

Outline

- Issues with RNNs
- Comparison with Transformers



Neural Machine Translation



wie

sind

sie

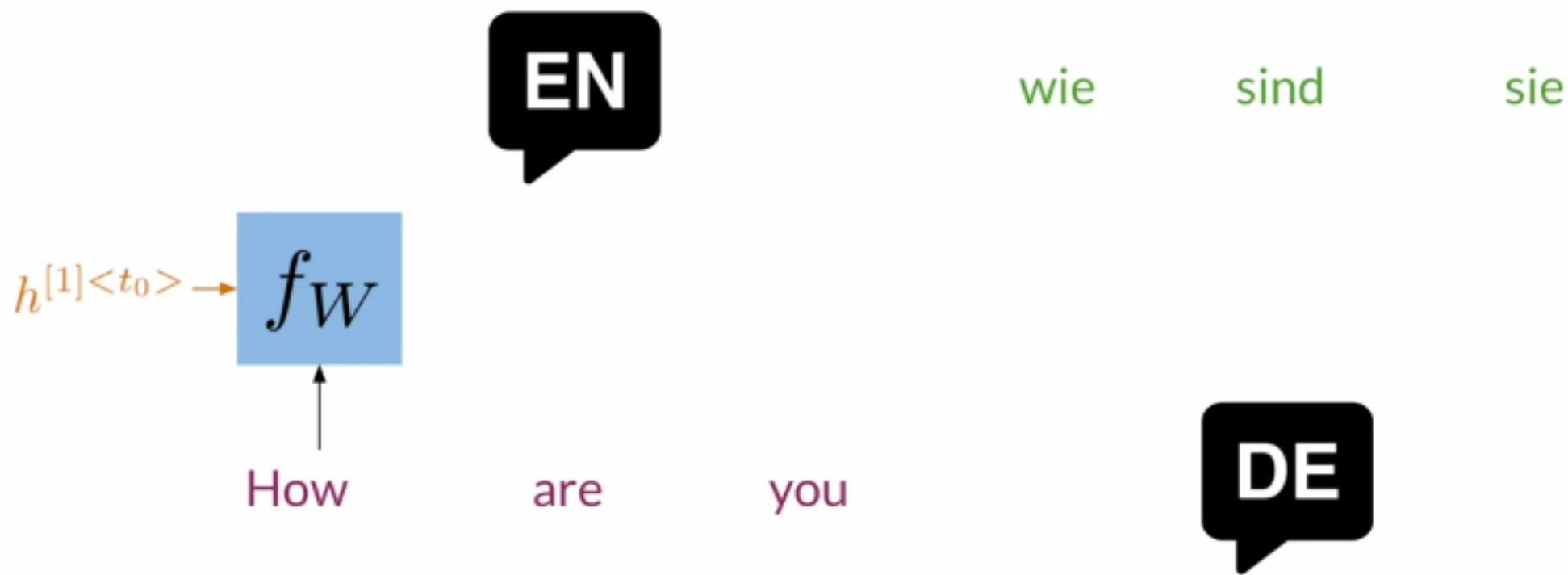
How

are

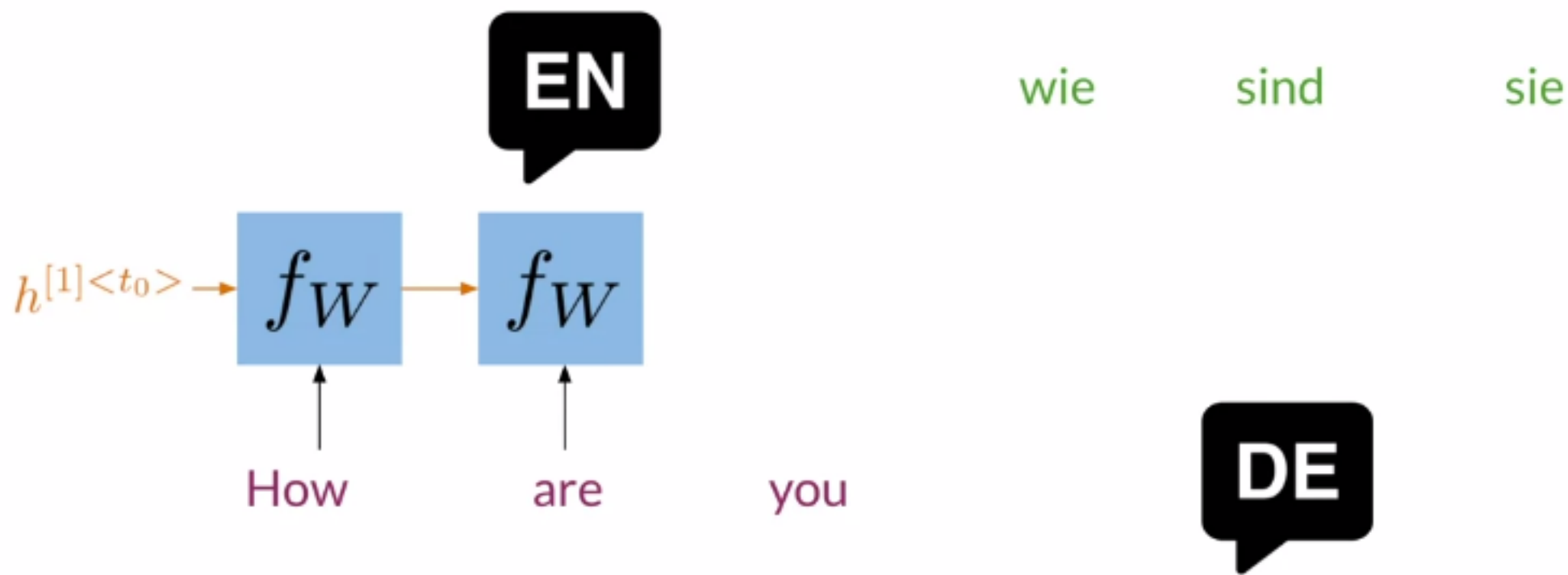
you



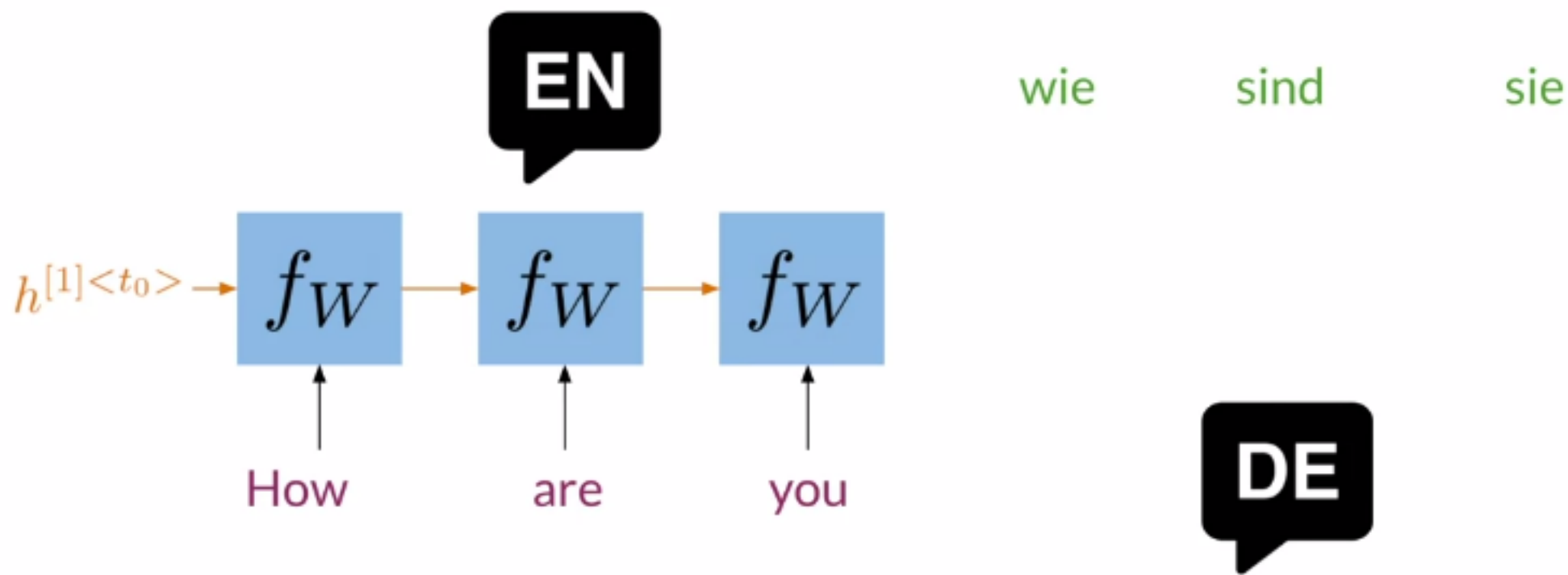
Neural Machine Translation



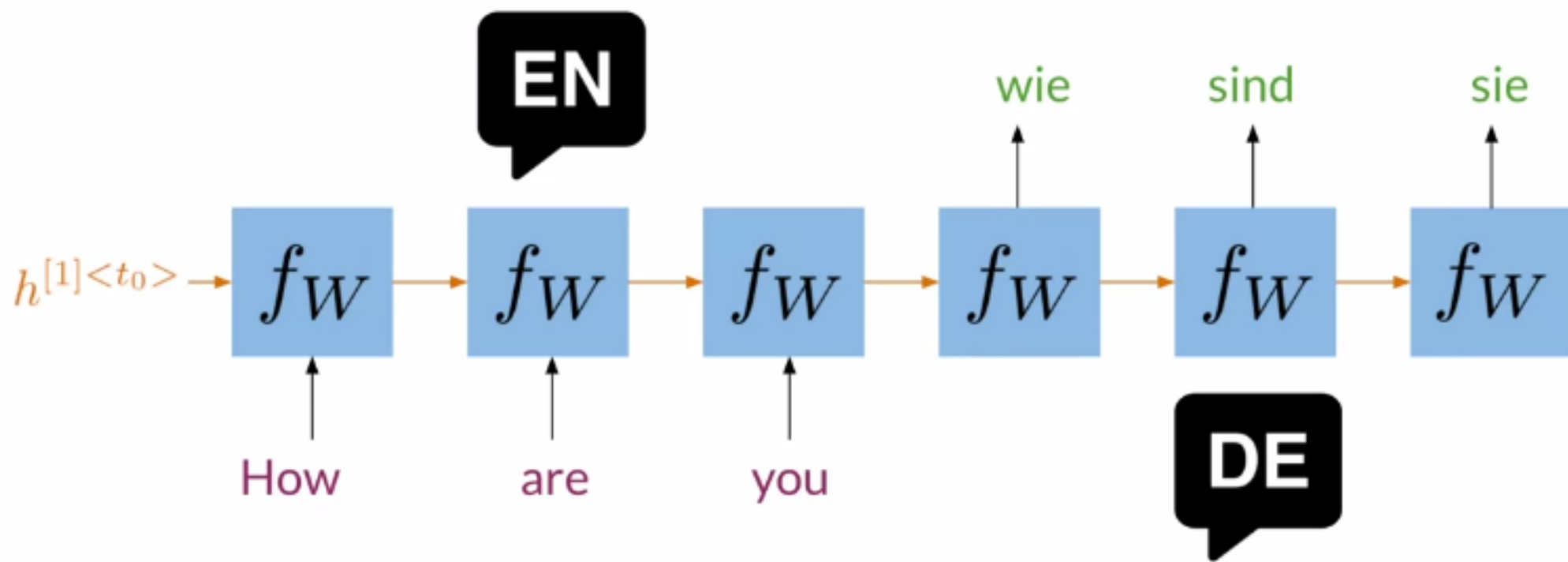
Neural Machine Translation



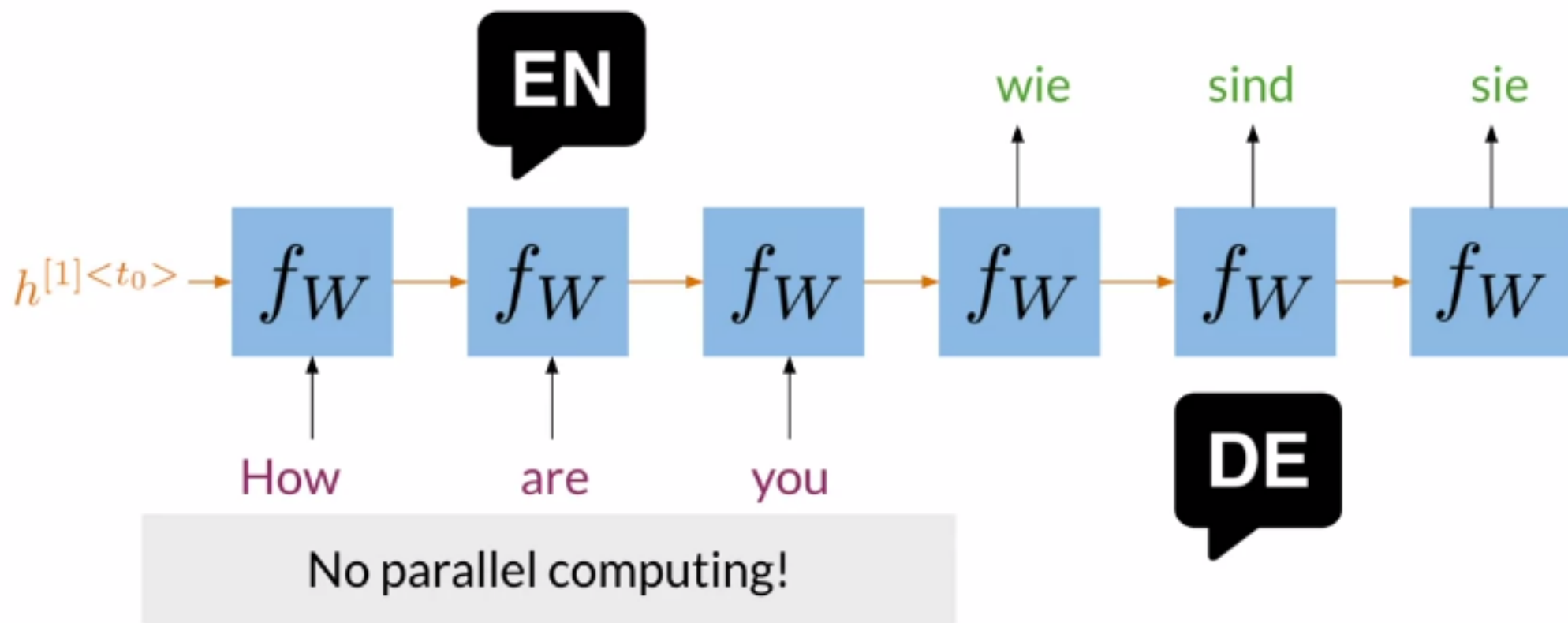
Neural Machine Translation



Neural Machine Translation

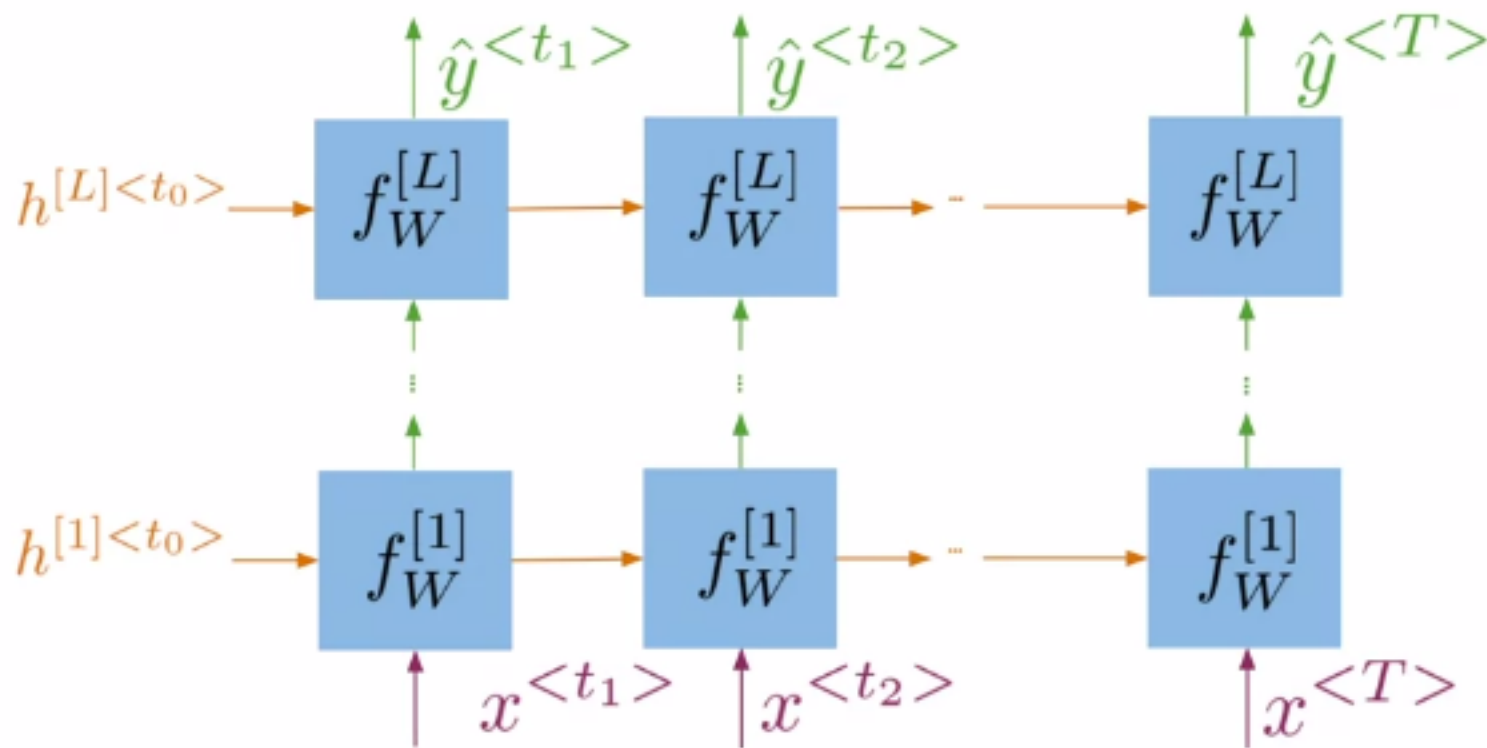


Neural Machine Translation

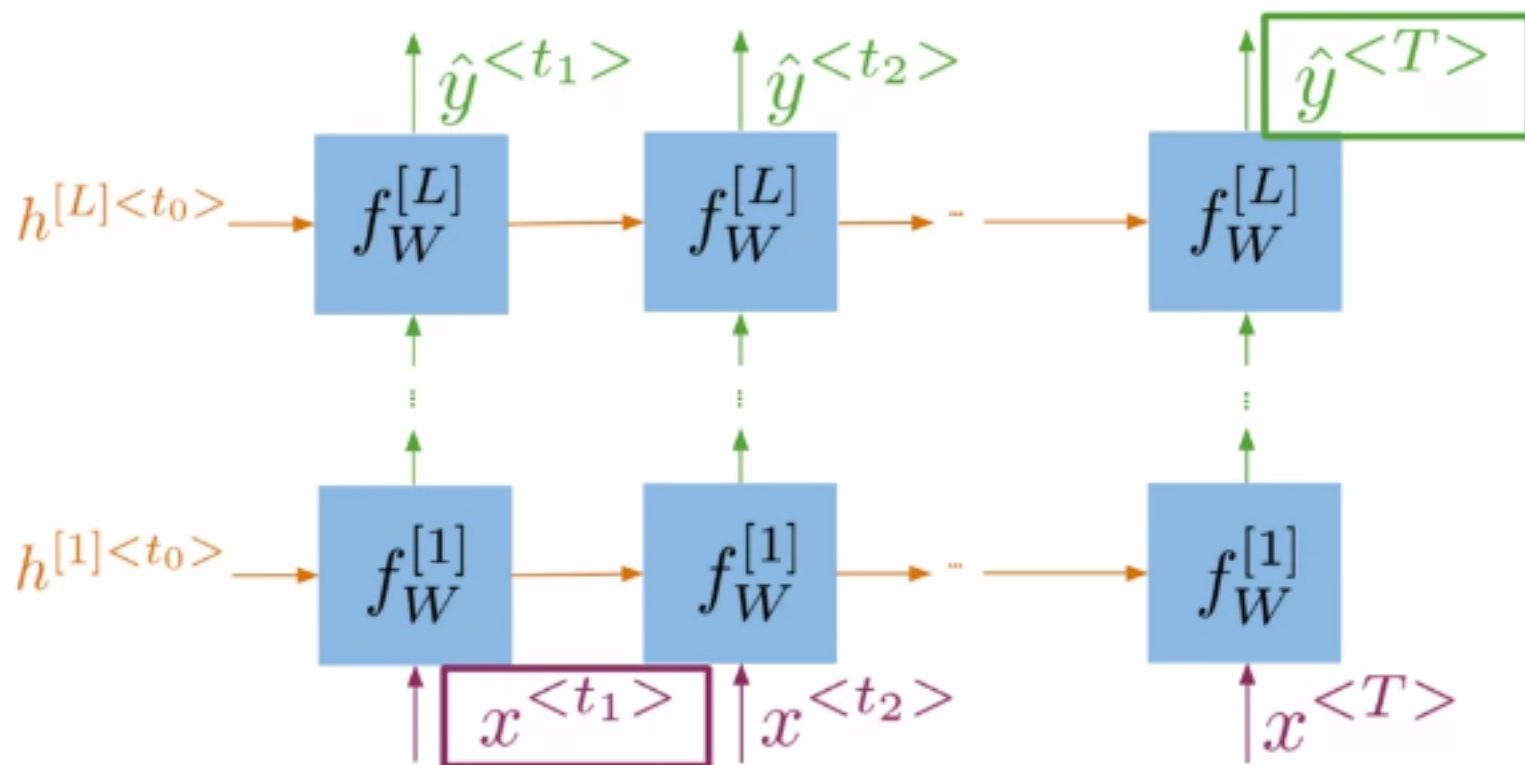


Seq2Seq Architectures

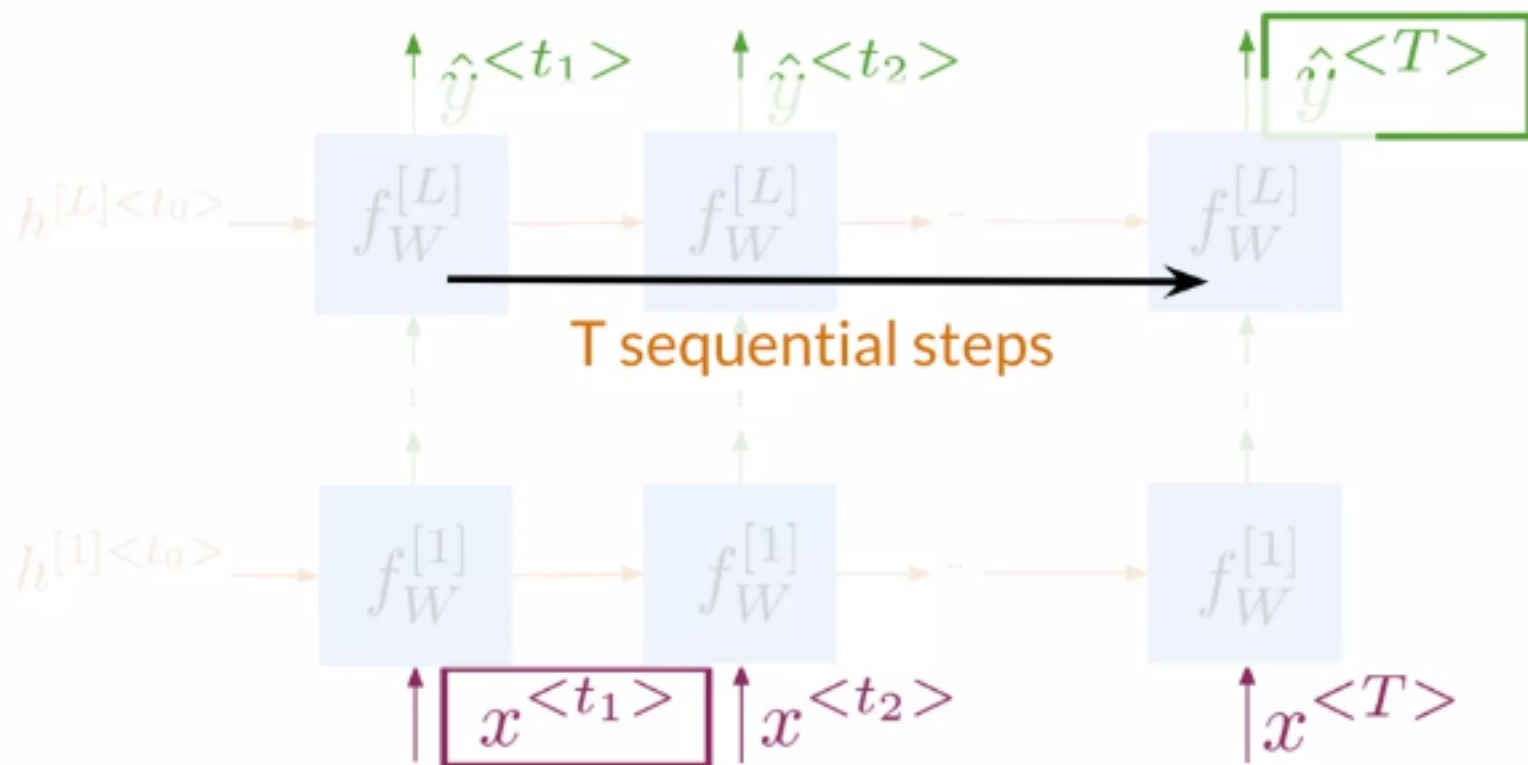
Seq2Seq Architectures



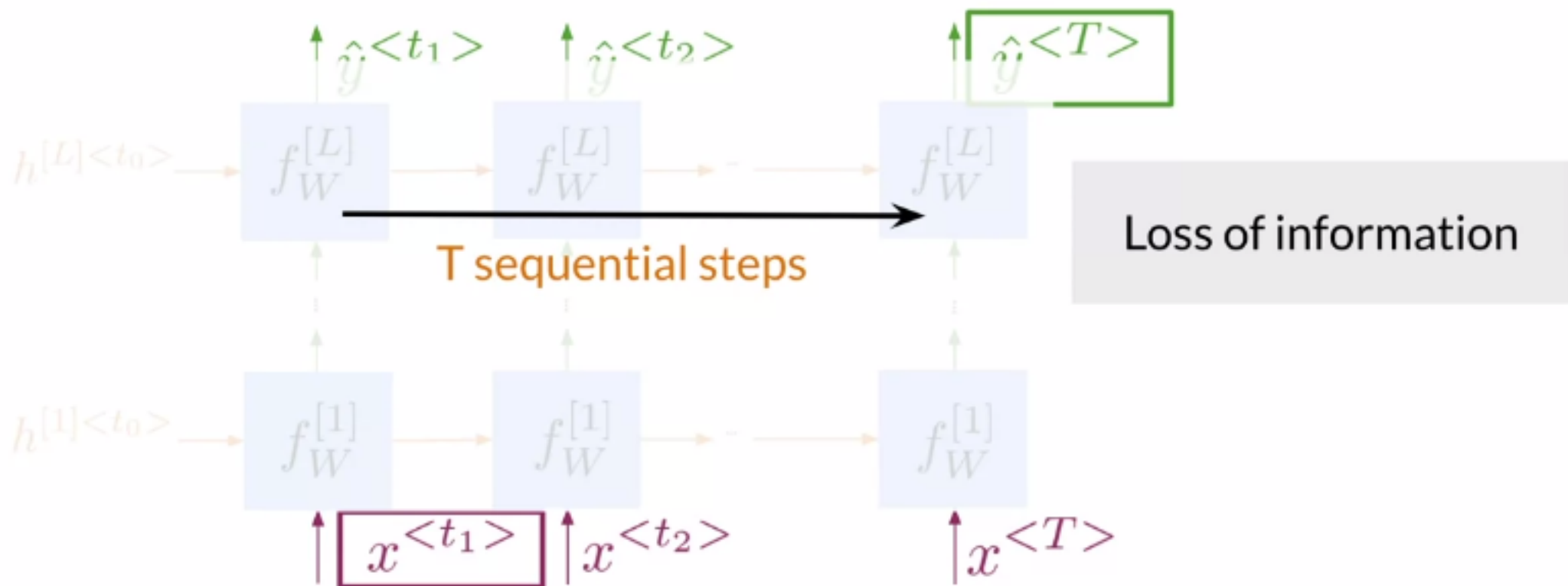
Seq2Seq Architectures



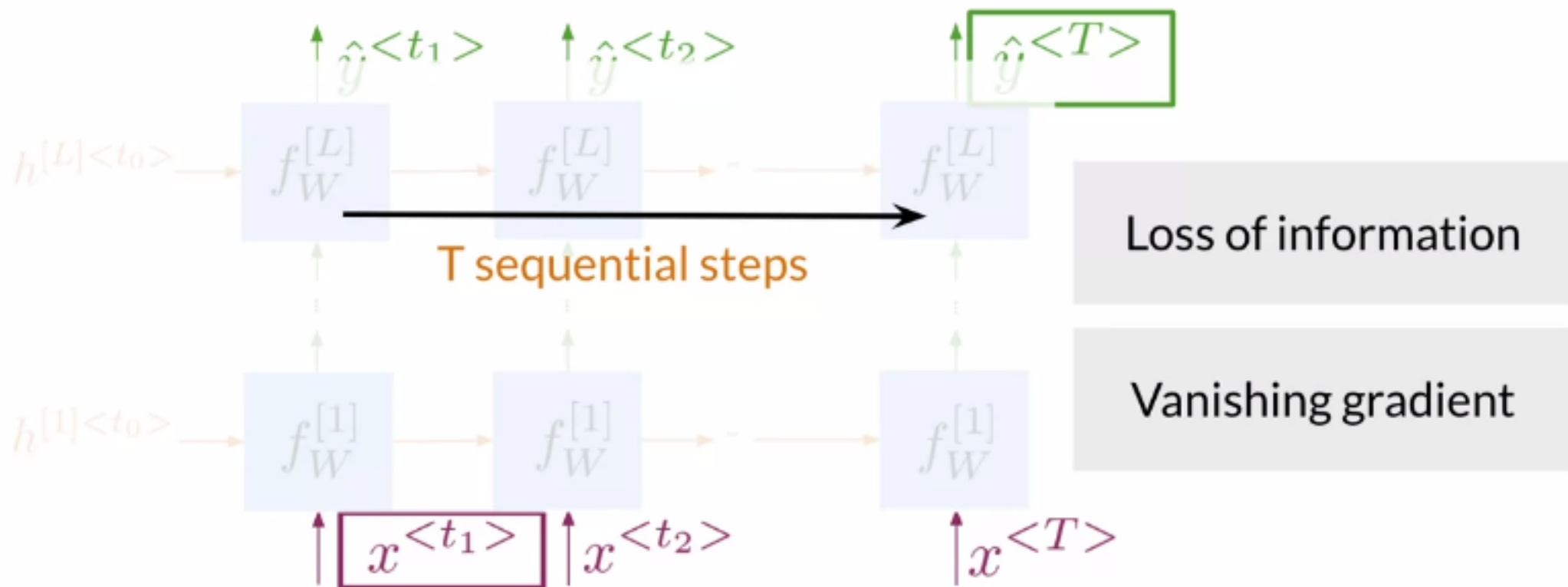
Seq2Seq Architectures



Seq2Seq Architectures

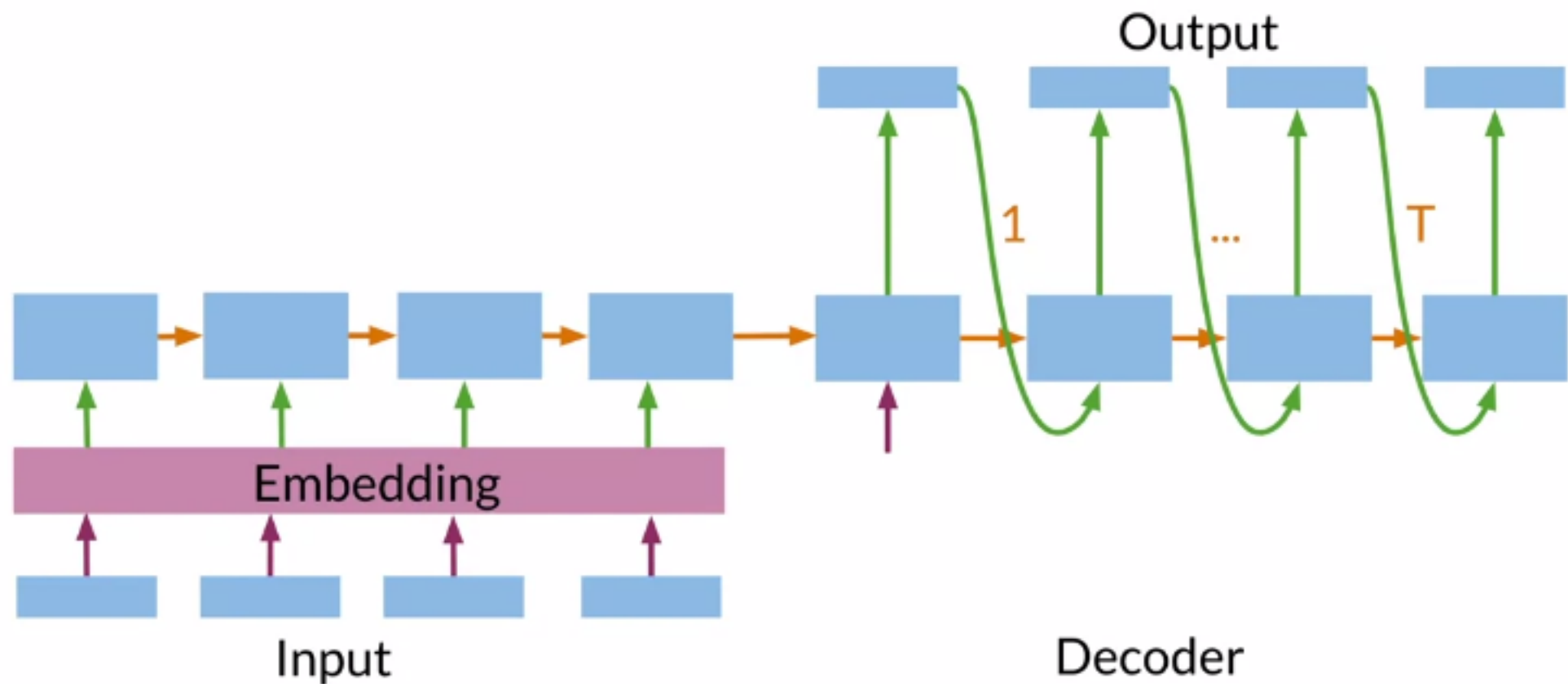


Seq2Seq Architectures

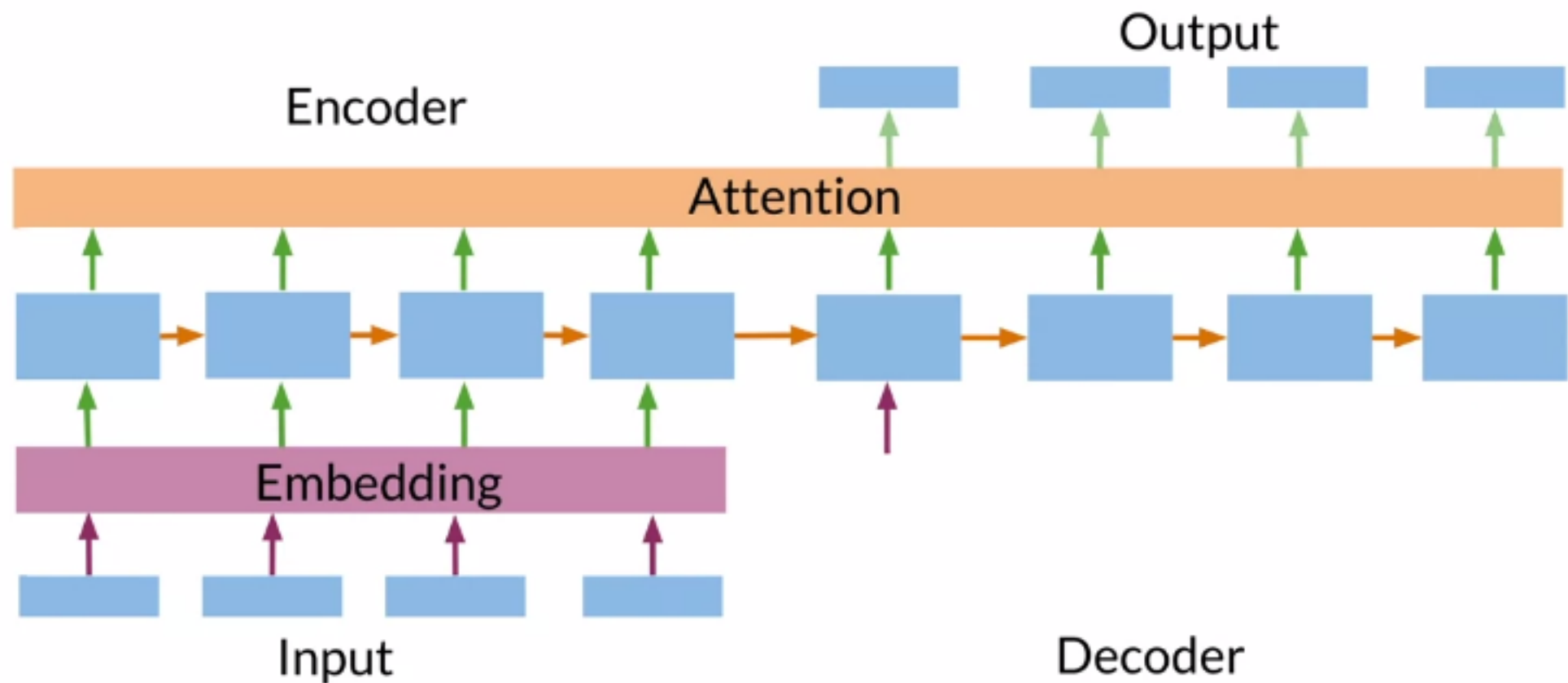


RNNs vs Transformer: Encoder-Decoder

RNNs vs Transformer: Encoder-Decoder

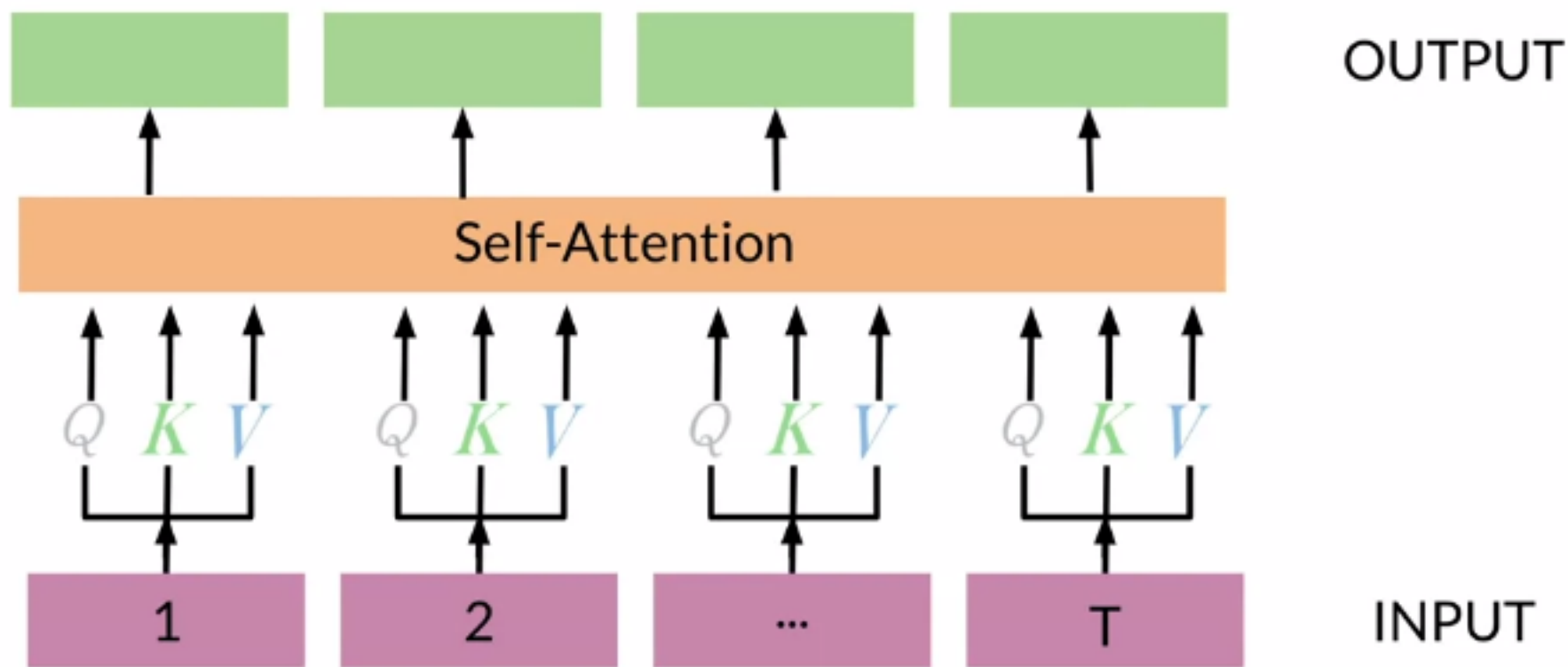


RNNs vs Transformer: Encoder-Decoder

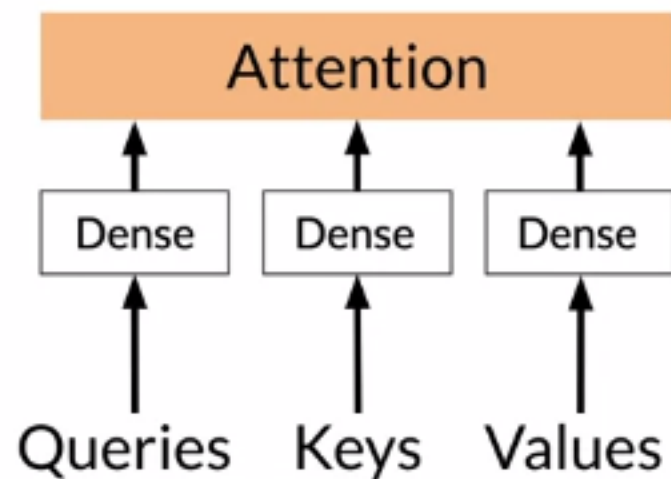


RNNs vs Transformer: Multi-headed attention

RNNs vs Transformer: Multi-headed attention



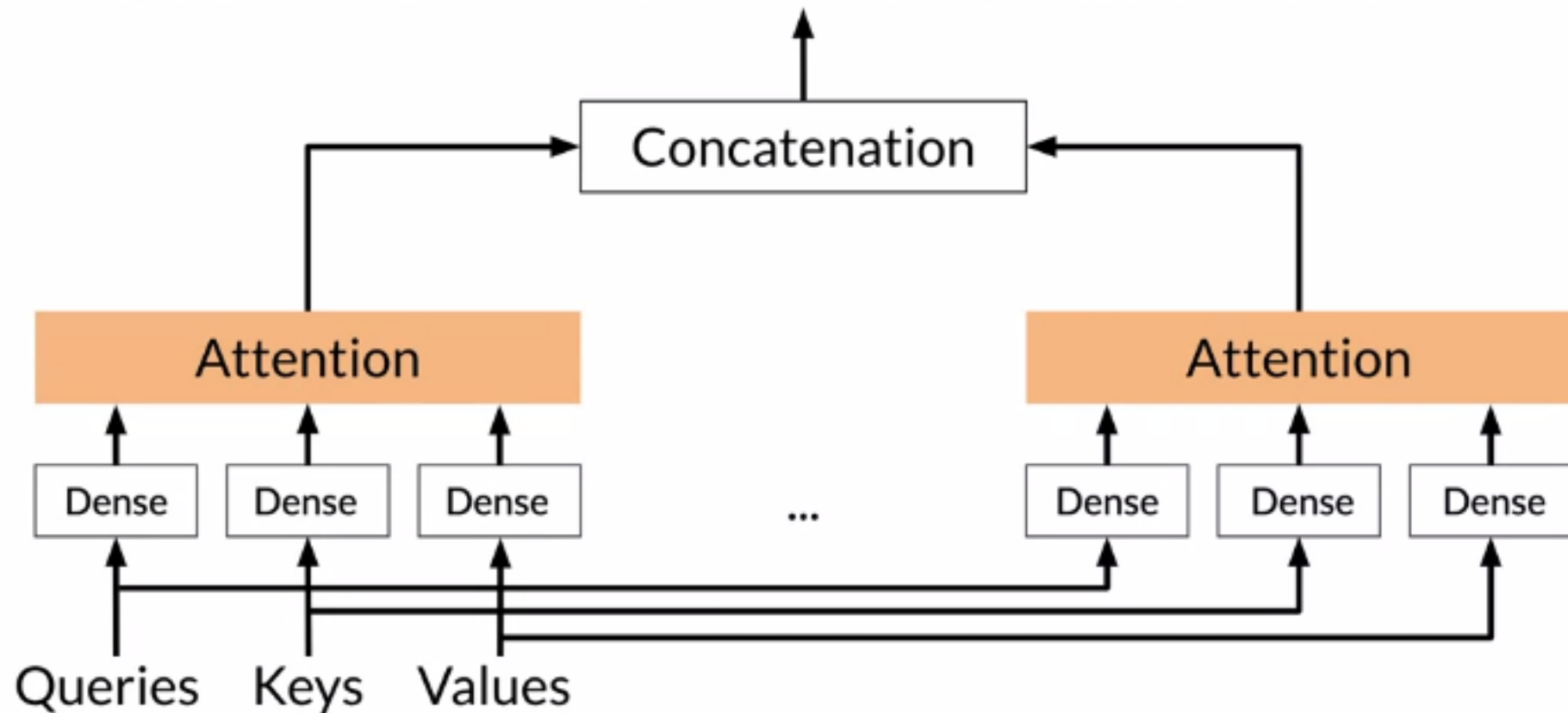
RNNs vs Transformer: Multi-headed attention



RNNs vs Transformer: Multi-headed attention



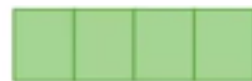
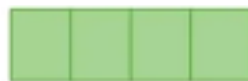
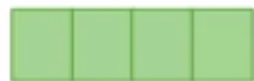
RNNs vs Transformer: Multi-headed attention



RNNs vs Transformer: Positional Encoding

RNNs vs Transformer: Positional Encoding

EMBEDDINGS



INPUT

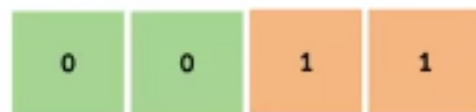
Ich

bin

glücklich

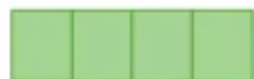
RNNs vs Transformer: Positional Encoding

POSITIONAL
ENCODING



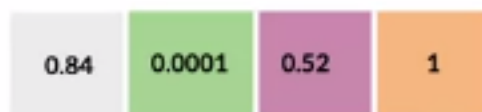
+

EMBEDDINGS

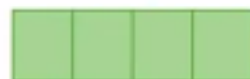


INPUT

Ich



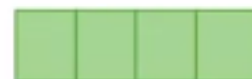
+



bin



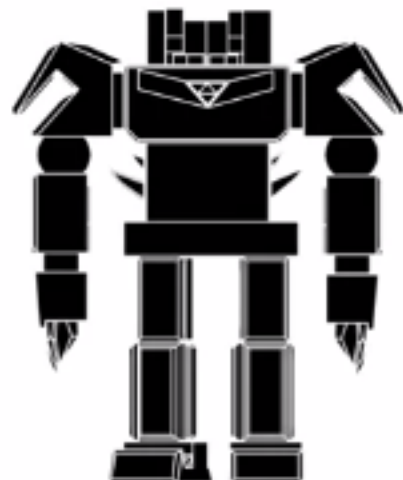
+



glücklich

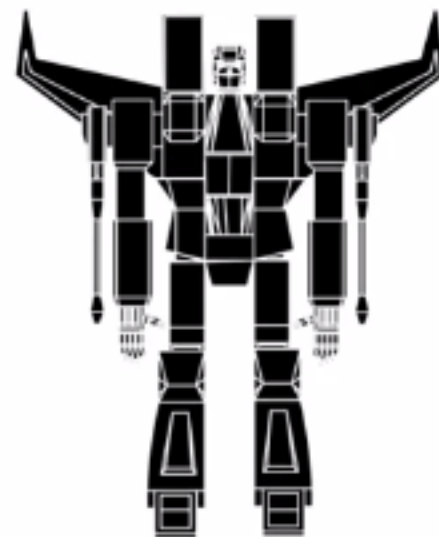
Summary

- In RNNs parallel computing is difficult to implement
- For long sequences in RNNs there is loss of information
- In RNNs there is the problem of vanishing gradient
- Transformers help with all of the above



Outline

- Transformers applications in NLP
- Some Transformers
- Introduction to T5



Transformer NLP applications

Transformer NLP applications

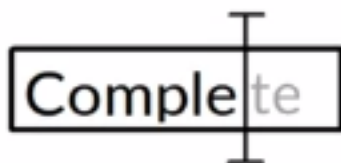


Text
summarization

Transformer NLP applications



Text
summarization

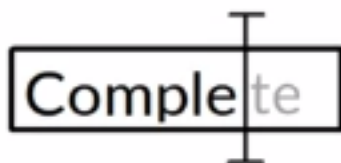


Auto-Complete

Transformer NLP applications



Text
summarization



Auto-Complete

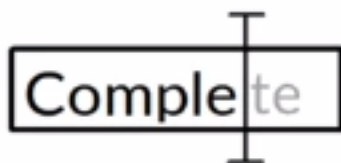


