Imperial College
Business School

Imperial means
Intelligent Business

Machine Learning in Finance

Lecture 8

RNN Applications and Attention Mechanisms

Arnaud de Servigny & Jeremy Chichportich

Outline:

- The Sentiment Analysis Pipeline
- The Various Applications of RNNs
- The Sequence to Sequence Framework
- Introducing the Attention Mechanisms
- A Brief description of the Transformer Architecture

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

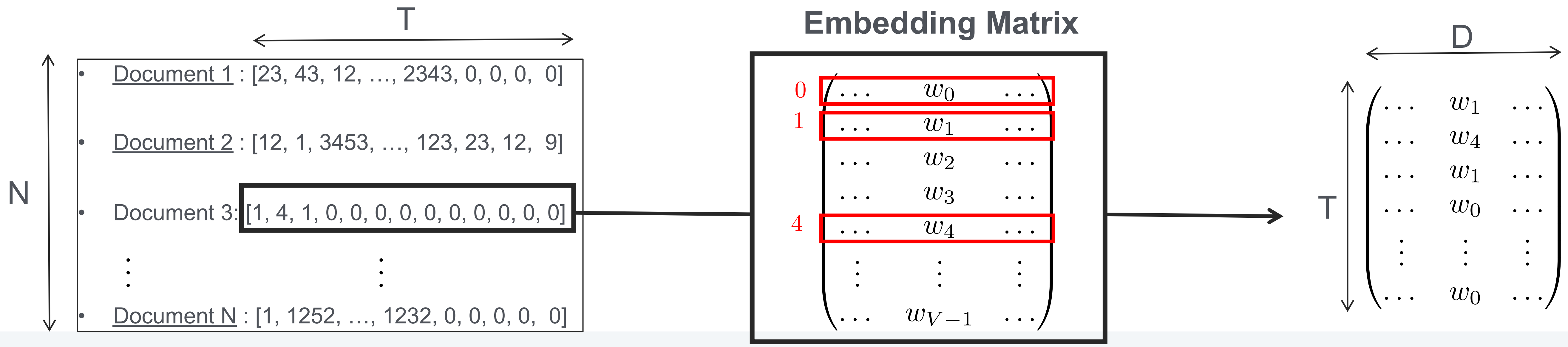
- Document 1 : [23, 43, 12, …, 2343, 0, 0, 0, 0]
- Document 2 : [12, 1, 3453, …, 123, 23, 12, 9]

⋮ ⋮

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

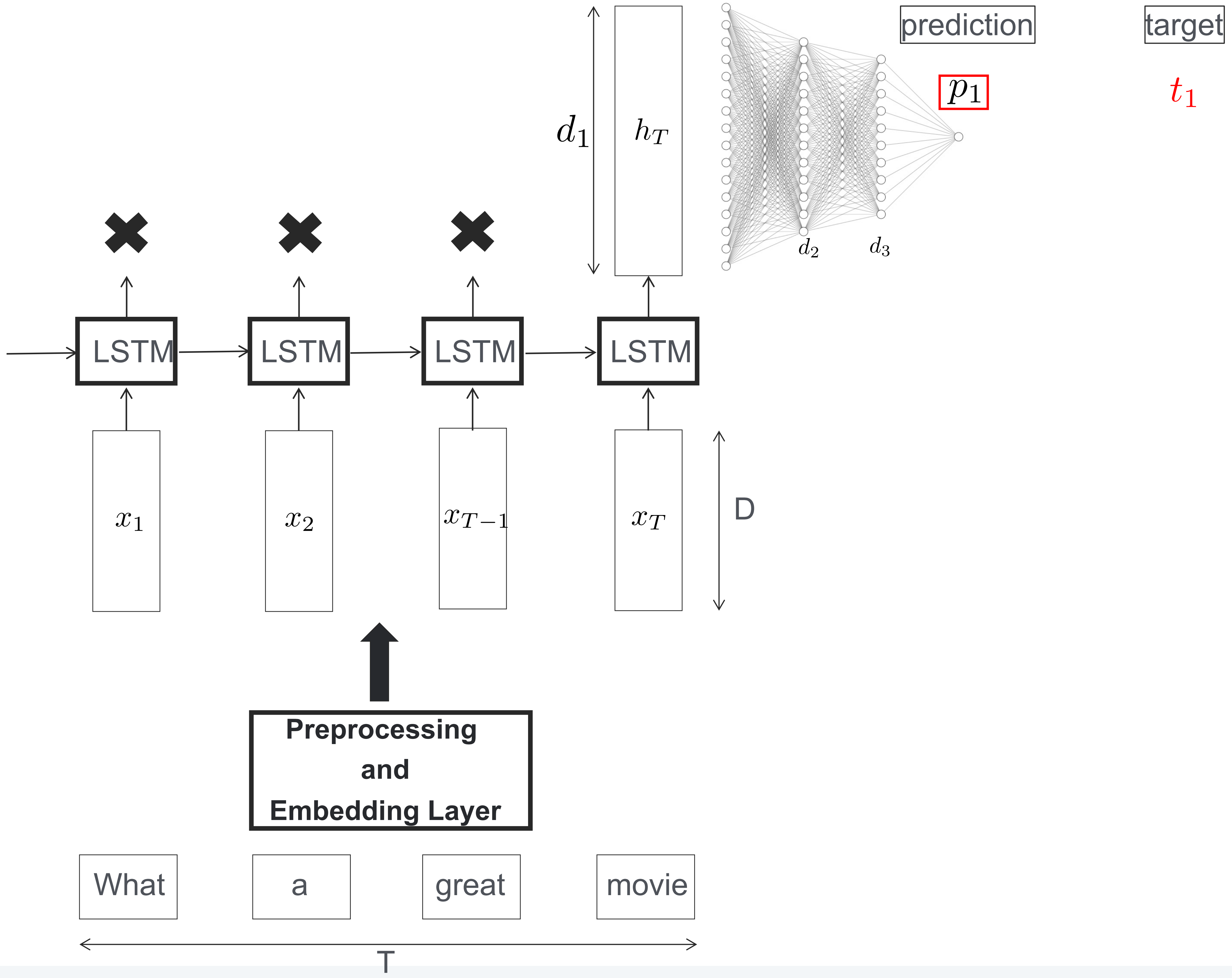
2D tensor of integers, of shape (N, T)

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


T
N
Document 1 : [23, 43, 12, …, 2343, 0, 0, 0, 0]
Document 2 : [12, 1, 3453, …, 123, 23, 12, 9]
Document 3: [1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Document N : [1, 1252, …, 1232, 0, 0, 0, 0, 0]
Embedding Matrix
0
1
4
D
T

$$
\begin{pmatrix}
\dots & w_0 & \dots \\
\dots & w_1 & \dots \\
\dots & w_2 & \dots \\
\dots & w_3 & \dots \\
\dots & w_4 & \dots \\
\vdots & \vdots & \vdots \\
\dots & w_{V-1} & \dots
\end{pmatrix}
\longrightarrow
\begin{pmatrix}
\dots & w_1 & \dots \\
\dots & w_4 & \dots \\
\dots & w_1 & \dots \\
\dots & w_0 & \dots \\
\vdots & \vdots & \vdots \\
\dots & w_0 & \dots
\end{pmatrix}
$$

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:

$$\overbrace{\boxed{\text{Input Data}}}^{\boxed{\text{Sequences of Integers}}} \qquad\qquad \boxed{\text{Predictions}}$$

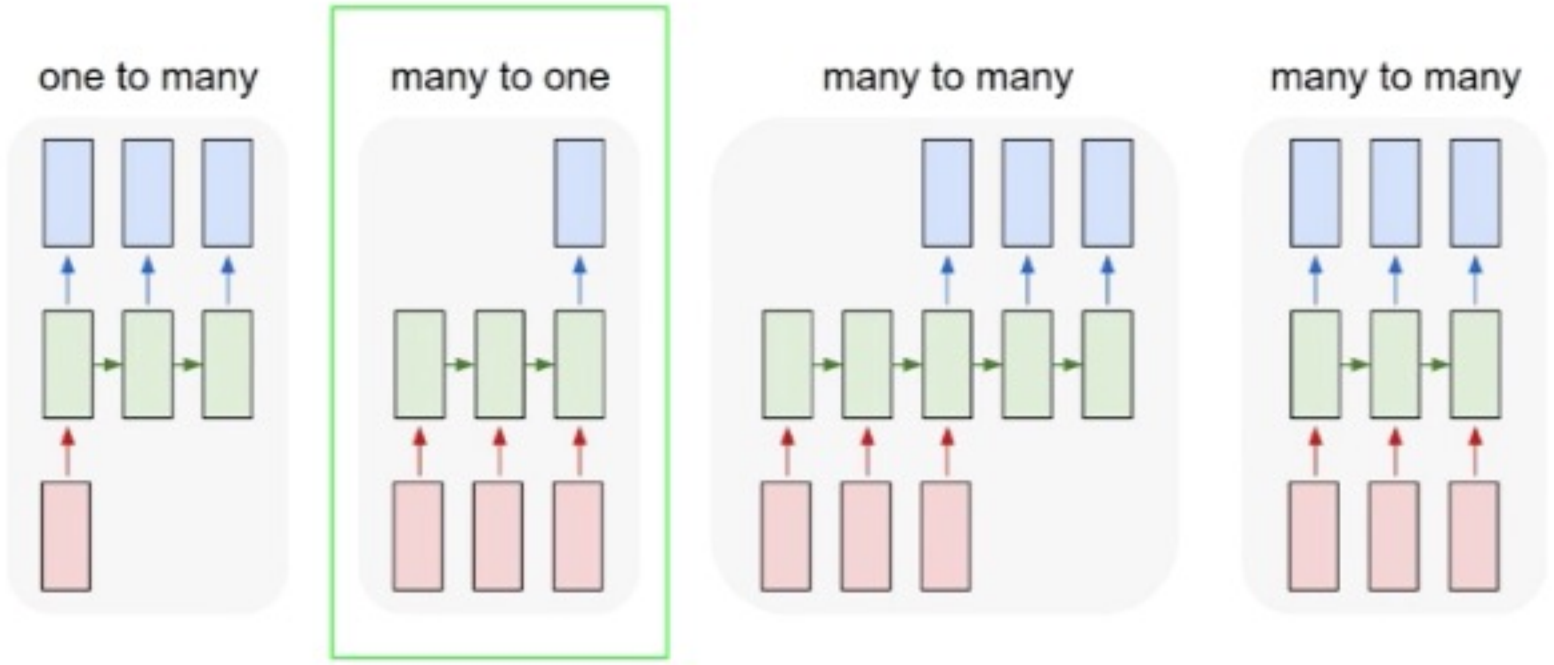
$$\begin{tikzcd} (N,T) \arrow[r, "\text{Embedding layer}"'] & (N,T,D) \arrow[r, "\text{LSTM layer}"'] & (N,d_1) \arrow[r, "\text{Dense layer}"'] & (N,d_2) \arrow[r, "\text{Dense layer}"'] & (N,d_3) \arrow[r, "\text{Dense layer}"'] & (N,) \end{tikzcd}$$

$$\xrightarrow[\text{The Forward Propagation}]{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

