

Recurrent Neural Networks

Intro to Deep Learning

Sequential data

So far, our focus has been on handling unordered data, where we simply fed a single matrix or image into the model.

However, when dealing with sequences like time-series data, disregarding the order of the data would lead to improper training.

Inputting such sequential data as-is without considering its order can adversely affect the learning process.

Sequential Data: Daily stock price, weather, document (text), sound, video, and so on.

Recurrent Neural Network

Working with time series data can be surprisingly straightforward.

The key is to input the data in chronological order.

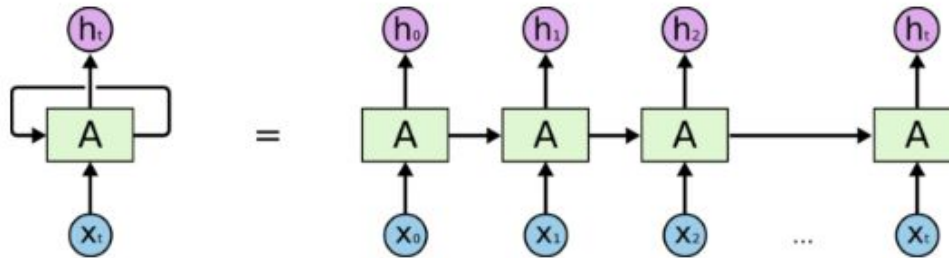
To build an effective model, you must incorporate both historical and current information.

This is achieved by feeding the model's previous output back into the input, a technique known as recursion.

In the realm of deep learning, this is referred to as a **Recurrent Neural Network** (RNN).

Why recurrent?

RNN is a method of storing previously input data for a while when multiple data are input in sequence. And it judges how important the memorized data is and gives it a separate weight and moves on to the next data. It does this for every input value in sequence, so it looks like it's hovering around the same layer before moving on to the next layer.



An unrolled recurrent neural network.

RNN - memory of previous node

In a recurrent neural network (RNN), the output of the previous node, known as the **memory** or hidden state, is utilized as input for the current node.

This characteristic holds significant importance as RNNs are specifically designed to handle sequential data, where each element's information in the sequence relies on the preceding elements.

By incorporating the previous memory as an input for the current node, the RNN **can effectively consider the contextual information from earlier steps**, enabling it to make informed predictions and decisions based on the historical context.

Memory of history

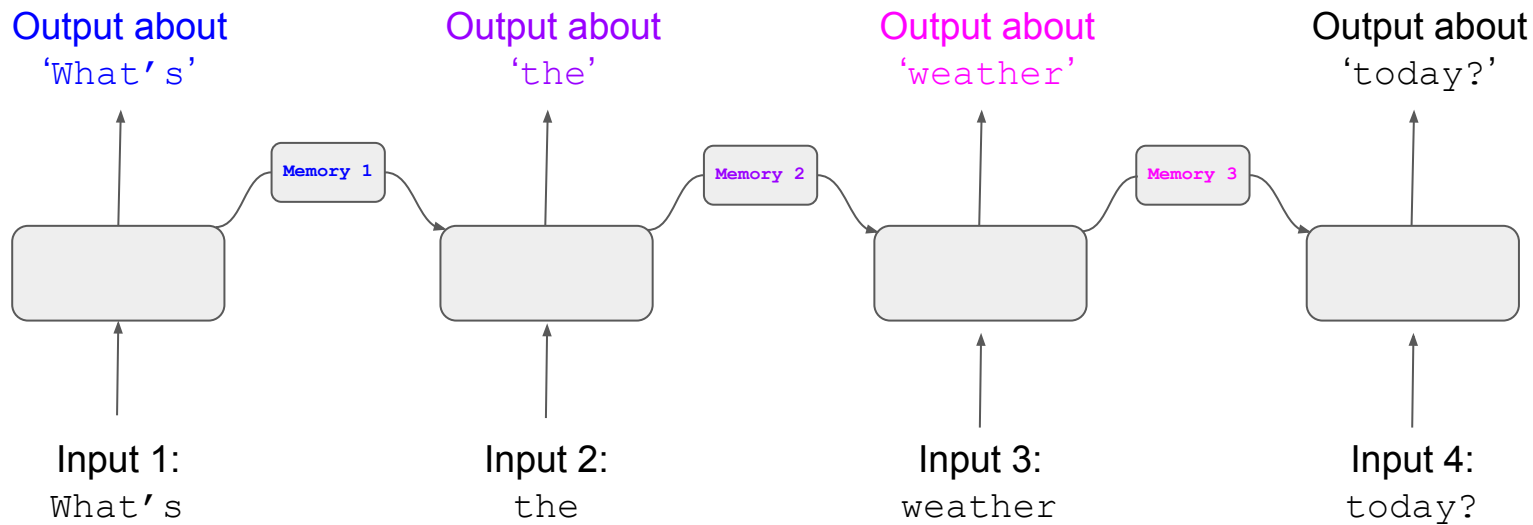
The incorporation of the previous memory serves a vital purpose, enabling the model to comprehend temporal dependencies effectively.

