Week 3 : Sequence Models & Attention Mechanism

1] Sequence to sequence architectures

1) Basic Models

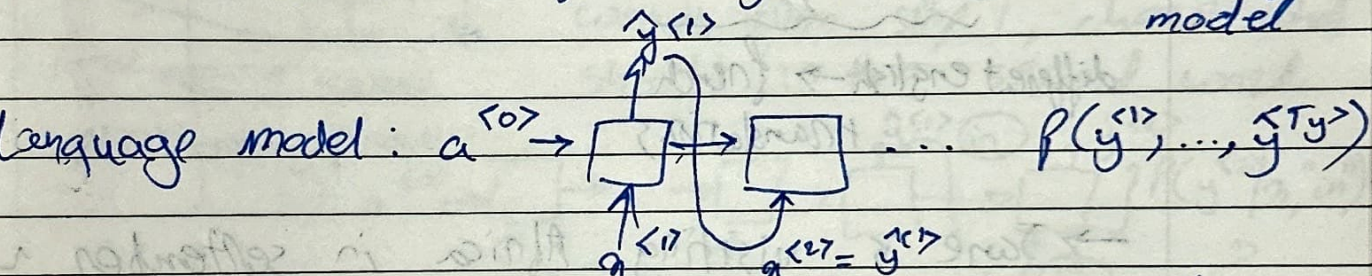
→ Machine translation & image captioning

↓

The input sentence is encoded using a NN (GRU or LSTM) then passed to a decoder which generates the output sentence. Similar methods using CNNs are applied for image captioning which will provide a short description of the input image.

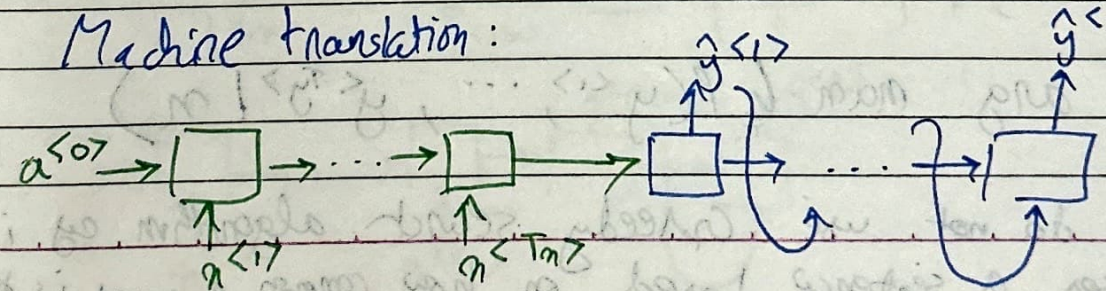
2) Picking the most likely sentence

Machine translation as building a conditional language model



Used to estimate the probability of a sentence or to generate novel sentences. (new/not seen).

Machine translation :



Page No.
Date

Instead of vectors of zeroes that are passed to the NN, it has an encoded network that figures out some representation for the input sentence rather than with representation of all zeros.

→ it is "Conditional language model"

because it is modeling $P(y^{<1>}, \ldots y^{<T_y>} | x^{<1>}, \ldots, x^{<T_x>})$

Probability of output English translation given (conditioned on) some french sentence.

Finding the most likely translation:

eg) Jane visite l'Afrique en septembre

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

different english → french translations

→ Jane is visiting Africa in septemfer ✓
→ Jane is going to be visiting Africa in septemba.
→ In september, Jane will visit africa
→ her ~~f~~African friend welcomed Jane in septemben

↓ coarse

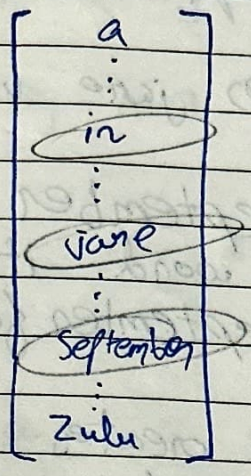
$$\arg\max_{y} P(y^{<1>}, \ldots, y^{<T_y>} | x)$$

