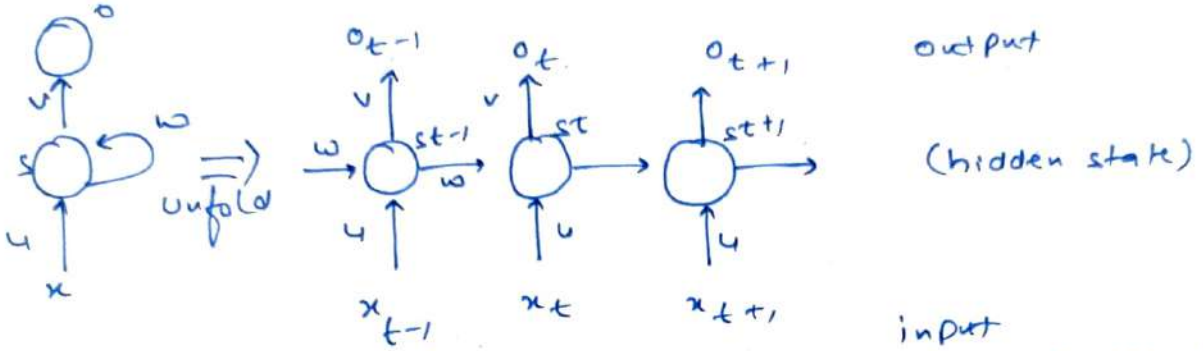
Recurrent neural network :- Difference between feed forward neural network and Recurrent neural network ①

RNN :- same task for every element of a sequence with the output being dependend on previous computation

- RNN have a memory which stores information about what has been calculated so far.

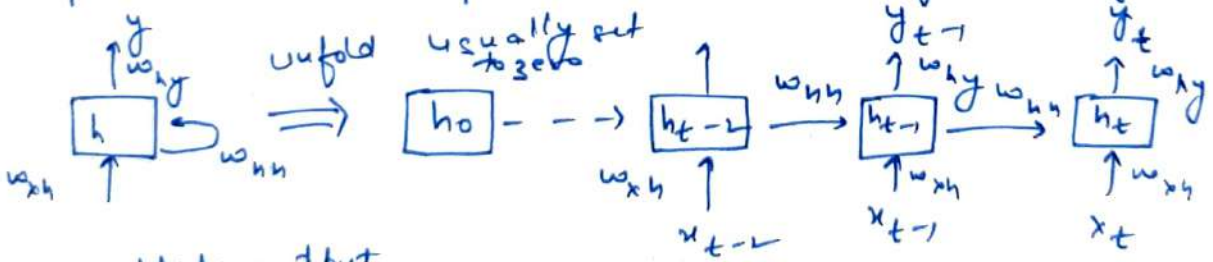


U, V, W :- weights, s, s_{t+1}, s_{t-1}, s_{t+1} :- hidden state

same weights used for every hidden layer ∴ it perform the same task on all the input of hidden layer to produce the output

No. of layer in RNN = No. of words in a sentence sequence

- output from previous step are fed as input to the current step.
- RNN used for sequential learning problem
- RNN save the output of a layer & feed this back to the input in order to predict the output of the layer



- multiple output
- single output

- there is a parameter sharing for

w_{hh} :- show relation b/w x_t & h_{t-1} is similar to relation ship b/w x_t & x_{t-1}

w_{xh} :- identical feature extraction from inputs

Parameter in RNN x_i, h_{i-1}, w_{xh}, w_{hh}, w_{hy}, b, y

Tasks for RNN :- 1. for each timestep of the input sequence & predict output y synchronously

2. for the input sequence and predict the scalar value of y (like end of the sequence)

3. for the input sequence x of length T_x generate output sequence y of different length T_y

- In RNN outputs are calculated not only by weights applied on input like a regular NN, but also by a "hidden state" vector representation the context based on prior, input/output
- In RNN, there are multiple copies of same network with same function and same parameters
- New / current hidden state h_t can be calculated as

$$h_t = f(h_{t-1}, x_t)$$

or $h_t = f(w_{hh} h_{t-1} + w_{xh} x_t + b)$

$h_t = \tanh(w_{hh} h_{t-1} + w_{xh} \cdot x_t + b)$

output vector at time step t calculated as

$$y_t = w_{hy} h_t$$

Training through RNN.

1. Words are first transformed to machine readable vector(x_t)
2. RNN process the sequence of vectors one by one (note: single time step of the input is supplied to the RNN (x_t is supplied to the RNN)
3. Calculated of hidden state h_t (using above formula)
4. The current h_t become h_{t-1} for the next time step.
5. We can go as many time steps the problem demands and combine the information from all previous state
6. finally RNN calculated output y_t (using above formula)
7. The out put is then compared to the actual output and the error is generated
8. The error is then back propagated to the network to update weights (using Backpropagation through time) and the network is trained.

Application of RNN :- 1. Sentiment classification (sequence classification) (2) video classification

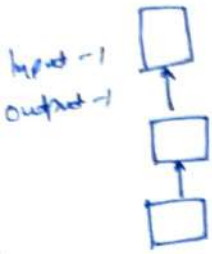
Sequence labelling Part of speech tagging (3) Name entity recognition

word prediction (1) Image captioning

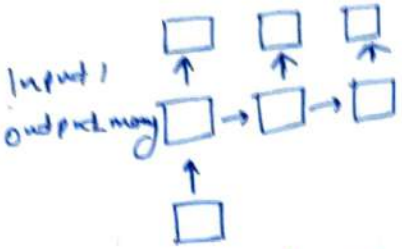
(2) Sequence generation eg machine translation

Various Architecture of RNN

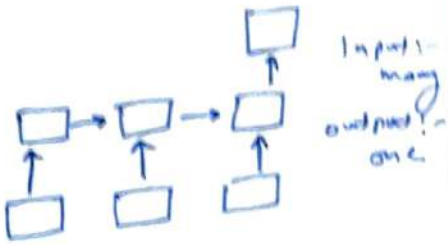
RNN has the flexibility for handling various type's of data



One to one
(Image classification)

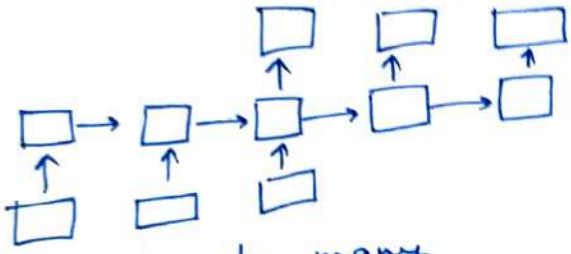


one to many
(Image captioning)

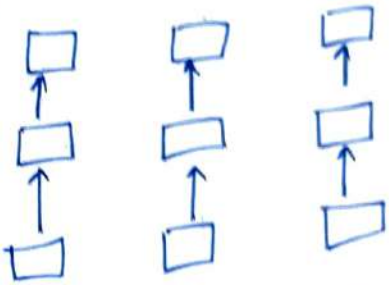


many to one
(sentiment analysis)

* Vanilla neural network



many to many
(machine translation)



many to many
(Labeling each frame of video)

Backpropagation through time (BPP(BPTT))

BPTT is the application of Backpropagation training algorithm to RNN applied to sequence of data like a time series

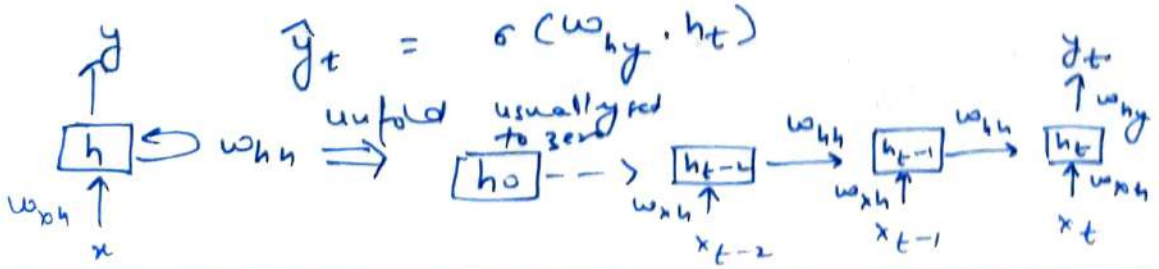
OR :- Backpropagation algorithm that works on sequence in time is called BPTT

The main goal of using this algorithm is to obtain the parameters that optimize the cost function.

Here formula used

$$h_t = \sigma(w_{hh} \cdot h_{t-1} + w_{xh} \cdot x_t + b)$$

$$\hat{y}_t = \sigma(w_{hy} \cdot h_t)$$



Let error is calculated using cross entropy loss function

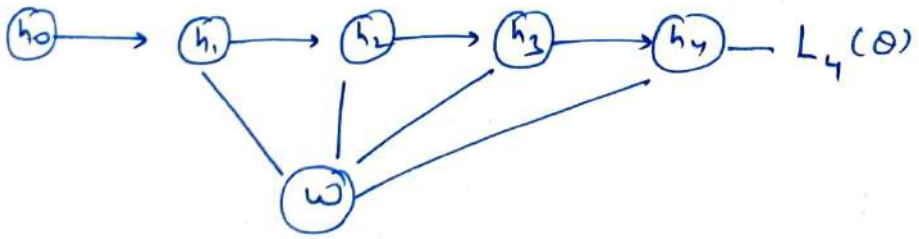
so overall loss of RNN

$$E(y, \hat{y}) = -\sum_t y_t \log \hat{y}_t$$

for each training epoch - start by training on shorter sequence and then train on progressively longer sequences until the length maximum sequence (1, 2, 3, ... n-1, n)

for each length of sequence unfold the network into a normal feed forward network that has k hidden layers.

- BPTT works by unrolling all input time stamp.
  Each time stamp has one input time stamp one copy of the network and one output
- Errors are then calculated and accumulated for each time ~~step~~. stamp.
- Backpropagate errors across the unfolded network and update the weights.



$$\frac{\partial L_4(\theta)}{\partial w} = \frac{\partial L_4(\theta)}{\partial h_4} \cdot \frac{\partial h_4}{\partial w}$$

$$\frac{\partial h_4}{\partial w} = \frac{\partial h_4}{\partial w} + \frac{\partial h_4}{\partial h_3} \cdot \frac{\partial h_3}{\partial w} + \frac{\partial h_4}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial w} + \frac{\partial h_4}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w}$$

$$\frac{\partial h_4}{\partial w} = \sum_{k=1}^{4} \frac{\partial h_4}{\partial h_w} \cdot \frac{\partial h_4}{\partial w}$$

$$\frac{\partial L_4(\theta)}{\partial w} = \frac{\partial L_4(\theta)}{\partial h_4} \sum_{k=1}^{4} \frac{\partial h_4}{\partial h_k} \cdot \frac{\partial h_k}{\partial w}$$

In general at time stamp 't'

$$\frac{\partial L_t(\theta)}{\partial w} = \frac{\partial L_t(\theta)}{\partial h_t} \cdot \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \cdot \frac{\partial h_k}{\partial w}$$

$$\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \times \frac{\partial h_{t-1}}{\partial h_{t-2}} \times \frac{\partial h_{t-2}}{\partial h_{t-3}} \cdots \frac{\partial h_{k+1}}{\partial h_k}$$

- high - Exploding gradient.
- low / less - vanishing gradient

~~Limitation of RNN~~ if we have long chain of gradient we can have two type of problems

(1) Vanishing gradients:- is going to be a product of many terms and if all these terms are very small then the gradient will vanish

(2) Exploding Gradients:- is going to be a product of many gradient terms and if all these terms are very large then the gradient will explode

Limitation of RNN:- (1) for deep RNN, RNN suffers by vanishing, exploding gradient problem

2. During back propagation RNNs suffer from vanishing gradient problem
   - The vanishing gradient problem is when the gradient shrink as it back propagates through time.
   - If the gradient value become extremely small, it doesn't contribute to much learning
   - It can't process very long sequence if using tanh or ReLu as an activation function
   - Training an RNN is very difficult task.

(2) RNN suffer from short term memory. If a sequence is long enough, RNN's may leave out important information from the beginning.

eg: If input is a long prop paragraph of text to do prediction

RNN can forget what it seen in longer sequences thus having a short term memory.

Need of long short term memory (LSTM)

LSTM is able to decide whether to keep the existing memory via the introduced gates.

RNN has two limitations:

1. Due to vanishing gradient problem, RNN's effectiveness is limited. When it needs to go back deep into the context
2. There is no finer control over which part of the context needs to be carried forward and how much of the part needs to be forgotten.

