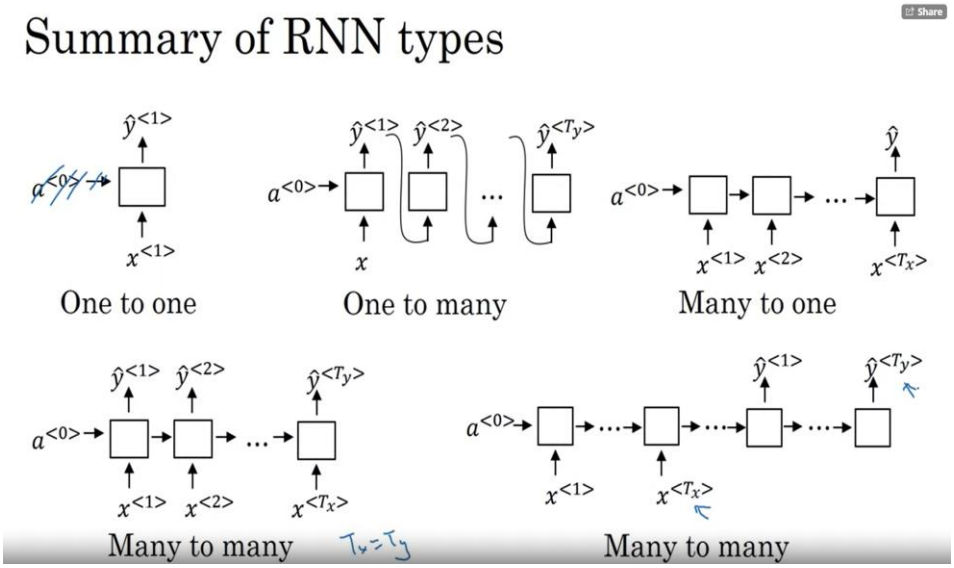
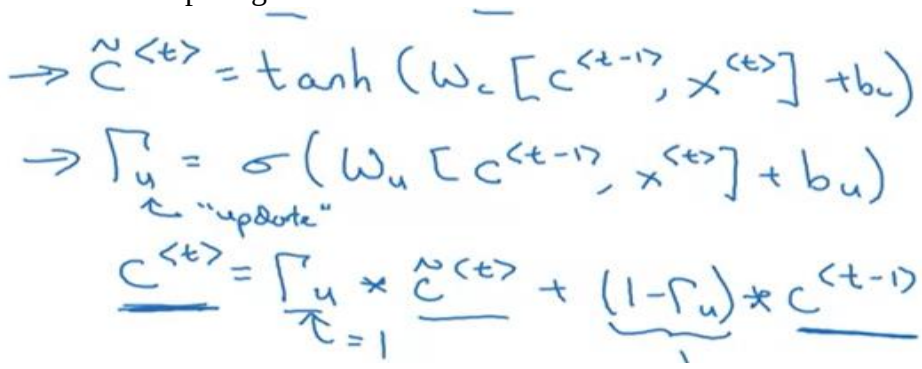
Sequence Model

Week 1:

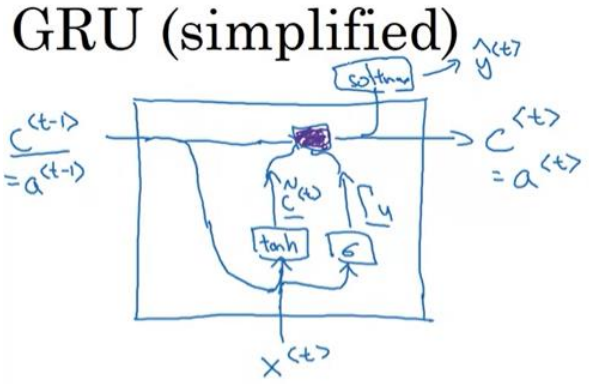
1. Recurrent NN:
    a. And what a recurrent neural network does is, when it then goes on to read the second word in the sentence, say x2, instead of just predicting y2 using only X2, it also gets to input some information from whether the computer that time step one. So in particular, deactivation value from time step one is passed on to time step two. Then at the next time step, recurrent neural network inputs the third word X3 and it tries to output some prediction, Y hat three and so on up until the last time step where it inputs x_TX and then it outputs y_hat_ty
    b. The recurrent neural network scans through the data from left to right. The parameters it uses for each time step are shared. So there'll be a set of parameters which we'll describe in greater detail on the next slide, but the parameters governing the connection from X1 to the hidden layer, will be some set of parameters we're going to write as Wax and is the same parameters Wax that it uses for every time step.
2. RNN types



