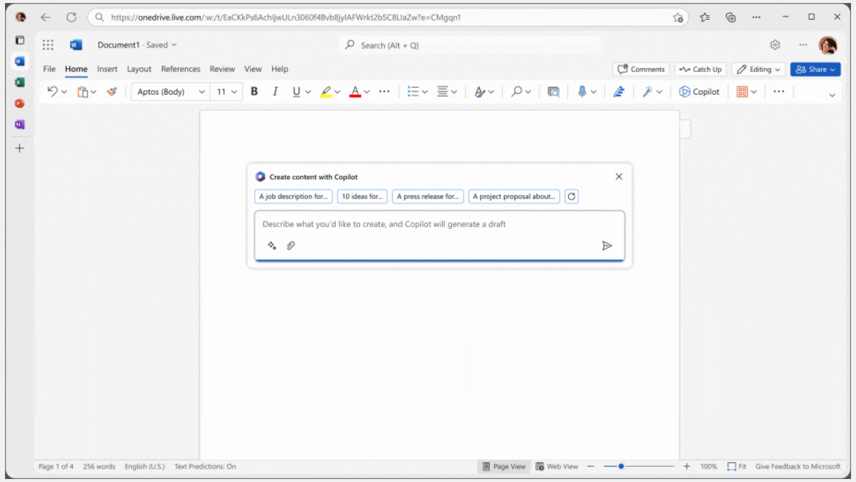
Generative AI

Generative AI

Generative AI is a branch of artificial intelligence primarily used for creative tasks

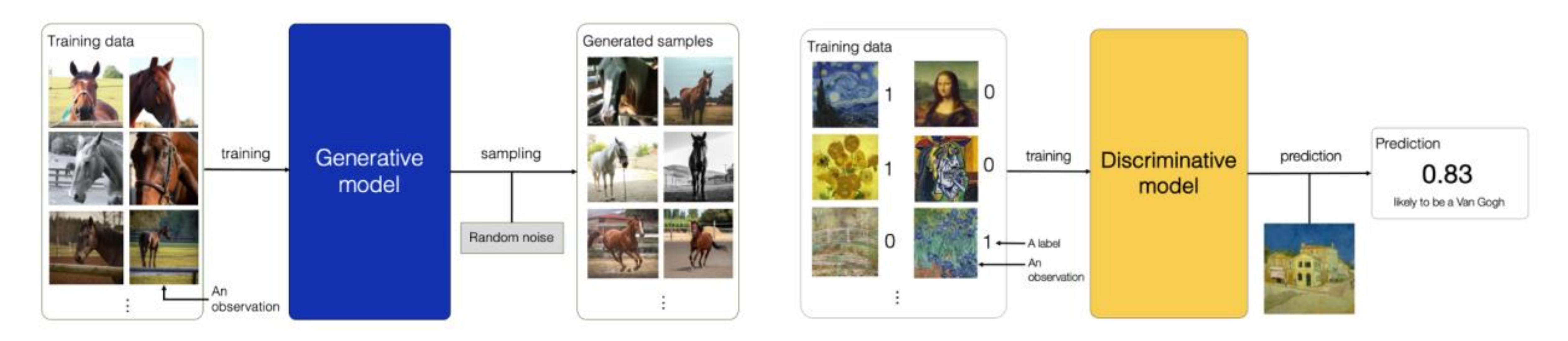
- ➢ Articles, images, music, and more
    - ChatGPT, Midjouney, Stable Diffusion, Office Copilot, and so on



Generative AI

Differences between generative AI and Discriminative AI

- Discriminative AI - predict labels or classifications based on training sets
  - Predict labels or classifications based on training sets
- Generative AI - generate new data samples based on training sets
  - Generate new data samples based on training sets



Source

Generative AI in Language

01

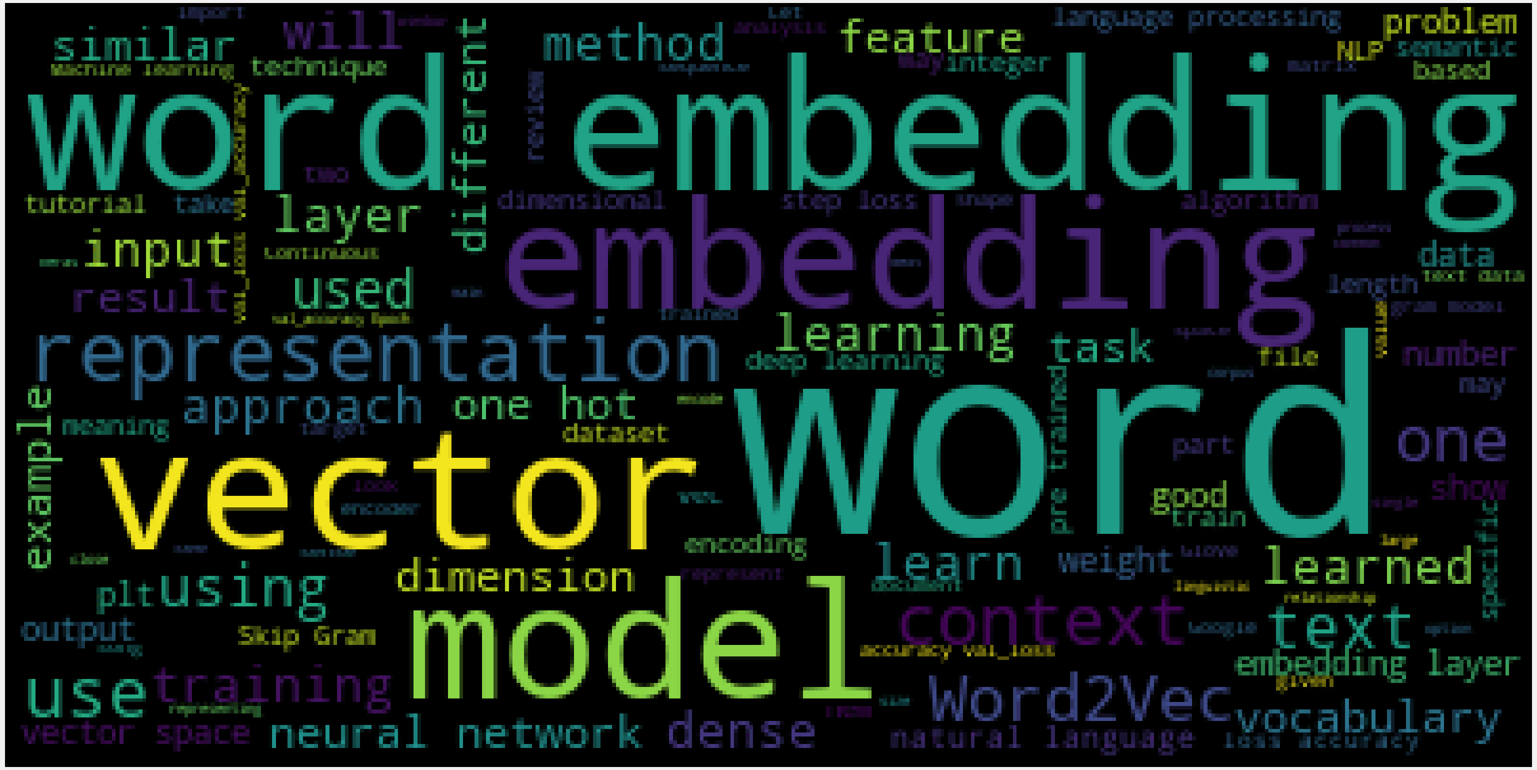
Word Embedding

Word Embedding

It is a method to embed "word" into "value"

It is a dimension reduction technology

The value is "vector"



Word Embedding

Word vector is generated with unsupervised learning

Car

↓

function

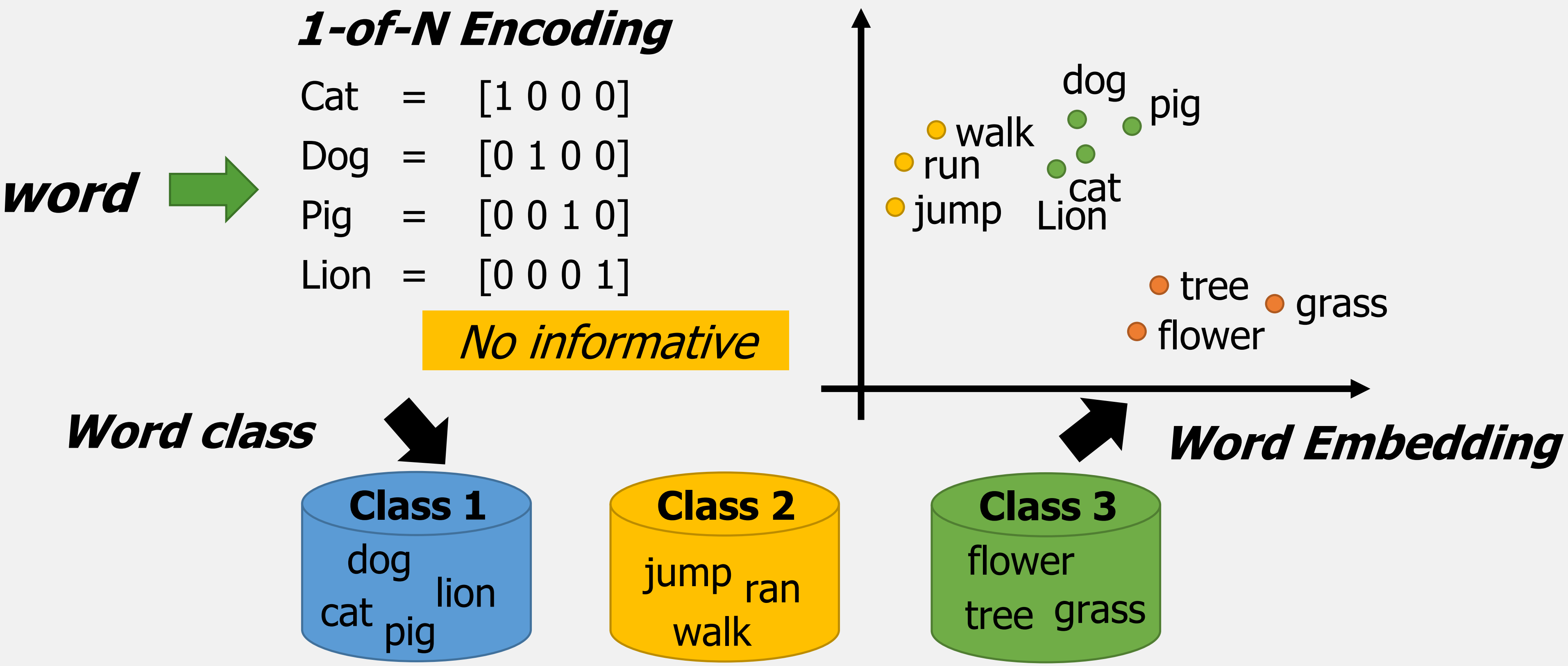
↓

[? ? ? ?]

Word Embedding

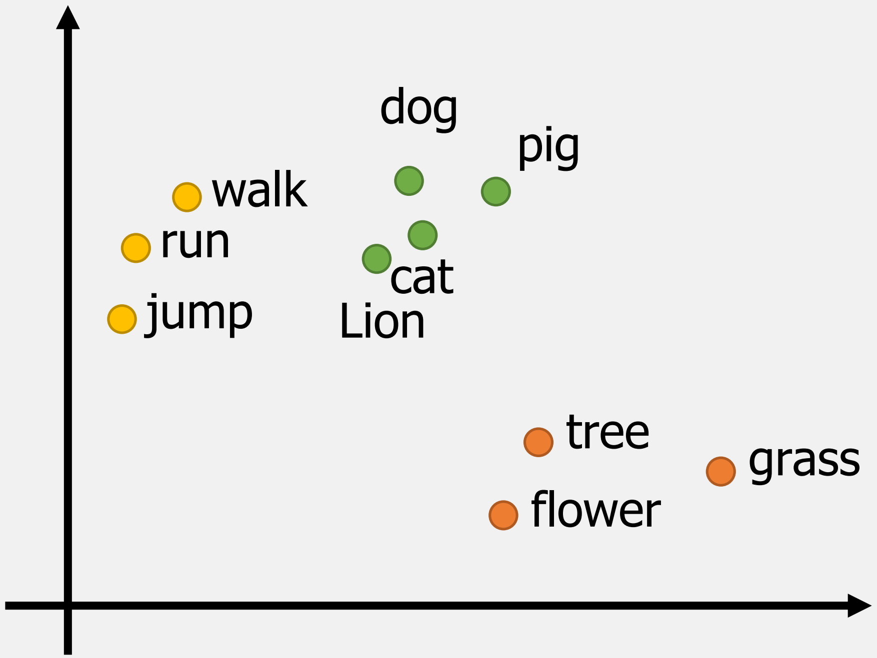
One-hot Encoding

Word vector is generated with unsupervised learning

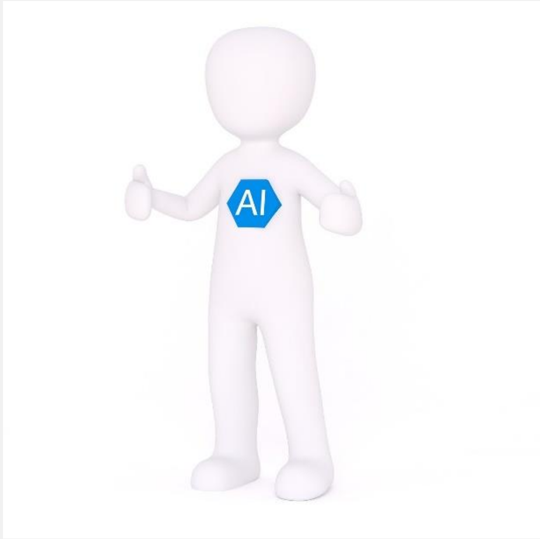


Word Embedding

Word Embedding



Documents



Word Embedding

How to do it?

➢ Counting-based

- if two words wi and wj often co-occur, the embedding vector vi and vj would be close to each other
- i.e. the larger inner product, the closer to each other

- The popular word embedding technology: Glove vector

Word Embedding

How to do it?

➢ Glove vector

- It uses the overall statistics and context to complete the word embedding

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Reference

Word Embedding

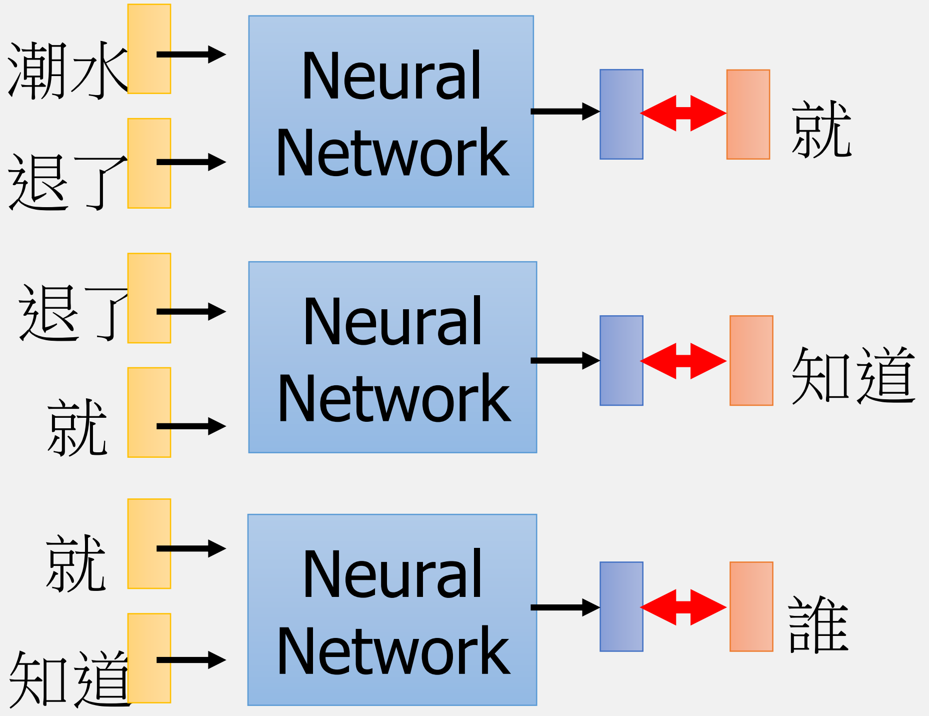
How to do it?

- Prediction-based

Ptter data:

道歉　時　需要 …
本來 是 想說　點 什麼的 …
不然　你要　投 …
潮水　退了　就　知道　誰 …
不爽　　不要　　買 …
公道價 八萬　一 …
………

Minimizing cross entropy

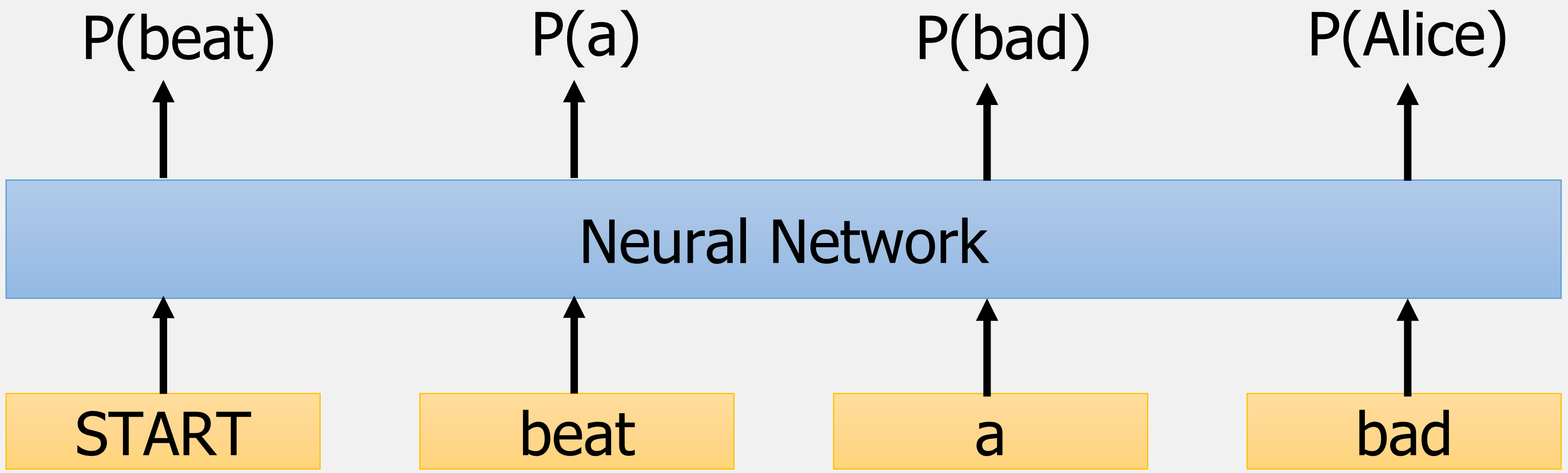


Word Embedding

Prediction-based - using a word xi to predict the next word xi+1

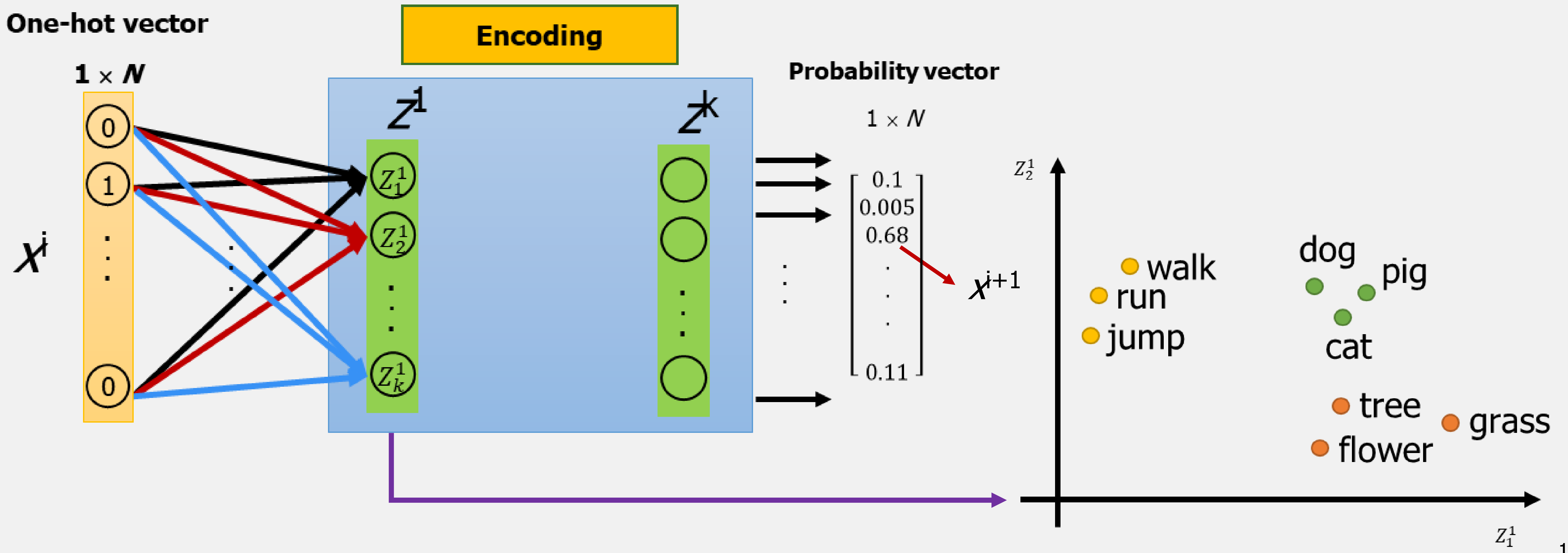
- P("beat a bad Alice") = P(beat|START)P(a|beat)P(bad|a)P(Alice|bad)

