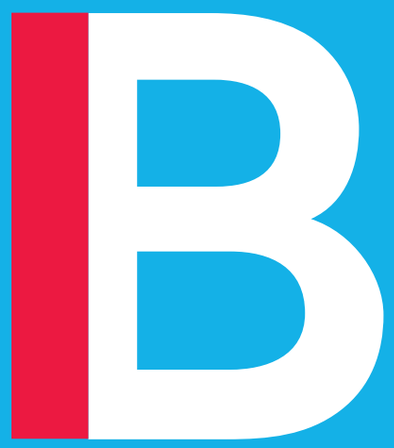
Imperial College Business School

Imperial means Intelligent Business

Machine Learning in Finance

Lecture 8

RNN Applications and Attention Mechanisms



Arnaud de Servigny & Hachem Madmoun

Outline:

- The Sentiment Analysis Pipeline
- The Various Applications of RNNs
- The Sequence to Sequence Framework
- Introducing the Attention Mechanism

Part 1 : The Sentiment Analysis Pipeline

The Embedding Layer

- The **Embedding Layer** takes as input the sequences of integers. But all the sequences should be of the same length T, so that we can pack them into the same tensor :
  - Sequences that are shorter than T are padded with zeros.
  - Sequences that are longer that T are truncated.

Row Data

- Document 1 : « There were no wolves in the movie. »
- Document 2 : « This movie has one star and that star is Ryan Gosling. Great flick, highly recommend it. »

⋮ ⋮

- Document N : « How many times must Willy be freed before he's freed?. »

Preprocessed Data

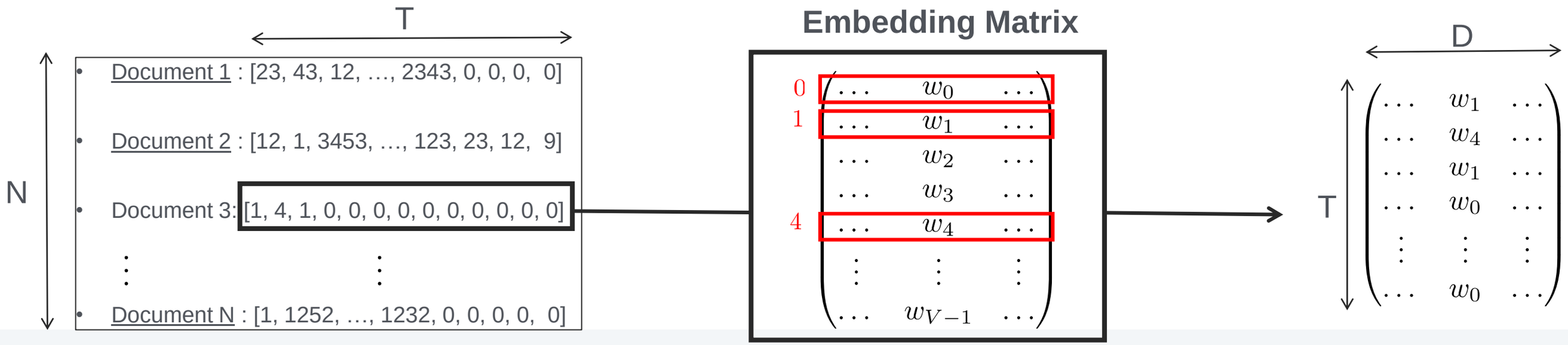
- Document 1 : [23, 43, 12, …, 2343, 0, 0, 0, 0]
- Document 2 : [12, 1, 3453, …, 123, 23, 12, 9]

⋮ ⋮

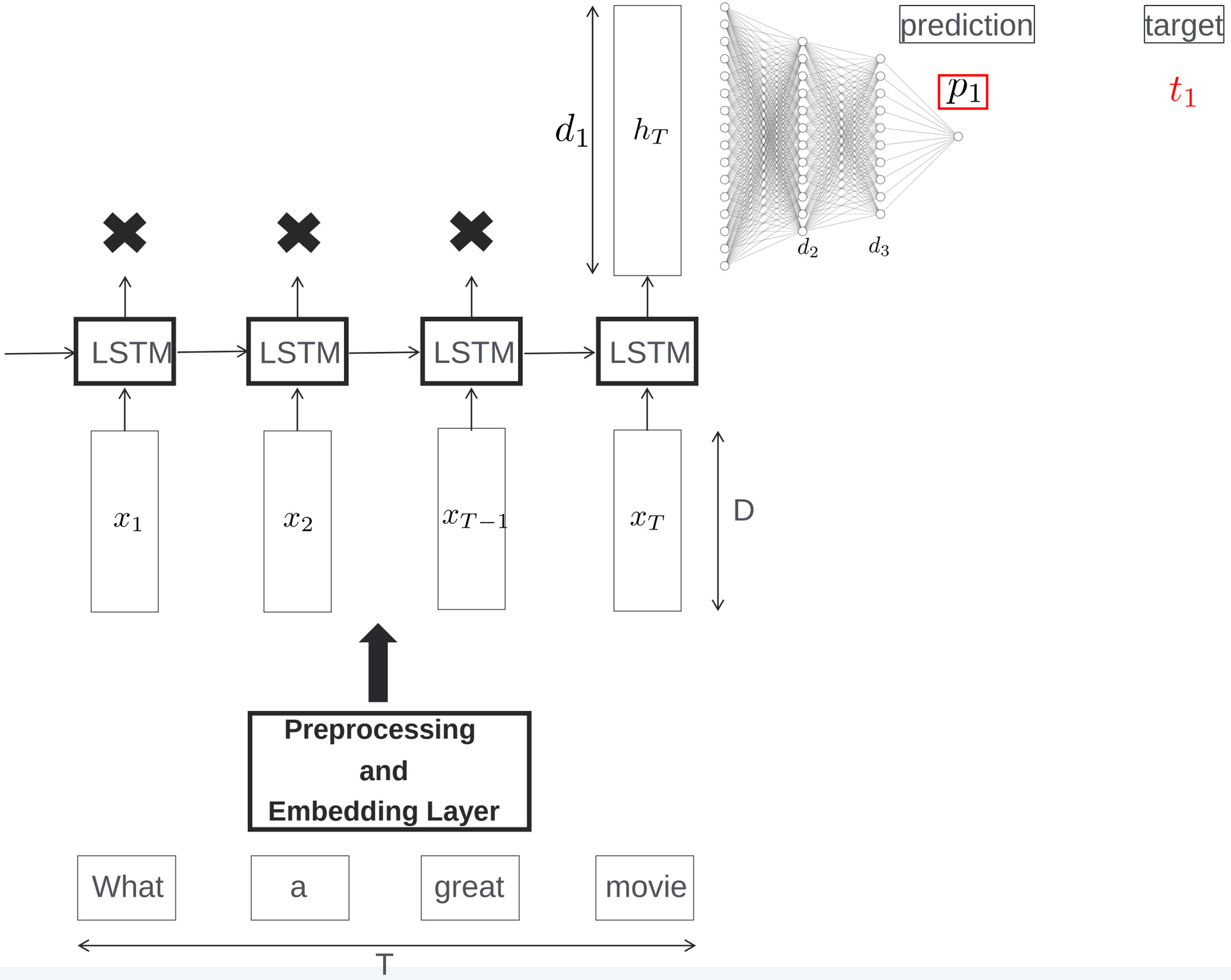
- Document N : [1, 1252, …, 1232, 0, 0, 0, 0, 0]

2D tensor of integers, of shape (N, T)

- The Embedding Layer transforms the 2-dim input tensor of shape (N, T) into a tensor of shape (N, T, D).