This feature proves highly beneficial across various applications like language modeling, speech recognition, and time series prediction.

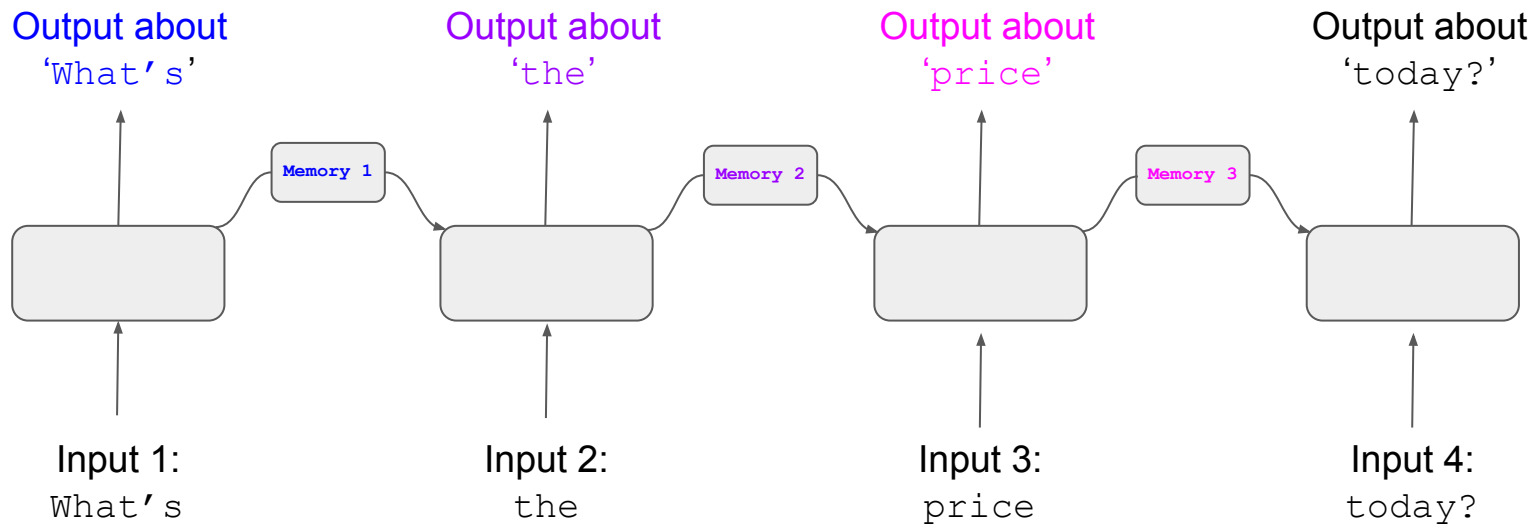
By preserving a memory of past inputs, the RNN becomes adept at capturing long-term dependencies and employing that knowledge to anticipate future outputs.

Essentially, the previous memory plays a pivotal role in facilitating the RNN's ability to learn from historical information, ultimately enhancing its predictive capabilities and decision-making prowess.

