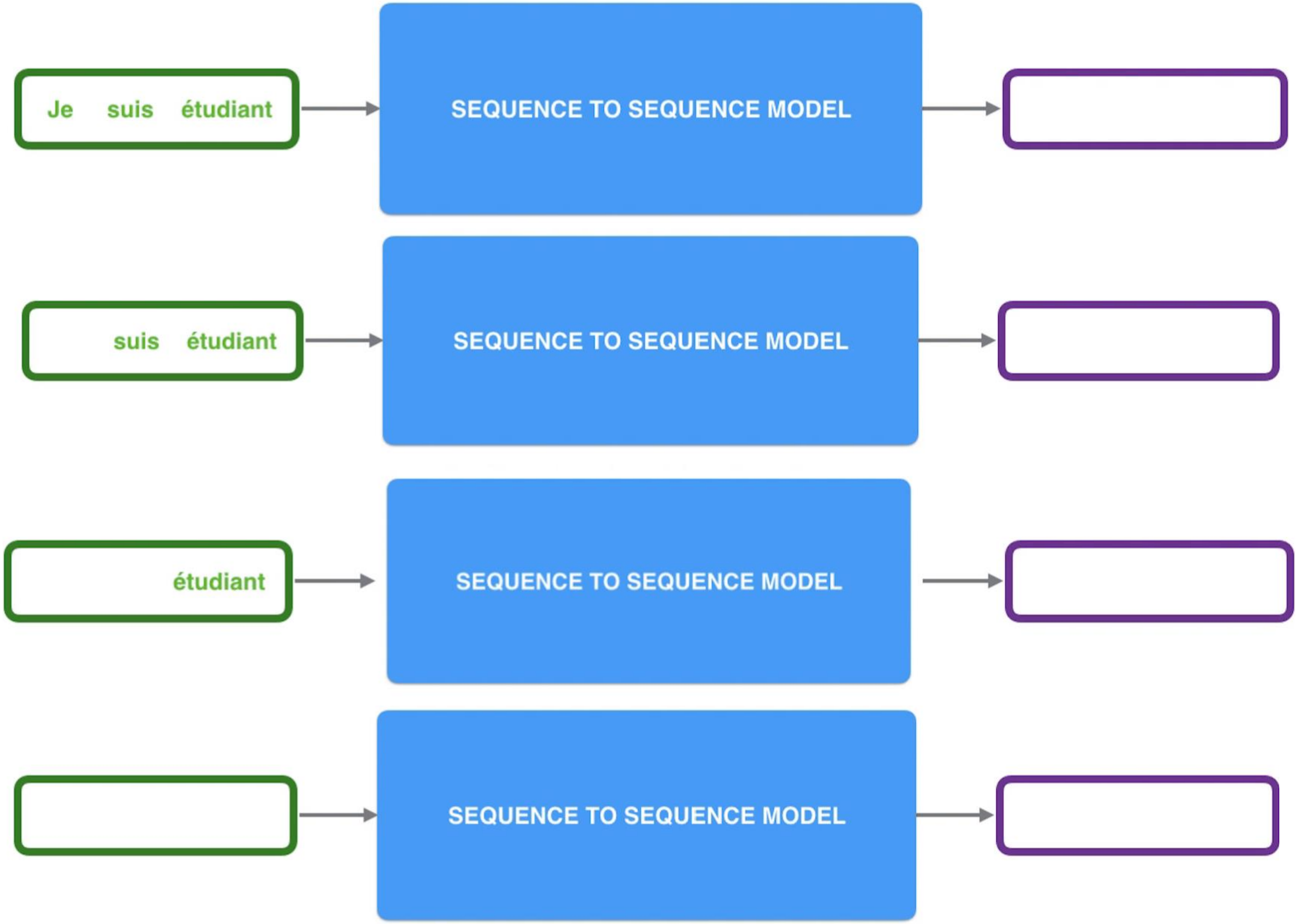
Sequence to Sequence Model

Syllabus

- Sequence to Sequence Model Overview
- Encoder-Decoder
- Attention Mechanism

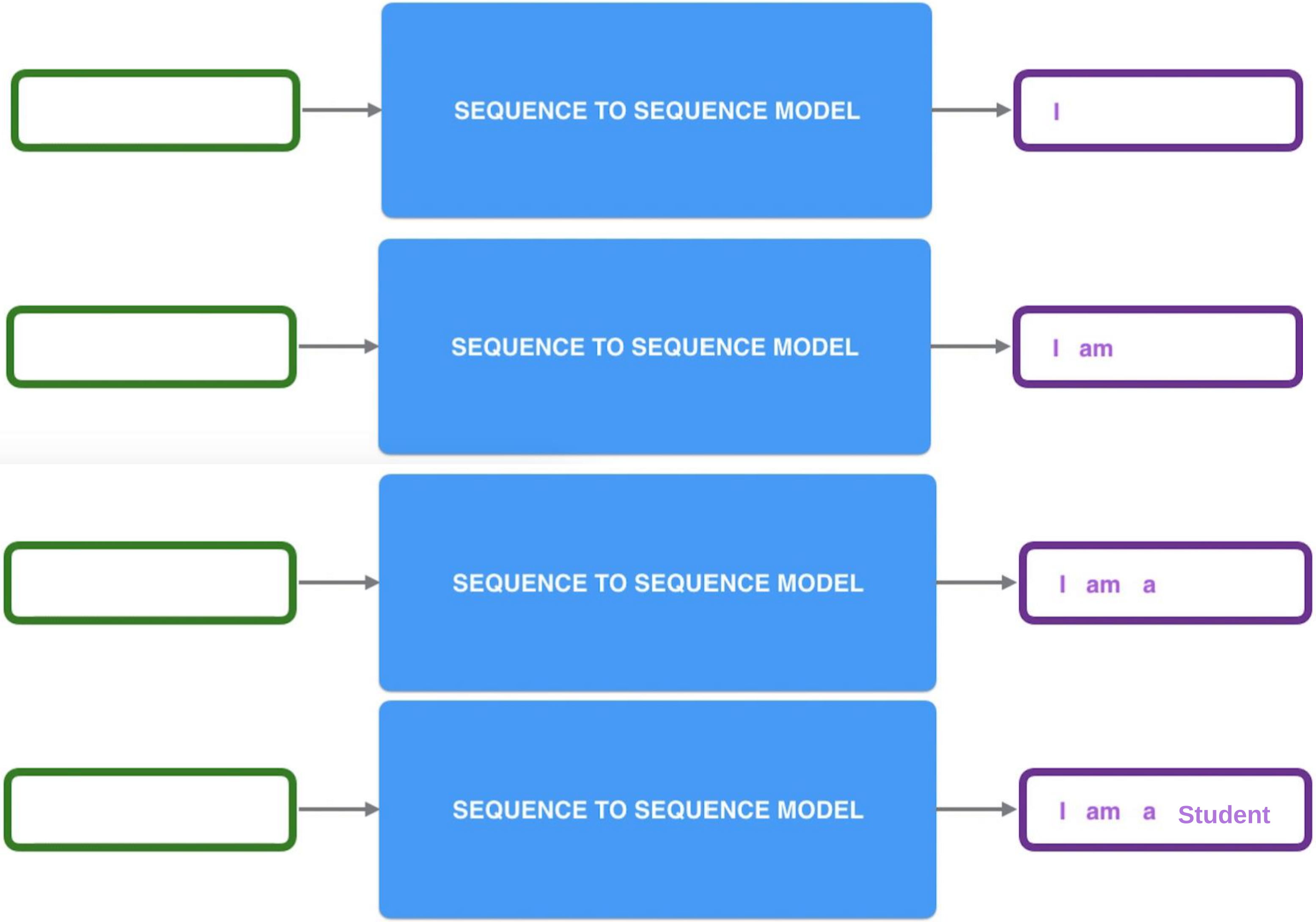
Sequence to Sequence Model Overview

Sequence and Sequence model is an interesting NLP model that can do **mapping** between input text and output text. Specifically, the input can be a **sequence of text** (source language).