Named entity
recognition (NER)

Transformer NLP applications



Text
summarization



Auto-Complete



Named entity
recognition (NER)



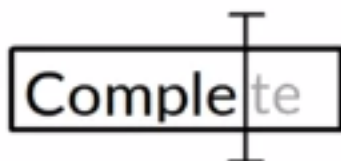
Question
answering (Q&A)

Transformer NLP applications



Text
summarization

Translation



Auto-Complete



Named entity
recognition (NER)



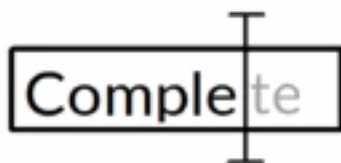
Question
answering (Q&A)

Transformer NLP applications



Text
summarization

Translation



Auto-Complete

Chat-bots



The	wind	blows	hard
Article	Noun	Verb	Adjective

Named entity
recognition (NER)

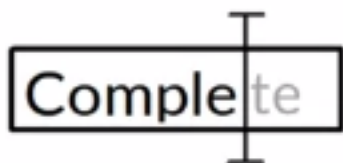


Question
answering (Q&A)

Transformer NLP applications



Text
summarization



Auto-Complete



Named entity
recognition (NER)



Question
answering (Q&A)

Translation



Chat-bots



Other NLP tasks

Sentiment Analysis
Market Intelligence
Text Classification
Character Recognition
Spell Checking

State of the Art Transformers

Radford, A., et al. (2018)
Open AI

GPT-2: Generative Pre-training for
Transformer

State of the Art Transformers

Radford, A., et al. (2018)
Open AI

Devlin, J., et al. (2018)
Google AI Language

GPT-2: Generative Pre-training for
Transformer

BERT: Bidirectional Encoder
Representations from Transformers

State of the Art Transformers

Radford, A., et al. (2018)
Open AI

Devlin, J., et al. (2018)
Google AI Language

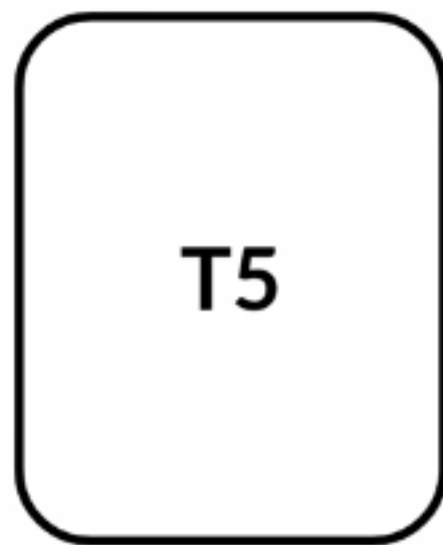
Colin, R., et al. (2019)
Google

GPT-2: Generative Pre-training for
Transformer

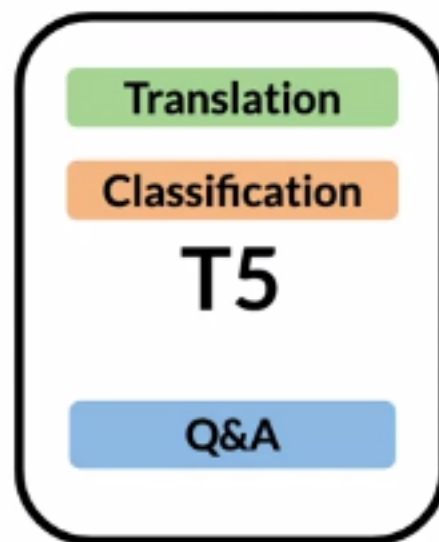
BERT: Bidirectional Encoder
Representations from Transformers

T5: Text-to-text transfer transformer

T5: Text-To-Text Transfer Transformer

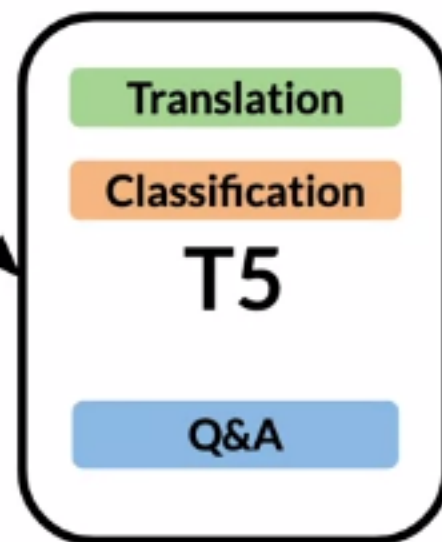


T5: Text-To-Text Transfer Transformer

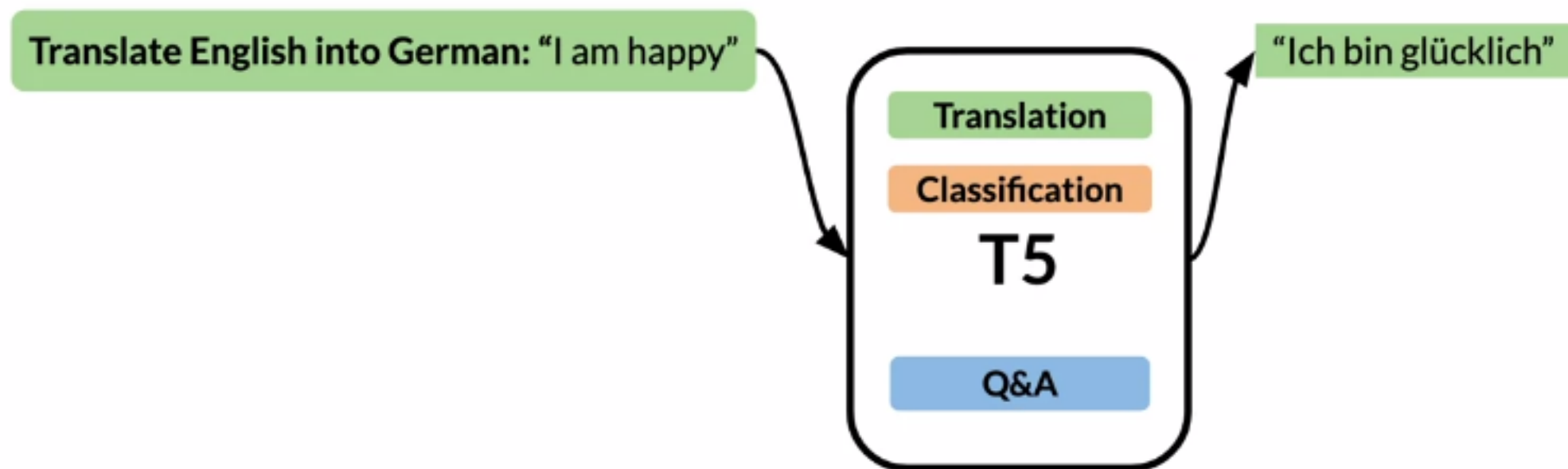


T5: Text-To-Text Transfer Transformer

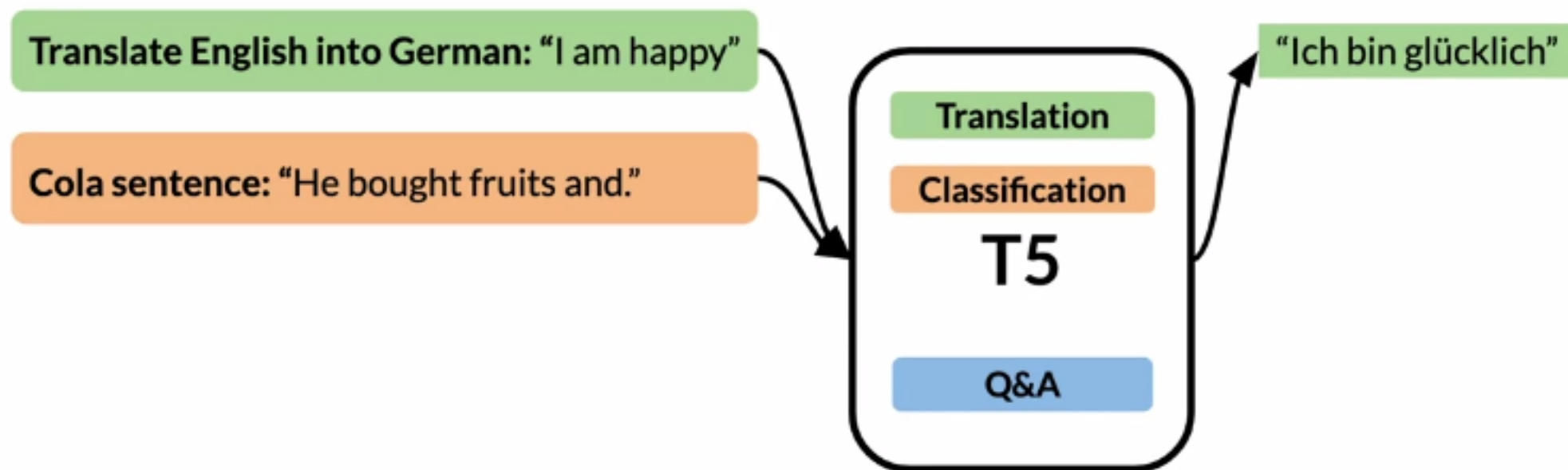
Translate English into German: "I am happy"



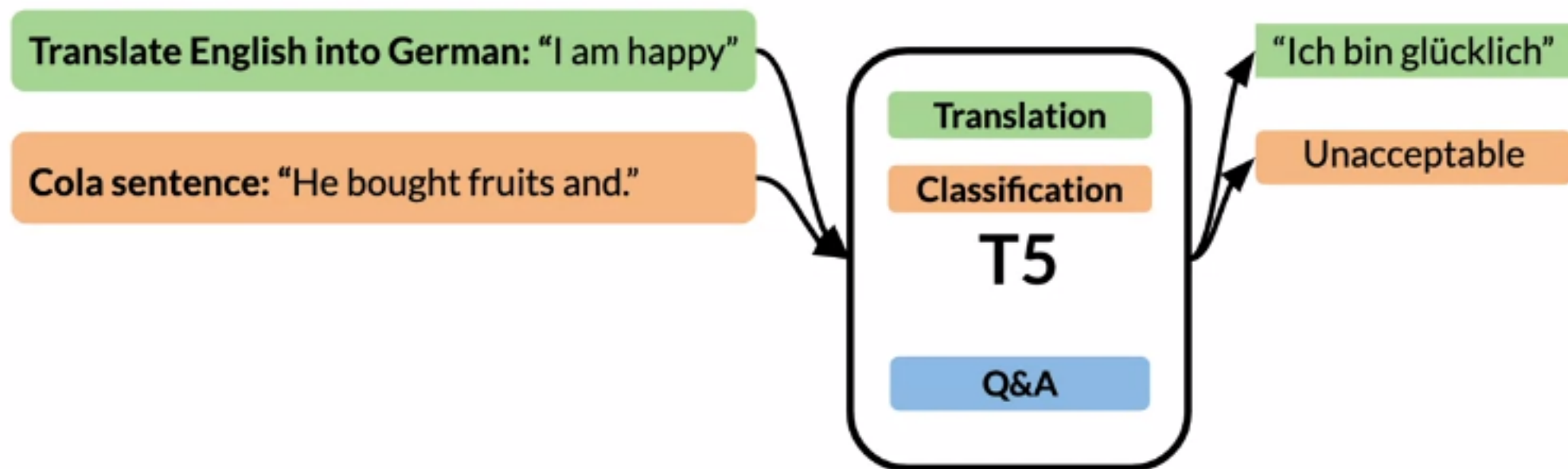
T5: Text-To-Text Transfer Transformer



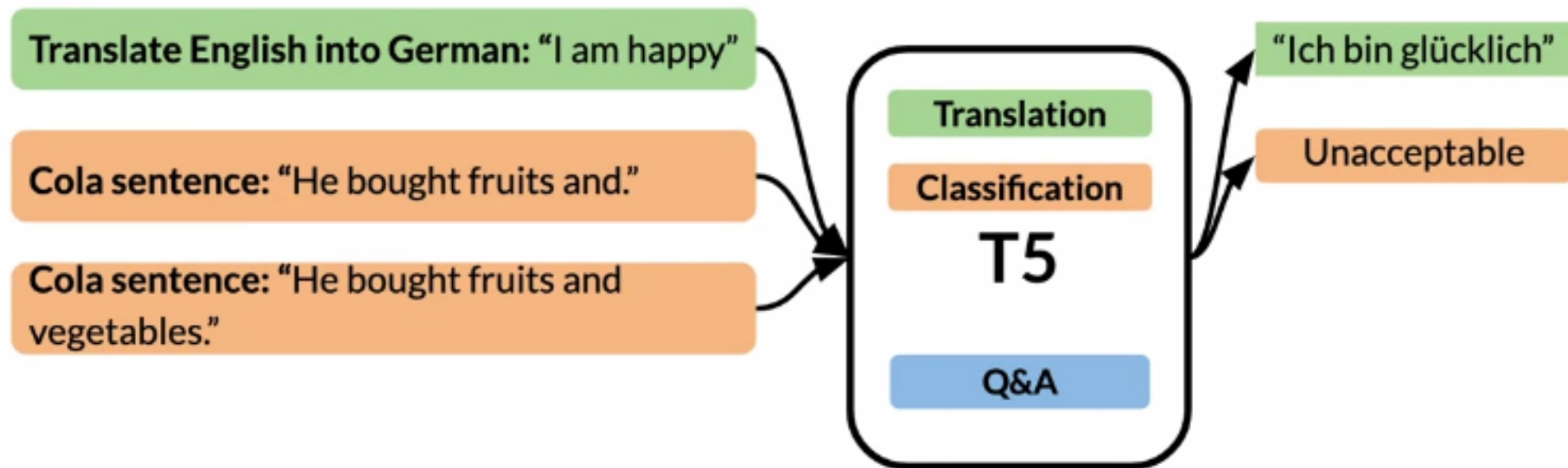
T5: Text-To-Text Transfer Transformer



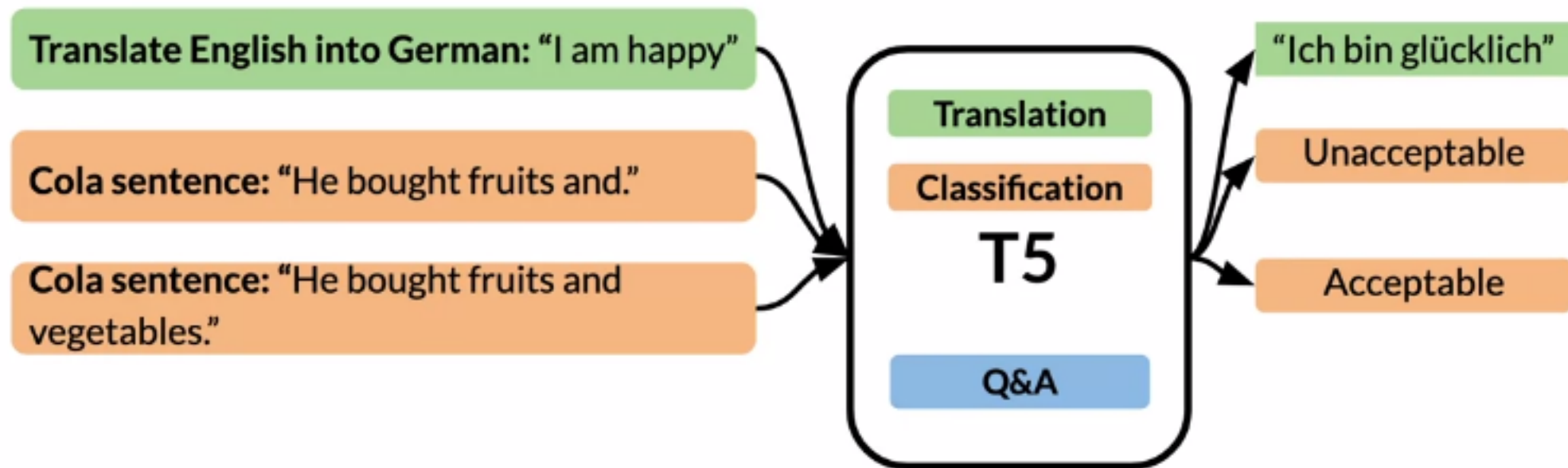
T5: Text-To-Text Transfer Transformer



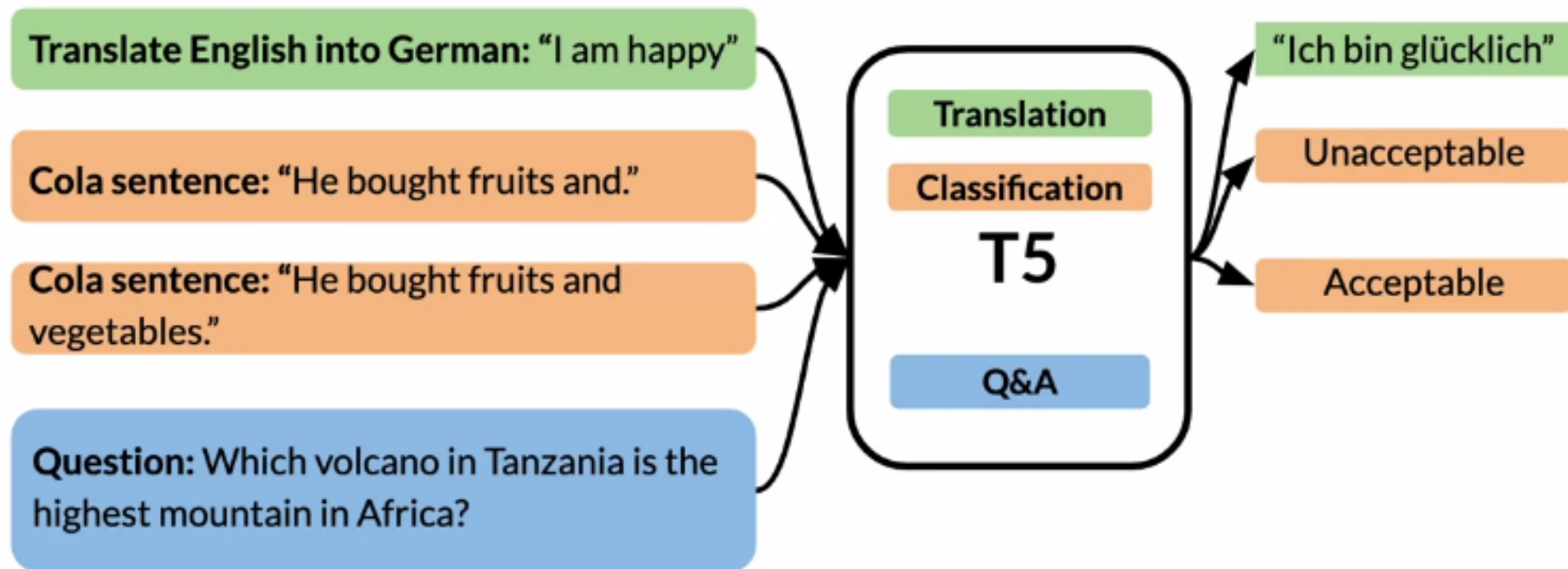
T5: Text-To-Text Transfer Transformer



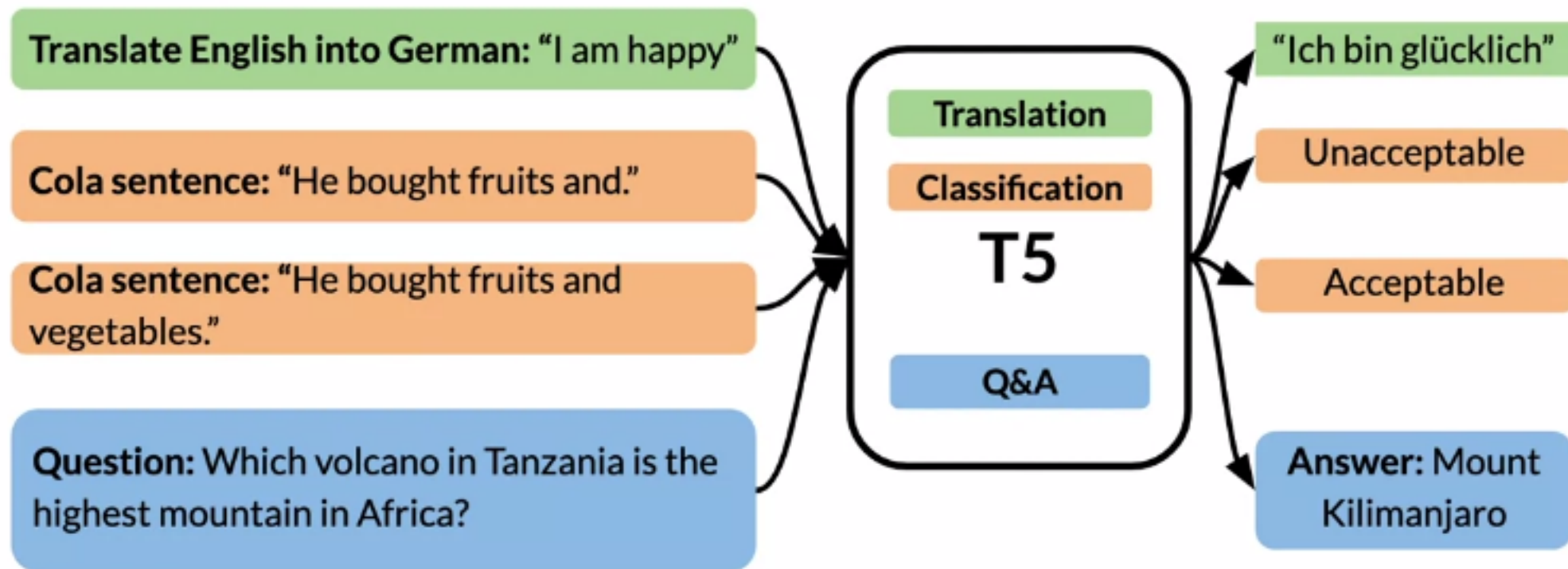
T5: Text-To-Text Transfer Transformer



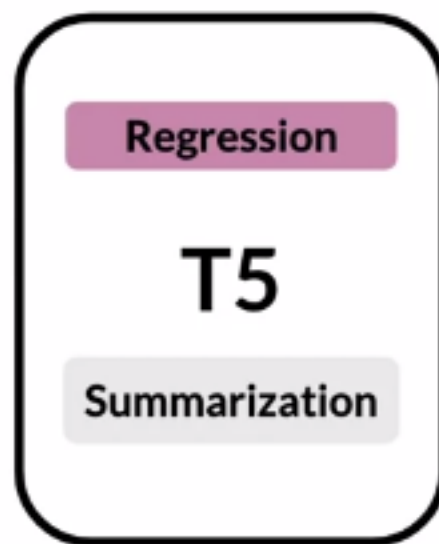
T5: Text-To-Text Transfer Transformer



T5: Text-To-Text Transfer Transformer

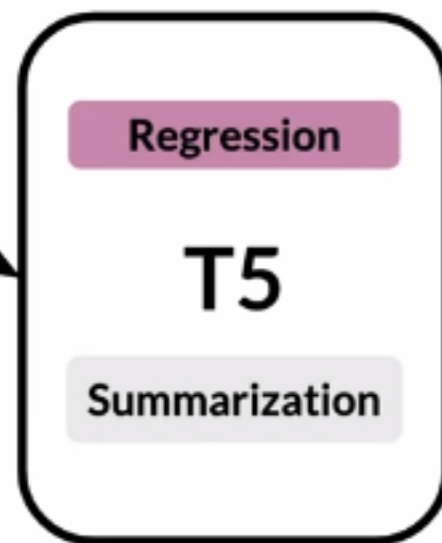


T5: Text-To-Text Transfer Transformer



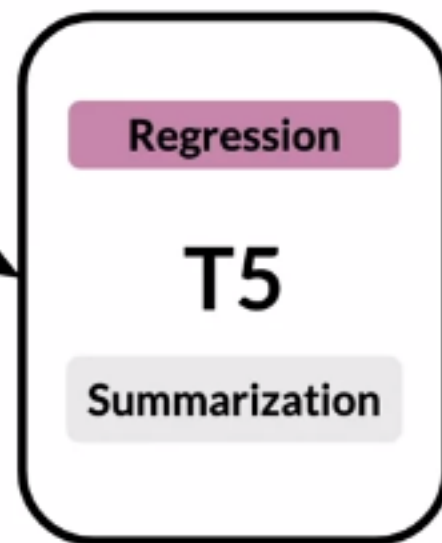
T5: Text-To-Text Transfer Transformer

Source sentence1: "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."



