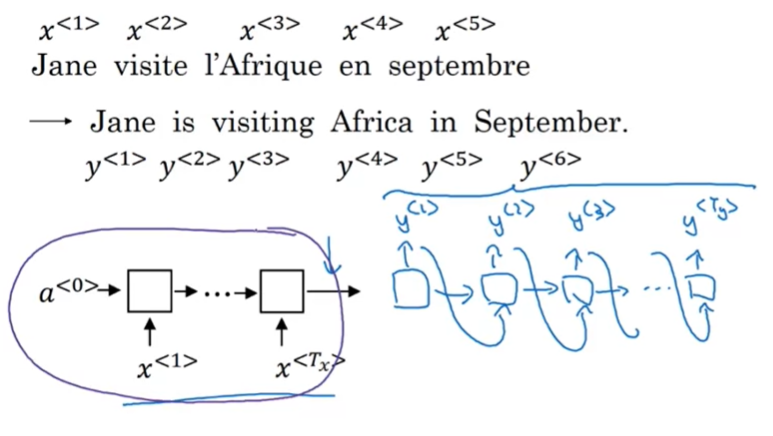
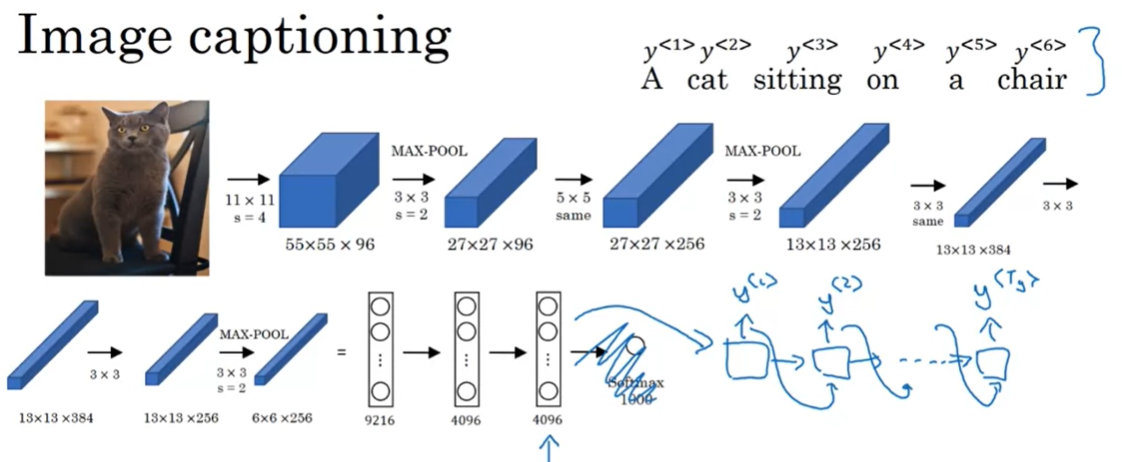
## Sequence to sequence Architecture (many to many RNN ) :



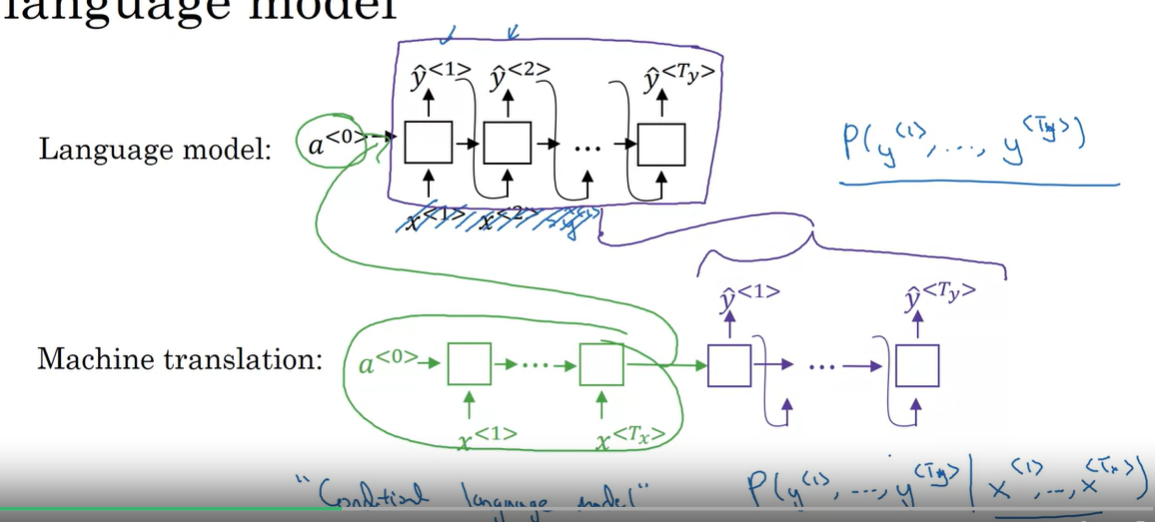
* As we already saw, the many to many expects a sequence of Tx size and the and the output is sequence of Ty size , where X represents the encoding port and y the decoding part
* We pass the whole input sequence ( x<1> , …,x<Tx> ) before starting to calculating the sequence representing the output Y
* The machine translation ( the image above ) is a good illustration of sequence to sequence model

