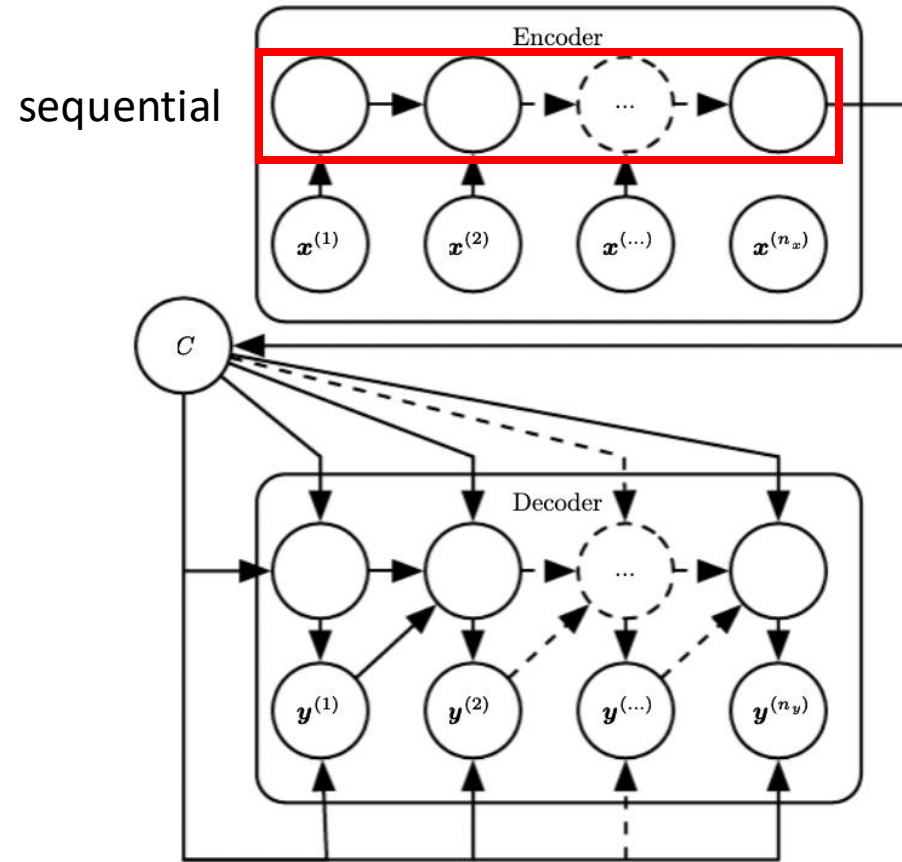


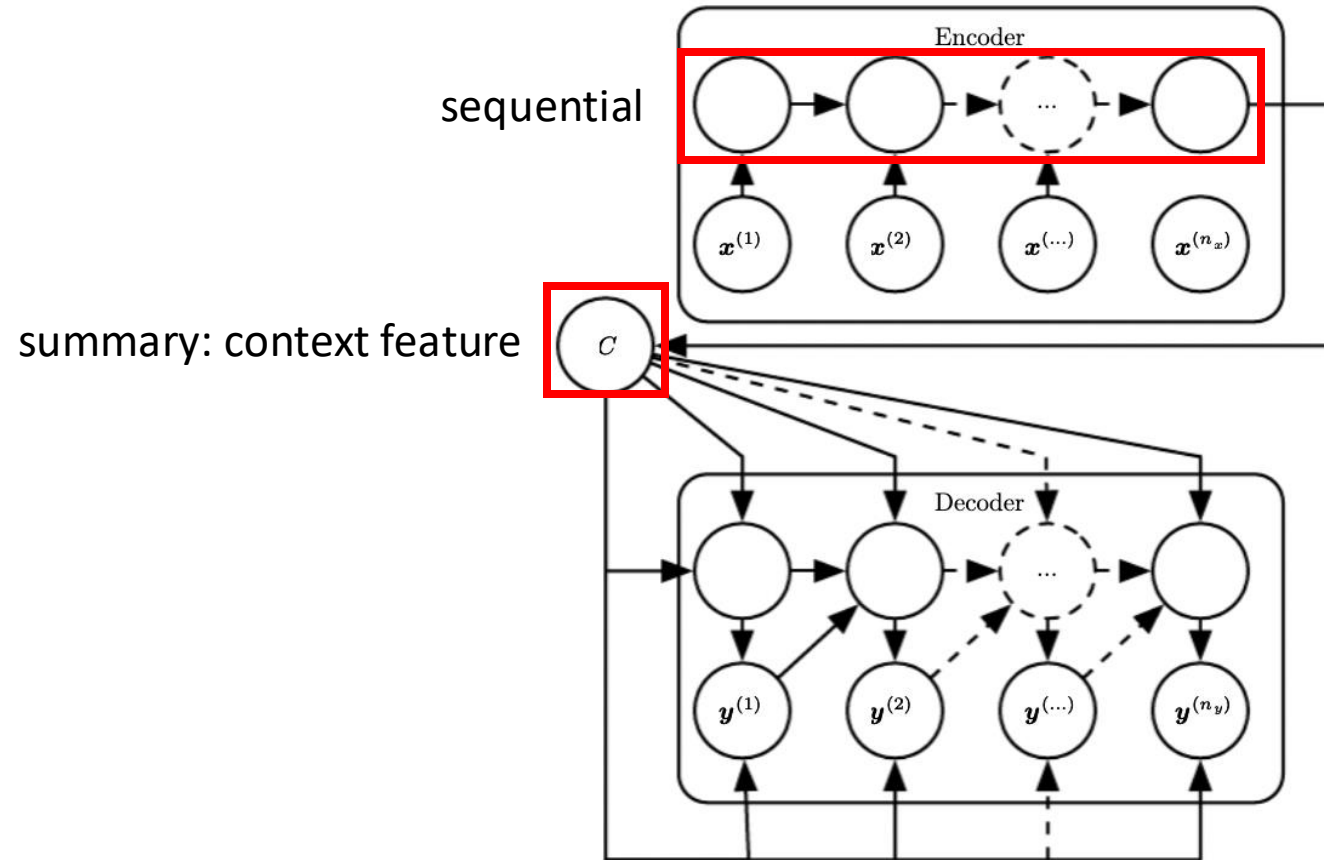
Attention and Transformers

Neural Networks Design And Application

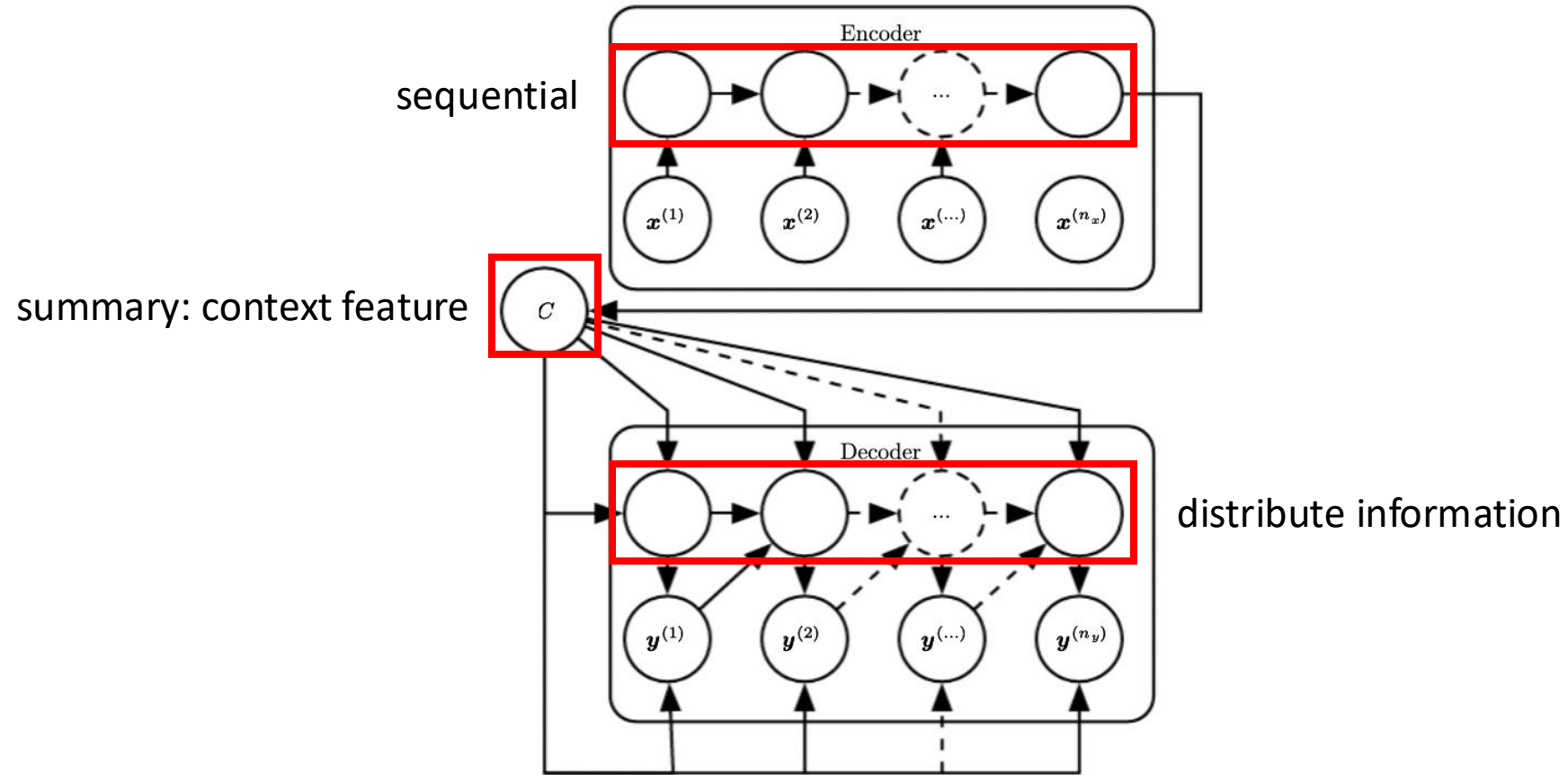
Encoder-decoder



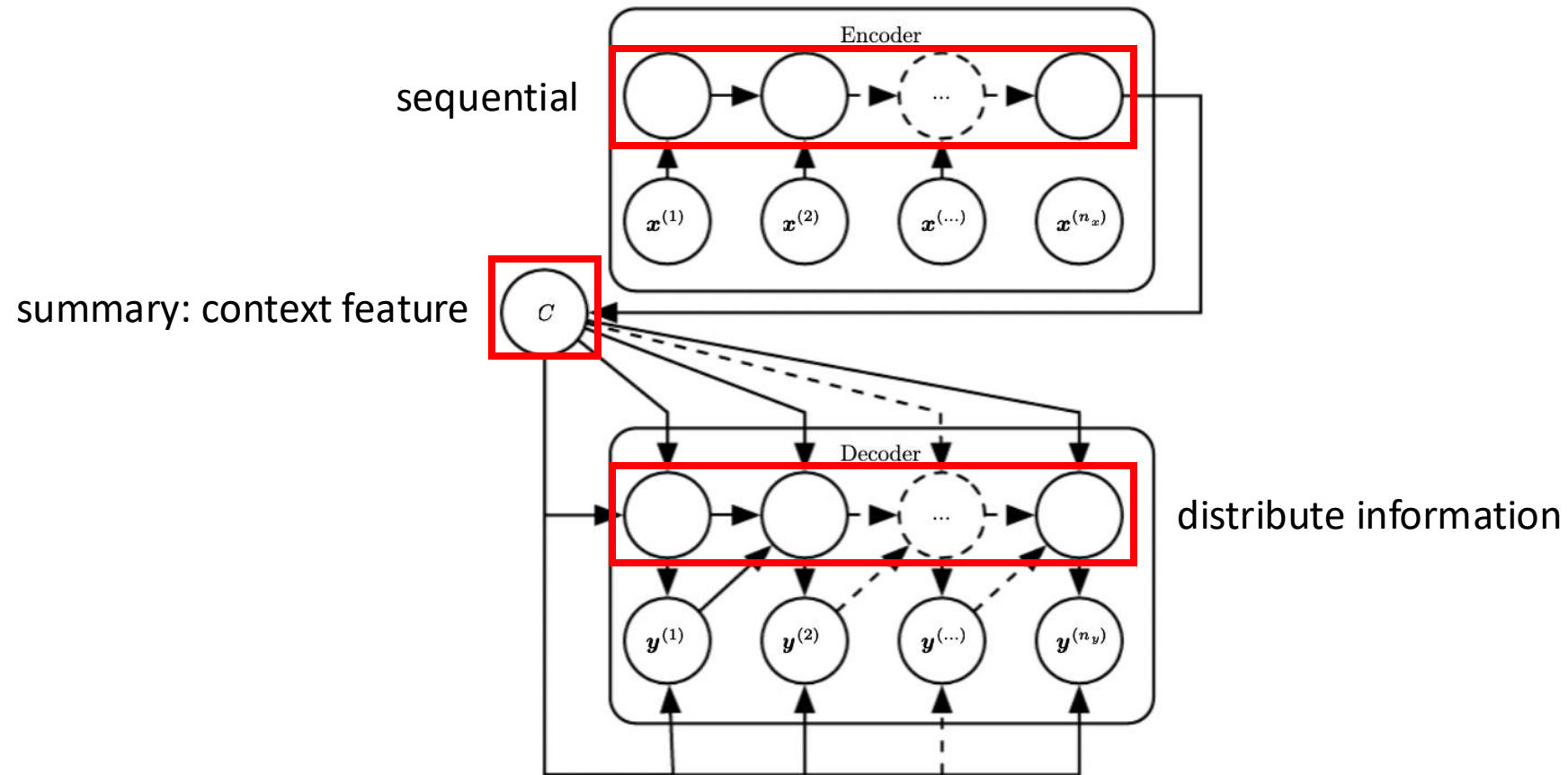
Encoder-decoder



Encoder-decoder

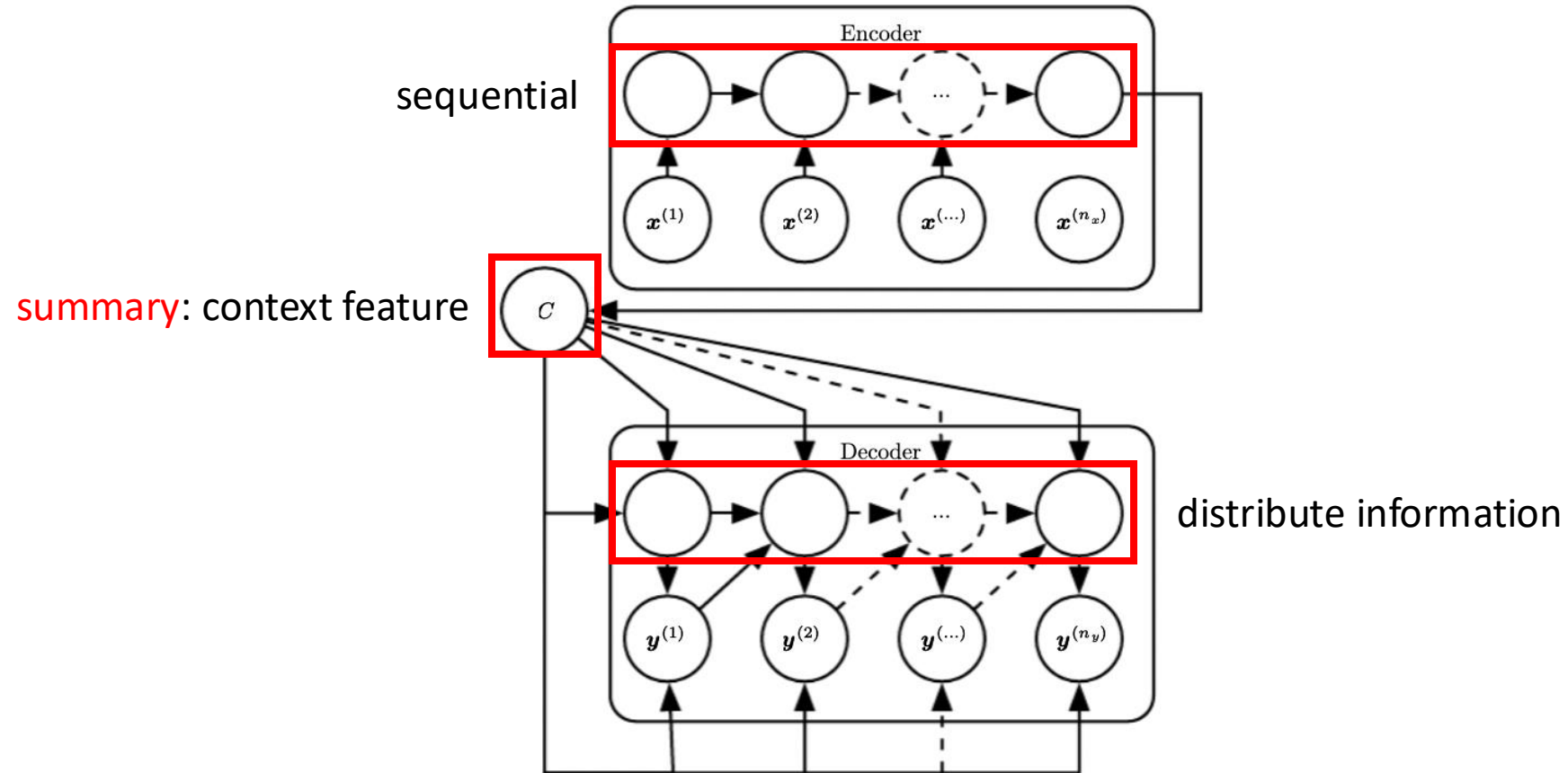


Encoder-decoder



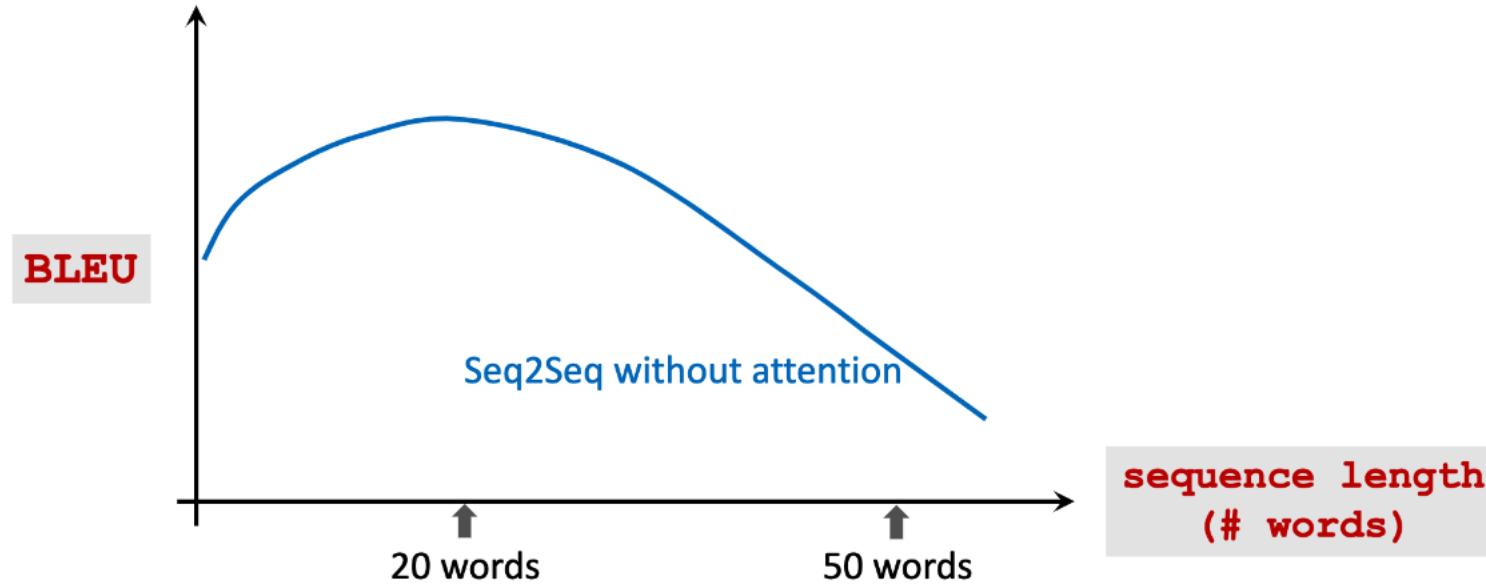
Q: can we use bi-directional RNN for input?

Encoder-decoder



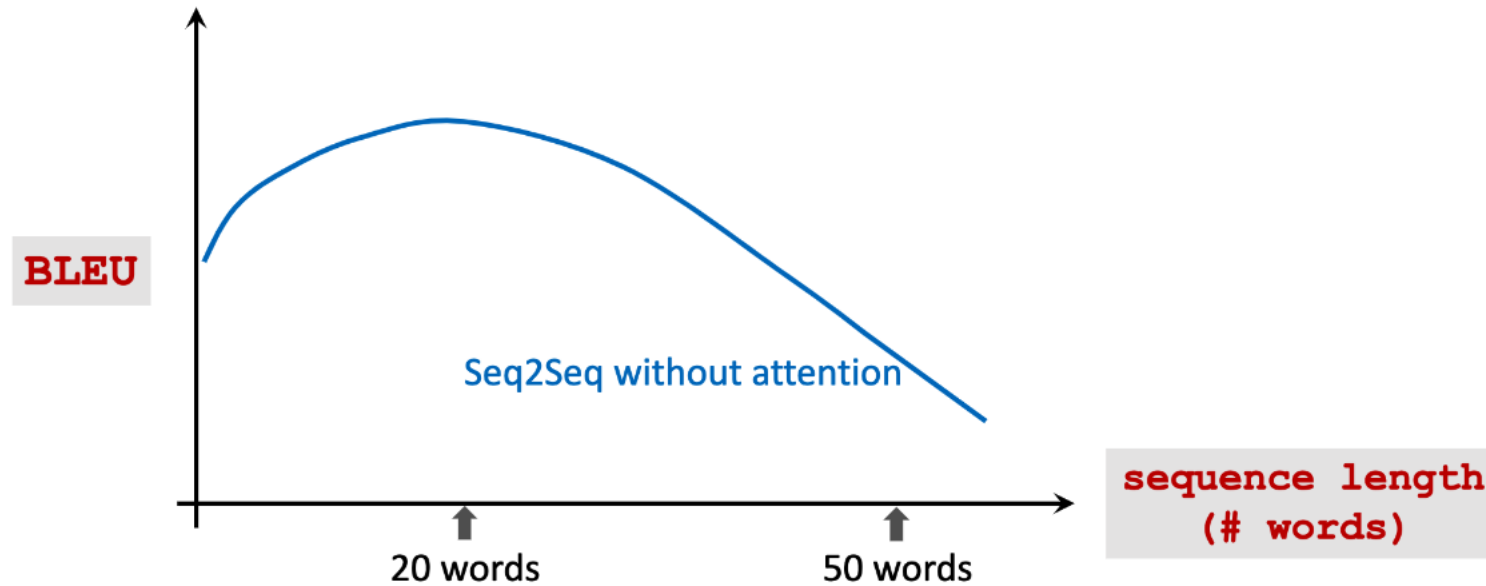
Q: can we use bi-directional RNN for input?

Seq2seq model performance



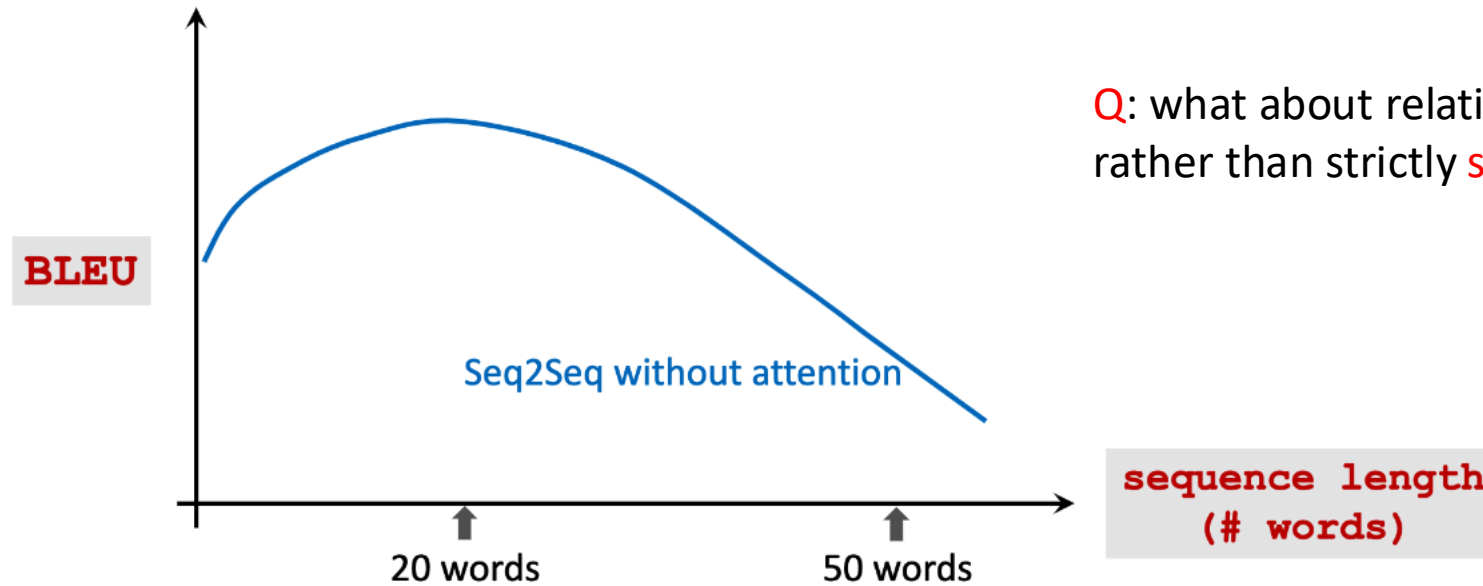
the clouds are in the sky

Seq2seq model performance



I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in **France**. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent **French**.

