3. Equalise pairs.

* Week 3.

Basic Model:

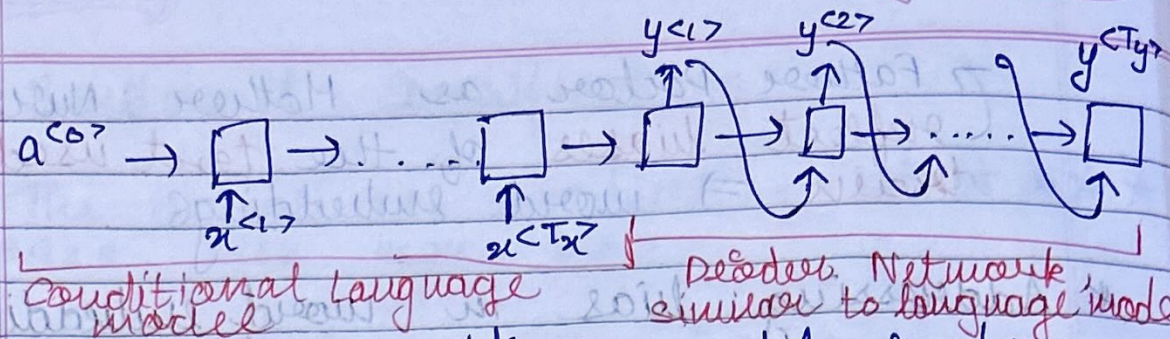
Machine Translation, Image Captioning

→ follow encoder - decoder architecture.

Input : French sentence Output : Translated english

Encoding network: input French sentence & generate its embeddings

Decoding network: find the corresponding similar english word and output it.



- In the image captioning model, encoder converts image to embeddings then applies softmax & decoder generates caption.

~~Ea~~

- The key difference b/w language model & ML model is that of conditional language model.
Here (ML), we're estimating probability of an English sentence given the condition of input french sentence

Whereas in language model we estimate accuracy of target.

* Finding most likely translation:

Jane visite l'Afrique en septembre

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

outputs — English — french

or random
→ Jane is visiting Africa in Sep.
→ Jane " going to be visiting Africa in Sep.
→ In September, Jane will visit Africa.
→ Her African friend welcome Jane in Sep.

Clearly, we do not want to distribute at random as it will not only not give best output but also at times give wrong output.

Thus, we want a func^n that maximise:

$$\arg\max P(y^{<1>}, \ldots, y^{<T_y>} \mid \underbrace{x}_{\text{input french sentence}})$$

* Why not greedy search?

- Greedy search is an algo which searches for the best most appropriate word wrt previous word.

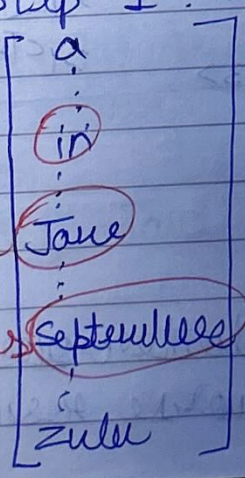
- Here, however we're estimating for the sentence as a whole than a word. Thus, it is not useful.
  For eg: Greedy search ~~will~~ will (might) choose the second estimate over first of previous page, which not the best trans.

* Beam Algorithm:

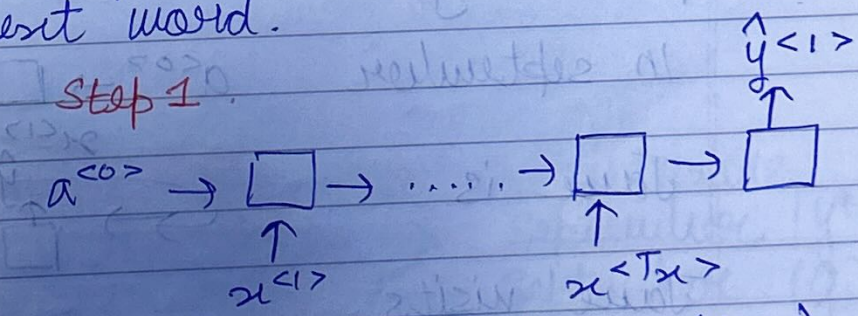
- Based off the previous words (or clippings) finds the next word.

