Transformer Basics

Danna Gurari
University of Colorado Boulder
Spring 2025

Review

- Last week:
  - Motivation: machine neural translation for long sentences
  - Encoder
  - Decoder: attention
  - Performance evaluation
  - Final project: ways to find a partner
- Assignments (Canvas):
  - Lab assignment 2 due Tuesday
- Questions?

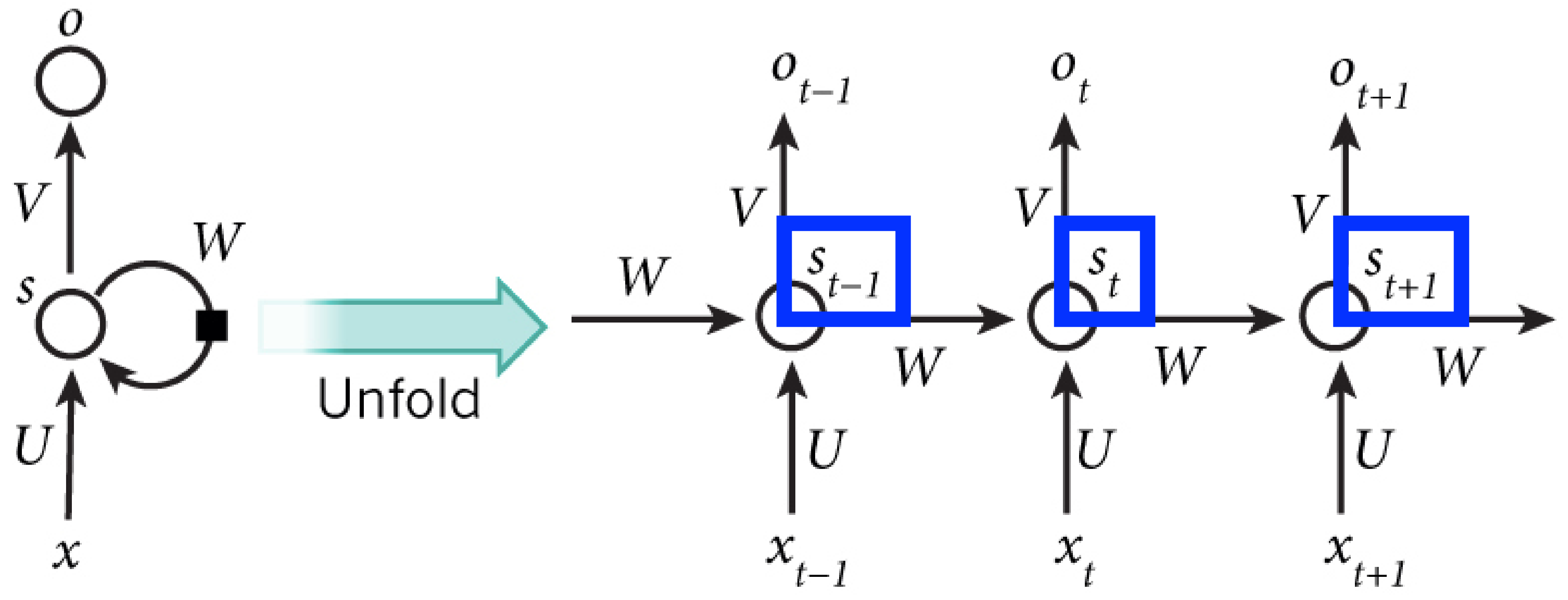
Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

