EMBEDDED LEARNING

SESSION 3

Long Short-Term Memory Networks (LSTM)

27/01/25

By:

Priyanka Saxena,

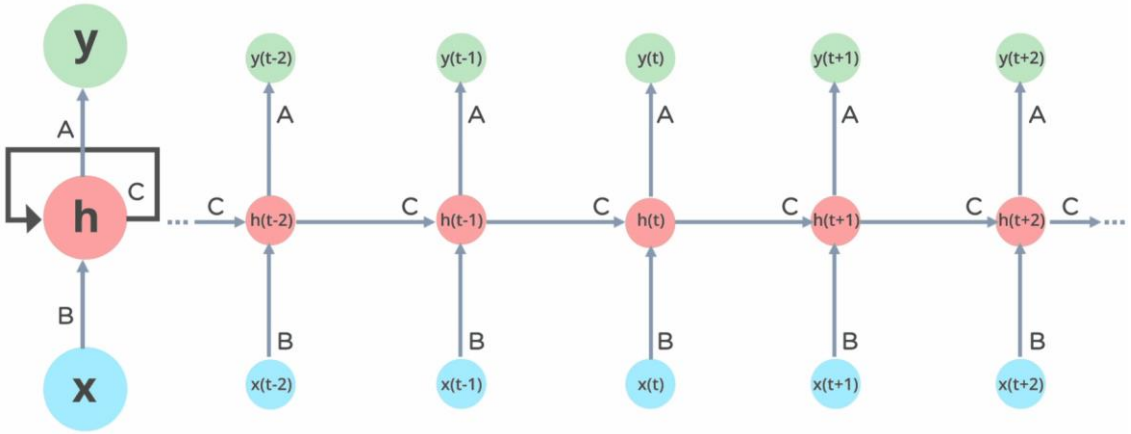
Assistant Professor, CSE, KMIT.

OUTLINE

- Background
- Definition of LSTM
- Need for LSTM
- Internal structure of LSTM
- Hands on simple LSTM.

BACKGROUND

- What are Recurrent Neural Networks?
- RNNs process sequential data, where the output of one step is fed as input to the next.



BACKGROUND

- What are the challenges with RNN?
- Exploding Gradient descent
  - Gradients grow too large, causing instability.
- Vanishing Gradient descent
  - Gradients become very small, making learning difficult.

NEED FOR LSTM

Why RNN fail for long sequences?

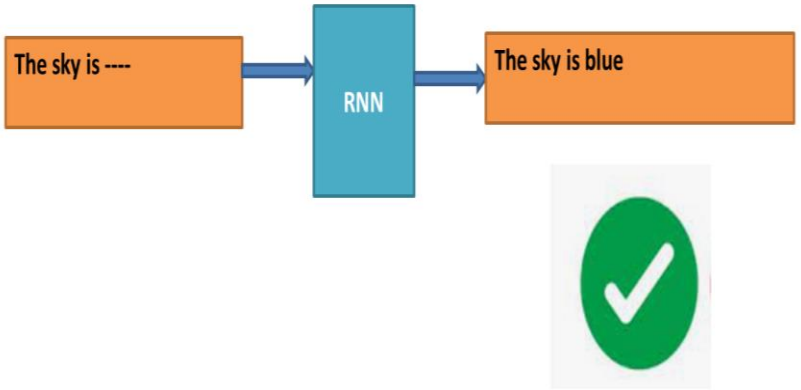
- Generally, in RNN, the earlier layers of the network don't learn much due to vanishing gradient descent problem.
- This leads RNN to "**FORGET**" and thus have a short term memory in "long sequences".
- **RNN cannot retain long term sequences in the memory!**
- LSTMs introduce gates to manage memory and handle long sequences effectively.

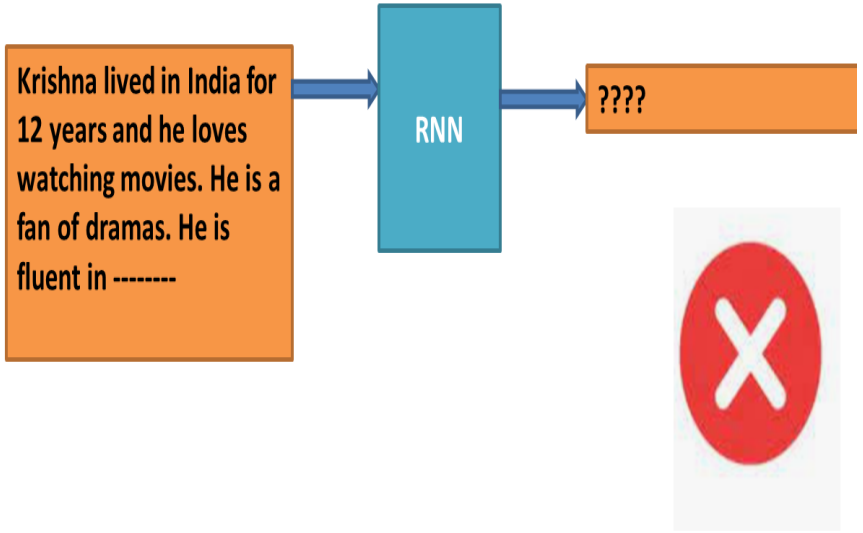
Q&A

- Why do you think “memory” is crucial for handling sequential data?
- What’s the difference between recognizing a word and understanding the context of a sentence?
  - **Recognizing words is straightforward, but understanding context requires memory of previous words—this is where LSTMs excel. Thus Memory cell is the requirement.**

NEED FOR LSTM

- While we use Vanilla RNN , have a look at the following example:





NEED FOR LSTM

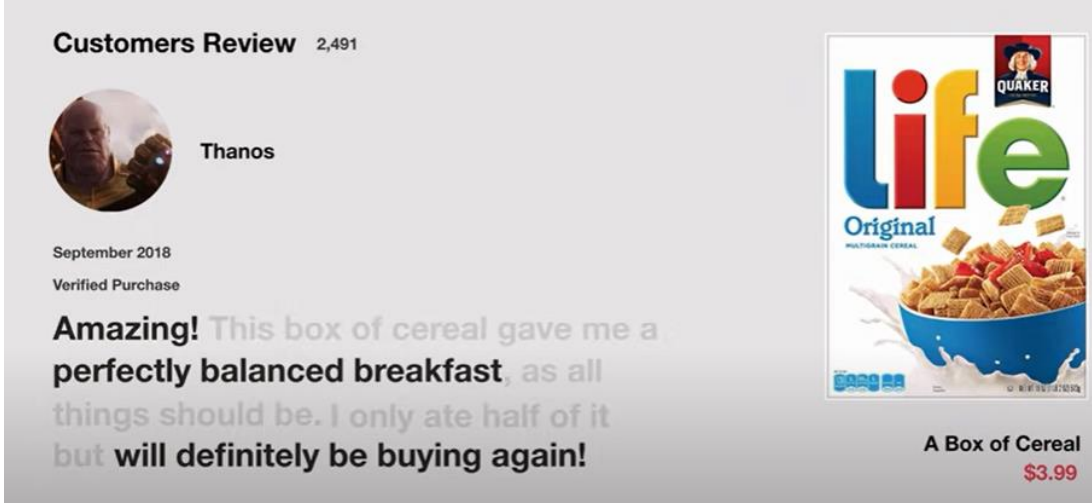
Reason of Failure:

- Long term sequences
- Inherent relationship between each sentences.

SOLUTION : LSTM

Q&A

- **How do you determine to buy something online??**
- Probably you will read some reviews about the product an buy!!
- You will subconsciously remember important keywords like
  - Amazing
  - Wonderful
  - Definitely buy again



Q&A

- Words you won't concentrate much:
  - They
  - Should
  - The
  - Like
  - This
- When recalled you would remember all key words and tell the feedback!! **That's what LSTM does!!**
- **They keep only relevant information to make the predictions!**