RNN



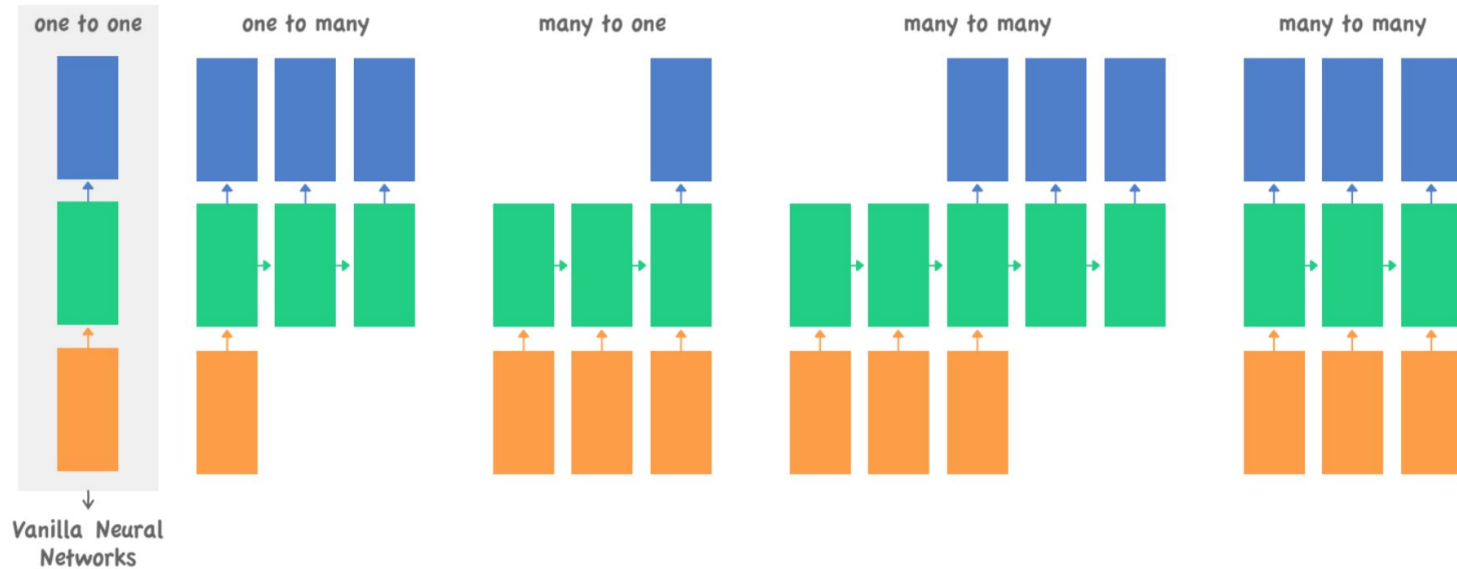
RNN



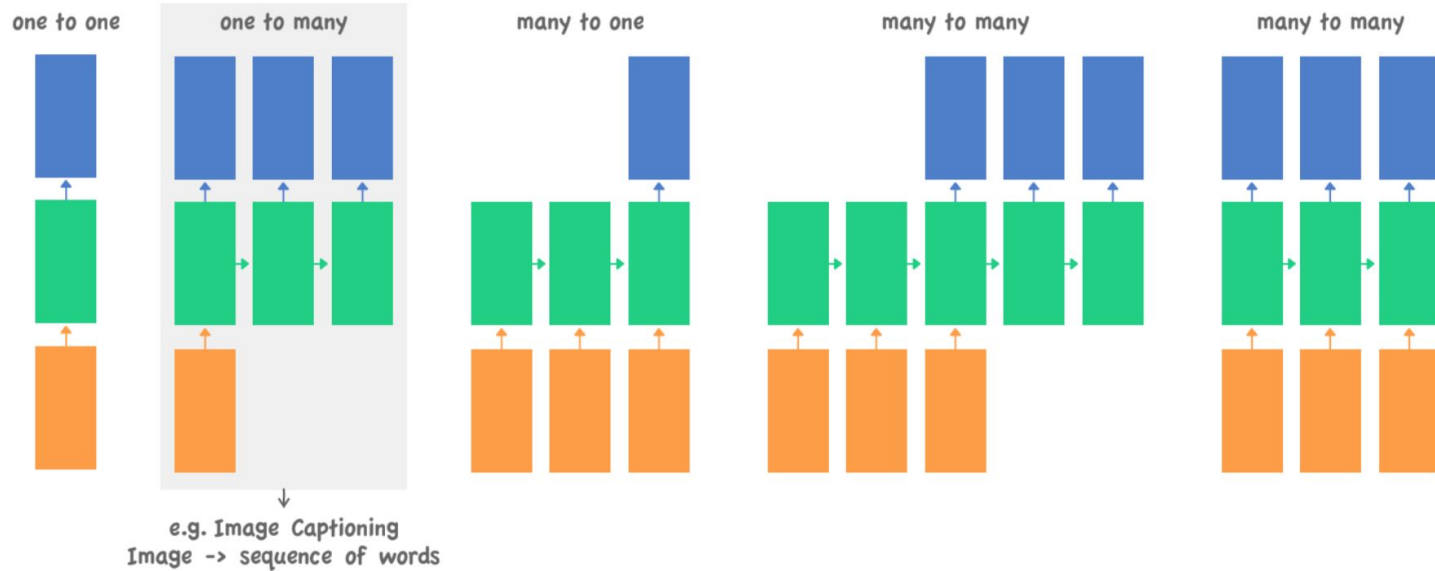
Applications

- Language Modeling
- Speech Recognition
- Machine Translation
- Conversation Modeling / Question Answering
- Image / Video Captioning