Today's Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

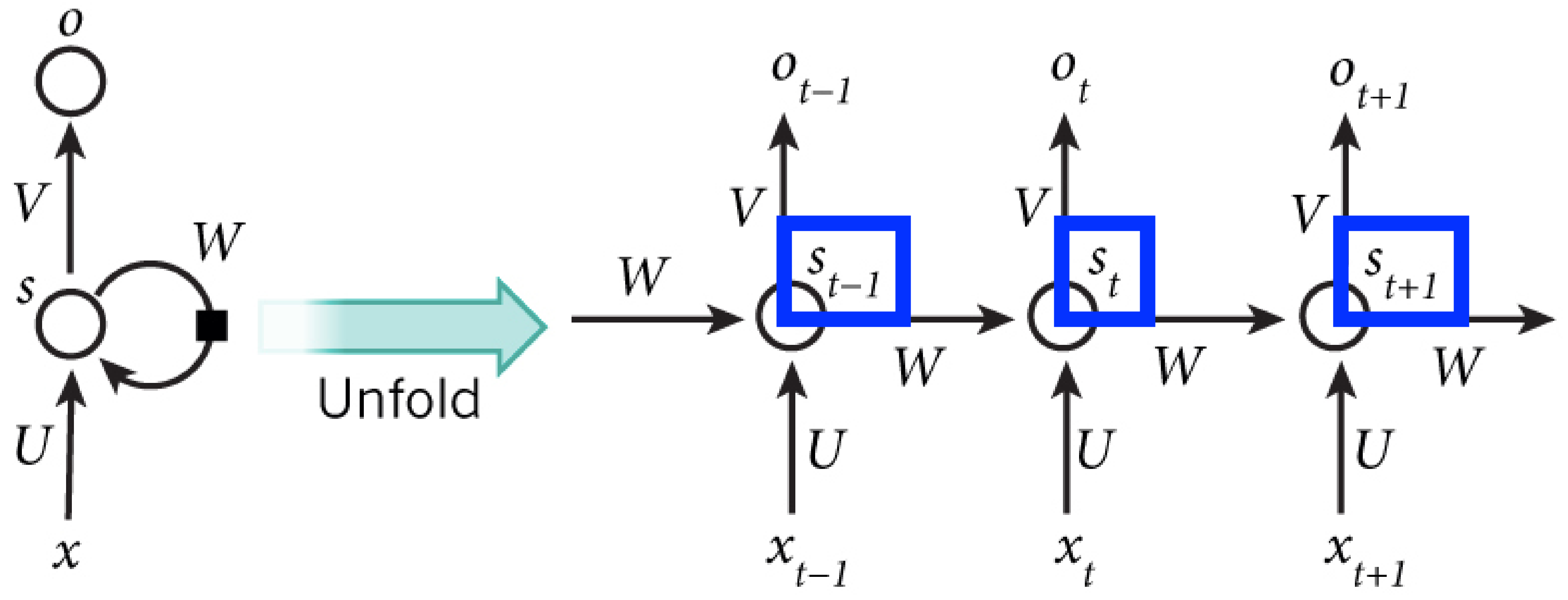
Goal: Model Sequential Data (Recall RNN)