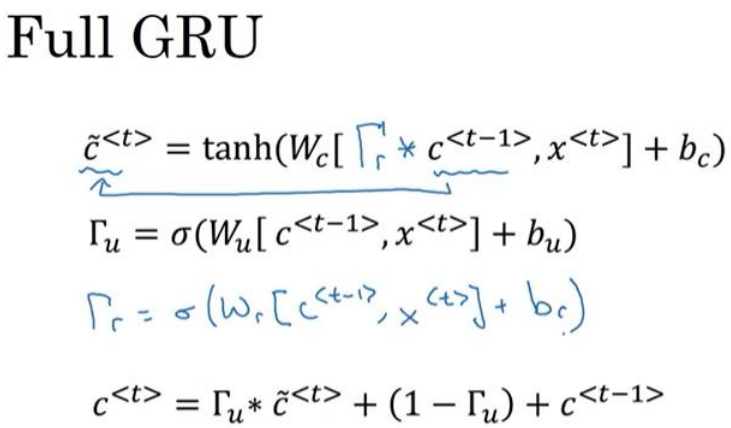

3. GRU:
    a. the GRU unit is going to have a new variable called c,which stands for cell, for memory cell. And what the memory cell do is it will provide a bit of memory to remember, for example, whether word was singular or plural
    b. Formula for computing the activation



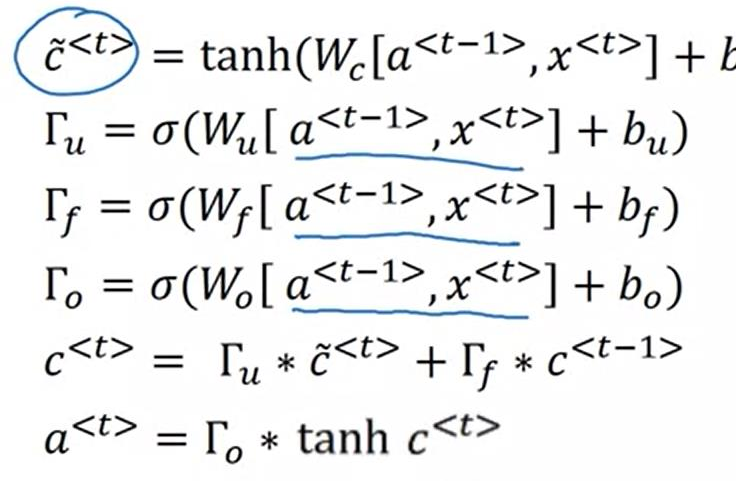

    c. GRU unit:



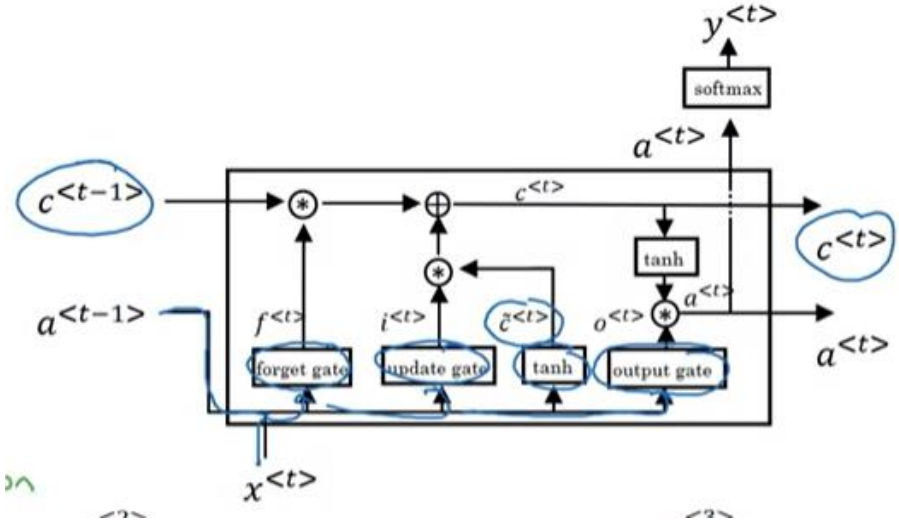

d. Full gru unit:



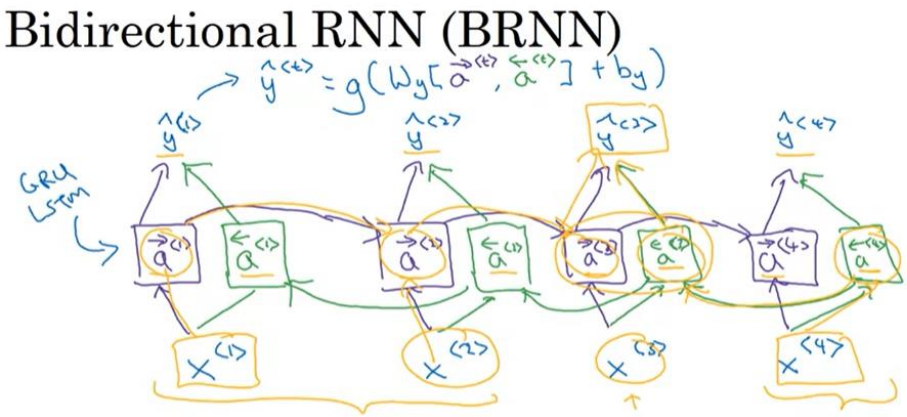

4. LSMT
   a. LSTM cell is meant to do: allow past information to be reinjected at a later time.
   b. Calculations of forward propagation



c. LSTM unit:




5. Bidirectional




Week 2

6. Embedding methods (alternatively referred to as “encoding”, “vectorising”, etc) convert symbolic representations (i.e. words, emojis, categorical items, dates, times, other features, etc) into meaningful numbers (i.e. real numbers that capture underlying semantic relations between the symbols).
7. **In the skip-gram model**, what we're going to do is come up with a few context to target errors to create our supervised learning problem. So rather than having the context be always the last four words or the last end words immediately before the target word, what I'm going to do is, say, randomly pick a word to be the context word. And what we're going to do is randomly pick another word within some window. Say plus minus five words or plus minus ten words of the context word and we choose that to be target word.

   The network is going to learn the statistics from the number of times each pairing shows up. So, for example, the network is probably going to get many more training samples of (“Soviet”, “Union”) than it is of (“Soviet”, “Sasquatch”). When the training is finished, if you give it the word “Soviet” as input, then it will output a much higher probability for “Union” or “Russia” than it will for “Sasquatch”.

8. **Negative sampling** addresses this by having each training sample only modify a small percentage of the weights, rather than all of them. Here’s how it works. When training the network on the word pair (“fox”, “quick”), recall that the “label” or “correct

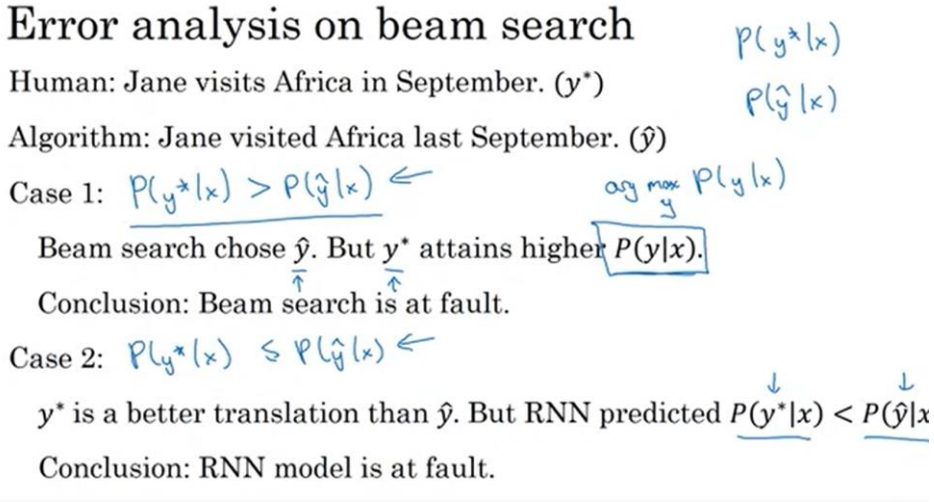
output" of the network is a one-hot vector. That is, for the output neuron corresponding to "quick" to output a 1, and for all of the other thousands of output neurons to output a 0. With negative sampling, we are instead going to randomly select just a small number of "negative" words (let's say 5) to update the weights for. (In this context, a "negative" word is one for which we want the network to output a 0 for). We will also still update the weights for our "positive" word (which is the word "quick" in our current example). Recall that the output layer of our model has a weight matrix that's 300 x 10,000. So we will just be updating the weights for our positive word ("quick"), plus the weights for 5 other words that we want to output 0. That's a total of 6 output neurons, and 1,800 weight values total. That's only 0.06% of the 3M weights in the output layer!

