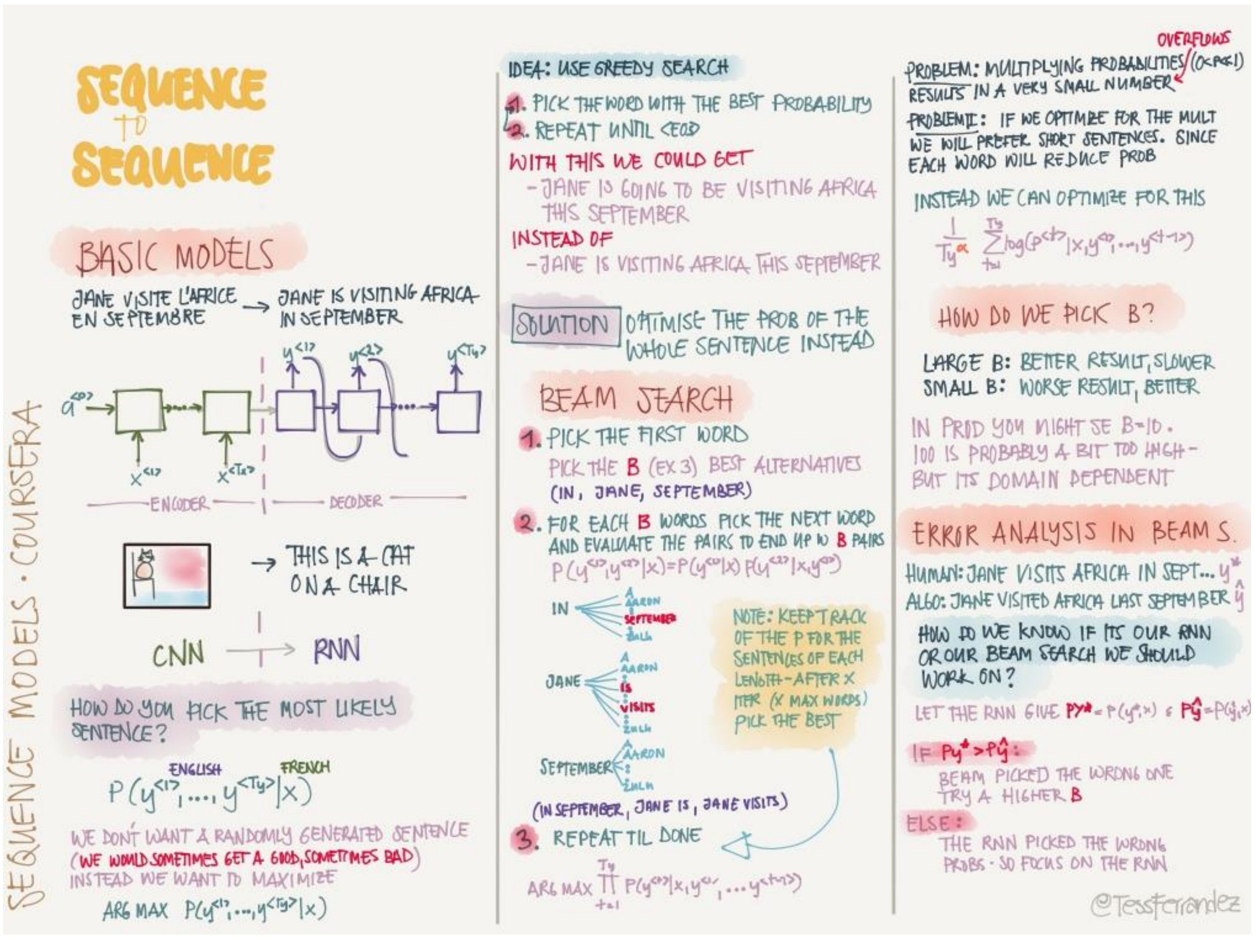
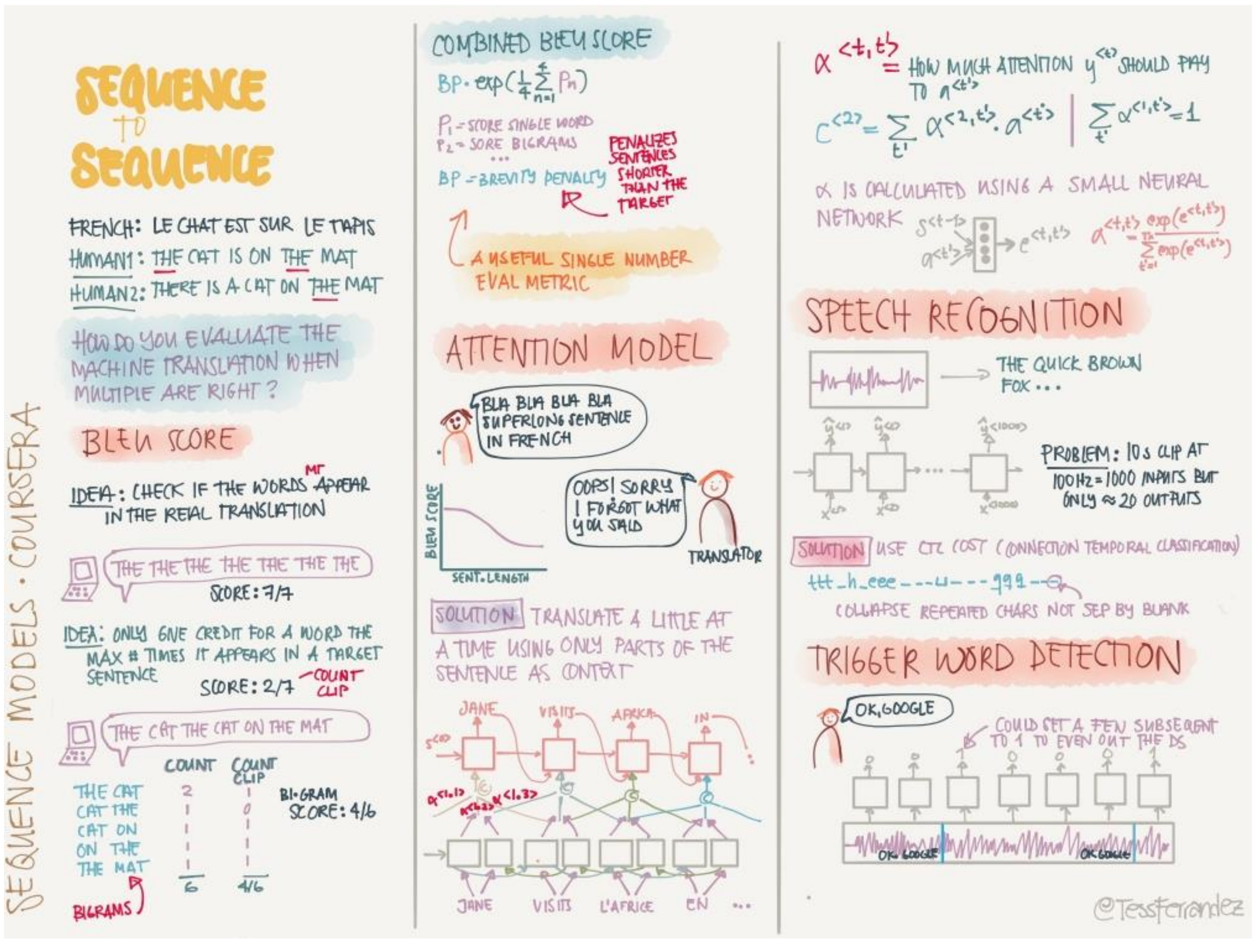


SEQUENCE TO SEQUENCE
SEQUENCE MODELS · COURSERA
BASIC MODELS
JANE VISITE L'AFRICE EN SEPTEMBRE → JANE IS VISITING AFRICA IN SEPTEMBER
$a^{<0>}$
$x^{<1>}$
$x^{<T_x>}$
$y^{<1>}$
$y^{<2>}$
$y^{<T_y>}$
ENCODER
DECODER
THIS IS A CAT ON A CHAIR
CNN → RNN
HOW DO YOU PICK THE MOST LIKELY SENTENCE?
ENGLISH
FRENCH
$P(y^{<1>},\dots,y^{<T_y>}|x)$
WE DON'T WANT A RANDOMLY GENERATED SENTENCE
(WE WOULD SOMETIMES GET A GOOD, SOMETIMES BAD)
INSTEAD WE WANT TO MAXIMIZE
ARG MAX $P(y^{<1>},\dots,y^{<T_y>}|x)$
IDEA: USE GREEDY SEARCH
1. PICK THE WORD WITH THE BEST PROBABILITY
2. REPEAT UNTIL <EOS>
WITH THIS WE COULD GET
- JANE IS GOING TO BE VISITING AFRICA THIS SEPTEMBER
INSTEAD OF
- JANE IS VISITING AFRICA THIS SEPTEMBER
SOLUTION
OPTIMISE THE PROB OF THE WHOLE SENTENCE INSTEAD
BEAM SEARCH
1. PICK THE FIRST WORD
PICK THE B (EX 3) BEST ALTERNATIVES
(IN, JANE, SEPTEMBER)
2. FOR EACH B WORDS PICK THE NEXT WORD AND EVALUATE THE PAIRS TO END UP W B PAIRS
$P(y^{<1>},y^{<2>}|x)=P(y^{<1>}|x)\,P(y^{<2>}|x,y^{<1>})$
IN
A
AARON
SEPTEMBER
ZULU
JANE
A
AARON
IS
VISITS
ZULU
SEPTEMBER
AARON
ZULU
NOTE: KEEP TRACK OF THE P FOR THE SENTENCES OF EACH LENGTH - AFTER X ITER (X MAX WORDS) PICK THE BEST
(IN SEPTEMBER, JANE IS, JANE VISITS)
3. REPEAT TIL DONE
ARG MAX $\prod_{t=1}^{T_y} P(y^{<t>}|x,y^{<1>},\dots y^{<t-1>})$
OVERFLOWS
PROBLEM: MULTIPLYING PROBABILITIES (0<P≤1) RESULTS IN A VERY SMALL NUMBER
PROBLEM II: IF WE OPTIMIZE FOR THE MULT WE WILL PREFER SHORT SENTENCES. SINCE EACH WORD WILL REDUCE PROB
INSTEAD WE CAN OPTIMIZE FOR THIS
$\frac{1}{T_y^{\alpha}}\sum_{t=1}^{T_y}\log(P^{<t>}|x,y^{<1>},\dots,y^{<t-1>})$
HOW DO WE PICK B?
LARGE B: BETTER RESULT, SLOWER
SMALL B: WORSE RESULT, BETTER
IN PROD YOU MIGHT SE B=10. 100 IS PROBABLY A BIT TOO HIGH - BUT ITS DOMAIN DEPENDENT
ERROR ANALYSIS IN BEAM S.
HUMAN: JANE VISITS AFRICA IN SEPT... y^*
ALGO: JANE VISITED AFRICA LAST SEPTEMBER $\hat{y}$
HOW DO WE KNOW IF ITS OUR RNN OR OUR BEAM SEARCH WE SHOULD WORK ON?
LET THE RNN GIVE $Py^* = P(y^*|x)$ & $P\hat{y} = P(\hat{y}|x)$
IF $Py^* > P\hat{y}$:
BEAM PICKED THE WRONG ONE TRY A HIGHER B
ELSE:
THE RNN PICKED THE WRONG PROBS - SO FOCUS ON THE RNN
@TessFerrandez


SEQUENCE TO SEQUENCE
SEQUENCE MODELS · COURSERA
FRENCH: LE CHAT EST SUR LE TAPIS
HUMAN1: THE CAT IS ON THE MAT
HUMAN2: THERE IS A CAT ON THE MAT
HOW DO YOU EVALUATE THE MACHINE TRANSLATION WHEN MULTIPLE ARE RIGHT?
BLEU SCORE
IDEA: CHECK IF THE WORDS MT APPEAR IN THE REAL TRANSLATION
THE THE THE THE THE THE THE
SCORE: 7/7
IDEA: ONLY GIVE CREDIT FOR A WORD THE MAX # TIMES IT APPEARS IN A TARGET SENTENCE
SCORE: 2/7
COUNT CLIP
THE CAT THE CAT ON THE MAT
COUNT
COUNT CLIP
THE CAT 2 1
CAT THE 1 0
CAT ON 1 1
ON THE 1 1
THE MAT 1 1
6
4/6
BI-GRAM SCORE: 4/6
BIGRAMS
COMBINED BLEU SCORE
BP · exp(1/4 Σ Pn)
P1 = SCORE SINGLE WORD
P2 = SCORE BIGRAMS
BP = BREVITY PENALTY
PENALIZES SENTENCES SHORTER THAN THE TARGET
A USEFUL SINGLE NUMBER EVAL METRIC
ATTENTION MODEL
BLA BLA BLA BLA SUPERLONG SENTENCE IN FRENCH
BLEU SCORE
SENT. LENGTH
OOPS! SORRY I FORGOT WHAT YOU SAID
TRANSLATOR
SOLUTION TRANSLATE A LITTLE AT A TIME USING ONLY PARTS OF THE SENTENCE AS CONTEXT
JANE VISITS AFRICA IN
JANE VISITS L'AFRIQUE EN ...
α<t,t'> = HOW MUCH ATTENTION y<t> SHOULD PAY TO a<t'>
c<2> = Σ α<2,t'> · a<t'> | Σ α<1,t'> = 1
α IS CALCULATED USING A SMALL NEURAL NETWORK
SPEECH RECOGNITION
THE QUICK BROWN FOX ...
PROBLEM: 10s CLIP AT 100Hz = 1000 INPUTS BUT ONLY ≈ 20 OUTPUTS
SOLUTION USE CTC COST (CONNECTION TEMPORAL CLASSIFICATION)
ttt_h_eee___ _ _ _qqq__
COLLAPSE REPEATED CHARS NOT SEP BY BLANK
TRIGGER WORD DETECTION
OK, GOOGLE
COULD SET A FEW SUBSEQUENT TO 1 TO EVEN OUT THE DS
OK GOOGLE
OK GOOGLE
@TessFerrandez

Sequence models & Attention mechanism

Sequence models can be augmented using an attention mechanism. This algorithm will help your model understand where it should focus its attention given a sequence of inputs. This week, you will also learn about speech recognition and how to deal with audio data.

Various sequence to sequence architectures

Basic Models

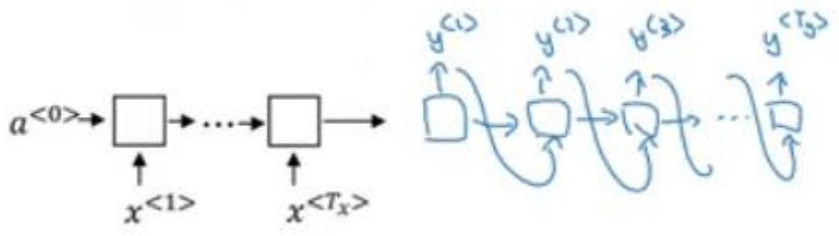
- In this section we will learn about sequence to sequence - *Many to Many* - models which are useful in various applications including machine translation and speech recognition.

- Let's start with the basic model:
  - Given this machine translation problem in which X is a French sequence and Y is an English sequence.

$$\overset{x^{<1>}}{\text{Jane}}\ \overset{x^{<2>}}{\text{visite}}\ \overset{x^{<3>}}{\text{l'Afrique}}\ \overset{x^{<4>}}{\text{en}}\ \overset{x^{<5>}}{\text{septembre}}$$

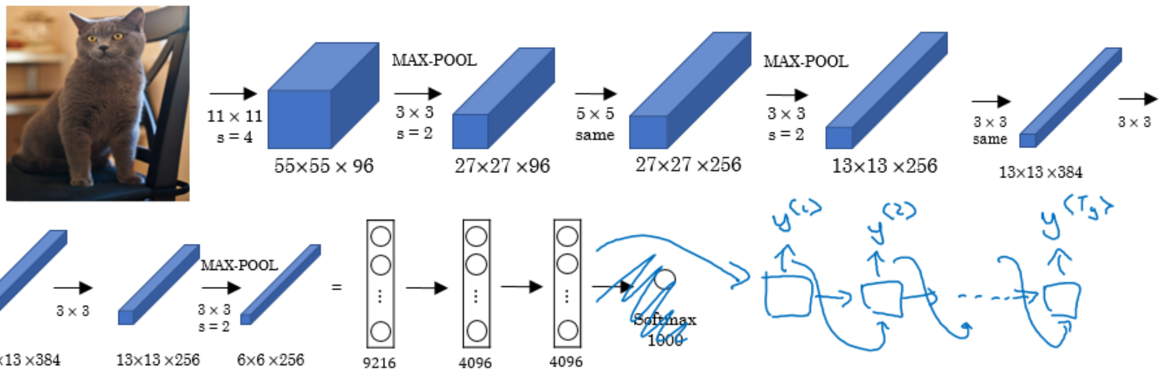
$$\longrightarrow \underset{y^{<1>}}{\text{Jane}}\ \underset{y^{<2>}}{\text{is}}\ \underset{y^{<3>}}{\text{visiting}}\ \underset{y^{<4>}}{\text{Africa}}\ \underset{y^{<5>}}{\text{in}}\ \underset{y^{<6>}}{\text{September.}}$$

  - Our architecture will include **encoder** and **decoder**.
  - The encoder is RNN - LSTM or GRU are included - and takes the input sequence and then outputs a vector that should represent the whole input.
  - After that the decoder network, also RNN, takes the sequence built by the encoder and outputs the new sequence.