Part 2 : The Various Applications of RNNs

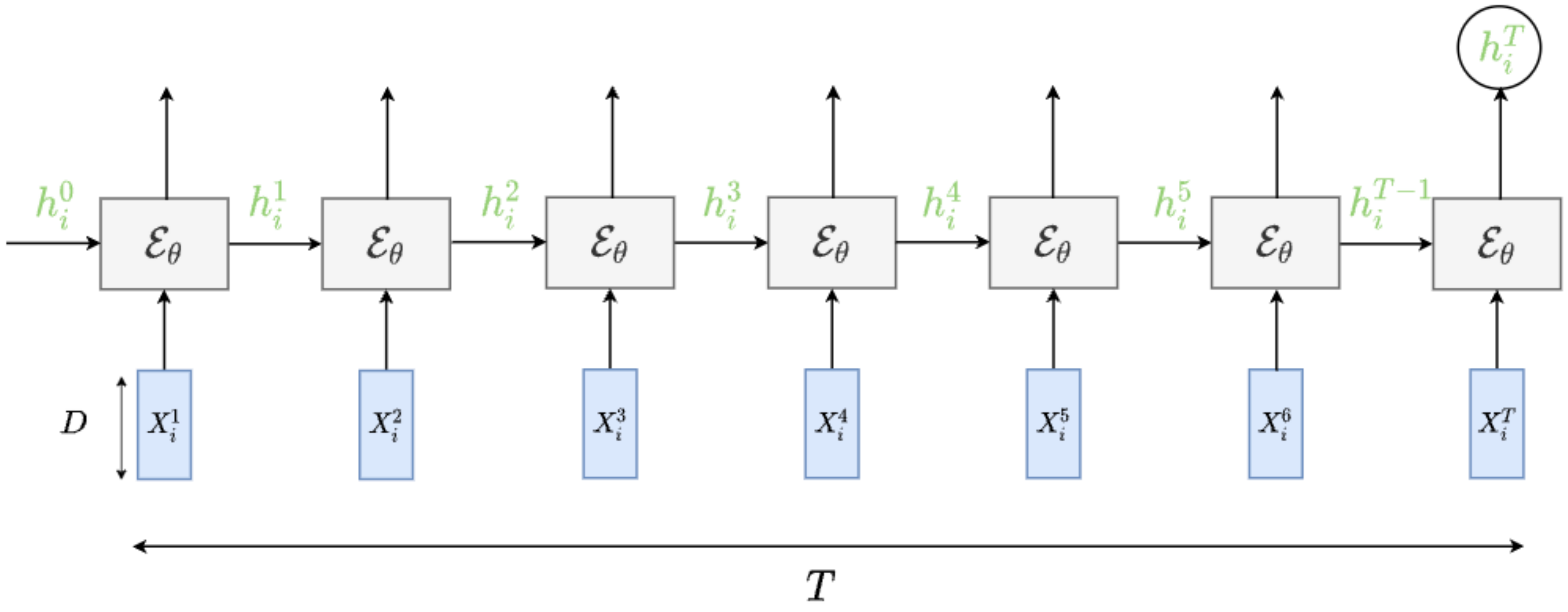
The Various Applications of RNNs

- 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 $(X_i^1, \dots, X_i^T) \in \mathbb{R}^{T \times D}$ into a vector $h_i^T \in \mathbb{R}^d$ 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.
    - News Classification.

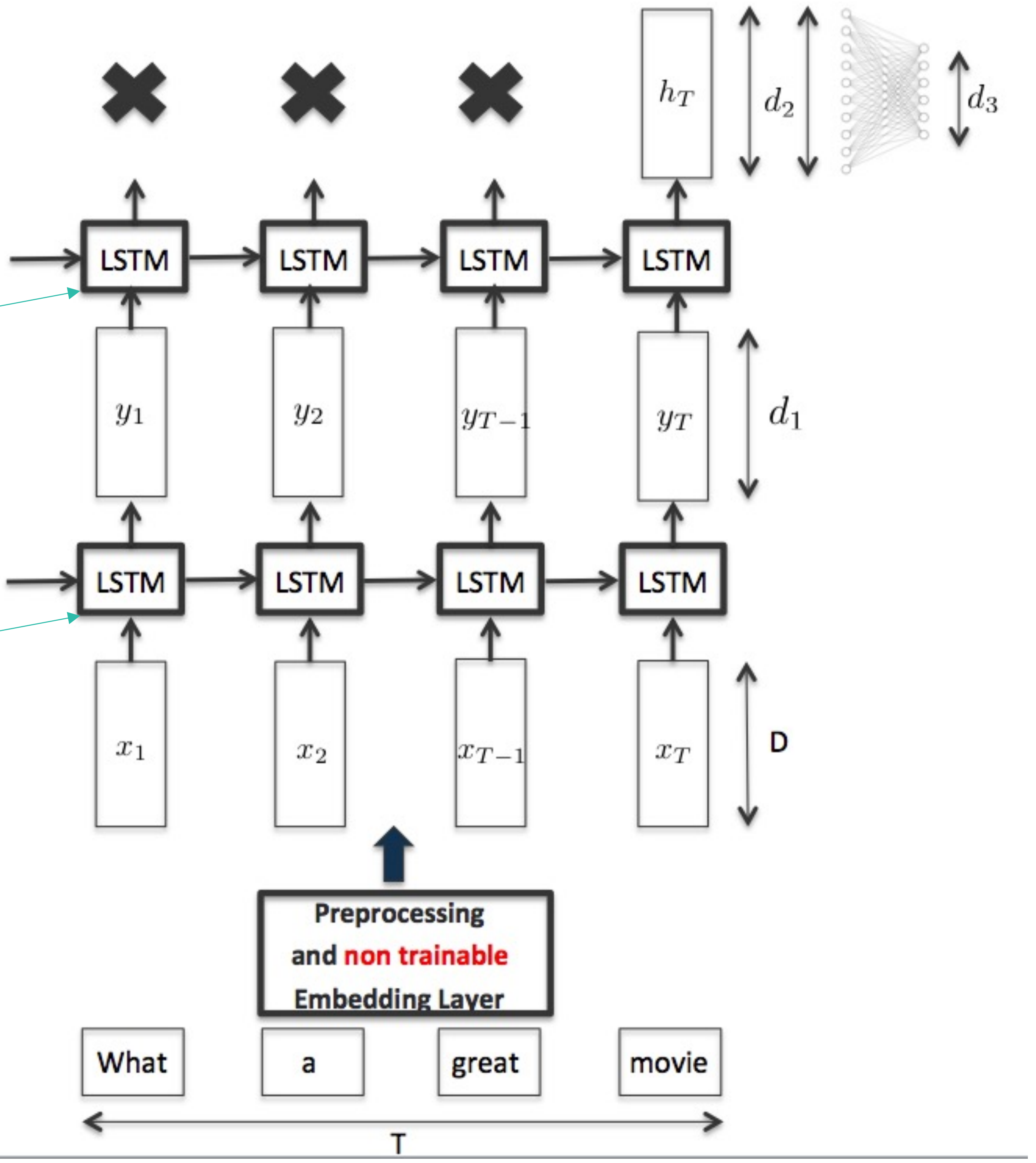
- Let us consider some examples in the next slides.

