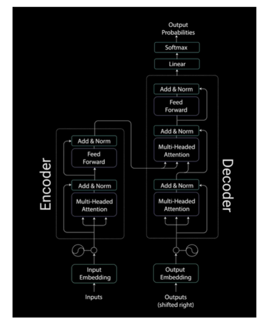
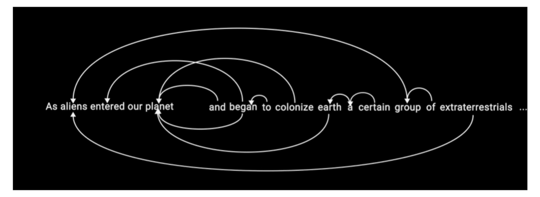
# Day 17: Transformer Architectrue

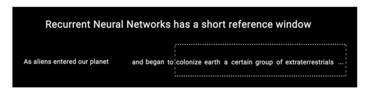
- **Intro**
  - Revolution in the field of machine learning
  - Paper published in 2017 - "Attention is All You Need"
  - It helps overcome the challenges seen in models like RNNs and LSTM
  - **Need**
    - RNN suffers from the vanishing gradient problem which causes long-term memory loss
    - Processing of text sequentially
    - Long sentence like- 'XYZ went to France in 2019 when there were no cases of COVID and there he met the president of that country.'
      - If we ask - what country?
      - It wont be able to tell
      - Encountered the word 'France' long before
    - Training in RNNs happens word by word, not sentence by sentence
    - With LSTMs and GRUs, vanishing gradient was solved, but not the sequntial processing problem
    - Ex: There is a word -' Point', and we use it in two different contexts given below
      - The needle has a sharp point.
      - It is not polite to point at people.
- **Transformer Architecture**



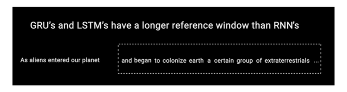

  - Attention Mechanism

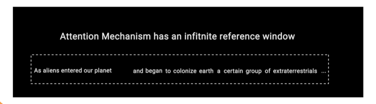


    - As the model generates text word by word...
      - It has the ability to reference to the words relevant to the generated word
        - How does it know which word, during backpropogation
    - Even RNNs can do it to an extent

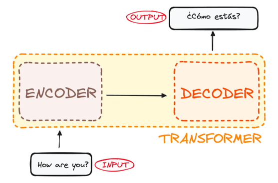



    - GRU and LSTM

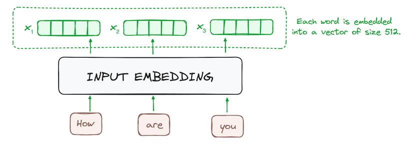


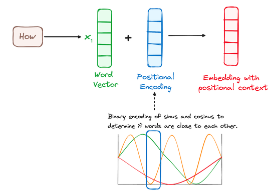

  - Two main parts:



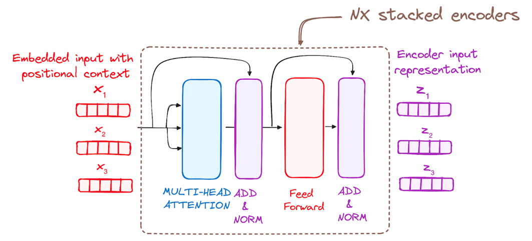

    - Encoder maps an input sequence into a continuous representation that holds all the learned information of that input
    - Decoder takes that representation, step-by-step generates a single output while also being fed the previous output
  - Working:
    - 1. Input Embedding Layer:
      - Converts each word/token into a dense vector of fixed size using learned embeddings.



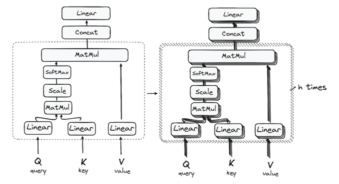

      - Positional encodings are added to the embeddings to inject sequence order information



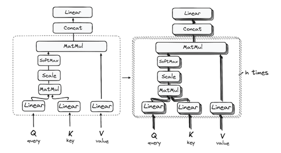
    - 2. Encoder:



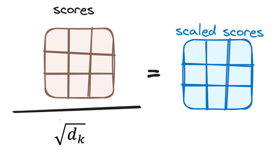

      - Each encoder layer consists of two main sub-layers
        - Multi-Head Self-Attention



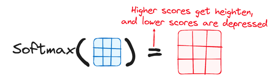
          - Calculate attention scores of all tokens in the input, allowing each token to focus on other relevant tokens in the sequence.
            - Matrix Multiplication (MatMul) - Dot Product of Query and Key



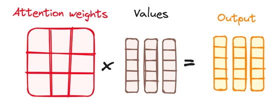
            - Reducing the Magnitude of attention scores




            - Applying Softmax to the Adjusted Scores




            - Combining Softmax Results with the Value Vector




          - Multiple attention heads capture different relationships.
        - Feed-Forward Neural Network
          - A fully connected network applied to each position independently
          - Includes ReLU activation and dropout
    - 3. Decoder
      - Similar structure to the encoder, with an additional attention layer
        - Masked Multi-Head Self-Attention
          - Prevents the decoder from attending to future tokens during training
        - Encoder-Decoder Attention
          - Allows the decoder to focus on relevant parts of the encoder's output.
        - Feed-Forward Neural Network
          - Processes each token representation
    - 4. Output Linear Layer
      - Projects the final decoder outputs to the vocabulary size, transforming them into logits.
    - 5. Softmax Layer
      - Converts logits into probability distributions, generating the most likely next token in the sequence.
  - efficient handling of complex dependencies in text.
- **Hands-on**

Presented with xmind