  - These ideas are from the following papers:
    - Sutskever et al., 2014. Sequence to sequence learning with neural networks
    - Cho et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation
- An architecture similar to the mentioned above works for image captioning problem:
  - In this problem X is an image, while Y is a sentence (caption).

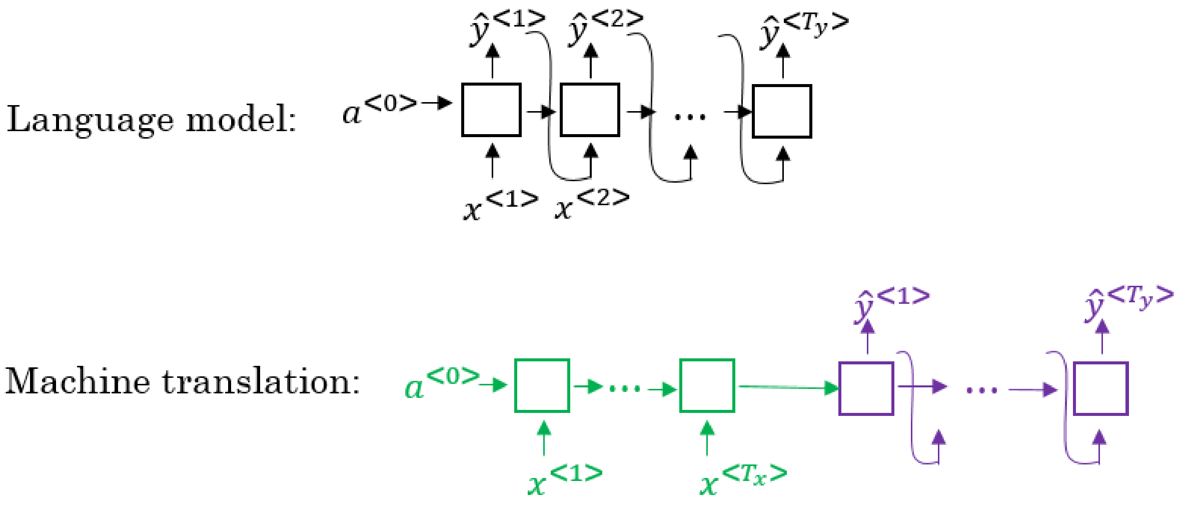
- The model architecture image:



- The architecture uses a pretrained CNN (like AlexNet) as an encoder for the image, and the decoder is an RNN.
- Ideas are from the following papers (they share similar ideas):
  - Maoet et. al., 2014. Deep captioning with multimodal recurrent neural networks
  - Vinyals et. al., 2014. Show and tell: Neural image caption generator
  - Karpathy and Li, 2015. Deep visual-semantic alignments for generating image descriptions

Picking the most likely sentence

- There are some similarities between the language model we have learned previously, and the machine translation model we have just discussed, but there are some differences as well.
- The language model we have learned is very similar to the decoder part of the machine translation model, except for $a^{<0>}$



- Problems formulations also are different:
  - In language model: $P(y^{<1>}, \dots, y^{<Ty>})$
  - In machine translation: $P(y^{<1>}, \dots, y^{<Ty>} \mid x^{<1>}, \dots, x^{<Tx>})$
- What we don't want in machine translation model, is not to sample the output at random. This may provide some choices as an output. Sometimes you may sample a bad output.
  - Example:
    - X = "Jane visite l'Afrique en septembre."
    - Y may be:
      - Jane is visiting Africa in September.
      - Jane is going to be visiting Africa in September.
      - In September, Jane will visit Africa.
- So we need to get the best output it can be:

$$\underset{y^{<1>},\dots,y^{<T_y>}}{\arg\max}\ P(y^{<1>}, \dots, y^{<T_y>} \mid x)$$

- The most common algorithm is the beam search, which we will explain in the next section.
- Why not use greedy search? Why not get the best choices each time?
  - It turns out that this approach doesn't really work!
  - Lets explain it with an example:
    - The best output for the example we talked about is "Jane is visiting Africa in September."
    - Suppose that when you are choosing with greedy approach, the first two words were "Jane is", the word that may come after that will be "going" as "going" is the most common word that comes after " is" so the result may look like this: "Jane is going to be visiting Africa in September.". And that isn't the best/optimal solution.
- So what is better than greedy approach, is to get an approximate solution, that will try to maximize the output (the last equation above).

Beam Search

- Beam search is the most widely used algorithm to get the best output sequence. It's a heuristic search algorithm.
- To illustrate the algorithm we will stick with the example from the previous section. We need Y = "Jane is visiting Africa in September."

- The algorithm has a parameter `B` which is the beam width. Lets take `B = 3` which means the algorithm will get 3 outputs at a time.
- For the first step you will get ["in", "jane", "september"] words that are the best candidates.
- Then for each word in the first output, get B next (second) words and select top best B combinations where the best are those what give the highest value of multiplying both probabilities - $P(y^{<1>}|x) * P(y^{<2>}|x,y^{<1>})$. Se we will have then ["in september", "jane is", "jane visit"]. Notice, that we automatically discard *september* as a first word.
- Repeat the same process and get the best B words for ["september", "is", "visit"] and so on.
- In this algorithm, keep only B instances of your network.
- If `B = 1` this will become the greedy search.

Refinements to Beam Search

- In the previous section, we have discussed the basic beam search. In this section, we will try to do some refinements to it.
- The first thing is **Length optimization**
  - In beam search we are trying to optimize:

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

  - And to do that we multiply:
    $P(y^{<1>} \mid x) * P(y^{<2>} \mid x, y^{<1>}) * \dots * P(y^{<t>} \mid x, y^{<y(t-1)>})$
  - Each probability is a fraction, most of the time a small fraction.
  - Multiplying small fractions will cause a **numerical overflow**. Meaning that it's too small for the floating part representation in your computer to store accurately.
  - So in practice we use **summing logs of probabilities** instead of multiplying directly.

    $$\arg\max_{y} \sum_{y=1}^{T_y} \log P(y^{<t>} \mid x, y^{<1>},\dots,y^{<t-1>})$$

  - But there's another problem. The two optimization functions we have mentioned are preferring small sequences rather than long ones. Because

multiplying more fractions gives a smaller value, so fewer fractions - bigger result.
  - So there's another step - dividing by the number of elements in the sequence.

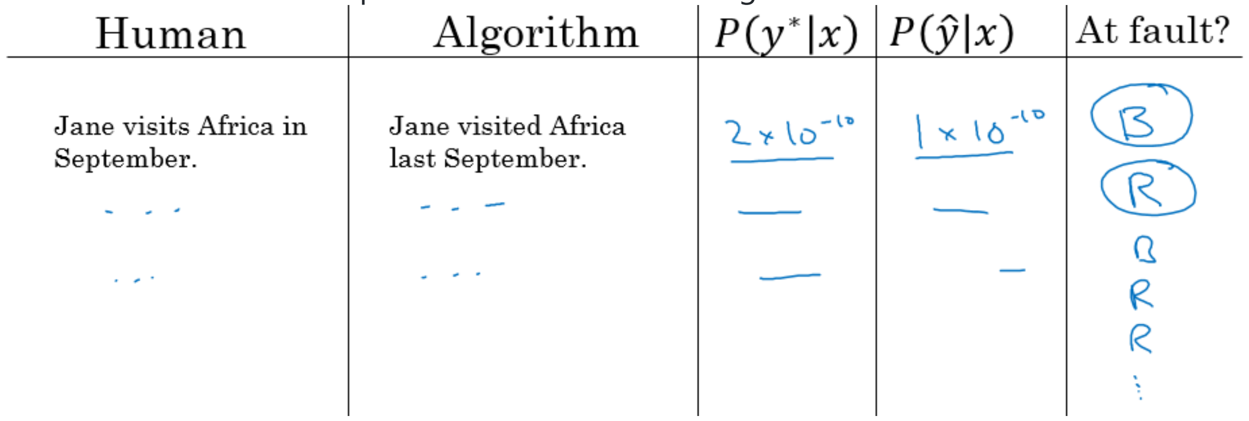
    $$\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \ldots, y^{<t-1>})$$

    - alpha is a hyperparameter to tune.
    - If alpha = 0 - no sequence length normalization.
    - If alpha = 1 - full sequence length normalization.
    - In practice alpha = 0.7 is a good thing (somewhere in between two extremes).
- The second thing is how can we choose best `B`?
  - The larger B - the larger possibilities, the better are the results. But it will be more computationally expensive.
  - In practice, you might see in the production setting `B=10`
  - `B=100`, `B=1000` are uncommon (sometimes used in research settings)
  - Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find the exact solution.

Error analysis in beam search

- We have talked before on **Error analysis** in "*Structuring Machine Learning Projects*" course. We will apply these concepts to improve our beam search algorithm.
- We will use error analysis to figure out if the `B` hyperparameter of the beam search is the problem (it doesn't get an optimal solution) or in our RNN part.
- Let's take an example:
  - Initial info:
    - x = "Jane visite l'Afrique en septembre."
    - y^* = "Jane visits Africa in September." - right answer
    - ŷ = "Jane visited Africa last September." - answer produced by model
  - Our model that has produced not a good result.
  - We now want to know who to blame - the RNN or the beam search.
  - To do that, we calculate $P(y^* | X)$ and $P(\hat{y} | X)$. There are two cases:

        - Case 1 (P(y* | X) > P(ŷ | X)):
            - Conclusion: Beam search is at fault.
        - Case 2 (P(y* | X) <= P(ŷ | X)):
            - Conclusion: RNN model is at fault.
- The error analysis process is as following:
    - You choose N error examples and make the following table:



Human	Algorithm	$P(y^*\|x)$	$P(\hat{y}\|x)$	At fault?
Jane visits Africa in September.	Jane visited Africa last September.	2×10^{-10}	1×10^{-10}	B
...	...	—	—	R
...	...	—	—	B
				R
				R
				⋮

    - B for beam search, R is for the RNN.
    - Get counts and decide what to work on next.

BLEU Score

- One of the challenges of machine translation, is that given a sentence in a language there are one or more possible good translation in another language. So how do we evaluate our results?
- The way we do this is by using **BLEU score**. BLEU stands for *bilingual evaluation understudy*.
- The intuition is: as long as the machine-generated translation is pretty close to any of the references provided by humans, then it will get a high BLEU score.
- Let's take an example:
    - X = "Le chat est sur le tapis."
    - Y1 = "The cat is on the mat." (human reference 1)
    - Y2 = "There is a cat on the mat." (human reference 2)
    - Suppose that the machine outputs: "the the the the the the the."
    - One way to evaluate the machine output is to look at each word in the output and check if it is in the references. This is called *precision*:
        - precision = 7/7 because "the" appeared in Y1 or Y2
    - This is not a useful measure!

  - We can use a modified precision in which we are looking for the reference with the maximum number of a particular word and set the maximum appearing of this word to this number. So:
    - modified precision = 2/7 because the max is 2 in Y1
    - We clipped the 7 times by the max which is 2.
  - Here we are looking at one word at a time - unigrams, we may look at n-grams too
- BLEU score on bigrams
  - The **n-grams** typically are collected from a text or speech corpus. When the items are words, **n-grams** may also be called shingles. An **n-gram** of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram".
  - X = "Le chat est sur le tapis."
  - Y1 = "The cat is on the mat."
  - Y2 = "There is a cat on the mat."
  - Suppose that the machine outputs: "the cat the cat on the mat."
  - The bigrams in the machine output:

Pairs	Count	C
the cat	2	1 (Y1)
cat the	1	0
cat on	1	1 (Y2)
on the	1	1 (Y1)
the mat	1	1 (Y1)
Totals	6	4

  - Modified precision = sum(Count clip) / sum(Count) = 4/6

- So here are the equations for modified precision for the n-grams case:

$$p_1 = \frac{\sum_{unigram \in \hat{y}} count_{clip}\,(unigram)}{\sum_{unigram \in \hat{y}} count\,(unigram)} \qquad p_n = \frac{\sum_{ngram \in \hat{y}} count_{clip}\,(ngram)}{\sum_{ngram \in \hat{y}} count\,(ngram)}$$

- Let's put this together to formalize the BLEU score:
  - **P_n** = Bleu score on one type of n-gram
  - **Combined BLEU score** = BP * exp(1/n * sum(P_n))
    - For example if we want BLEU for 4, we compute P_1, P_2, P_3, P_4 and then average them and take the exp.
  - **BP** is called **BP penalty** which stands for brevity penalty. It turns out that if a machine outputs a small number of words it will get a better score so we need to handle that.

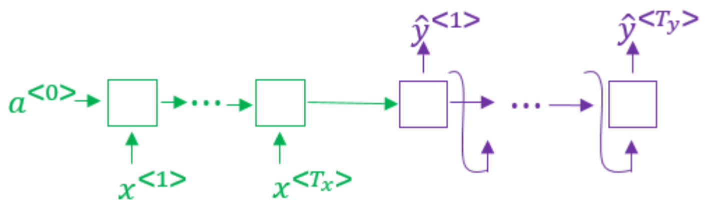
$$\text{BP} = \begin{cases} 1 & \text{if MT\_output\_length > reference\_output\_length} \\ \exp(1 - \text{MT\_output\_length}/\text{reference\_output\_length}) & \text{otherwise} \end{cases}$$

- BLEU score has several open source implementations.
- It is used in a variety of systems like machine translation and image captioning.

Attention Model Intuition

- So far we were using sequence to sequence models with an encoder and decoders. There is a technique called *attention*which makes these models even better.
- The attention idea has been one of the most influential ideas in deep learning.
- The problem of long sequences:

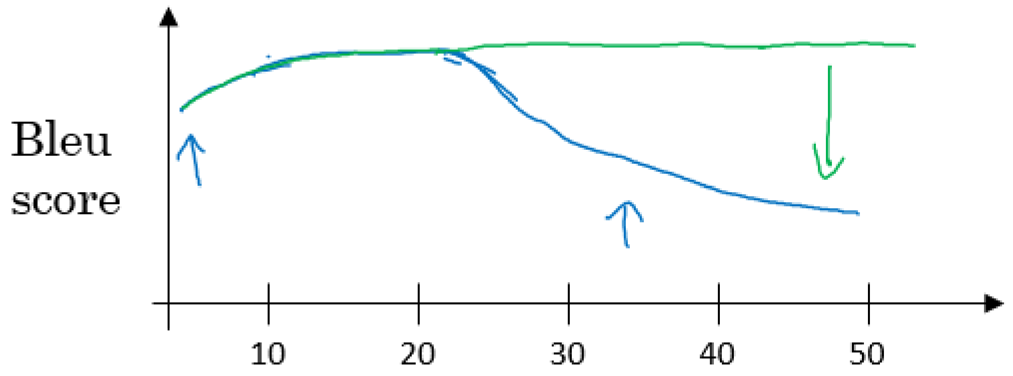
- Given this model, inputs, and outputs.



Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

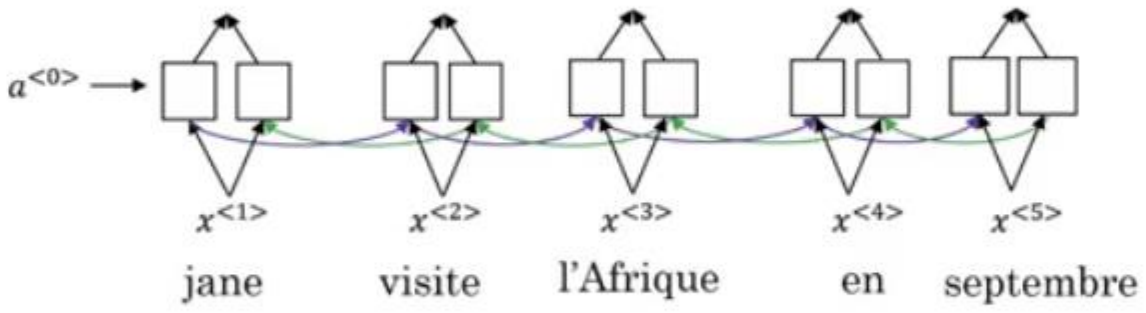
- The encoder should memorize this long sequence into one vector, and the decoder has to process this vector to generate the translation.
- If a human would translate this sentence, he/she wouldn't read the whole sentence and memorize it then try to translate it. He/she translates a part at a time.
- The performance of this model decreases if a sentence is long.
- We will discuss the attention model that works like a human that looks at parts at a time. That will significantly increase the accuracy even with longer sequence:



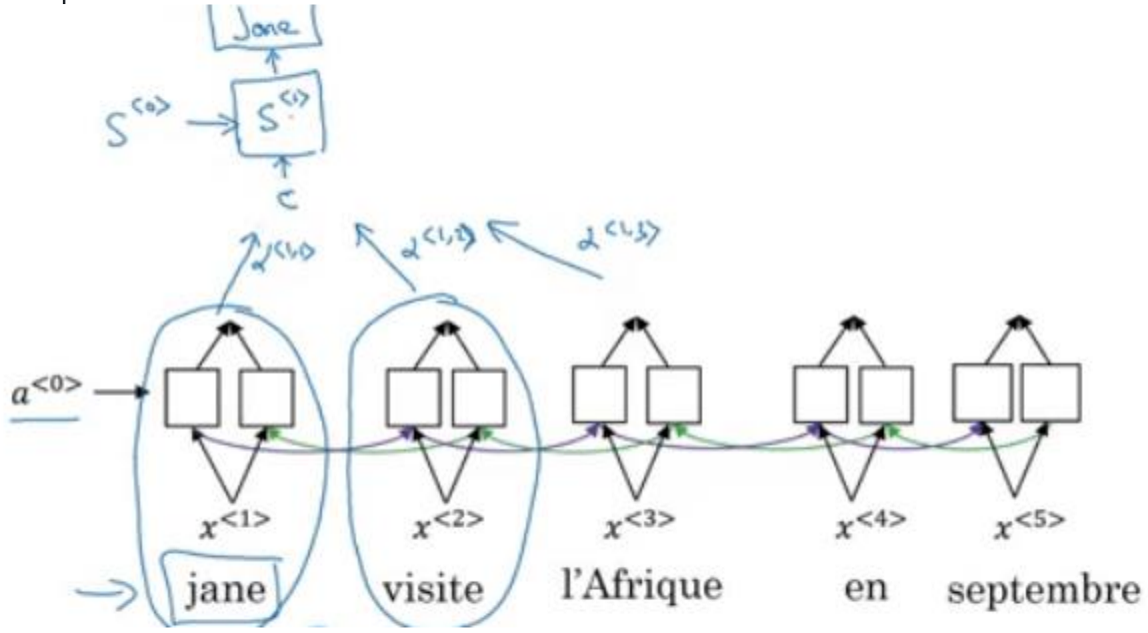
  - Blue is the normal model, while green is the model with attention mechanism.

- In this section we will give just some intuitions about the attention model and in the next section we will discuss it's details.
- At first the attention model was developed for machine translation but then other applications used it like computer vision and new architectures like Neural Turing machine.
- The attention model was descried in this paper:
  - Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate

- Now for the intuition:
    - Suppose that our encoder is a bidirectional RNN:

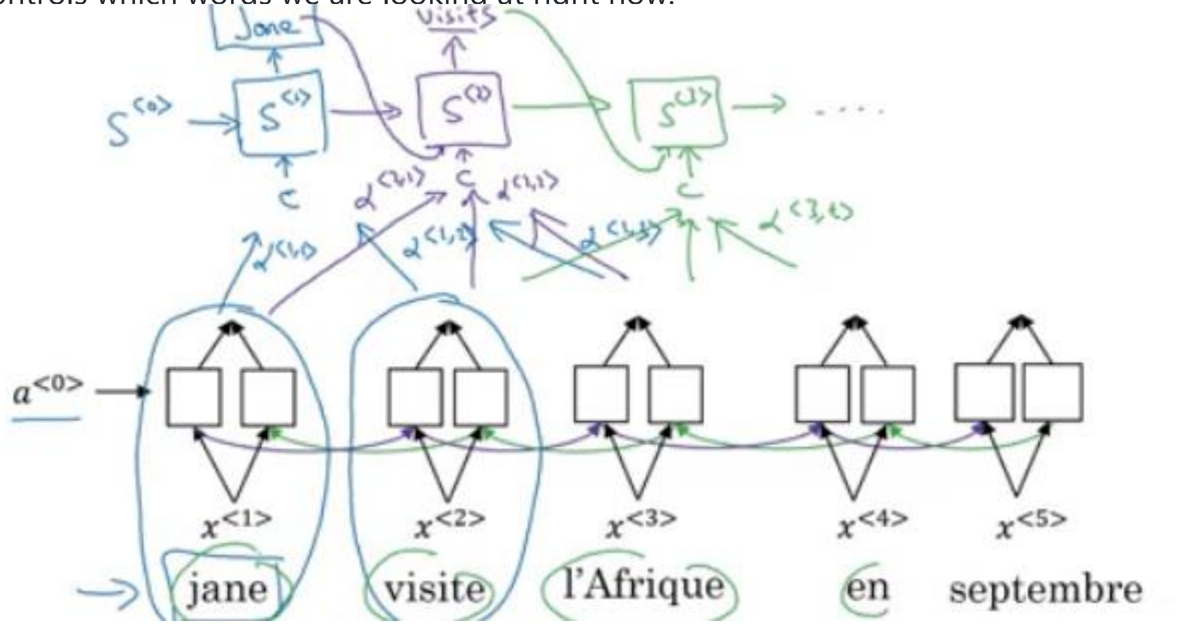


    - We give the French sentence to the encoder and it should generate a vector that represents the inputs.
    - Now to generate the first word in English which is "Jane" we will make another RNN which is the decoder.
    - Attention weights are used to specify which words are needed when to generate a word. So to generate "jane" we will look at "jane", "visite", "l'Afrique"



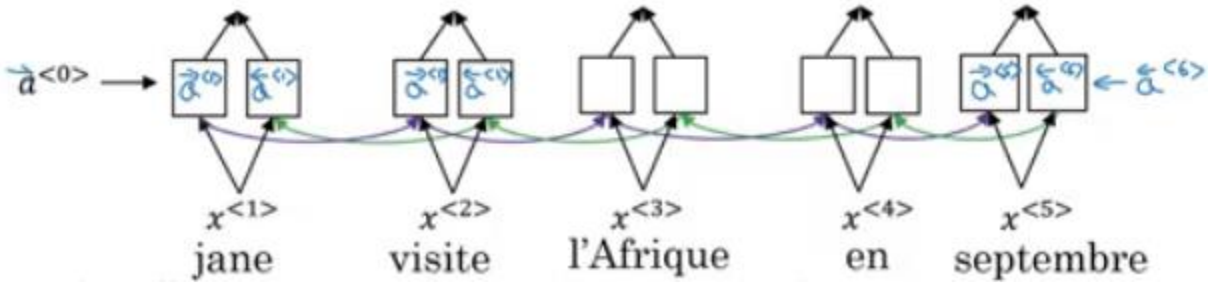
    - alpha$^{<1,1>}$, alpha$^{<1,2>}$, and alpha$^{<1,3>}$ are the attention weights being used.

- And so to generate any word there will be a set of attention weights that controls which words we are looking at right now.



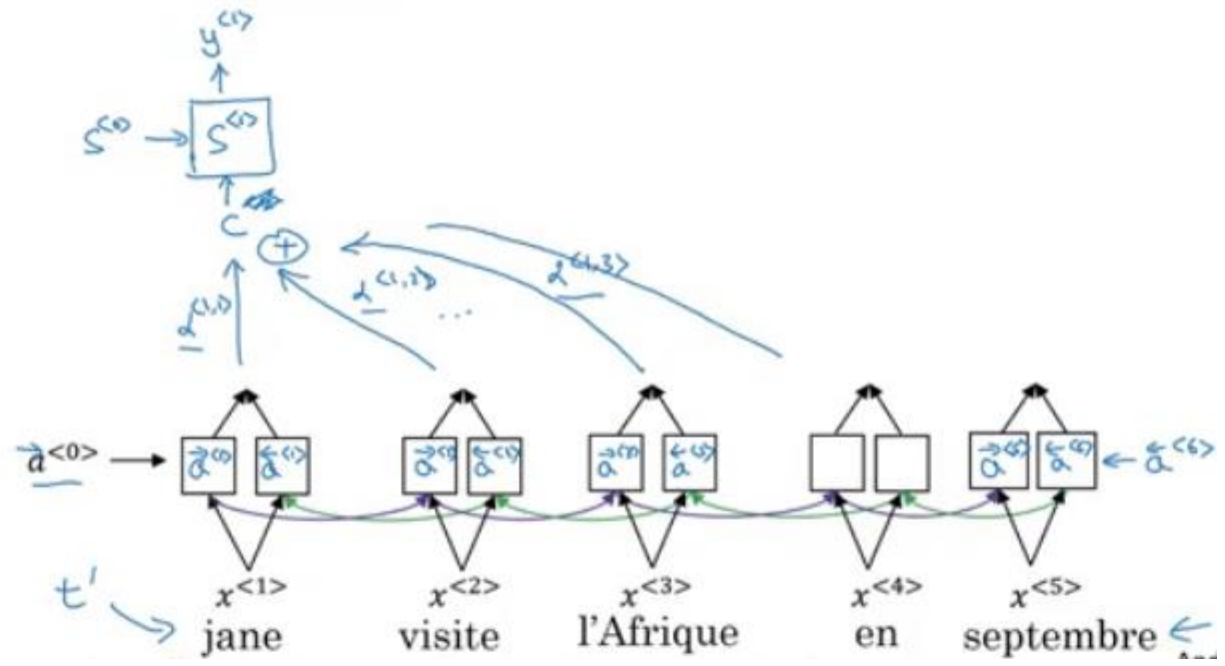
Attention Model

- Lets formalize the intuition from the last section into the exact details on how this can be implemented.
- First we will have an bidirectional RNN (most common is LSTMs) that encodes French language:



- For learning purposes, lets assume that $a^{<t'>}$ will include the both directions activations at time step t'.
- We will have a unidirectional RNN to produce the output using a context c which is computed using the attention weights, which denote how much information

does the output needs to look in $a^{<t'>}$



- Sum of the attention weights for each element in the sequence should be 1:

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

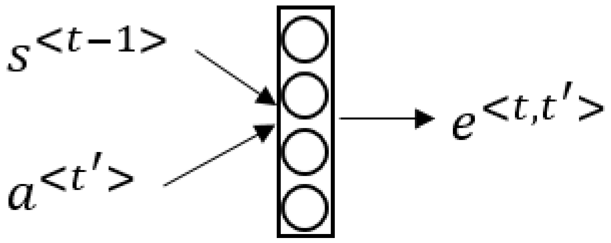
- The context c is calculated using this equation:

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

- Lets see how can we compute the attention weights:
  - So alpha$^{<t, t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$
    - Like for example we payed attention to the first three words through alpha$^{<1,1>}$, alpha$^{<1,2>}$, alpha$^{<1,3>}$
  - We are going to softmax the attention weights so that their sum is 1:

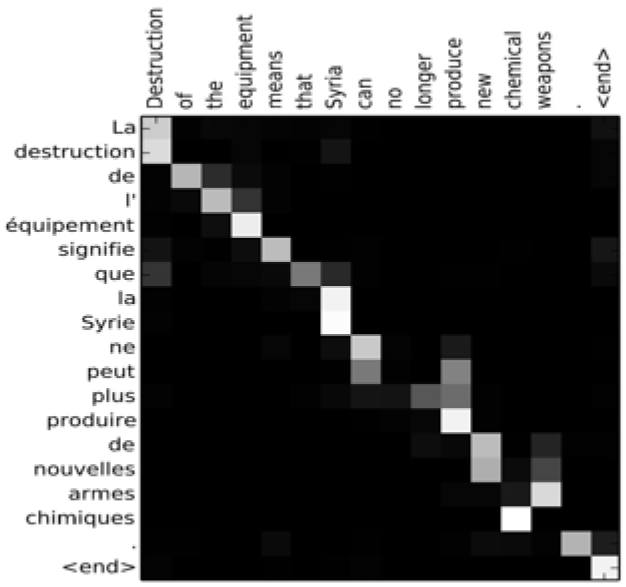
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{Tx} \exp(e^{<t,t'>})}$$

  - Now we need to know how to calculate $e^{<t, t'>}$. We will compute e using a small neural network (usually 1-layer, because we will need to compute this a lot):



- $s^{<t-1>}$ is the hidden state of the RNN s, and $a^{<t'>}$ is the activation of the other bidirectional RNN.

- One of the disadvantages of this algorithm is that it takes quadratic time or quadratic cost to run.
- One fun way to see how attention works is by visualizing the attention weights:

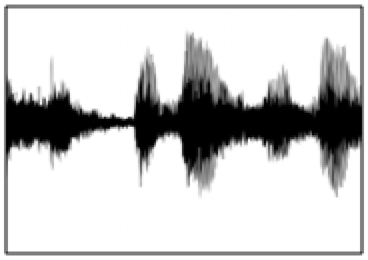


Speech recognition - Audio data

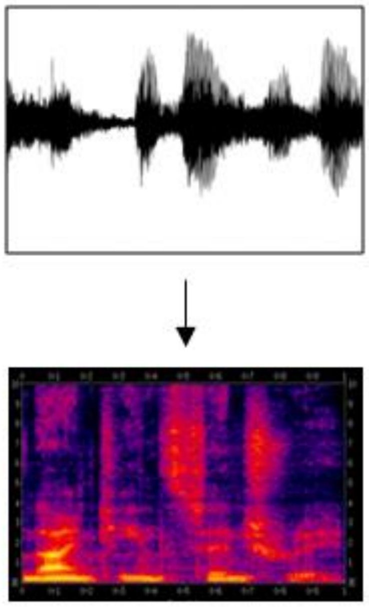
Speech recognition

- One of the most exciting developments using sequence-to-sequence models has been the rise of very accurate speech recognition.
- Let's define the speech recognition problem:
  - X: audio clip
  - Y: transcript

- If you plot an audio clip it will look like this:



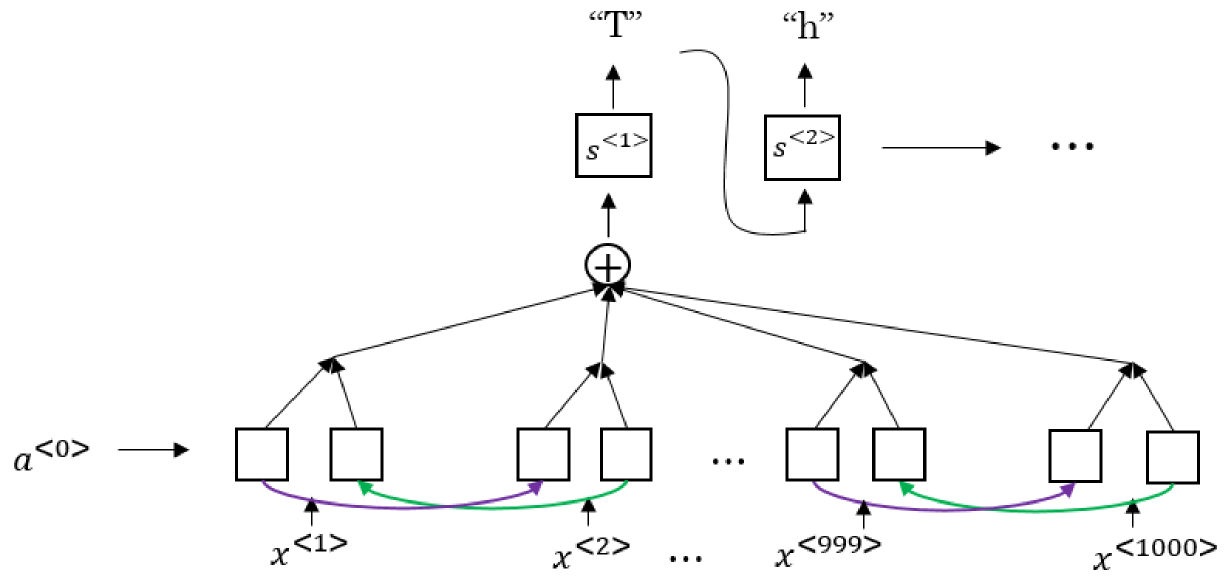
  - The horizontal axis is time while the vertical is changes in air pressure.
- What really is an audio recording? A microphone records little variations in air pressure over time, and it is these little variations in air pressure that your ear perceives as sound. You can think of an audio recording is a long list of numbers measuring the little air pressure changes detected by the microphone. We will use audio sampled at 44100 Hz (or 44100 Hertz). This means the microphone gives us 44100 numbers per second. Thus, a 10 second audio clip is represented by 441000 numbers (= 10 * 44100).
- It is quite difficult to work with "raw" representation of audio.
- Because even human ear doesn't process raw wave forms, the human ear can process different frequencies.
- There's a common preprocessing step for an audio - generate a spectrogram which works similarly to human ears.



  - The horizontal axis is time while the vertical is frequencies. Intensity of different colors shows the amount of energy - how loud is the sound for different frequencies (a human ear does a very similar preprocessing step).