P(b|a): the probability of model to predict the next word



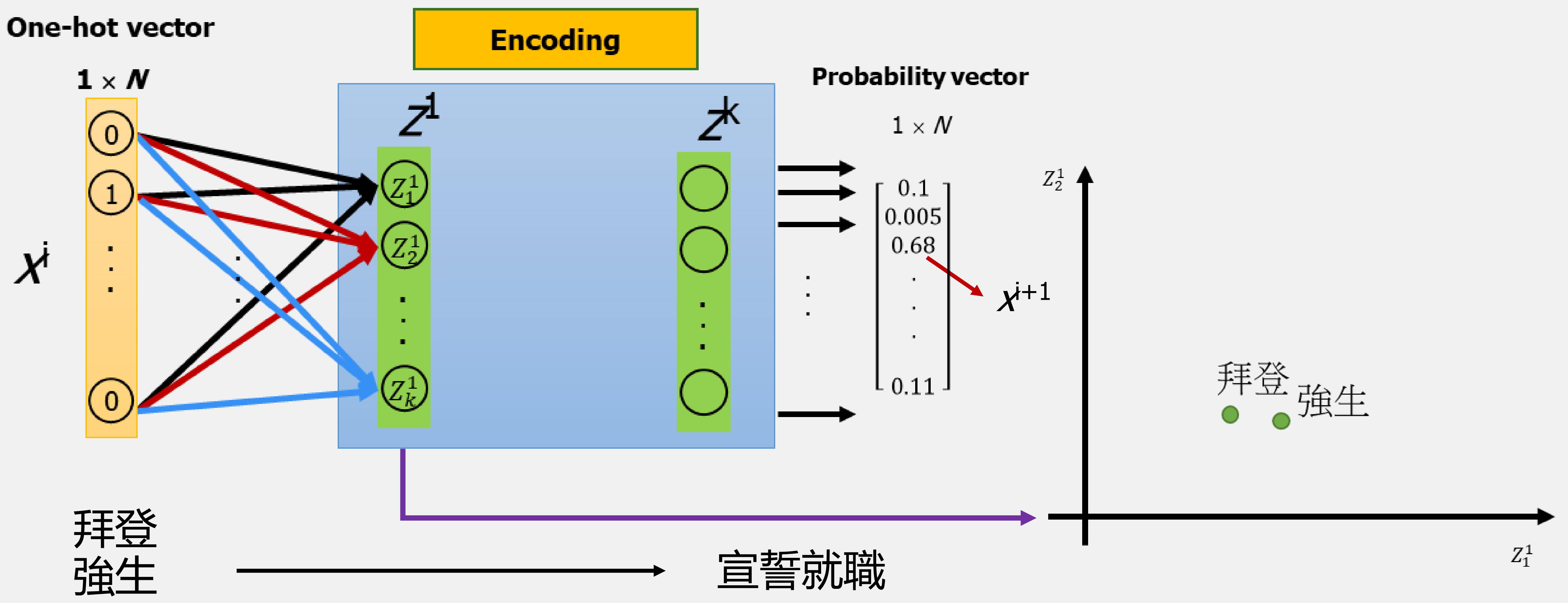
Word Embedding

Prediction-based



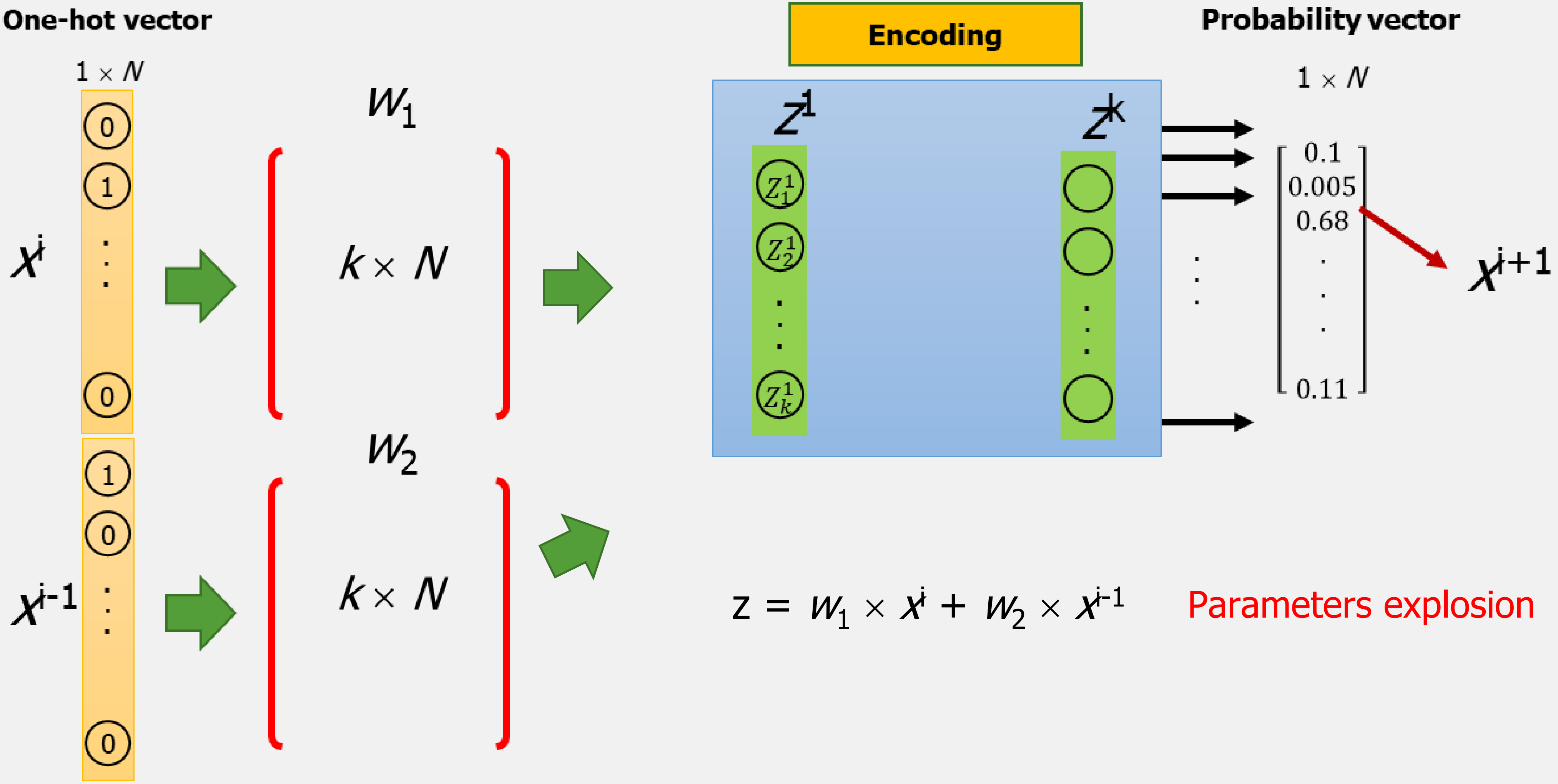
Word Embedding

Prediction-based



Word Embedding

Prediction-based - More and more words

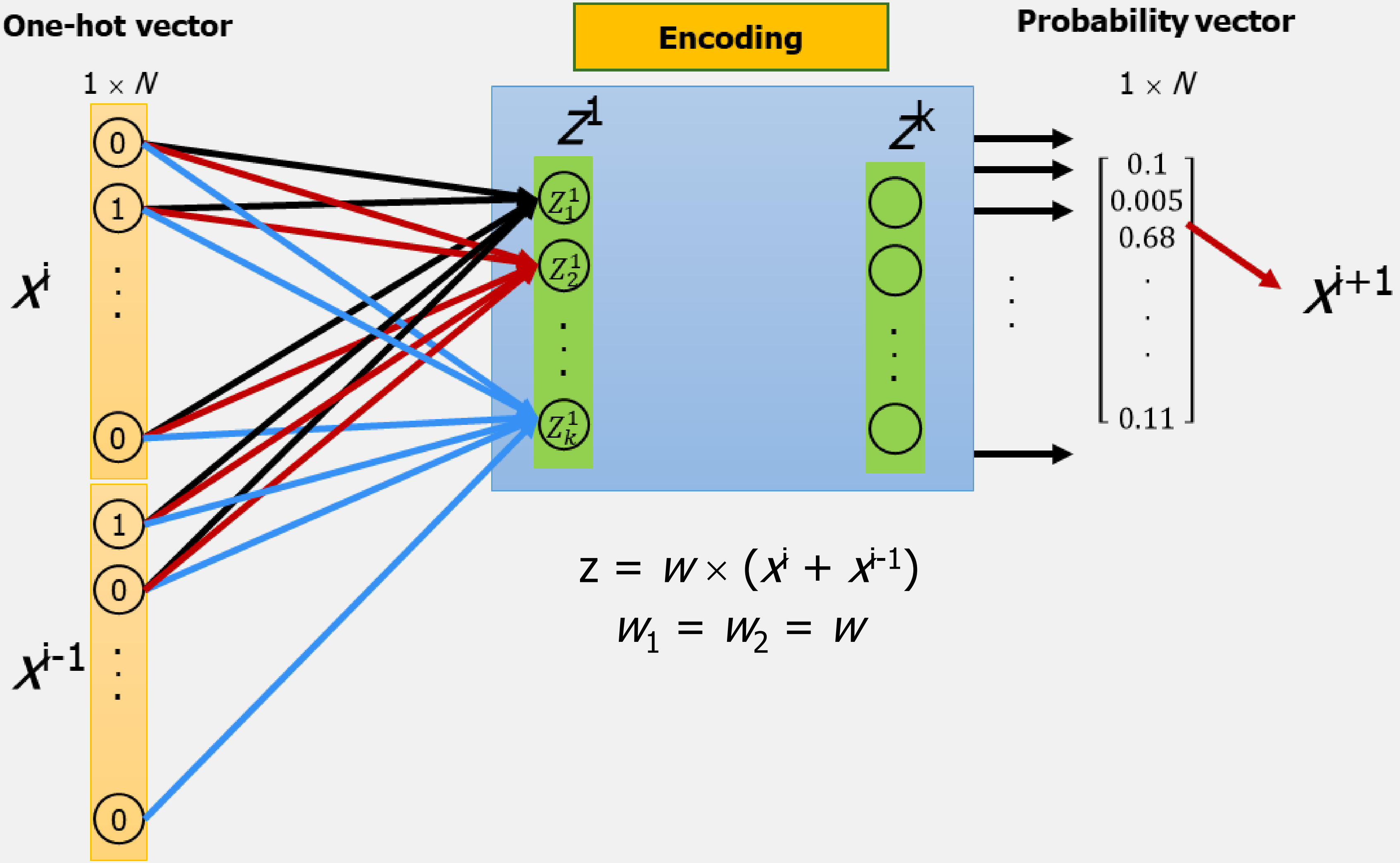


$$z = W_1 \times x^i + W_2 \times x^{i-1}$$

Parameters explosion

Word Embedding

Prediction-based - More and more words



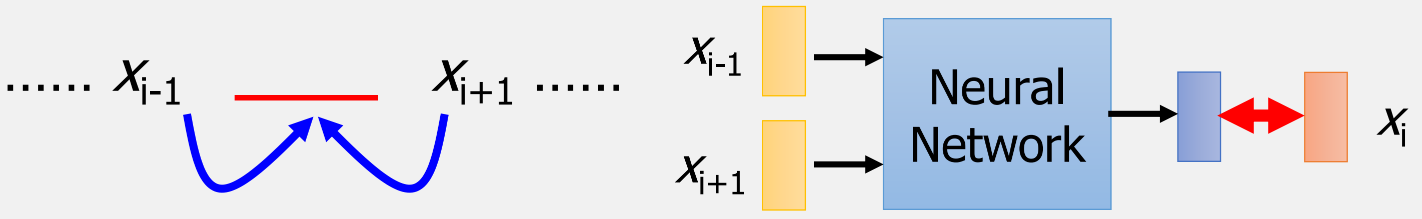
$$z = W \times (x^i + x^{i-1})$$

$$W_1 = W_2 = W$$

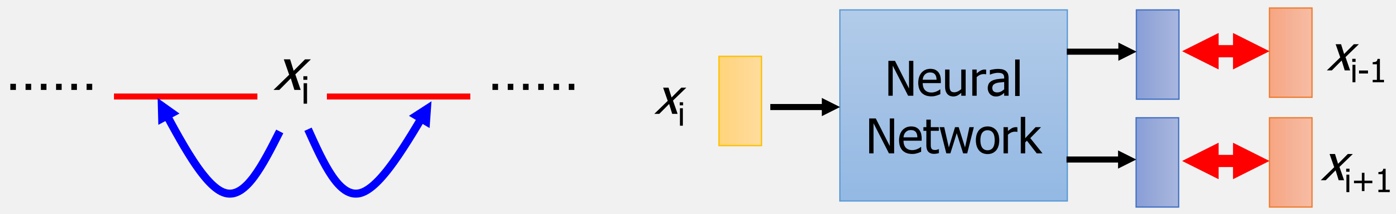
Word Embedding

Other methods

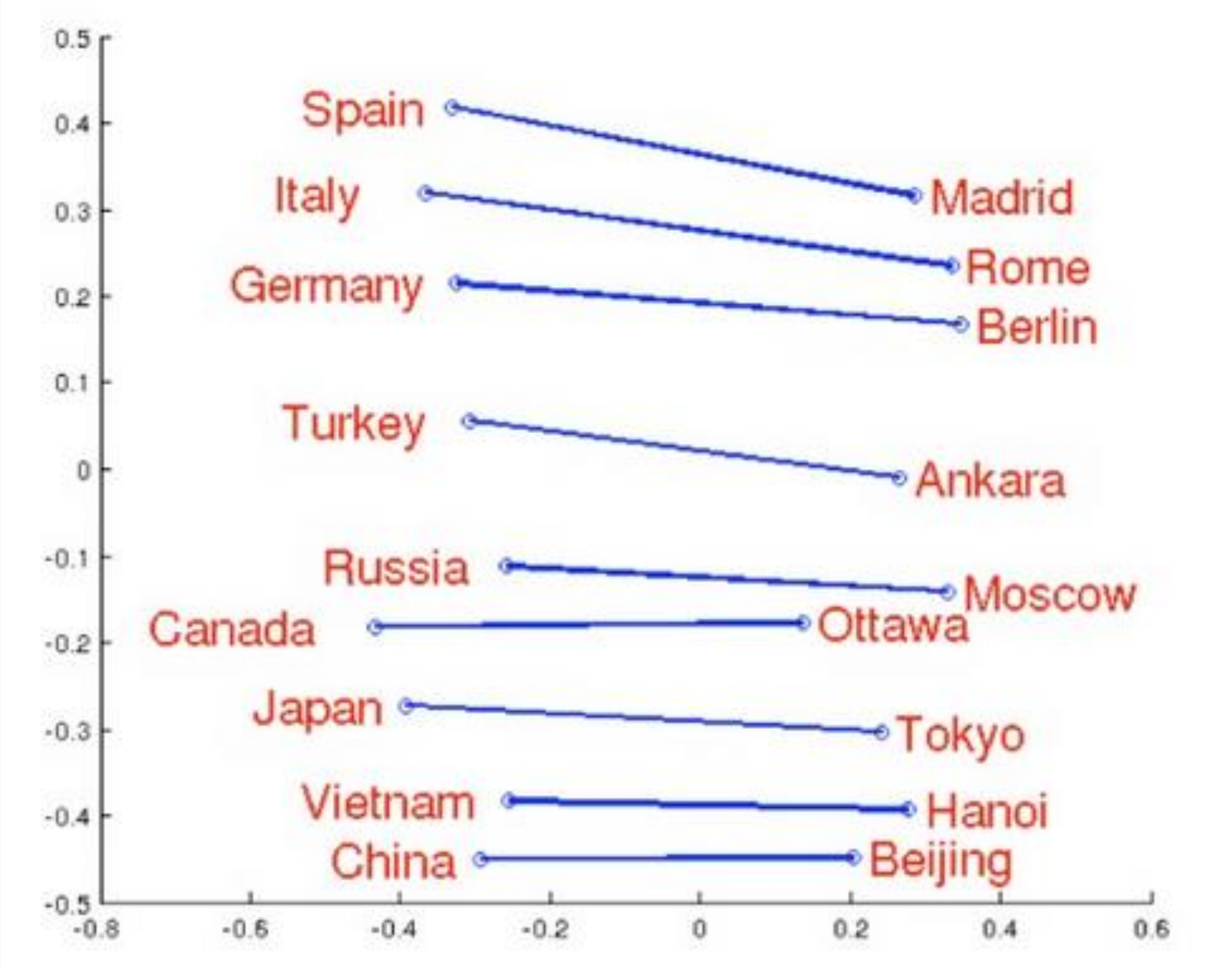
- Continuous bag of word (CBOW)
  - predict a word from given context

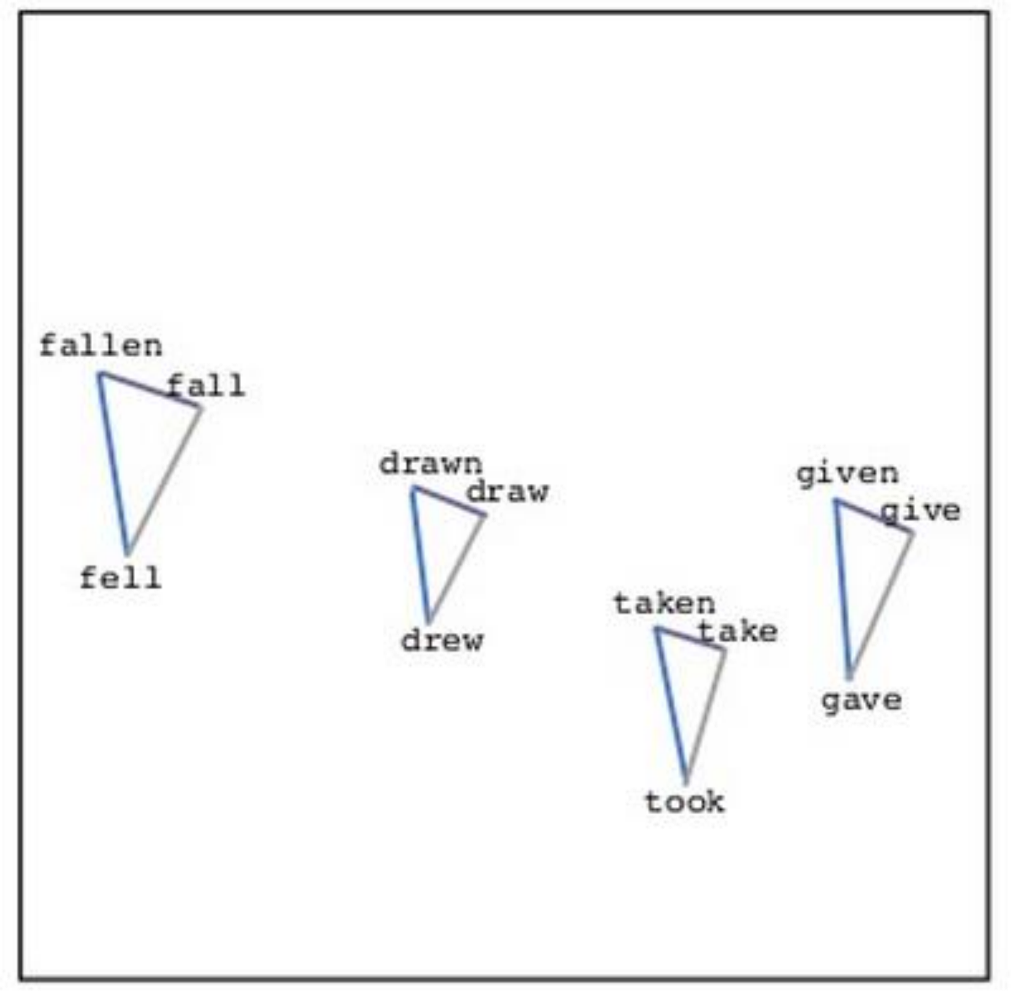


- Skip-gram
  - predict a context from given a word



Word Embedding



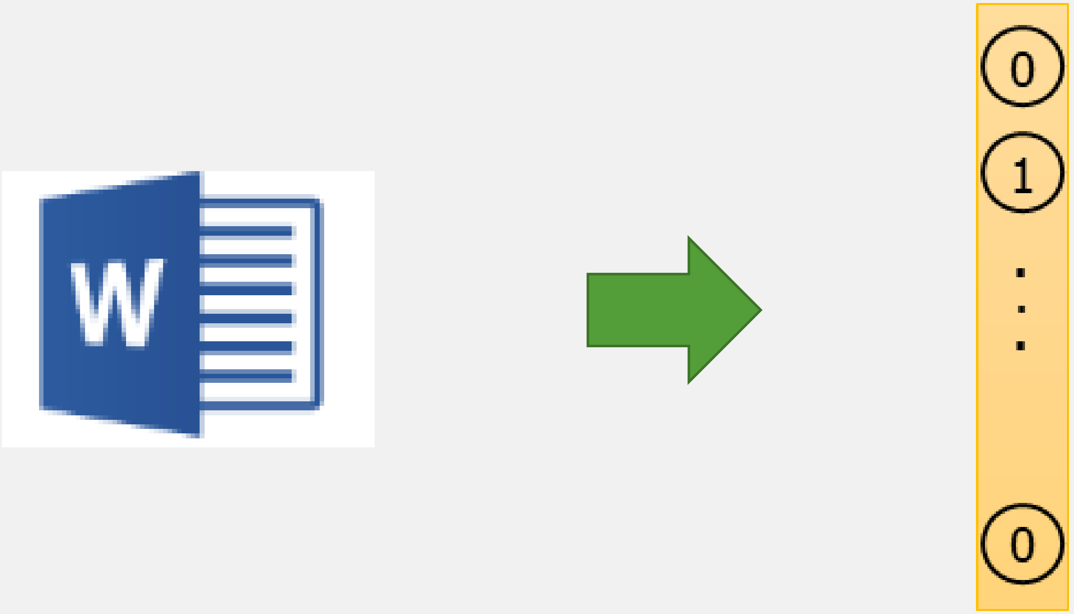


Reference

Document Embedding

Could the document embedding be achieved too?

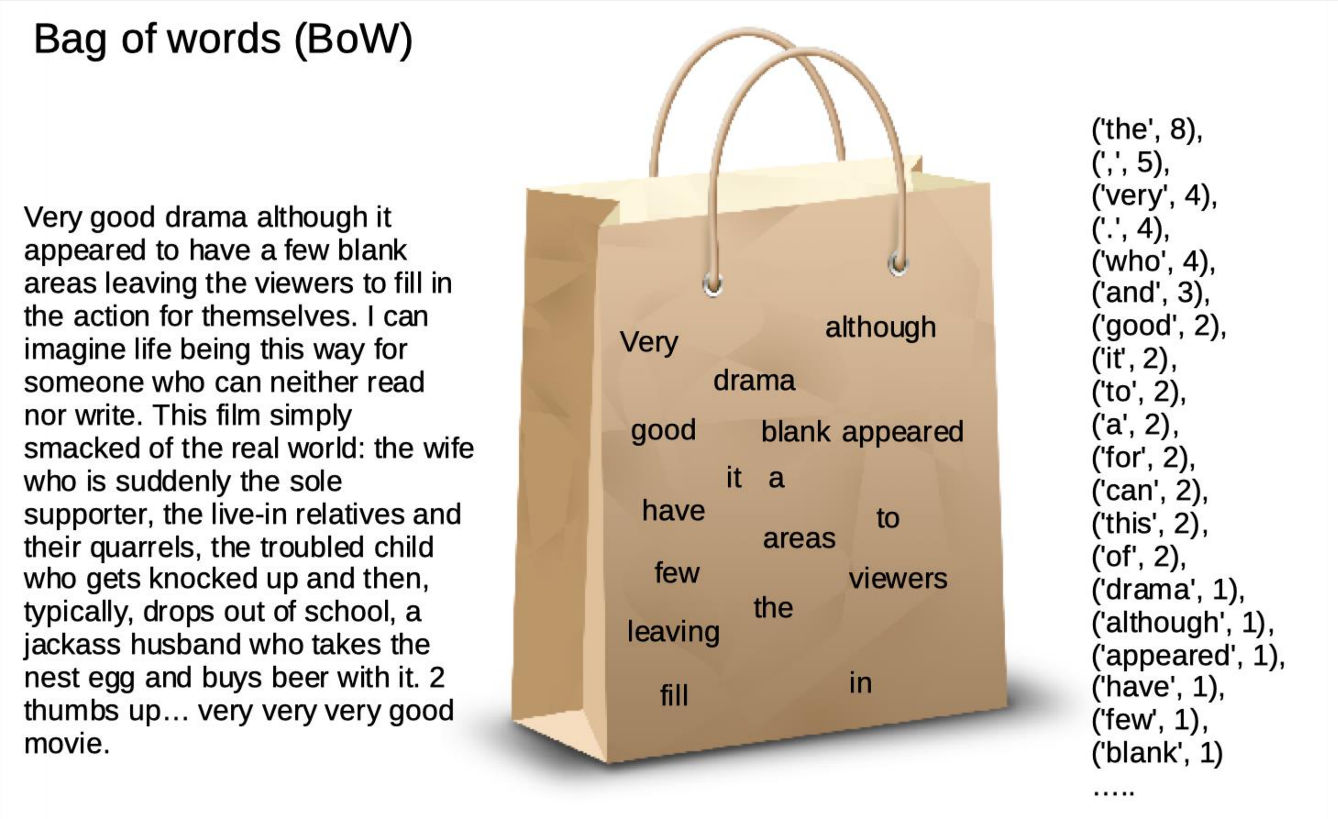
- word -> word sequences with different lengths -> the vector with the same length
- The vector represents the meaning of the word sequence
- A word sequence can be a document or a paragraph



Document Embedding

Bag-of-word (BOW)

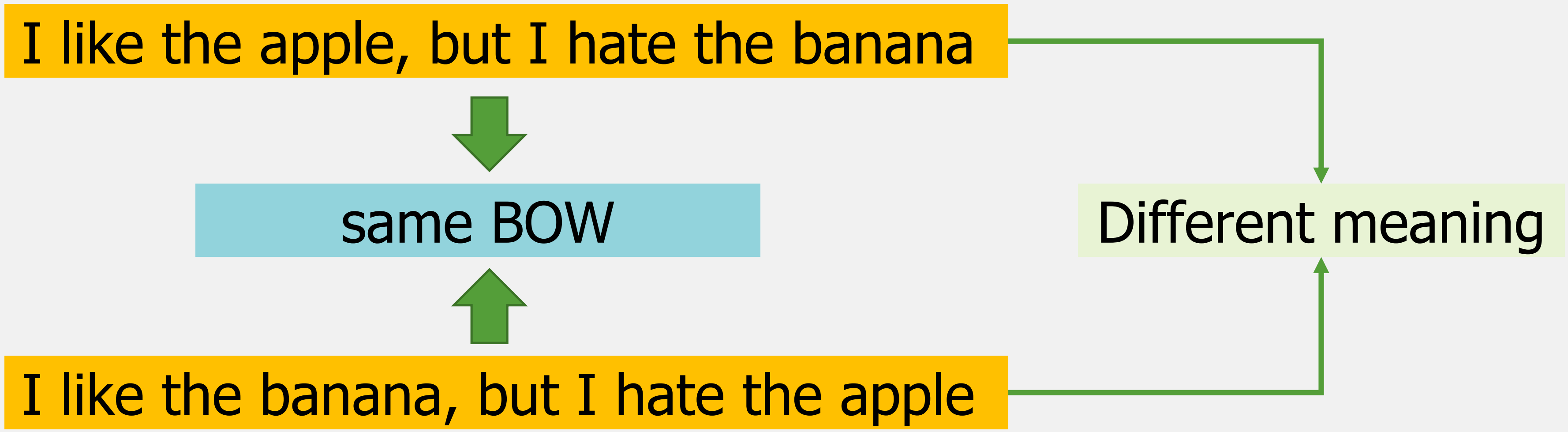
- Process a sentence or a document as a bag of words
- Each document is converted to a <word, count> map
- Document similarity
  - Euclidean distance
  - Cosine
  - Dot-product
  - …



Document Embedding

The problem of BOW

- the information of order of the words is ignored



Word Embedding

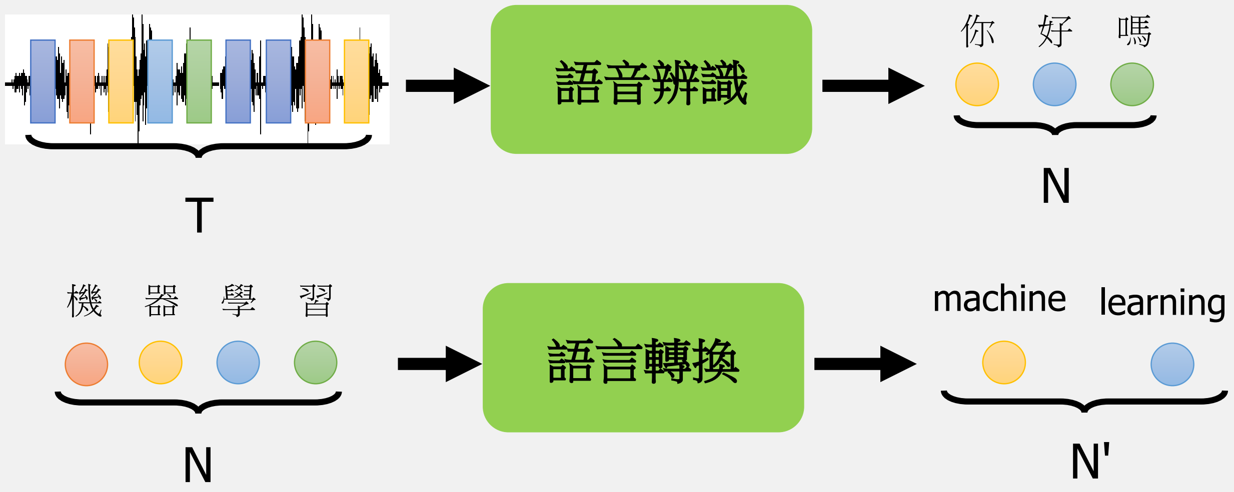
[Example]

02

Seq2Seq

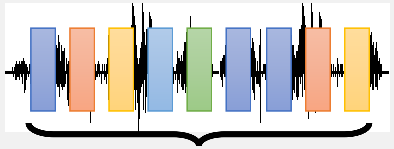
Seq2Seq

Input a sequence, output a sequence whose length is determined by model

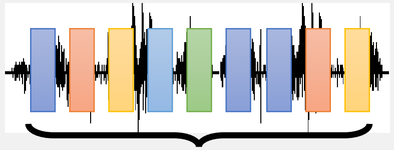


Seq2Seq

In 台語



Speech in 台語 → 語音辨識 → 挖 德 諾 (N)



Speech in 台語 → 語言翻譯 → 我 就 爛 (N')

Seq2Seq

Chatbot

"Hi"

"I am good thank you, how are you"

Seq2Seq Model

"Hello! How are you today?"

"Great, thanks! My ..."

Training data:

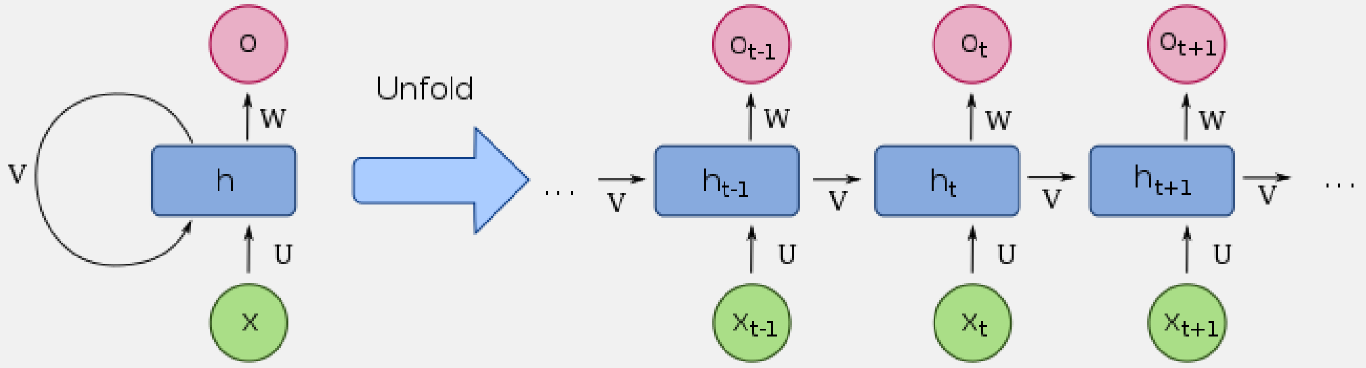
[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

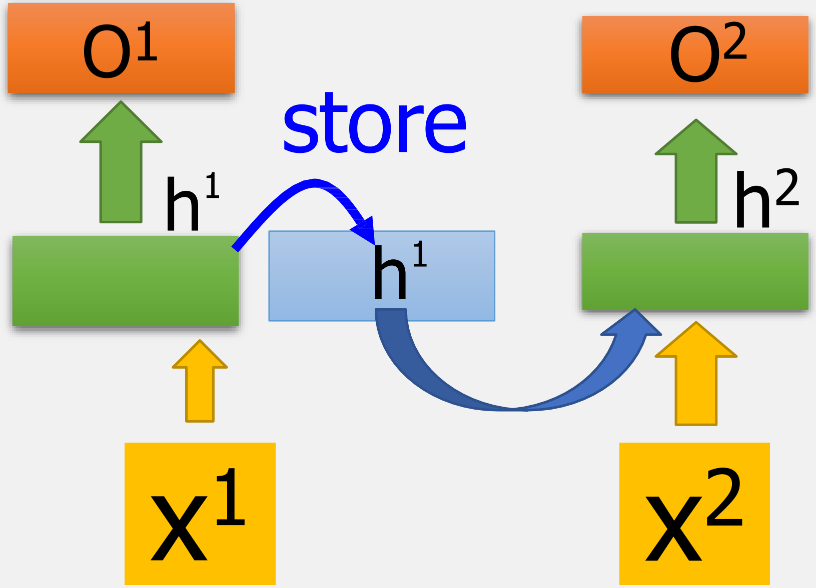
A

RNN and LSTM

RNN and LSTM

RNN is designed for the time-series problem

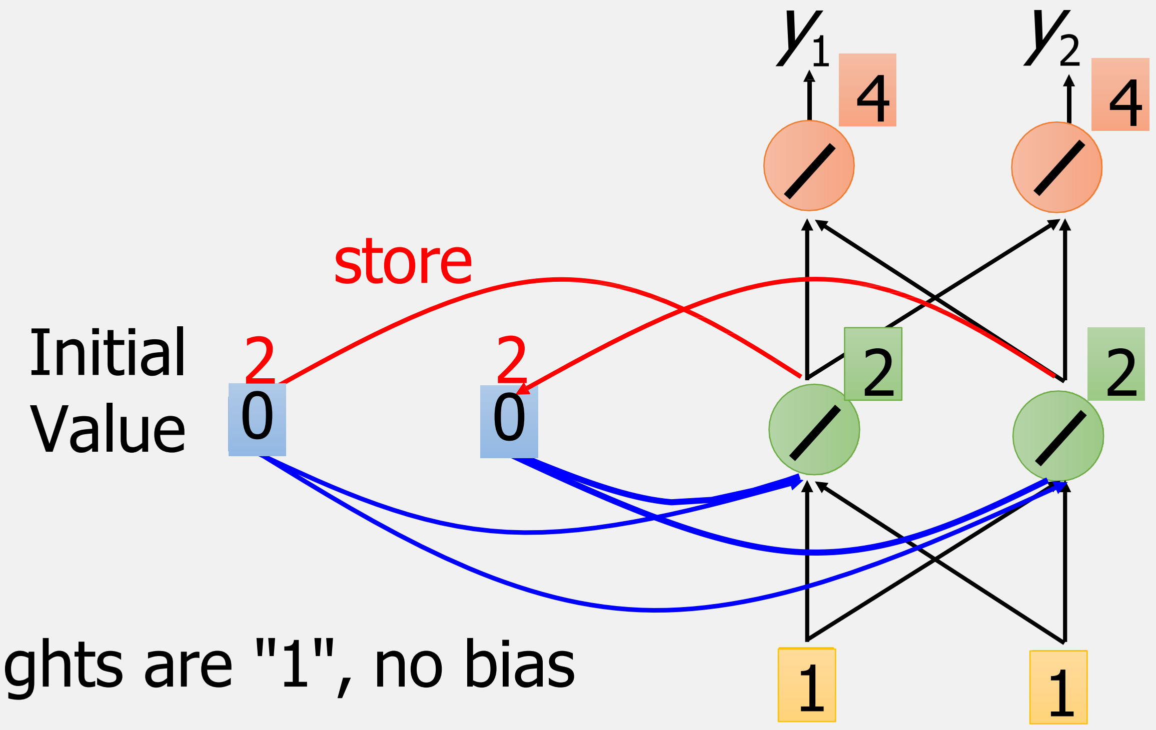




RNN and LSTM

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}\cdots$

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\quad\cdots$



All the weights are "1", no bias

All activation functions are linear

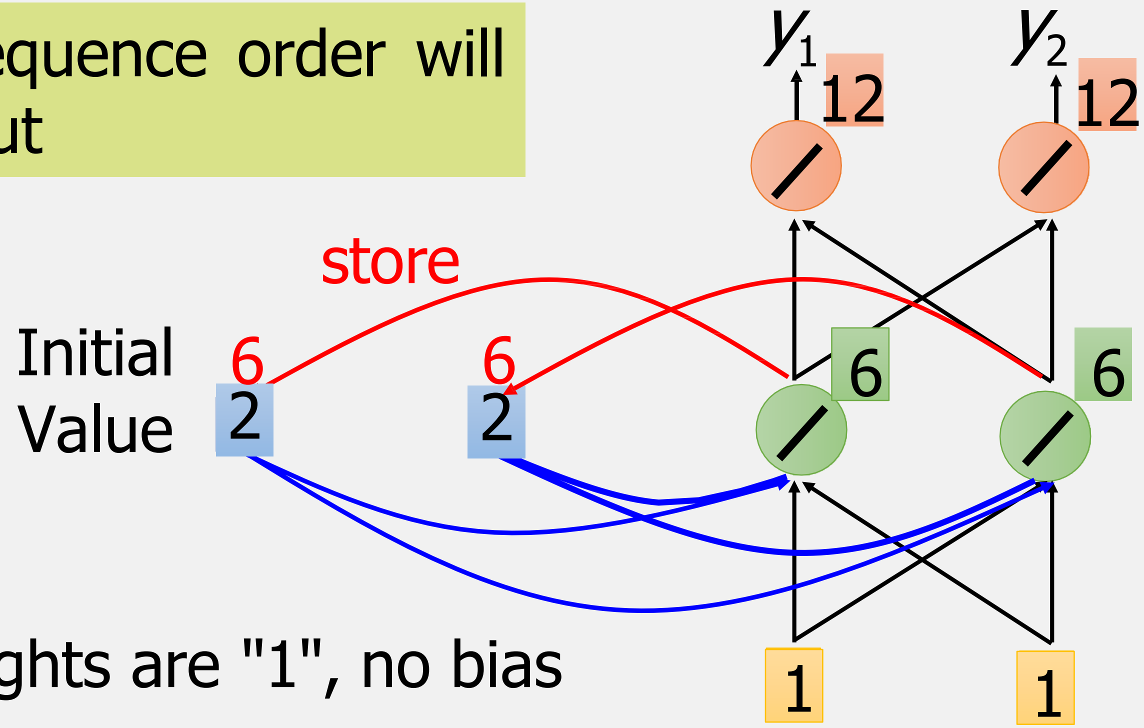
Reference

RNN and LSTM

Input sequence: $\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix}\begin{bmatrix}2\\2\end{bmatrix}\cdots$

output sequence: $\begin{bmatrix}4\\4\end{bmatrix}\begin{bmatrix}12\\12\end{bmatrix}\cdots$