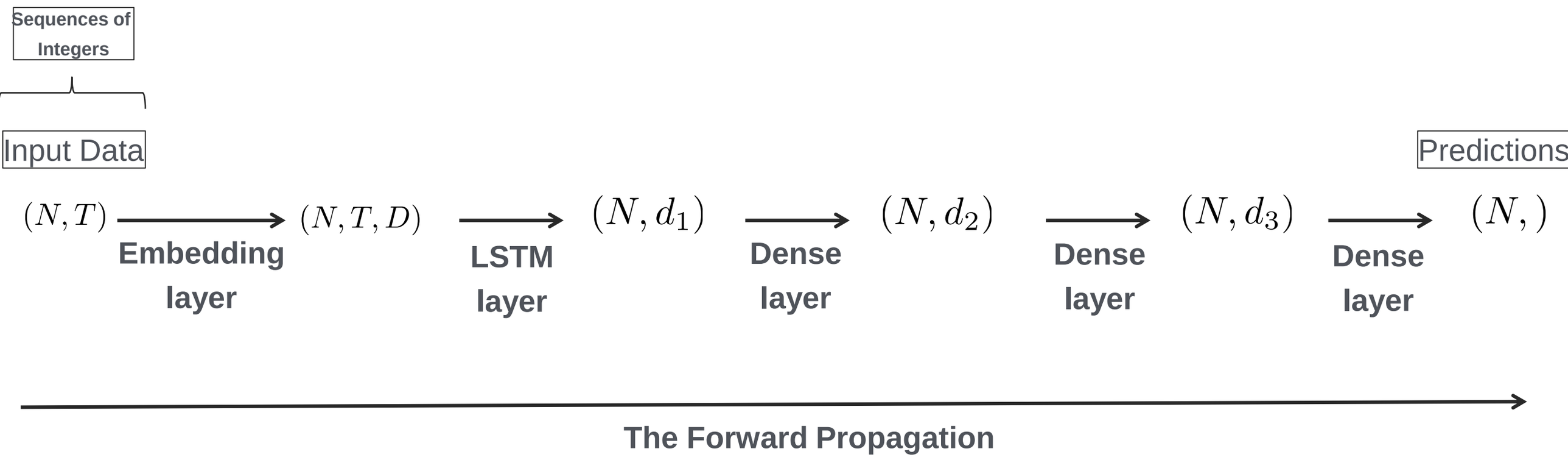


The Sentiment Analysis Pipeline – Part 1 –



The Sentiment Analysis Pipeline – Part 2 –

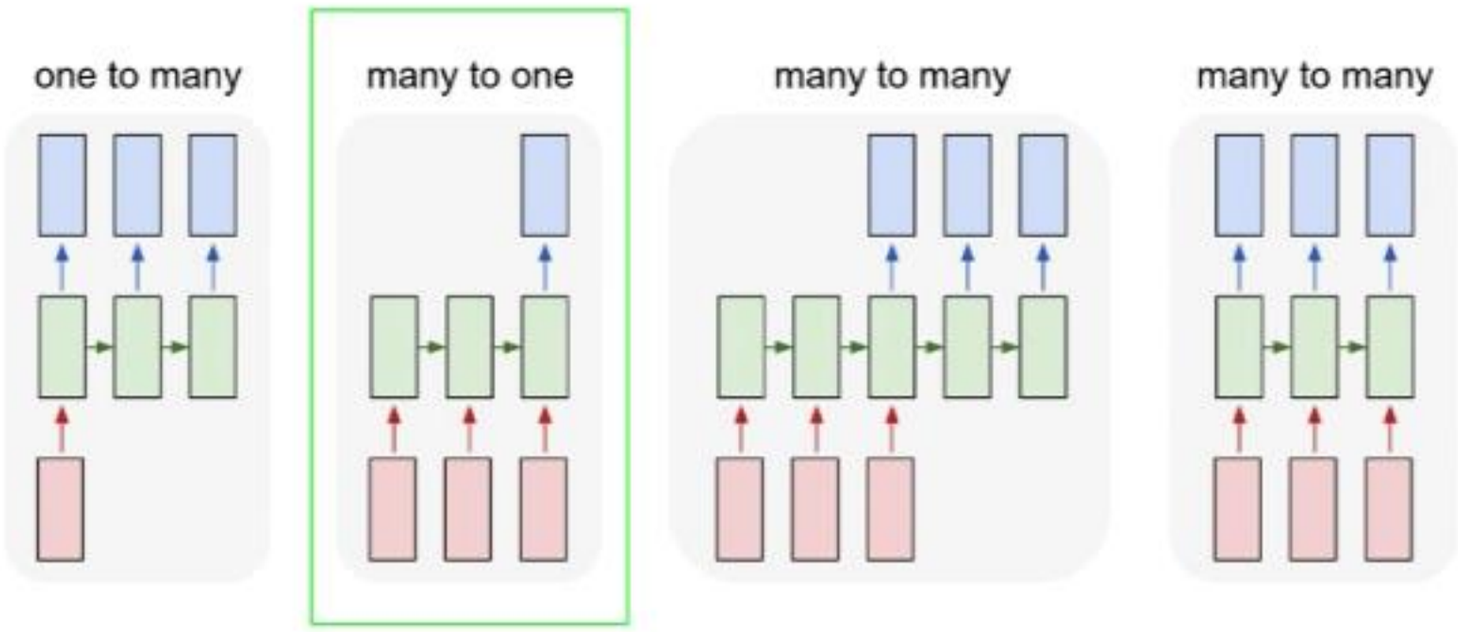
- Let’s keep track of the evolution of the tensor shape after each layer transformation:



Part 2 : The Various Applications of RNNs

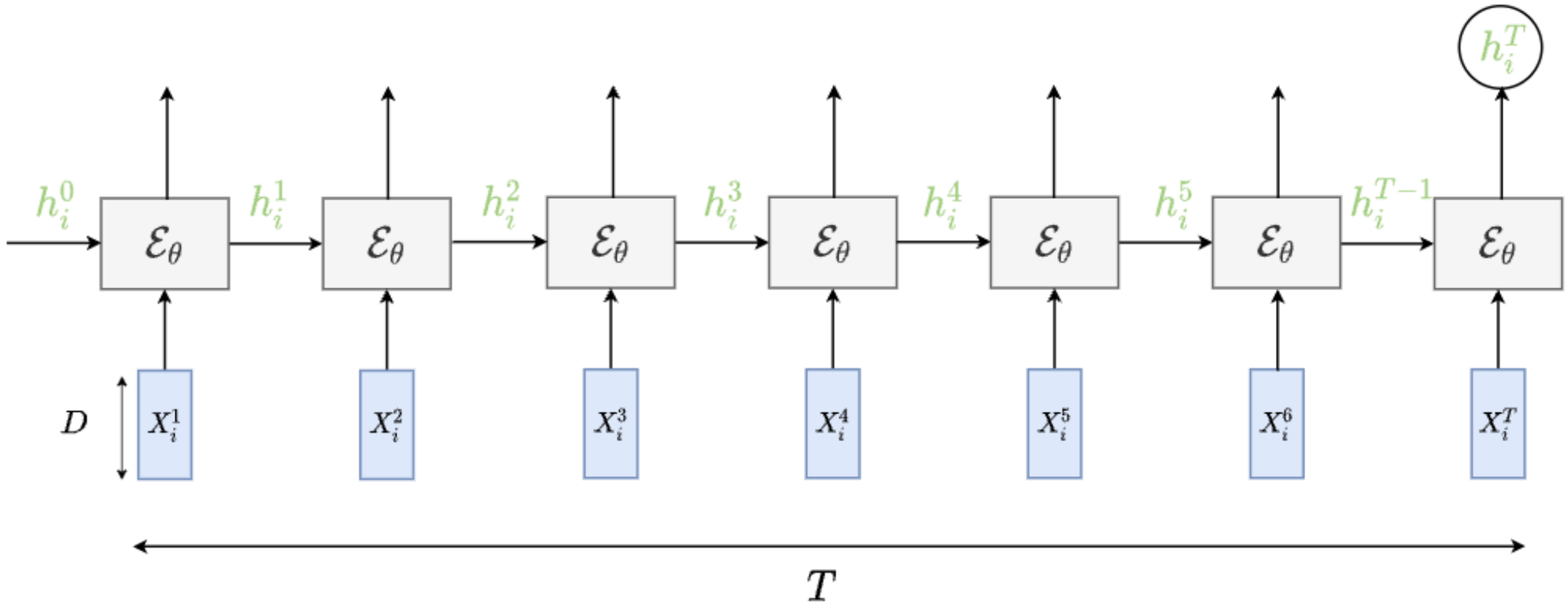
The Various Applications of RNNs

- There are principally 4 types of applications to Recurrent Neural Networks.
    - **One to Many:** Mapping a vector to a sequence of vectors.
    - **Many to One:** Mapping a sequence of vectors to one vector.
    - **Many to Many:**
        - Aligned case: Mapping a sequence to another sequence of the same length T
        - Unaligned case: Mapping a sequence of length T_x into another sequence of length T_y $(\text{with } T_x \neq T_y)$



The Many to One problem – The architecture –

- In the Many to One framework, the objective is to map a sequence $(X_i^1, \ldots, X_i^T) \in \mathbb{R}^{T \times D}$ into a vector $\boldsymbol{h}_i^T \in \mathbb{R}^d$ using the LSTM layer $\mathcal{E}_\theta$ parameterized by θ



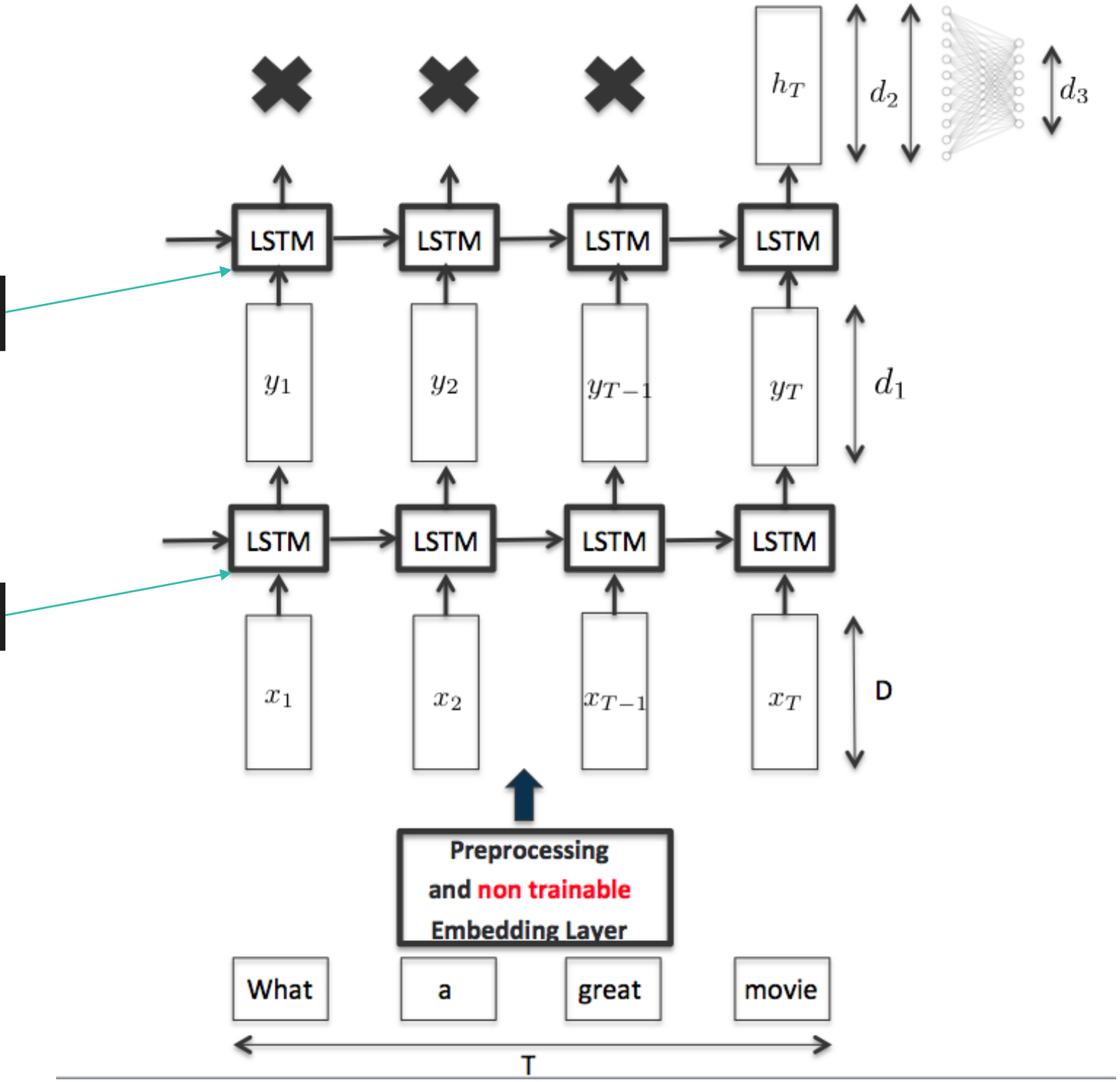
- So far, we have only discussed models that are part of the Many to One framework.
  - Sentiment Analysis (Lecture 6).
  - News Classification (programming session 7).
- Let us consider some examples in the next slides.

