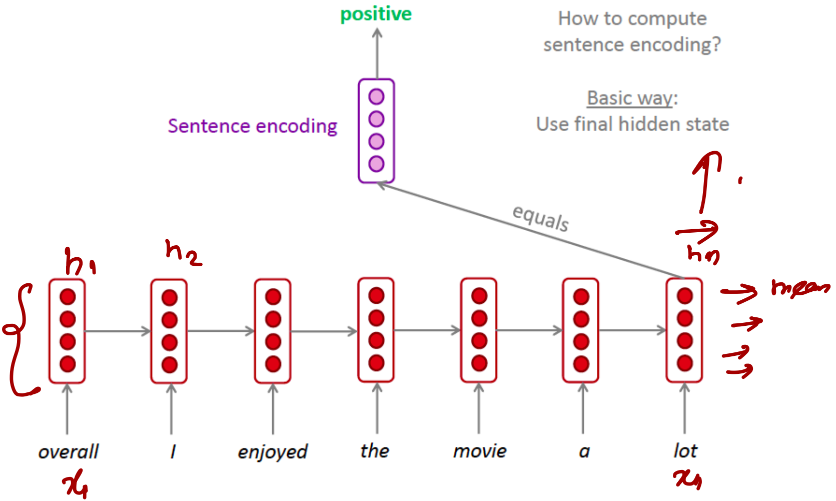
RNNs can be used for sentence classification

e.g., sentiment classification



How to compute sentence encoding?