Changing the sequence order will change the output

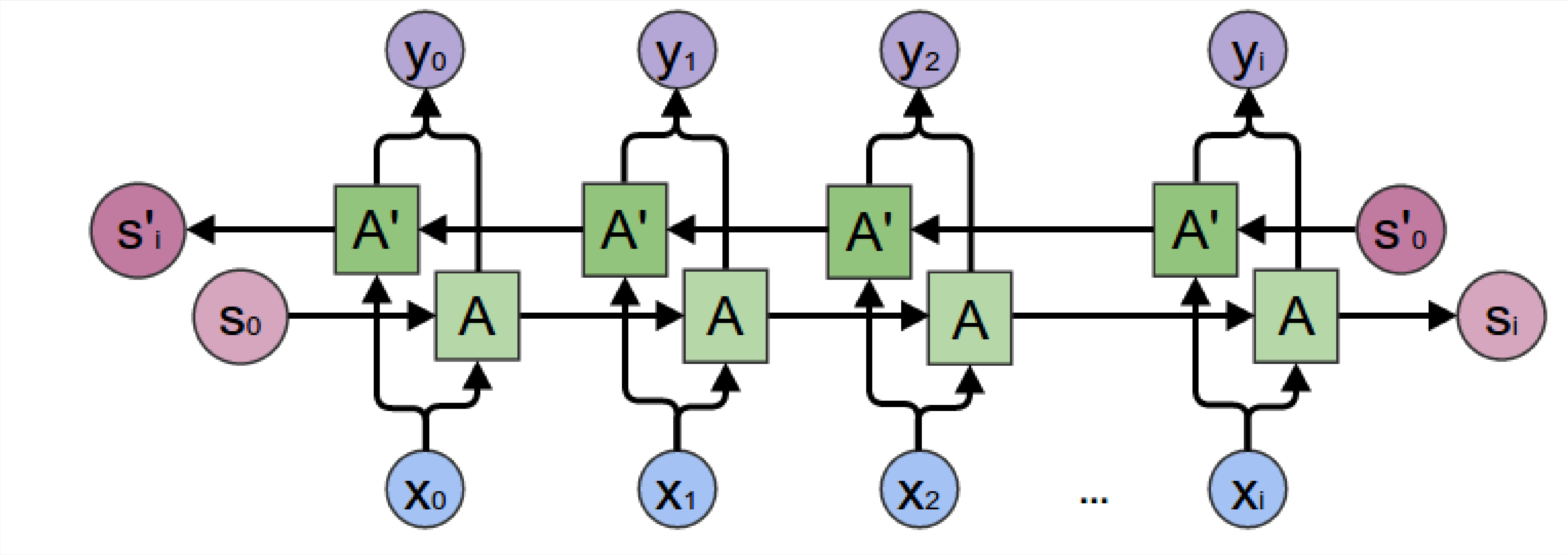


All the weights are "1", no bias

All activation functions are linear

RNN and LSTM

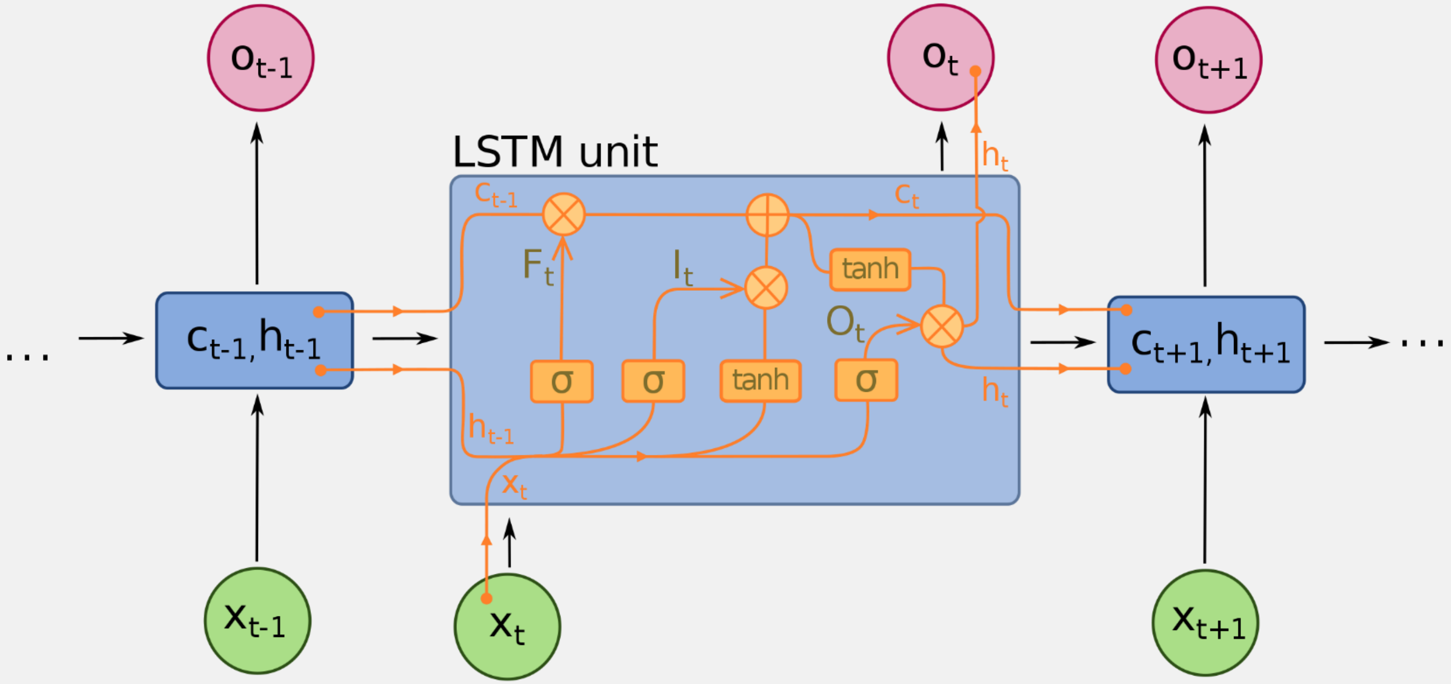
Bi-directional RNN

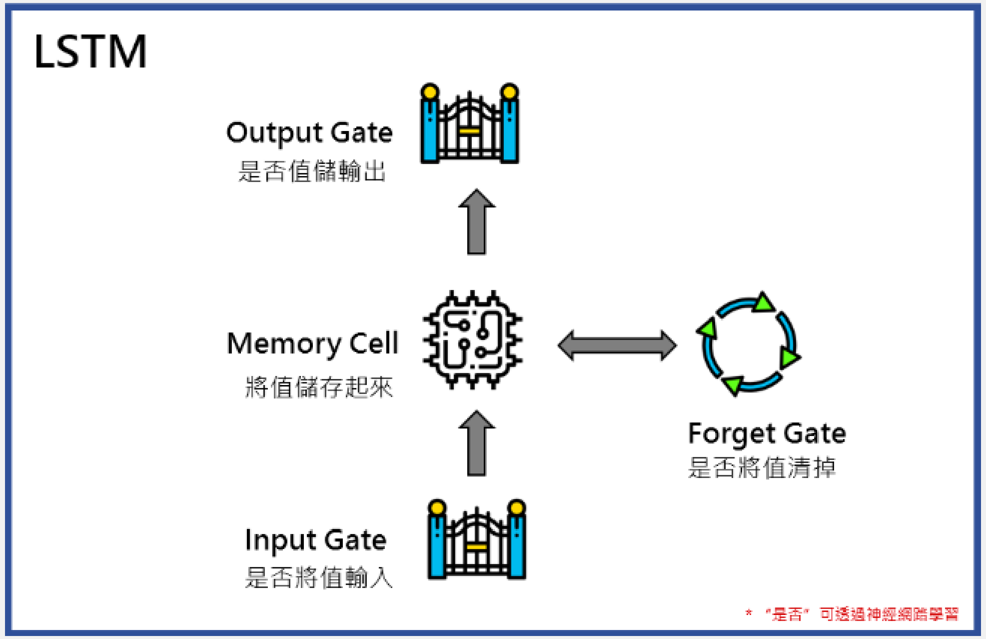


[Reference](#)

RNN and LSTM

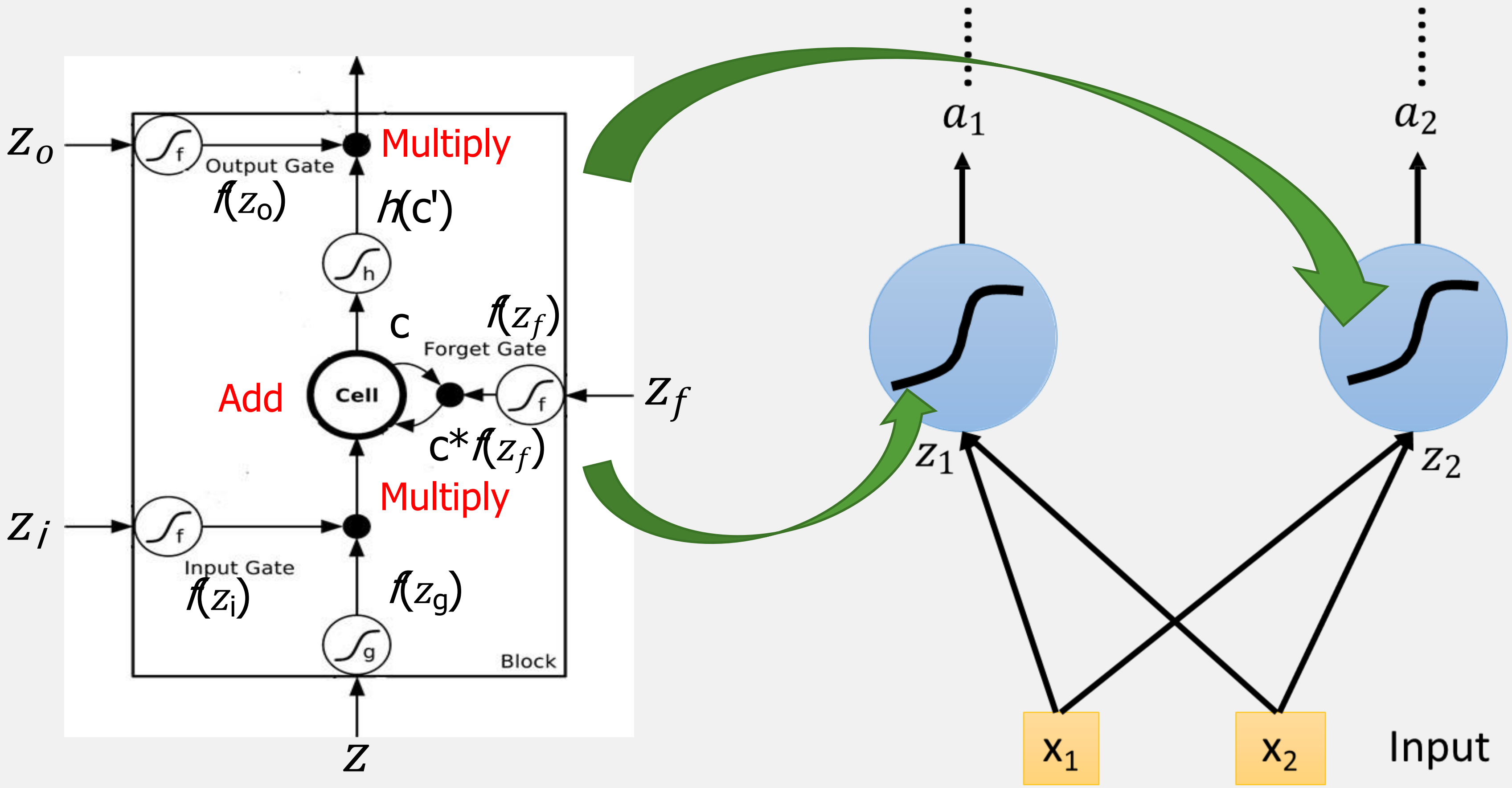
LSTM is the best famous model in RNN





[Reference](#)

RNN and LSTM

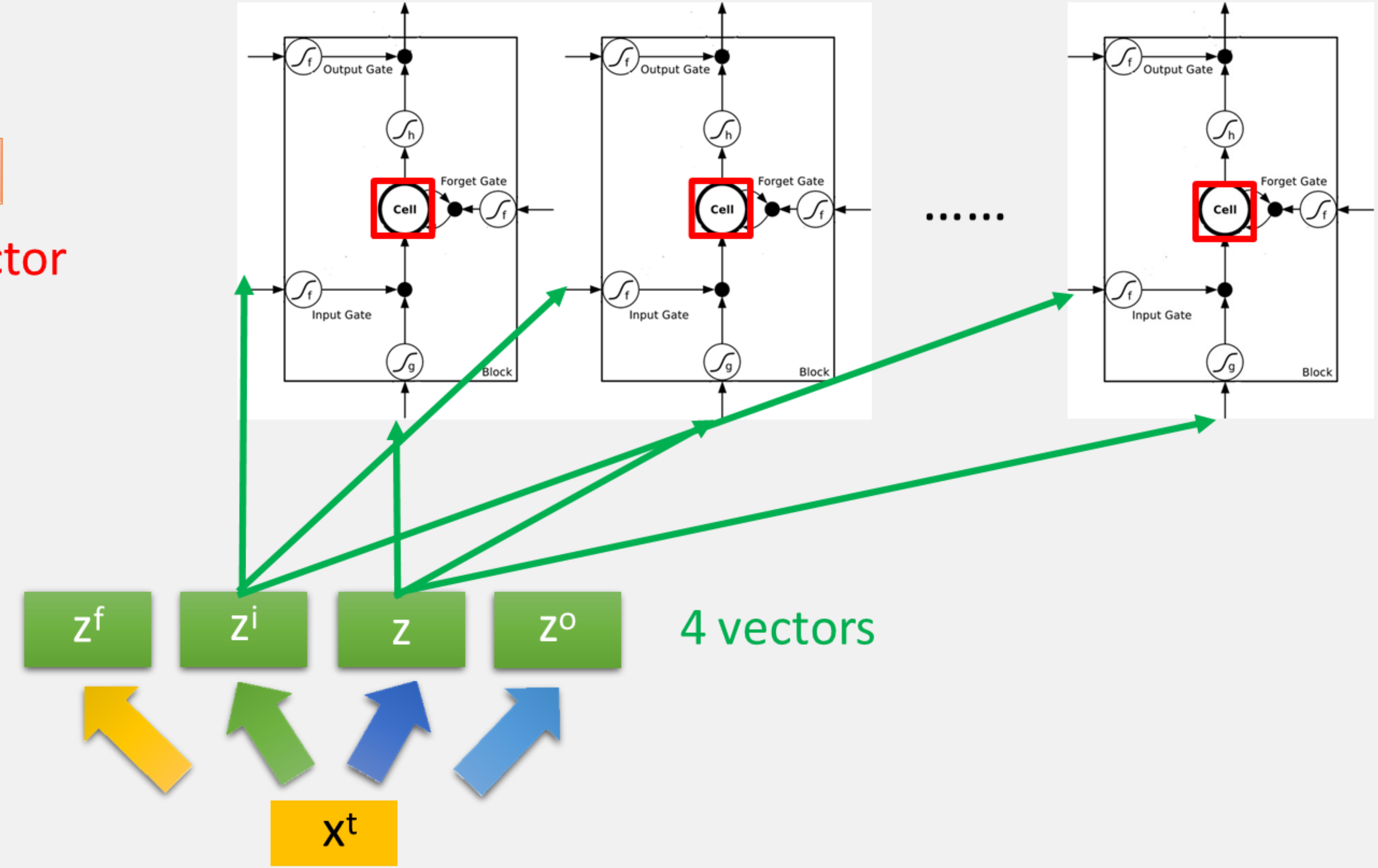


z_o
Output Gate
Multiply
$f(z_o)$
$h(c')$
c
$f(z_f)$
Forget Gate
Add
Cell
z_f
$c*f(z_f)$
Multiply
z_i
Input Gate
$f(z_i)$
$f(z_g)$
Block
z
a_1
a_2
z_1
z_2
x_1
x_2
Input

RNN and LSTM

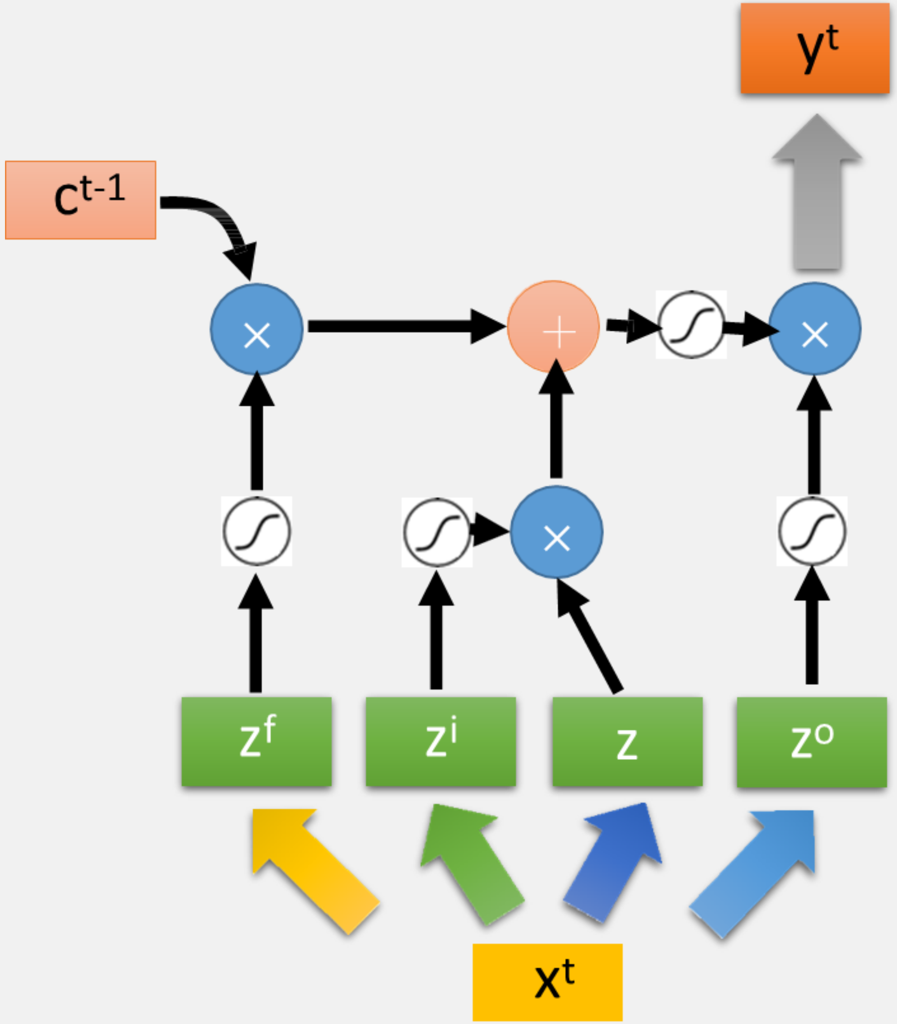
c^{t-1}

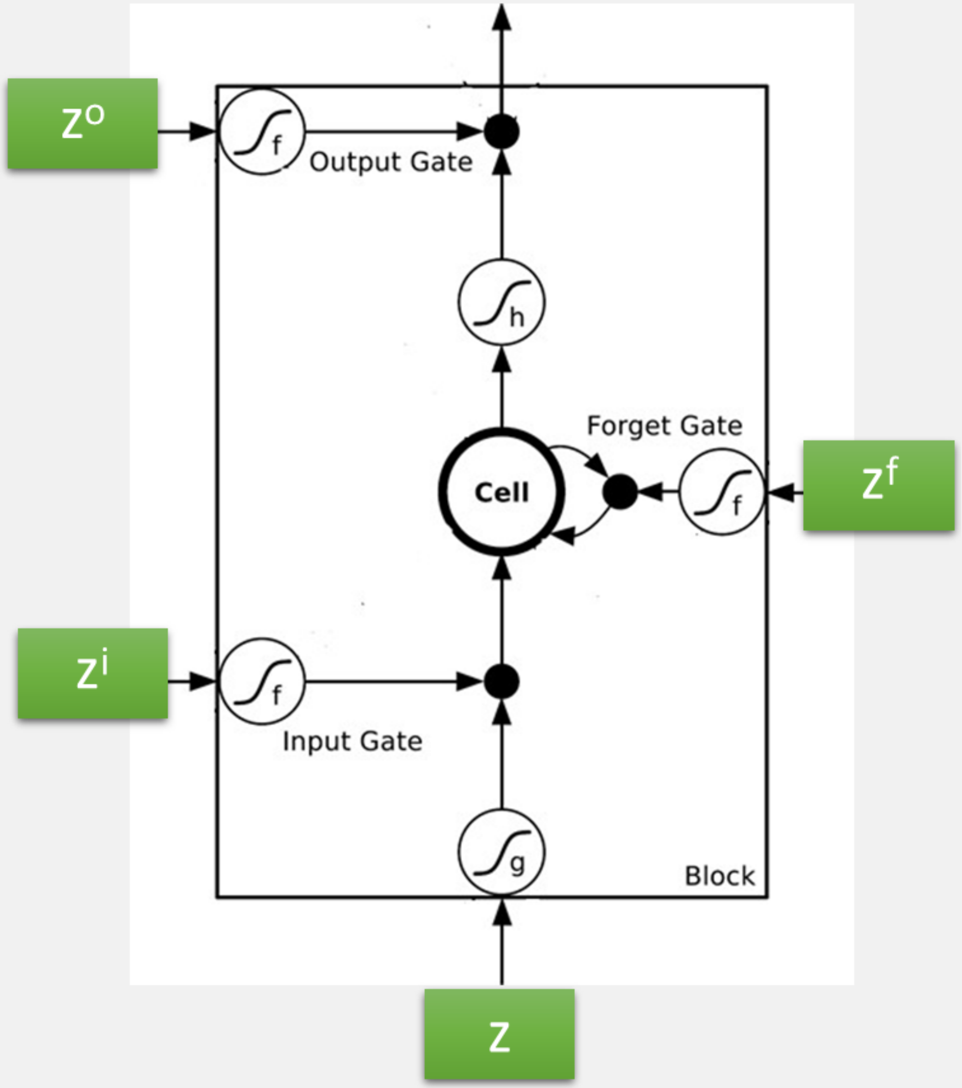
vector



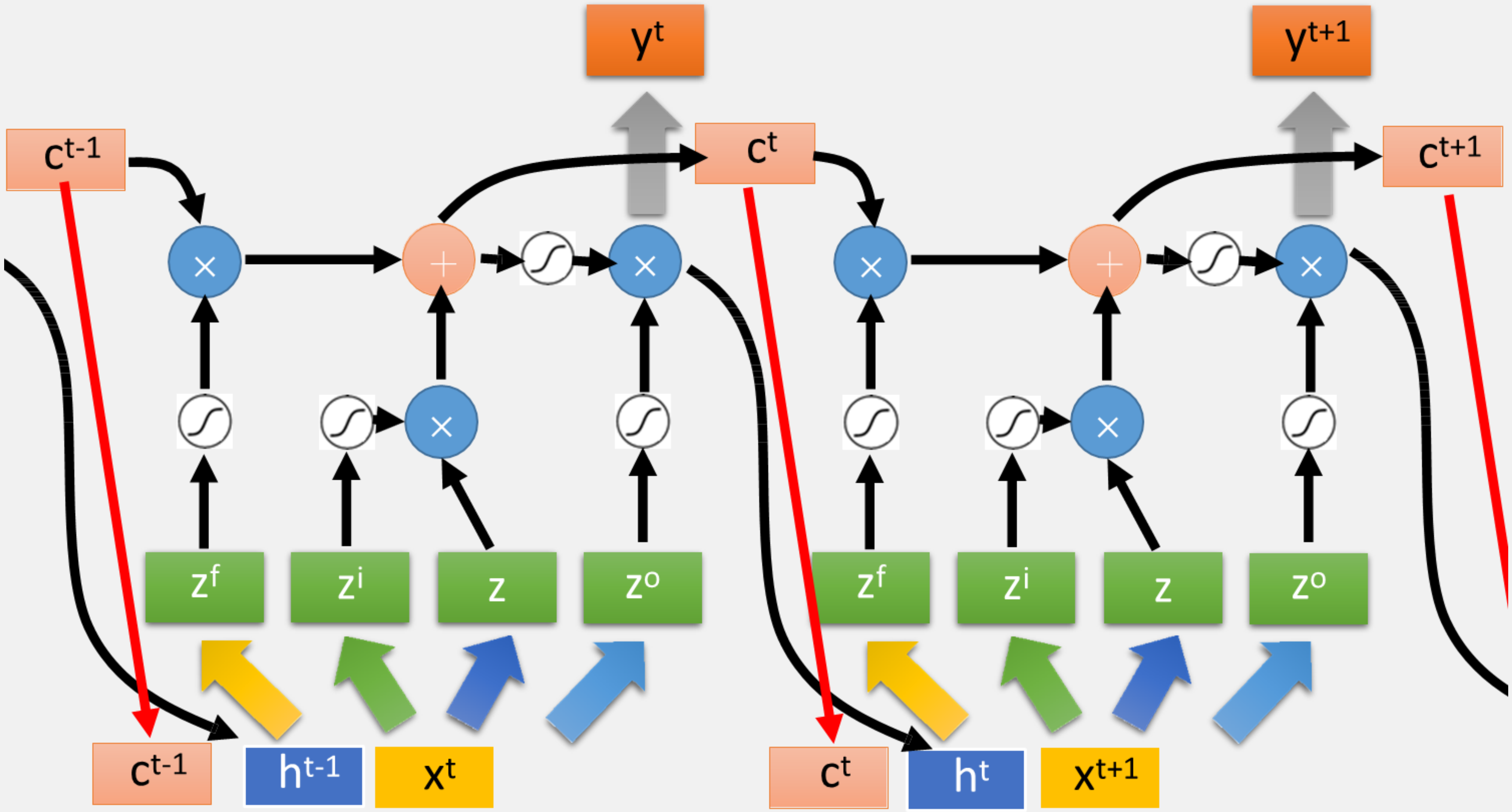
4 vectors

RNN and LSTM


y^t
c^{t-1}
z^f
z^i
z
z^o
x^t


z^o
Output Gate
Forget Gate
Cell
z^f
z^i
Input Gate
Block
z

RNN and LSTM



Reference

B

Seq2Seq

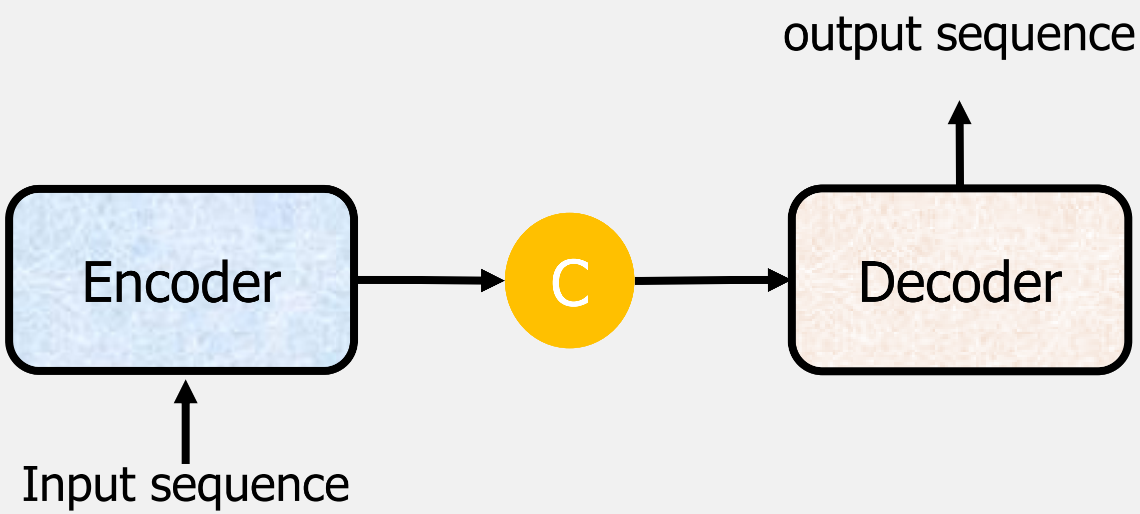
Seq2Seq

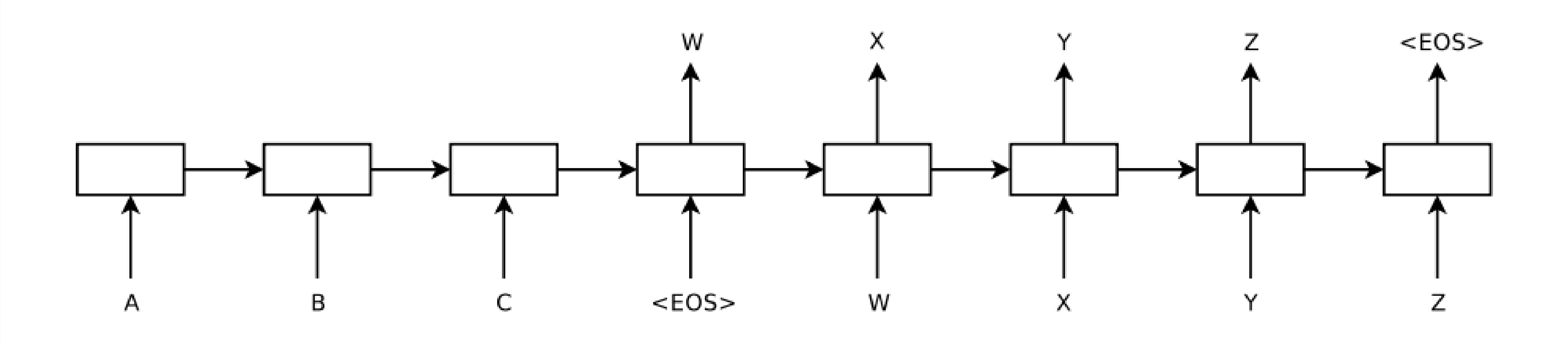
Seq2Seq is built by two RNN or LSTM models

The first model, encoder, is to convert the sequence with length M into a vector with a fixed length

The second model, decoder, is to reconstruct the vector back to a sequence with length N

Seq2Seq





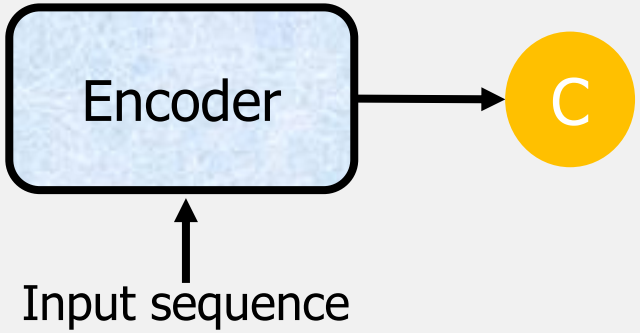
2014, I Sutskever et al., Sequence to Sequence Learning with Neural Networks