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)
```



See Programming Session 7 for the code

Stacking LSTM layers for a Multiclass classification Problem

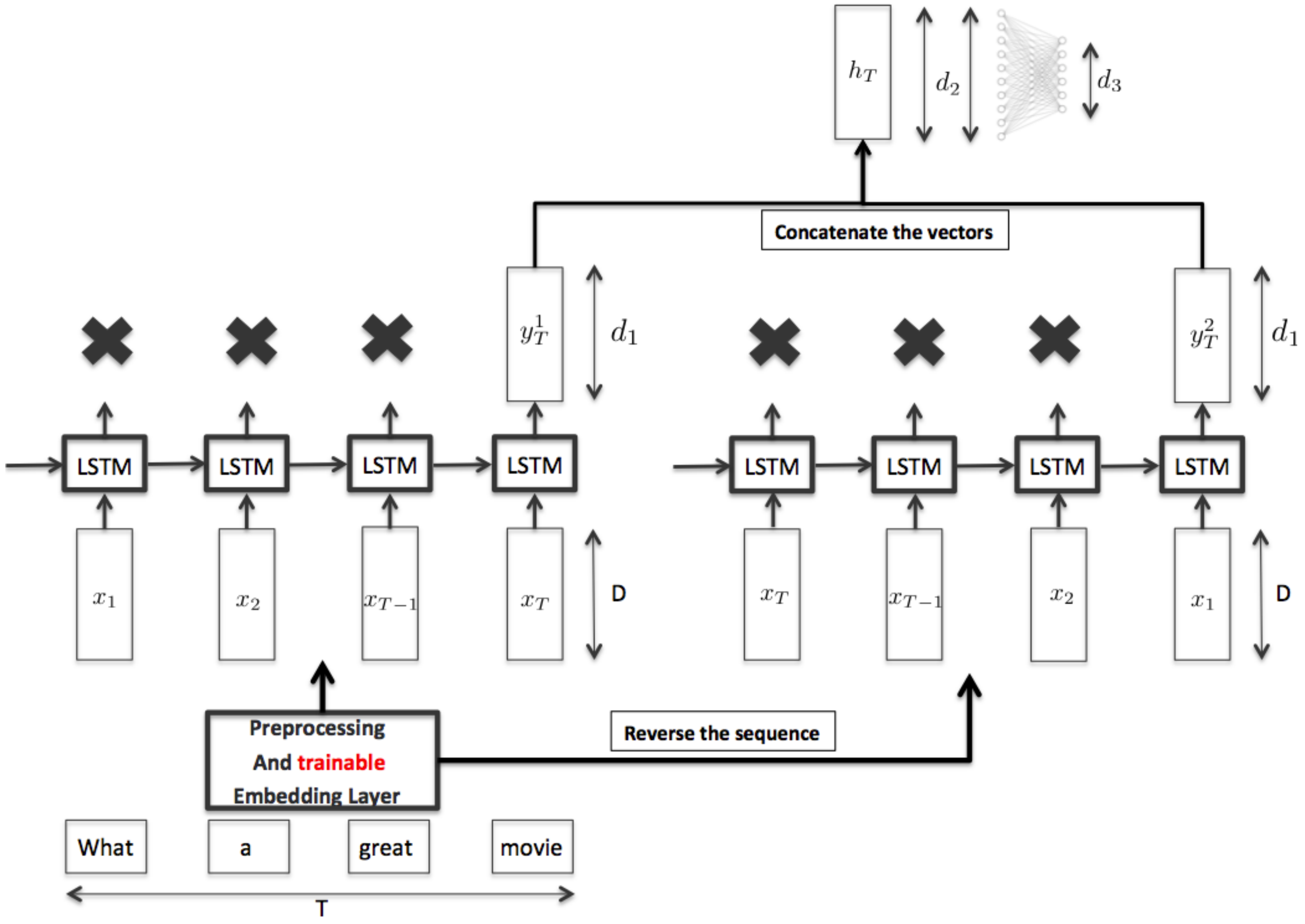
- Let’s keep track of the evolution of the tensor shape after each layer transformation:

$$\overbrace{\boxed{\text{Input Data}}}^{\boxed{\text{Sequences of Integers}}} \qquad \boxed{\text{Predictions}}$$

$$\begin{tikzcd}
(N,T) \arrow[r, "\text{Embedding layer}"'] & (N,T,D) \arrow[r, "\text{First LSTM layer}"'] & (N,T,d_1) \arrow[r, "\text{Second LSTM layer}"'] & (N,d_2) \arrow[r, "\text{Dense layer}"'] & (N,d_3)
\end{tikzcd}$$

$$\xrightarrow[\text{The Forward Propagation}]{\hspace{20em}}$$

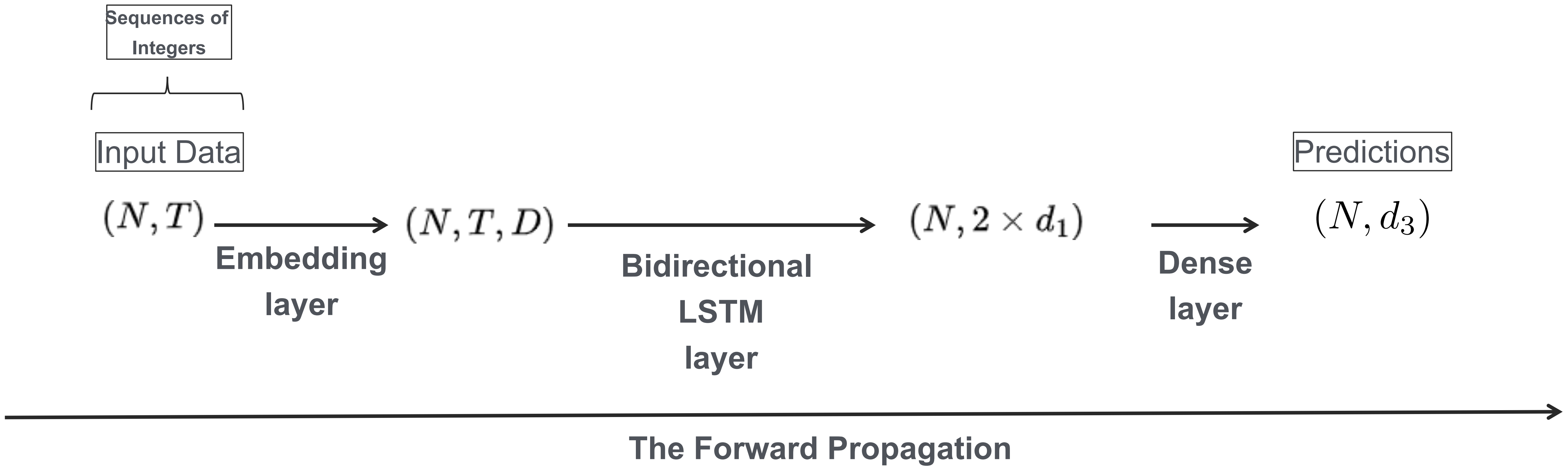
Bidirectional LSTM for a Multiclass classification Problem



See Programming Session 7 for the code

Stacking LSTM layers for a Multiclass classification Problem

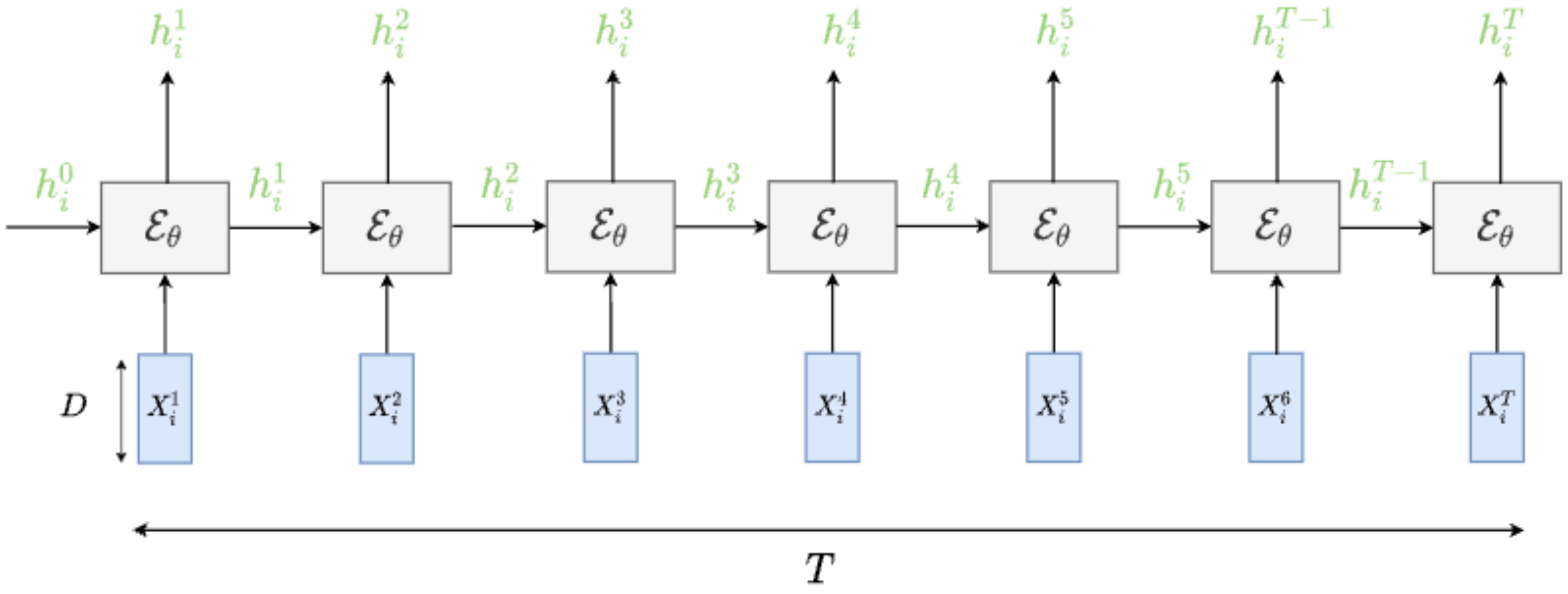
- Let's keep track of the evolution of the tensor shape after each layer transformation:



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

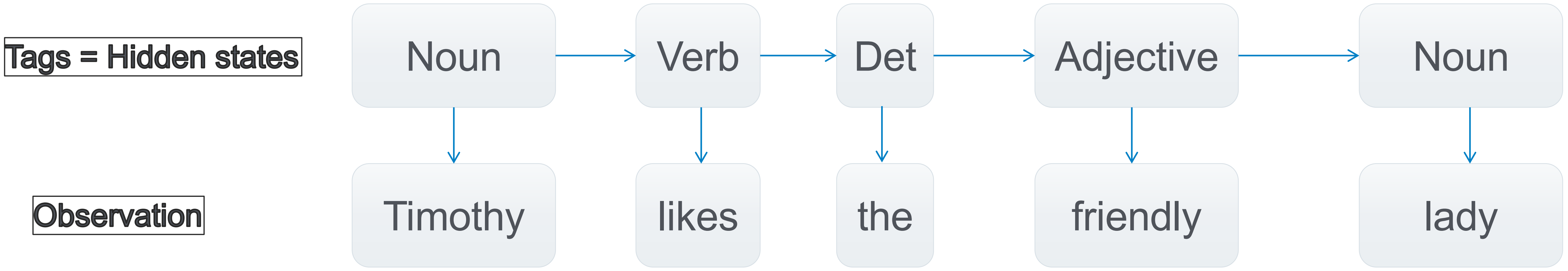
- In the Many to Many framework, the objective is to map a sequence $(X_i^1, \ldots, X_i^T) \in \mathbb{R}^{T \times D}$ into a sequence $(h_i^1, \ldots, h_i^T) \in \mathbb{R}^{T \times d}$ using the LSTM layer $\mathcal{E}_\theta$ parameterized by θ

- 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 –

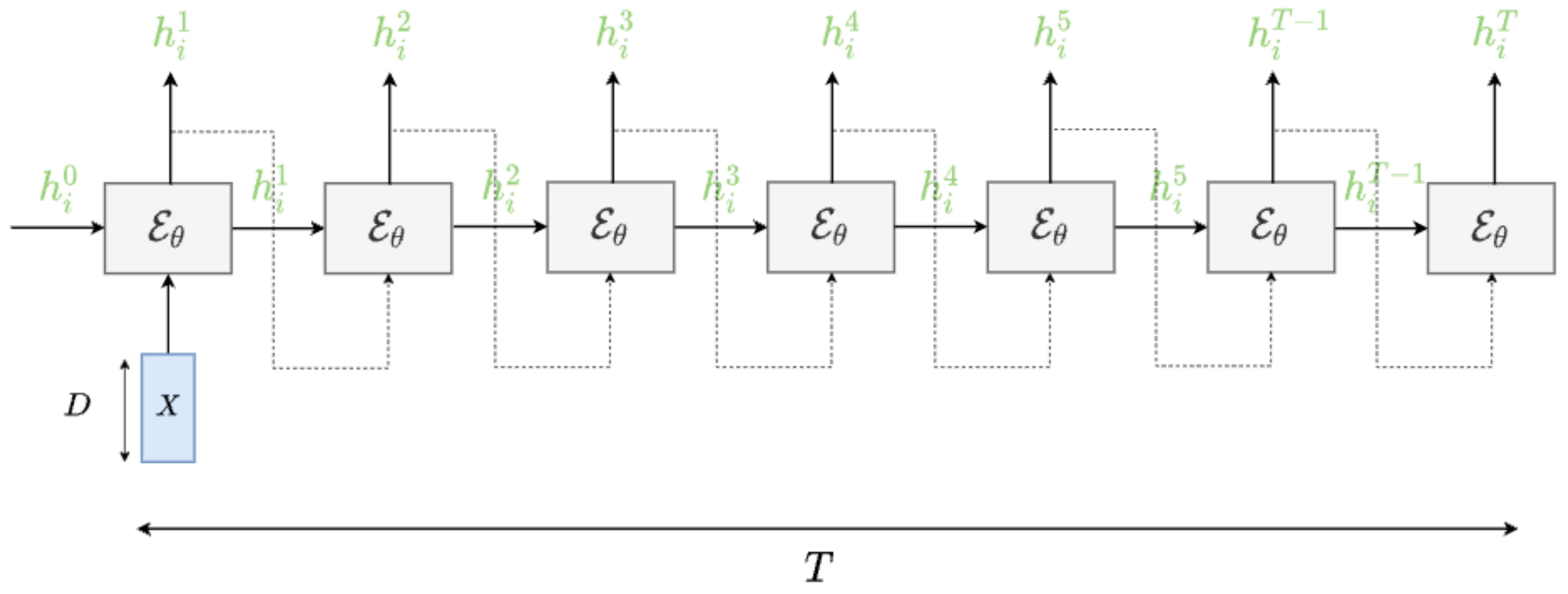
- POS (Part Of Speech) Tagging is a typical example, where the objective is to tag each word of a sentence with its "Part-of-Speech" tag.
- Another popular model can be used for POS tagging: The Hidden Markov Model (HMM).



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

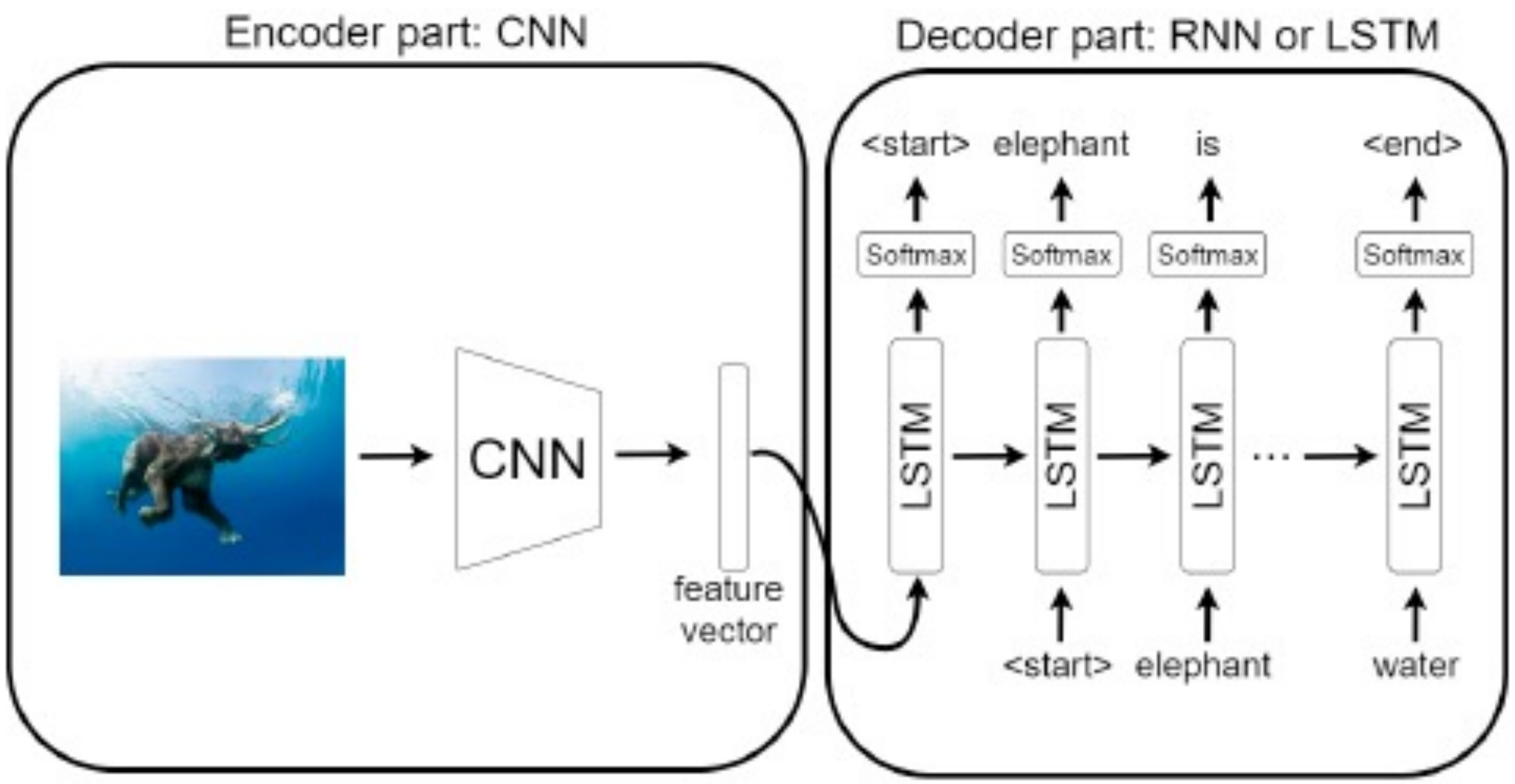
The One to Many Problem – The Architecture –