Step 1:

Let's consider it selects these 3 to be most probable first words

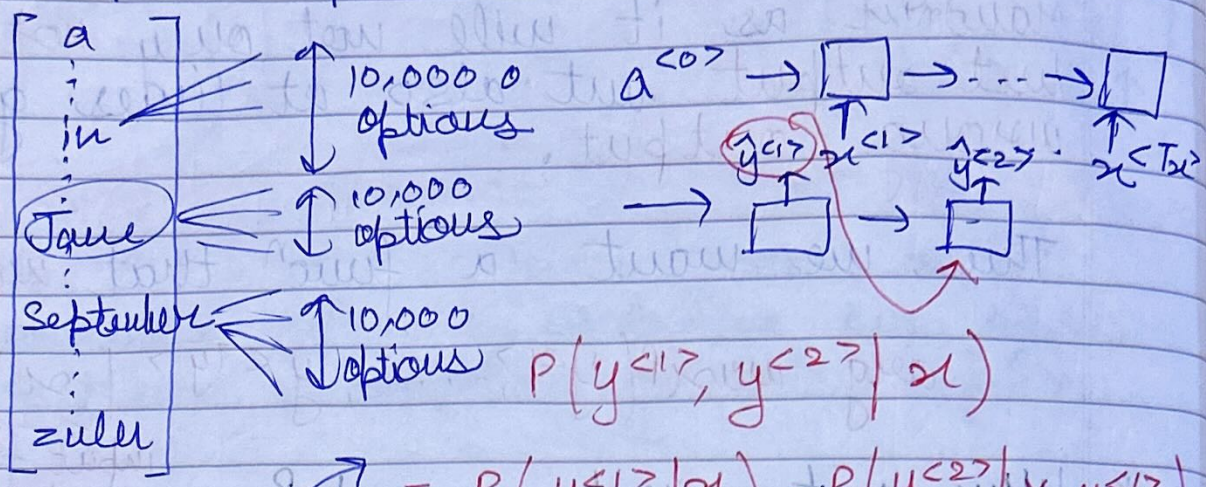


Step 1



B = 3 (Breath of Beam Algo) → implies that the algo finds 3 most appropriate words as 1st word (& so on).

Step 1 Step 2



$$P(y^{<1>}, y^{<2>} \mid x)$$

$$= P(y^{<1>} \mid x) \cdot P(y^{<2>} \mid x, y^{<1>})$$

This step is repeated thrice for in, Jane & September.

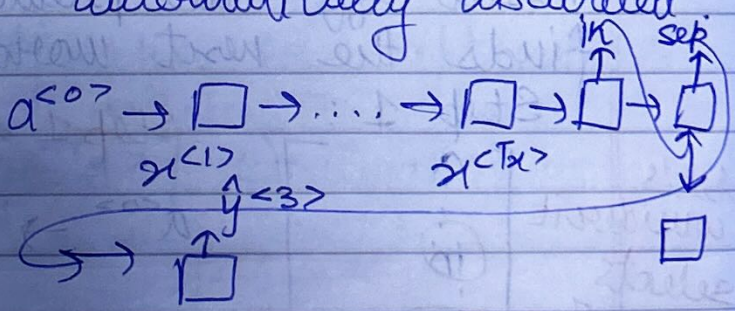
Here, $y^{<1>}$ will be hardcoded to in/ Jane/ September.

Similar for third step and so on.

In step 2, if algo recognises in Sep, Jane is & Jane visits as the three outcomes ($\hat{y}^{<2>}$) then September for from $\hat{y}^{<1>}$ is automatically discarded.

In september

Jane is



Jane visits

B (Beam Width) ⇒ ↑B, better result, slower

B=1 (Greedy Search) ↓B, worse result, faster

* Length Normalisation:

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

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

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

→ maximising log will give same result as maximising product

→ Further we can normalise above by ~~not~~ dividing by T_y for outputting longer translations.

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

for softer normalisation, $\alpha = 0.7$.

- Beam Search is faster than BFS and DFS but does not guarantee accurate result.

* Error Analysis:

Eg: Jane visite l'Afrique en septembre

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

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

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

Beam Search chose $\hat{y}$, but y^* is more accurate. Beam Search is at fault.