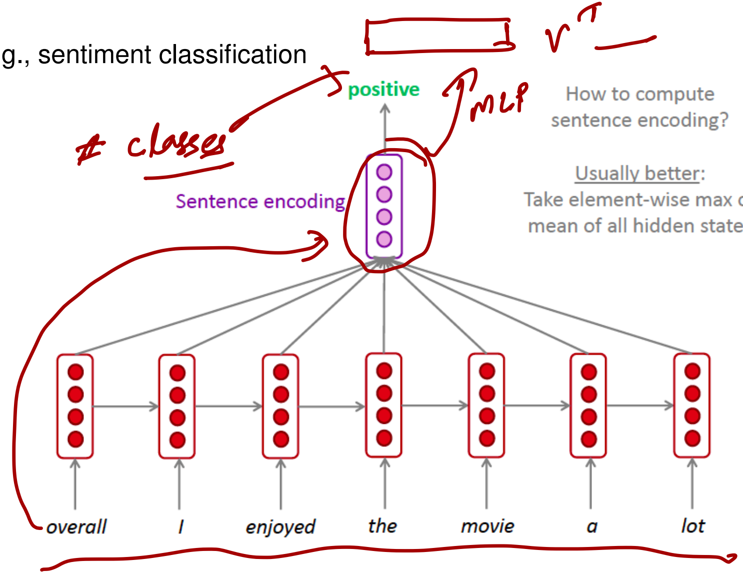
Basic way:
Use final hidden state

RNNs can be used for sentence classification

e.g., sentiment classification

v^T

CE loss

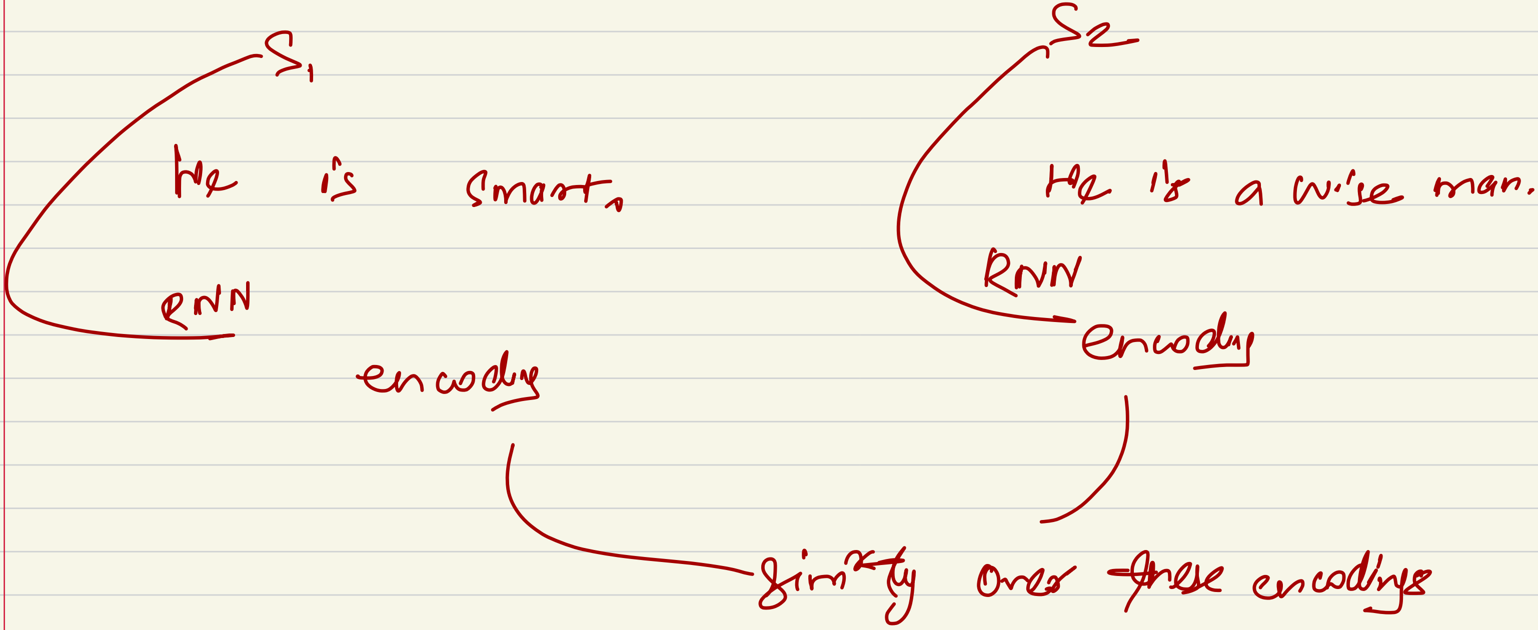


How to compute sentence encoding?

Usually better:
Take element-wise max or mean of all hidden states

element-wise max/mean

Sentence Similarity



S_1

He is smart.

RNN

encoding

S_2

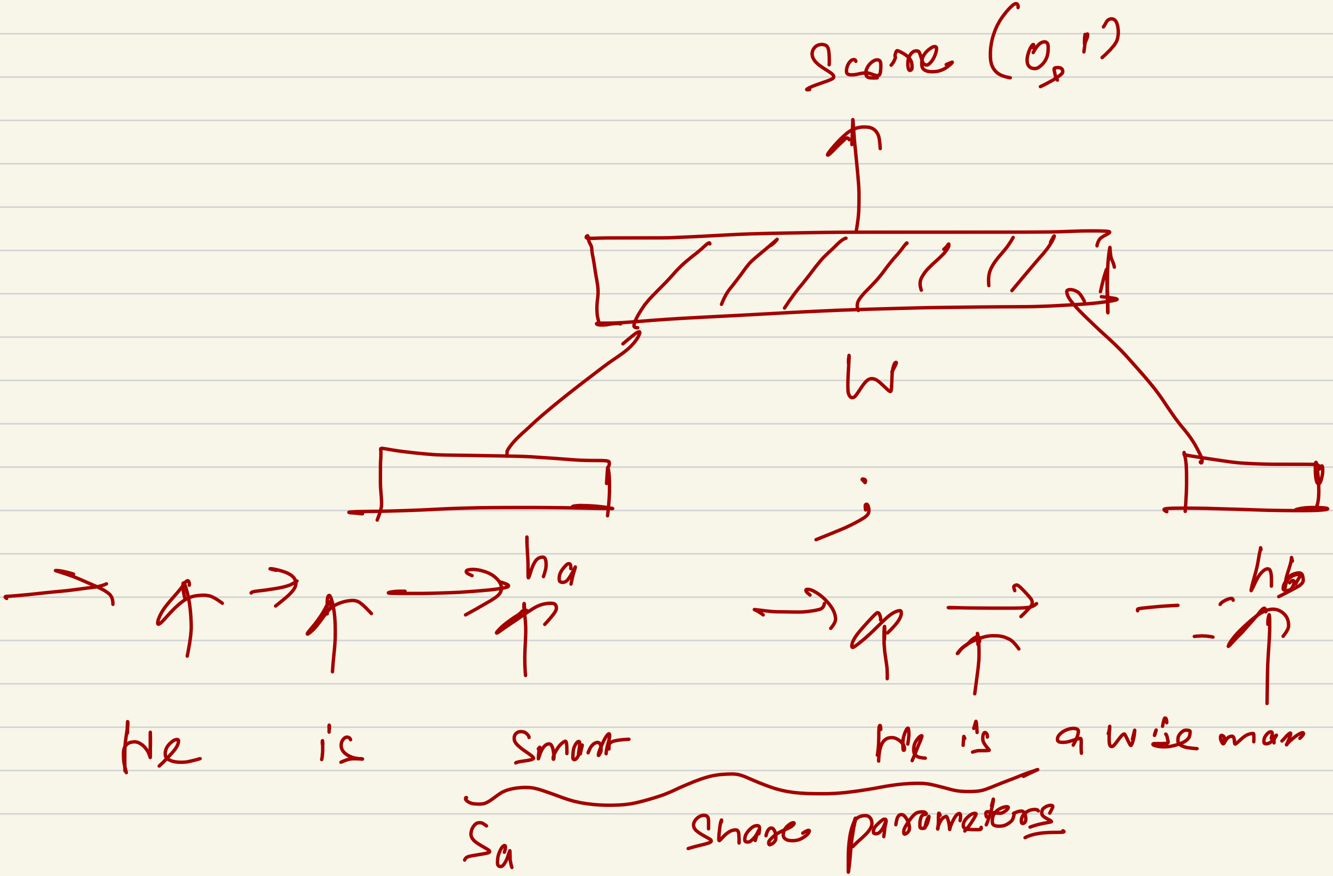
He is a wise man.

