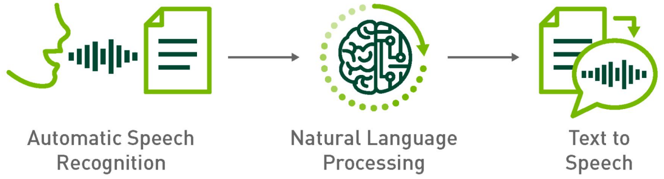
SEQUENTIAL NATURAL LANGUAGE PROCESSING

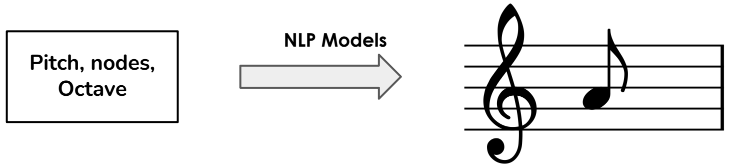
Agenda

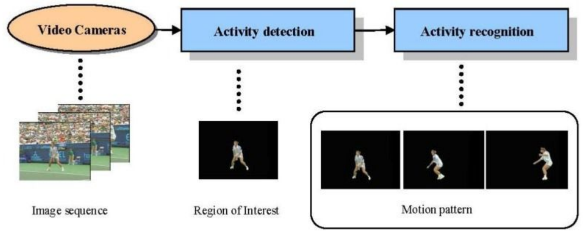
- Understanding Sequential Data
- Introduction to RNN
- Introduction to LSTM
- Hands-on Case study

What is Sequential Data?

- Sequential data is **a series of observations**, measurements, or events that **occur over time.**
- Examples of sequential data include **speech signals, time series data, video frames, and musical signals.**
- The **time component** is a critical feature of sequential data, this time component provides us context and tells us how the events are related to each other in a **sequential fashion.**
- Sequential data is used in a variety of applications, including natural language processing, **speech recognition, music generation, time series prediction, and video analysis.**





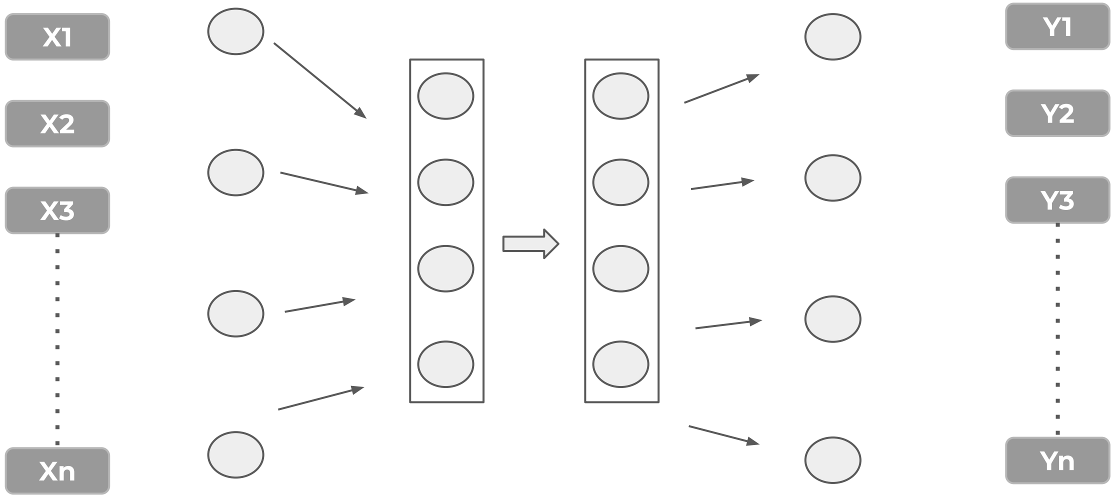


Why Sequential NLP?

Text/Audio data is always in the form of sequences and the order of the words in the data obviously matters

Contextual Understanding	Wide range of Applications	High demand
Sequential NLP enables us to understand the meaning and context of text data that occurs in a sentence	Applications include language translation, sentiment analysis and text generation	With the growing volume of sequential text data, it has become essential to learn and understand sequential NLP
Better predictions are made based on the understanding of this context	Helps in solving real world problems and making informed decisions	Many organizations want to make informed decisions based on sequential text data

Why does a standard NN not work?