- 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.



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).

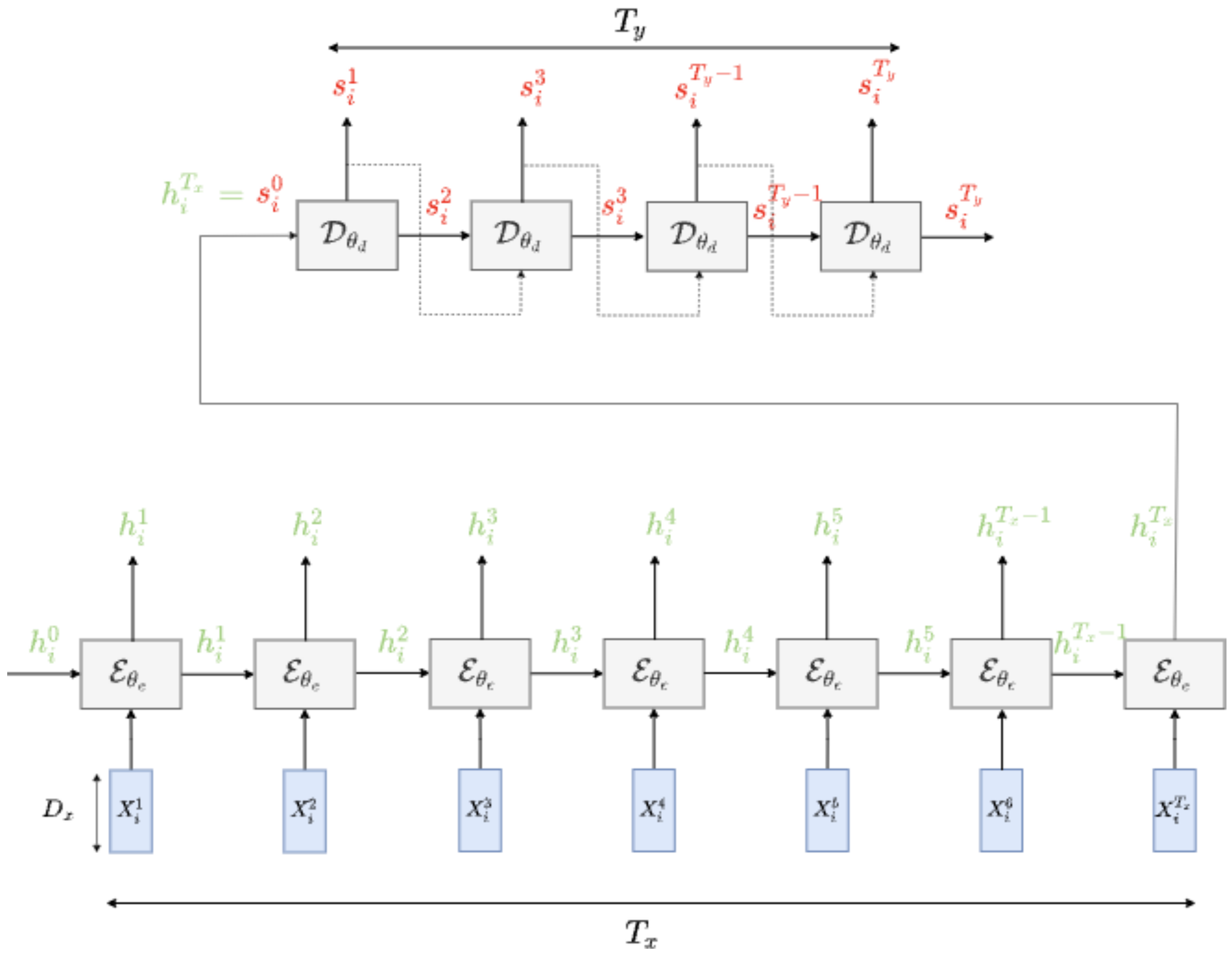
- However if we want to learn a mappy from a sequence of inputs into a sequence of output of different length, we need to introduce a new framework, composed of two steps:

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

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

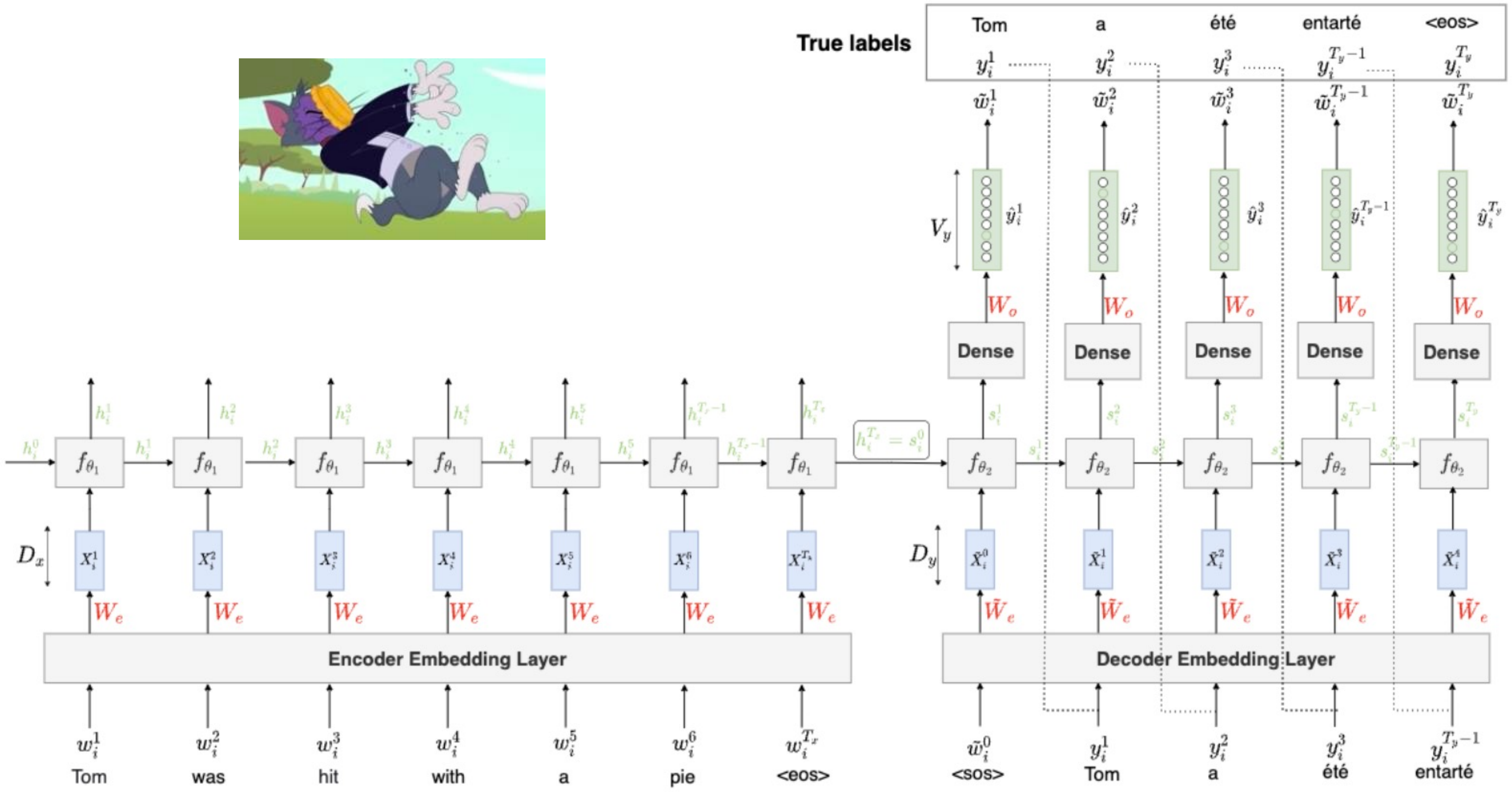
$$h_i^{T_x} = s_i^0$$

- We can the generate the sequence of hidden states associated with the decoder: $(s_i^1, \dots, s_i^{T_y})$



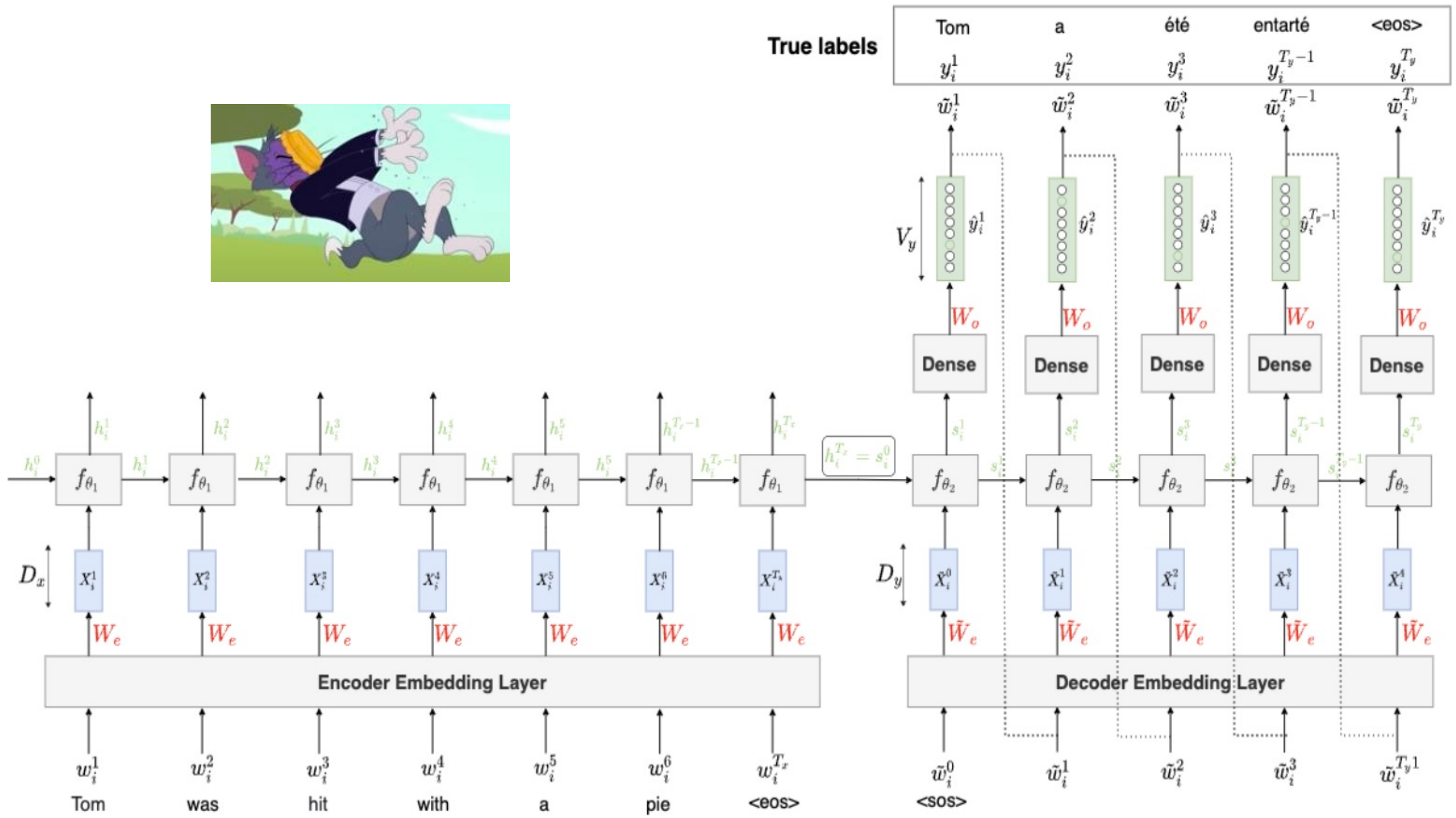
The Sequence to Sequence Framework –an Example–

- An exemplary use case of the SeqtoSeq architecture is seen in Neural Machine Translation.
- Typically, we employ **Teacher Forcing** in the training phase.



The Sequence to Sequence Framework –an Example–

- In the prediction phase, at every iteration, the decoder's output is looped back into the model.



True labels
Tom
a
été
entarté
<eos>
y_i^1 y_i^2 y_i^3 $y_i^{T_y-1}$ $y_i^{T_y}$
$\tilde{w}_i^1$ $\tilde{w}_i^2$ $\tilde{w}_i^3$ $\tilde{w}_i^{T_y-1}$ $\tilde{w}_i^{T_y}$
V_y
$\hat{y}_i^1$ $\hat{y}_i^2$ $\hat{y}_i^3$ $\hat{y}_i^{T_y-1}$ $\hat{y}_i^{T_y}$
W_o
Dense
h_i^0 h_i^1 h_i^2 h_i^3 h_i^4 h_i^5 $h_i^{T_x-1}$ $h_i^{T_x}$
$h_i^{T_x} = s_i^0$
s_i^1 s_i^2 s_i^3 $s_i^{T_y-1}$ $s_i^{T_y}$
f_{θ_1}
f_{θ_2}
D_x
X_i^1 X_i^2 X_i^3 X_i^4 X_i^5 X_i^6 $X_i^{T_x}$
D_y
$\tilde{X}_i^0$ $\tilde{X}_i^1$ $\tilde{X}_i^2$ $\tilde{X}_i^3$ $\tilde{X}_i^4$
W_e
$\tilde{W}_e$
Encoder Embedding Layer
Decoder Embedding Layer
w_i^1 w_i^2 w_i^3 w_i^4 w_i^5 w_i^6 $w_i^{T_x}$
Tom
was
hit
with
a
pie
<eos>
$\tilde{w}_i^0$ $\tilde{w}_i^1$ $\tilde{w}_i^2$ $\tilde{w}_i^3$ $\tilde{w}_i^{T_y1}$
<sos>

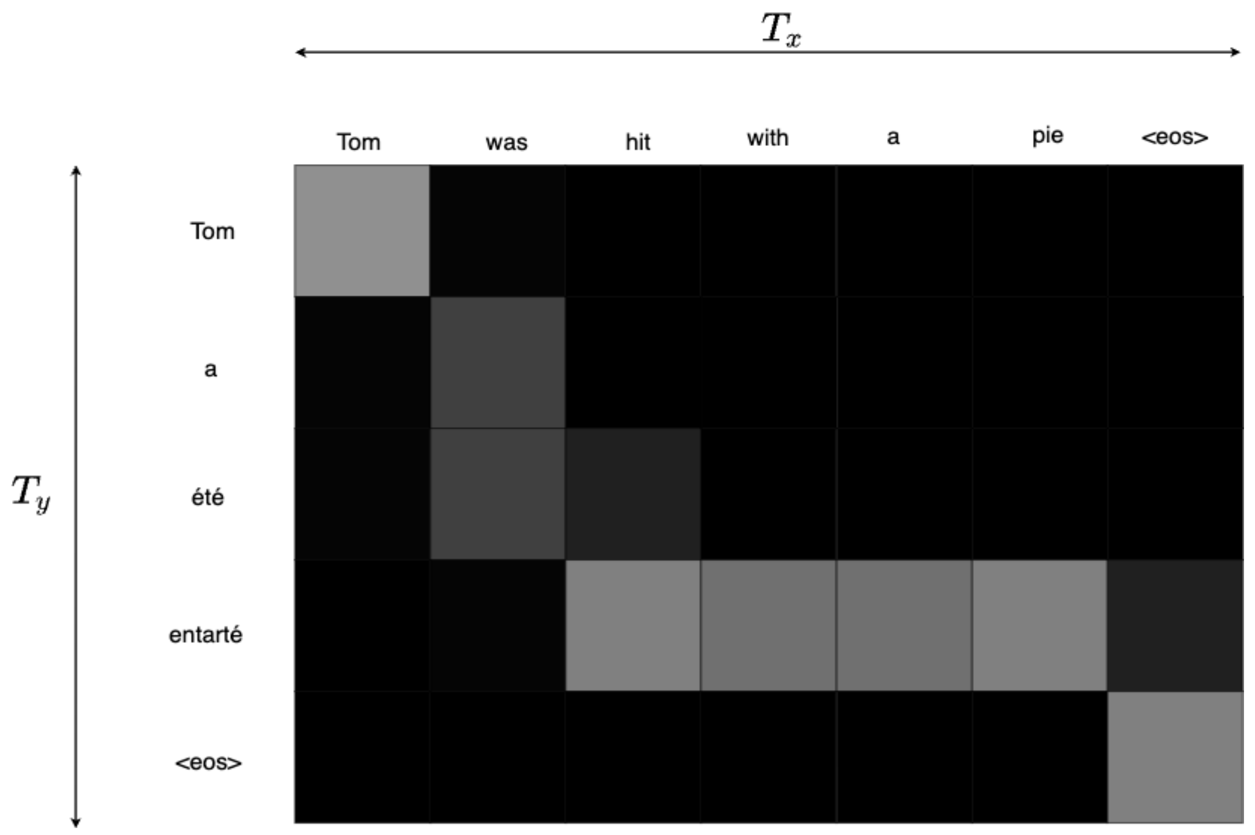
The Sequence to Sequence Framework –Limitations–