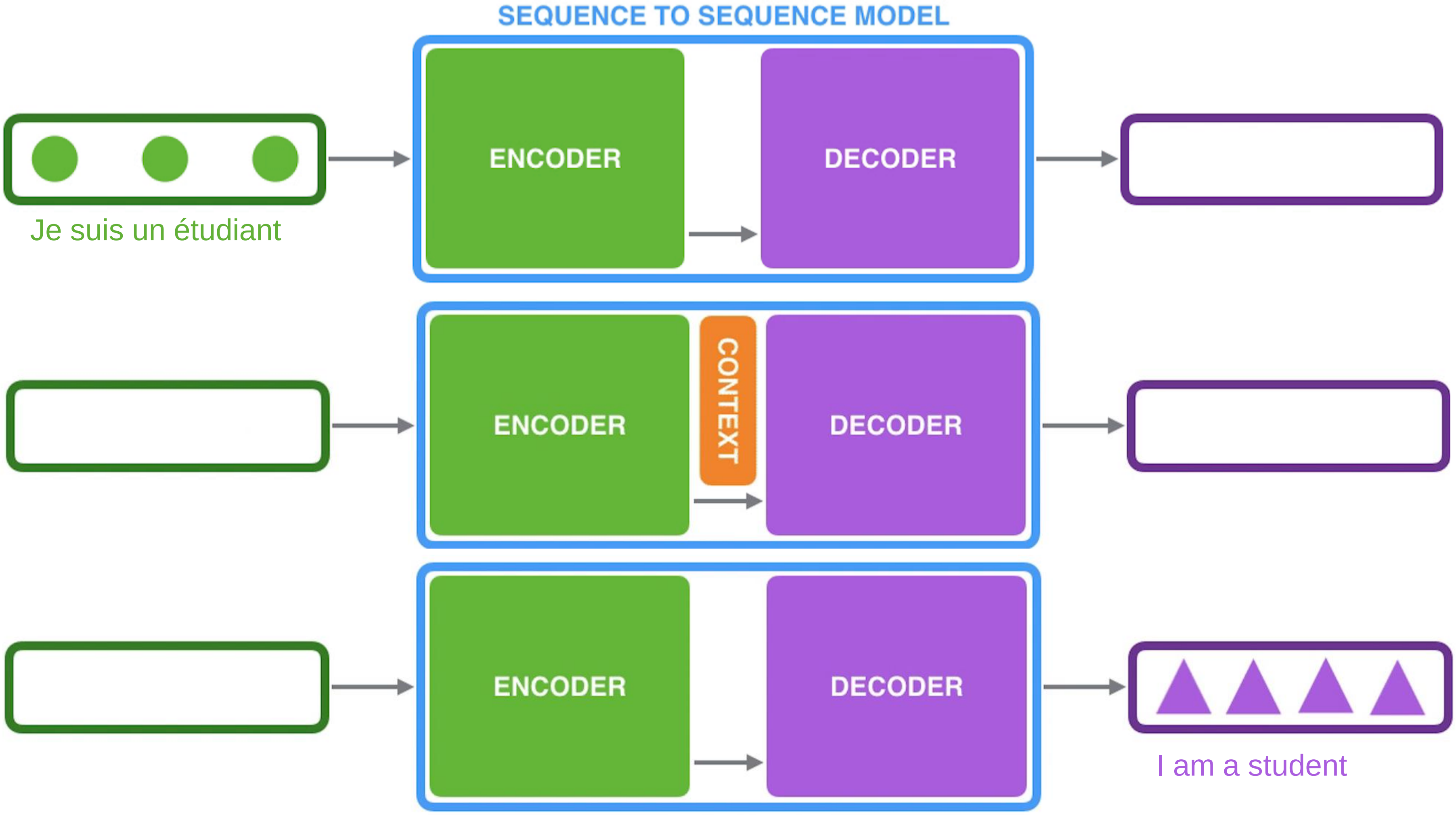
Sequence to Sequence Model Overview

The output is also a sequence of text (target language) in machine translation use case.



Sequence to Sequence Model Structure: Encoder-Decoder

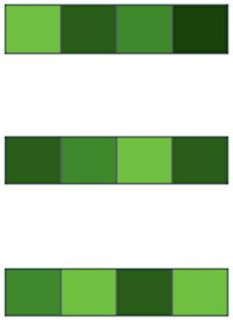
Encoder: Handles **each element** of text sequence (a word) can transform them into **Context Vector**
Decoder: Generates **output text sequence** based on Context Vector



Input: Word Embedding

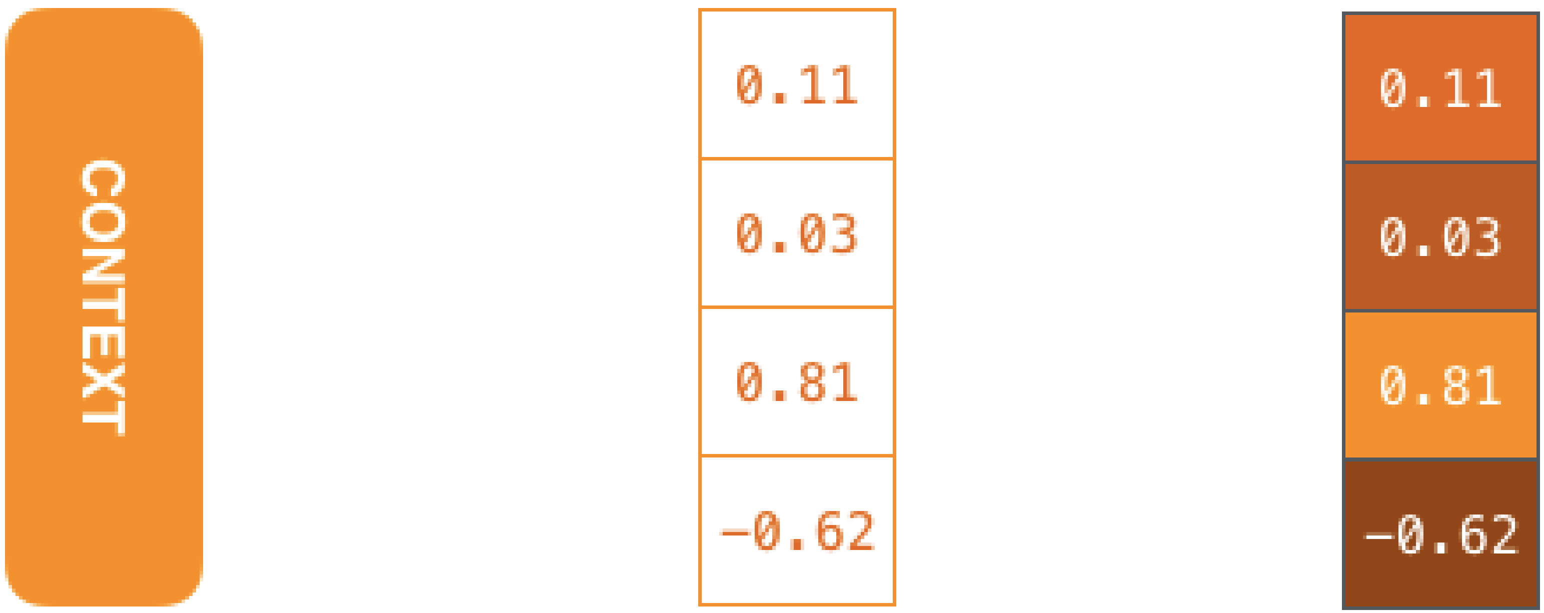
Word Embedding: Normally, the input text sequence is encoded as a **dense vector**

Je	0.901	-0.651	-0.194	-0.822
suis	-0.351	0.123	0.435	-0.200
étudiant	0.081	0.458	-0.400	0.480