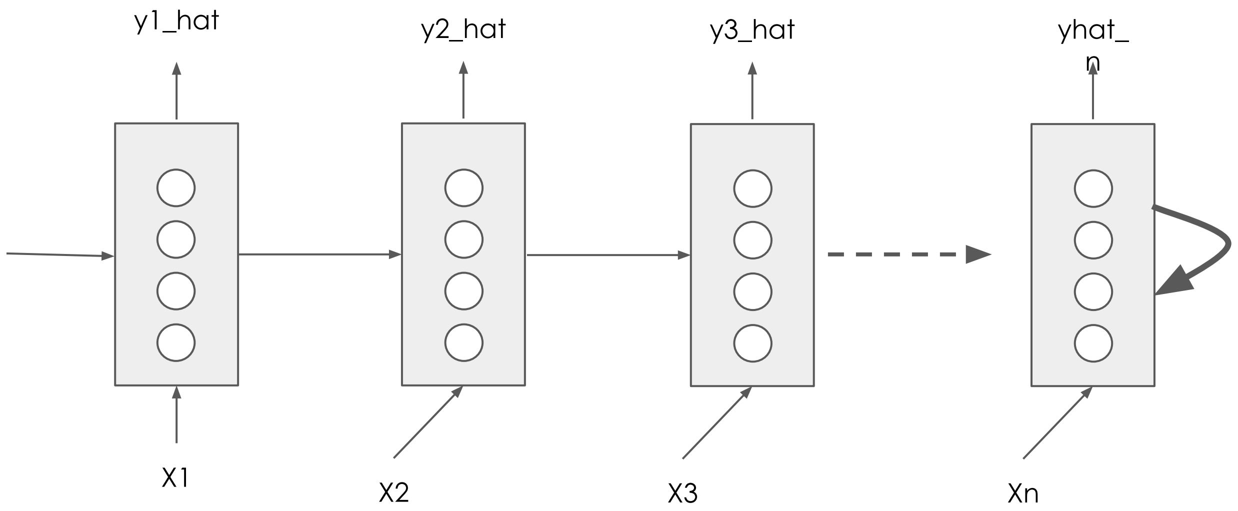


- **Inputs and outputs could be of different lengths in various NLP use cases.**
- **A traditional NN also does not share features learned across different text positions.**

Recurrent Neural Networks

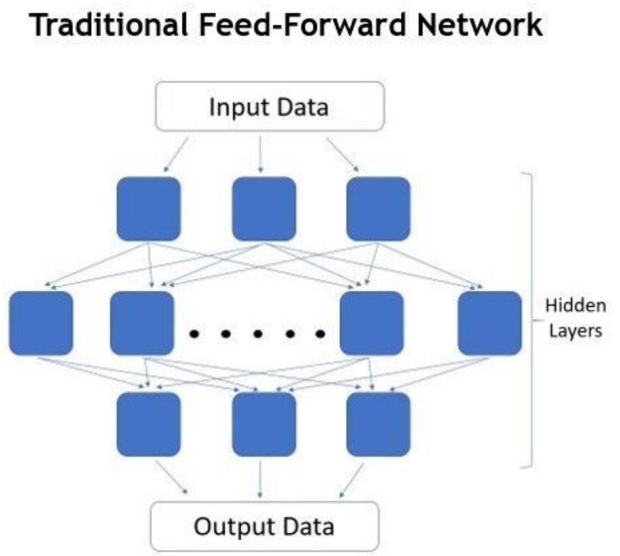
- An RNN is a type of Neural Network designed to handle sequential data by **modeling the dependencies between elements in a sequence.**

- RNNs use loops to allow information to be passed from one step of the sequence to the next, making them well-suited for tasks such as language modeling, speech recognition, and machine translation.

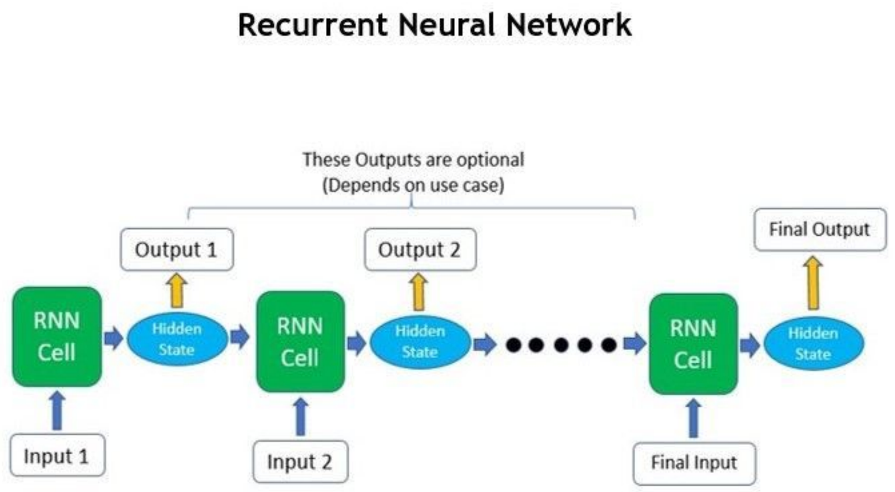


RNNs have a recurrent structure, **where the hidden state of the network at time step t is influenced by the hidden state at time step t-1.** This allows the network to **have a "memory" of past information**.