RNN

encoding

Simty over these encodings

Siamese Architecture

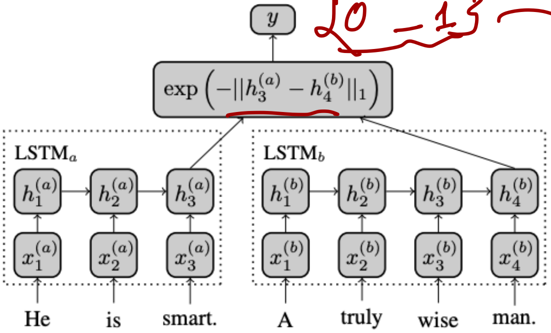

Score (0, 1)
W
;
ha
hb
He is Smart
He is a wise man
Sa
Share parameters

RNN
Copy

RNNs can be used to find sentence similarity

Siamese Architecture

- Pass both sentences through RNNs: the parameters are shared across these
- Use (a function of) the difference between the final hidden states

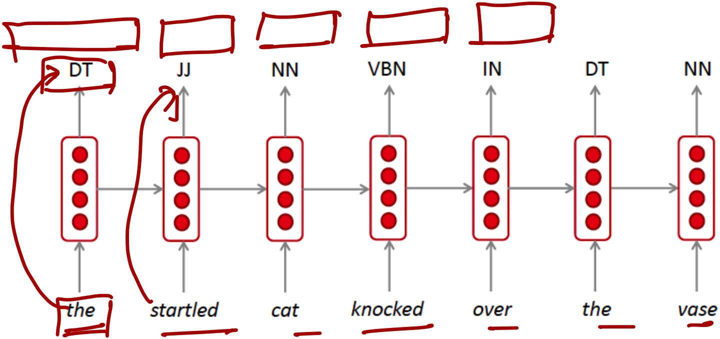


y
$\exp\left(-\|h_3^{(a)} - h_4^{(b)}\|_1\right)$
LSTM$_a$
LSTM$_b$
$h_1^{(a)}$ $h_2^{(a)}$ $h_3^{(a)}$
$x_1^{(a)}$ $x_2^{(a)}$ $x_3^{(a)}$
$h_1^{(b)}$ $h_2^{(b)}$ $h_3^{(b)}$ $h_4^{(b)}$
$x_1^{(b)}$ $x_2^{(b)}$ $x_3^{(b)}$ $x_4^{(b)}$
He is smart.
A truly wise man.

[0 – 1] → Can vary based on dataset

RNNs can be used for tagging

e.g., part-of-speech tagging, named entity recognition → Sequence labeling

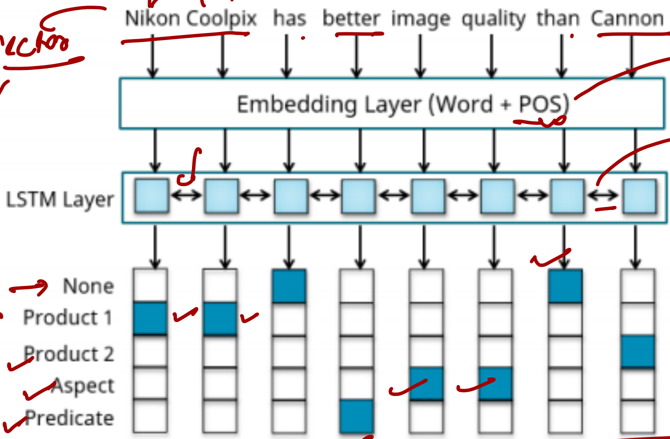


output at every step

size of o/p vector ?

| POS tags |

RNNs for compararison mining in Reviews


Nikon Coolpix has better image quality than Cannon
Embedding Layer (Word + POS)
LSTM Layer
None
Product 1
Product 2
Aspect
Predicate

adjective

postagger

vector

POS

bidirection

1-hot

00100

300-d POS embed.

initialized embeddings

1-hot vector / trained word embeddings

Bidirectional RNNs

h_t

$x^{(1)}$ - - $x^{(t-1)}$ $x^{(t)}$ x^{t+1} - - $x^{(n)}$

What we have seen till now?

- The state at time t only captures information from the past $x^{(1)}, \ldots, x^{(t-1)}$, and the present input $x^{(t)}$
- Some models also allow information from past y values to affect the current state when the y values are available

Text classification

LM?