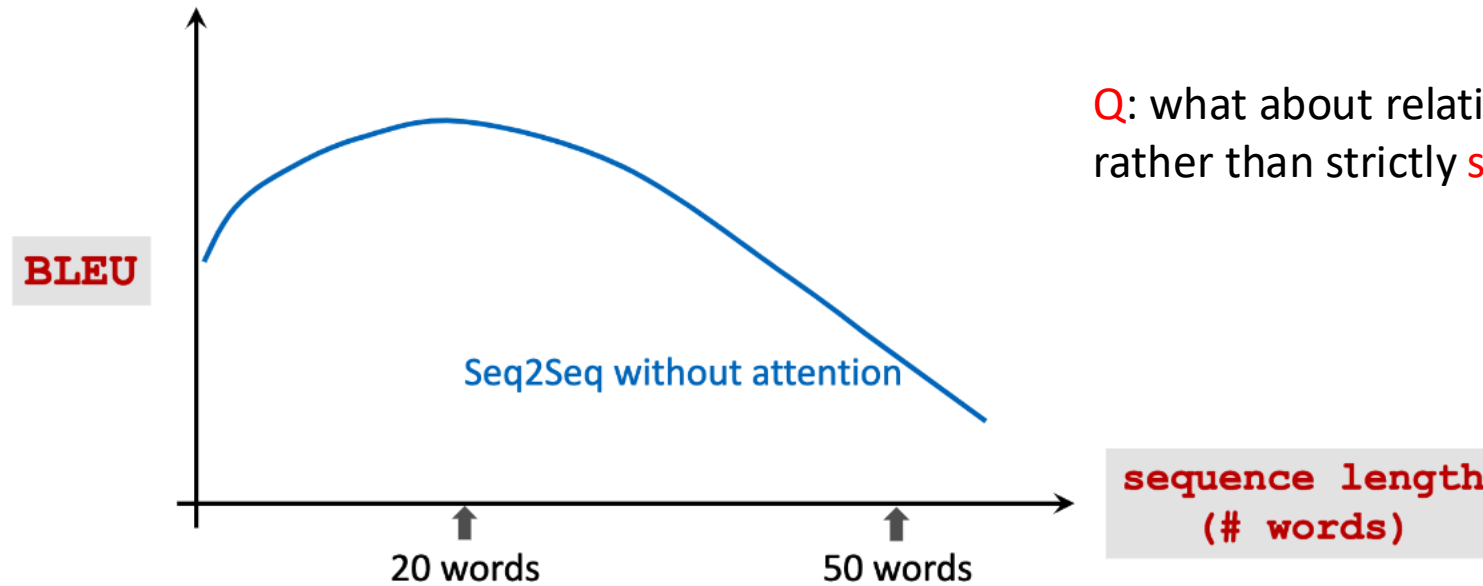
Seq2seq model performance



Q: what about relatively **parallel** correlation rather than strictly **sequential** correlation?

I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in **France**. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent **French**.

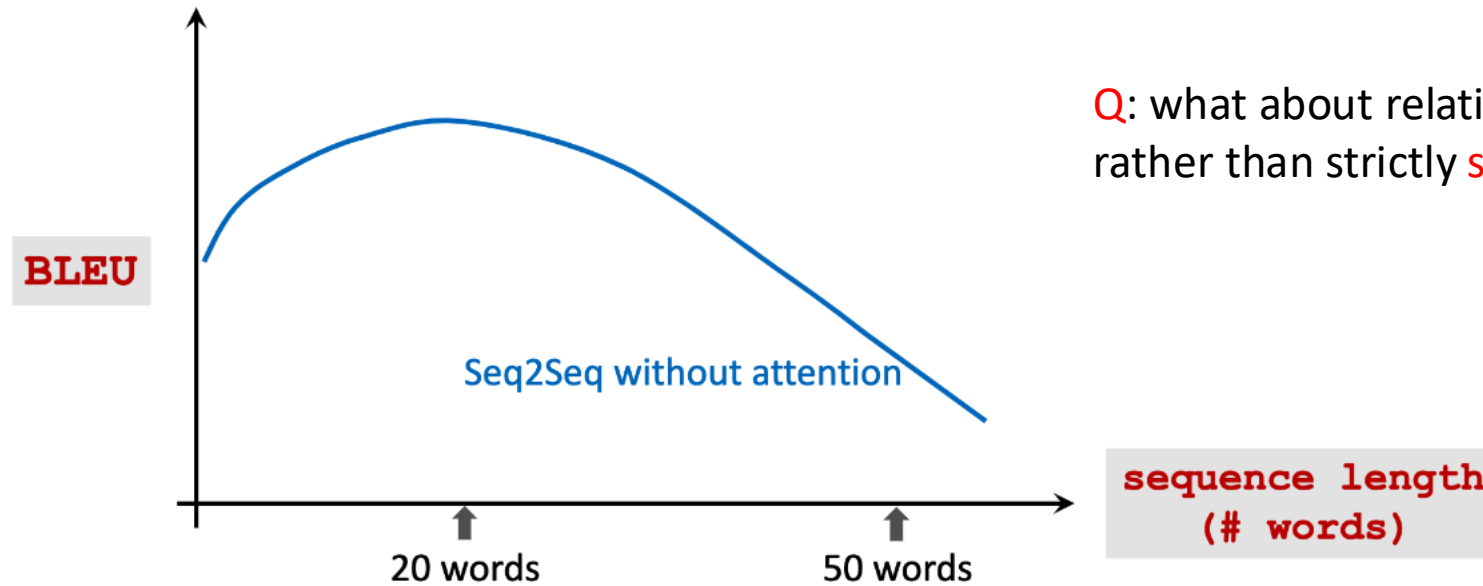
Seq2seq model performance



Q: what about relatively **parallel** correlation rather than strictly **sequential** correlation?

I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in **France**. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent **French**.

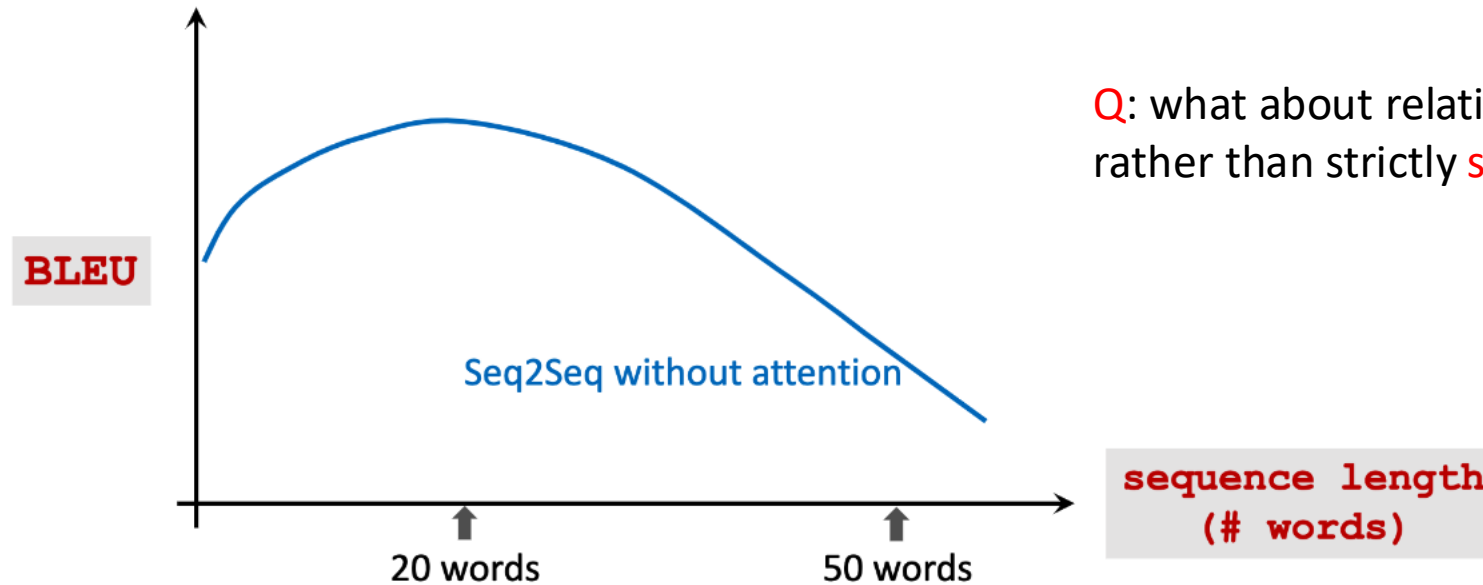
Seq2seq model performance



Q: what about relatively **parallel** correlation rather than strictly **sequential** correlation?

I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in **France**. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent **French**.

Seq2seq model performance

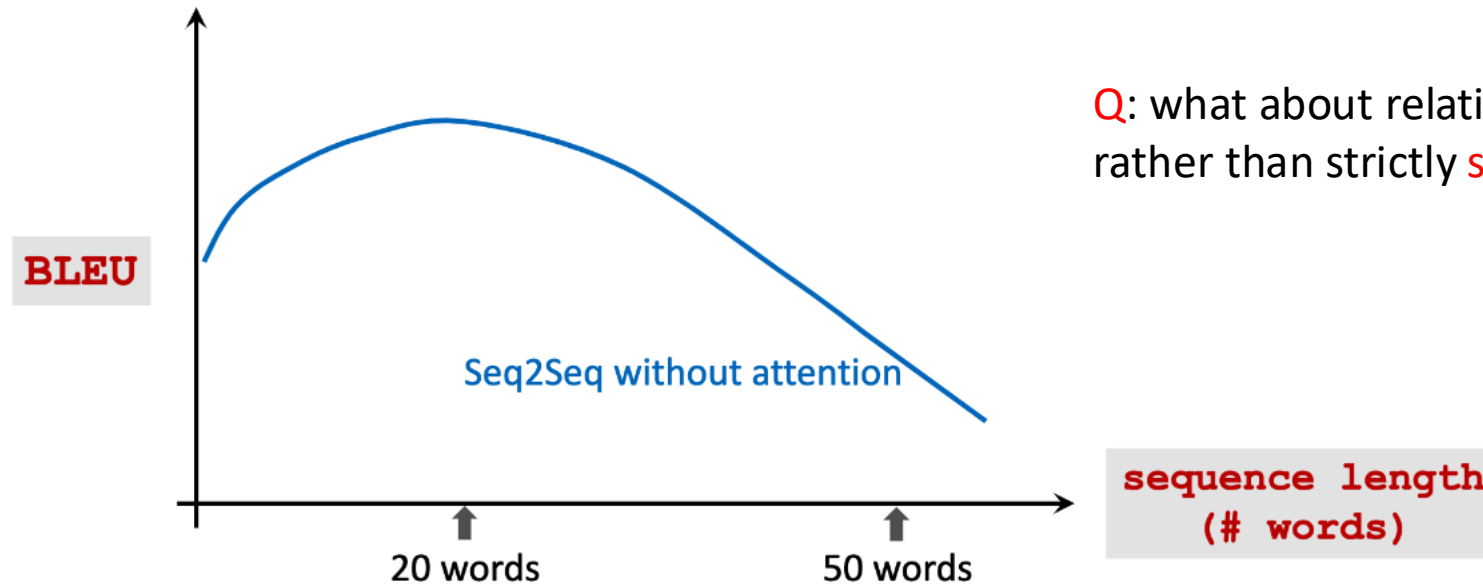


Q: what about relatively **parallel** correlation rather than strictly **sequential** correlation?

We need to summarize all context

I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in France. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent *French*.

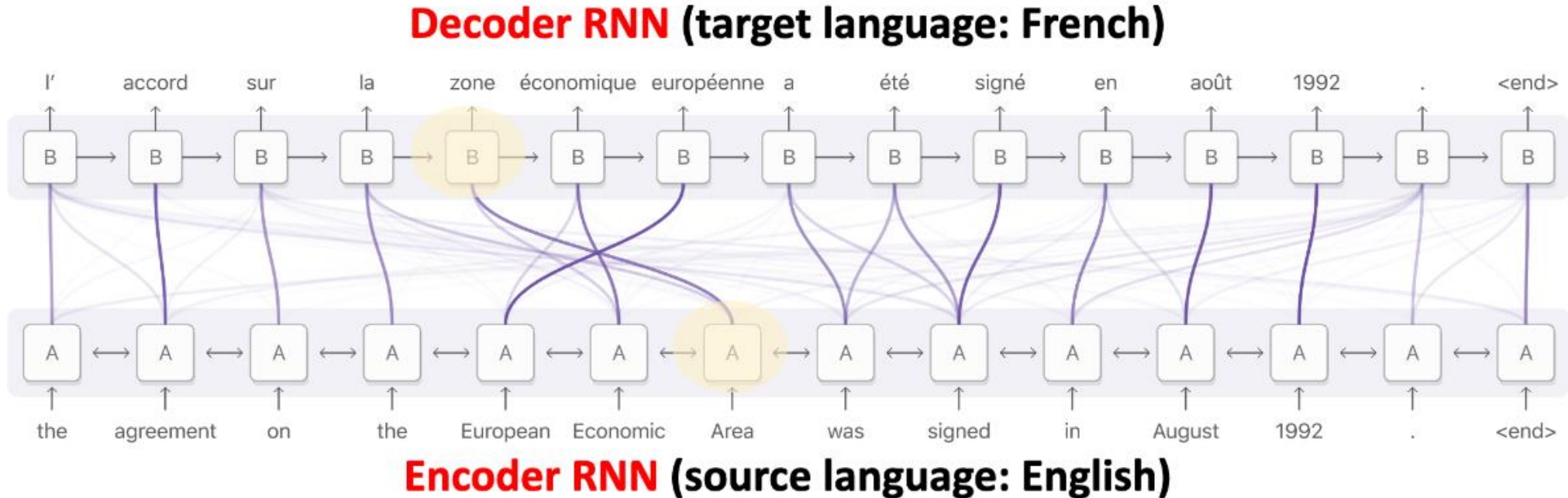
Seq2seq model performance



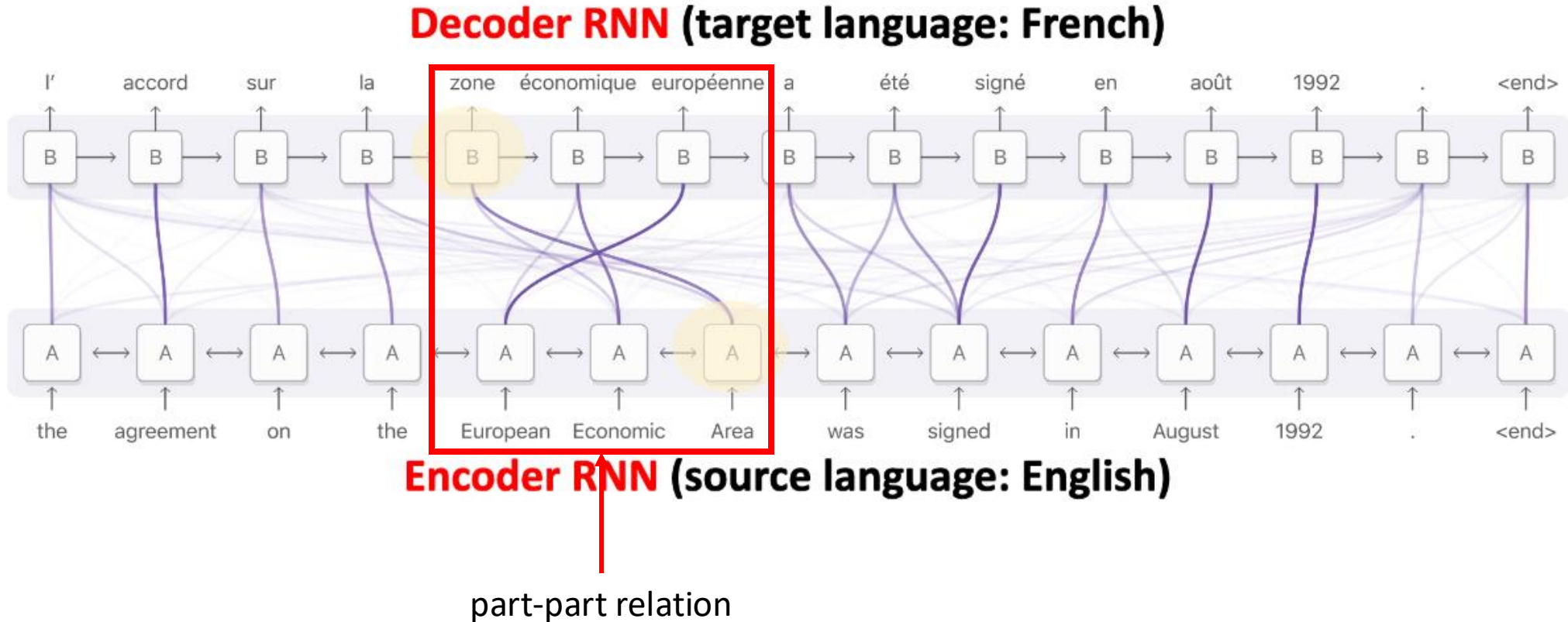
Q: what about relatively **parallel** correlation rather than strictly **sequential** correlation?

I like this town very much. I started my undergraduate study in 2020 and my major is computer science. I like programming and reading. I usually get up at 7AM and do some exercise. I also go fishing at weekend. I grew up in **France**. I spent my childhood outdoors. Whether it was riding my bicycle around my neighborhood pretending it was a motorcycle, making mud cakes, going on treasure hunts, making and selling perfume out of strong smelling flowers, or simply laying on the grass underneath the sun with a soccer ball waiting for someone to come out and play with me, the outdoors was where I spent my childhood and I cannot be more appreciative of it. I speak fluent **French**.

Input-output correlation

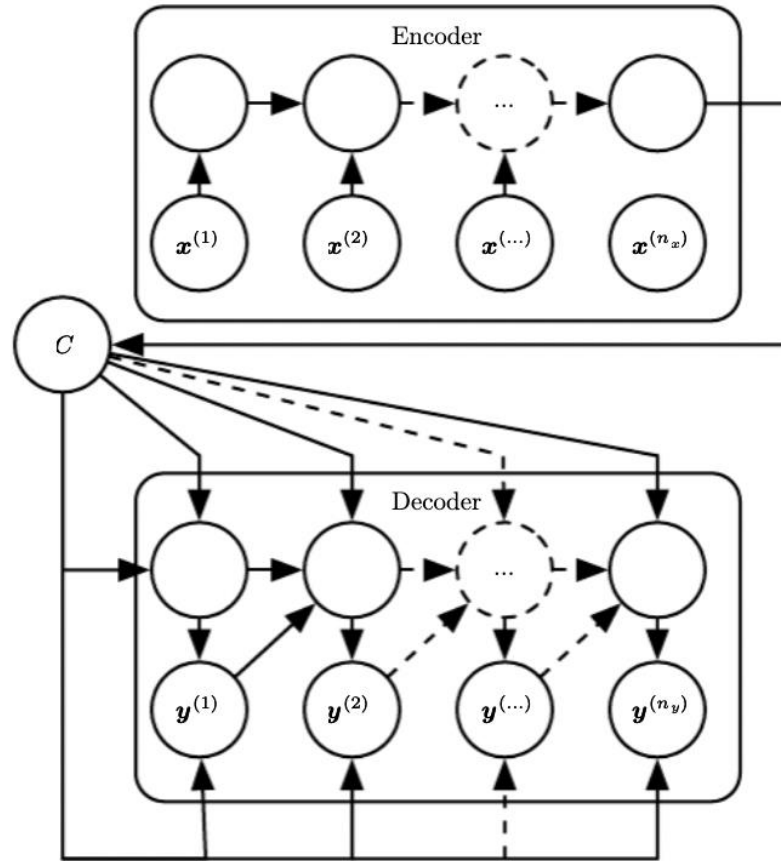


Input-output correlation



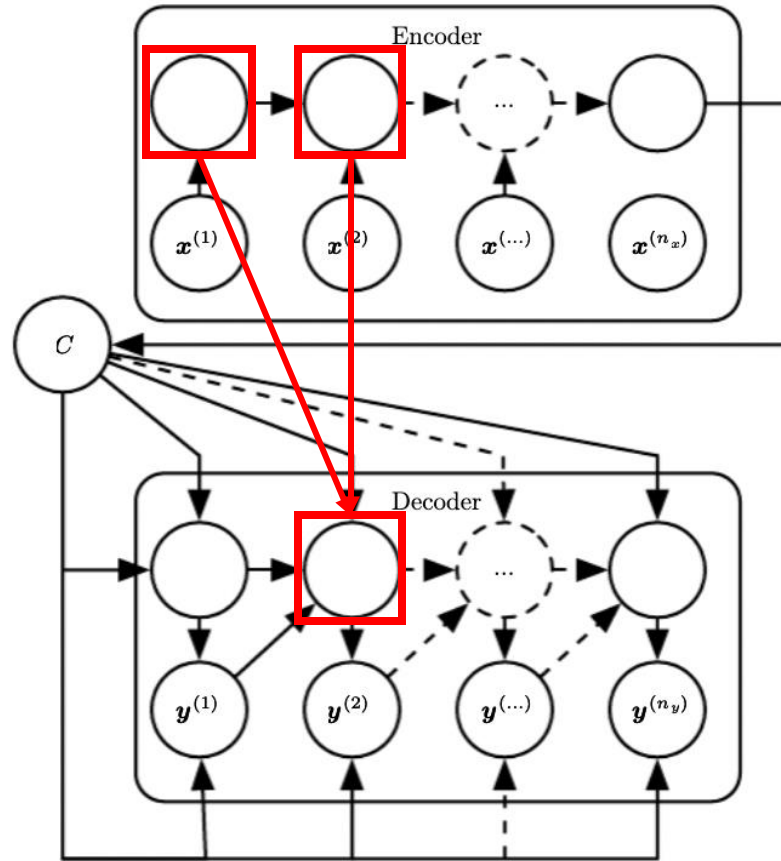
Encoder-decoder

Q: can we create information flow between encoder and decoder nodes?

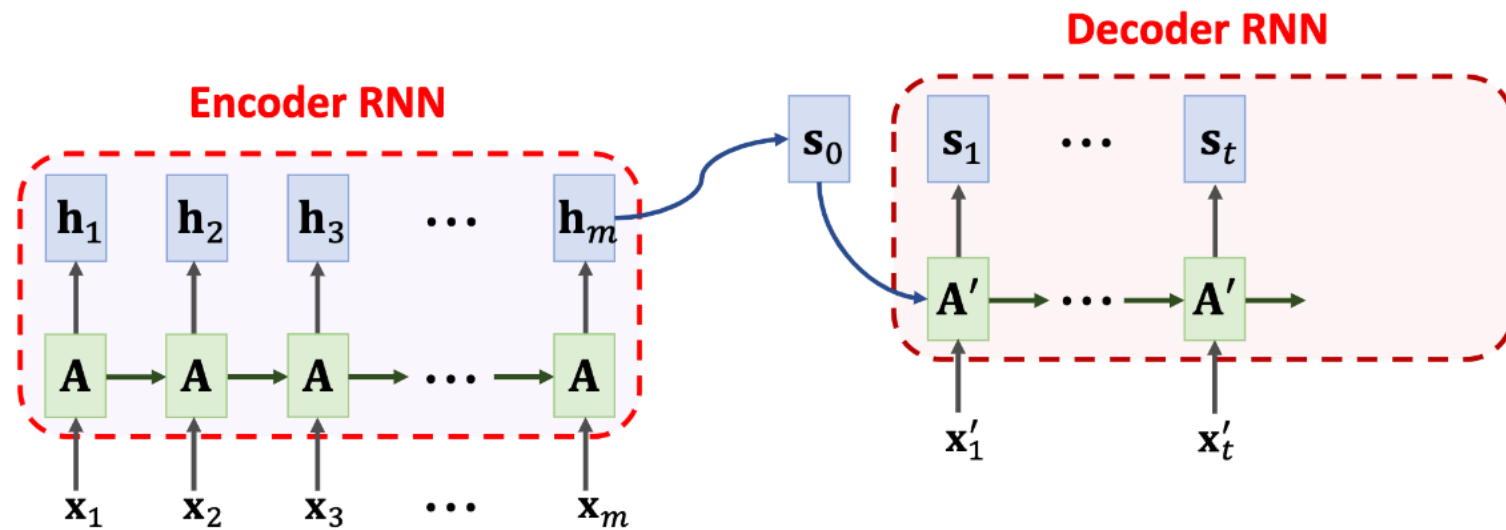


Encoder-decoder

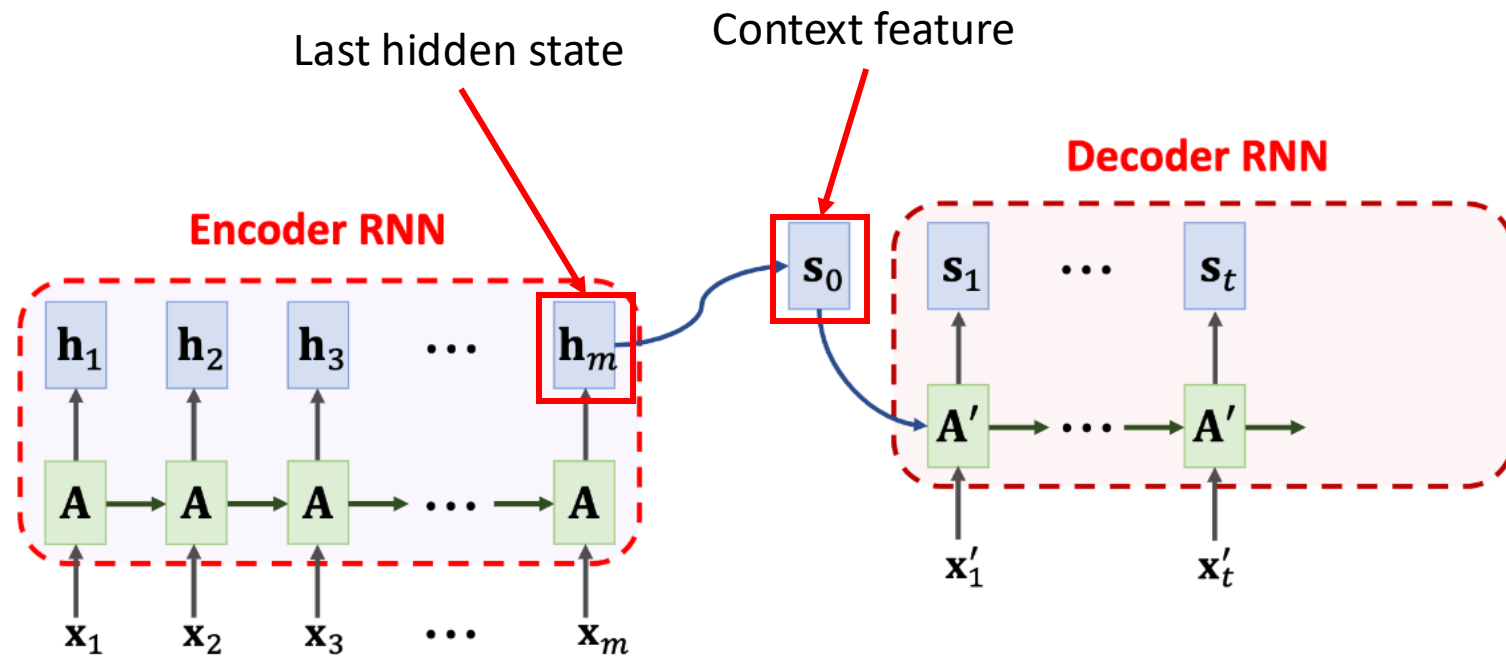
Q: can we create **information flow** between encoder and decoder nodes?