RNNs vs ANNs for Text



VS



The **main advantage** of using RNNs instead of standard neural networks is that they give scope for features to be shared across various inputs. **Weights are shared across time in an RNN**. RNNs can remember their previous inputs to some level, but simple Artificial Neural Networks are not capable of doing so, and hence cannot provide scope for word dependencies in sentences.

How do RNNs work?

Input Sequence

An input sequence is fed into the network one time step at a time. The input at each time step can be any type of data, such as text, speech, or even time series data.

Hidden State Computation

The hidden state at each time step is computed based on the current input and the previous hidden state. This computation is done using a set of weights and biases that are learned during training. The hidden state acts as a memory of past inputs and is used to make predictions for the current time step.

Output Computation

The output of the RNN at each time step is computed based on the current hidden state. This allows the network to produce a sequence of outputs for a sequence of inputs.

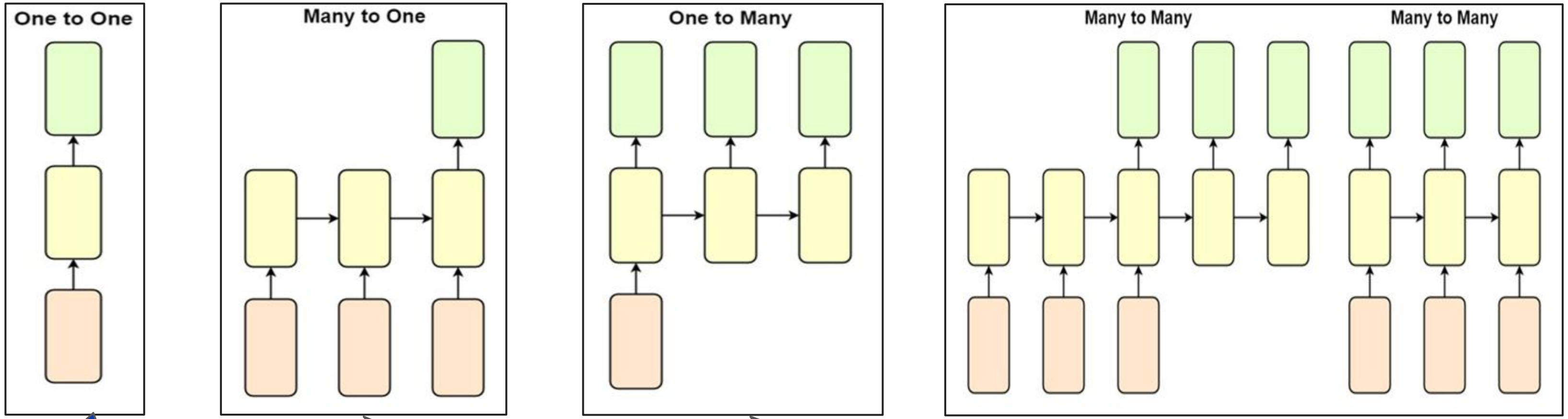
Back Propagation

During training, the error is back propagated through time to update the weights and biases in the network. This allows the network to learn patterns in the input sequences and improve its performance over time.

Recurrent Structure

The hidden state of the network at time step t is influenced by the hidden state at time step t-1. This recurrent structure allows the network to have a **"memory"** of past information.

RNN Types



A single input and a single output, similar to a traditional NN

The input sequence is mapped to a single output
Task - Sentiment Analysis

The input is mapped to an output sequence of different length
Task - Image Captioning

The input sequence is mapped to an output sequence of the same length
Task - Stock Price Prediction

Each rectangle is a vector and arrows represent functions (e.g. Matrix Multiply)
Input vectors are in Orange, output vectors are in green and lime vectors hold the RNN's state

Limitations of RNNs

Long Term Dependencies

It refers to the problem of preserving information from previous time steps over a long sequence, causing vanishing gradients and poor performance in training and prediction.

Overfitting

RNNs can easily overfit to training data, especially when the sequence is long and the number of parameters is large.

Inadequate memory

RNNs have limited memory capacity, making it challenging to model long-term dependencies in sequences.