- Image / Music / Dance Generation

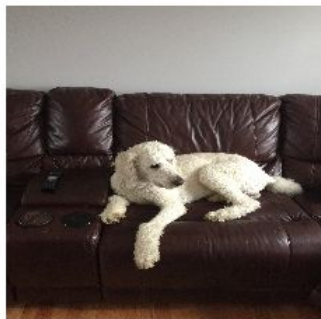
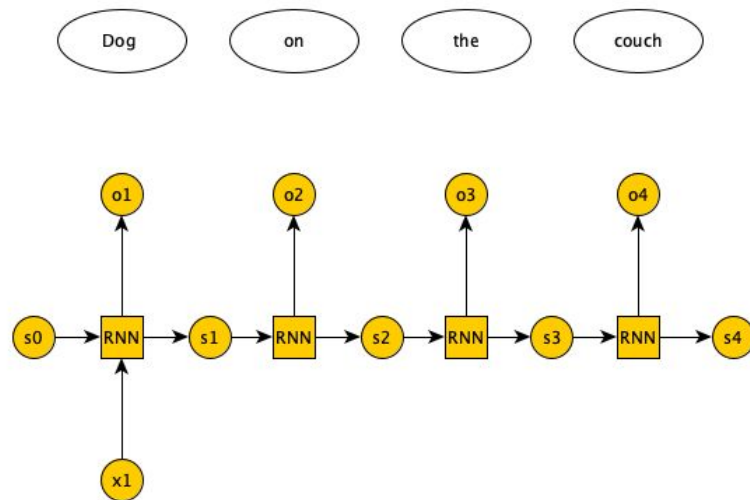
Recurrent Networks Offer a Lot of Flexibility



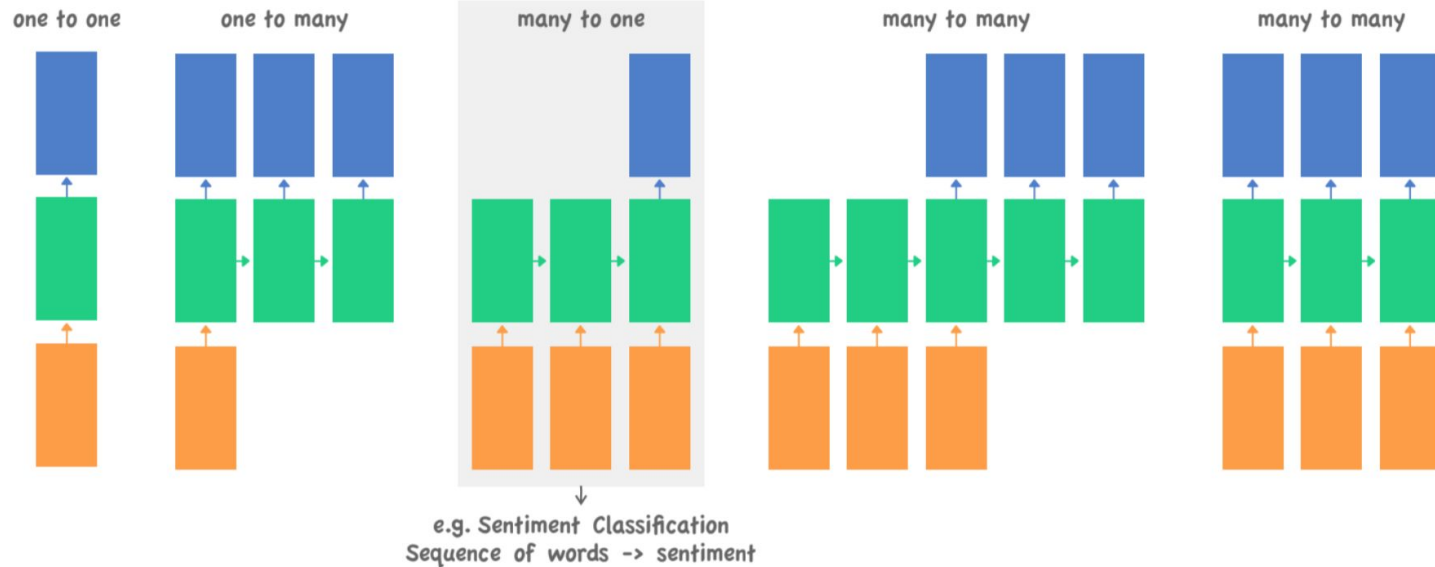
Recurrent Networks Offer a Lot of Flexibility