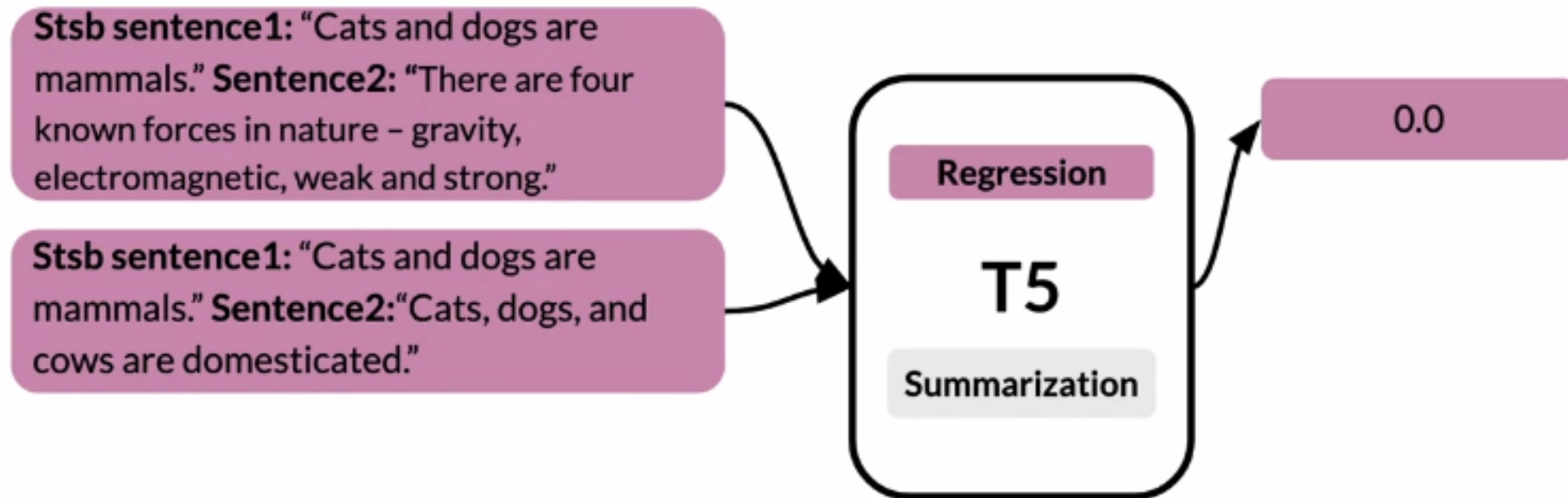
T5: Text-To-Text Transfer Transformer

Source sentence1: "Cats and dogs are mammals." **Sentence2:** "There are four known forces in nature – gravity, electromagnetic, weak and strong."

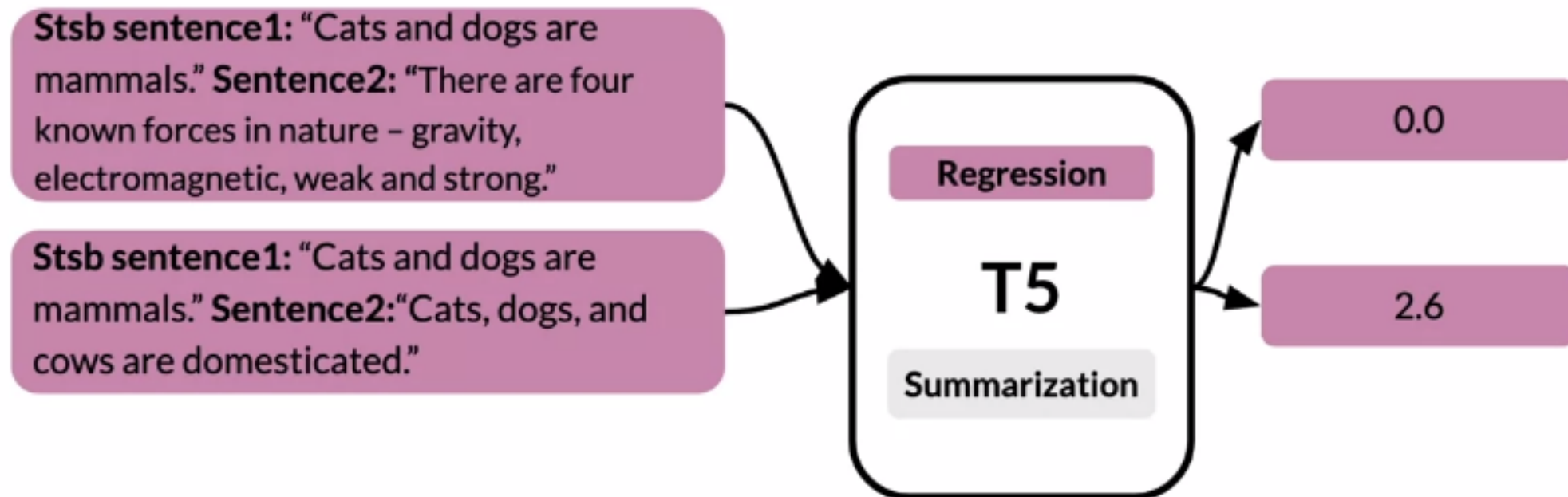


0.0

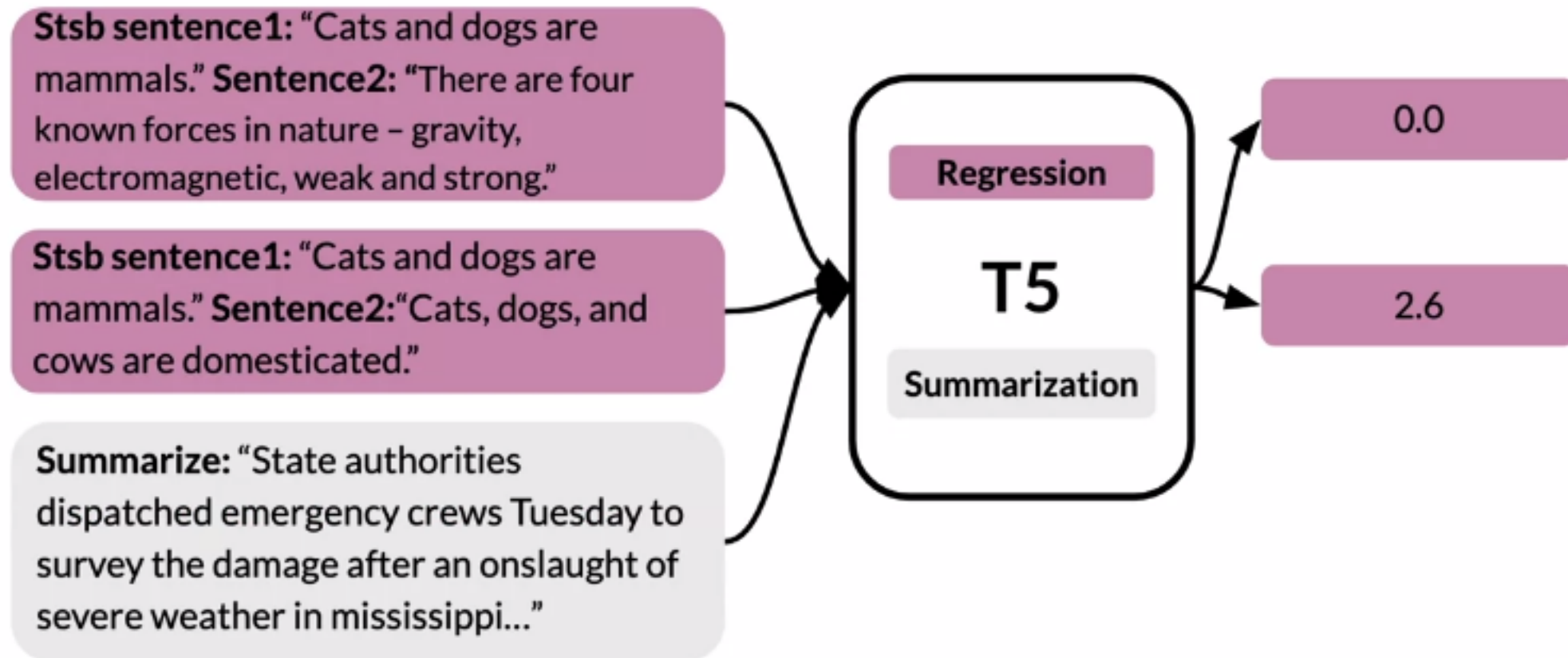
T5: Text-To-Text Transfer Transformer



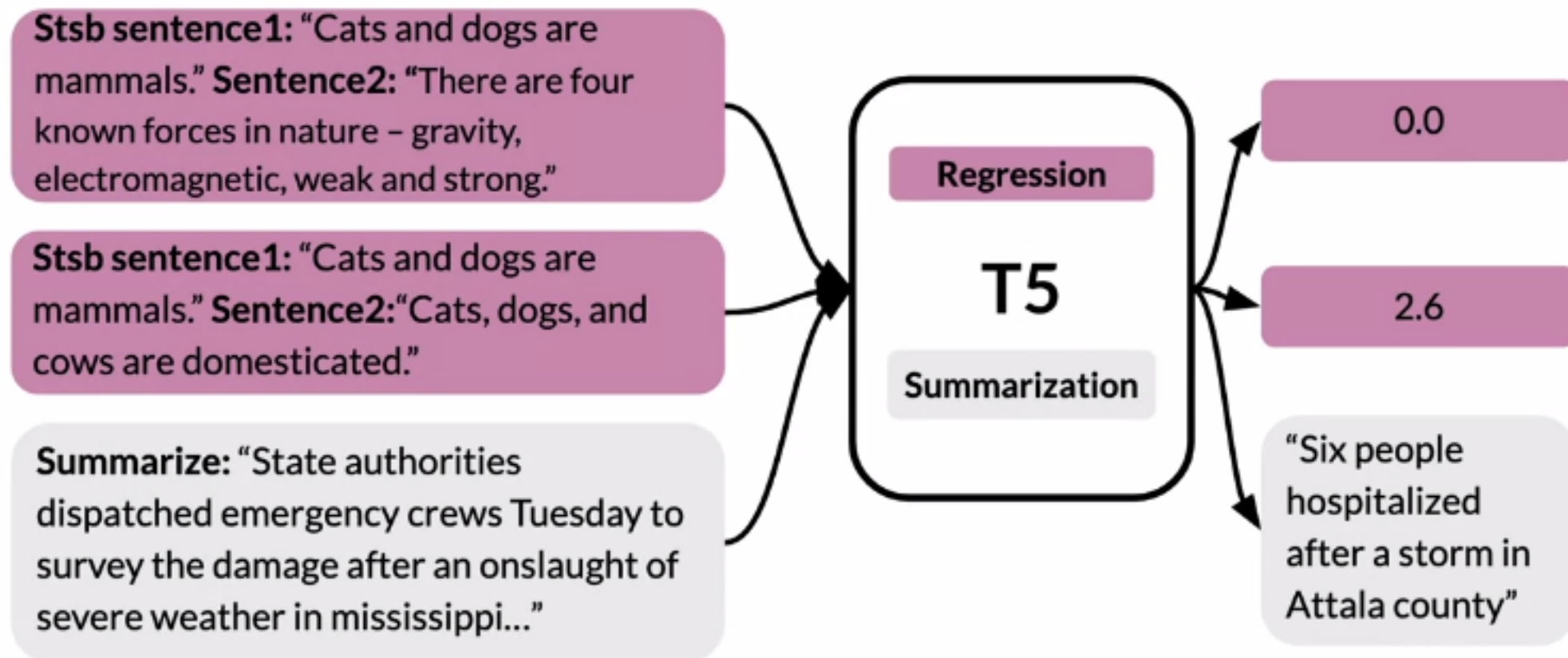
T5: Text-To-Text Transfer Transformer



T5: Text-To-Text Transfer Transformer



T5: Text-To-Text Transfer Transformer



T5 Demo

question 2

What colour hair did Charles Dickens' character David Copperfield have?

You

P I

SUBMIT

1

T5

LOCKED IN!

1

T5 Demo

question 1

What country is the largest oil producer in Africa?

You 0

T5 0

0:05 / 0:25

T5 Demo

The screenshot displays a web-based interface for a T5 model demo. At the top, a black header bar is followed by a light blue bar containing a blue button labeled "question 1". Below this, the question "What country is the largest oil producer in Africa?" is shown. The interface then displays two rows of text: "You" followed by "Nigeria" and "T5" followed by "Nigeria". To the right of each answer is a green button labeled "CORRECT!" and a purple circle containing the number "1". A blue button labeled "NEXT QUESTION" is positioned below the answers. Underneath the button, the text "Correct answer: **nigeria**" is displayed, followed by a link "► See all accepted answers". The bottom of the interface features a video player with a red progress bar, a timestamp of "0:09 / 0:25", and standard video controls (play, next, volume, settings, full screen, and close).

T5 Demo

The screenshot displays a web-based interface for a T5 model demo. At the top, a black bar is followed by a light blue bar containing a blue button labeled "question 2". Below this, the question "What colour hair did Charles Dickens' character David Copperfield have?" is presented in a monospaced font. The interface shows two rows of input and output: "You" with the input "Pink" and a red button labeled "NOPE :(" with a circled "1" next to it; and "T5" with the input "Brown" and another identical red button and circled "1". A blue "NEXT QUESTION" button is centered below the inputs. The bottom section of the interface has a light gray background and displays the text "Correct answer: RED" and a link "▶ See all accepted answers". At the very bottom, a video player control bar shows a progress bar at 0:14 / 0:25 and various control icons.

question 2

What colour hair did Charles Dickens' character David Copperfield have?

You Pink NOPE :(1

T5 Brown NOPE :(1

NEXT QUESTION

Correct answer: RED

▶ See all accepted answers

0:14 / 0:25

T5 Demo

The screenshot displays a web-based interface for a T5 model demo. At the top, a black bar is followed by a light blue header containing a blue button labeled "question 3". Below this, the question "Who wrote the 1961 novel The Prime of Miss Jean Brodie ?" is shown in a monospaced font. The interface then compares two answers. The "You" section shows an empty text input field and a "SUBMIT" button, with a score of 1 in a blue circle. The "T5" section shows the generated answer "J.K. Rowling" and a "REMOVE" button, also with a score of 1 in a blue circle. A mouse cursor is positioned over the "REMOVE" button. The bottom of the interface features a large, empty light blue rectangular area and a video player control bar at the very bottom with a red progress line and icons for play, next, volume, and a timestamp of 0:16 / 0:25.

question 3

Who wrote the 1961 novel The Prime of Miss Jean Brodie ?

You SUBMIT 1

T5 J.K. Rowling REMOVE 1

T5 Demo

The screenshot displays a web-based interface for a T5 model demo. At the top, a black bar is followed by a light blue bar containing a blue button labeled "question 3". Below this, the question "Who wrote the 1961 novel The Prime of Miss Jean Brodie ?" is shown. Two responses are listed: "You" with the text "Muriel Spark|" in a light blue box and a blue "SUBMIT" button, and "T5" with a series of asterisks "*****" and a blue "LOCKED IN!" button. To the right of each response is a circular icon with the number "1". Below the responses is a large video player with a black mouse cursor in the center. The video player's control bar at the bottom shows a red progress line, a play/pause button, a volume icon, the text "0:23 / 0:25", and icons for settings, full screen, and a share icon.

question 3

Who wrote the 1961 novel The Prime of Miss Jean Brodie ?

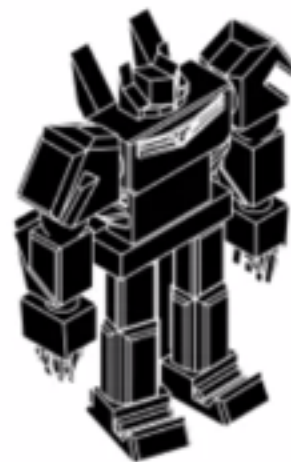
You Muriel Spark| SUBMIT 1

T5 ***** LOCKED IN! 1

0:23 / 0:25

Summary

- Transformers are suitable for a wide range of NLP applications
- GPT-2, BERT and T5 are the cutting-edge Transformers
- T5 is a powerful multi-task transformer



Outline

- Introducing attention (Translation example)
- Mathematics behind Attention



Introducing attention - Translation example



Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

I am happy



Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

I am happy



Ich bin glücklich

Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).

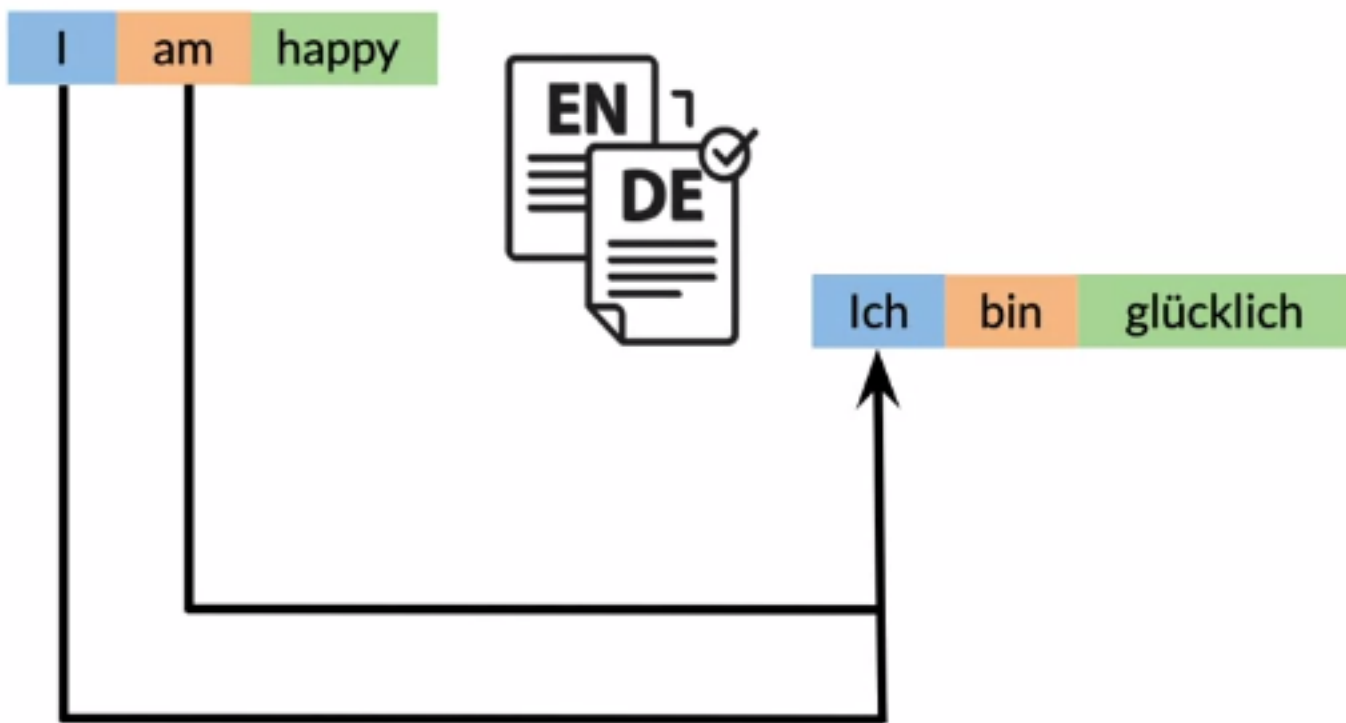
I am happy



Ich bin glücklich

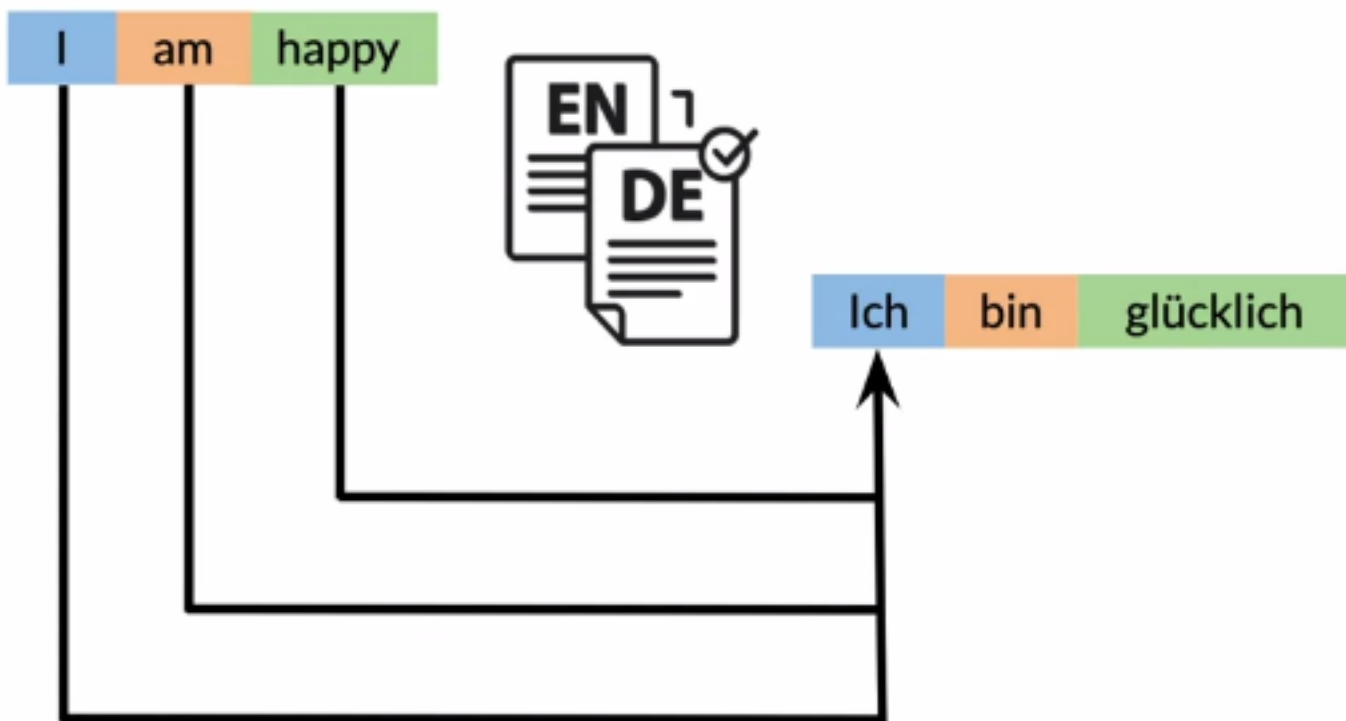
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



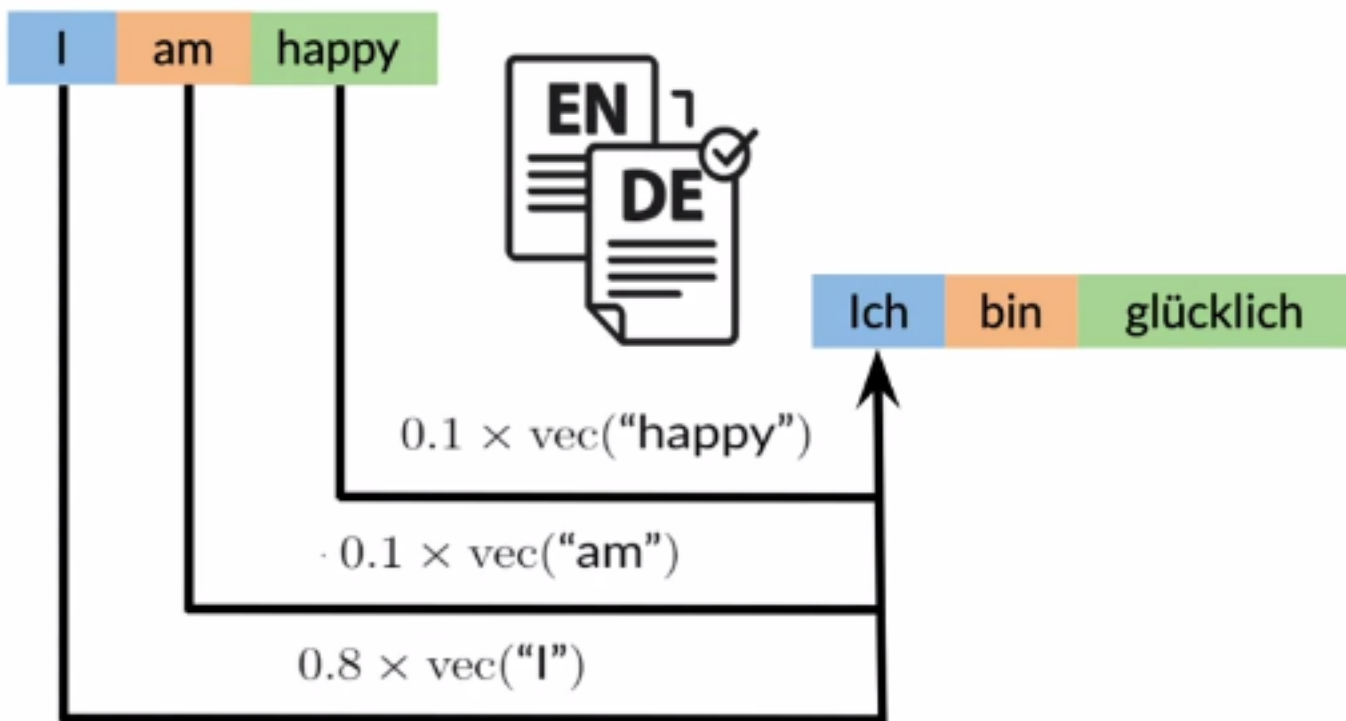
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



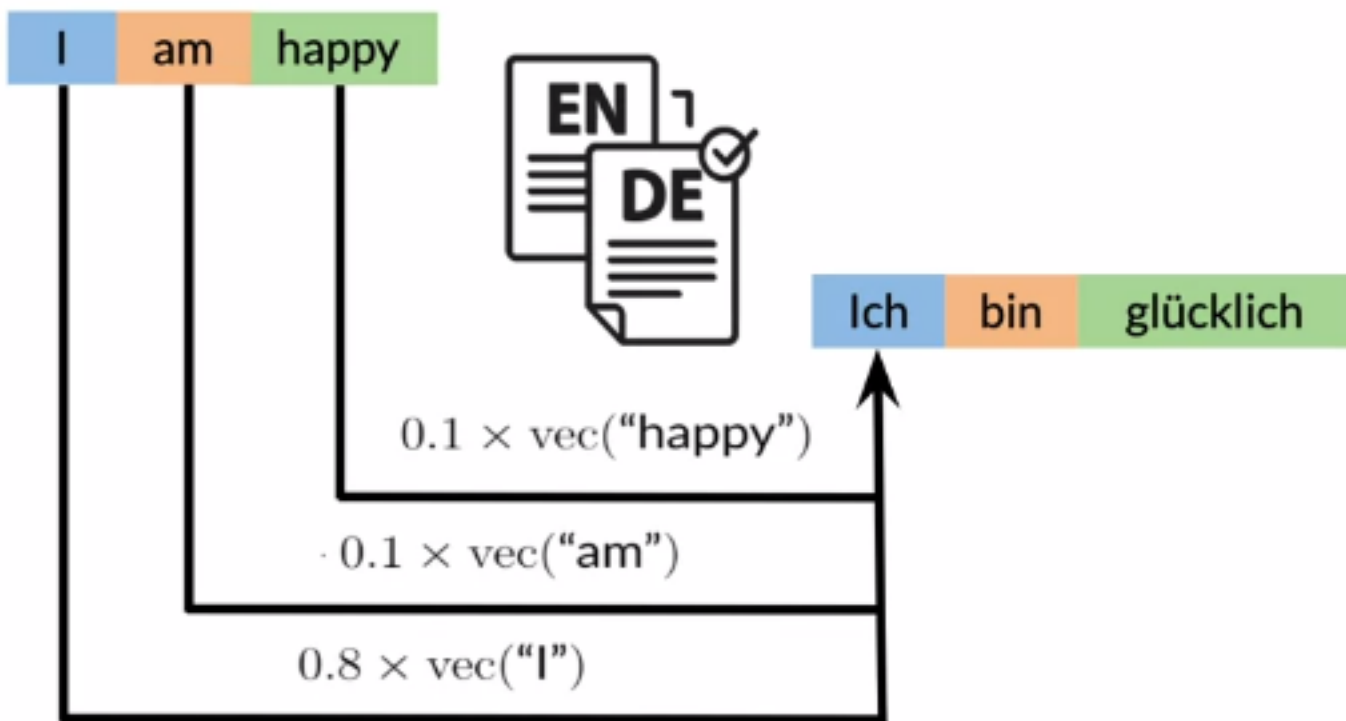
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.



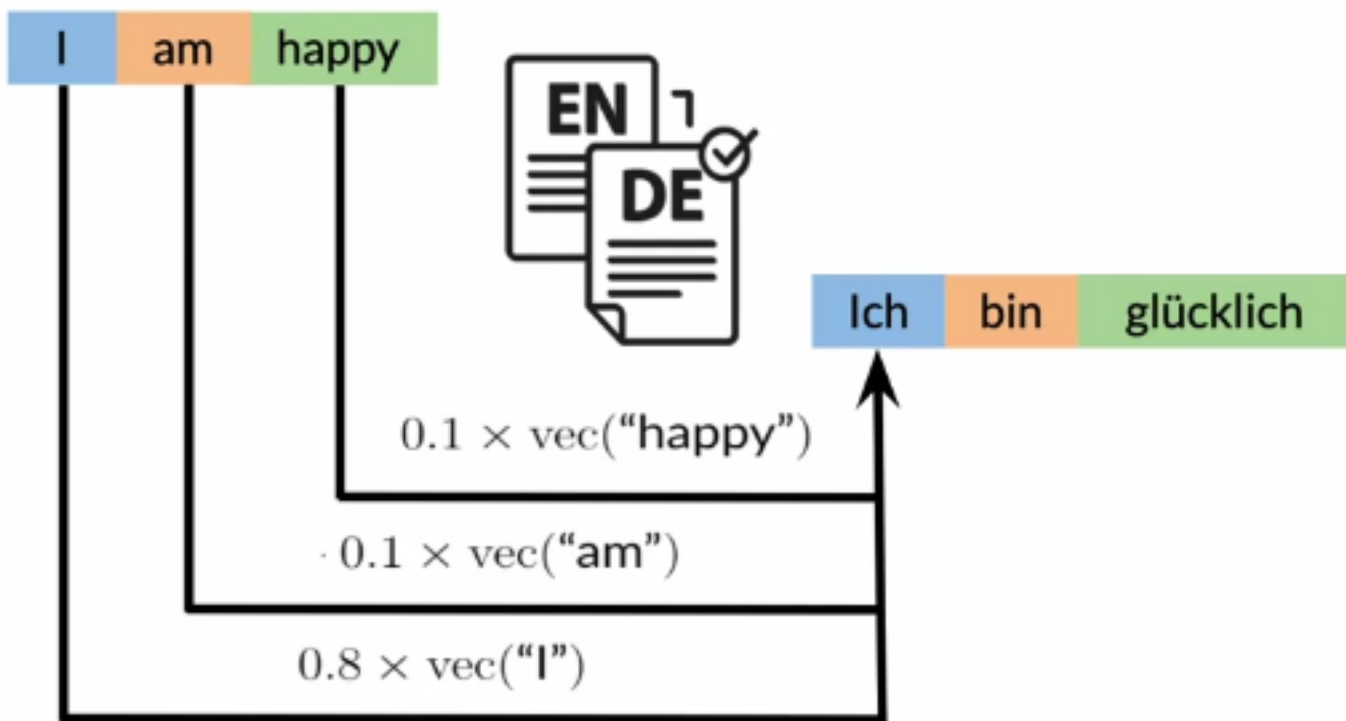
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



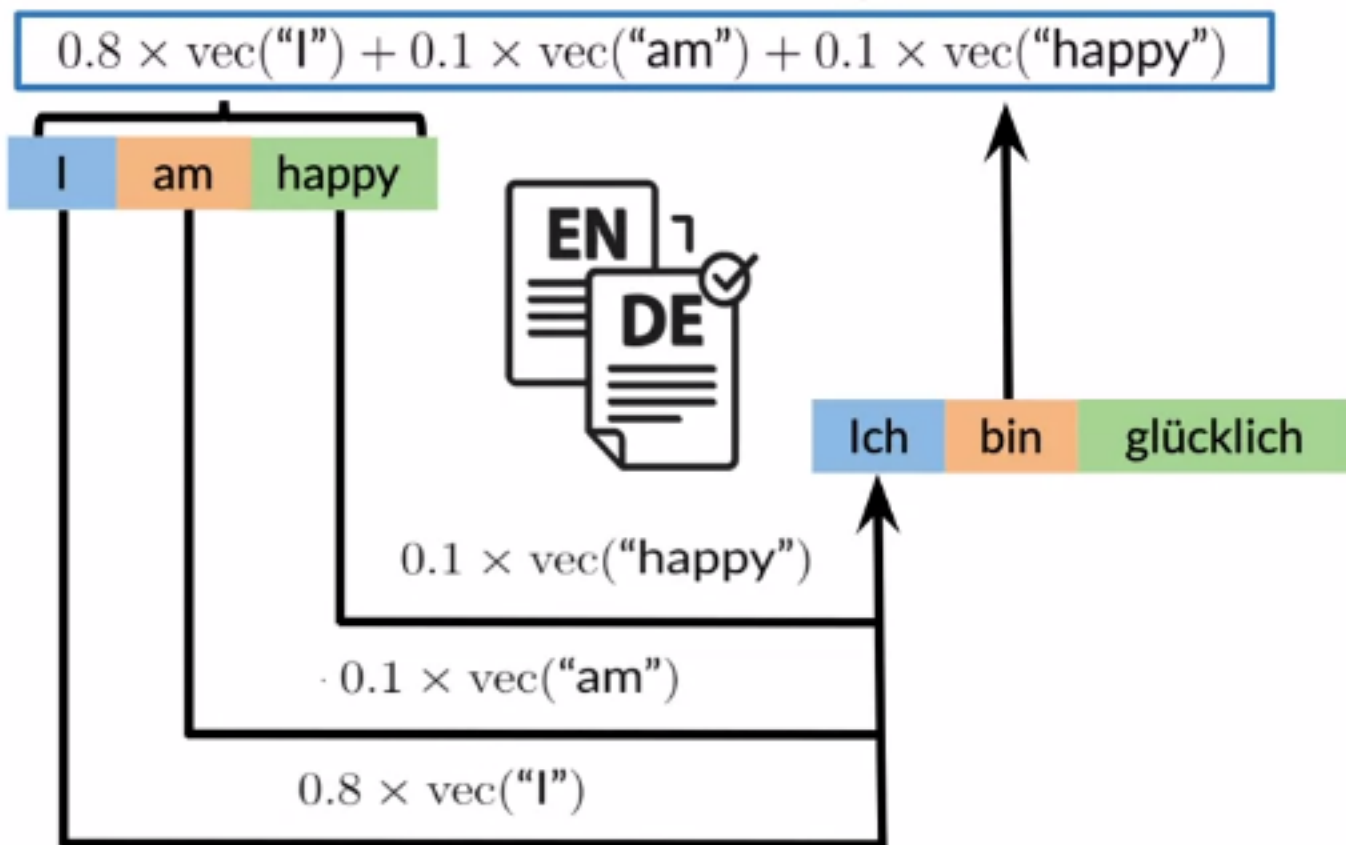
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



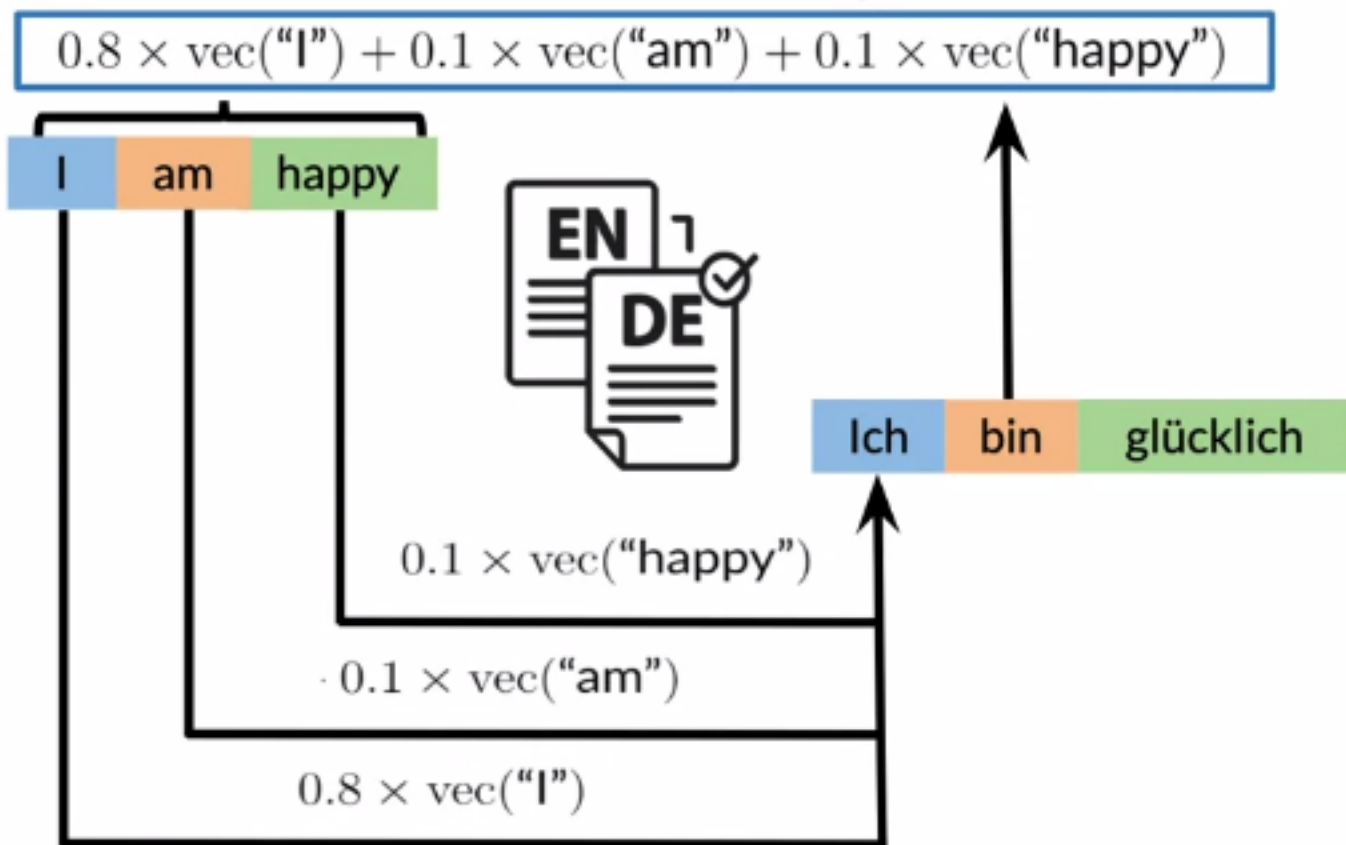
Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



Introducing attention - Translation example

- A query (German word) looks for similar keys (English words).
- Each key has a probability of being a match for the query.
- Return sum of the keys weighted by their probabilities.



