TÜBİTAK

NATURAL LANGUAGE PROCESSING RESEARCH GROUP

EURO²

Lect. Tuğba Pamay Arslan

ITUNLP Research Group

AI & Data Engineering, İstanbul Technical University

ncc@ulakbim.gov.tr

The Power of LLMs: Transformer (Part 1)

Table of Contents

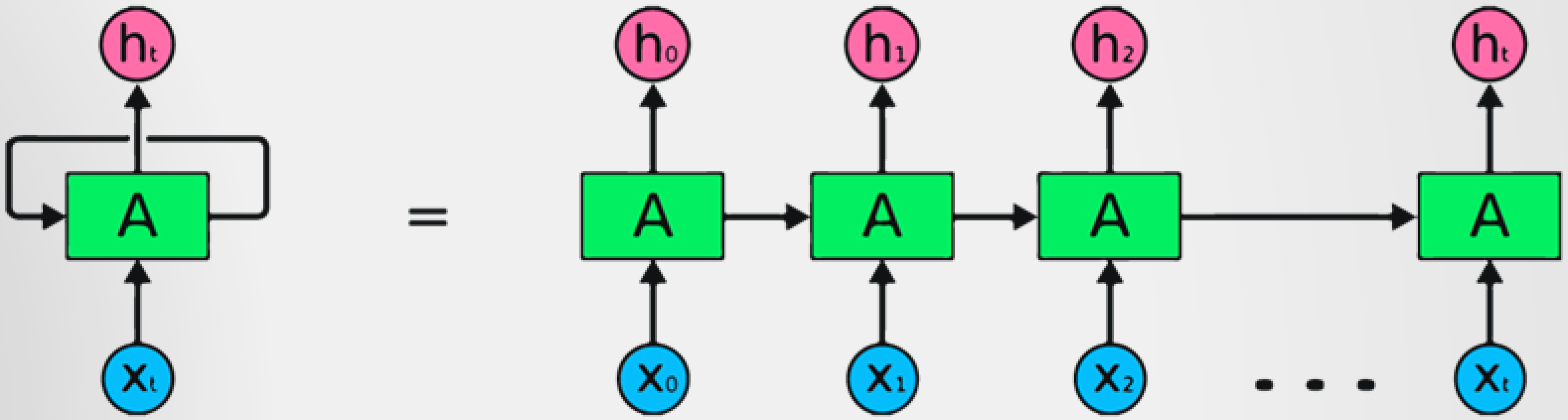
Recurrent Neural Networks

➔ A specialized version of feed-forward networks for sequence modeling
  - e.g. time series, speech, text.

➔ Have connections that form cycles, allowing them to use information from previous inputs to inform the current output.

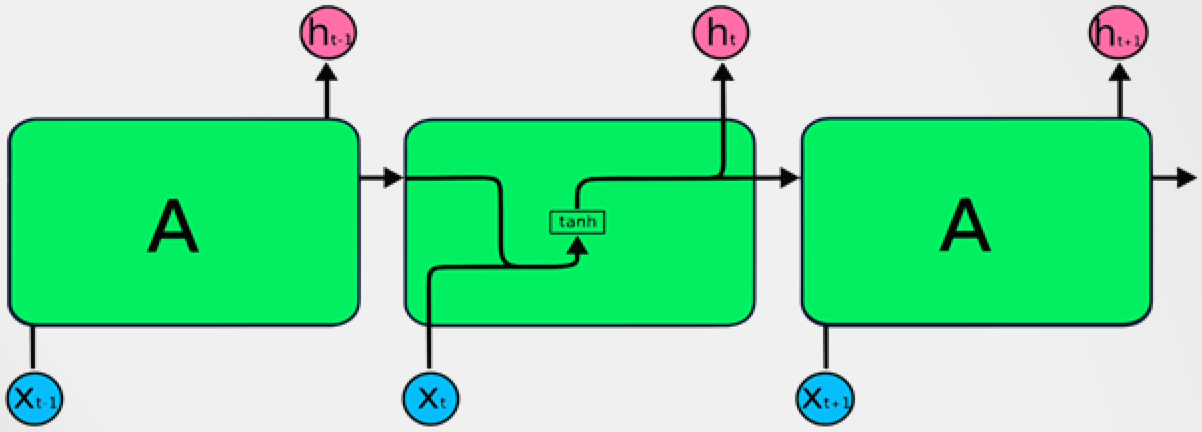
➔ Effective for tasks where context matters.

