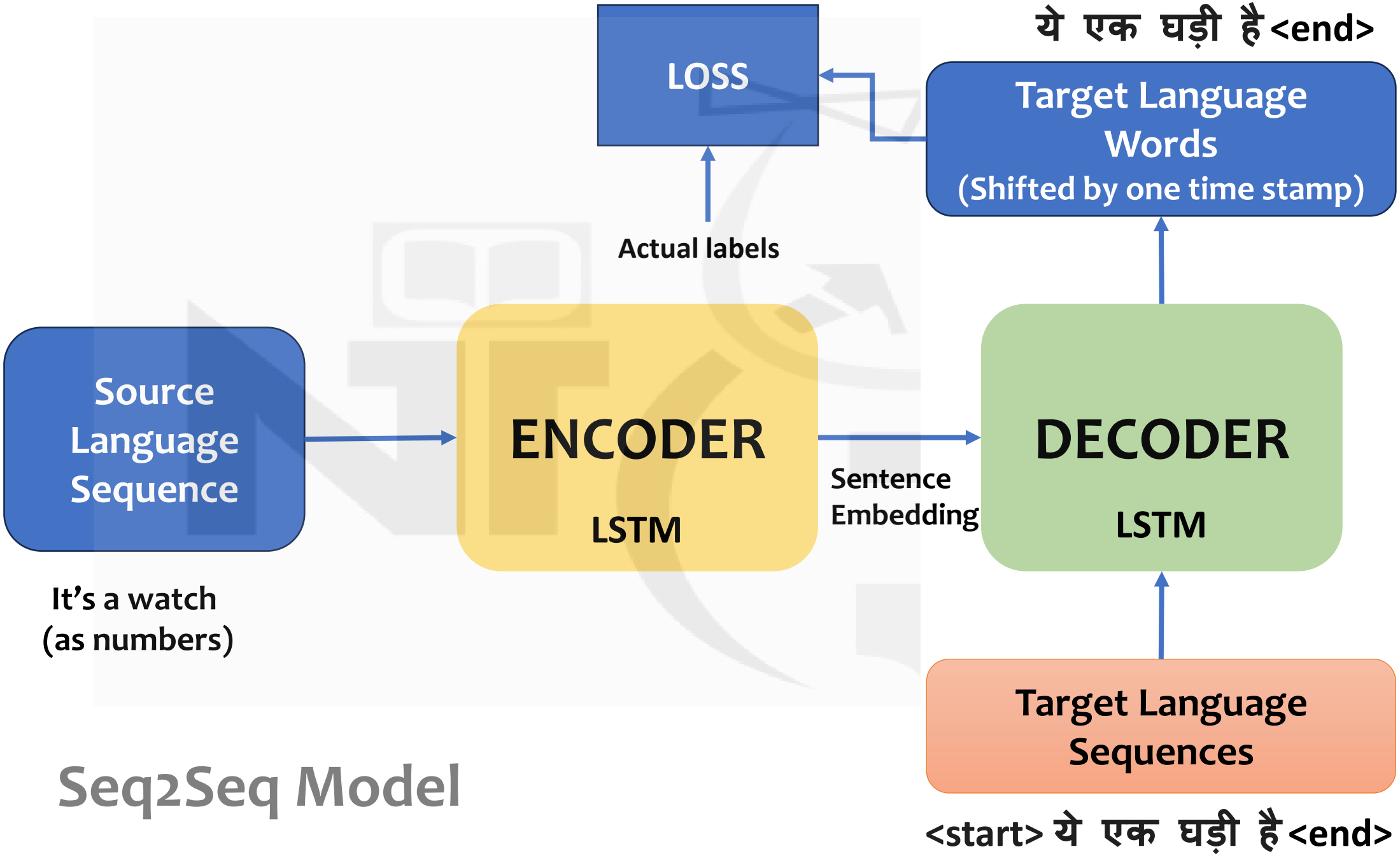
Transformers

MUKESH KUMAR



Seq2Seq Model

Key issue with LSTM

- What is the major issue with LSTM
- It cannot take advantage of parallel processing available in GPU
- And also, large data take more time because it process data word by word

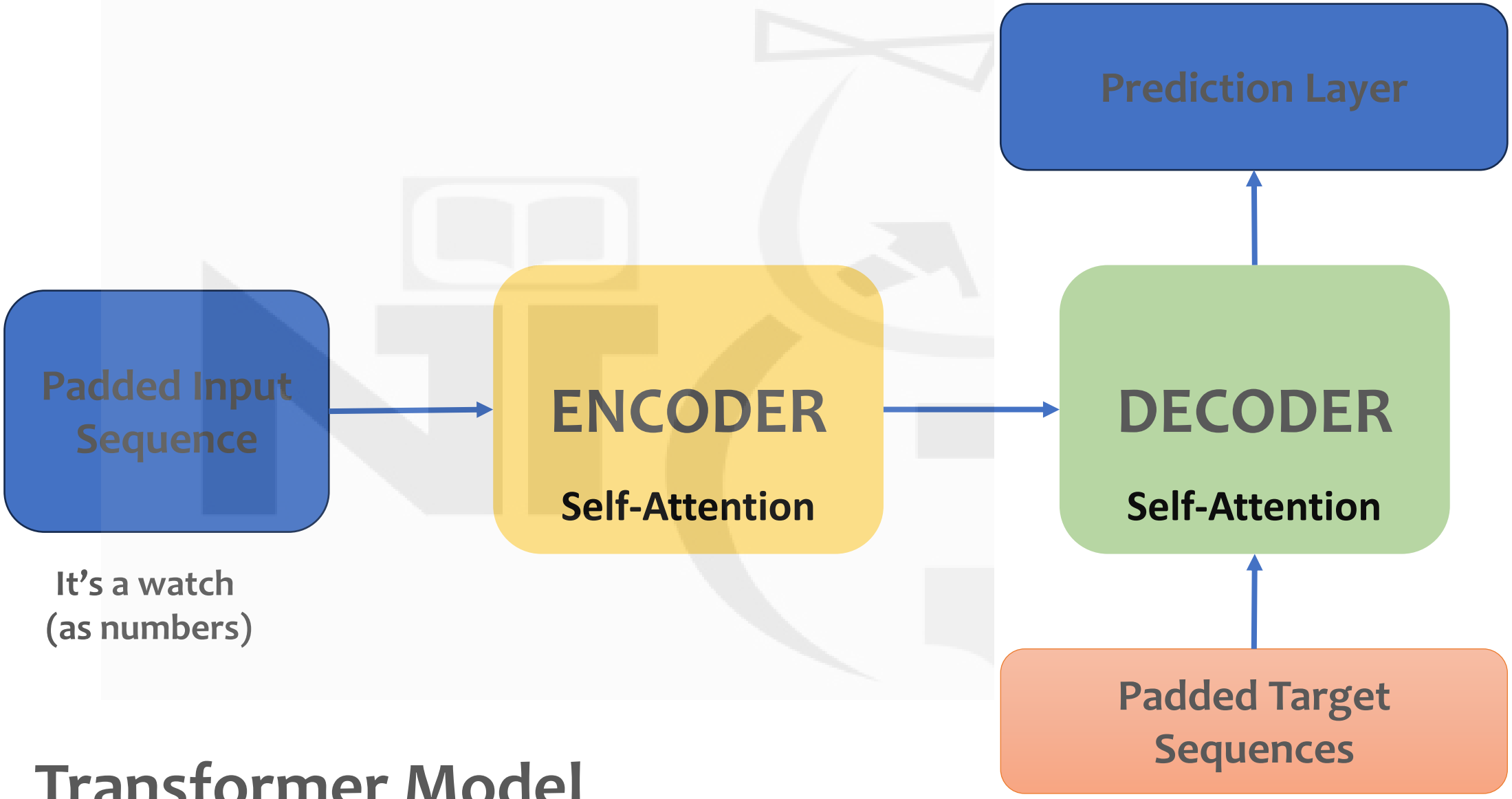
Attention is all you need

https://arxiv.org/abs/1706.03762

Moving away from LSTM

Attention

- Came around 2017
- Build without LSTM



Transformer Model

Self Attention

- Self attention layer is a dense layer

How does Encoder understand sequence in transformer model?

