Encoder And Decoder

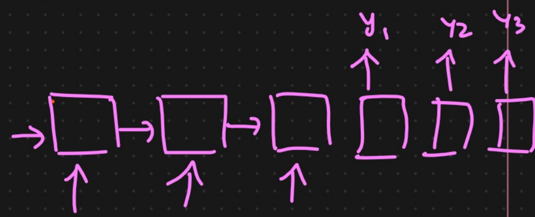
1. Simple RNN → Vanishing Gradient Problem
2. LSTM RNN →
3. GRU RNN →

} → Long Short Term Memory.

4. Bidirectional RNN ←

Type of RNN

1. Many to Many RNN



Encoder And Decoder?

Sequences I/p

o/p Sequence Of Words

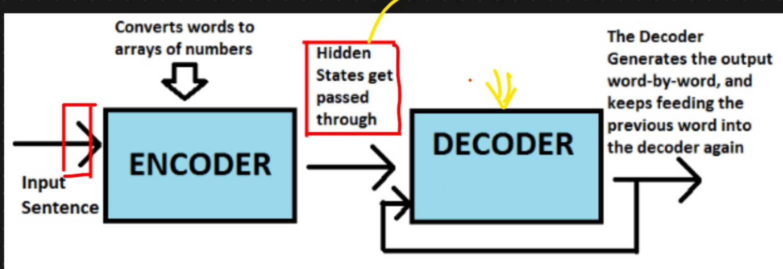
Eg:- One Language To other

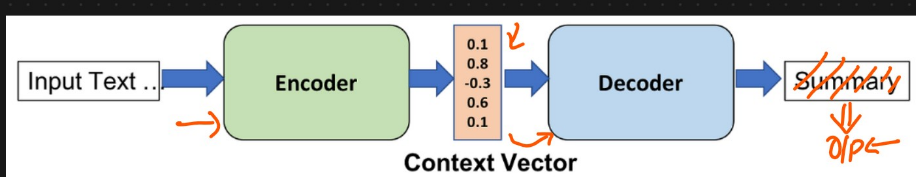
English → French

Eg:- Linked Chat → Hi, How are you?

