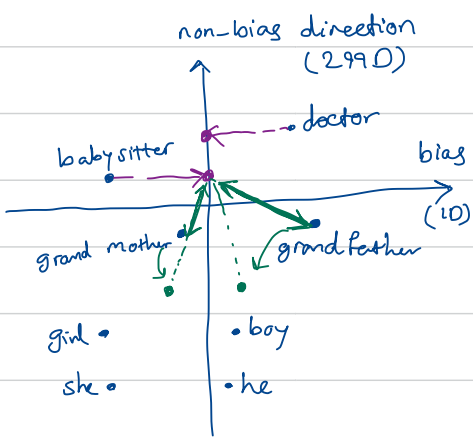
we want to eliminate these type of biases which is learned from texts used for training.

addressing bias in word embeddings.

1. Identify bias direction

$$\begin{cases} e_{he} - e_{she} \\ e_{male} - e_{female} \\ \vdots \\ \text{average.} \end{cases} \longrightarrow \text{find the bias direction.}$$

1.



2. Neutralize: for every word that is not definitional, project to get rid of bias. (doctor, babysitter, ---)

3. Equalize pairs: (like the distant of grand Mom / father to baby sitter, doctor transpose to new position to be equally distanced.

What words has to be neutralized or not?

→ a linear classifier could be trained to find which has to be neutralized. (small set like grand Mom/father are definitional).

→ same for equalization pairs as well.

Sequence to sequence models.

application: machine translation, speech recognition, ..

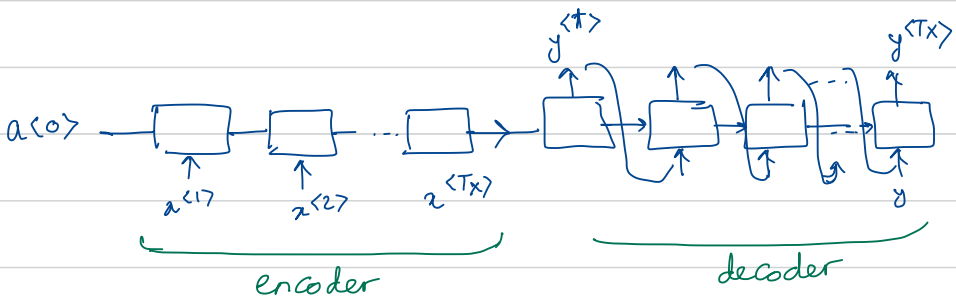
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

Jane visite l'Afrique en septembre

↓

Jane is visiting Africa in september

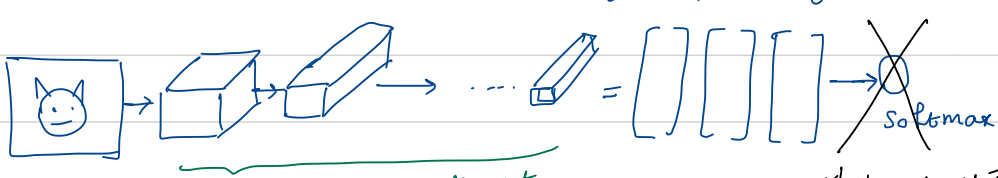
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$



→ This architecture actually work!

(given enough pairs en/fr for training)

Similar architecture also used for image captioning.



→ give this output to decoder to generate caption.

Many groups: Moo 2014

Vinyals 2014

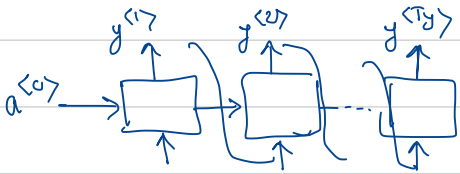
Korpathy and Feifei 2015

:

This as well works pretty decent!