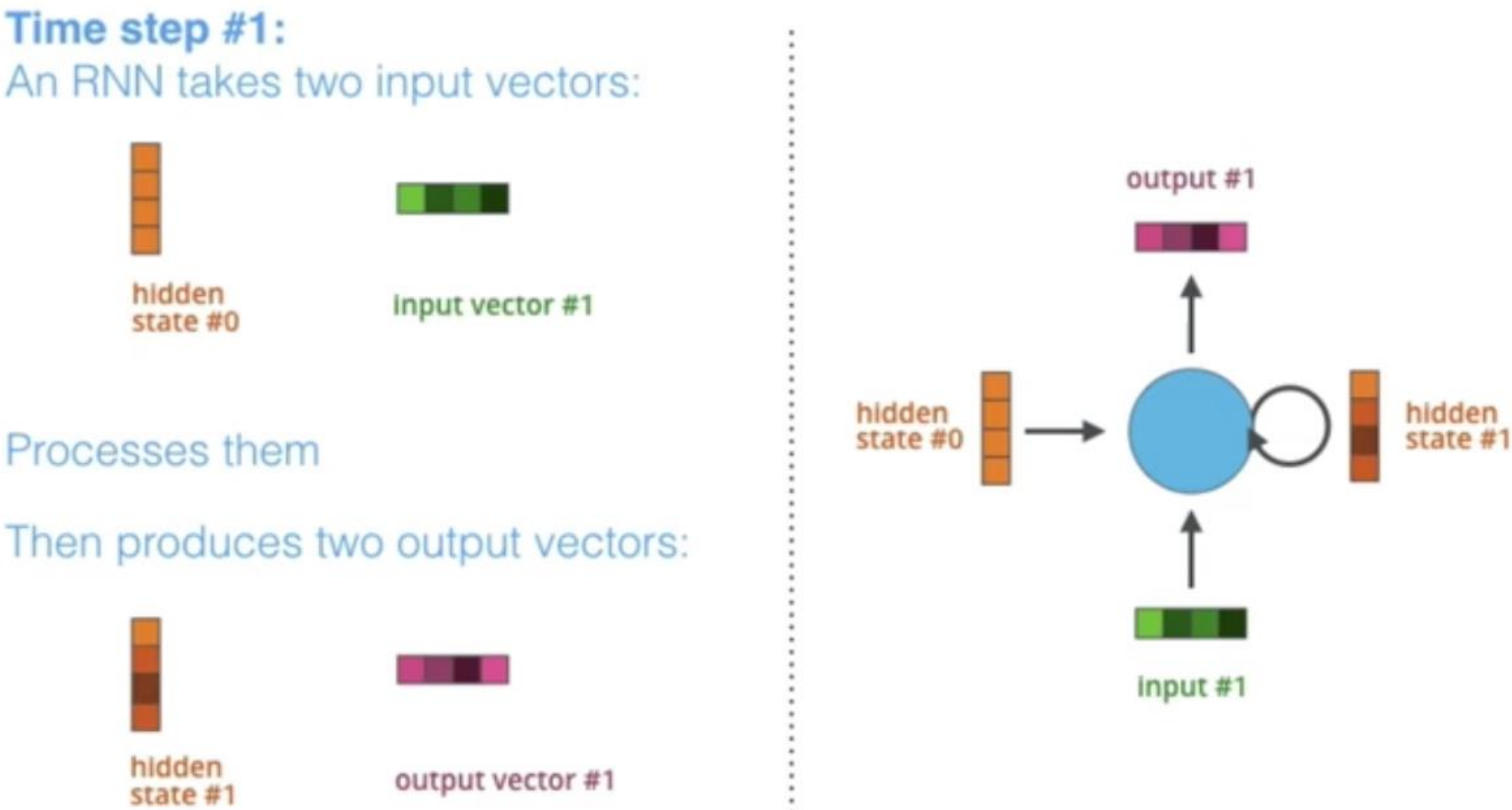
Context Vector

Context Vector: Generated by **Encoder** and will be feed into **Decoder**



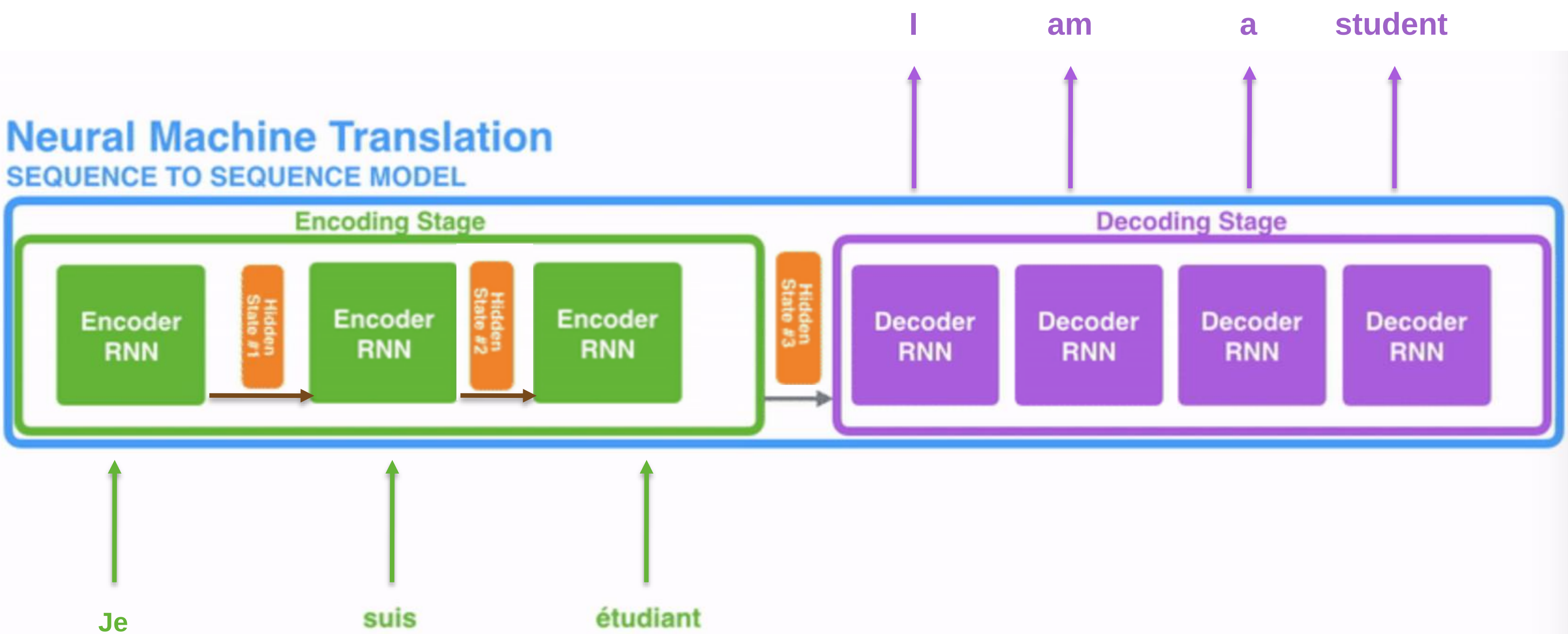
Implementation of Encoder and Decoder: Recurrent Neural Network

Recurrent Neural Network: The **input vector** of each timestamp will be processed with the **previous hidden state** and then generate the corresponding **hidden state** and output.