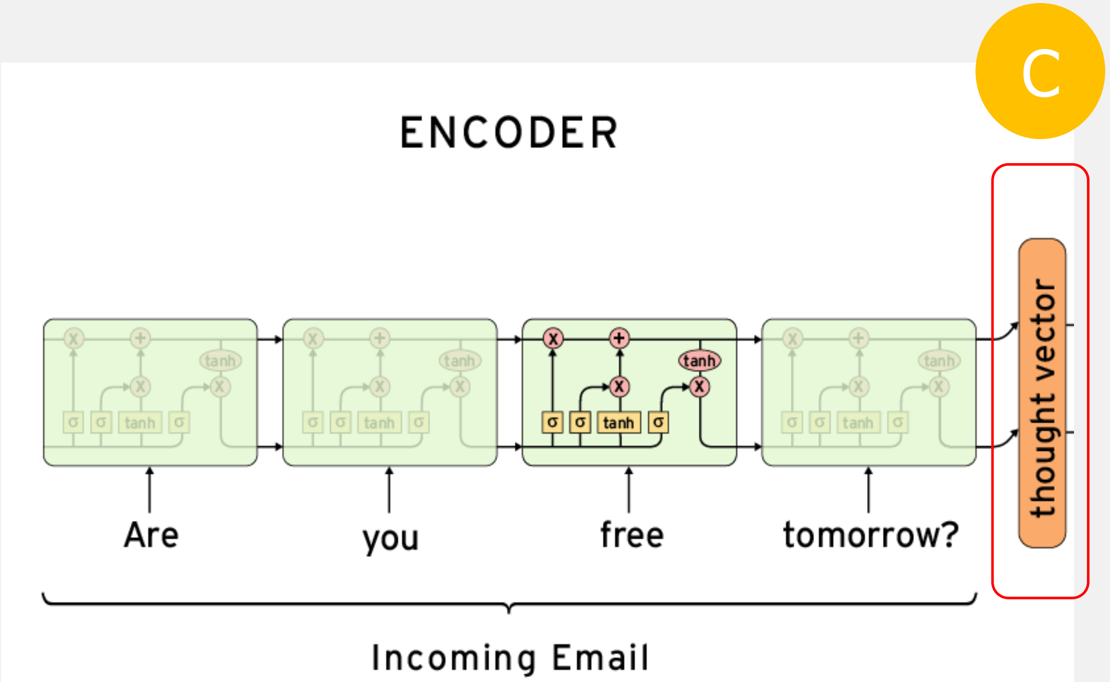
Seq2Seq

Encoder is responsible for conversion of input sequence into a vector

The vector is context vector composed of important information of input sequence

It can be done by RNN or LSTM

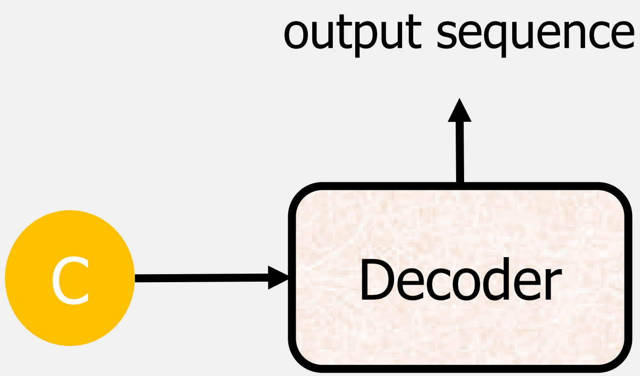


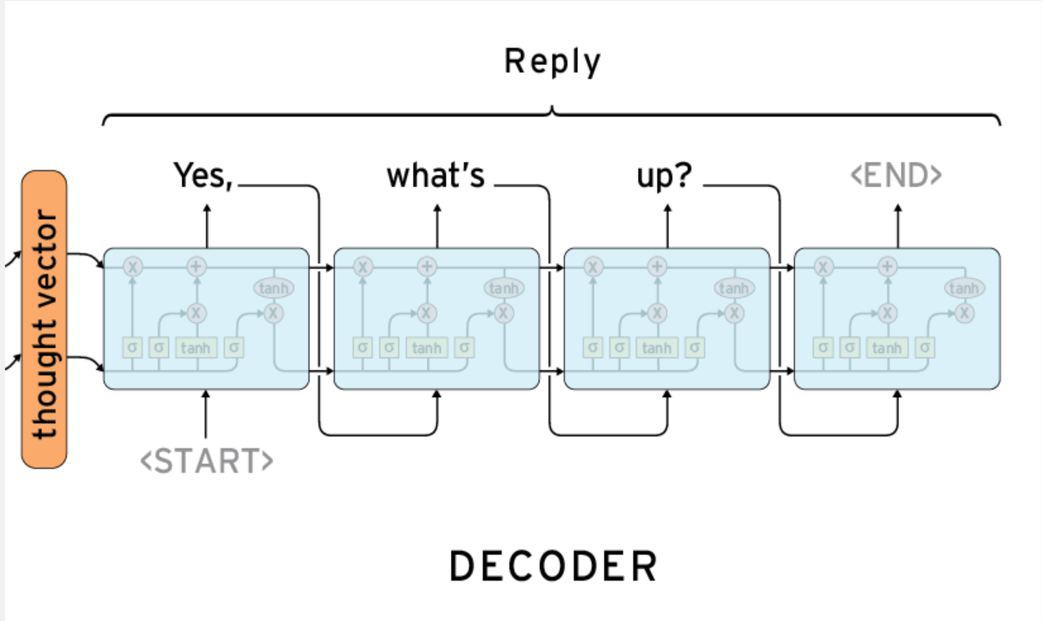


Seq2Seq

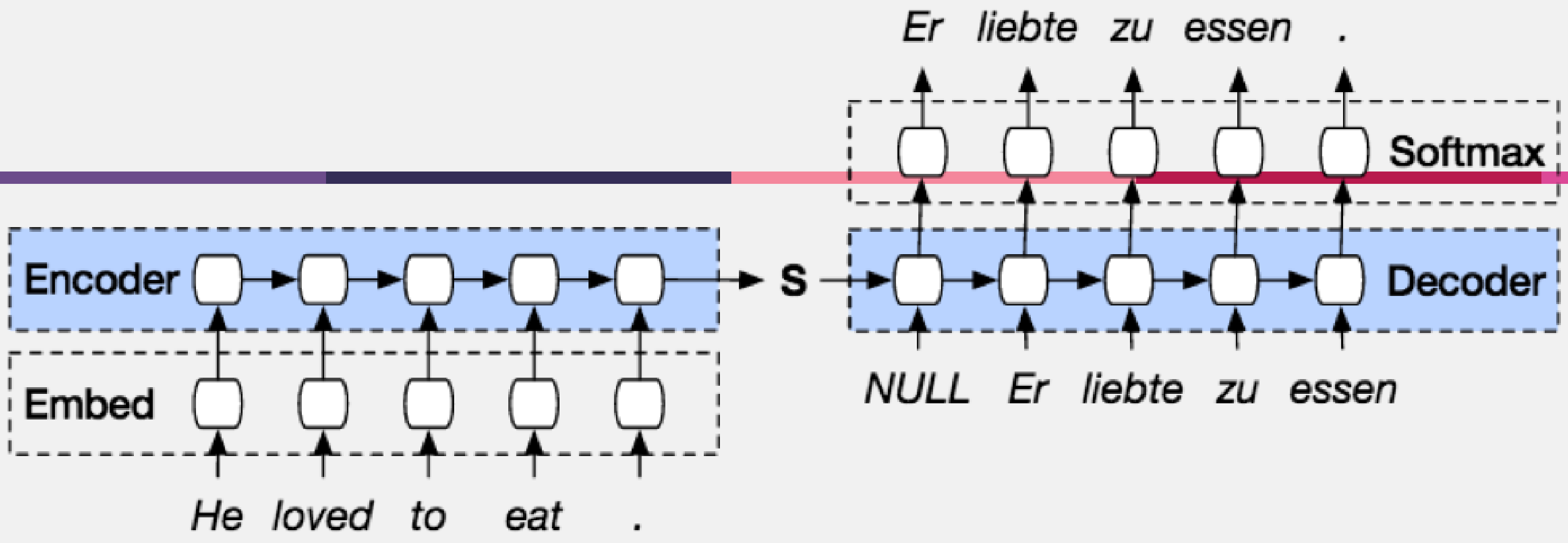
Decoder is responsible for word generation from context vector

It also can be done by RNN or LSTM

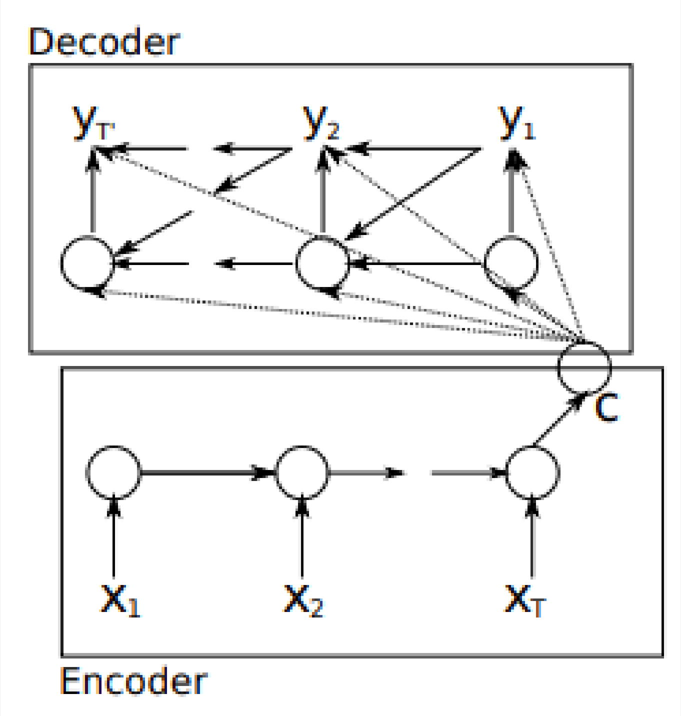




Seq2Seq



Example



$$p(y_1, \dots y_{T'} | x_1, \dots x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \dots y_{t-1})$$

Reference

C

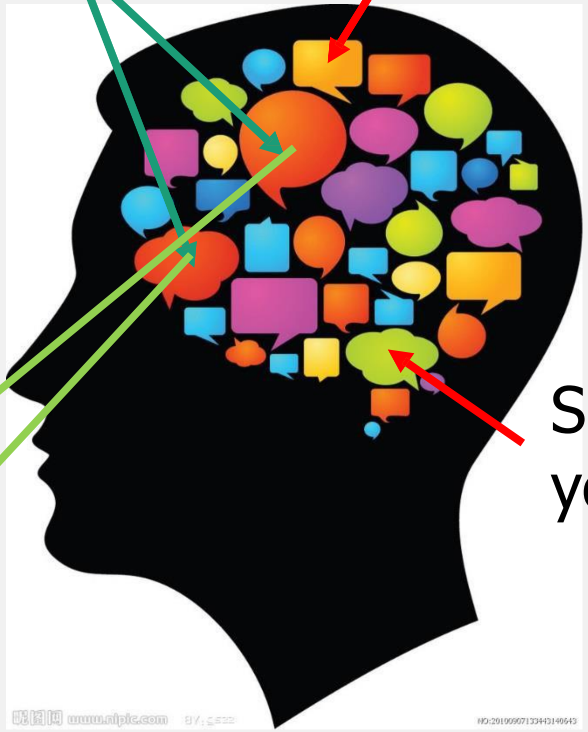
Attention

Attention

What you learned in these lectures

Breakfast today

What is deep learning?

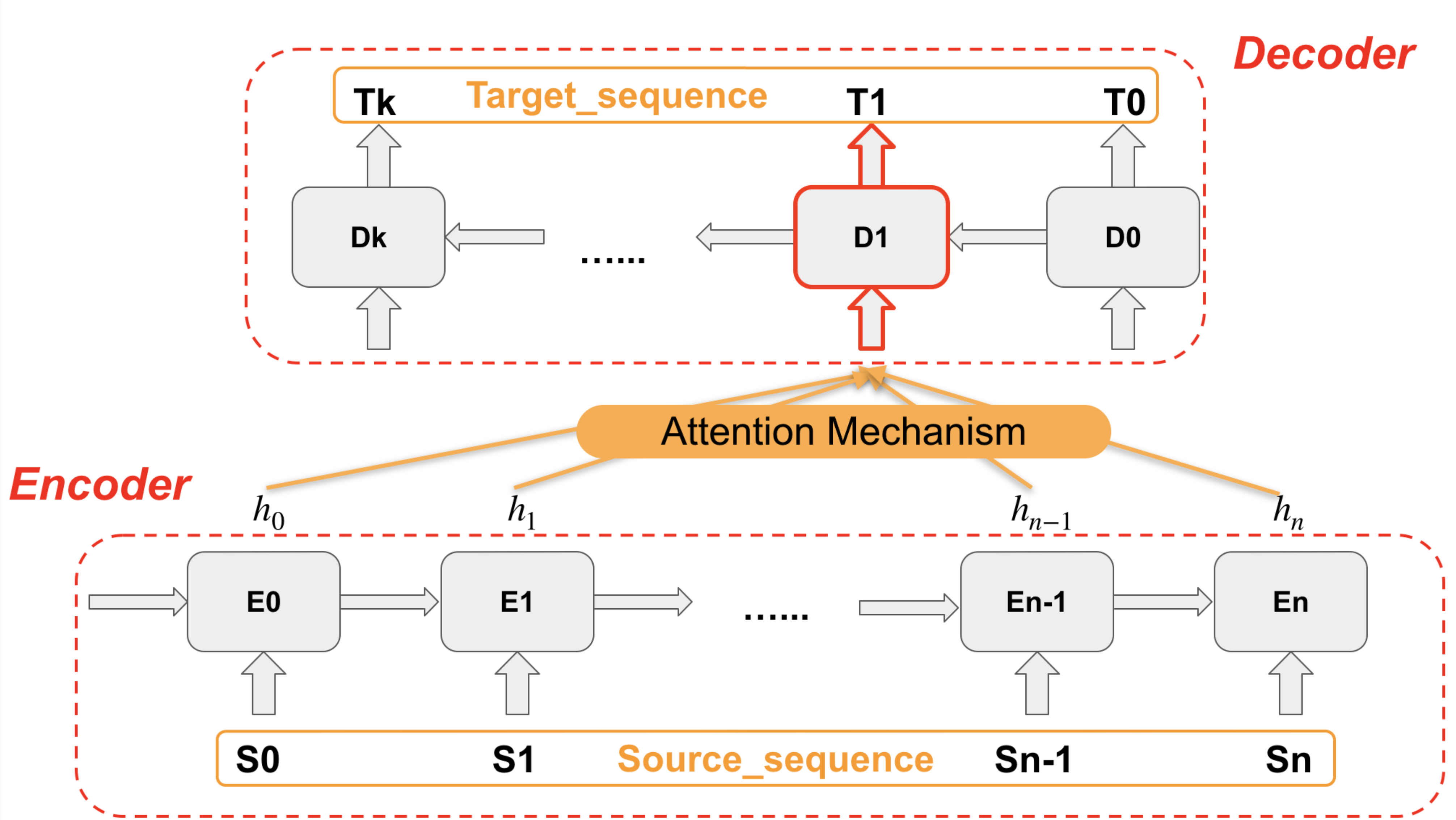


Summer vacation 10 years ago

Answer ← Organize

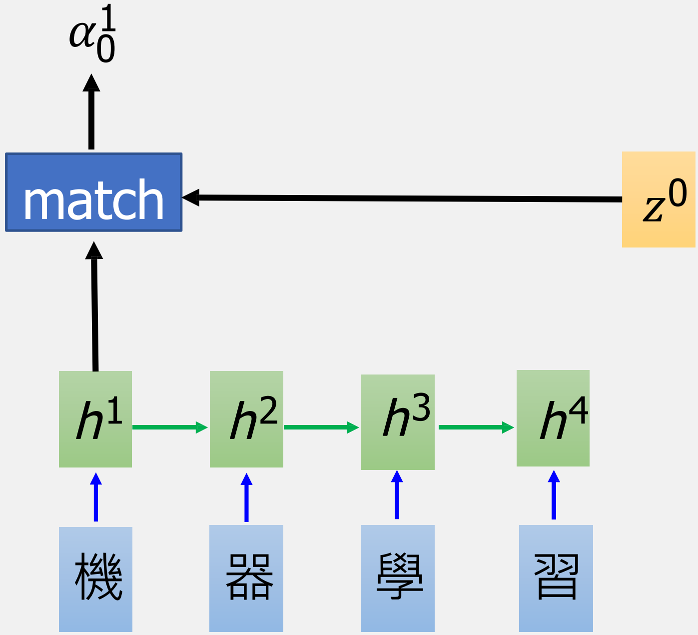
[Reference]

Attention

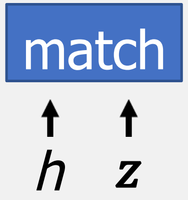

Decoder
Tk
Target_sequence
T1
T0
Dk
……
D1
D0
Attention Mechanism
Encoder
h_0
h_1
h_{n-1}
h_n
E0
E1
……
En-1
En
S0
S1
Source_sequence
Sn-1
Sn

Attention

Attention-based model



Jointly learned with other part of the network

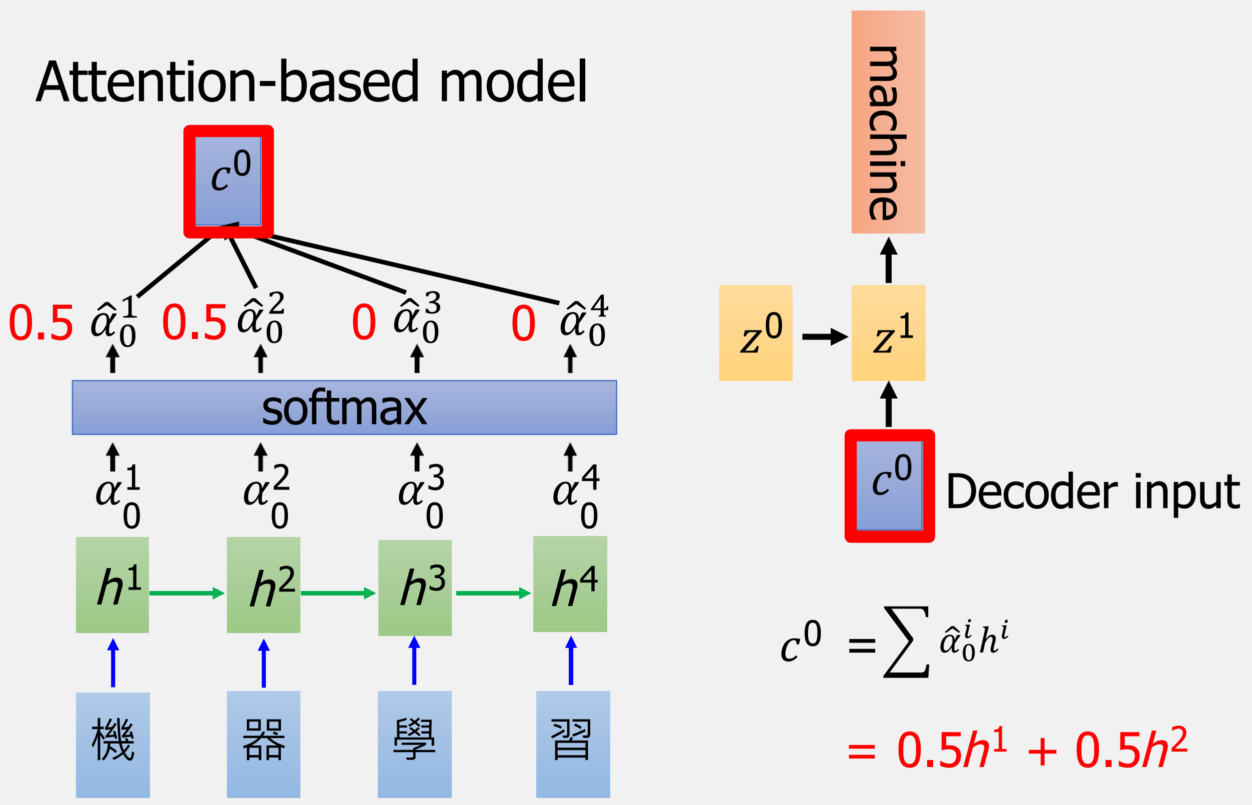


What is match ?

Design by yourself

- Cosine similarity of z and h
- Small NN whose input is z and h, output a scalar
- $\alpha = h^T W z$

Attention

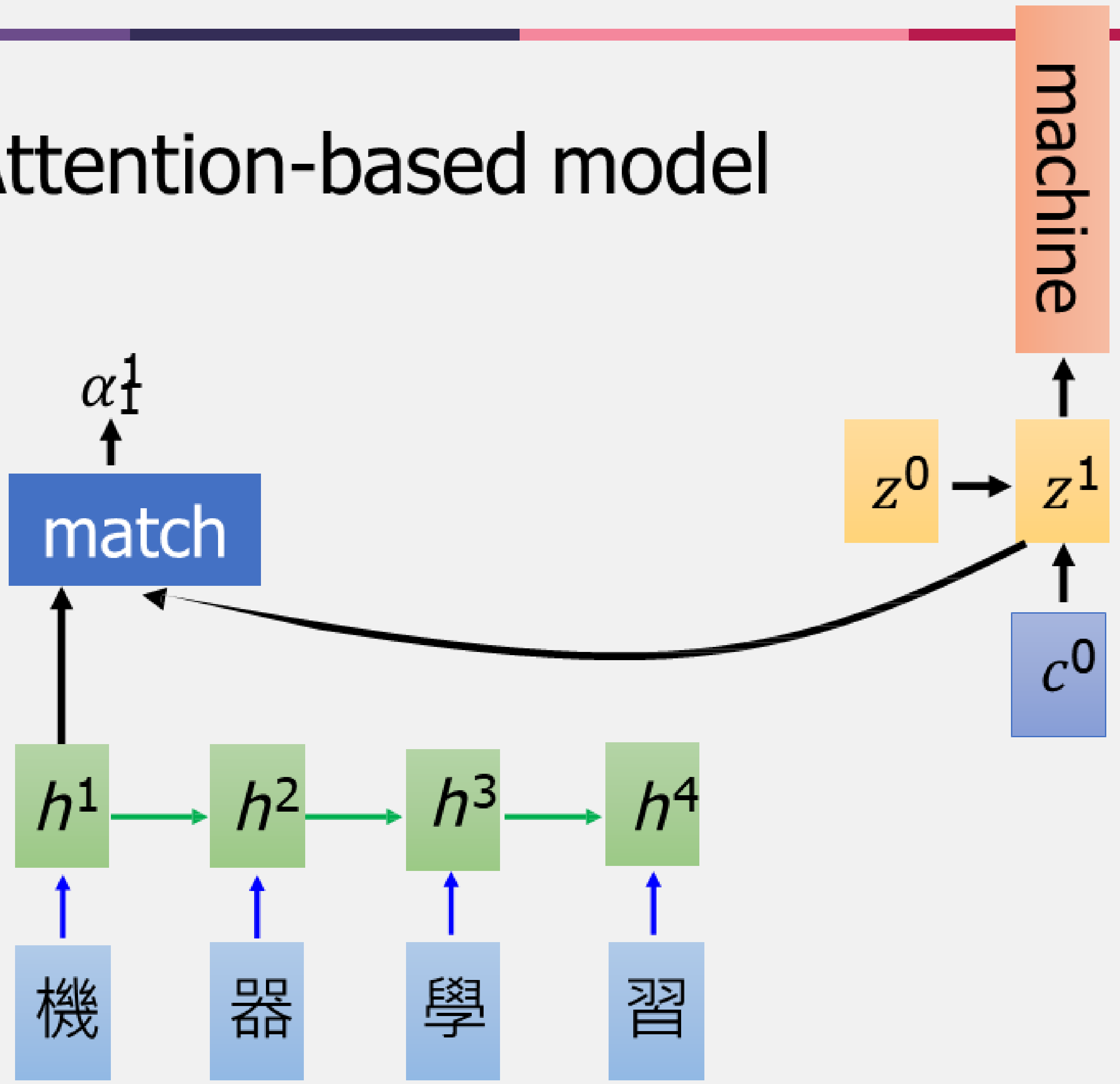


$$c^0 = \sum \hat{\alpha}_0^i h^i$$

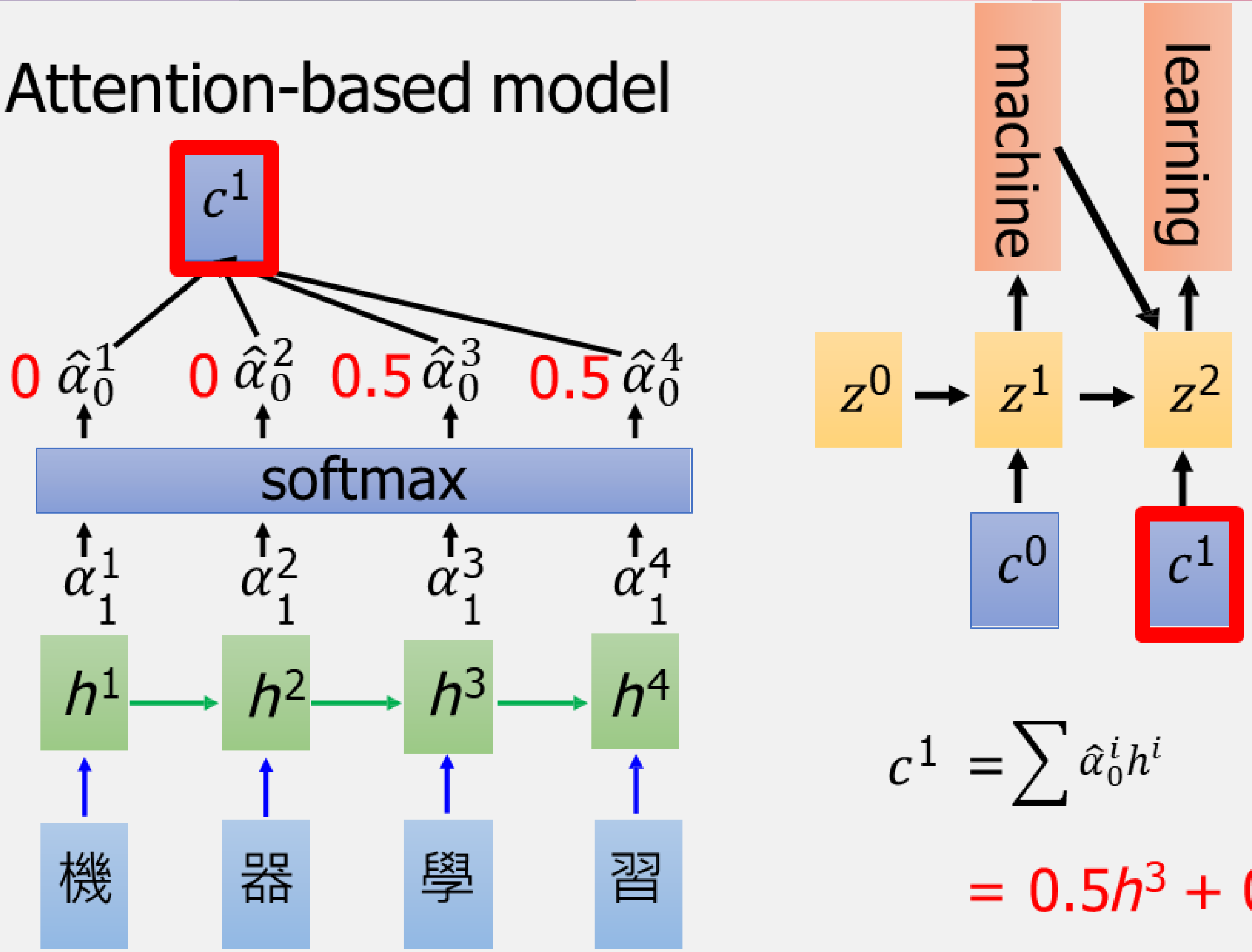
$$= 0.5h^1 + 0.5h^2$$

Attention

Attention-based model



Attention

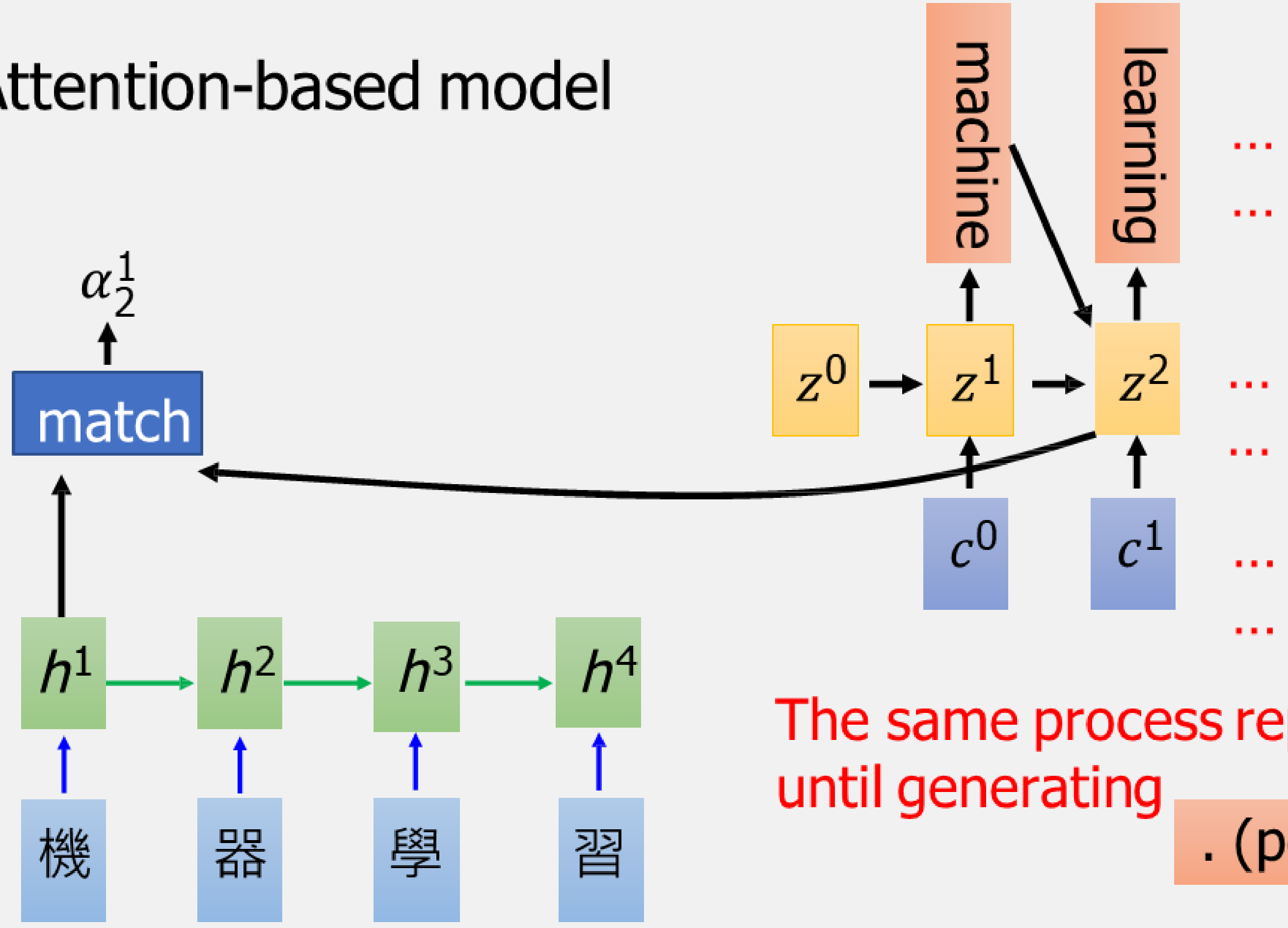


$$c^1 = \sum \hat{\alpha}_0^i h^i$$

$$= 0.5h^3 + 0.5h^4$$

Attention

Attention-based model



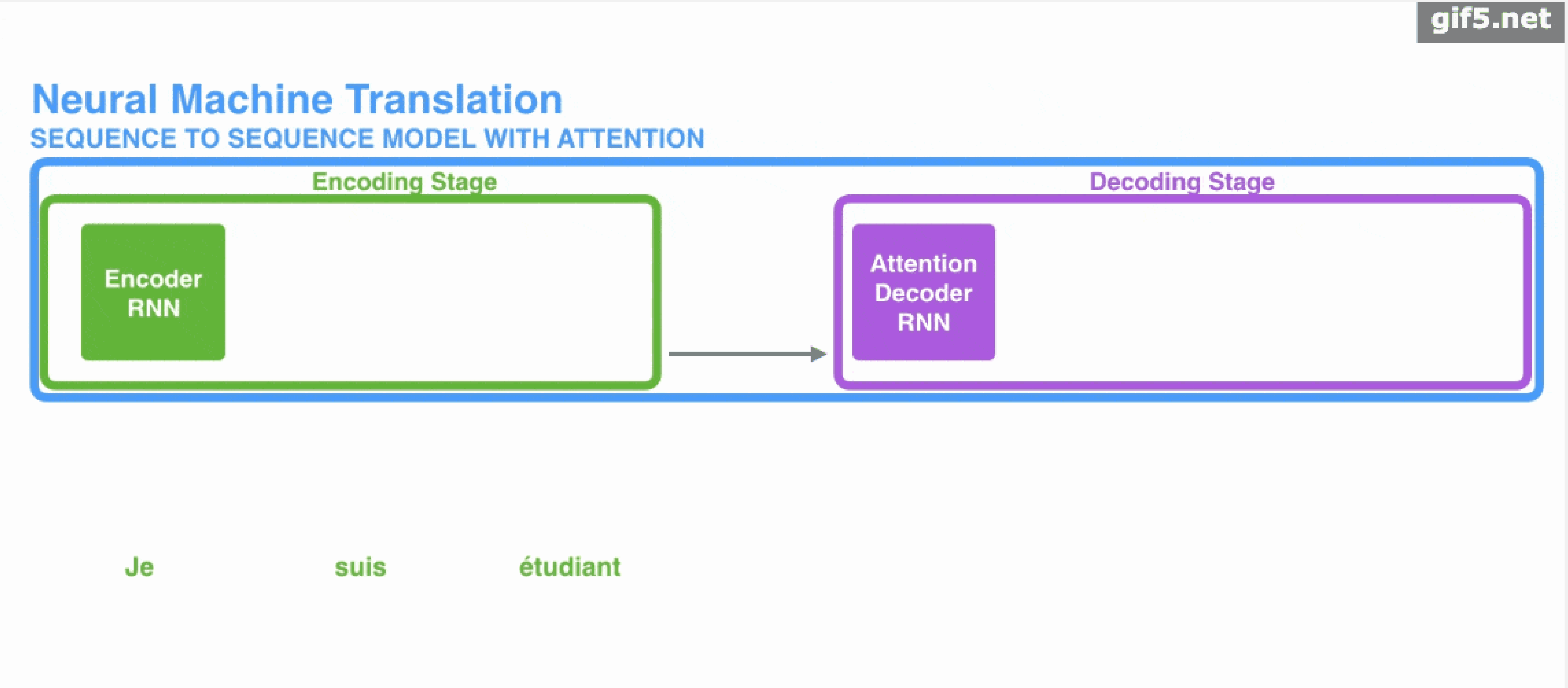
The same process repeat until generating . (period)