Truncated Backpropagation through Time (BPTT)

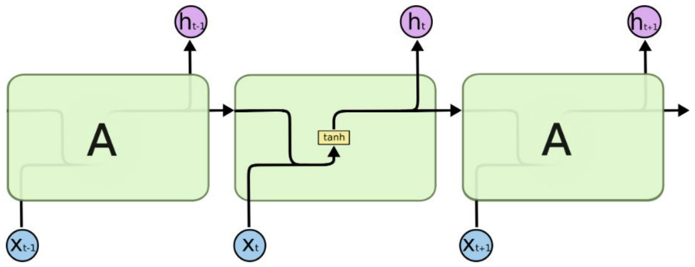
BPTT is limited, leading to a tradeoff between computation and accuracy

Vanishing and Exploding Gradients

Long Short-Term Memory (LSTM)

LSTMs are a special kind of Recurrent Neural Network that are explicitly designed **to avoid the long-term dependency problem.** They were created to help solve the RNN limitation of remembering information for longer periods of time.

All Recurrent Neural Networks have the form of a **chain of repeating modules of a Neural Network**. In standard RNNs, this repeating module will have a very simple structure, such as a single Tanh layer.

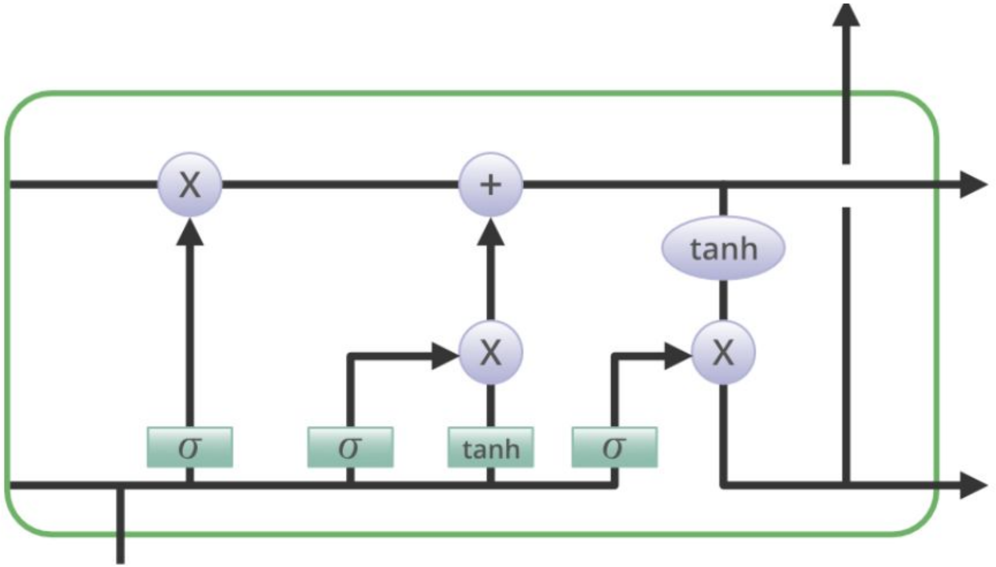


The repeating module in a standard RNN contains a single layer

The LSTM Cell

An LSTM also has a similar control flow as a Recurrent Neural Network.
It processes data passing on information as it propagates forward.
The difference is in the operations taking place within the LSTM's cell.

🟩	**Neural Network Layer**
●	**Pointwise Operation**
→	**Vector Transfer**
≻	**Concatenate**



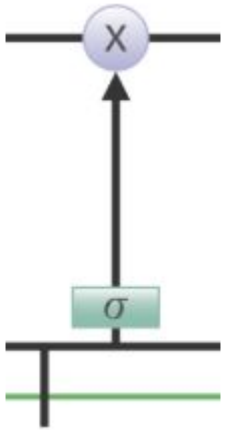
Decoding the LSTM

The memory cell is controlled by three gates:

- **The Forget Gate:** The Forget Gate controls what information is *removed* from the memory cell.
- **The Input Gate:** The Input Gate controls what information is *added* to the memory cell.
- **The Output Gate:** and the Output Gate controls what information is *output* from the memory cell.