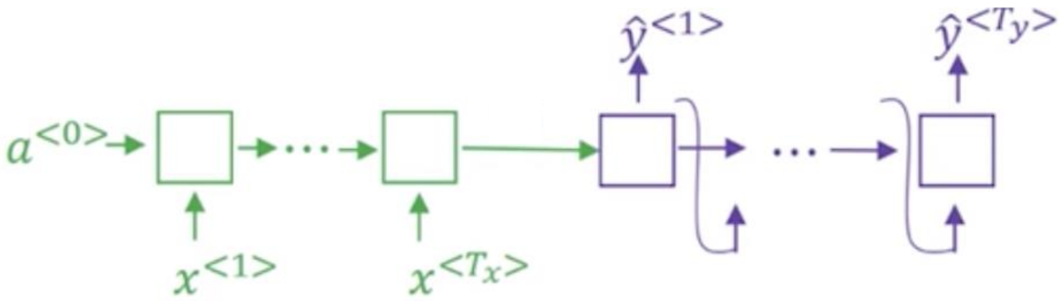


$S_t=f(UX_t+WS_{t-1})$ $O_t = \mathrm{softmax}(VS_t)$

RNN in Encoder and Decoder



Problems of long sequences

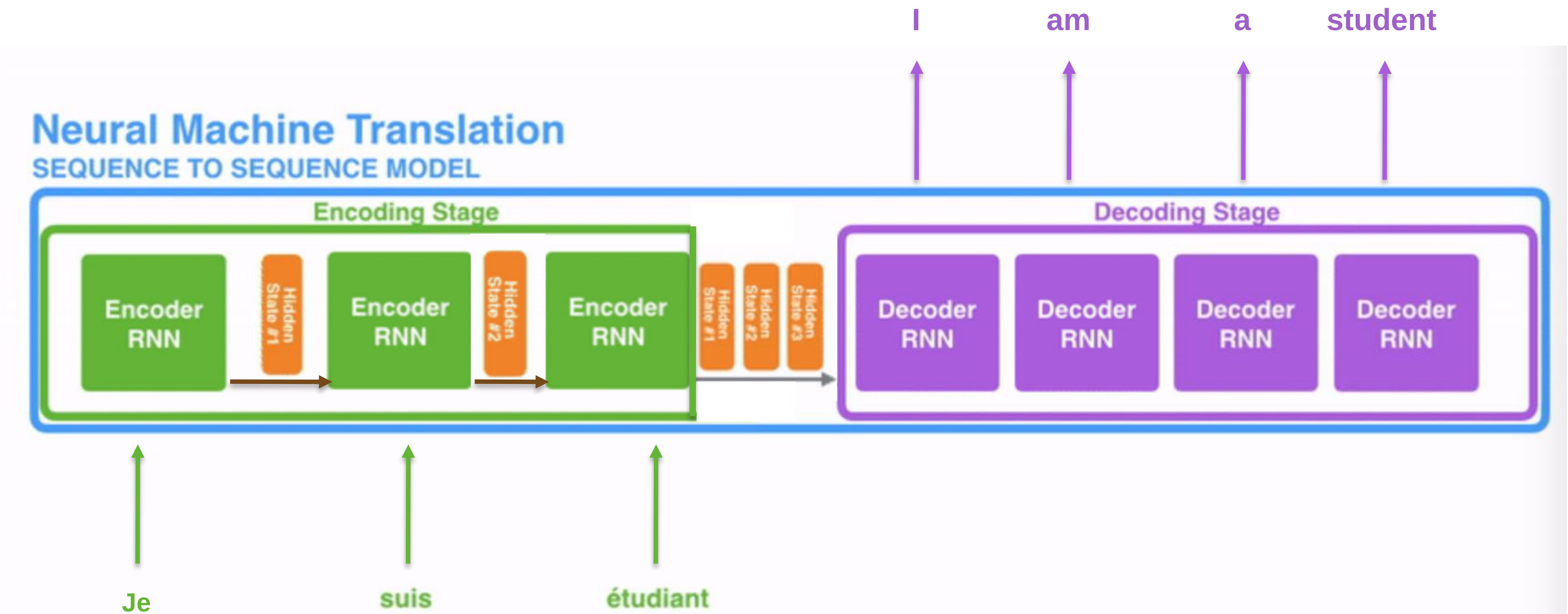


Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.

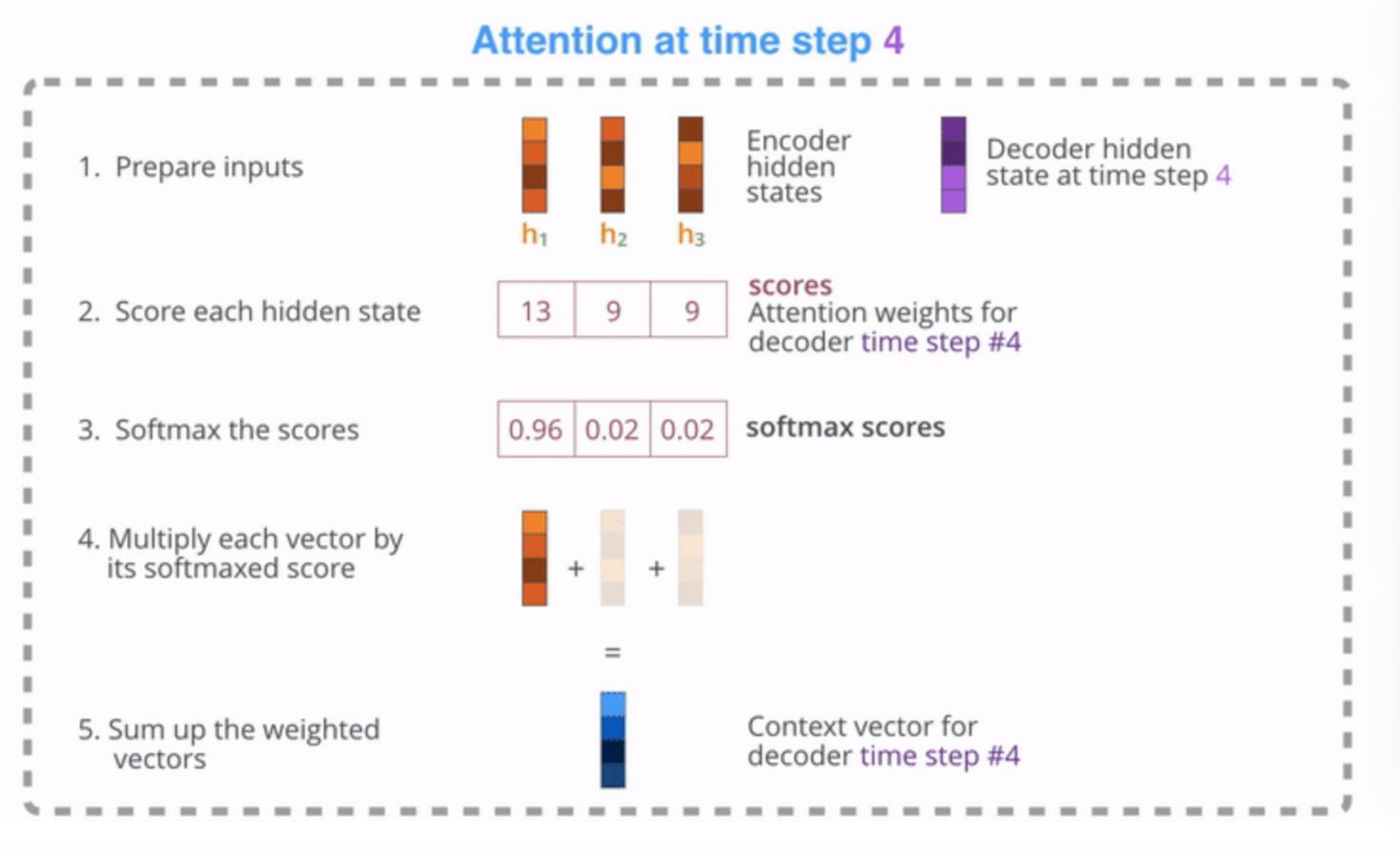
Solution: Attention in Encoder and Decoder

To learn the alignment between the source language and the targeted language, **Attention** is applied in Sequence and Sequence model instead of using the context information that is put into a single context vector.