➔ Flexible as the length of inputs and outputs can be changed



https://www.datacamp.com/tutorial/tutorial-for-recurrent-neural-network

Recurrent Neural Networks



$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

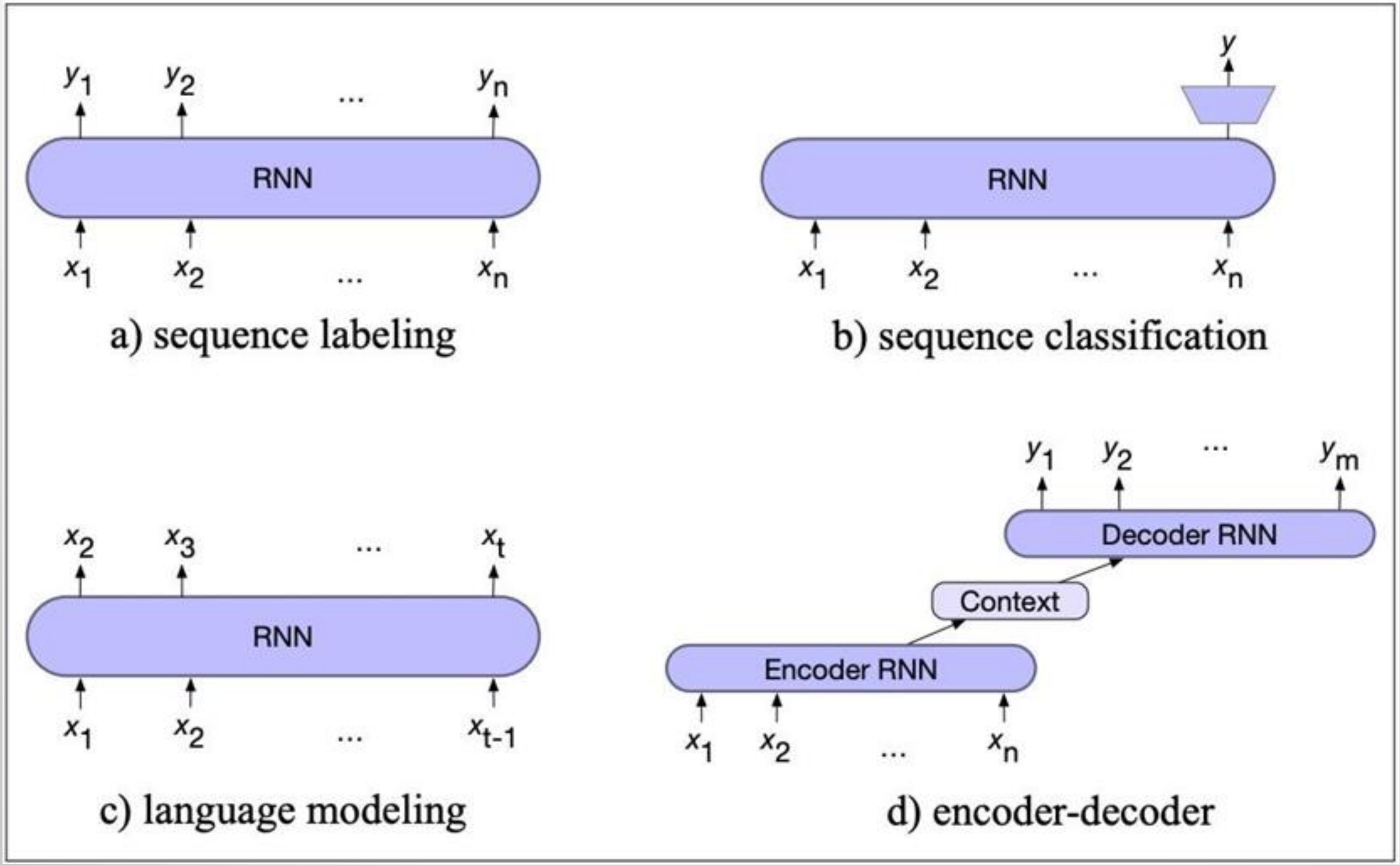
current state

function parametrized by W

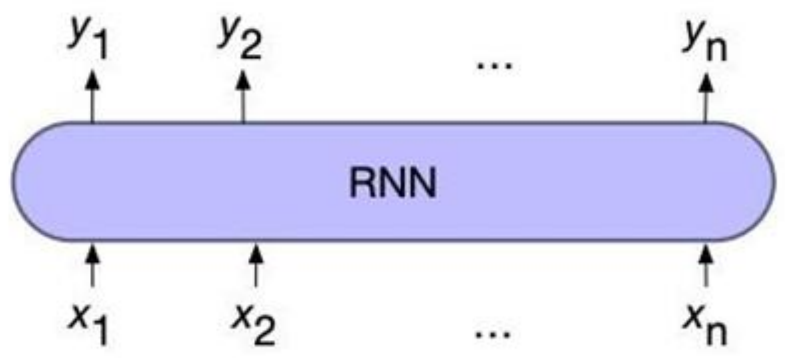
previous state

input at time step t

RNN Architectures in Sequence Modeling



Sequence Labeling



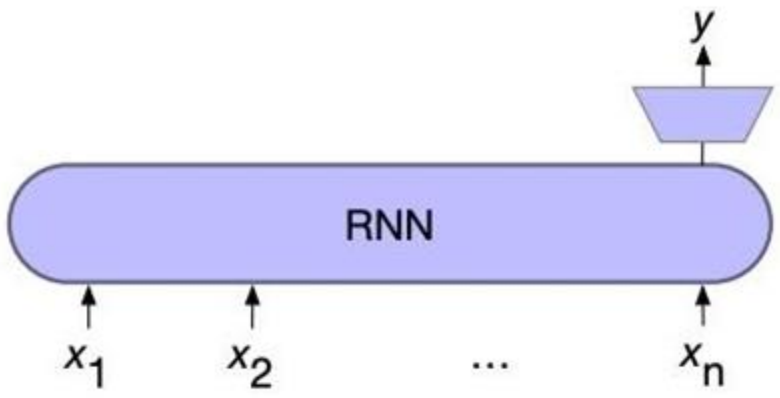
a) sequence labeling

➔ One-to-One: Each element of the input sequence is assigned a label.