It tries to understand sequence using **Self Attention**

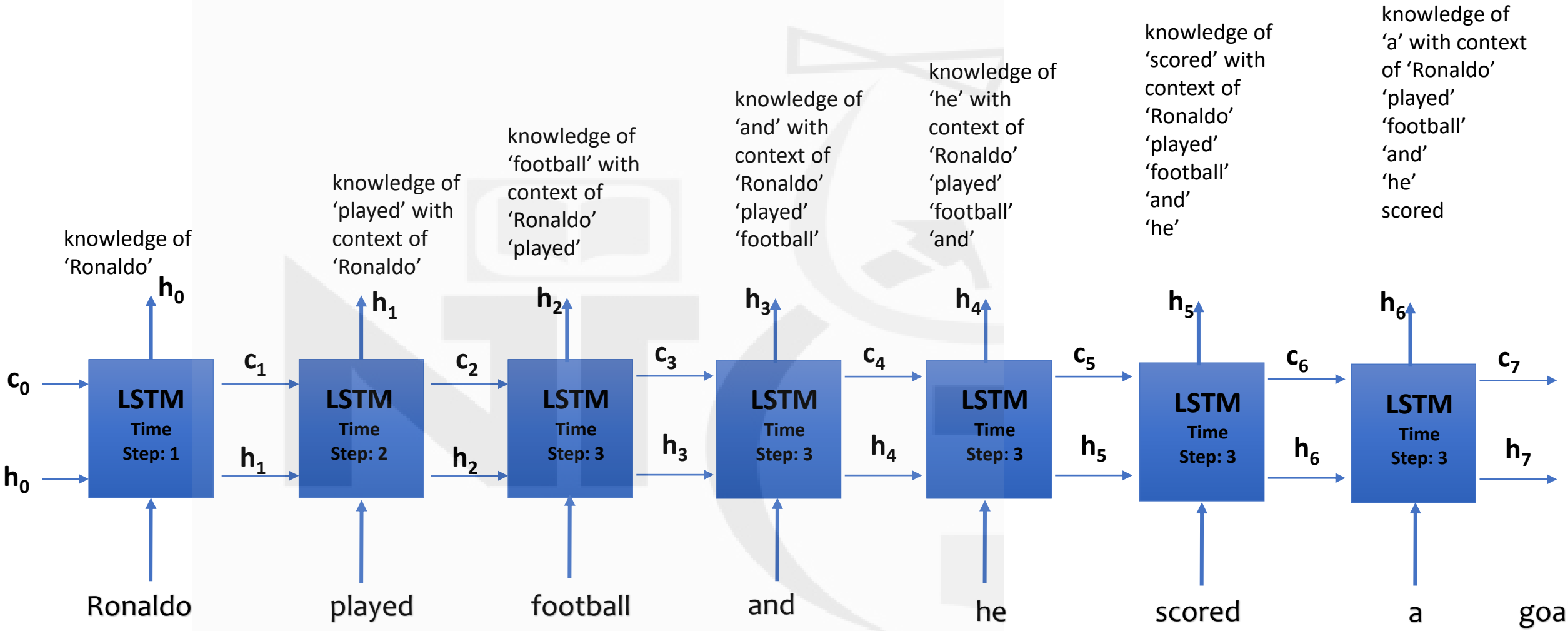
Self-Attention Vs LSTM

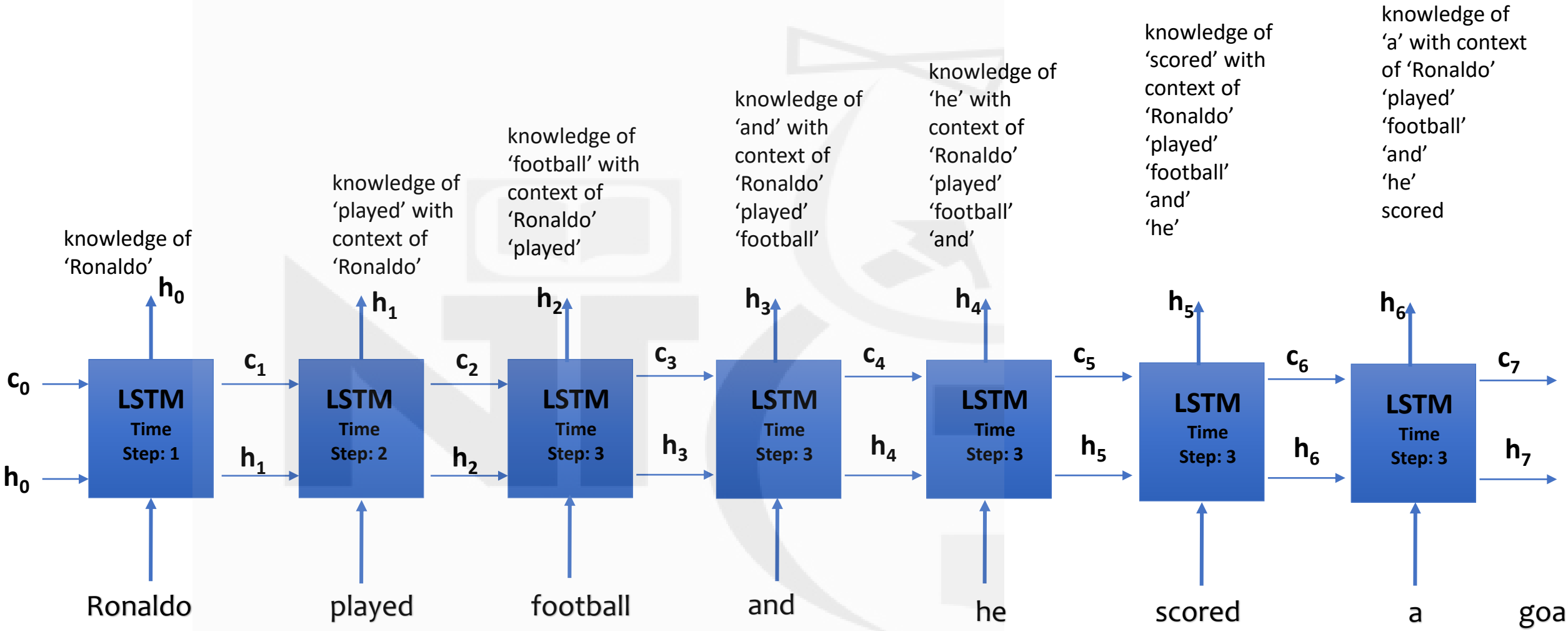
- Let’s consider an example:

Ronaldo played football and he scored a goal

How do we understand this sentence?

LSTM


knowledge of 'Ronaldo'
knowledge of 'played' with context of 'Ronaldo'
knowledge of 'football' with context of 'Ronaldo' 'played'
knowledge of 'and' with context of 'Ronaldo' 'played' 'football'
knowledge of 'he' with context of 'Ronaldo' 'played' 'football' 'and'
knowledge of 'scored' with context of 'Ronaldo' 'played' 'football' 'and' 'he'
knowledge of 'a' with context of 'Ronaldo' 'played' 'football' 'and' 'he' scored
c_0
h_0
LSTM Time Step: 1
h_0
c_1
h_1
LSTM Time Step: 2
h_1
c_2
h_2
LSTM Time Step: 3
h_2
c_3
h_3
LSTM Time Step: 3
h_3
c_4
h_4
LSTM Time Step: 3
h_4
c_5
h_5
LSTM Time Step: 3
h_5
c_6
h_6
LSTM Time Step: 3
h_6
c_7
h_7
Ronaldo
played
football
and
he
scored
a
goal



LSTM builds a memory (hidden and cell state) as it's new data elements in the sequence. This allows LSTM to understand individual data elements e.g words based on the context.

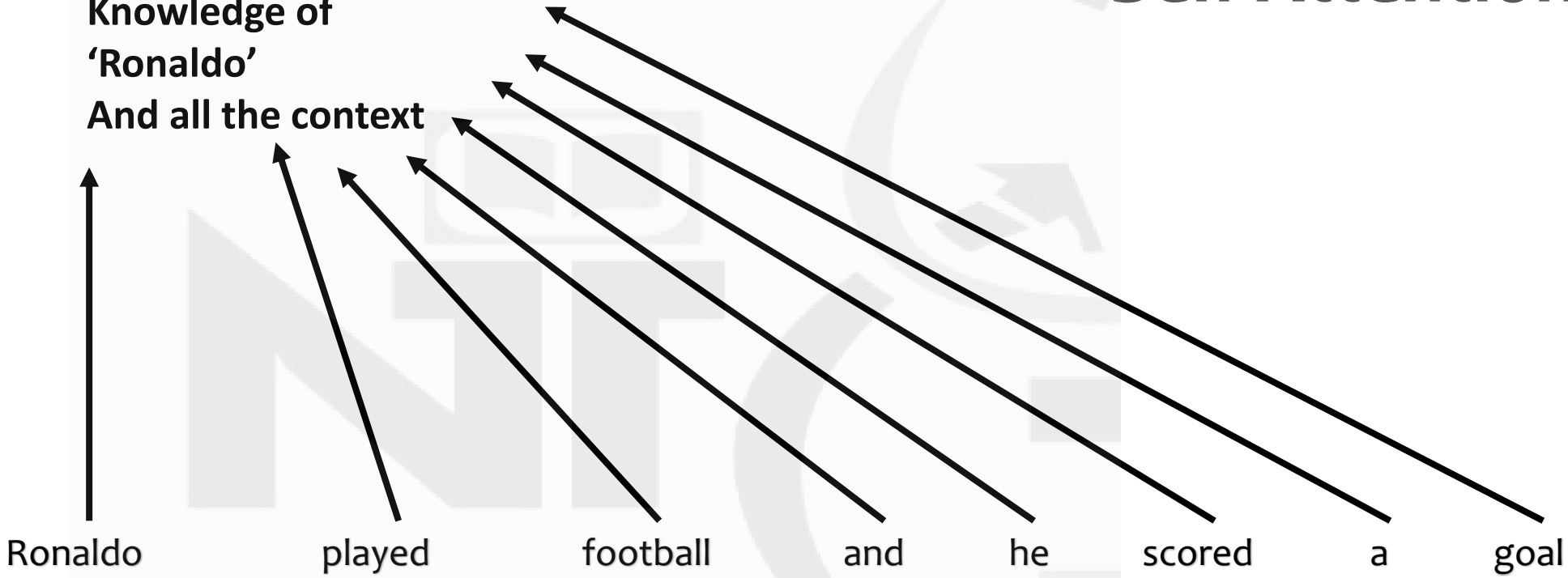
LSTM looks at Sequence one step at a time!!!

How do we get ‘hidden State’ type on information in Transformer Encoder

We will need to accomplish the same without ‘memory’

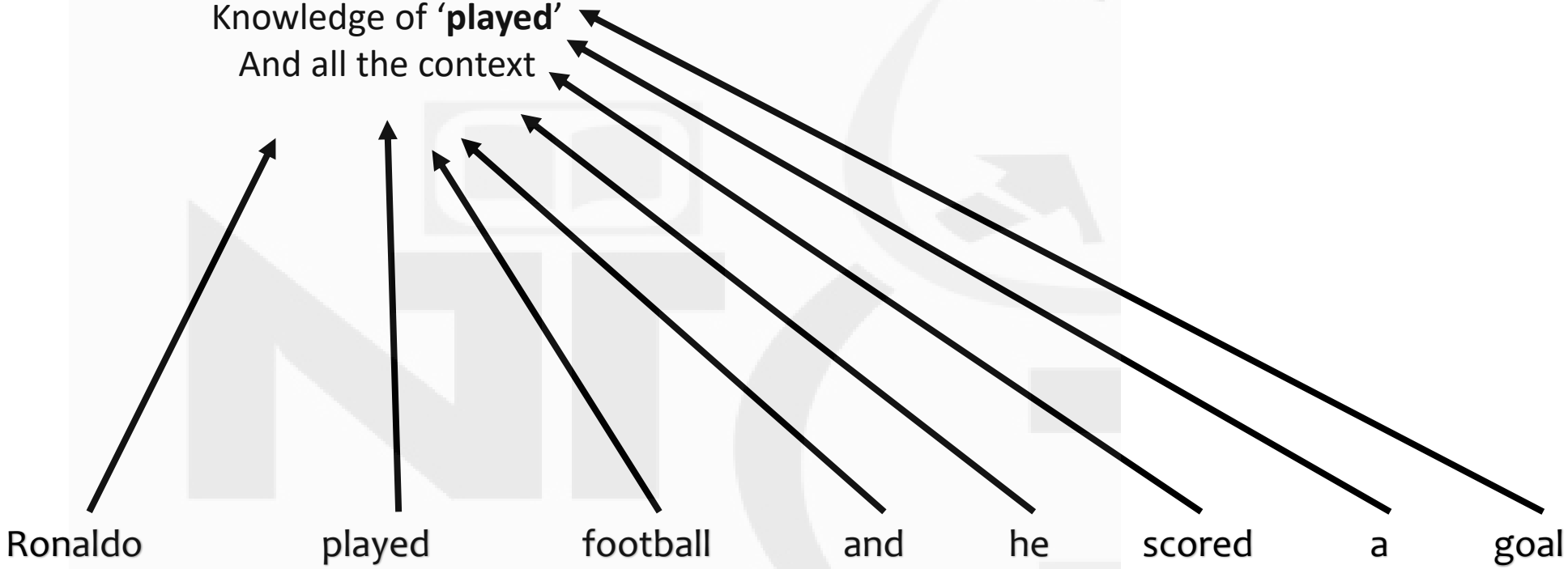
Self-Attention

Self Attention



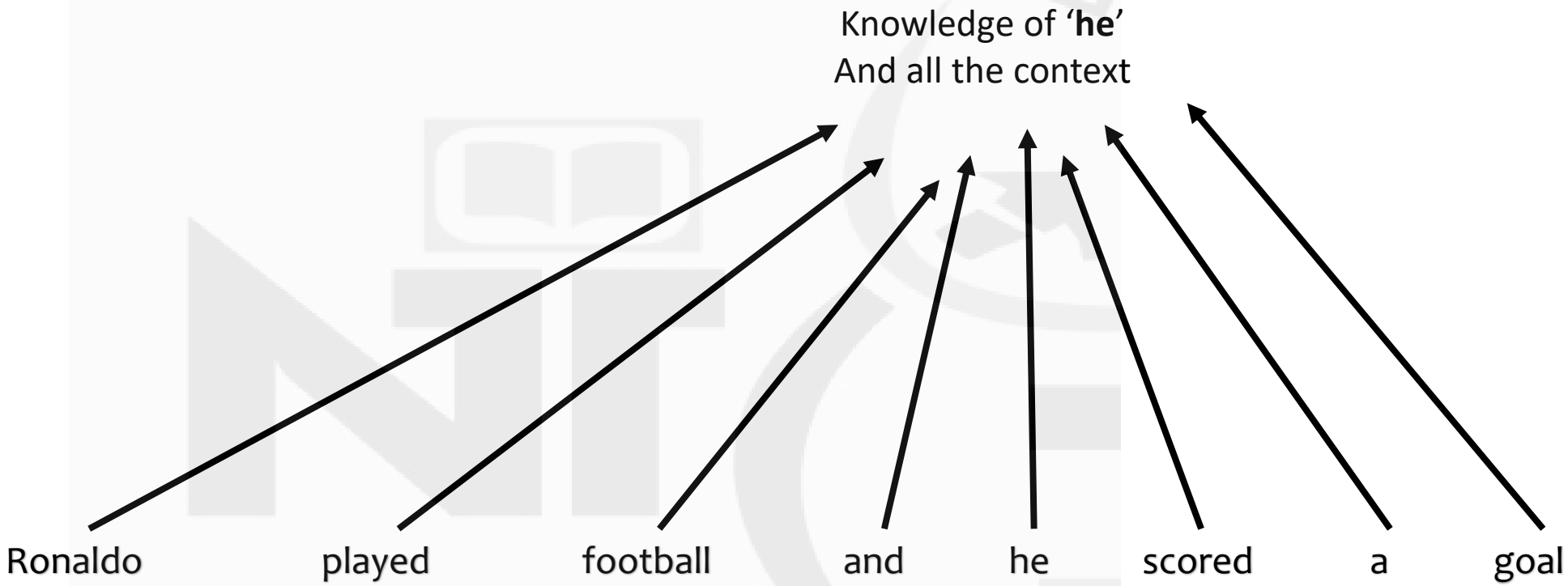
To create encoding for word 'Ronaldo' Self-Attention looks at all the words in the sequence