"Picking the most likely sentence"

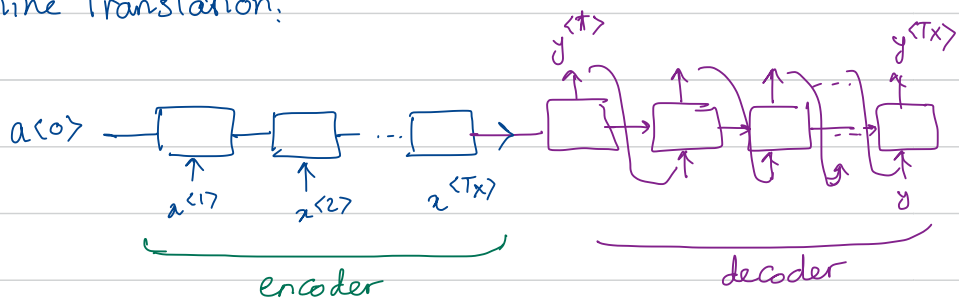
language Model
(to estimate the
probability of sentence)



$P(y^{\langle 1\rangle}, y^{\langle 2\rangle}, \ldots, y^{\langle T_y\rangle})$

→ such architecture as well, we used to generated Novel sentences.

Machine Translation:



decoder is very similar to language
→ model, just the input is not ø, and actually is generated from other sequence (encoder). because we call this:

"Conditional language model"

↳ $P(y^{\langle 1\rangle}, \ldots, y^{\langle T_y\rangle} \mid x^{\langle 1\rangle}, x^{\langle 2\rangle}, \ldots x^{\langle T_x\rangle})$

exp. P(english sent. / french sentence).

but, what we dont want is to sample randomly (similar to what we did while generating Novel sentences).

Jane visite l'Afrique en septembre

different samples, not all good!

1. Jane is visiting Africa in september
2. Jane is going to be visiting ----
3. In september, Jane will ---

we want an english sentence which maximize:

$$\arg\max_{y^{\langle 1\rangle} y^{\langle 2\rangle} \cdots y^{\langle T_y\rangle}} P(y^{\langle 1\rangle}, \ldots y^{\langle T_y\rangle} \mid x)$$

(x: french sentence.)

→ why not just use a greedy search?

greedy search: first $P(\hat{y}^{\langle 1\rangle} \mid x)$ then $P(\hat{y}^{\langle 2\rangle} \mid x, \hat{y}^{\langle 1\rangle}), \ldots$

best first ≠ best 2nd, ...

this is not good, indeed we are interested in best joint probability $P(\hat{y}^{\langle 1\rangle}, \hat{y}^{\langle 2\rangle} y^{\langle 3\rangle}, \ldots \mid x)$

for example $P(\text{sentence 1} \mid x) > P(\text{sen. 2} \mid x)$

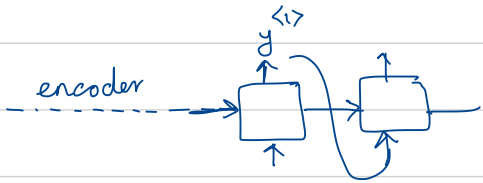
→ if we use greedy $P(\text{Jane is going} \mid x) > P(\text{Jane is visiting} \mid x)$

is more frequent in english

but not correct choice, given whole sentence.

also given 10k, vocab list, searching for all combination is also not good!

→ Beam search Algorithm



① $P(y^{\langle 1 \rangle} / X)$ → Find n most probable n: Beam width = 3 (here).

1. in 2. Jane 3. september.

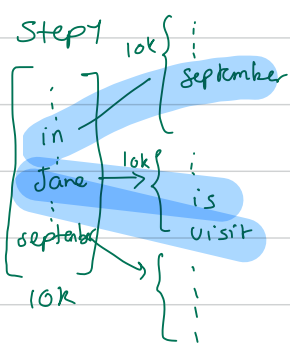
② a. Take First best Find $P(\hat{y}^{\langle 2 \rangle} | X, \hat{y}^{\langle 1 \rangle})$

→ $P(\hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle} | X) = P(y^{\langle 1 \rangle} / X)\ P(y^{\langle 2 \rangle} | X, y^{\langle 1 \rangle})$

we have this from ① 10K choice

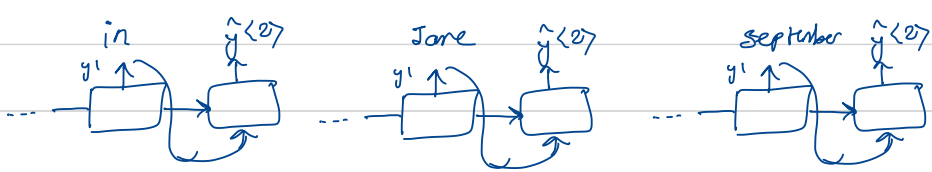
b. Same for second and 3 word as well 10k+10k choice.