Gradient are values used to update a neural network weight $w' = w - \eta \frac{\partial L}{\partial w}$

$w' = 0.001$ (small change in weight values)

So layers that get a smaller gradient update steps learning (usually early layers)

As earlier layers don't learn, RNN's can forget what it seen in longer sequences thus having a short term memory

RNN:- Suffers from short term memory. If a sequence / sentence is long enough, it will not carry information from earlier time steps to later ones

For example:-

If we try to process a paragraph of text consist of many words, to do predictions using RNN's,

, it may leave out out important information from beginning eg The clouds are in the ——

Ans: sky

Like consider a language model trying to predict the next word based on the previous one

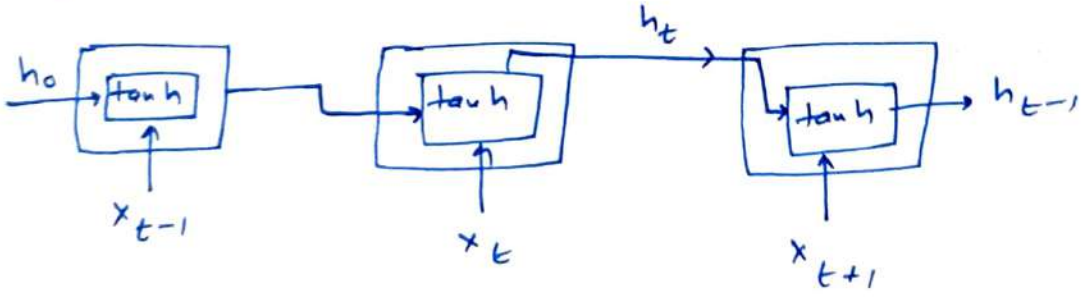
In such cases where the gap between the relevant information and the place that it is needed is small RNN can learn to use the past of information.

eg:- I grew in france..... I speaks fluent——

Ans :- french.

Here the gap between the relevant information and the point where it is needed to become very large. RNNs unable to predict

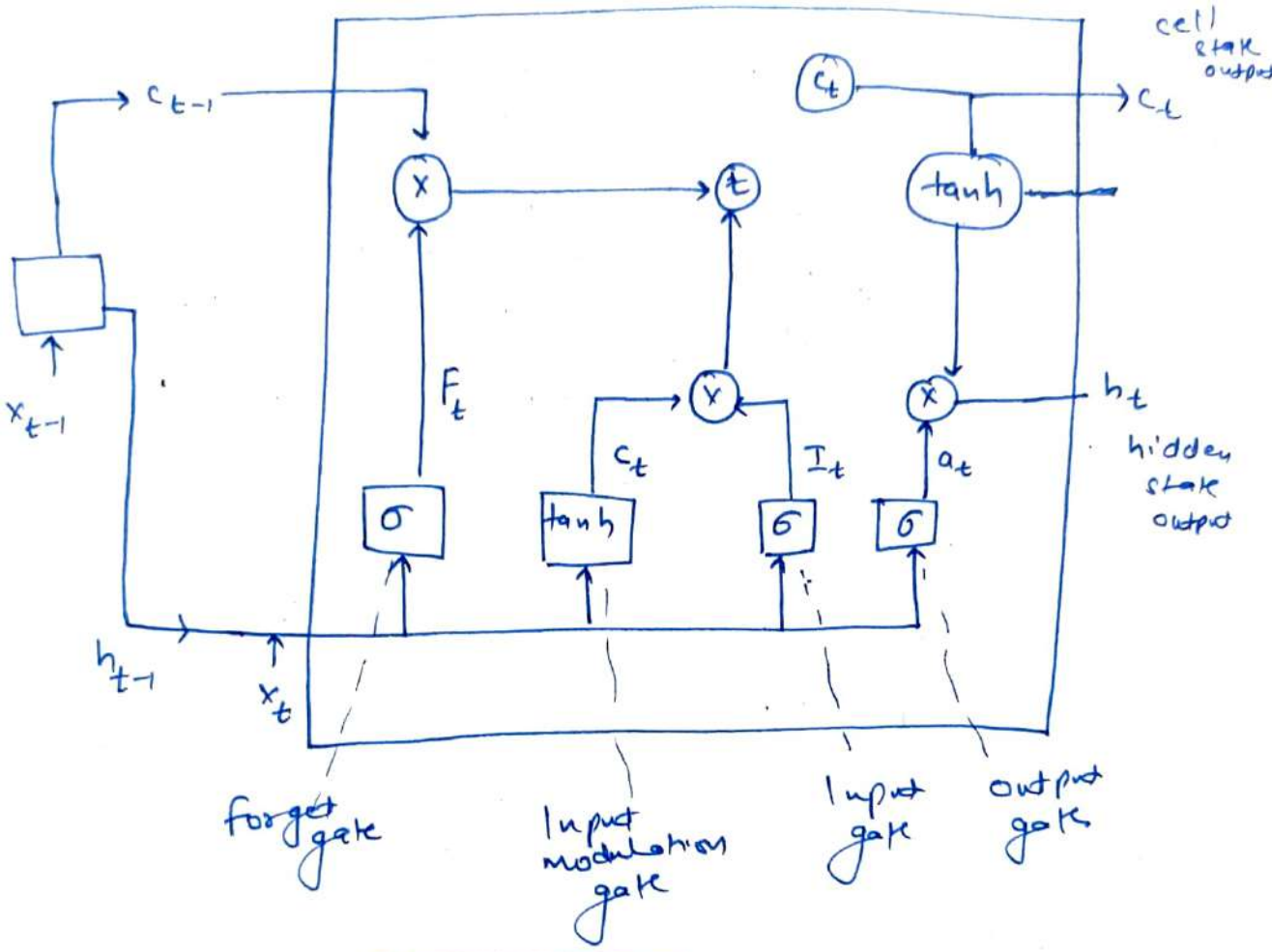
General introduction of LSTM LSTM networks are special kind of RNN, capable of learning long term dependencies



RNN.

The repeating module in a standard RNN contains a single layer (tanh)

LSTM contain four interacting layers in a repeating module and different memory blocks called cells



Terms used in LSTM

cell state :- Memory of LSTM network. It carry relevant information throughout the processing of the sequence. As the cell state goes on its journey information gets added / removed to the cell state via gates

Gates — Input gates
Forget gate
output gate

The gates are different neural networks that decide which information is allowed on the cell state

During training, the gates can learn what information is relevant to kept forged

Sigmoid activation function:- range [0, 1]. It is helpful to update or forget data because any number getting multiplied by 0 is zero, causing values to disappear or be forgotten. and if any number multiplied by 1 is the same value, therefore value stay the same or is kept

Element wise / pointwise product ⊙

$$[1\ 2] \odot [2\ 3]$$

$$[1\times 2 \quad 2\times 3] = [2\ 6]$$

working of LSTM Part 1

Step by step procedure of LSTM

step 1:- forget gate layer [What to keep and what to throw from old information]

In first step, it decided what information to throw away from the cell state and this decision is made by a sigmoid layer (or forget gate layer)

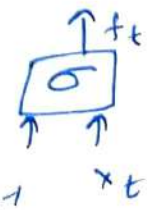
forget gate layer / sigmoid layer takes h_{t-1} and x_t as input and outputs a number between 0 & 1 for each number in the cell state

C_{t-1}

Here 1 represents - completly keep this information

0 represent - completly ignore / forget information from this state

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f$$



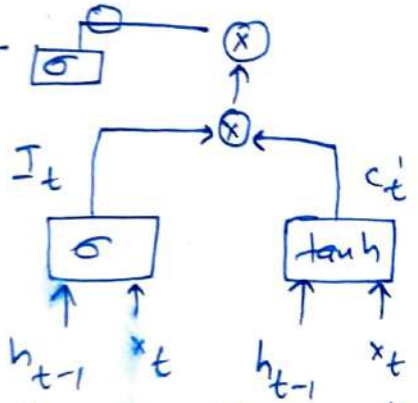
Step 2:- Input Modulation gate Layer and output gate Layer.

In second step, it is decided what new information to store in the cell state. or additional of useful information to the cell state is done by input gate

It has two parts

① A sigmoid layer (inputgate) decides which values to update layer

② Next a tanh layer (input modulation gate layer) creates a vector of new candidates values C_t' that could be added to the state. Next combine these two to create an update to the state



Sigmoid function filter the values to be remembered using input h_{t-1} and x_t

tanh function creates a vector that gives output from -1 to +1 which contains all possible values from h_{t-1} and x_t

Next combine these two to create an update to the state

$$I_t = \sigma(w_i(h_{t-1}, x_t) + b_i$$

$$C_t' = \tanh(w_c[h_{t-1}, x_t] + b_c)$$

Now update of old cell state C_{t-1} to the new cell state C_t

(i) Multiply c_{t-1} by f_t - forgetting the things we decided to forget earlier $(f_t \odot c_{t-1})$

(ii) $c_t = (f_t (t-1)) + (I_t \odot c_t')$ Memory cell update

$I_t * c_t'$ is the new candidate values, scaled by how much we decided to update each state value.

Working of LSTM part 2

step 3:- Calculation of output

In this step, we decide what will be the output of this hidden layer

This output will be based on cell state (filtered form)

Sigmoid layer (output gate layer)

It decides what part of the cell state goto output [The o/p gate decides what the next hidden state should be.]

first cell state pass through the tanh, it pushes the values between -1 and +1 and then multiply it by output of the sigmoid gate (output layer) so that we only output the parts we decided to.

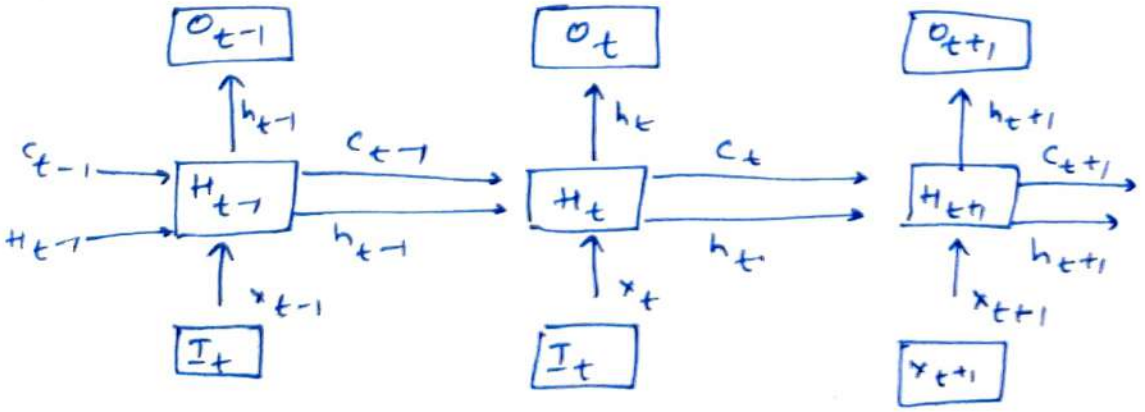
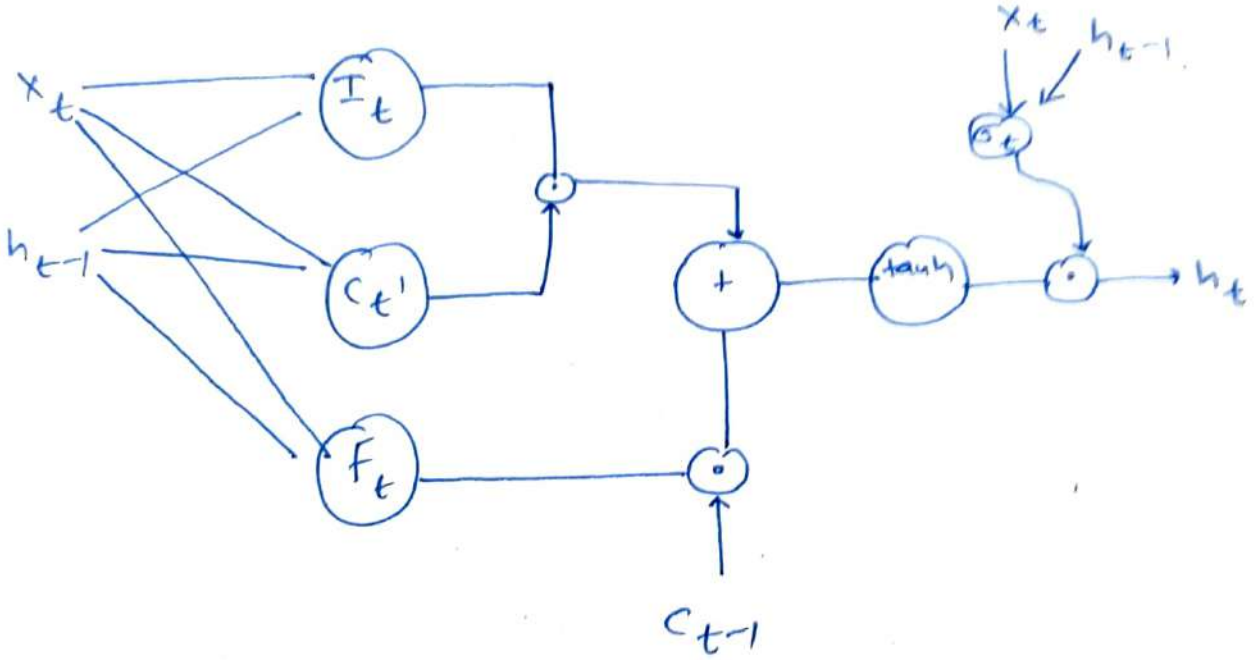


fig LSTM

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o)$$

Hidden Layer output $h_t = o_t \otimes \tanh(c_t)$



$C_t \rightarrow$ next state

h_t

Gated Recurrent ~~Neural~~ Unit (GRU)

RNN

GRU LSTM

Both solve the vanishing gradient problem of standard RNN

- Similar to LSTM, but there is no memory cell state it uses hidden state to transfer information
- It only has two gates
  - a reset gate
  - a update gate

Update gate

- The update gate acts similar to the forget and input gate of LSTM
- It decides what information to throw away and what new information to add.