Self Attention



To create encoding for word 'played'
Self-Attention looks at all the words in the sequence

Self Attention



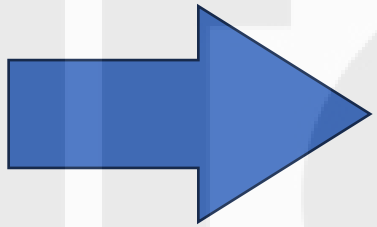
Great news encoding for each word in sequence can be done in parallel

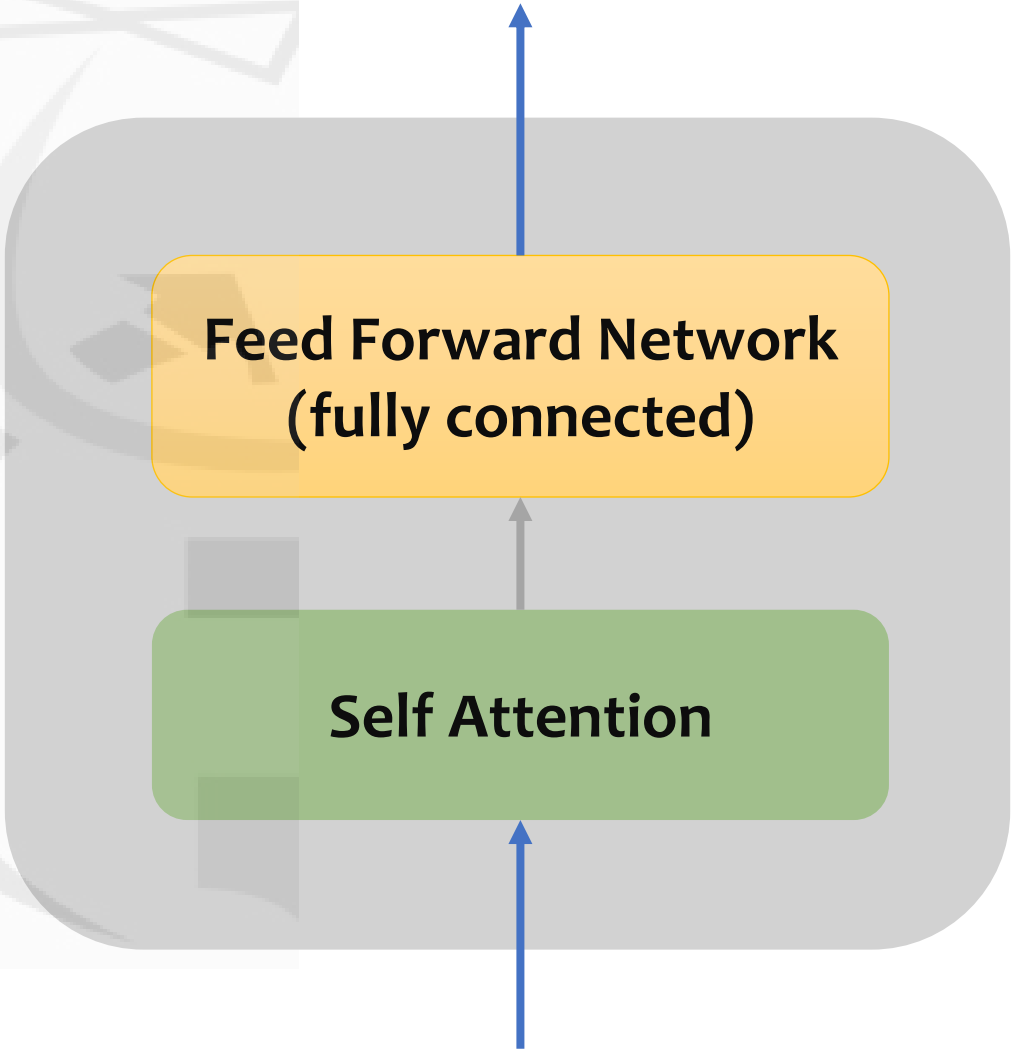
Understanding Encoder

Transformer Model

ENCODER

With Dense layers





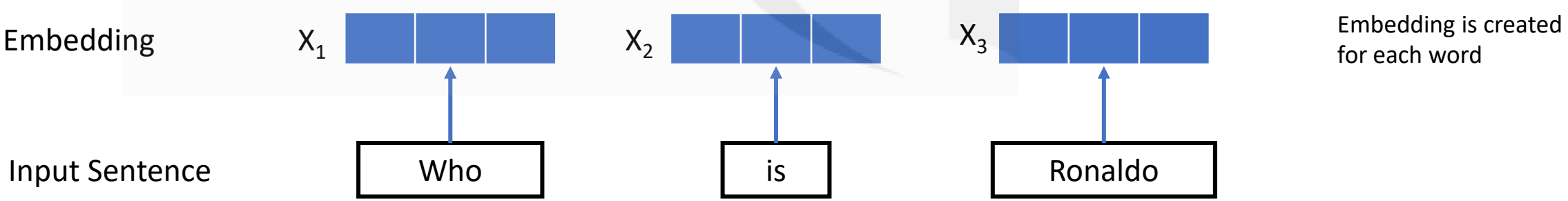
In Transformer model, Encoder is made up of two types of layers as shown on right

Implementing Self Attention

Implementing Self Attention

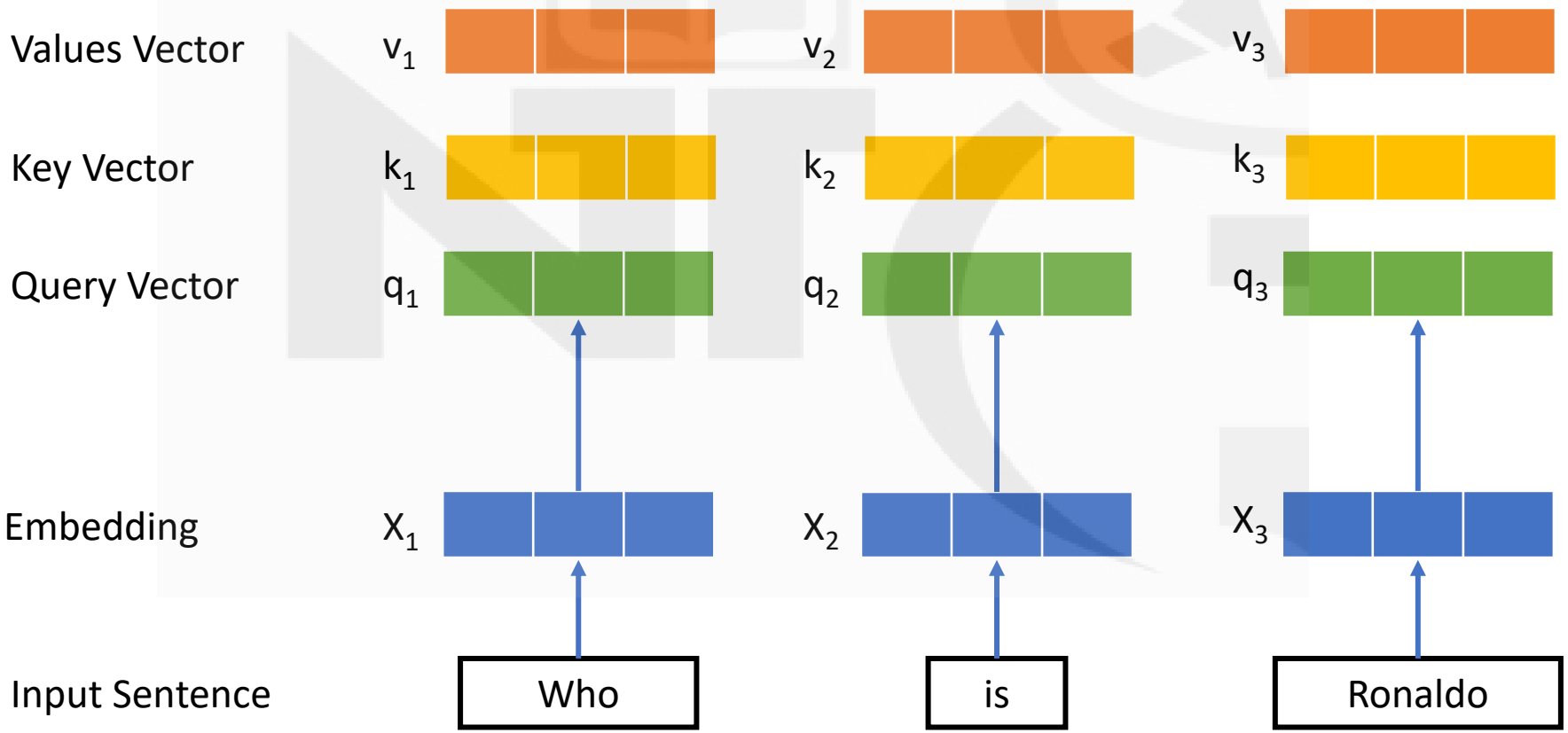
Input Sentence | Who | is | Ronaldo

Implementing Self Attention



Embedding is created for each word

Implementing Self Attention

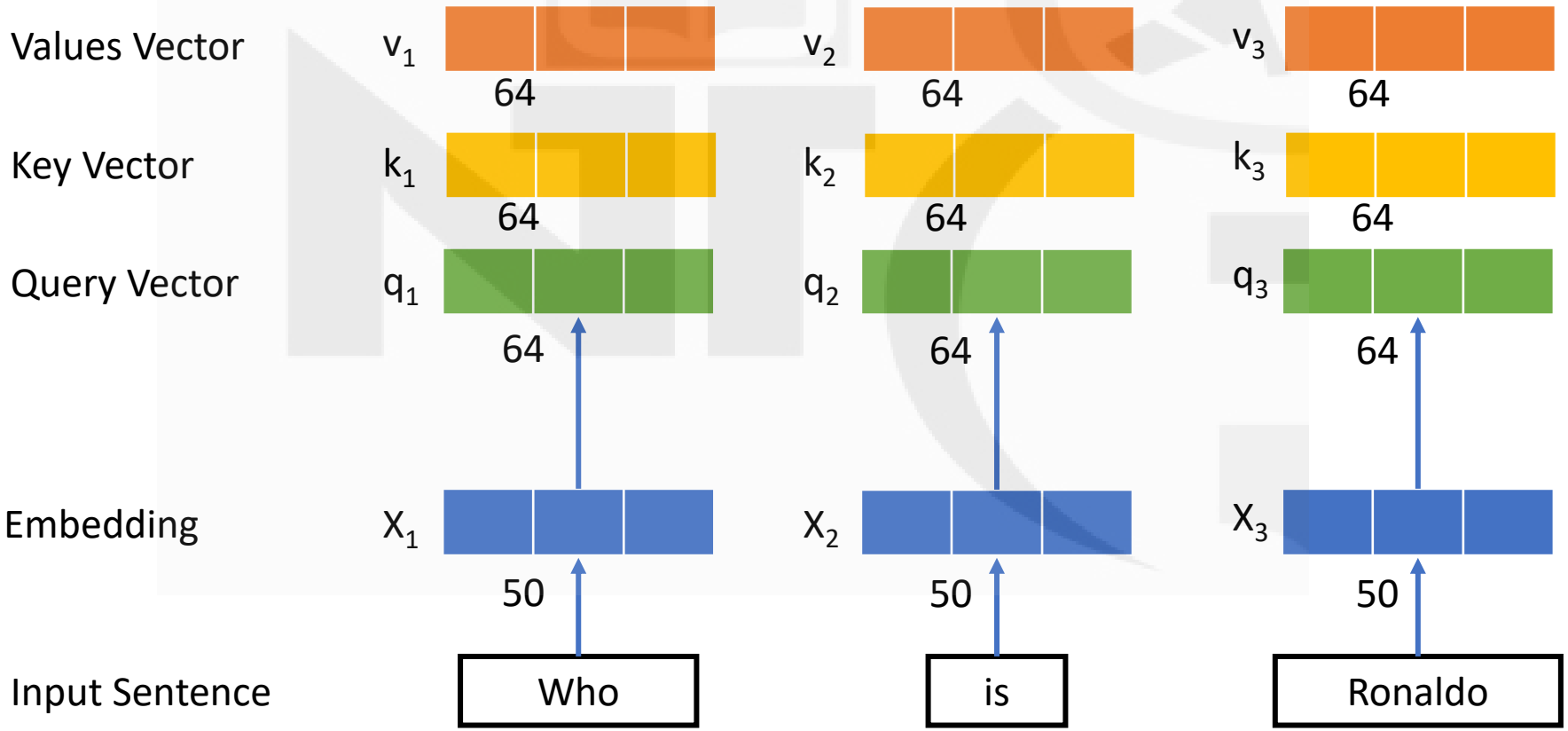


For each word, 3 vectors are created using the embedding as input. These vectors are created using 3 different Dense layers