POS tagging

↑ ↑ ↑ ↑ ↑

x_1 x_n

Given x_1 - - x_{t-1} predict x_t

Bidirectional RNNs

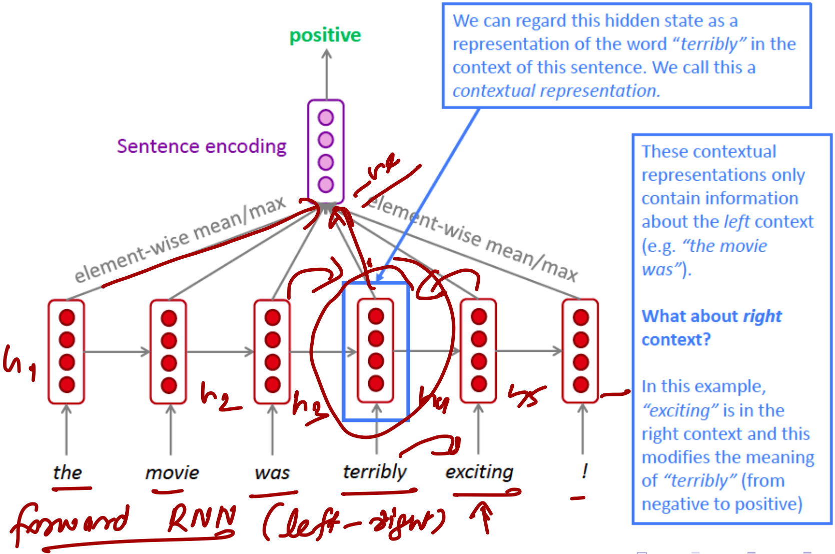
What we have seen till now?

- The state at time t only captures information from the past $x^{(1)}, \ldots, x^{(t-1)}$, and the present input $x^{(t)}$
- Some models also allow information from past y values to affect the current state when the y values are available

However, in many applications we want to output a prediction of $y^{(t)}$ which may depend on the whole input sequence. e.g., sentiment analysis.

Bidirectional RNNs: motivation for sentiment classification

Task: Sentiment Classification



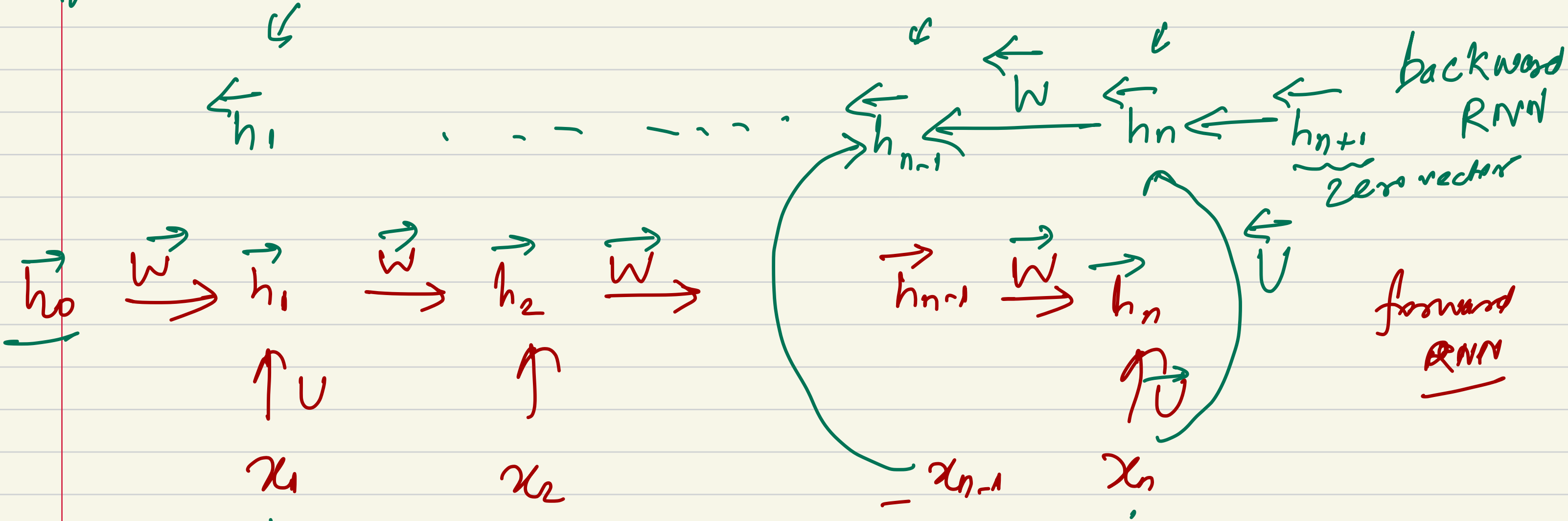
We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation.*

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)

