Bidirectional Encoder Representations from Transformers (BERT)

Rajdipa Chowdhury

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Bidirectional - This model reads text from both directions, left to right as well as right to left.

Encoder - The architecture it employs.

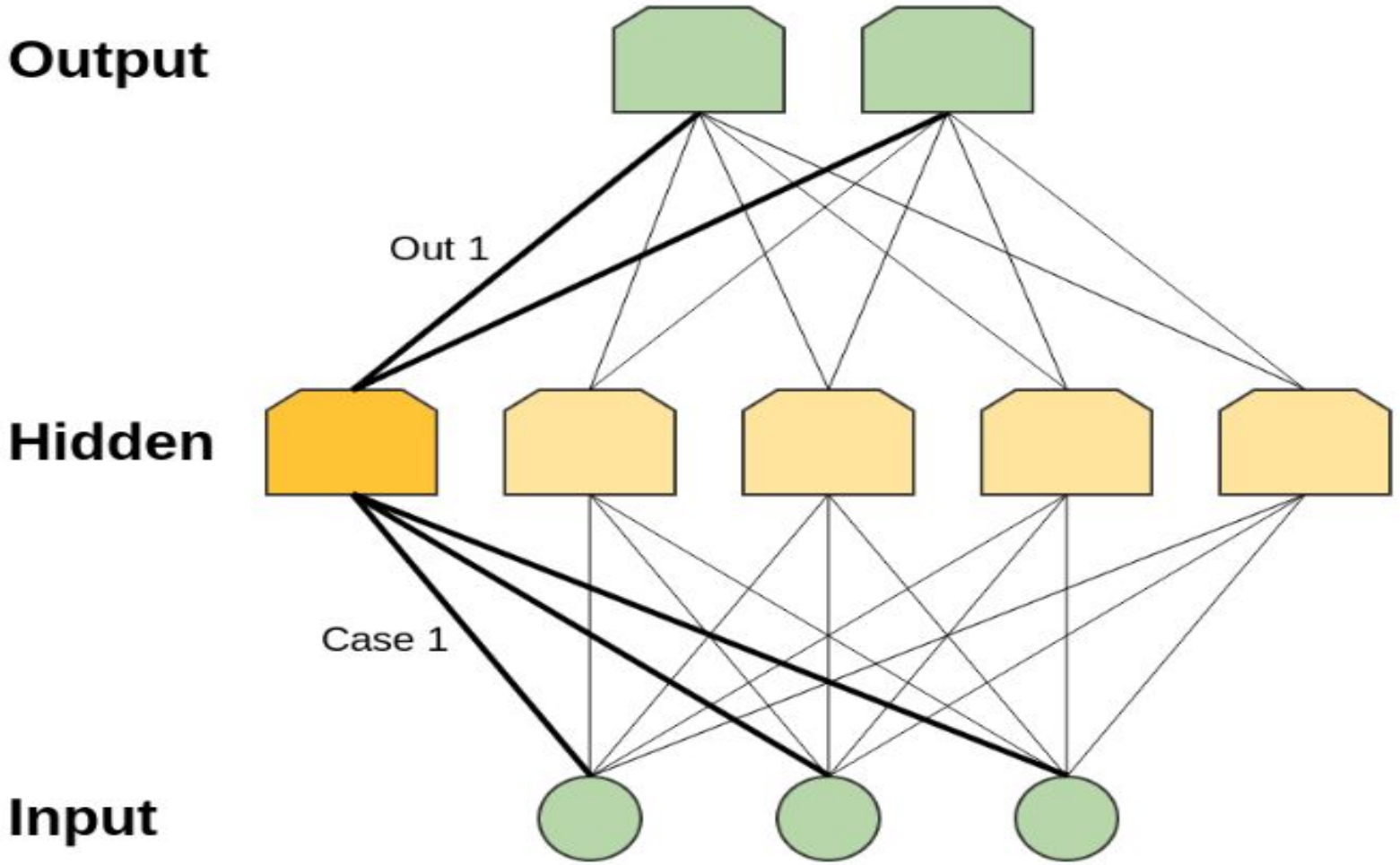
Representations - Encoder decoder architecture is represented using Transformers


Transformers???
But first, we need to understand a few other concepts !

What is Deep Learning?