NEED FOR LSTM

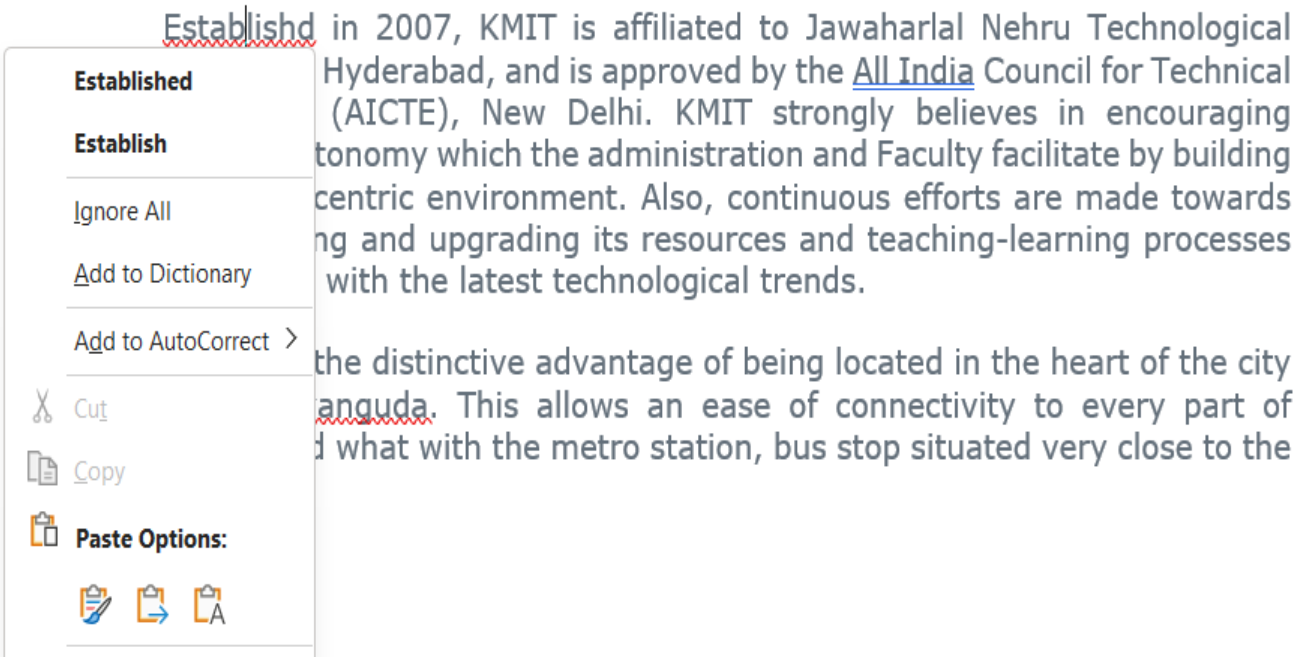
- LSTM have the mechanism with **gates** which
  - can regulate the flow of information
  - Recognise which data is important to keep / throw away!!
  - Uses relevant information to make predictions

NLP - APPLICATIONS

- Spell correction
- Machine translation
- Chatbot
- Sentiment analysis
- Email filtering
- Text summarization

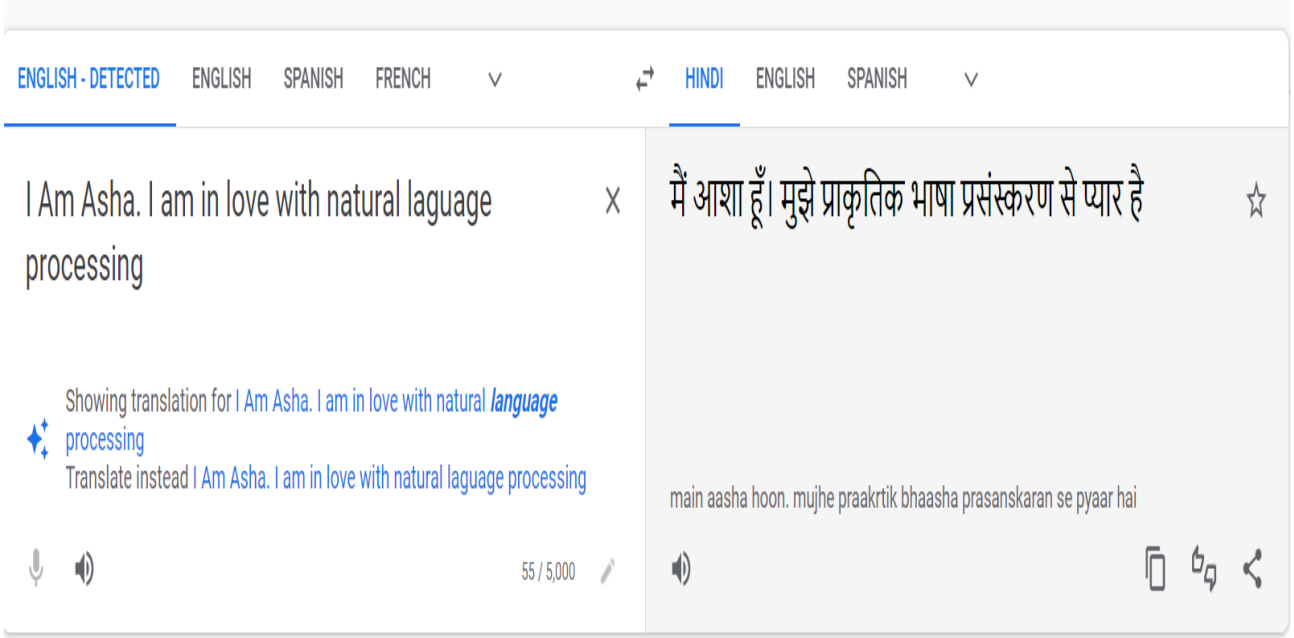
Spelling correction

Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.



Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

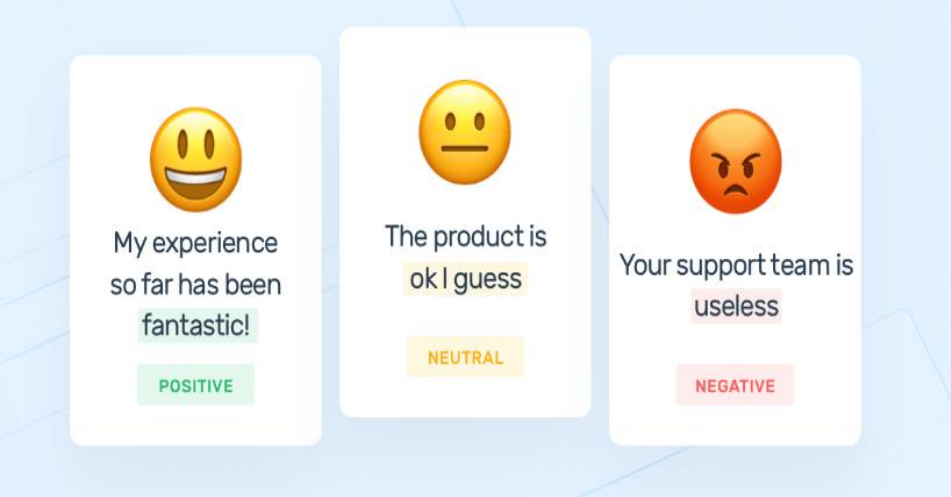


Chatbot

Implementing the Chatbot is one of the important applications of NLP. It provides the customer chat services

Sentiment Analysis

Sentiment Analysis is also known as **opinion mining.** It is used on the web to analyse the attitude, behaviour, and emotional state. This application is implemented through a combination of NLP and statistics by assigning the values to the text, identify the mood of the context.