Each hidden state is a function of the previous hidden state

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

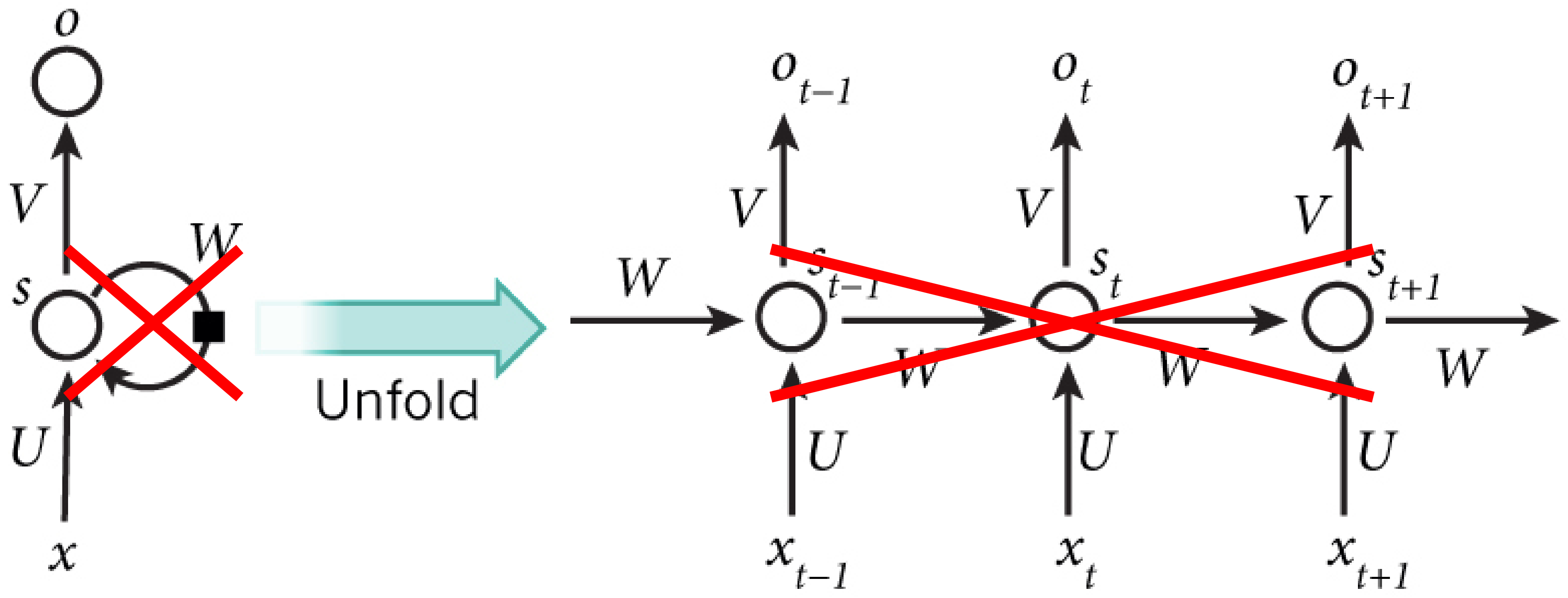
Problem: RNNs Use Sequential Computation



RNNs struggle to carry information through hidden states across many time steps and train/testing is slow

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Idea: Model Sequential Data Without Recurrence