➔ One Input >> One Output

➔ Example: Named Entity Recognition

Sequence Classification

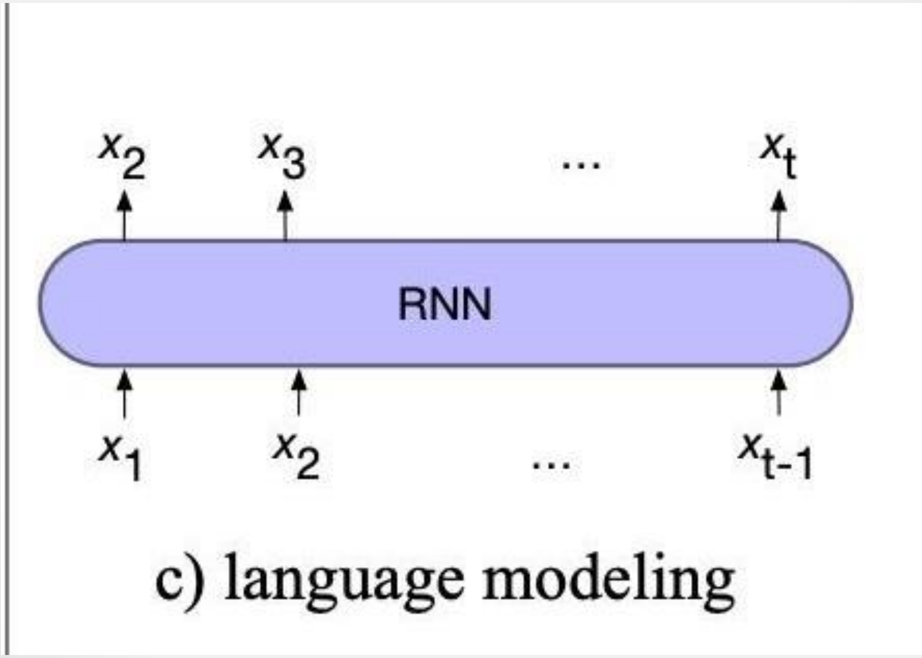


b) sequence classification

➔ Many-to-One: Classifying an entire sequence into one label

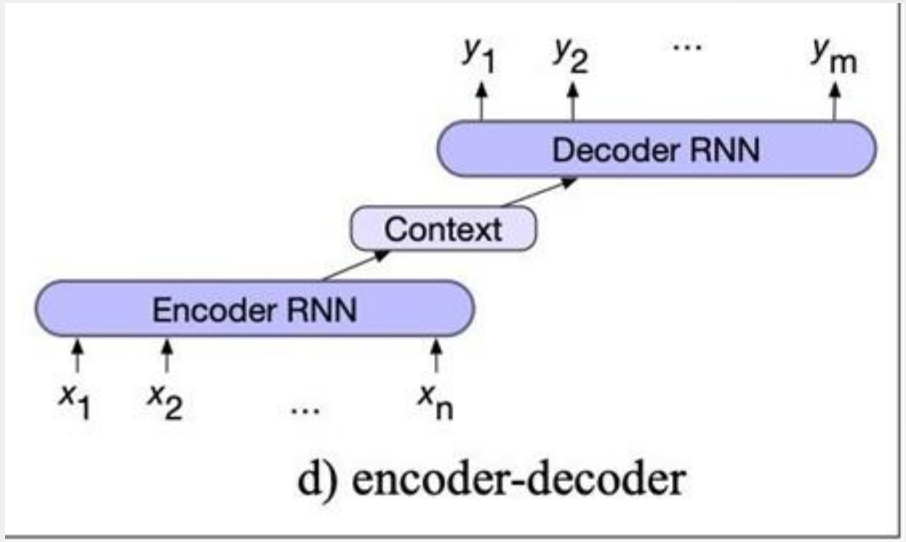
➔ Example: Sentiment Analysis

Language Modeling



- ➔ One-to-One: Predicting the next word in a sequence given previous words
- ➔ For this time, the output is continuous and represents the likelihood of the next token.
- ➔ Then, this next token is used as an input in the next time step.

Encoder-Decoder