Again Pick 3(Beam width) most probable.



for example in second step we rejected "september" as the First word.

→ we only need 3 initantine of network to evaluate 30k output.



Same routine continues:...

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

in september → 10k { Jane

$P(\hat{y}^{\langle 3 \rangle} \mid x, y^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle})$

Jane is → 10k { visiting

Jane visits → 10k { africa

This routine of search continues, untill it find <EOS> and most likely sentence.

Beam_width = 1 → becomes greedy search.

→ Refinement to beam search

length normalization

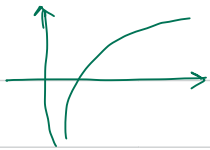
$$\arg\max_{y} \prod_{t=1}^{T_y} P(y^{\langle t \rangle} \mid x, y^{\langle 1 \rangle}, \dots y^{\langle t-1 \rangle})$$ → these probabilities are small number and product will be very small for float

instead we use log

↳ $$\arg\max_{y} \sum \log P(y^{\langle t \rangle} \mid x, y^{\langle 1 \rangle}, \dots y^{\langle t-1 \rangle})$$

→ More numerically stable.

log is strictly monotonically increasing function, should give same results.



→ One of the problem with these objective function is that it always prefers smallest sentence. (as $T_y \uparrow$ $P \downarrow$)

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

the term helps alot to decrease the penalty for longer sentences. or heuristically $\frac{1}{T_y^{\alpha}}$

- $\alpha = 0.7$ (good).
- $\alpha = 1$ full normalization
- $\alpha = 0$ No ~ at all

Summary :

we run the algorithm of Beam-search and we find a lot of possible sentence. (till $T_y = 30$)
Then we use the "Normalized log-likelihood" to pick the best.

→ Beam-Width

- larger : More probable to find bes / More computation
- small : worse results / faster.

10 ~ 100 (For products, 100 is already huge), research 1k ~ 3k to get the best results. (improvement ofcourse diminish as goes very high)

→ Beam search Unlike (BFS) or (DFS) is not guaranteed to find the best but much faster. (Breadth / Depth First search).

Error Analysis in Beam search

Beam search is heuristic search algorithm and does not gwarntee to find the best possible. suppose:

→ Input : Jane visite l'Afrique en septembre

human : Jane visits Africa in september (y^*)

Algo : Jane visited Africa last september $\hat{y}$ (Pretty bad translation).

who is to blame?

since here we had 

two part 1. RNN model

2. Not globally optimal search

error Analysis: there are two possiblity:

case1 : $P(y^*|x) > P(\hat{y}|x)$

Means, our RNN correctly knows that y^* is more probable (B.S is fault)

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

Beam search was fine, RNN is fault. → Increase Beam-width.

(small complication, May be due to bad normalization at B.S could be responsible as well. → all relevant technique for net (More data, archite---)

Normally analysis done for a number of examples to find the most impactful of errors.

human	Algo	$P(y^*\|x)$	$P(\hat{y}\|x)$	fault
- -	- - -	2×10^{-10}	1×10^{-10}	B
- -	- -	⋮	⋮	R
- -	- - -	⋮	⋮	R
				⋮
				B

%

"Bleu score" (bilingual evaluation understudy)

How about if there are multiple good translations?

*understudy is theatre is the person that learn to take the role of senior player if necessary)

Paper: Papinei 2002, Bleu ...

Imagine example that we have 2 reference human translation →

ref 1: The cat is on the mat

ref 2: there is a cat on the mat

MT: the the the the the the the (Pretty awful).

