NLP

Introduction to RNNs

Natural Language Processing

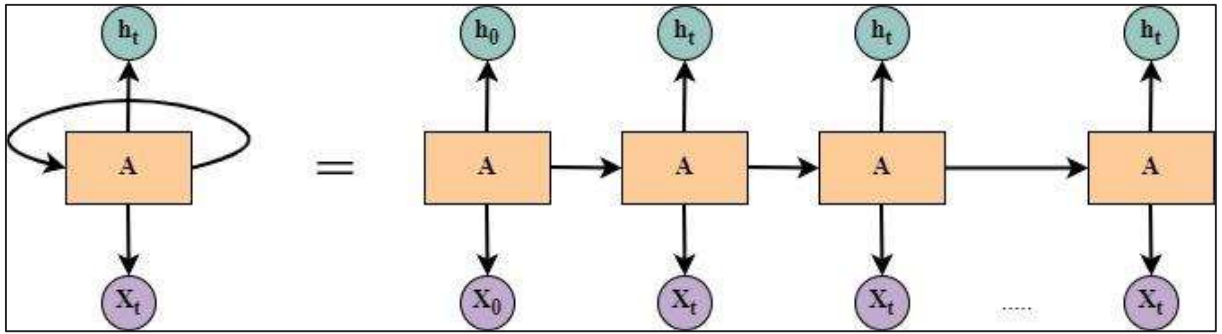
Agenda

- What are RNNs?
- Advantages of RNNs over ANNs
- Types of analysis using Recurrence
- Limitations of RNNs

What are RNNs?

- Recurrent Neural Networks (RNNs) are a Neural Network architecture that can be applied to sequential data such as text, audio, video (a sequence of image frames), and time series data.

- RNNs are named **Recurrent Neural Networks** because they perform **the same recurring task for each element in a sequence, with the outcomes based on prior computations**. RNNs **can be viewed as having a "memory"** that stores information about these prior computations.



An unrolled Recurrent Neural Network (RNN)

Advantages of RNNs over ANNs

When it comes to applying Deep Learning to sequential data, RNNs show some advantages over traditional feed-forward Artificial Neural Networks (ANNs):

1. **RNNs are well-suited for processing sequential data**, as they store information about the sequence of the data. This often results in better predictions on such data than ANNs, as ANNs do not have the capacity to store hidden states or sequential information.

2. **RNNs are not bound to the size of the input** the way ANNs are. This would make RNNs more generalizable to use cases such as text where the size of the input sequence is not necessarily fixed.

3. **RNNs also use shared parameters/weights, unlike ANNs**, which require training unique parameters for every connection. This allows RNNs to reduce the number of parameters that the model needs to learn while training, resulting in computational efficiency.

Types of Analysis using Recurrence

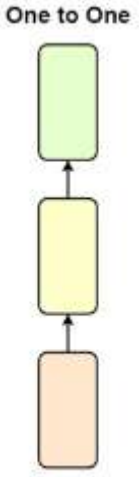
- The flexibility of RNNs also lies in the fact that **we can modify the RNN architecture** to solve different classes of problems relating to sequential data.
- There are **4 types of RNN architectures** we can use depending on the nature of the problem:

1. **One to One**
2. **One to Many**
3. **Many to One**
4. **Many to Many**

Types of Analysis using Recurrence - One to One RNNs

- In the '**One to One**' sequence problem, there is just **a single input and a single output**. Since there's only one input and one output, we use this architecture to solve **binary classification problems**.

- This would **work exactly like the traditional Artificial Neural Network.**

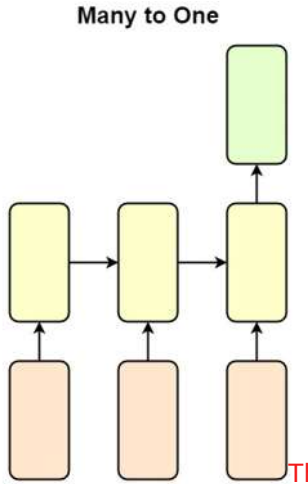


Types of Analysis using Recurrence - Many to One RNNs

- In **'Many to One'** sequence problems, there are **many input elements in the sequence but just one final output.** This type of model takes inputs at every time step and gives one final output.

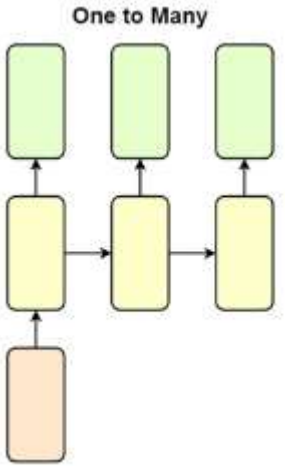
- With this type of modeling, we can perform tasks such as:

    - **Text Classification** (predicting one class for a sentence with many words).
    - **Stock Price Forecasting** (predicting one future stock price for a history of stock prices).