Simple Working

Context Vectors



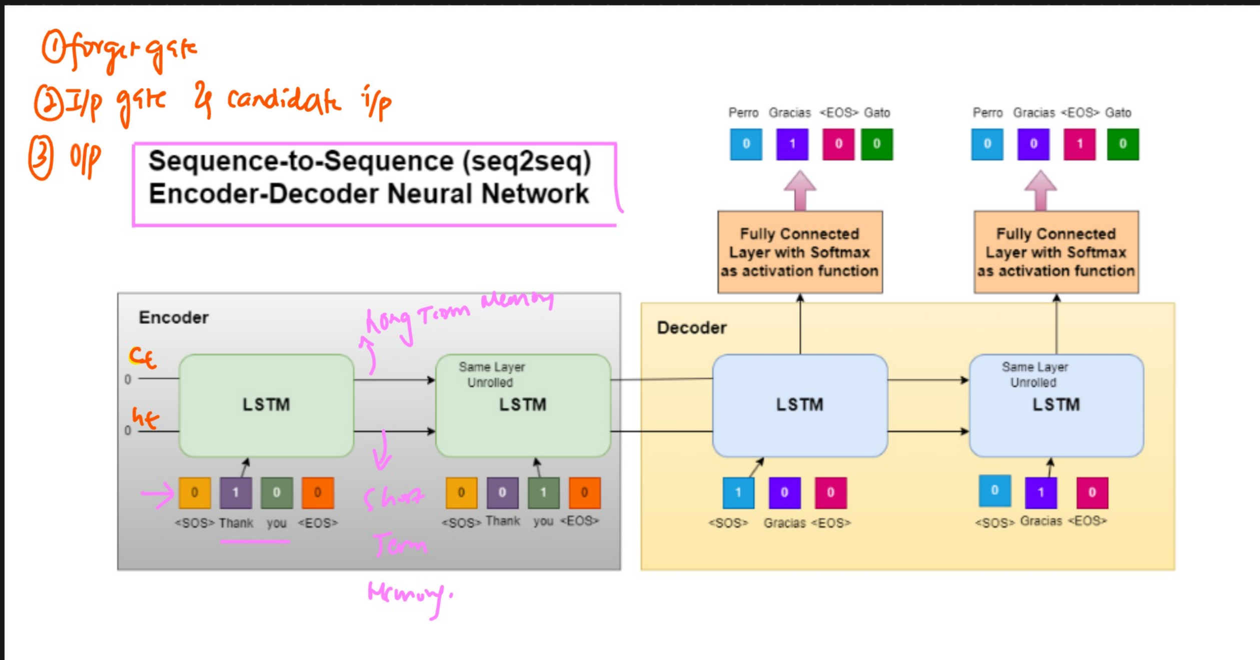


1. Encoder ⇒ I/p ⇒ Context Vector ⇐ Vectors
2. Decoder ⇒ ⇒ O/P

Usecase

1. Language Translation
2. Text Generation
3. Text Suggestion.

RNN → Vanishing Gradient Problem



$\hat{y}$

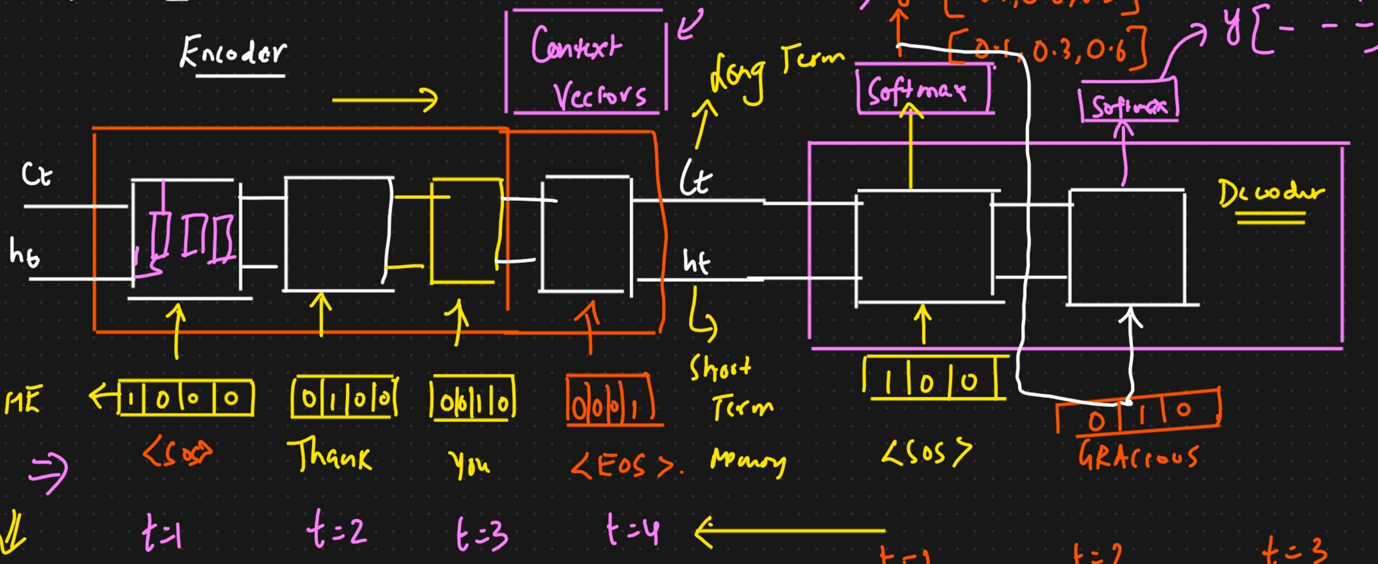
Dataset