Simple Precision: $\frac{7 \text{ (appeard on ref)}}{7 \text{ (word on MT)}} \leq 1$ ← bad!!

on MT words (the)

Modifid $P = \frac{2 \text{ (count\_clip)}}{7}$ → Maximum number of appearance over all reference sentence.

→ 2nd exp: MT: The cat the cat on the mat

bigrams (P on every two-pairs).

	count	$count_{clip}$
the cat	2	1
cat the	1	0
cat on	1	1
⋮	1	1
		1

$= 4/6$

$$P_1 = \frac{\sum_{\text{unigrams} \in \hat{y}} count_{clip}(\text{unigram})}{\sum_{\text{unigrams} \in \hat{y}} count(\text{unigram})}$$

↓ unigram

$$P_n = \frac{\sum_{\text{n-grams} \in \hat{y}} count_{clip}(\text{n-gram})}{\sum_{\text{n-grams} \in \hat{y}} count(\text{n-gram})}$$

↓ n-gram

if one exact ref is found $P_1, \ldots P_n = 1.0$

More detail:

p_n: Bleu score on n-grams only

combined bleu score $\quad BP \cdot \exp\left(\frac{1}{K}\sum_{n=1}^{K} p_n\right)$

BP (heavy penalty): to penalize generating short sentences.
(short sentence have higher chance of getting higher score).

BP (heuristically defined)

$$BP = \begin{cases} 1 & \text{if MT\_out\_len} > \text{ref\_out\_len} \\ \exp(1 - \text{MT\_out\_len}/\text{ref\_out\_len}) & \text{otherwise} \end{cases}$$

"Attention Models"

Paper: Bahdanau, et al 2014. Neural machine translation.

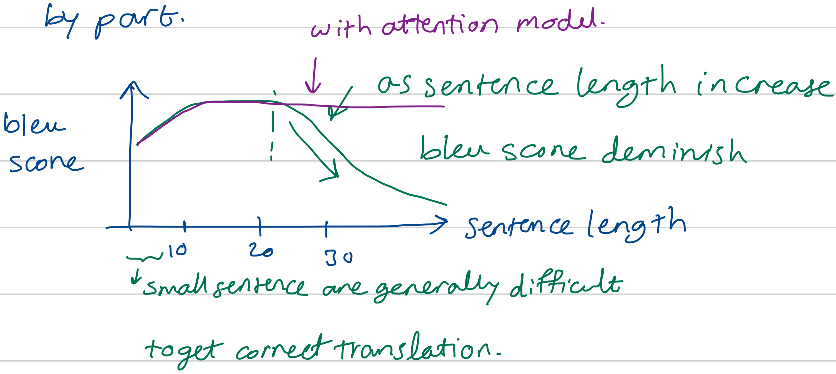
one of the most influential idea's in deep learning.

Problem with longe sentences.

exp → fr: Jane s'est rendu en Afrique ——————— —, - - - , - -.

en: Jane went to Africa last - - - - - - - -

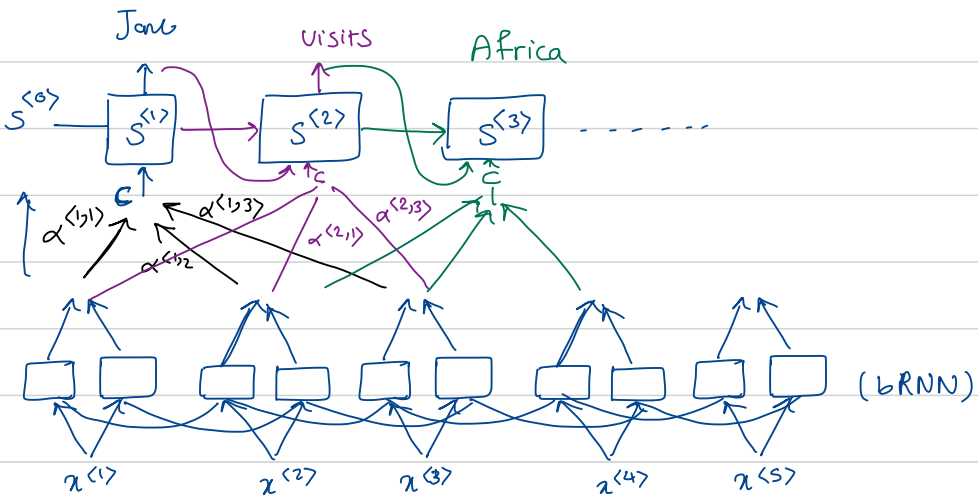
since for MT, we use encoder/decoder, having a long sentence, it is very difficult for encoder to memorize whole sentence and then generate the sentence. human as well, does the translation part by part.



