Lecture-9

Contents

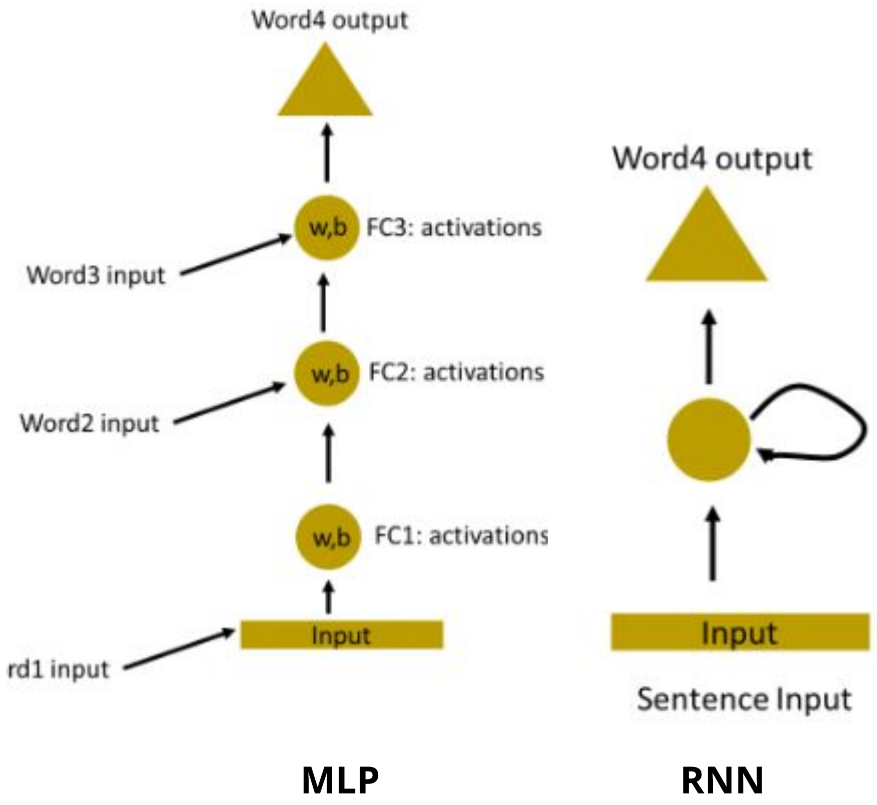
Previous concepts

- ANN
  - SLP
  - MLP
- Activation functions
- FFNN

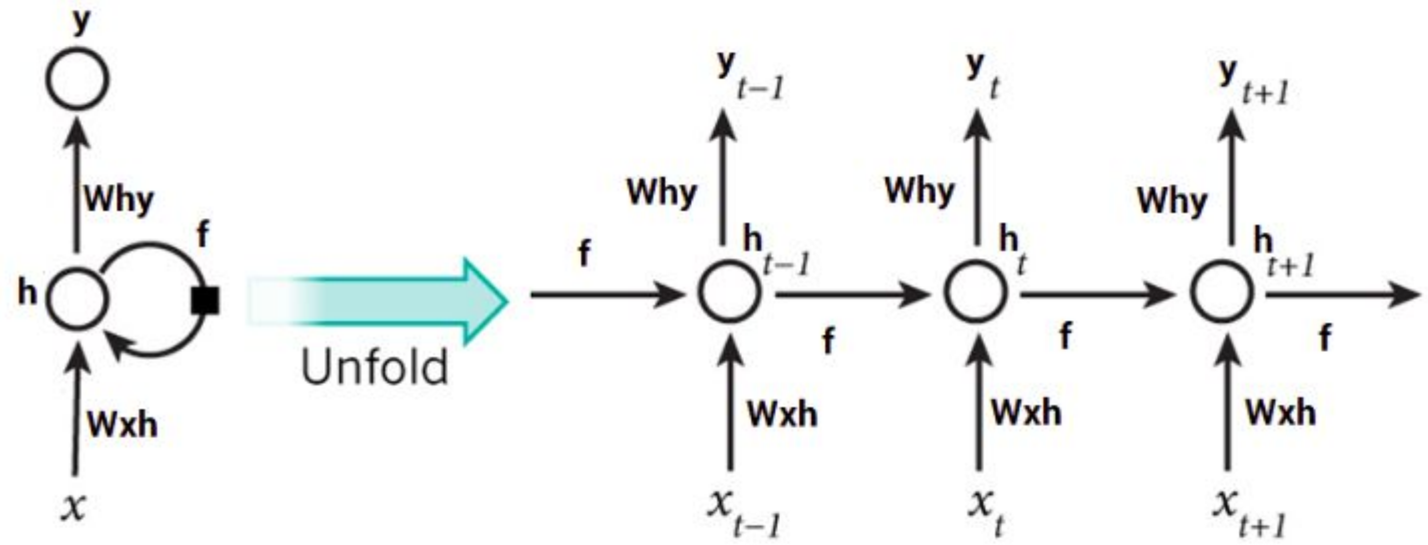
Essence of Recurrent Layers

Let's say the task is predict the next word in a sentence. Let's try to accomplish with MLP.

- Each hidden layer is characterized by its own weights and biases.
- Here, the weights and bias of these hidden layers are different. And hence each of these layers behave independently and cannot be combined together.
- To combine these hidden layers together, we shall have the same weights and bias for these hidden layers.
- We can now combines these layers together, that the weights and bias of all the hidden layers is the same. All these hidden layers can be rolled in together in **a single recurrent layer**.



Unfold the Recurrent Layer

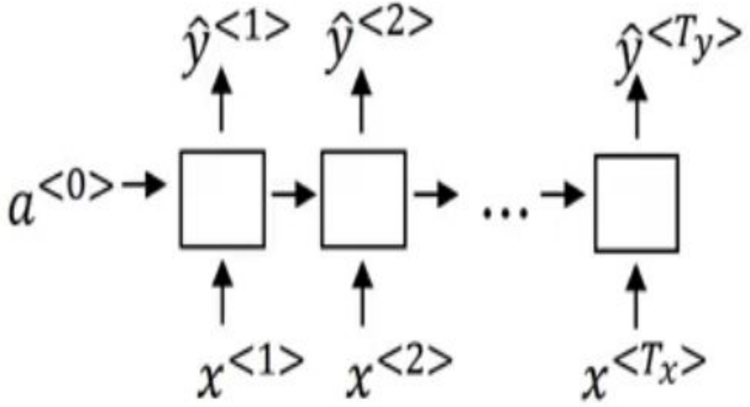


Usage of RNN:

1. We need a representation that will help us to *parse through different sentence lengths as well as reduce the number of parameters* in the model. This is where we use a recurrent neural network