English | True o/p French | ytruth {Training}

<SOS> Thank you <EOS>. Gracias → <SOS> GRACIAS <EOS>. GRACIOUS

$[y\ \hat{y}]$ <EOS>

Encoder → Context Vectors

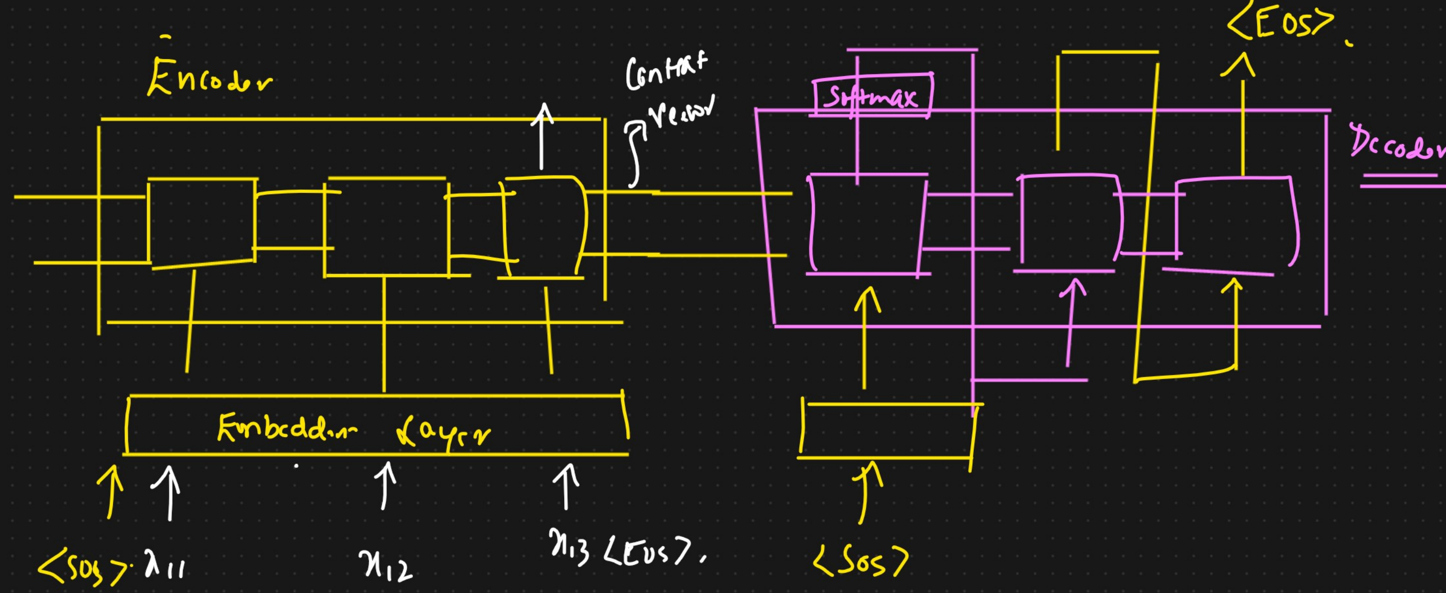


⇒ Context of this sentence.

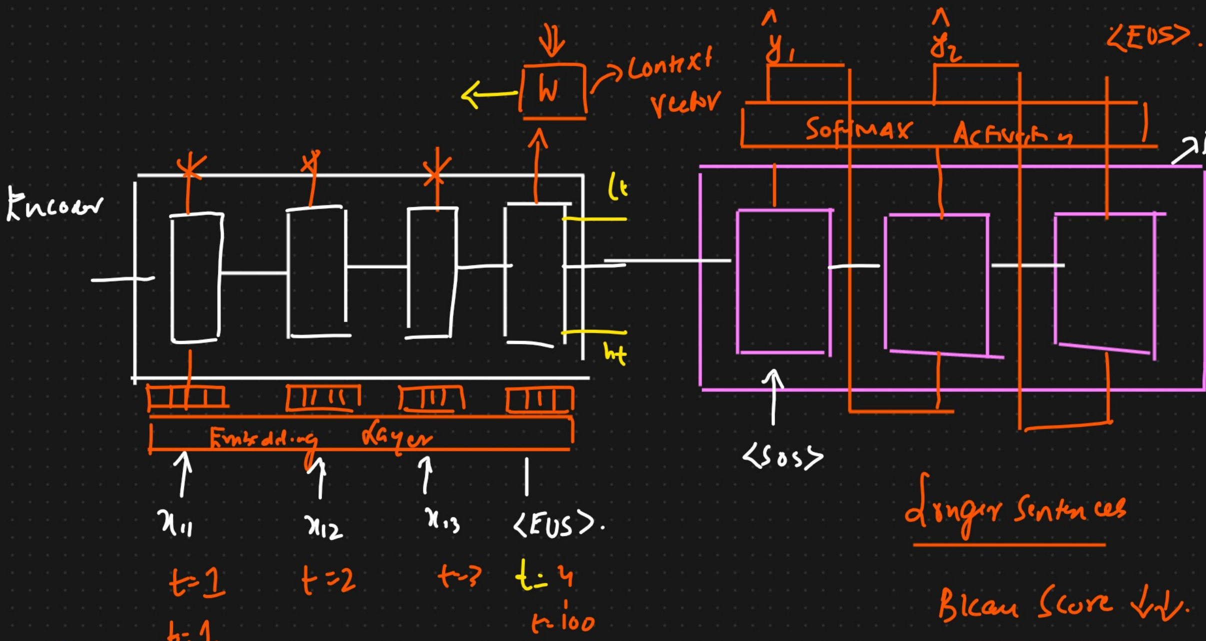
	t=1	t=2	t=3
$\hat{y}$	[- - -]	[- - -]	[- - -]
y	[- - -]	[- - -]	[- - -]