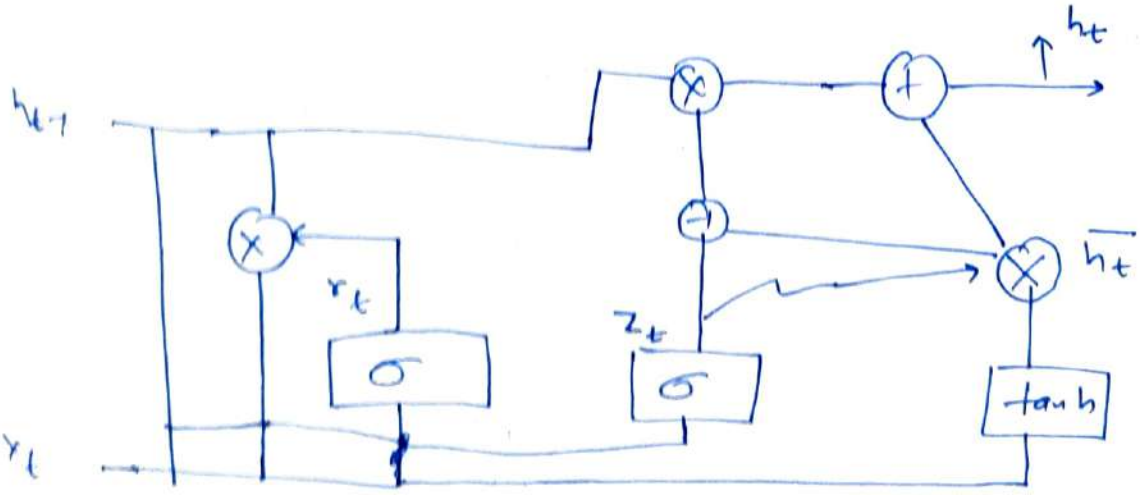
Reset gate:- It is used to decide how much past information to forget

Advantages of GRU

1) It is simple
2) faster
3) optimizes quicker (atleast on sentiments)

It is similar to LSTM is using gating functions, but differ from LSTM in that it doesn't have a memory cell / cell state and used the hidden state to transfer information.

- Each GRU unit consist of update gate and reset gate while LSTM consist of four (4) gates
- LSTM or GRU can learn to keep only relevant information to make prediction and forget non relevant data.



$$z_t = \sigma(w_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(w_r \cdot [h_{t-1}, x_t])$$

$$\bar{h}_t = \tanh(w \cdot [x_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \bar{h}_t$$

Machine translation translate a text from one natural language to another

english ⟶ hindi.

Sequence to sequence model are used for machine translation which are RNN based model.
- finds the most relevant word in sentence for a target language

Two type of sequence to sequence models

1. Encoder- Decoder model
2. ~~Attention~~ model.

Encoder-Decoder model for machine translation.

Given (English) Input sequence $x_1, x_2, \ldots x_m$

where x_i - words in input sequence

m = number of words in input sequence

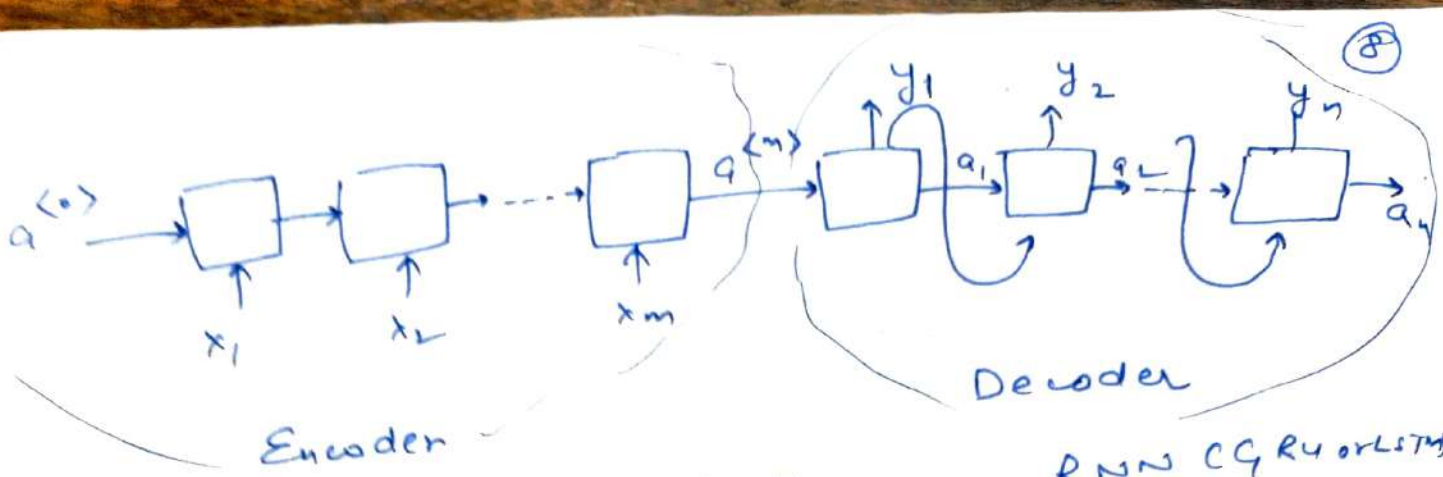
model have to find out / predict

(Hindi) output sequence - $y_1, y_2 \ldots y_n$

where y_i - words in output sequence

n = no. of word in output sequence

where m and n may be equal / different



The encoder network is built as an RNN (GRU or LSTM) where we feed the English words, one word at a time.

After processing the input sequence the output of the RNN is a vector that represent the input sequence

- After encoder, there is a decoder network which takes as input the encoding output produced by encoder network
- Decoder is then trained to output the translation one word at a time until it outputs the whole O/P sequence
- Decoder acts mainly as a Language model probabilistic model that are able to predict the next word in the sequence gives the word that preceede it) instead of having an input of all zeros here the input of decoder is an output vector of the encoder network

So machine translation model (like encoder-Decoder) can be thought of as conditional language model for a system that translates English to hindi That means instead of modeling the probability of any ~~sequence~~ sentence it is now shaping the likehood of the output (Hindi translation) conditioned on some input sentence (English text)

Due to the probabilistic behaviour of model there can be more than one the translation to input sentence English sentence

→Main aim → $\text{argmax} P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, \cdots y^{\langle T_y \rangle} | x)$

or

$\text{argmax} P(y_1, y_2, \cdots y_n | x)$

L most likely translation

Our Aim to find the most suitable translation

so model doesn't sample the outputs at random, but it find the Hindi sentence that maximize the conditional probability.

- To maximize the condtional probability or to find a most likely translation for a given input sentence the most common algorithm is used that is called Beam search algorithm

Beam search To explain beam search, let we are using a sequence to sequence model that was an encoder and decoder framework with LSTM or GRU as the basic block

Here encoder maps a source sequence encodes the source information (Let here English sentence) and the decoder takes the encoded data from encoder as input to produced an output sequence (Let here hindi sentence)

we always want the best and most likely words to be selected for the translation that matches the meaning of the English sentence

- There could be multiple likely translation of the source sentence (English) in the target language (Hindi).

So our goal is to pick the best and most likely translated word, so we choose the target word with the maximum probability based on the source sentence

- The Beam search algorithm selects multiple alternatives for an input sequence at each timestamp based on conditional probability

J - [illegible]

The number of multiple alternatives depends on a parameter called Beam width B

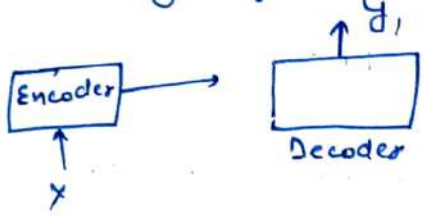
e.g eg B = 2

So at each time, step the beam search selects B number of best alternatives with the highest probability as the most likely possible choices for the timestep.

Example :- Let Beam width = 3

Vocabulary of Hindi word = 10,000 words

Step 1:-



(I have some money)

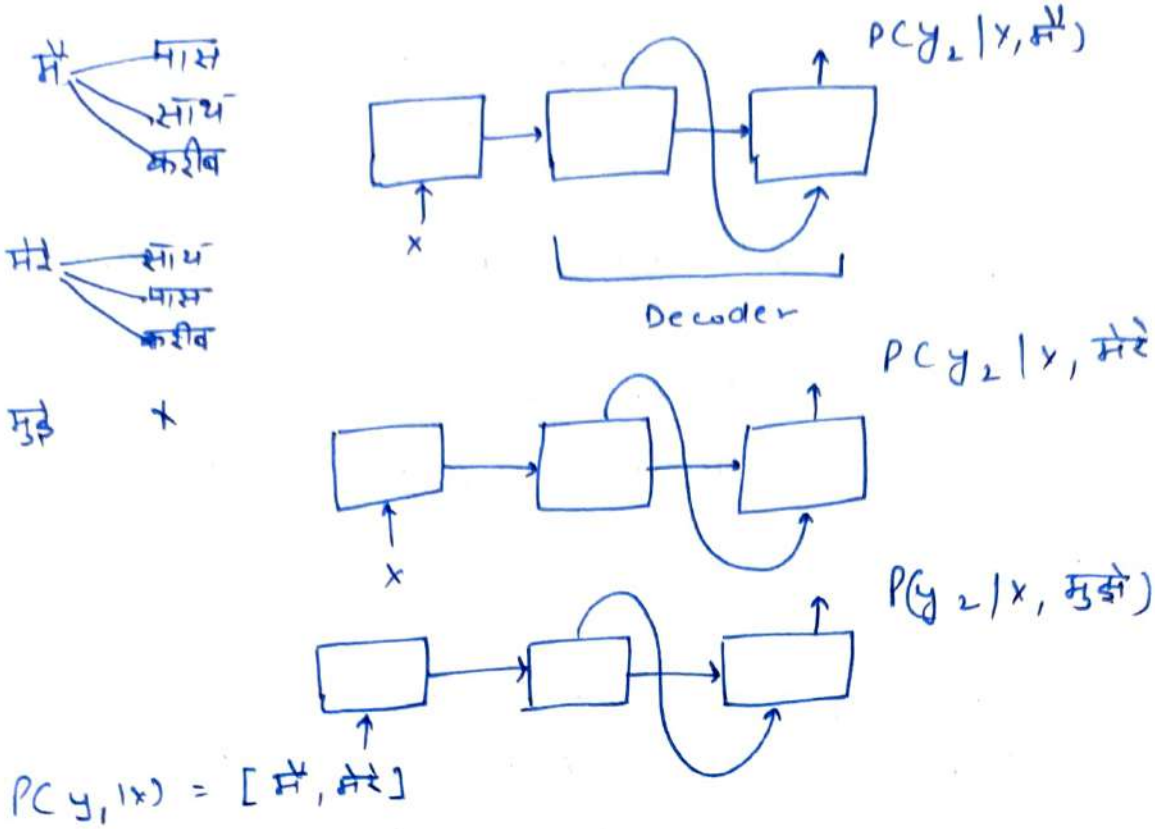
$P(y_1|x)$

= [मैं, मेरे, मुझे

Hindi vocabulary

step 1 finds the 3 words with the highest softmax probability based on the input sequence

Step 2 This step finds top 3 pairs of words for first and second word of the translated sequence (Hindi)



$P(y_1|x)$ = [मैं, मेरे]

$P(y_1, y_2|x)$ = [मैं पास, मेरे पास, मेरे साथ]

- The first three selected words (मैं, मेरे, मुझे) from step 1 is given as input to second step
- then softmax function is applied to all the 10,000 words in the hindi vocabulary to find the 3 best alternatives for the second word that are most likely to form a pair with first word conditional probability
- first, we will take word मैं and apply the softmax function to all 10,000 words in the vocabulary which calculated the probability with मैं (10000)

$10,000 + 10,000 + 10,000 = 30,000$

similarly probability for other two words, मेरे, मुझे is calculated using softmax functions ($10,000 + 10,000 = 20000$)

- Run 30,000 different combinations to check the top 3 pairs of the first and 2nd words.
- Let the top 3 first and second word pair combinations are मैं पास, मेरे पास, मेरे साथ

Let we did not find any word pair with मुझे as first word and second word having a high conditional probability so we dropped it.

Some process is repeated to find 3 best pair for the first, second and third word based on the input sequence and the chosen first and second word.

$P(y_1|x)$ = [मैं, मुझे, मुझे]

$P(y_1, y_2|x)$ = [मैं पास, मेरे पास, मेरे साथ]

$P(y_1, y_2, y_3|x)$ [मैं पास कुछ, मेरे पास कुछ]

This process continues and finally we get two output sentences with highest conditional probability and different lengths

I have some money — [मैं पास कुछ पैसे हैं (0.2)

मेरे पास कुछ पैसे हैं (0.8)

(input sentence)

- we finally pick the output of the decoder as the sentence with highest probability

I have some money — मेरे पास कुछ पैसे हैं।

So Beam search consider multiple best options based on beam width using conditional probability

- If we take higher beam width, it will give a better translation but would use a lot of memory and computational power

— So beam search consider multiple best options based on beam width using conditional probability

- If we take higher beam width, it will give a better translation but would use a lot of memory and computational power

for calculating conditional probability for 30000 words at each time step.