Queries, Keys and Values

Queries, Keys and Values

Q

K

V

Queries, Keys and Values

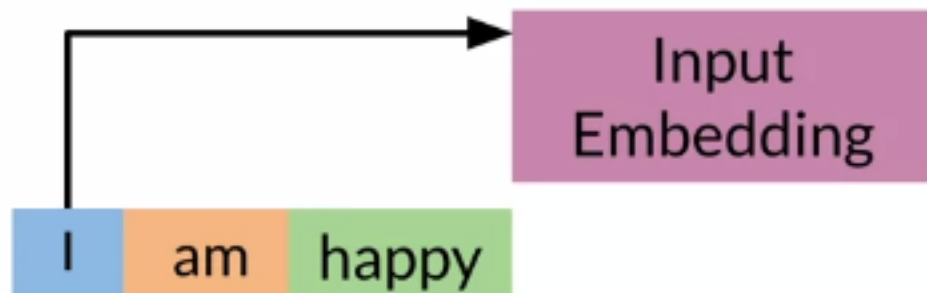
I am happy

Q

K

V

Queries, Keys and Values

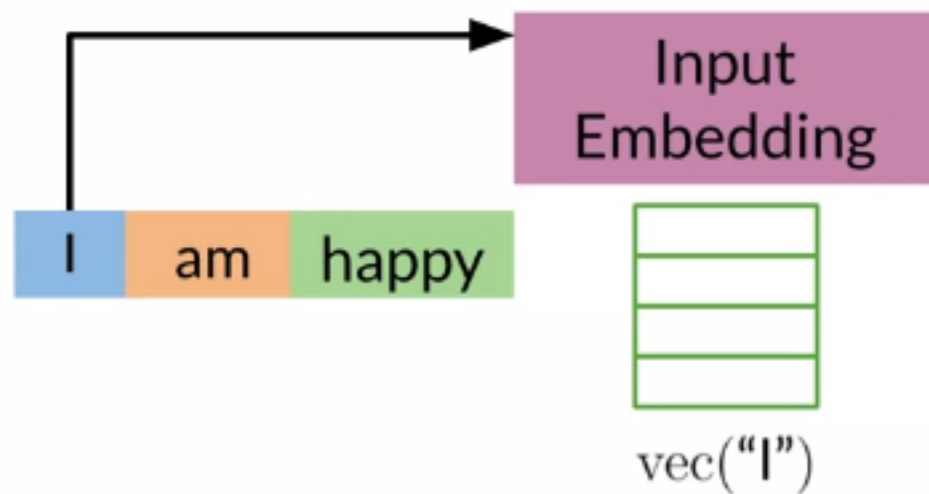


Q

K

V

Queries, Keys and Values

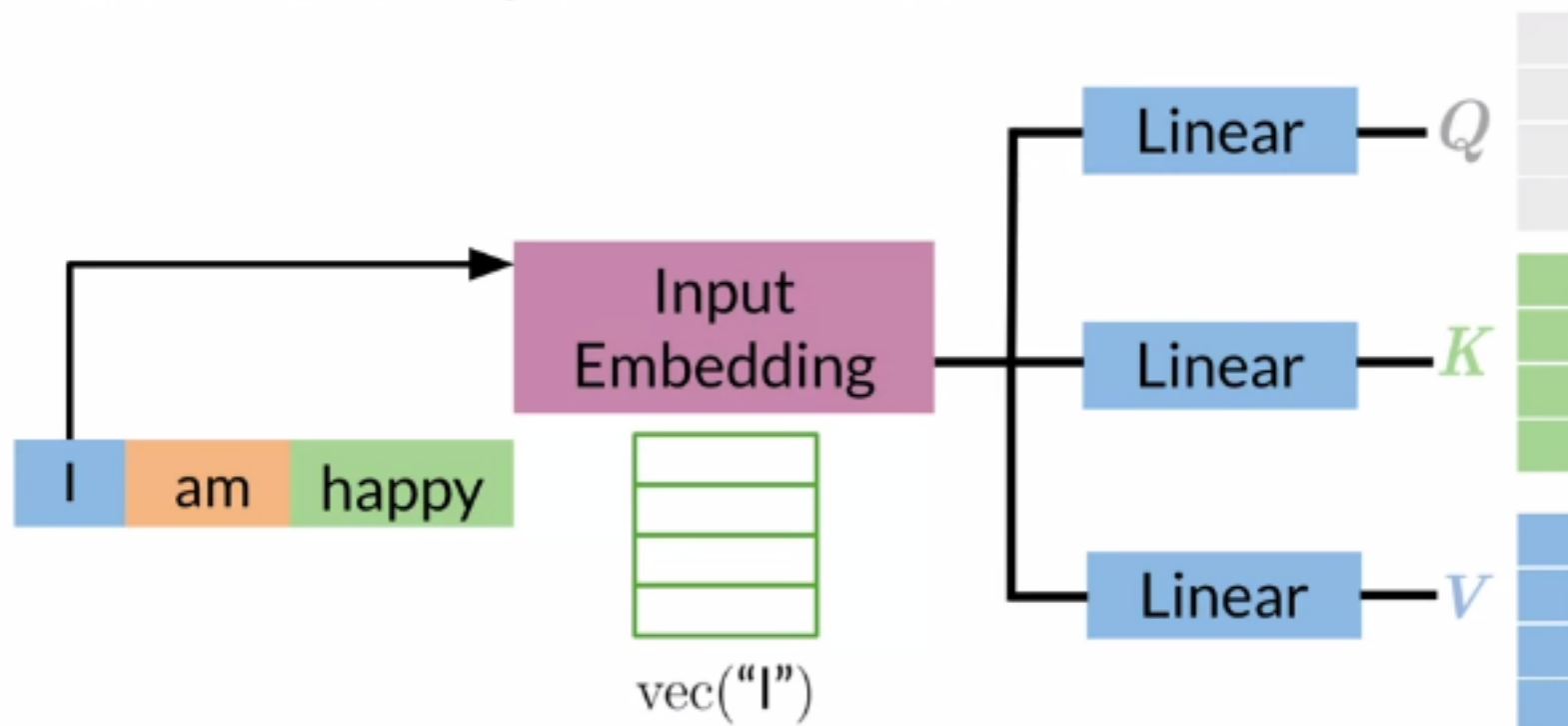


Q

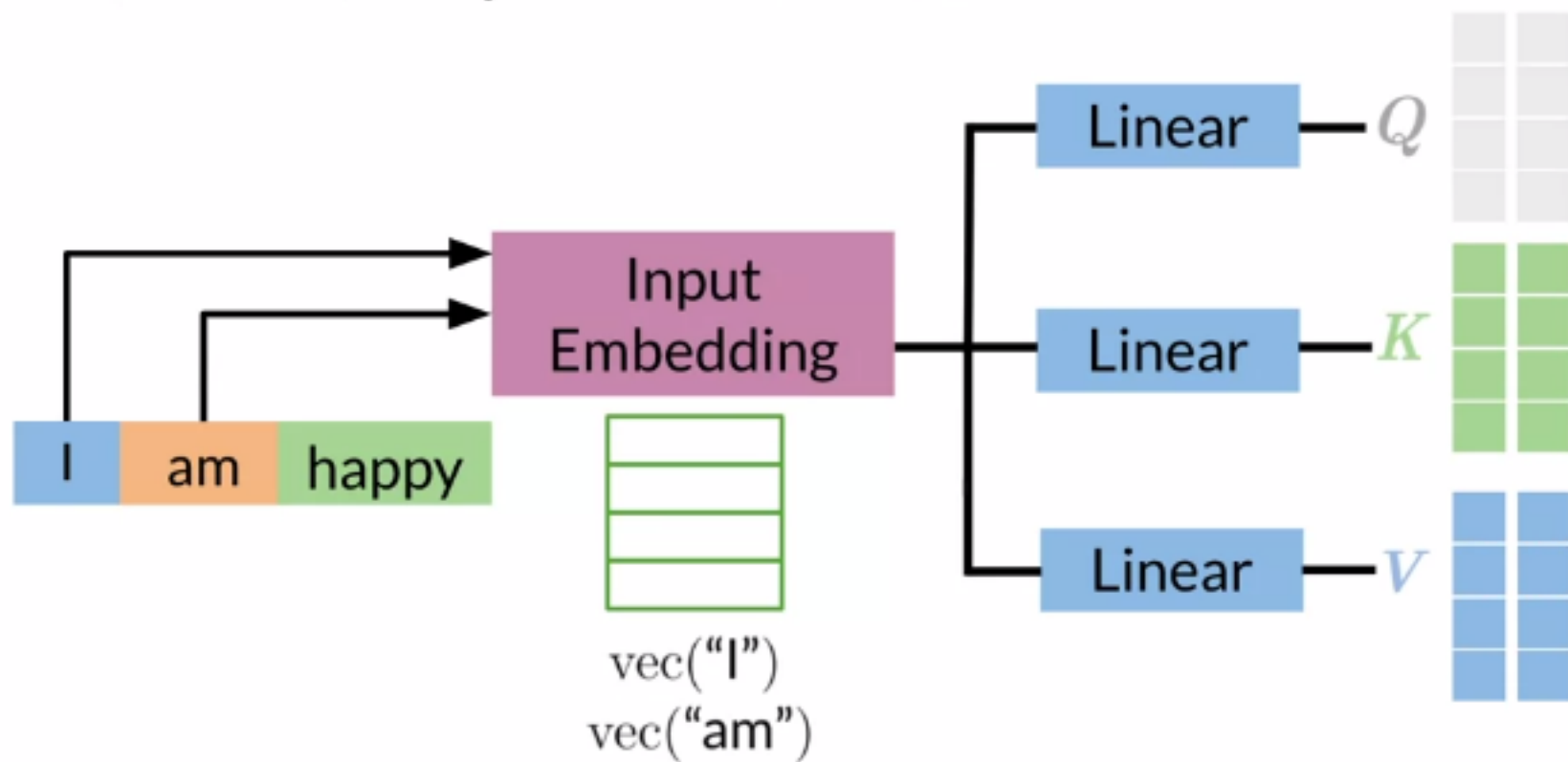
K

V

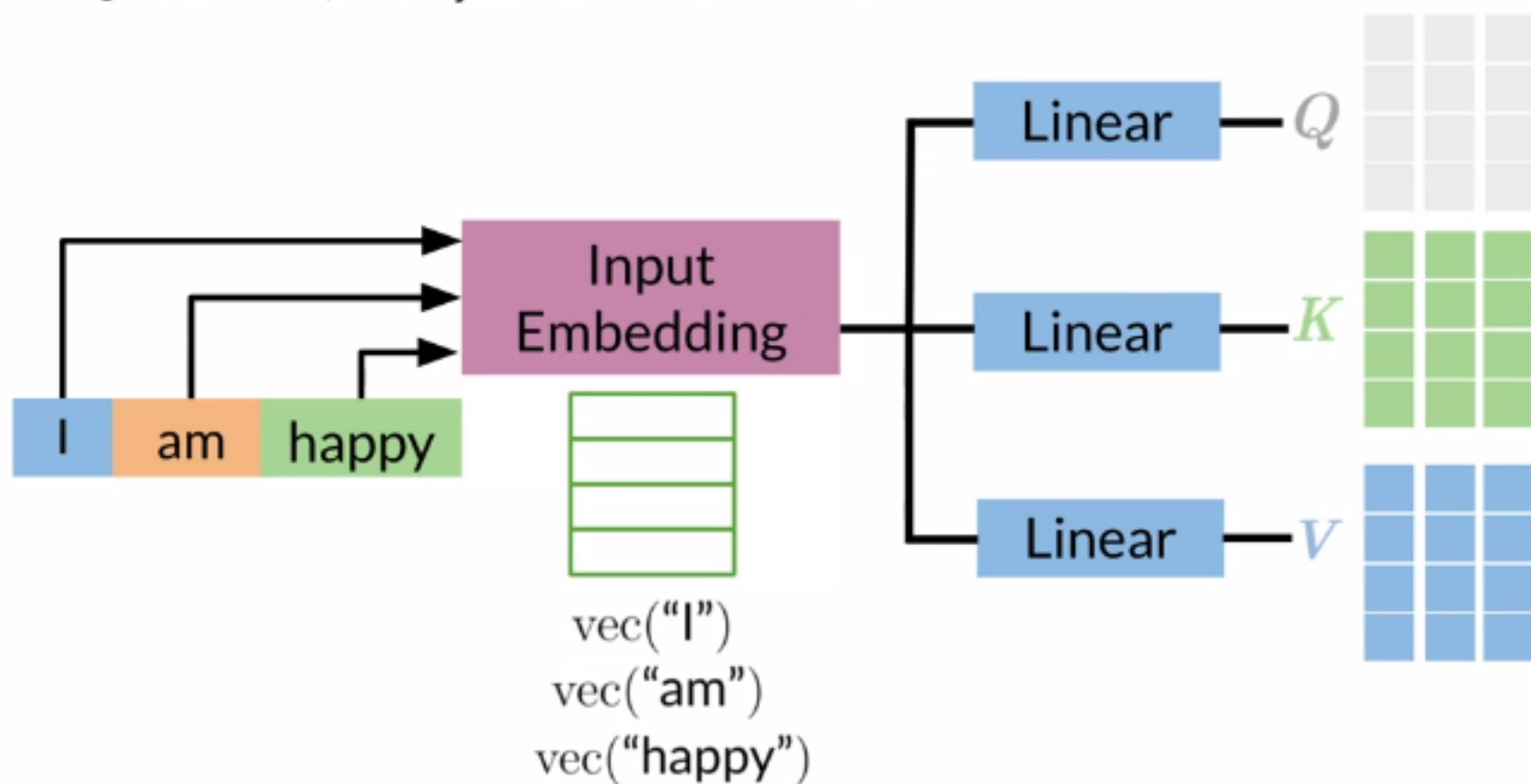
Queries, Keys and Values



Queries, Keys and Values



Queries, Keys and Values



Concept of attention



Q



K

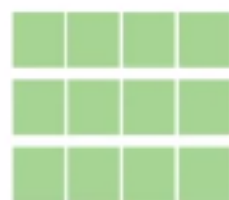


V

Concept of attention



Q

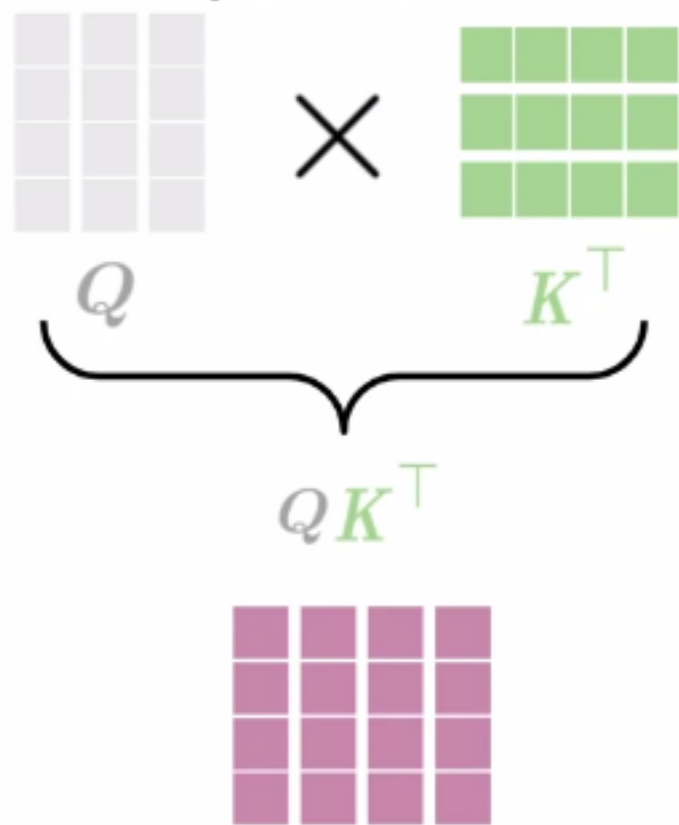


K^T



V

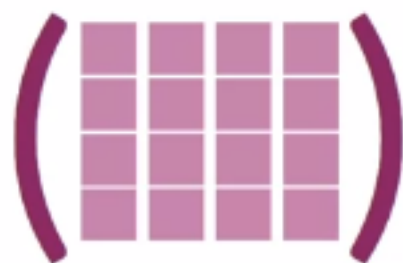
Concept of attention



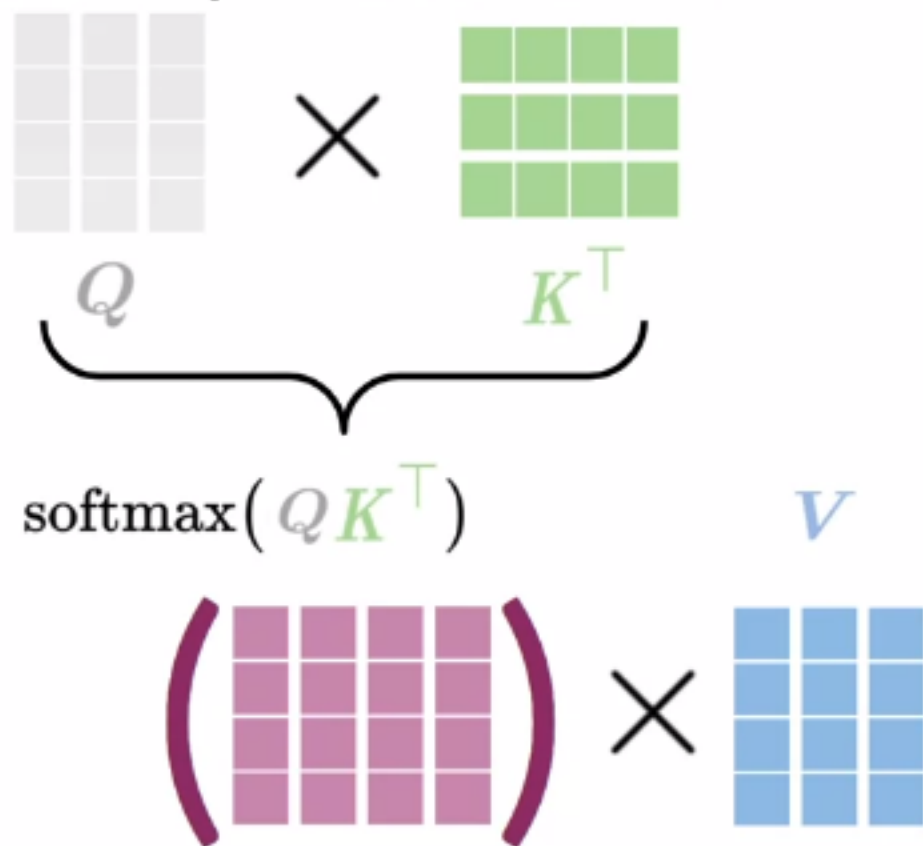
Concept of attention



$\text{softmax}(QK^T)$



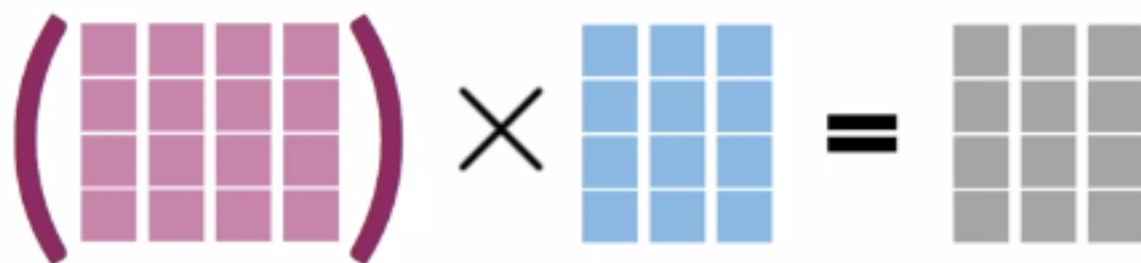
Concept of attention



Concept of attention



$\text{softmax}(QK^T)$



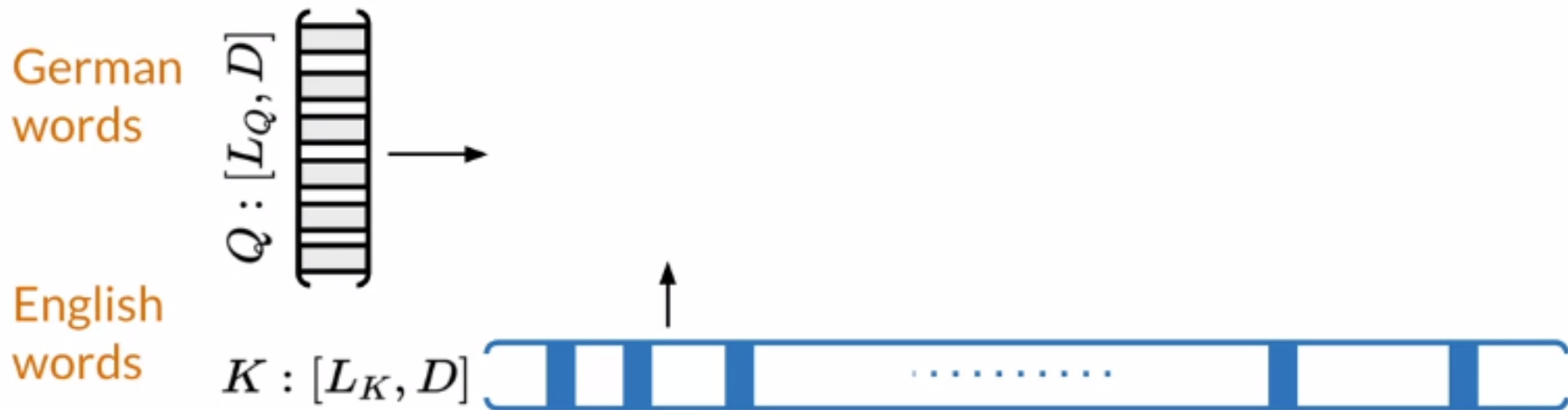
Attention math

English
words

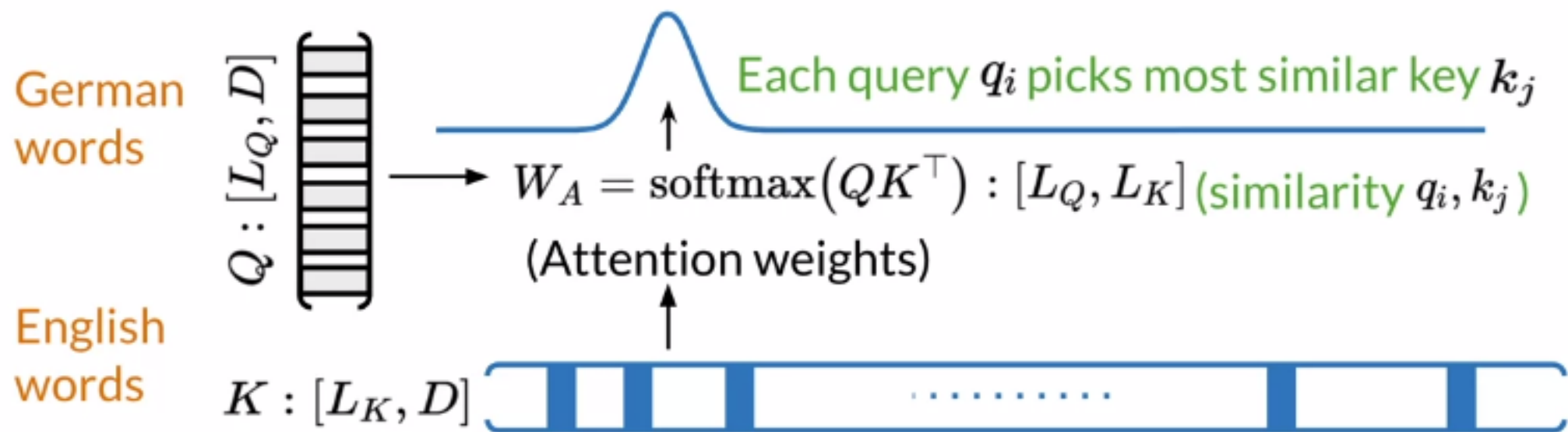
$K : [L_K, D]$



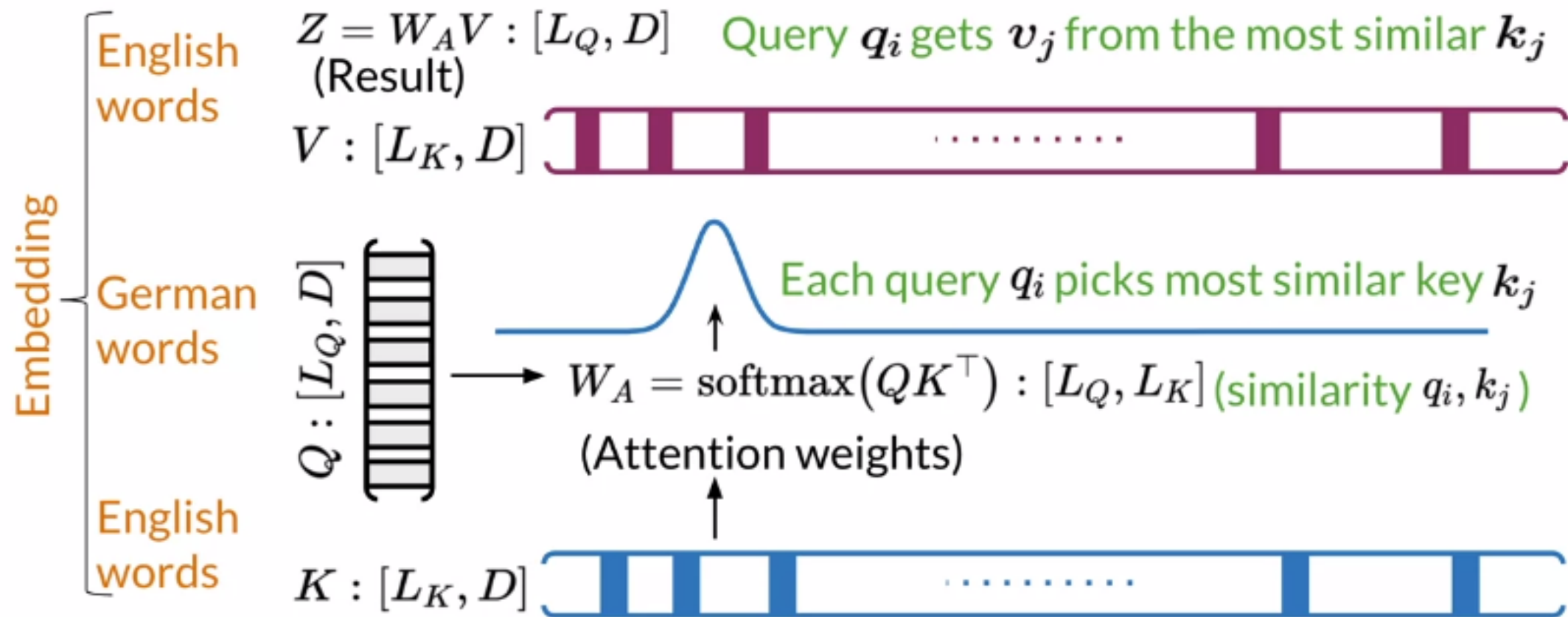
Attention math



Attention math



Attention math



Attention formula

Attention formula

Z

(Result)

Attention formula

$$Z = \text{attention}(Q, K, V)$$

(Result)

Attention formula

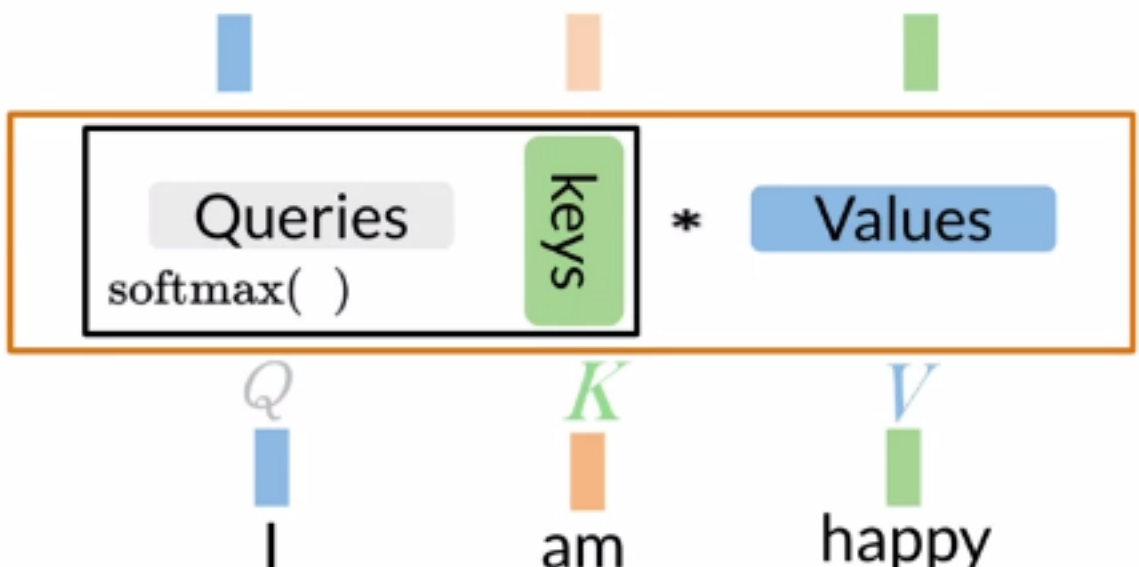
$$Z = \text{attention}(Q, K, V) = \text{softmax}(QK^T)V$$

(Result)

Attention formula

$Z = \text{attention}(Q, K, V) = \text{softmax}(QK^T)V = W_A V$

(Result)



The diagram illustrates the components of the attention formula. At the top, the formula $Z = \text{attention}(Q, K, V) = \text{softmax}(QK^T)V = W_A V$ is shown. A purple box highlights the $\text{attention}(Q, K, V)$ part. Below this, three colored vertical bars (blue, orange, green) correspond to Q , K , and V respectively. Below these bars is an orange box containing three elements: a grey box labeled 'Queries' with 'softmax()' below it, a green box labeled 'keys', and a blue box labeled 'Values'. An asterisk (*) is between the 'keys' and 'Values' boxes. Below the orange box, three colored vertical bars (blue, orange, green) correspond to the words 'I', 'am', and 'happy' respectively. Below these words are the labels Q , K , and V in blue, orange, and green respectively.

Attention formula

$$Z = \underset{\text{(Result)}}{\text{attention}}(\underset{\substack{\text{Queries} \\ \text{softmax}()}}{Q}, \underset{\text{keys}}{K}, \underset{\text{Values}}{V}) = \text{softmax}(QK^T)V = W_A V$$

The diagram illustrates the attention formula with color-coded components and their corresponding inputs:

- Queries** (grey box) is associated with the input **I** (blue bar).
- keys** (green box) is associated with the input **am** (orange bar).
- Values** (blue box) is associated with the input **happy** (green bar).

The formula shows that the attention result Z is calculated by applying a softmax function to the product of the Queries and the transpose of the keys, and then multiplying the result by the Values.