Reference

D

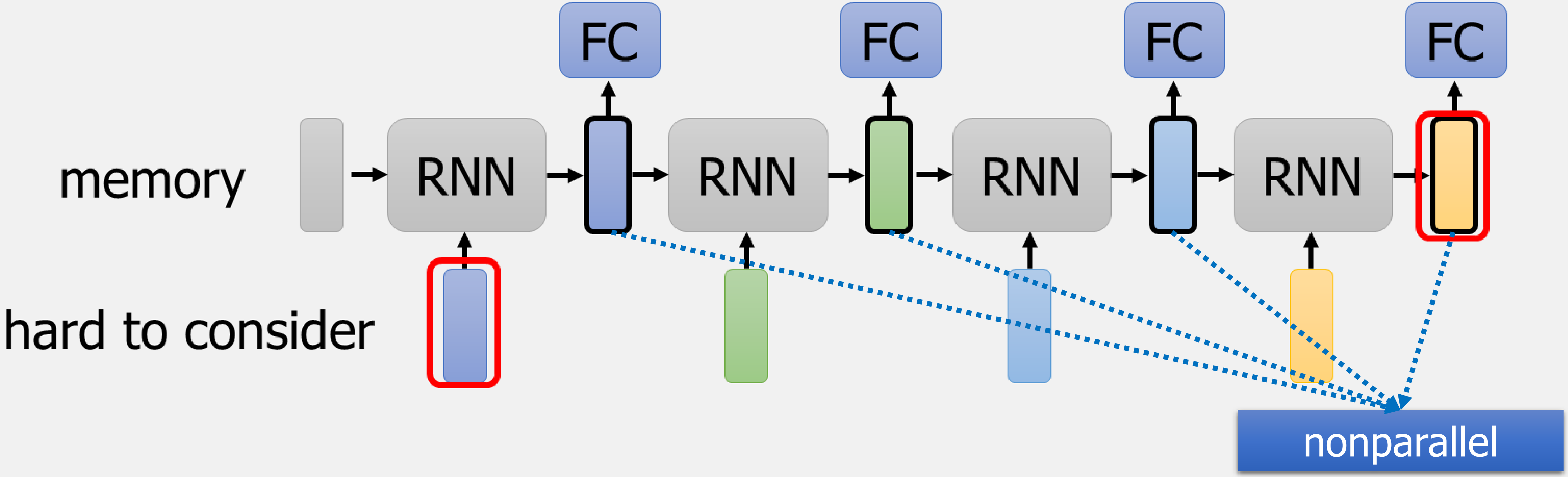
Self-attention

Attention

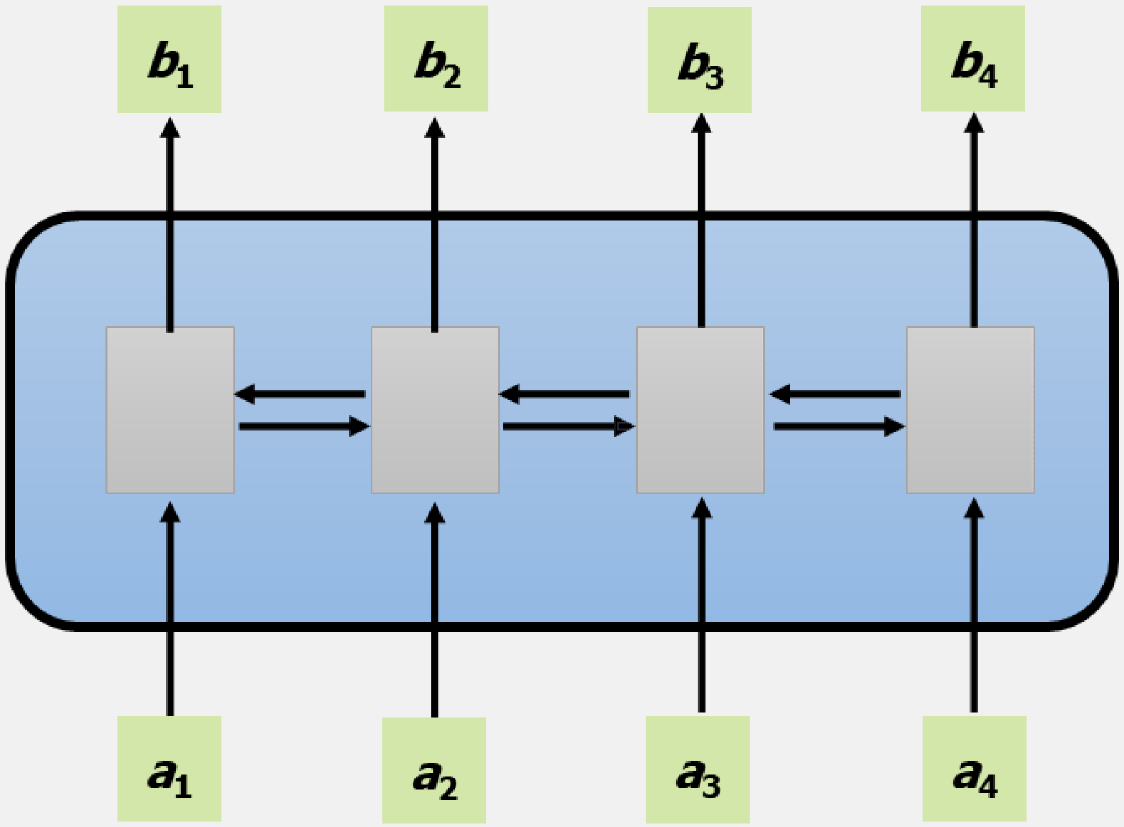


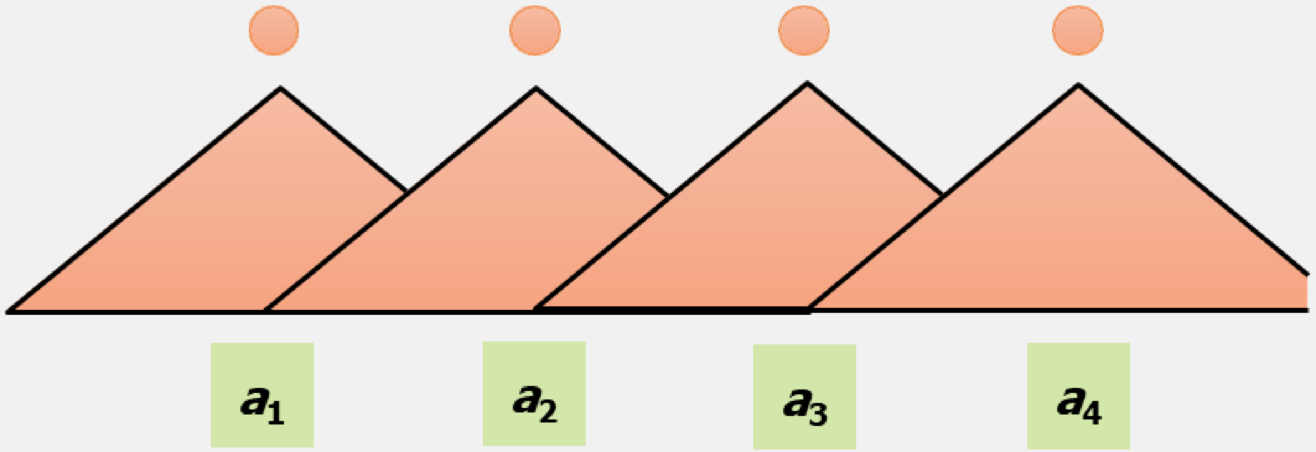
[Reference](#)

Attention


FC
FC
FC
FC
memory
RNN
RNN
RNN
RNN
hard to consider
nonparallel

Attention

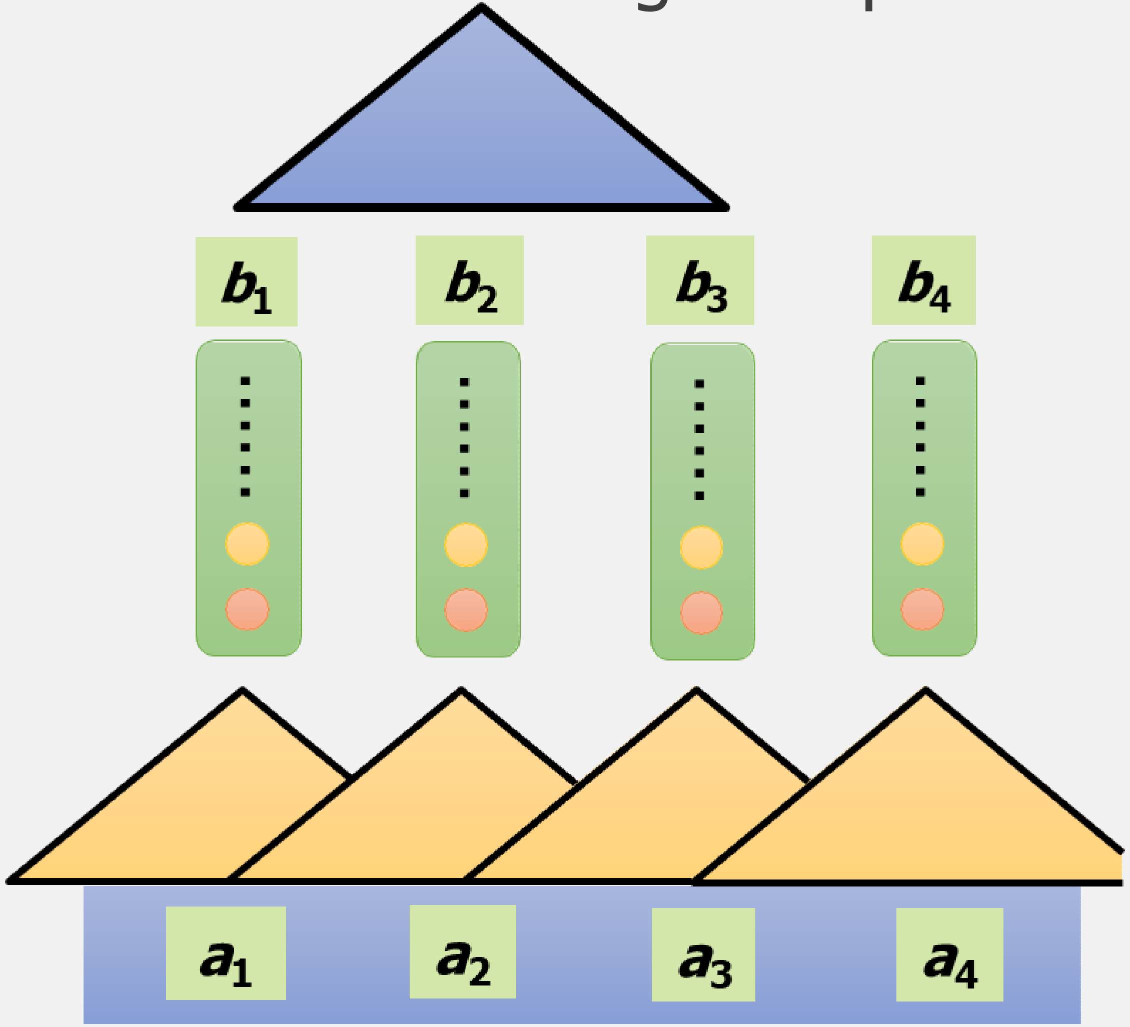




Using CNN to replace RNN

Attention

Filters in higher layer can consider longer sequence

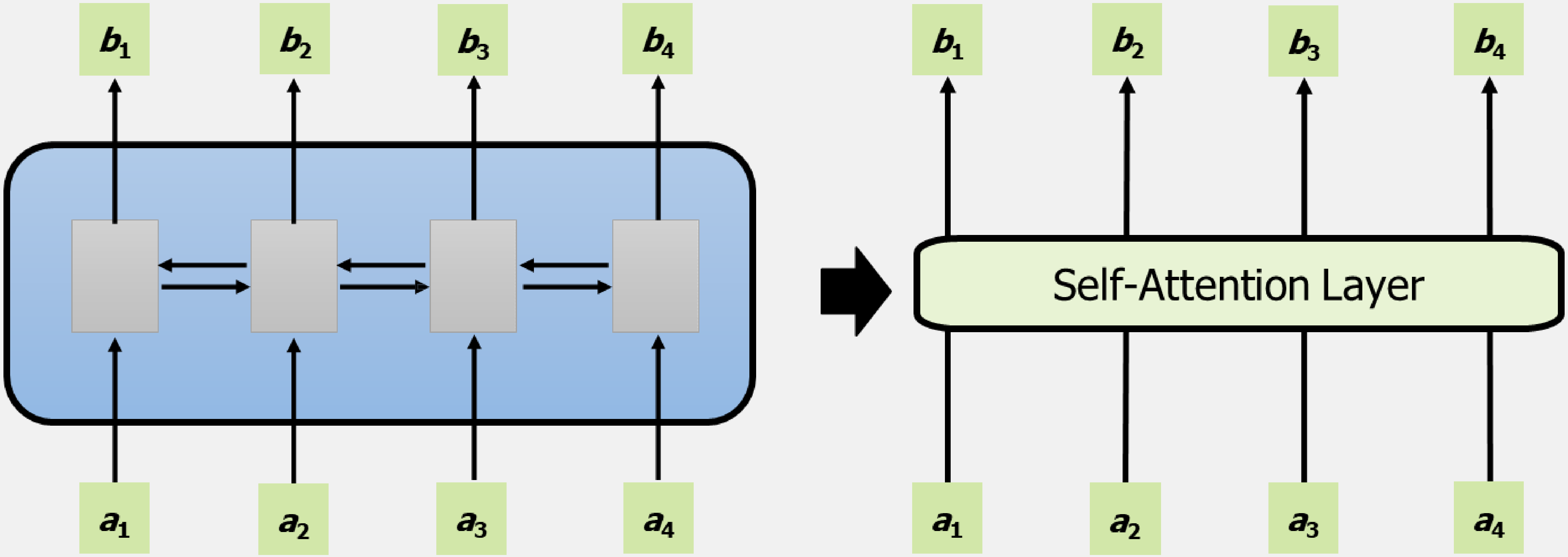


Attention

Self-attention

b_i is obtained based on the whole input sequence

b_1, b_2, b_3, b_4 can be parallelly computed

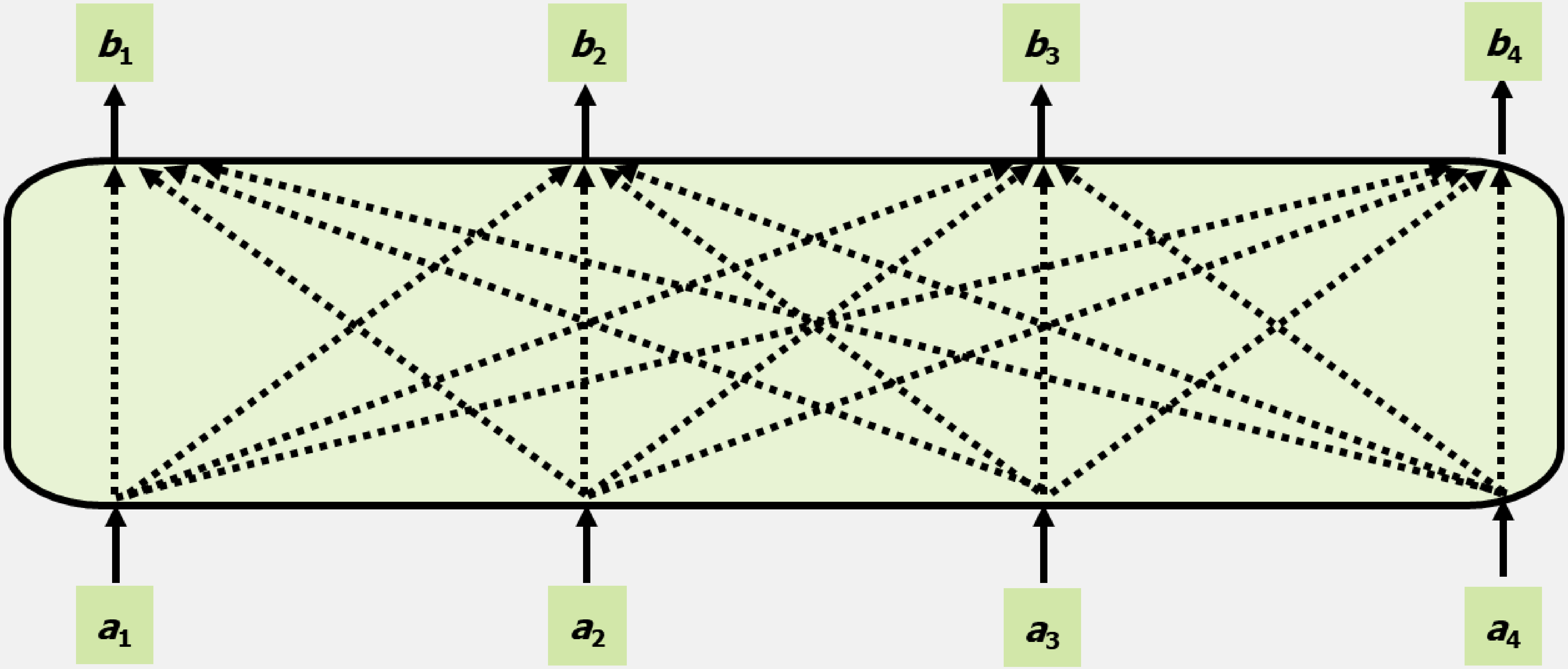


You can try to replace any thing that has been done by RNN with self-attention

[Reference]()