$B = 3$ 100000 words vocabulary

If we take lower beam width will result in more low quality translation but will be fast and efficient in terms of memory usage and computation power

BLEU score A method for Automatic evaluation of m/c translation

- BLEU is a score for comparing a candidate machine translation of text to one or more reference translations (human translations)
- It can also be used to evaluate text generated for a suite of NLP tasks (like generation, recognition, image caption generation, text summation, sequence recognition)
- BLEU is a metric for evaluating a generated sentence (often translation) to a reference sentence
- BLEU metric ranges from 0 to 1
  - rare score = 1.0 A perfect match result
  - score = 0.0 A perfect mismatch results
- Bleu score can be used to evaluate the performance of a MT system, particularly when there are multiple equally good answers
- Bleu score was developed for evaluating the prediction made by automatic machine translation system, but It is not perfect evaluation method.
- Benefit of Bleu score
  1. Quick and inexpensive to calculate
  2. Easy to understand
  3. Language independent
  4. It correlates highly with human evaluation
  5. Widely adopted method.

Machine translation evaluation system requires
(1) A numerical translation closeness metrics

(D) A good quality human reference translation

- BLEU score is based on "text string matches".
- It is a score that quantifies how good a machine translation is by computing a similarity score based on n-gram precision.

n-gram precision:- Look at each word in the candidate sentence and design it a score of 1 if it show up in any of the reference sentence otherwise 0

example: MT output- The cat the cat on the mat
(1) (2) (3)

Reference 1 :- There is a cat on the mat

Reference 2 :- The cat is on the mat

Unigram precision:-

candidate sentence	Unique Unigram	count / score
The cat the cat on the mat	the	3
	cat	2

Sequence of n consecutive words ← n-gram

Unigram one words sequence
bigram two words sequence
trigram three words sequence
4-gram four words sequence

- The main task of a BLEU implementor is to compare n-gram of the candidate with Two n-grams of the reference translation and count the number of matches
- Their number of matches are position-independent
- If there are no. of more no. of matches between candidates and reference translations, then better is the translation
- Higher the BLEU score better is the translation
- More reference human translations result in better and accurate score

Unigram precision

Candidate sentence

The cat the cat on the mat

1 2 3 4 5 6 7

number of unigrams in candidate sentence = 7

Unique Unigram	count / scan
the	3
cat	2
on	1
mat	1
	7

Total counts for the unique unigram in the candidate sentence

unigram precision = $\frac{7}{7} = 1.0$

Bigram precision :- candidate sentence - The cat the cat on the mat

the cat

Bigram: the cat, cat the, cat on, on the, the mat

1 2 3 4 5 = 6

Unique Bigram	count / scan
the cat	2
cat the	0
cat on	1
on the	1
the mat	1
Total count	05

Bigram precision = $\frac{5}{6}$ = 0.83

Drawback Unigram precision is very easy to be overconfident about the quality of MT

eg of poor MT output with high precision

candidate sentence the the the the the the the

unigram precision = $\frac{7}{7} = 1.0$

- To overcome the unigram precision problem, clipped count and modified precision were proposed

modified n-gram precision.

Unigram modified precision

Unique unigram	count	clipped count
the	3	2
cat	2	1
on	1	1
mat	1	1
	7	5

unigram modified precision $\frac{5}{7} = 0.714$

Unique Bigram	count	clipped count	
the cat	2	1	
cat the	0	0	$\frac{4}{6} = 0.667$
cat on	1	1	
on the	1	1	
the mat	1	1	
		4	

Unigram modified precision

unique unigram	count	clipped count
the	7	2
precision	$\frac{2}{7}$	

Problem in modified n gram precision

Bleu score tend to find favour short translation, which can produced very high precision scores, even using modified precision

Example :- candidate sentence :- the cat

Reference sentence :- the cat is on the mat

Modified unigram precision $= (1+1)/2 = 1$

solution is to Brevity penalty.

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \le r \end{cases}$$

r = length of reference sentence

c = length of candidate sentence

for above example

$c = 2$, $r = 6$

as $c < r$ $\quad BP = e^{(1-6/2)} = e^{(1-3)} = e^{-2}$

Bleu score

$$BP \times \exp\left[\frac{1}{N}\sum_{n=1}^{N} P_n\right]$$

Geometric mean of all n-gram precision

P_n - modified precision for n-gram

Bleu only utilizes a BP for shorter sentence

The redundant of layer sentences are not penalized properly

Attention Model

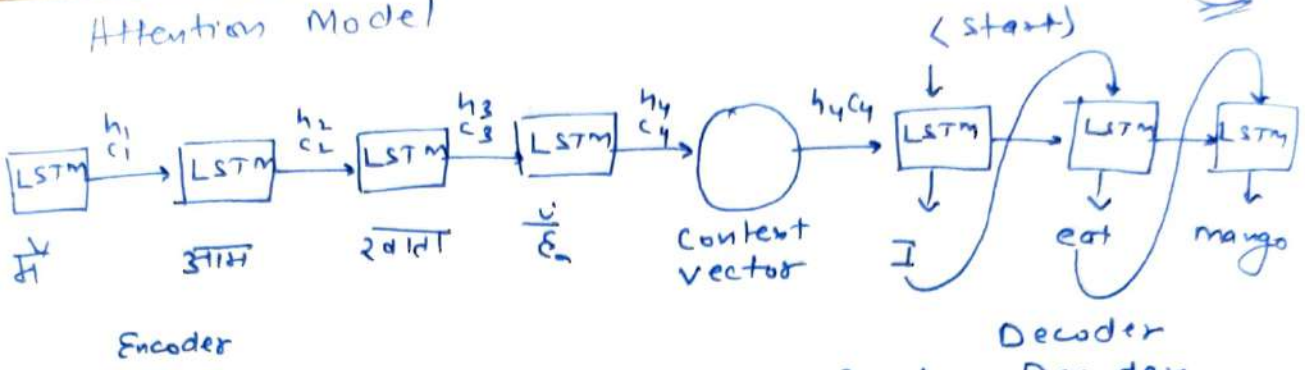


fig translation using Encoder - Decoder

- For long sentences (input) context vector may not have all necessary information
- To solve this problem, attention mechanism is used with Encoder - Decoder RNN / LSTM
context vector → $h_4 c_4$ contain good summary of मैं आम खाता है
↑ fixed size

Attention means directly your focus at something and taking greater notice or concentrating on one or a few things while ignoring others

- The attention mechanism is an improvement over the encoder - decoder based on neural based machine translation system in NLP.
- In encoder-decoder RNN / LSTM
  - Encoder process the entire input ~~seq~~ sentence and encode it into a context vector, which is the last hidden state of the LSTM / RNN.

मैं आम खाता है
I eat mango

t_1 मैं [1 0 0 0] I
t_2 खाता [0 0 0.5 0.5] eat
t_3 आम [0 1 0 0] mango

improvement

	मैं	आम	खाता	है
t_1	1	0	0	0
	α_{11}	α_{21}	α_{31}	α_{41}
t_2	0	0	0.5	0.5
	α_{12}	α_{22}	α_{32}	α_{42}
t_3	0	1	0	2
	α_{13}	α_{23}	α_{33}	α_{43}

α_{ij} i = word
j = timestamp.

It is expected that context vector contain good summary of input sequence. All intermediate states of the encoder are ignored and the final state is supposed to be the initial hidden state of the decoder

- Decoder unit produce the word in a sentence one after another

Drawback of Encoder-Decoder machine translation system:- If the encoder makes a bad summary, the translation will also be bad.

Usually encoder creates a bad summary when it tries to understand long sentences

- RNN cannot ~~remember too~~ remember long sentence and sequences due to vanishing/exploding gradient problem. It can remember the parts which it has just seen

So the performance of the encoder-decoder network degrades rapidly as the length of the input sentence increases

LSTM is supposed to be better than RNN, but it become forgetful in specific case

But in both RNN and LSTM there is no mechanism to give more importance to some of the input words compared to other while translating the sentence

So when model generates a sentence (machine translation) It searches for a set of position in the encoder hidden states where the most relevant information is available. This idea is called attention.

- Attention is proposed as a solution to the limitation of encoder-decoder model encode the input sequence to one fixed length vector from which to decode each output timestep. This issue is believed to be more of a problem when decoding long sequences.

The attention model allows an RNN to pay attention to specific part of the input that is considered as being important which improves the performance of the resulting model in practice

In attention model, instead of sending last hidden state output of decoder, we could take a weighted average of the corresponding word representation and feed to the decoders.

for example; at timestamp 2, we can take a weighted average of the representation खाली ई to get output 'eat' from the decoder. By doing this, we are not overloading the decoder with irrelevant information (like मैं मेरा)

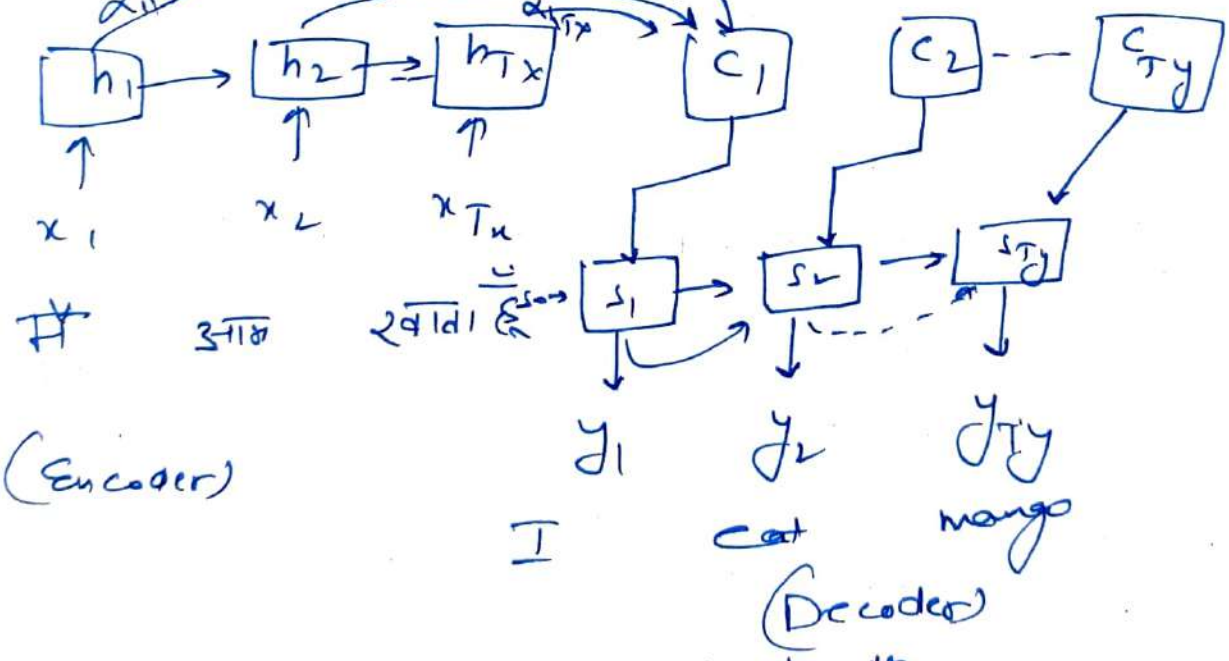


fig. ~~Attention~~ ~~mechanism~~ Encoder-decoder model.