Summary

- Dot-product Attention is essential for Transformer
- The input to Attention are queries, keys, and values
- A softmax function makes attention more focused on best keys
- GPUs and TPUs is advisable for matrix multiplications



Outline

- Ways of Attention
- Overview of Causal Attention
- Math behind causal attention



Three ways of attention

Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)

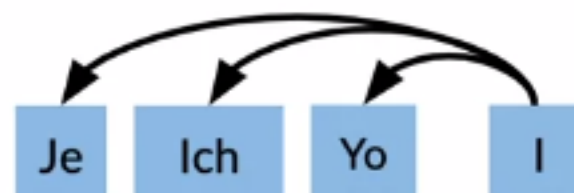
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)

Je Ich Yo I

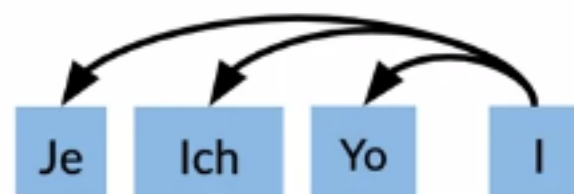
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)



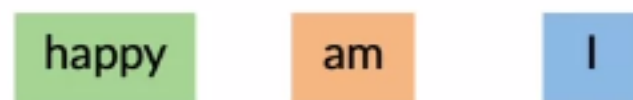
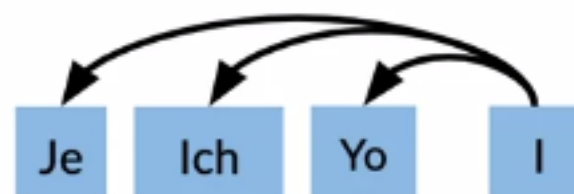
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



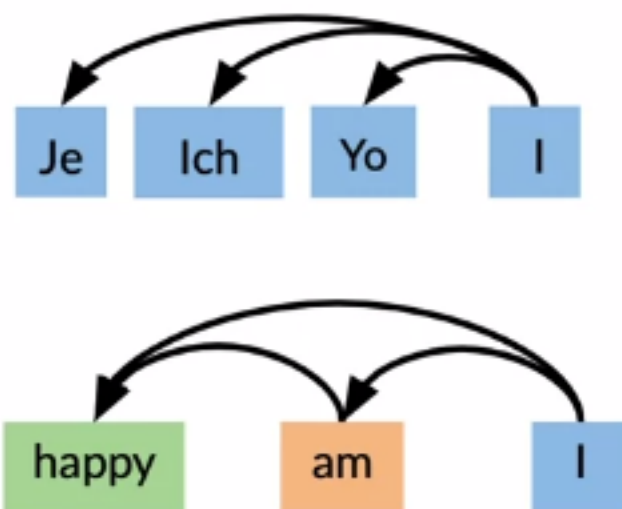
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



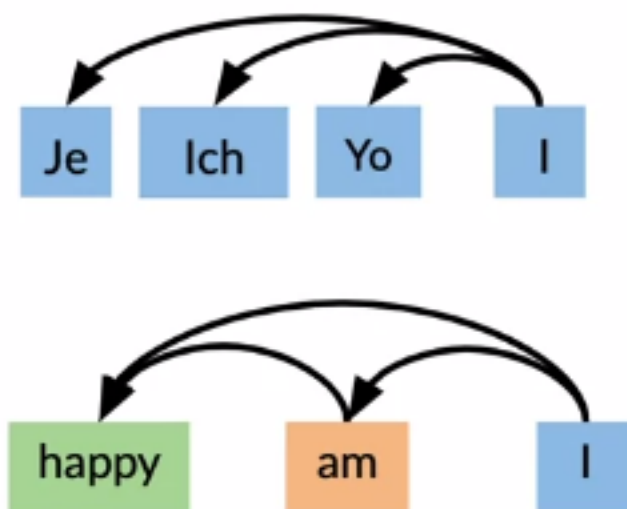
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)



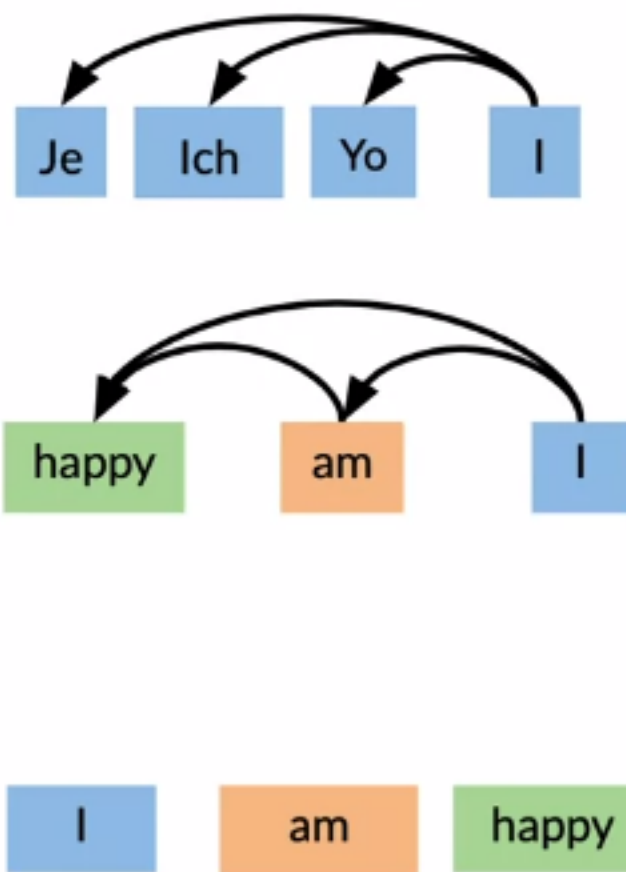
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



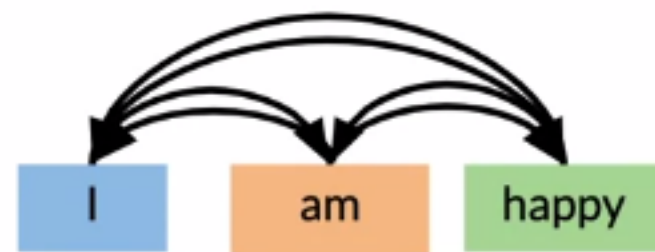
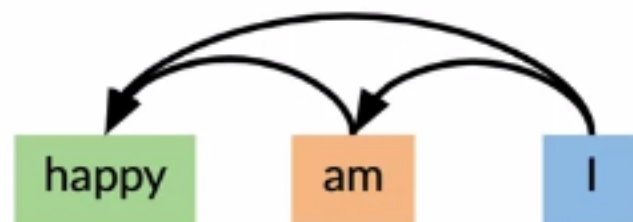
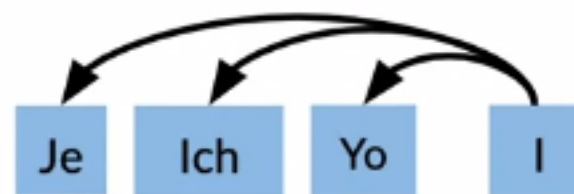
Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



Three ways of attention

- **Encoder/decoder attention:** One sentence (decoder) looks at another one (encoder)
- **Causal (self) attention:** In one sentence, words look at previous words (used for generation)
- **Bi-directional self attention:** In one sentence, words look at both previous and future words



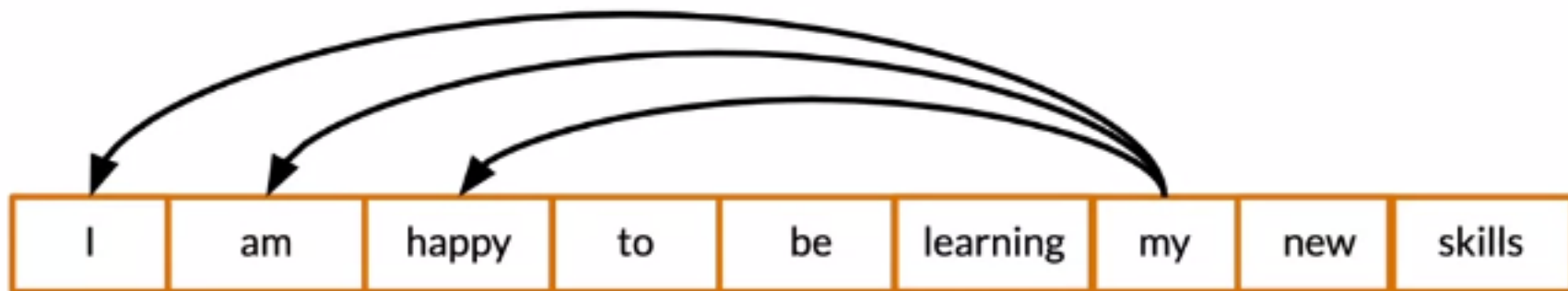
Causal attention

- Queries and keys are words from the same sentence

I	am	happy	to	be	learning	my	new	skills
---	----	-------	----	----	----------	----	-----	--------

Causal attention

- Queries and keys are words from the same sentence
- Queries should only be allowed to look at words before



Causal attention math

Causal attention math



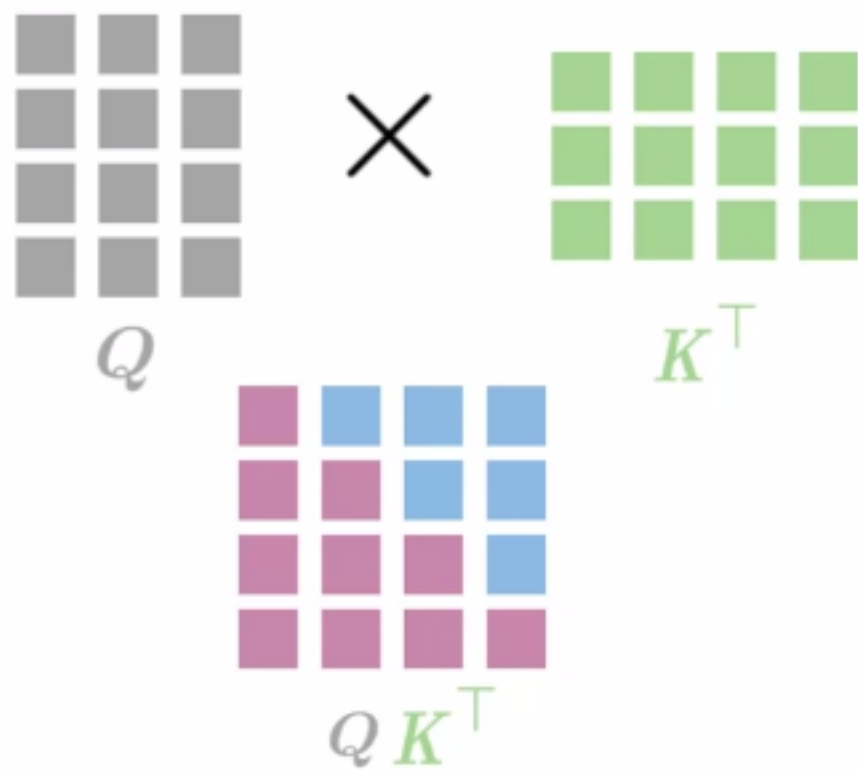
Q

Causal attention math

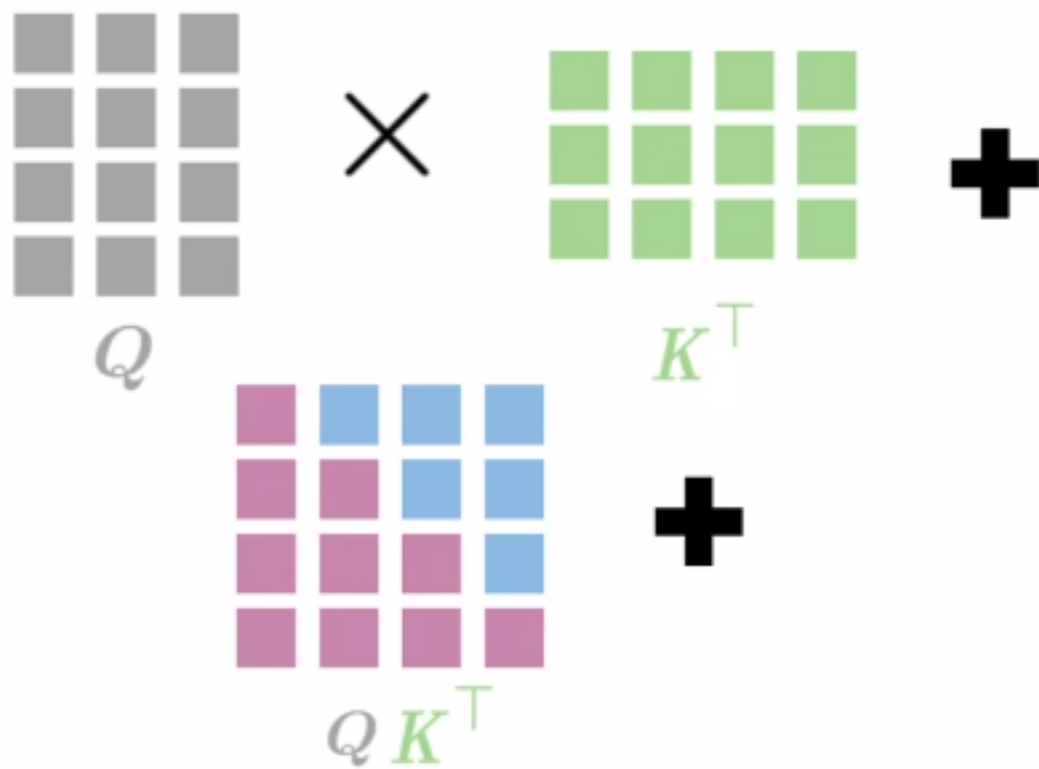


The diagram illustrates the matrix multiplication of Q and K^T . On the left, a 4x3 grid of gray squares represents matrix Q , with the label Q centered below it. In the middle is a large black multiplication symbol \times . On the right, a 3x4 grid of green squares represents matrix K^T , with the label K^T centered below it.

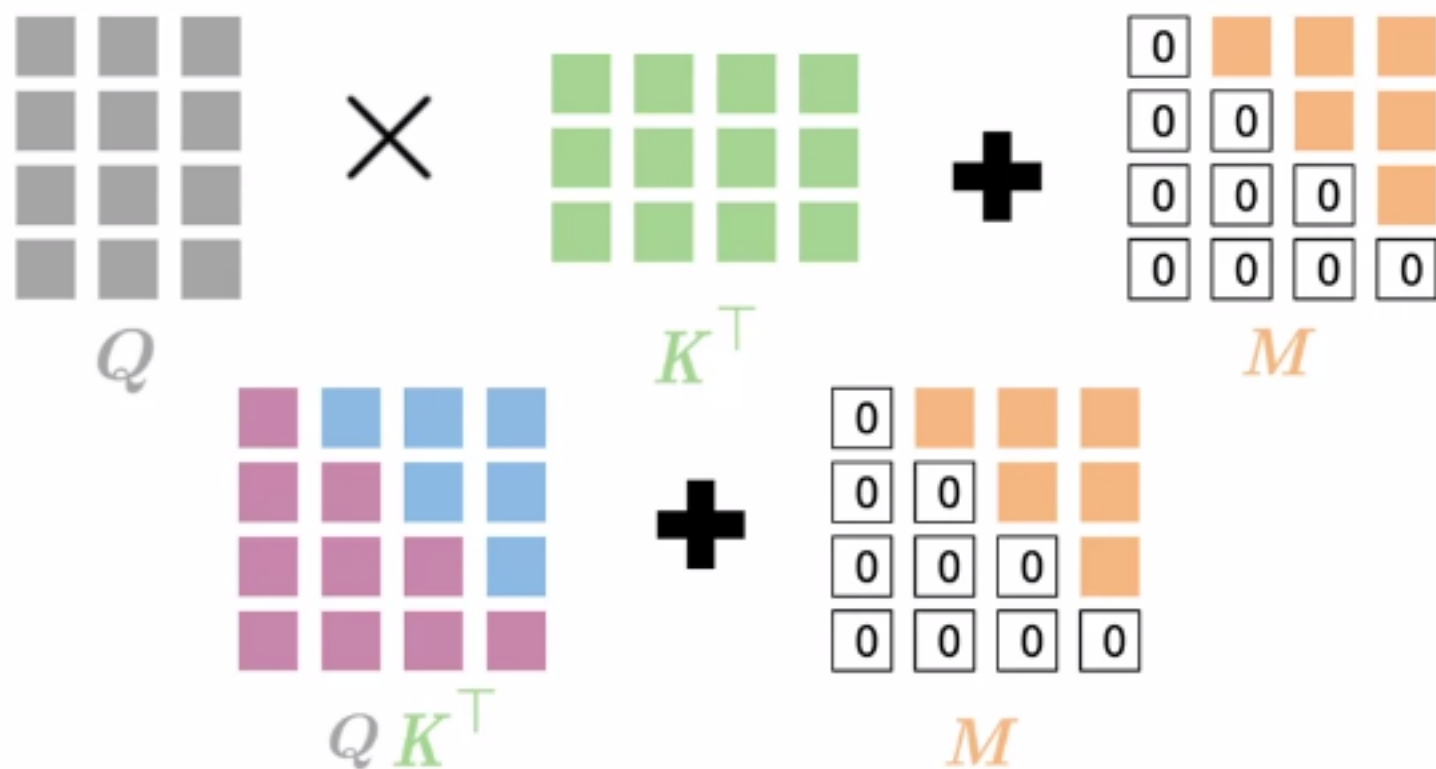
Causal attention math



Causal attention math

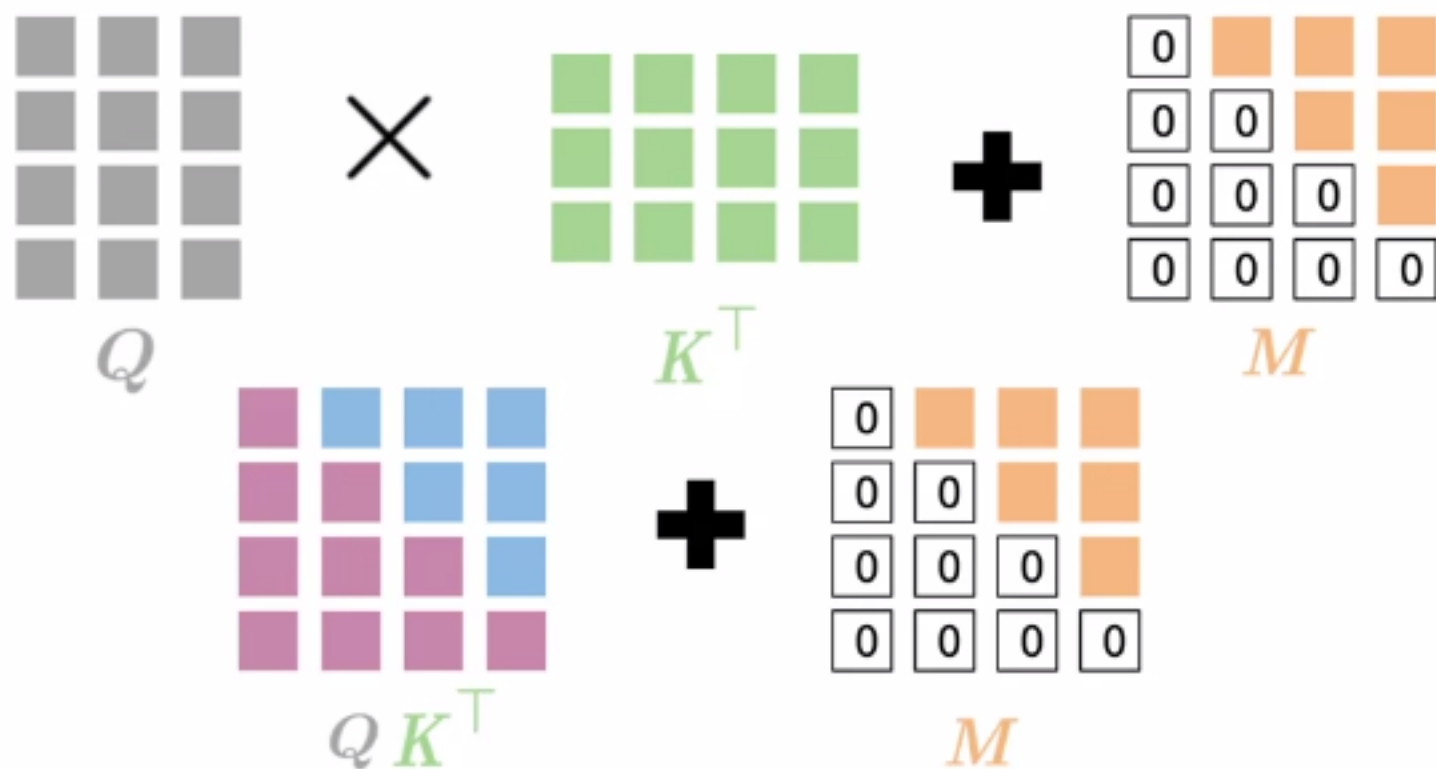


Causal attention math



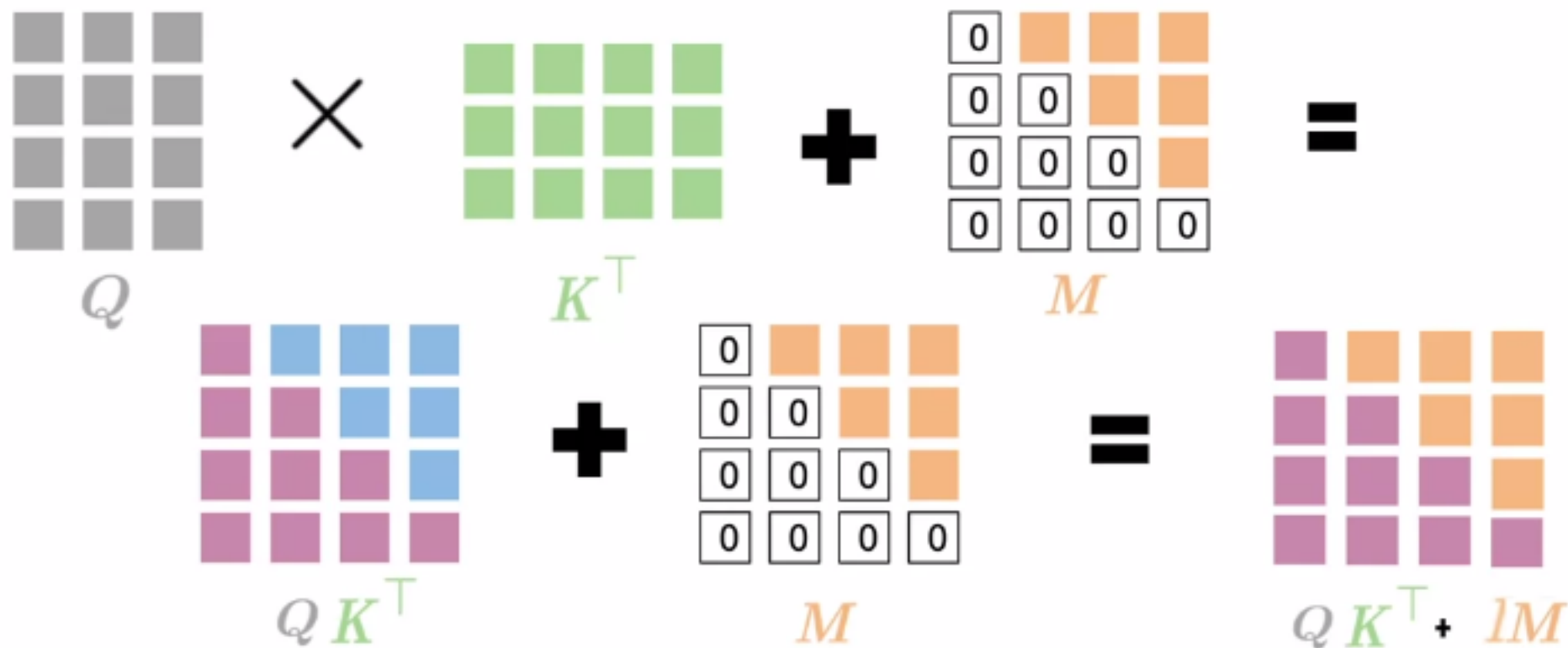
Causal attention math

■ → Minus infinity -in practice, a huge negative number



Causal attention math

■ → Minus infinity -in practice, a huge negative number



Causal attention math

$$W_A = \text{softmax} \left(QK^T + M \right) : [L, L]$$

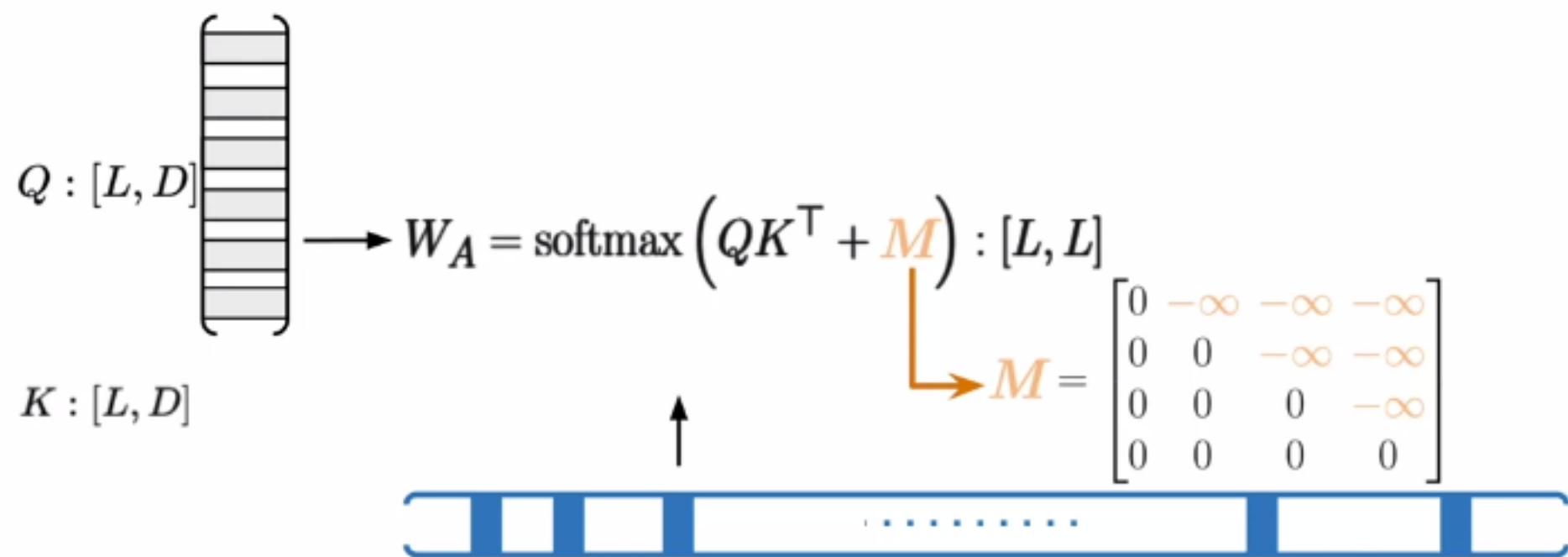
$K : [L, D]$

$M =$

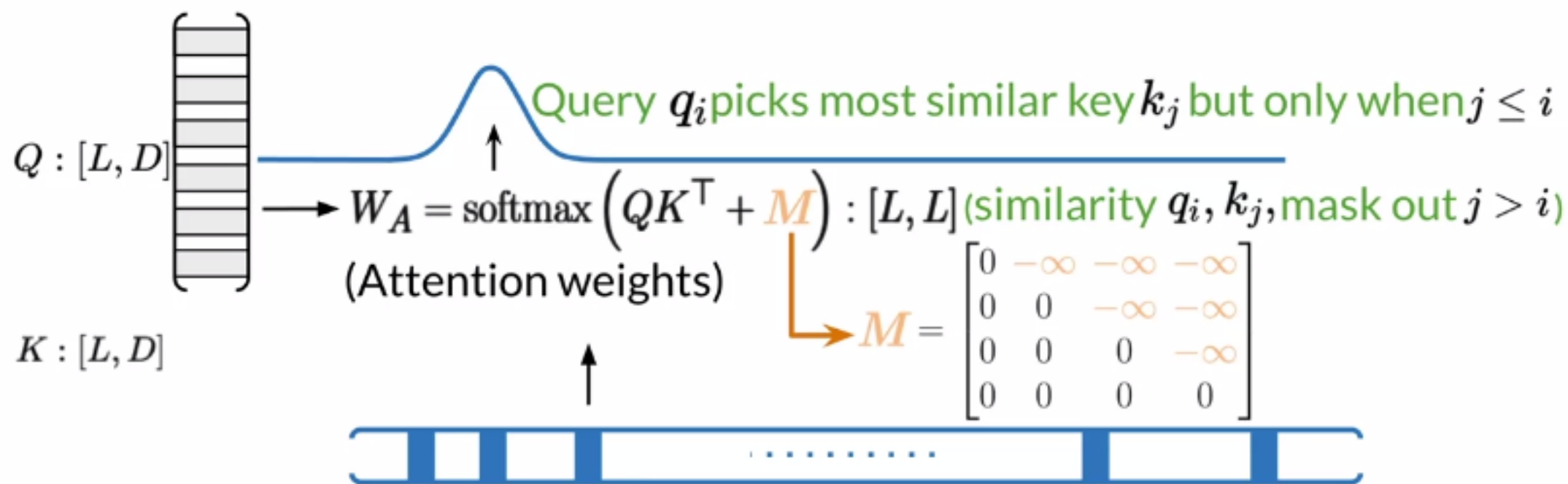
$$\begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



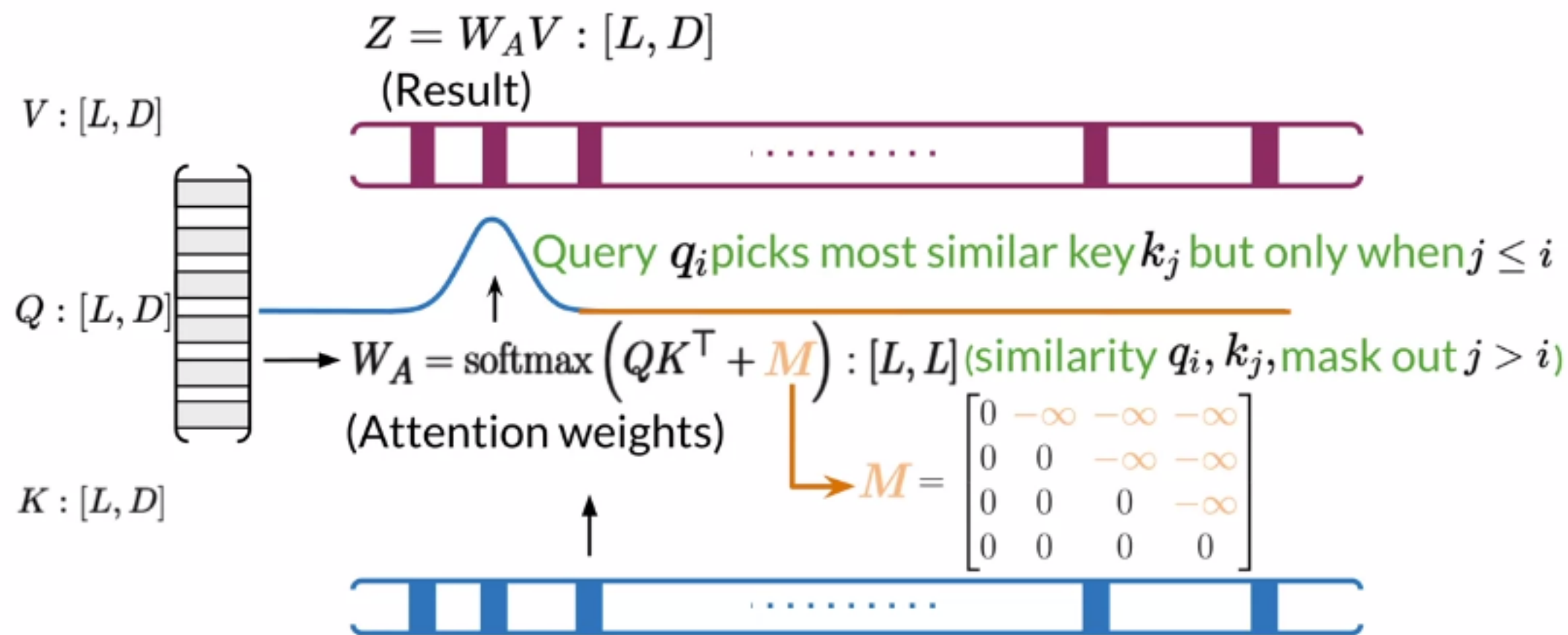
Causal attention math



Causal attention math



Causal attention math



Summary

- There are three main ways of Attention: Encoder/Decoder, Causal and Bi-directional type
- In causal attention, queries and keys come from the same sentence and queries search among words before only

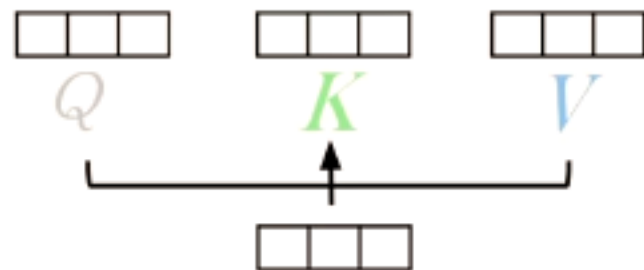


Outline

- Intuition of Multi-Head Attention
- Scaled dot-product and concatenation
- Multi-Head Attention formula

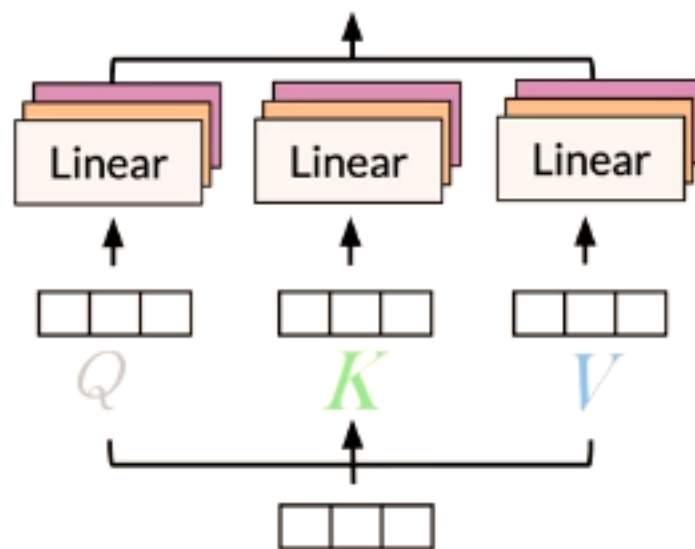


Multi-Head Attention



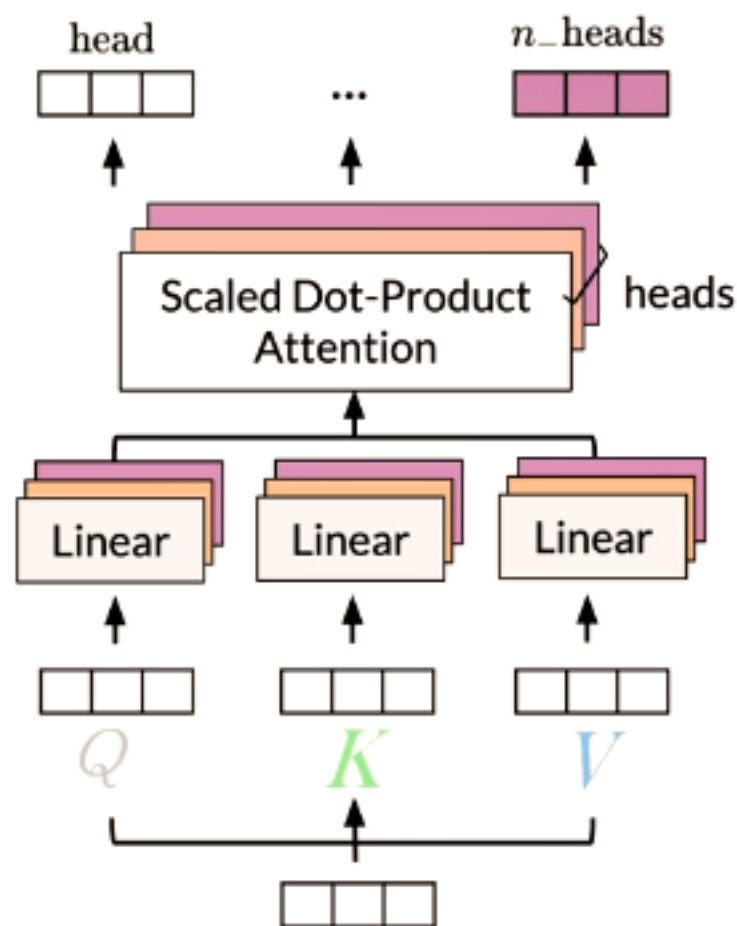
Multi-Head Attention

- Each head uses different linear transformations to represent words



Multi-Head Attention

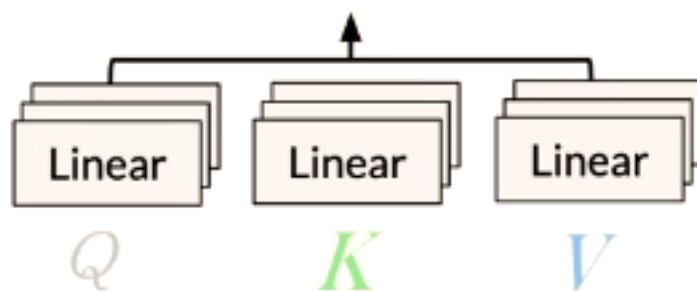
- Each head uses different linear transformations to represent words
- Different heads can learn different relationships between words