Loss $(y-\hat{y})^2$

Optimizer



Problems With Encoder-Decoder Seq2seq Architecture

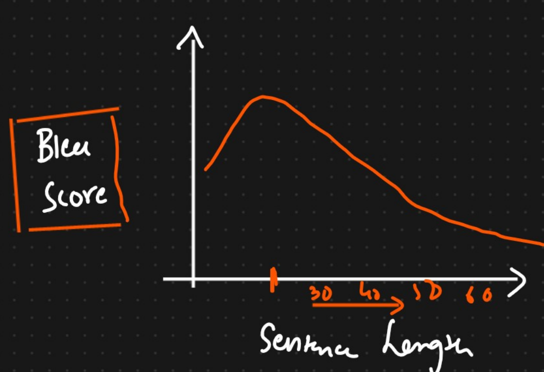


Context vector ⇓ Represents the entire sentence.

Longer sentences

Bleu Score ↓↓.

Researchers : Sentences of varying length

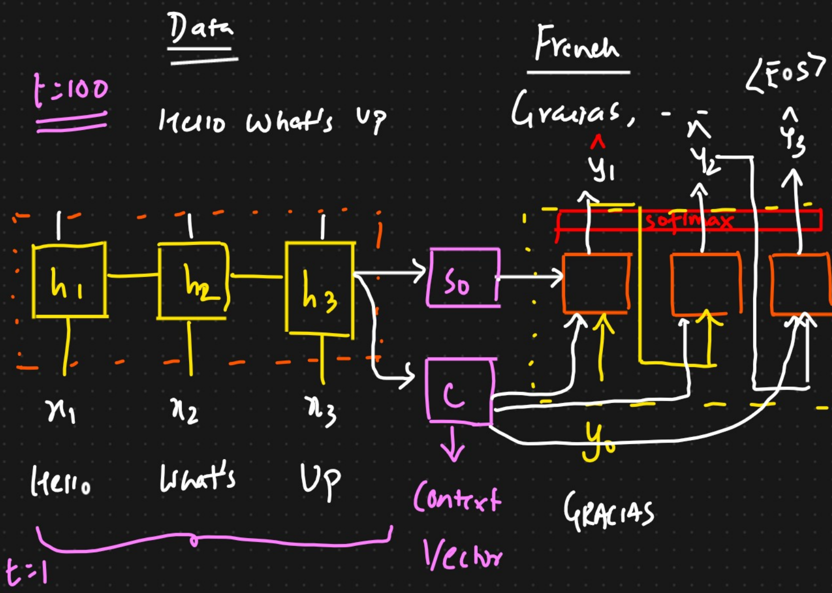


⇒ Seq to Seq Data.

(*) Attention Mechanism → Seq2Seq Network

longer paragraph → {context vector}.
+
{context}

Attention Mechanism | Seq2Seq Networks



3 LEARNING TO ALIGN AND TRANSLATE

In this section, we propose a novel architecture for neural machine translation. The new architecture consists of a bidirectional RNN as an encoder (Sec. 3.2) and a decoder that emulates searching through a source sentence during decoding a translation (Sec. 3.1).

3.1 DECODER: GENERAL DESCRIPTION

In a new model architecture, we define each conditional probability in Eq. (2) as:

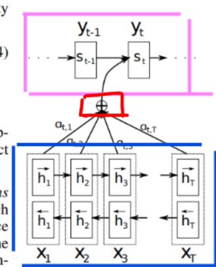
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, c_i), \qquad (4)$$

where s_i is an RNN hidden state for time i, computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder–decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i.

The context vector c_i depends on a sequence of *annotations* $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th word of the input sequence. We explain in detail how the annotations are computed in the next section.



The context vector c_i is, then, computed as a weighted sum of thes
annotations h_i:

$$\left\{ c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \right\} \qquad (5$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \left\{ \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \right\}$$