Bidirectional RNN



$h_1 = [\overrightarrow{h_1} ; \overleftarrow{h_1}]$

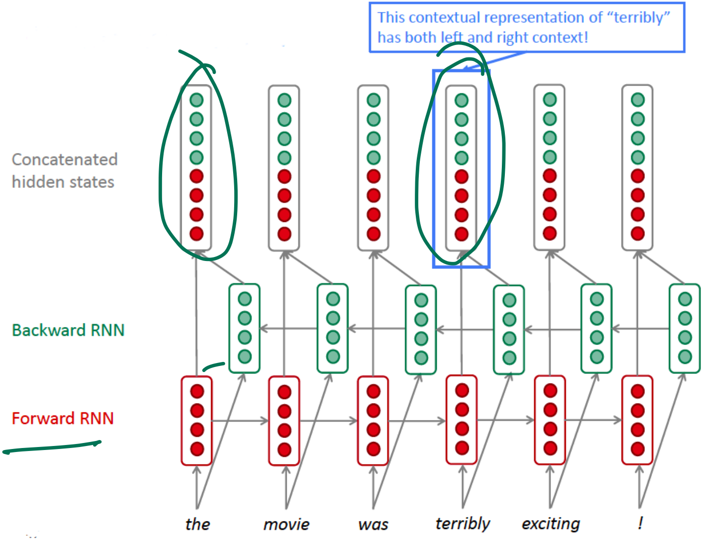
$\underset{\sim}{h_i} \; [\underset{\uparrow}{\overrightarrow{h_i}} ; \underset{\uparrow}{\overleftarrow{h_i}}]$

left context

right context

$h_n = [\overrightarrow{h_n} ; \overleftarrow{h_n}]$

Bidirectional RNNs



This contextual representation of "terribly" has both left and right context!

Not for LM

Bidirectional RNNs

On timestep *t*:

This is a general notation to mean "compute one forward step of the RNN" – it could be a vanilla, LSTM or GRU computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \boxed{\mathrm{RNN}_{\mathrm{FW}}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

Generally, these two RNNs have separate weights

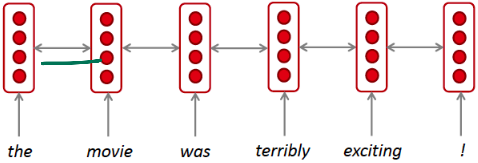
Concatenated hidden states $\boxed{\boldsymbol{h}^{(t)}} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

2h

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

O_t

Bidirectional RNNs: simplified diagram



The two-way arrows indicate bidirectionality and the depicted hidden states are assumed to be the concatenated forwards+backwards states.

Bi-RNNs are only applicable if you have access to the entire input sequence

- thus not applicable to Language Modeling where only the left context is available

Using Char-RNN for word embeddings

OOV words

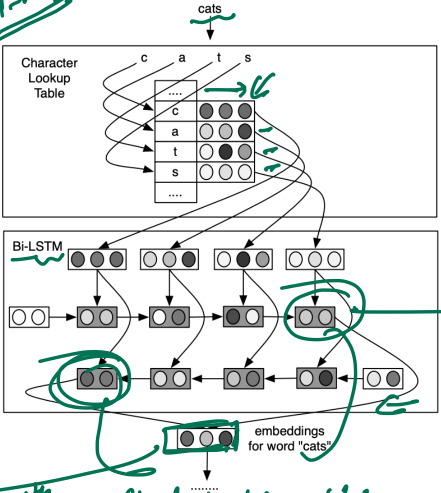
- what are character embeddings?

1-hot

- How do I train my network?

$y-\hat{y}$ MSE loss

Compare with "Cats"



Cats

c a t s

final hidden f/w

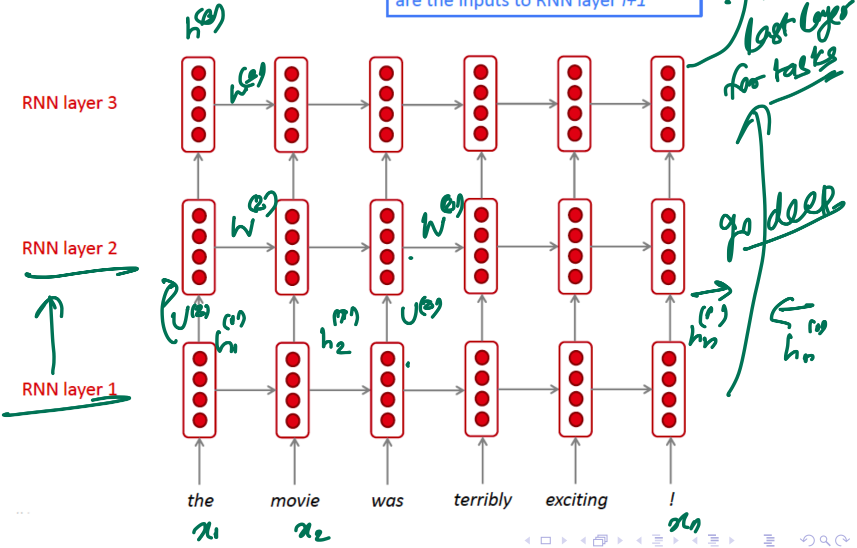
Concatenate

final hidden b/w

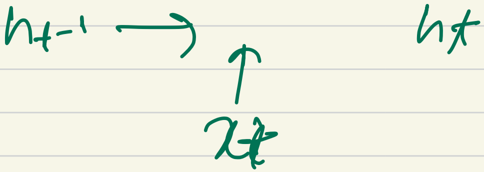
Multi-layer RNNs

Multi-layer RNNs

The hidden states from RNN layer *i* are the inputs to RNN layer *i+1*



Vanilla RNN



GRU, LSTMs (complicated units)

Bi-RNN, Bi-LSTM

Multi-layer

Multi-layer Bi-LSTM

What other paradigm can RNN be used for?