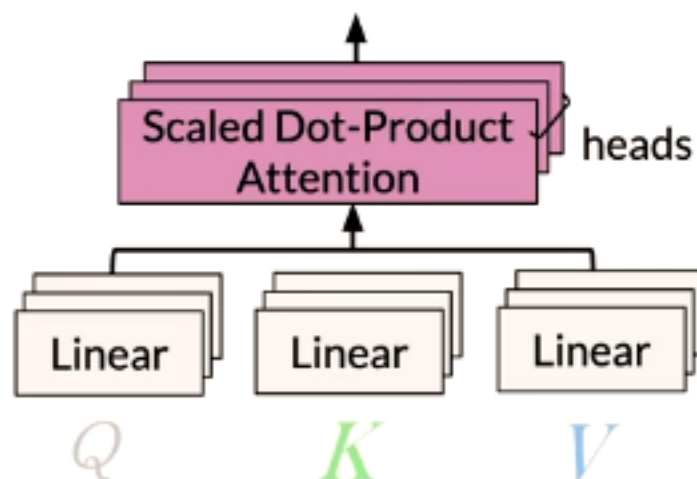
Multi-Head Attention - Overview

Q K V

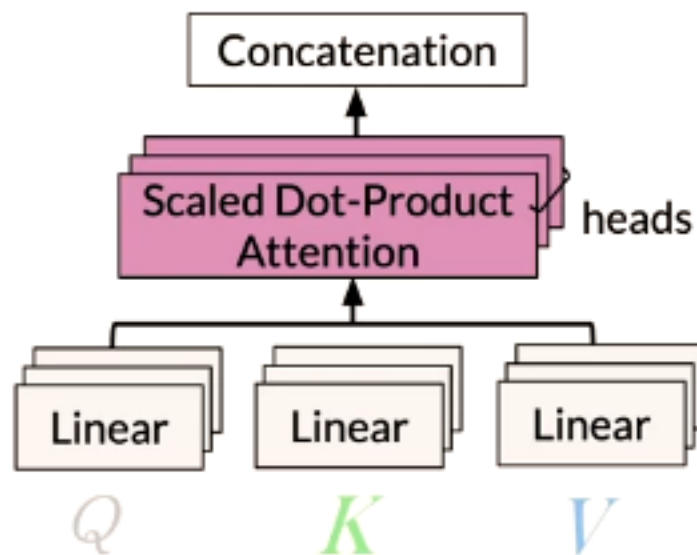
Multi-Head Attention - Overview



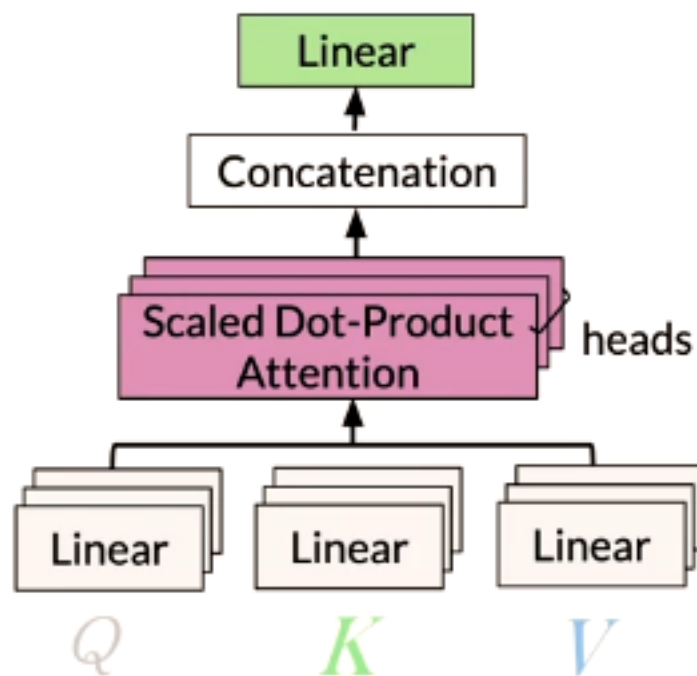
Multi-Head Attention - Overview



Multi-Head Attention - Overview



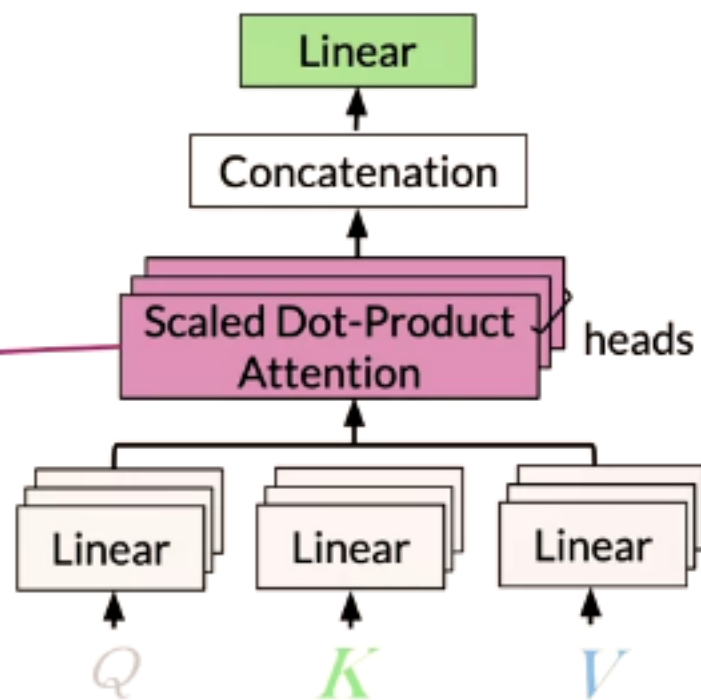
Multi-Head Attention - Overview



Multi-Head Attention - Scaled dot product

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

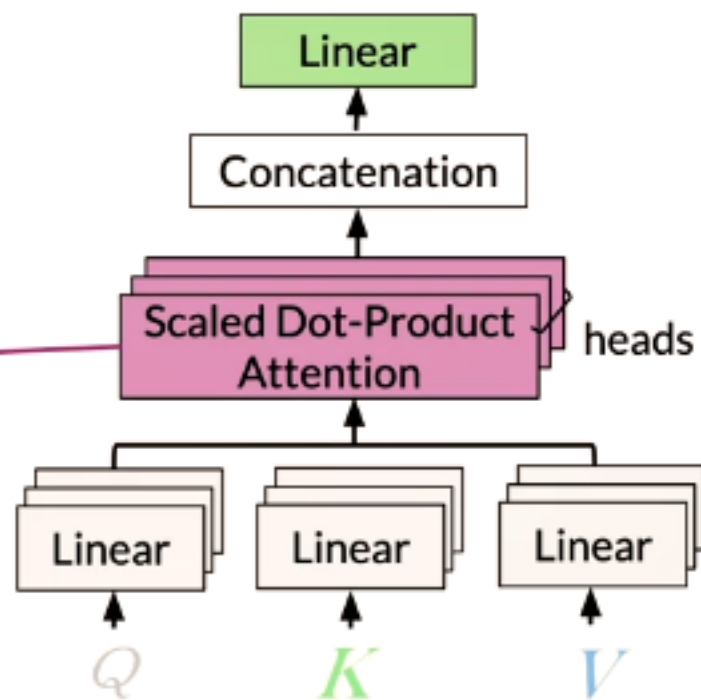
Query and key
dimension



Multi-Head Attention - Scaled dot product

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

Query and key
dimension



Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]



Multi-Head Attention - Concatenation

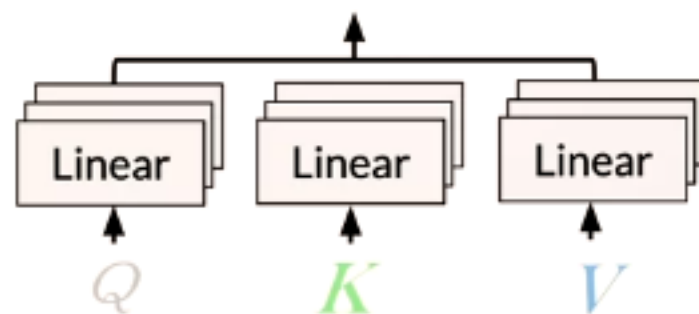
- Input(Q, K, V): [batch, length, d_model]
512, 1024

↑ ↑ ↑

Q *K* *V*

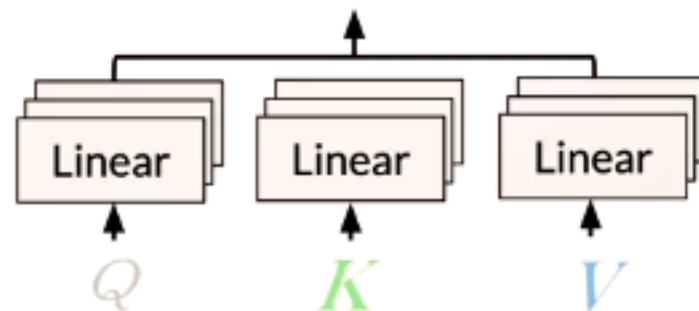
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]



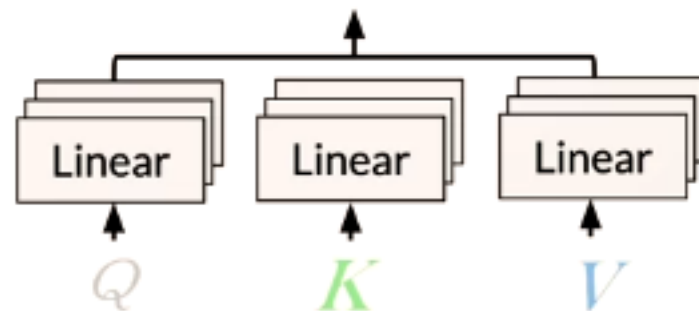
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]



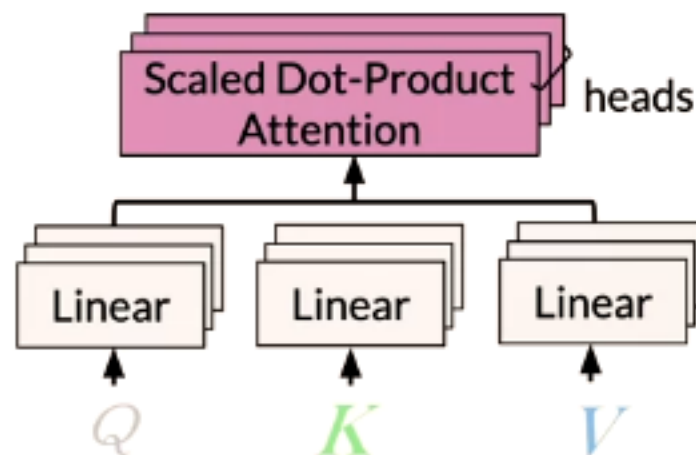
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]
- Transpose: [batch, n_heads, length, d_head]



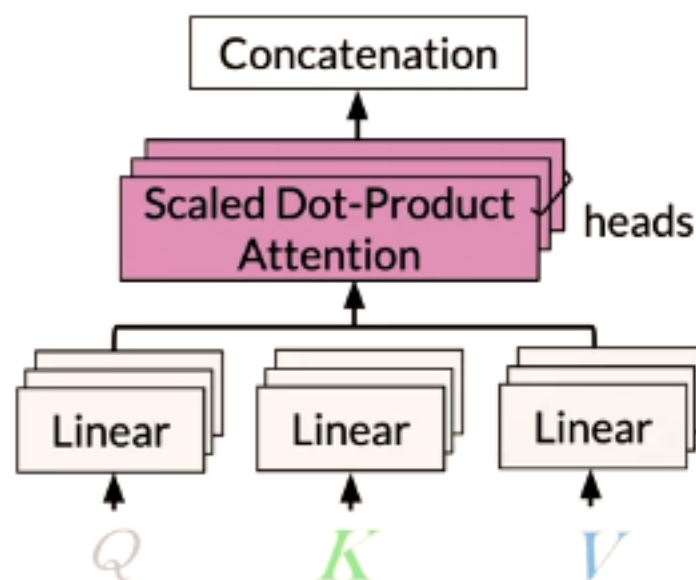
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]
- Transpose: [batch, n_heads, length, d_head]
- Apply attention treating n_heads like batch



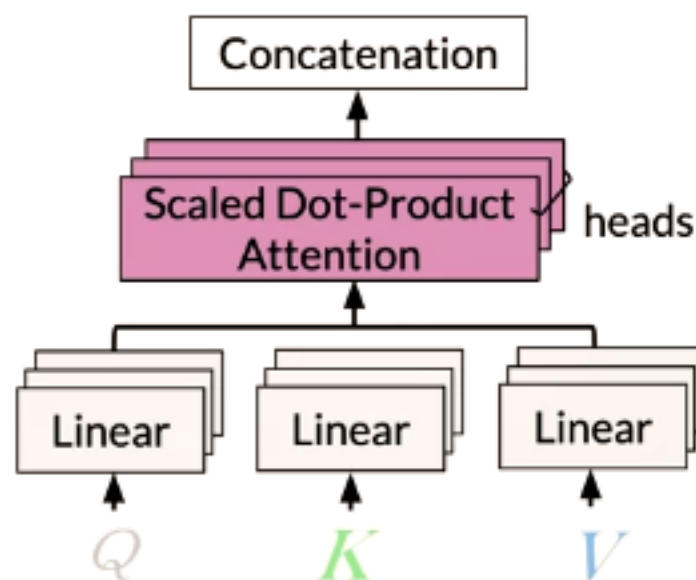
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]
- Transpose: [batch, n_heads, length, d_head]
- Apply attention treating n_heads like batch
- Result shape: [batch, n_heads, length, d_head]



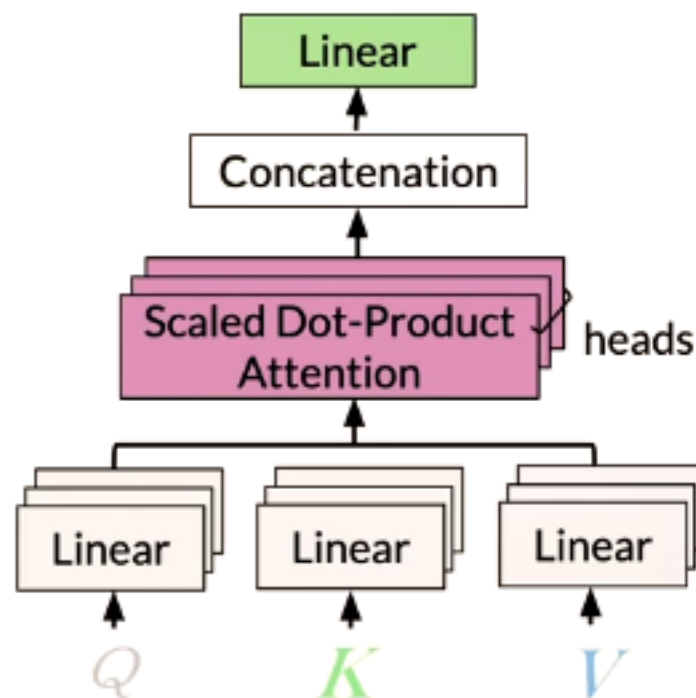
Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]
- Transpose: [batch, n_heads, length, d_head]
- Apply attention treating n_heads like batch
- Result shape: [batch, n_heads, length, d_head]
- Transpose: [batch, length, n_heads * d_head]



Multi-Head Attention - Concatenation

- Input(Q, K, V): [batch, length, d_model]
- Linear layer: [batch, length, n_heads * d_head]
- Transpose: [batch, n_heads, length, d_head]
- Apply attention treating n_heads like batch
- Result shape: [batch, n_heads, length, d_head]
- Transpose: [batch, length, n_heads * d_head]
- Linear layer into: [batch size, length, d_model]



Multi-Head Attention math

Multi-Head Attention math

Embedding

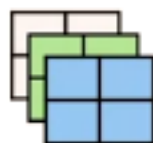


Multi-Head Attention math

Embedding

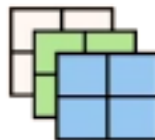


Q_i K_i V_i



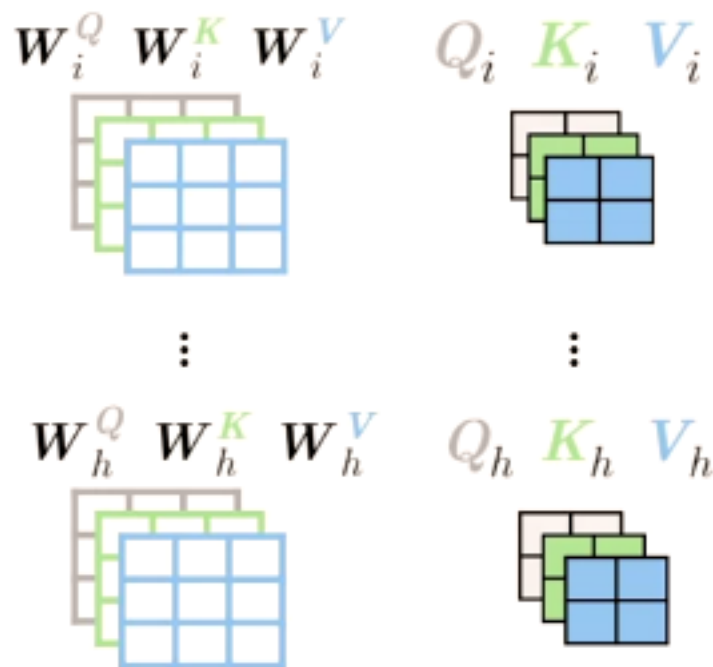
\vdots

Q_h K_h V_h



Multi-Head Attention math

Embedding

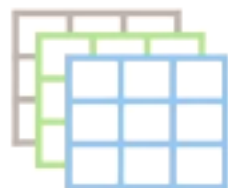


Multi-Head Attention math

Embedding

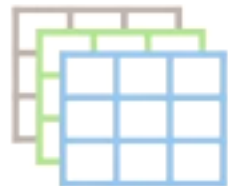


W_i^Q W_i^K W_i^V

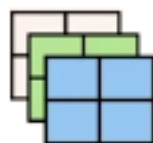


\vdots

W_h^Q W_h^K W_h^V

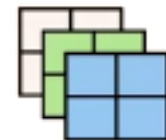


Q_i K_i V_i



\vdots

Q_h K_h V_h

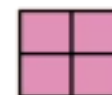


Z_i



\vdots

Z_h

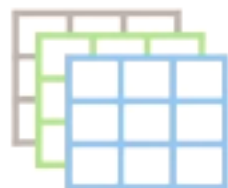


Multi-Head Attention math

Embedding

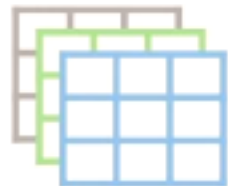


W_i^Q W_i^K W_i^V

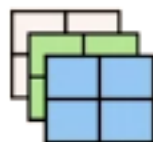


\vdots

W_h^Q W_h^K W_h^V

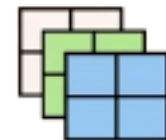


Q_i K_i V_i



\vdots

Q_h K_h V_h

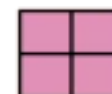


Z_i

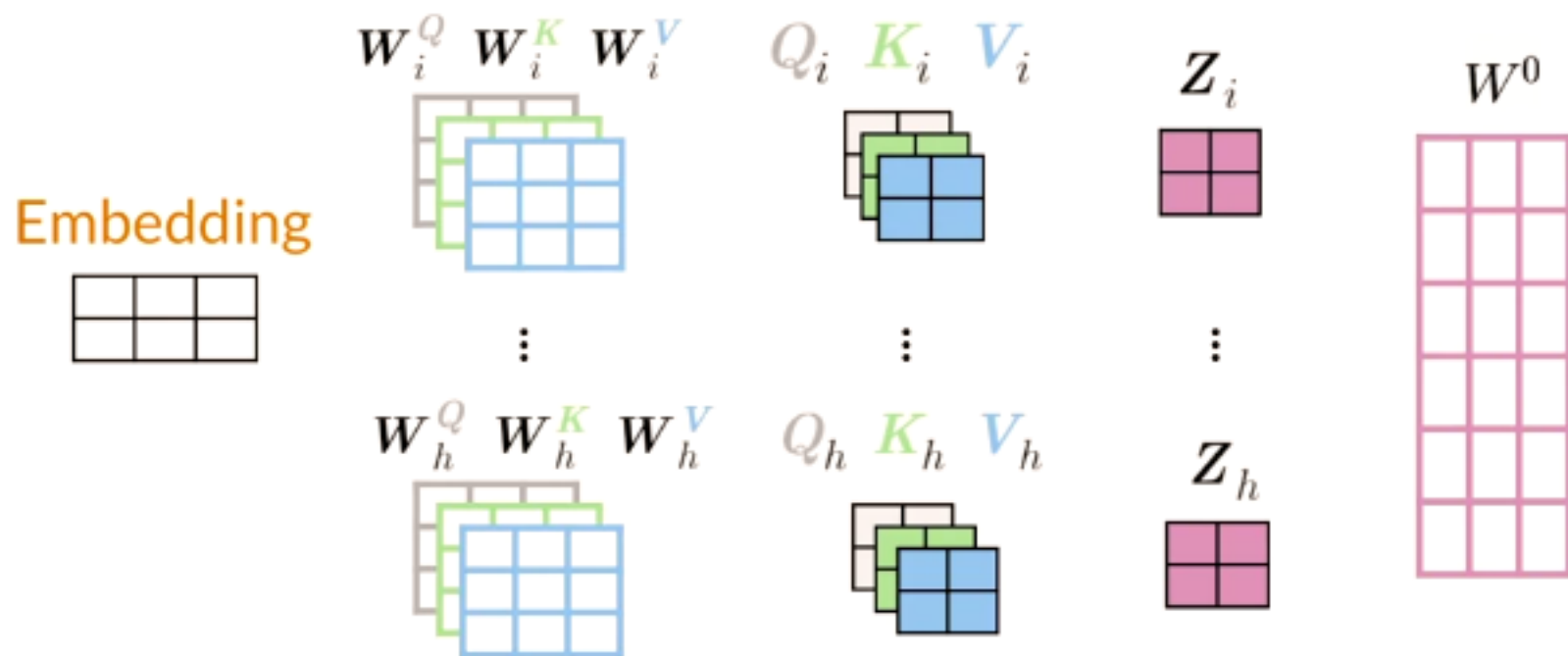


\vdots

Z_h



Multi-Head Attention math



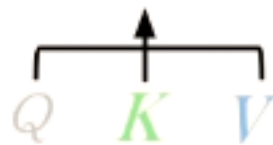
Multi-Head Attention Formula

Q K V

Q K V

Multi-Head Attention Formula

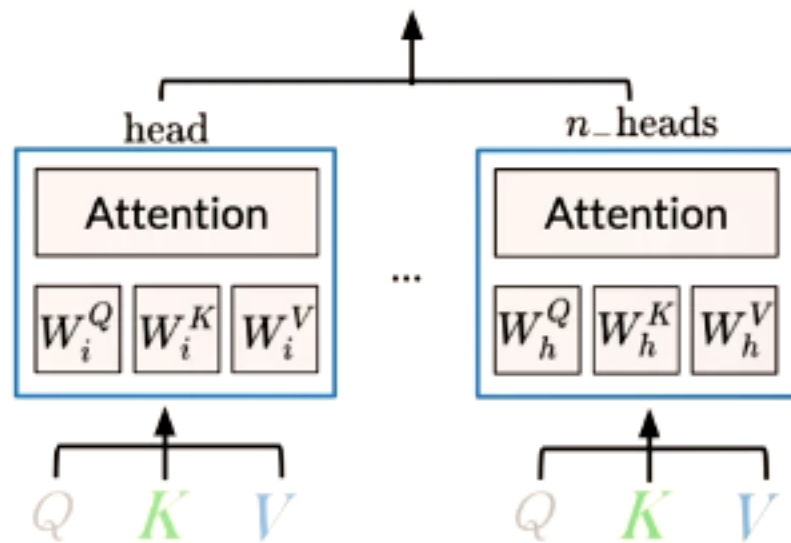
$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^0$$



Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^O$$

where $h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

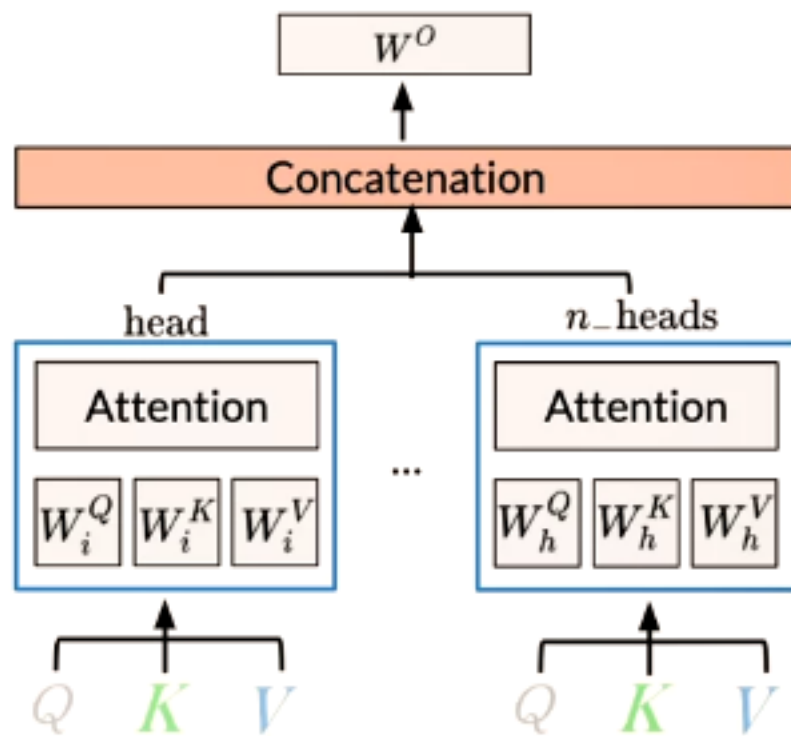


Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^O$$

$$\text{where } h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Each head h_i is the attention function of **Query**, **Key** and **Value** with trainable parameters (W_i^Q, W_i^K, W_i^V)

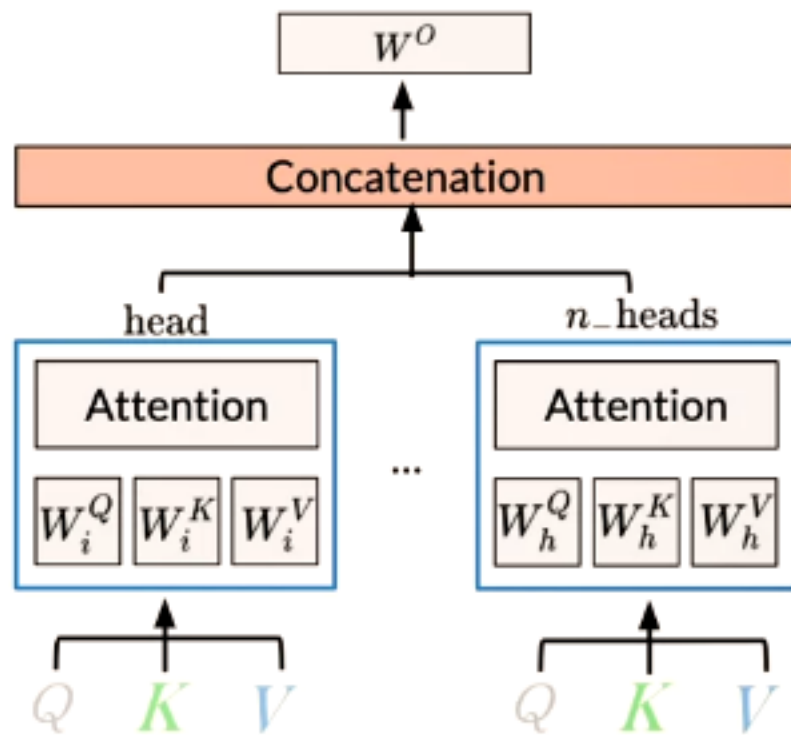


Multi-Head Attention Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \dots, h_h)W^O$$

$$\text{where } h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Each head h_i is the attention function of **Query**, **Key** and **Value** with trainable parameters (W_i^Q, W_i^K, W_i^V)



Summary

- Different heads can learn different relationship between words
- Scaled dot-product is adequate for Multi-Head Attention
- Multi-Headed models attend to information from different representations at different positions



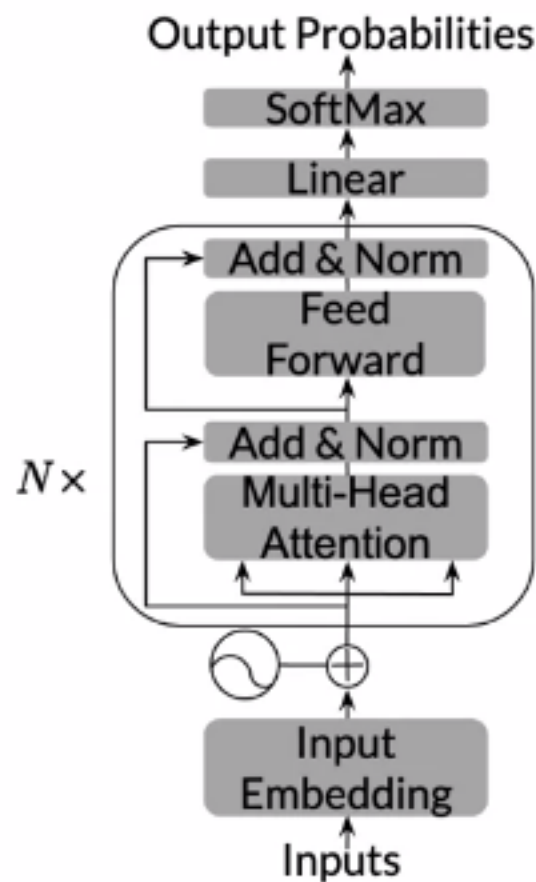
Outline

- Overview of Transformer decoder
- Implementation (decoder and feed-forward block)

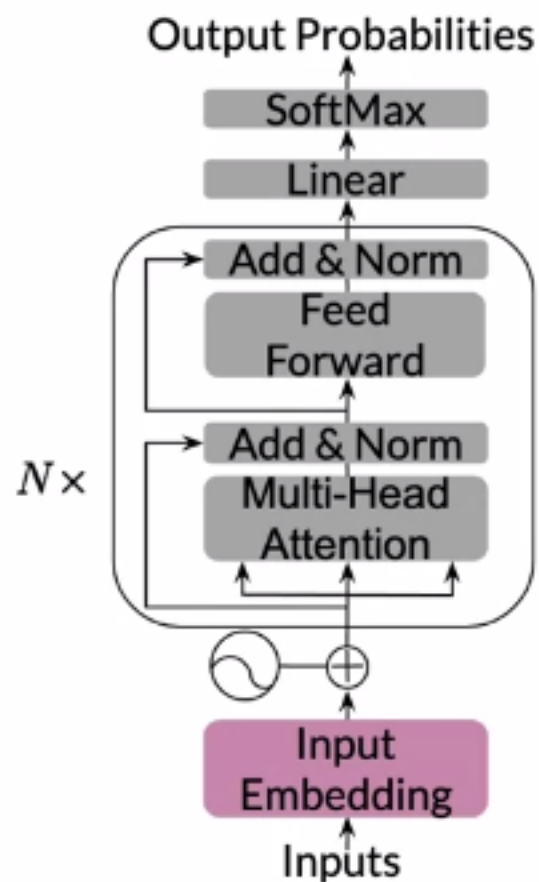


Transformer decoder

Overview



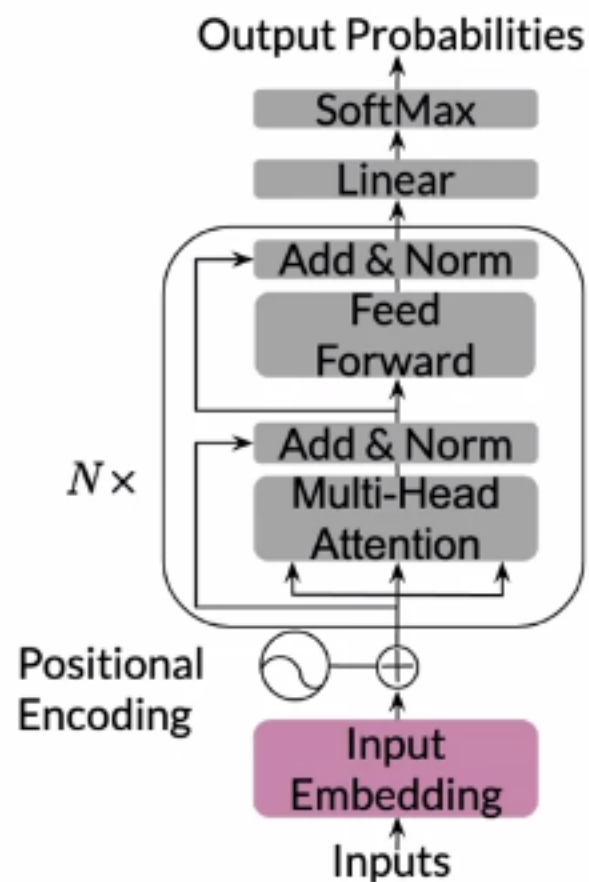
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word

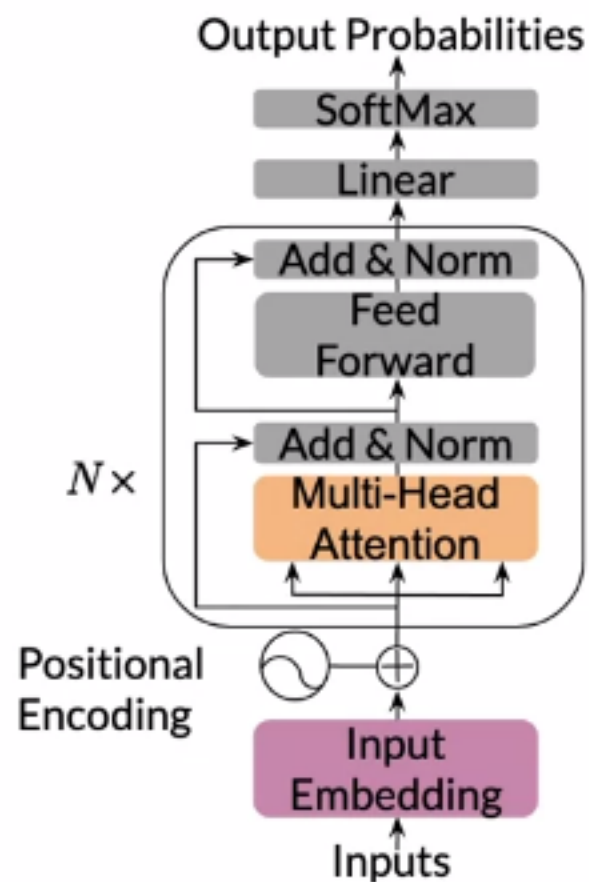
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)

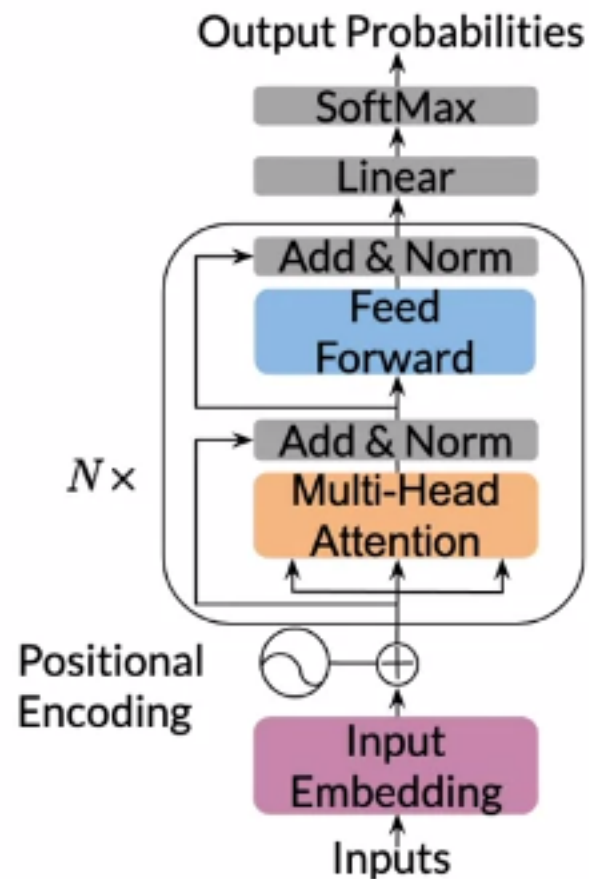
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words

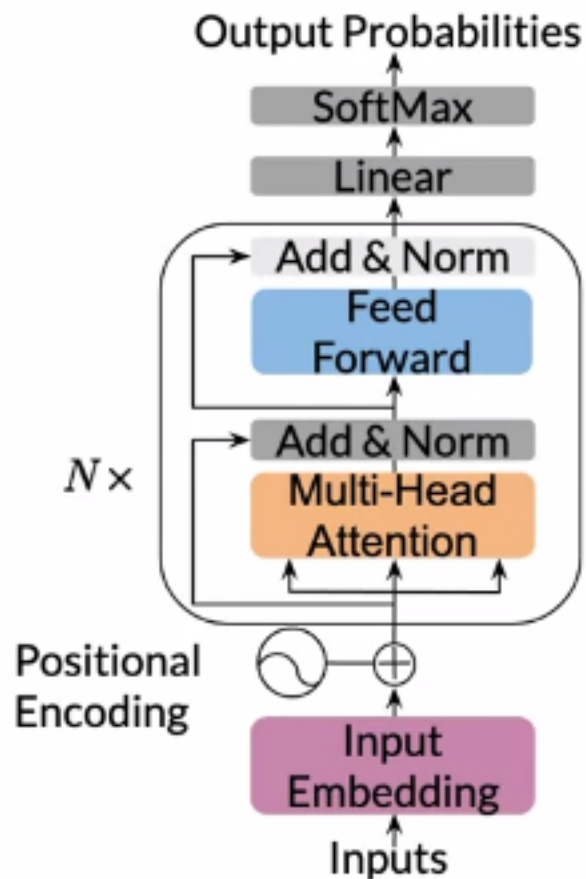
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
 - that's where most parameters are!