Attention

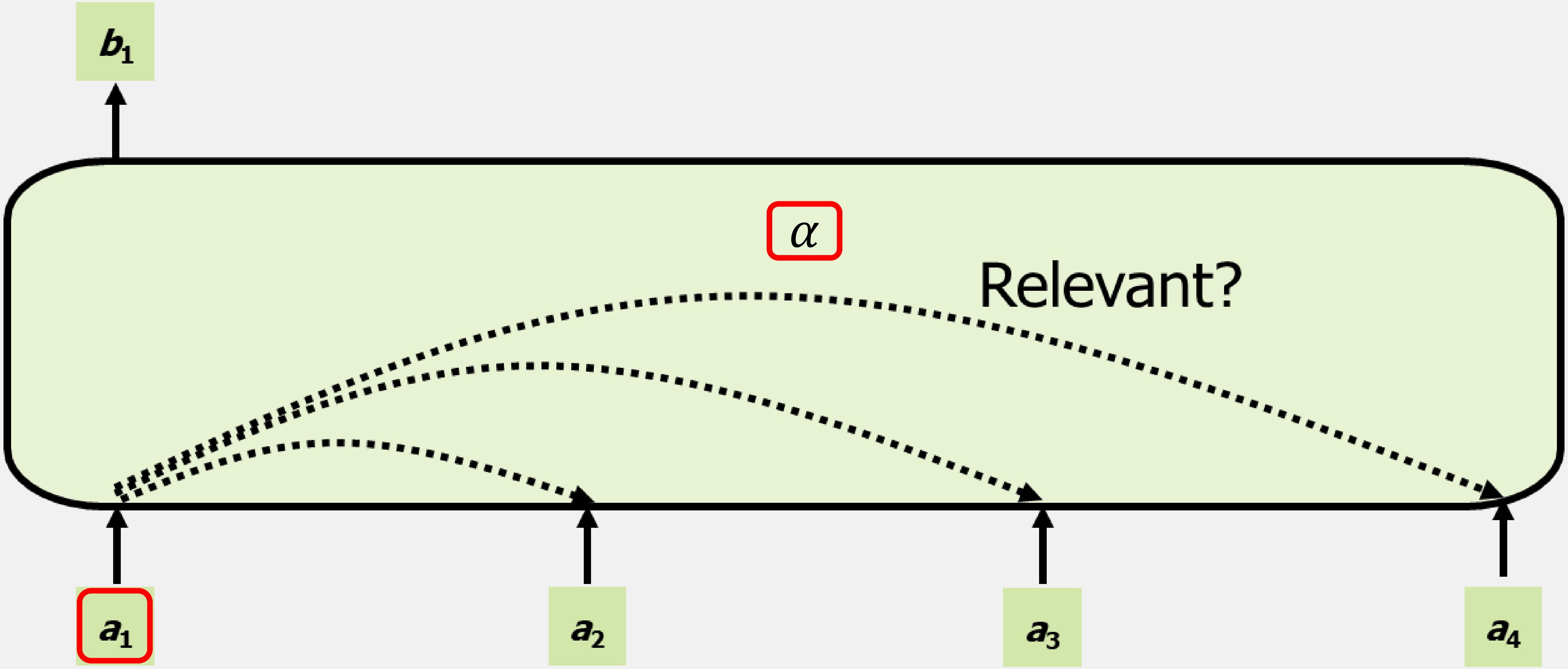
Self-attention



Can be either **input** or a **hidden layer**

Attention

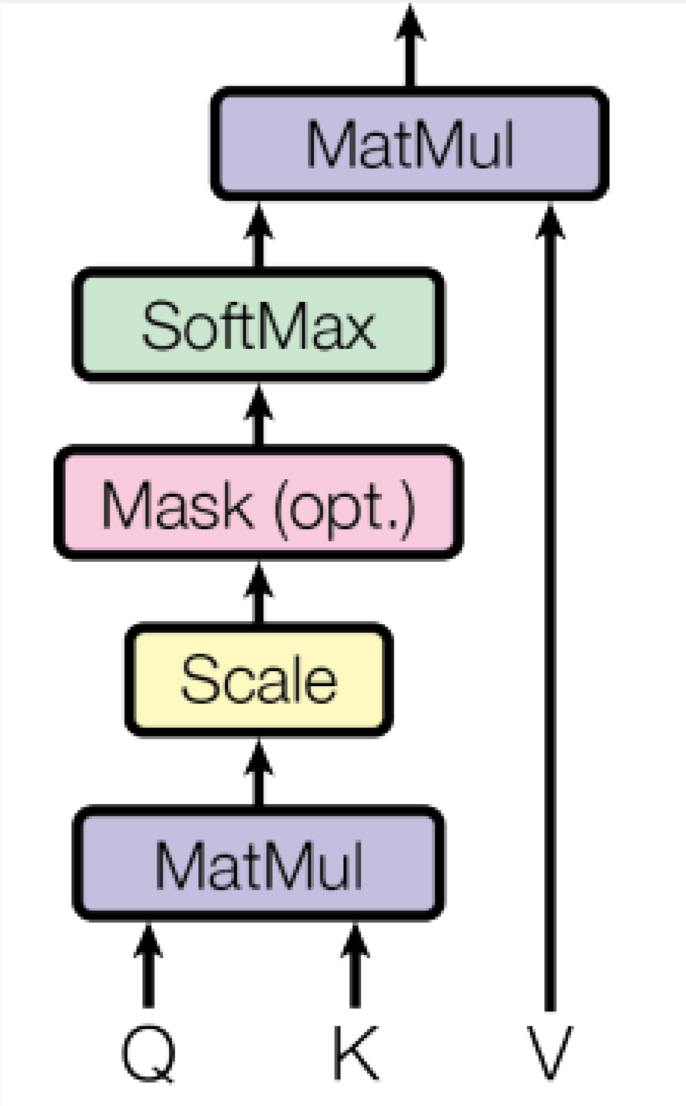
Self-attention

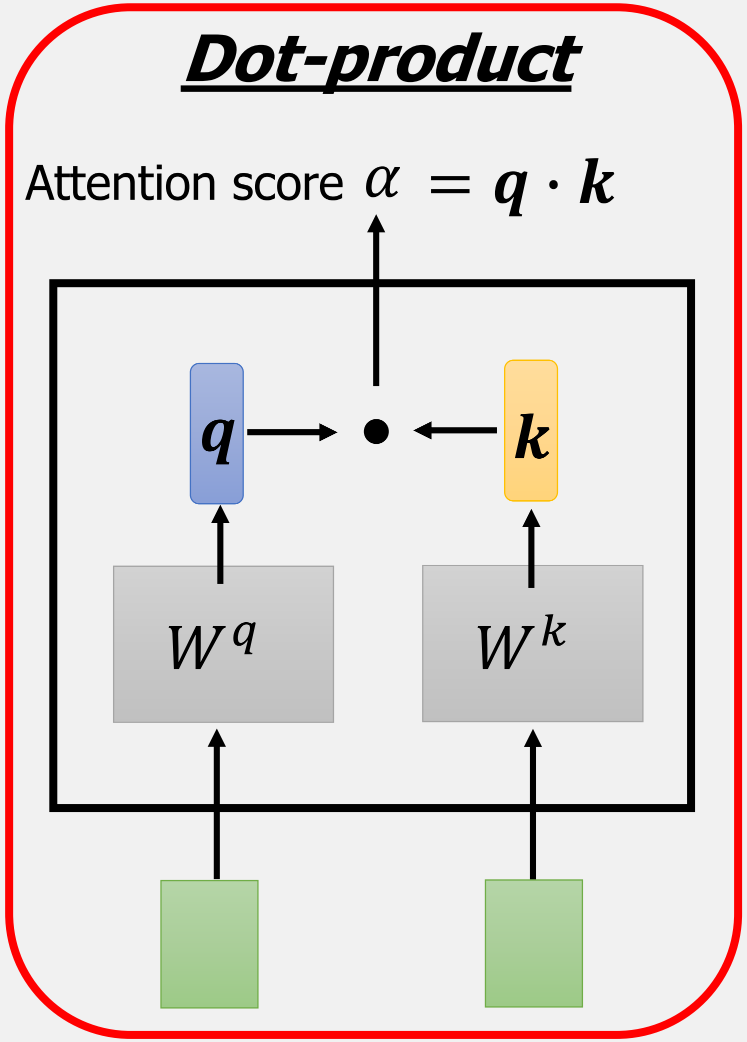


Find the relevant vectors in a sequence

Attention

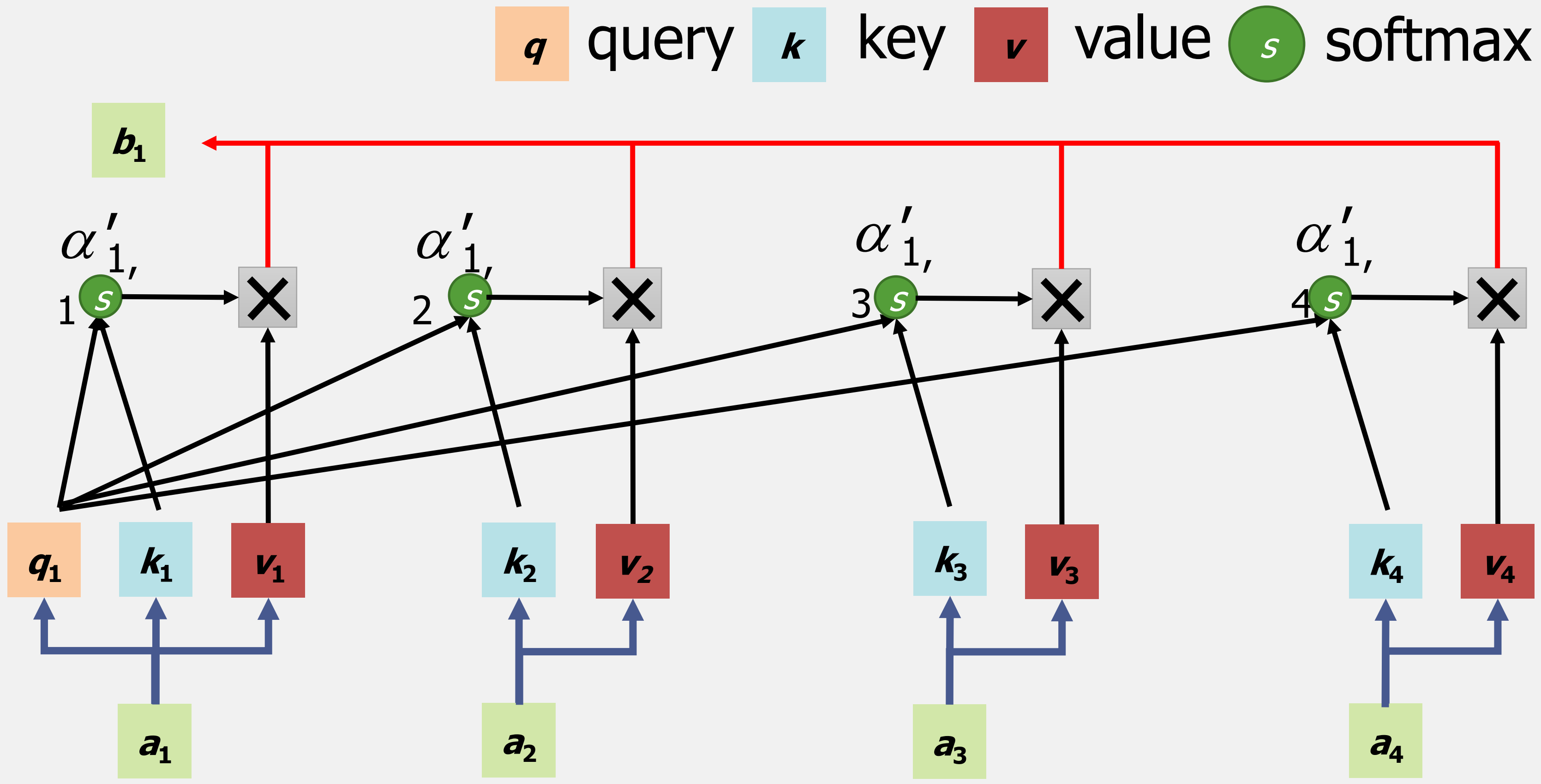
Self-attention





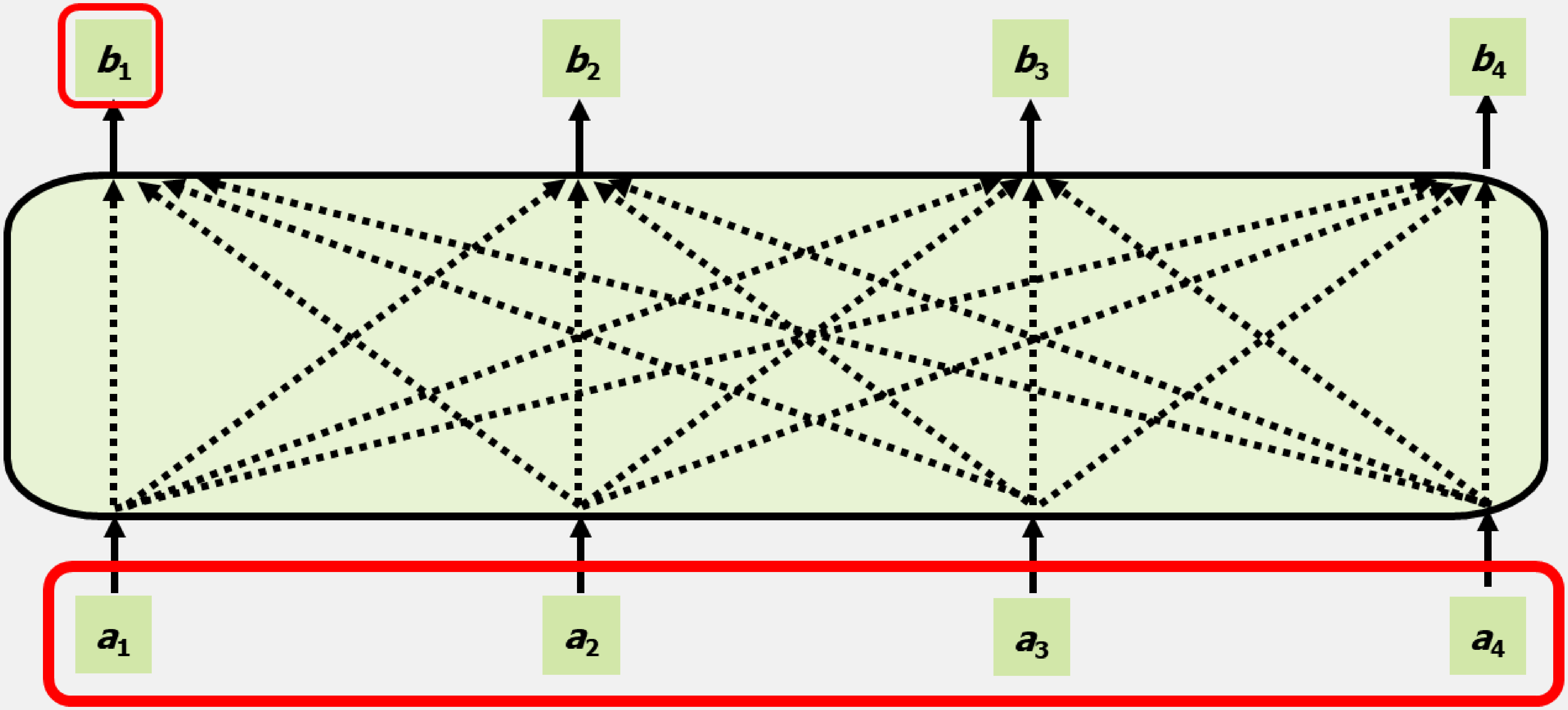
Attention

Self-attention



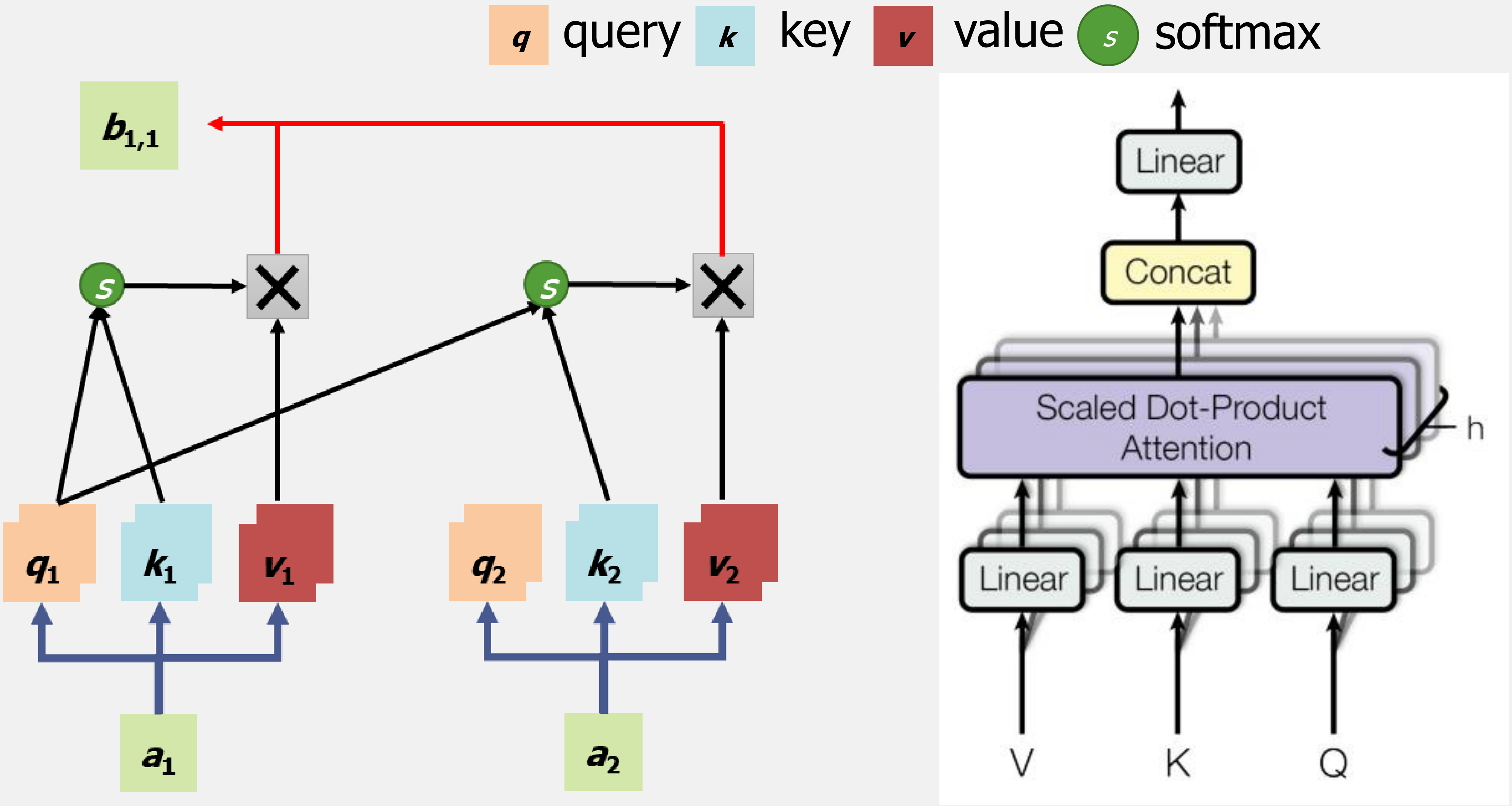
$q_1 = W_q \cdot a_1$ $k_1 = W_k \cdot a_1$ $k_2 = W_k \cdot a_2$ $k_3 = W_k \cdot a_3$ $k_4 = W_k \cdot a_4$

Attention
Self-attention



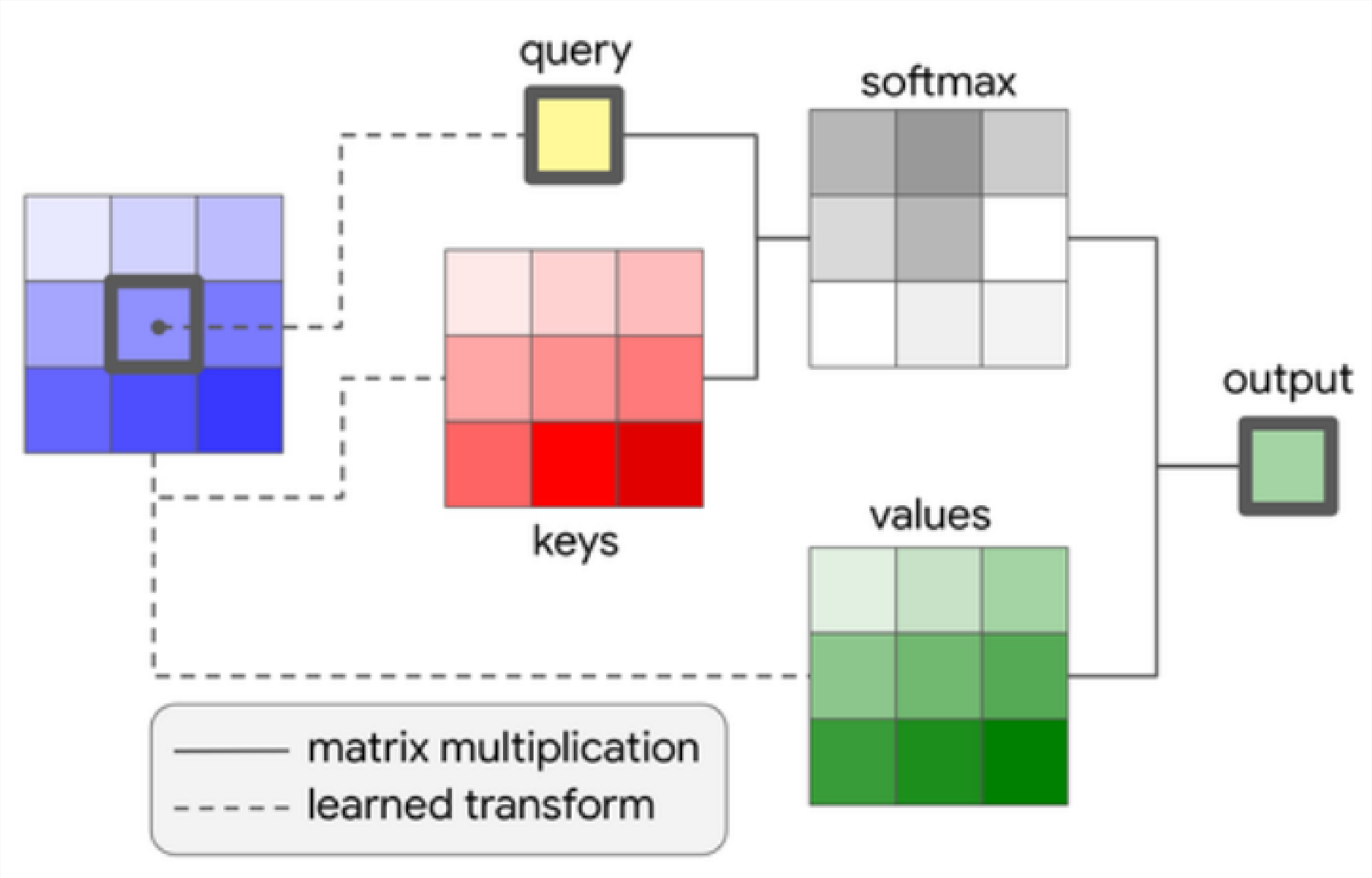
Attention

Self-attention



Attention

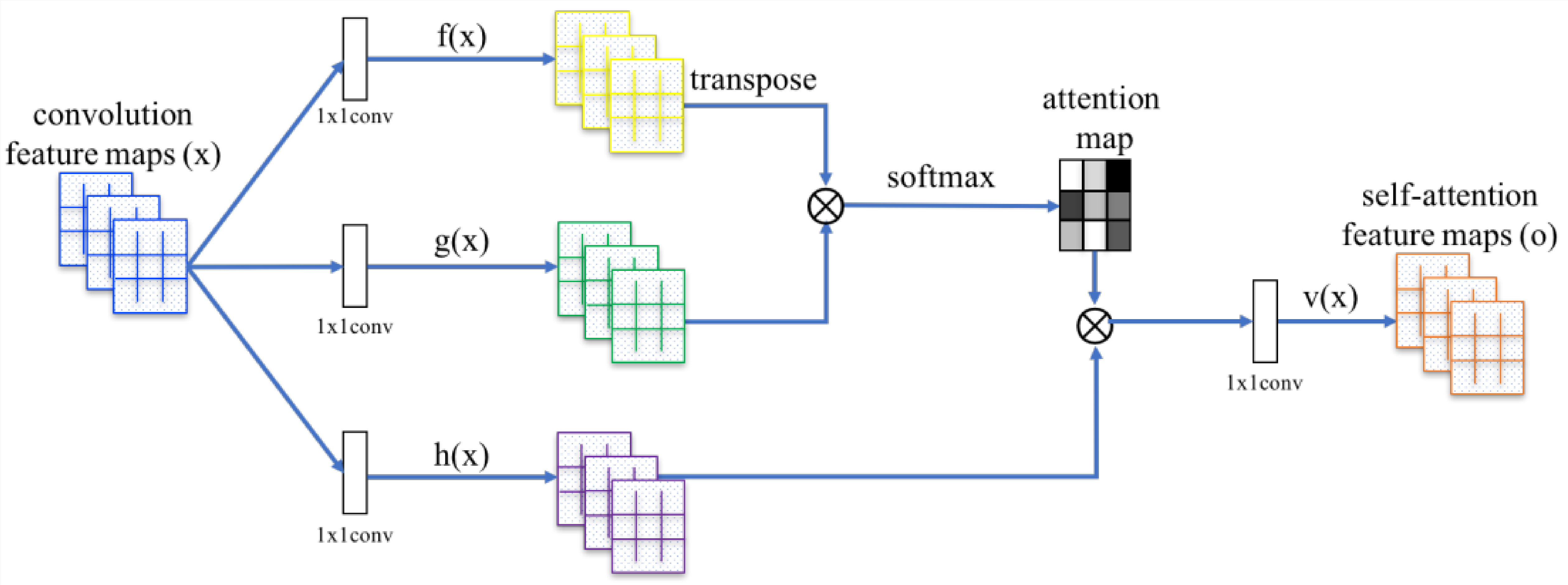
Self-attention in Vision model



[Reference](#)

Attention

Self-attention GAN

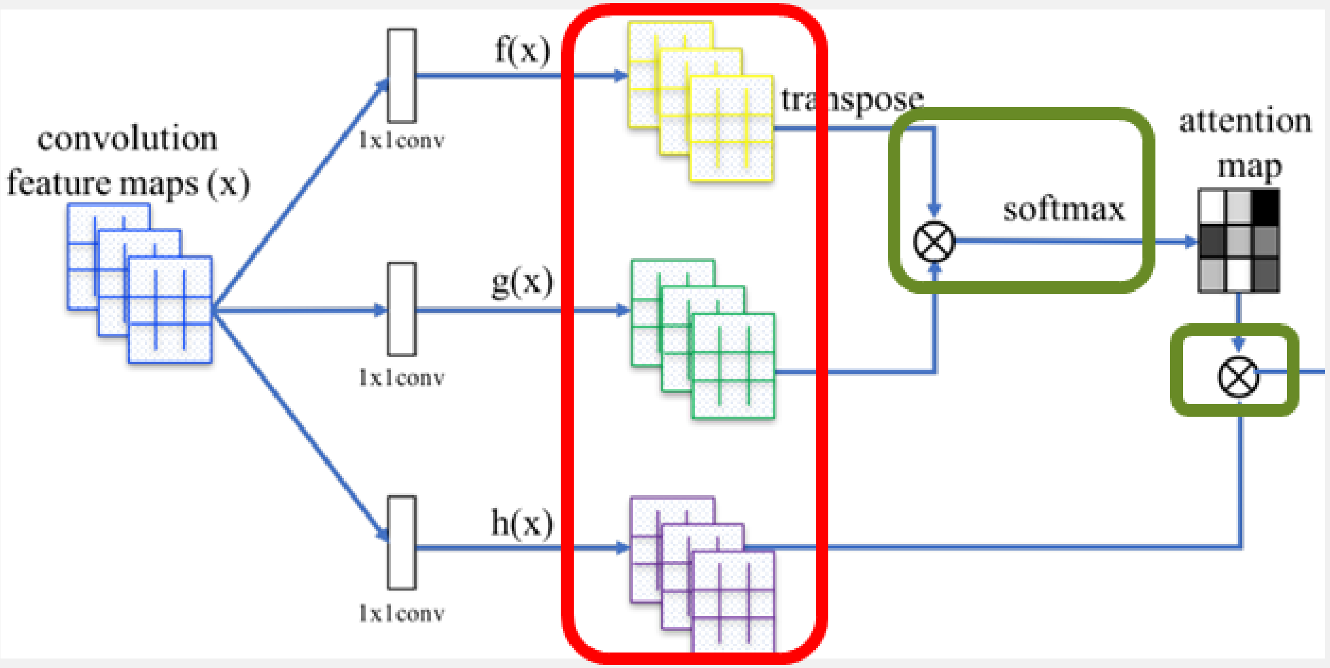


[Reference](#)

Attention

Self-attention GAN

```python
def __init__(self, in_channels):
    super(Self_Attn, self).__init__()
    self.in_channels = in_channels
    self.snconv1x1_theta = snconv2d(in_channels=in_channels, out_channels=in_chan
    self.snconv1x1_phi = snconv2d(in_channels=in_channels, out_channels=in_channe
    self.snconv1x1_g = snconv2d(in_channels=in_channels, out_channels=in_channels
    self.snconv1x1_attn = snconv2d(in_channels=in_channels//2, out_channels=in_ch
```

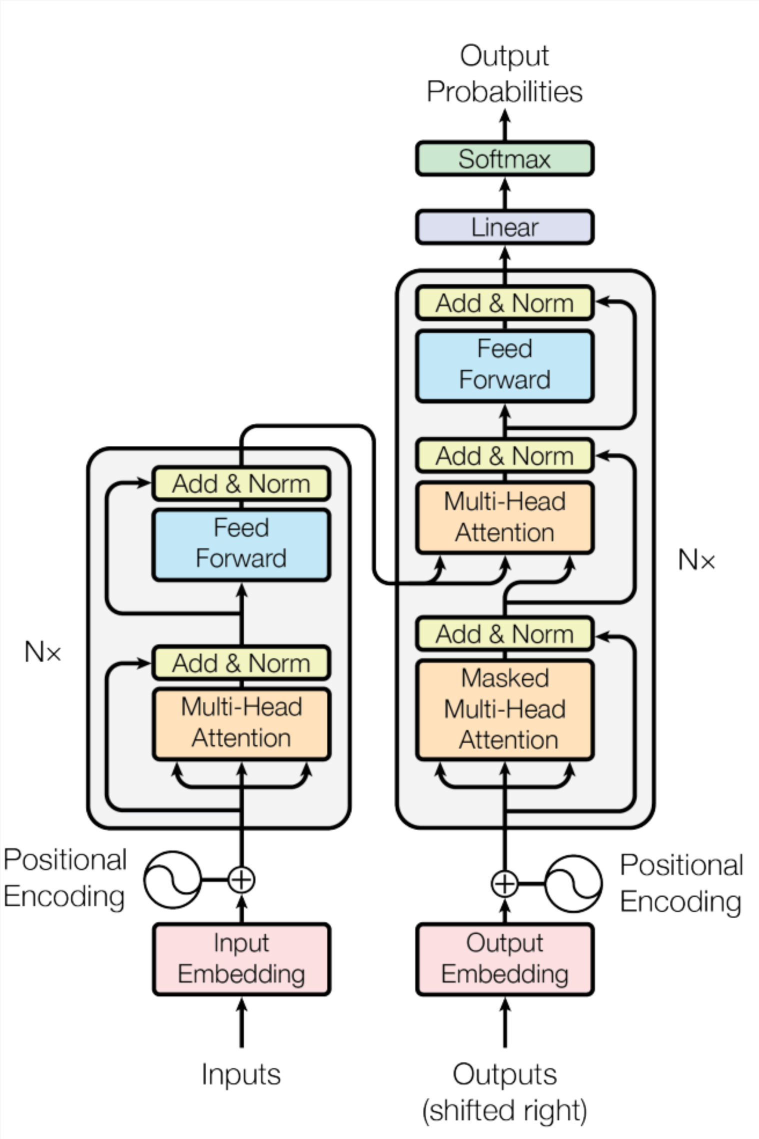



```python
# Theta path
theta = self.snconv1x1_theta(x)
theta = theta.view(-1, ch//8, h*w)
# Phi path
phi = self.snconv1x1_phi(x)
phi = self.maxpool(phi)
phi = phi.view(-1, ch//8, h*w//4)
# Attn map
attn = torch.bmm(theta.permute(0, 2, 1), phi)
attn = self.softmax(attn)
# g path
g = self.snconv1x1_g(x)
g = self.maxpool(g)
g = g.view(-1, ch//2, h*w//4)
# Attn_g
attn_g = torch.bmm(g, attn.permute(0, 2, 1))
attn_g = attn_g.view(-1, ch//2, h, w)
attn_g = self.snconv1x1_attn(attn_g)
# Out
out = x + self.sigma*attn_g
return out
```