2. A RNN takes the first word ($x^{<1>}$) and feeds it into a neural network layer which predicts an output ($y'^{<1>}$). This process is repeated until the last time step $x^{<Tx>}$ which generates the last output $y'^{<Ty>}$. This is the network where the number of words in *input as well as the output are same*
3. A potential **weakness** of RNN is that it only takes information from the previous timesteps and not from the ones that come later. This problem can be solved using bi-directional RNNs
4. RNN shares the same parameters across all steps. This reflect the fact that we are performing the same task at each step, just with different inputs. This greatly *reduces the total number of parameters* the model has to learn.

Types of RNNs

1. Many to many
2. Many to one
3. One to many



For every input word, we predict a corresponding output word.

Limitations of RNN

1. Can able to deal effectively with **short term dependencies**

   Example: **The color of the sky is __________**

   Produces correct answer: The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue.

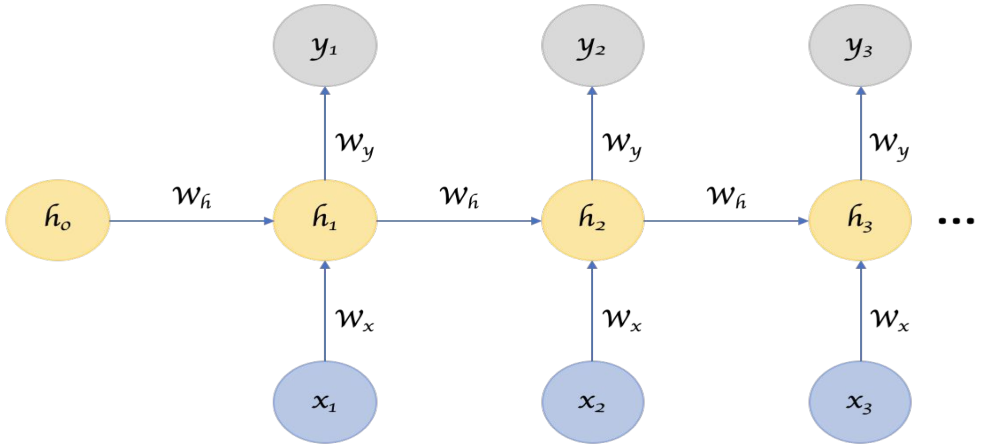
2. May not work effectively for **long term dependencies**