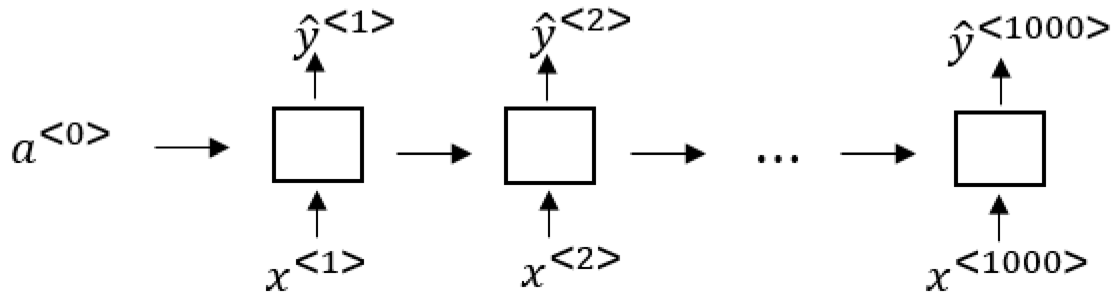
- A spectrogram is computed by sliding a window over the raw audio signal, and calculates the most active frequencies in each window using a Fourier transformation.
- In the past days, speech recognition systems were built using *phonemes* that are a hand engineered basic units of sound. Linguists used to hypothesize that writing down audio in terms of these basic units of sound called *phonemes* would be the best way to do speech recognition.
- End-to-end deep learning found that phonemes was no longer needed. One of the things that made this possible is the large audio datasets.
- Research papers have around 300 - 3000 hours of training data while the best commercial systems are now trained on over 100,000 hours of audio.

- You can build an accurate speech recognition system using the attention model that we have descried in the previous section:



- One of the methods that seem to work well is *CTC cost* which stands for "Connectionist temporal classification"
  - To explain this let's say that Y = "the quick brown fox"

- We are going to use an RNN with input, output structure:



- Note: this is a unidirectional RNN, but in practice a bidirectional RNN is used.
- Notice, that the number of inputs and number of outputs are the same here, but in speech recognition problem input X tends to be a lot larger than output Y.
  - 10 seconds of audio at 100Hz gives us X with shape (1000,). These 10 seconds don't contain 1000 character outputs.
- The CTC cost function allows the RNN to output something like this:
  - `ttt_h_eee<SPC>___<SPC>qqq___` - this covers "the q".
  - The _ is a special character called "blank" and `<SPC>` is for the "space" character.
  - Basic rule for CTC: collapse repeated characters not separated by "blank"
- So the 19 character in our Y can be generated into 1000 character output using CTC and it's special blanks.
- The ideas were taken from this paper:
  - Graves et al., 2006. Connectionist Temporal Classification: Labeling unsegmented sequence data with recurrent neural networks
  - This paper's ideas were also used by Baidu's DeepSpeech.

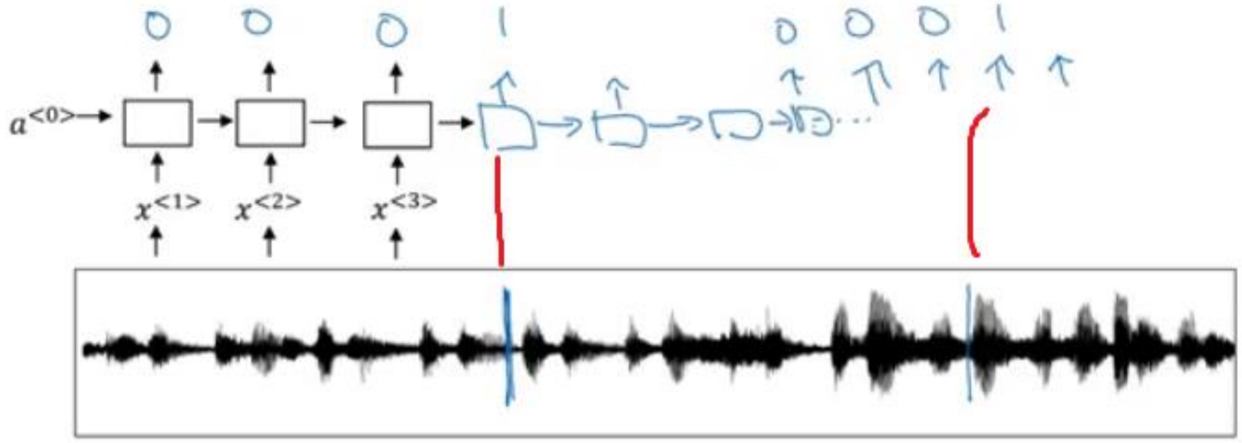
- Using both attention model and CTC cost can help you to build an accurate speech recognition system.

Trigger Word Detection

- With the rise of deep learning speech recognition, there are a lot of devices that can be waked up by saying some words with your voice. These systems are called trigger word detection systems.
- For example, Alexa - a smart device made by Amazon - can answer your call "Alexa, what time is it?" and then Alexa will respond to you.

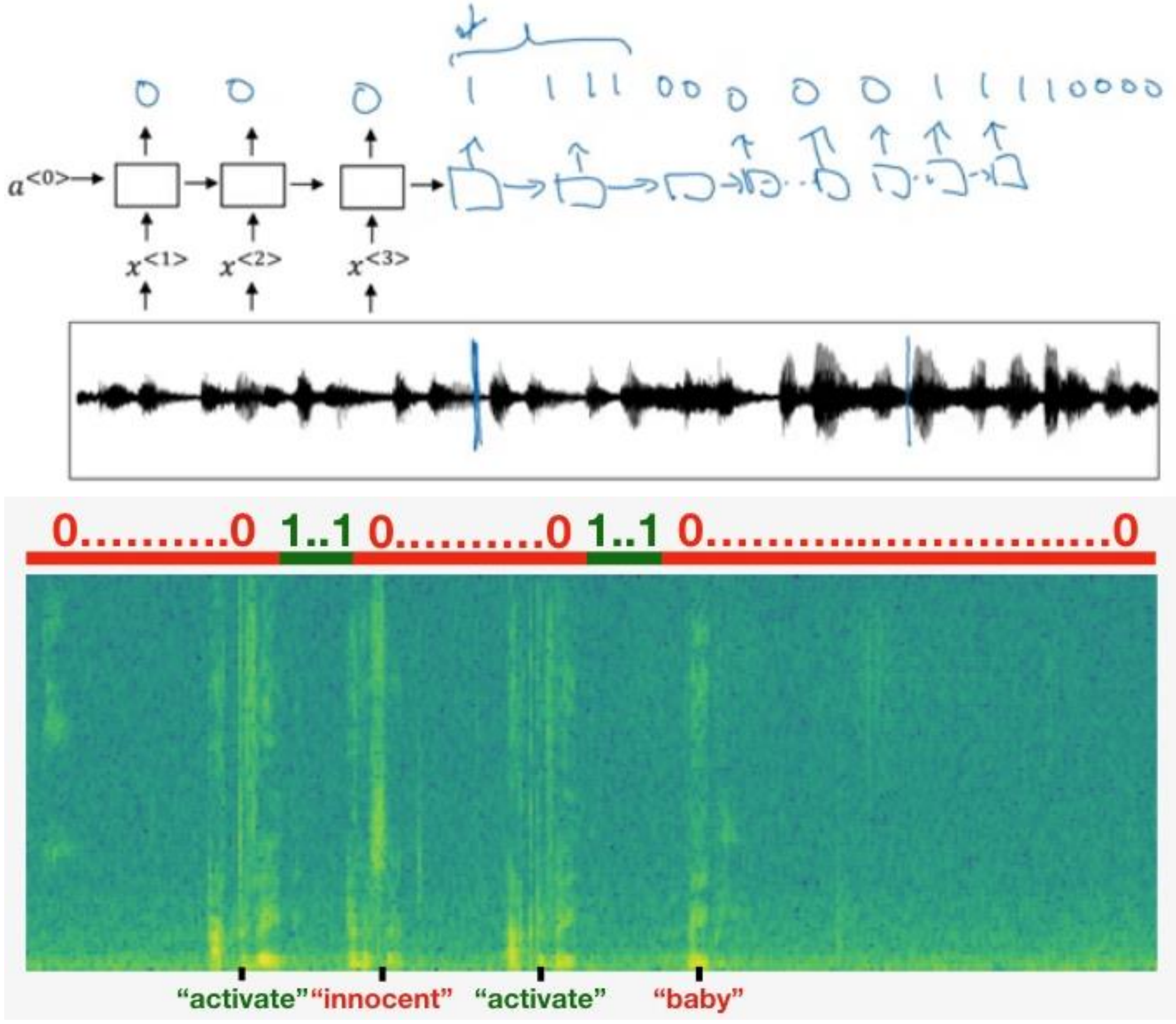
- Trigger word detection systems include:

- For now, the trigger word detection literature is still evolving so there actually isn't a single universally agreed on the algorithm for trigger word detection yet. But let's discuss an algorithm that can be used.
- Let's now build a model that can solve this problem:
  - X: audio clip
  - X has been preprocessed and spectrogram features have been returned of X
    - $X^{<1>}, X^{<2>}, \ldots, X^{<t>}$
  - Y will be labels 0 or 1. 0 represents the non-trigger word, while 1 is that trigger word that we need to detect.
  - The model architecture can be like this:



    - The vertical lines in the audio clip represent moment just after the trigger word. The corresponding to this will be 1.

- One disadvantage of this creates a very imbalanced training set. There will be a lot of zeros and few ones.
- A hack to solve this is to make an output a few ones for several times or for a fixed period of time before reverting back to zero.

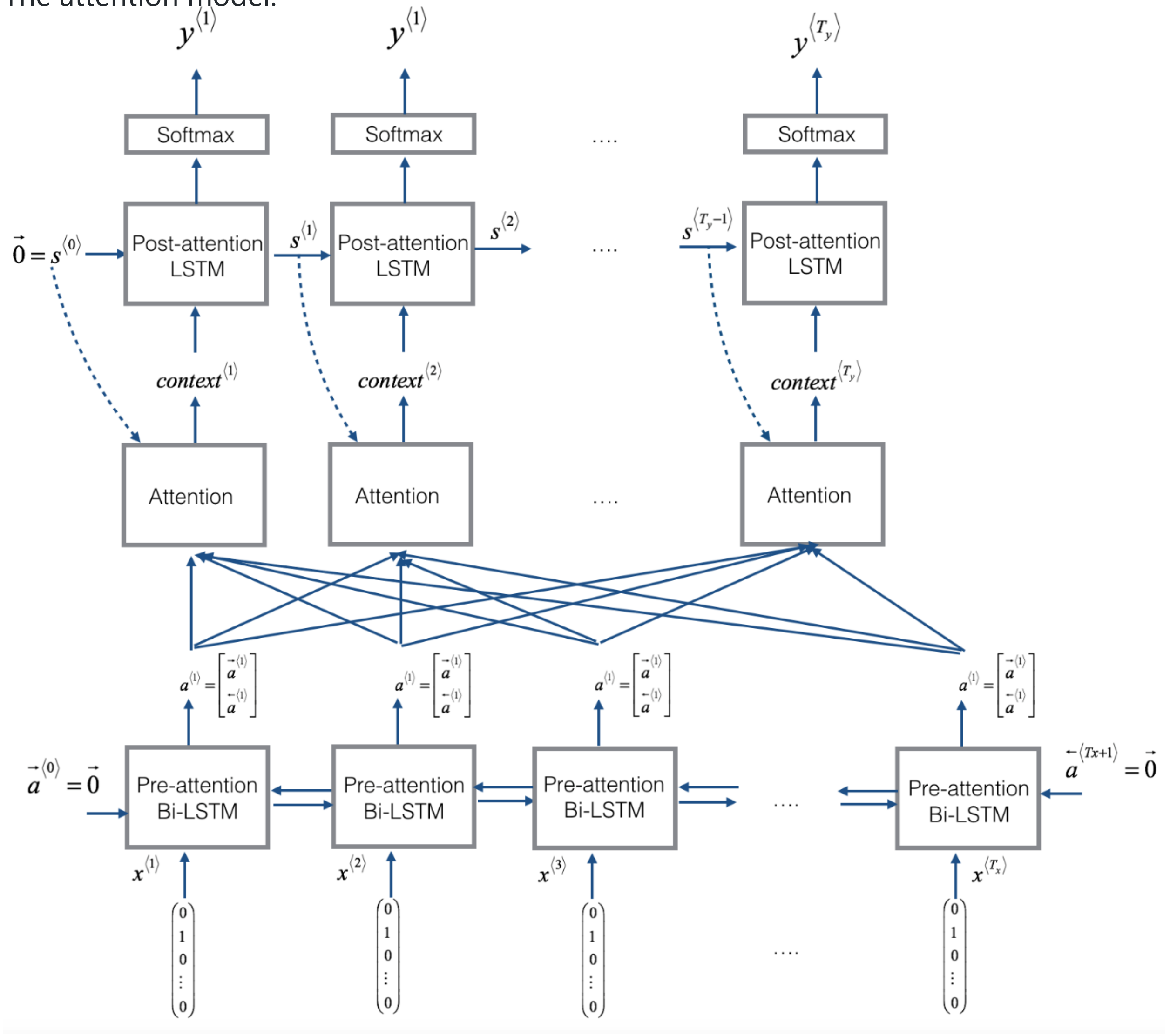


Extras

Machine translation attention model (from notebooks)

- The model is built with keras layers.

- The attention model.

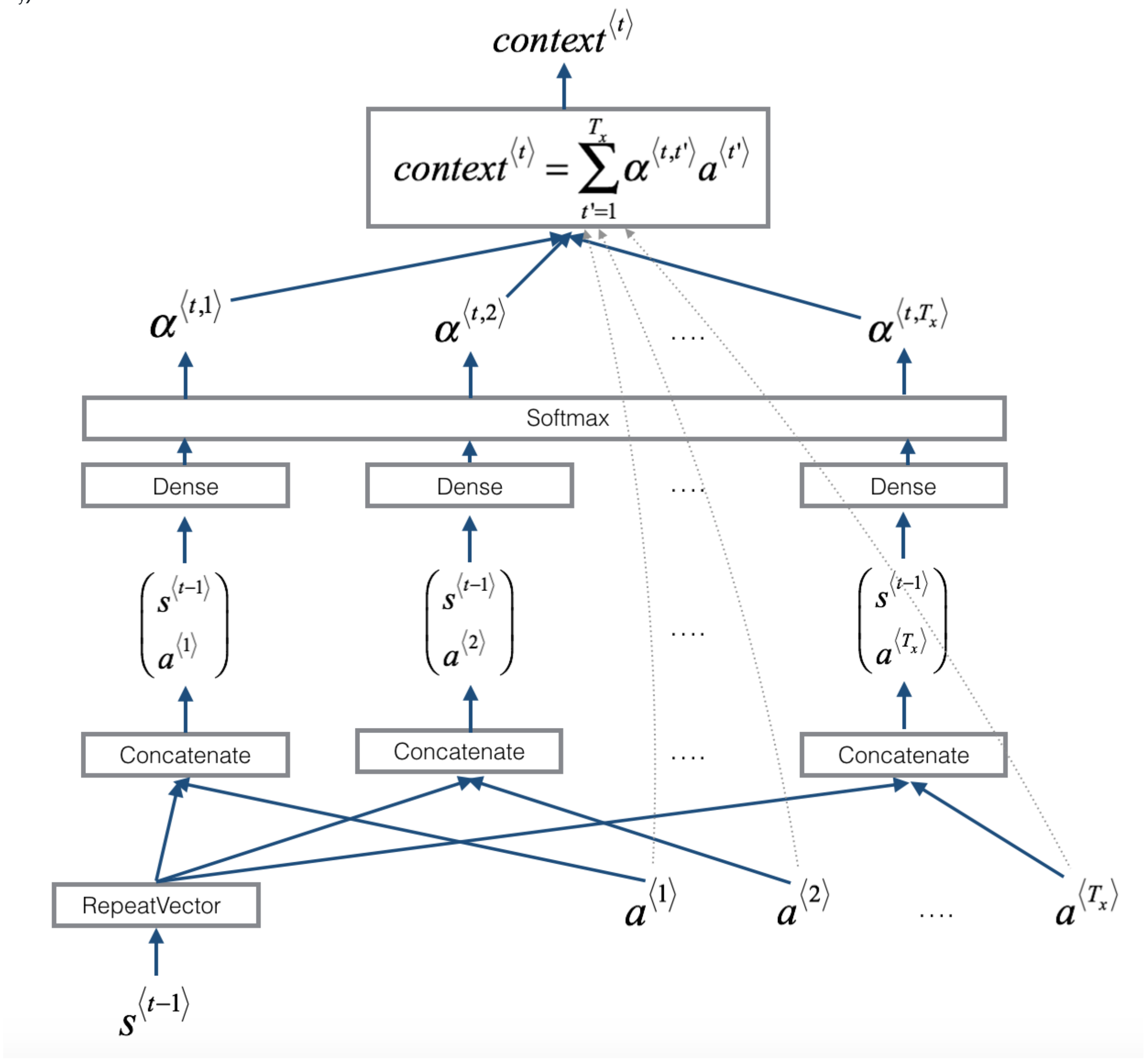


$y^{\langle 1 \rangle}$
$y^{\langle 1 \rangle}$
$y^{\langle T_y \rangle}$
Softmax
Softmax
....
Softmax
$\vec{0} = s^{\langle 0 \rangle}$
Post-attention LSTM
$s^{\langle 1 \rangle}$
Post-attention LSTM
$s^{\langle 2 \rangle}$
....
$s^{\langle T_y - 1 \rangle}$
Post-attention LSTM
$context^{\langle 1 \rangle}$
$context^{\langle 2 \rangle}$
$context^{\langle T_y \rangle}$
Attention
Attention
....
Attention
$a^{\langle 1 \rangle} = \begin{bmatrix} \vec{a}^{\langle 1 \rangle} \\ \overleftarrow{a}^{\langle 1 \rangle} \end{bmatrix}$
$\vec{a}^{\langle 0 \rangle} = \vec{0}$
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
Pre-attention Bi-LSTM
....
Pre-attention Bi-LSTM
$\overleftarrow{a}^{\langle Tx+1 \rangle} = \vec{0}$
$x^{\langle 1 \rangle}$
$x^{\langle 2 \rangle}$
$x^{\langle 3 \rangle}$
$x^{\langle T_x \rangle}$

  - There are two separate LSTMs in this model. Because the one at the bottom of the picture is a Bi-directional LSTM and comes *before* the attention mechanism, we will call it *pre-attention* Bi-LSTM. The LSTM at the top of the diagram comes *after* the attention mechanism, so we will call it the *post-attention* LSTM. The pre-attention Bi-LSTM goes through T_x time steps; the post-attention LSTM goes through T_y time steps.
  - The post-attention LSTM passes $s^{<t>}$, $c^{<t>}$ from one time step to the next. In the lecture videos, we were using only a basic RNN for the post-activation sequence model, so the state captured by the RNN output activations $s^{<t>}$. But since we are using an LSTM here, the LSTM has both the output activation $s^{<t>}$ and the hidden cell state $c^{<t>}$. However, unlike previous text generation examples (such as Dinosaurus in week 1), in this model the post-activation LSTM at time `t` does will not take the specific generated $y^{<t-}$

$^{1>}$ as input; it only takes $s^{<t>}$ and $c^{<t>}$ as input. We have designed the model this way, because (unlike language generation where adjacent characters are highly correlated) there isn't as strong a dependency between the previous character and the next character in a YYYY-MM-DD date.