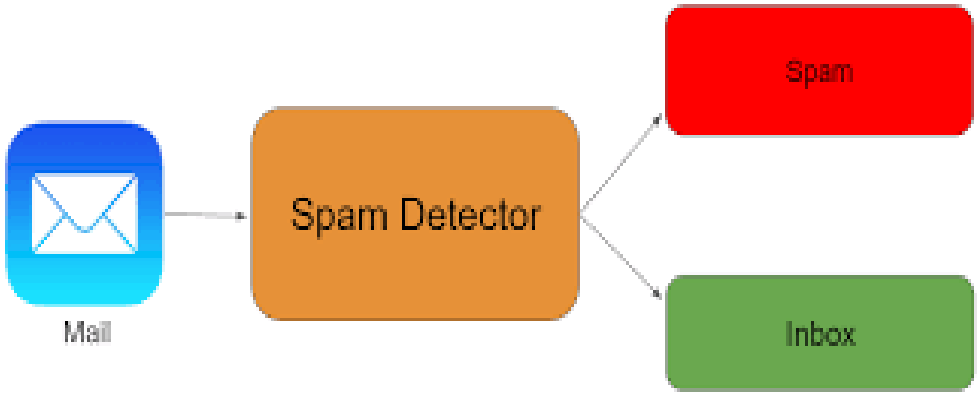


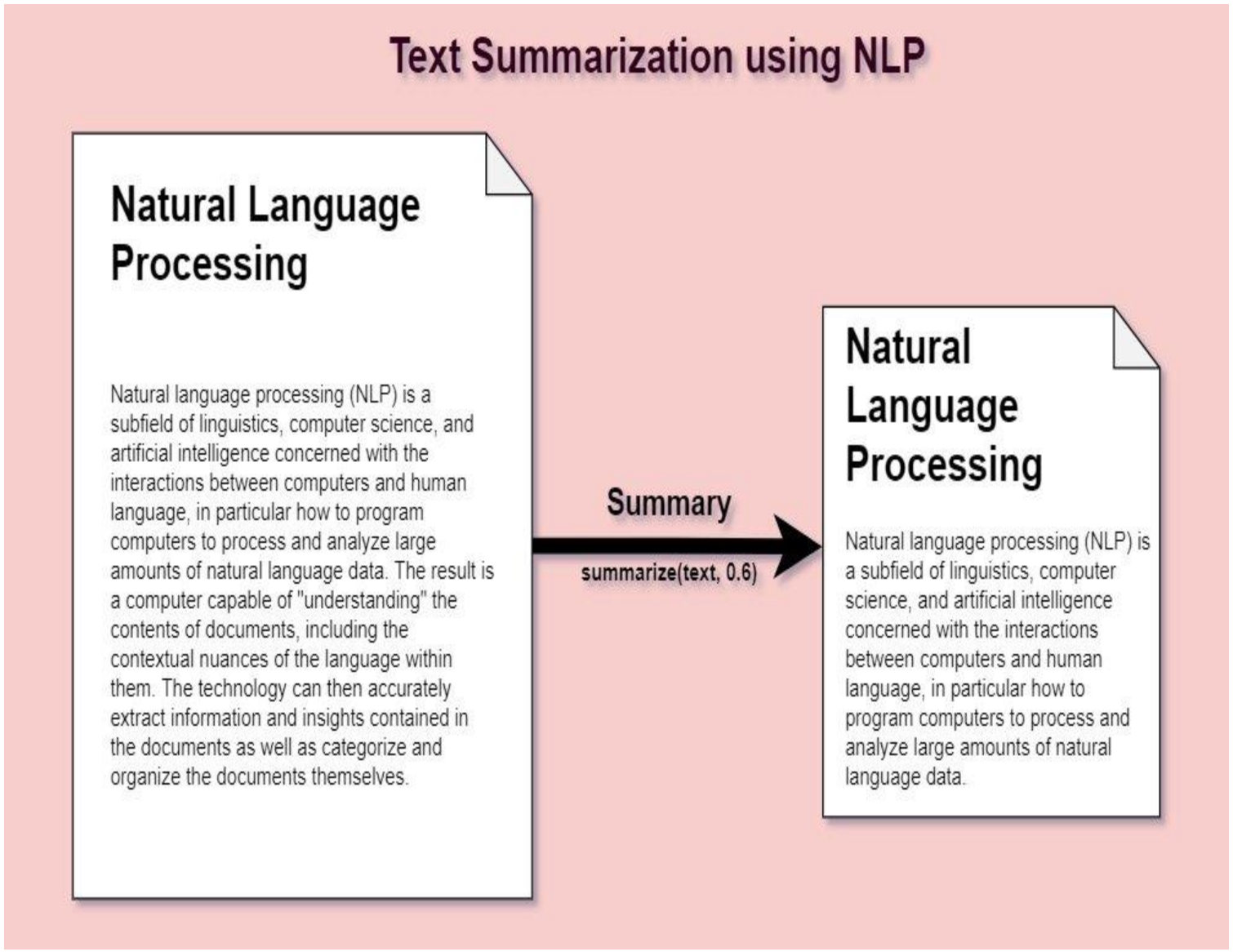
Email Filtering

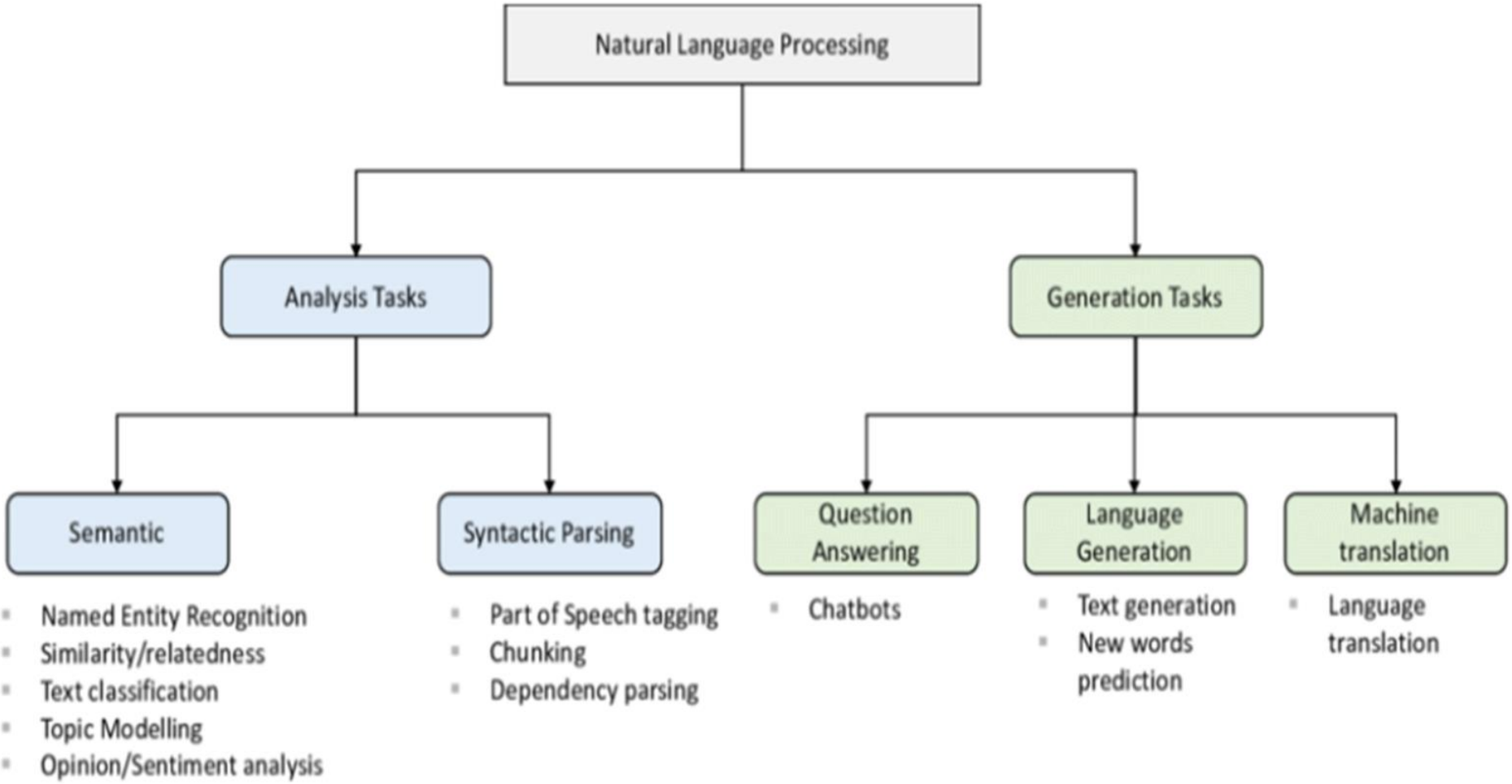
It checks/ analyzing incoming emails for red flags that signal spam or phishing content and then automatically moving those emails to a separate folder.

eg: look for common trigger words,

such as "free" and "earn money".




Text Summarization using NLP
Natural Language Processing
Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. The result is a computer capable of "understanding" the contents of documents, including the contextual nuances of the language within them. The technology can then accurately extract information and insights contained in the documents as well as categorize and organize the documents themselves.
Summary
summarize(text, 0.6)
Natural Language Processing
Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.


Natural Language Processing
Analysis Tasks
Generation Tasks
Semantic
Syntactic Parsing
Question Answering
Language Generation
Machine translation
Named Entity Recognition
Similarity/relatedness
Text classification
Topic Modelling
Opinion/Sentiment analysis
Part of Speech tagging
Chunking
Dependency parsing
Chatbots
Text generation
New words prediction
Language translation

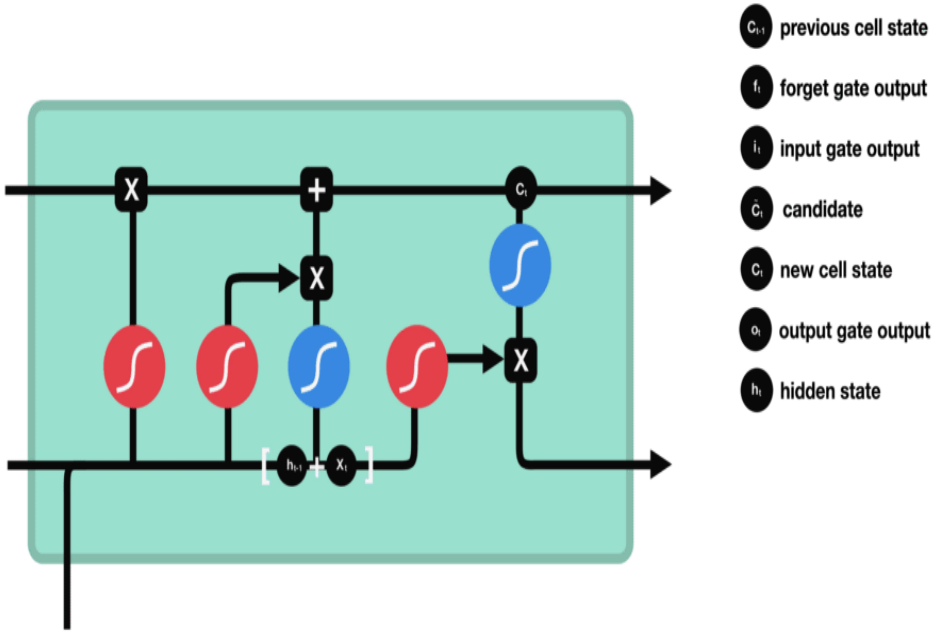
LSTM

- Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning.
- It was proposed in 1997 by **Sepp Hochreiter** and **Jurgen schmidhuber**.
- Unlike standard feed-forward neural networks, LSTM has feedback connections.
- It can process not only single data points but also entire sequences of data (such as speech or video).