Stacking LSTM layers for a Multiclass classification Problem

```
from tensorflow.keras.layers import LSTM
```

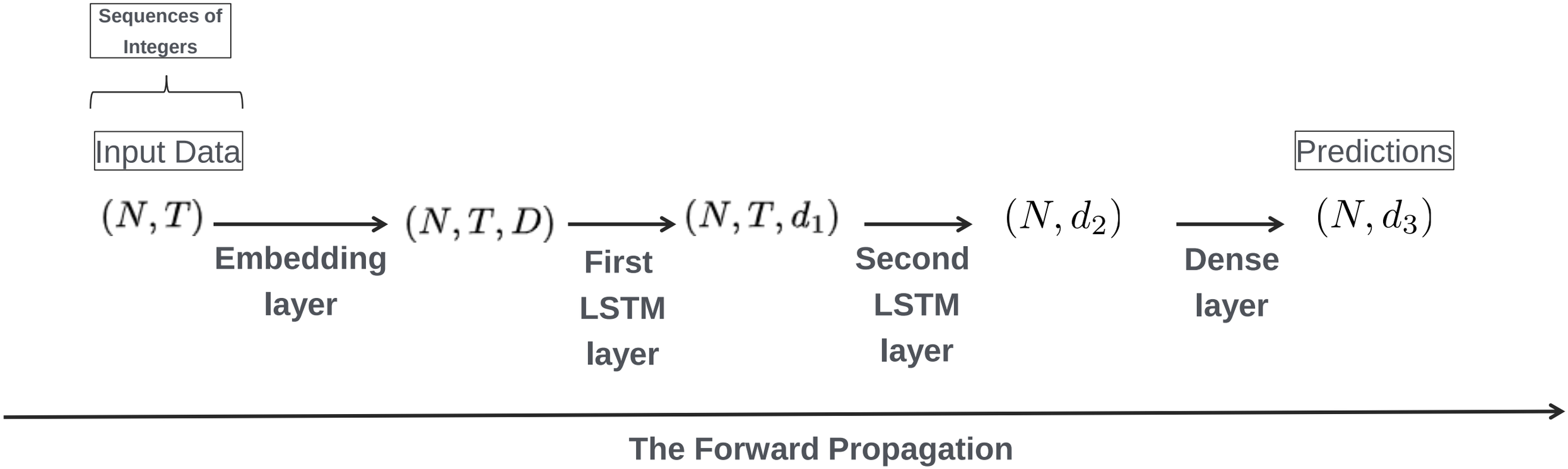
```
lstm_2 = LSTM(d_2, return_sequences = False)
```

```
lstm_1 = LSTM(d_1, return_sequences = True)
```

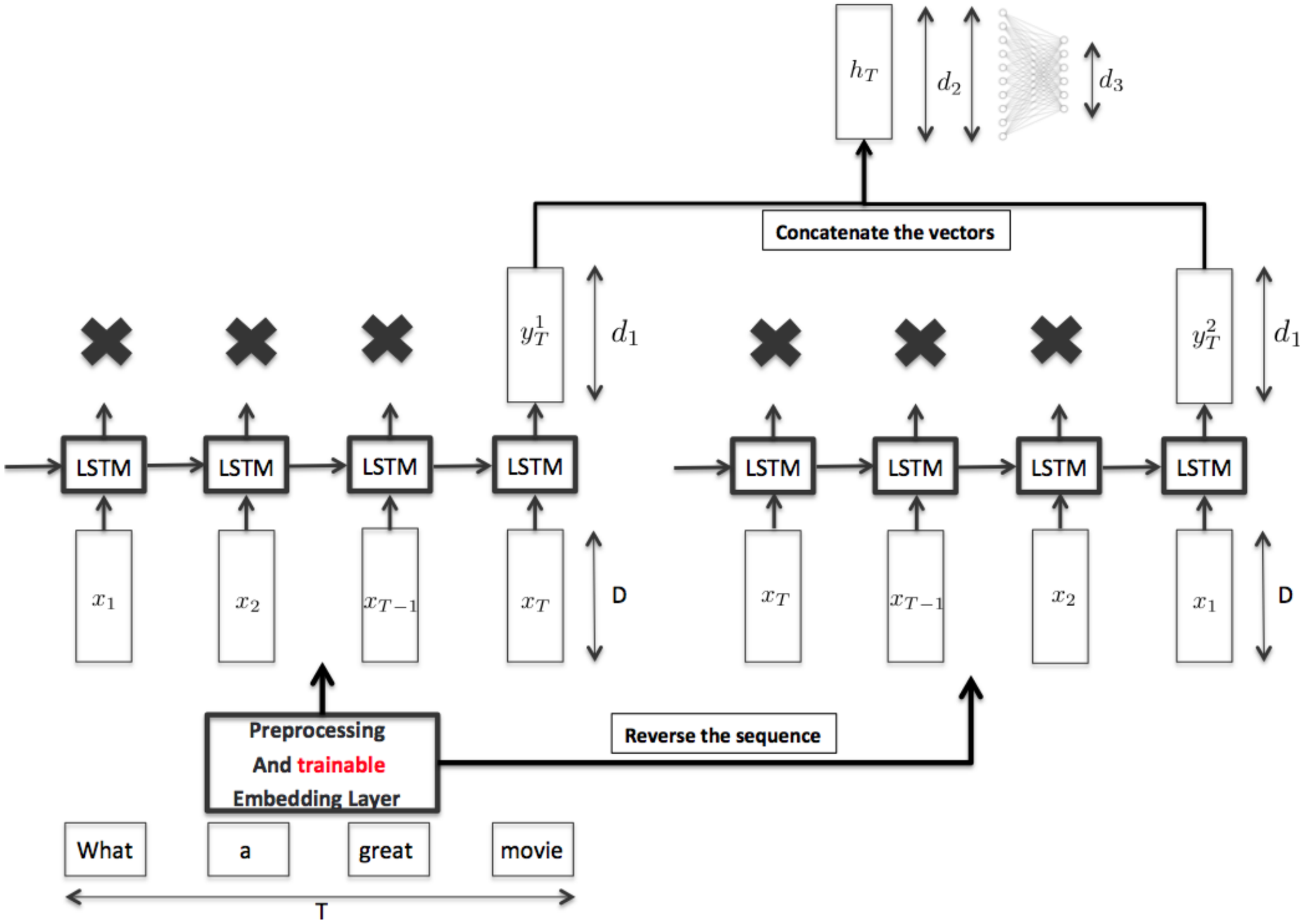


Stacking LSTM layers for a Multiclass classification Problem

- Let’s keep track of the evolution of the tensor shape after each layer transformation:

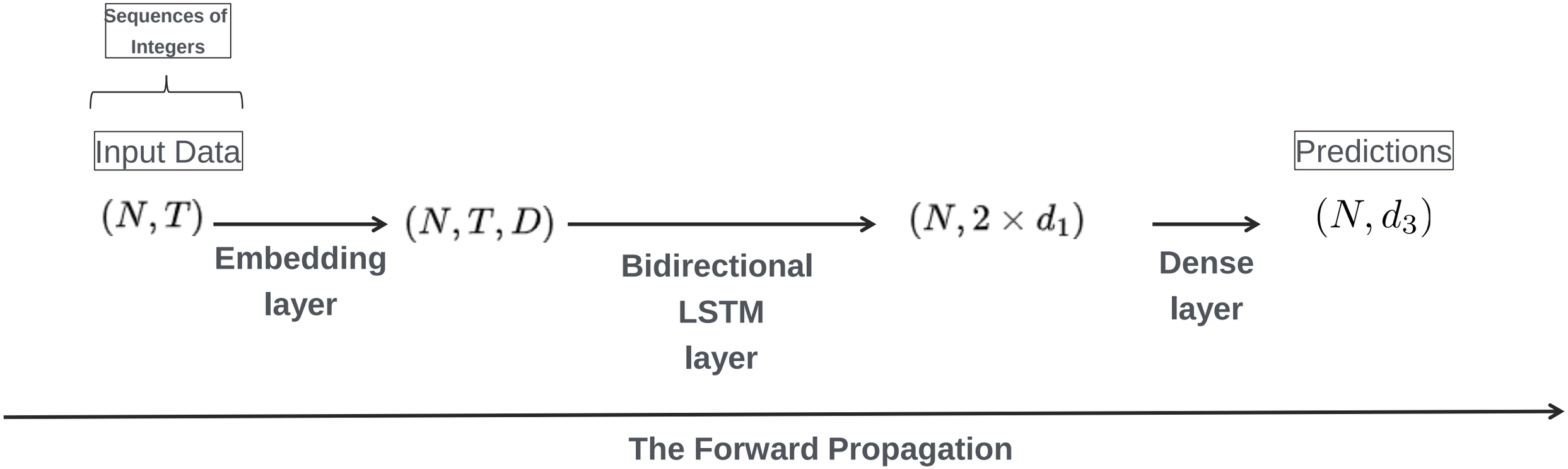


Bidirectional LSTM for a Multiclass classification Problem



Bidirectional LSTM for a Multiclass classification Problem

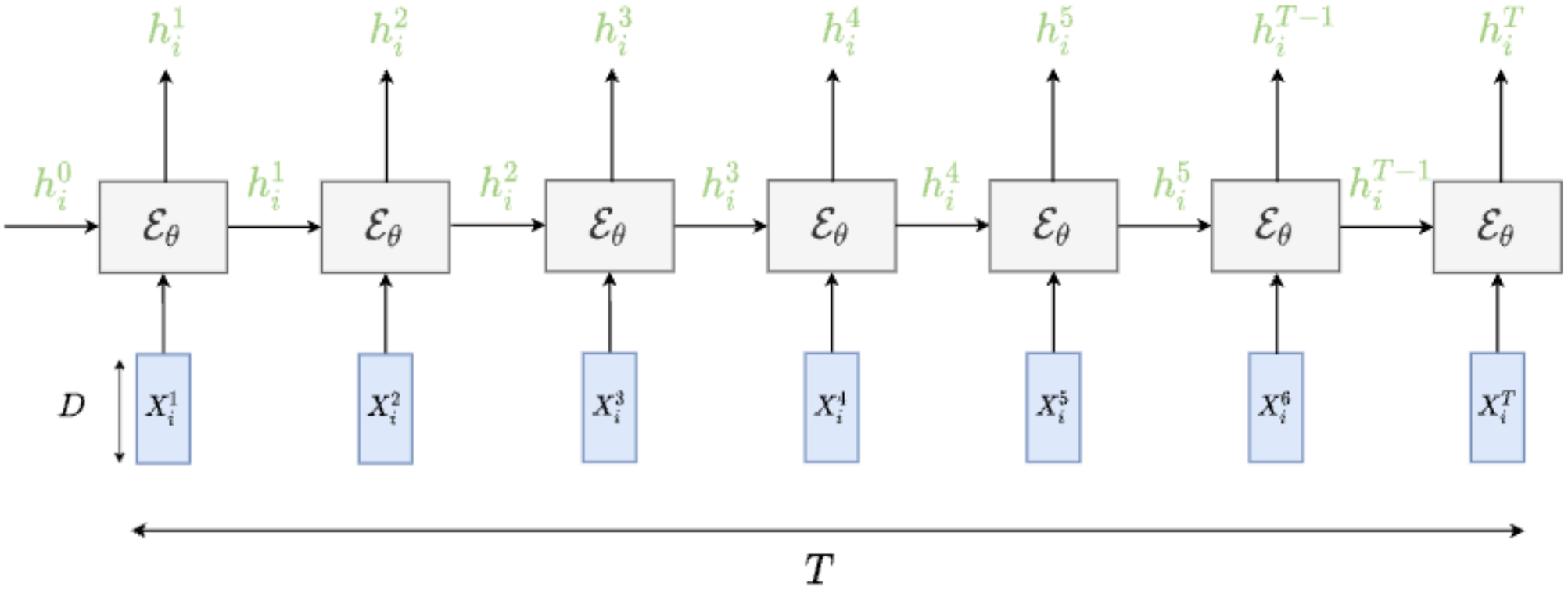
- Let’s keep track of the evolution of the tensor shape after each layer transformation:



The Many to Many Problem (Aligned case) – The Architecture –

- In the Many to Many framework, the objective is to map a sequence $(X_i^1, \ldots, X_i^T) \in \mathbb{R}^{T \times D}$ into a sequence $(h_i^1, \ldots, h_i^T) \in \mathbb{R}^{T \times d}$ using the LSTM layer $\mathcal{E}_\theta$ parameterized by θ

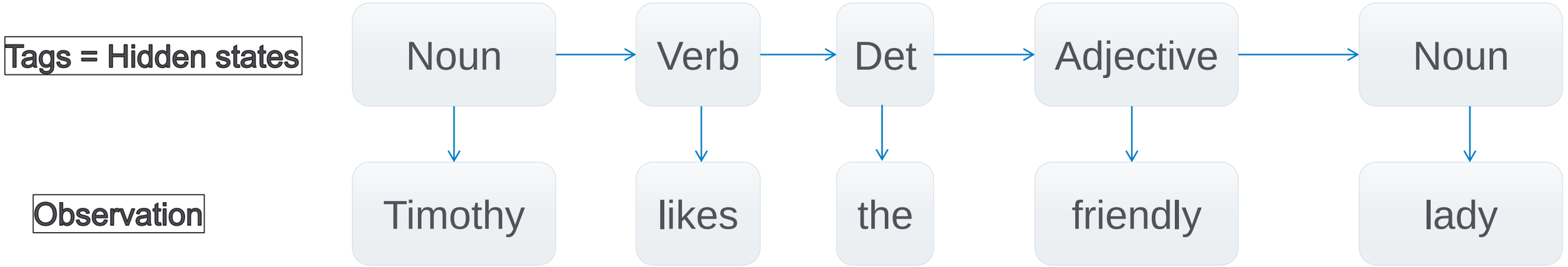
- We are considering the **aligned case** where the input and the output sequences are of the same length T



The Many to Many Problem (Aligned case) – an Example –

- POS (Part Of Speech) Tagging is a typical example, where the objective is to tag each word of a sentence with its "Part-of-Speech" tag.

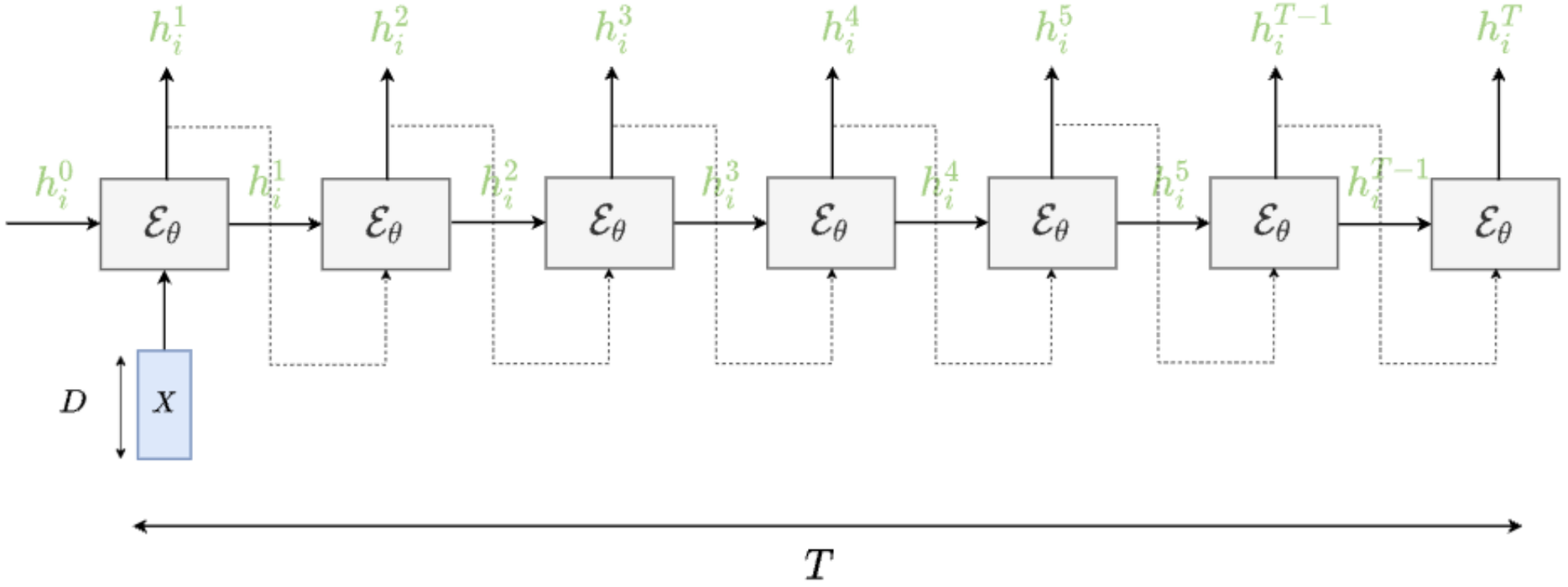
- Another popular model can be used for POS tagging: The Hidden Markov Model (HMM).



(See the Optional Reading) for more details about the HMM

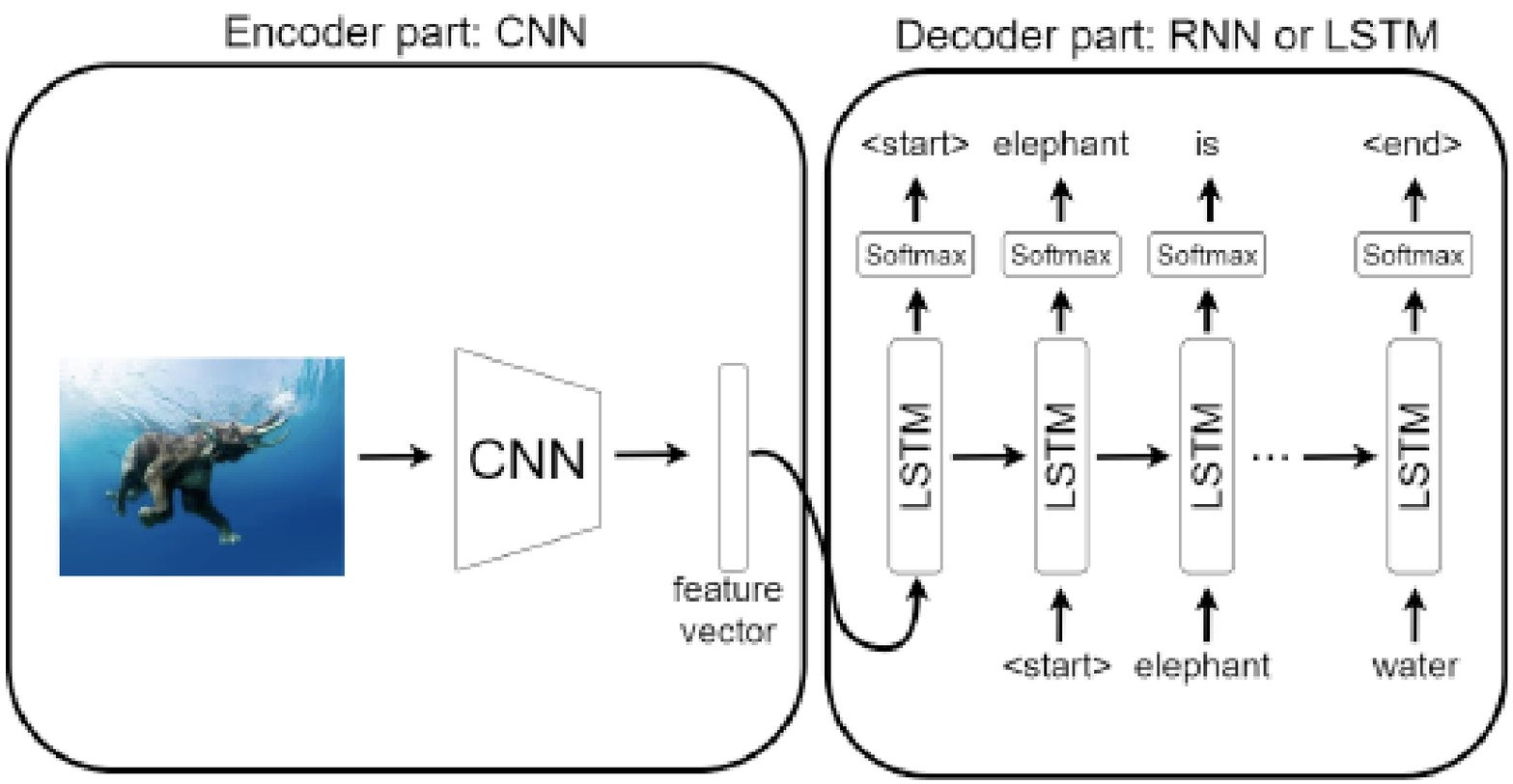
The One to Many Problem – The Architecture –

- In the One to Many framework, the objective is to map a vector $X \in \mathbb{R}^D$ into a sequence $(h_i^1, \ldots, h_i^T) \in \mathbb{R}^{T \times d}$ using the LSTM layer $\mathcal{E}_\theta$ parameterized by θ
- The vector $X \in \mathbb{R}^D$ is typically the output of an encoder layer processing an image or another sequence for instance.
- At each step of the generation process, the output h_i^t is fed back into the model to get the new hidden state h_i^{t+1}



The One to Many Problem – an Example –

- **Image captioning** is a typical example, where the description of an image is generated.
- An image is mapped into a **feature vector**, which in turn becomes the input for an LSTM architecture.