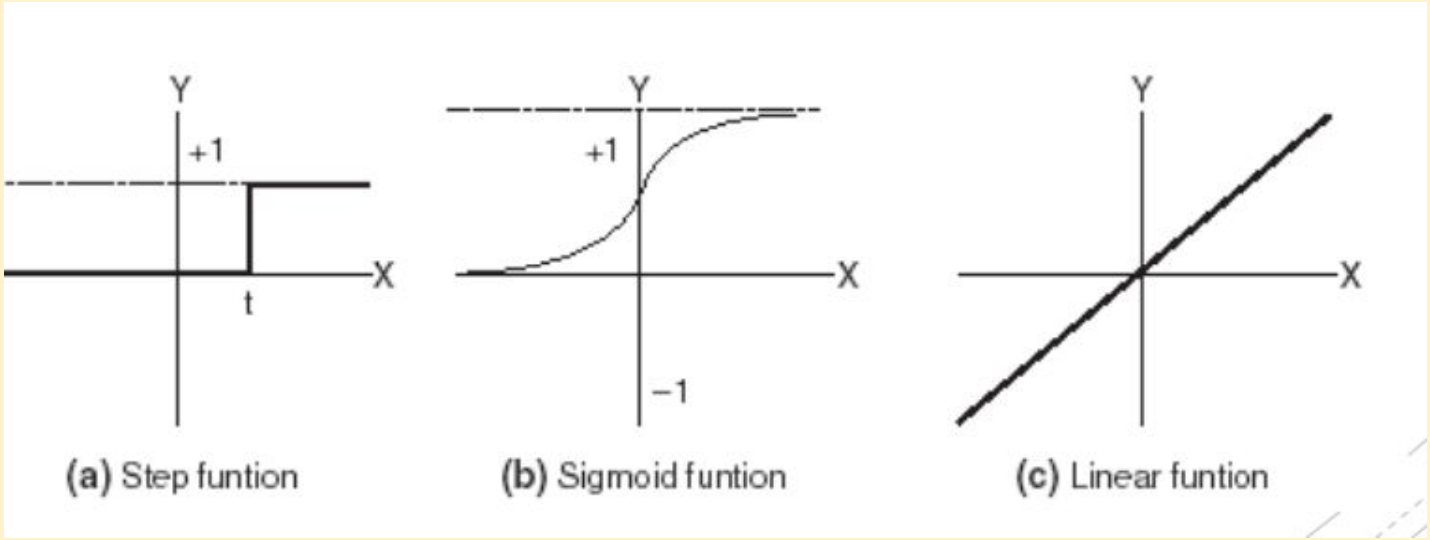
It is a sub field of machine learning concerned with algorithms inspired by the structure and function of the brain.

Neural Networks


Output
Out 1
Hidden
Case 1
Input

- based on biological neurons
- Each neuron in the network receives one or more inputs
- An activation function is applied to the inputs, which determines the output of the neuron – the activation level.

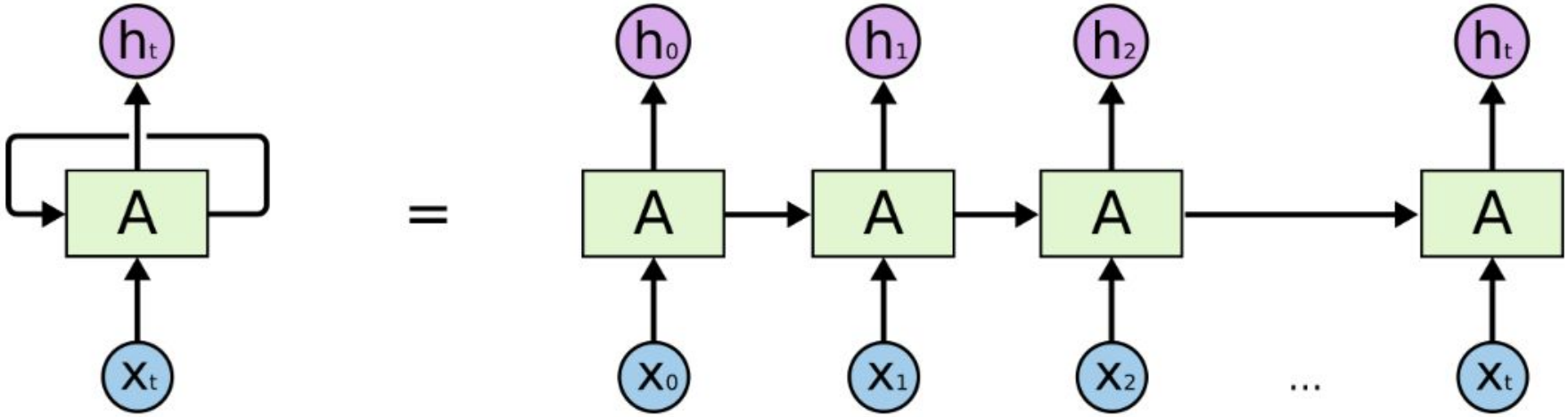
Some Activation functions



Recurrent Neural Network(RNN)

Motivation:

- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output
- Recurrent networks allow us to operate over sequences of vectors
- invokes context (Vanilla Neural Network cannot do this)
- RNNs have loops in them allowing information to persist i.e. the output is fed back into the network.