### Image captioning is also an interesting illustration:



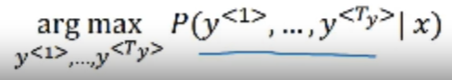
* The task here is “give me a suitable caption about the image” which is “a cat sitting on a chair” in this example
* Instead of doing a softmax classifier to predict what’s inside the pic as we used to do in computer visions, we inject the output vector from the last layer (the 4096 dimension vector ) to the decoding part of our RNN model .

## Machine translation as building a conditional language model:



For one-to-many architecture like language modeling : the first hidden state a<0> is just a zeros-vector , but this is different for sequence to sequence architectures where instead of having a zero-vector a<0> we are having a set of x<t> that’s goanna update the a<0> before getting passed to the decoding part of our RNN model . So we can consider the sequence X as a condition to consider where generating the output Y ( which is a zero vector for one-to-many architectures like language modeling )

* So we are going to generate the most suitable set of y<1>….y<t> given x<1>,….x<Tx> as an input by maximizing the probability of P( y<1> , ….., y<Ty> | x<1> , ……, x<Tx>)



### Greedy search as a potential solution :

* The greedy search algorithms consists of : starting to generate y<1> by finding the best one which responds to Max(P( y<1> | X ) ) and then going to generate y<2> by finding the best one which responds to Max( P ( y<1> , y<2> | X )) and so on

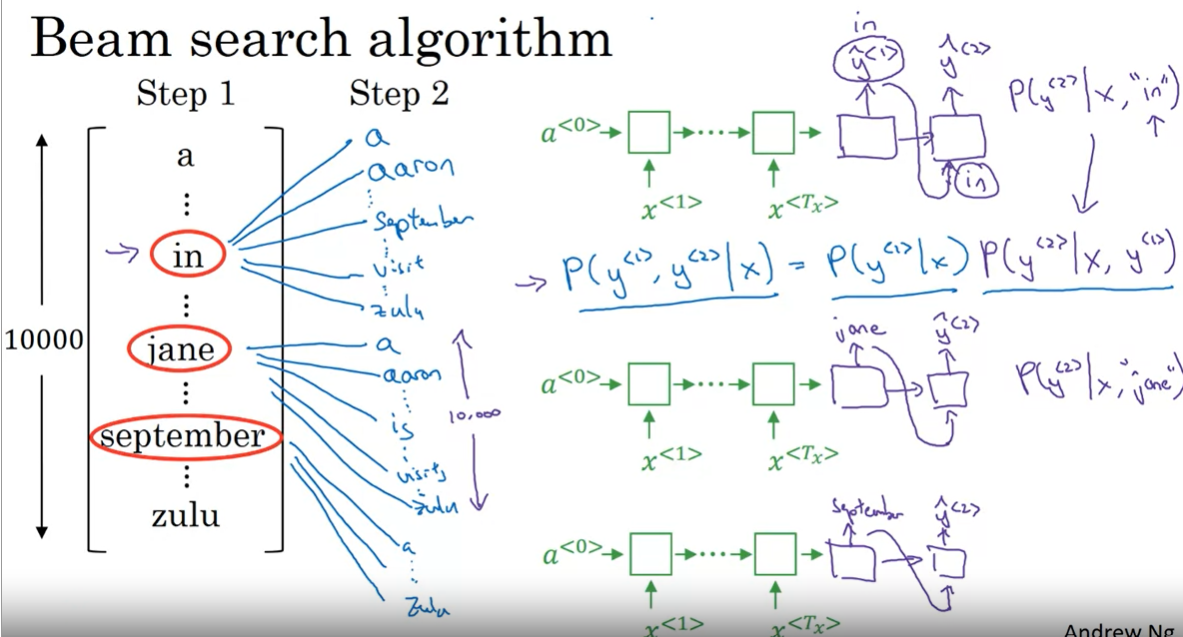
#### Greedy search isn’t a good solution:

* Suppose we wanna translate the sentence “Jean visite l’Afrique en Septembre” to English:

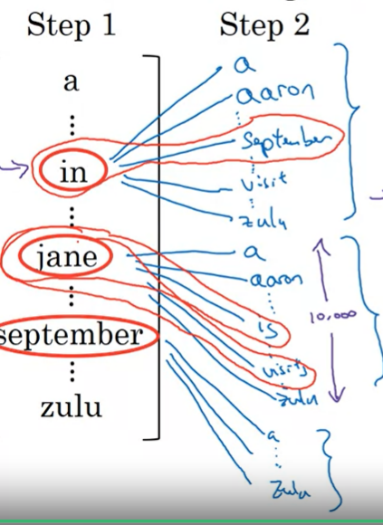


* The greedy search will favorize “Jane is going to be visiting Africa in September” , although “Jane is visiting Africa in September” is more suitable and précised and this is due to having P ( “Jane is going” | X ) > P( “Jane is visiting” | X ) because “going” is a very popular word in English but it took us to wrong path

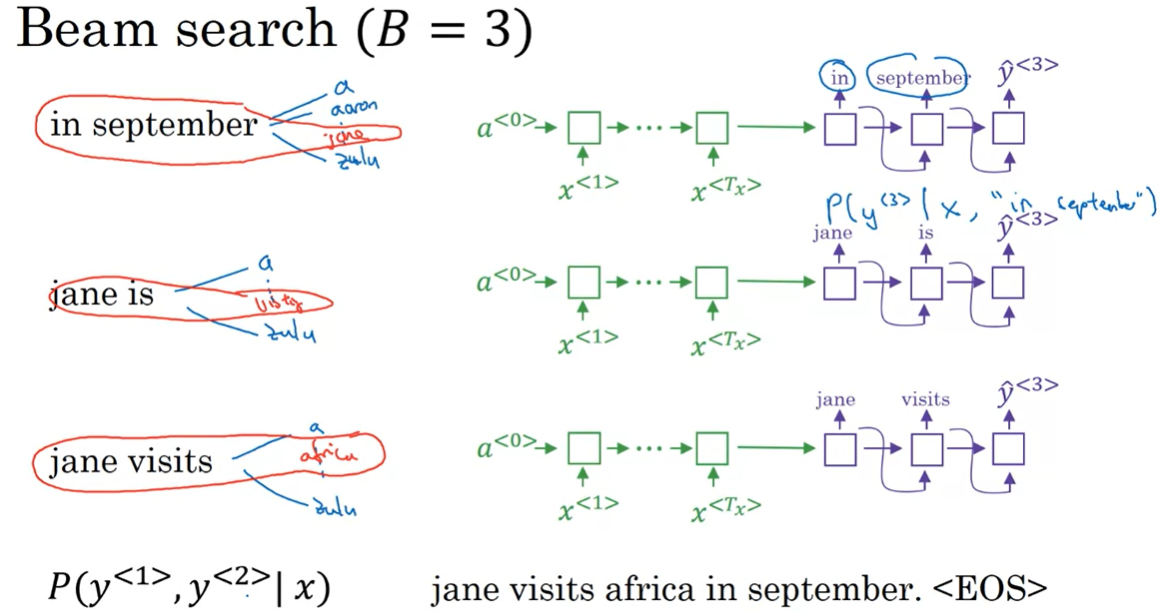
### Beam search is the ideal search algorithm for sequence-to-sequence models :