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

~~RNN~~ y^* is better, but RNN predicted $P(y^*|x) \leq P(\hat{y}|x)$.
Thus, RNN model is at fault.

Error Analysis process:

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

Figure out what fraction of errors are due to Beam Search & RNN.

* BLEU Score

Bilingual Evaluation of understody

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

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

P_n = Blue score on n grams only.

Combined BLEU score (to avoid very short trans)

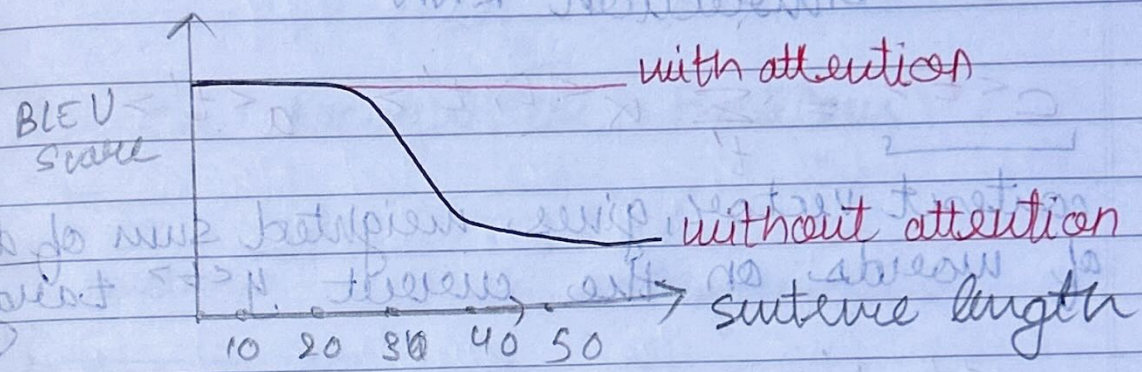
$$BP = \begin{cases} 1 & \text{if } MT > ref \\ e^{1 - ref/MT} \end{cases}$$

ref output, model output

$$BP \exp\left(\frac{1}{4} \sum_{n=1}^{4} P_n\right)$$

→ till 4 n-grams

* Attention Model



α^{11} = when computing first word (english), how attention to give to first word

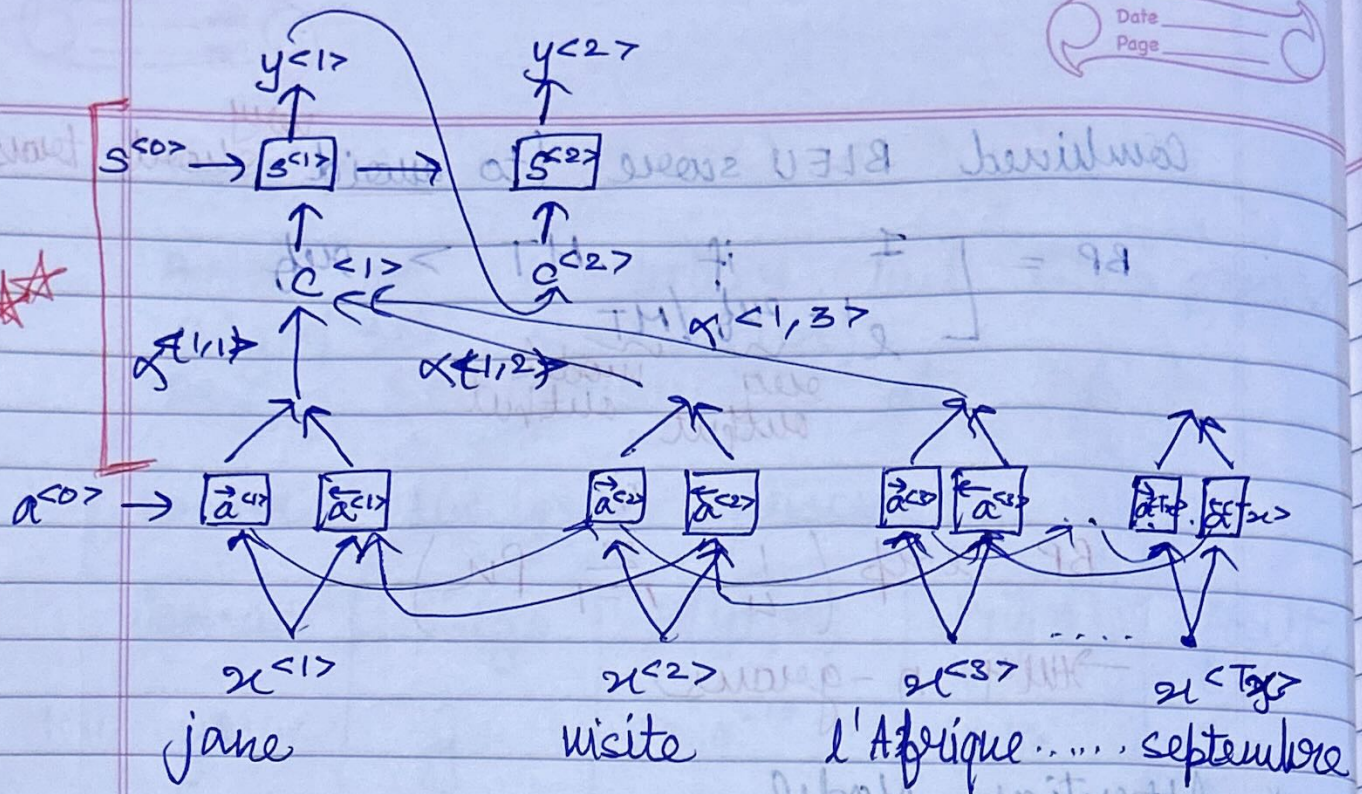
α^{12} = ~~input~~ attention to 2nd (french) while computing 1st word (english)