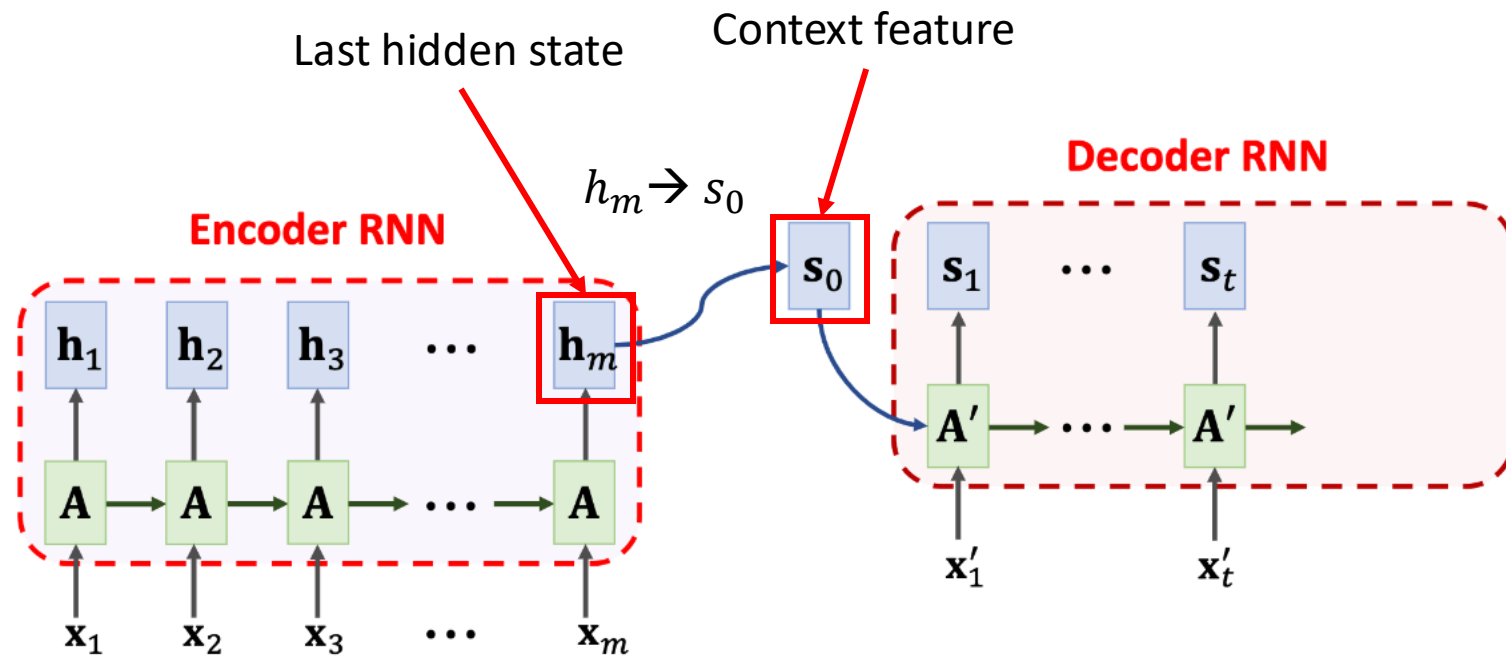
Attention mechanism



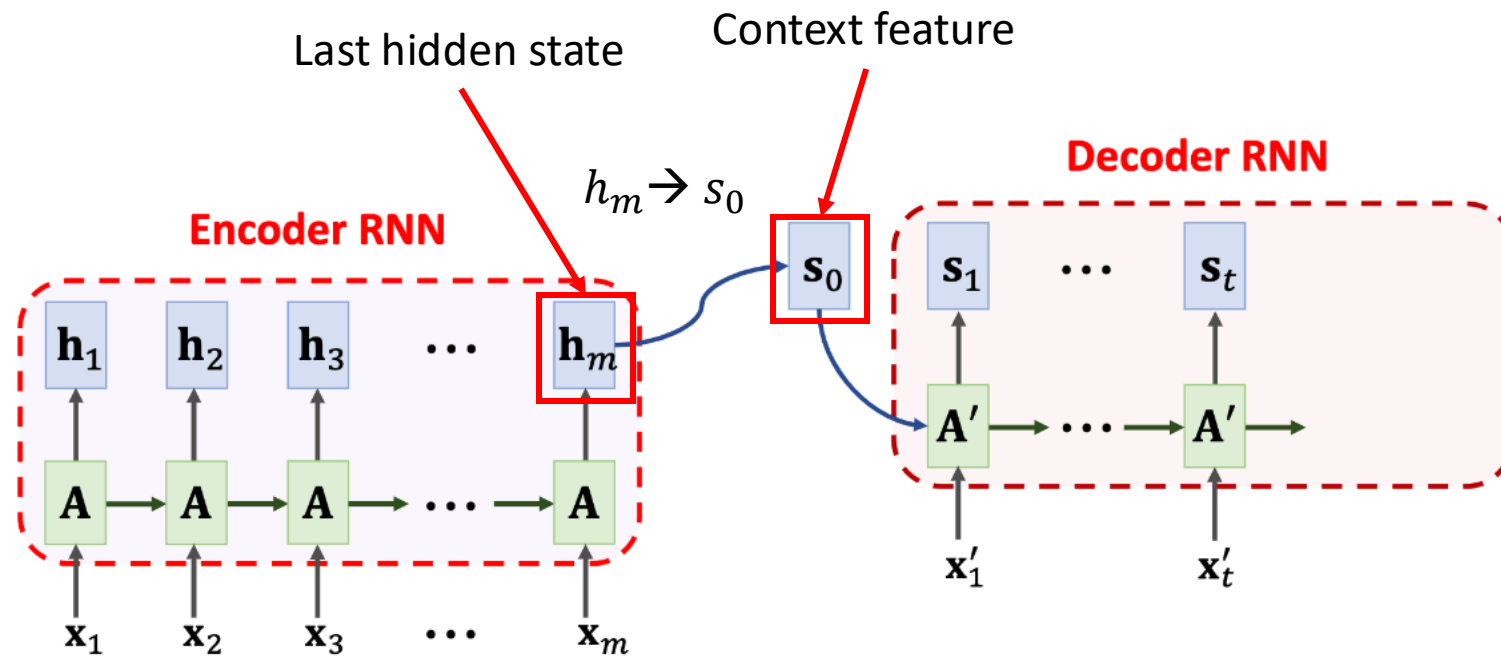
Attention mechanism



Attention mechanism

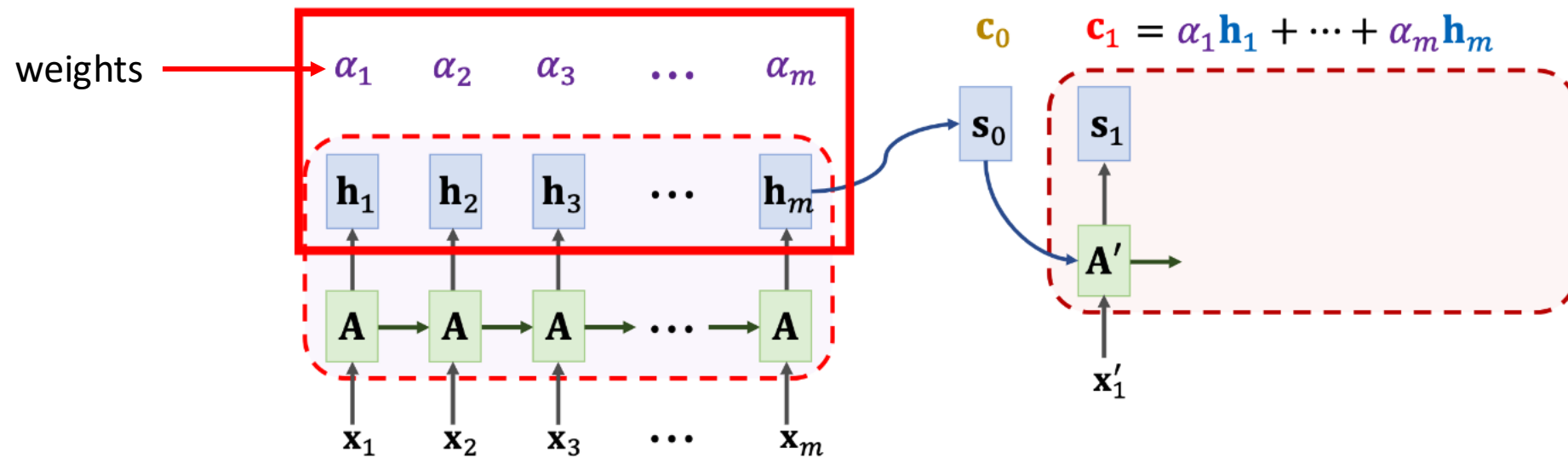


Attention mechanism

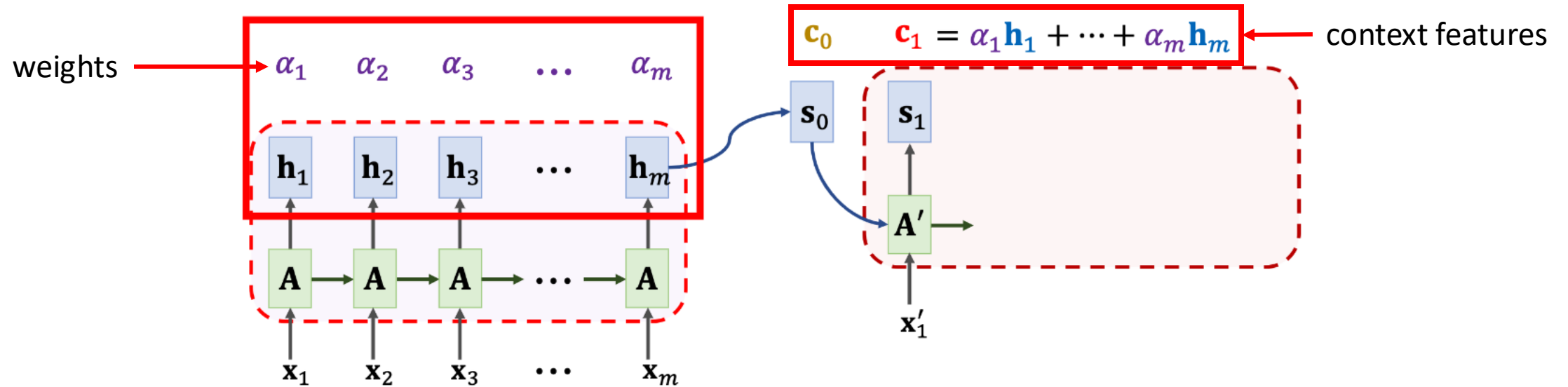


No direct connection between hidden states of encoder and decoder

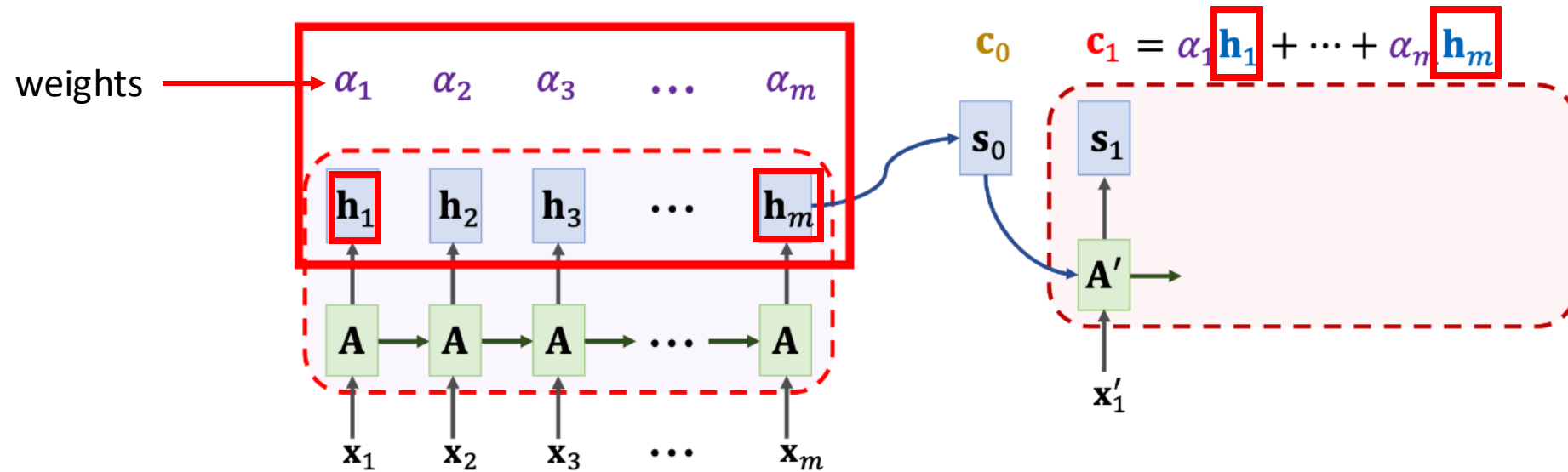
Attention mechanism



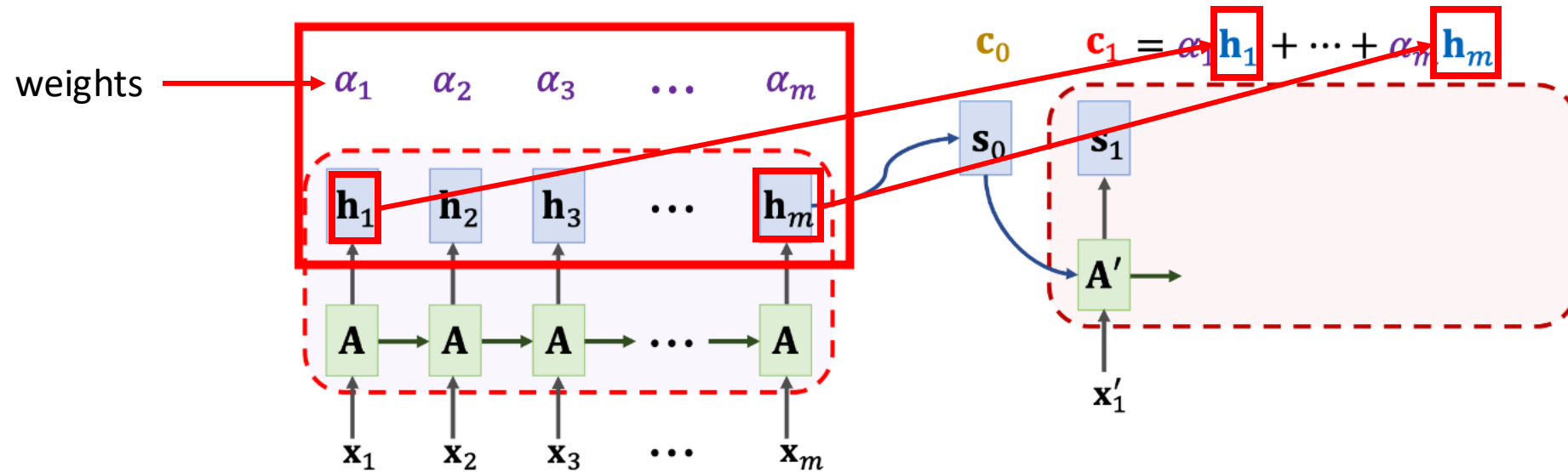
Attention mechanism



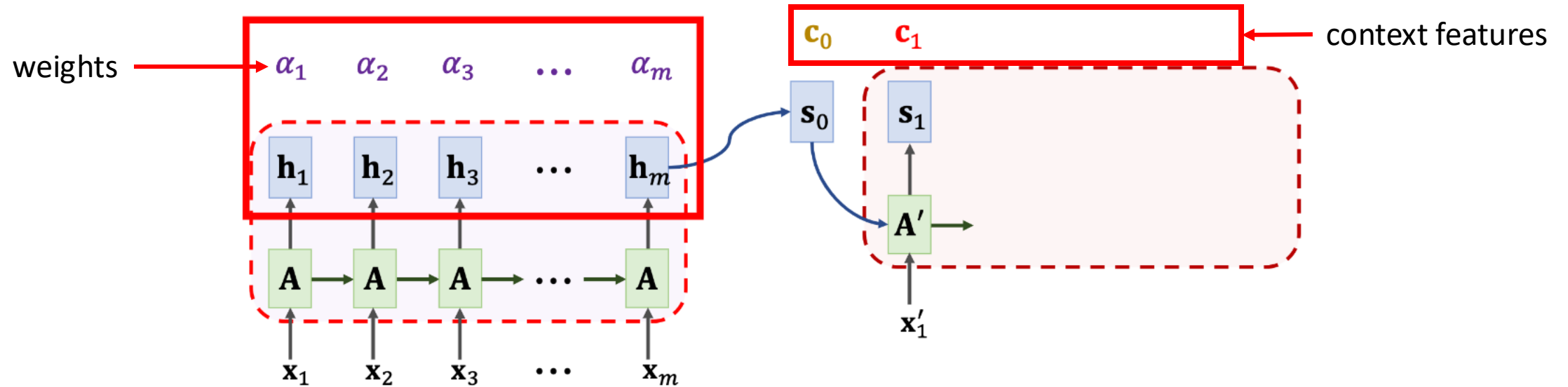
Attention mechanism



Attention mechanism

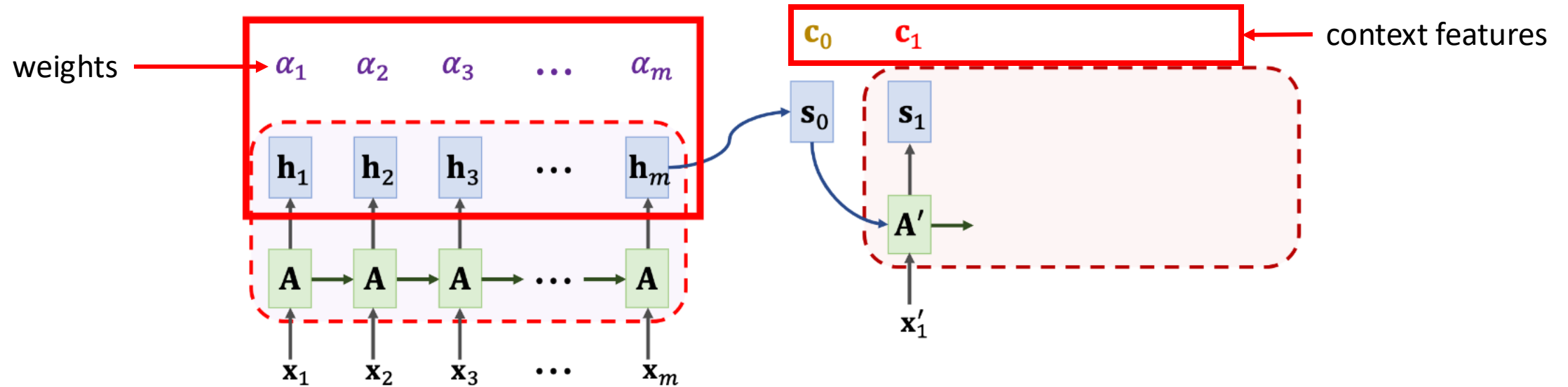


Attention mechanism



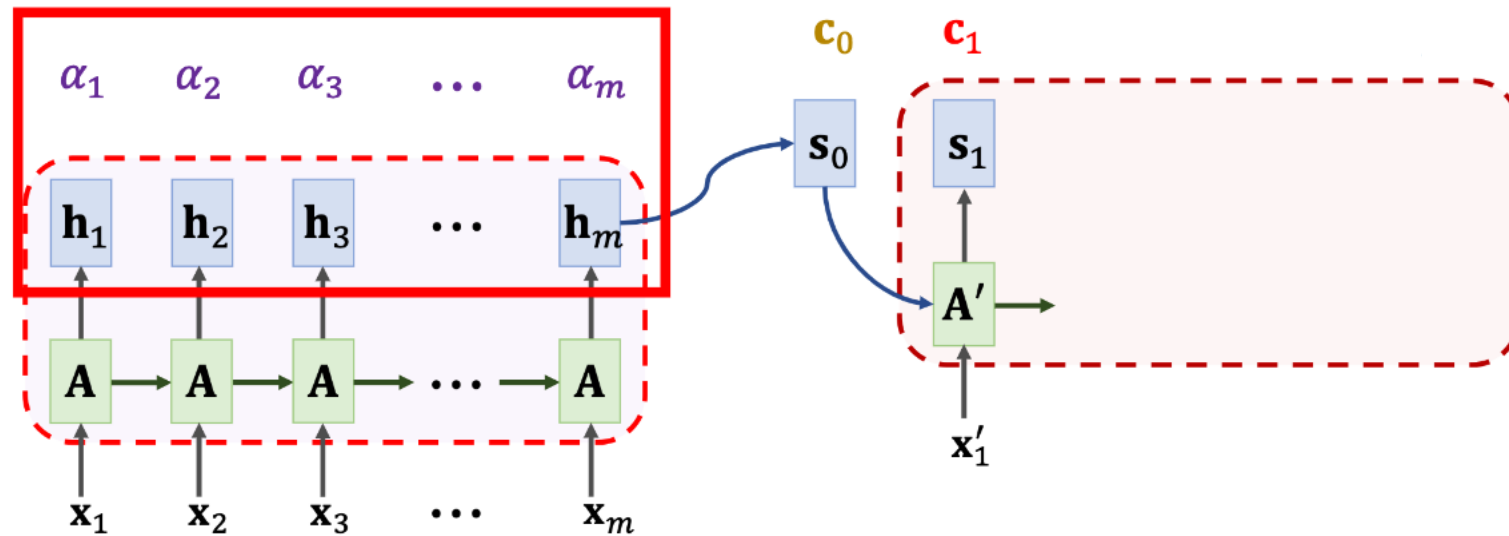
Attention mechanism

How to use the two variables to build information flow?