Interactive Session

Part 3 : The Sequence to Sequence Framework

The Sequence to Sequence Framework –The architecture –

- For Many to Many applications, the LSTM models can only be applied in the aligned case (i.e, if the input and the output sequences are of the same length).

- However, if we want to learn a mapping from a sequence of input vectors of length T_x into a sequence of output vectors of length T_y (where $T_x \neq T_y$), we need to introduce a new framework, composed of two steps.

  - An encoder $\mathcal{E}_{\theta_e}$ maps the input sequence $(X_i^1, \ldots, X_i^{T_x}) \in \mathbb{R}^{T_x \times D_x}$ into the final hidden state $h_i^{T_x}$

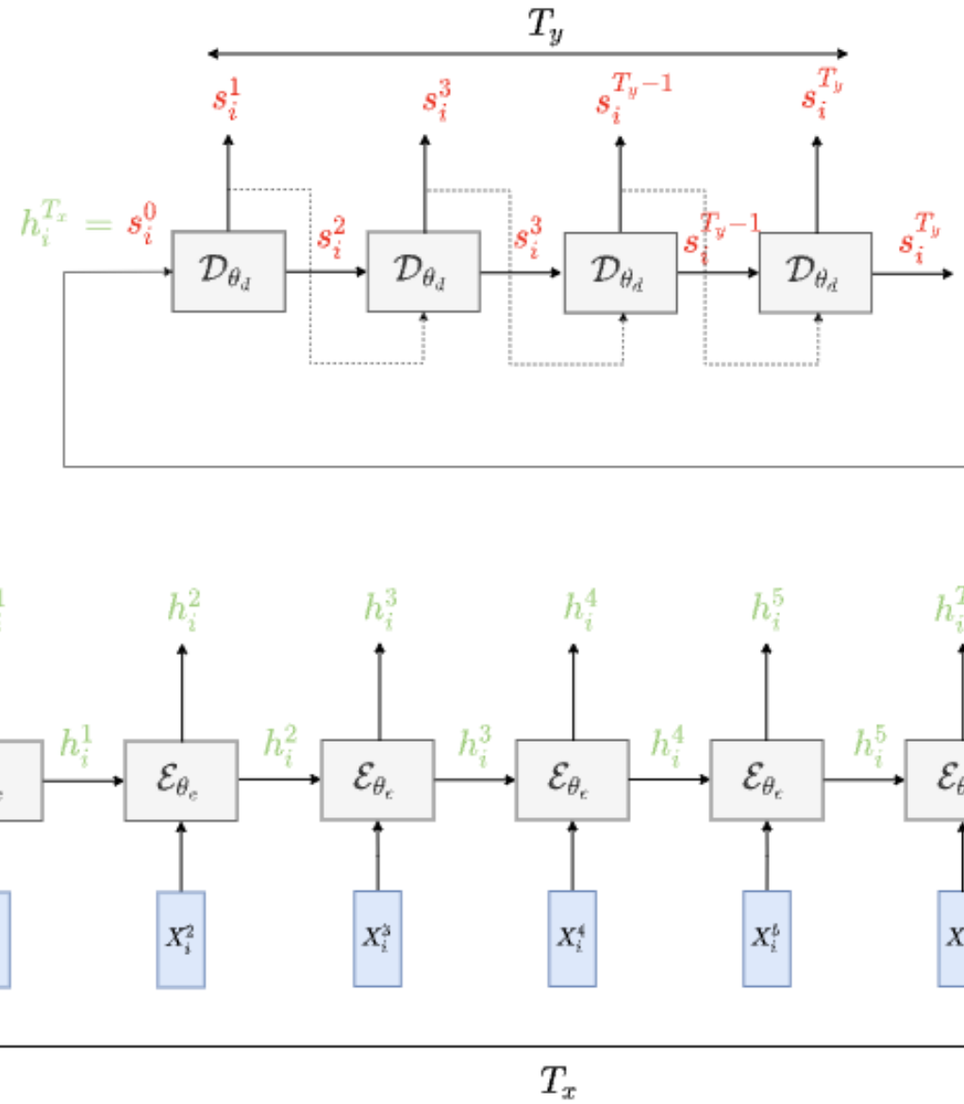
  - A decoder $\mathcal{D}_{\theta_d}$ is initialized with the final encoder hidden state:

$$h_i^{T_x} = s_i^0$$

.

- We can then generate the sequence of hidden states associated with the decoder

$$(s_i^1, \ldots, s_i^{T_y})$$

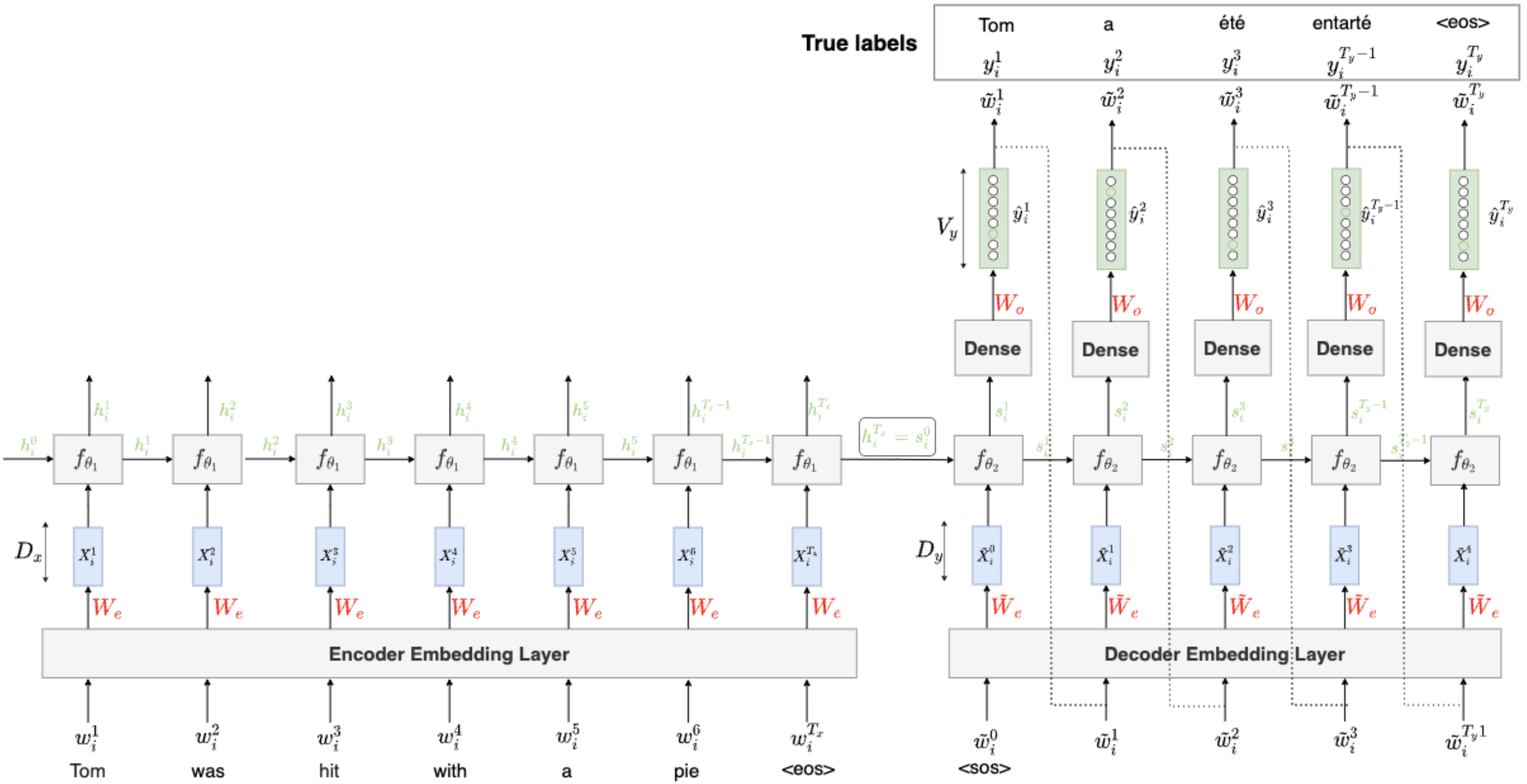


The Sequence to Sequence Framework – an Example –

- A Typical example for the Sequence to Sequence Framework is Neural Machine Translation (NMT).
- We usually use **Teacher Forcing** during the training process.