An unrolled recurrent neural network.

credit: Christopher Olah

The clouds are in the ____

The clouds are in the sky

The clouds are in the sky

When the gap between the relevant information and the place that it’s needed is small, RNNs are can use past information effectively.

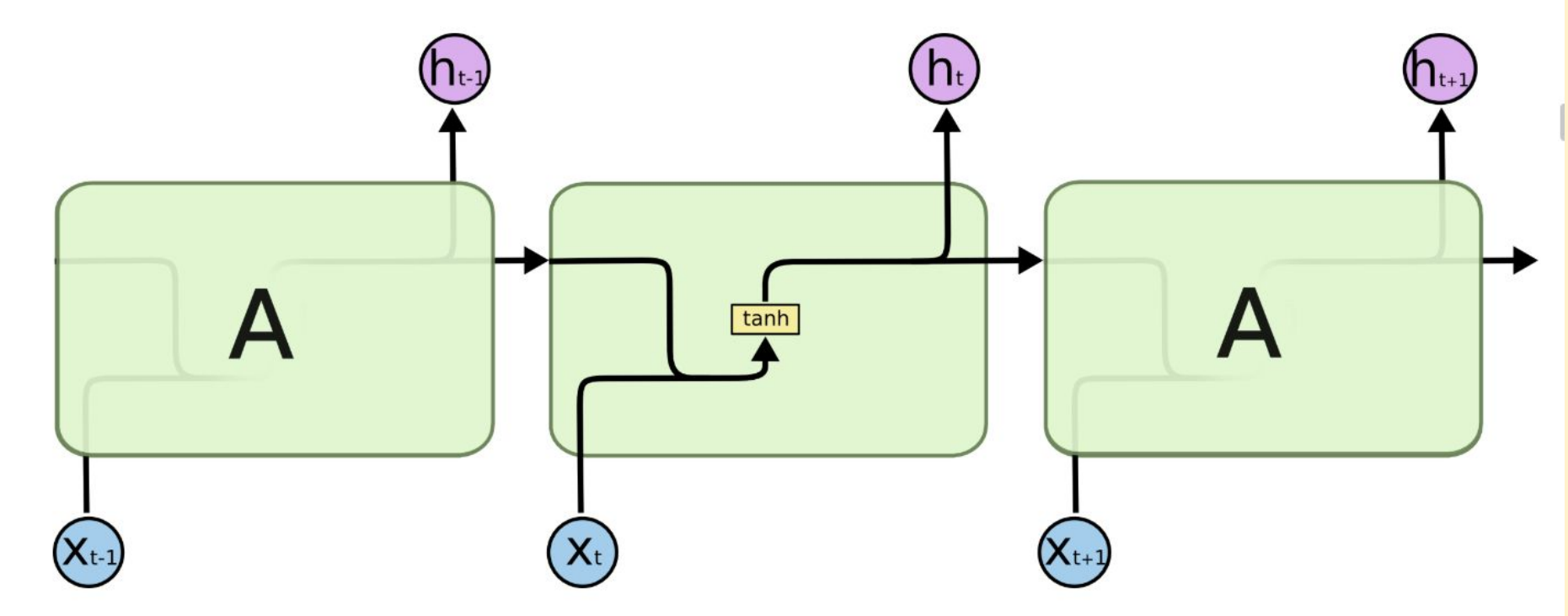
I grew up in France … I speak fluent _____

I grew up in France … I speak fluent french

I grew up in France … I speak fluent french

This requires more context.