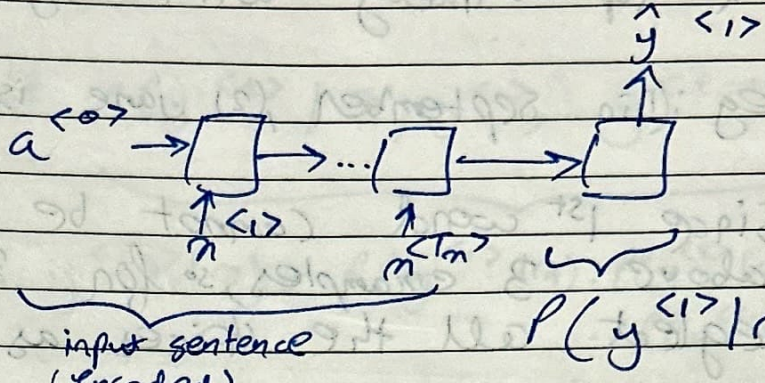
we do not use Greedy search algorithm eg it chooses a sentence based on how common a word is than appropriate

Page No.
Date

3) Beam search → B (beam width)
no. of possibilities that are considered

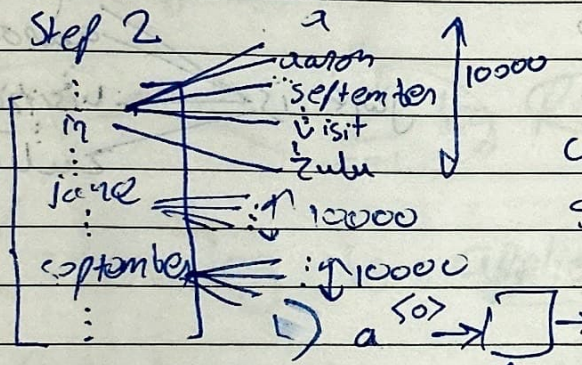
Step 1



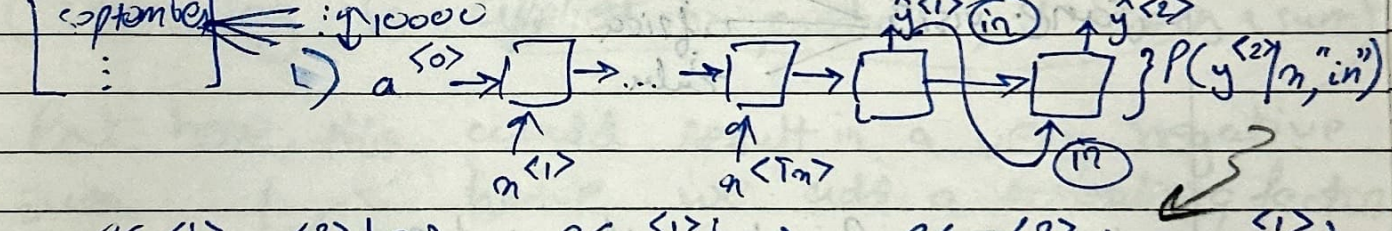


$\hat{y}^{<1>}$ is a softmax layer overall outputing 10000 possibilities and keep the top 3 words in the memory given that $B = 3$.

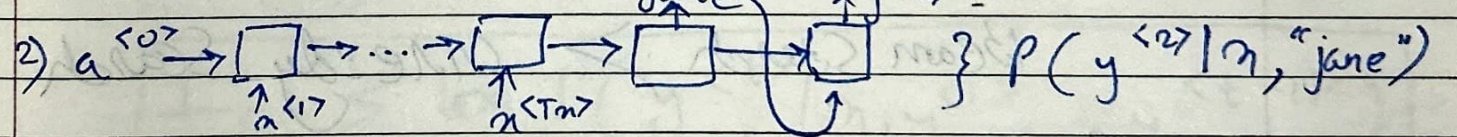
Step 2

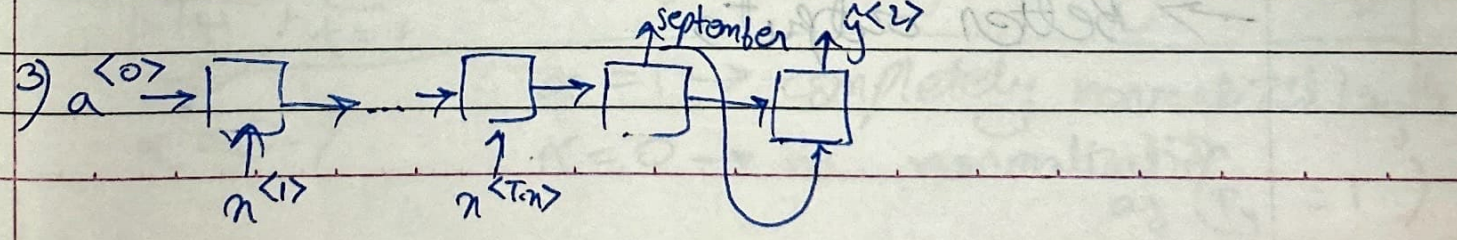


For each of the 3 chosen words from step 1, choose what should be the second word.