Reference

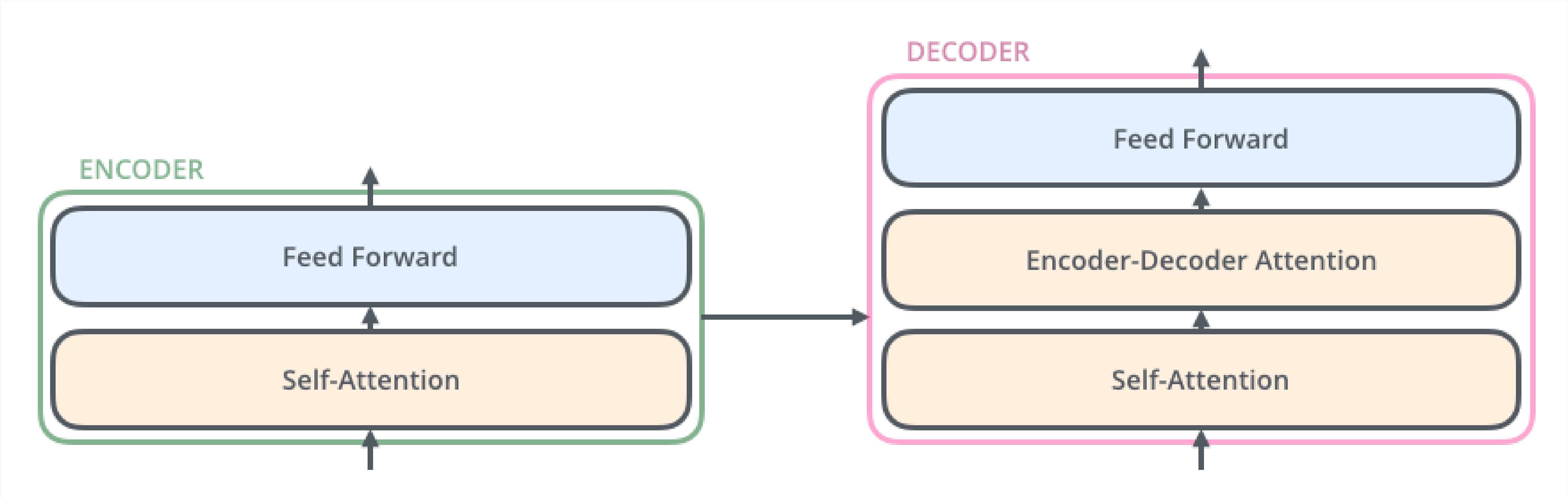
02
Transformer

Transformer



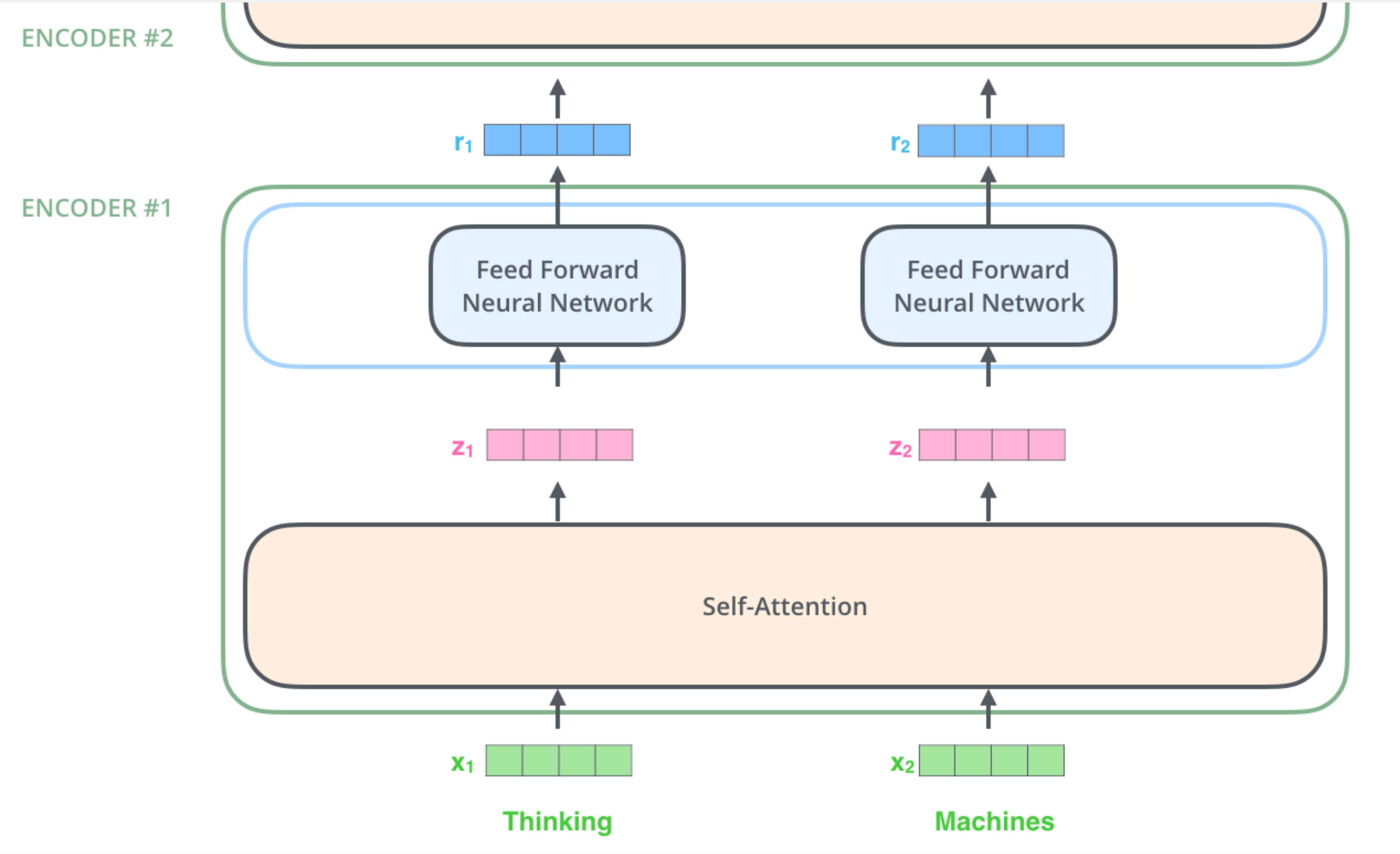
Reference

Transformer

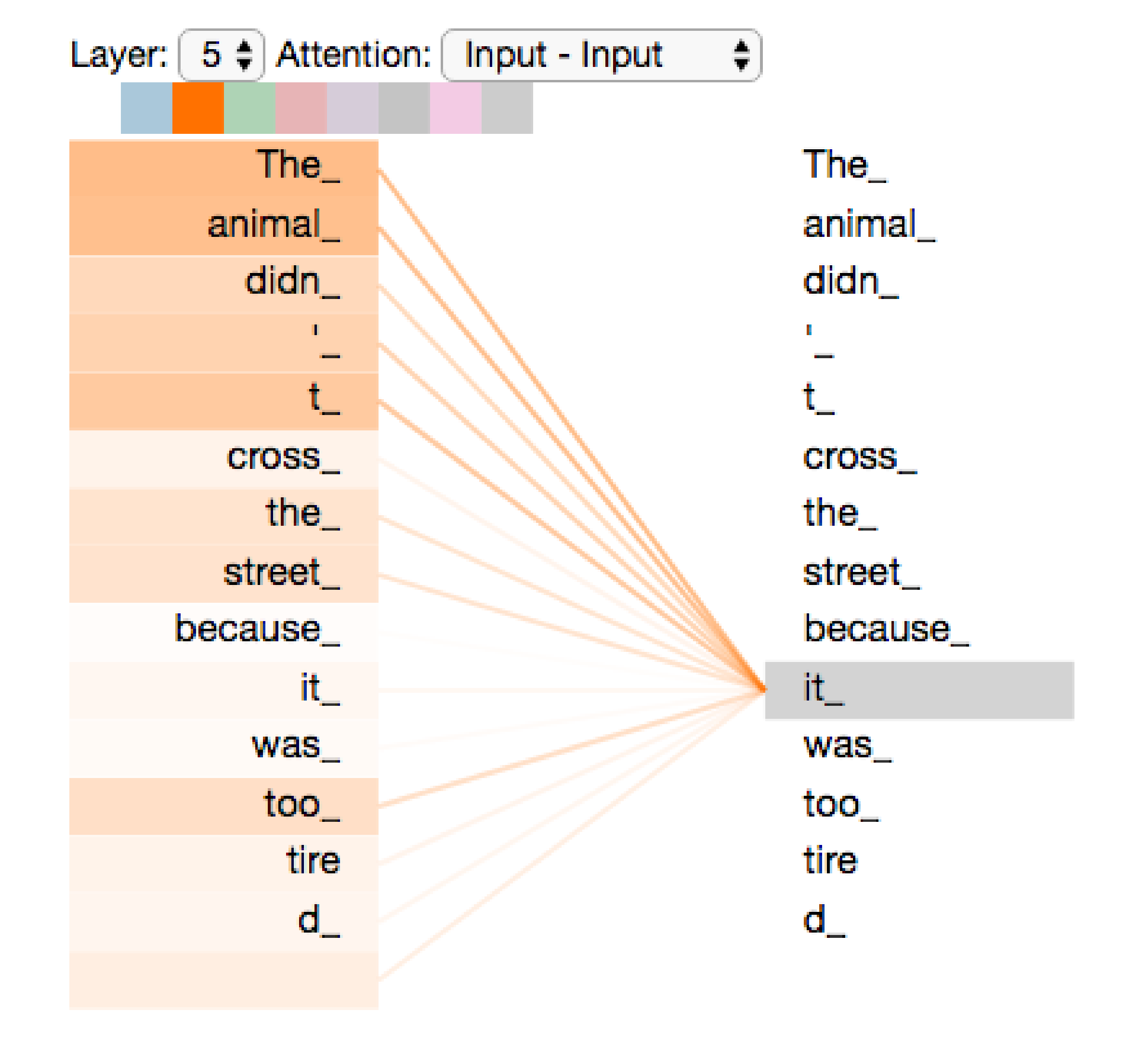


Reference

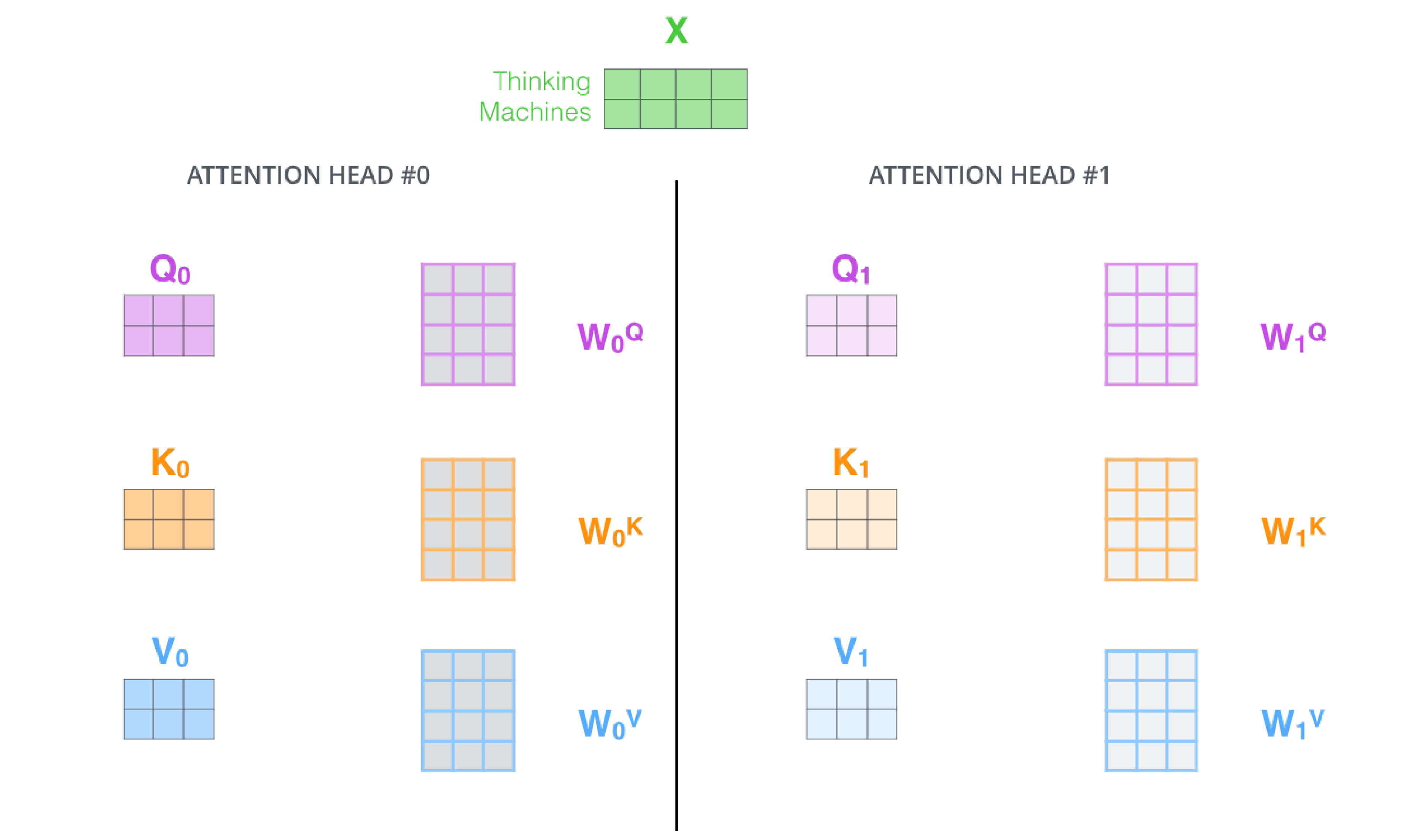
Transformer



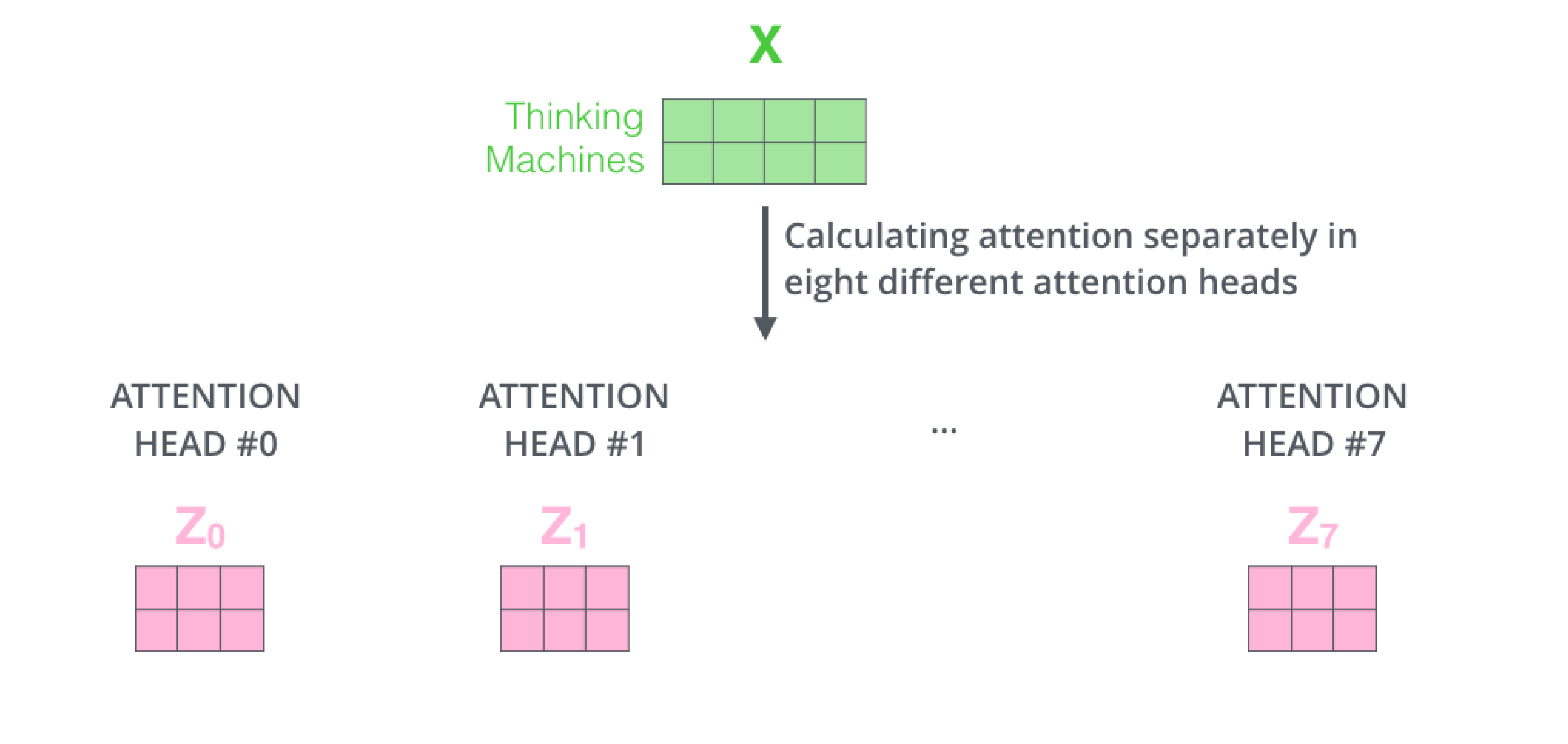
Transformer



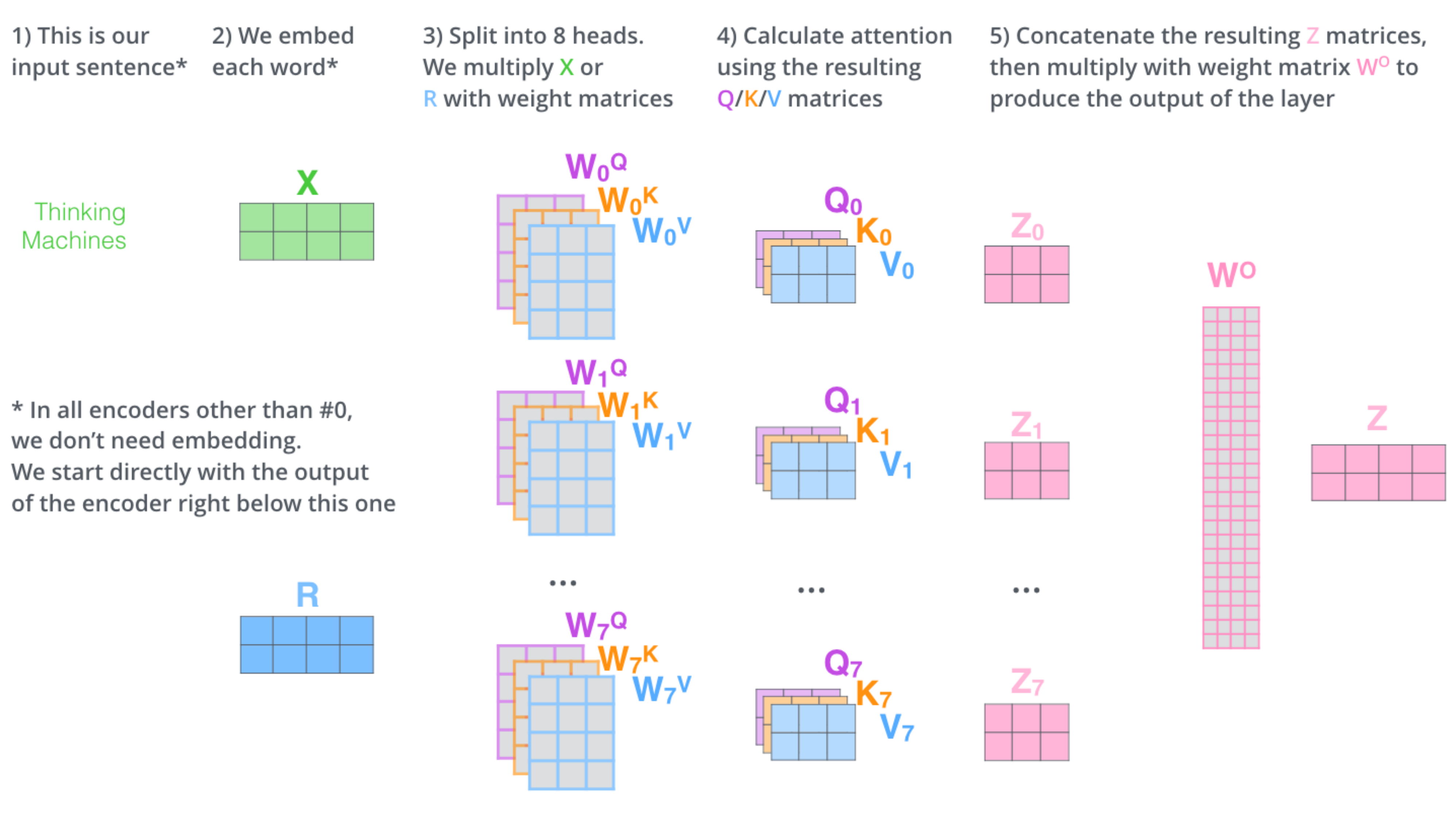
Transformer



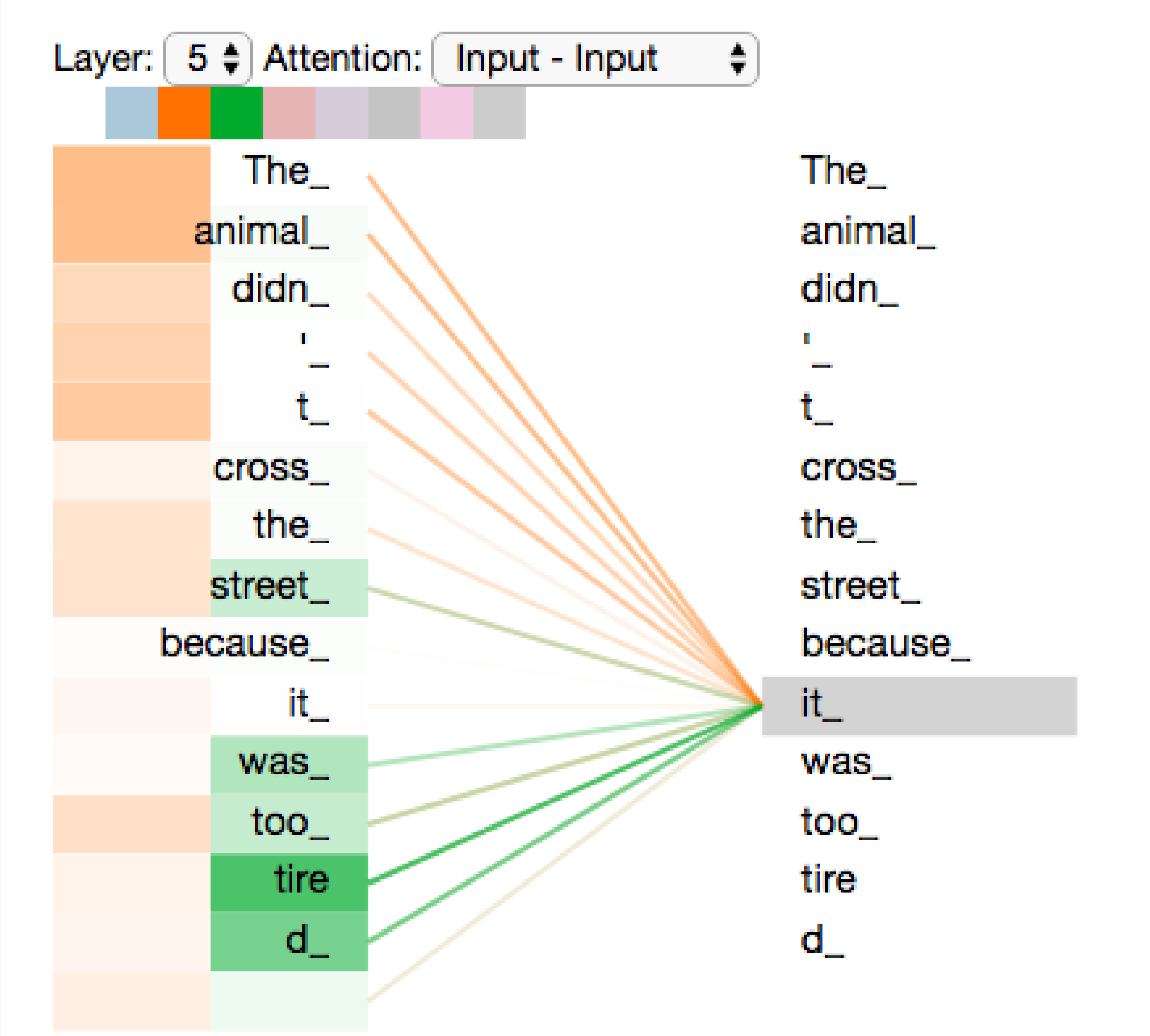
Transformer



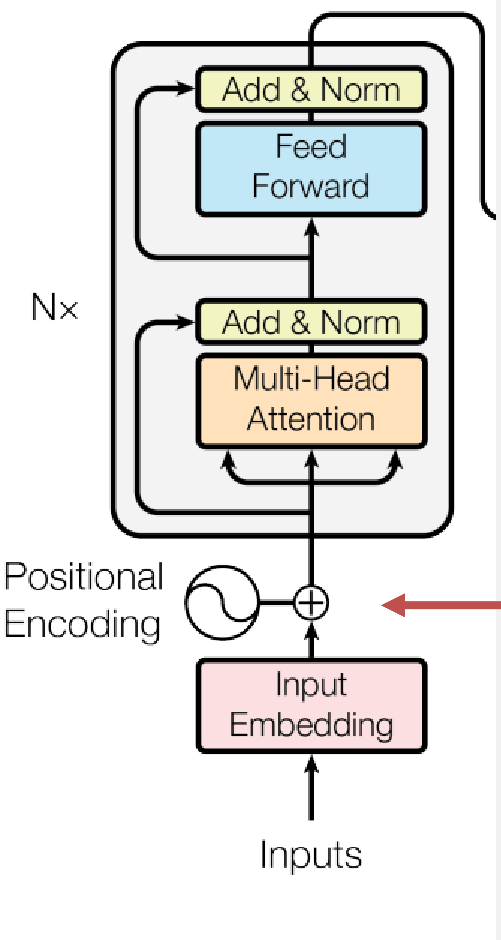
Transformer

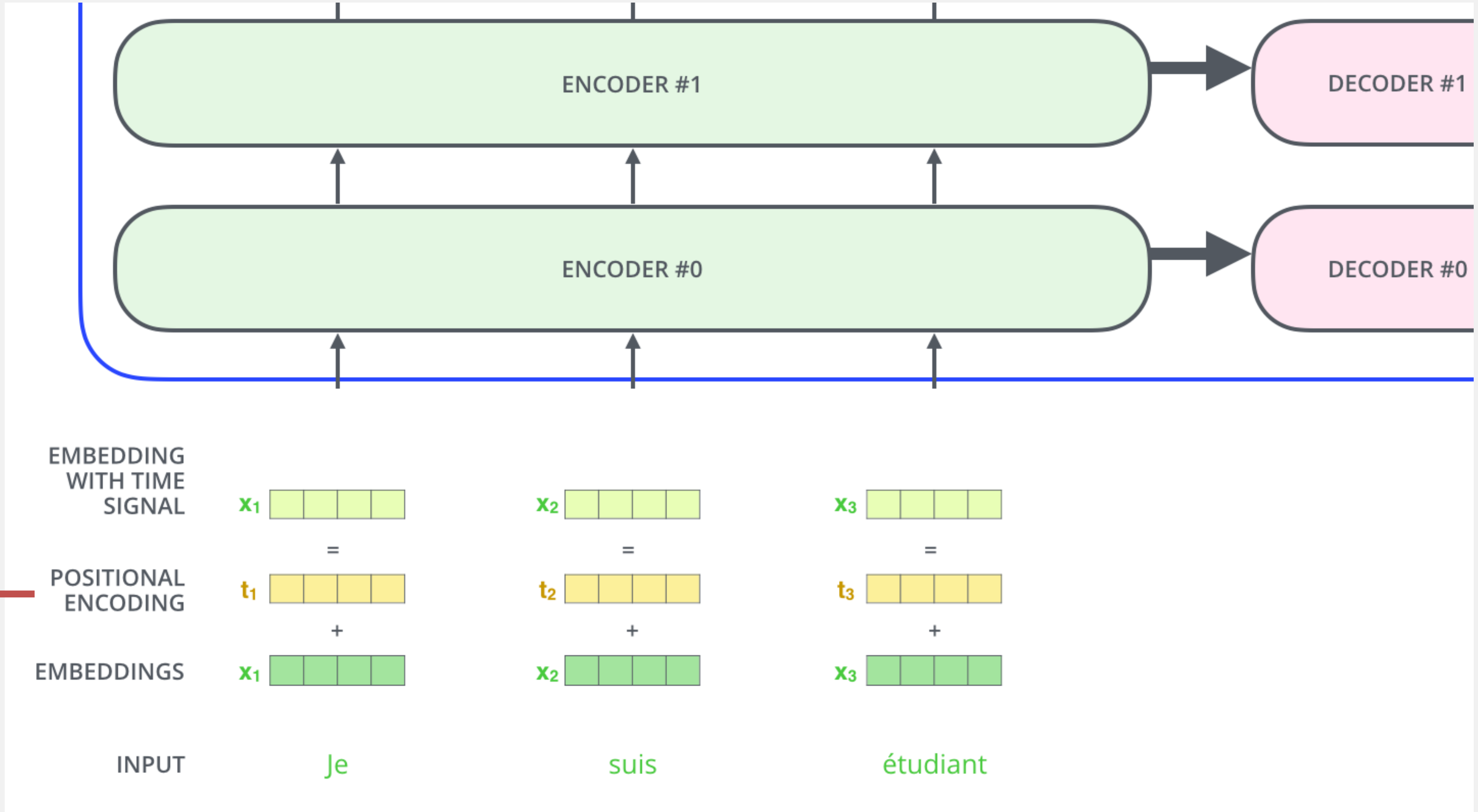


Transformer

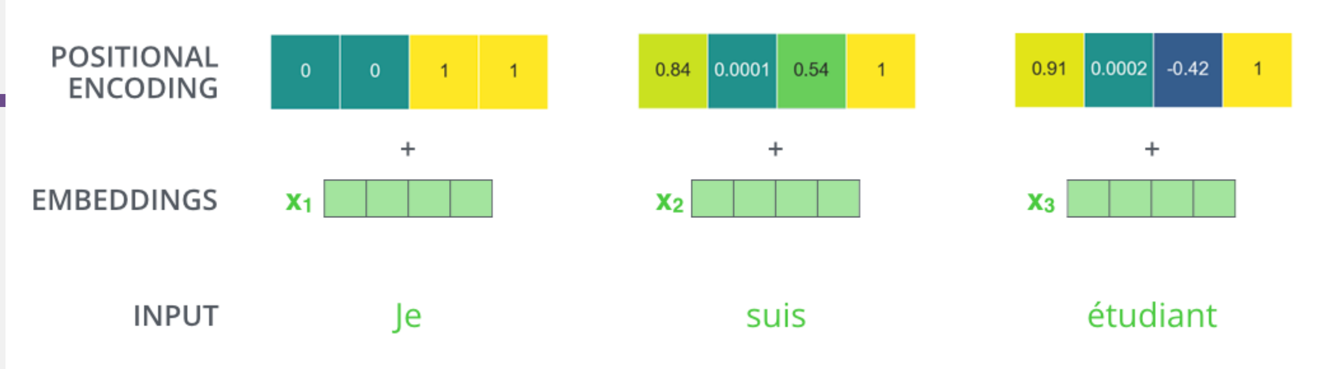


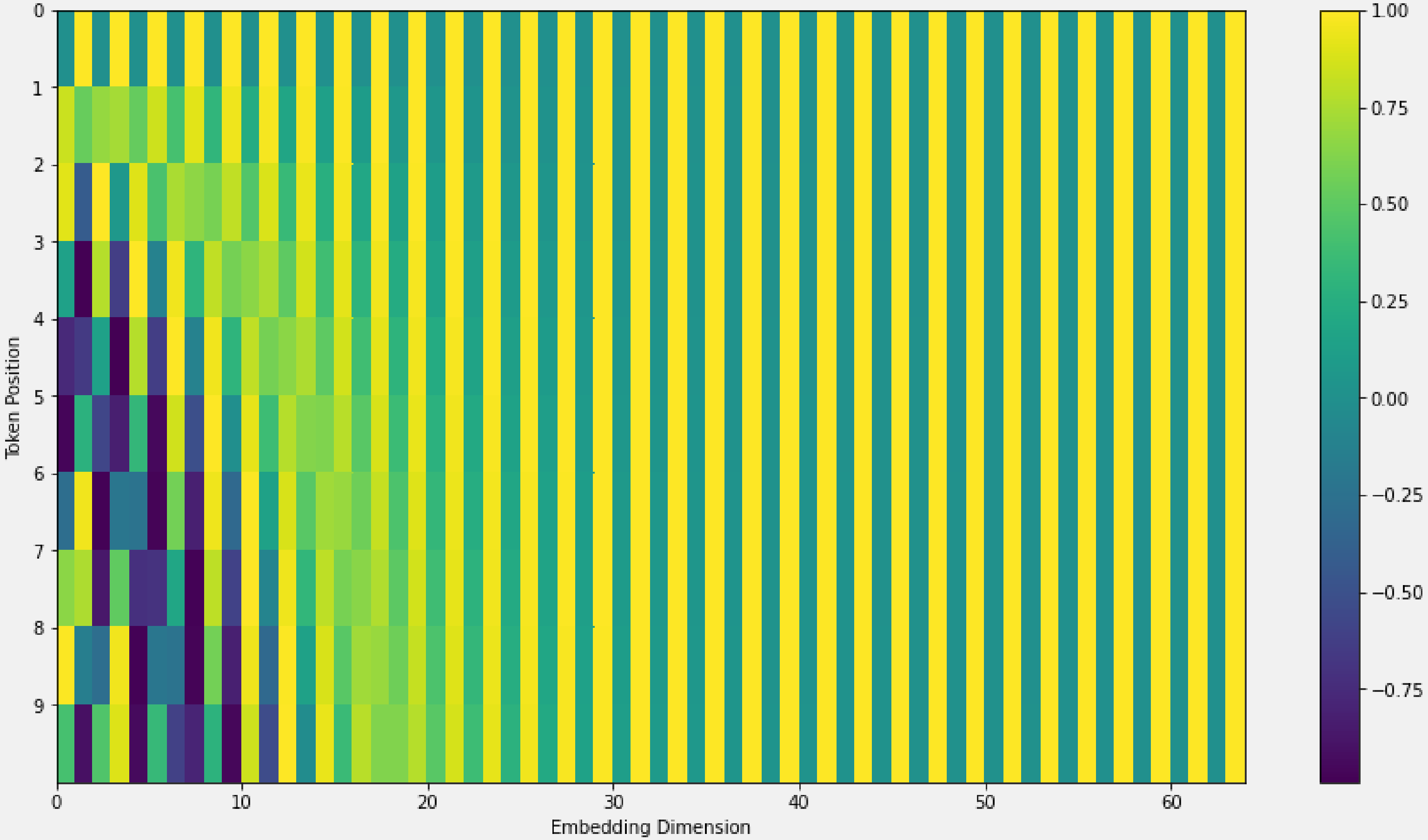
Transformer





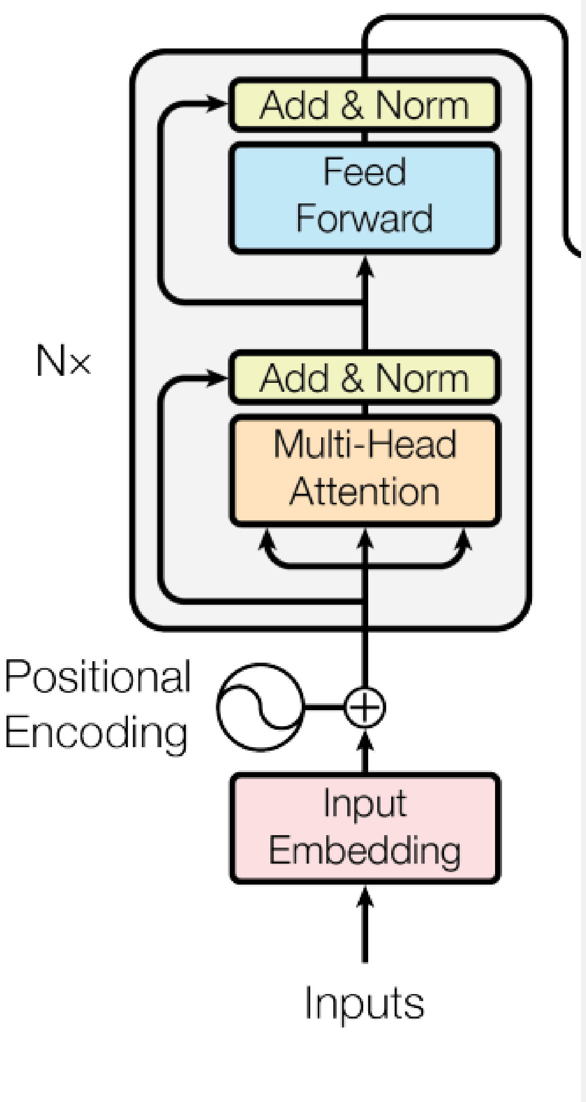
Transformer

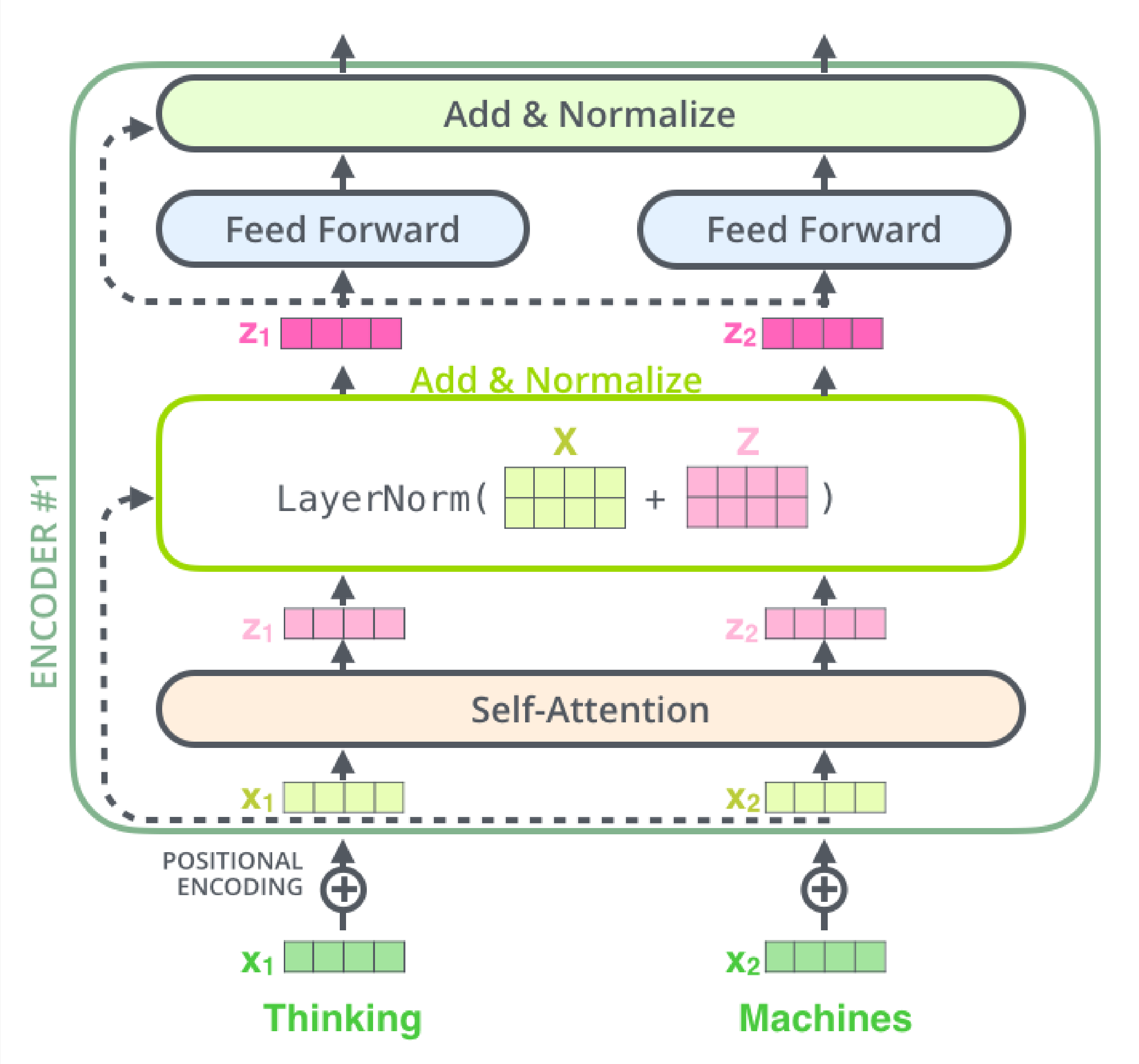




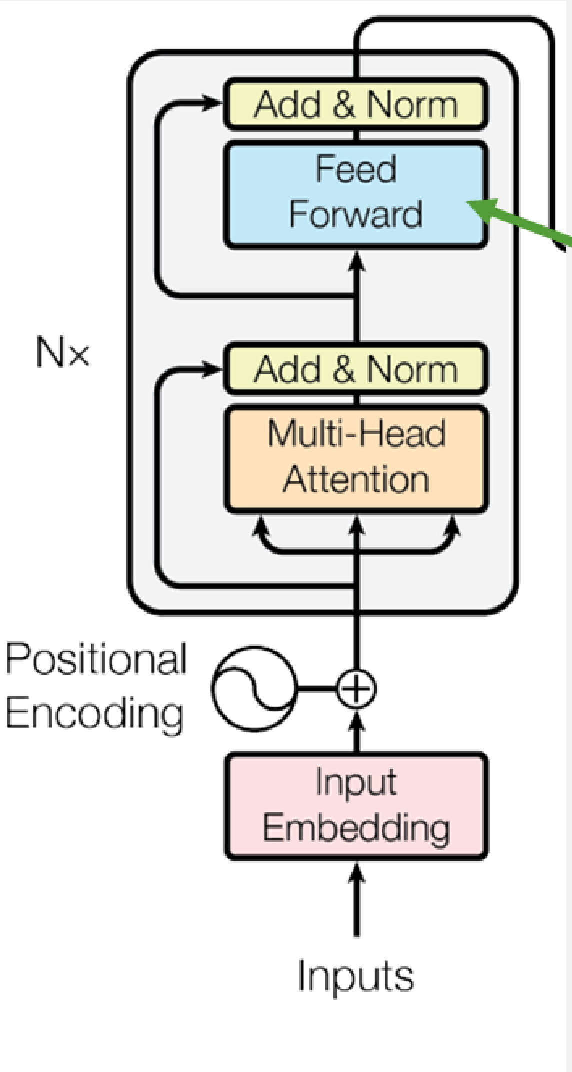
[Reference]