- **Handling long sequences**: 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.

- **Variable-length sequences**: Another challenge with SeqToSeq models is handling variable-length input and output sequences. In many real-world applications, the length of the input and output sequences can vary significantly, which can make it difficult to design a model that can handle all possible sequence lengths.

- **Lack of interpretability**: SeqToSeq models can be difficult to interpret, as the model's internal representations can be complex and hard to understand. This can make it challenging to diagnose and correct errors in the model's predictions.

- We want the model to be able to dynamically focus on the most relevant parts of the input sequence when generating each output token, rather than compressing all of the input information into a single fixed-length vector.



Part 4 : Introducing the Attention Mechanisms

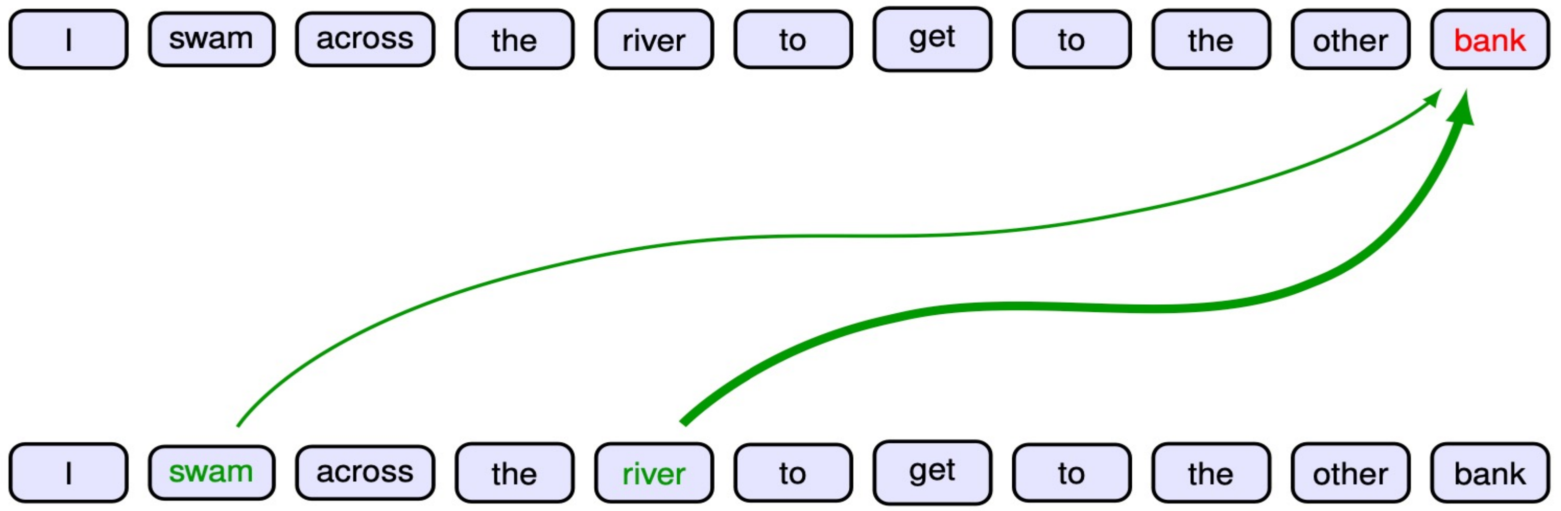
Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Intuition

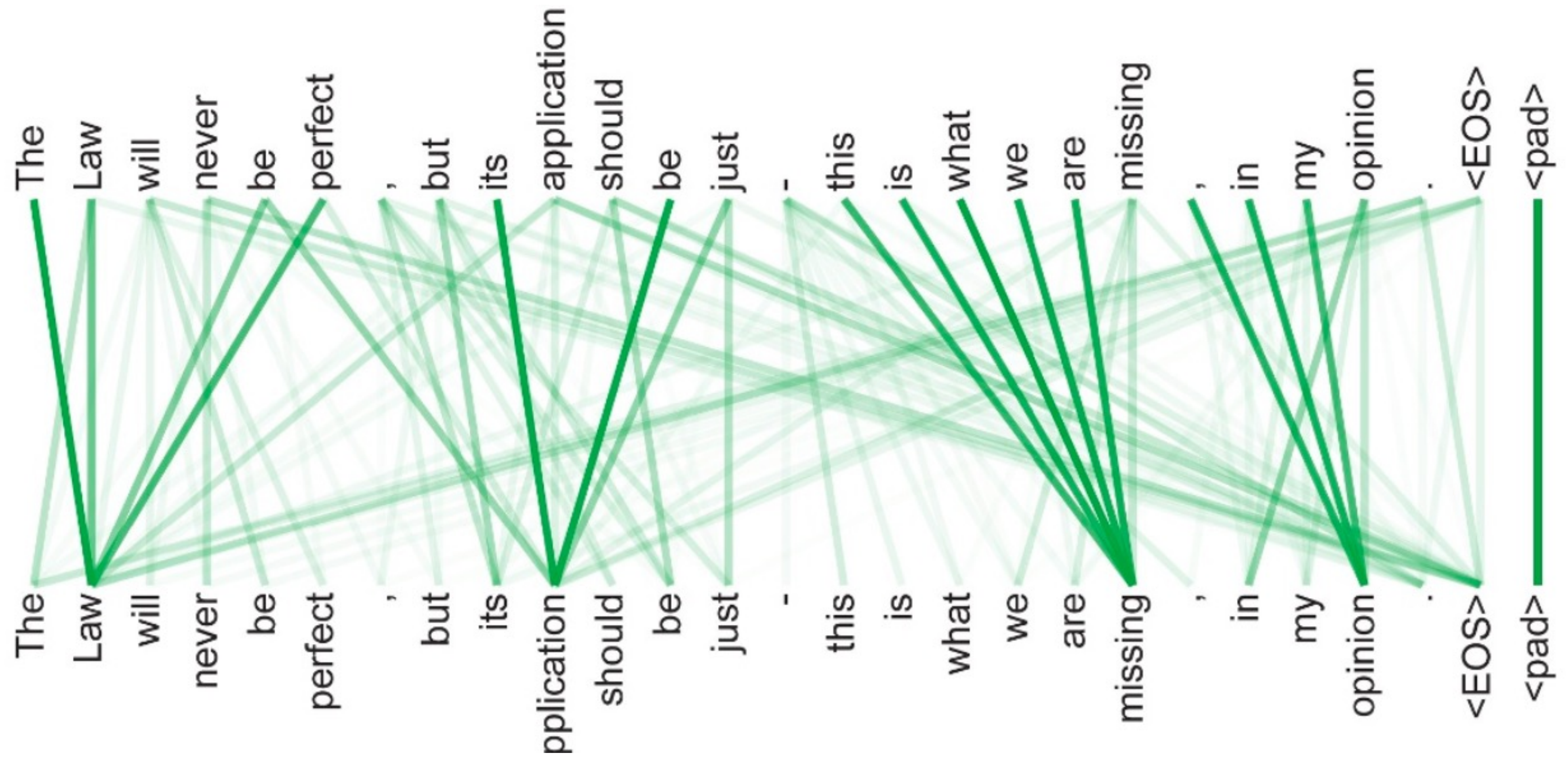
- Consider the following two sentences:
    1. I swam across the river to get to the other bank.
    2. I walk across the road to get cash from the other bank

- Context is key for understanding bank:
    1. In the first sentence, swam and river suggest bank means riverbank.
    2. In the second sentence, cash suggests bank means a financial institution.

- Traditional recurrent neural networks use fixed importance for each word → we need adjustable importance to deal with that issue.

Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Part 1





Source Christopher M. Bishop • Hugh Bishop

Deep Learning Foundations and Concepts Springer

Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Part 2

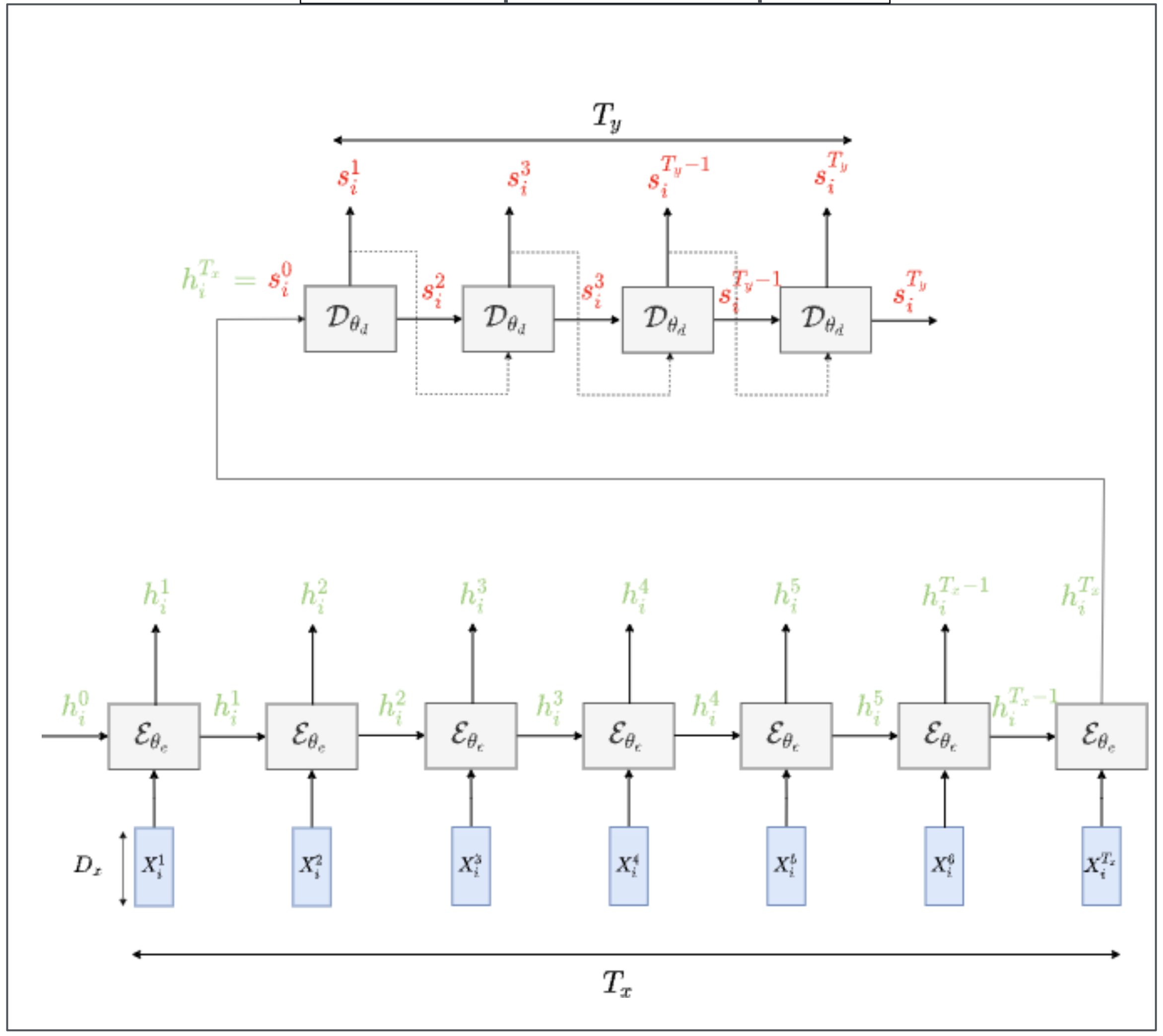
- In the basic Sequence to Sequence model, the entire input sequence is condensed into a single fixed-length vector.