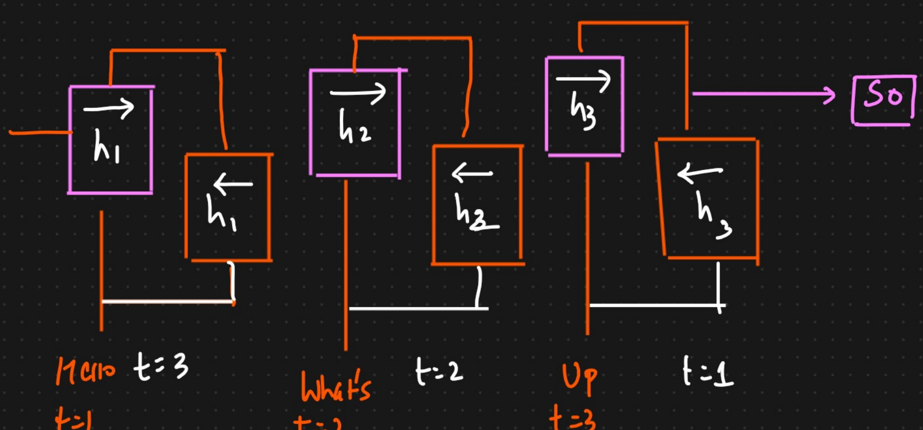
where

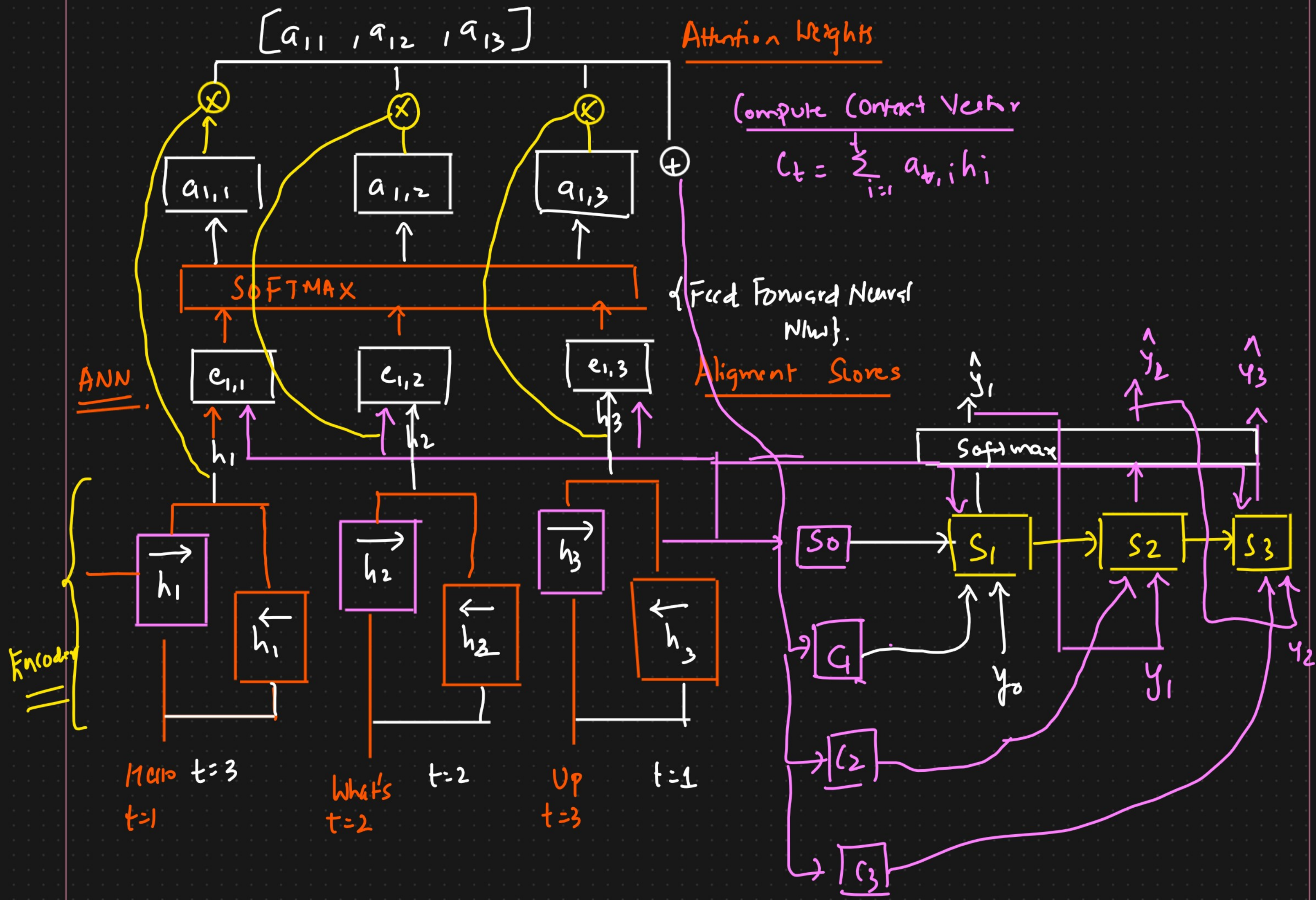
$$e_{ij} = a(s_{i-1}, h_j)$$

Encoder Decoder Architecture

Attention Mechanism

https://erdem.pl/2021/05/introduction-to-attention-mechanism





Hello , What's up