Prediction of II word

$t=2$ $\quad c_2 = \alpha_{12} \cdot h_1 + \alpha_{22} h_2 + \alpha_{32} h_3 + \alpha_{42} h_4$

$\theta_{12} = a(s_1, h_1)$ $\quad \alpha_{12} = \dfrac{e^{\theta_{12}}}{e^{\theta_{12}} + e^{\theta_{22}} + e^{\theta_{32}} + e^{\theta_{42}}}$

$$e_{11} = a(s_1, h_1) \quad \alpha_{11} = \frac{e^{e_{11}}}{e^{e_{11}} + e^{e_{12}} + e^{e_{13}} + e^{e_{14}}}$$

$$e_{12} = a(s_1, h_2) \quad - \quad - \quad -$$

$$c_t = \sum_{j=1}^{T} \alpha_{jt} h_j$$

h_j - encoder hidden state at j^{th} timestamp

α_{jt} - Probability of focusing on the j^{th} words to produce the word

$$\alpha_{jt} = \frac{e^{e_{jt}}}{\sum_{k=1}^{T} e^{e_{kt}}}$$

$$e_{jt} = a(s_{t-1}, h_j)$$

↑ tanh

$$\alpha_{jt} = \frac{\exp(e_{jt})}{\sum_{k=1}^{T} \exp(e_{kt})}$$

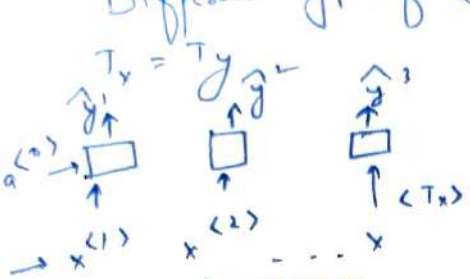
The importance of the j^{th} input word for decoding the t^{th} output word can be calculated by

$$e_{jt} = F_{ATT}(s_{t-1}, h_j) \quad \& \; F_{ATT} = \text{feed forward neural network}$$

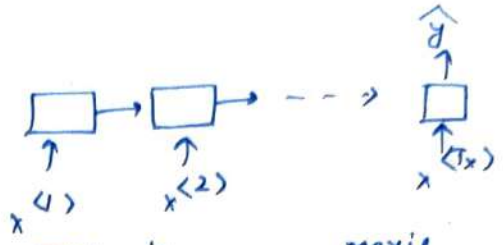
the alignment model scores :- how well each encoder input (h_j) matches the current output of the decoder (s_{t-1})

Different types of RNN

A.NG 1

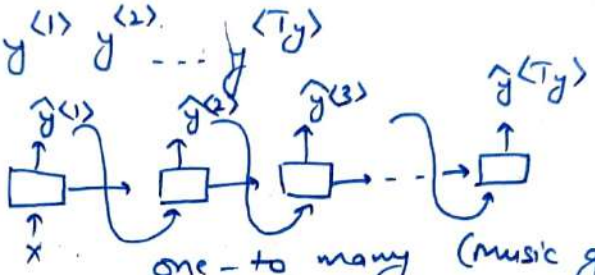


Many-to-many
eg: sentiment analysis
x = text
y = 0/1 ... 1 ... 5

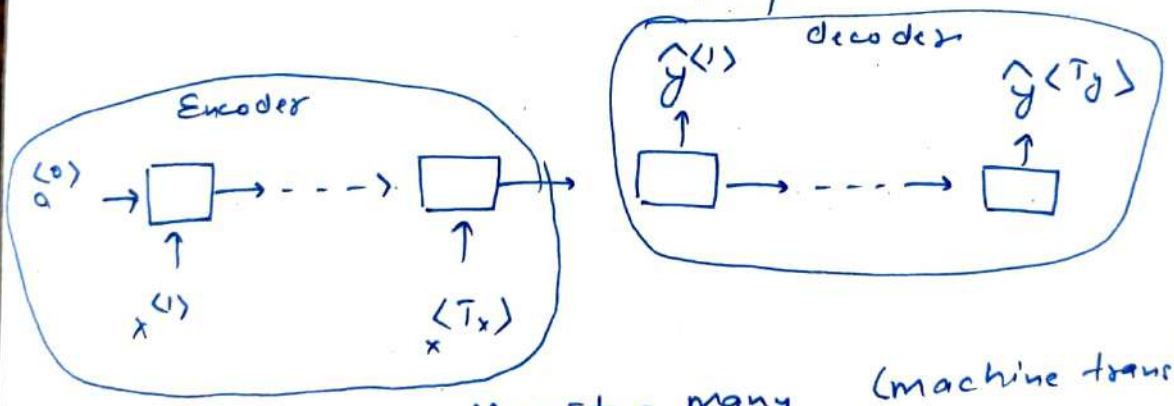


Many-to-one

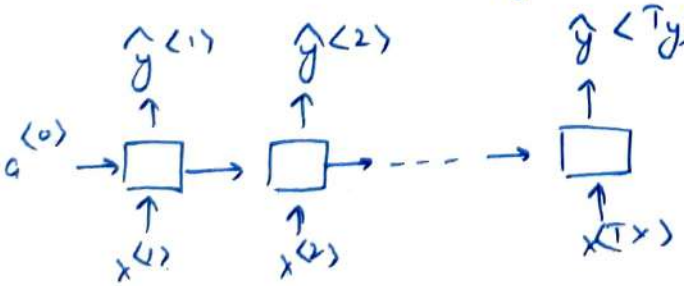
one-to-one



one-to-many (music generation)
$x = \phi$



Many-to-many (machine translation)



Language model and sequence generation

what is language modelling?

speech recognition

① — The apple and pair salad

② — The apple and pear salad

$P(1) = 3.2 \times 10^{-13}$

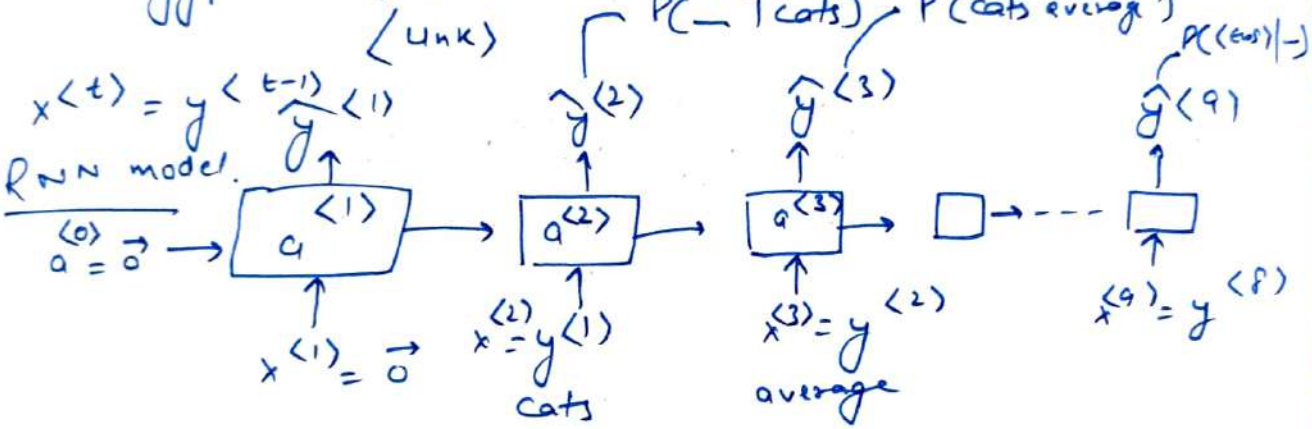
$P(2) = 5.7 \times 10^{-10}$

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

Training set: Large corpus of english text

Cats average 15 hours of sleep a day. <EOS>
$y^{\langle 1\rangle}$ $y^{\langle 2\rangle}$ $y^{\langle 3\rangle}$ $y^{\langle 4\rangle}$ ..... $y^{\langle 9\rangle}$
Tokenize end of sentence

The Egyptian Mau is a bread of cat <EOS>
<Unk>



P(a) P(aaron) .. P(cats) .. P(zulu) .. P(UNK) .. P(<EOS>)

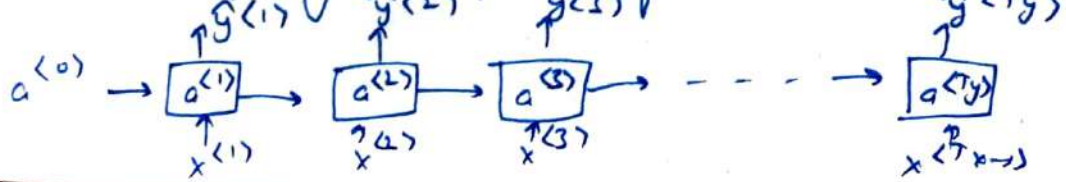
$$L(\hat{y}^{\langle t\rangle}, y^{\langle t\rangle}) = -\sum_i y_i^{\langle t\rangle} \log \hat{y}_i^{\langle t\rangle}$$

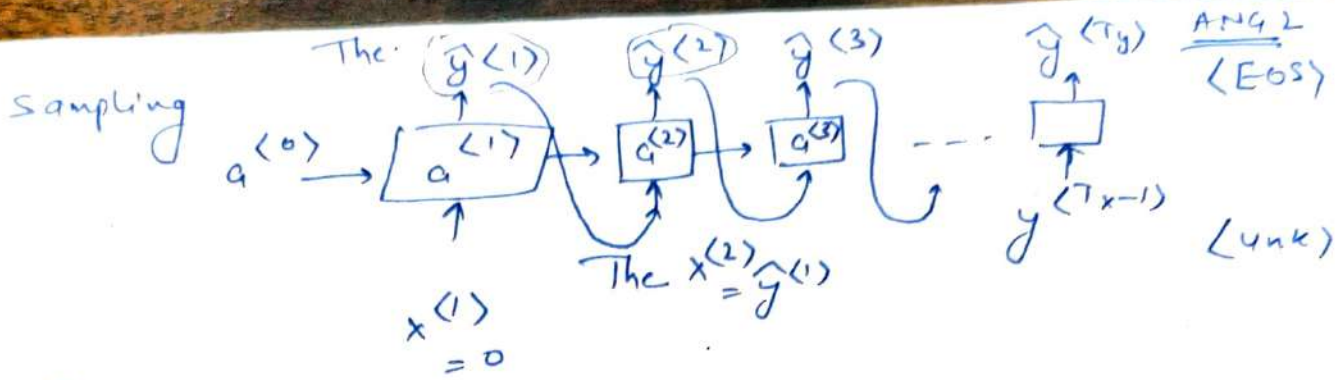
$$L = \sum_t L^{\langle t\rangle}(\hat{y}^{\langle t\rangle}, y^{\langle t\rangle})$$

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

Sampling novel sequence

Sampling a sequence from a trained RNN





P(a) P(aaron) -- P(Zulu) P(<unk>) np.random.choice

P(— | the)

Vocabulary = [a, b, c, .. z — A, .. Z]

$y^{\langle 1 \rangle}, y^{\langle 2 \rangle}, y^{\langle 3 \rangle}, y^{\langle 4 \rangle}$ (cat average)

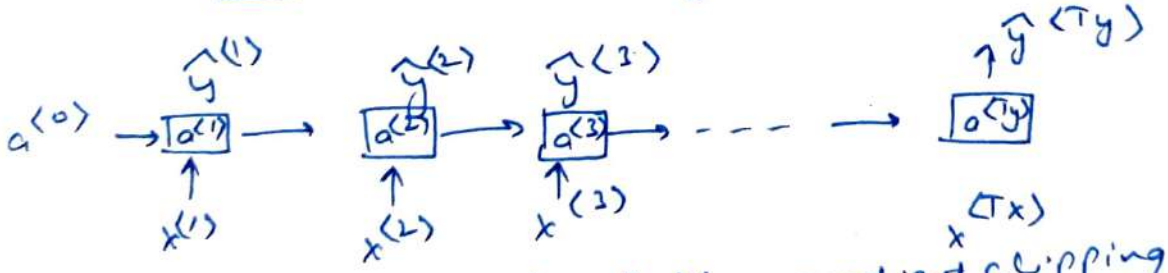
c a t space

Sequence generation

Vanishing gradient with RNNs

The [cat], which already ate ---, [was] full

The [cats], which already ate ---, [were] full

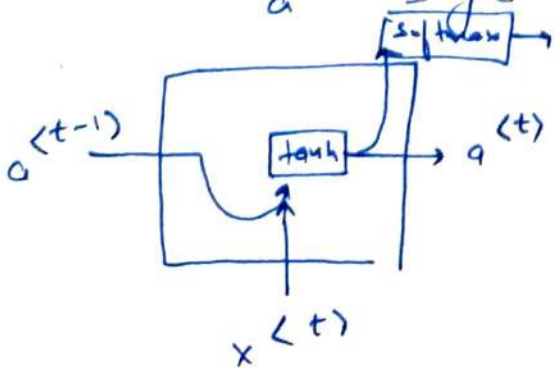


NaN → gradient clipping

Gated Recurrent Unit (GRU)

RNN unit

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



RNN (simplified)

c = memory cell

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

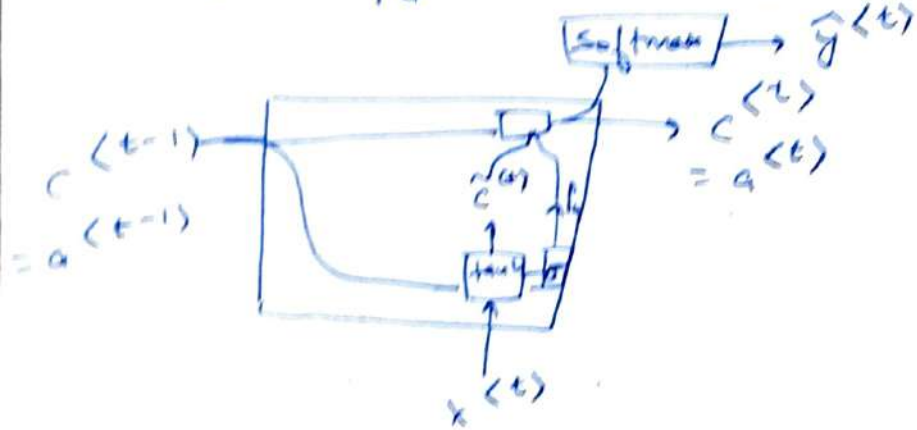
$\Gamma_u = \sigma(w_u [c^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$

$\Gamma_u^{\langle t \rangle} = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 1$ $\Gamma_u = 0$