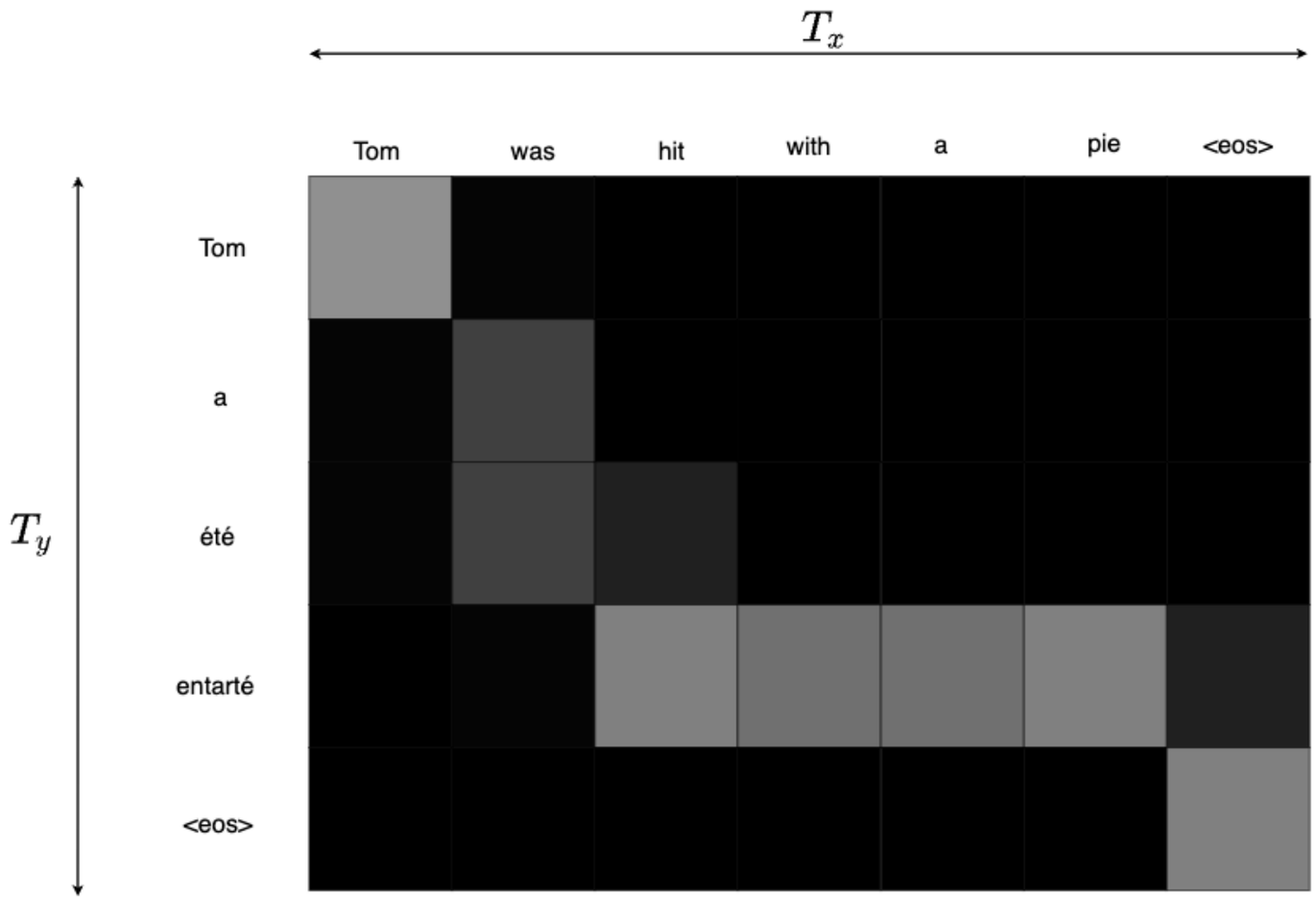
The Sequence to Sequence Framework – an Example –

- During the prediction phase, at each iteration, the decoder output is fed back into the model.



Limitations of the Sequence to Sequence Framework

- There are two main challenges with the sequence to sequence framework using RNNs:
    - First, by feeding a single fixed length vector to the decoder, the encoder has to compress all the input information in few dimensions, which leads to a loss of information.
    - This architecture doesn't allow model alignment between the input and the output sequences.
- We would like each output sequence to selectively focus on relevant parts of the input sequence.



Part 4 : Introducing the Attention Mechanism

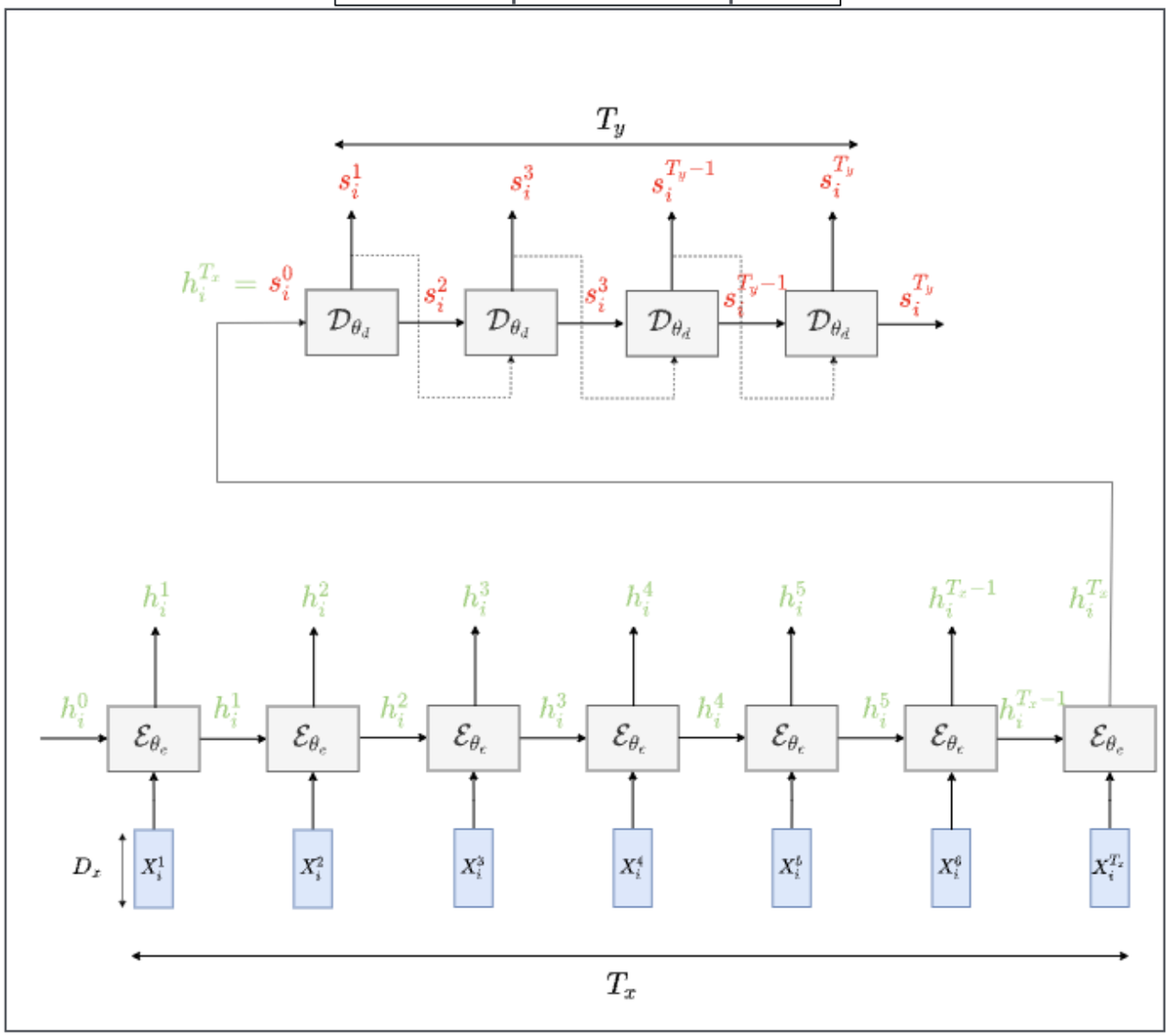
Sequence to Sequence with Attention Mechanisms

- The vanilla Sequence to Sequence model has to boil the entire input sequence into a single vector.
- At each decoder time step $t_y \in \{1, \ldots, T_y\}$, we would like the input vector to be: $c_i^{t_y} = \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} h_i^{t_x}$

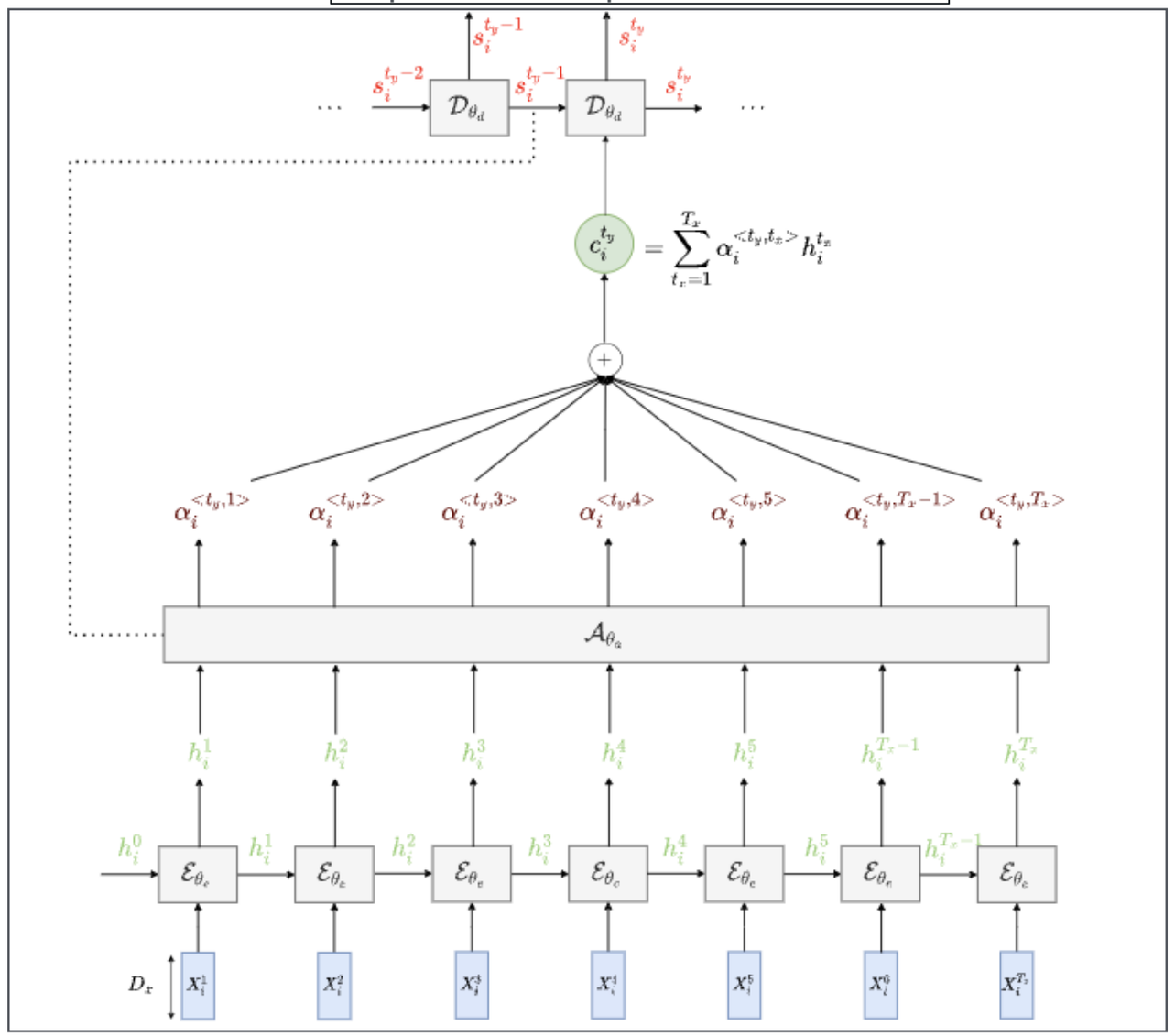
  such that: $\forall t_x \in \{1, \ldots, T_x\} \quad \alpha_i^{<t_y,t_x>} \geq 0 \quad \text{and} \quad \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} = 1$