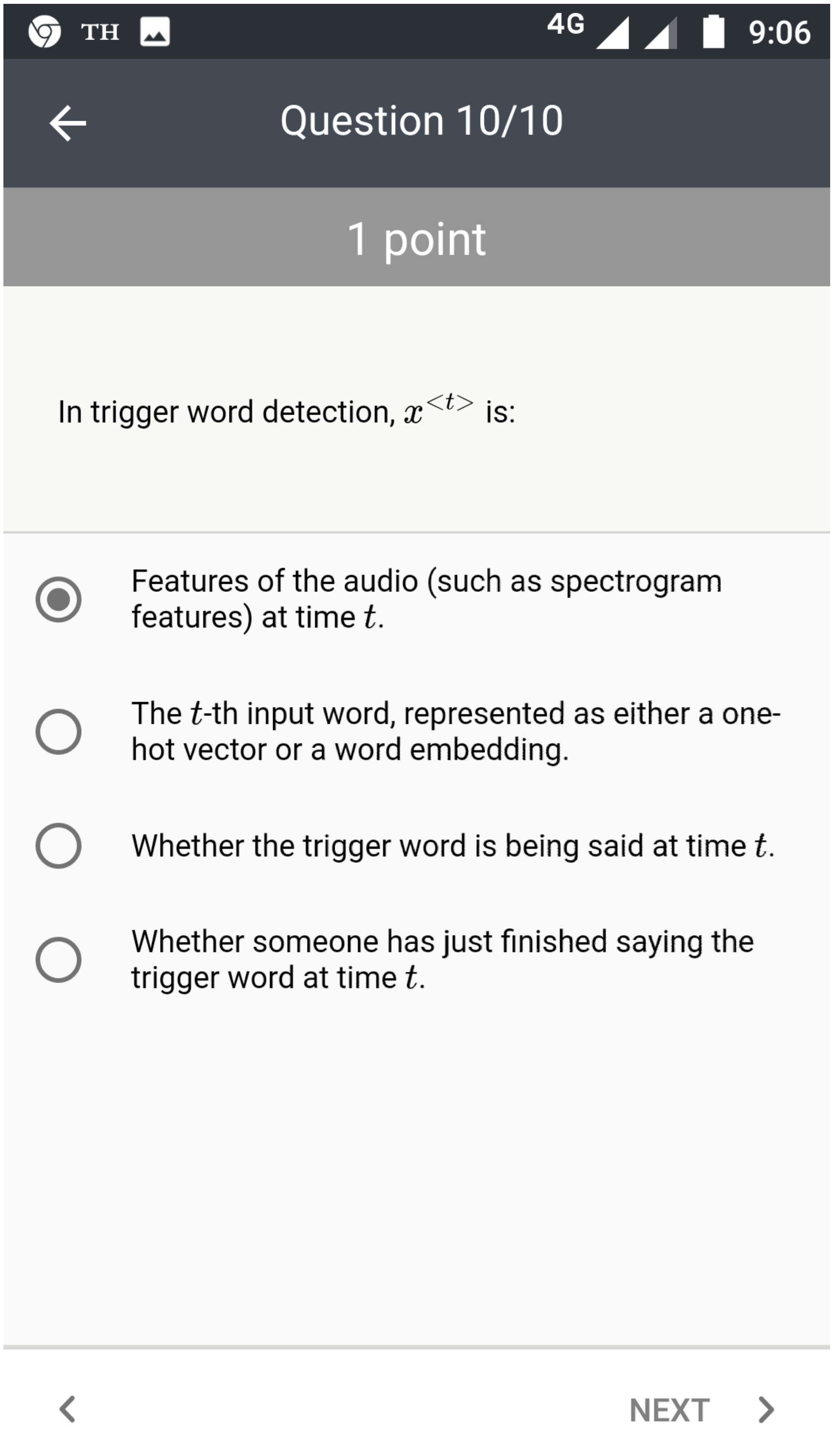
- What one "Attention" step does to calculate the attention variables $\alpha^{<t, t>}$, which are used to compute the context variable context$^{<t>}$ for each timestep in the output (t=1, ..., T_y).



$context^{\langle t \rangle}$
$context^{\langle t \rangle} = \sum_{t'=1}^{T_x} \alpha^{\langle t,t' \rangle} a^{\langle t' \rangle}$
$\alpha^{\langle t,1 \rangle}$
$\alpha^{\langle t,2 \rangle}$
$\alpha^{\langle t,T_x \rangle}$
Softmax
Dense
Dense
Dense
$s^{\langle t-1 \rangle}$ $a^{\langle 1 \rangle}$
$s^{\langle t-1 \rangle}$ $a^{\langle 2 \rangle}$
$s^{\langle t-1 \rangle}$ $a^{\langle T_x \rangle}$
Concatenate
Concatenate
Concatenate
RepeatVector
$a^{\langle 1 \rangle}$
$a^{\langle 2 \rangle}$
$a^{\langle T_x \rangle}$
$s^{\langle t-1 \rangle}$

  - The diagram uses a `RepeatVector` node to copy $s^{<t-1>}$'s value T_x times, and then `Concatenation` to concatenate $s^{<t-1>}$ and $a^{<t>}$ to compute $e^{<t, t>}$, which is then passed through a softmax to compute $\alpha^{<t, t>}$.

Quiz Notes:


TH 4G 9:06

Question 10/10

1 point

In trigger word detection, $x^{<t>}$ is:

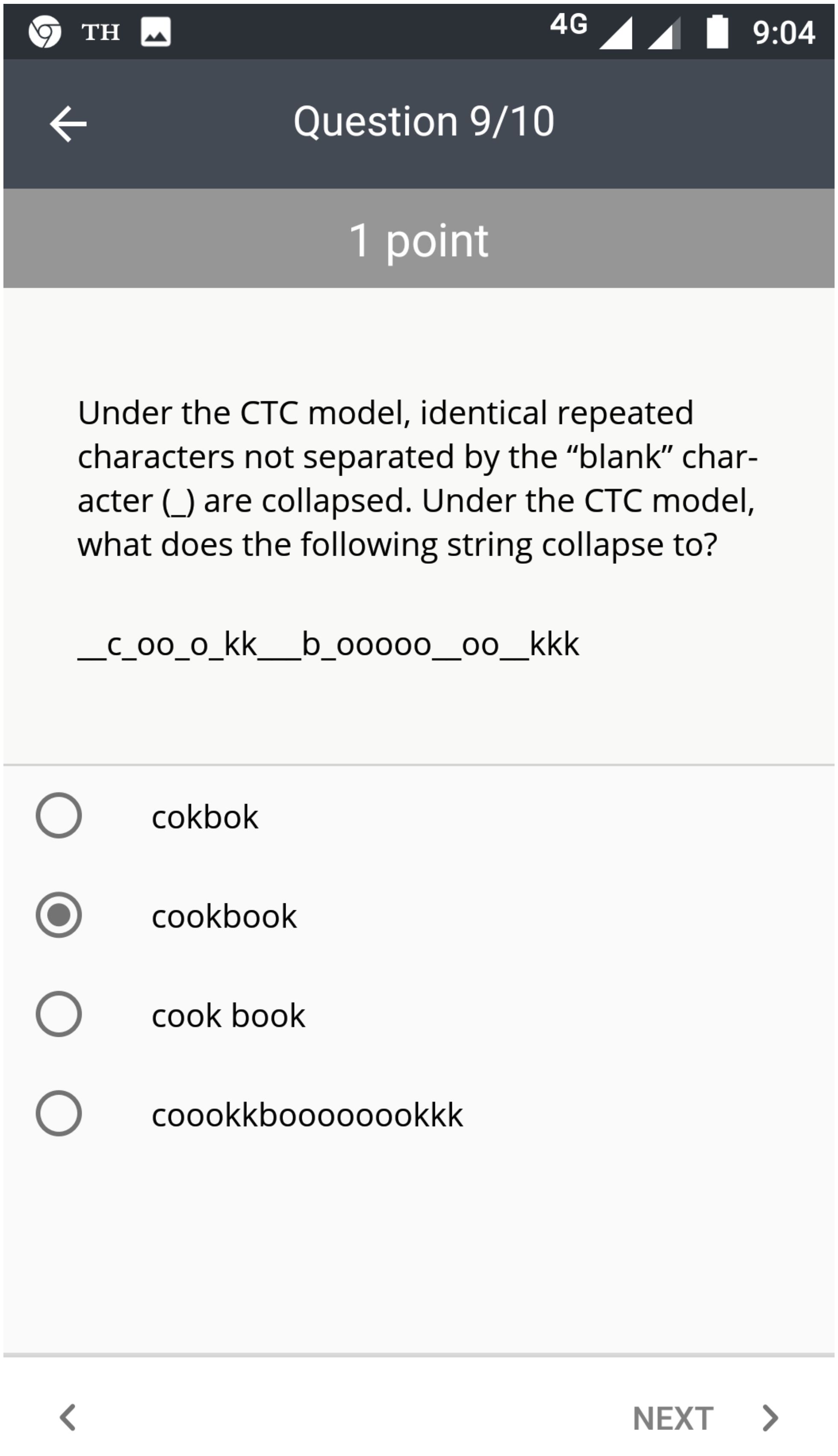
[✓ Features of the audio (such as spectrogram features) at time t.]

The t-th input word, represented as either a one-hot vector or a word embedding.

Whether the trigger word is being said at time t.

Whether someone has just finished saying the trigger word at time t.

NEXT



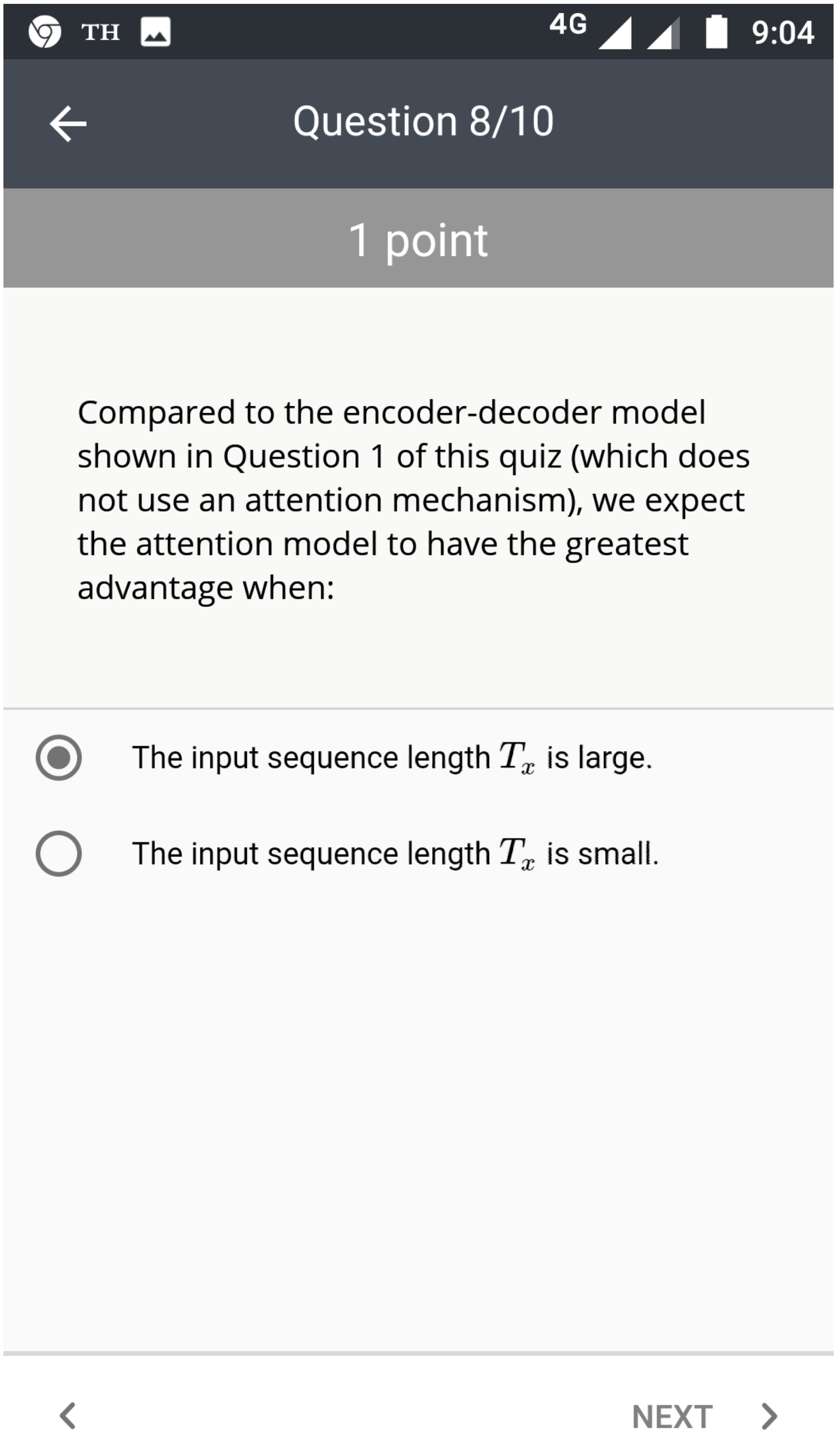
Question 9/10

1 point

Under the CTC model, identical repeated characters not separated by the "blank" character (_) are collapsed. Under the CTC model, what does the following string collapse to?

__c_oo_o_kk___b_ooooo__oo__kkk

- ○ cokbok
- [✓ ● cookbook]
- ○ cook book
- ○ coookkbooooookkk

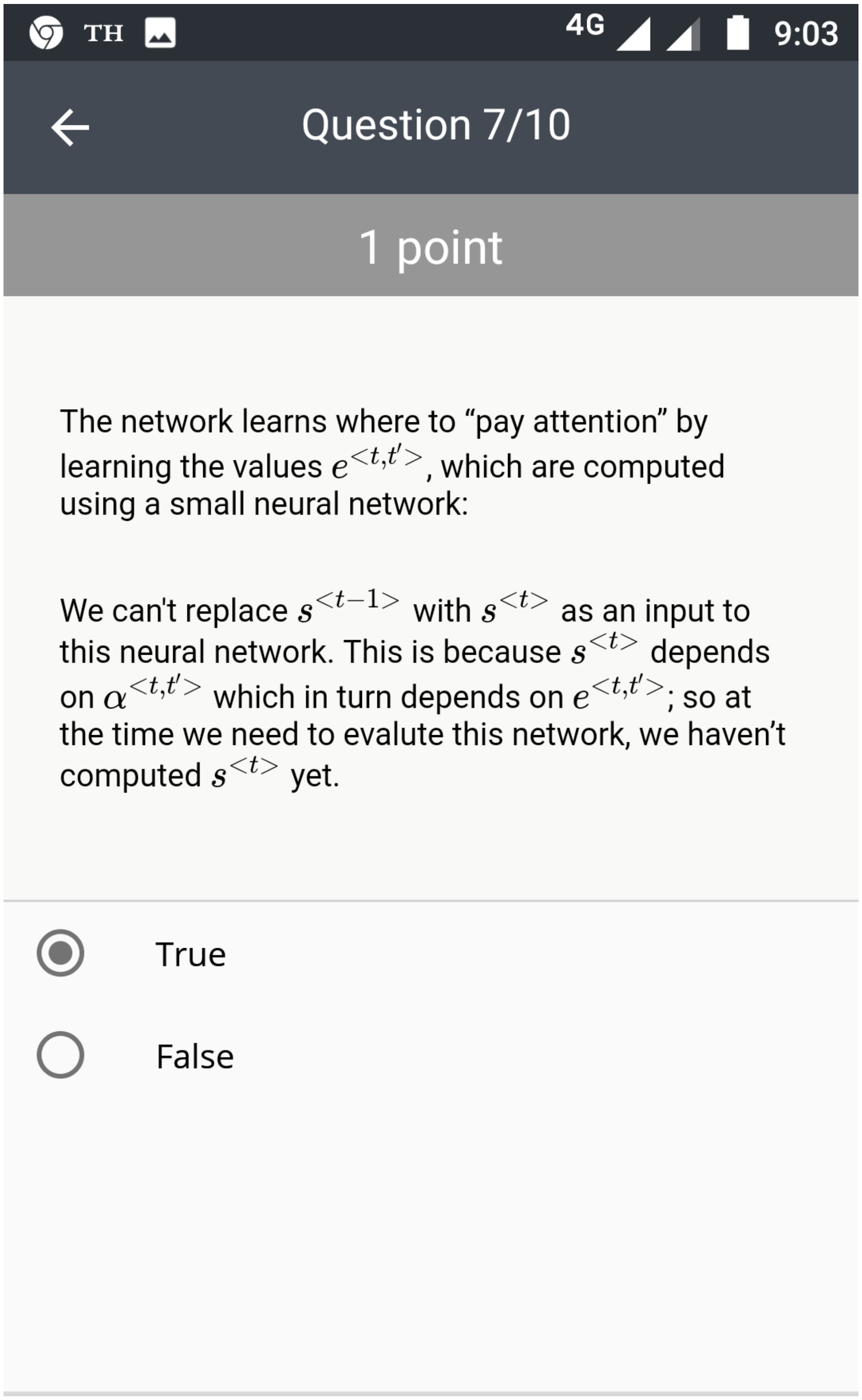


Question 8/10

1 point

Compared to the encoder-decoder model shown in Question 1 of this quiz (which does not use an attention mechanism), we expect the attention model to have the greatest advantage when:

- [✓ The input sequence length T_x is large.]
- The input sequence length T_x is small.