   Example: **I spent 20 long years for the under-previleged kids in spain. I then moved to Africa**

   Ques: I can speak fluent__________.

   The reason behind this is the problem of **Vanishing Gradient.**

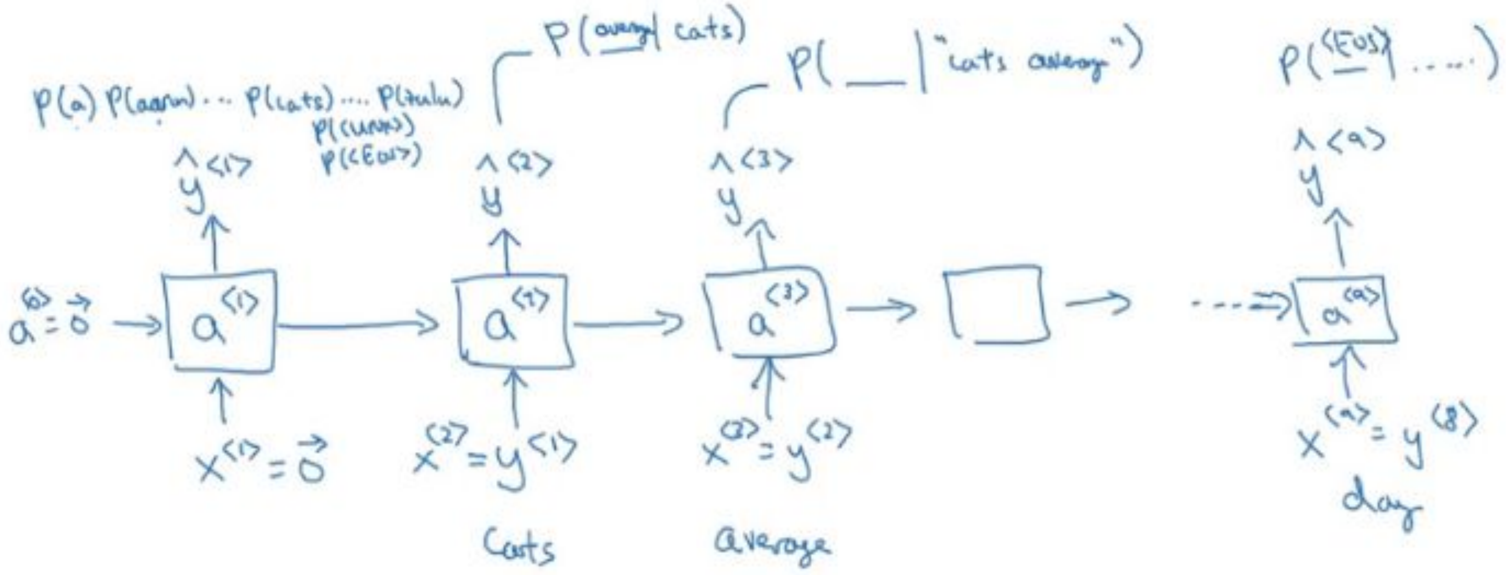
Recurrent Neural Networks



There are primarily two problems with this:

1. Inputs and outputs do not have a fixed length, i.e, input sentences can be of 10 words while others could be < > 10. The same is true for the eventual output
2. We will not be able to share features learned across different positions of text if we use a standard neural network

We take the first input word and make a prediction for that. The output here tells us what is the probability of any word in the dictionary. The second output tells us the probability of the predicted word given the first input word:



Each step in our RNN model looks at some set of preceding words to predict the next word. There are various challenges associated with training an RNN model and we will discuss them in the next section.