attention weights

Vanilla Sequence to Sequence



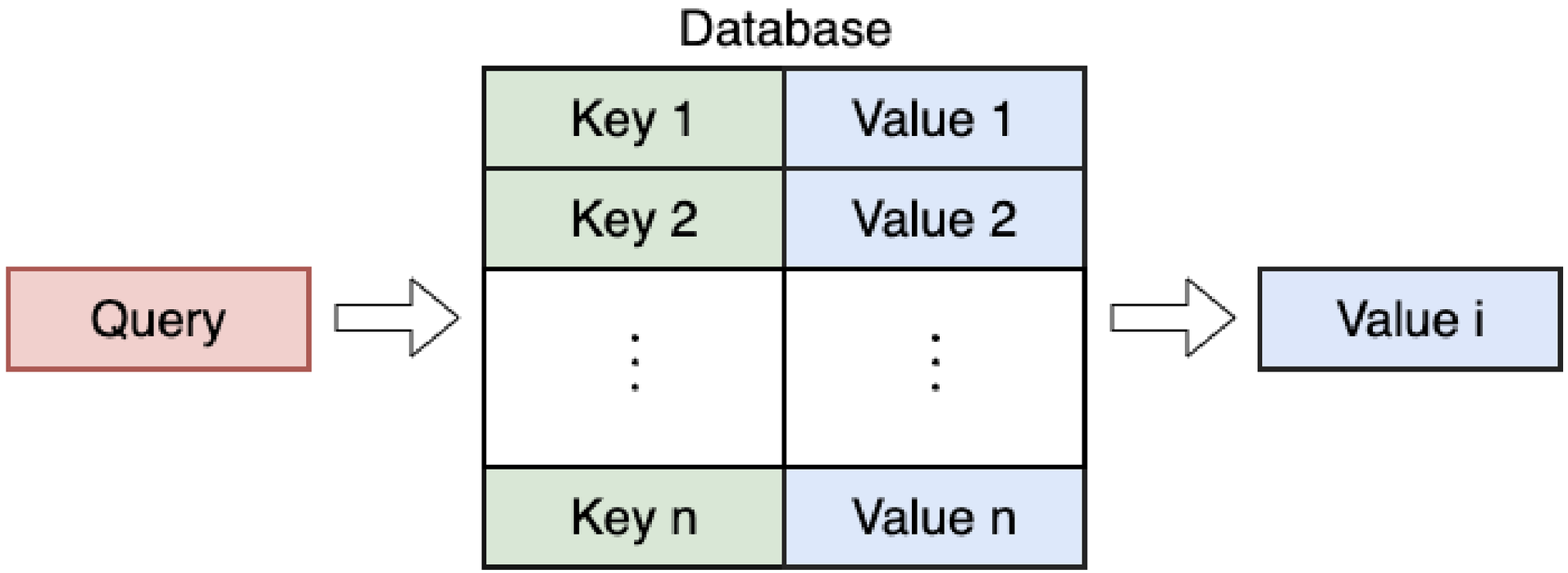
Sequence to Sequence with Attention



Interactive Session

Query-Retrieval Modeling

- Attention mechanisms intuition originates from database Query-Retrieval Problems.
- In the following database, the query retrieval problem consists in searching a query through the keys in order to retrieve a value.



Query Retrieval Modeling – an Example –

query

Attention mechanism

Database (key/value)

key

Yoshua Bengio: Attention and Consciousness (NeurIPS 2019)

value

key

C5W3L07 Attention Model Intuition

value

key

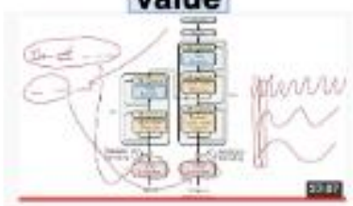
Lecture 10: Neural Machine Translation and Models with Attention