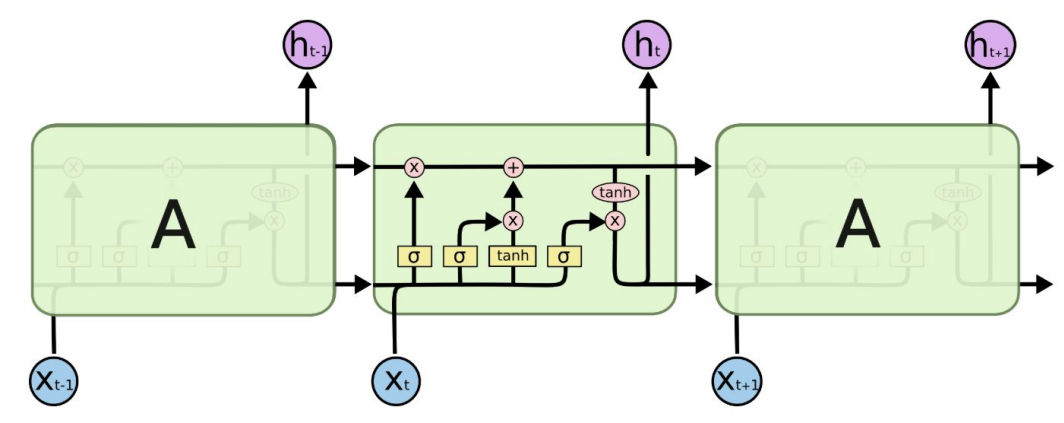
When the gap between the relevant information and the place that its needed grows, standard RNNs are unable to connect the information.



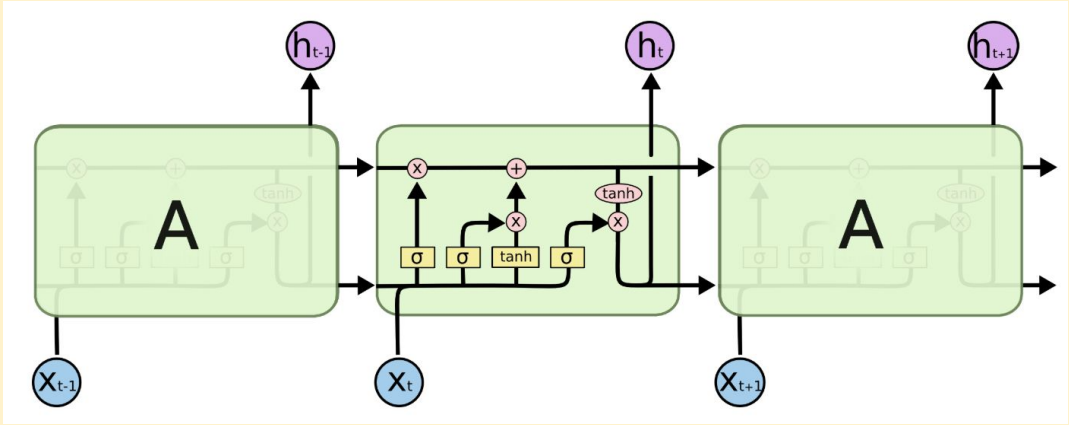
Structure of a standard RNN

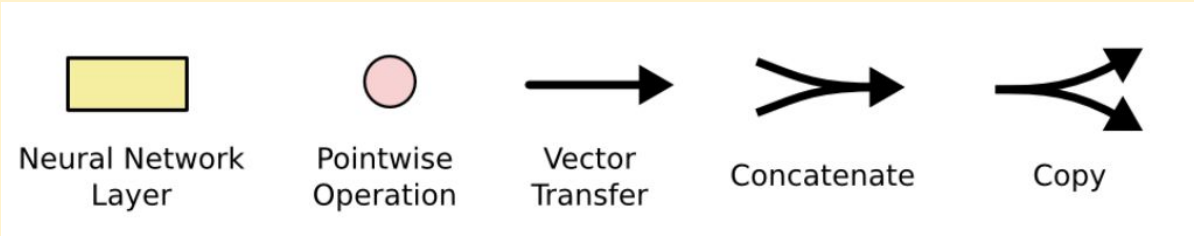
Long Short Term Memory networks (LSTMs)

A special kind of RNN, which was specifically designed to learn long term dependencies.

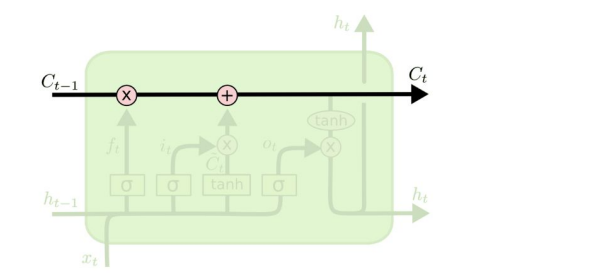


Long Short Term Memory networks (LSTMs)