Embedding is created for each word

Implementing Self Attention

Say embedding =50, and
the size of vectors =64

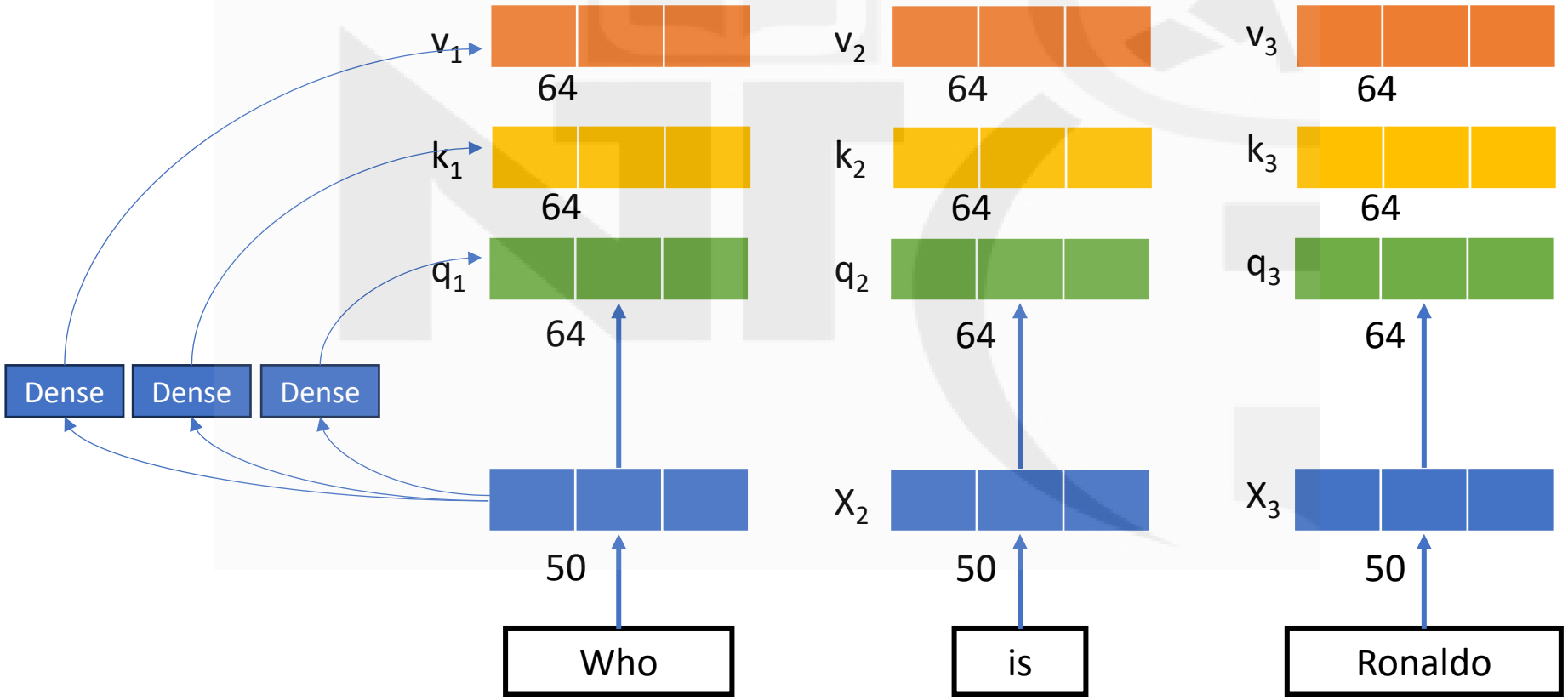


For each word,
3 vectors are created using the embedding as input. These vectors are created using 3 different Dense layers

Embedding is created for each word

Implementing Self Attention

How do we convert 50 num embedding to 64 number vectors



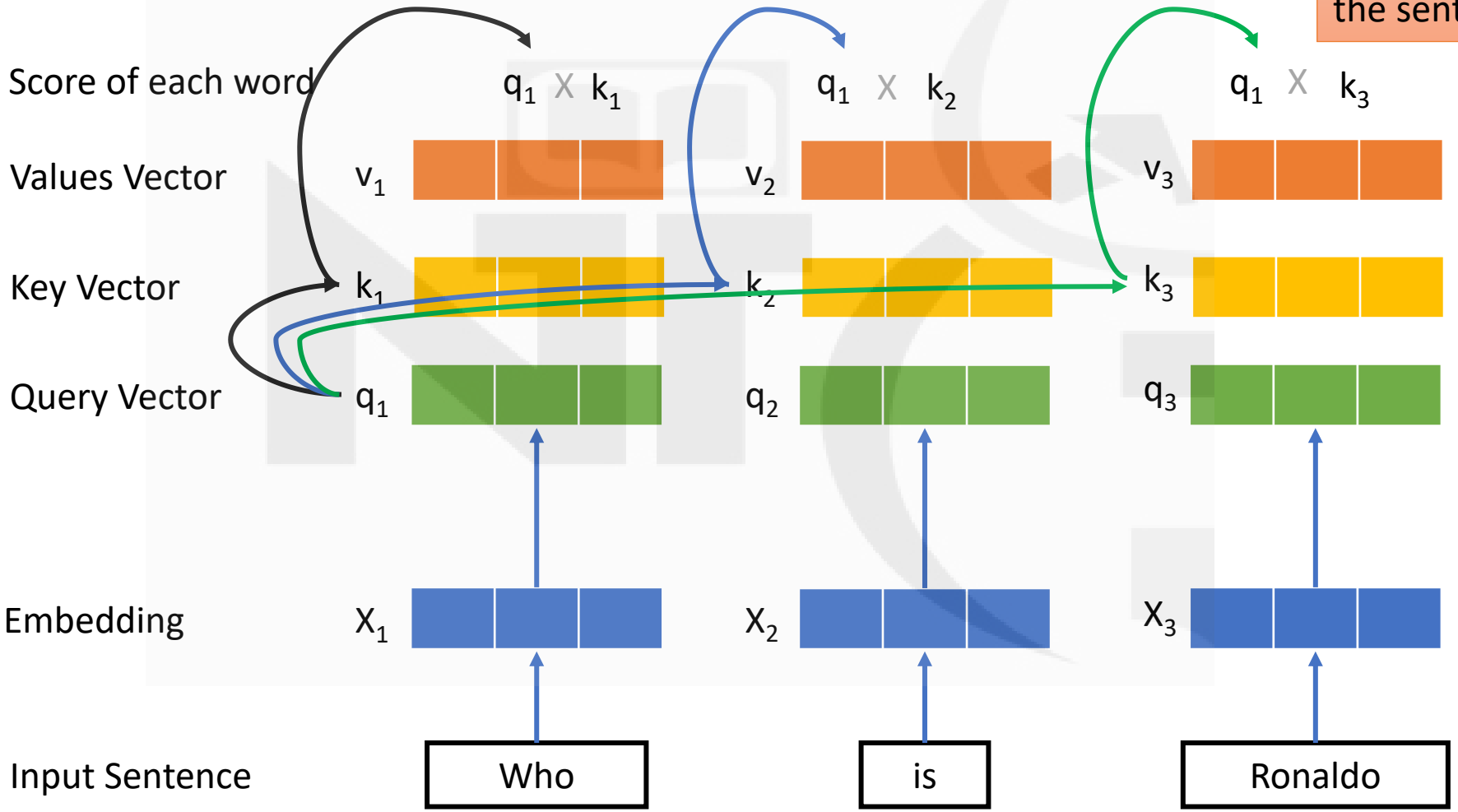
For each word,
3 vectors are created using the embedding as input. These vectors are created using 3 different Dense layers

Embedding is created for each word

Implementing Self Attention

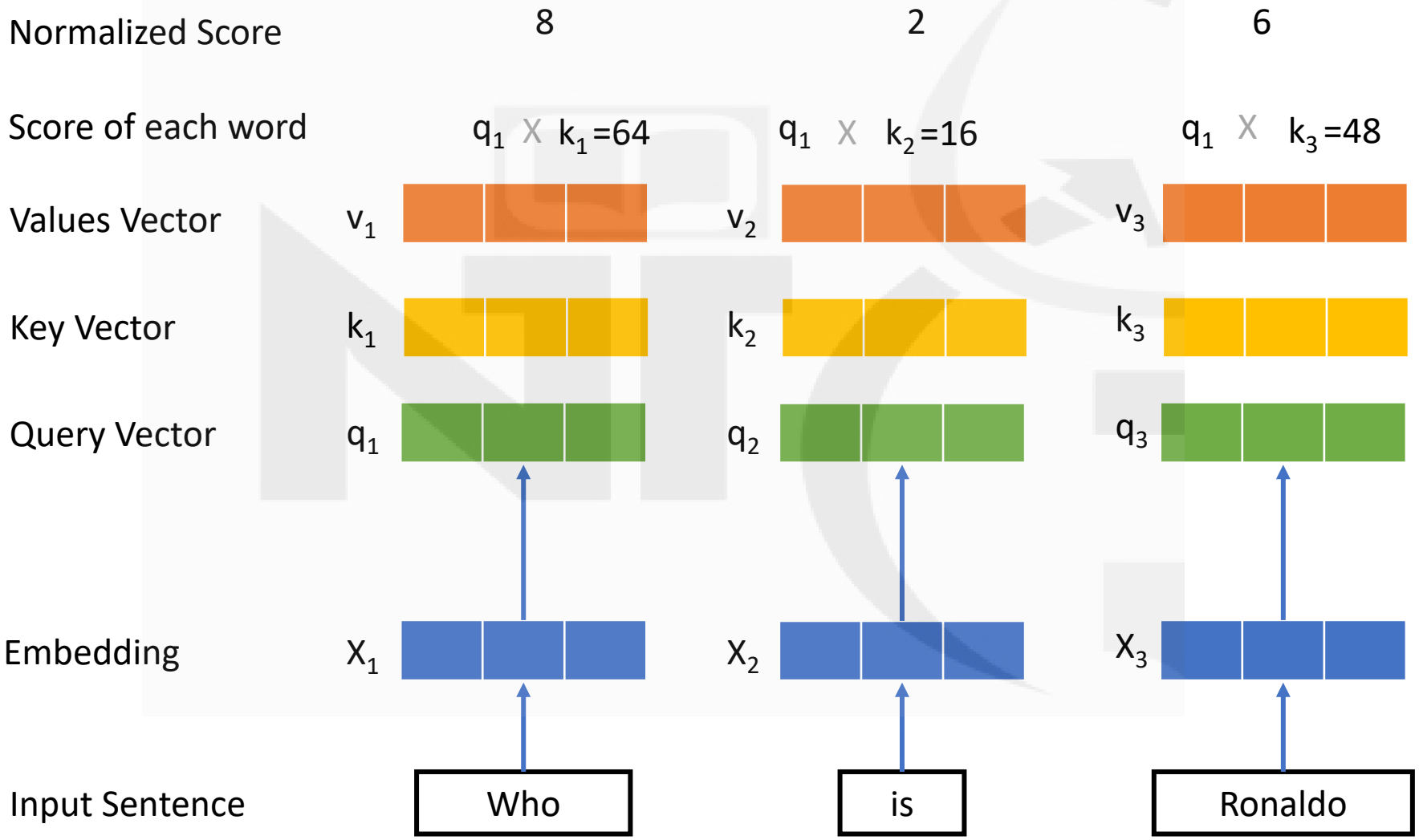
We multiply Query and key vector to check if there is a match

Dot product of query vector of 1st word and key vector of each word in the sentence.