$$\rightarrow P(y^{<1>}, y^{<2>}|x) = P(y^{<1>}|x) * P(y^{<2>}|x, y^{<1>})$$





Page No.
Date

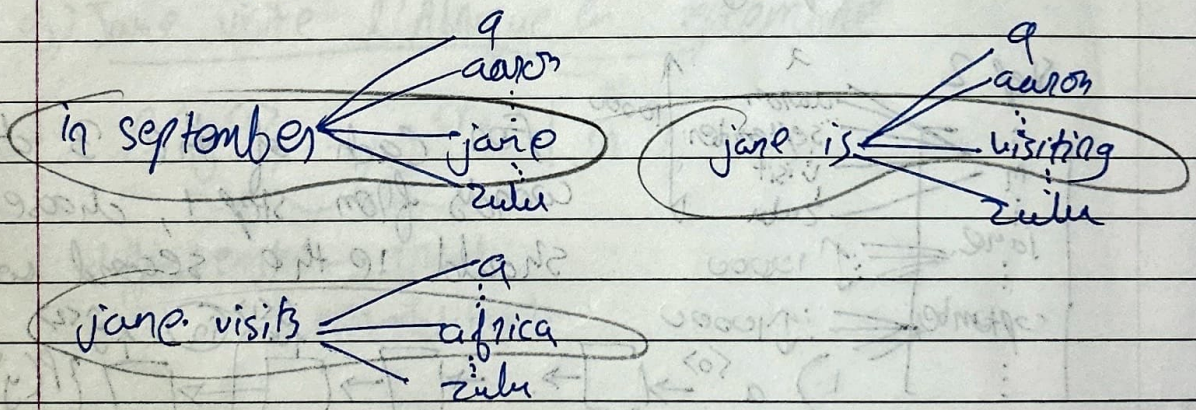
Total $3 \times 10000 = 30000$ possibilities.
Evaluate all the 30000 options and pick the top 3 likely with highest probabilities.

eg: (1) in september, (2) jane is (3) jane visits

Since 1st word cannot be september acc. to above 3 examples, so for 3rd word it will neglect all the choices as september for 1st word.

Also since $B = 3$, the algorithm creates 3 copies of different $\hat{y}^{<1>}$ which are further used to predict 2nd word.

Next Step,



Note: If $B = 1$, this algo becomes a greedy search algo.

Beam Search > Greedy Search

→ Better outputs

Page No.
Date

4) Refinements to Beam Search (Length Normalisation)

previously we used :

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

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

as we can see that the above result will be a product of probabilities ~~her~~ and each prob. being <1 will result in a number very close to zero ~~hence~~ and since the small value cannot be stored in the memory, we shall add log to the function.

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

(when log is applied to product, it becomes a sum)

But here, this would result in a very negative sum hence we add a normalizing factor.

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

$\alpha = 0.7$

$\alpha = 1 \rightarrow$ completely normalized length

$\alpha = 0 \rightarrow$ no normalization as $(T_y^{0} = 1)$

Page No.
Date

Beam width B:

1000 / 3000 → large B: better results, but slower (computationally)
also memory requirements ↑

3 → small B: worse results, (faster computationally)

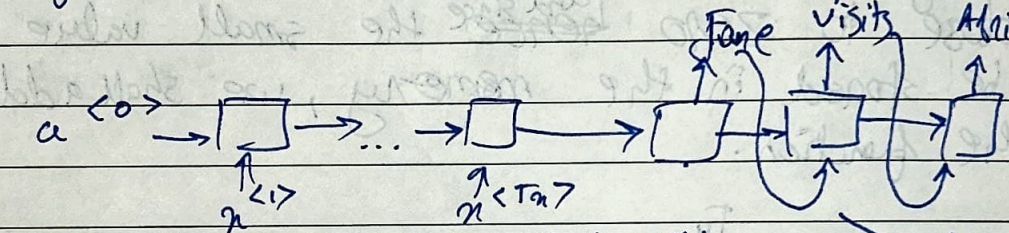
5) Error analysis in Beam Search

~~RNN computes $P(y|x)$~~

Jane visite l'Afrique en septembre.

