1.Encoder-Decoder

ENCODER

It reads input sequence $x_1, x_2,...,x_T$.

$$h_i = EncoderRNN(h_{i-1}, x_i)$$

Thus it produces a sequence of hidden states:

$$h_1, h_2,...,h_T.$$

DECODER

Decoder uses only the final encoder state.

$$s_0 = h_T$$

Then

$$s_t = DecoderRNN(s_{t-1}, y_{t-1})$$

Drawbacks

- Entire input sequence is compressed into a single vector h_T
- Long sequences cause information loss due to fixed-length bottleneck

2.Attention

Let's take an example.

I ate an apple because it was tasty

When predicting “it”, the models should focus on “Apple” and not everything.
This is achieved by attention mechanism.

Instead of relying on a single vector, attention allows the decoder to dynamically focus on different parts of the input sequence at each time step. This improves handling of long-range dependencies and alignment between input and output tokens.

At each decoder time step t:

Step 1: Computer alignment scores

For each encoder state h_i:

$$e_{t,i} = score(s_{t-1}, h_i)$$

$e_{t,i}$ tells us how relevant encoder position i is for predicting output at time t

Step 2: Normalize attention weights

$$\alpha = \frac{exp(e_{t,i})}{\sum_{k=1}^{T} exp(e_{t,i})}$$

Step 3: Compute context vector

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i$$

c_t is a **summary of the input**, for time step t.

DECODER WITH ATTENTION

Now the decoder state update becomes:

$$s_t = RNN(s_{t-1}, y_{t-1}, c_{t-1})$$

And output:

$$P(y_t | y_{<t}, x) = softmax(V[s_t; c_t] + b$$

DRAWBACK OF ATTENTION

h_i for all i cannot be computed in parallel.

Thus we cannot parallelize the sequence of computation across time steps.

3.Transformer

SELF ATTENTION

Let $h_1, h_2, \ldots, h_T$ denote input sequence.

$$\alpha_{ij} = softmax(f(h_i, h_j))$$

For a fixed i, we compare h_i with every h_j.
This gives a score for how relevant token j is to token i.

$$s_i = \sum_{j=1}^{T} \alpha_{ij} h_j$$

We can see that now s_t can be computed parallely.

Instead of directly computing $f(h_i, h_j)$, we first project h_i and h_j into different spaces.

$$q_i = W_Q h_i$$
$$k_i = W_K h_i$$
$$v_i = W_V h_i$$

- The query vector q_i represents what information token i is looking for.
- The key vector k_j represents what kind of information token j contains.
- The value vector v_j represents the actual information content that token j can contribute.

$$f(h_i, h_j) = q_i^T k_j$$

$$\alpha = \frac{exp(q_i^T k_j)}{\sum_{m=1}^{T} exp(q_i^T k_m)}$$

$$s_i = \sum_{j=1}^{T} \alpha_{ij} v_j$$

MULTI-HEAD ATTENTION

A single attention mechanism is often insufficient to capture the different types of relationships present in a sequence.
Thus we can have more than one self-attention heads with different parameter (W^i_Q, W^i_K, W^i_V) matrices so that it captures more meaningful interactions between inputs.

MASKED ATTENTION

Some tokens of the input sequence are purposefully omitted (masked) from contributing to attention weights. For example, future words are masked when training the decoder layer of a machine translation model.

Masking is done by inserting negative infinite at the respective positions.

q1•k1	$-\infty$	$-\infty$	$-\infty$	$-\infty$
q2•k1	q2•k2	$-\infty$	$-\infty$	$-\infty$
q3•k1	q3•k2	q3•k3	$-\infty$	$-\infty$
q4•k1	q4•k2	q4•k3	q4•k4	$-\infty$
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

AUTO-REGRESSIVE MODEL

It is a probabilistic model in which each element of a sequence is predicted using only the previously observed elements of the same sequence.
Mathematically, an autoregressive model estimates the joint probability of a sequence x_1, x_2,...,x_T as:

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, x_2, \ldots, x_T)$$

4.GPT

GPT (Generative Pre-trained Transformer) is an autoregressive language model that predicts the most probable next token in a sequence based on all previously generated tokens. Given an input prompt, GPT estimates the conditional probability distribution of the next token and generates text one token at a time.

The power of GPT models comes from two key aspects:

- **Generative pretraining** that teaches the model to detect patterns in unlabeled data, then apply those patterns to new inputs.
- A **transformer architecture** that enables the model to process all portions of an input sequence in parallel.