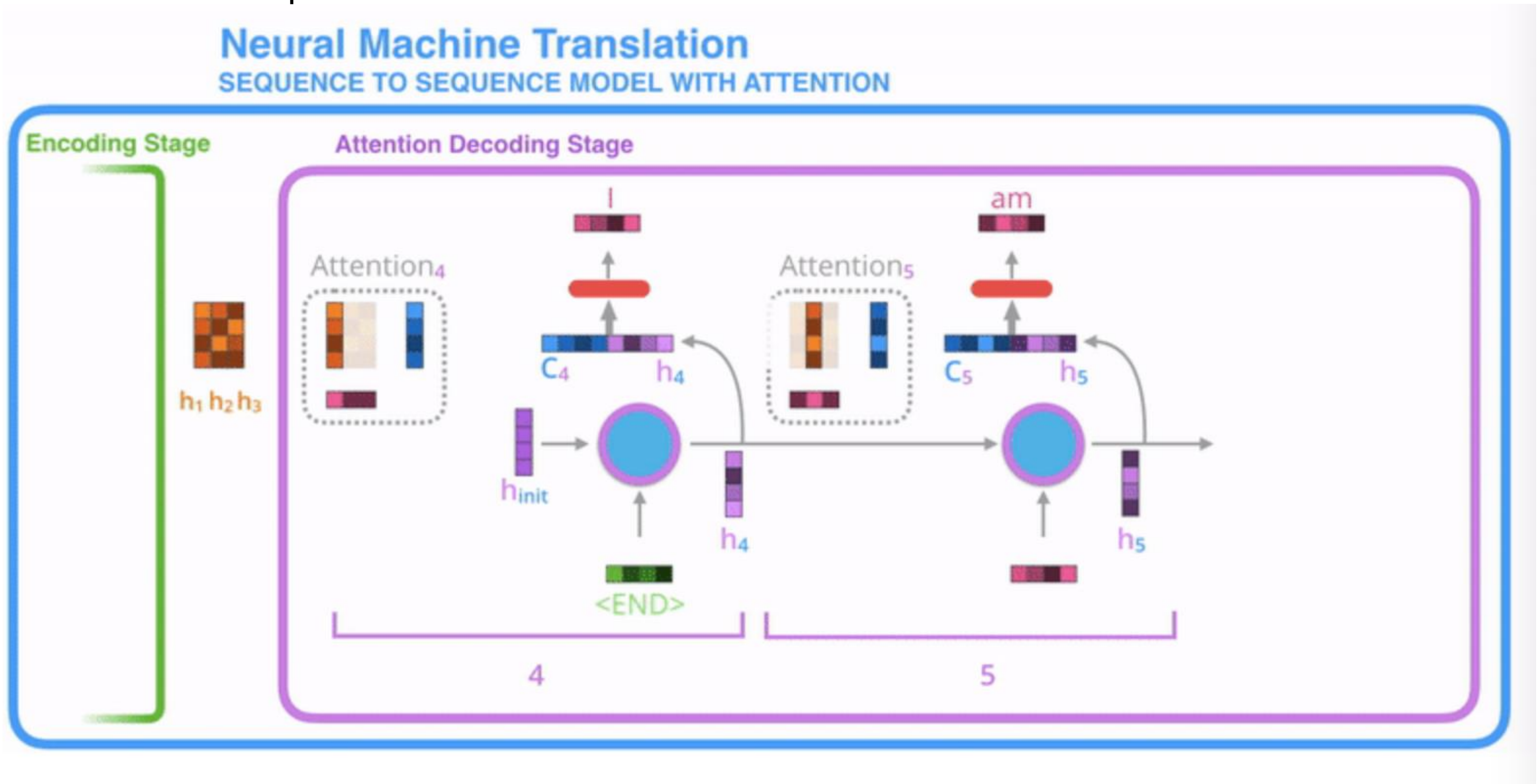
Attention Mechanism

Attention mechanism can be understood as a **weighted approach(softmax)** based on the **importance** of input. Specifically, it sets **different weights** for the **hidden states** generated by **Encoder**.

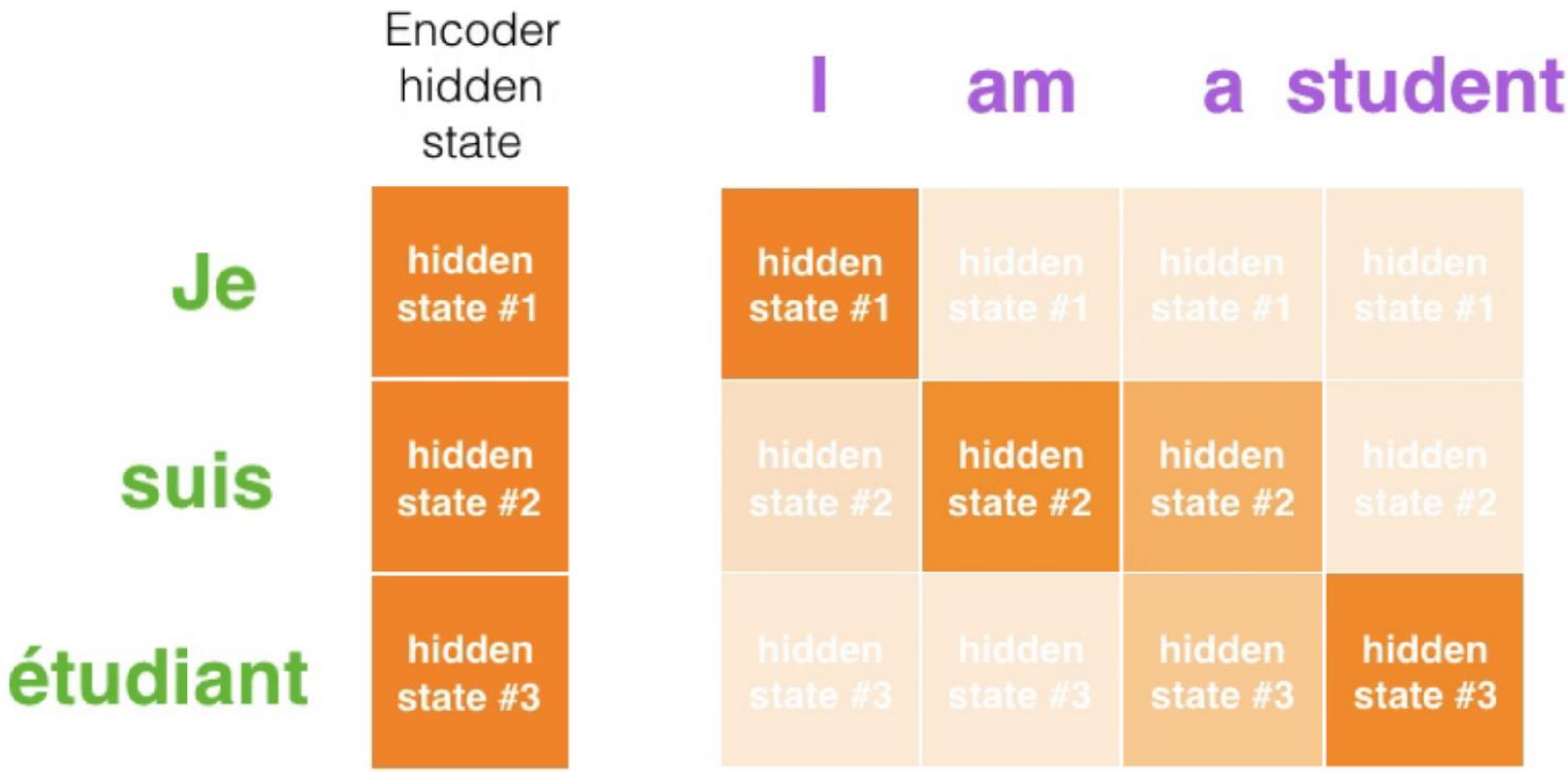


Attention Mechanism

1. RNN Decoder with attention receives **word embedding** and a **init hidden state** as **input**.
2. RNN cell handles input, generates output(discard) and a **new hidden state** (h4).
3. Then **attention** mechanism is applied to **calculate context vector (c4) using Encoder's hidden states and the new hidden state(h4).**
4. **Concatenates** context vector(c4) with the new hidden state (h4).
5. Finished decoder step with the concatenated vector as input using **fully connected layer (softmax).**
6. **Repeats** the above steps for each RNN cell

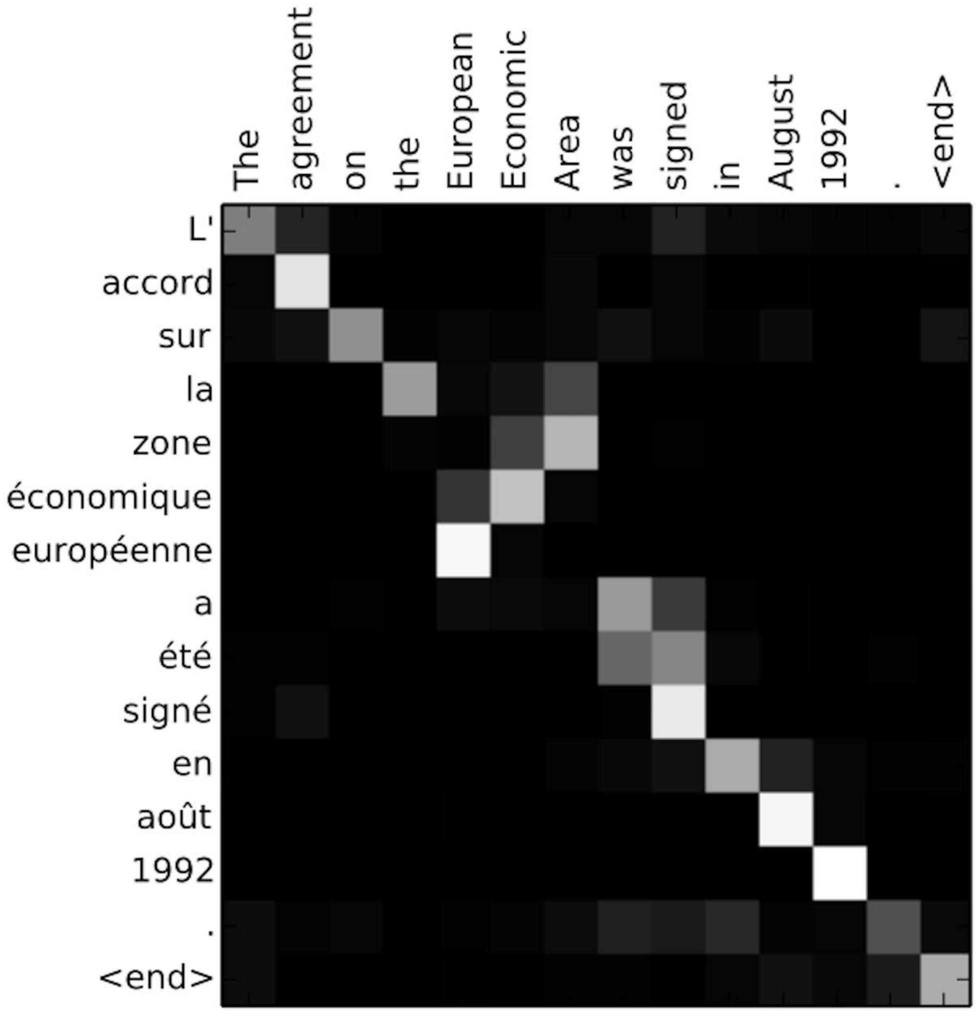


Attention Mechanism



Attention Mechanism

Attention mechanism can used to learn the alignment between the source language and the targeted language