→ $c^{\langle t \rangle} = 1$

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

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



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

LSTM Long short term memory

GRU

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

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

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

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

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

L = 1:Ty

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

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

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

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

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

[Peephole connection $c^{\langle t-1 \rangle}$

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

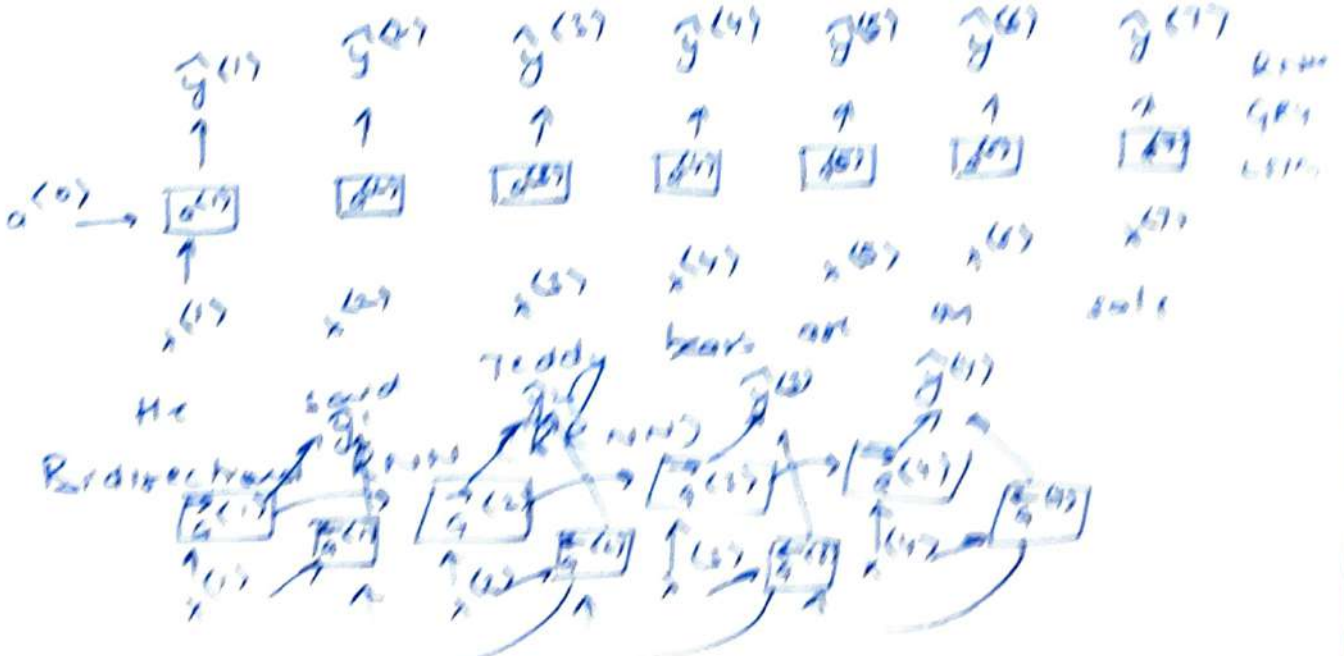
$$a^{\langle t \rangle} = \Gamma_o * \tanh c^{\langle t \rangle}$$

Bride Bidirectional RNNs

Getting information from the future.

He said, "Teddy bears are on sale!"

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



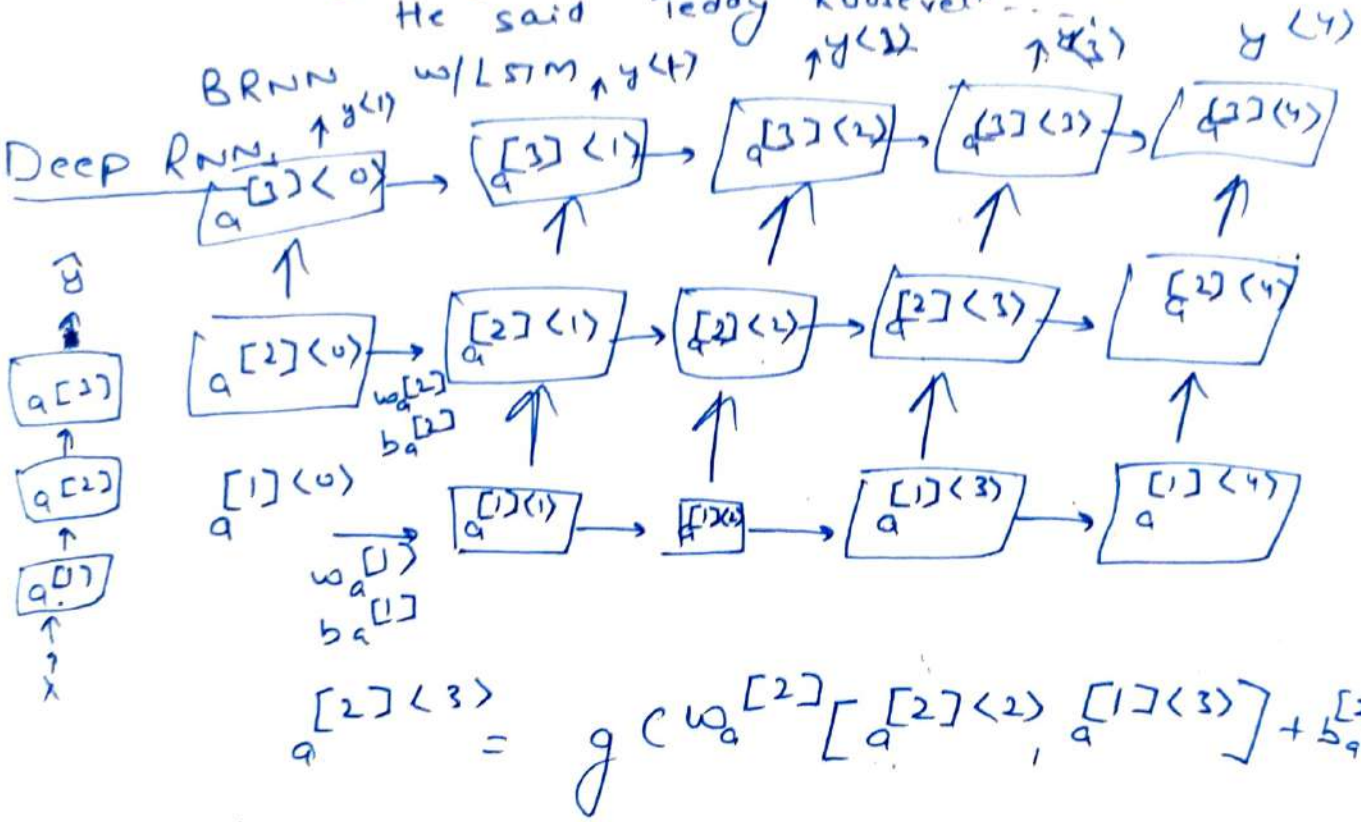
Acyclic graph

$$\hat{y}^{\langle t \rangle} = g\left(W_y\left[\overrightarrow{a}^{\langle t \rangle}, \overleftarrow{a}^{\langle t \rangle}\right] + b_y\right)$$

He said "Teddy Roosevelt ..."

BRNN w/ LSTM

Deep RNN

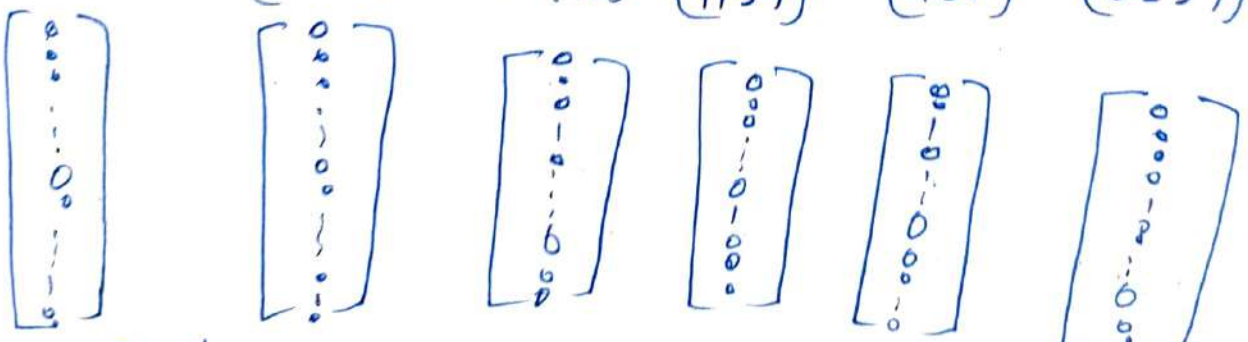


$$a^{[2]\langle 3 \rangle} = g\left(W_a^{[2]}\left[a^{[2]\langle 2 \rangle}, a^{[1]\langle 3 \rangle}\right] + b_a^{[2]}\right)$$

Word Representation

1- hot representation

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



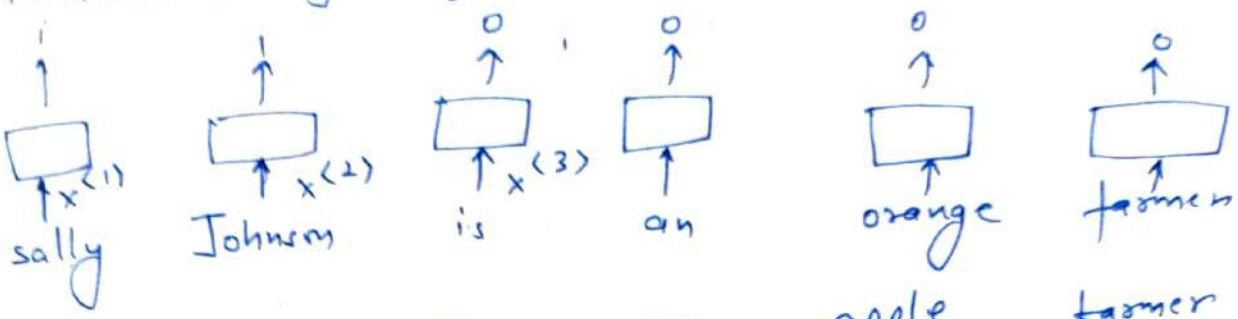
featurized representations : word embedding

	Man	woman	King	Queen	Apple	Orange
Gender	-1	1	0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.7	0.69	0.03	-0.02
Food	0.04	0.01	0.02	0.01	0.95	0.97

Visualizing word embeddings

Using word embedding

Named entity recognition example



Robert Lin i's an apple farmer

→ 1B words — 100B words

→ 100k words

Transfer learning and word embeddings

(A) 1. Learn word embedding from large text corpus (1–100B words)
(or download pretrained embedding online.)

(B) 2. Transfer embedding to new task with smaller training set (say 100k words)

3. optional: continue to finetune the word embeddings with new data.

Relation to face encoding

Properties of word embedding

Man → woman

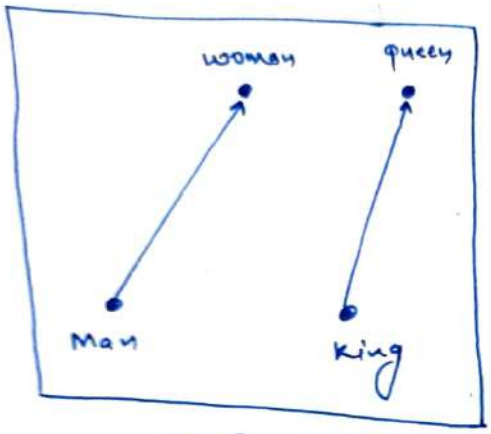
$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

King → ? (queen)

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

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

↑ queen



300 D

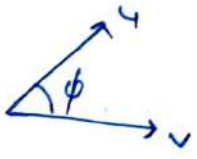
$e_{man} - e_{woman} \approx e_{king} - e_?$ (e_w)

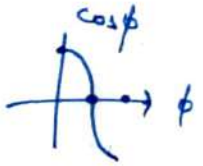
$\underset{w}{\text{argmax}}\ sim(e_w, e_{king} - e_{man} + e_{woman})$

(30 - 75 %)

Cosine similarity

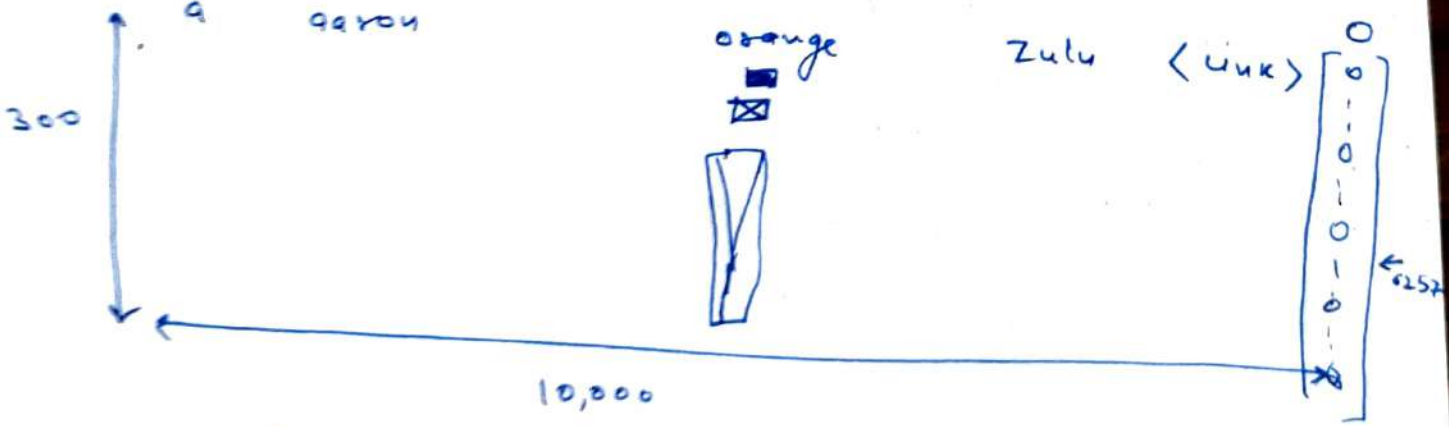
$$sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$