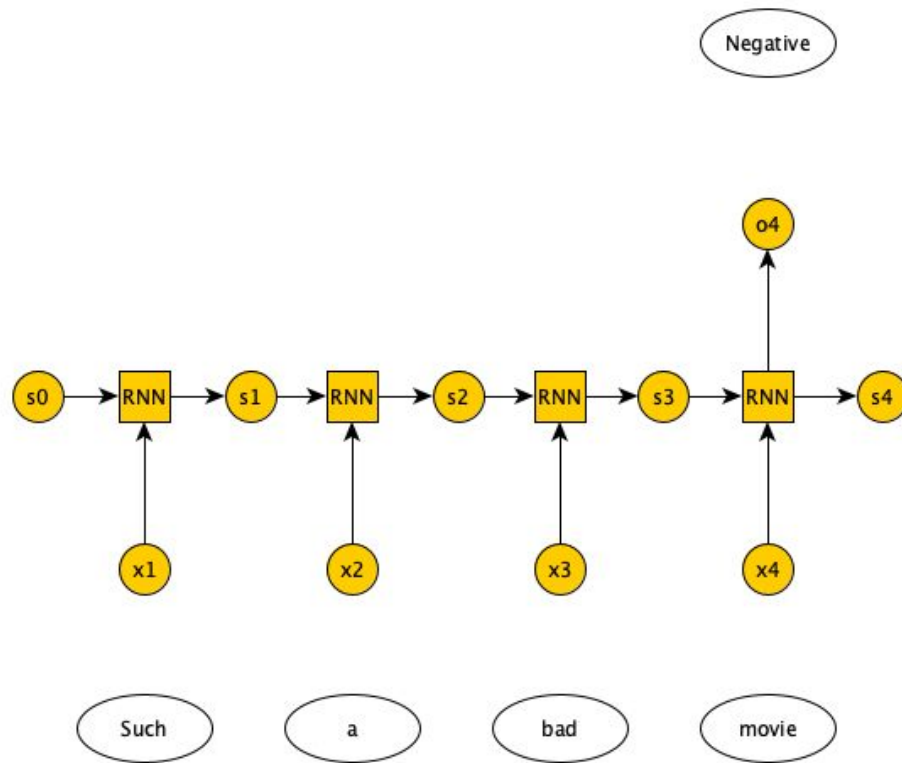
Example



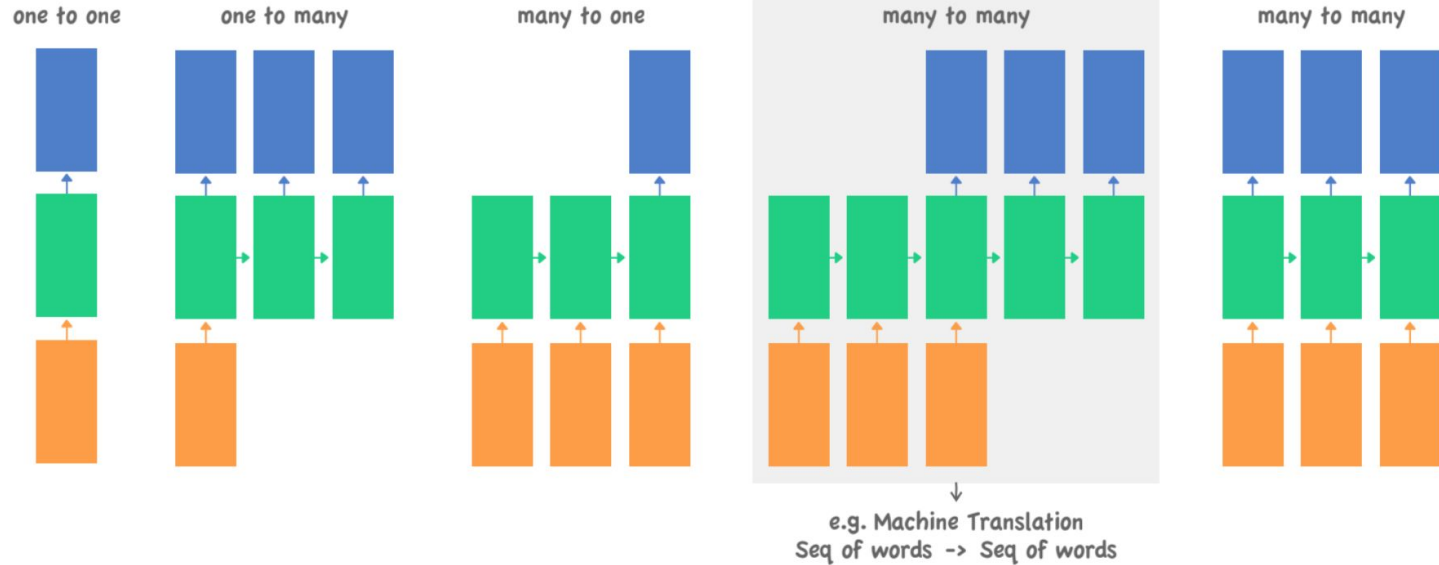
Recurrent Networks Offer a Lot of Flexibility