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

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



RNN computes $P(y^*|x) \geq P(\hat{y}|x)$

To see which one is bigger to clearly ascribe/identify and attribute a specific error to a cause or source within the process.

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

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

Conclusion: Beam search is at fault.

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

in ~~other~~ y^* is a better translation than $\hat{y}$. But RNN predicts $P(y^*|x) \leq P(\hat{y}|x)$

Conclusion: RNN model is at fault.

Page No.
Date

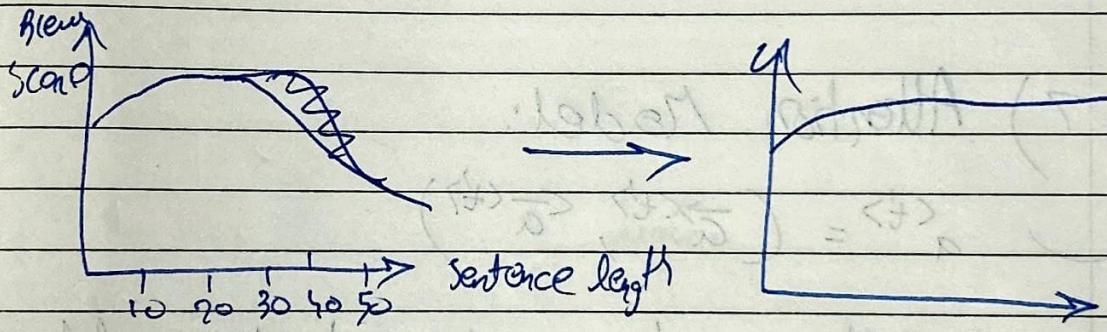
Error Analysis Process

Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
...	...	2×10^{-10}	1×10^{-10}	Beam
		:	:	RNN
				Beam
				:

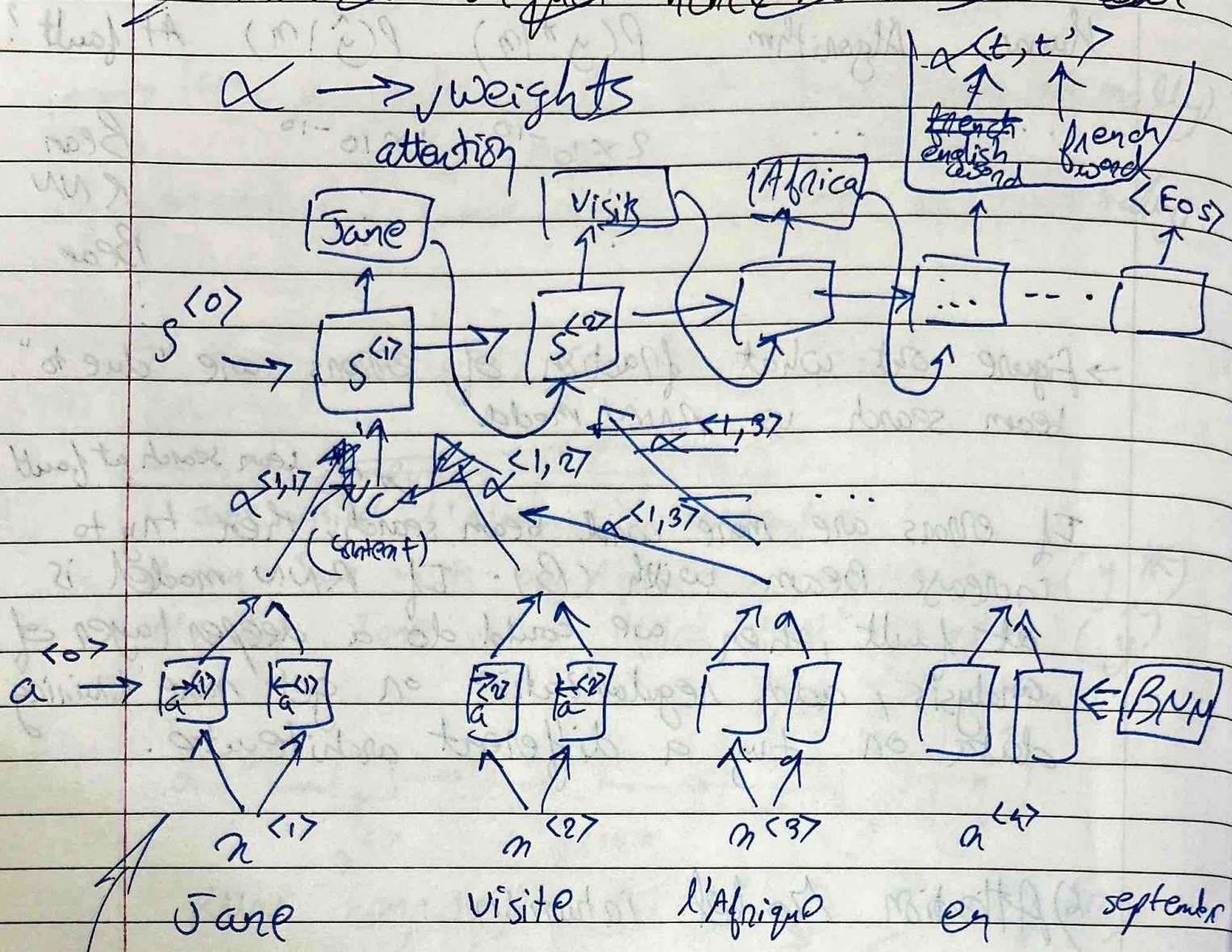
→ figure out what fraction of errors are "due to" beam search vs RNN model