INTERNAL STRUCTURE OF LSTM

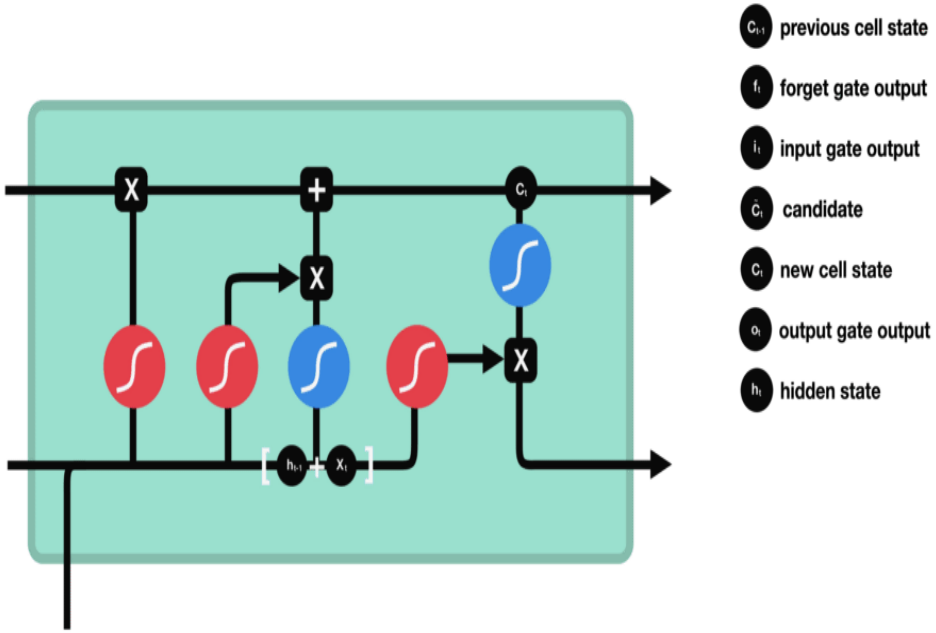
Key components:

- The input gate controls what information is added to the memory cell.
- The forget gate controls what information is removed from the memory cell.
- The output gate controls what information is output from the memory cell.



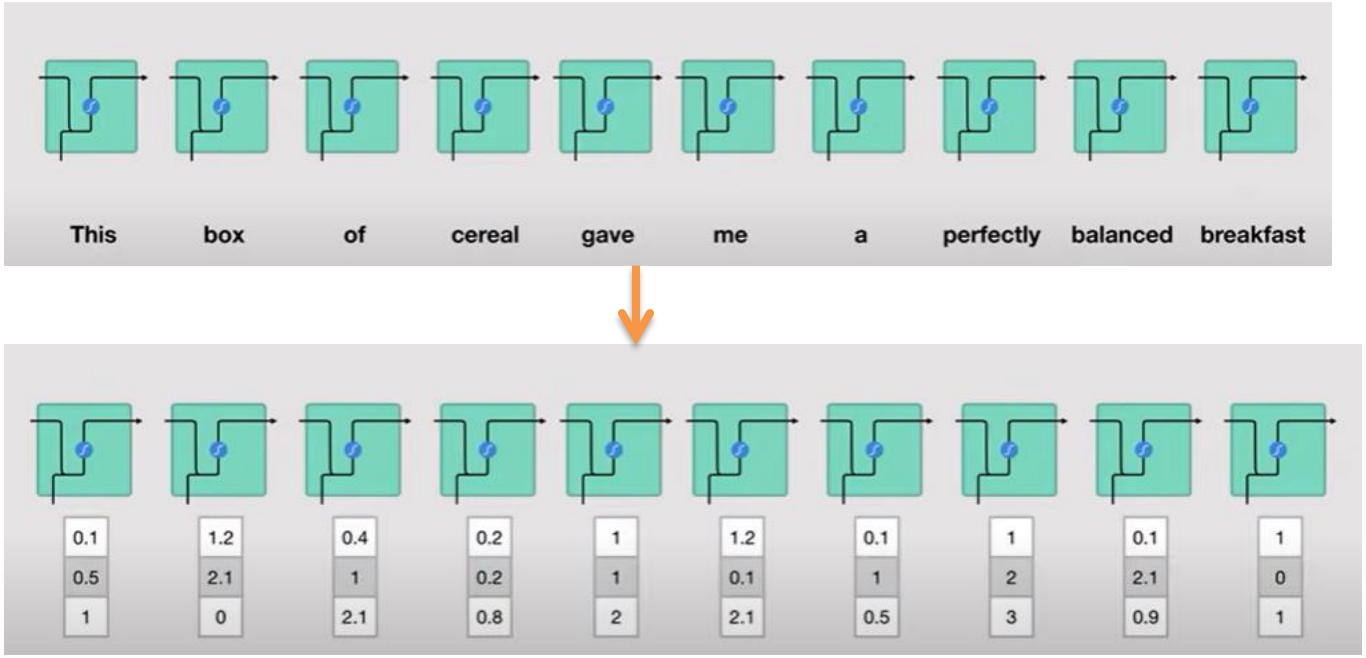
INTERNAL STRUCTURE OF LSTM

- The LSTM maintains a hidden state, which acts as the short-term memory of the network.
- The hidden state is updated based on the input, the previous hidden state, and the memory cell's current state.



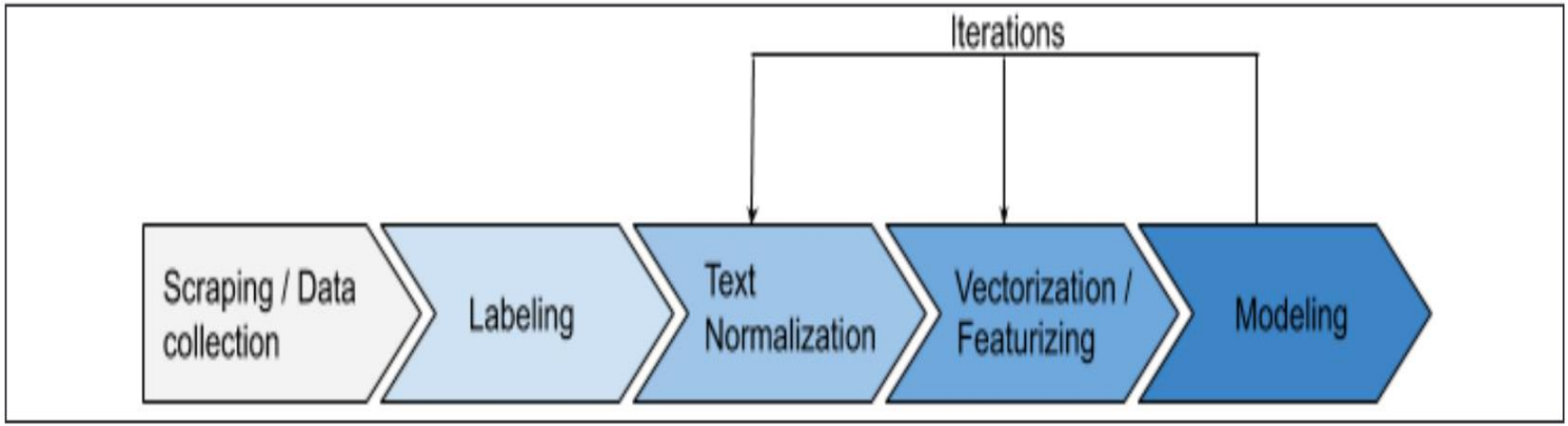
Simple process.. RNN vs LSTM

- Before going to actual architecture lets understand simple working of cells of RNN & LSTM individually.
- Lets consider RNN: first words get transformed into machine readable vectors



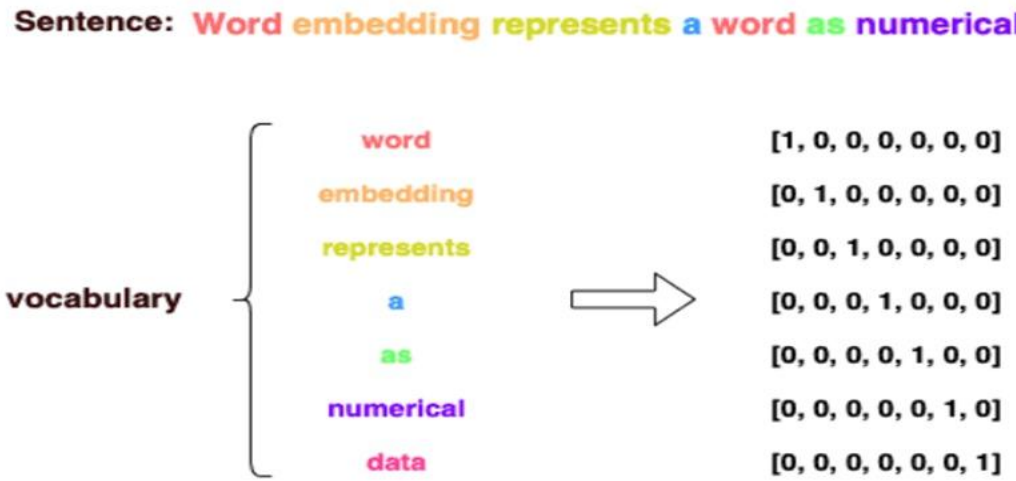
Text Pre-Processing

- A typical text processing workflow is:



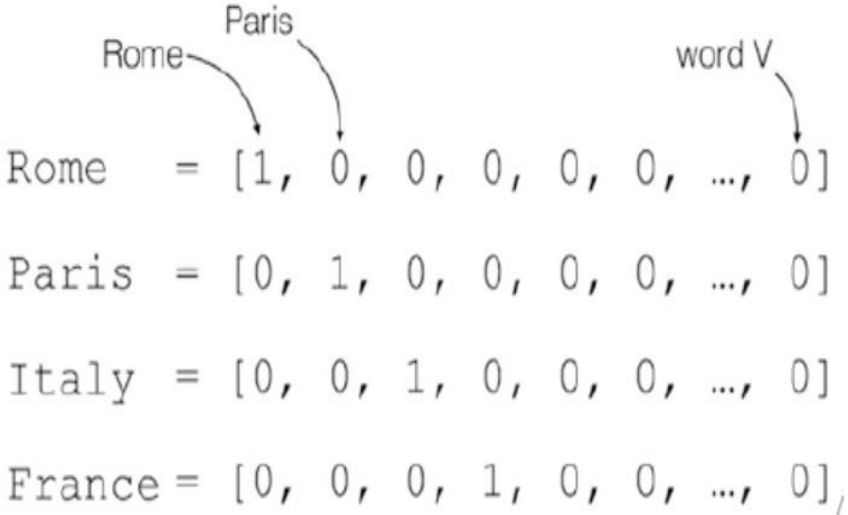
What Are Word Embeddings?

- The word generated on each timestamp is a character, not a numerical value.
- Neural networks cannot directly process string data.
- Conversion of words or characters into numerical values becomes particularly critical, in neural network applications like NLP.
- **Word embeddings** are a way to represent words as dense, continuous-valued vectors in a fixed-dimensional space.



Sparse Representation (One-Hot Encoding)

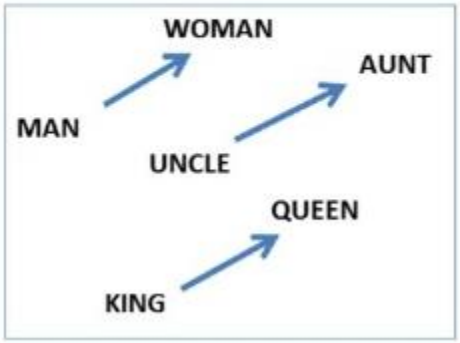
- The one-hot encoding vector is high-dimensional and extremely sparse, with a large number of positions as 0s.
- Therefore, it is computationally expensive and also not conducive to the neural network training.
- It ignores the **semantic relevance inherent in words.**

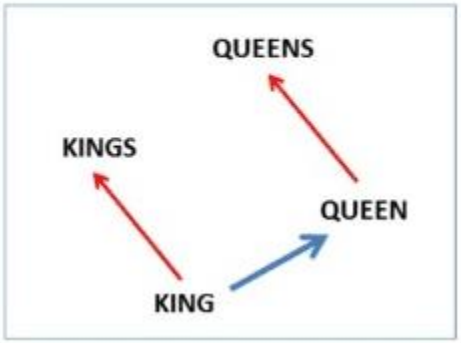


Dense Representation (Embeddings)

- Example: Each word is represented as a vector of, say, 100 dimensions.
- These vectors are learned in such a way that semantically similar words (e.g., "king" and "queen") are close to each other in the vector space.
- Meaningful relationships are encoded:*king - man + woman ≈ queen*.

vec("man") – vec("king") + vec("woman") = vec("queen")





Importance of word embeddings

- Capture Semantic Similarity
- Dimensionality Reduction
- Context Understanding
- Transfer Learning etc.

How Does the Embedding Layer Work?

Input Tokenization

Embedding Lookup

- Emb
- Eac
  mat
- The
  eac

Example sentence: I am happy

- {‘I’, ‘am’, ‘happy’}
- {‘I’: 0, ‘am’: 1, ‘happy’: 2}
- A sentence like "I am happy" is tokenized into [0, 1, 2]
- For a vocabulary of 10,000 words and embedding size of 100, the embedding matrix has dimensions 10,000×100 (assumption)
- For the word "happy" (index 2), the embedding layer retrieves the vector from the 2nd row of the matrix:

Embedding[2]=[0.12,−0.3,0.56,…,0.91]

(100-dimensionalvector)

How Does the Embedding Layer Work?

- An embedding matrix can be randomly initialised or Pre- trained Embeddings like **Word2Vec, FastText** or **GloVe etc** can be used.
- The **embedding size** is a hyperparameter (e.g., 50, 100, 300).Larger dimensions capture more information but increase computation.
- Input to an Embedding layer is a batch of sentences with word indices.
- Example: Batch of 2 sentences, each with 4 words:

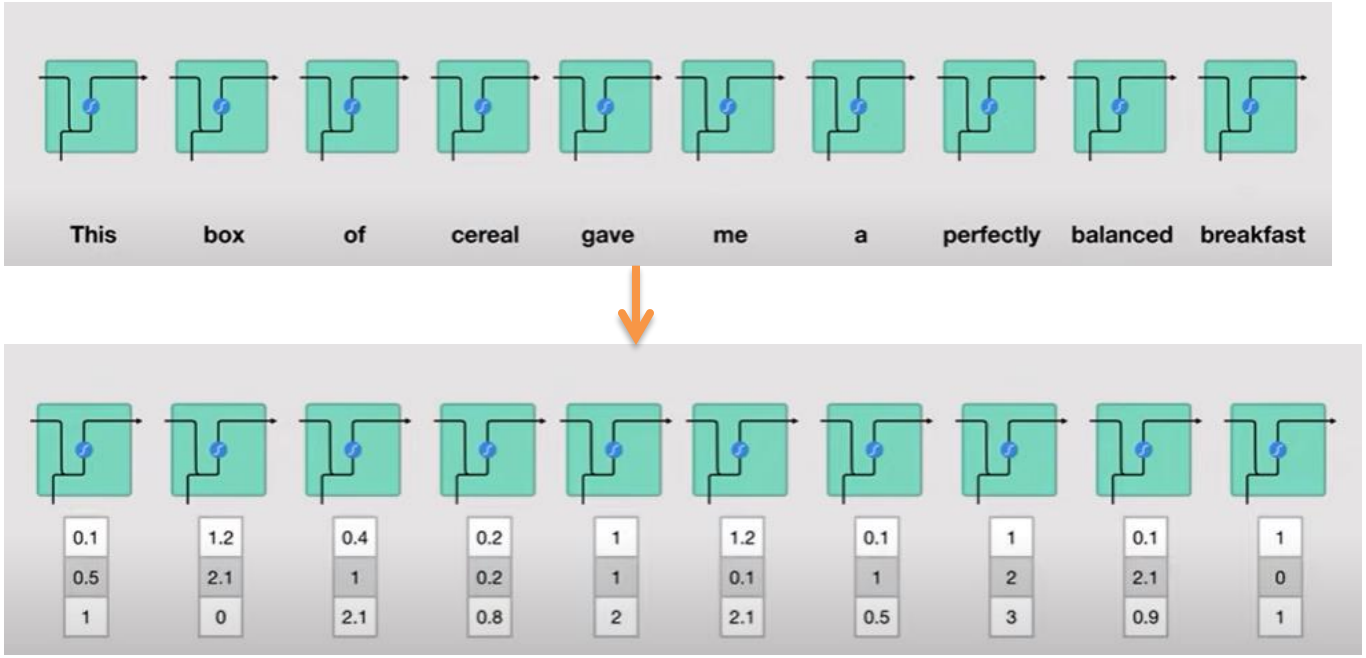
  [[0, 1, 2, 3], [4, 5, 6, 7]]
- Output: A 3D tensor with shape:
  (Batch Size)×(Sequence Length)×(Embedding Dimension)
- Example: For batch size = 2, sequence length = 4, embedding size = 100 the Output shape = 2x4x100

How Does the Embedding Layer Work?