Calculating Self Attention of First word

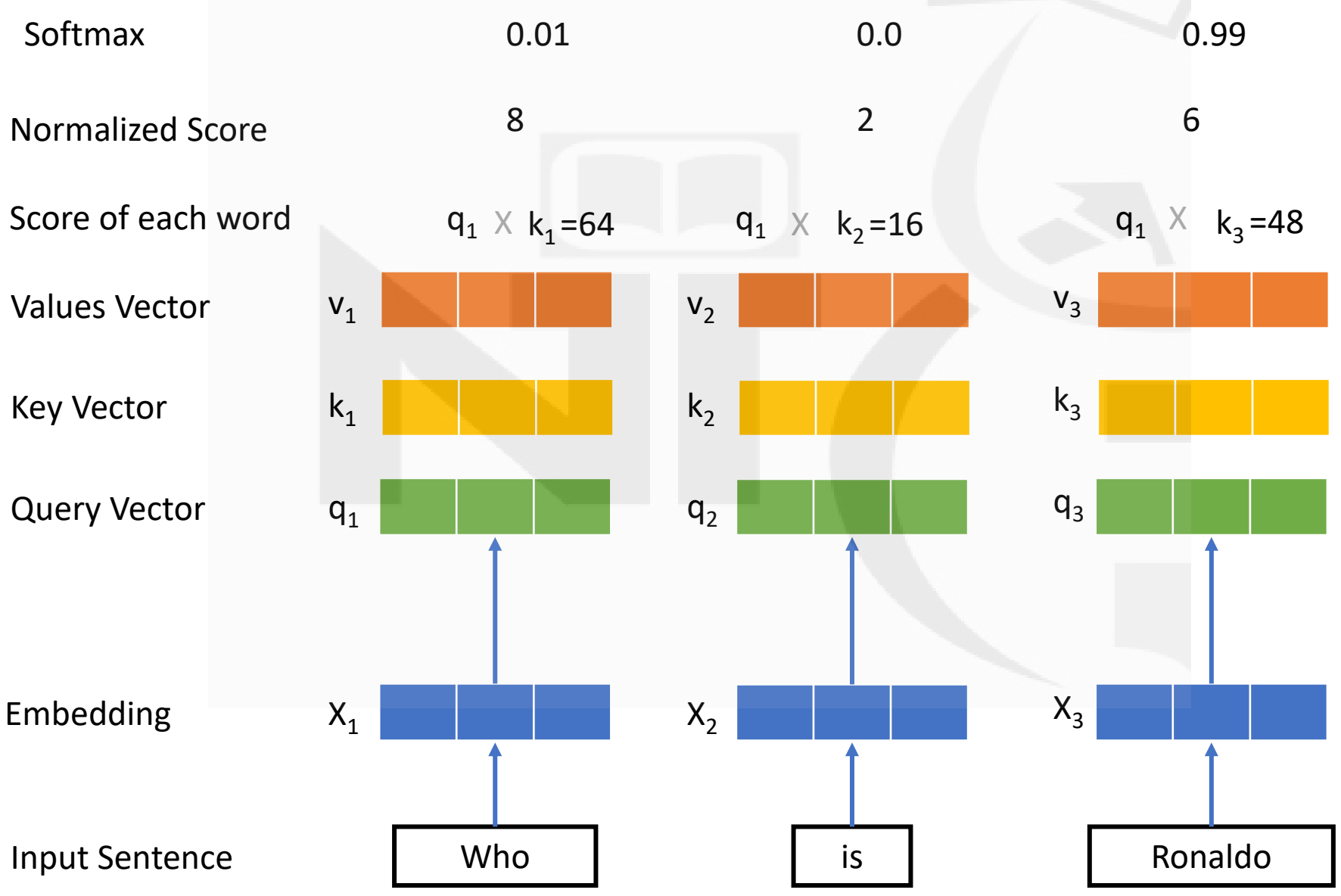
Implementing Self Attention



Scores are divided by square root of length of key vector e.g 8 (square root of 64)

Calculating Self Attention of First word

Implementing Self Attention



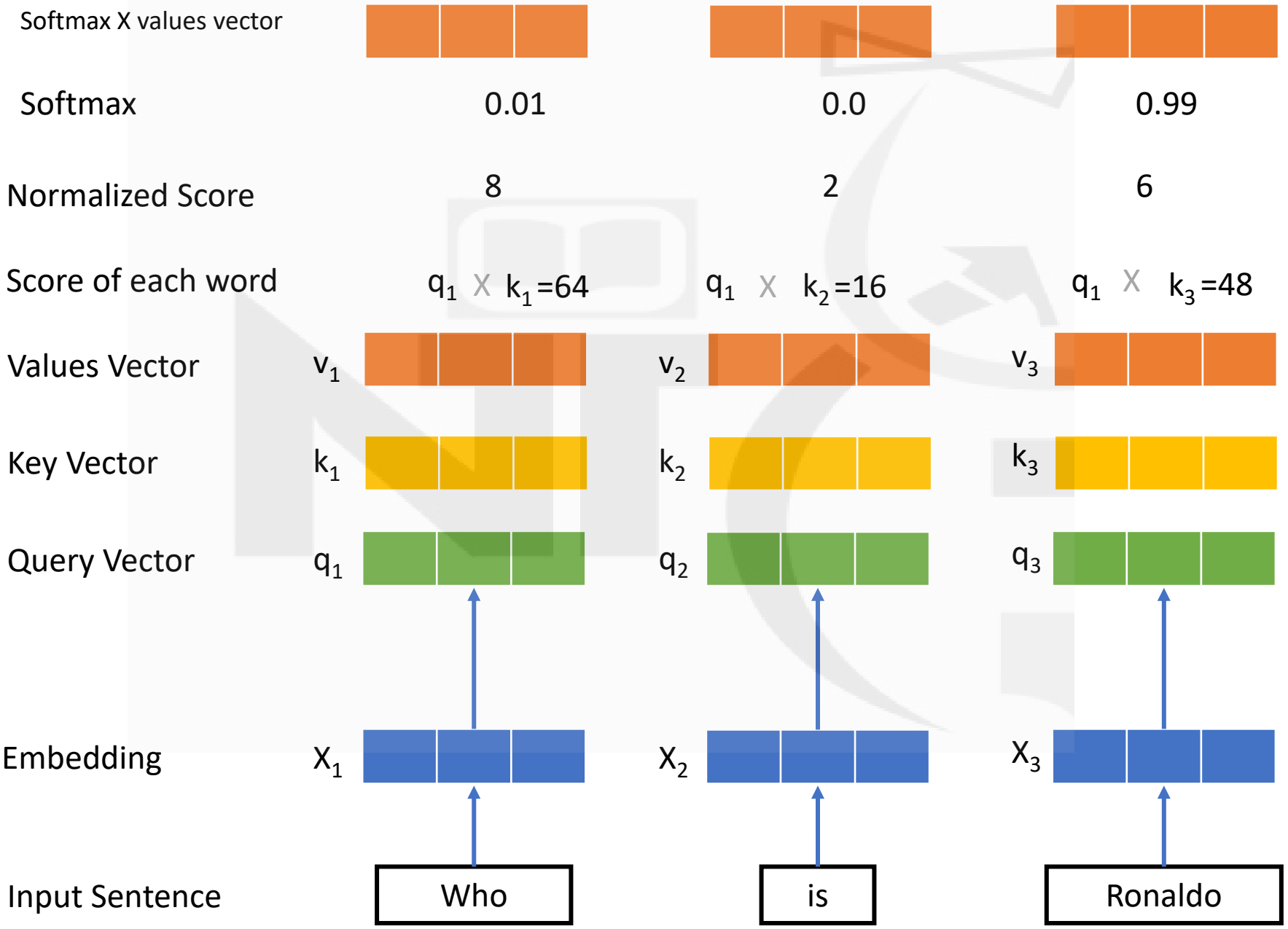
Values for each word are multiplied with respective softmax score

Softmax provides relative importance of each word for 1st word

Scores are divided by square root of length of key vector e.g 8 (square root of 64)

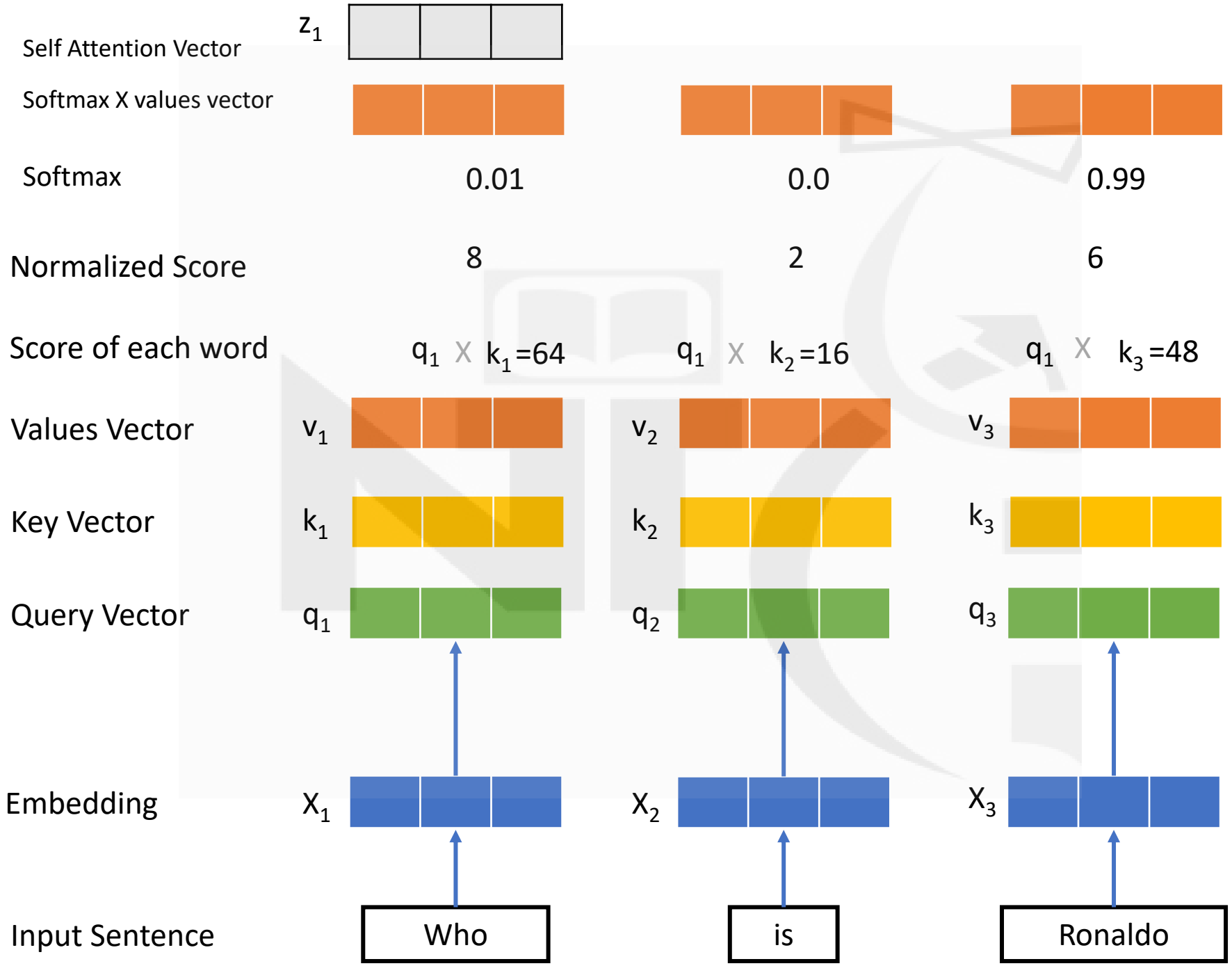
Dot product of query vector of 1st word and key vector of each word in the sentence.