with attention model, Network does the translation more like human in small parts.

Originally dweloped for MT but later extended for many others.

"Attention model intuition"



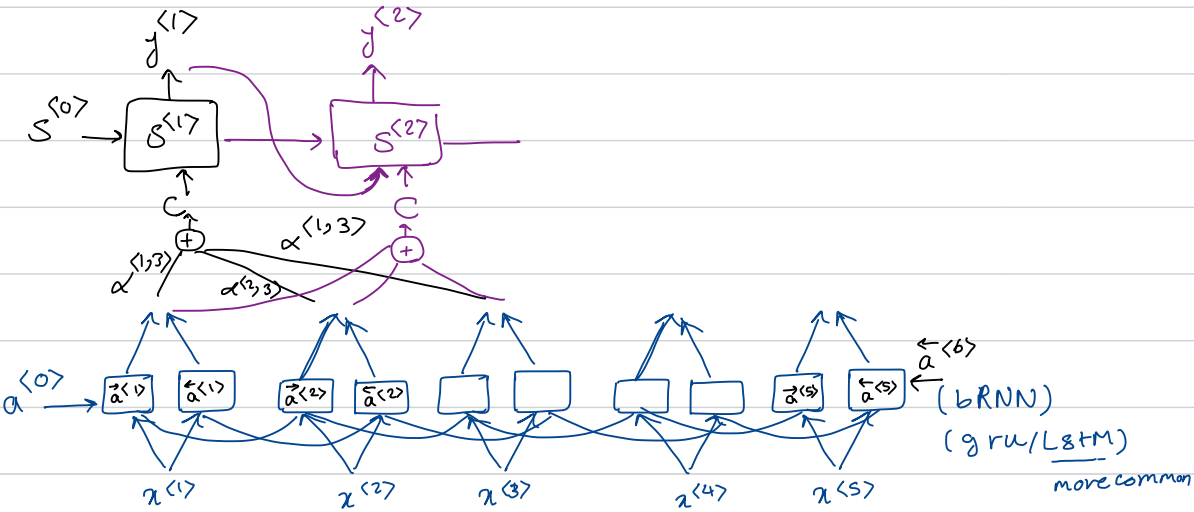
Jane visite l'Afrique en septembre

the idea is that while generating the translation we also gave a context from similar position of french sentence

$\alpha^{\langle t,t'\rangle}$ t: english, t': French

$\rightarrow \begin{cases} s^{\langle t-1\rangle} \\ \overrightarrow{a}^{\langle t\rangle}, \overleftarrow{a}^{\langle t\rangle} \end{cases}$: this help MT to also look at the local position of original sentence.

Attention model : (detail)



t' : used for French sentence

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

→ context calculated by a sum: $\sum_{t'} \alpha^{<1,t'>} = 1$ *

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

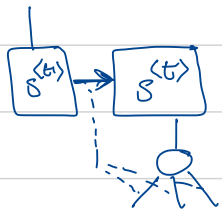
* Indeed $\alpha^{<t,t'>}$ is amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

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

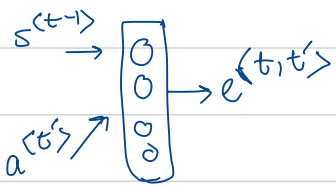
→ ensure to sum to 1

How to compute $e^{\langle t,t' \rangle}$?