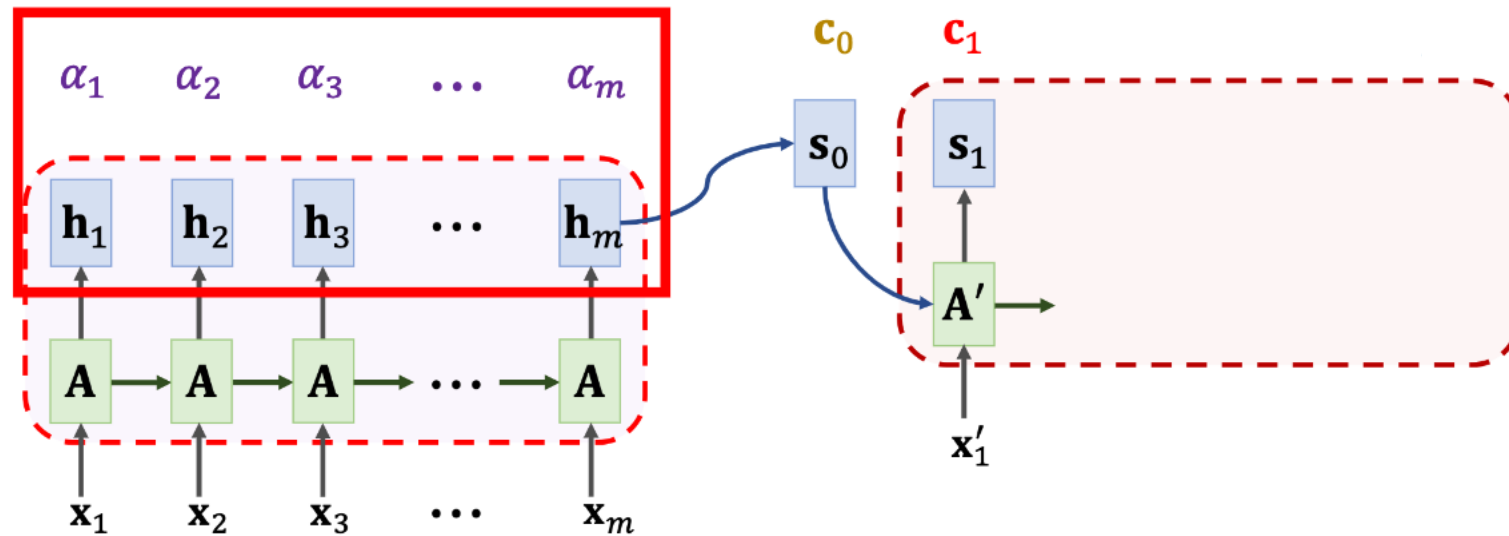
Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



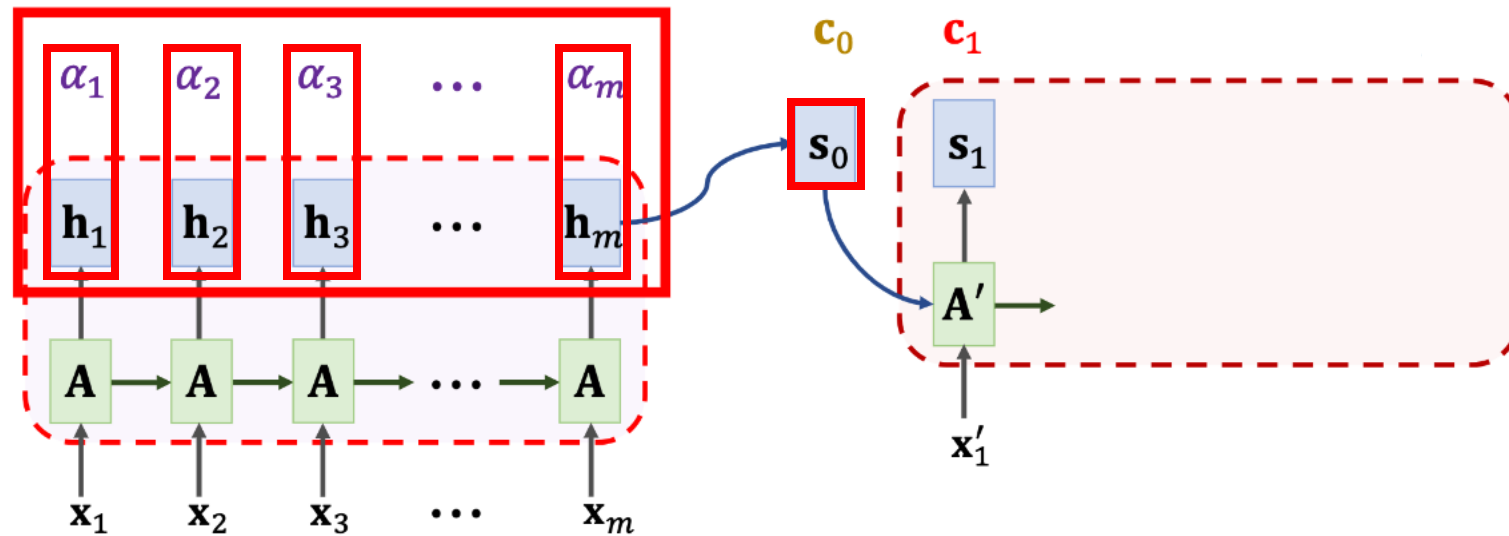
Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



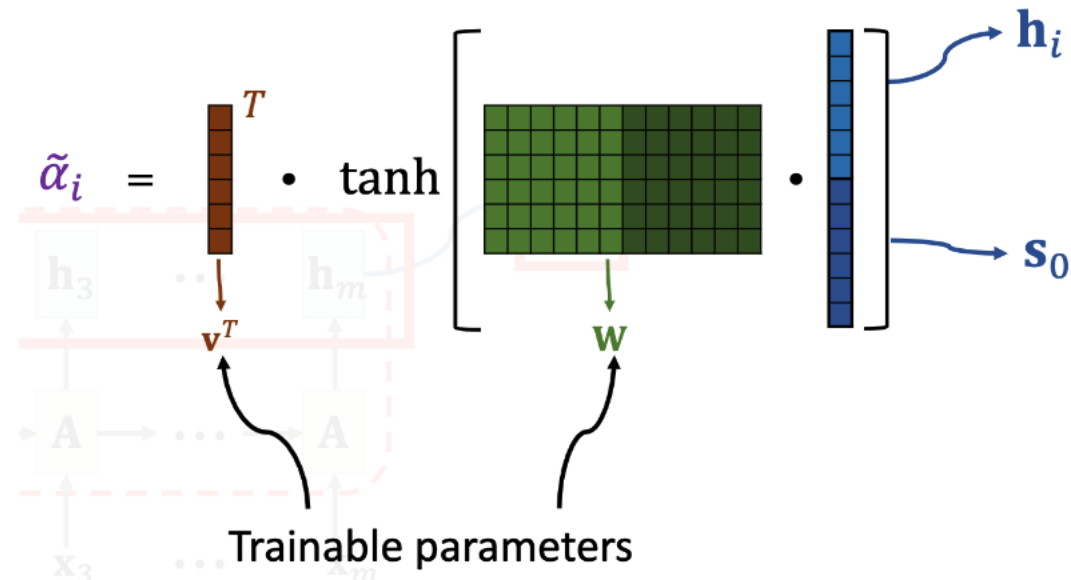
Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

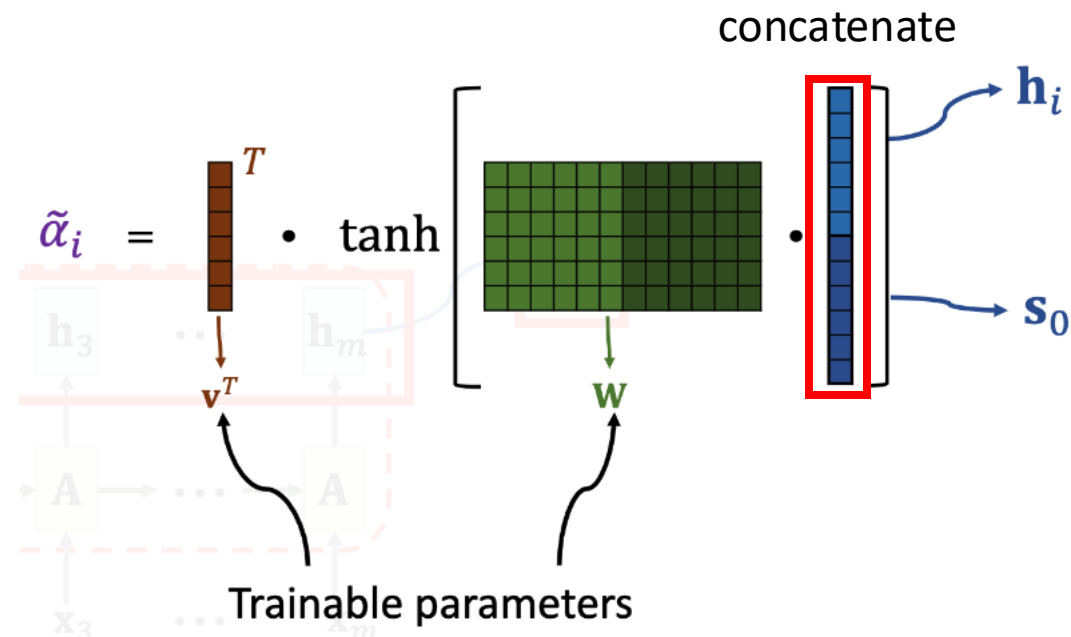


Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

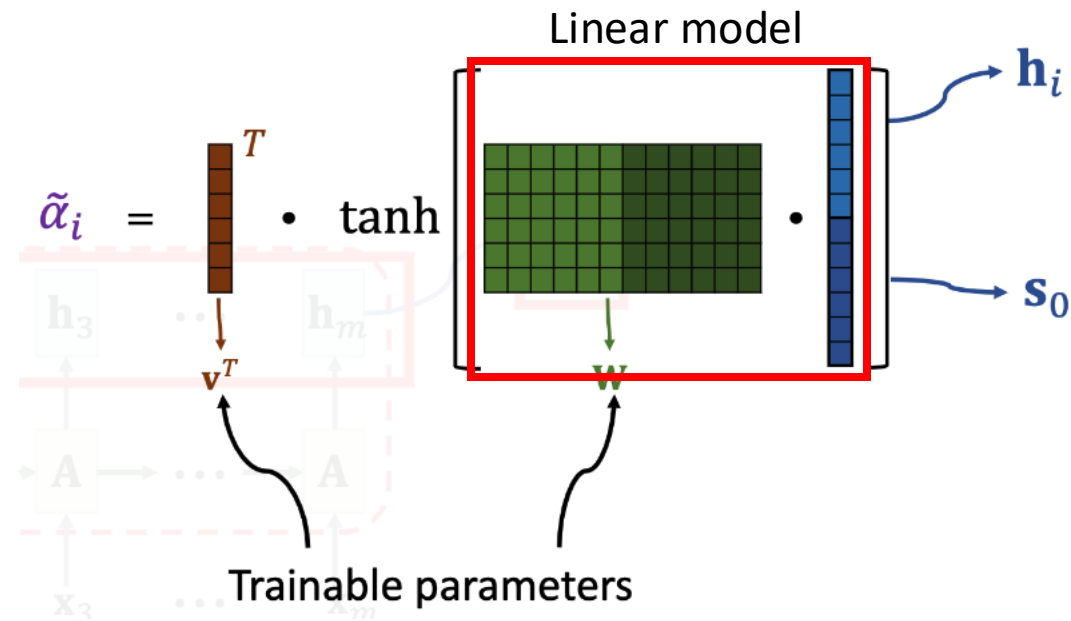


Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

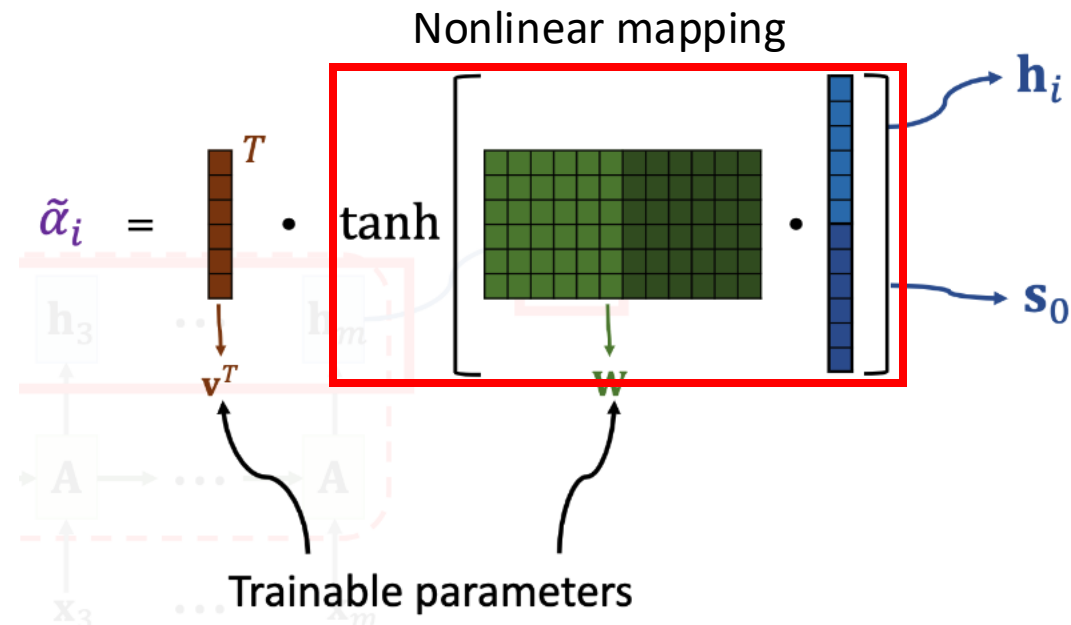


Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

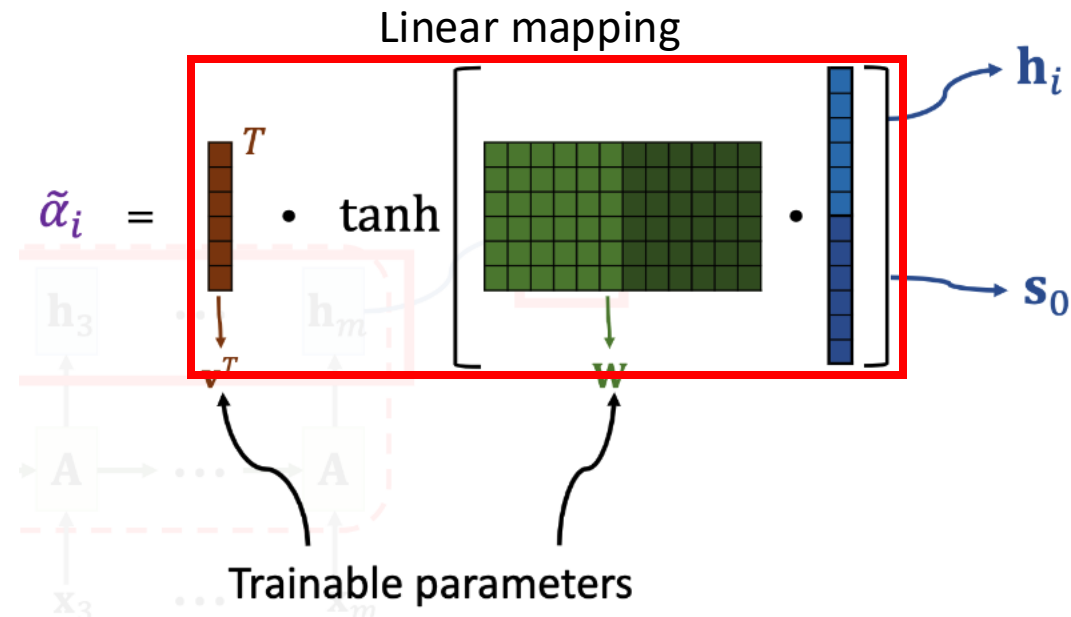


Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

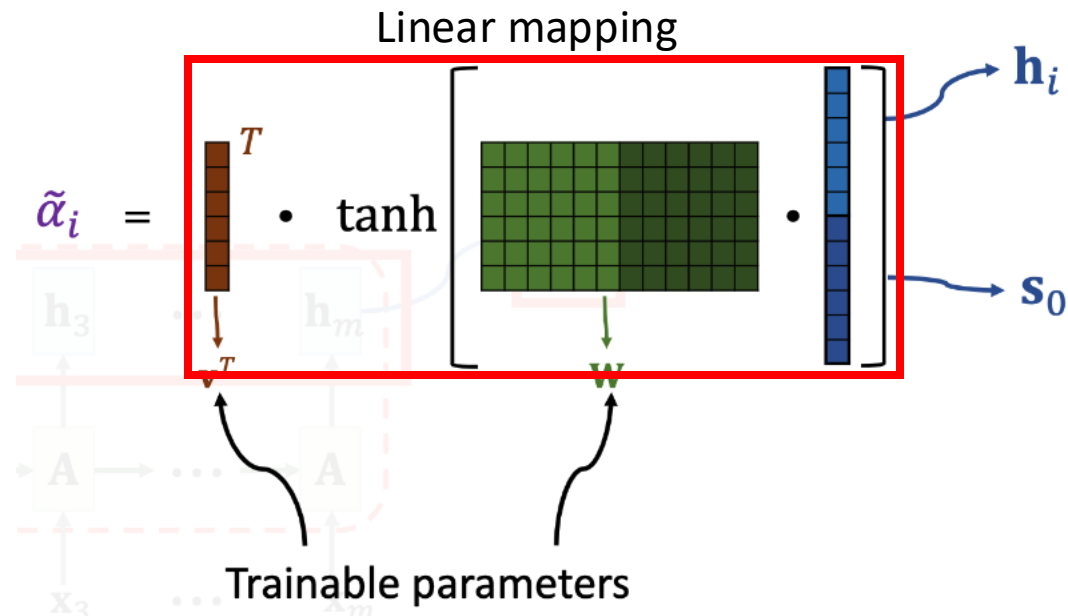


Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.



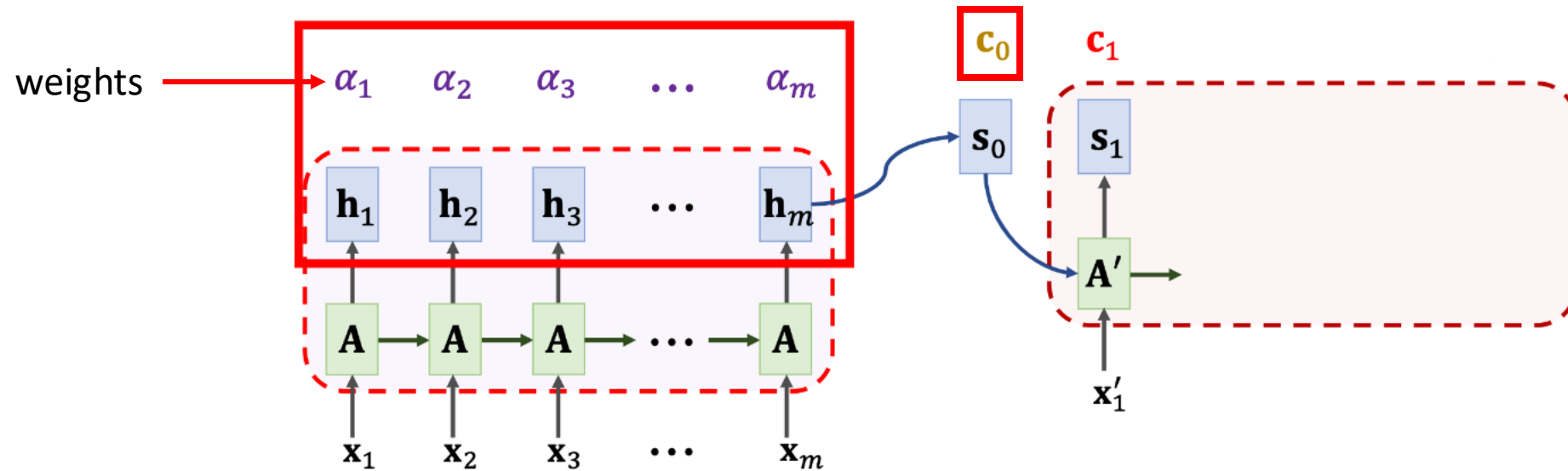
Then **normalize** $\tilde{\alpha}_1, \dots, \tilde{\alpha}_m$ (so that they sum to 1):

$$[\alpha_1, \dots, \alpha_m] = \text{Softmax}([\tilde{\alpha}_1, \dots, \tilde{\alpha}_m]).$$

Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

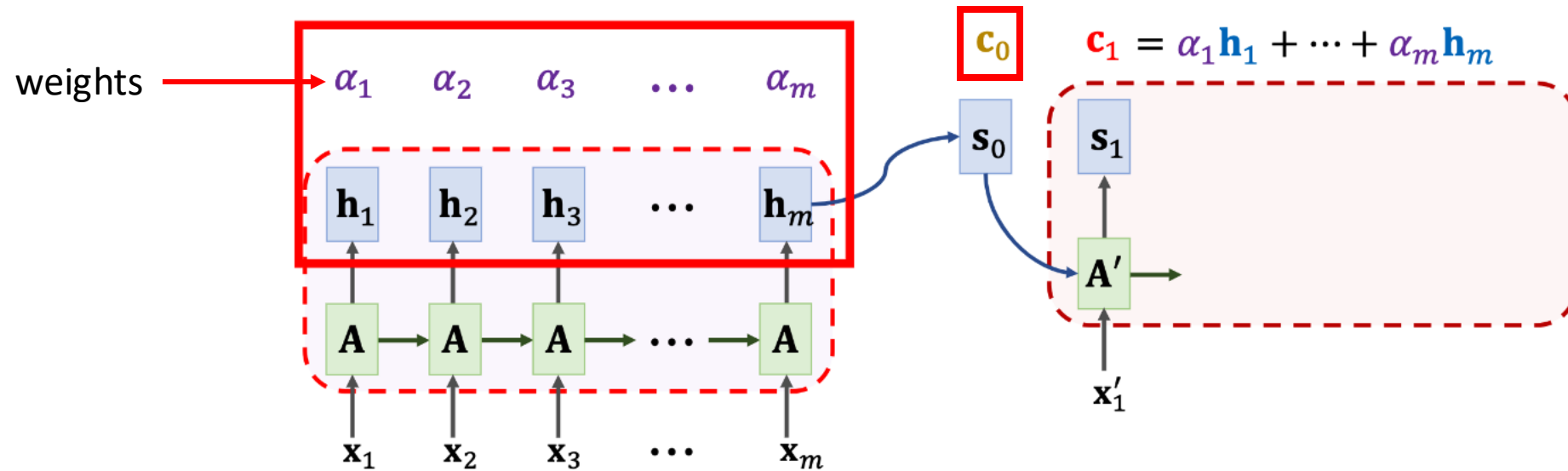
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.



Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

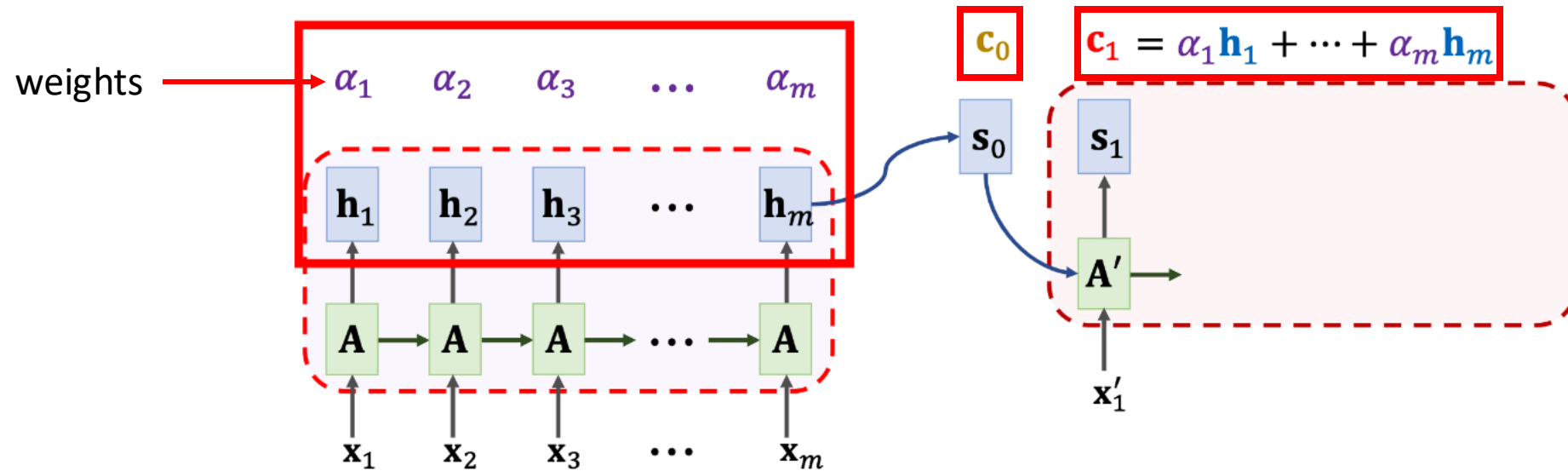
Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.



Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.

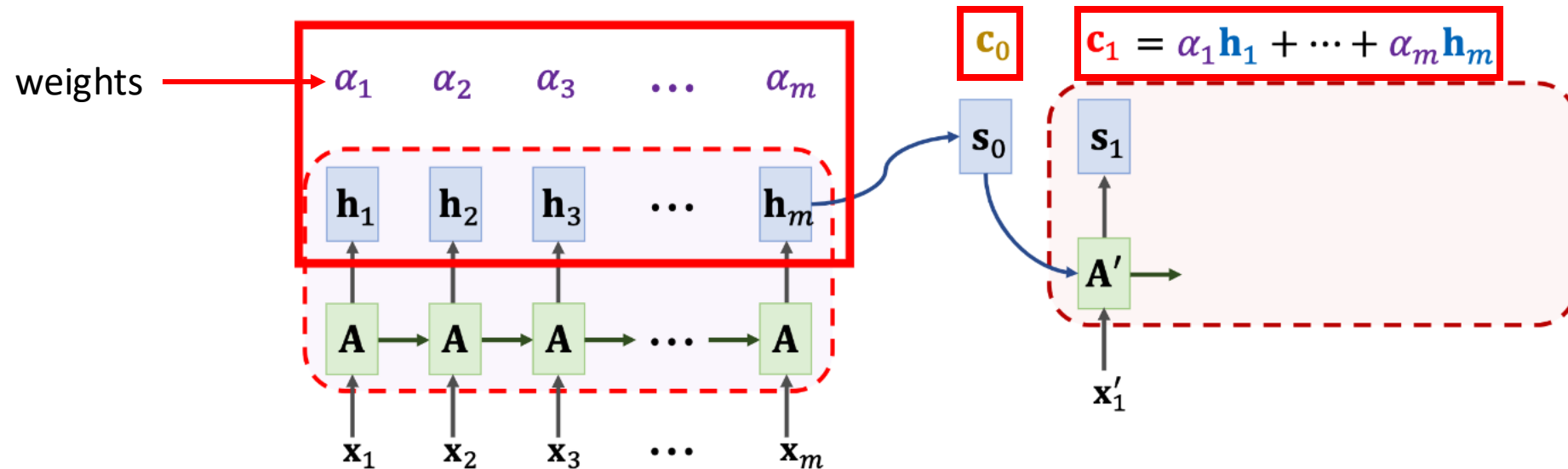


Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.

Q: should we use different α' s for different c_t ?