$\|u - v\|^2$

Embedding matrix



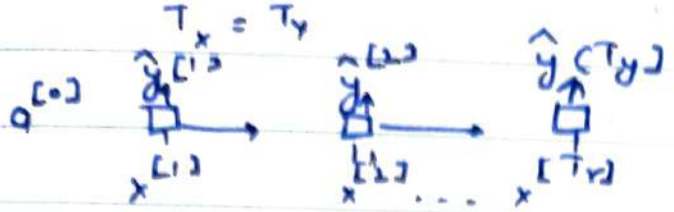
$E \cdot O_{6257} = e_{6257}$ e_w = embedding for w

(300, 10K) (10K, 1) (300, 1)

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

Recurrent Neural network (RNN)

Different types of RNN



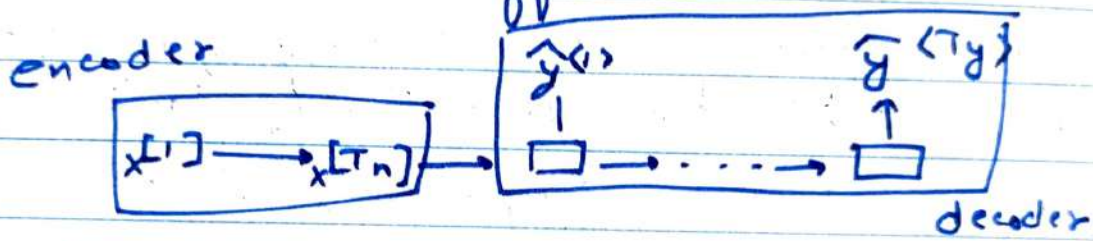
Many - to - many

Many - to - one eg Movie review

One - to - one

One - to - many e.g Music generation

Many - to - many Input and output line is different



→ Language Model and sequence operation

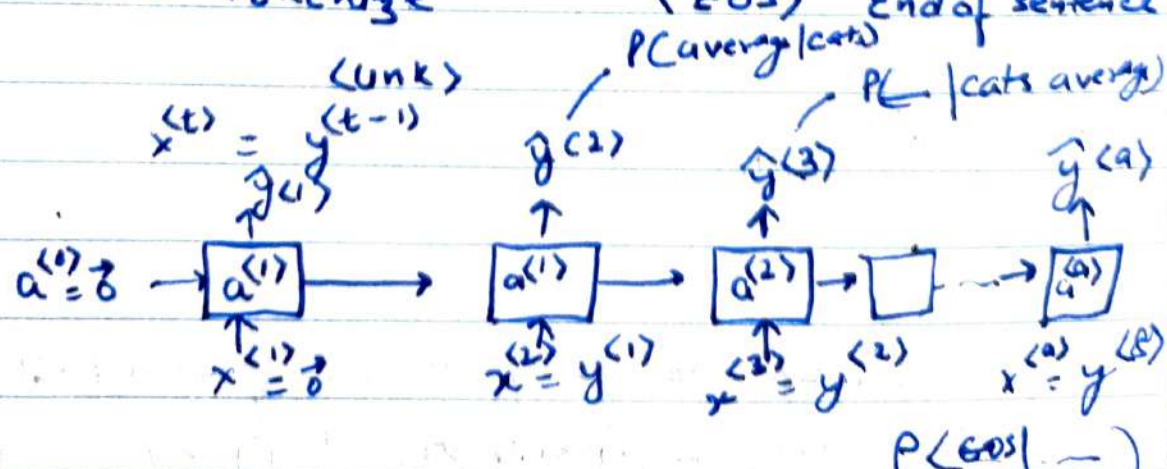
What is language modelling?

Speech recognition

Language modelling with an RNN

Training set

- Tokenize



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

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

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

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

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

→ Sampling novel sequences

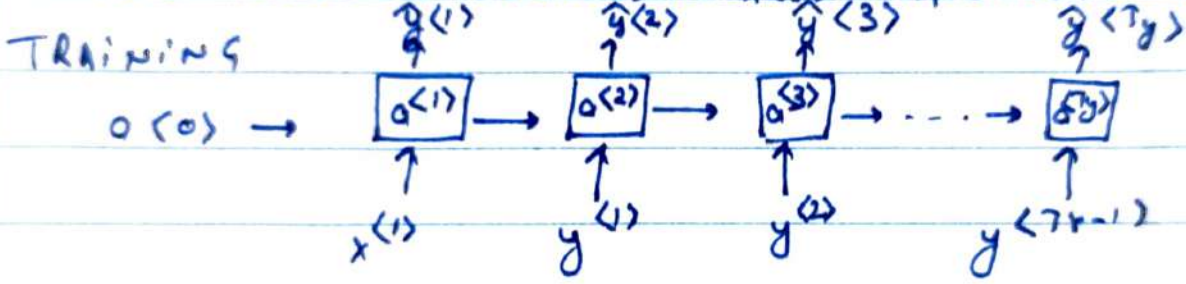
Sampling a sequence from a trained RNN

Training

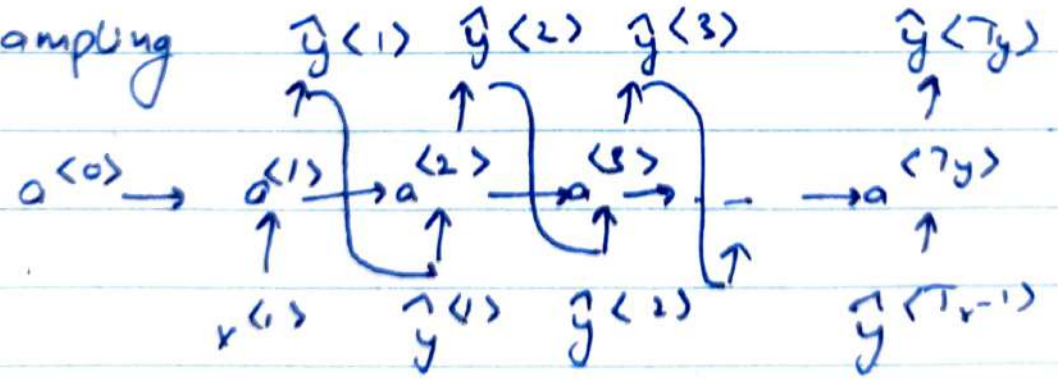
Sampling

P(a) P(aaron) ... P(Zulu) P(UNK) np.random.choice

TRAINING



Sampling



→ Vanishing gradients with RNN

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

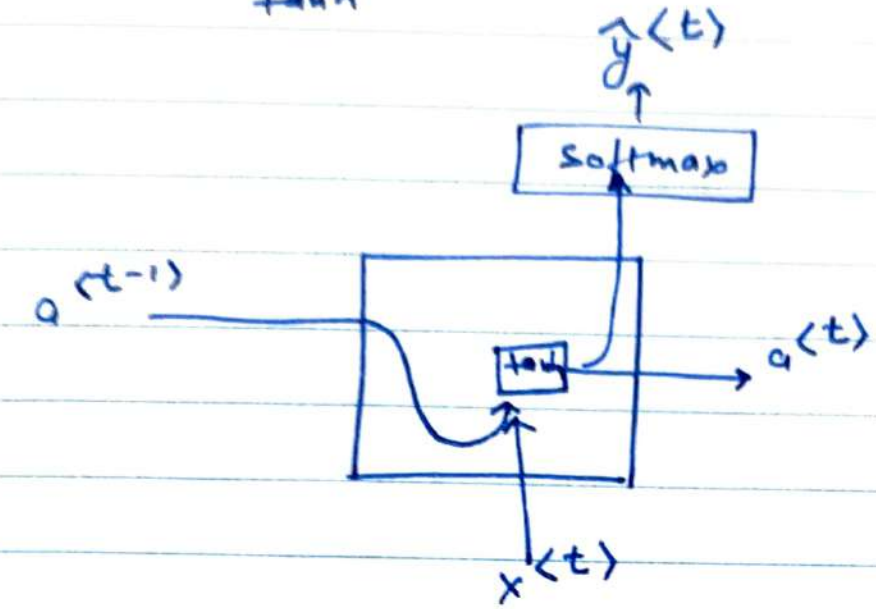
The cats, which already ate ..., were full

Nan gradient clipping

→ Gated Recurrent Unit GRU

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

↑ tanh



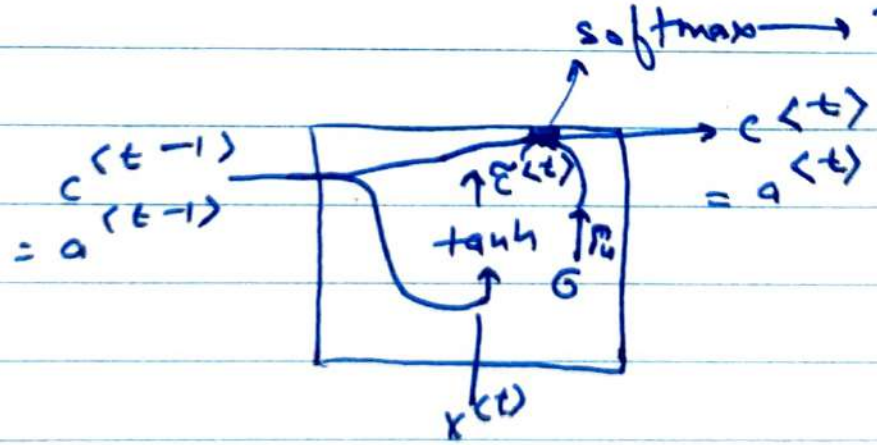
c = memory cell

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

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

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

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



→ LSTM (Long short term memory) unit

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

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

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

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

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

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

(peephole connection)

→ Biderectional RNN (BRNN)

Getting information from the future.

He said "Teddy bears are on sale"!

He said "Teddy Roosevelt was a great President!

→ Deep RNNs

→ Word Representation

Featurized representation: word embedding

Visualizing word embeddings

→ Using word embeddings

Relation to face encoding (embedding)

→ Properties of word embeddings.

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

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

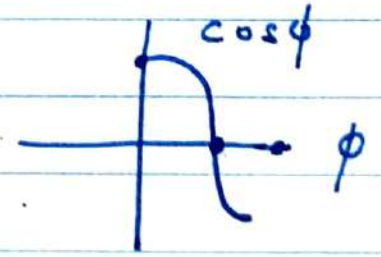
$$e_{man} - e_{woman} \approx e_{king} - e_{?queen}$$

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

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

cosine similarity

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



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

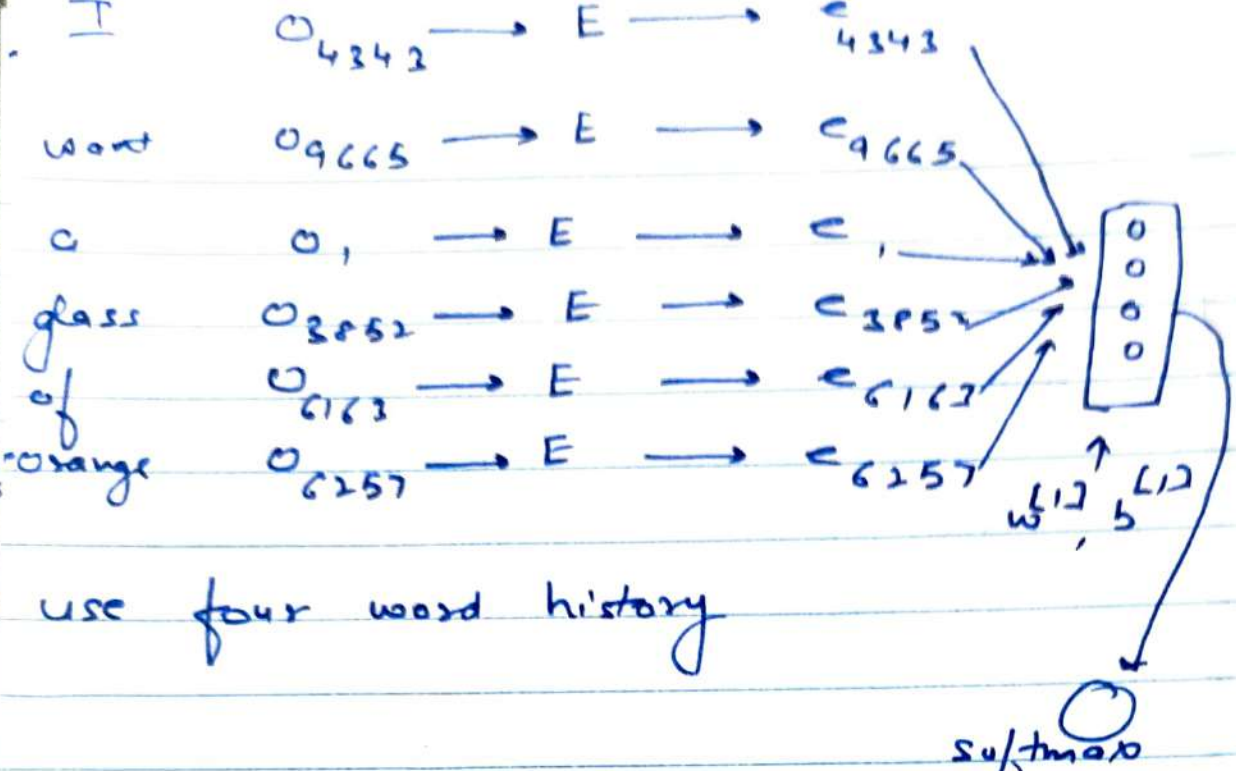
→ Embedding matrix

→ Learning word embedding

neural language model

I want a glass of orange ?