* Beam search is a heuristic algorithm , it means that he will not give assure us the best solution , but it guarantees us a good enough solution
* For the step 1 : instead of picking a single word y<1> that maximizes P( y<1> | X ) , we are goanna pick B=3 words that have the most big probabilities ( If B=1 then we are doing the greedy search approach ) , we found that the max 3 probabilities goes to “In” ,”Jane’ , ‘September’
* For the step 2: we instantiate 3 copies of our RNN model , the first one will be fed by y<1>=”In” , the second one by y<1>=”Jane”, and the third one by y<1> =”September” . And then for each y<1> of these three words : we calculate the probabilities ( 3\*10000 probabilities ) of P(y<2> | X , y<1> ) and we will pass to the 3rd step with the y<1> , y<2> which has the 3 best probabilities

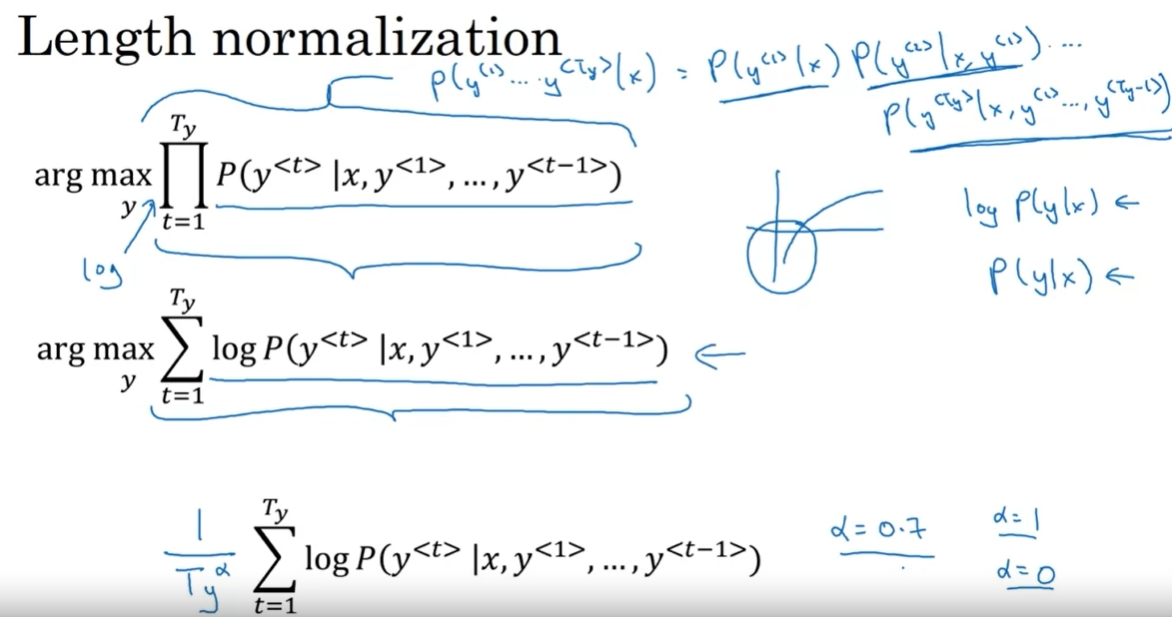


* We found that the best probabilities goes to y<1>,y<2> = “In September” , “Jane is” and “Jane visits”



* And we continue with same approach ( calculating B\*vocab\_size probabilities ) until we get <EOS> a token which defines the end of Sentence

#### Refining the Beam Search algorithm using Length normalization :



* Explaining the problem with Beam search calculations : By calculating P ( y<1> , y<2> , …y<t> | X ) we are doing a multiplication of P(y<1> | X ) \* P(y<2> | X , y<1> ) \*… P( y<t> | X , y<1> , y<2> , ….y<t> ) ( look to the first equation ) : and since 0<P(x) <1 … Doing several multiplications will make the result goes to a very tiny values so close to 0 that the computer may find hard to store their values …. So We will apply the Log() that will transform the multiplication of values between 0 and 1 into a sum of negative values ( < log(1) = 0 ) and we will take the max value ( the most closer one to zero ) ( look to the second equation )
* We can refine the calculations more by doing the normalization, by dividing the sum of logs by the length of the output Ty to get the average ( look to the third equation ) or replacing Ty by Tyalpha  with alpha between 0 and 1 ( alpha =0 => no normalization and alpha =1 => full normalization )