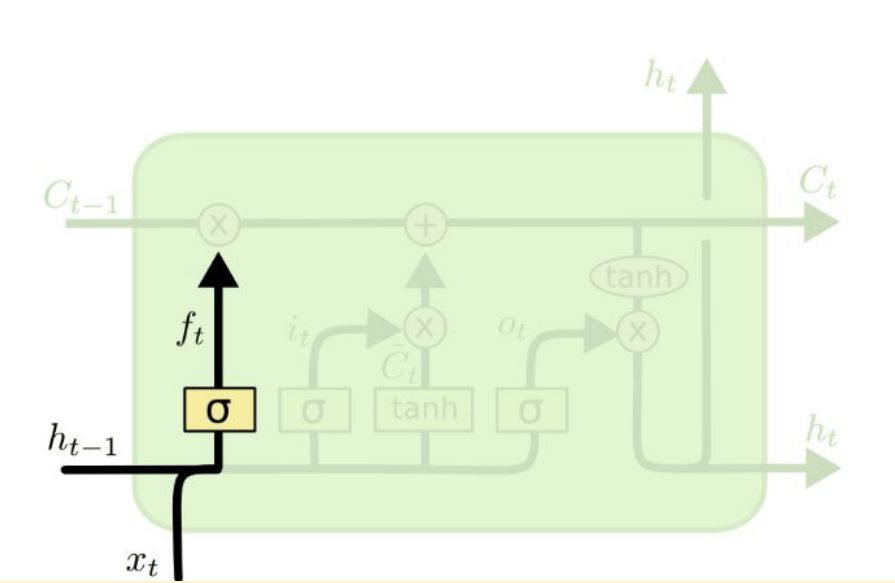


Core Idea



- The key to LSTMs is the cell state.
- Cell state runs straight down the entire chain, with only some minor linear interactions.
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.
- Gates are a way to optionally let information through.
- An LSTM has three of these gates, to protect and control the cell state.

STEP 1: What to forget



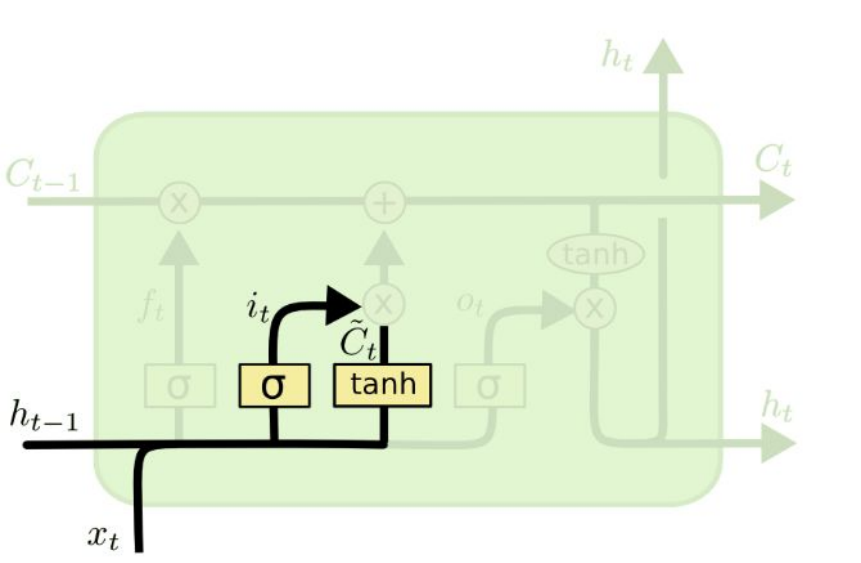
- sigmoid layer called the “forget gate layer.”
- outputs a number between 0 and 1 for each number in the cell state.

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

W and b are weights and biases and [x,y] is the concatenation of x and y.

When we see a new subject, we want to forget the gender of the old subject.

STEP 2: What to store in cell state



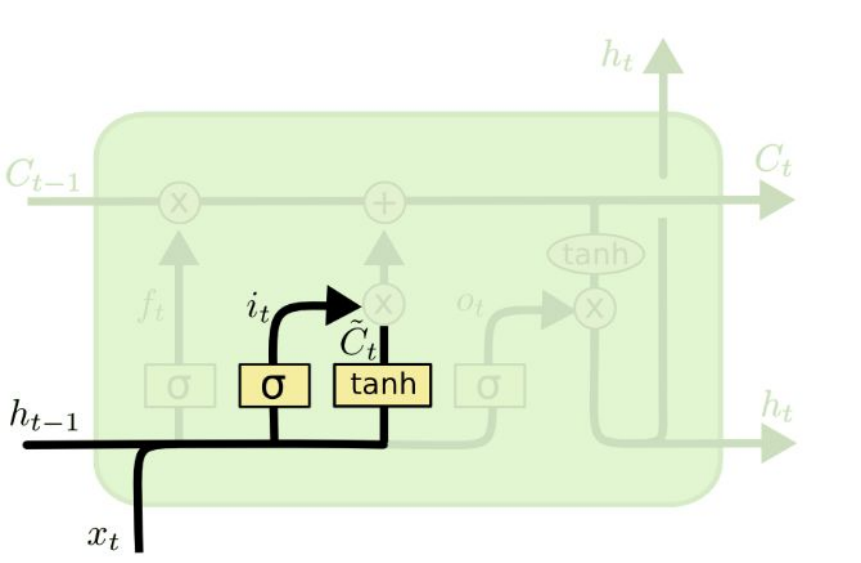
- sigmoid layer "input gate layer" decides which values we'll update
- a tanh layer creates a vector of new candidate values,that could be added to the state
- we'll combine these two to create an update to the state