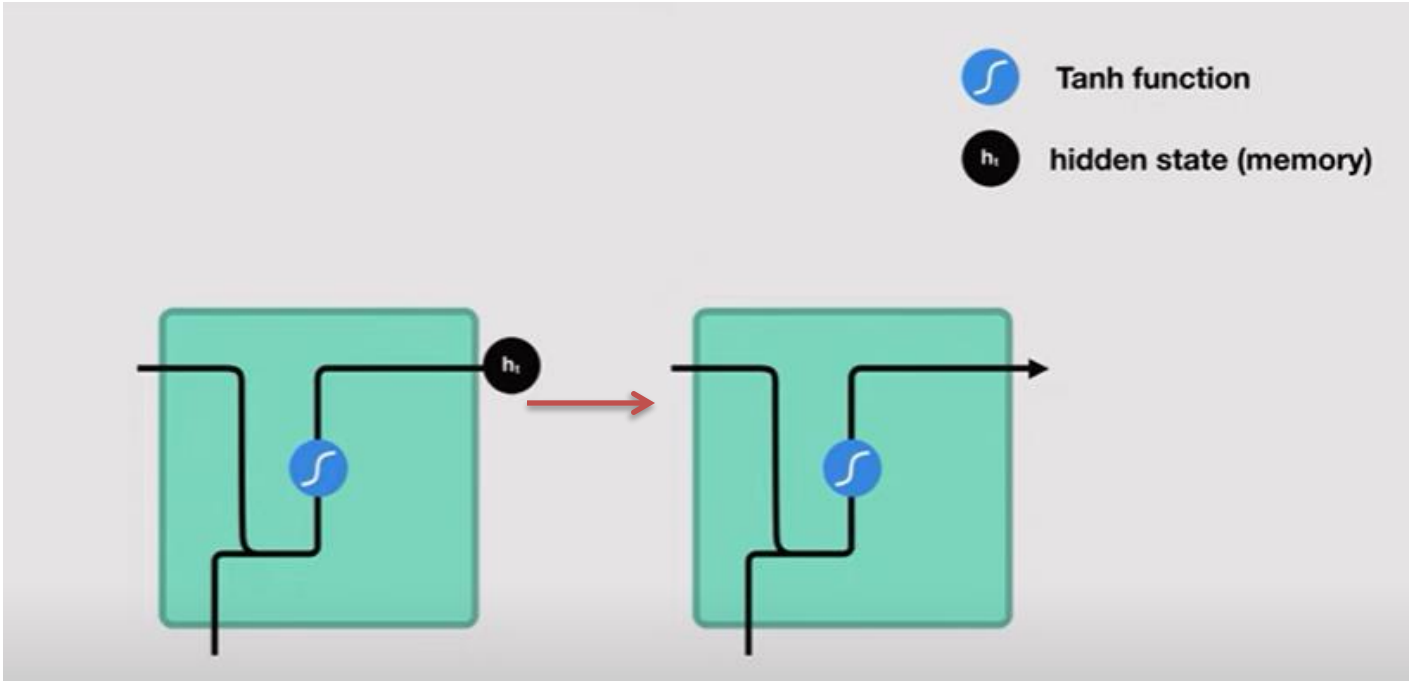
- The embedding layer is trainable, meaning it learns to assign optimal vector representations to words during model training. Here's how:
- **Initial Embedding Matrix**: Start with random or pre-trained vectors.
- **Training with Loss Function**:
  - Example: If you're training for sentiment analysis, the model adjusts the embeddings to better separate positive and negative words.
- **Gradient Updates**:
  - During backpropagation, the embedding vectors are updated along with other model weights.

Simple process.. RNN vs LSTM

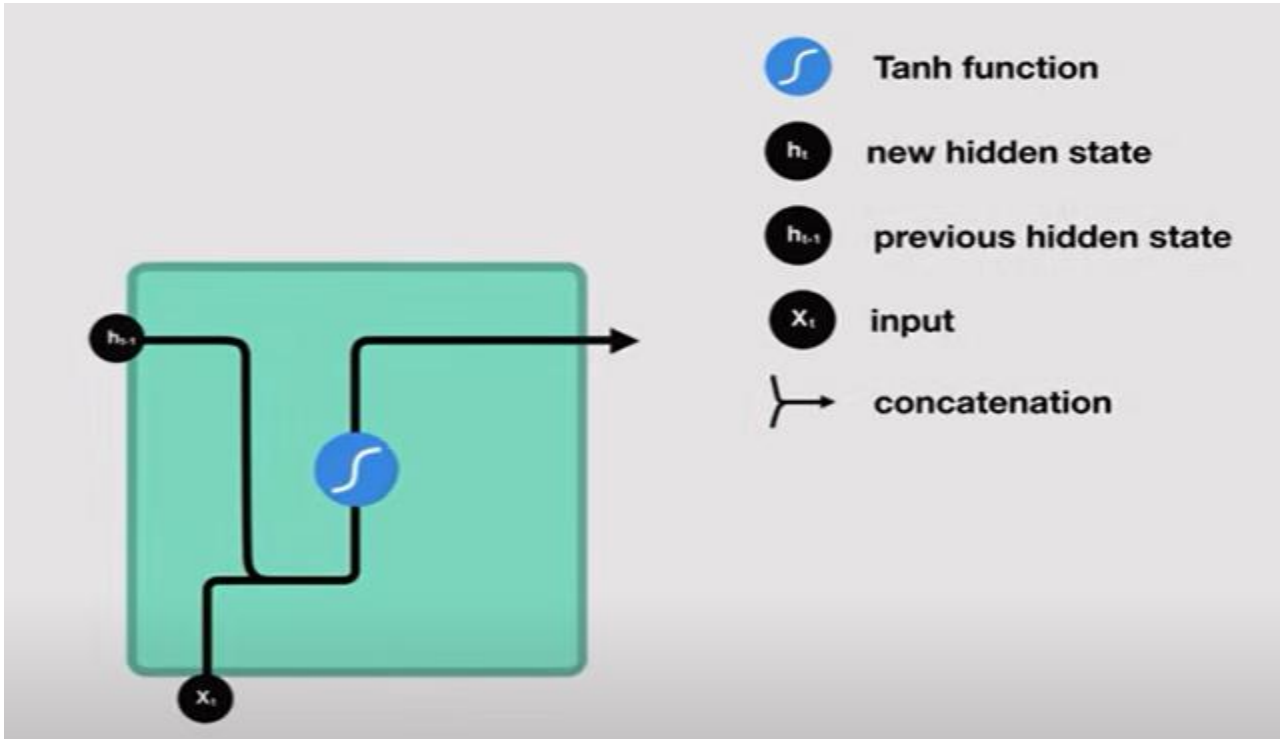
- Before going to actual architecture lets understand simple working of cells of RNN & LSTM individually.
- Lets consider RNN: first words get transformed into machine readable vectors



- These sequence of words are processed one by one where in previous hidden state is passed to next step of the sequence

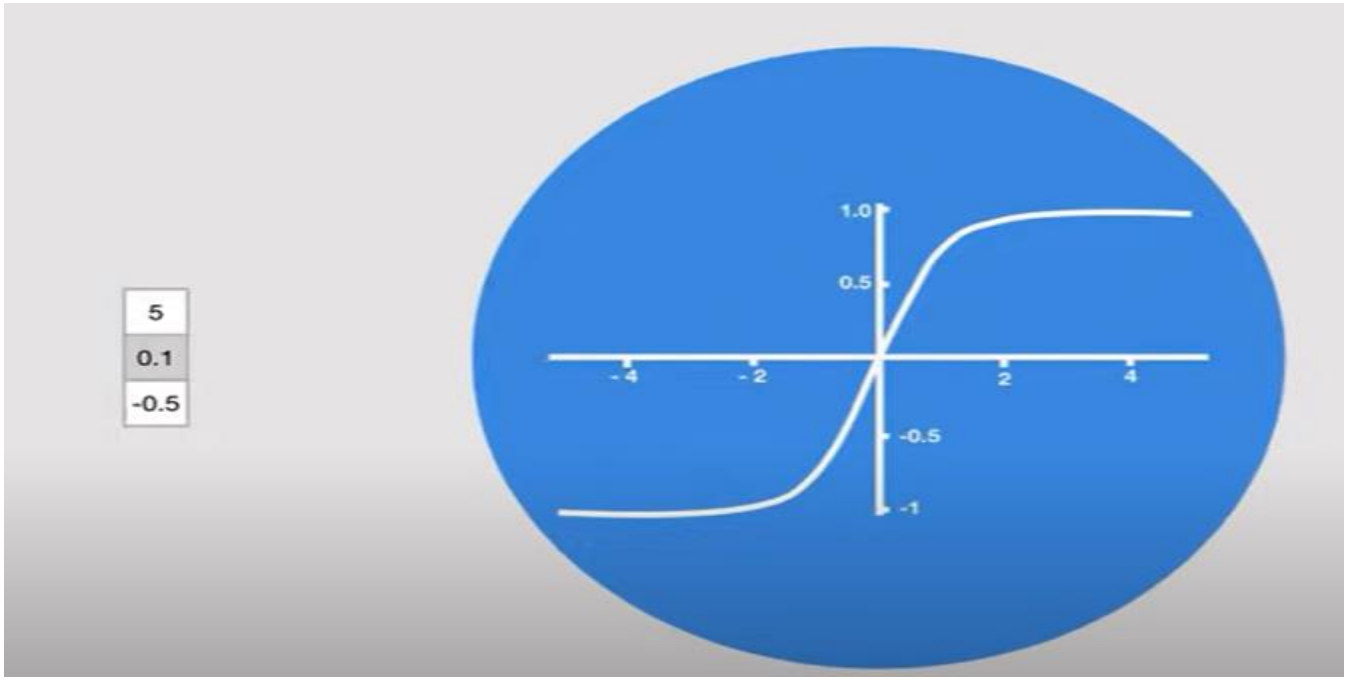


- Lets see how a hidden state is calculated?
- A vector with current input & previous input is created and its passed to a Tanh activation function.
- Output of this will be a new hidden state.