#### Choosing the right Beam Width B :

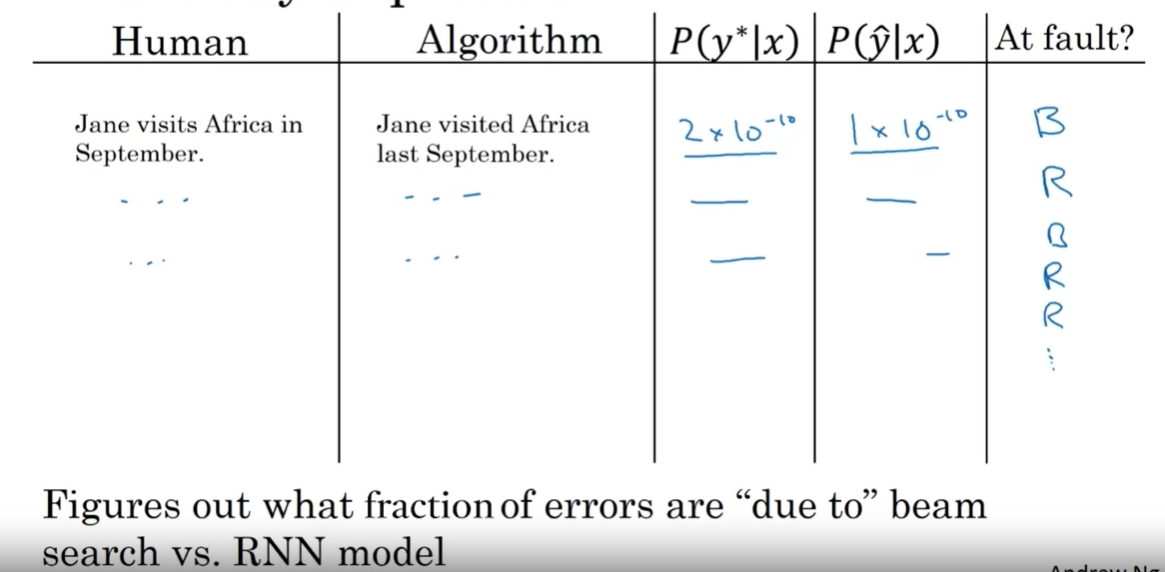
* Choosing a little B ( 10 or less ) will make the calculations faster but will give us a not very good solution
* Choosing a big B will slow the calculations because there is many possibilities to compute for each iterations but It will give us a more precise solution in the end

#### Error analysis : Our model didn’t predict well , is it because the RNN architecture or because the Beam Search with a wrong B :

#### 

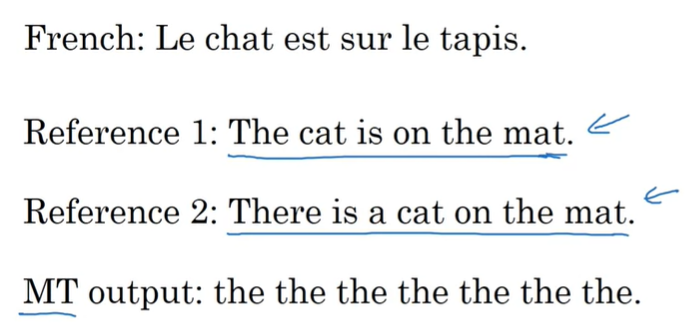
* The role of our RNN is to calculate and define P( Y | X ) and the role of the beaml search algorithm to find the Y that gives the max ( or a very high ) value of P( X | Y)
* If we note the perfect translation of our input “Jane visite l’Afrique en Septembre” by y\* and a not good translation by y^ we will fall in one of two unique cases :
  + Case 1 : our RNN model gives us P(y\*|X) > P(y^|X) but we got at the end as output y^ :
    - We notice that RNN did his job correctly by favorizing and maximizing P( y\* | X ) over the wrong translations like y^ so we conclude that **the fault is in the beam search algorithm**
  + Case 2 : our RNN model gives us P(y\*|X) < P(y^|X)
    - We notice that the RNN favorizes the wrong translation y^ over the perfect one y\* , so it’s not the Beam Search algorithm fault to gives us as an output y^ so we conclude that **the fault is in our RNN architecture**