TH 4G 9:03

Question 7/10

1 point

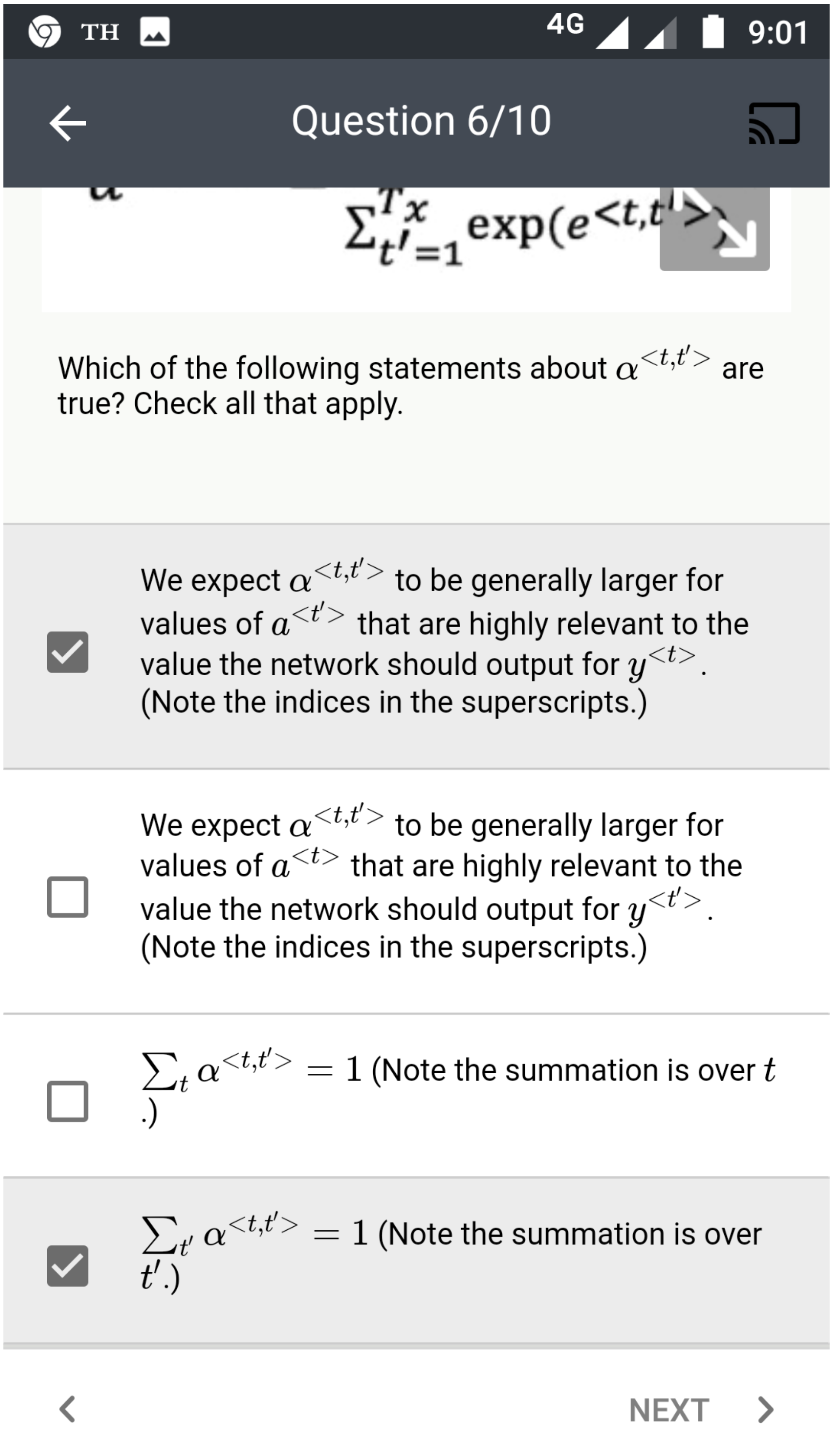
The network learns where to “pay attention” by learning the values $e^{<t,t'>}$, which are computed using a small neural network:

We can't replace $s^{<t-1>}$ with $s^{<t>}$ as an input to this neural network. This is because $s^{<t>}$ depends on $\alpha^{<t,t'>}$ which in turn depends on $e^{<t,t'>}$; so at the time we need to evalute this network, we haven’t computed $s^{<t>}$ yet.

[✓ True]

False

NEXT

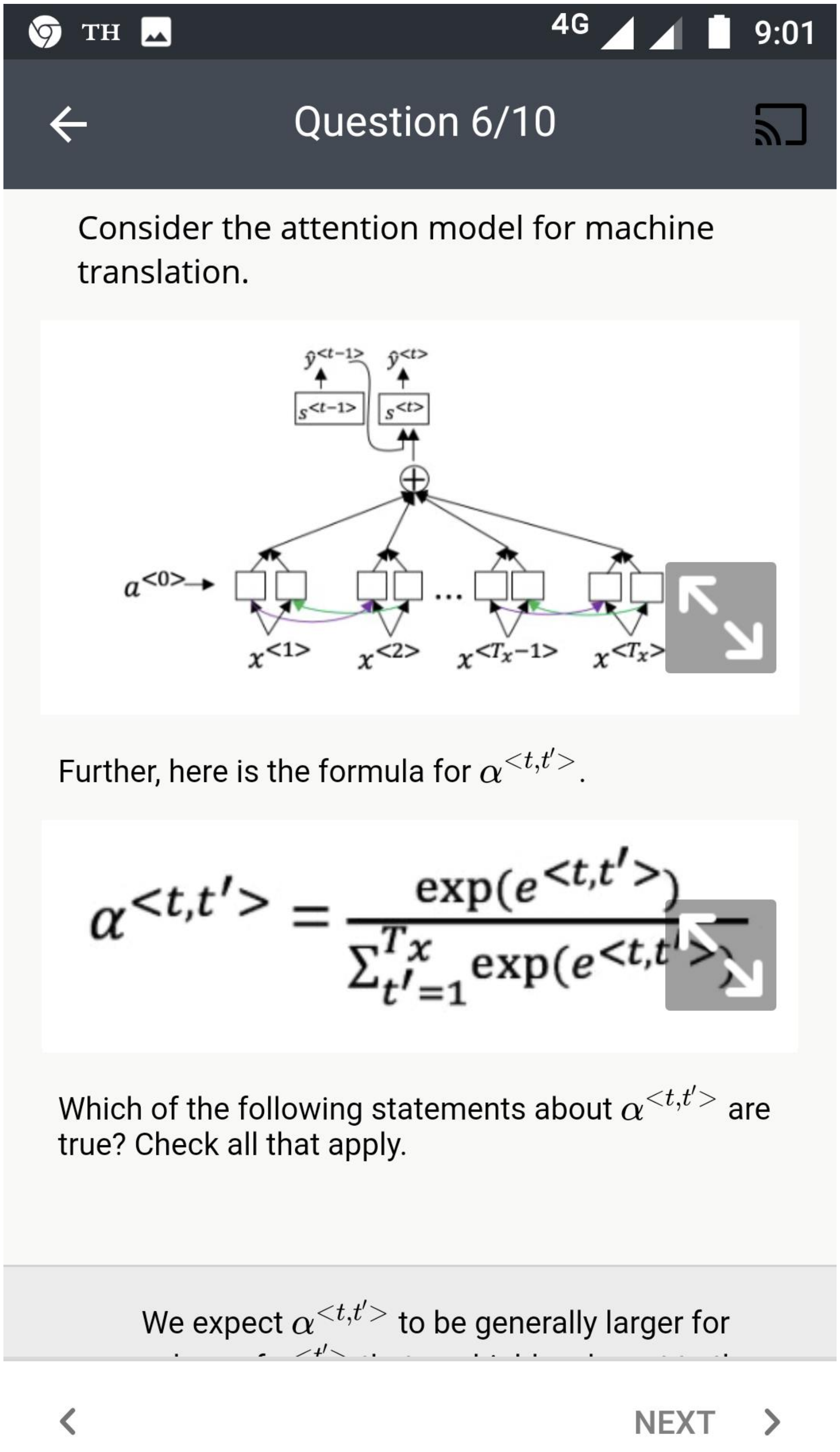


Question 6/10

$\Sigma_{t'=1}^{T_x} \exp(e^{<t,t'>}$

Which of the following statements about $\alpha^{<t,t'>}$ are true? Check all that apply.

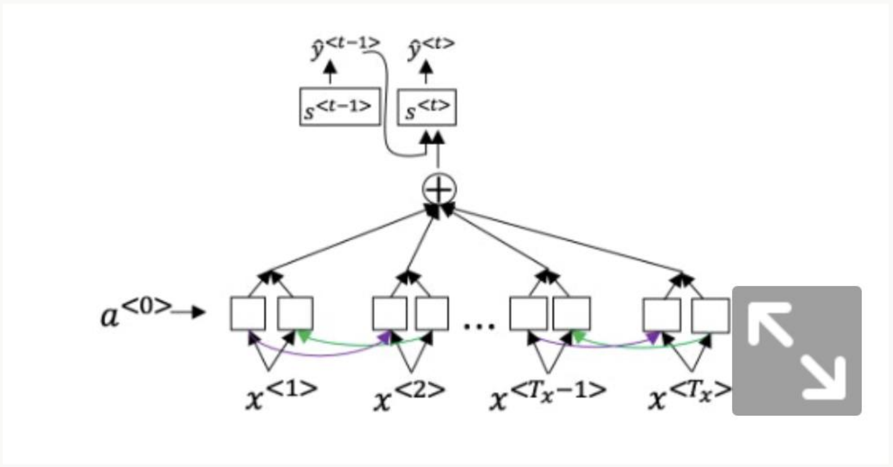
- [✓] We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t'>}$ that are highly relevant to the value the network should output for $y^{<t>}$. (Note the indices in the superscripts.)
- [ ] We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t>}$ that are highly relevant to the value the network should output for $y^{<t'>}$. (Note the indices in the superscripts.)
- [ ] $\sum_t \alpha^{<t,t'>} = 1$ (Note the summation is over t.)
- [✓] $\sum_{t'} \alpha^{<t,t'>} = 1$ (Note the summation is over t'.)



TH 4G 9:01

Question 6/10

Consider the attention model for machine translation.



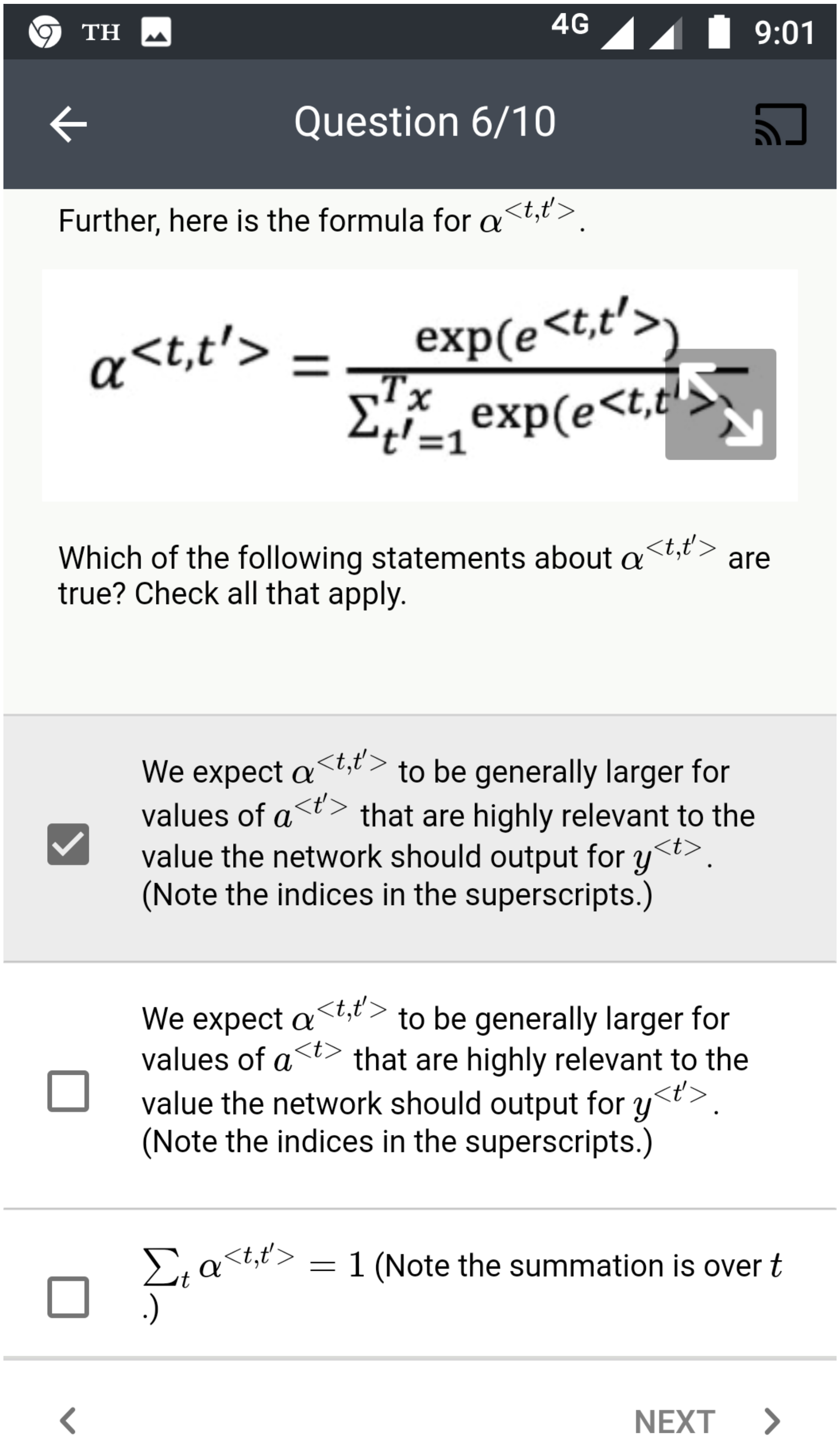
Further, here is the formula for $\alpha^{<t,t'>}$.

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

Which of the following statements about $\alpha^{<t,t'>}$ are true? Check all that apply.

We expect $\alpha^{<t,t'>}$ to be generally larger for

NEXT



Question 6/10

Further, here is the formula for $\alpha^{<t,t'>}$.