Gates are a way to optionally let information through. They are composed of a Sigmoid Neural Net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. **A value of zero means "let nothing through," while a value of one means "let everything through!"**



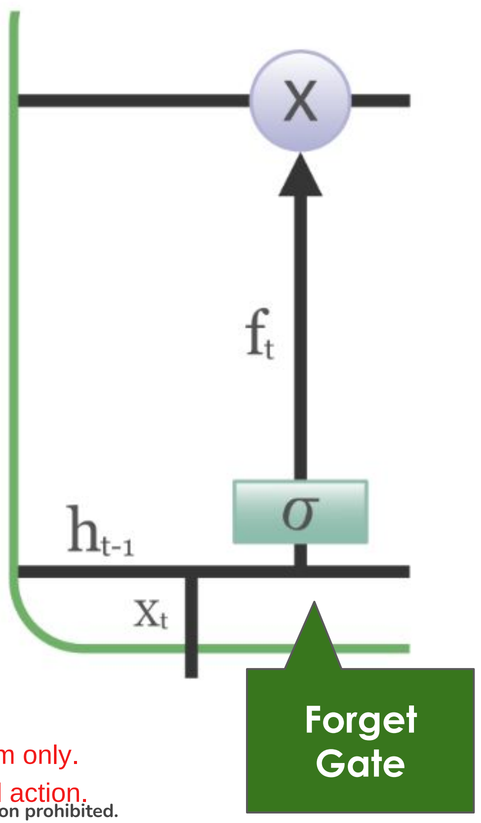
Step 1: The Forget Gate Layer

1) The Forget Gate is controlled by an activation function, typically a Sigmoid Function, **that outputs values between 0 and 1.**

2) **Two inputs x_t (input at particular time) and h_{t-1} (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of a bias. This is of course just a linear combination.**

3) The result is passed through a **Sigmoid Function** which gives a binary output.

4) **A 1 represents the "completely keep this" choice, while a 0 represents "completely get rid of this information".**



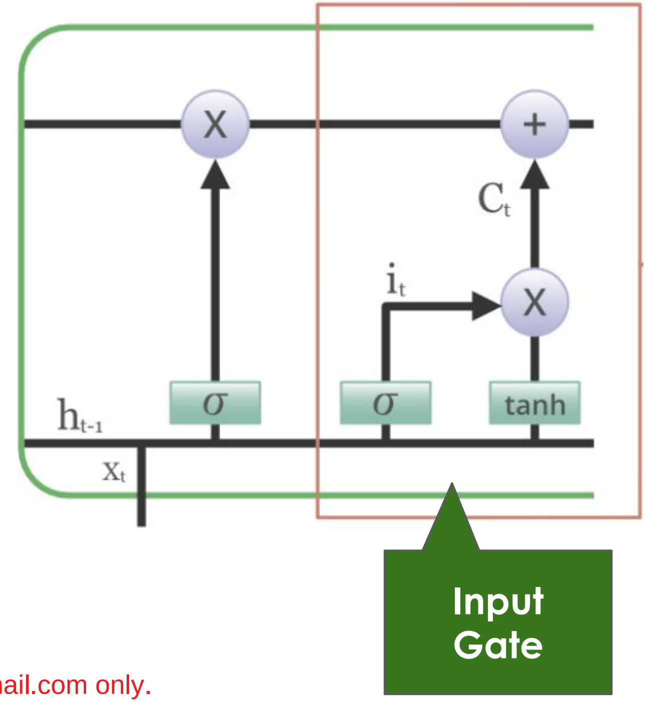
Step 2: The Input Gate Layer

The Input Gate determines which information from the current input should be added to the hidden state and which information should be ignored. This allows the network to selectively choose what information to remember from the past and what information to ignore.

The information is regulated using the Sigmoid Function

A vector is created using the *tanh* function that gives output from -1 to +1

The values of the vector and the regulated values are multiplied to obtain the useful information



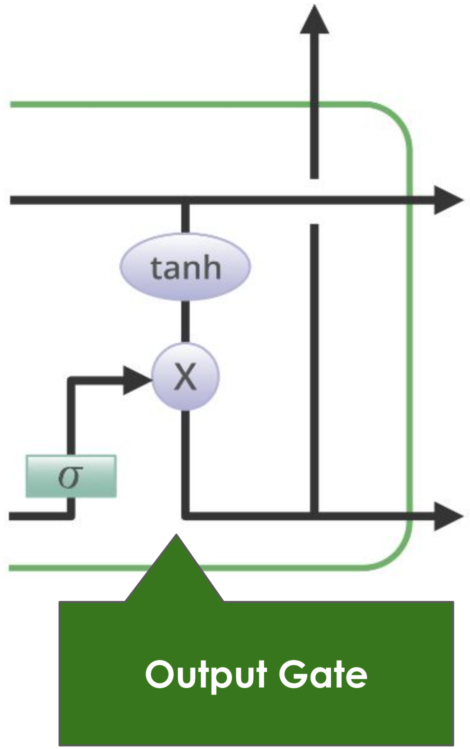
Step 3: The Output Gate Layer

The task of extracting useful information from the current cell state to be presented as output is done by the output gate.The output gate determines which information from the hidden state should be passed on to the output

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.



Hands-on Case Study