Calculating Self Attention of First word



Scores are divided by square root of length of key vector e.g 8 (square root of 64)

Calculating Self Attention of First word



Values for each word are multiplied with respective softmax score

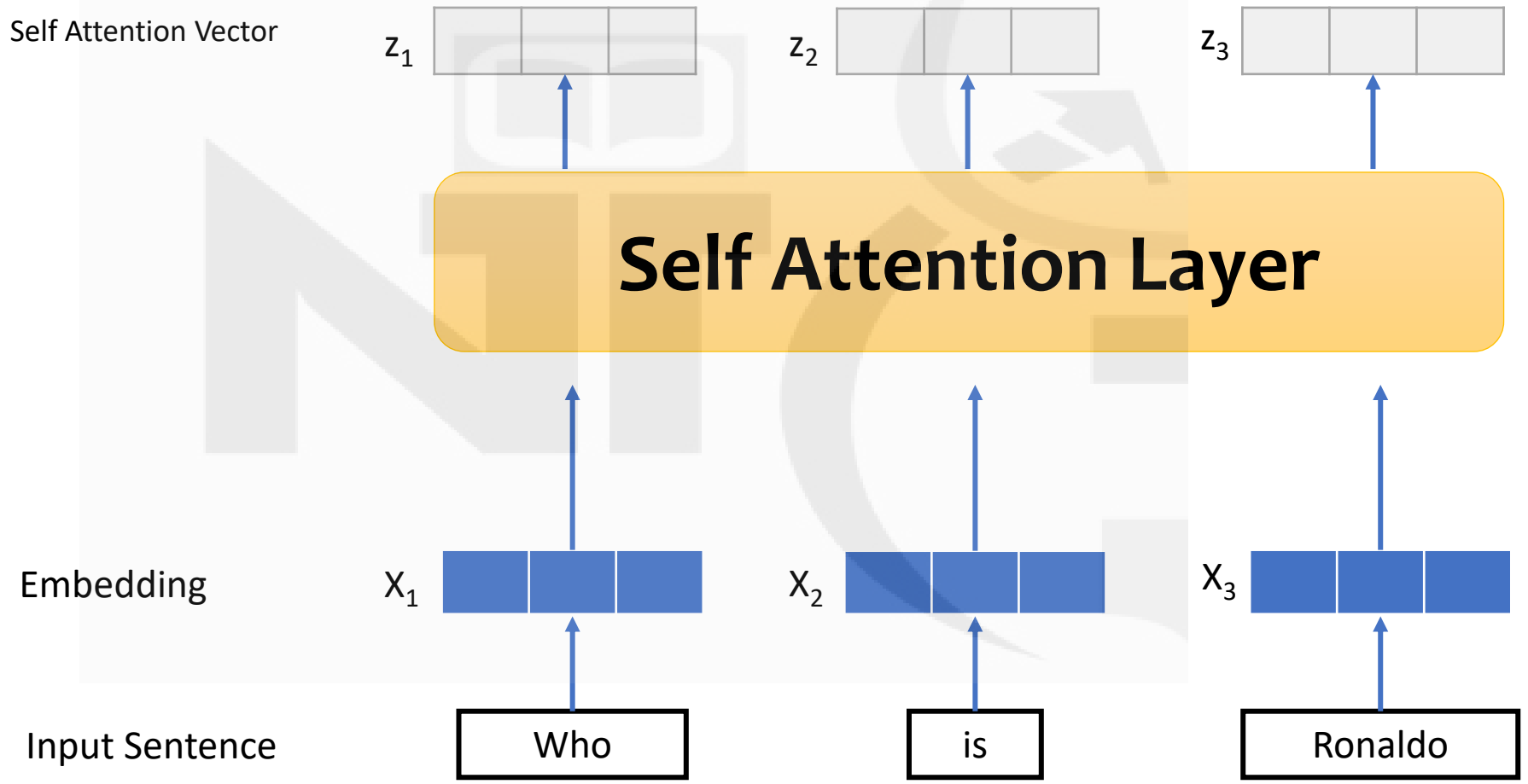
Softmax provides relative importance of each word for 1st word

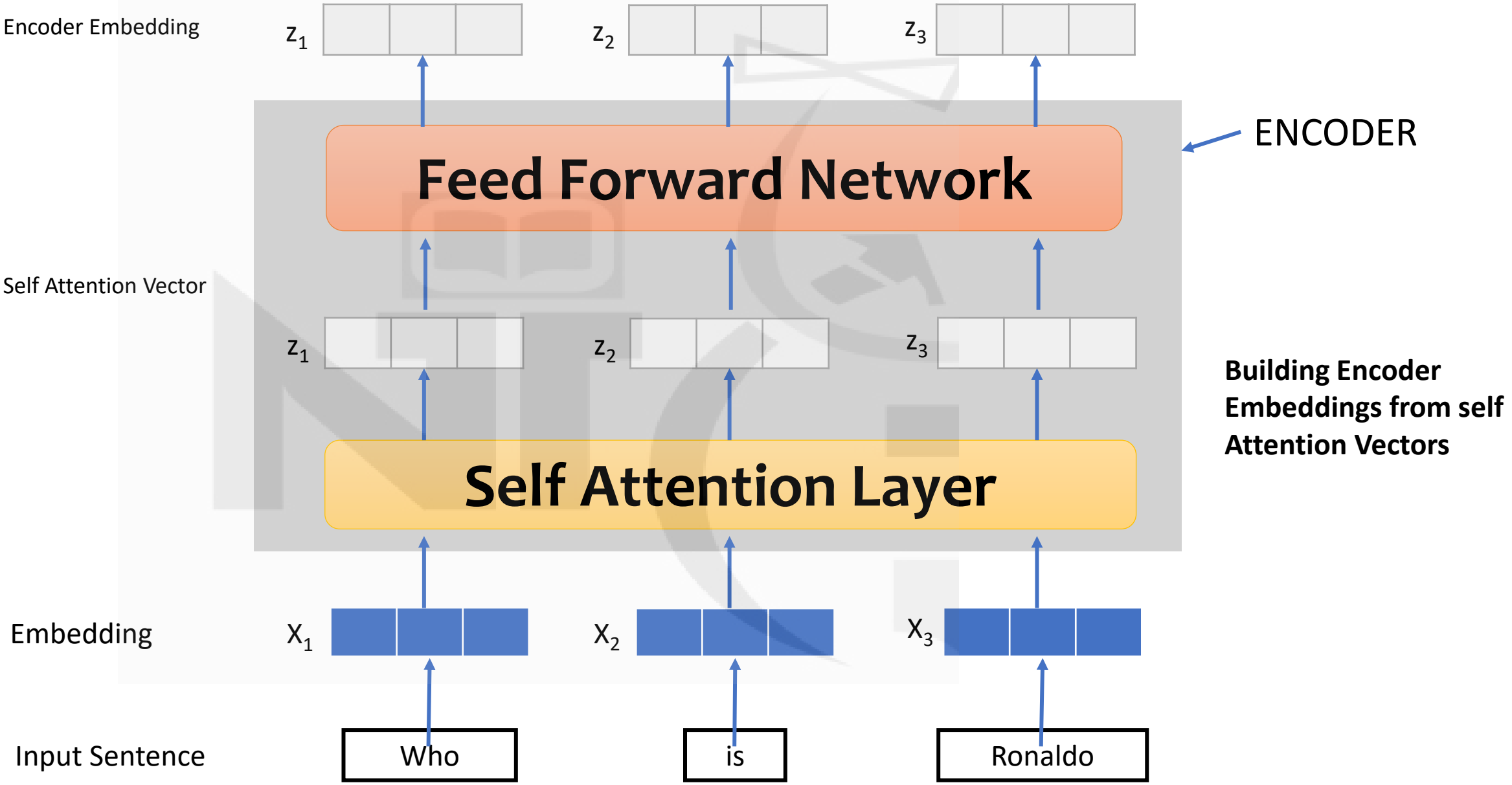
Scores are divided by square root of length of key vector e.g 8 (square root of 64)

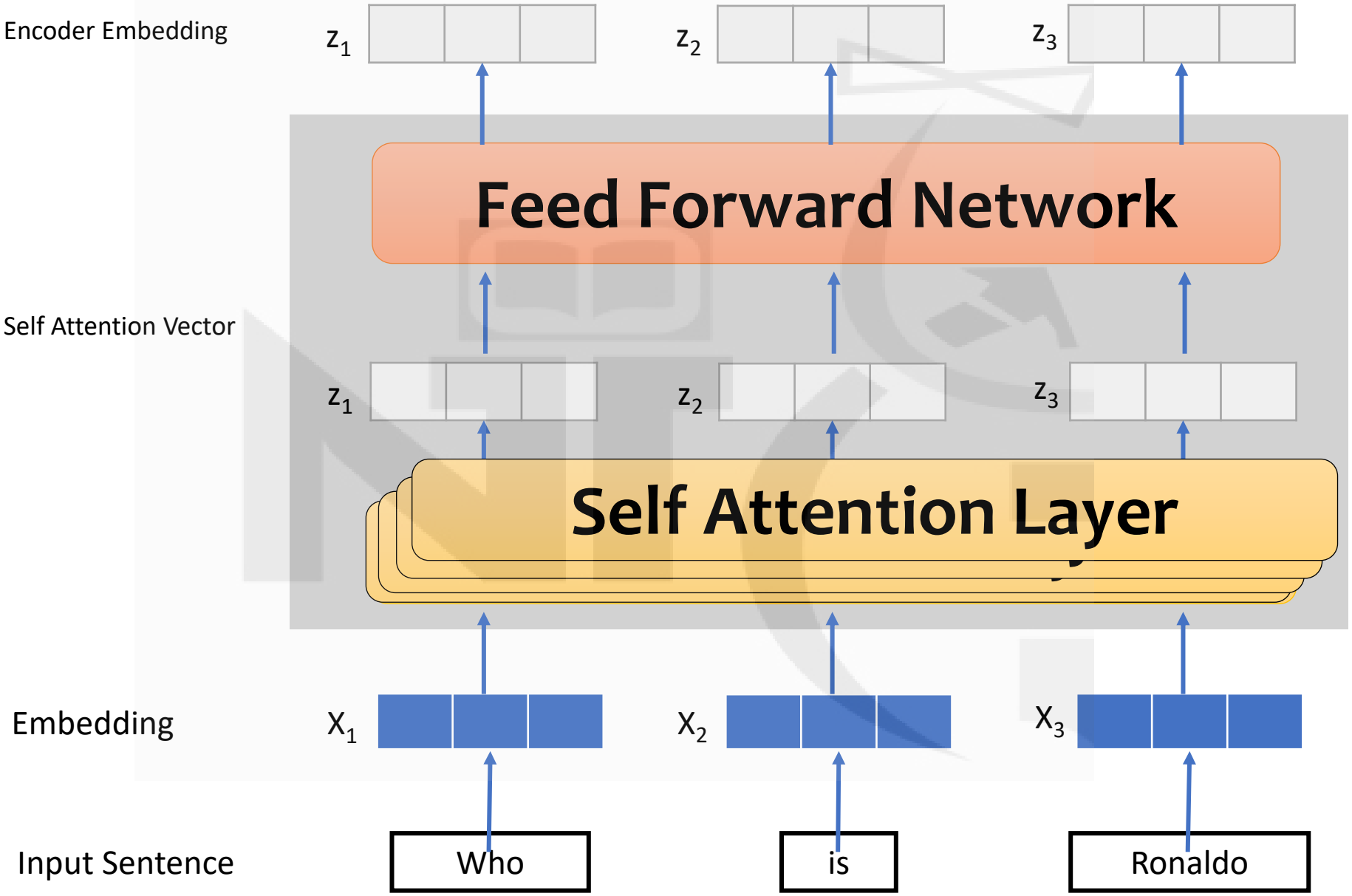
Dot product of query vector of 1st word and key vector of each word in the sentence.