- ➔ Example: Machine Translation, Text Summarization, or Question Answering
- ➔ Two continuous steps:
  - ➔ Encoding: The input is encoded into a representation
  - ➔ Decoding: Generates a corresponding output sequence based on the encoded input (context)

The Encoder-Decoder with RNNs

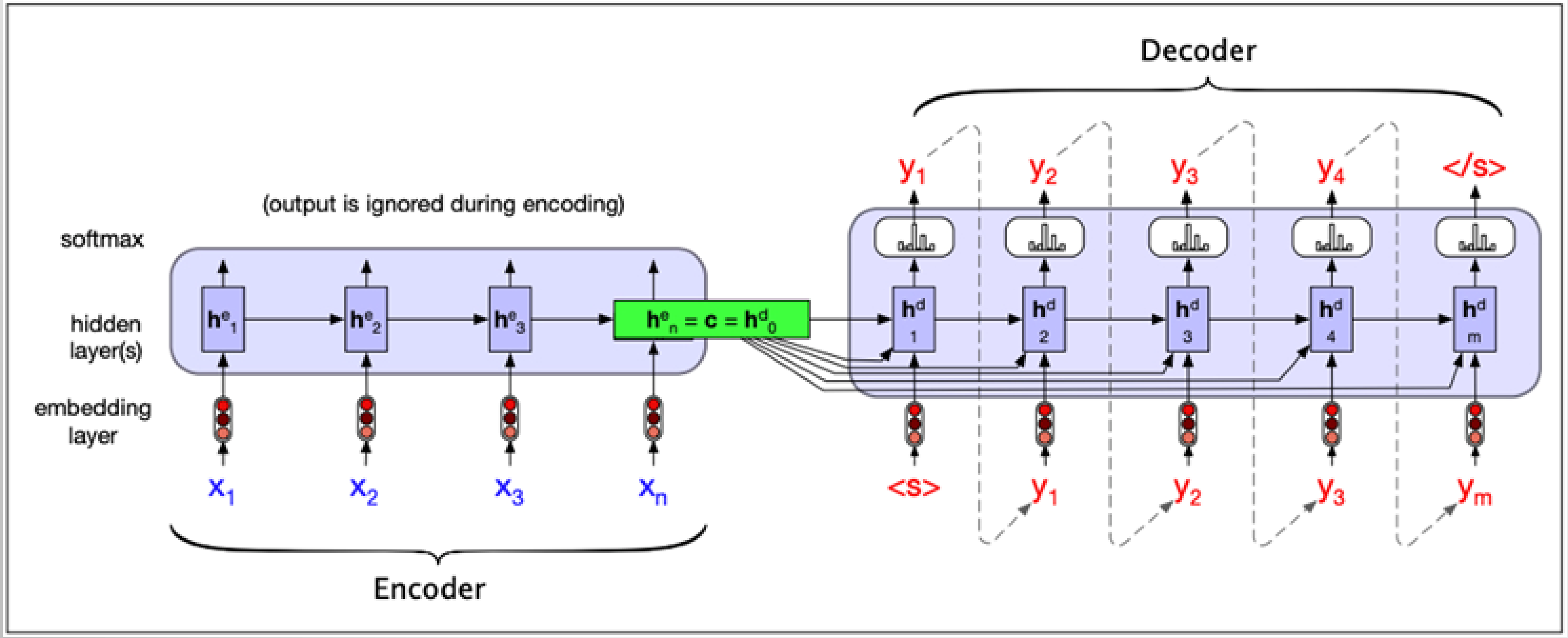
- ➔ also known as sequence-to-sequence networks (seq2seq)
- ➔ |Input| may vary from |Output|
- ➔ Generate contextually appropriate output sequences of arbitrary length, given an input sequence.
- ➔ Particularly popular for Machine Translation. But also,
  - ◆ Summarization, Question Answering etc.