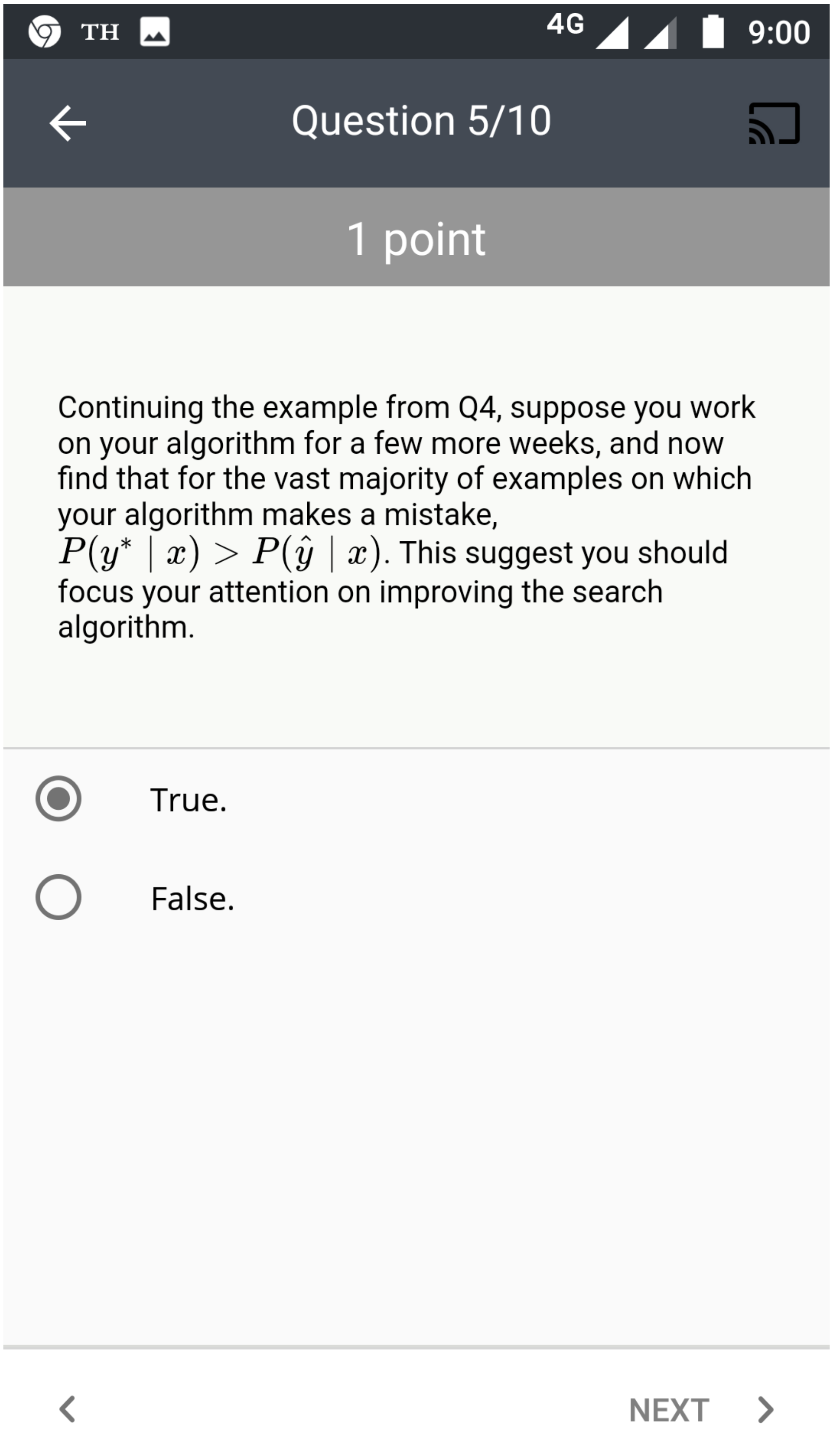
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

Which of the following statements about $\alpha^{<t,t'>}$ are true? Check all that apply.

[✓] We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t'>}$ that are highly relevant to the value the network should output for $y^{<t>}$. (Note the indices in the superscripts.)

☐ We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t>}$ that are highly relevant to the value the network should output for $y^{<t'>}$. (Note the indices in the superscripts.)

☐ $\sum_t \alpha^{<t,t'>} = 1$ (Note the summation is over t.)



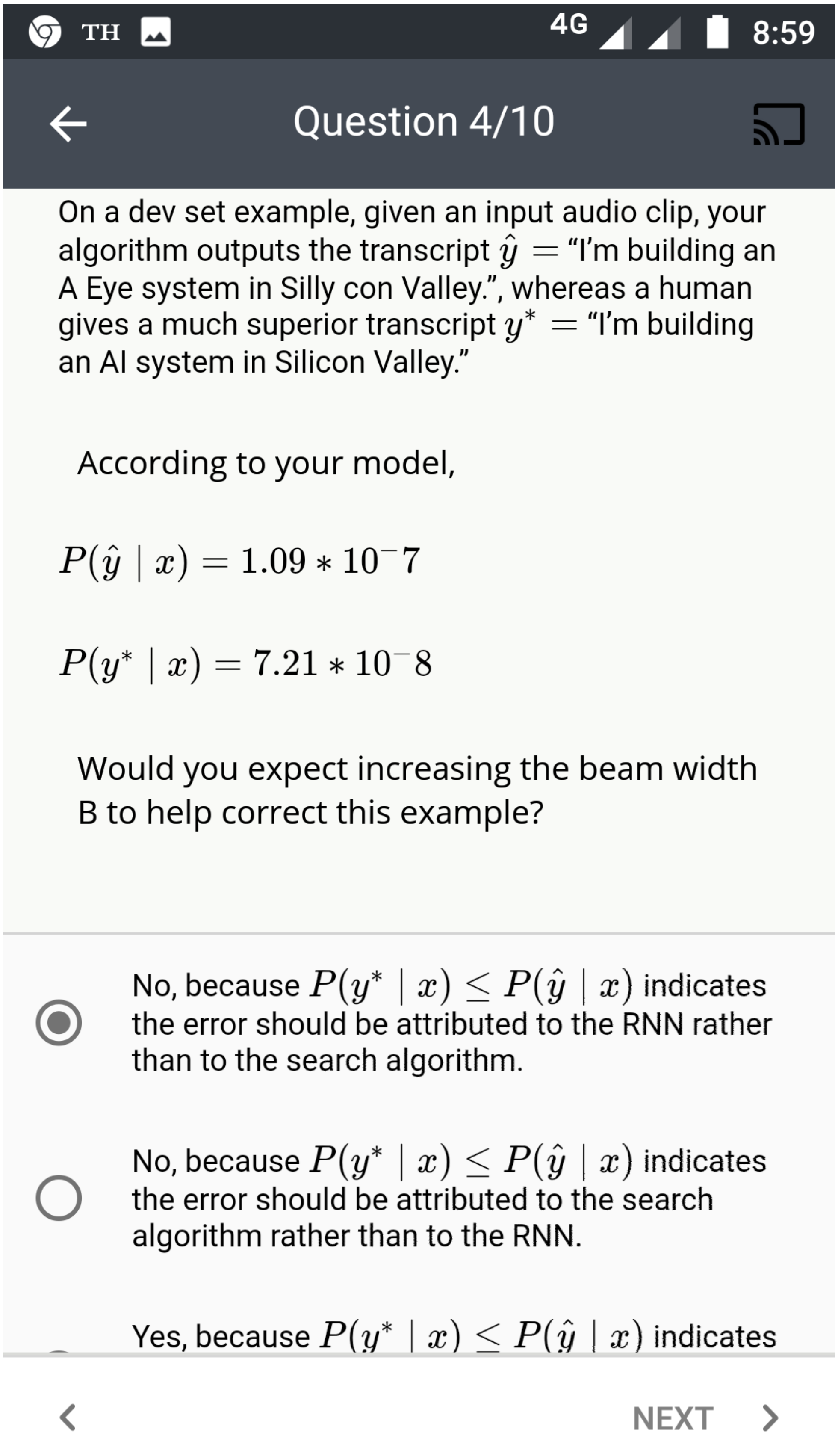
Question 5/10

1 point

Continuing the example from Q4, suppose you work on your algorithm for a few more weeks, and now find that for the vast majority of examples on which your algorithm makes a mistake, $P(y^* \mid x) > P(\hat{y} \mid x)$. This suggest you should focus your attention on improving the search algorithm.

[✓ True.]

False.



Question 4/10

On a dev set example, given an input audio clip, your algorithm outputs the transcript $\hat{y}$ = "I'm building an A Eye system in Silly con Valley.", whereas a human gives a much superior transcript y^* = "I'm building an AI system in Silicon Valley."

According to your model,

$$P(\hat{y} \mid x) = 1.09 * 10^-7$$

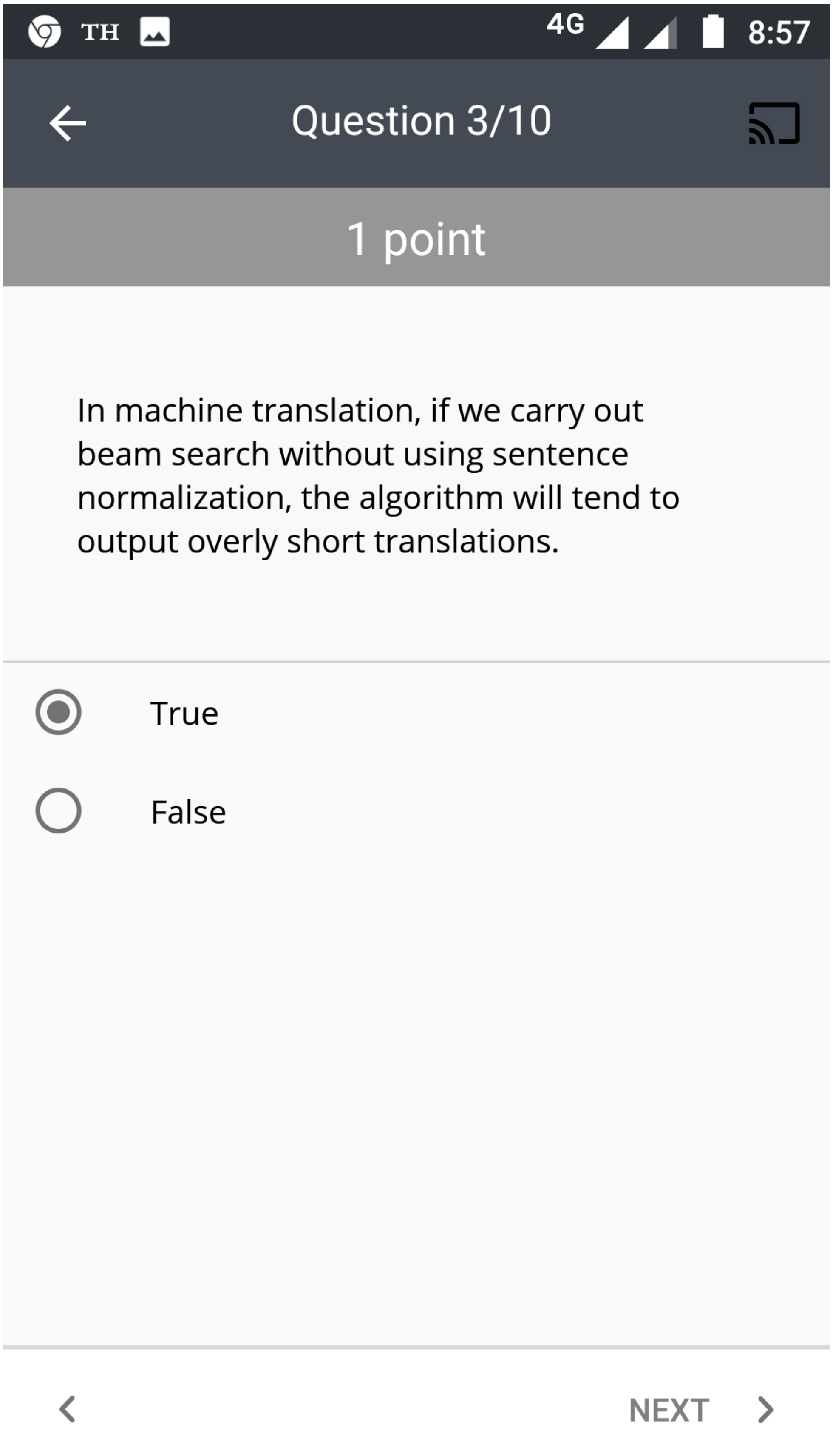
$$P(y^* \mid x) = 7.21 * 10^-8$$

Would you expect increasing the beam width B to help correct this example?

[✓ No, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the RNN rather than to the search algorithm.]

No, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the search algorithm rather than to the RNN.

Yes, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates



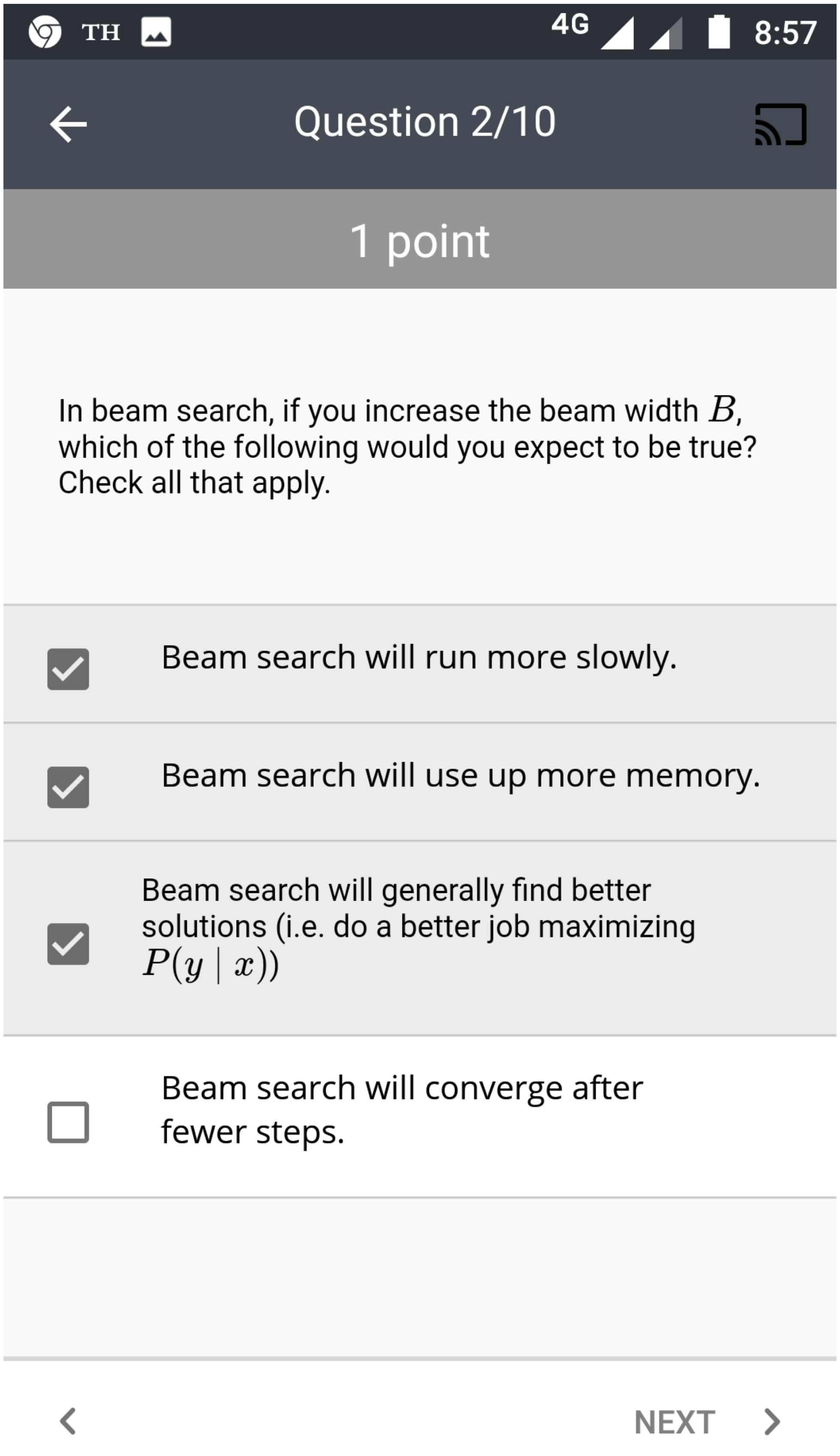
Question 3/10

1 point

In machine translation, if we carry out beam search without using sentence normalization, the algorithm will tend to output overly short translations.