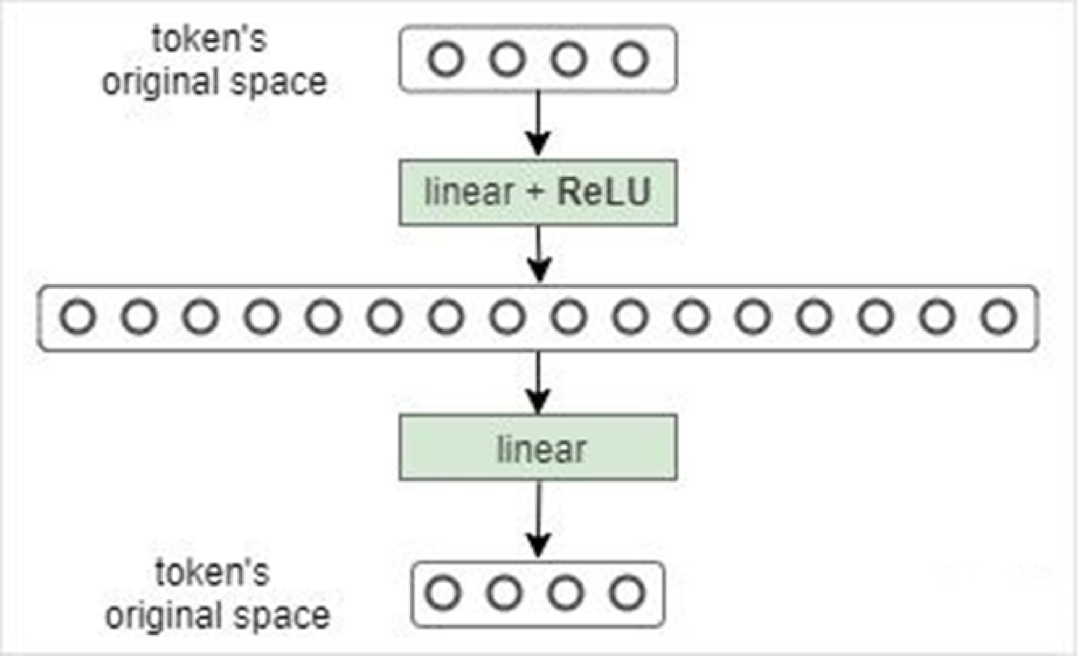
Transformer



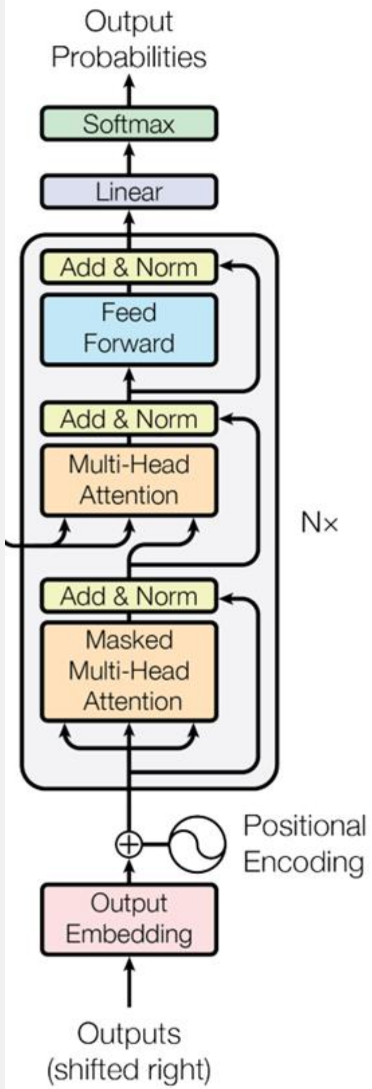


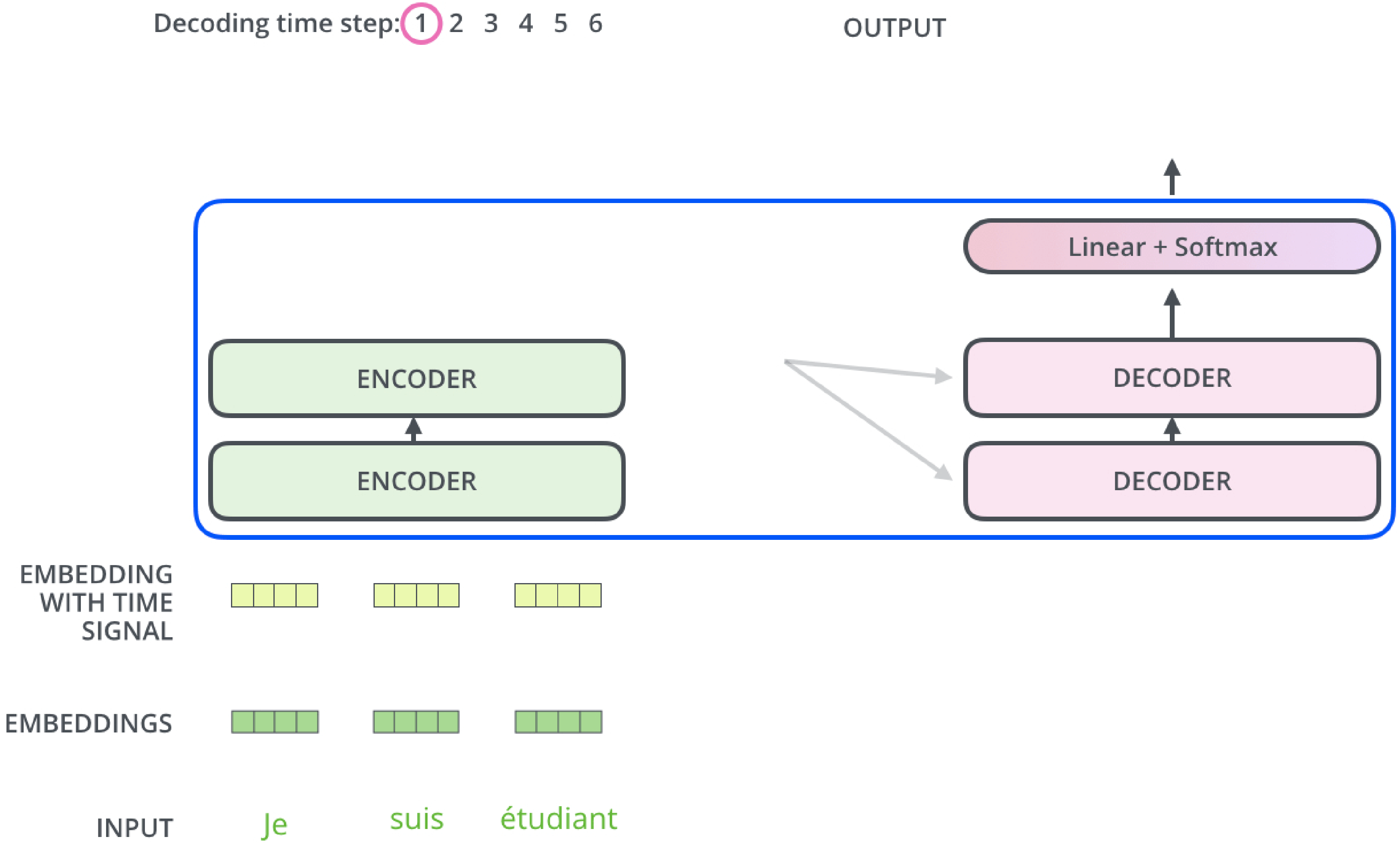
Transformer



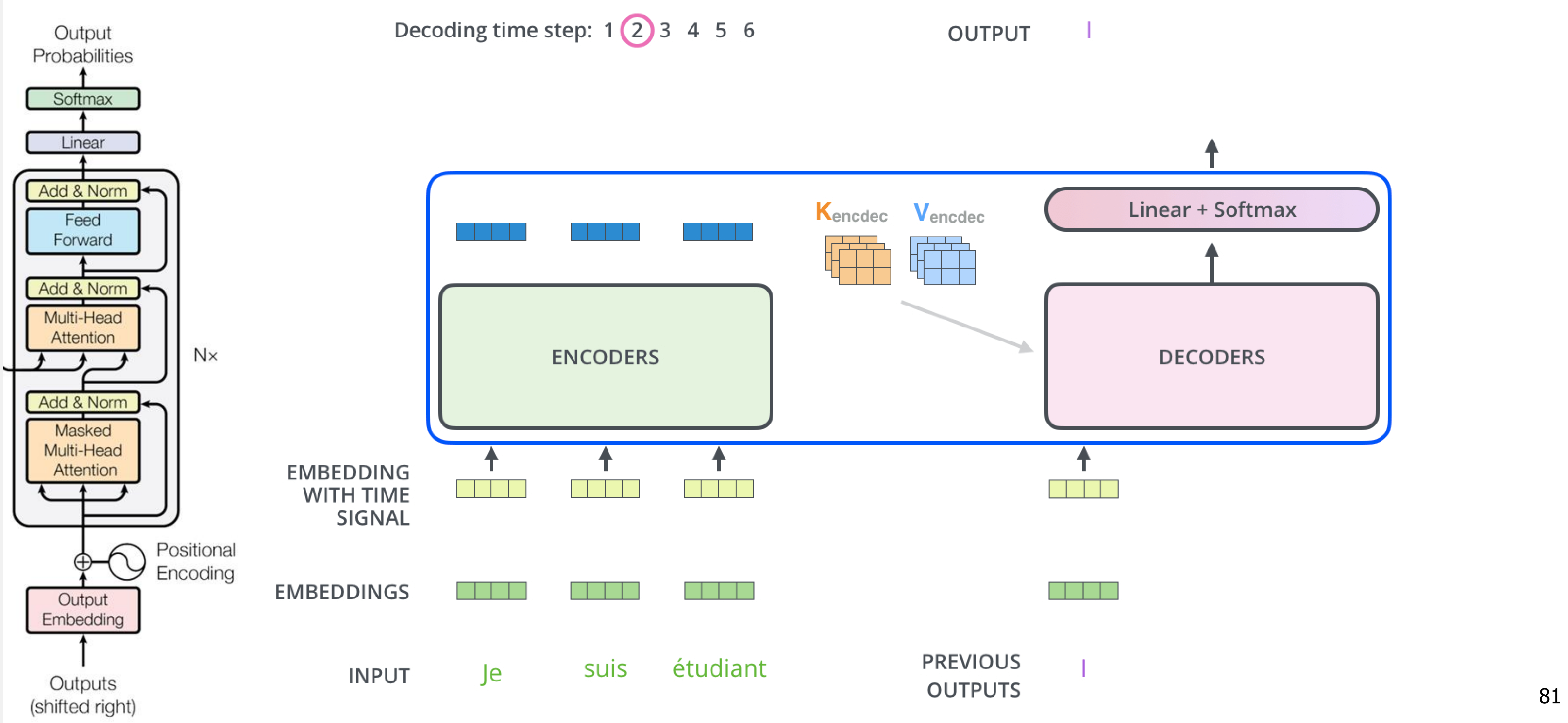


Transformer





Transformer

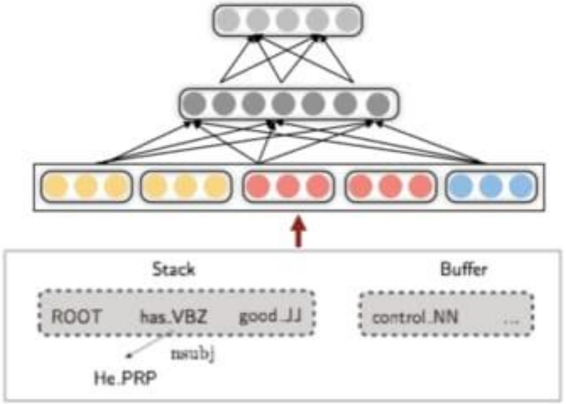


03

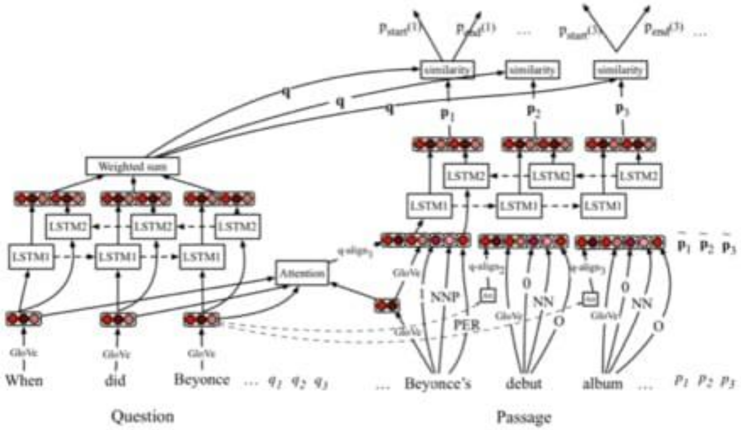
Two Models

BERT

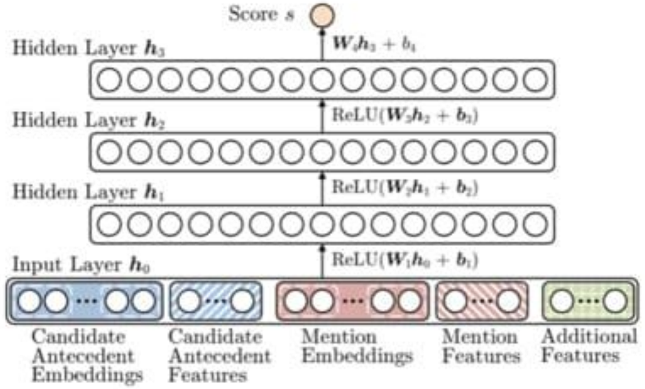
Various Model Architectures for Different NLP Tasks



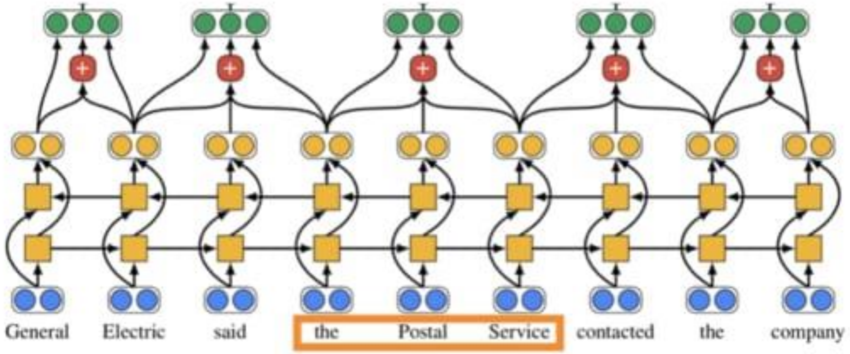
Dependency Parsing



Question Answering



Coreference example 1



Coreference example 2

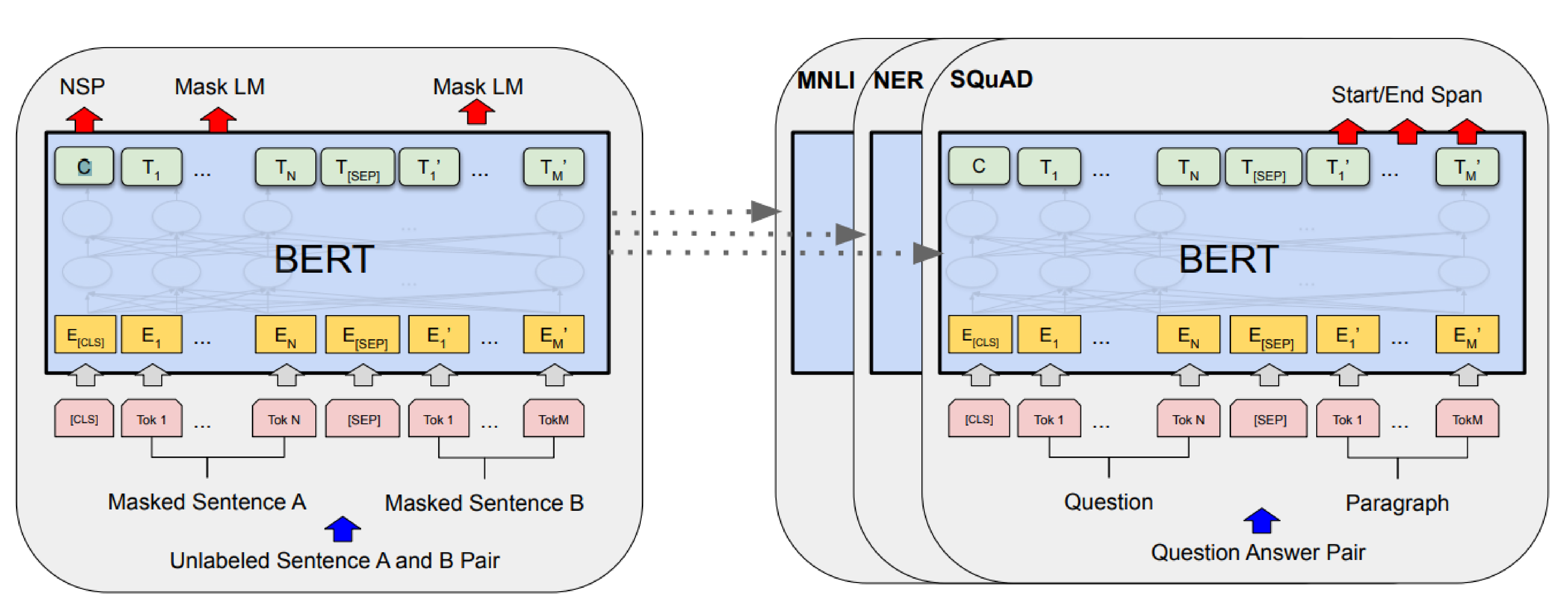
BERT

Bidirectional Encoder Representation from Transformers

- Architecture
  - Built by multiple Transformers' encoder
- Features
  - Bidirectionality
  - Pre-training
  - Semantic representation
- Objective
  - Provide a universal language representation model to be fine-tuned for various natural language processing tasks

BERT

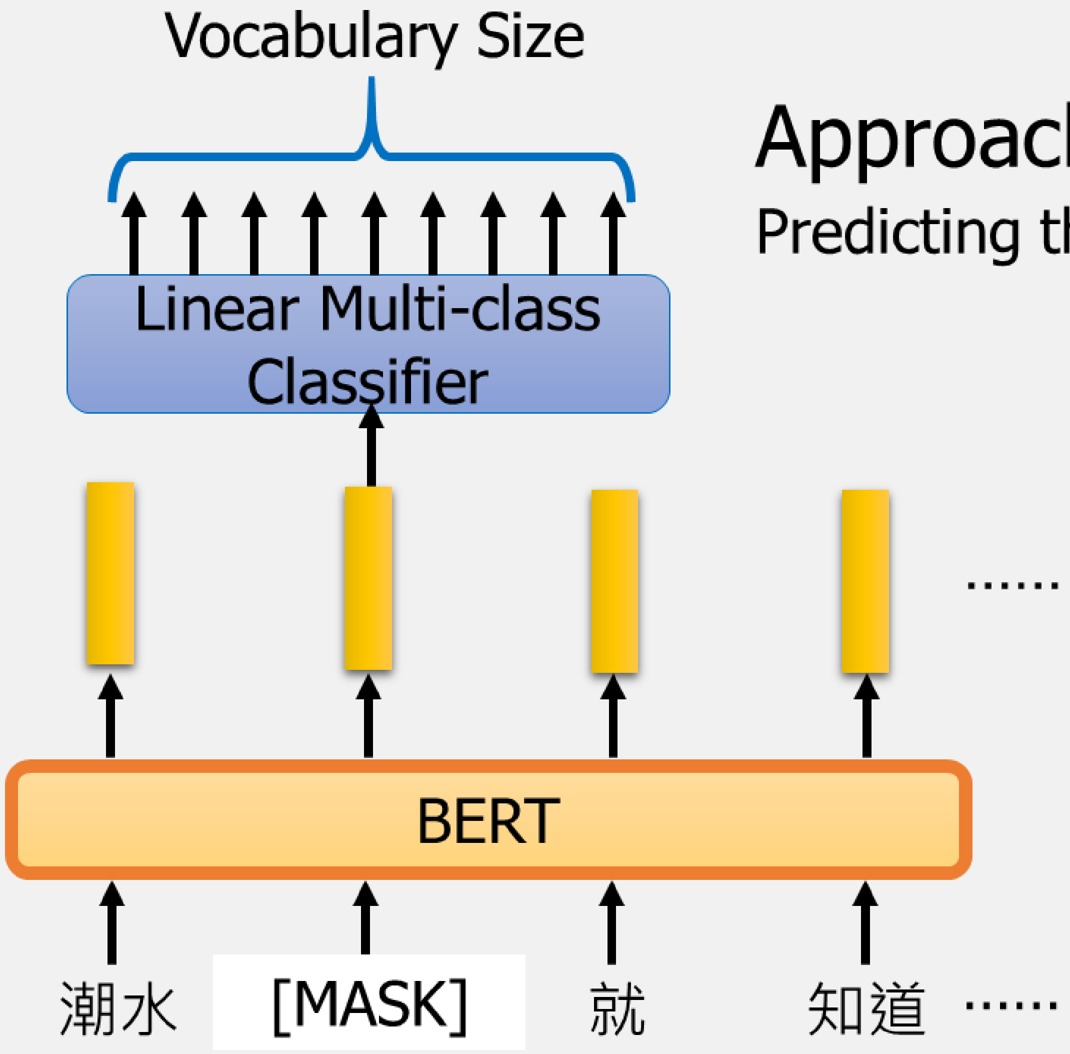
A model for all NLP task



Pre-training

Fine-tuning

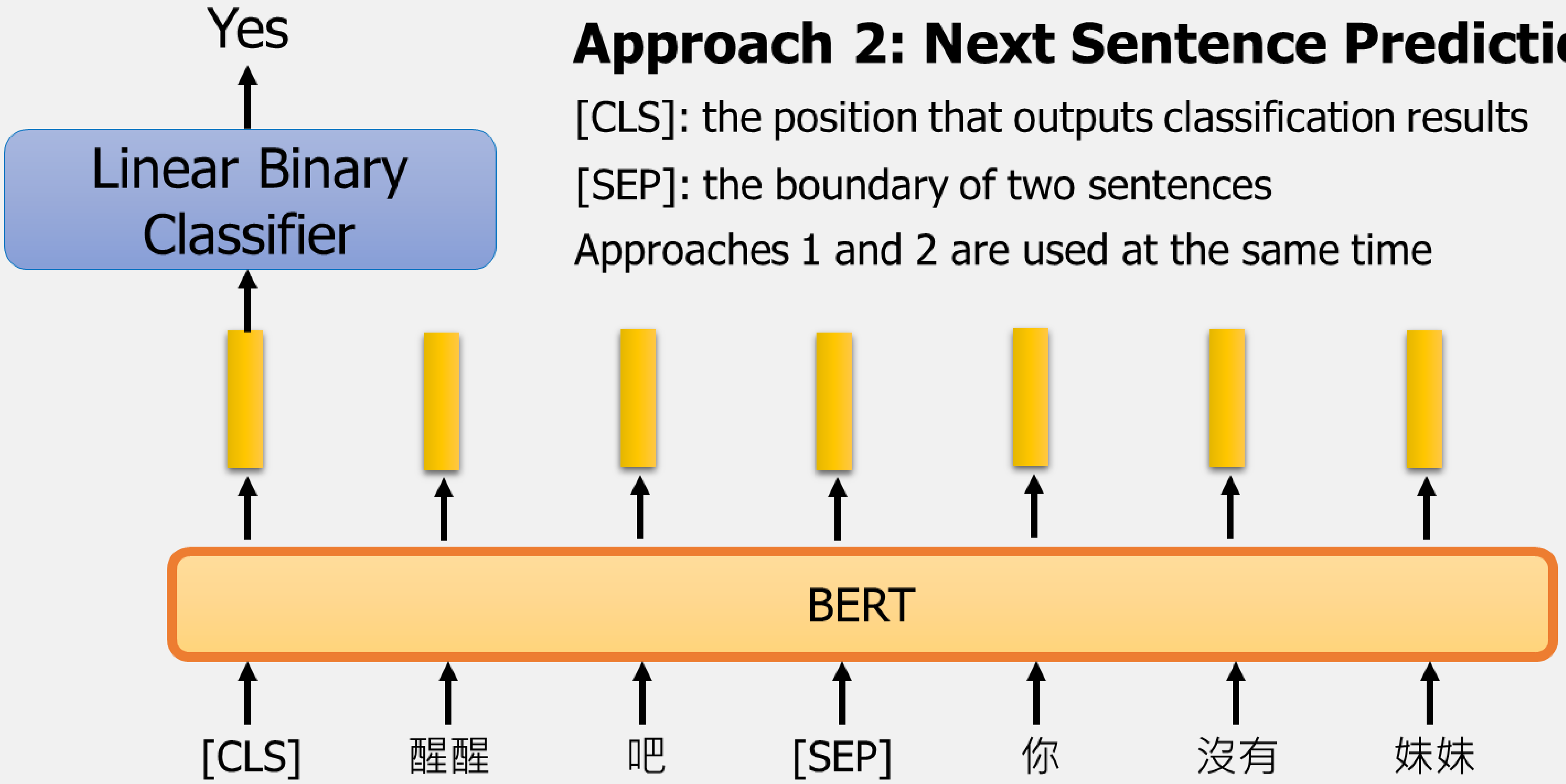
BERT



Approach 1: Masked LM

Predicting the masked word

BERT



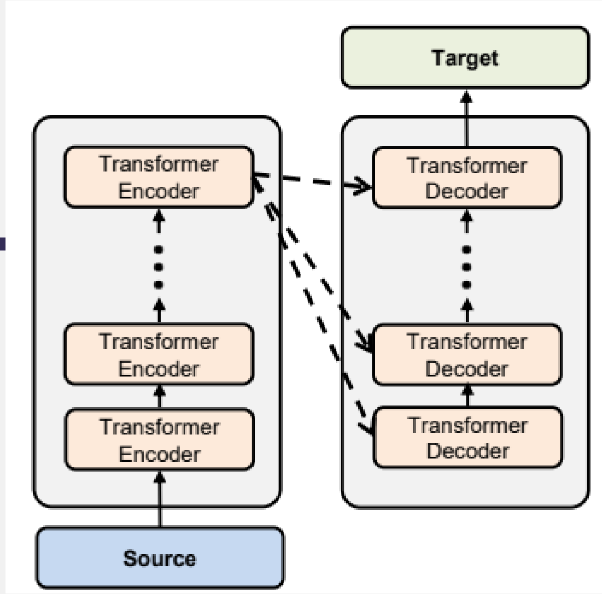
Approach 2: Next Sentence Prediction

[CLS]: the position that outputs classification results

[SEP]: the boundary of two sentences

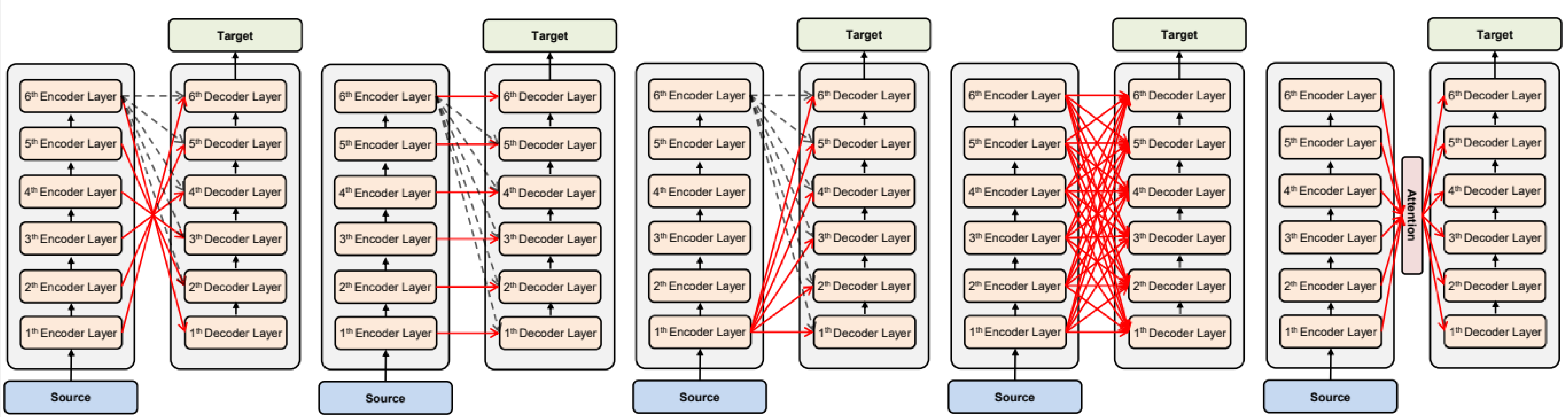
Approaches 1 and 2 are used at the same time

BERT



Conventional Transformer

[Reference](#)



(a) Granularity Consistent Attention (b) Granularity Parallel Attention (c) Fine-Grained Attention (d) Full Matching Attention (e) Adaptive Matching Attention

Various Transformer

GPT

Generative Pre-trained Transformer

- Architecture
  - Also Built by multiple Transformers' encoder
- Features
  - Generative model
  - Large-scale pre-training
  - Transfer learning
- Objective
  - Provide a universal language model to understand and generate for various text generation and dialogue capabilities

GPT vs BERT

Comparisons

- Pre-training objectives
  - GPT - using "predicting the next word" to learn the language structure and contextual relationships within sentences
  - BERT - using "masked language model" and "next sentence prediction" to predict masked words and to judge whether two sentences are consecutive
- Input
  - GPT - use a unidirectional Transformer architecture and take the text sequence as input
  - BERT - employ a bidirectional Transformer architecture and divide the sentence into left and right parts as input
- Fine-tuning
  - Both models use a similar method of adjusting the last layer of the pre-trained model to adapt to specific tasks