9. So what the **GloVe model** does is, it optimizes the following. We're going to minimize

$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(\theta_i^T e_j + b_i - b_j' - \log X_{ij})^2$$

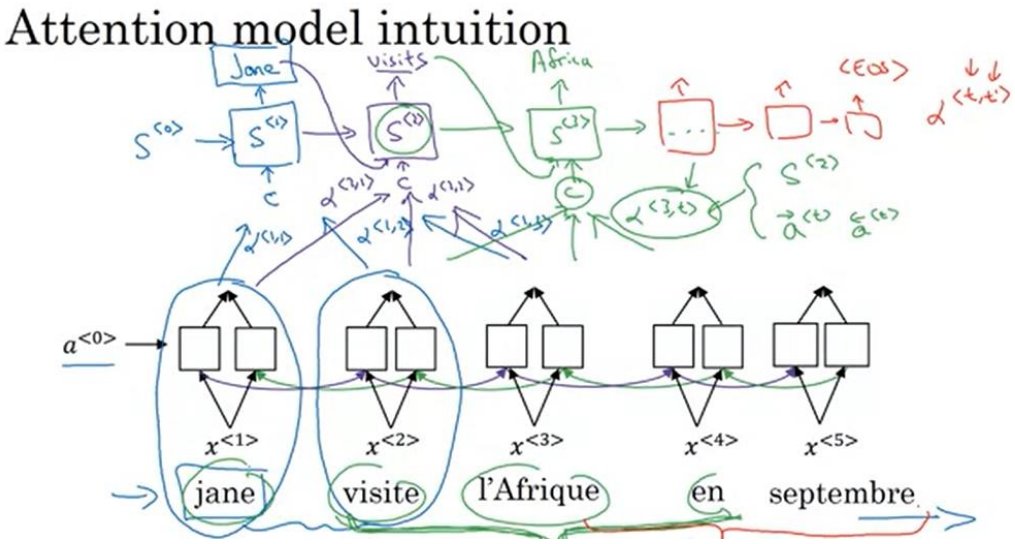
Week 3

10. Greedy Search - greedy search is an algorithm from computer science which says to generate the first word just pick whatever is the most likely first word according to your conditional language model. Going to your machine translation model and then after having picked the first word, you then pick whatever is the second word that seems most likely, then pick the third word that seems most likely.
11. Beam search -
    a. Beam Search algorithm has a parameter called B, which is called the beam width and for this example I'm going to set the beam width to be with the three. And what this means is Beam search will cause that not just one possibility but consider three at the time. So in particular, let's say evaluating this probability over different choices the first words
    b. for each of these three choices consider what should be the second word. I'm just listing words from the vocabulary, from the dictionary. So, to evaluate the probability of second word, it will use this new network and for the decoder portion when trying to decide what comes after in.\
12. Error beam search analysis –




    a.
13. Bleu score - is given a machine generated translation, it allows you to automatically compute a score that measures how good is that machine translation.

14. Attention model -




15. **The CTC cost function** allows the RNN to generate an output like this ttt, there's a special character called the blank character, which we're going to write as an underscore here, h_eee___, and then maybe a space, we're going to write like this, so that a space and then ___ qqq__. And, this is considered a correct output for the first parts of the space, quick with the Q, and the basic rule for the CTC cost function is to collapse repeated characters not separated by "blank". So, to be clear, I'm using this underscore to denote a special blank character and that's different than the space character. So, there is a space here between the and quick, so I should output a space. But, by collapsing repeated characters, not separated by blank, it actually collapse the sequence into t, h, e, and then space, and q, and this allows your network to have a thousand outputs by repeating characters allow the times. So, inserting a bunch of blank characters and still ends up with a much shorter output text transcript.