4343 9665 1 3852 6163 6257



use four word history

Bengio et al., 2003, A neural probabilistic Language model.

I want a glass of orange juice to go along
context target context

with my cereal

4 word on left & right

Last 1 word orange ?

Near by 1 word glass ?

→ Word2Vec

Skip-grams

Model

Vocab size = 10,000 K

$O_c \rightarrow E \rightarrow e_c \rightarrow O \rightarrow \hat{y}$

$e_c = E o_c$ softmax

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

θ_t = parameter associated with output t

$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10000} y_i \log \hat{y}_i$

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

Problems with softmax classification

hierachial

$\log |V|$

→ Negative sampling

Defining a new learning problem

context word target

10,000 binary classification K+1

Selecting negative examples

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

→ GloVe word vectors

global vector for word representation

$X_{ij} = X_{ji}$ ←

Pennington et. al., 2014. GloVe vector for word representation.

Model

Minimize $\sum_{i=1}^{10,000}\sum_{j=1}^{10,000} f(x_{ij})(\theta_i^T e_j + b_i + b_j - \log x_{ij})^2$

weighting term: $f(x_{ij})$ — $\theta_t^T e_c$ — $\log x_{ij}$

$f(x_{ij}) = 0$ if $x_{ij} = 0$ "$0 \log 0$" $= 0$

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

A note on the featurization view of word embeddings

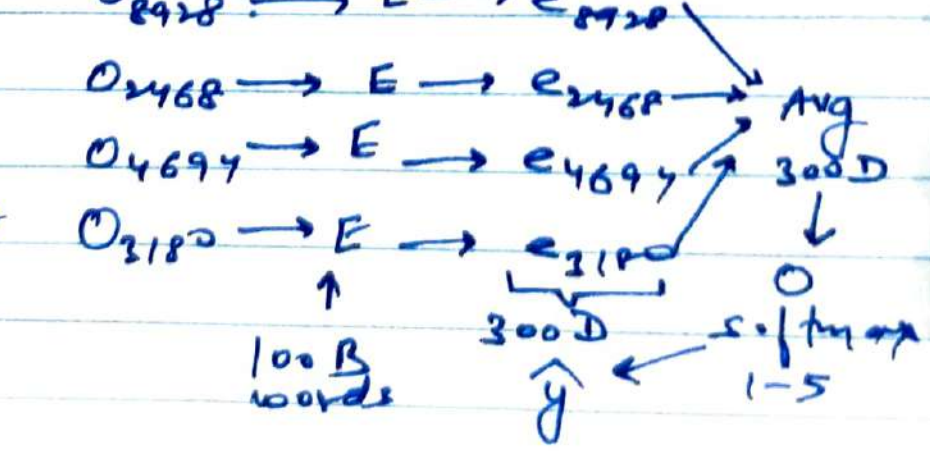
$$\text{minimize} \sum_{i=1}^{10,000}\sum_{i=1}^{10,000} f(x_{ij})(\theta_i^T e_j + b_i + b_j' - \log x_{ij})^2$$

→ Sentimental classification

Simple sentiment classification model

The dessert is excellent

8928 2468 4694 3180



→ Debiasing word embeddings

Bolukbasi et al 2016 Man is to computer programmer as woman is to homemaker

? Debiasing word embeddings

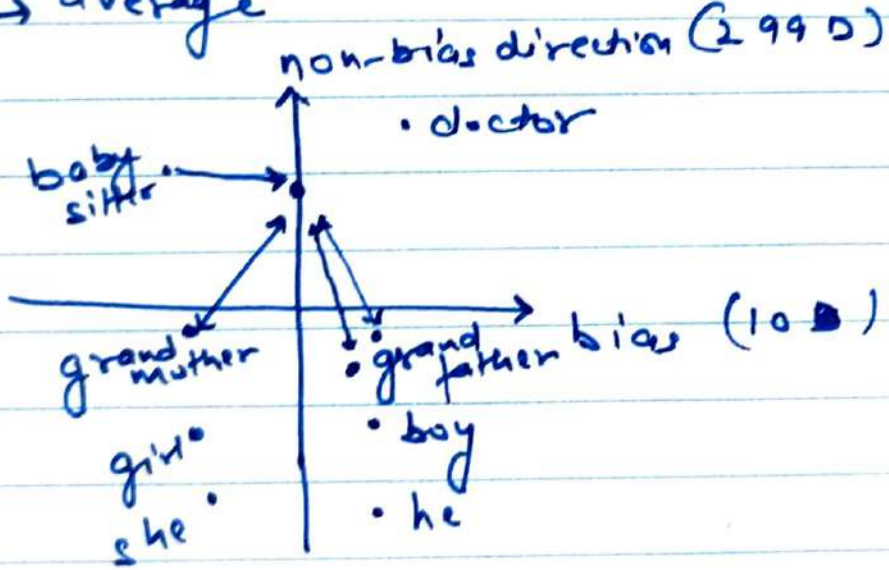
Addressing bias in word embeddings

1. Identify bias directions

$e_{he} - e_{she}$

$e_{male} - e_{female}$

→ average



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

3. Equalize pairs

grandmother — grandfather

girl — boy

Bolukbasi et al., 2016

→ Basic models

Sequence to sequence model

$x^{\langle 1 \rangle}$ $x^{\langle 2 \rangle}$ $x^{\langle 3 \rangle}$ $x^{\langle 4 \rangle}$ $x^{\langle 5 \rangle}$

Jane visite l'Afrique en septembre

Jane is visiting Africa in september

$y^{\langle 1 \rangle}$ $y^{\langle 2 \rangle}$ $y^{\langle 3 \rangle}$ $y^{\langle 4 \rangle}$ $y^{\langle 5 \rangle}$ $y^{\langle 6 \rangle}$

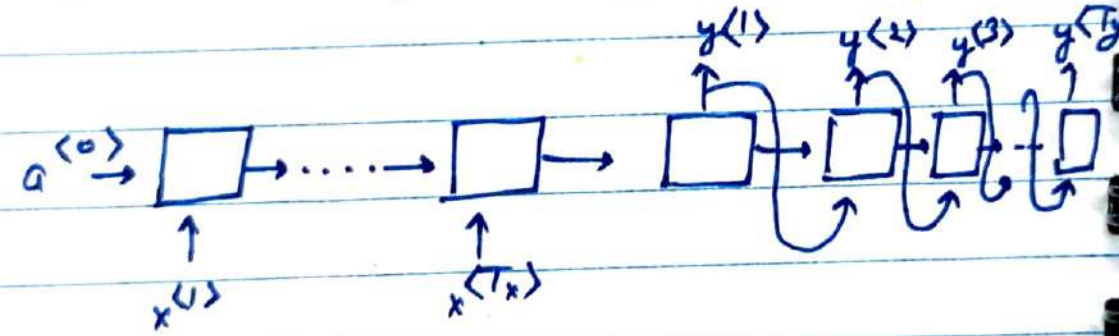


Image captioning

A cat sitting on a chair

→ Picking the most likely sentence

Machine translation as building a conditional language model

Finding the most likely translation

$P(y^{\langle 1 \rangle}, \dots y^{\langle T_y \rangle} | x)$

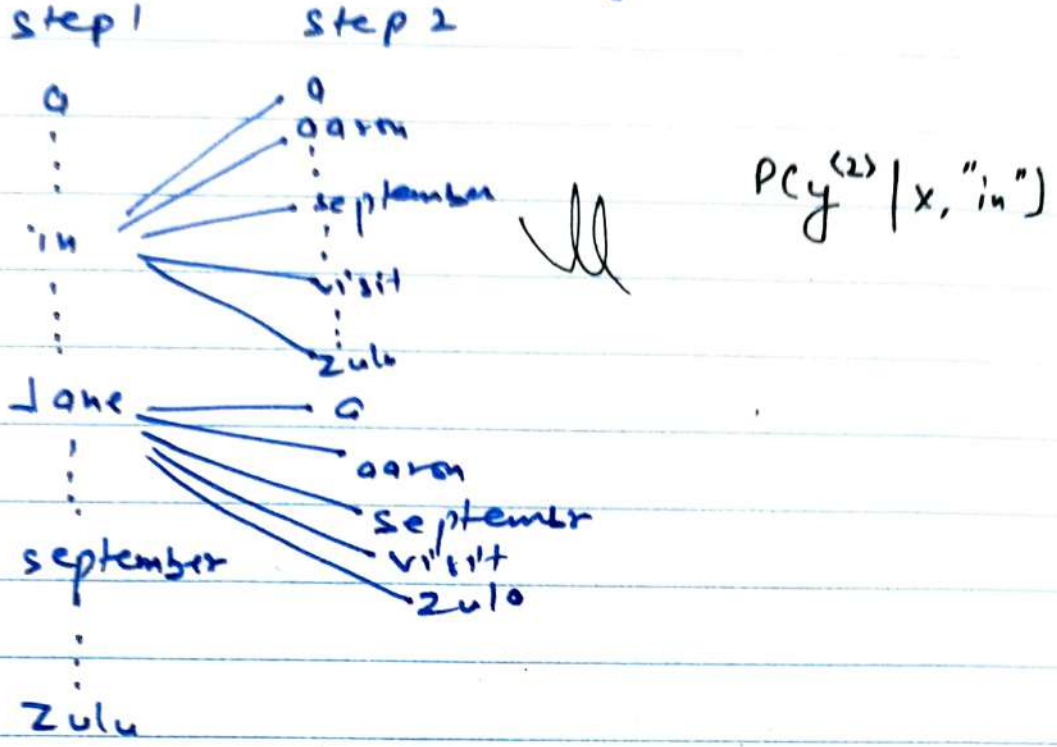
$\arg\max_{y^{\langle 1 \rangle}, \dots y^{\langle T_y \rangle}} P(y^{\langle 1 \rangle}, \dots y^{\langle T_y \rangle} | x)$

why not a greed search?

→ (Basic search) Beam search

beam width

$P(y^{\langle 1 \rangle} | x)$



Beam search ($B = 3$) (Possibly 3 at a time)

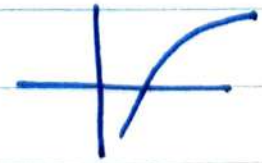
$P(y^{\langle 1 \rangle}, y^{\langle 2 \rangle} | x)$

→ Refinements to beam search

Length normalization

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

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



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

$P(y|x) \leftarrow$

$$\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{\langle t \rangle} | x, y^{\langle 1 \rangle}, \ldots y^{\langle t-1 \rangle})$$

$\alpha = 0.7$ $\alpha = 1$

$\alpha = 0$

Beam width B?

Large B better result, slower

small B worse result, faster

1, 3 → 10, 100, 1000 → 3000

Unlike exact search algorithm like BFS or DFS, Beam search run faster but is not guaranteed to find exact max. for $\arg\max_y P(y|x)$.

→ Error analysis in Beam search

— Bleu score

Papineni et al 2002, A method for automatic evaluation of machine translation

Bilingual evaluation understanding

Precision Modified precision

Blue Bleu score on bigrams

Bleu score on unigrams

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

$P_1, P_2, = 1.0$

Bleu details

P_n = Bleu score on n-grams only

combined Bleu score

$$BP \exp\left(\frac{1}{\psi}\sum_{n=1}^{y} P_n\right)$$

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

→ Attention model intution

The problem of long sequences

Bahdanau et al 2014 Neural machine translation by jointly learning to align and translate

→ Attention model

$$c^{\langle 1 \rangle} = \sum_{t'} \alpha^{\langle 1, t' \rangle} a^{\langle t' \rangle}$$

$$a^{\langle t' \rangle} = (\vec{a}^{\langle t' \rangle}, \overleftarrow{a}^{\langle t' \rangle})$$

computing attention $\alpha^{\langle t t' \rangle}$

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

$$\alpha^{\langle t, t' \rangle} = \frac{\exp e^{\langle t, t' \rangle}}{\sum_{t'=1}^{T_x} \exp e^{\langle t, t' \rangle}}$$

Xu et.al., 2015, show attend and tell: neural image caption generation with visual attention

~~Attention~~ examples

→ speech recognition

x
audio clip

y
transcript

"The quick brown fox"

CTC cost for speech recognition (Connectionist temporal classification)

Graves et al 2006 Connectionist temporal classification labeling unsegmented sequence data with RNN

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