Attention Mechanism Details

Input： $x=(x1,\ldots,xTx)$
Output： $y=(y1,\ldots,yTy)$

(1) $ht=RNNenc(xt,ht-1)$: each RNN cell in Encoder receives **word embedding** of each input word and the **hidden state generated by the previous cell**. The output is the **hidden state of this current cell**.

(2) $st=RNNdec(yt-1\hat{},st-1)$: each RNN cell in Decoder receives **word embedding** of each targeted word and the **hidden state generated by the previous cell**.

(3) $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$: **Context vector** is a **weighted** average of hidden states generated by Encoder.

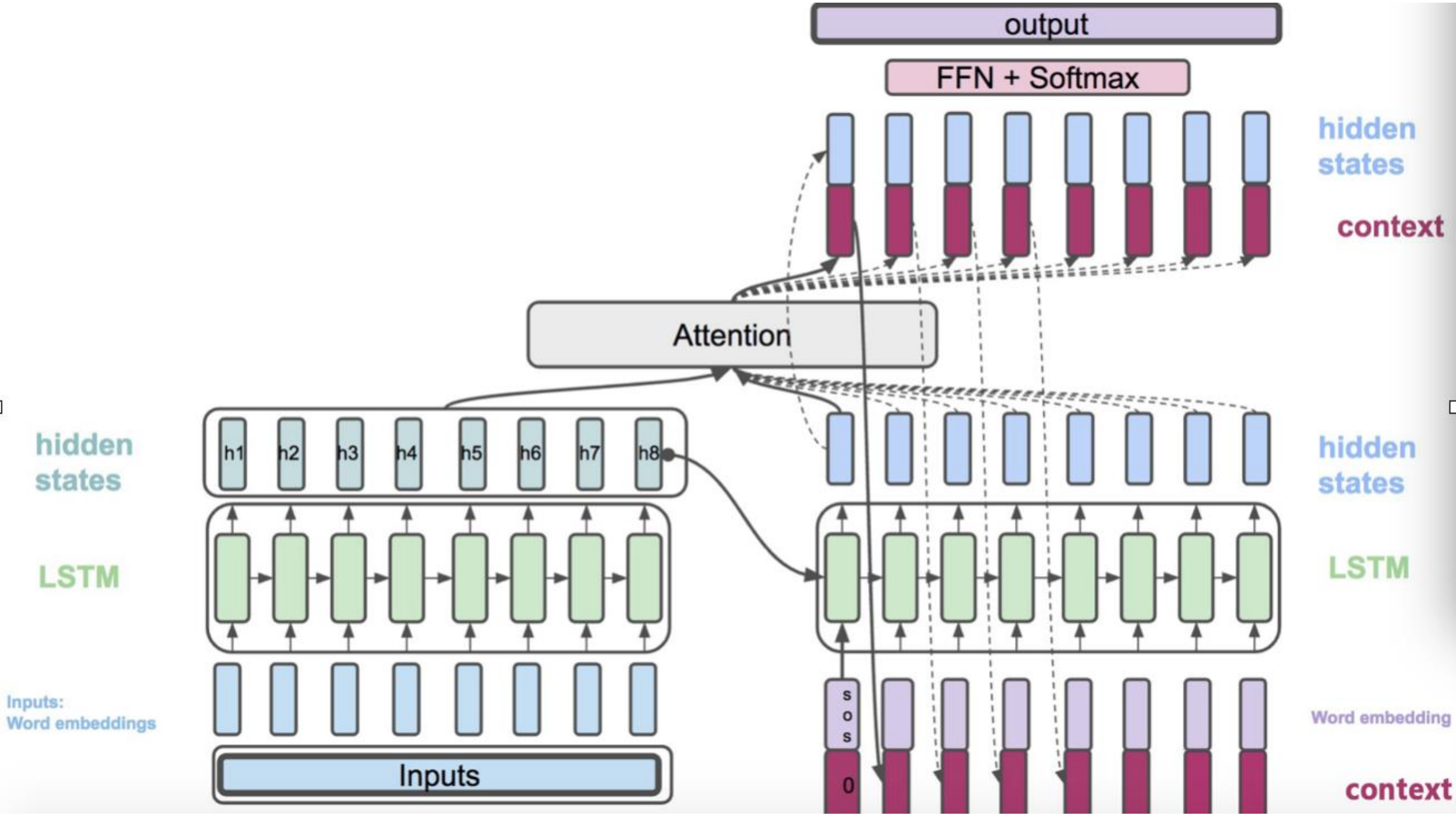
(4) $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$: The **weight** of each hidden state generated by Encoder.

(5) $e_{ij} = score(s_i, h_j)$: The **score** is calculated by **hidden states generated by both Encoder and Decoder.** It is used for the weight calculation in step (4).

(6) $\hat{s_t} = tanh(W_c[c_t; s_t])$: **Concatenates** context vector with the hidden state generated by Decoder.

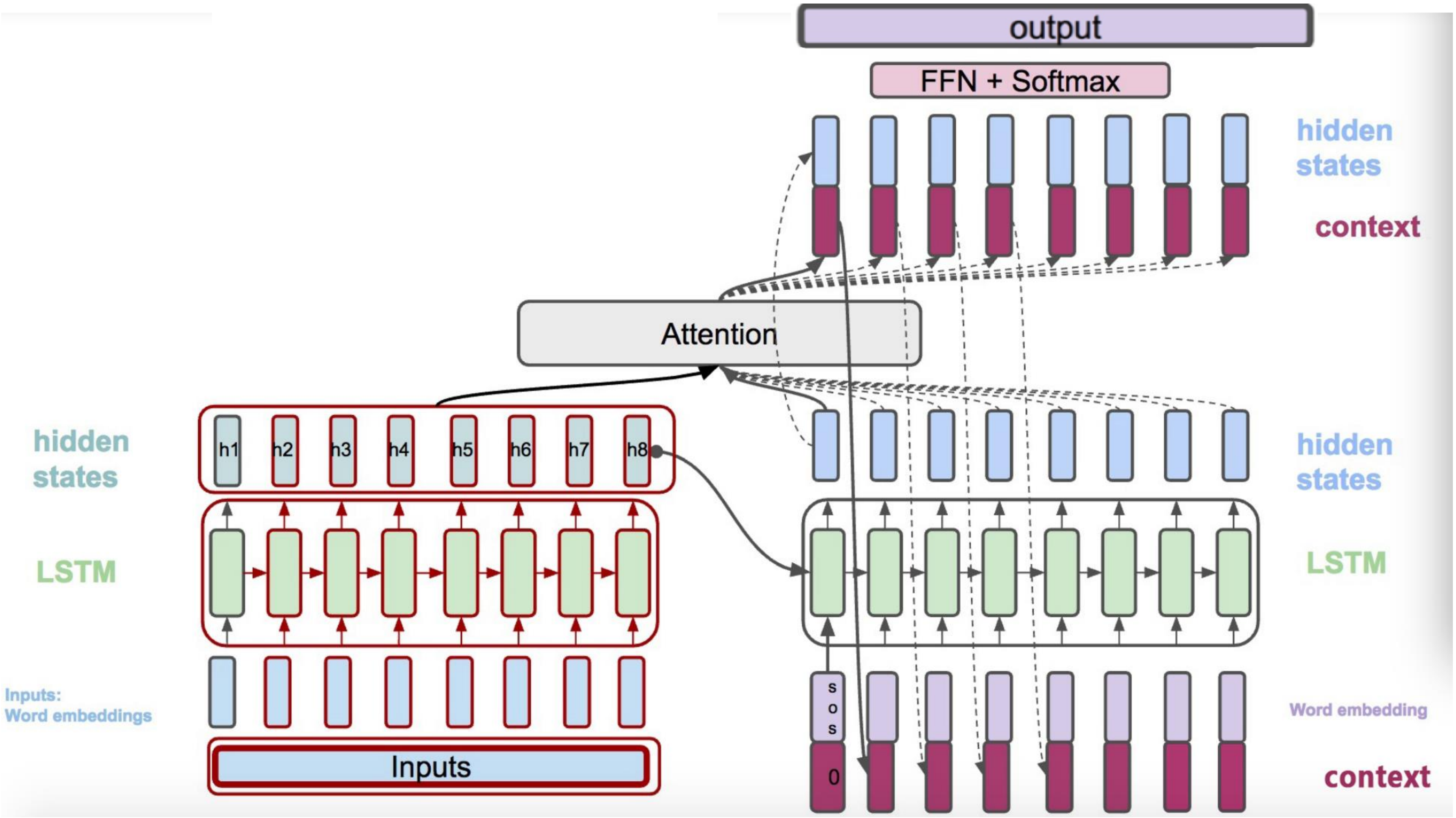
(7) $p(y_t|y_{<t}, x) = softmax(W_s \hat{s_t})$: Calculates the final output probability of each word in dictionary.