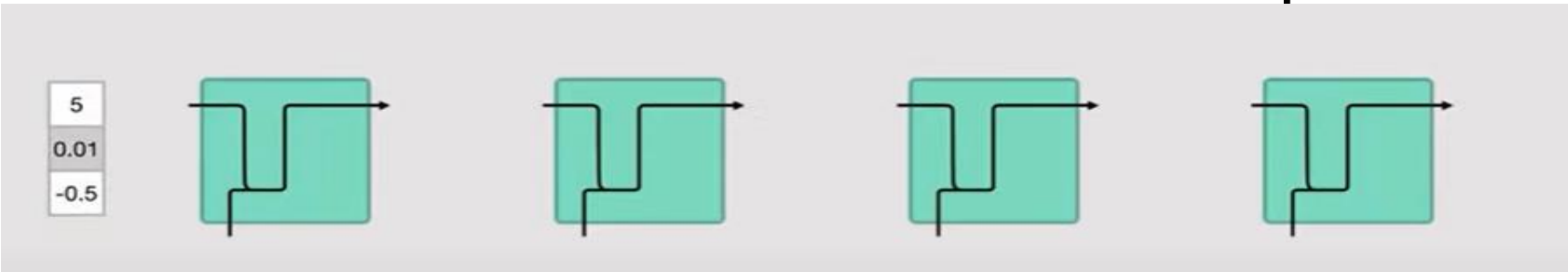


Why Tanh Activation function is important?

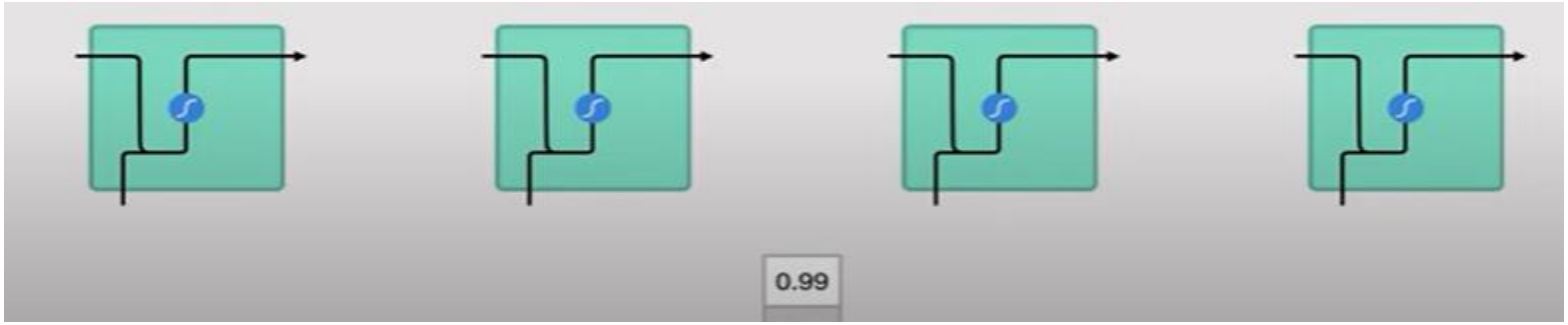
- Tanh activation function is a squishing function used to regulate the value of a neural network



- Information vectors undergo various mathematical transformations in this process.

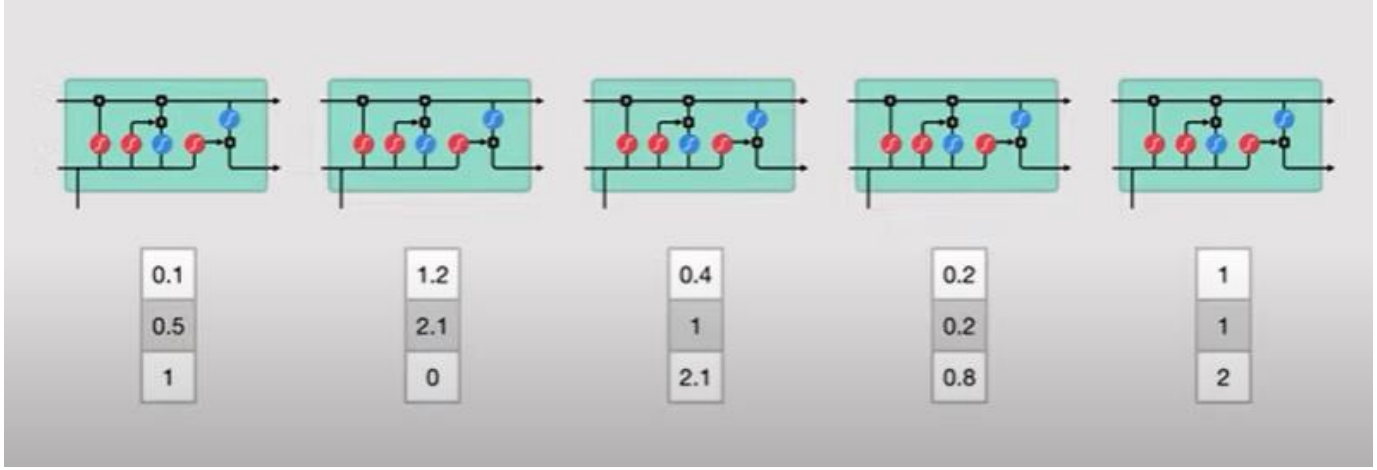


- Some values can explode in this process, causing other values to be insignificant. Tanh function ensures the value remains between -1 and +1



Let's see LSTM now….

- LSTM has similar control flow as that of RNN

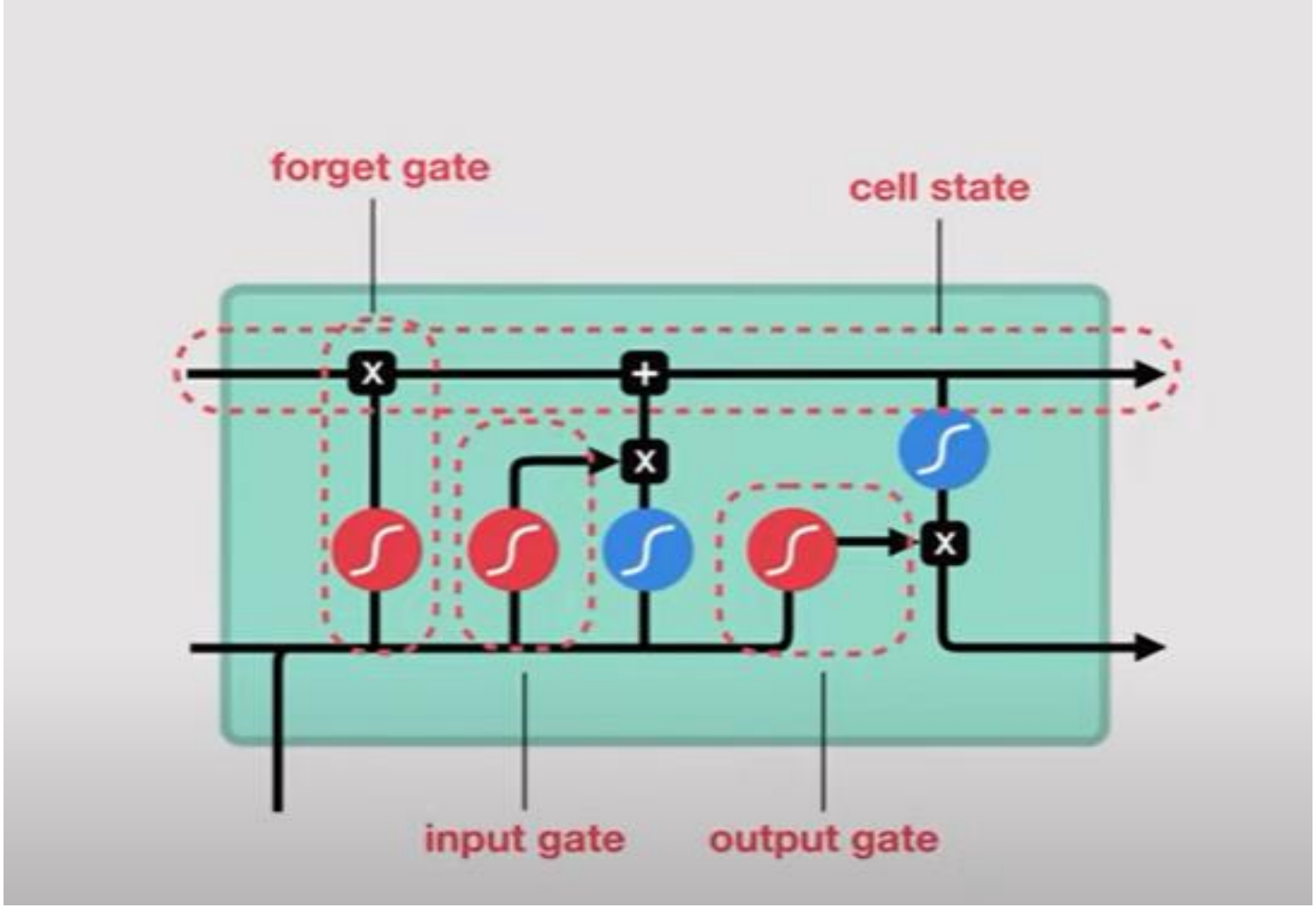


- It processes data sequentially passing on the information as it propagates forward .
- Major difference lies in the operations within the cells of LSTM!!