#### The error analysis Process :



* We iterate through our training set and particularly for the inputs where we got a wrong translations , we calculate and compare the values of P(y\*|X ) and p( y^|X ) and conclude for each sequence wither it’s Beam search algorithm B fault or the RNN architecture fault R , we compare the two proportions and we decide in which part we gonna focus our ameliorations to have a better prediction later on

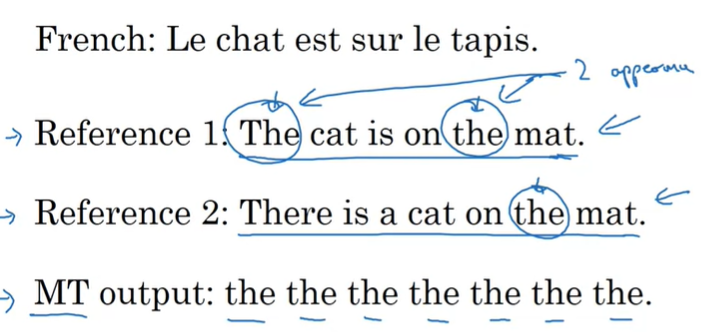
## Evaluating the machine translations using the BLEU Score:



Let’s say we have in the training set a French sentcne “ le chat est sur le tapis” and its corresponding human perfect translations “The cat is on the mat” and “There is a cat on the mat” , if our model output is “the the the the the the the” … How can we know mathemathically it’s a wrong output with a very low precision close to 0 ?

The solution is : **BLEU ( Bilingual evaluation understudy ) Score**

### Understanding Bleu score : calculating unigrams bleu score ( P1 ) :

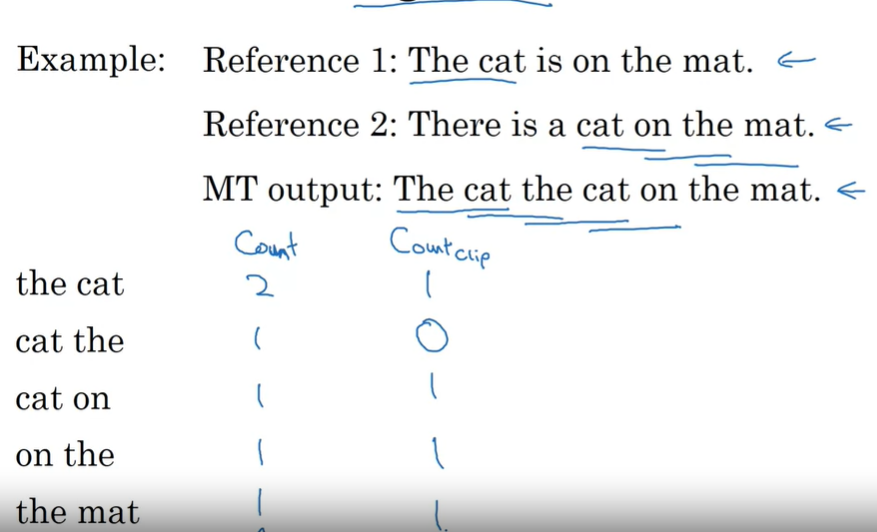


For each distinct word w in the MT output ( which is only “the” in this case) , we calculate the ratio CountClip(w)/Count(w) and we do the sum

* Count : is the number of occurrences of the word in the Mt output ( = 7 in this case )
* Count Clip : is the maximum number of appearance of the word in the Human translations : in this example Count Clip(“the”) = Max(Ref1Count(‘the’) , Ref2Count(‘the’) ) = Max(2,1) = 2

The final precision is **2/7**  which is quite far from the perfect one ( 1) so we conclude that we have a bad translation