$e^{\langle t,t' \rangle}$ should be calculated from $\begin{cases} s^{\langle t-1 \rangle} \\ a^{\langle t' \rangle} \end{cases}$



we could train a very simple model to learn this function and trust it



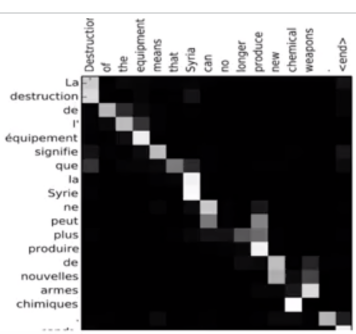
due to using attention, this algorithm runs at quadratic time/cost but for MT where T_x, T_y are not very big, it could be acceptable. ($T_x \cdot T_y$ times).

→

same idea of attention used in image captioning and other area's as well.

Paper: Xu et. al. 2015. attend and tell:...

Visualization of $\alpha^{\langle t,t' \rangle}$:

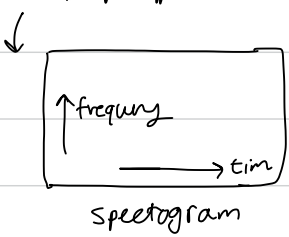


→ plotting the attention

→ usually pays attention to correct word.

"speech recognition"

x
audio clip → y
transcript



→ these are used as input to network.

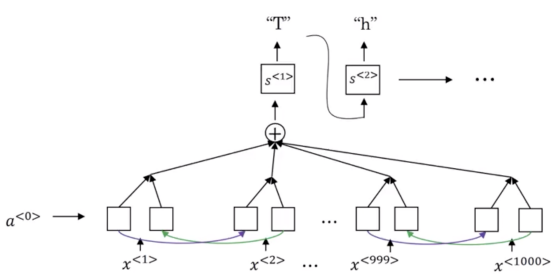
(human ear as well does something similar).

before phonemes concepts used: like.

The quick brown → de kuik braun (hand engineered features).

{ 300h / 3000h used in academia
{ commercial product 100Kh of data for training.

one solution could be to use Attention models for speech recognition

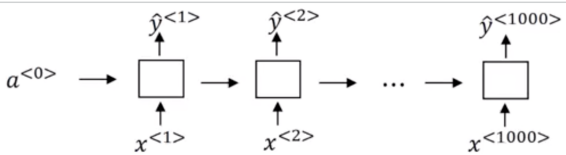


Second option

CTC cost for speech recognition

(connectionist temporal classification) Graves et. al 2006.

since imagine $x^{<1>} \_ x^{<1000>}$ are from, 'to' second speech, then y is really dont have that many characters.



→ inreality bidirectional gru/ LSTM is used. and deeper models.

output may be like:

space

ttt _ h - eee _ _ _ _ _ u _ _ _ qqq _ _ _ _ _

↳ blank

CTC collapse → the quick ...

→ this allows network to have 1000 output by repeating letters.

both options (using attention models and CTC models both works)

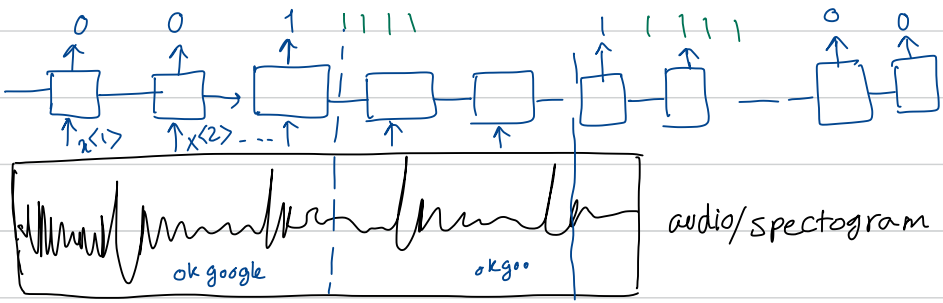
Trigger word detection

(or keyword detection).

examples: devices which wake by a keyword; hey siri, ok google

(still evolving area).

one way could be..



everytime in training data trigger word finishes, output will become 1.

To improve on unbalanced output label (lots of 0 and rare 1s), we could output for few more timestamps.