What makes LSTM special?

- It can achieve long term dependency.
- It knows which information to discard or retain.
- How does LSTM achieve it????
- Answer: using GATE and CELLS
- LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.
- Information is retained by the cells and the memory manipulations are done by the **gates.**

LSTM CELL



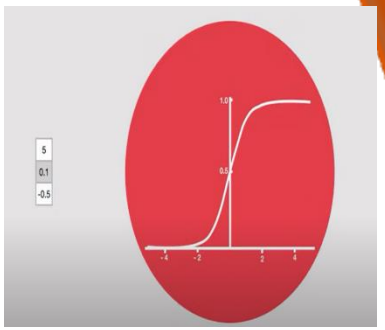
- The **cell state** acts like a sequence highway which transfers the relative information down to the sequence chain. **Its like a memory of the network!!**
- **Cell state can transfer information from previous time steps** as well thus it reduces the effect of short term memory!
- Information can get added or removed via the **Gates** present in the cell.
- Gates are a kind of neural networks which decides which information to retain or discard on the cell state!!
- **Gates are trained to learn which information to retain or discard.**

LSTM CELL

- **Hidden state** in LSTM is broken into two states:
  - CELL STATE: called internal memory where all the information is stored
  - HIDDEN STATE: used for computing the output

NOTE: cell state and hidden state are shared across every time step.

LSTM CELL: GATES



- Gates contain sigmoid activation.
- Sigmoid function squishes values between 0 & 1.
- This can help to update or forget the data.
- Thus a network learns which data to remember and which to forget!!!!
- There are 3 different gates which regulate the information flow in an LSTM cell.

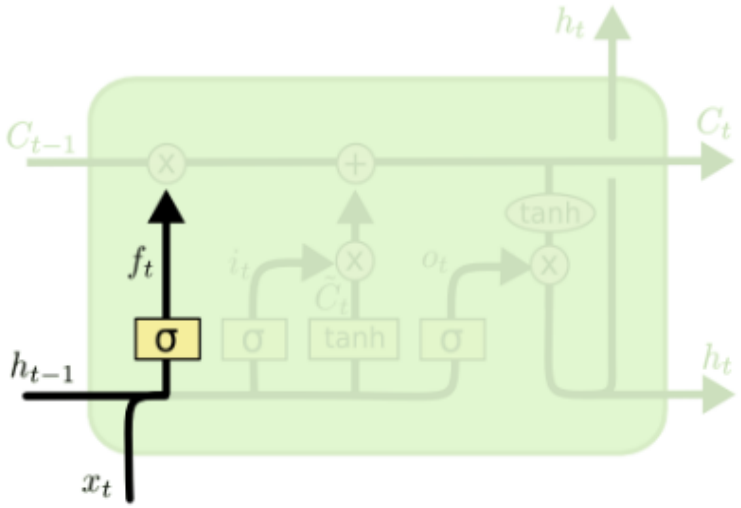
FORGET GATE

- The first step in our LSTM is to decide what information we're going to throw away from the cell state.
- This decision is made by a sigmoid layer called the "forget gate layer."
- It looks at h_{t-1} i.e previous hidden state and x_t i.e current input and outputs a number between 0 and 1 for each number in the cell state Ct−1.
  - 0 means FORGET
  - 1 means RETAIN

Example

- Let's say we're training an LSTM to predict the next word in a conversation. Here's the dialogue so far: "***Hi, my name is Alice and I love to play***"
- In this case, the forget gate might "forget" the greeting **"Hi, my name is"**, and keep **"Alice and I love to play"**

FORGET GATE EQUATION



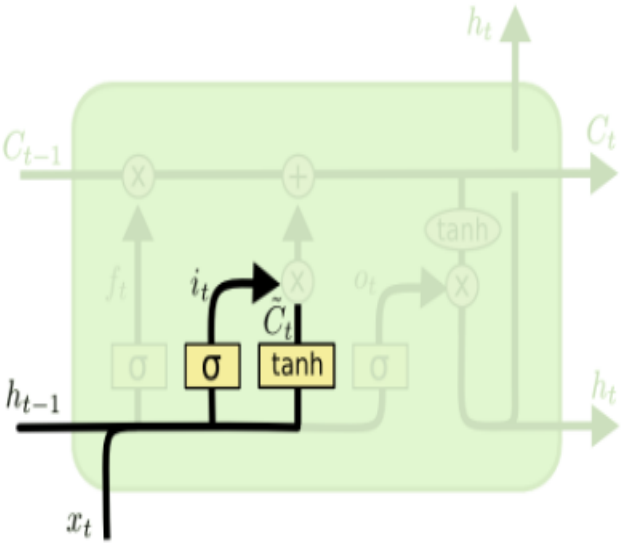
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

INPUT GATE

- The input gate decides how much of the new information should be **added** to the memory.
- In this case, the current input is **"play"**. The input gate evaluates how relevant this is for updating the memory.
- The LSTM already has the phrase **“Alice and I love to play”** stored, and the word **"play"** is useful because it tells the model that Alice is probably talking about a hobby or activity.
- The input gate adjusts the memory by adding more detailed information about what Alice loves to do, perhaps by focusing on a verb or activity that can help predict the next word in the sentence.

INPUT GATE

- The next step is to decide what new information we're going to store in the cell state. This has two parts.
- First, a sigmoid layer called the "input gate layer" decides which values we'll update.
- Next, a tanh layer creates a vector of new candidate values, $C_t^{\sim}(t)$ that could be added to the state.

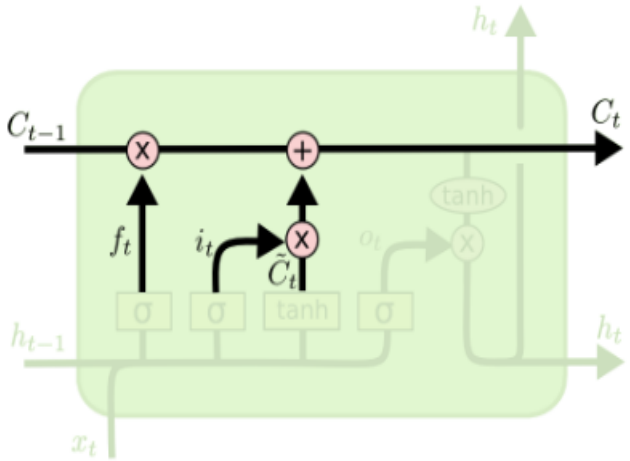


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

INPUT GATE

- In the example taken, we'd want to add the the new subject JACOB to the cell state, to replace the old one we're forgetting i.e MARK.
- It's now time to update the old cell state C_{t-1} into the new state C_t .
- So, we multiply f_t with C_{t-1} (forgetting the things we decided to forget earlier) and also , we add [$C_t^{\sim}$*(t)* * i_t] to obtain new cell state C_t.



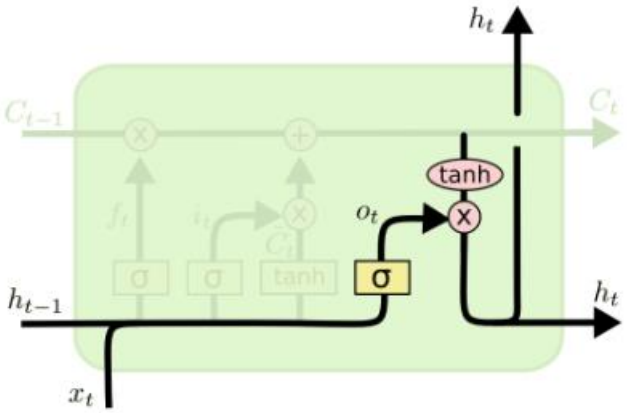
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

OUTPUT GATE

- Finally, we need to decide what we're going to output.
- This output will be based on our cell state, but will be a filtered version.
- First, we run a sigmoid layer which decides what parts of the cell state we're going to output.

OUTPUT GATE

- Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.



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

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

OUTPUT GATE

- The output gate determines what part of the **updated memory** (cell state) should be sent out as the hidden state, which will influence the next word prediction.
- After the forget gate has decided which information to forget and the input gate has updated the memory, the output gate looks at the current state of the memory and decides which parts to output
- . The output gate focuses on the most relevant parts of the memory, which in this case is **“Alice and I love to play”** and uses that to predict the next word.
- For example, the model might predict the next word to be **"guitar"**, **"soccer"**, or **"chess"**, based on the context that Alice loves to play something

TIME FOR THE QUIZ!!!!