Calculating Self Attention of First word

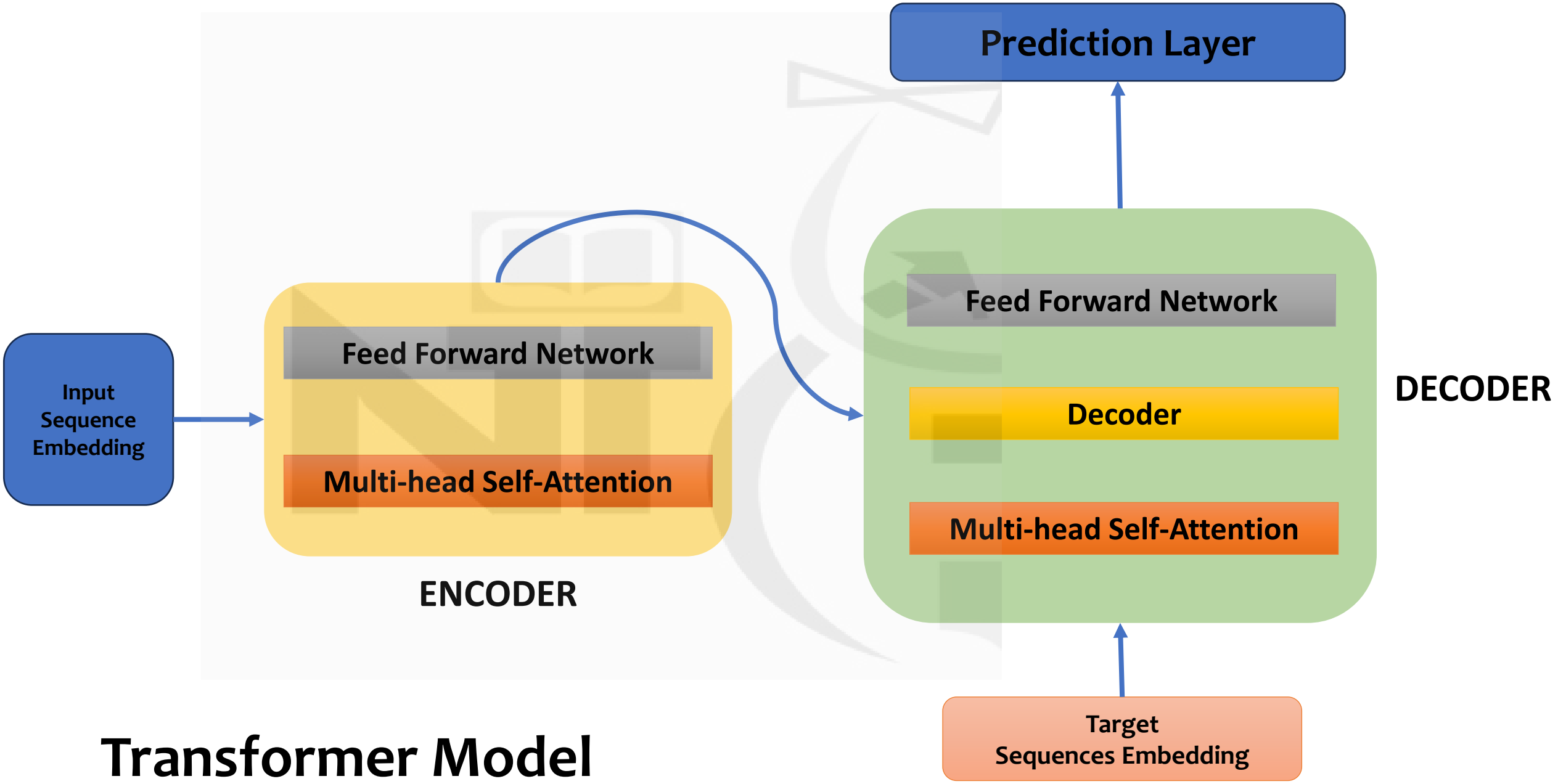
Repeat the process to calculate the self Attention vectors of all the words




Encoder Embedding
z_1
z_2
z_3
ENCODER
Feed Forward Network
Self Attention Vector
z_1
z_2
z_3
Building Encoder Embeddings from self Attention Vectors
Self Attention Layer
Embedding
X_1
X_2
X_3
Input Sentence
Who
is
Ronaldo



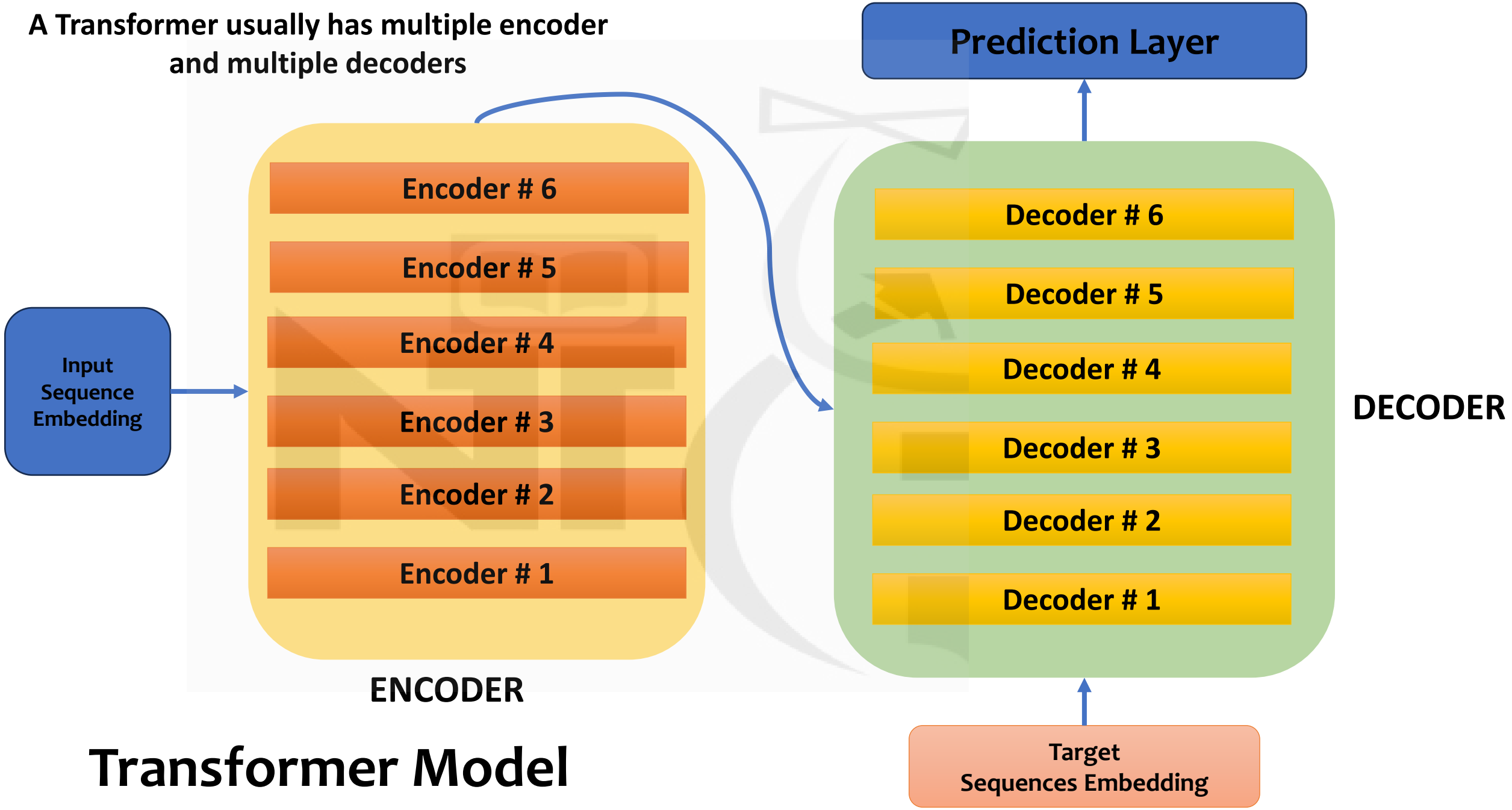
Encoder uses multiple Self-Attention layers each with its own Key, Query and Values vector



Transformer Model

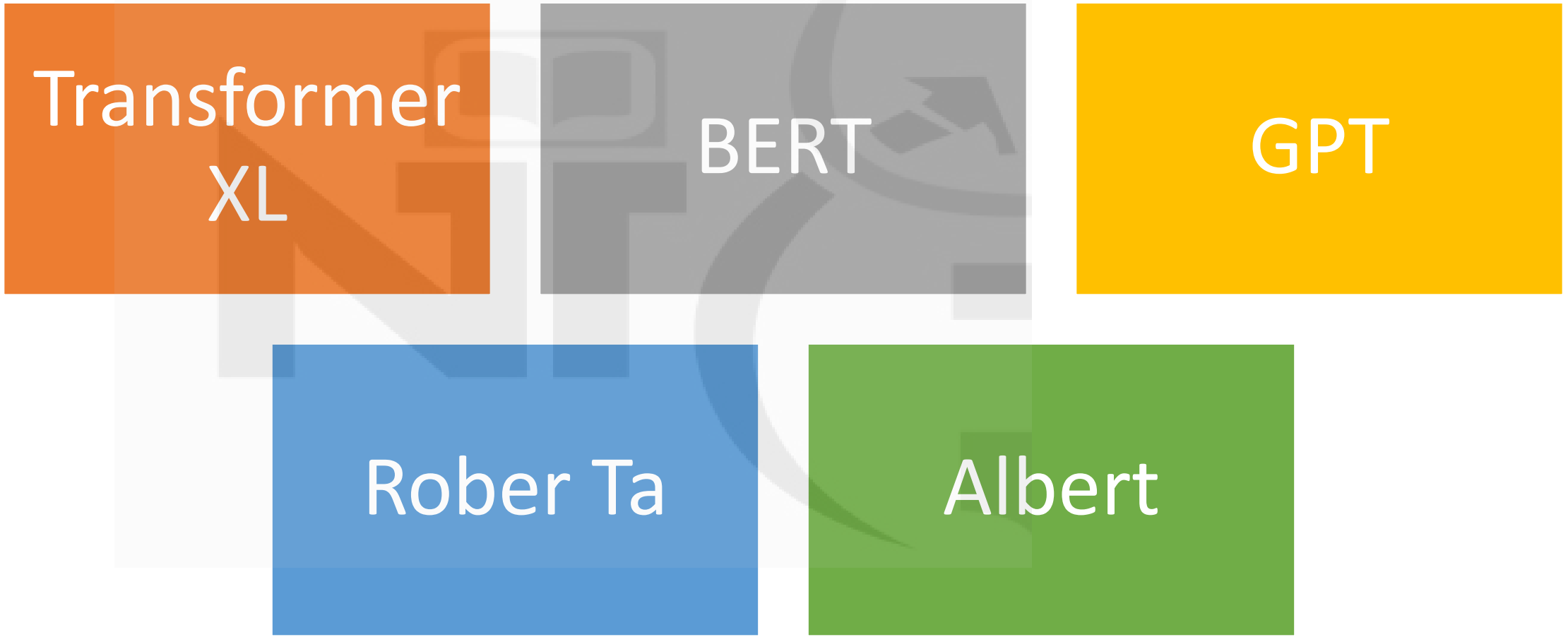
The authors of Transformers dint stop here