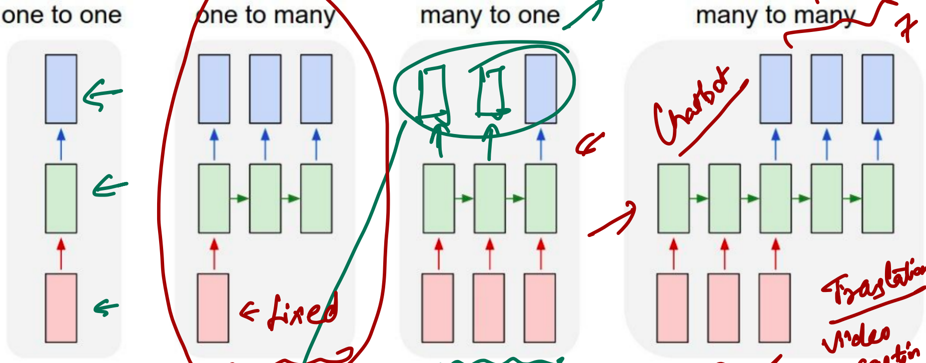


image captioning

Text classification

o/p fixed dim

o/p variable

10

7

Chatbot

fixed

NN

Seq. labeling

o/p at every step

i/p variable length

i/p variable

5

8

Translation

video caption

ASR

Enc.

dec.

Seq2Seq

Encoder

→ decoder

Sequence to Sequence Models

Pawan Goyal

CSE, IIT Kharagpur

NLP - Interaction Hour

What we have seen?

- Input sequence to a fixed-sized vector
- Input sequence to an output sequence of the same length

What we have seen?

- Input sequence to a fixed-sized vector
- Input sequence to an output sequence of the same length

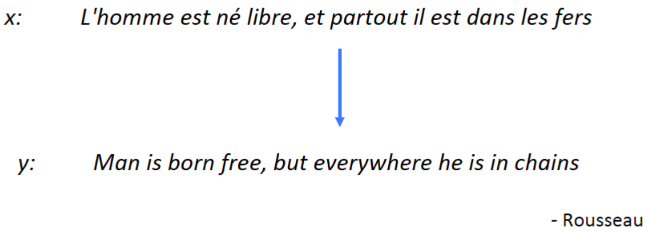
Any other constraint?

Mapping input sequence to an output sequence, not necessarily of the same length

machine translation, question answering, chatbots, summarization, ...

What is machine translation?

Machine Translation (MT) is the task of translating a sentence x from one language (the source language) to a sentence y in another language (the target language)