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

we’d want to add the gender of the new subject to the cell state, to replace the old one we’re forgetting.

STEP 3: Updating the old cell state

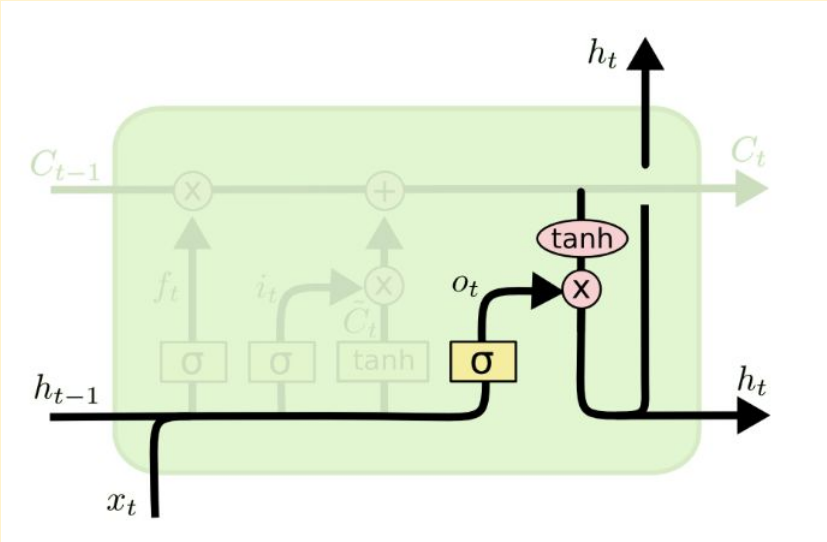


- We multiply the old state by f_t, forgetting the things we decided to forget earlier
- Then we add $i_t * \tilde{C}_t$
- This is the new candidate values, scaled by how much we decided to update each state value.

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

we’d actually drop the information about the old subject’s gender and add the new information.

STEP 4: Output



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

$$h_t = o_t * \tanh\left(C_t\right)$$

- First, we run a sigmoid layer which decides what parts of the cell state we're going to output
- we put the cell state through tanh (to push the values to be between −1 and 1)
- multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

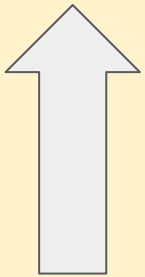
If the input is a subject, lstm might want to output information relevant to a verb, in case that’s what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that’s what follows next.

Drawbacks of LSTMs

- We processed Language using encoder- decoder systems that implement LSTMs.
- The problem with this model is that the encoder has to read the entire input, store it and memorize it.
- After this step, the decoder can output anything.

Drawbacks of LSTMs

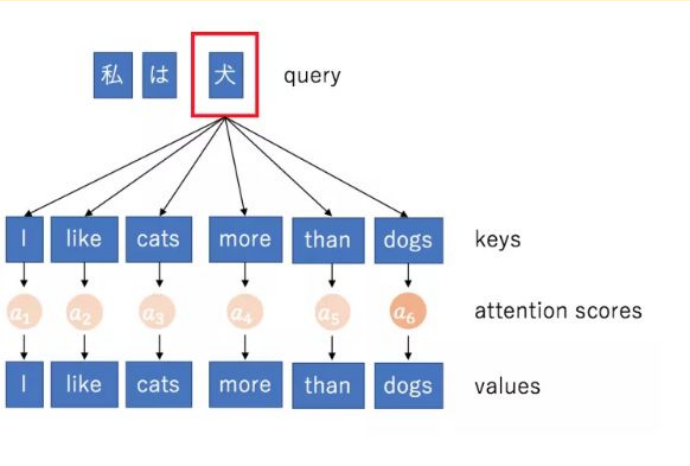
- We processed Language using encoder- decoder systems that implement LSTMs.
- The problem with this model is that the encoder has to read the entire input, store it and memorize it.
- After this step, the decoder can output anything.