Essence of GRU and LSTM

Generally, variants of Recurrent Neural Networks (RNNs), i.e. Gated Recurrent Neural Network (GRU) or Long Short Term Memory (LSTM), are preferred as the encoder and decoder components. This is because they are *capable of capturing long term dependencies* by overcoming the problem of vanishing gradient

Long Short Term Memory (LSTM)

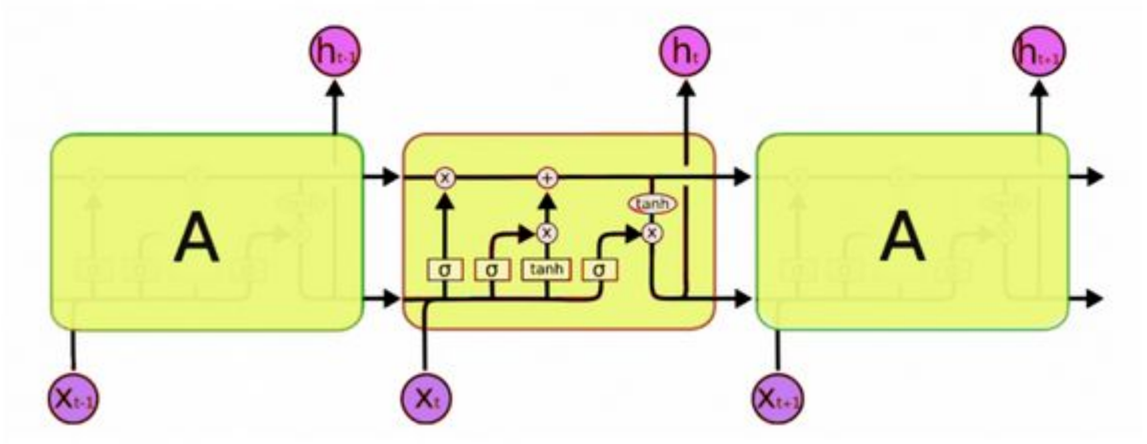
The advantage with GRU is that it has a simpler architecture and hence we can build bigger models, but LSTM is more powerful and effective as it has 3 gates instead of 2.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

These dependencies can be generalized to any problem as:

1. The previous cell state *(i.e. the information that was present in the memory after the previous time step)*
2. The previous hidden state *(i.e. this is the same as the output of the previous cell)*
3. The input at the current time step *(i.e. the new information that is being fed in at that moment)*

LSTM Architecture



1. LSTM compariese of different memory blocks called as cells (rectangular boxes present in image)
2. There are two states that are being transferred to the next cell; the **cell state** and the **hidden state**.
3. Manipulations and additions were responsibility of memory blocks which are called as **gates.**

Forget gate

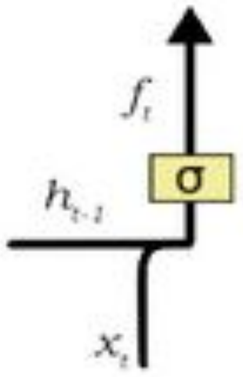
Example: Bob is a nice person. Dan on the other hand is evil.

Functionality: After the first full stop, forget gate realizes that change of context may occur in the next sentence. Place for the subject is evacuated and allocated to ‘Dan’. This process of forgetting the subject is brought about by the forget gate.

Advantage:

1. The forget gate is responsible for **removing the less important** information via multiplication of a filter.
2. Optimizing the performance of the LSTM network

1. Inputs to the forget gate are X_t and h_t-1, output is f_t.
2. h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step
3. Given inputs are multiplied with weights and later they will passed to a sigmoid function.
4. Sigmoid outputs are ranging from 0 to 1.
5. Output 0: that piece of information is discarded/ forget.
6. Output 1: keep the information



Input gate

Example: **Bob knows swimming. He told me over the phone that he served the navy for 4 long years.**

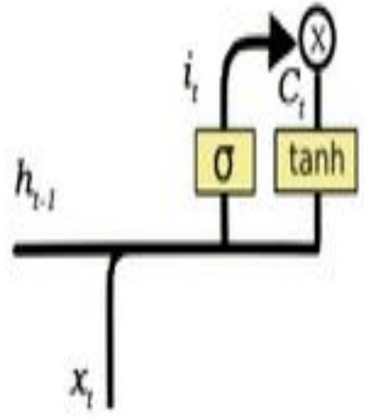
Important information: “Bob” knows swimming and that he has served the Navy for four years.