If errors are more (i.e. beam search at fault) with beam search, then try to increase Beam with (B). If RNN model is at fault, then we could do a deeper layer of analysis, add regularization or get more training data or try a different architecture.

6) Attention Model intuition :



This score goes down as the model cannot remember sentences of high lengths at once. Hence, attention model is used which will translate the sentence part-by-part just like human translators do.



7) Attention Model:

$$a^{\langle t \rangle} = (\overrightarrow{a}^{\langle t \rangle}, \overleftarrow{a}^{\langle t \rangle})$$

α tells us how much the context would depend on the features / activations we're getting from different time steps.

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

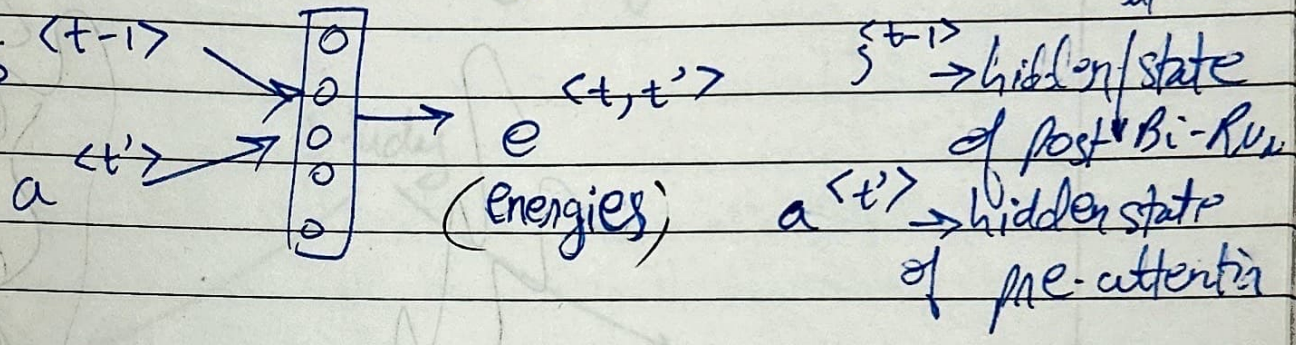
$$c^{\langle 1 \rangle} = \sum_{t'} \alpha^{\langle 1, t' \rangle} a^{\langle t' \rangle}$$

Page No.
Date

$\alpha^{<t,t'>}$ = amount of "attention" $y^{<t>}$ should pay to $a^{<t'>}$ i.e. importance of that input word.

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

Can be computed using a small NN:



2] Trigger word detection

→ Alexa, Google Home