This is not how the human brain works and it is very inefficient.

Attention is all you need!

- attention mechanism is used as a way for the model to **focus on relevant information** based on what it is currently processing.

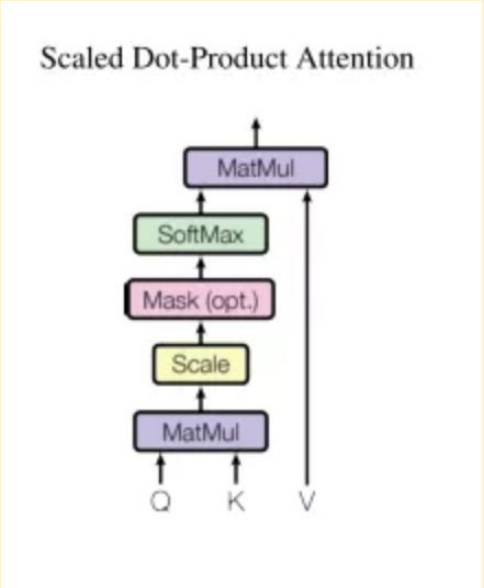


both the keys and values are the source sentence. The attention score represents the relevance, and in this case is large for the word “dog” and small for others.

Scaled dot Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Where Q,K,V are the matrices of Queries,Keys and Values respectively. dk represents the dimensionality of the queries and keys.



The basic attention mechanism is simply a dot product between the query and the key.

Multi-Head Attention

- If we only computed a single attention weighted sum of the values, it would be difficult to capture various different aspects of the input.

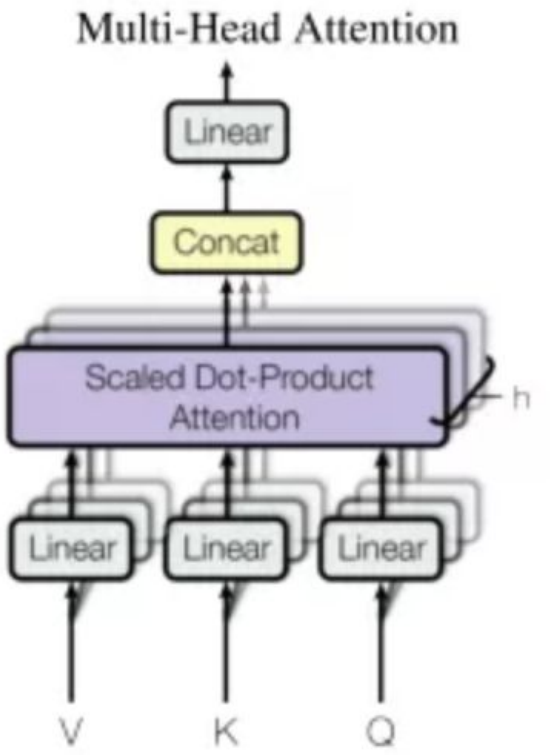
Multi-Head Attention

- If we only computed a single attention weighted sum of the values, it would be difficult to capture various different aspects of the input.

- For instance, in the sentence “I like cats more than dogs”, you might want to capture the fact that the sentence compares two entities, while also retaining the actual entities being compared.

Multi-Head Attention Block:

computes *multiple* attention weighted sums instead of a *single* attention pass over the values