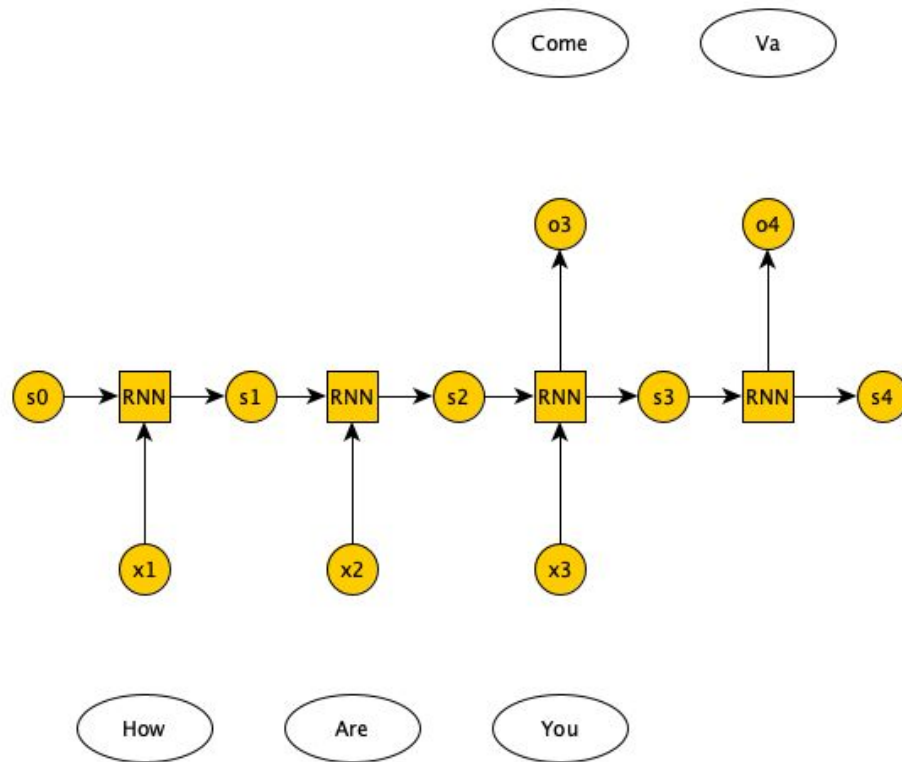
Example



Recurrent Networks Offer a Lot of Flexibility



Example



Recurrent Networks Offer a Lot of Flexibility