Attention Mechanism Details

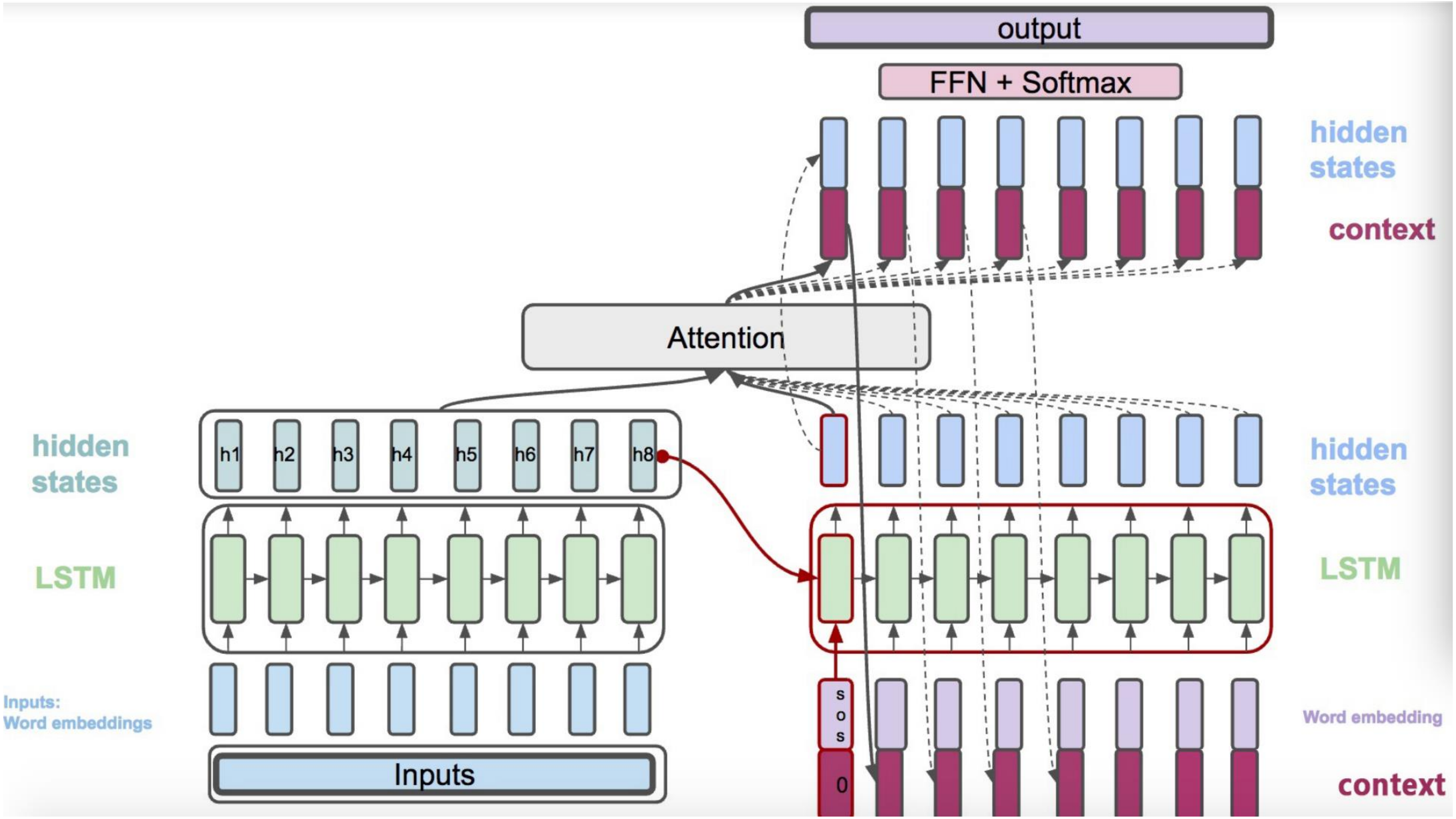


Attention Mechanism Details

1) $ht = RNNenc(xt, ht-1)$: each RNN cell in Encoder receives **word embedding** of each input word and the **hidden state generated by the previous cell**. The output is the **hidden state of this current cell**.



(2) $st=RNNdec(yt-1\hat{},st-1)$: each RNN cell in Decoder receives **word embedding** of each targeted word and the **hidden state generated by the previous cell**.