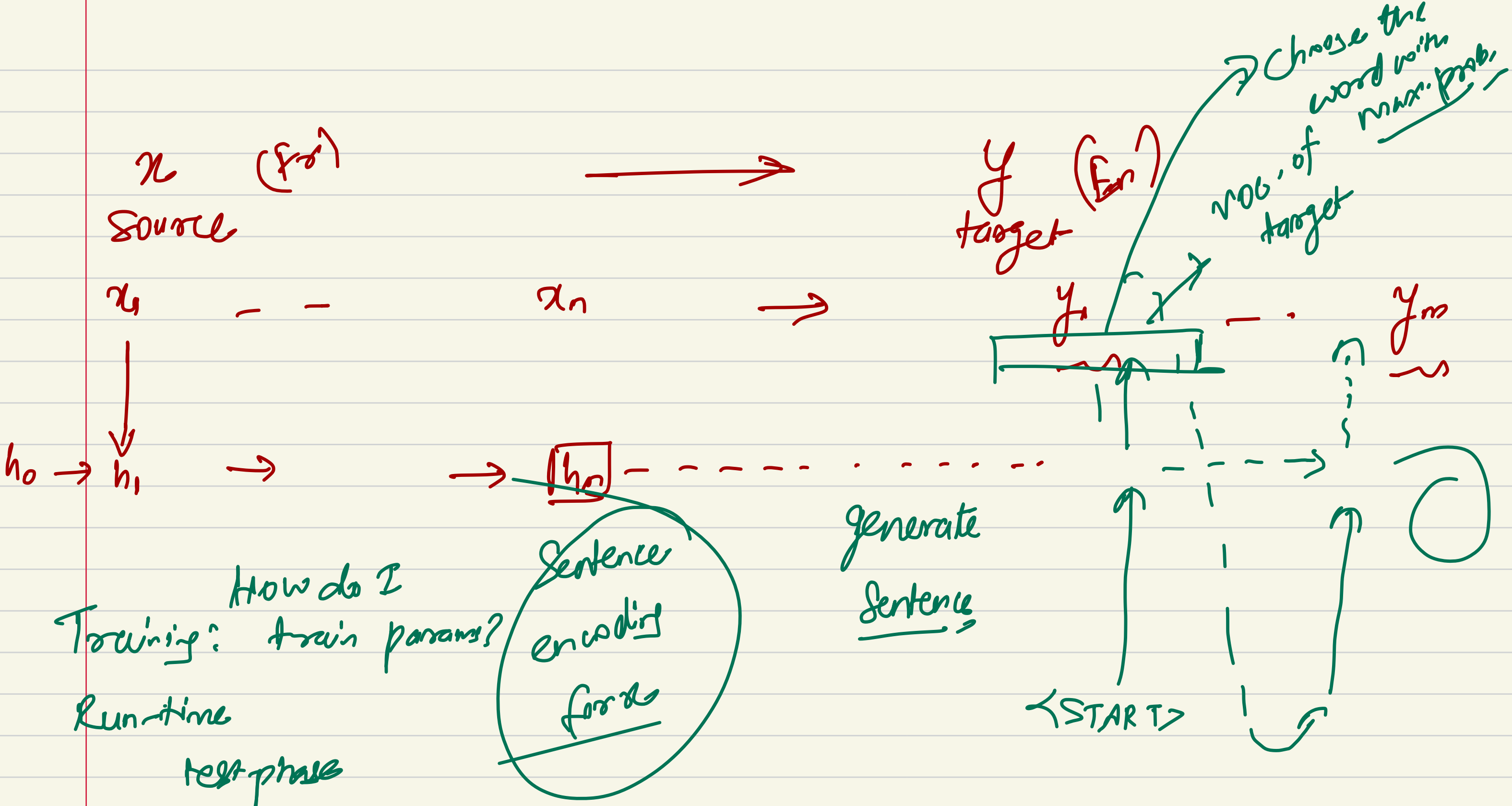
x: *L'homme est né libre, et partout il est dans les fers*

y: *Man is born free, but everywhere he is in chains*

- Rousseau

French

English



How do I

Training: train params?

Run-time

test-phase

↳ How do I use my learnt parameters to generate target sequence?

Generation using LM

Sequence-to-sequence architecture

Also known as encoder-decoder architecture

- Input sequence $X = (\underline{x^{(1)}, \ldots, x^{(n_x)}})$
- Output sequence $Y = (y^{(1)}, \ldots, \underline{y^{(n_y)}})$

Sequence-to-sequence architecture

Source encoding

Also known as encoder-decoder architecture

- Input sequence $X = (x^{(1)}, \ldots, x^{(n_x)})$
- Output sequence $Y = (y^{(1)}, \ldots, y^{(n_y)})$
- Encoder (reader/input) RNN: Emits the context C, a vector summarizing the input sequence, usually as a simple function of its final hidden state

✓

C learnable

summarization

$$\boxed{h_n} \xrightarrow{W} h_0'$$

d_1 En

d_2 Fr

Sequence-to-sequence architecture

Also known as encoder-decoder architecture

- Input sequence $X = (x^{(1)}, \ldots, x^{(n_x)})$
- Output sequence $Y = (y^{(1)}, \ldots, y^{(n_y)})$
- Encoder (reader/input) RNN: Emits the context C, a vector summarizing the input sequence, usually as a simple function of its final hidden state
- Decoder (writer/output) RNN: Is conditioned on the context C to generate the output sequence.

Sequence-to-sequence architecture

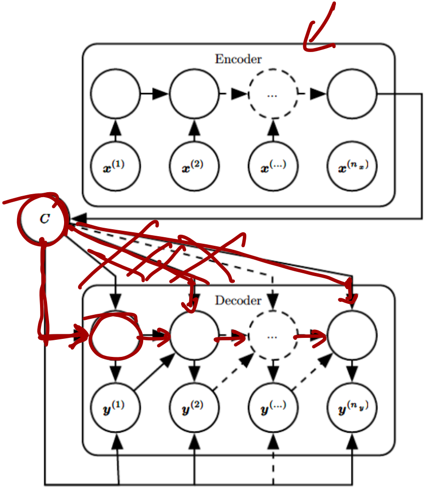
Also known as encoder-decoder architecture

- Input sequence $X = (x^{(1)}, \ldots, x^{(n_x)})$
- Output sequence $Y = (y^{(1)}, \ldots, y^{(n_y)})$
- Encoder (reader/input) RNN: Emits the context C, a vector summarizing the input sequence, usually as a simple function of its final hidden state
- Decoder (writer/output) RNN: Is conditioned on the context C to generate the output sequence.

What is the innovation?