- At each decoder time step $t_y \in \{1, \ldots, T_y\}$ we would like the input vector to be: $c_i^{t_y} = \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} h_i^{t_x}$

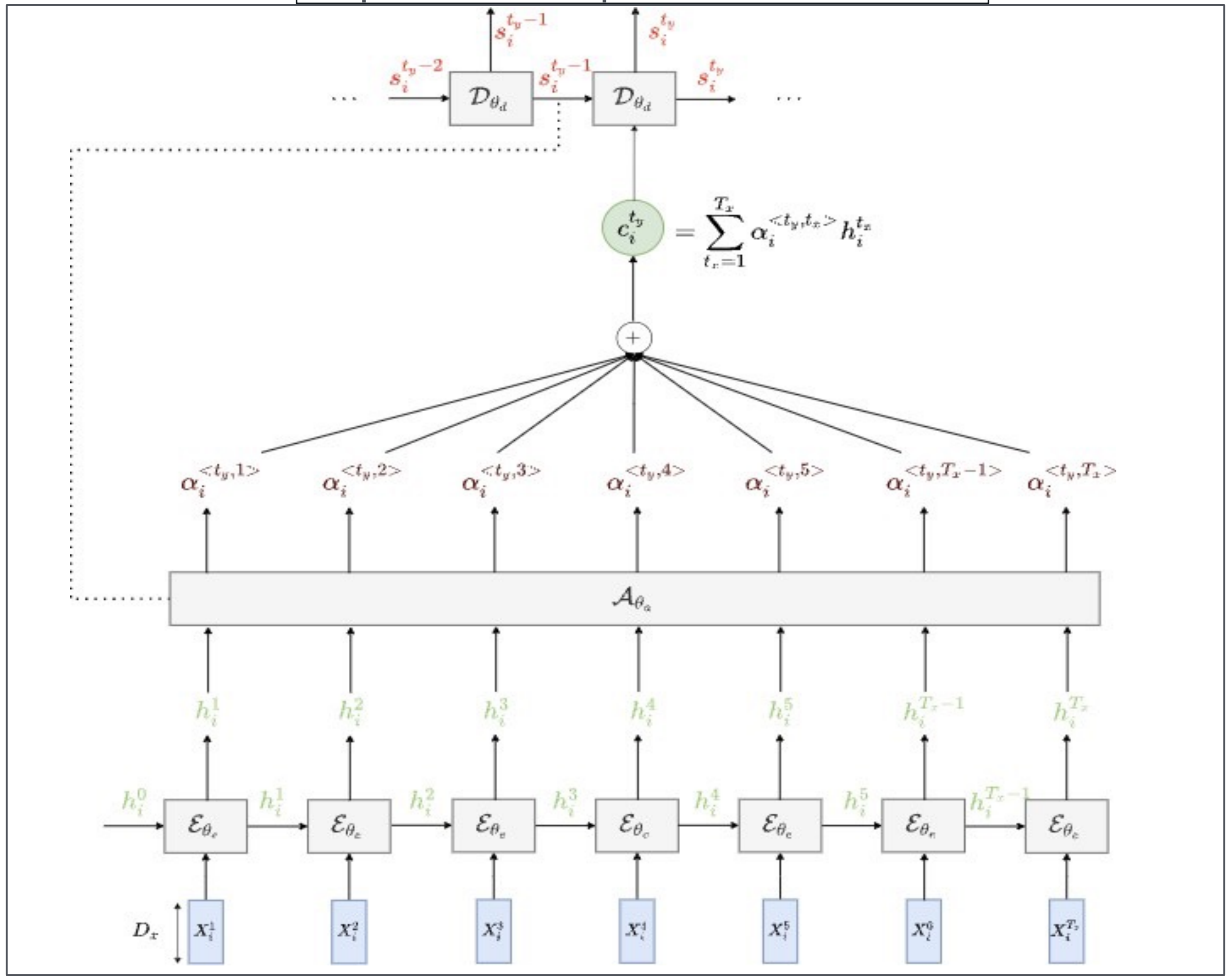
  such that: $\forall t_x \in \{1, \ldots, T_x\} \quad \alpha_i^{<t_y,t_x>} \geq 0 \quad \text{and} \quad \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} = 1$

attention weights

Vanilla Sequence to Sequence

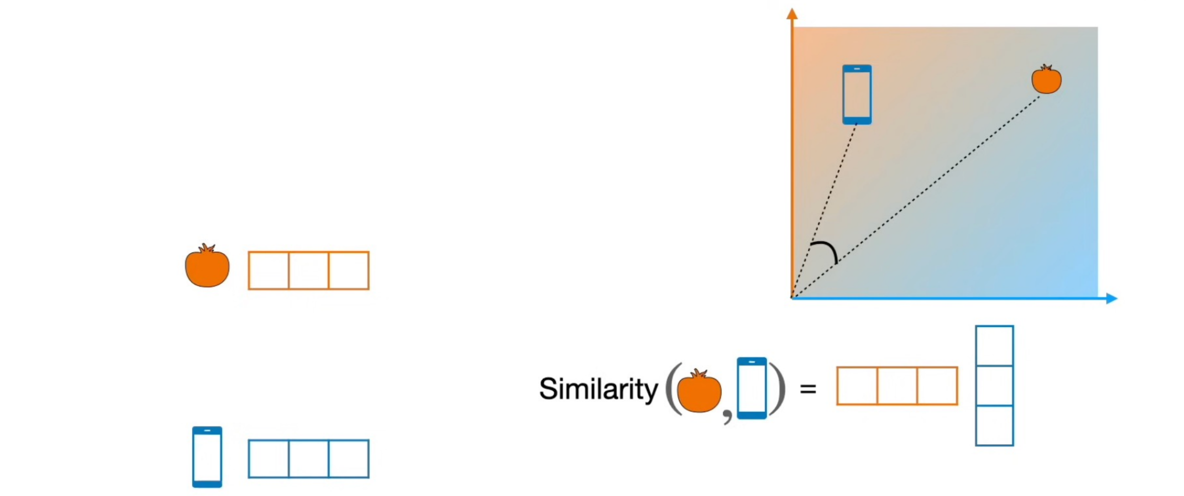


Sequence to Sequence with Attention

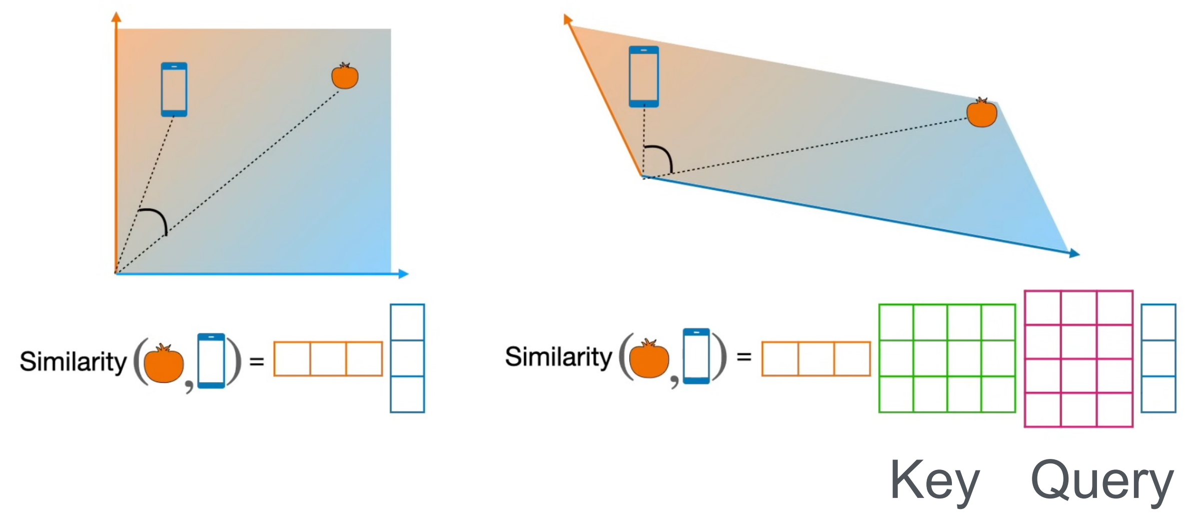

$s_i^{t_y-1}$
$s_i^{t_y}$
$s_i^{t_y-2}$
$s_i^{t_y-1}$
$s_i^{t_y}$
$\mathcal{D}_{\theta_d}$
$c_i^{t_y} = \sum_{t_x=1}^{T_x} \alpha_i^{<t_y,t_x>} h_i^{t_x}$
+
$\alpha_i^{<t_y,1>}$ $\alpha_i^{<t_y,2>}$ $\alpha_i^{<t_y,3>}$ $\alpha_i^{<t_y,4>}$ $\alpha_i^{<t_y,5>}$ $\alpha_i^{<t_y,T_x-1>}$ $\alpha_i^{<t_y,T_x>}$
$\mathcal{A}_{\theta_a}$
h_i^1 h_i^2 h_i^3 h_i^4 h_i^5 $h_i^{T_x-1}$ $h_i^{T_x}$
h_i^0
$\mathcal{E}_{\theta_e}$
D_x
X_i^1 X_i^2 X_i^3 X_i^4 X_i^5 X_i^6 $X_i^{T_x}$

Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Part 3

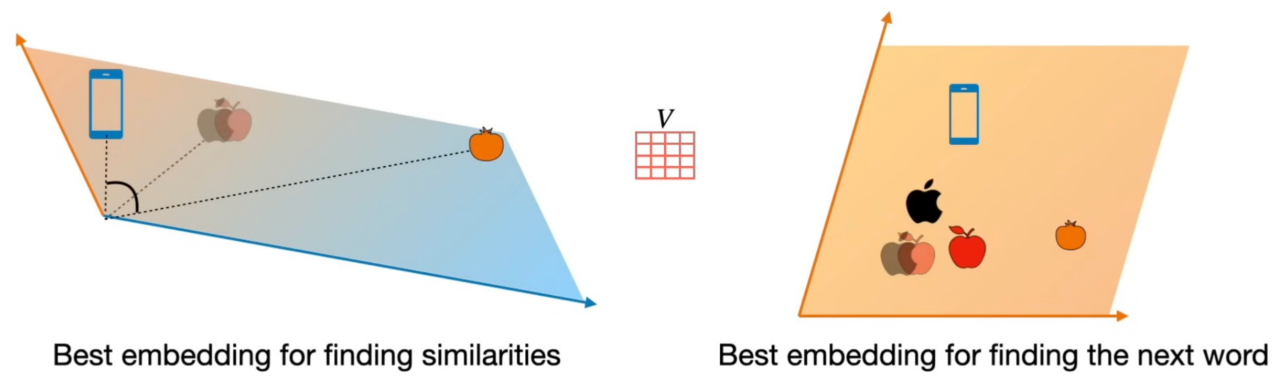
Similarity: the Intuition behind Key, Query and Value



Similarity between two objects is provided by the 'sum-product' outcome



By rotating the two objects of concern, thanks to 'Key' and 'Query' matrices the Similarity measure gets clarified


V
Best embedding for finding similarities
Best embedding for finding the next word

By adding a 'Value' matrix transformation, the objective is to make the search for the next word optimal

Source: Serrano Academy

Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Part 4

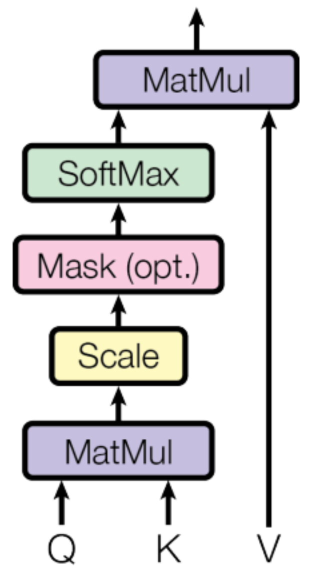
- The (Key, Query, Value) system is a common approach used in attention mechanisms to compute the attention weights. The input data is first transformed into three separate components: keys, queries, and values.