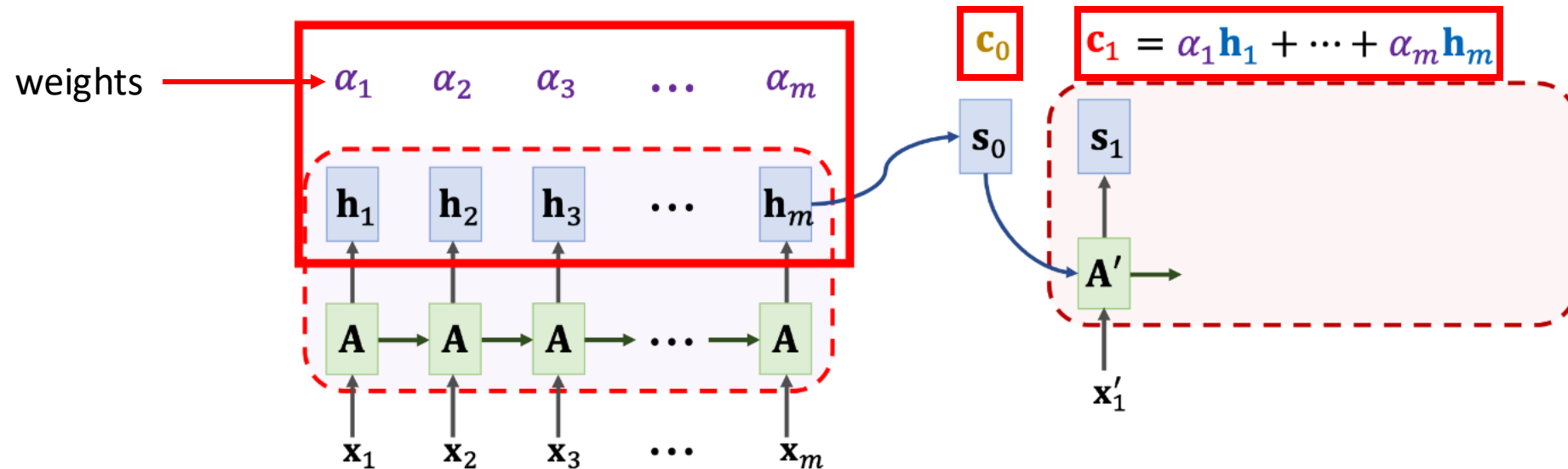
Attention mechanism

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.

Q: should we use different α 's for different c_t ?

Different α 's. Why?



Attention mechanism

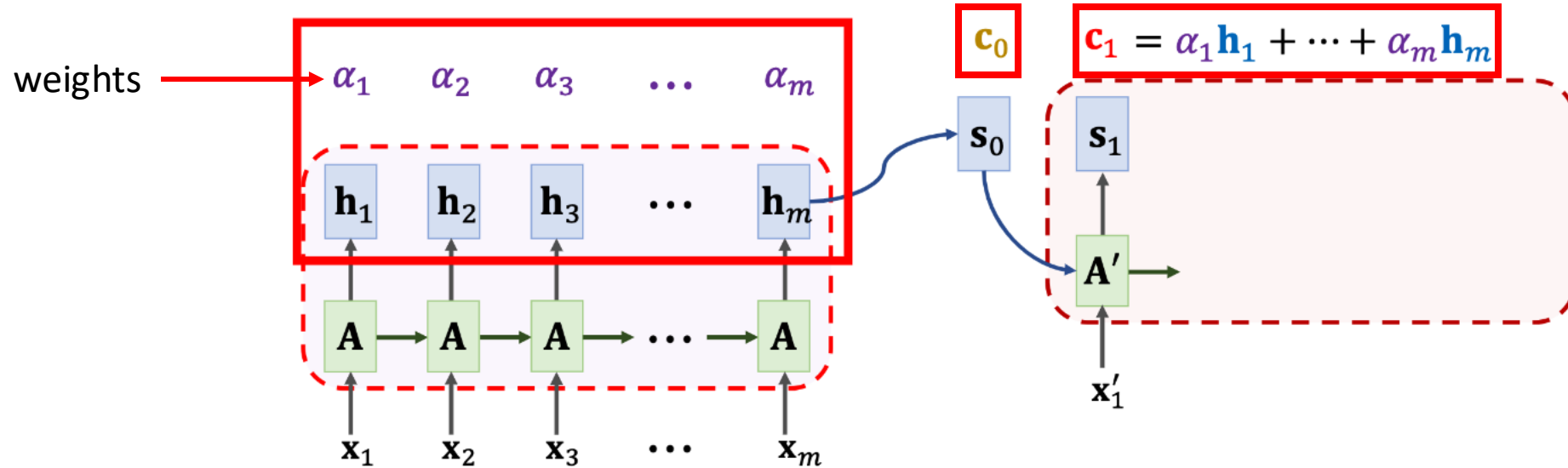
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

In the original paper

Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0).$

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m.$

Q: should we use different α' 's for different c_t ?
Different α' 's. Why?



Attention mechanism

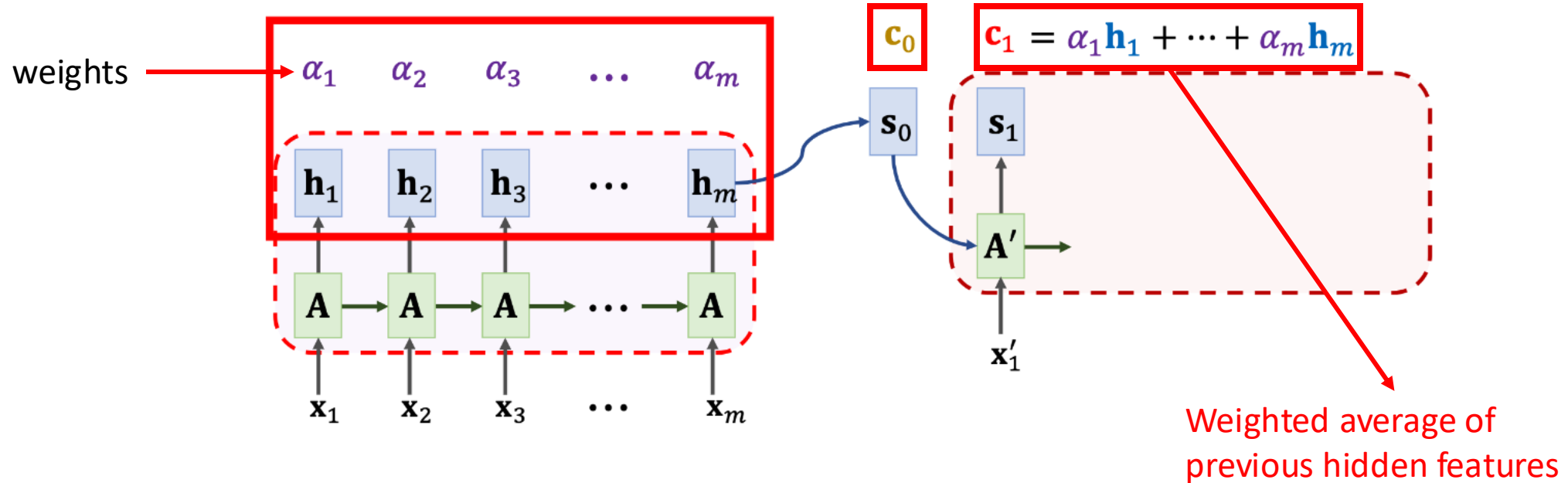
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

In the original paper

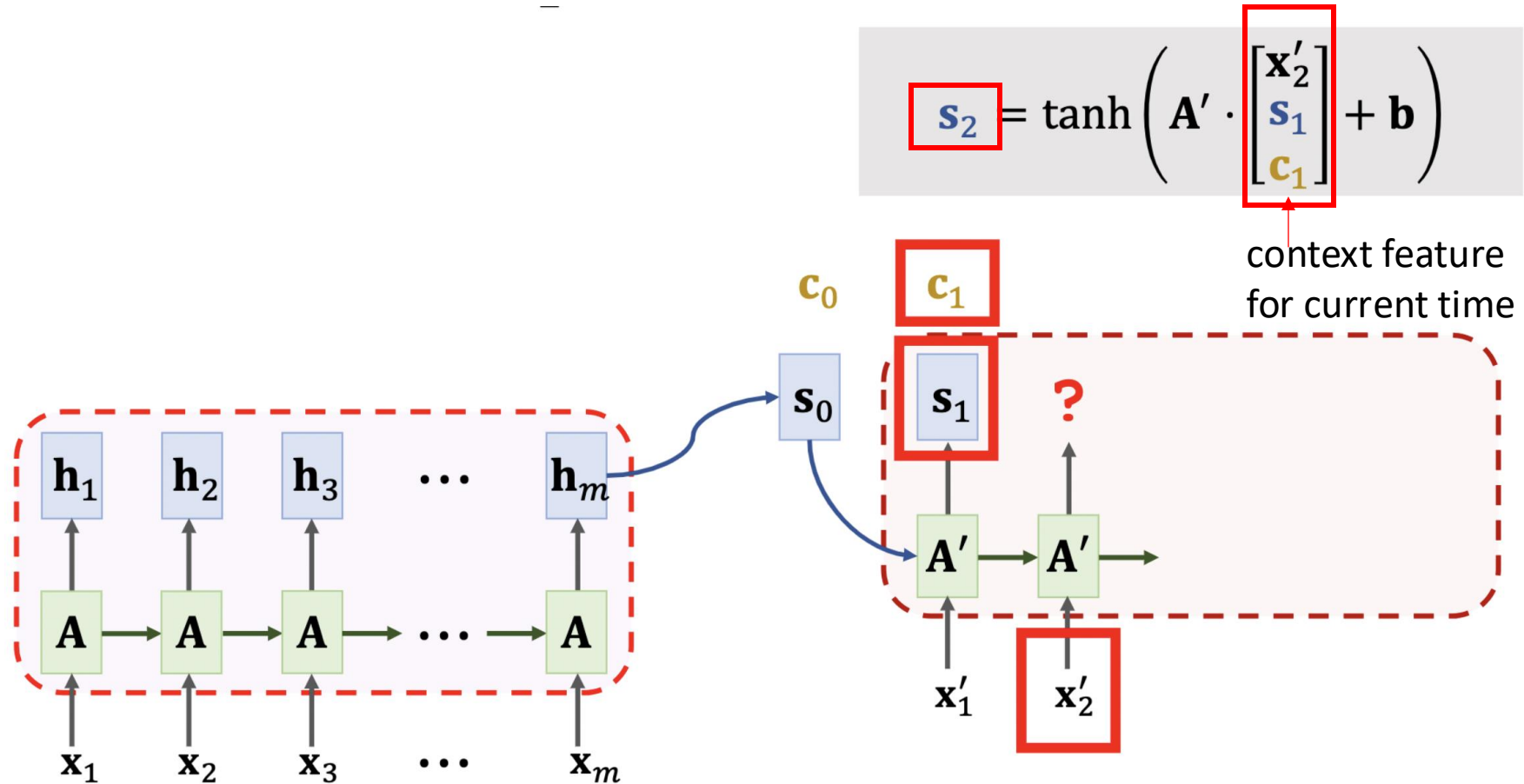
Weight: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{s}_0)$.

Context vector: $\mathbf{c}_0 = \alpha_1 \mathbf{h}_1 + \dots + \alpha_m \mathbf{h}_m$.

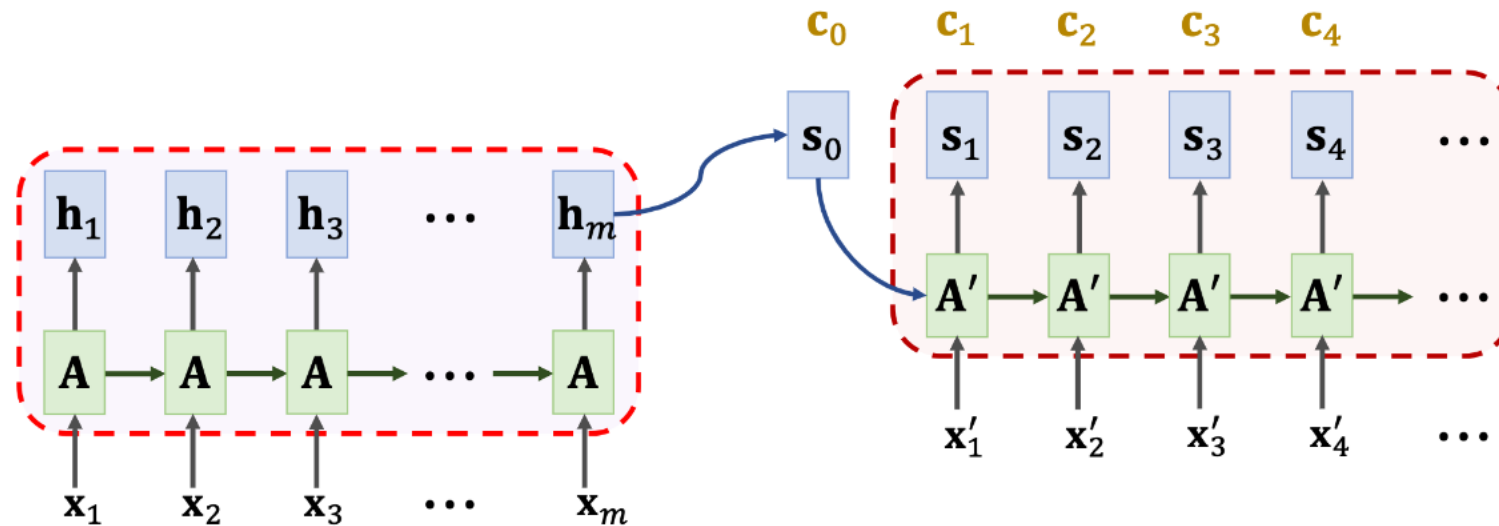
Q: should we use different α' 's for different c_t ?
Different α' 's. Why?



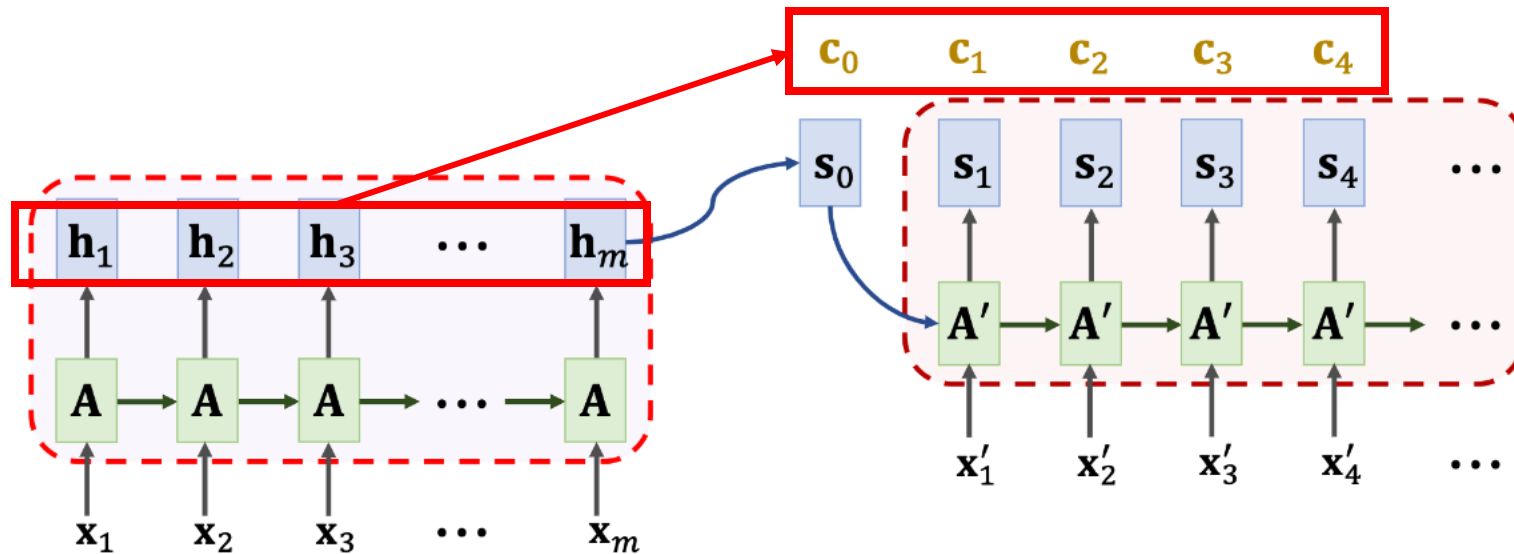
Attention mechanism



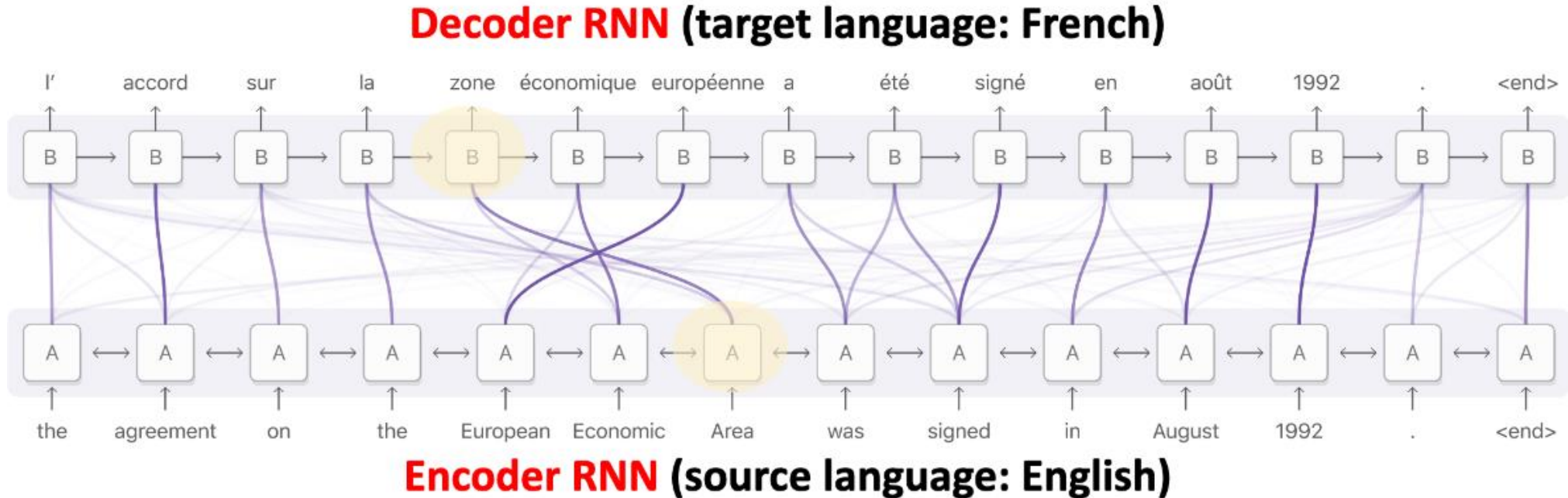
Attention mechanism



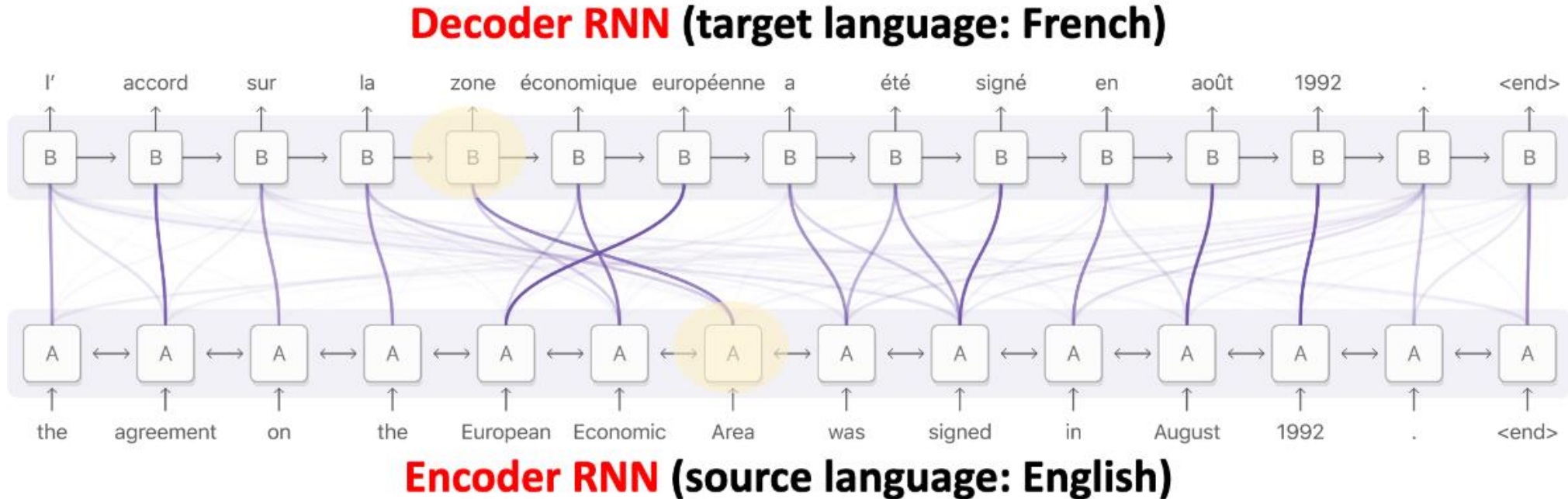
Attention mechanism



Input-output correlation



Input-output correlation



Q: can we build attention mechanism in a single RNN (e.g., the encoder)?

Self-attention (intra-attention)

SimpleRNN:

$$\mathbf{h}_1 = \tanh\left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b}\right)$$

\mathbf{c}_0



Self-attention (intra-attention)

SimpleRNN:

$$\mathbf{h}_1 = \tanh \left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{h}_0 \end{bmatrix} + \mathbf{b} \right)$$

SimpleRNN + Self-Attention:

$$\mathbf{h}_1 = \tanh \left(\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{c}_0 \end{bmatrix} + \mathbf{b} \right)$$

\mathbf{c}_0

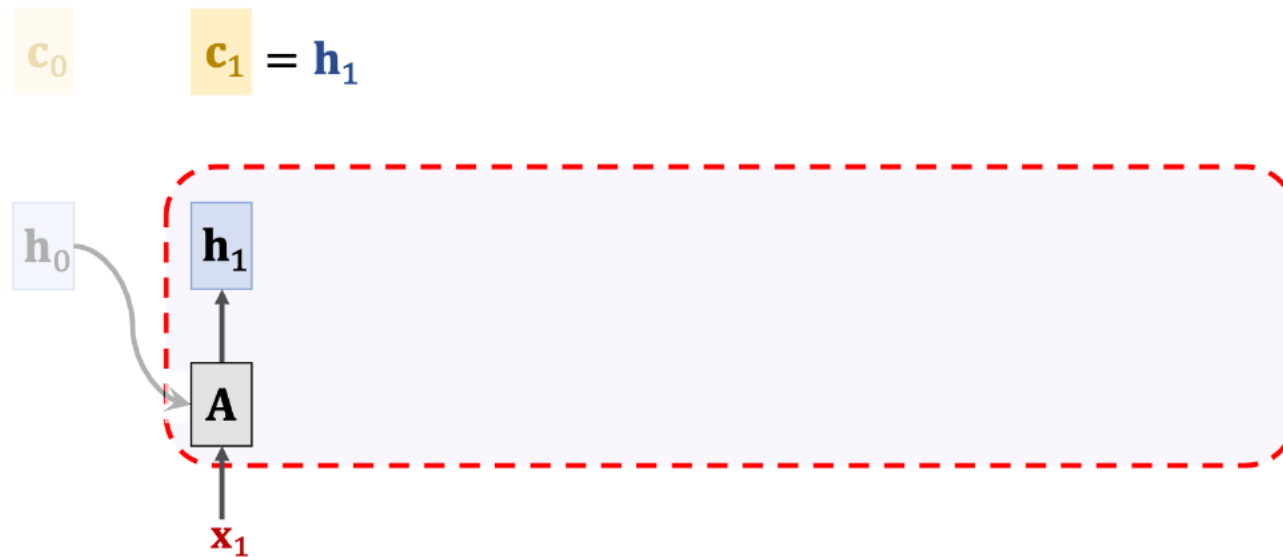
\mathbf{h}_0

?