The Encoder-Decoder with RNNs

Consist of 3 conceptual components:

➔ **Encoder:**

- Process the input sequence, $x_{1:n}$
- Create a contextualized representation (i.e., the context), $h_{1:n}$

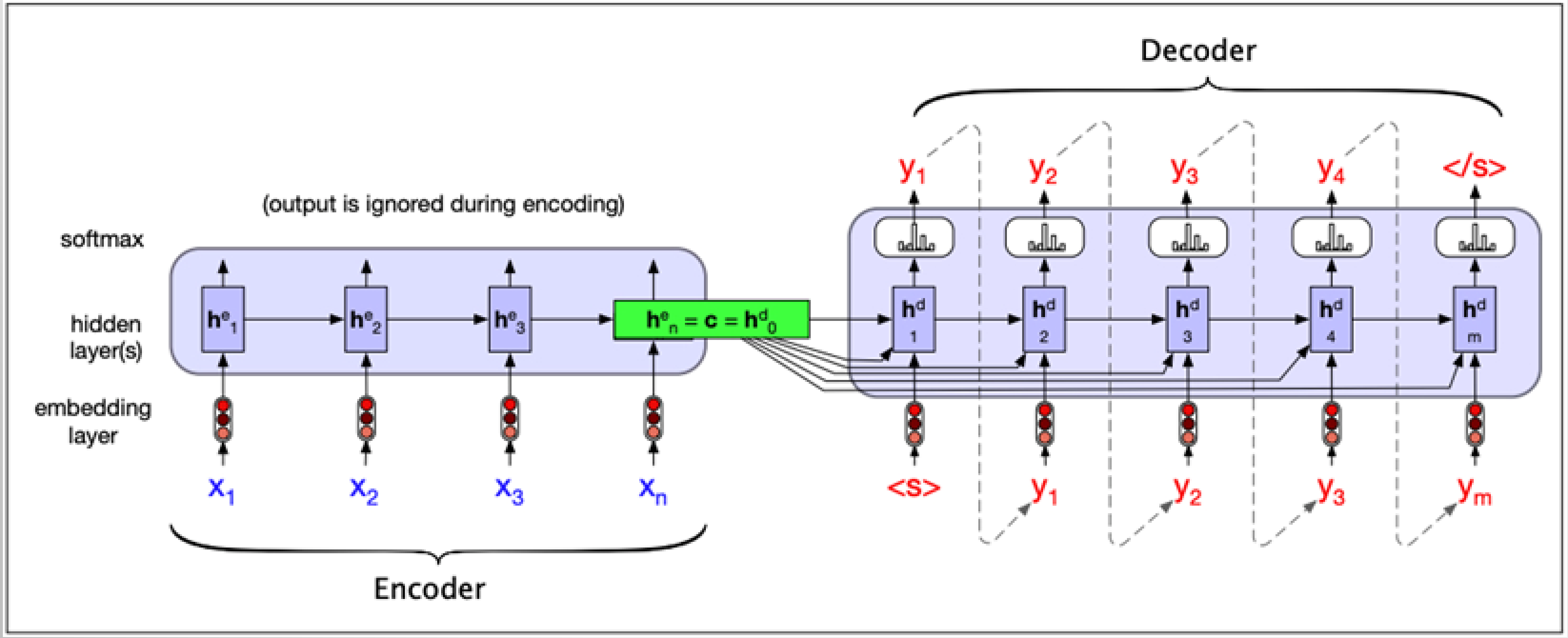


The Encoder-Decoder with RNNs

Consist of 3 conceptual components:

➔ **Context Vector, c:**

- A function of $h_{1:n}$
- Conveys the essence of the input to the decoder.

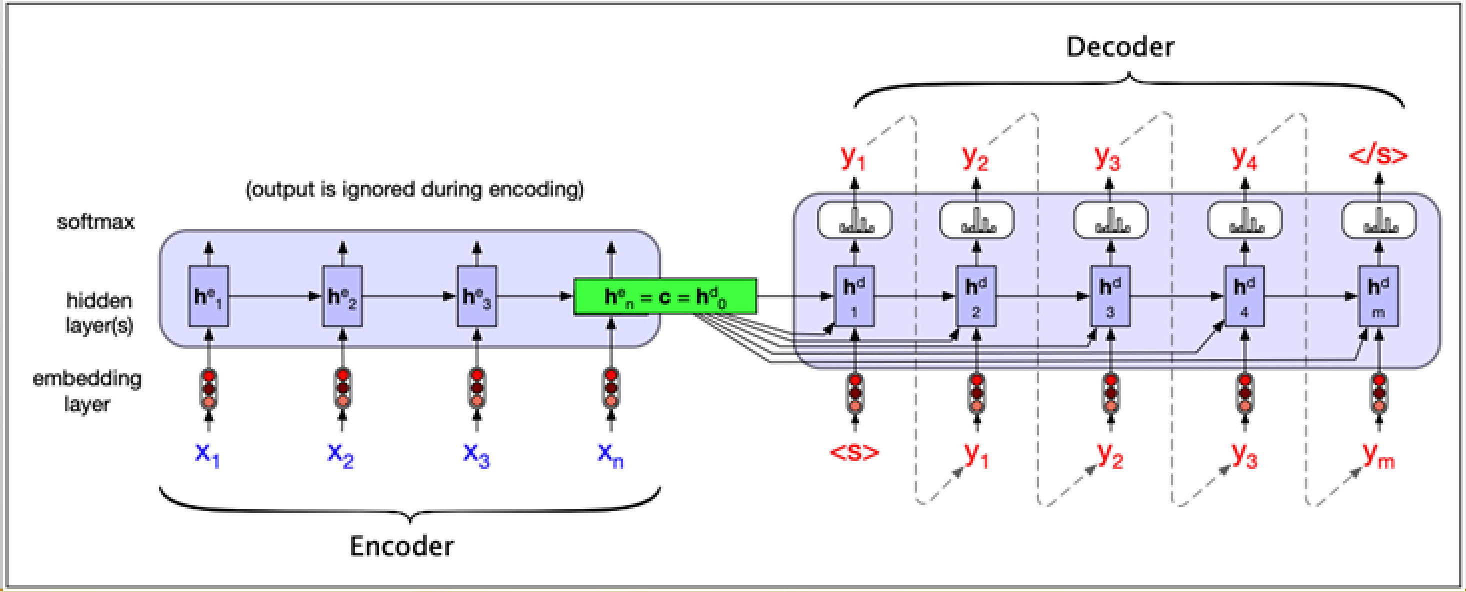


The Encoder-Decoder with RNNs

Consist of 3 conceptual components:

➔ **Decoder:**

- Accepts c as input
- Generates an arbitrary length sequence of hidden states, $h_{1:m}$
- Also, a corresponding sequence of output states $y_{1:m}$ can be obtained.



Limitations ?

Fixed-Length Encoding Bottleneck

Encoder compressing the entire input sequence into a single fixed-length context vector.

|Input sequence| ↑ >>> Crucial details may be lost

Capturing Long-Term Dependencies

RNNs struggle with long-term dependencies due to vanishing gradients and limited memory.