- **Keys**: The keys represent the input elements that the network will attend to. *For example, in a machine translation task, the keys could be the words in the source language sentence.*

- **Queries**: The queries represent the input elements that the network will use to compute the attention weights. *For example, in a machine translation task, the queries could be the words in the target language sentence that the network is currently generating.*

- **Values**: The values represent the input elements that the network will use to compute the output, based on the attention weights. *For example, in a machine translation task, the values could be the word embeddings for the source language sentence.*

Scaled Dot-Product Attention

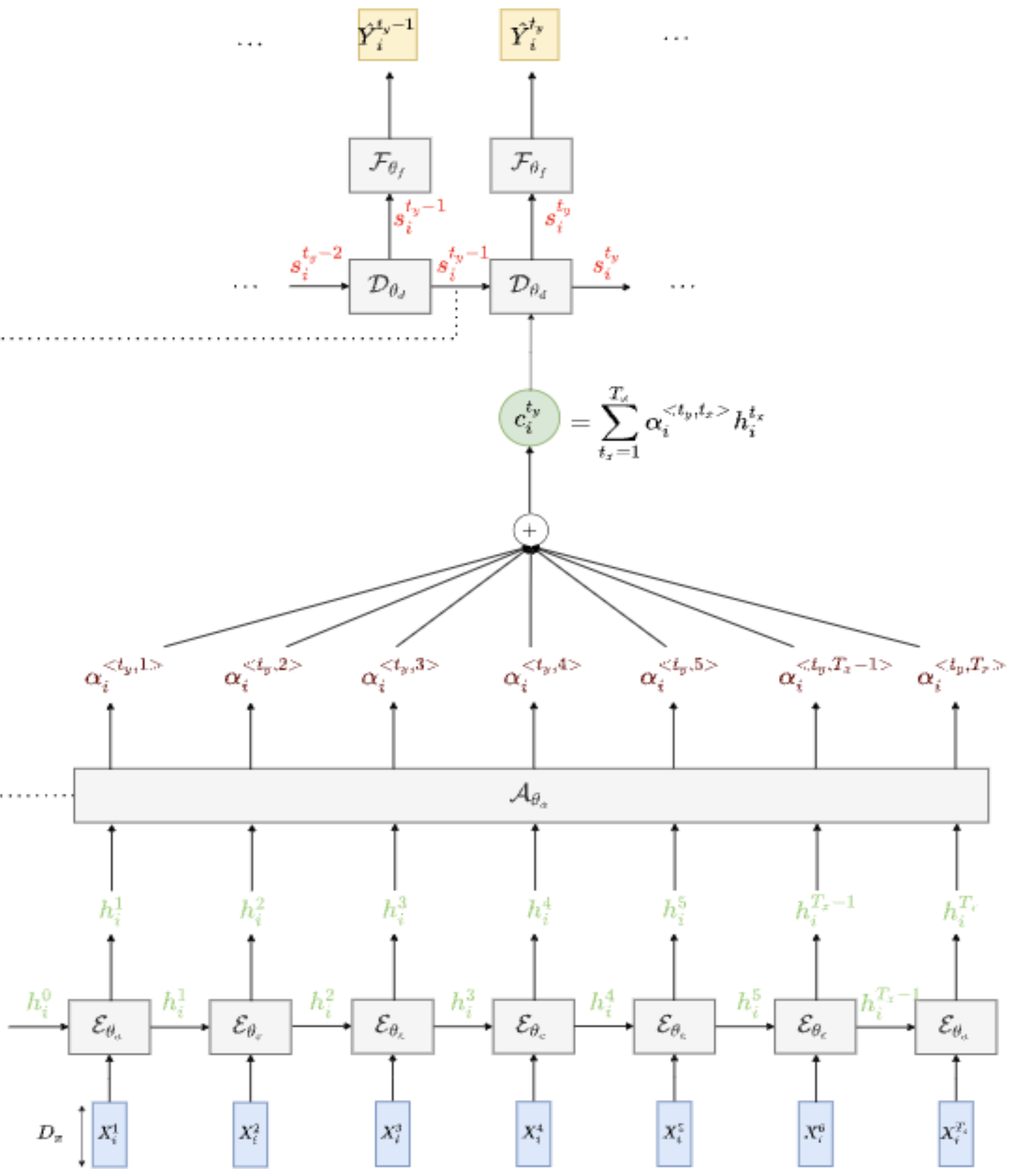


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Addressing the Challenges of Sequence Modeling with Attention Mechanisms – Part 5

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

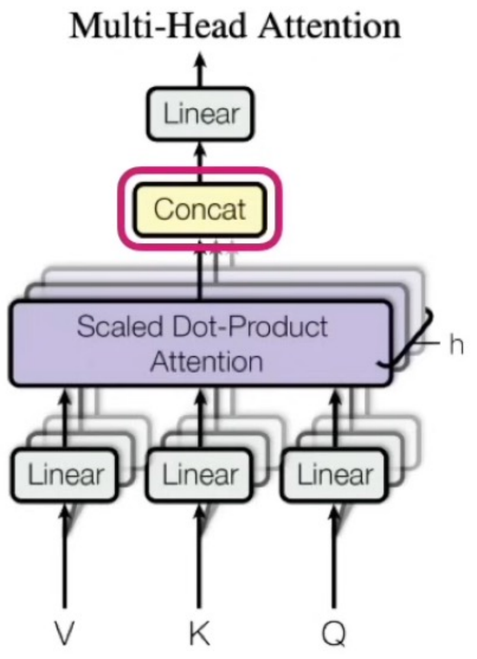
- An Encoder $\mathcal{E}_{\theta_e}$ parameterized by θ_e maps the input embeddings $(X_i^1, \ldots, X_i^{T_x})$ to the decoder hidden states $(h_i^1, \ldots, h_i^{T_x})$
- An Attention Layer $\mathcal{A}_{\theta_a}$ parameterized by θ_a is used to compute the attention weights $\alpha_i^{<t_y,t_x>}$ 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 $\hat{Y}_i^{t_y}$



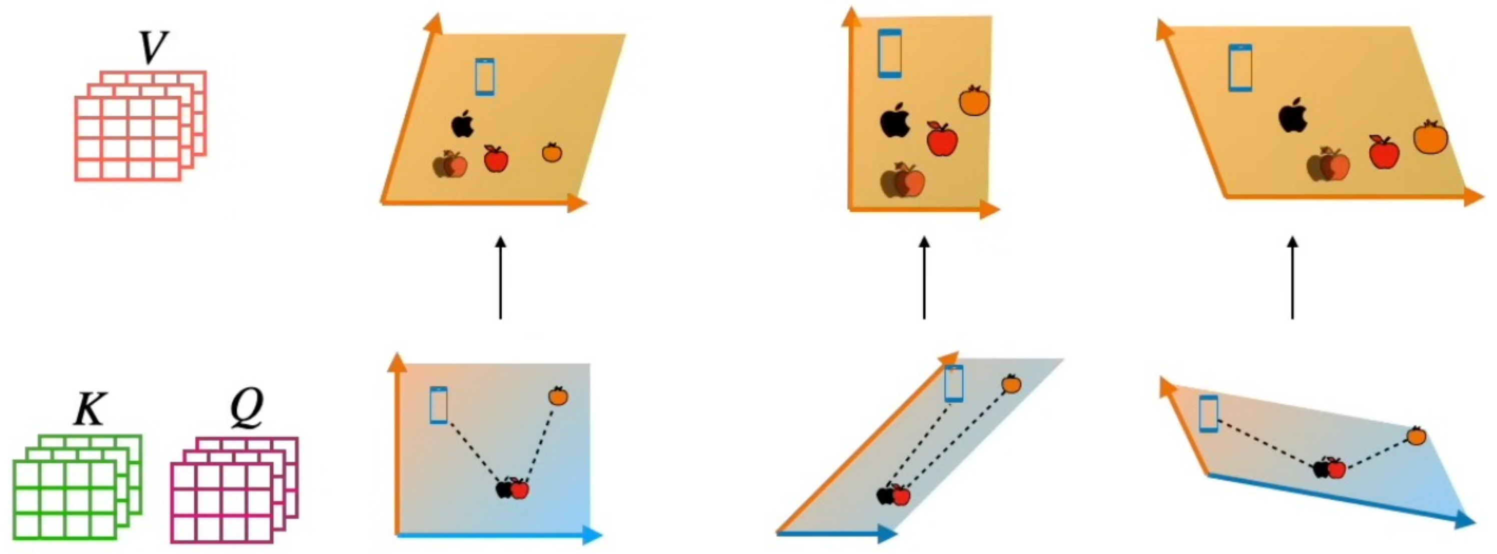
Part 5 : A Brief description of the Transformer Architecture

From Attention to Multi Head Attention – Part 1

The Intuition behind Multi-head attention



$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},...,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

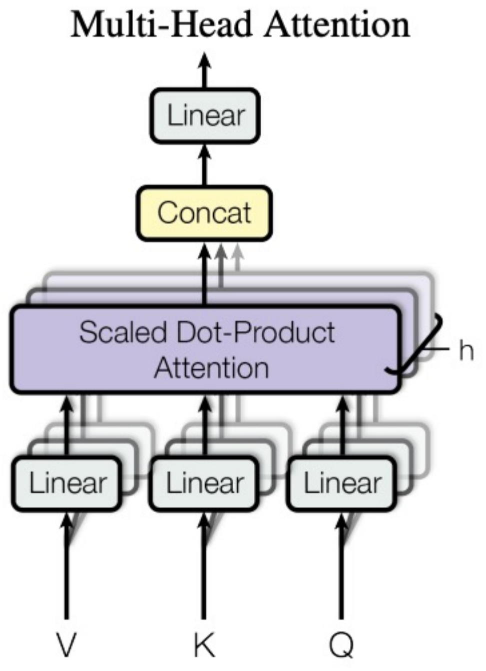


The multiple replication of the attention processes enables us to zoom on different aspects of the complex relationships.