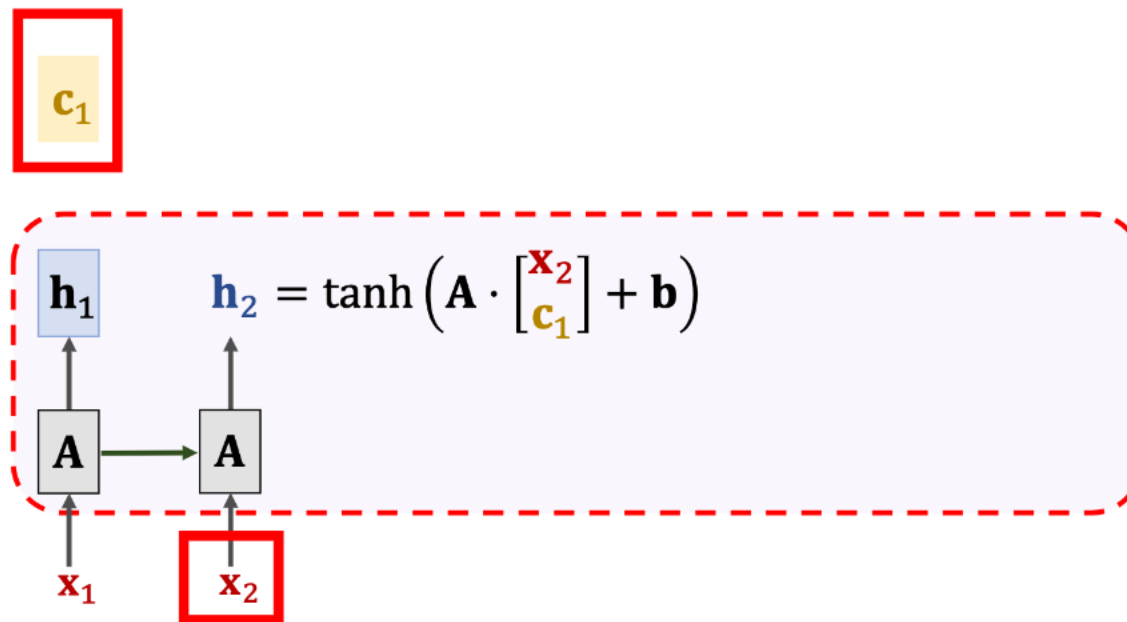
\mathbf{A}

\mathbf{x}_1

Self-attention (intra-attention)

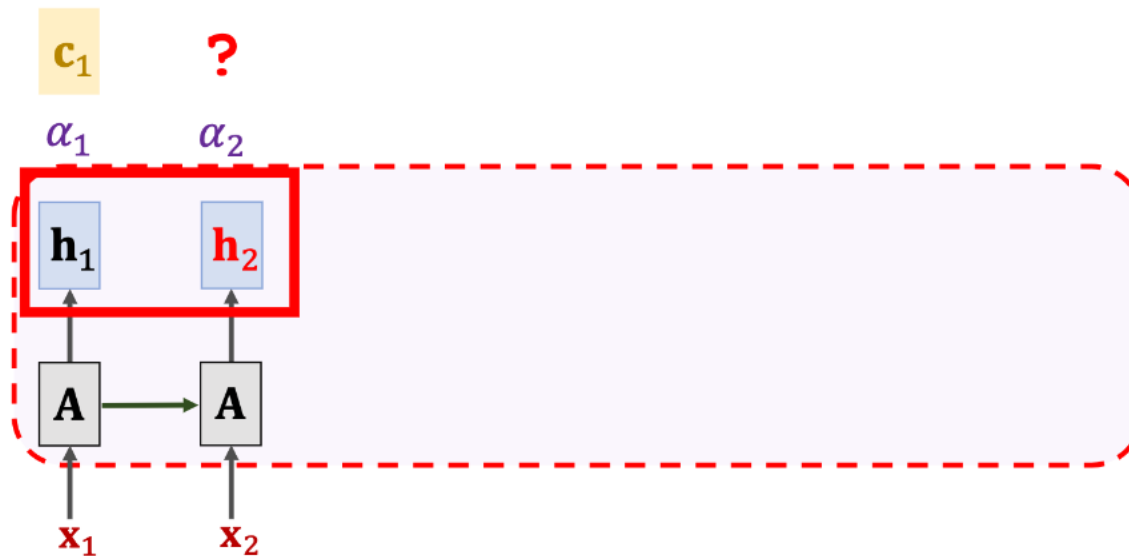


Self-attention (intra-attention)

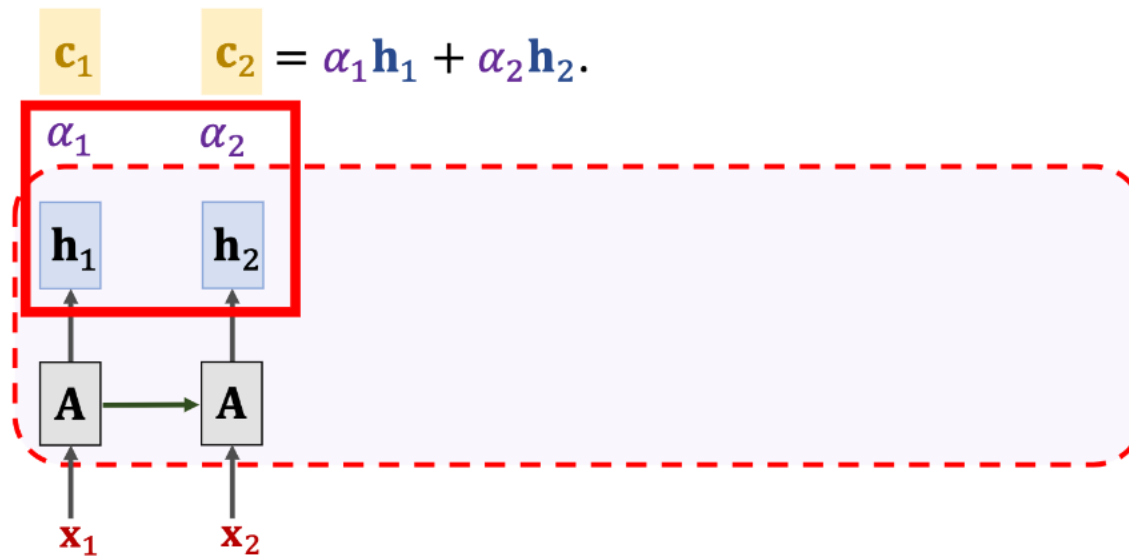


Self-attention (intra-attention)

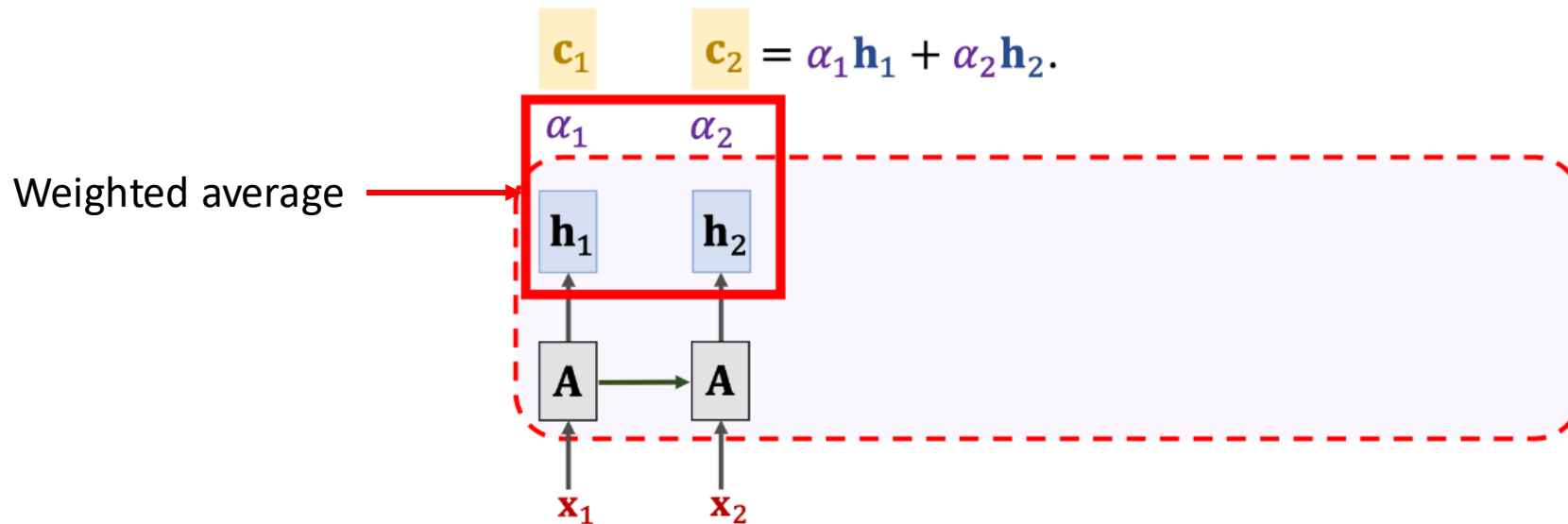
Weights: $\alpha_i = \text{align}(\mathbf{h}_i, \mathbf{h}_2)$.



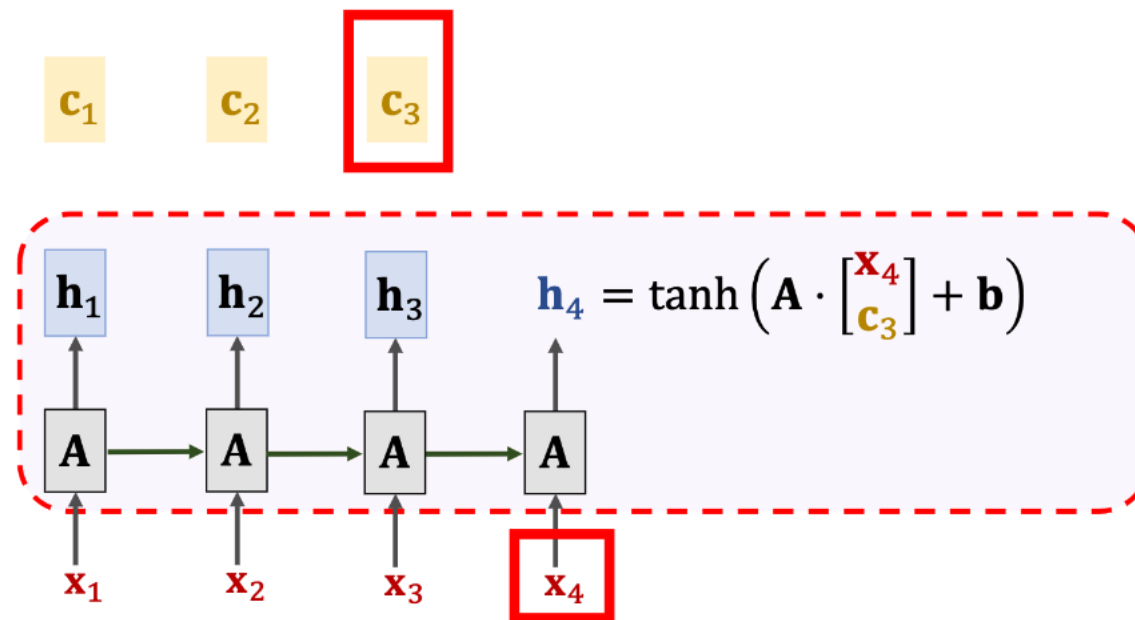
Self-attention (intra-attention)



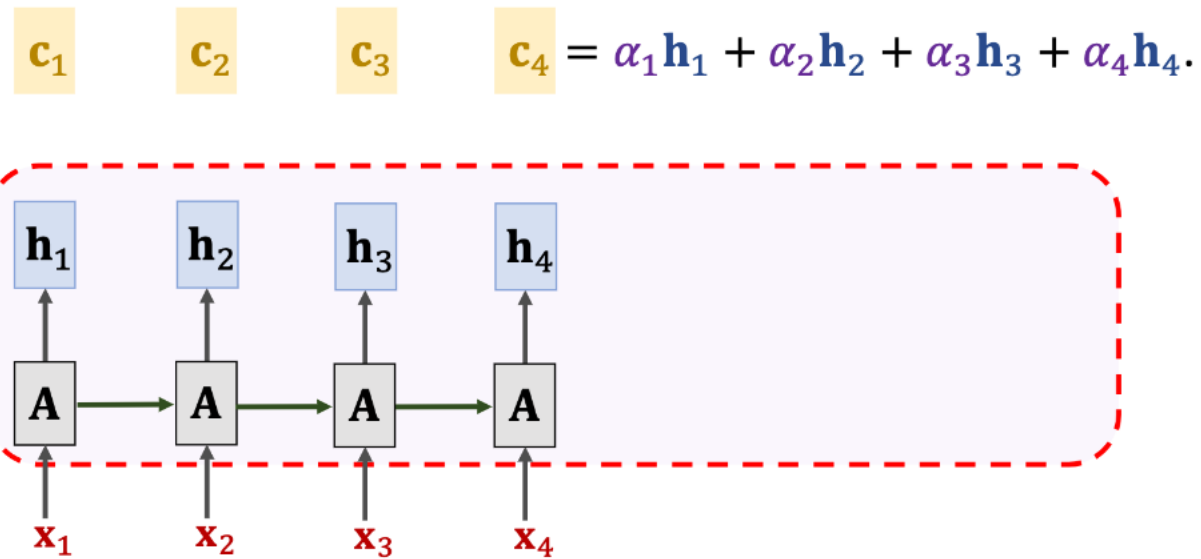
Self-attention (intra-attention)



Self-attention (intra-attention)



Self-attention (intra-attention)



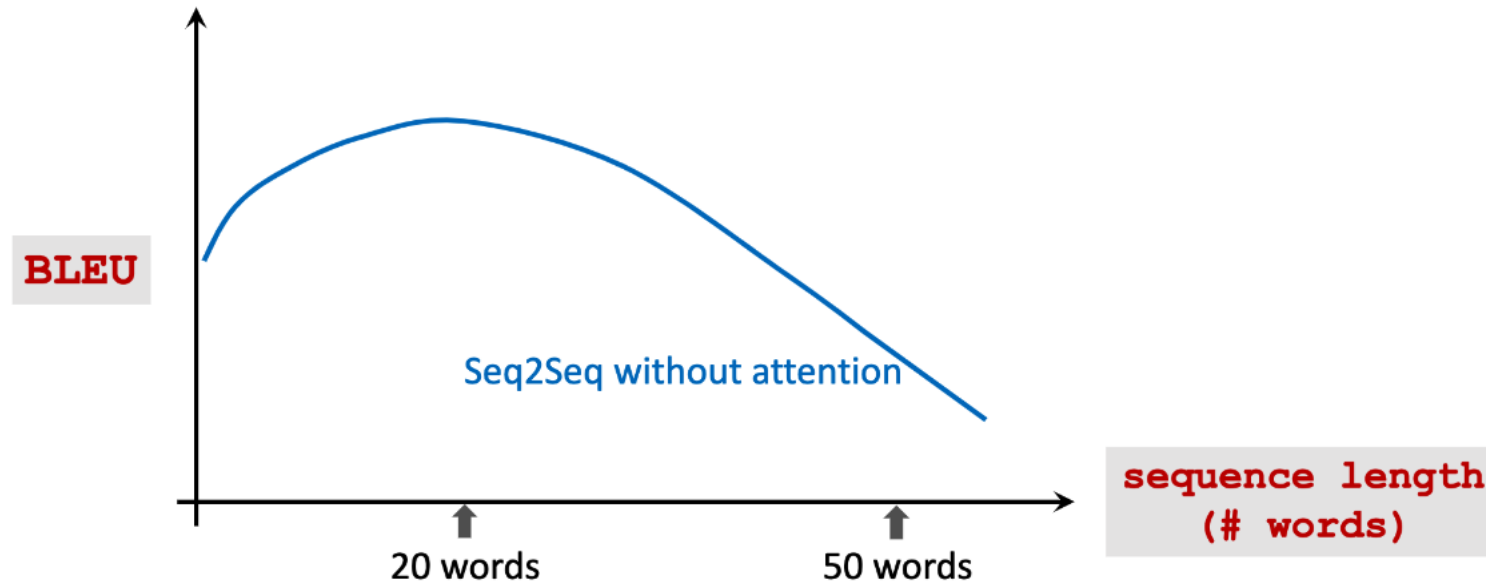
Self-attention (intra-attention)

The
The FBI
The FBI is
The FBI is chasing
The FBI is chasing a
The FBI is chasing a criminal
The FBI is chasing a criminal on
The FBI is chasing a criminal on the
The FBI is chasing a criminal on the run
The FBI is chasing a criminal on the run .

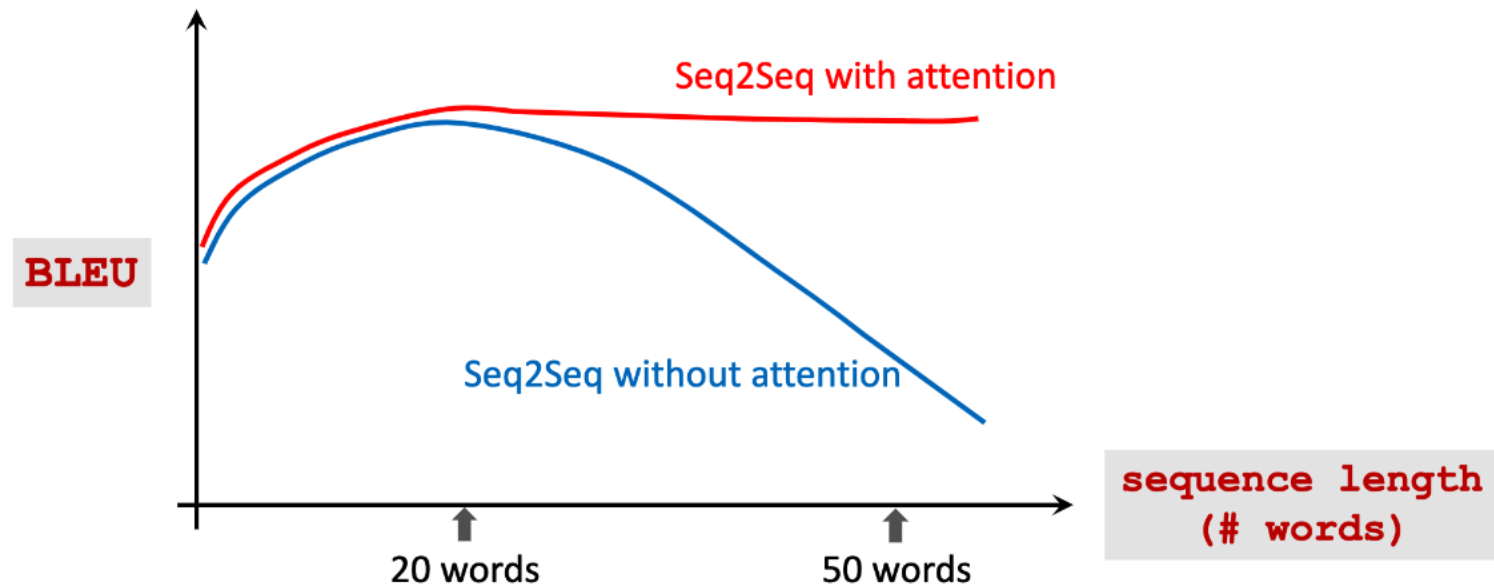
Figure is from the paper "Long Short-Term Memory-Networks for Machine Reading."

Pay attention to the context relevant to the **new input**

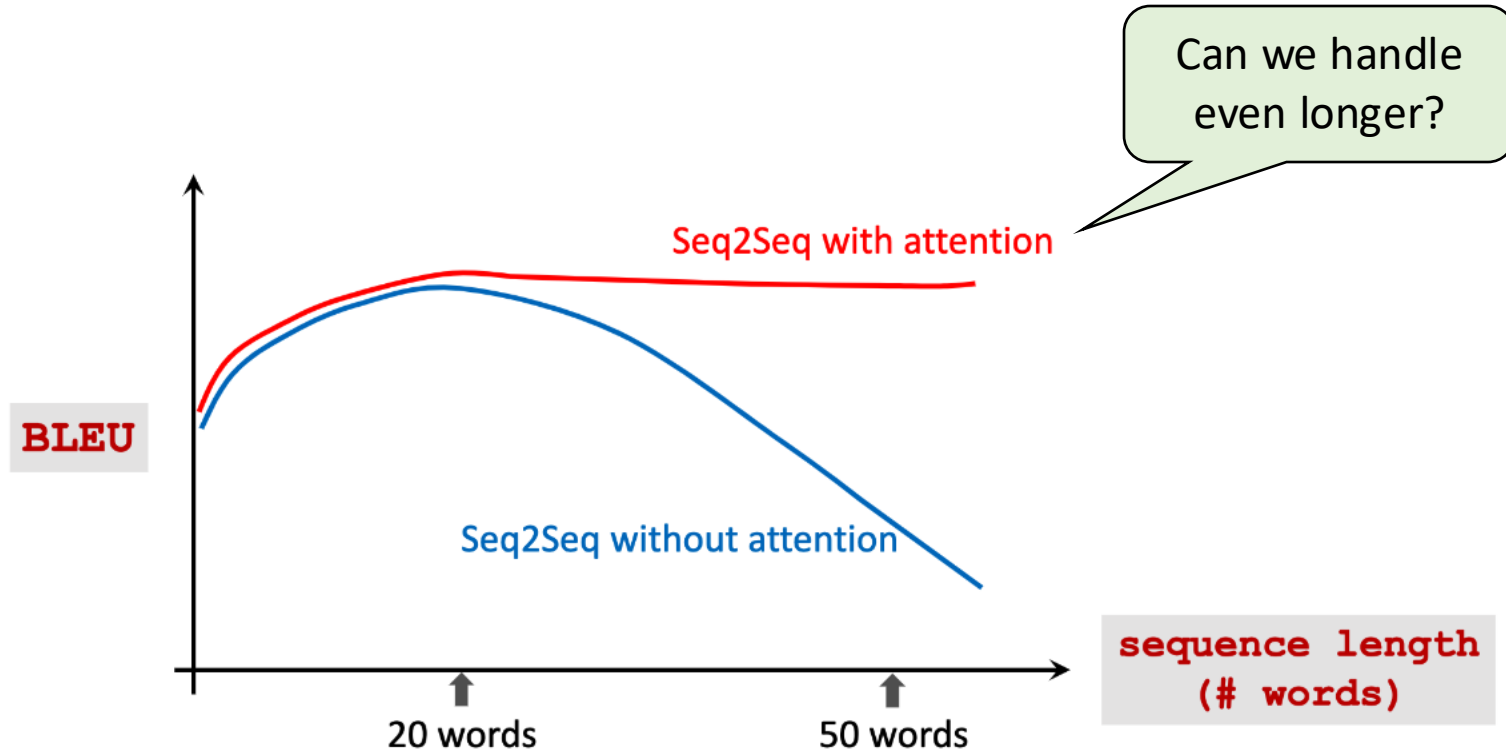
Seq2seq model performance



Seq2seq model performance



Seq2seq model performance



Transformer networks

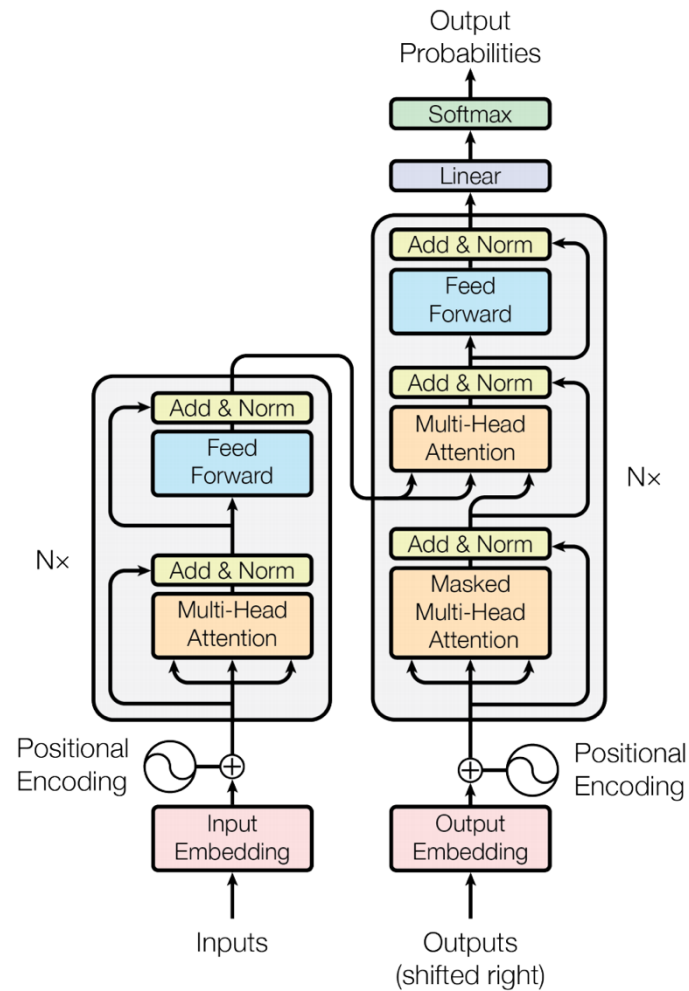


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

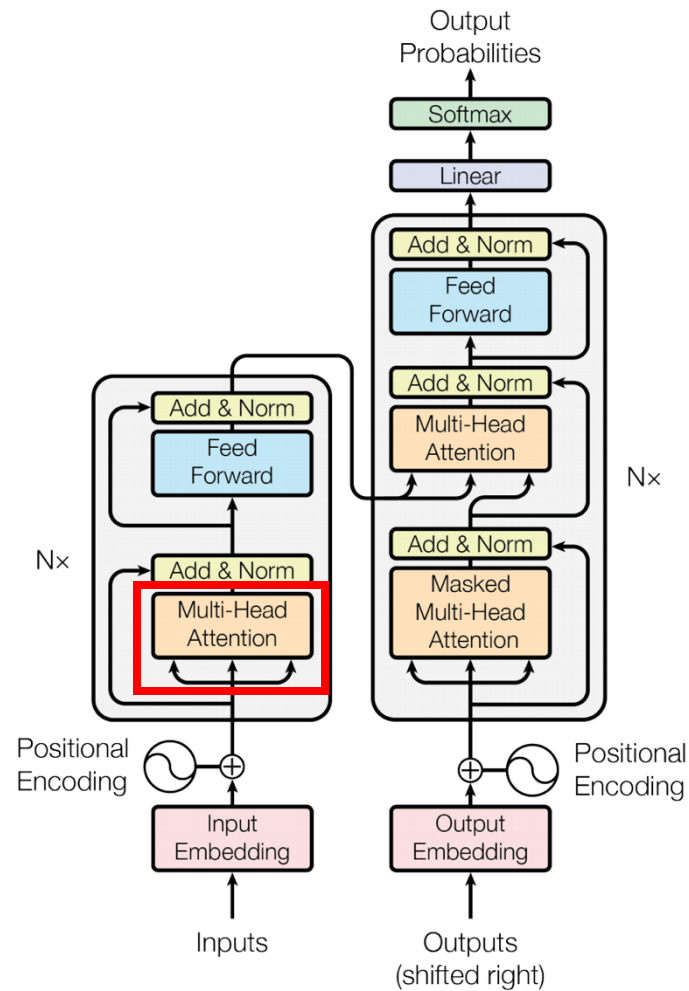


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

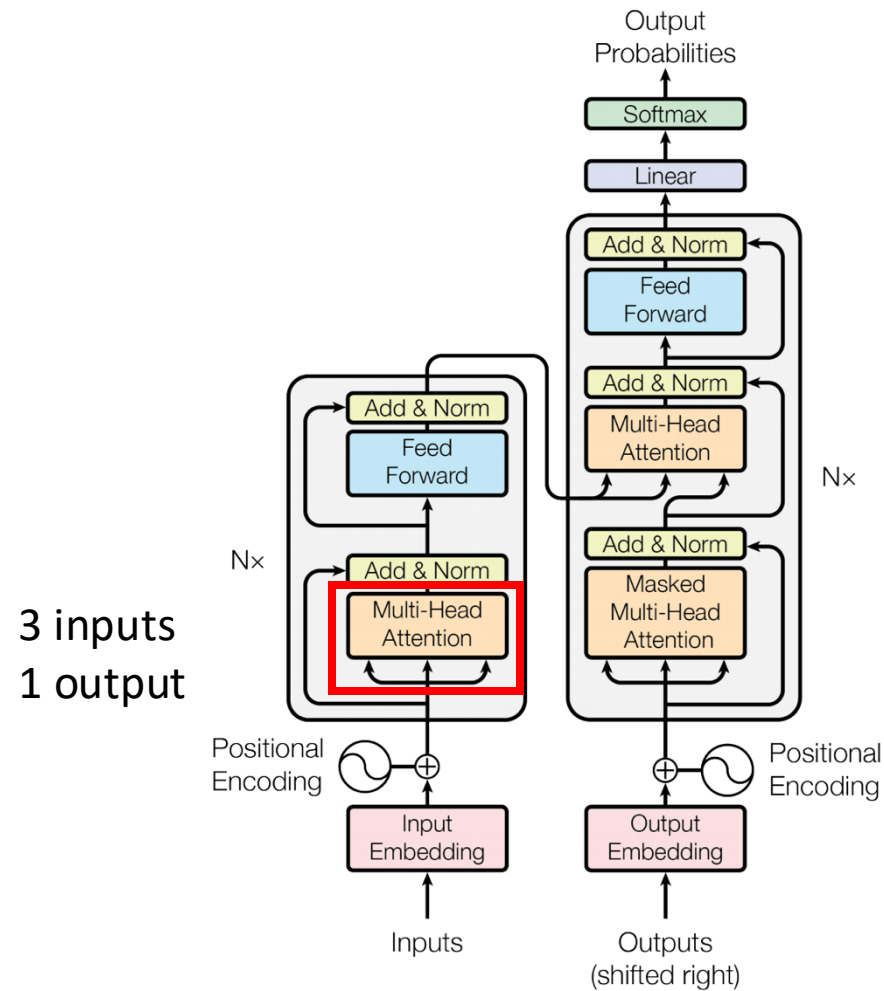
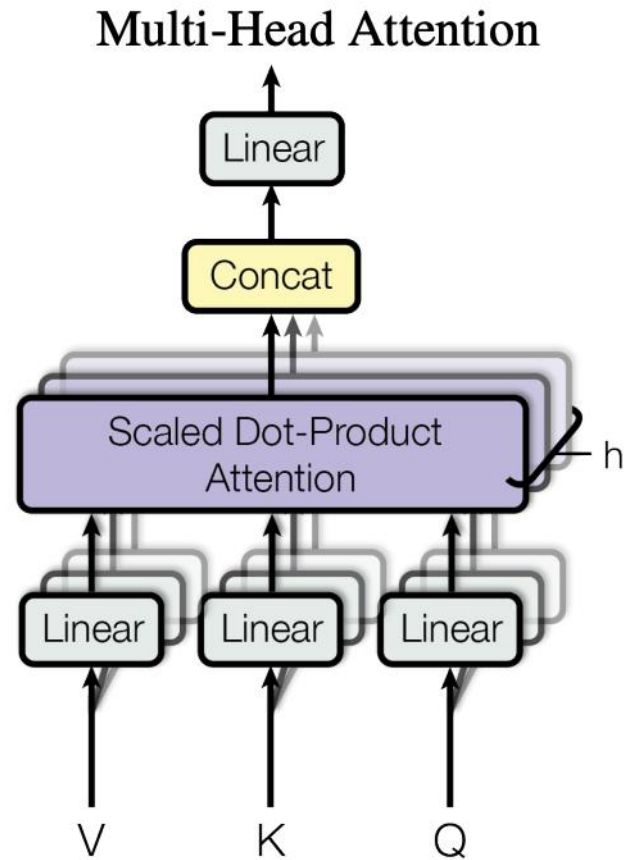


Figure 1: The Transformer - model architecture.