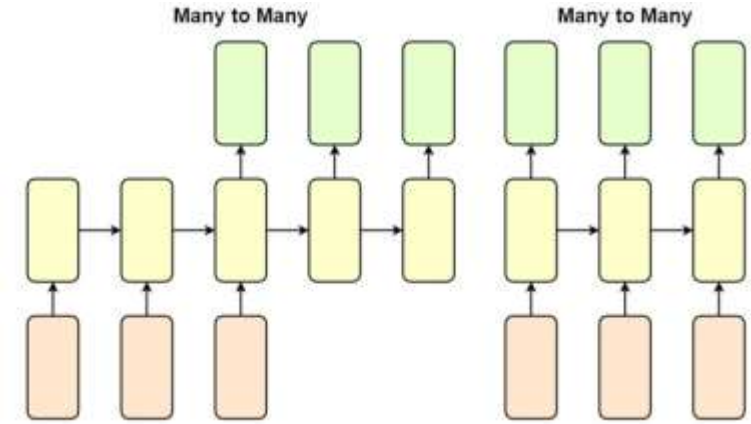
Types of Analysis using Recurrence - One to Many RNNs

- In **'One to Many'** sequence problems, there's **one input and many sequential outputs**.
- With this type of modeling, we can perform generative tasks such as:
    - **Text Generation** (provide one word to generate many subsequent words)
    - **Music Generation** (provide one music note to generate many notes in sequence)
- This type of model takes one input and gives outputs at multiple timesteps.



Types of Analysis using Recurrence - Many to Many RNNs

- In '**Many to Many**', as the name suggests, there are multiple inputs and multiple outputs.
- With this type of modeling, we can perform tasks such as:
  - **Named Entity Recognition (NER)**
  - **Machine Translation**



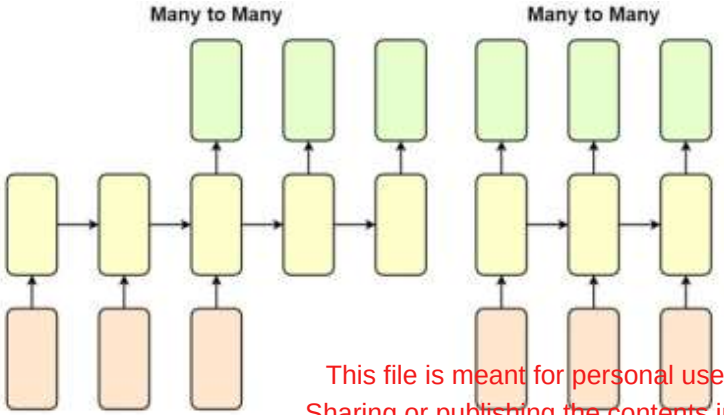
Types of Analysis using Recurrence - Many to Many RNNs

- However, even in Many to Many, there are two cases depending on the application:

  a. **One output is provided for every given input** (Named Entity Recognition).

  b. **The number of inputs is not necessarily equal to the number of outputs** (Machine Translation).

- When the number of inputs and the number of outputs are not equal, we need to use an **Encoder** (a set of LSTM layers) that encodes the input data sequence and gives that encoding as input to a **Decoder** (another set of LSTM layers) that will decode the data.



Limitations of RNNs

Although RNNs are a useful first step for modeling sequential data, they have some limitations:

- Training RNNs can be a **slow and complex** procedure.
- **RNNs don't perform well on long-term sequential data** if Tanh or ReLU are used as the activation function in the network. This is primarily due to the problem of vanishing and exploding gradients, as the products that give gradients include the weights of every later layer. This product can be very big (exploding) or very small (vanishing); in either scenario, the learning quality of the network is affected.

To address these issues, we shall look at an attempt to improve RNNs through the idea of LSTMs (Long Short-Term Memory Networks).

Summary

- Recurrent Neural Networks (RNNs) are **the first Deep Learning architecture created specifically for sequential data modalities.**

- The main types of analysis possible on sequential data using recurrence are:

    1. **One to One**
    2. **Many to One**
    3. **One to Many**
    4. **Many to Many**

- The problem of **vanishing/exploding gradients is quite common in RNNs**, and basically occurs when a Recurrent Neural Network (RNN) is unable to backpropagate meaningful gradient information from the model's output to the layers close to its input end.

  LSTM networks have helped address this situation.

Great Learning

Happy Learning !