A Transformer usually has multiple encoder and multiple decoders

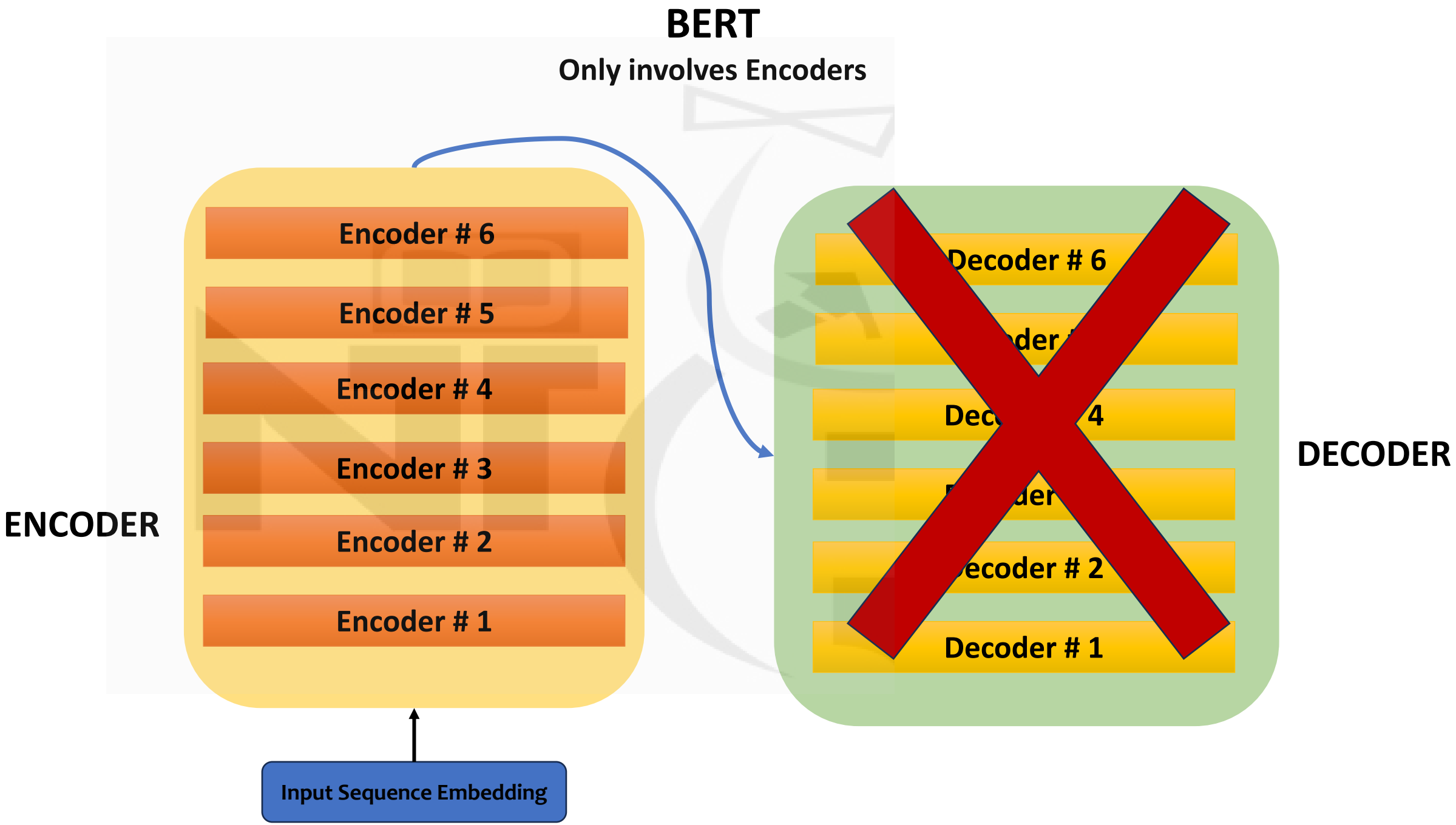


Transformer Model

Different Models build using Transformer Architecture


Transformer XL
BERT
GPT
Rober Ta
Albert

Bidirectional Encoder Representations from Transformers


BERT
Only involves Encoders
ENCODER
Encoder # 6
Encoder # 5
Encoder # 4
Encoder # 3
Encoder # 2
Encoder # 1
Input Sequence Embedding
DECODER
Decoder # 6
Decoder # 1

How do we train BERT?

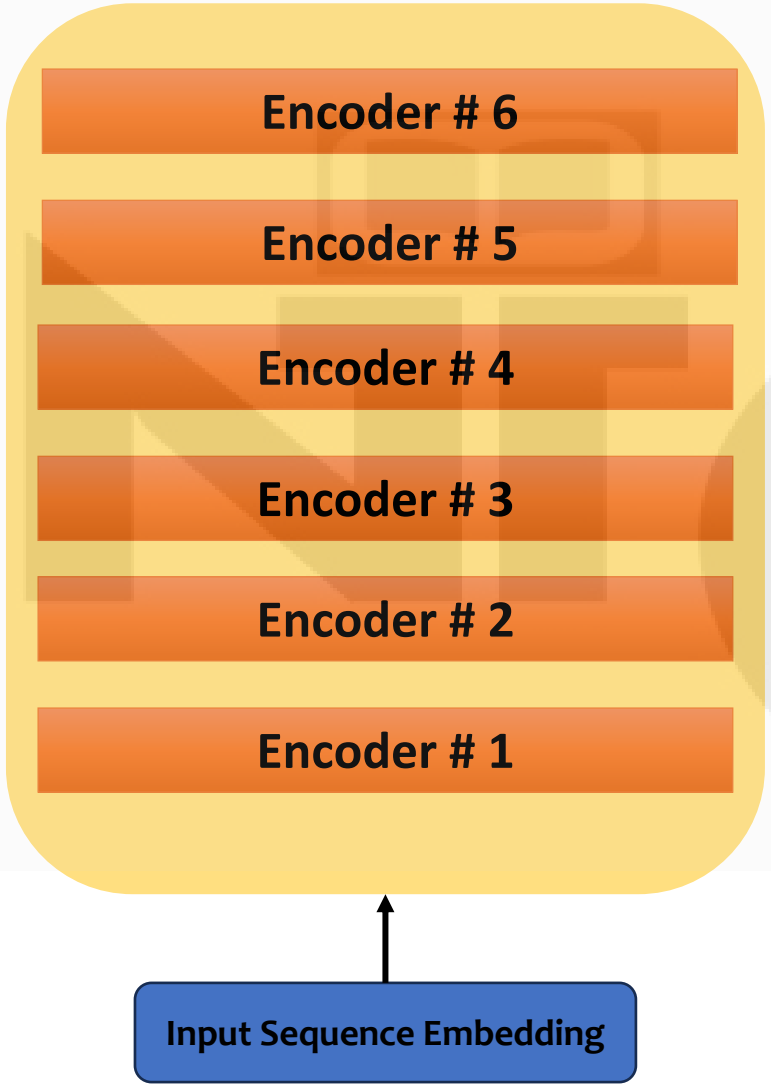
- With tons of data but in self-supervised way

How BERT is trained in self supervised way

- Masked language Modelling (MLM)
- Next Sentence Prediction

BERT

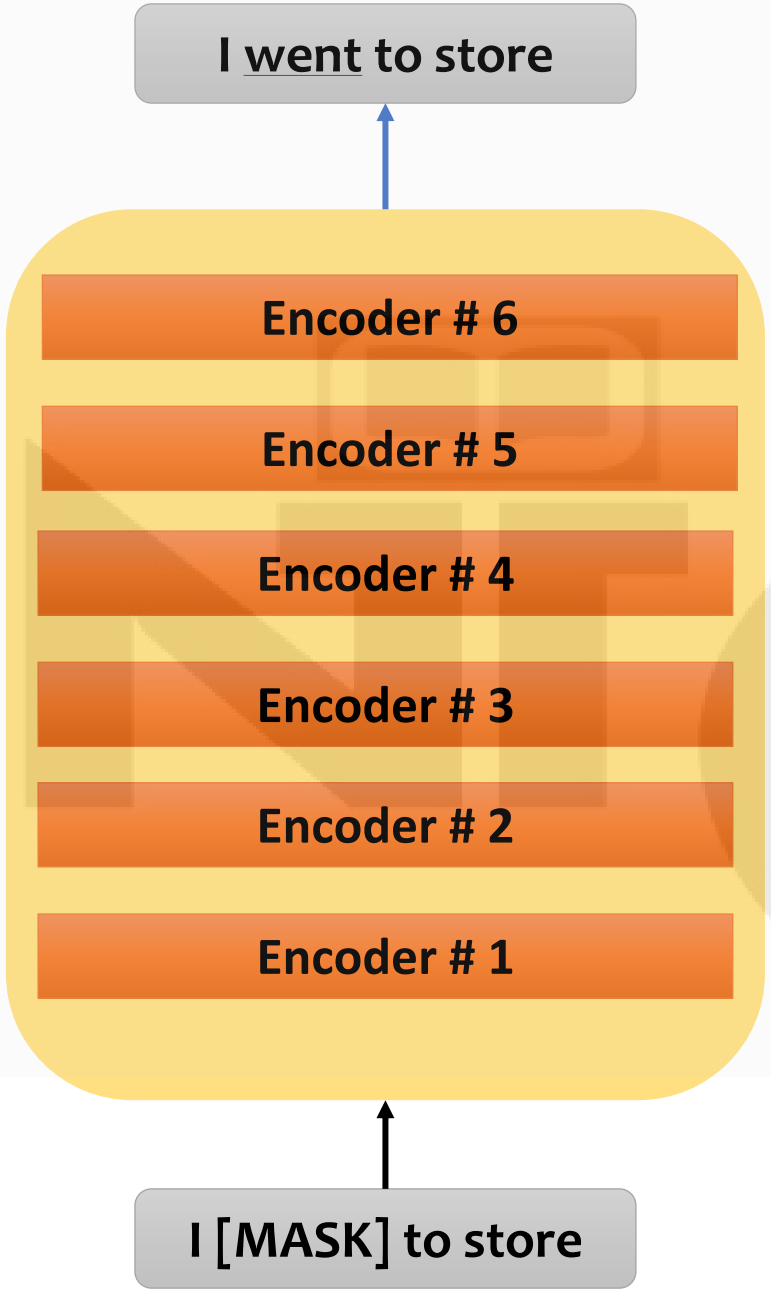
Masked Language Modelling



1. Up to 15% words in the input sequence are masked.
2. 2. Model is expected to predict masked word(s).

BERT

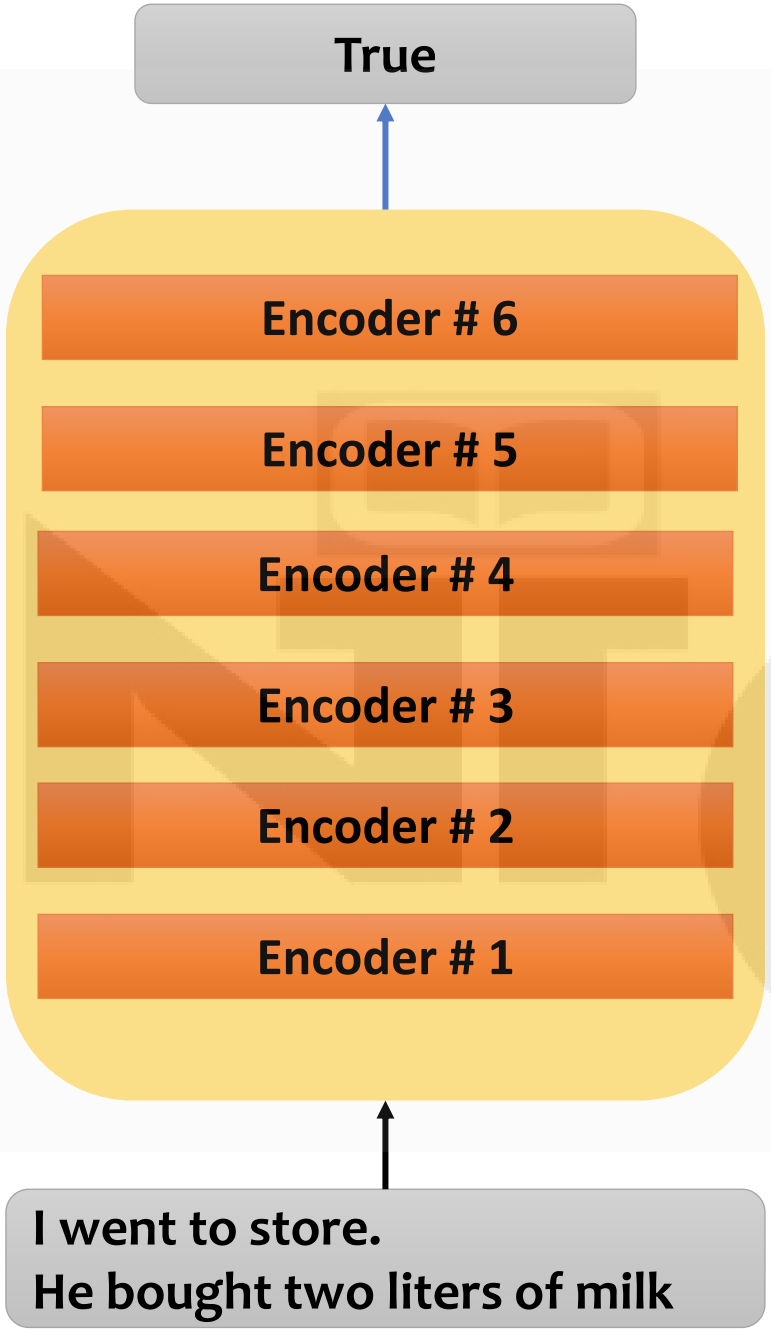
Masked Language Modelling



1. Up to 15% words in the input sequence are masked.
2. 2. Model is expected to predict masked word(s).

BERT

Next Sentence Prediction(NSP)



1. Feed two sentences to the model
2. Model will predict if 2nd sentence should follow 1st sentence.