value

key

Attention Is All You Need

value



key

CS480/680 Lecture 19: Attention and Transformer Networks

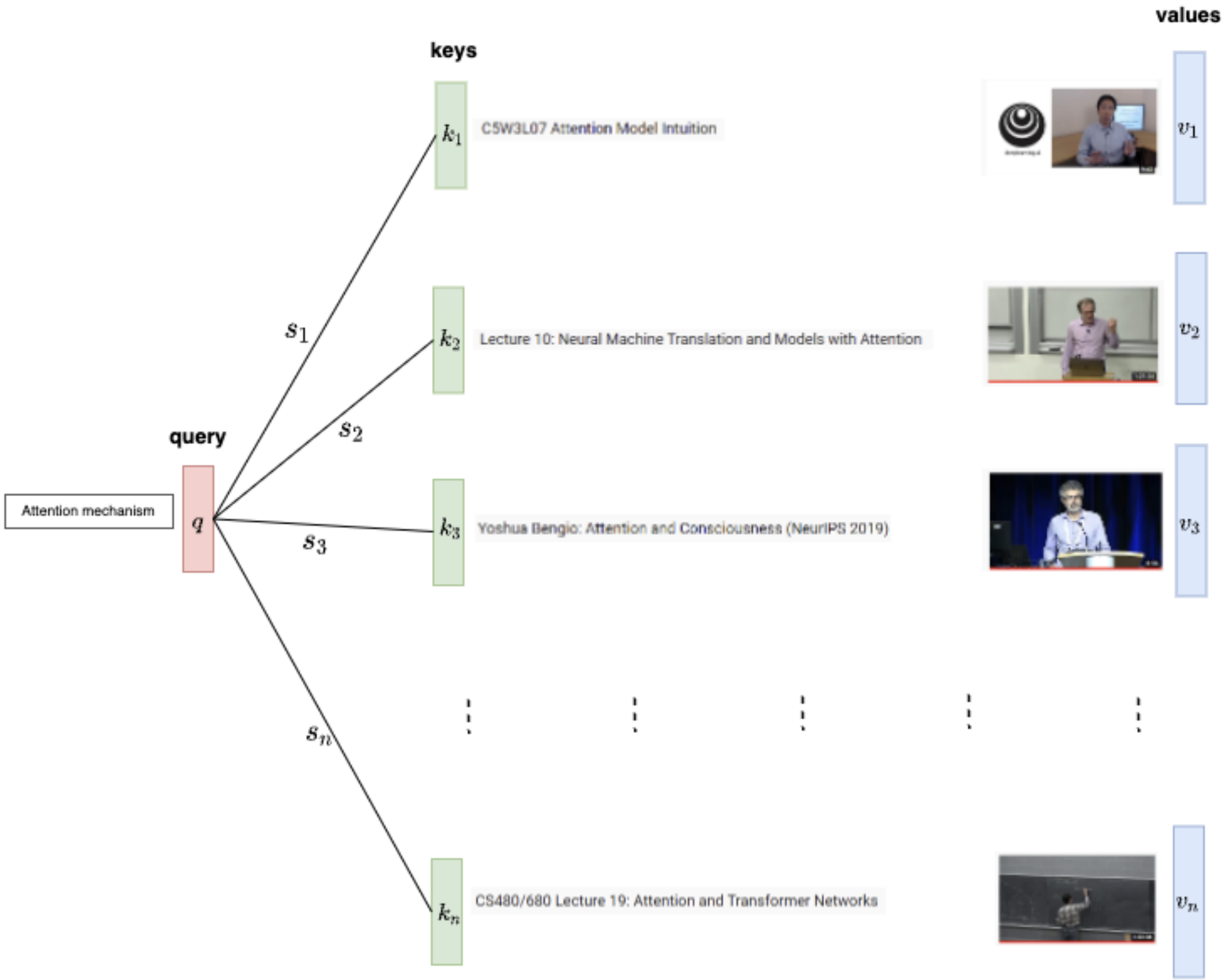
value

key

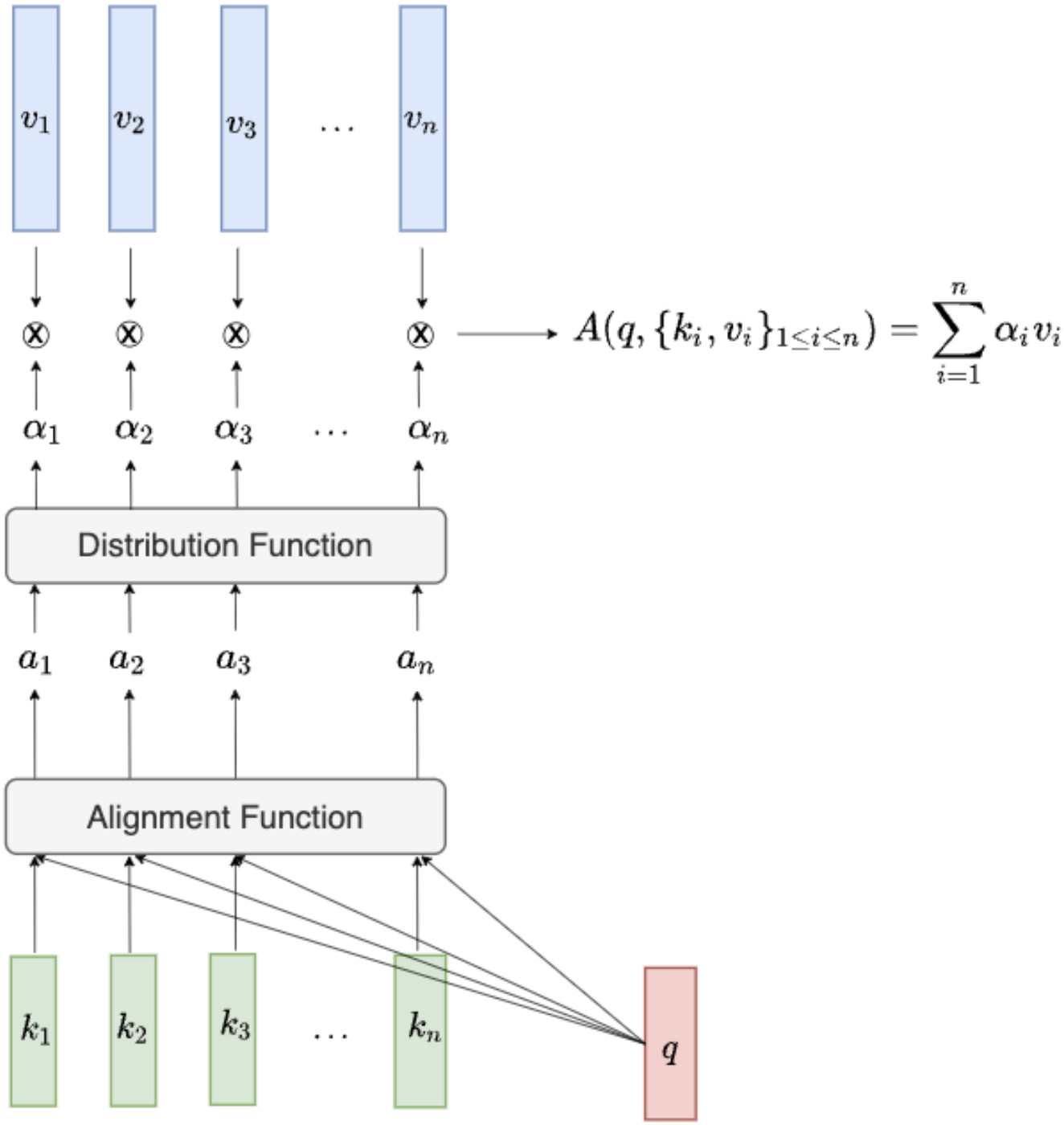
Deep Learning 7. Attention and Memory in Deep Learning

value

Query Retrieval Modeling – an Example –



Attention Mechanism as a Soft Query-Retrieval Problem

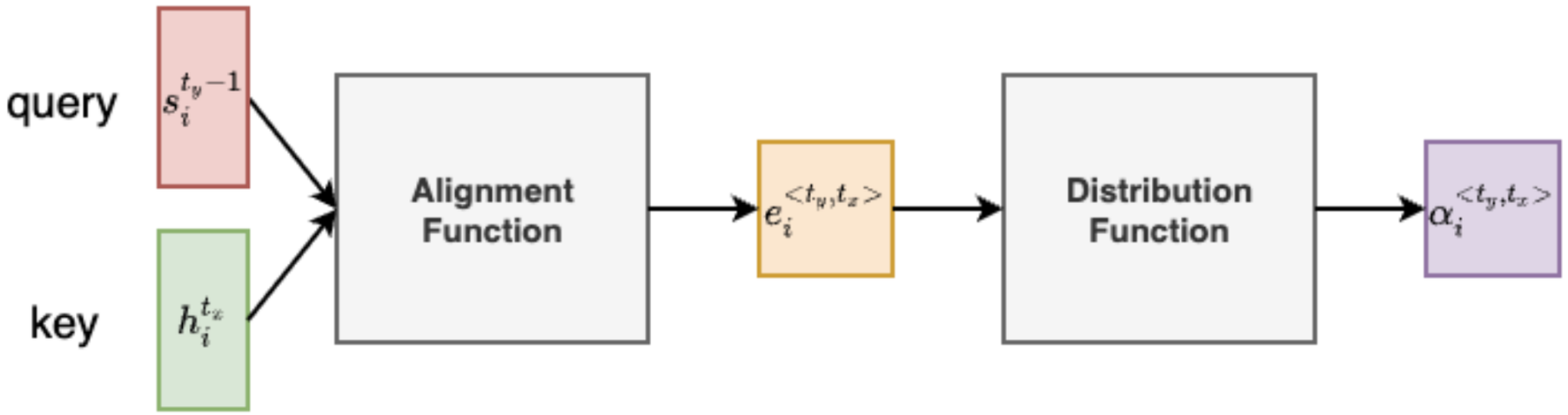


Function	Equation
Dot Product	$a(q, k_i) = q^T k_i$
Scaled Dot Product	$a(q, k_i) = \frac{q^T k_i}{\sqrt{d_k}}$
Luong's Multiplicative alignment	$a(q, k_i) = q^T W k_i$
Bahdanau's Additive alignment	$a(q, k_i) = v_a^T \tanh(W_1 q + W_2 k_i)$
Feature-based	$a(q, k_i) = W_{imp}^T \text{act}(W_1 \phi_1(k_i) + W_2 \phi_2(q) + b)$
Kernel Method	$a(q, k_i) = \phi(q)^T \phi(k_i)$

Interactive Session

The Attention Weights

- The **Attention weights**:



- The decoder input at time $t_y \in \{1, \ldots, T_y\}$, also called the **context vector** is:

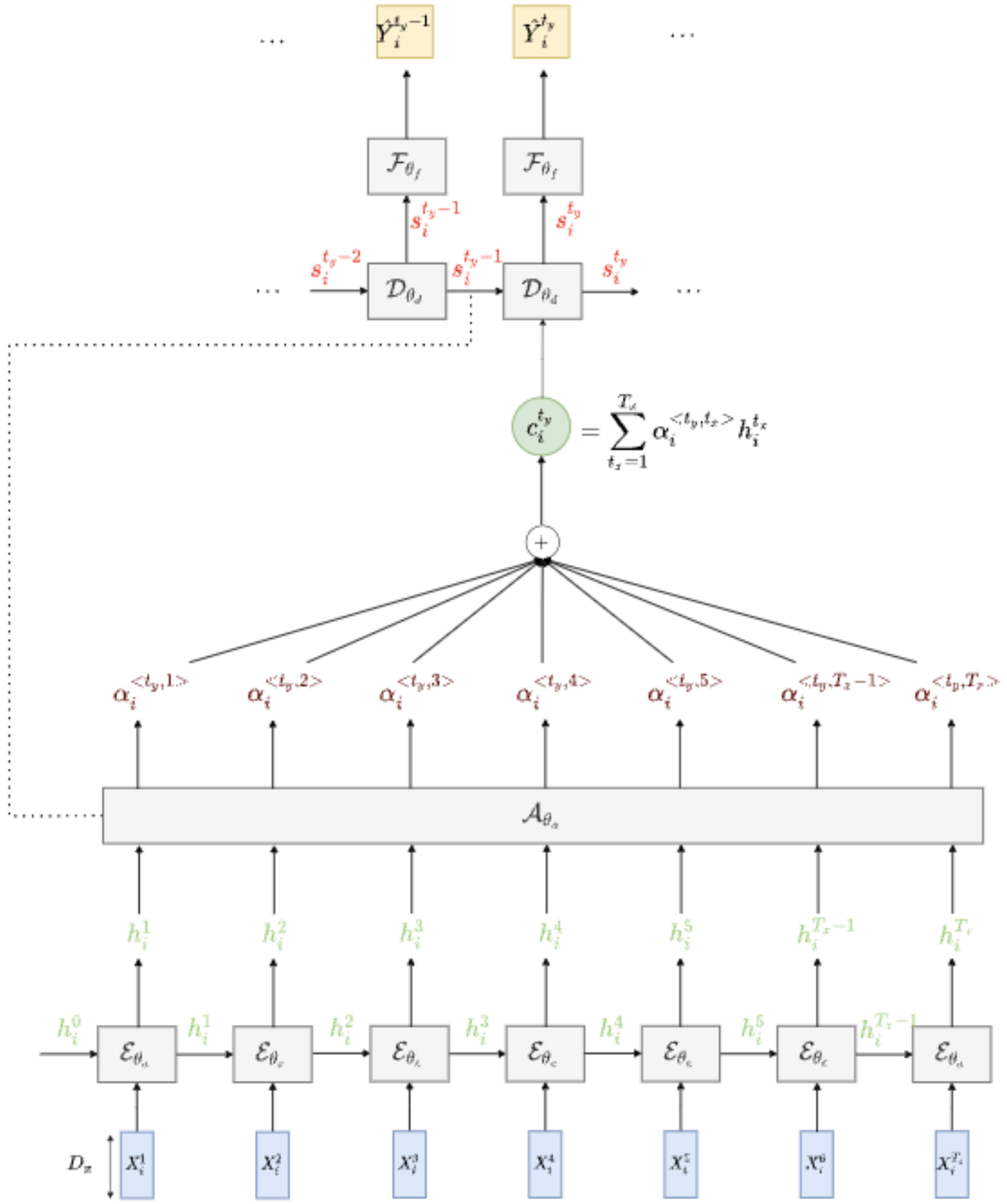
$$c_i^{t_y} = \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} h_i^{t_x}$$

values

Wrap-up: The Sequence to Sequence model with Attention

Generating $(\hat{Y}_i^1, \ldots, \hat{Y}_i^{T_y})$ using the final model:

- An Encoder $\mathcal{E}_{\theta_e}$ parameterized by θ_e maps the input embeddings $(X_i^1, \ldots, X_i^{T_x})$ to the decoder hidden states $(h_i^1, \ldots, h_i^{T_x})$
- An Attention Layer $\mathcal{A}_{\theta_a}$ parameterized by θ_a is used to compute the attention weights $\alpha_i^{<t_y, t_x>}$ in order to get the context vector $c_i^{t_y}$, which be fed into the decoder at time $t_y \in \{1, \ldots, T_y\}$
- A Decoder Layer $\mathcal{D}_{\theta_d}$ parameterized by θ_d which generates the decoder hidden states $(s_i^1, \ldots, s_i^{T_y})$
- A final Dense Layer $\mathcal{F}_{\theta_f}$ parameterized by θ_f can be used to map each decoder hidden state $s_i^{t_y}$ into the prediction $\hat{Y}_i^{t_y}$



Programming Session

Go to the following link and take Quiz 8 :

https://mlfbg.github.io/MachineLearningInFinance/