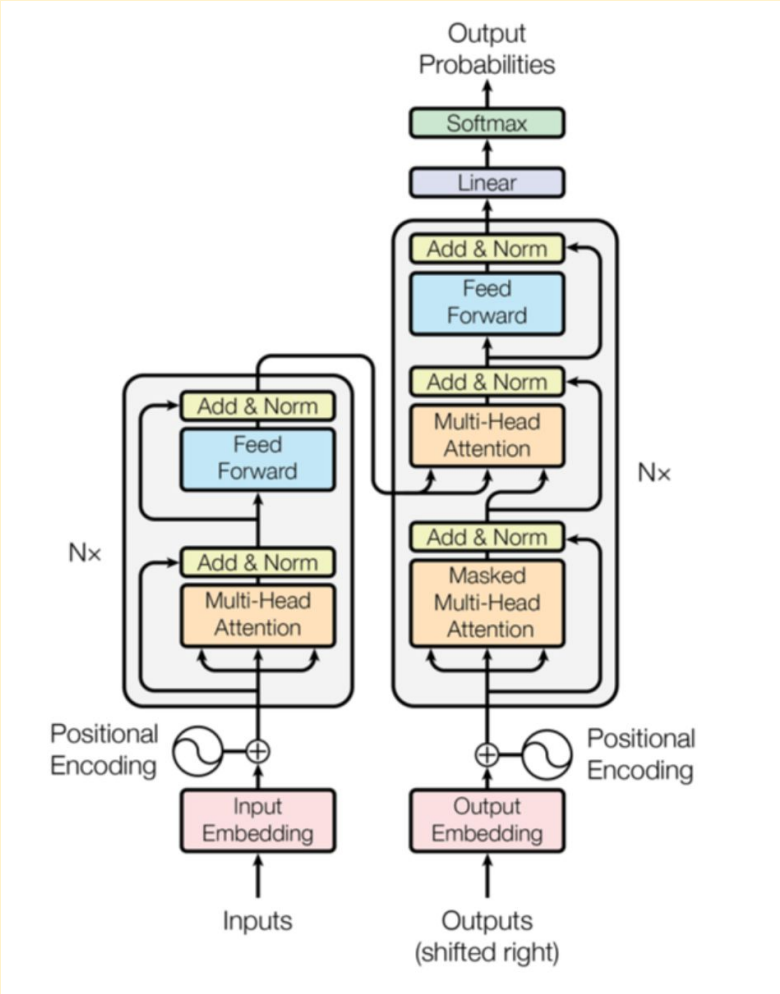
The Multi-Head Attention block

- a unique linear transformation to its input queries, keys, and values, computes the attention score between each query and key, then uses it to weight the values and sum them up.

- The Multi-Head Attention block just applies multiple blocks in parallel, concatenates their outputs, then applies one single linear transformation.

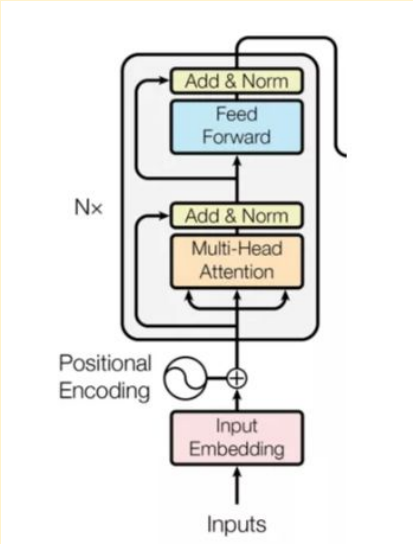
We are finally ready to talk about

Transformers



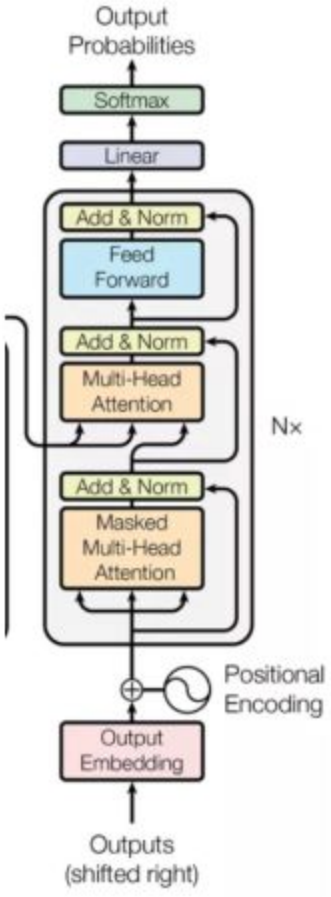
The Transformer - Model architecture

The Encoder



- encoder uses source sentence for its keys, values, and queries
- what each encoder block is doing is **actually just a bunch of matrix multiplications** followed by a couple of element-wise transformations.

- This is why the Transformer is so fast: everything is just parallelizable matrix multiplications.



The Decoder

- decoder uses the encoder's outputs for its keys and values and the target sentence value as its queries.
- decoders are generally trained to predict sentences based on all the words before the current word.
- masked multi head attention block because we need to mask the inputs to the decoder from future time.
- if we give the decoder access to the entire target sentence, the model can just repeat the target sentence

Positional Encodings

explicitly encode the relative/absolute positions of the inputs as vectors

Positional Encodings

explicitly encode the relative/absolute positions of the inputs as vectors

I like cats more than dogs

I like dogs more than cats

BERT’s key technical innovation is applying the bidirectional training of Transformer

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

- bi directional training makes bert more generalizable to different natural language processing tasks.
- bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.
- bidirectional training is possible through “masked language model”

References:

- “Attention is All You Need” : Ashish Vaswani
- Olah, Christopher. “Understanding LSTM Networks.” *Understanding LSTM Networks -- Colah's Blog*, colah.github.io/posts/2015-08-Understanding-LSTMs/.
- “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” : Jacob Devlin