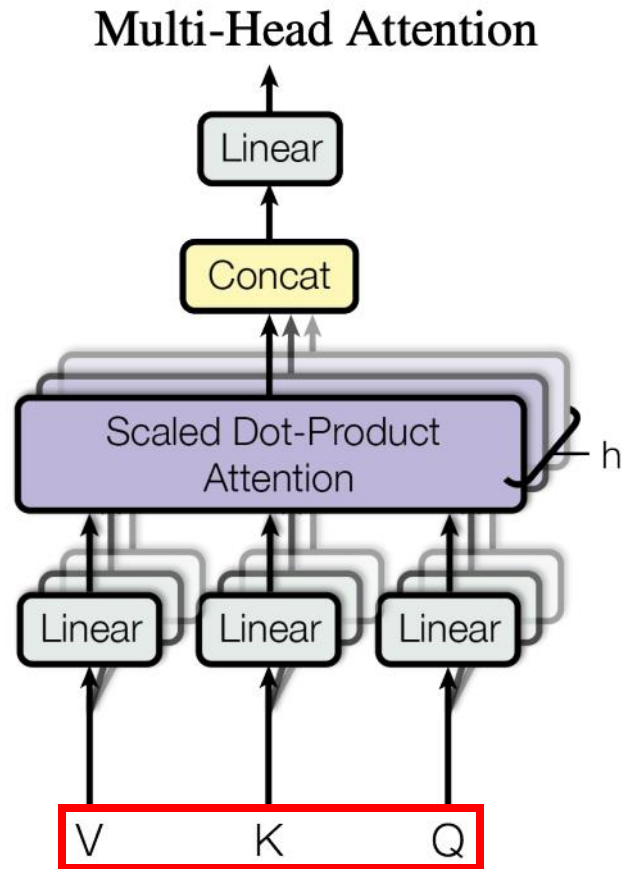
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

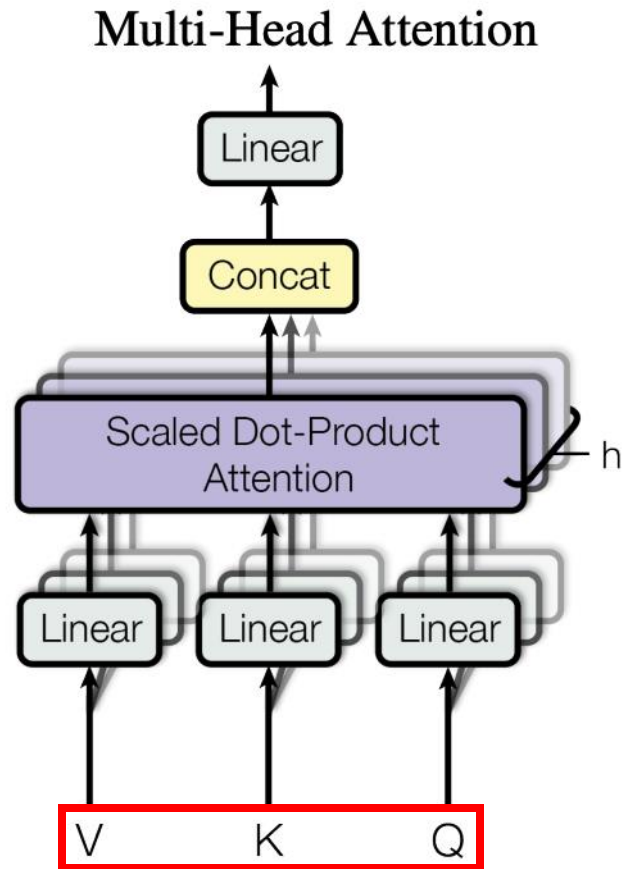
Transformer networks



V: value
K: key
Q: query

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks



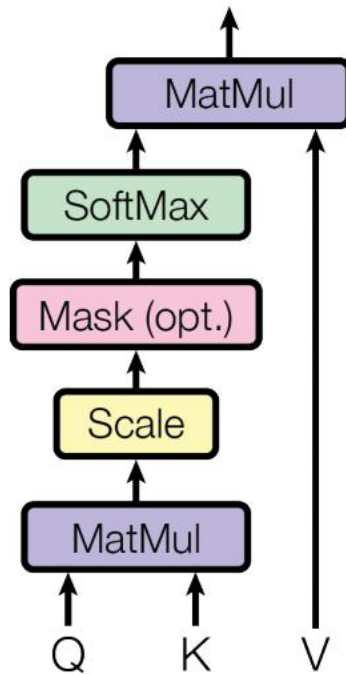
V: value
K: key
Q: query

Interpreted from information retrieval systems:
<https://stats.stackexchange.com/questions/421935/what-exactly-are-keys-queries-and-values-in-attention-mechanisms>

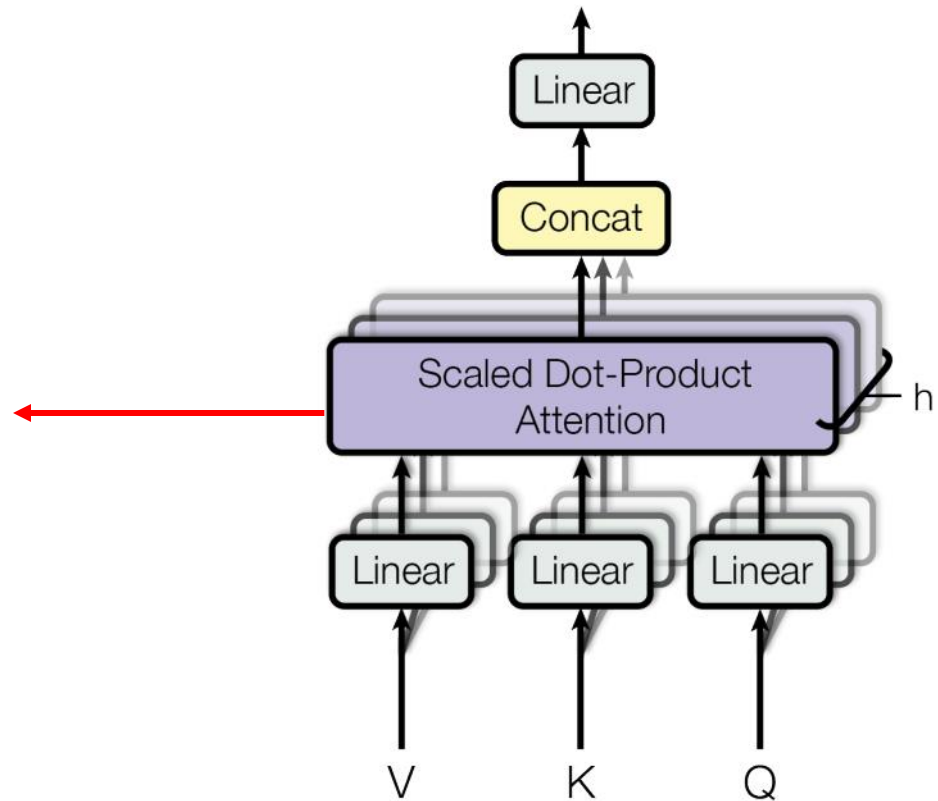
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention



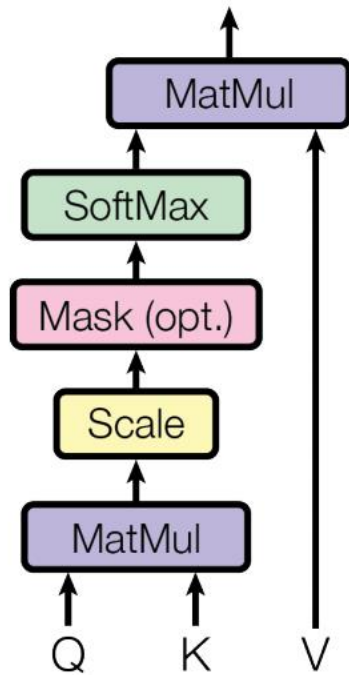
Multi-Head Attention



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

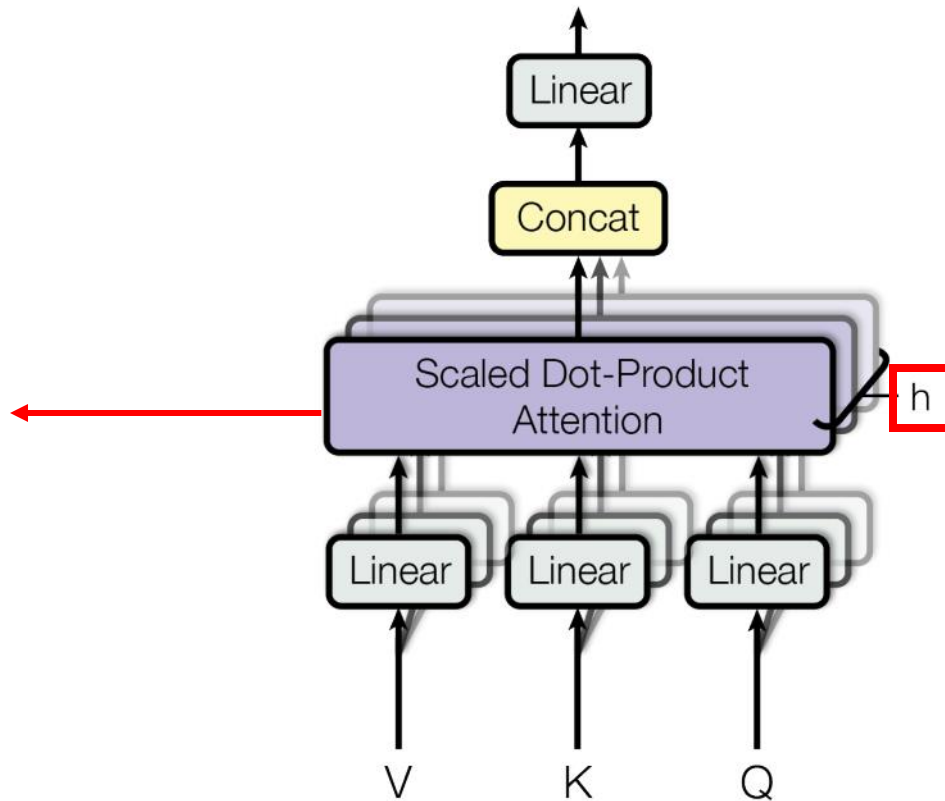
Transformer networks

Scaled Dot-Product Attention



stack h modules together

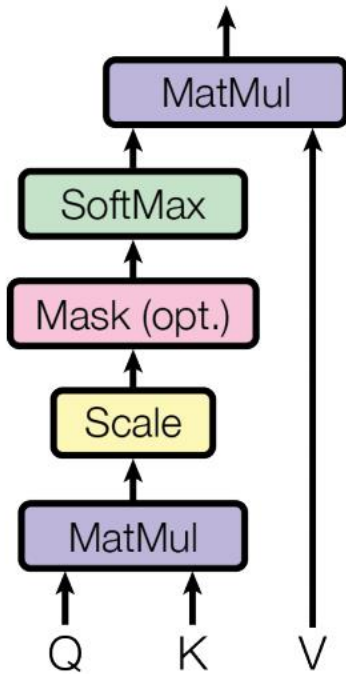
Multi-Head Attention



Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention

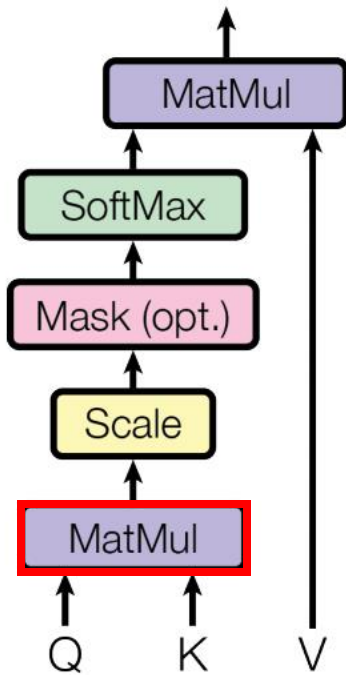


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention

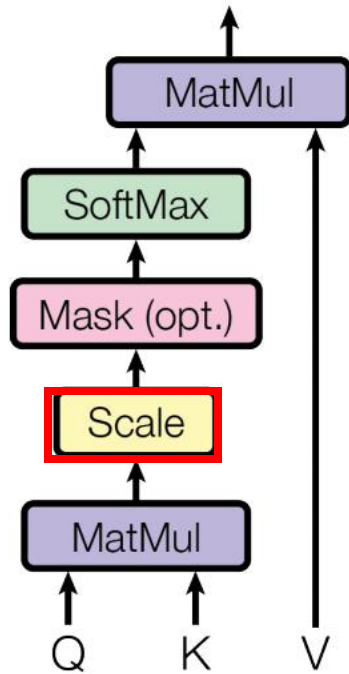


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention

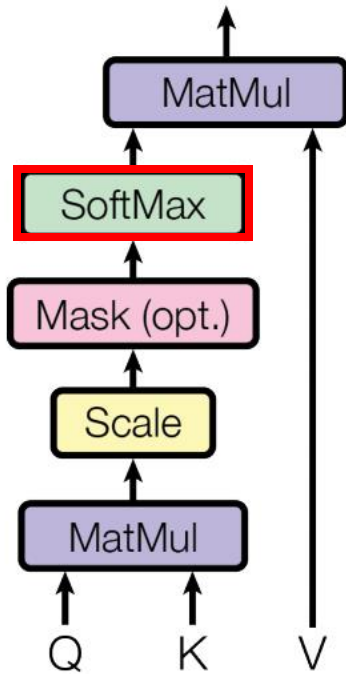


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Dimension of keys

Transformer networks

Scaled Dot-Product Attention

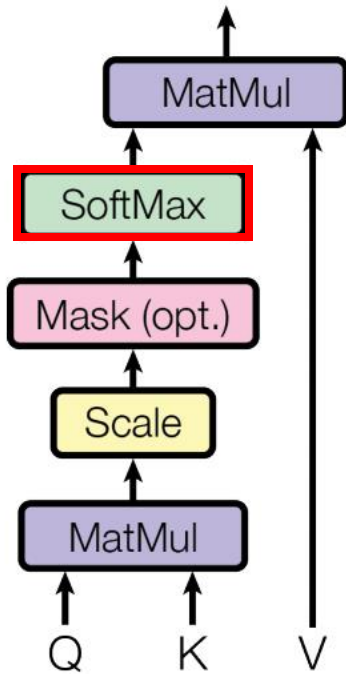


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention



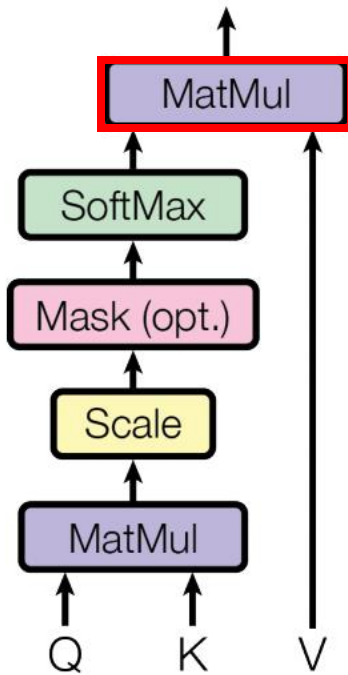
weights for selection (keys
in a retrieval system)

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

An example: one-hot format

Transformer networks

Scaled Dot-Product Attention

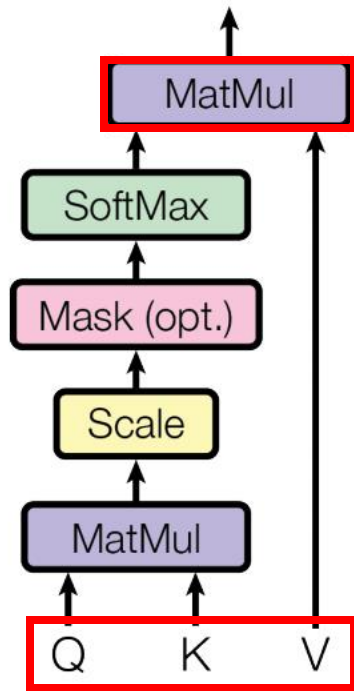


$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention



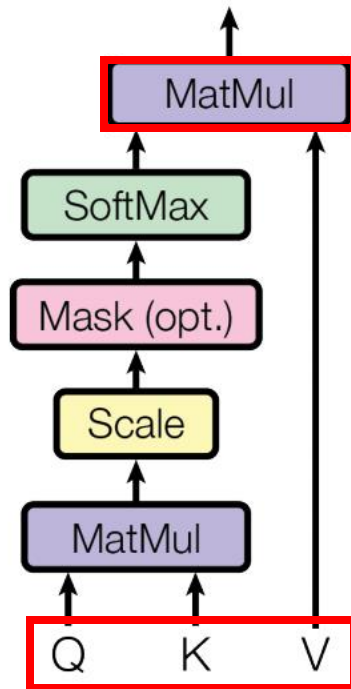
How to generate

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

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention



In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{\text{ff}} = 2048$.

Transformer networks

Scaled Dot-Product Attention

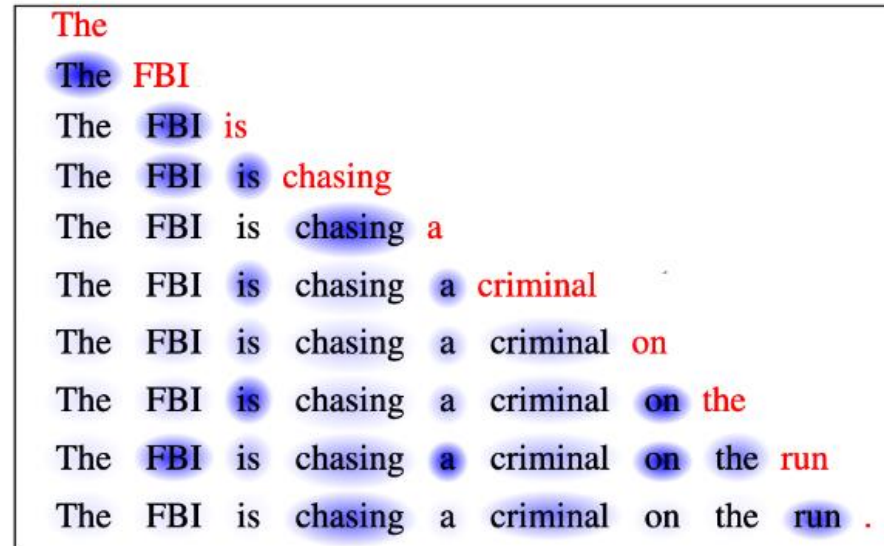
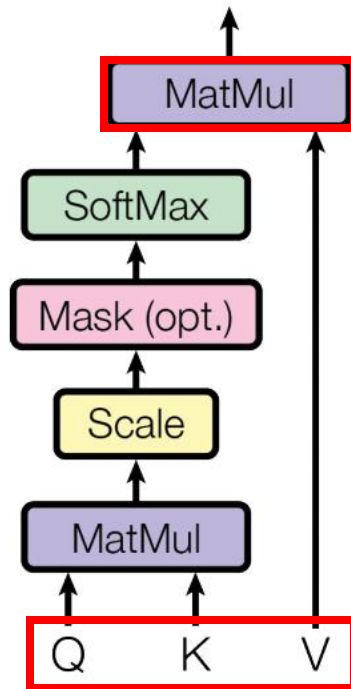
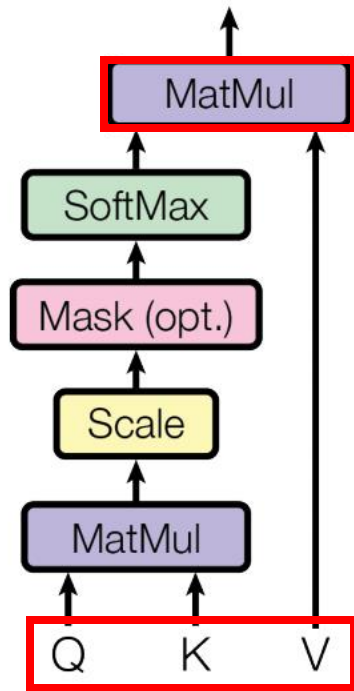


Figure is from the paper "Long Short-Term Memory-Networks for Machine Reading."

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

Scaled Dot-Product Attention



Greater weights (larger Softmax output)

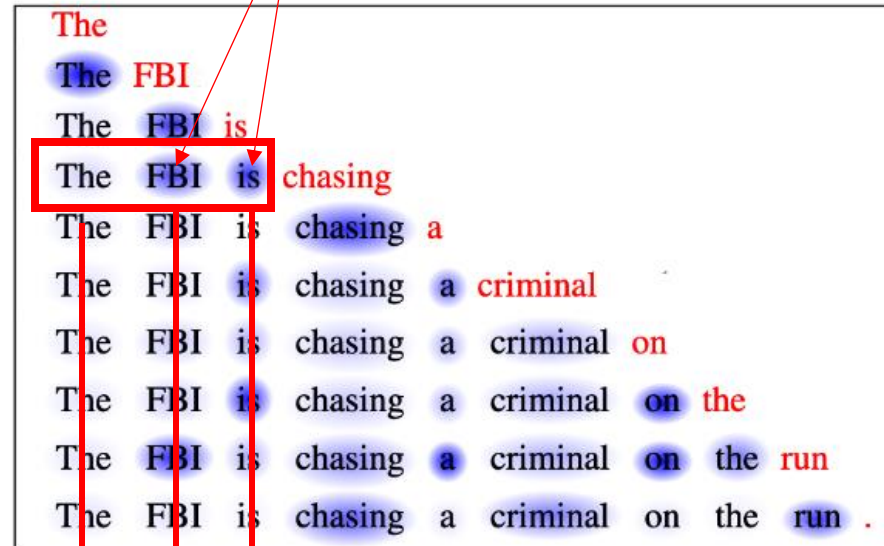


Figure is from the paper "Long Short-Term Memory-Networks for Machine Reading."