### Understanding Bleu score : calculating bigrams bleu score ( P2 ) :

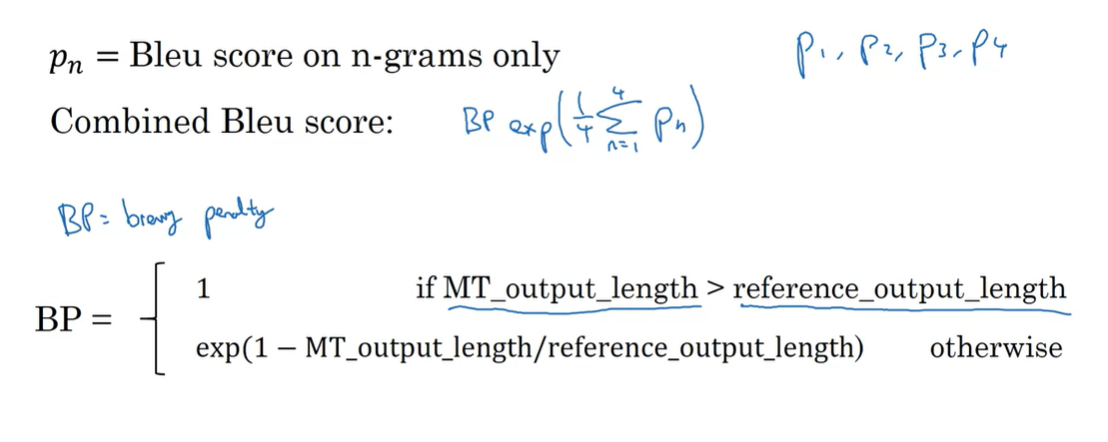


For the bigrams , Instead of picking by a single word , we pick all the possible combinations of 2 of the closest words and we count their Count and their CountClip

* For w=”The cat” for example , its count is 2 in the MT output and its count clip is Max(1,1)=1 ( because “the cat” exists in both human references )

We get in the end precision equals to 4/6 for “The car the cat on the mat” which is bigger than the ones we got with “ the the …the” and it makes a sense !

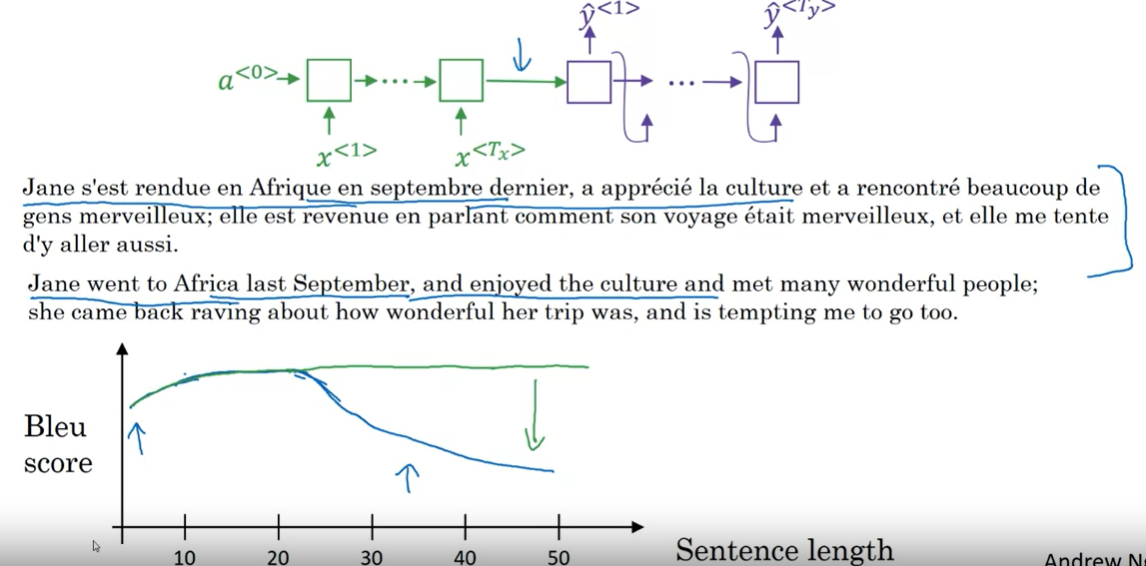
### The final Formula of Bleu Score :