It allows the NN to pay attention to only part of input sequence over the entire sentence.

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

→ feature vector of time stamp t

→ using $a^{<t'>}$ for french word inputs



Bidirection RNN

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

context vector, gives weighted sum of dependencies of words on the current $y^{<t>}$ target.

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

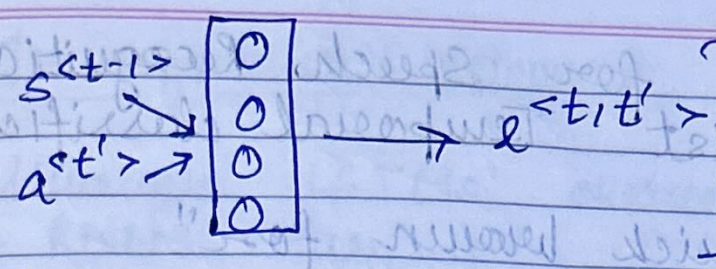
$$\sum_{t'} \alpha^{<t,t'>} = 1$$

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

ensures that sum is 1.

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

We know, that $\alpha^{<t,t'>}$ depends on $a^{<t'>}$ and $s^{<t-1>}$, thus $e^{<t,t'>}$ will also depend on it.



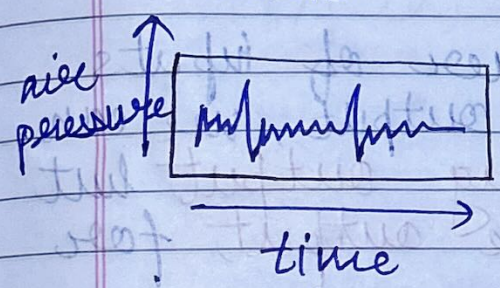
We build a small NN, & train it (train set) as we don't know what the function is

Thus, we trust back prop., & gradient descent to train the NN and identify (implement) the correct func^n.
(learn)

Disadvantage: Quadratic Cost ($T_x \times T_y$), however acceptable for ML translation sentences as most are not very long.