Less important: He told over the phone

Functionality: The **process of adding new information (decides which values we’ll update)** done via input gate

Addition of information is three step process:

1. Using forget gate regulating what values added to cell state (to filter the information)
2. Creating a vector of all possible values to be added to the cell state (using **tanh** function), the output ranges in b/w -1 to 1.
3. Multiplying the values of regulatory filter (sigmoid) and the created vector (the tanh function). Adding this useful information via addition operation.



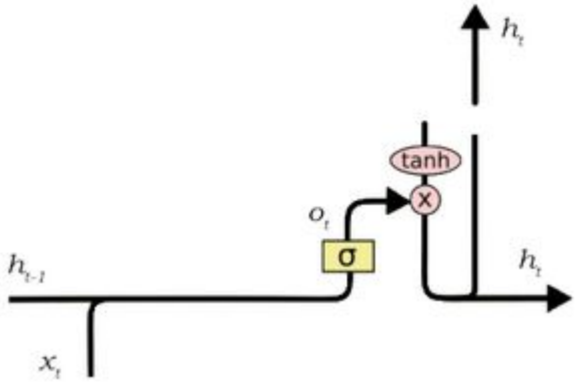
Output gate

Example: Bob fought single handedly with the enemy and died for his country. For his contributions brave__________

---> Numerous possible answers are there, most apt one is Bob.

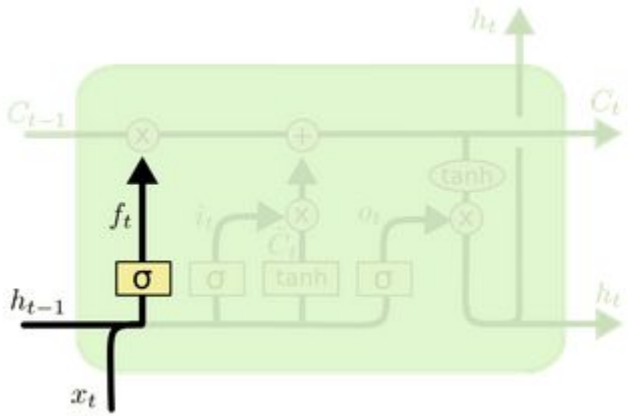
Functionality: The job of **selecting useful information** from the current cell state and showing it out as an output is done via the output gate.

1. Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

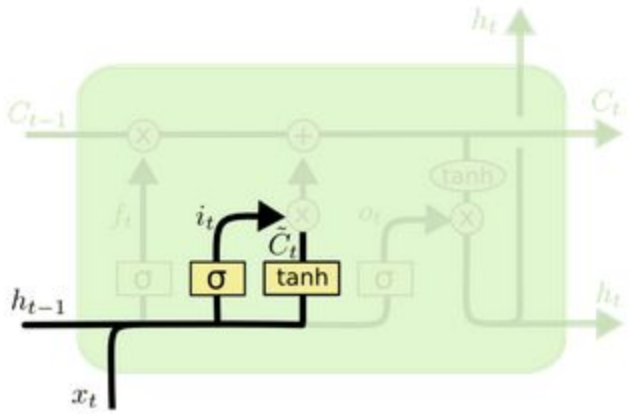


The filter in the above example will make sure that it diminishes all other values but 'Bob'.