Embedding feature vectors
(a matrix)

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

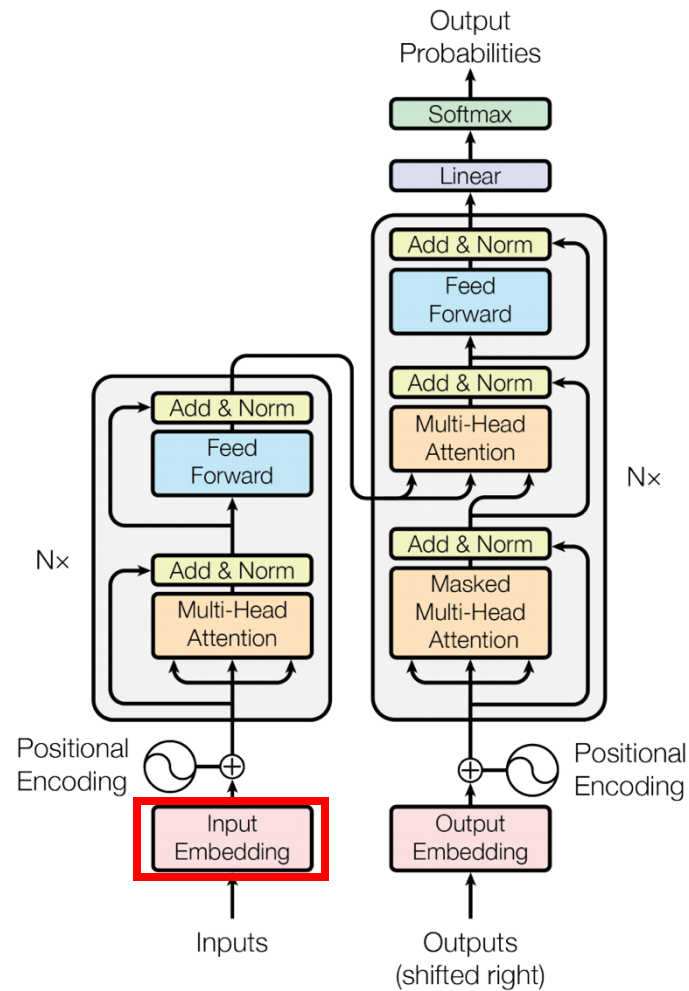


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

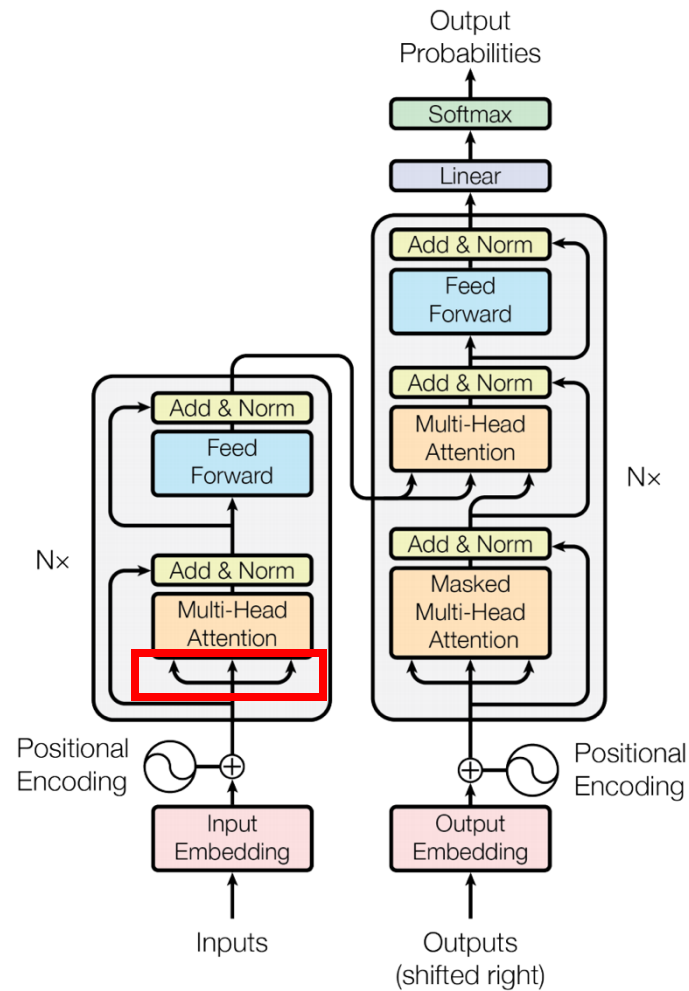


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

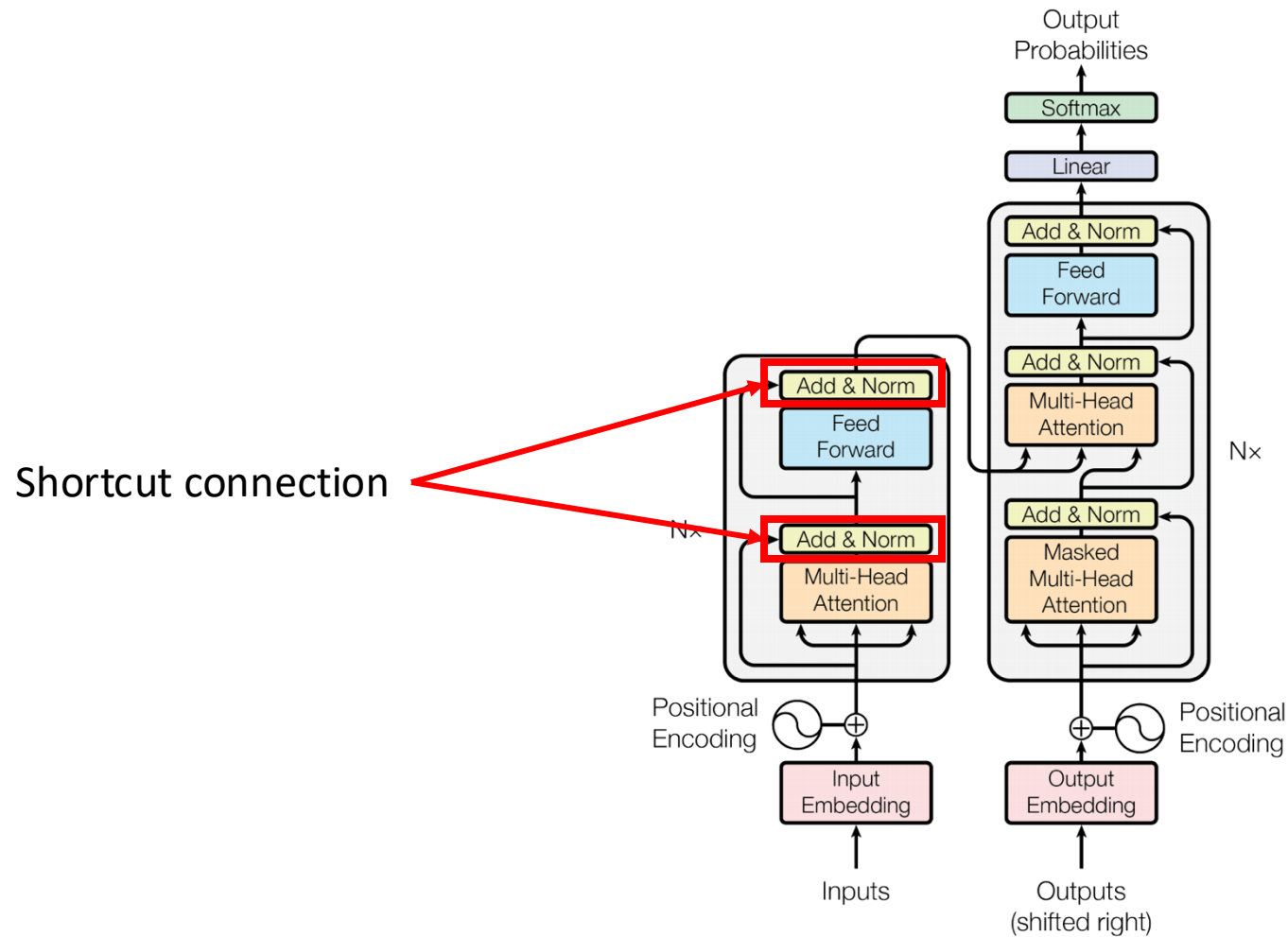


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

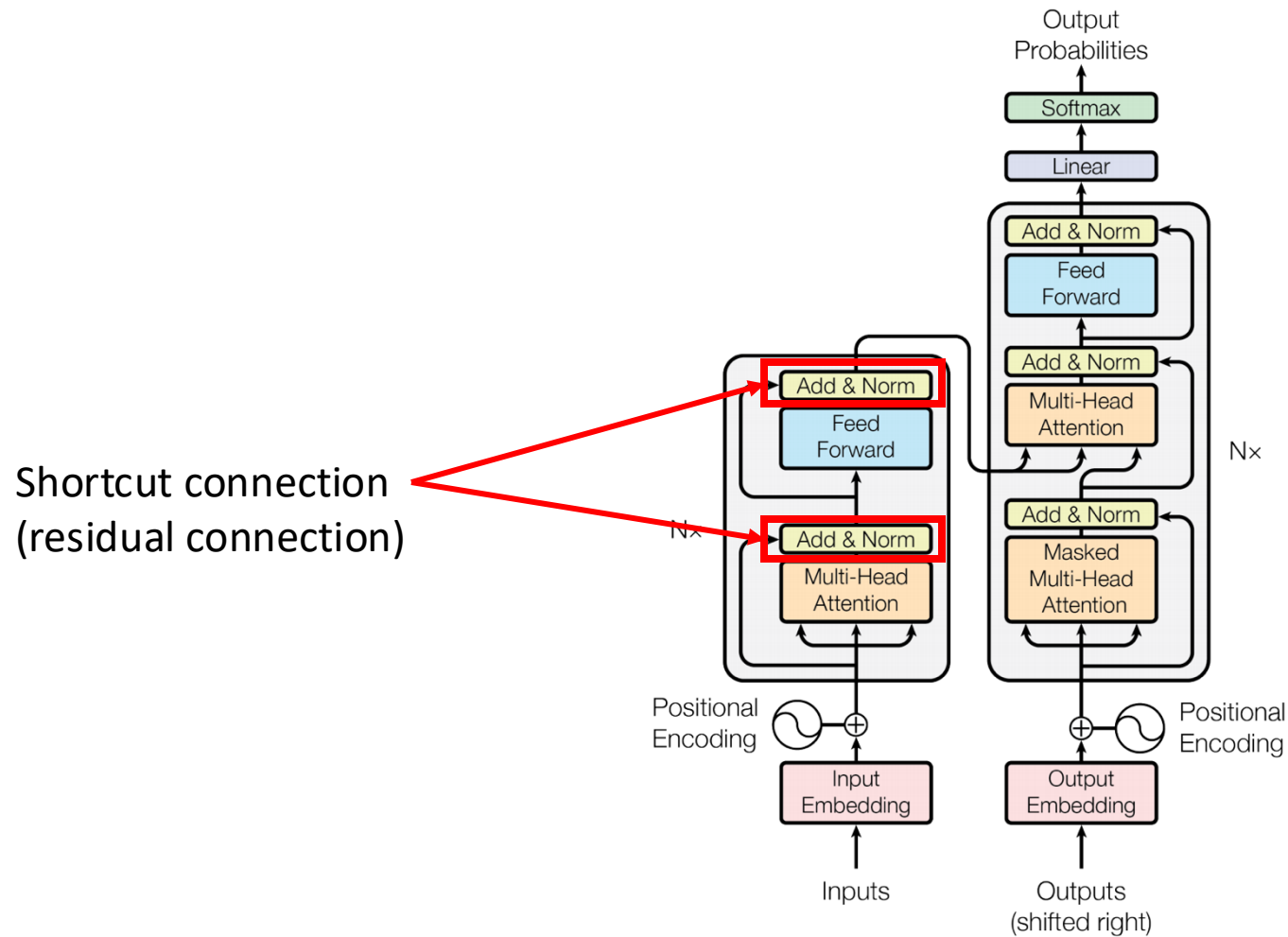


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

No recurrent structure → not RNN

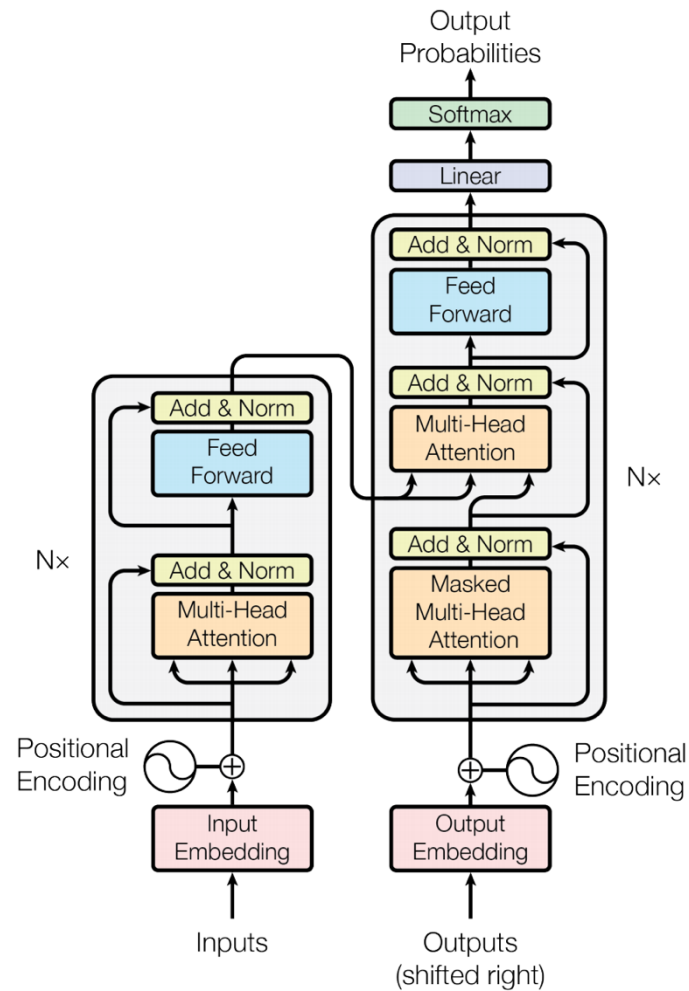


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

No recurrent structure → not RNN

Q: any benefit?

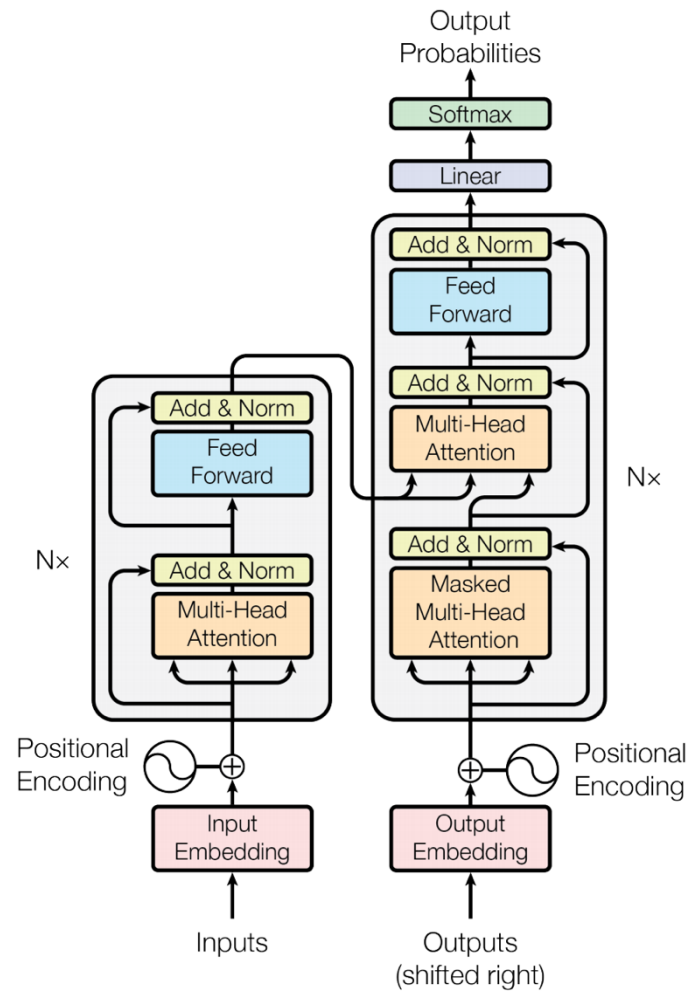


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

No recurrent structure → not RNN

Q: any benefit?

1. No sequential dependence

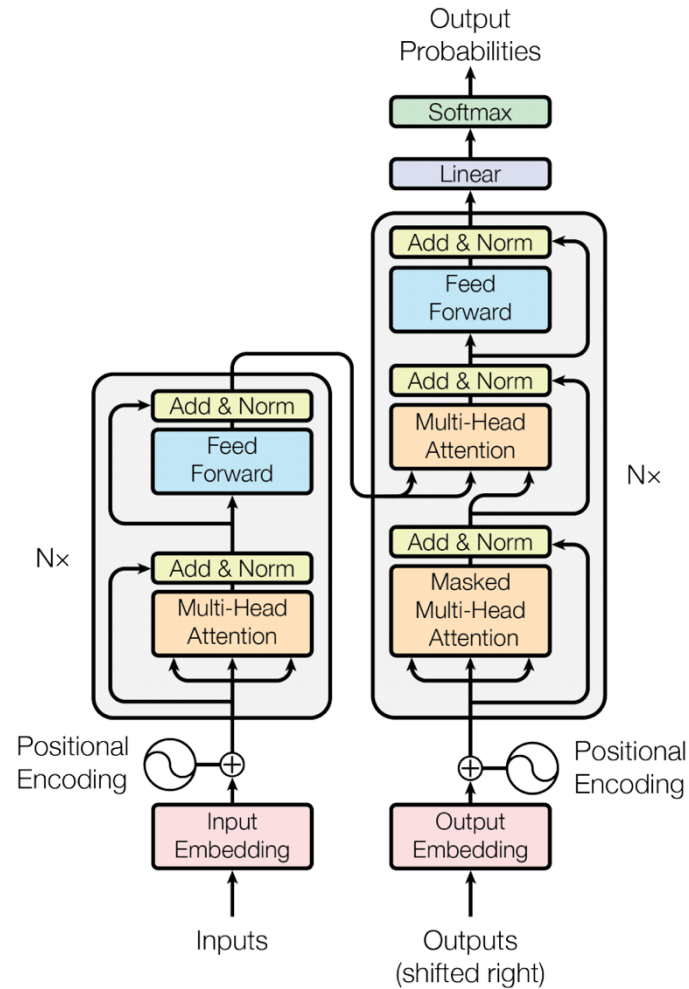


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

No recurrent structure → not RNN

Q: any benefit?

1. No sequential dependence
2. Parallel processing

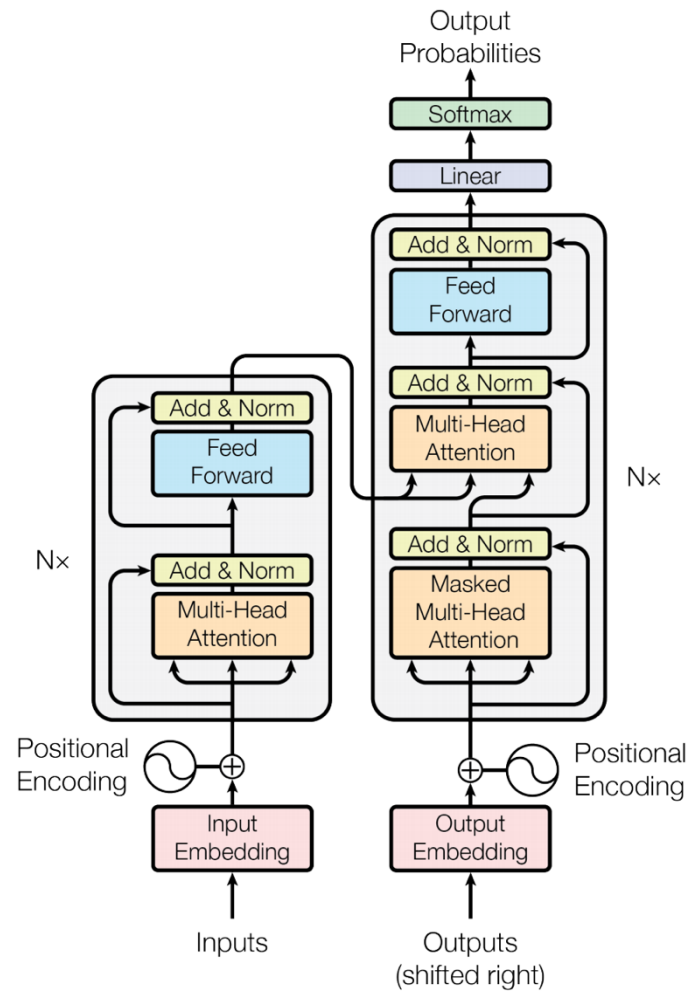


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

Transformer networks

No recurrent structure → not RNN

Q: any benefit?

1. No sequential dependence
2. Parallel processing
3. ...

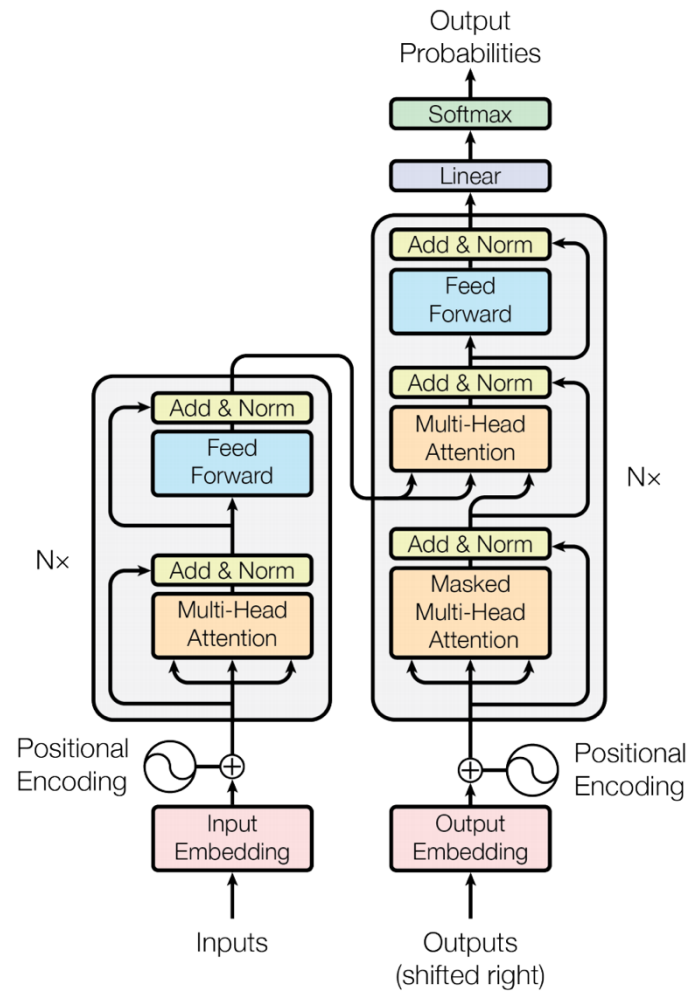
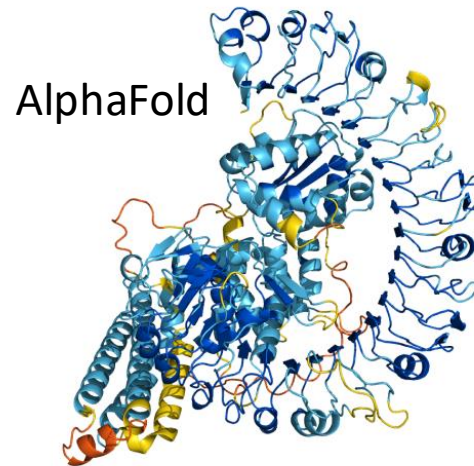
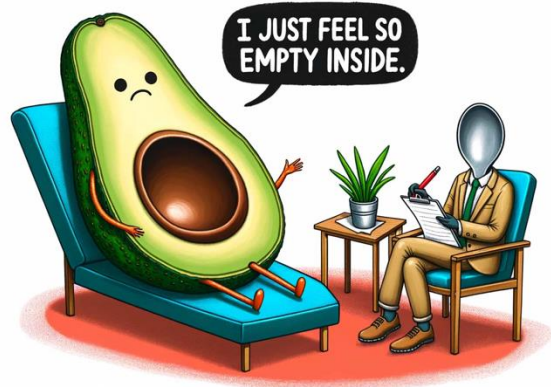


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need." *arXiv preprint arXiv:1706.03762* (2017).

AI products based on Transformers



AlphaFold

A screenshot of the GitHub Copilot web interface. At the top, it says "The world's most widely adopted AI developer tool" and "Get started with GitHub Copilot >". Below this is a chat window titled "GITHUB COPILOT: CHAT" with a user named "monalisa" asking "Write unit tests for this function". The Copilot response shows Python code for unit testing a function that parses expense strings. The code includes test cases for valid input, empty input, and various date and value formats. A small 3D head icon is visible in the top right corner of the interface.

GitHub Copilot

The world's most widely adopted AI developer tool

Get started with GitHub Copilot >

GITHUB COPILOT: CHAT

monalisa

Write unit tests for this function

GitHub Copilot

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK\n2023-01-03 -2.72 EUR'
        expected_output = [(datetime.datetime(2023, 1, 2), -34.01, 'USD'), (datetime.datetime(2023, 1, 3), 2.59, 'DKK'), (datetime.datetime(2023, 1, 3), -2.72, 'EUR')]
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

```
import datetime

def parse_expenses (expenses_string):
    """Parse the list of expenses and return the list of triples (date, amount, currency)
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
    2023-01-02 -34.01 USD
    2023-01-03 2.59 DKK
    2023-01-03 -2.72 EUR
    """
    expenses = []

    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"), float(value), currency))

    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

Ask a question or type "/" for commands