* Speech Recognition Problem:

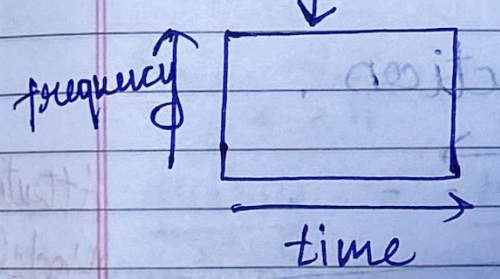
x audio $\longrightarrow$ y transcripts



$\longrightarrow$ "the quick brown fox"

Phonemes: de kwik braon

handwritten, used earlier & recognised as best method, however, not necessary today



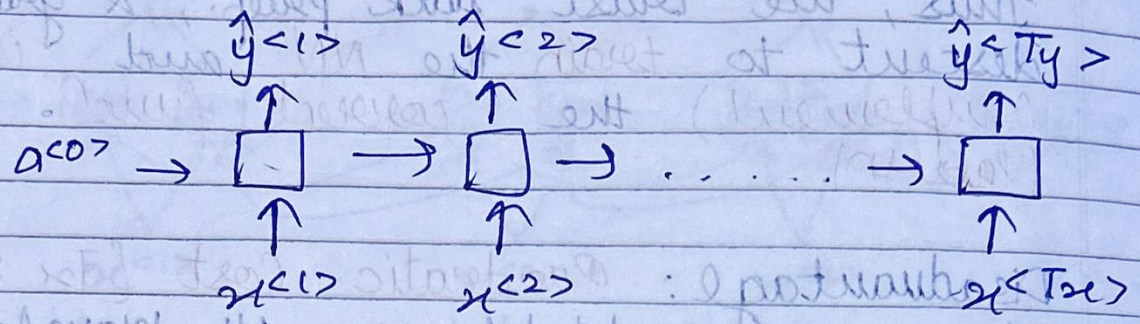
300 hour
3000 hour } train set

100,000 hour → commercial data set

→ We can use attention models over phonemes or a CTC approach ~~to implement it~~ ($T_x = T_y$).

* CTC cost for Speech Recognition: (Connectionist Temporal classification)

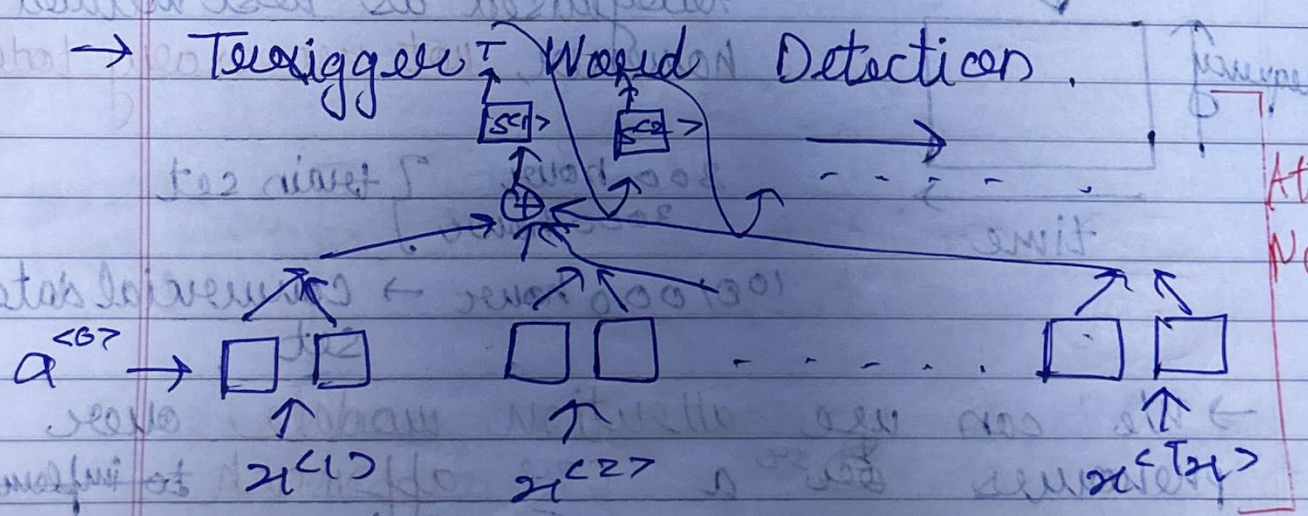
→ y ⇒ "the quick brown fox"



→ t t t _ h _ eee --- ⊔ --- qqq --

space blank

Rule: Collapse repeated character not separated by "blank".

Thus, for the number of inputs being greater than output, we can not only generate long output but also then concatenate output for correct result.

→ Trigger Word Detection.