The lengths n_x and n_y can vary from each other

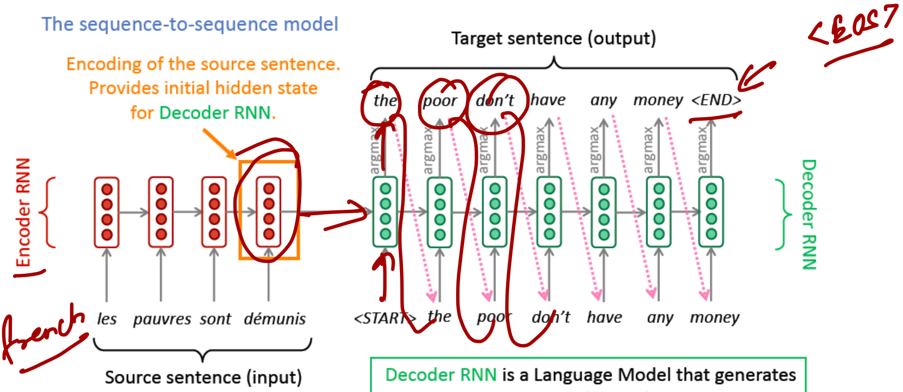
Encoder-Decoder Architecture



Encoder and Decoder RNNs

- Last state of the encoder RNN is used as a representation C of the input sequence, provided as input to the decoder RNN
- Decoder RNN is a vector-to-sequence RNN, described earlier.
- Two ways to receive input: as the initial state of the RNN, or can be connected to the hidden units at each time step. These two ways can also be combined.

Sequence to Sequence Models for Machine Translation



The sequence-to-sequence model

Encoding of the source sentence. Provides initial hidden state for Decoder RNN.

Decoder RNN is a Language Model that generates target sentence conditioned on encoding.

Encoder RNN produces an encoding of the source sentence.

Note: This diagram shows **test time** behavior: decoder output is fed in ······> as next step's input

Sequence to sequence is versatile

Many NLP tasks can be phrased as sequence-to-sequence

- Summarization (long text → short text)
- Dialogue (previous utterances → next utterance)

Neural Machine Translation (NMT)

An example of a Conditional Language Model

- **Language Model** because the decoder is predicting the next word of the target sentence y
- **Conditional** because the predictions are also conditioned on the sentence x

NMT directly calculates $P(y|x)$:

$$\underbrace{P(y|x)}_{y} = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots \underbrace{P(y_T|y_1, \ldots, y_{T-1}, x)}_{\text{Probability of next target word, given target words so far and source sentence } x}$$

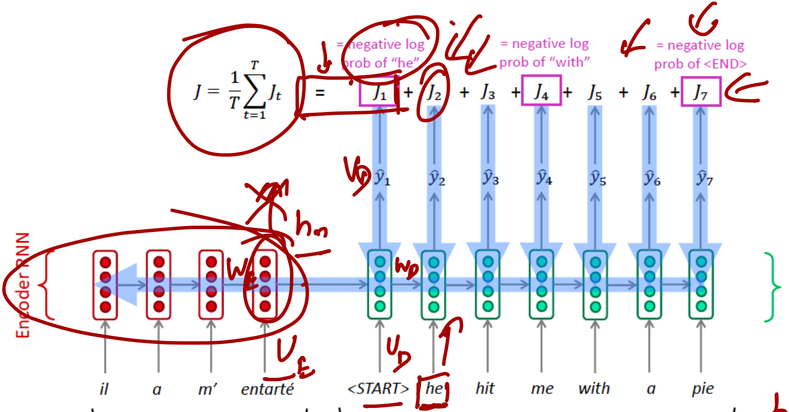
Parliament
Sessions
websites

How to train an NMT system?

Get a big parallel corpus ...

En S_{En} ⟷ S_{Fr} Fr

Training an NMT system



$J = \frac{1}{T}\sum_{t=1}^{T} J_t = J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7$
= negative log prob of "he"
= negative log prob of "with"
= negative log prob of <END>
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_3$ $\hat{y}_4$ $\hat{y}_5$ $\hat{y}_6$ $\hat{y}_7$
Encoder RNN
Decoder RNN
il a m' entarté
<START> he hit me with a pie
V_D
h_n
W_E
W_D
U_E
U_D
x_n
x_1

Source sentence (from corpus)

Target sentence (from corpus)

Seq2seq is optimized as a single system. Backpropagation operates "end-to-end".

teacher forcing

being trained simultaneously

Greedy decoding

One possibility is to generate (decode) the target sentence by taking argmax on each step of the decoder

Seq2seq

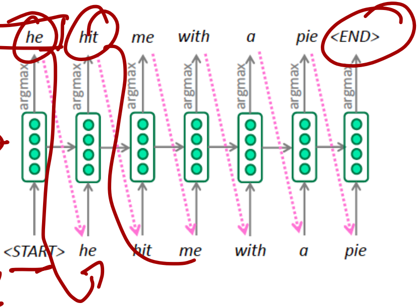
Encoder C.T.C.

+

LM

$h_n^{(e)}$ →

Train end-to-end



or whenever I reached a limit

— K words

This is greedy decoding

Problems with this method?

Seq2seq. large i/p dataset