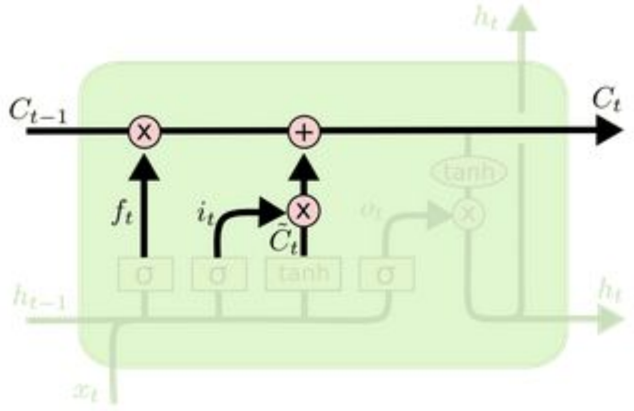
Mathematical description



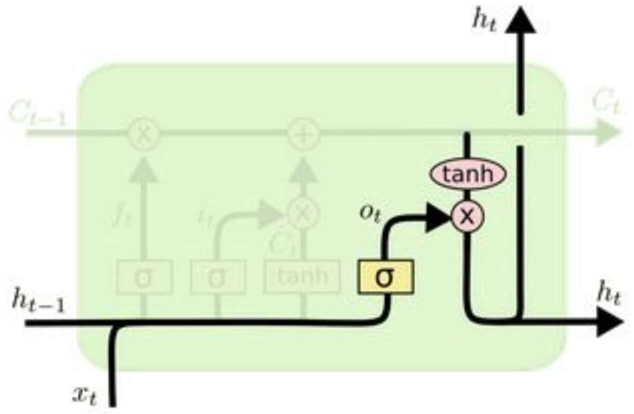
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$



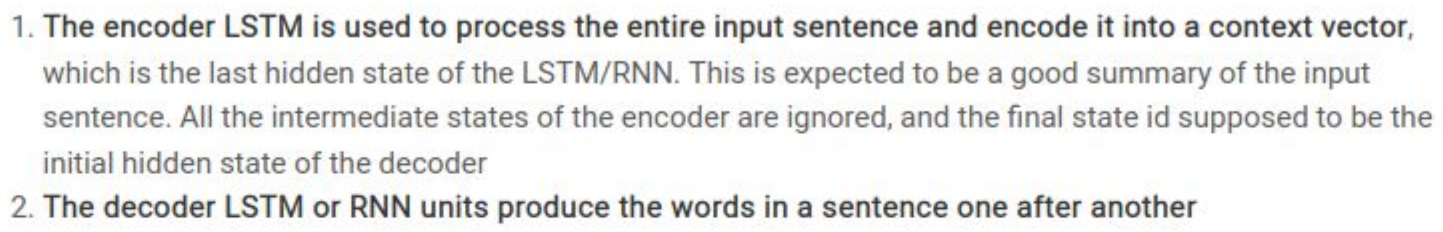
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



1. **The encoder LSTM is used to process the entire input sentence and encode it into a context vector,** which is the last hidden state of the LSTM/RNN. This is expected to be a good summary of the input sentence. All the intermediate states of the encoder are ignored, and the final state id supposed to be the initial hidden state of the decoder
2. **The decoder LSTM or RNN units produce the words in a sentence one after another**

Note: The performance of the encoder-decoder network degrades rapidly as the length of the input sentence increases.

Example: Despite originally being from Uttar Pradesh, as he was brought up in Bengal, he is more comfortable in Bengali.

I want to predict a word “ Bengali”