Sequential Processing / Limited Parallelism

Computationally slow, NOT compute the outputs of different time steps in parallel

Attention

➔ *Attend to* different parts of the **another sequence** when making predictions.

Processing all parts of the input **equally**

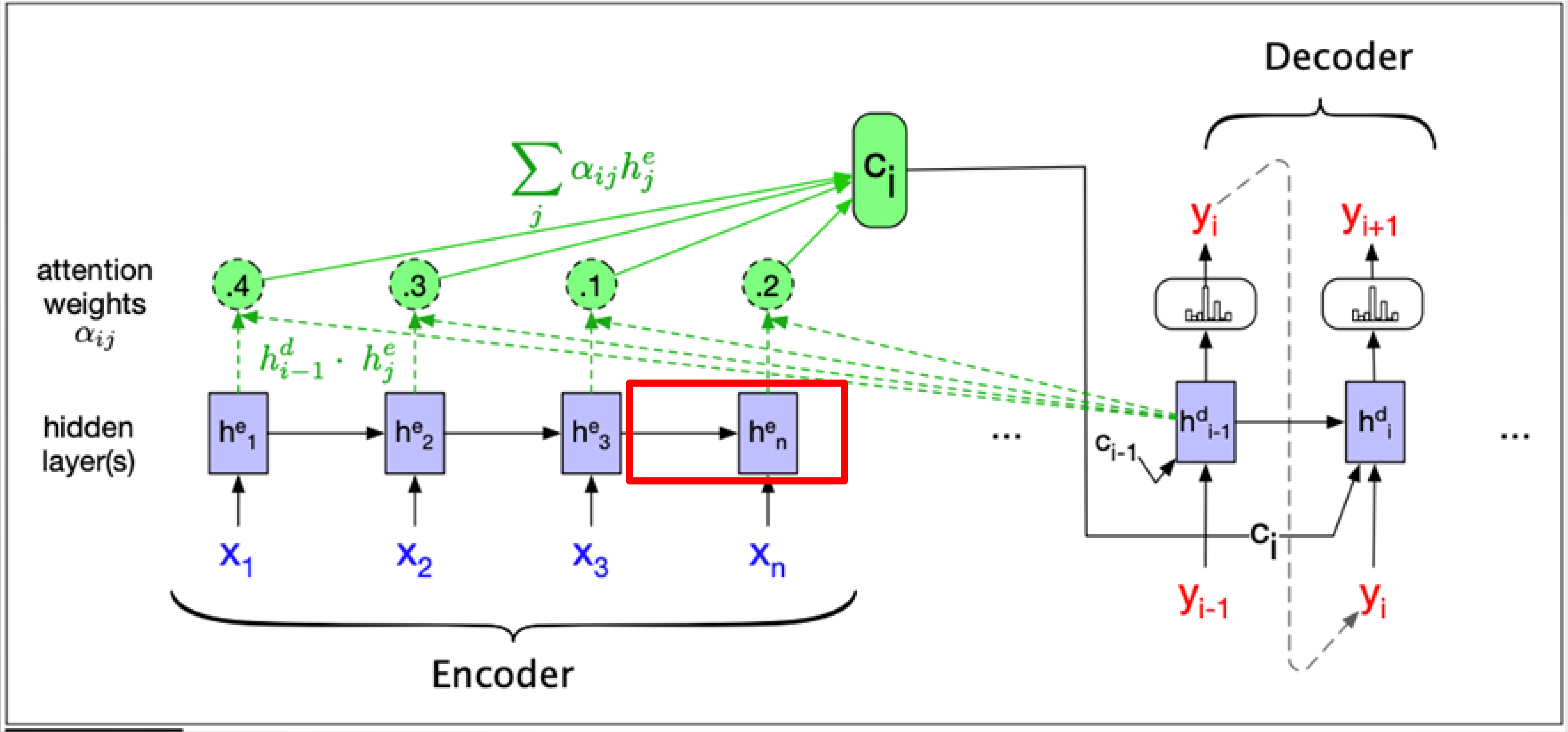
⇒

Attend to the most relevant information at each step

➔ Replaces the static context vector, c, with one that is dynamically derived from the encoder hidden states, different for each token in decoding.