- 

* We will Sum P1 , P2 , P3 and P4 and we apply the exponential function to the sum
* We multiply the result by a function called BP : Brevity Penalty
  + The Brevity Penalty function role is reduce the score for the short MT translations since it’s easy to get a high score for the short MT outputs even if it’s a wrong translation

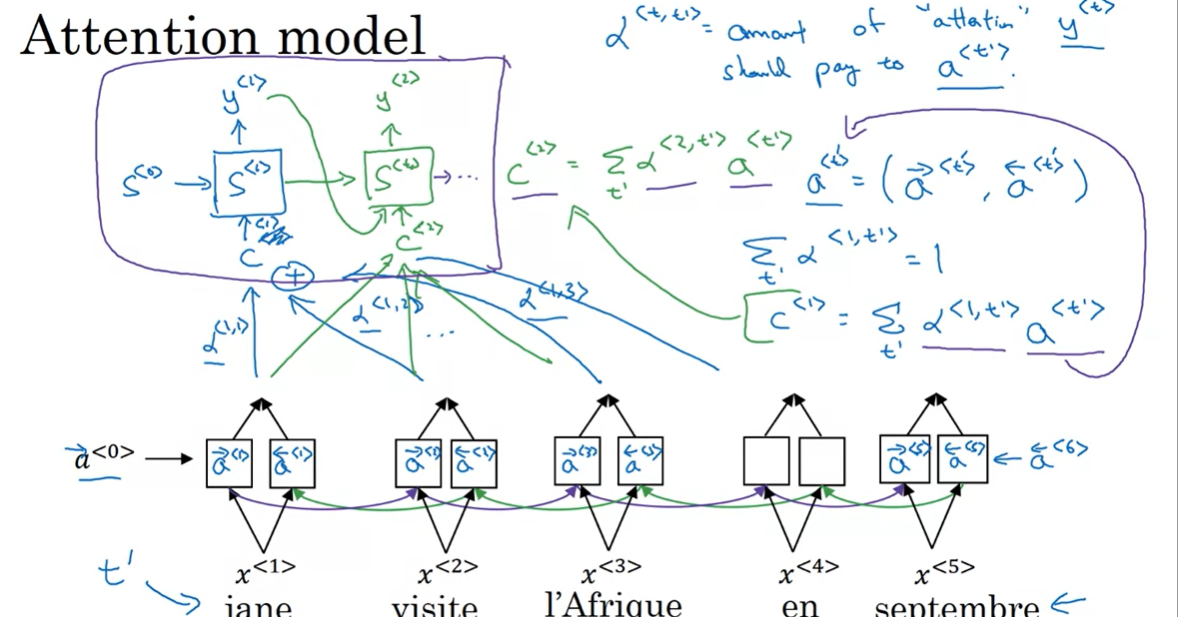
## Attention Model:

### Understanding the problem with long sequences:



If we want for example translate this three lines long French sentence , it would be so hard for the human brain to memorize the whole sequence right before doing the translation , The human translator will just try to do the translations part by part , for each part there is just some words that he have to keep in his mind ( which are the context ) for the given part , And we want to do the same approach with our RNN model because it’s proven that it has a lower bleu score for the sentences with a big length like it’s shown In the graph

### Attention Model Architecture:



* We are going to have 2 RNN models:
  + The first one (the bottom of the picture):
    - Its role is to calculate the attention weights α(i , j ) as outputs
      * α (i, j): how much you should give attention to the jth input word ( X ) while generating the ith  outputword (Y )
    - it’s a BRNN because we have to consider the whole sequence and not just the previous words while calculating the attention weights
    - we give a notation a<t> to represent the forward and backward hidden states ( a<t>-> , a<t><-)
  + The second one (the top of the picture) :
    - It’s the one who will generate the output (The Machine translation)
    - It has its own hidden states, we notate him by S<t> to distinguish it from the first model’s hidden state
    - It takes as an input (its x<t>) the different attention weights related to α(t , j ) ( j from 1 to Tx ) , they all represent the context c<t> ( which is x<t> for RNN ) for the output y<t> and the hidden state S<t-1>

### Formulas to calculate α <t,t’>