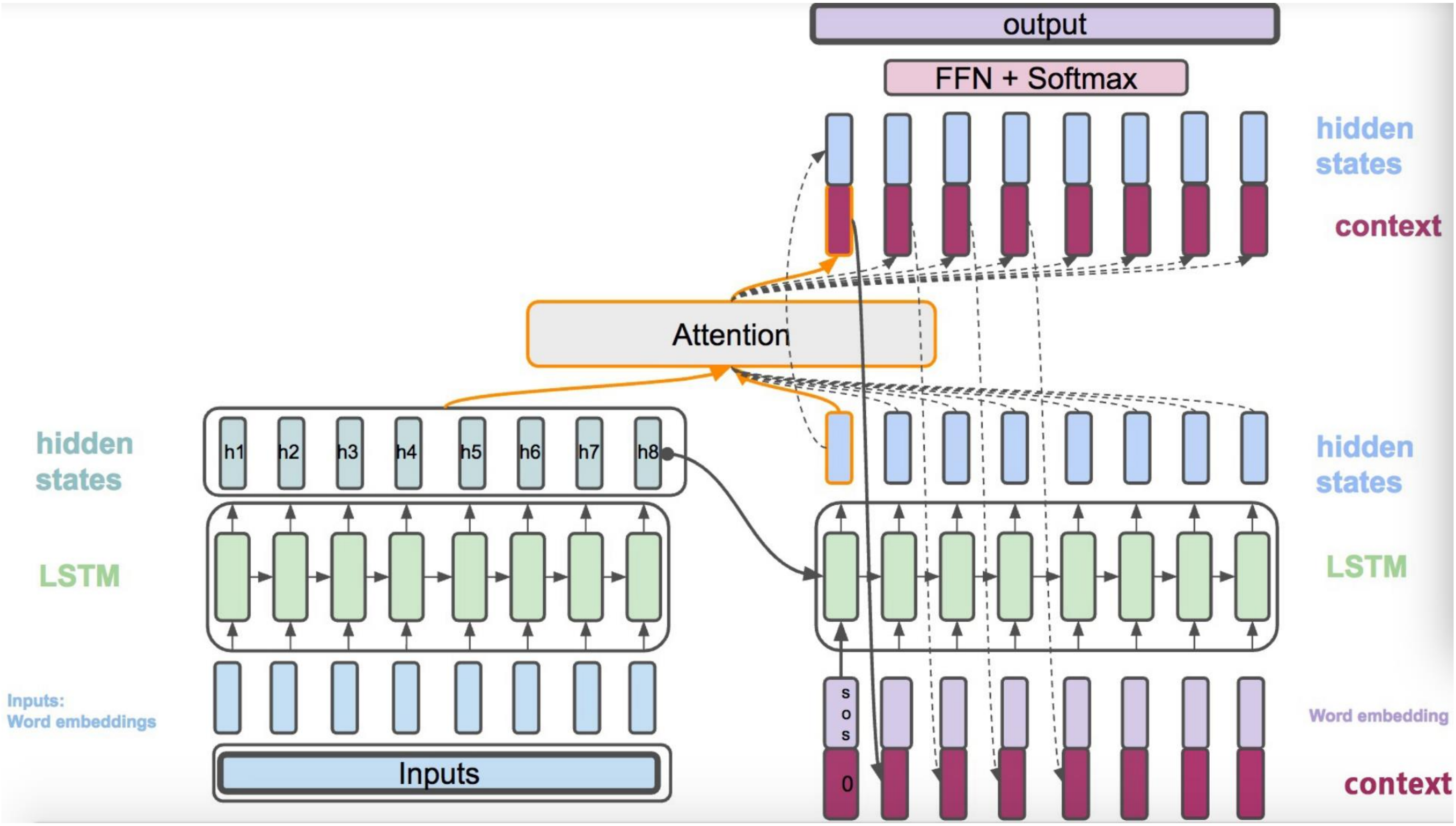


Attention Mechanism Details

(3) $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$: **Context vector** is a **weighted** average of hidden states generated by Encoder.

(4) $\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{Tx} exp(e_{ik})}$: The **weight** of each hidden state generated by Encoder.

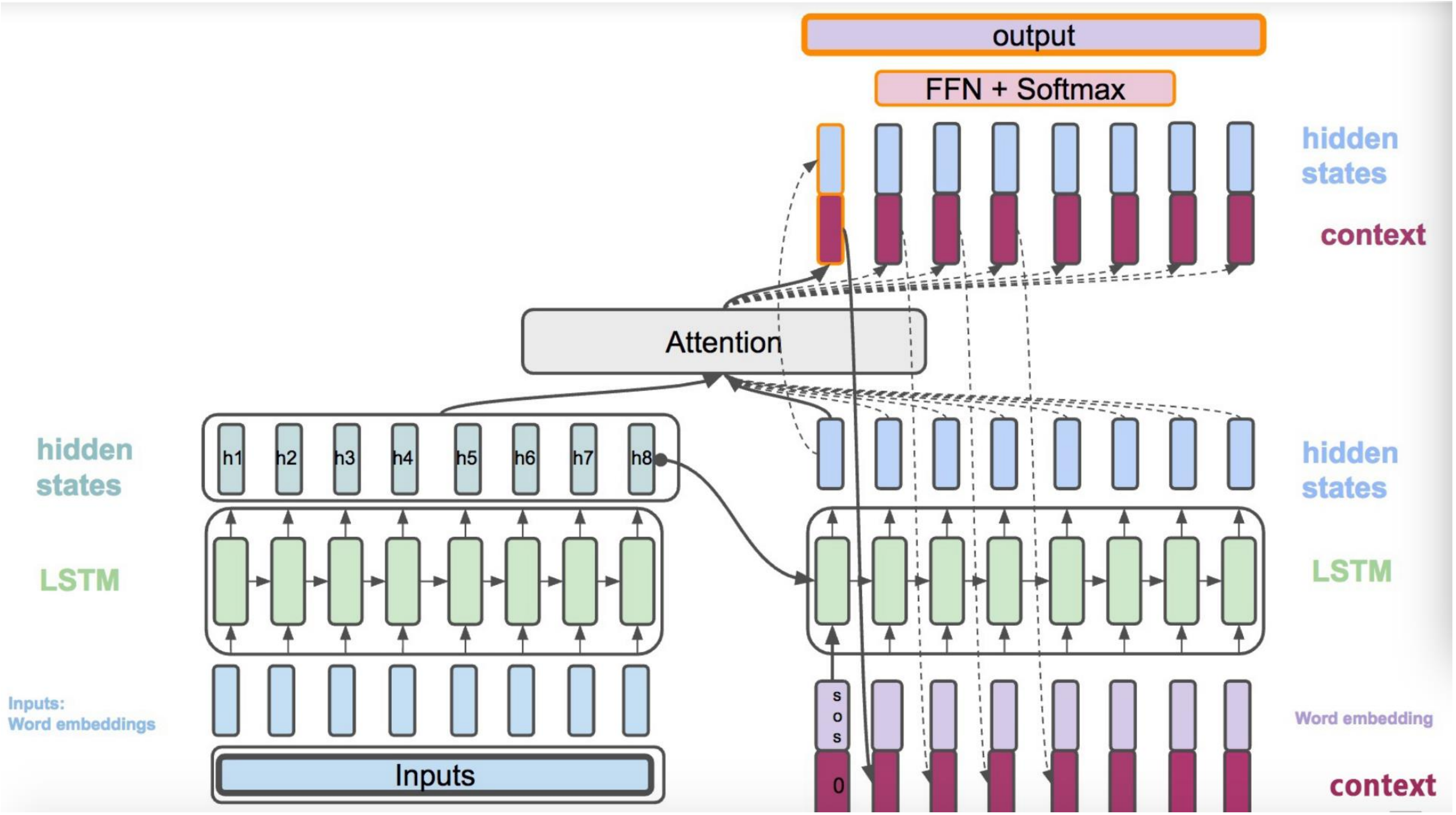
(5) $e_{ij} = score(s_i, h_j)$: The **score** is calculated by **hidden states generated by both Encoder and Decoder.** It is used for the weight calculation in step (4).



Attention Mechanism Details

(6) $\hat{s}_t = tanh(W_c[c_t; s_t])$: **Concatenates** context vector with the hidden state generated by Decoder.

(7) $p(y_t|y_{<t}, x) = softmax(W_s\hat{s}_t)$: Calculates the final output probability of each word in dictionary.



Attention Mechanism: Score Calculation

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s] & \textit{concat} \end{cases}$$

Attention Mechanism: Score Calculation

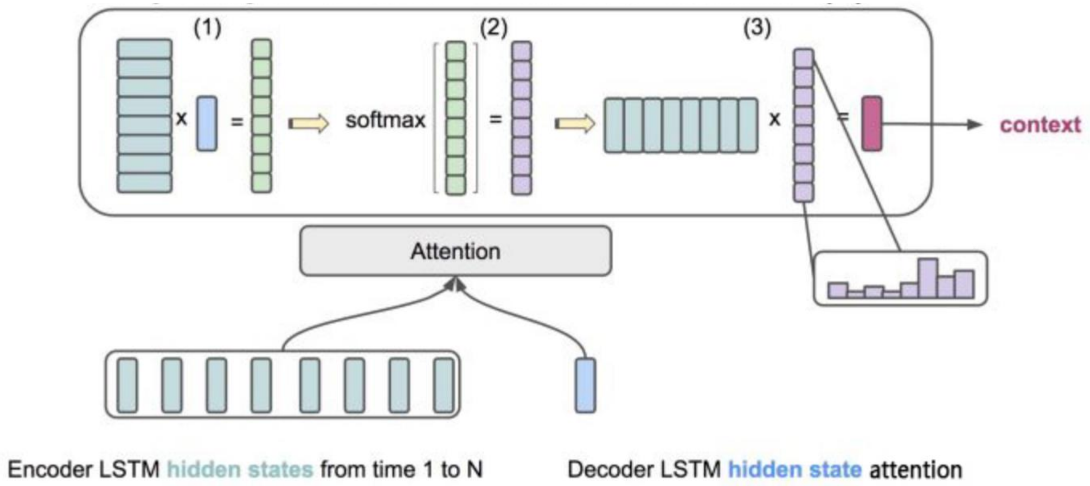
Approach 1
Input: The dimension of matrix represents **all hidden states generated by Encoder** (H):
(hidden dimension, input sequence length)

The dimension of **hidden state generated by decoder at any time point** (S):
(hidden dimension, 1)

1) Get score matrix (sequence length, 1) by transposing H to (sequence length, hidden dimension) and conducting **dot** with S

2) Applying **softmax** to these scores to normalize the sum of weighted scores = 1

3) Generating Context Vector by conducting Dot to H and the weighted scores of step (2)



Sequence to Sequence Model Use Cases

- Chatbot
- Machine Translation (Github: https://github.com/tensorflow/nmt)
- Auto Summarization