[✓ True]

False

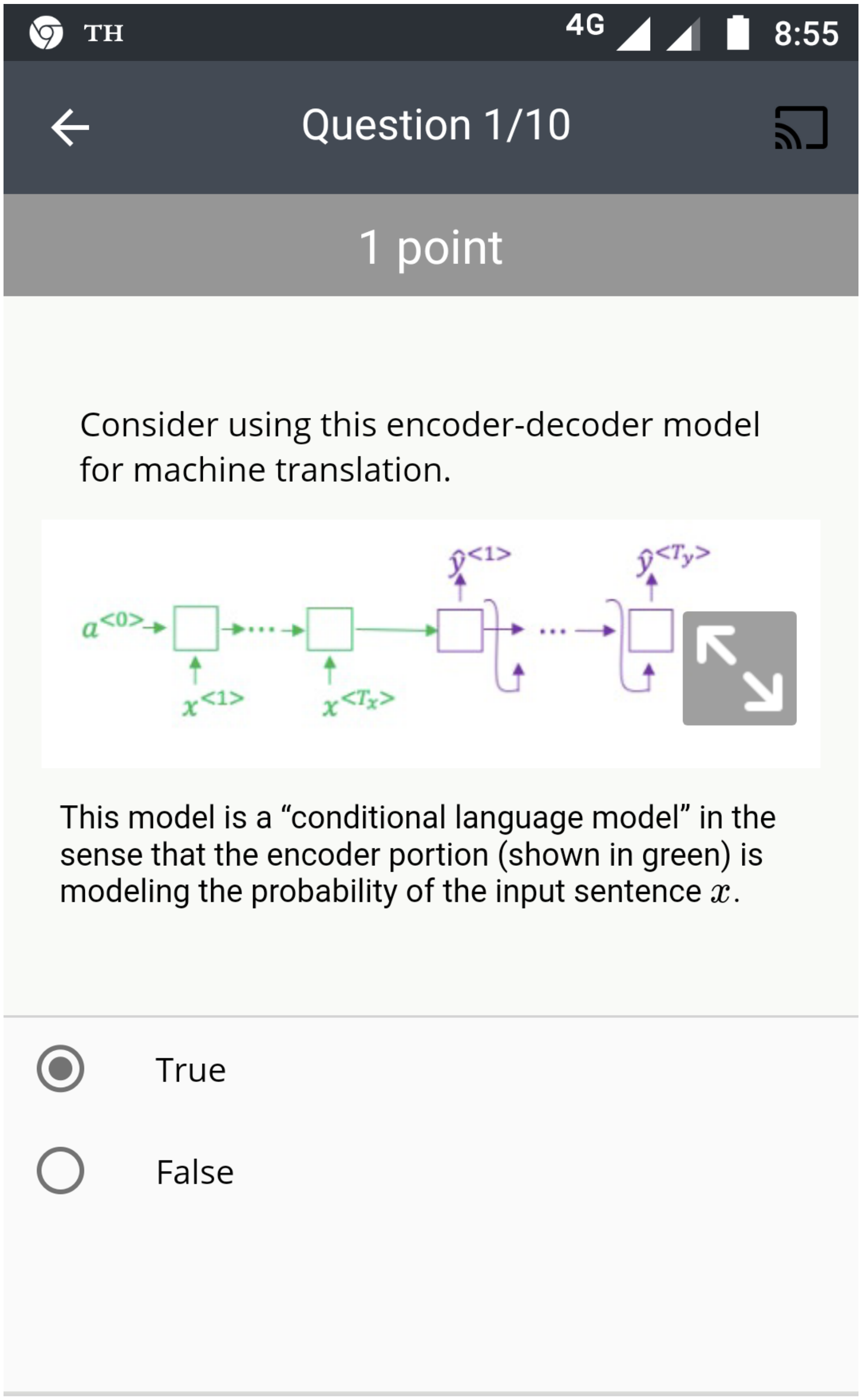


Question 2/10

1 point

In beam search, if you increase the beam width B, which of the following would you expect to be true? Check all that apply.

- ✓ Beam search will run more slowly.
- ✓ Beam search will use up more memory.
- ✓ Beam search will generally find better solutions (i.e. do a better job maximizing $P(y \mid x)$)
- ☐ Beam search will converge after fewer steps.

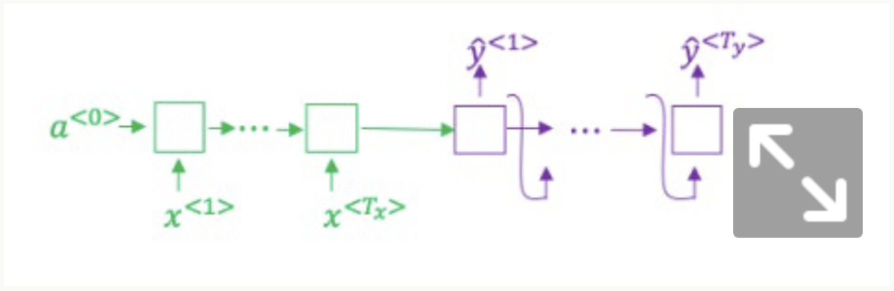


TH 4G 8:55

Question 1/10

1 point

Consider using this encoder-decoder model for machine translation.

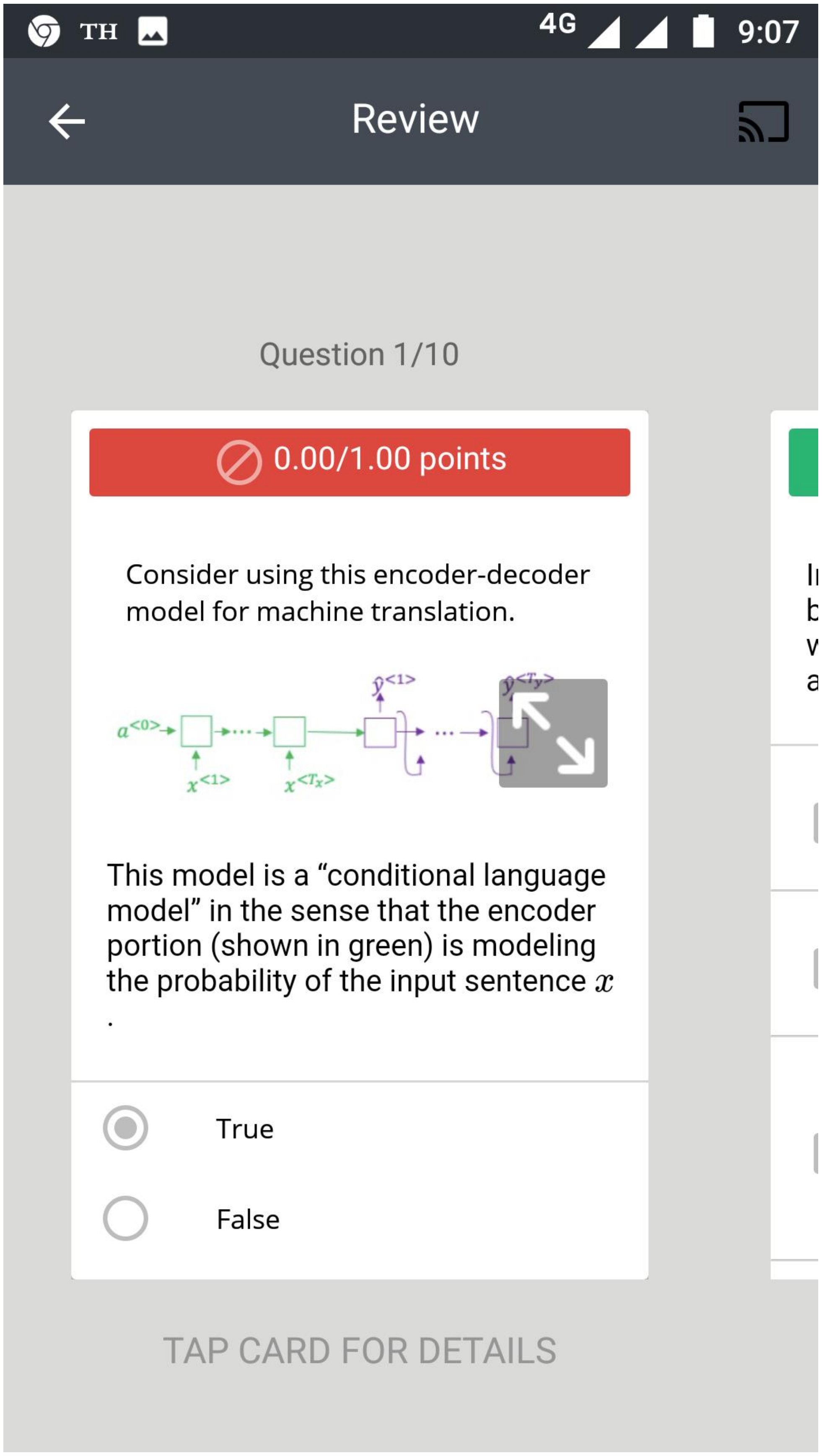


This model is a “conditional language model” in the sense that the encoder portion (shown in green) is modeling the probability of the input sentence x.

[✓ True]

False

NEXT



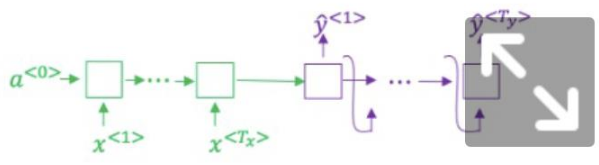
TH 4G 9:07

Review

Question 1/10

0.00/1.00 points

Consider using this encoder-decoder model for machine translation.



This model is a “conditional language model” in the sense that the encoder portion (shown in green) is modeling the probability of the input sentence x
.

[✓ True]

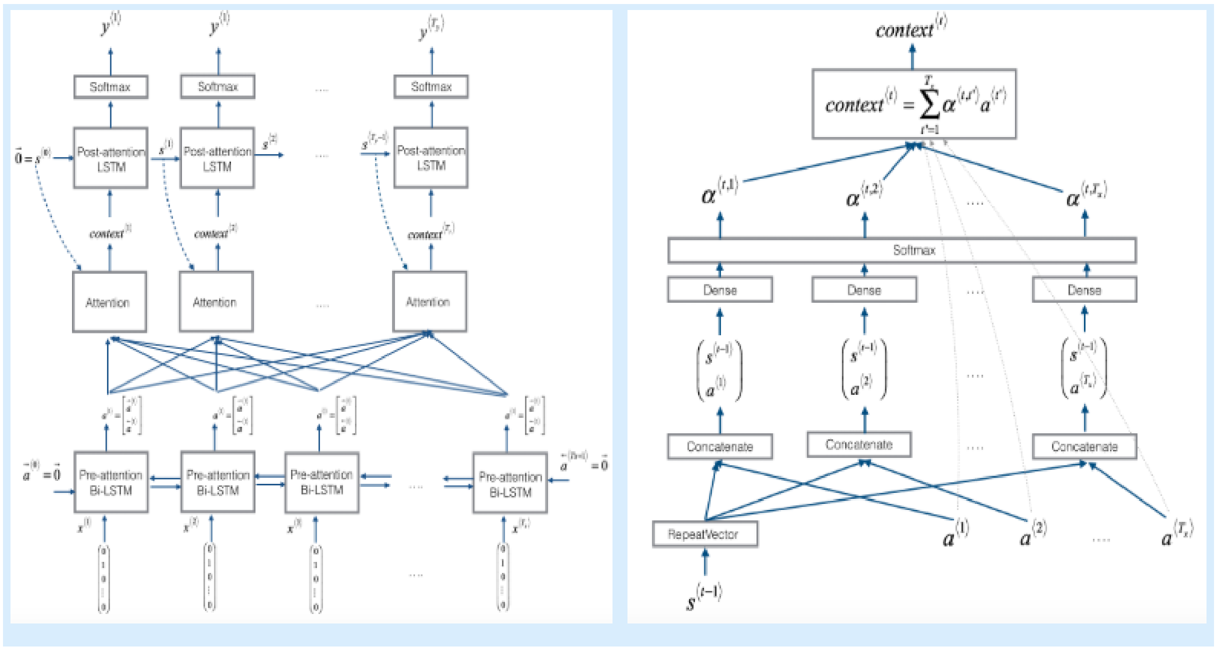
False

TAP CARD FOR DETAILS

Assignment Notes:

Translating human readable dates into machine readable dates

- Pre-attention Bi-LSTM encoder + attention nw + post-attention LSTM
- An attention network creating the context vector



Softmax
Post-attention LSTM
Attention
Pre-attention Bi-LSTM
$context^{\langle t \rangle} = \sum_{t'=1}^{T_x} \alpha^{\langle t,t' \rangle} a^{\langle t' \rangle}$
Dense
Concatenate
RepeatVector

Figure 1: Neural machine translation with attention

Visualizing Attention:

3 - Visualizing Attention (Optional / Ungraded)

Since the problem has a fixed output length of 10, it is also possible to carry out this task using 10 different softmax units to generate the 10 characters of the output. But one advantage of the attention model is that each part of the output (say the month) knows it needs to depend only on a small part of the input (the characters in the input giving the month). We can visualize what part of the output is looking at what part of the input.

Consider the task of translating "Saturday 9 May 2018" to "2018-05-09". If we visualize the computed $\alpha^{\langle t,t' \rangle}$ we get this:

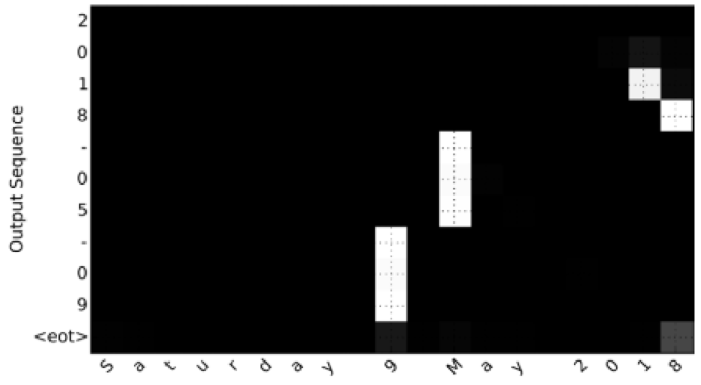


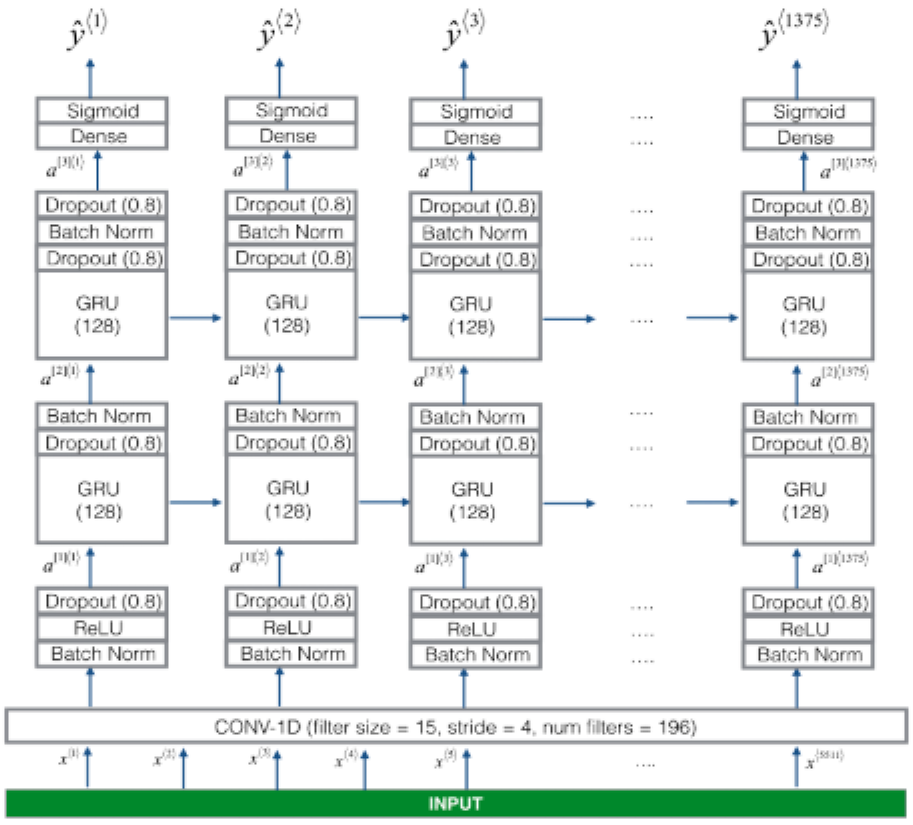
Figure 8: Full Attention Map

Notice how the output ignores the "Saturday" portion of the input. None of the output timesteps are paying much attention to that portion of the input. We see also that 9 has been translated as 09 and May has been correctly translated into 05, with the output paying attention to the parts of the input it needs to to make the translation. The year mostly requires it to pay attention to the input's "18" in order to generate "2018."

- Machine translation models can be used to map from one sequence to another. They are useful not just for translating human languages (like French->English) but also for tasks like date format translation.
- An attention mechanism allows a network to focus on the most relevant parts of the input when producing a specific part of the output.
- A network using an attention mechanism can translate from inputs of length T_x to outputs of length T_y, where T_x and T_y can be different.
- You can visualize attention weights $\alpha^{\langle t,t' \rangle}$ to see what the network is paying attention to while generating each output.

Trigger Speech Detection:

Learnt about how to generate training data for speeches


$\hat{y}^{\langle 1 \rangle}$
$\hat{y}^{\langle 2 \rangle}$
$\hat{y}^{\langle 3 \rangle}$
$\hat{y}^{\langle 1375 \rangle}$
Sigmoid
Dense
$a^{[3]\langle 1 \rangle}$
Dropout (0.8)
Batch Norm
Dropout (0.8)
GRU (128)
$a^{[2]\langle 1 \rangle}$
Batch Norm
Dropout (0.8)
GRU (128)
$a^{[1]\langle 1 \rangle}$
Dropout (0.8)
ReLU
Batch Norm
CONV-1D (filter size = 15, stride = 4, num filters = 196)
$x^{\langle 1 \rangle}$
$x^{\langle 5511 \rangle}$
INPUT

The 1D Conv layer plays a role similar to the 2D convolutions you saw in Course 4, of extracting low-level features and then possibly generating an output of a smaller dimension.