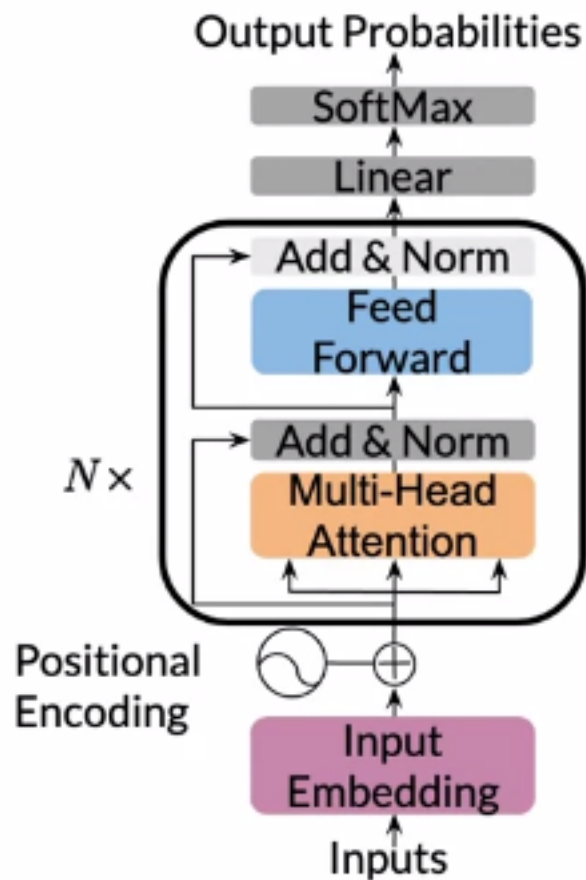
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
 - that's where most parameters are!
- residual connection with layer normalization

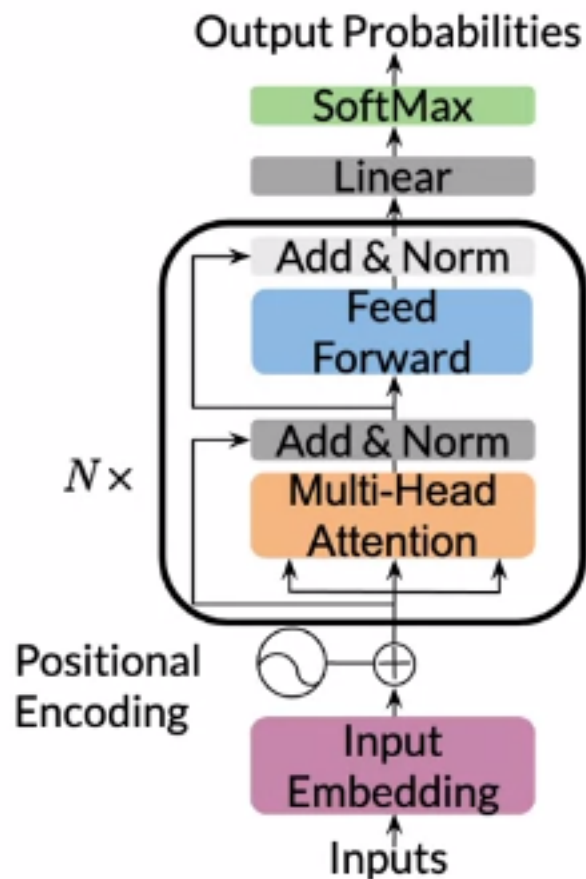
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
 - that's where most parameters are!
- residual connection with layer normalization
- repeat N times

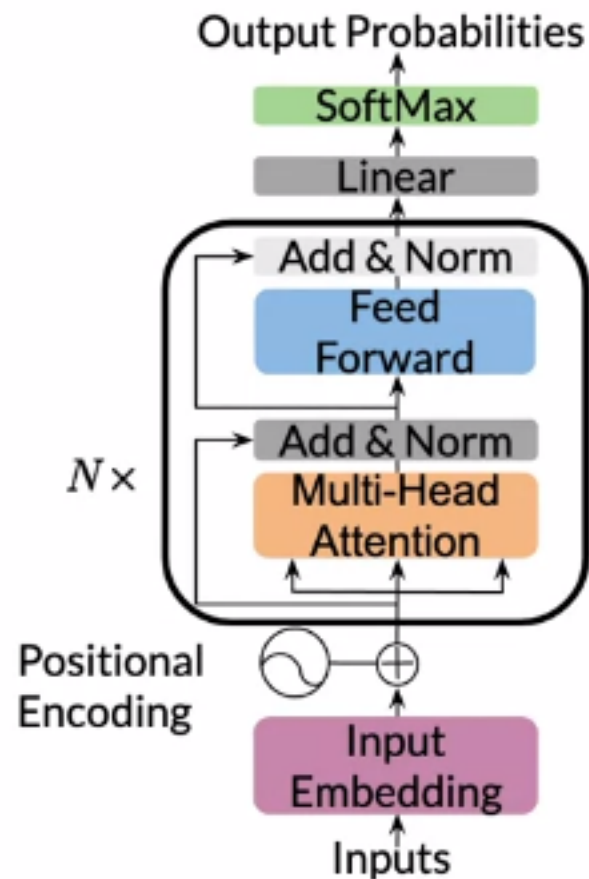
Transformer decoder



Overview

- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
 - that's where most parameters are!
- residual connection with layer normalization
- repeat N times
- dense layer and softmax for output

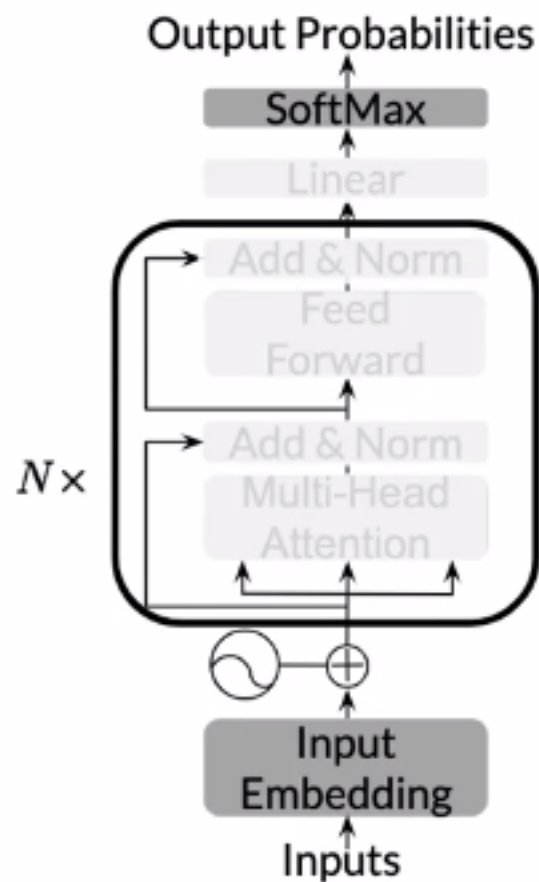
Transformer decoder



Overview

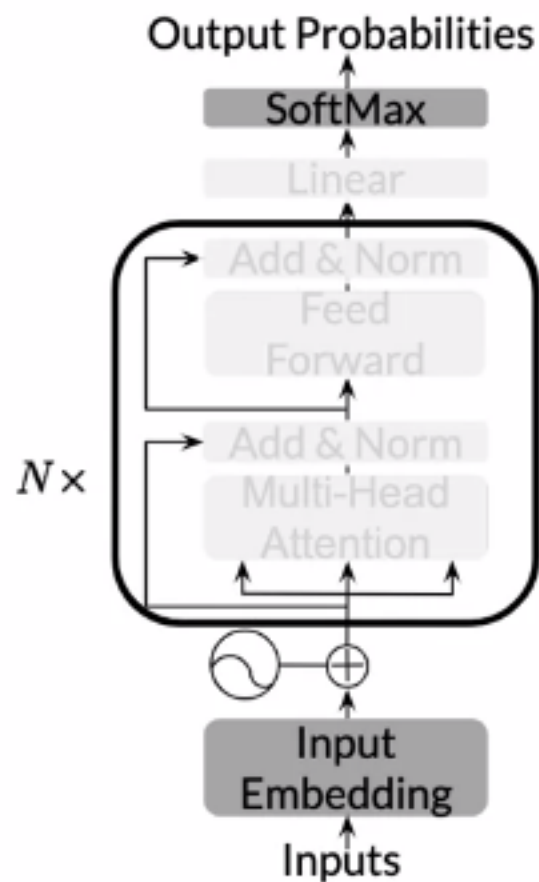
- input: sentence or paragraph
 - we predict the next word
- sentence gets embedded, add positional encoding
 - (vectors representing $\{0, 1, 2, \dots, K\}$)
- multi-head attention looks at previous words
- feed-forward layer with ReLU
 - that's where most parameters are!
- residual connection with layer normalization
- repeat N times
- dense layer and softmax for output

Transformer decoder



Explanation

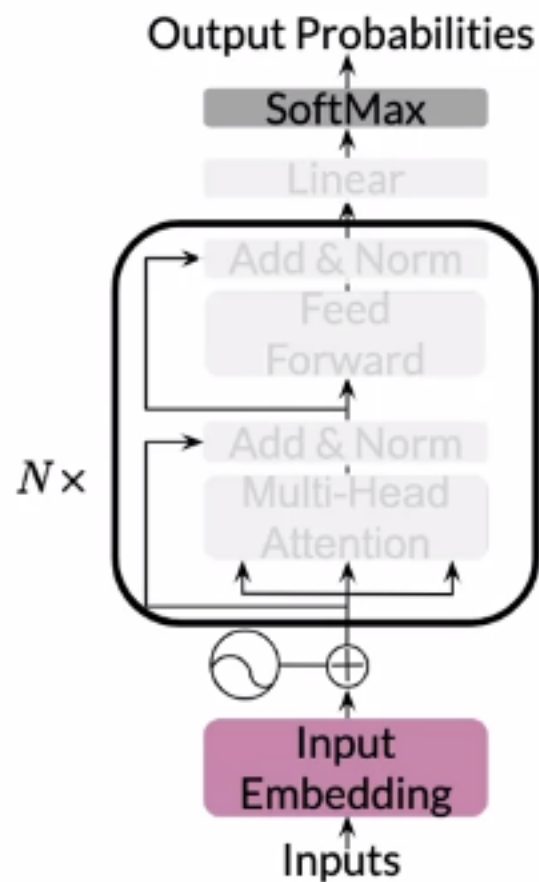
Transformer decoder



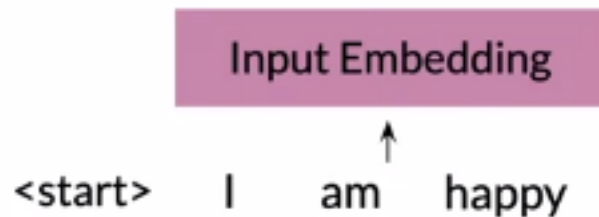
Explanation

<start> I am happy

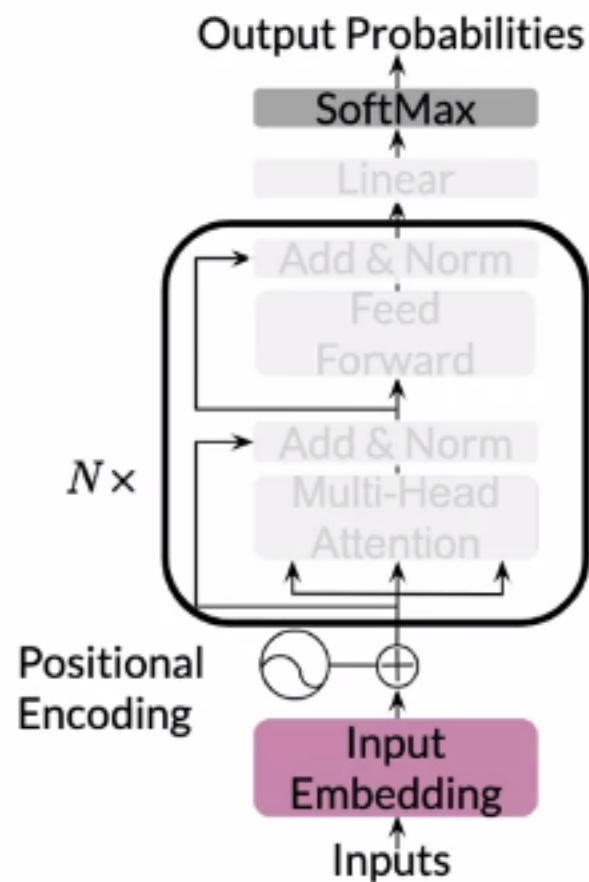
Transformer decoder



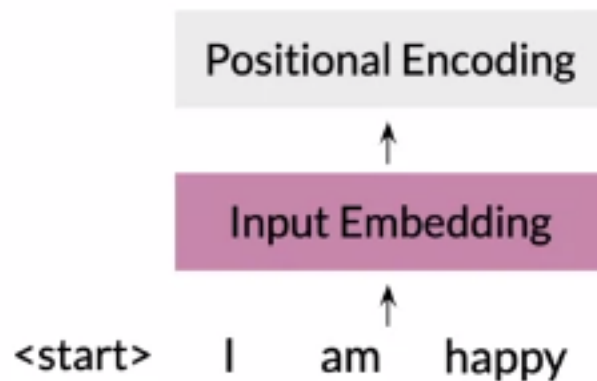
Explanation



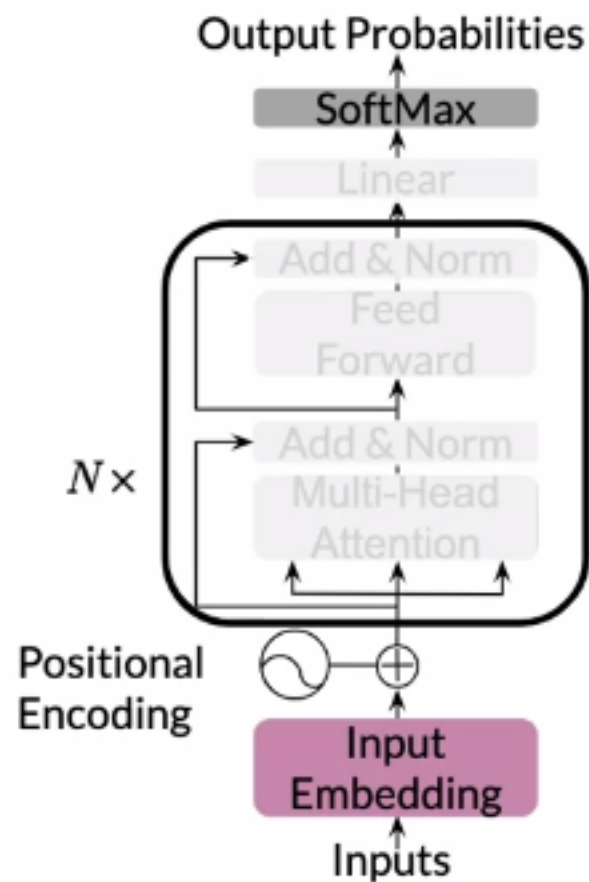
Transformer decoder



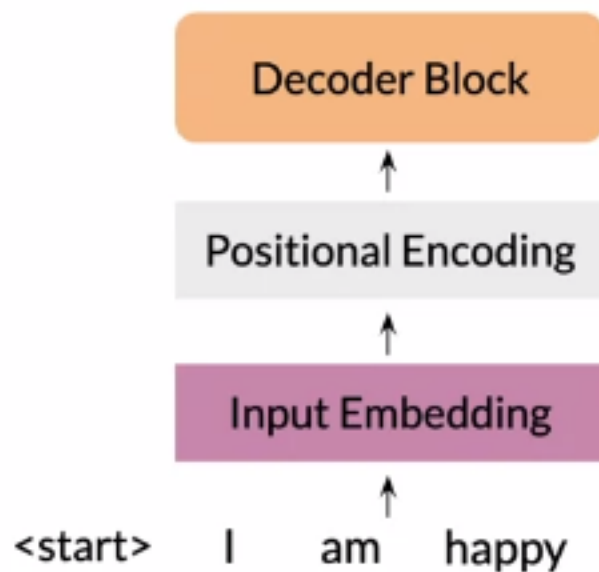
Explanation



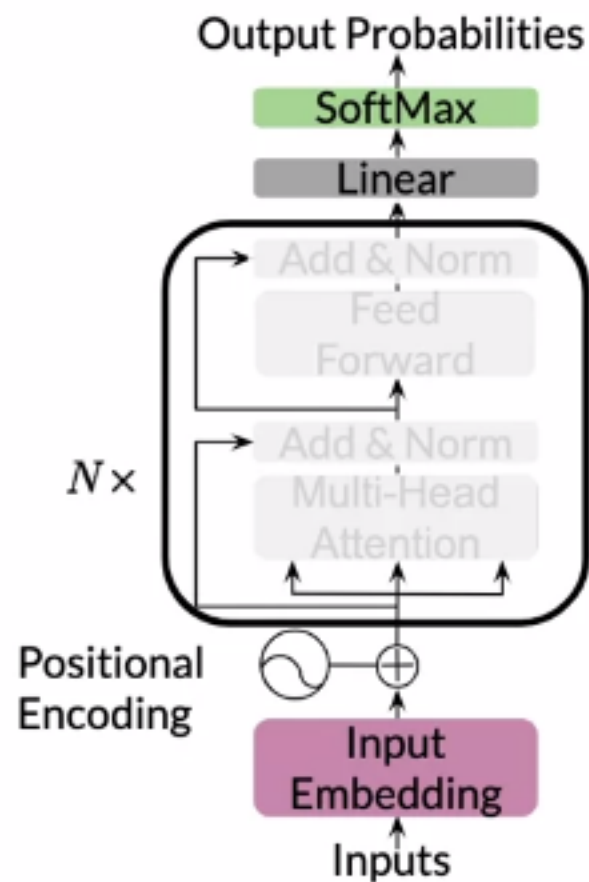
Transformer decoder



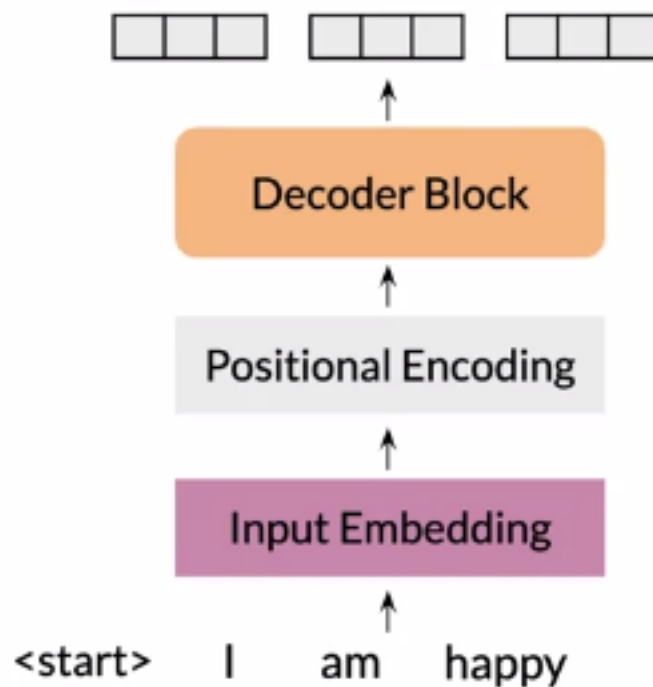
Explanation



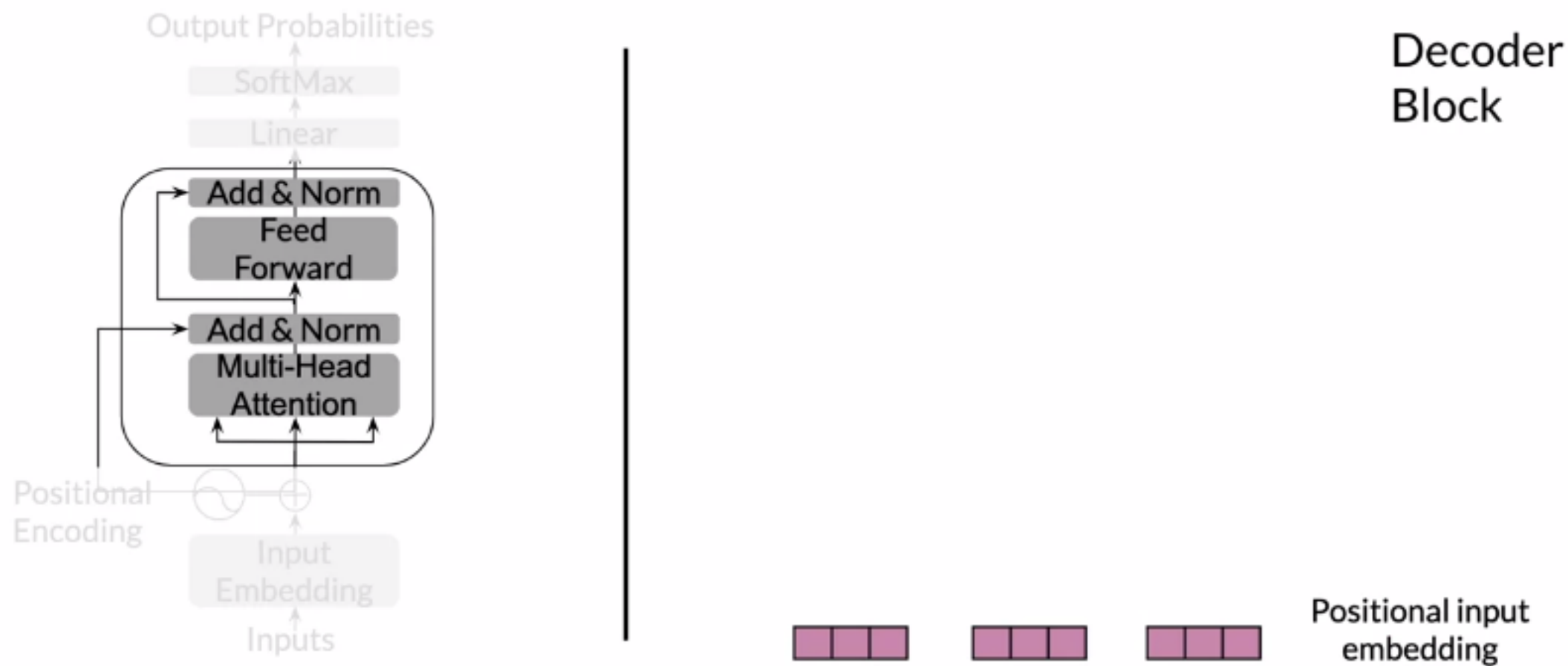
Transformer decoder



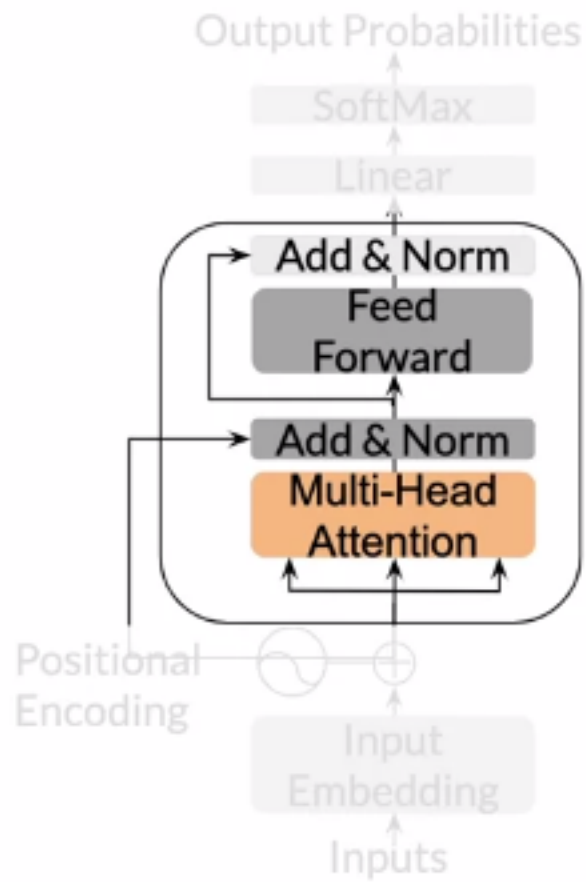
Explanation



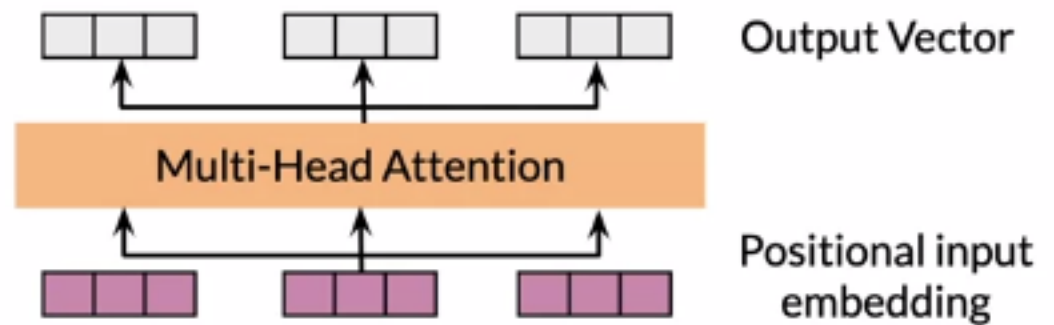
The Transformer decoder



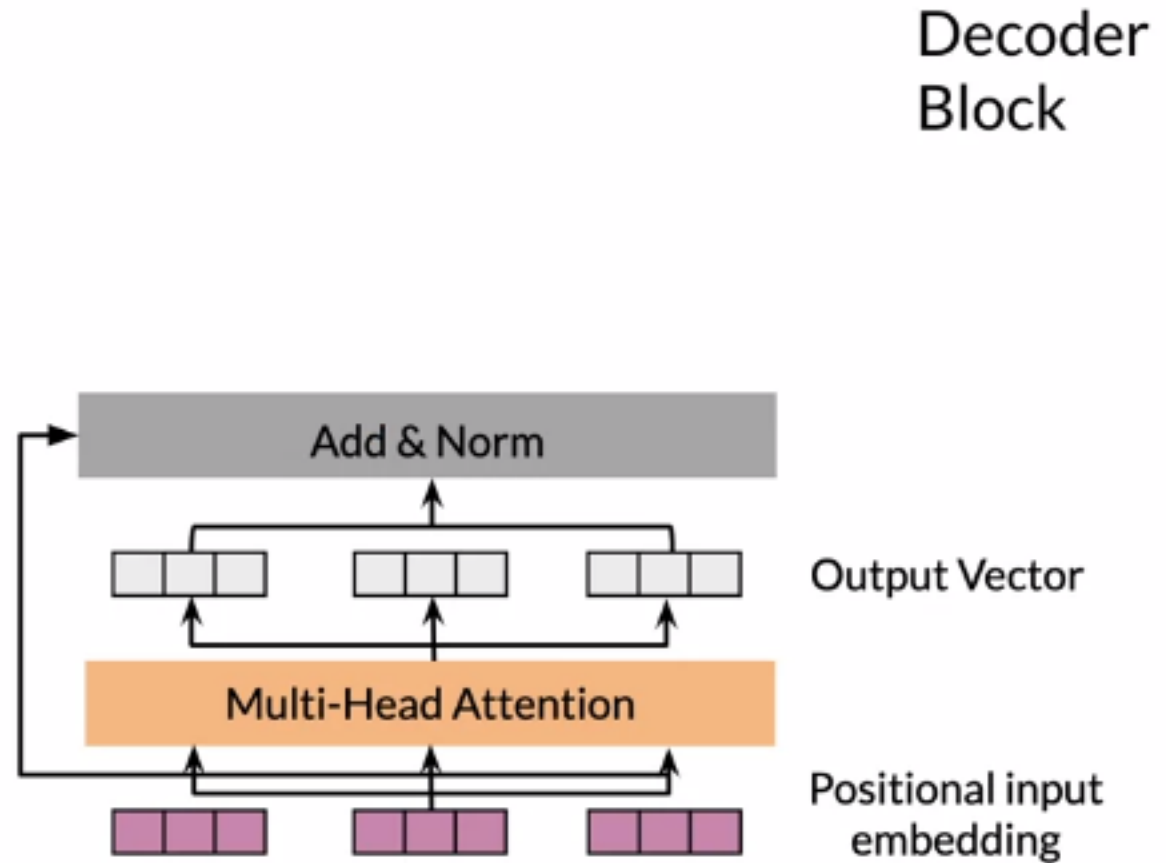
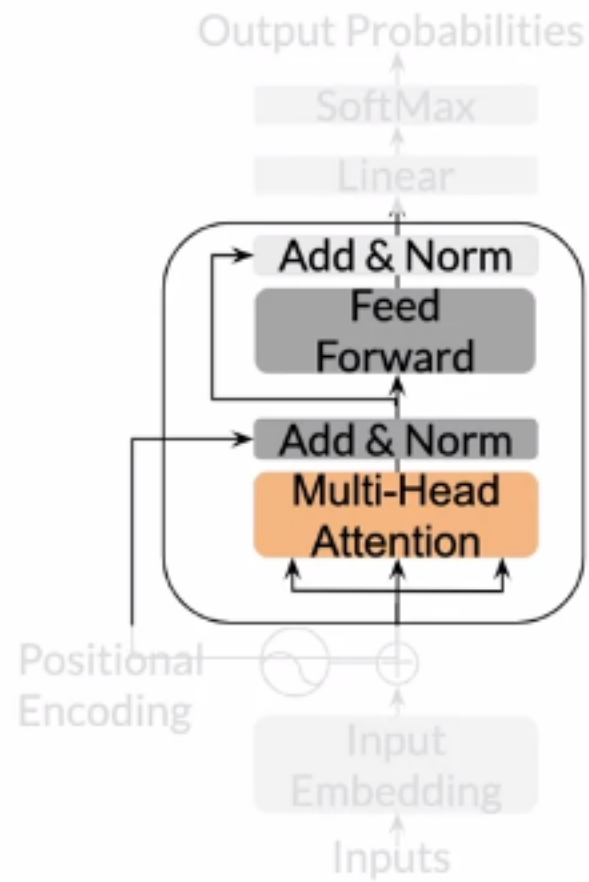
The Transformer decoder



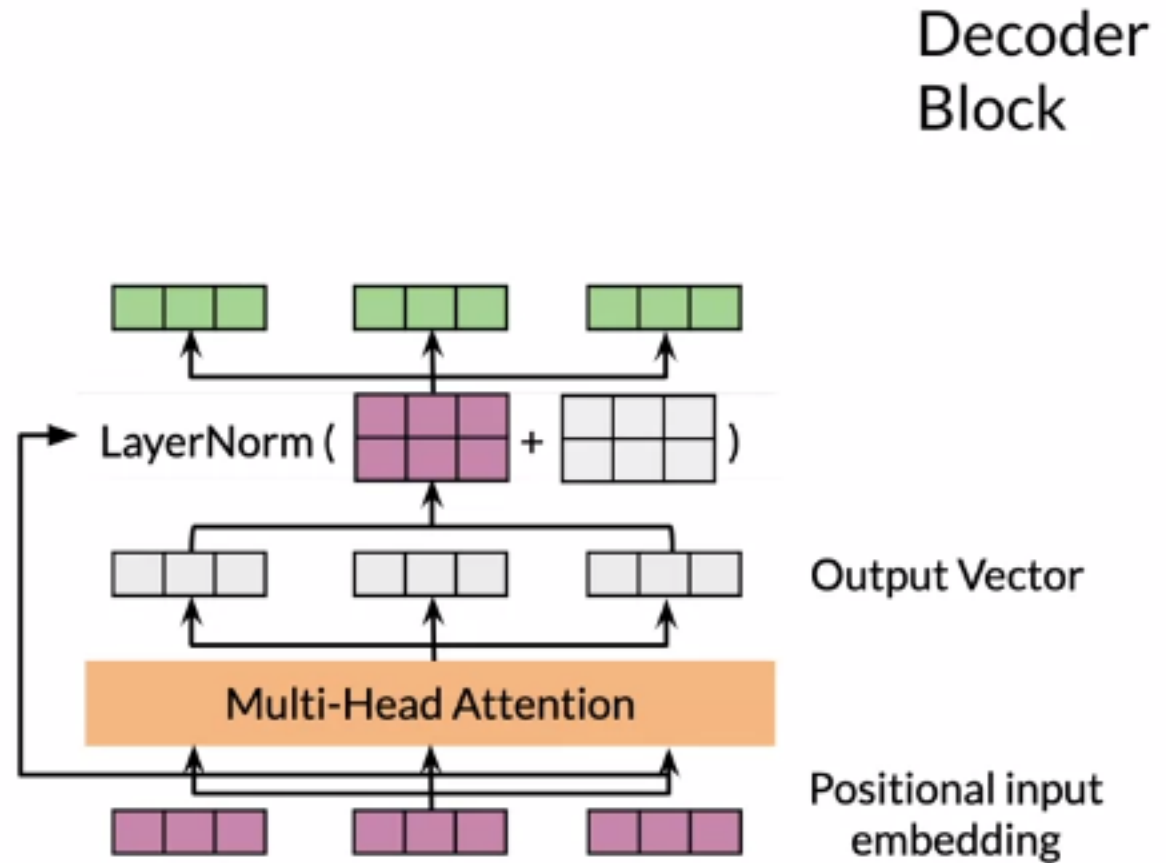
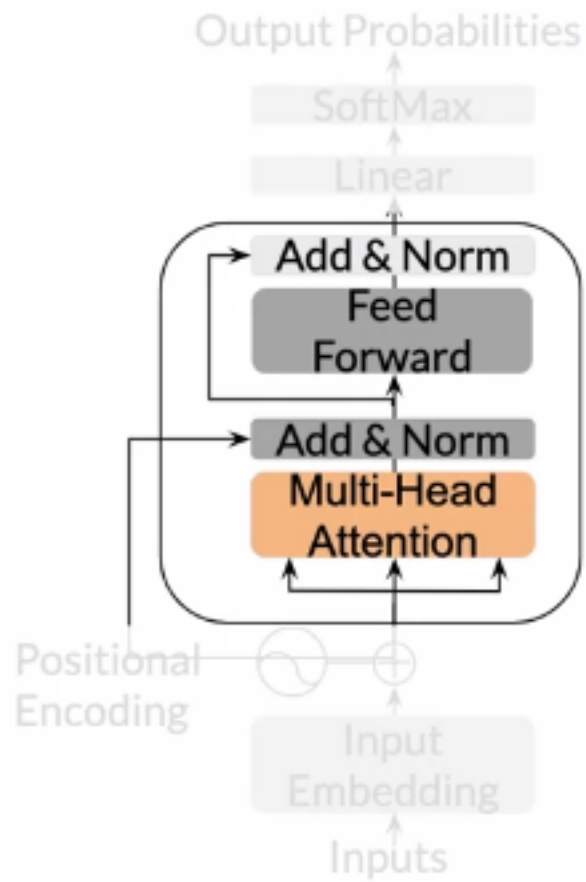
Decoder
Block