Source: Serrano Academy

From Attention to Multi Head Attention - Part 2

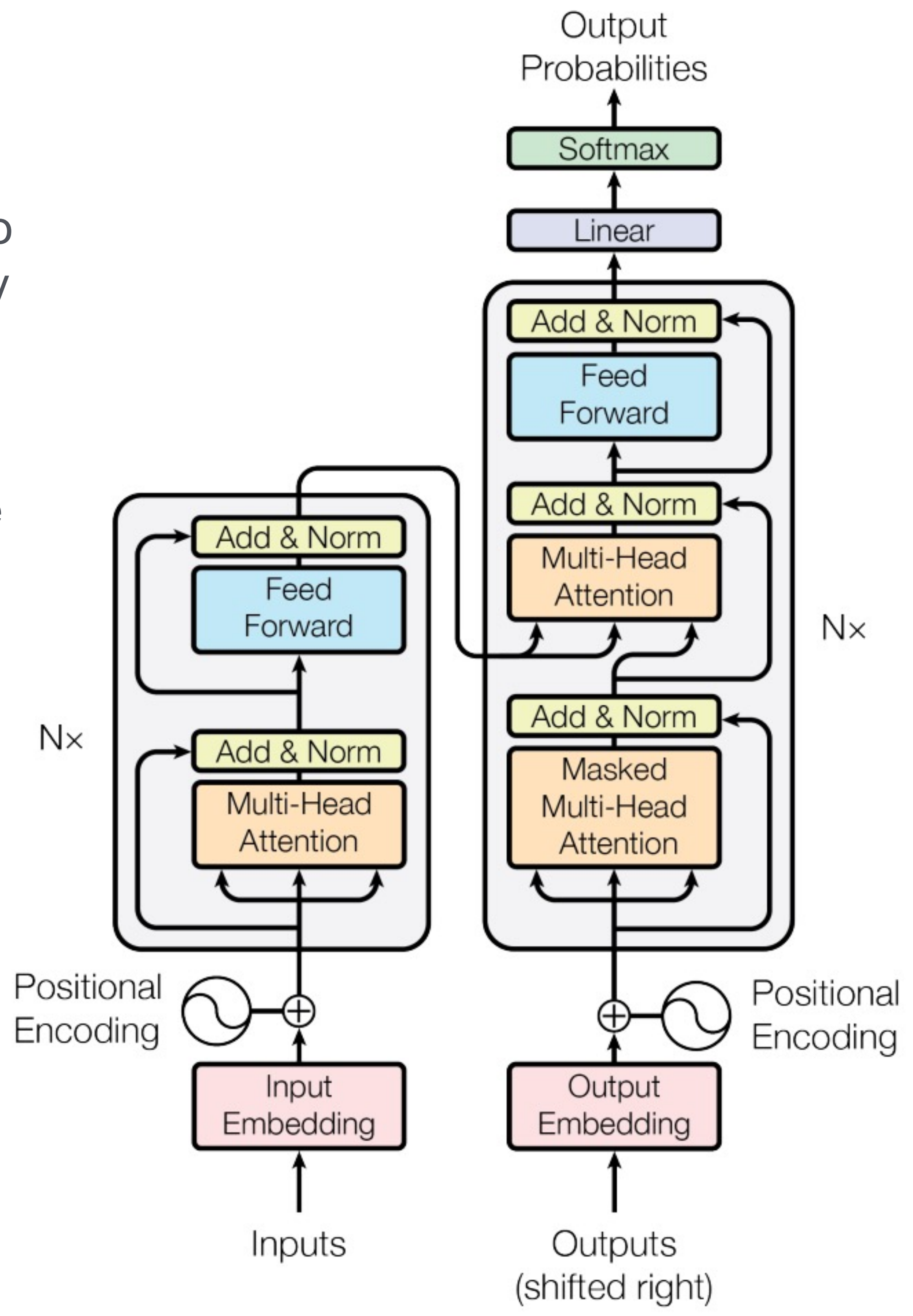
- **Scalability Enhancement**:
  - Traditional attention mechanisms often face limitations in handling complex relationships in large sequences due to computational constraints.
  - Multi-Head Attention addresses this by splitting the attention mechanism into multiple heads, allowing the model to focus on different parts of the input simultaneously, thus improving scalability.
- **Increased Representation Capacity**:
  - Multi-Head Attention facilitates capturing diverse and nuanced patterns in the data by enabling the model to attend to different aspects of the input through parallel attention heads.
  - This expanded representation capacity enhances the model's ability to capture long-range dependencies and subtle relationships, leading to improved performance in various tasks.



$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\ldots,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Transformer Architecture

- The **Transformer architecture** consists of an encoder and a decoder, each composed of multiple layers.
- **Multi-Head Attention** enhances the model's ability to focus on different parts of the input simultaneously by performing multiple attention operations in parallel.
- Transformers incorporate **positional encodings** to provide information about the position of words in the input sequence.
- Positional encodings are added to the input embeddings, allowing the model to distinguish between words based on their positions.
- **Layer normalization** helps stabilize training by normalizing the inputs to each layer, while residual connections facilitate gradient flow, mitigating the vanishing gradient problem.


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)