Attention

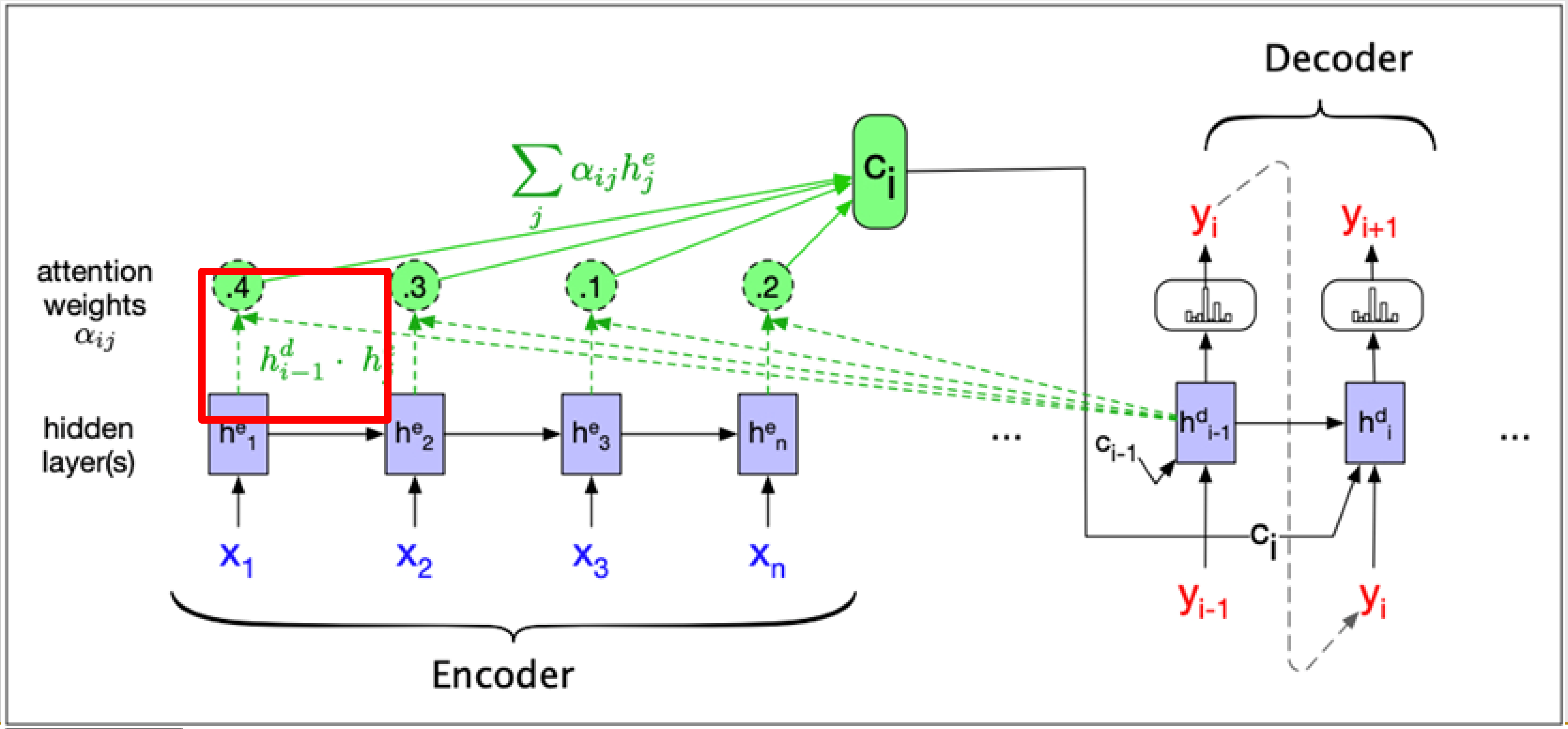
Step #1: $$\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) = \mathbf{h}^d_{i-1} \cdot \mathbf{h}^e_j$$



Attention

Step #2:

$$\begin{aligned}\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)) \\ &= \frac{\exp(\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)}{\sum_k \exp(\text{score}(\mathbf{h}^d_{i-1}, \mathbf{h}^e_k))}\end{aligned}$$



Attention

Step #3:

$$\mathbf{c}_i = \sum_j \alpha_{ij}\,\mathbf{h}^e_j$$