Words having more weightage: brought up, Bengal

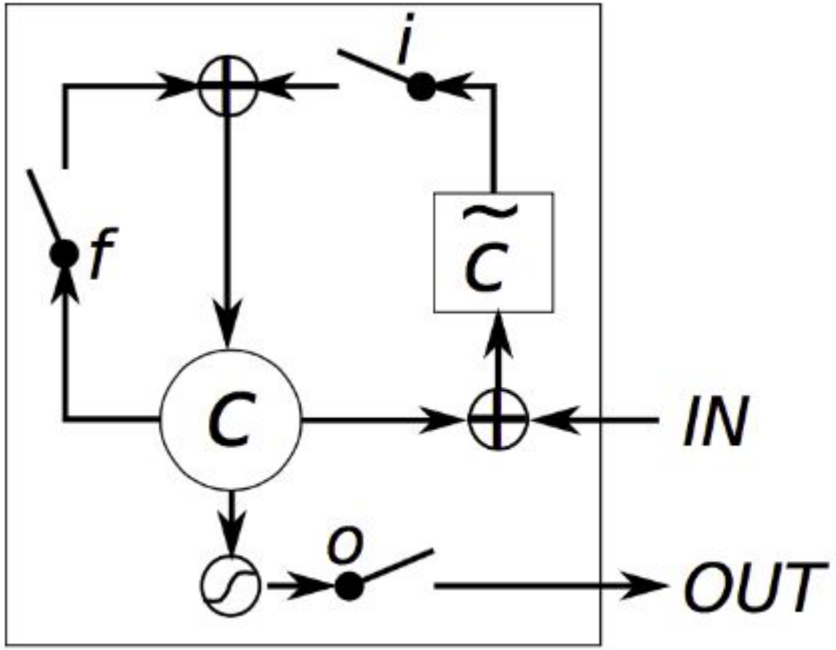
Words can ignore: Uttar Pradesh

Question: Is there any way to keep all the relevant information in the input sequence?

Gated RNNs (GRUs)

GRU has two gates

1. Reset gate (r):
   Determines how to combine the new input with the previous memory
2. Update gate (z):
   How much of the previous memory to keep around
   → Combination of input and forget gate



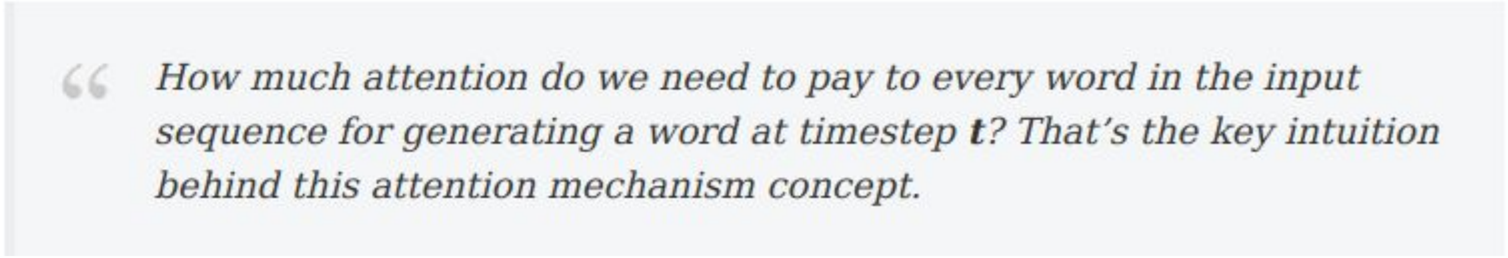
LSTM Gating. Chung, Junyoung, et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling." (2014)

What is the difference b/w LSTM and GRU?

- A GRU has two gates, an LSTM has three gates.
- GRUs don’t possess and internal memory that is different from the exposed hidden state. They don’t have the output gate that is present in LSTMs.
- The input and forget gates are coupled by an update gate and the reset gate is applied directly to the previous hidden state. Thus, the responsibility of the reset gate in a LSTM is really split up into both and .
- We don’t apply a second nonlinearity when computing the output.

z r r z c_t

Intuition Behind Attention Mechanism



> *How much attention do we need to pay to every word in the input sequence for generating a word at timestep **t**? That's the key intuition behind this attention mechanism concept.*

Example:

- **Source sequence:** "Which sport do you like the most?
- **Target sequence:** "I love cricket"

I ---> You

Love ---> Like

Note: Instead of looking at whole sequence, increase the importance of specific parts of the source sentence

Attention

Question: What if we get very long sentences as input, it becomes very hard for the model to memorize the entire sentence.

1. What attention models do is they take small samples from the long sentence and translate them, then take another sample and translate them, and so on
2. We use an alpha parameter to decide how much attention should be given to a particular word while we generate the output.

α<1, 2> = For generating first word, how much attention should be given to the second input word

References

1. https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/?utm_source=blog&utm_medium=comprehensive-guide-text-summarization-using-deep-learning-python (LSTM)
2. https://www.analyticsvidhya.com/blog/2017/01/ultimate-guide-to-understand-implement-natural-language-processing-codes-in-python/ (NLP basics)
3. https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/ (RNN)
4. https://medium.com/swlh/how-to-implement-gradient-descent-in-python-7bbd958ee24b (implementation of Gradient descent)
5. http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/ (GRU)