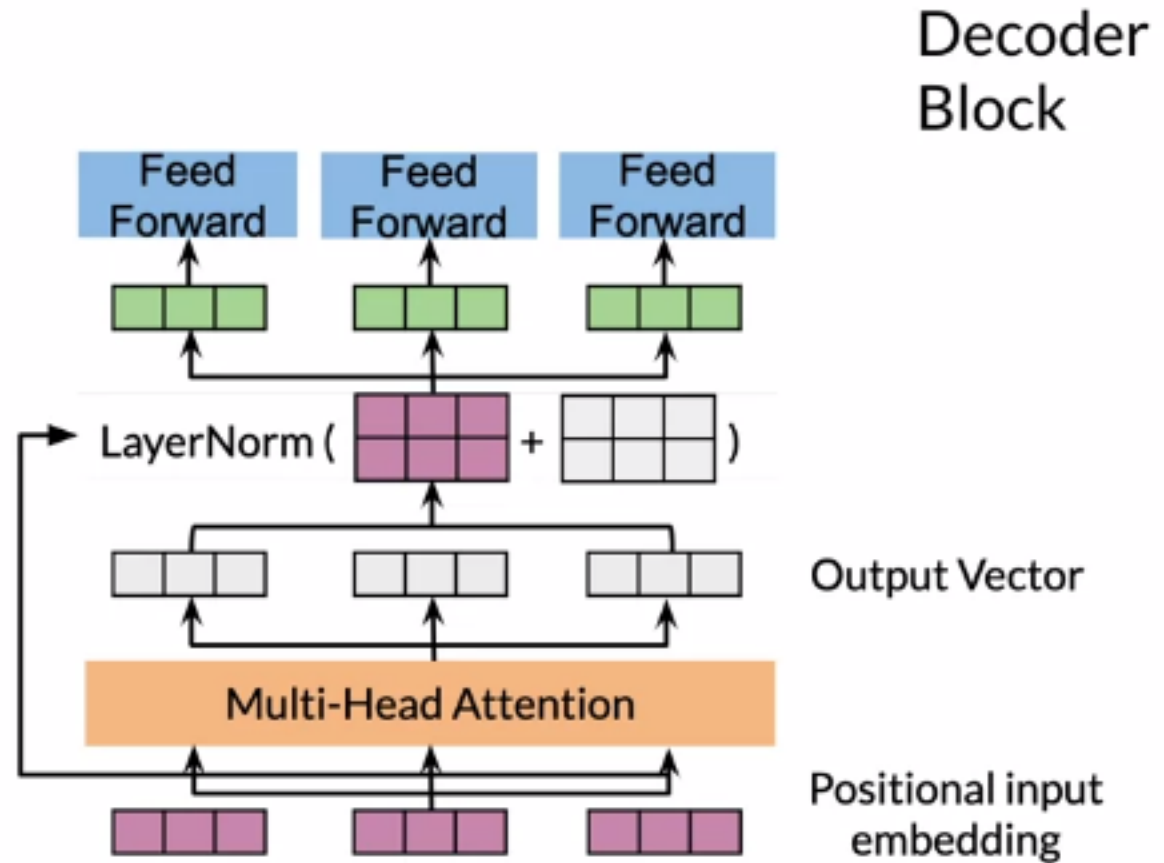
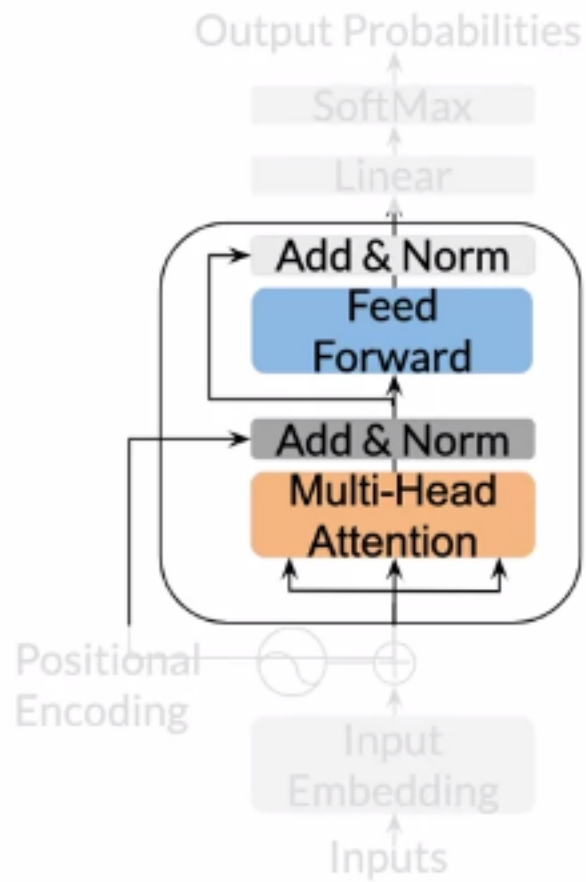
The Transformer decoder



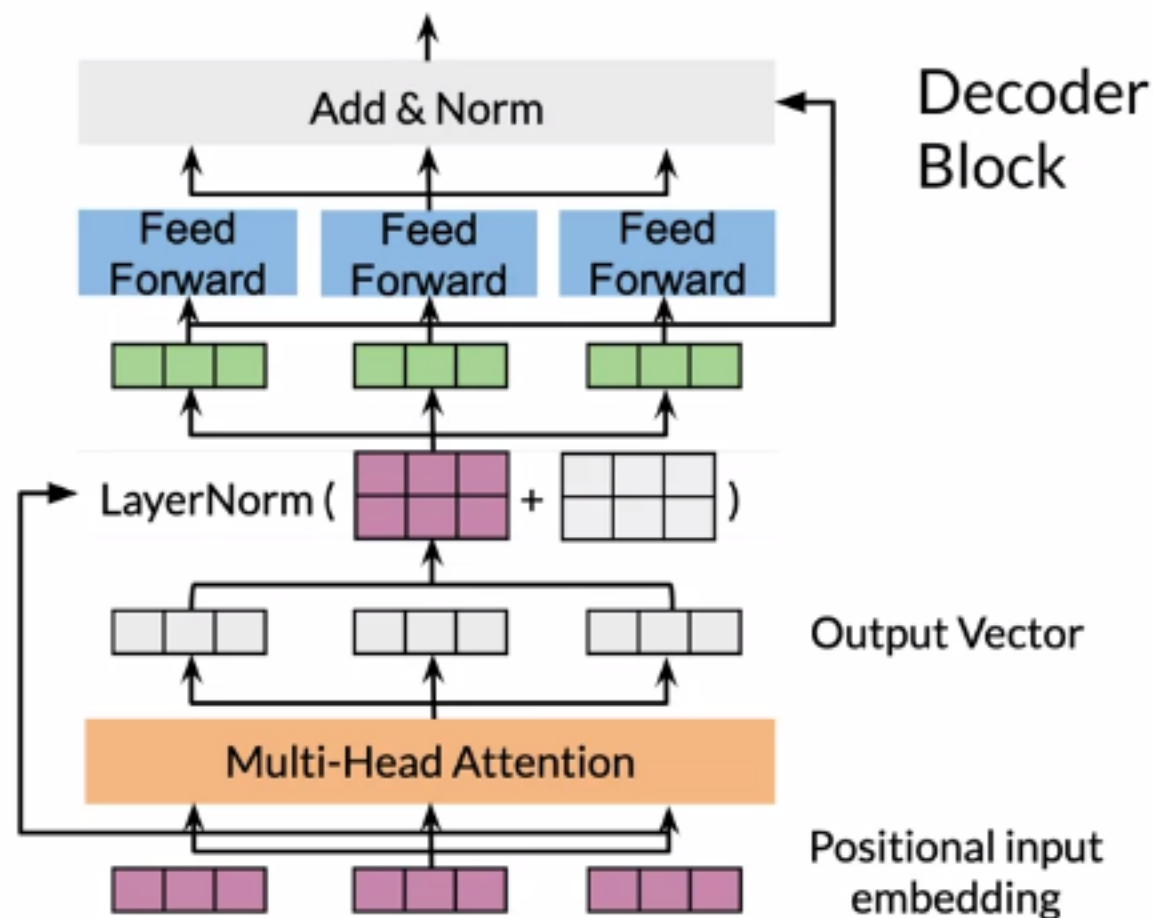
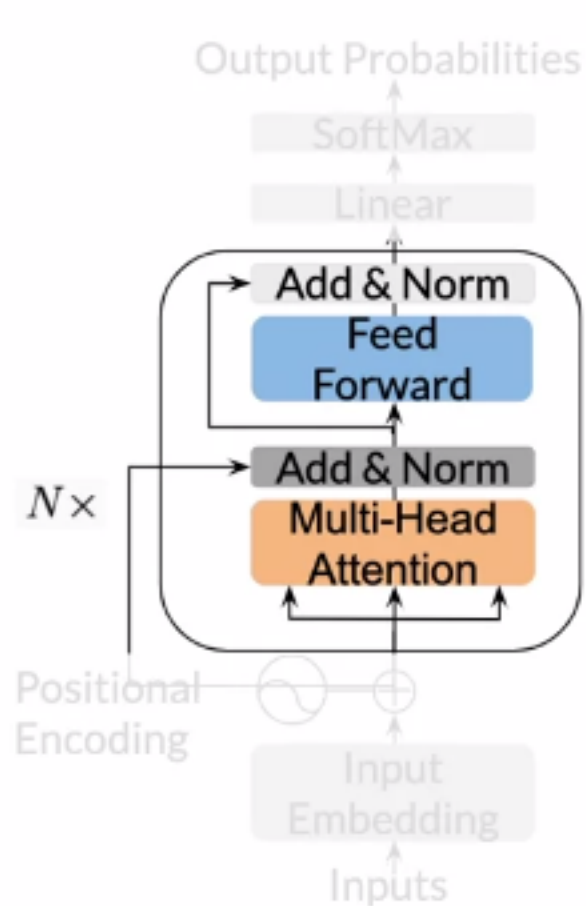
The Transformer decoder



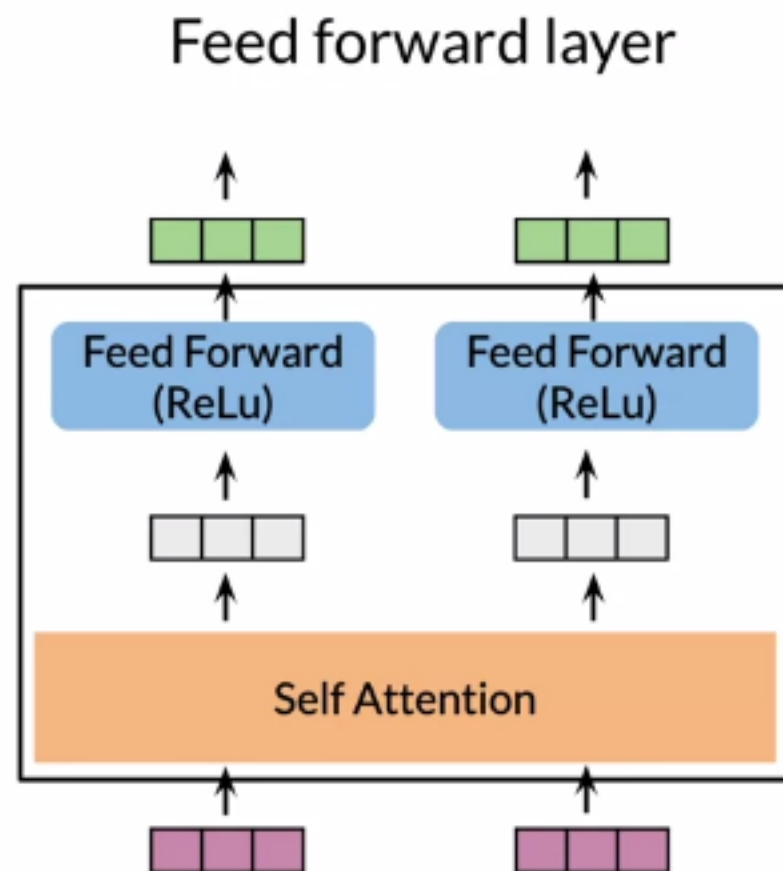
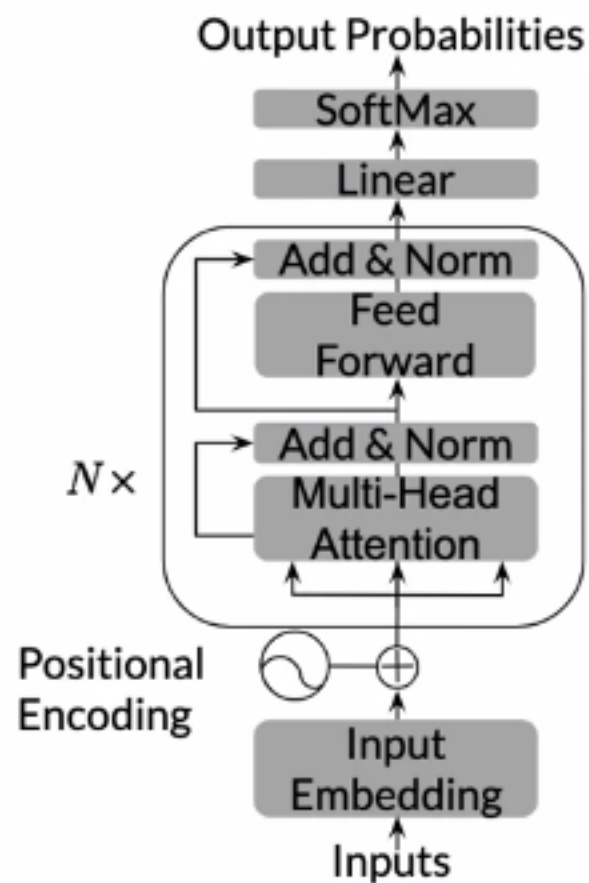
The Transformer decoder



The Transformer decoder



The Transformer decoder



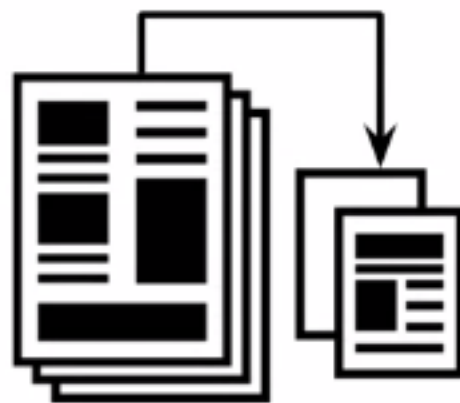
Summary

- Transformer decoder mainly consists of three layers
- Decoder and feed-forward blocks are the core of this model code
- It also includes a module to calculate the cross-entropy loss

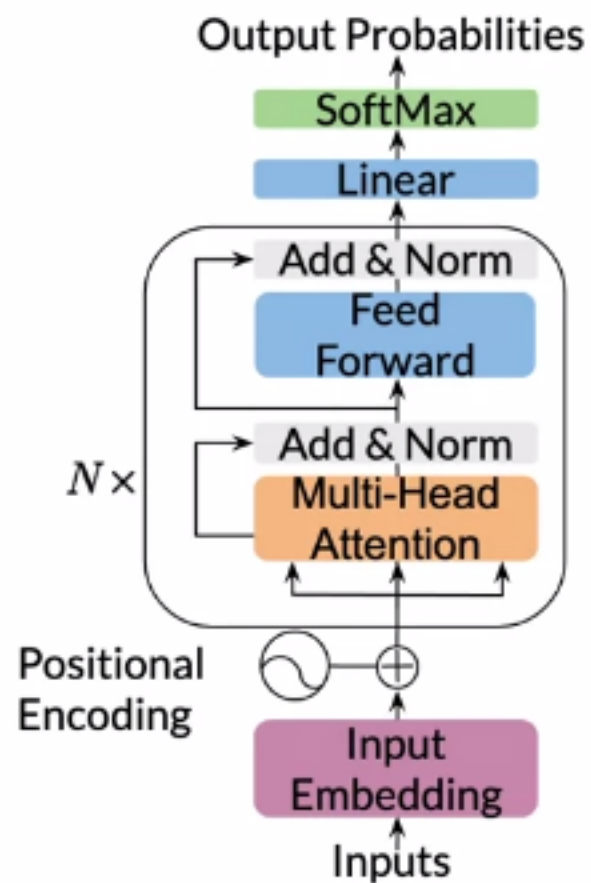


Outline

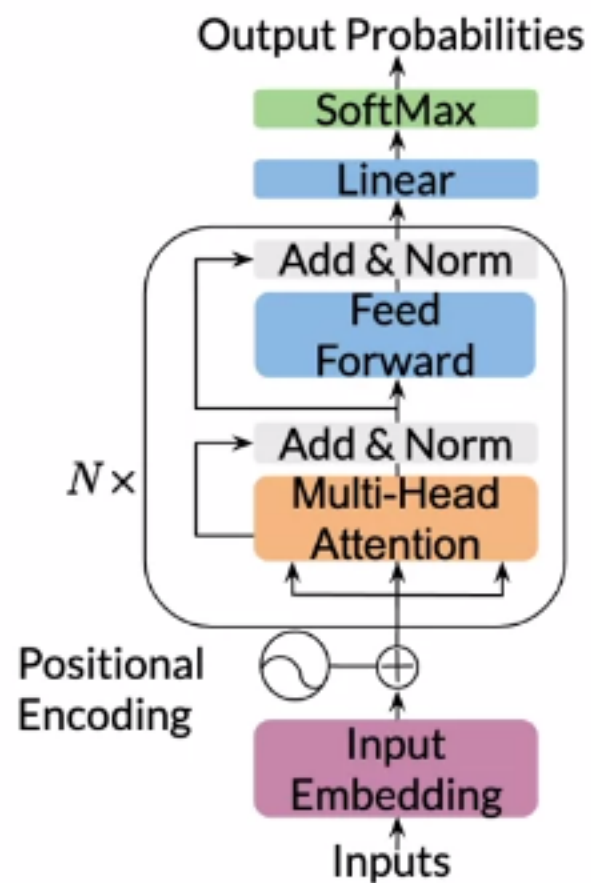
- Overview of Transformer summarizer
- Technical details for data processing
- Inference with a Language Model



Transformer for summarization



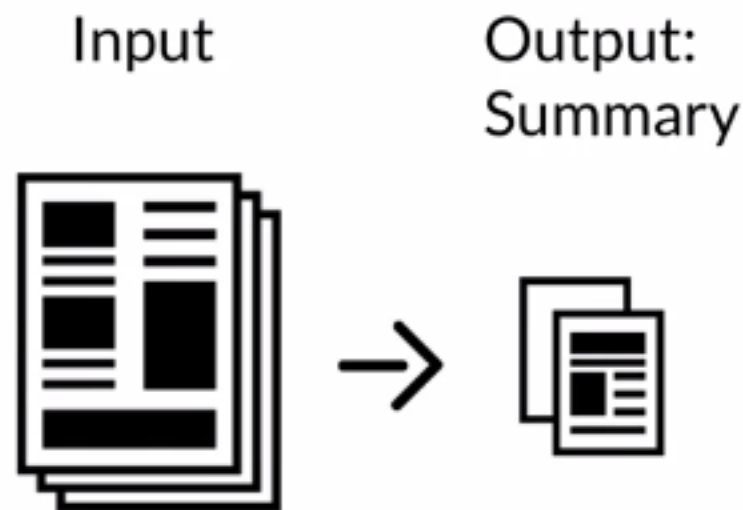
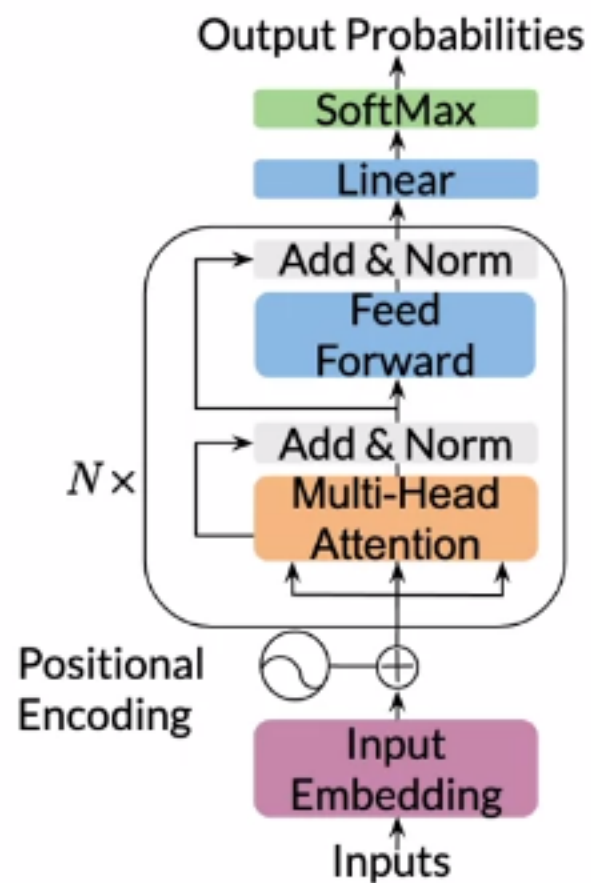
Transformer for summarization



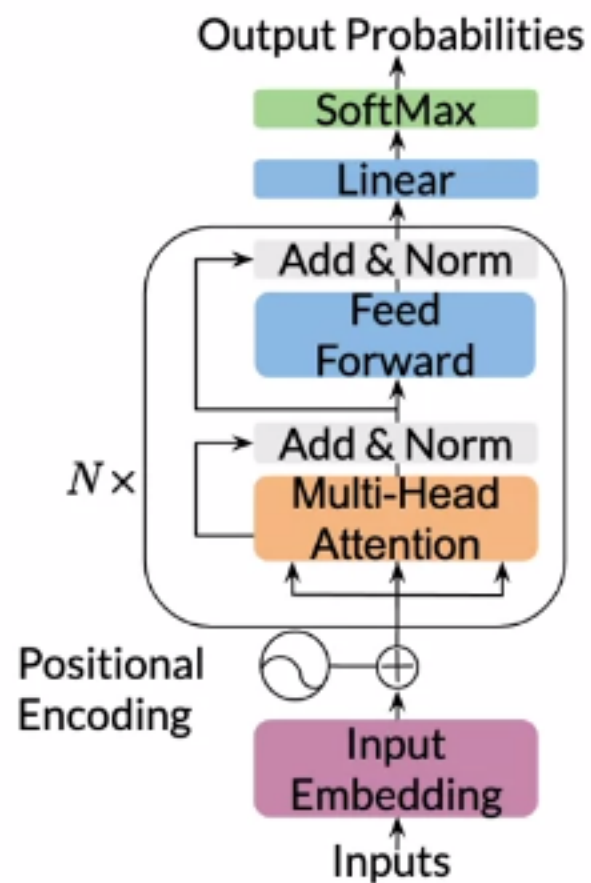
Input



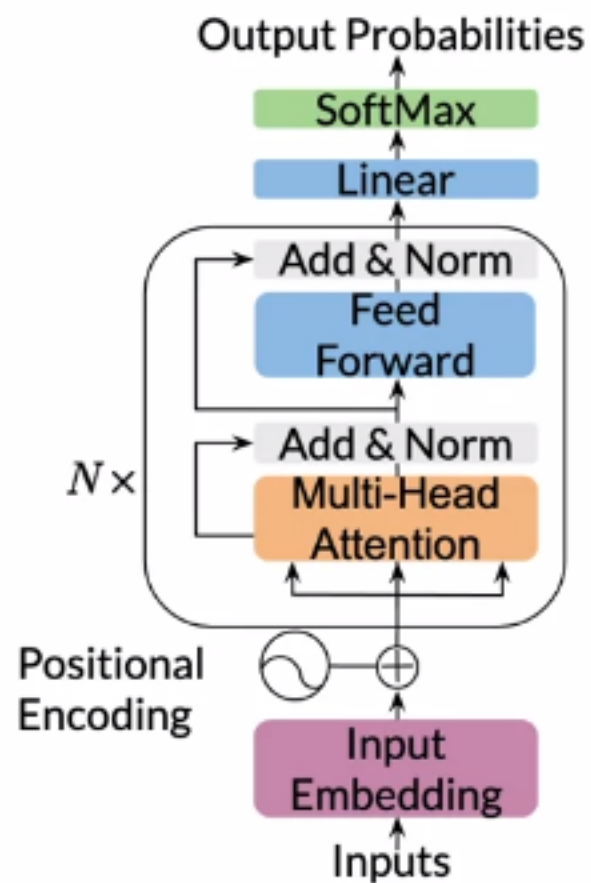
Transformer for summarization



Technical details for data processing



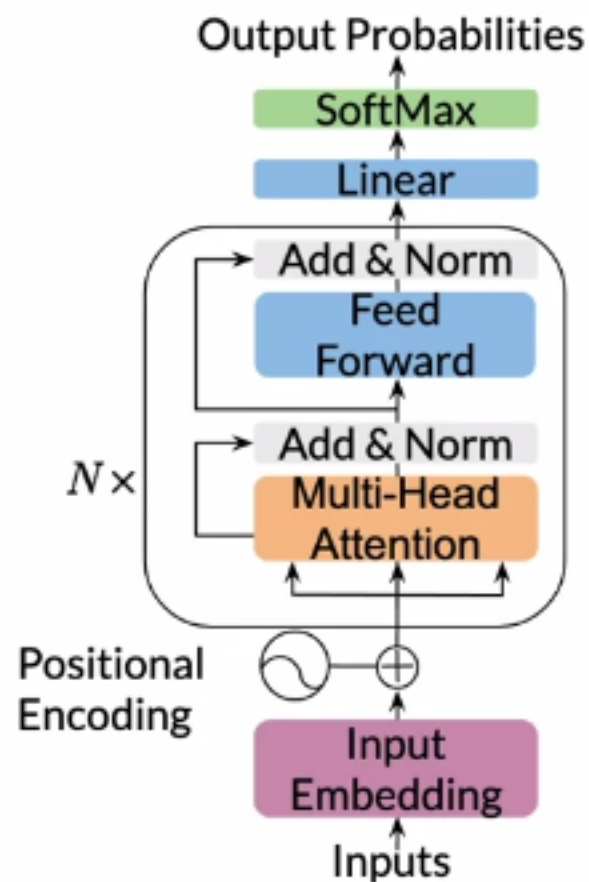
Technical details for data processing



Model Input:

ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

Technical details for data processing



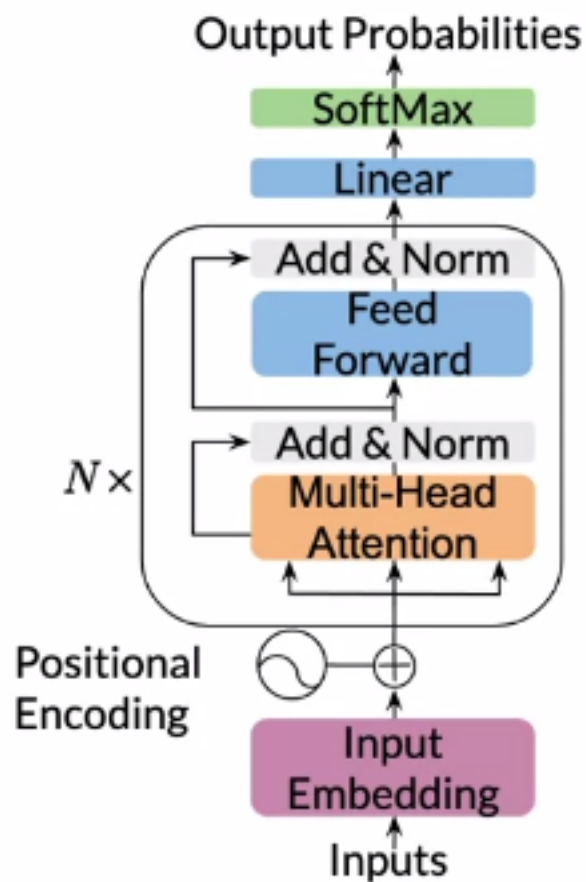
Model Input:

ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

Tokenized version:

[2,3,5,2,1,3,4,7,8,2,5,1,2,3,6,2,1,0,0]

Technical details for data processing



Model Input:

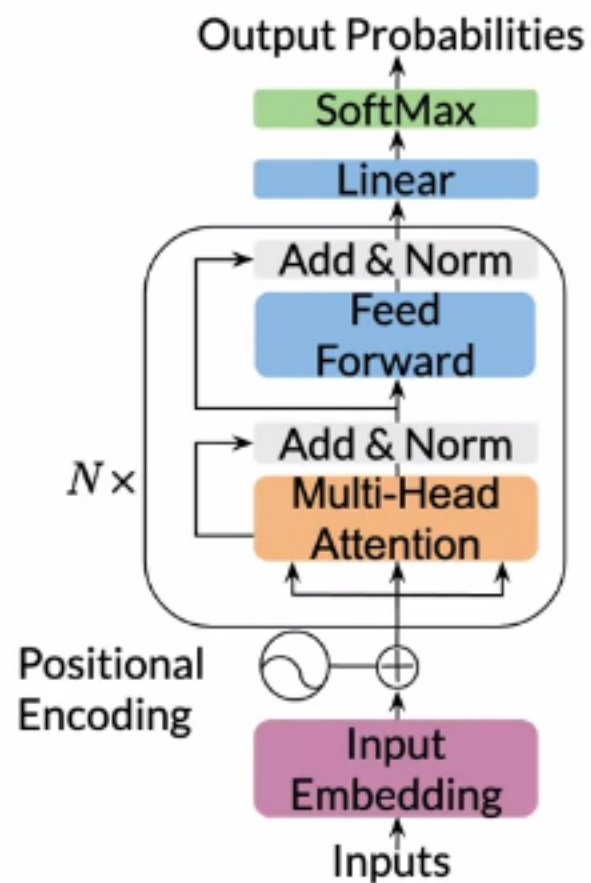
ARTICLE TEXT <EOS> SUMMARY <EOS> <pad> ...

Tokenized version:

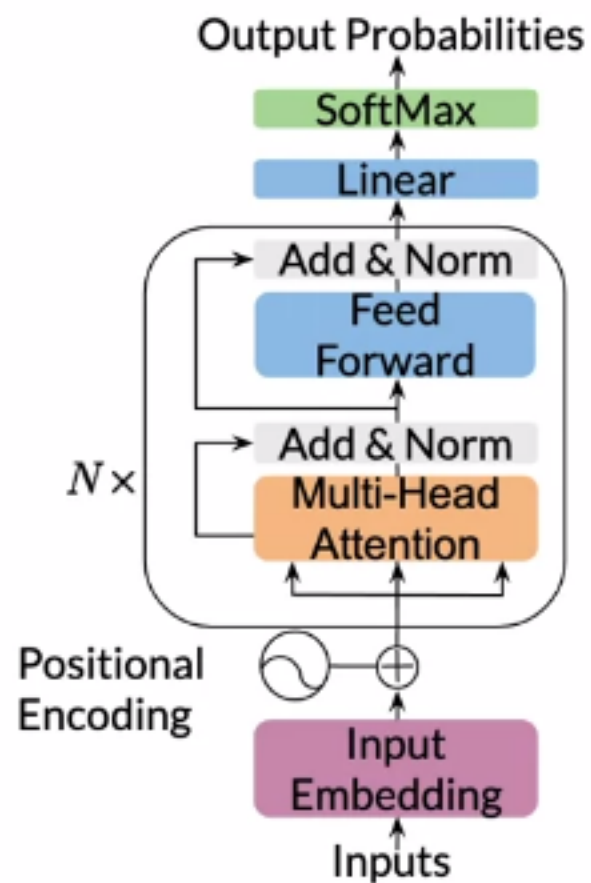
[2, 3, 5, 2, 1, 3, 4, 7, 8, 2, 5, 1, 2, 3, 6, 2, 1, 0, 0]

Loss weights: 0s until the first <EOS> and then 1 on the start of the summary.

Cost function



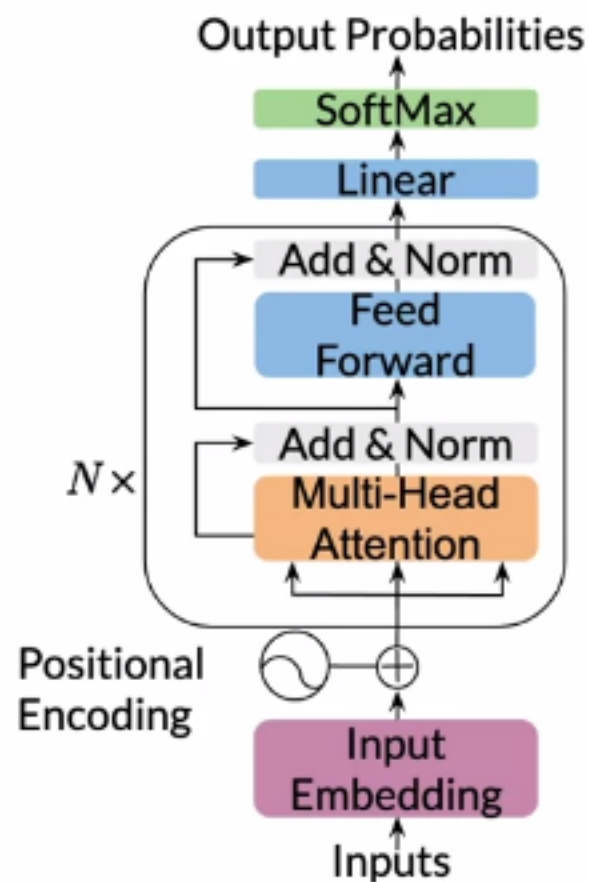
Cost function



$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$



Cost function



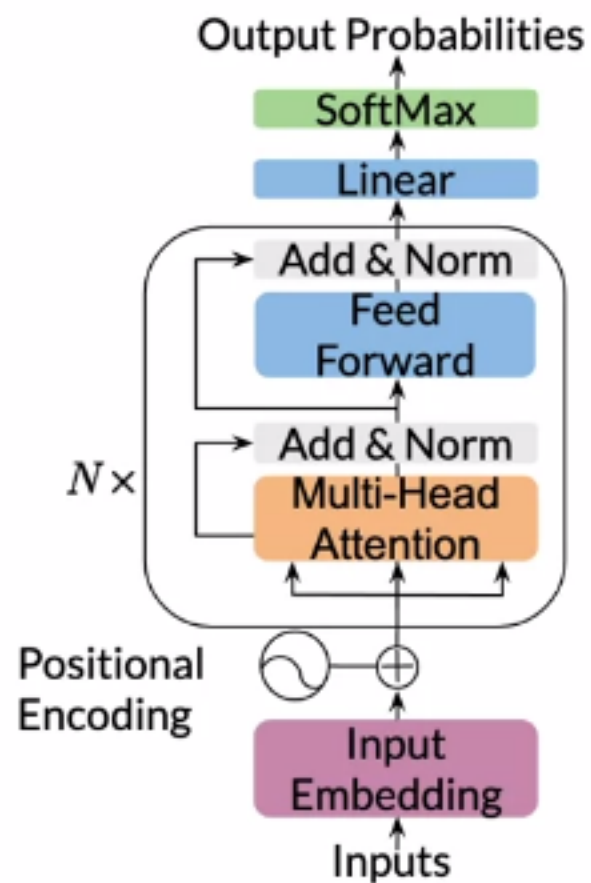
$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$

j : over summary

i : bach elements



Cost function



Cross entropy loss

$$J = -\frac{1}{m} \sum_j^m \sum_i^K y_j^i \log \hat{y}_j^i$$

j : over summary

i : batch elements



Inference with a Language Model

Inference with a Language Model

Model input:

[Article] <EOS> [Summary] <EOS>

Inference with a Language Model

Model input:

[Article] <EOS> [Summary] <EOS>

Inference:

Inference with a Language Model

Model input:

[Article] <EOS> [Summary] <EOS>

Inference:

- Provide: [Article] <EOS>

Inference with a Language Model

Model input:

[Article] <EOS> [Summary] <EOS>

Inference:

- Provide: [Article] <EOS>
- Generate summary word-by-word
 - until the final <EOS>

Inference with a Language Model

Model input:

[Article] <EOS> [Summary] <EOS>

Inference:

- Provide: [Article] <EOS>
- Generate summary word-by-word
 - until the final <EOS>
- Pick the next word by random sampling
 - each time you get a different summary!

Summary

- For summarization, a weighted loss function is optimized
- Transformer Decoder summarizes predicting the next word using
- The transformer uses tokenized versions of the input