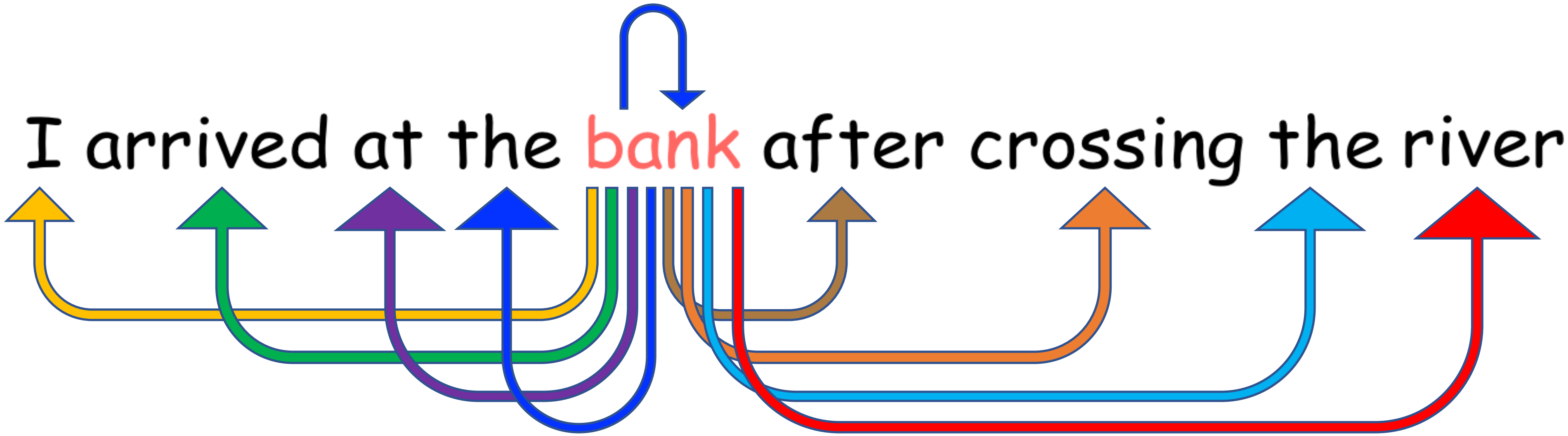


Replace sequential hidden states for capturing knowledge of other inputs with a new representation of each input that shows its relationship to all other inputs (i.e., self-attention)

http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Transformer Key Idea: Self-Attention

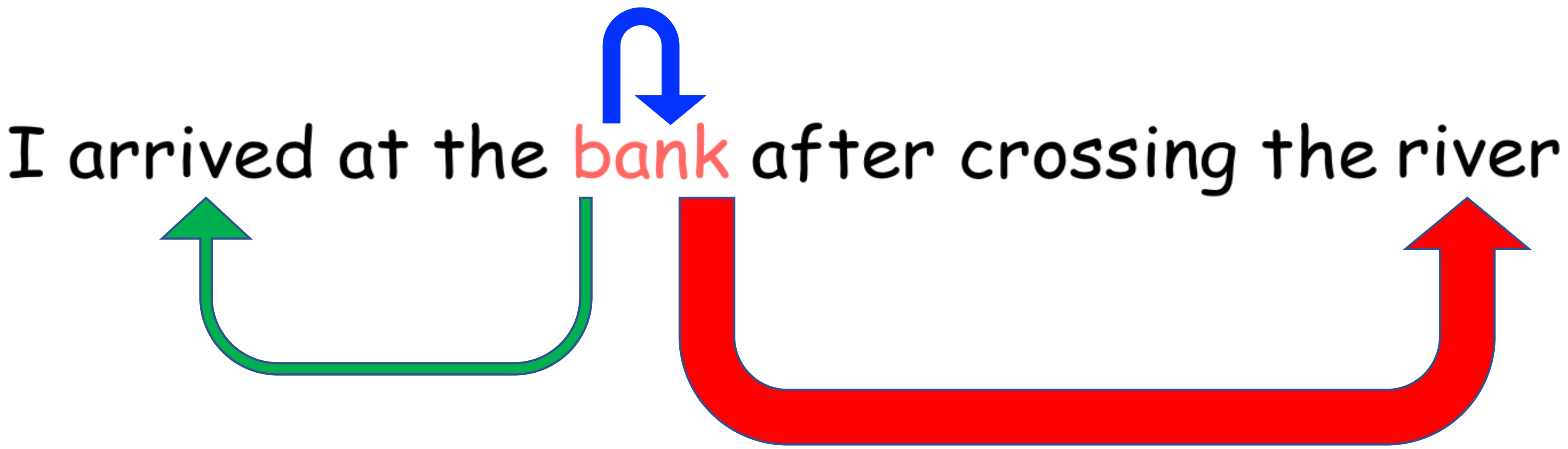
New representation of each token in a sequence showing its relationship to all tokens; e.g.,



https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Arrow thickness is indicative of attention weight

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Key Idea: Self-Attention

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

A large attention score means the other word will strongly inform the new representation of the word

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Intuition

What does bank mean in this sentence?

I arrived at the bank after crossing the …

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer Intuition

What does bank mean in this sentence?
- new word representation disambiguates meaning by identifying other relevant words (e.g., high attention score with “river”)

I arrived at the bank after crossing the river

vs

I arrived at the bank after crossing the street

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the ...

$\boxed{\text{...street?} \quad \text{...river?}}$

What does **bank** mean in this sentence? Meaning depends on other input words

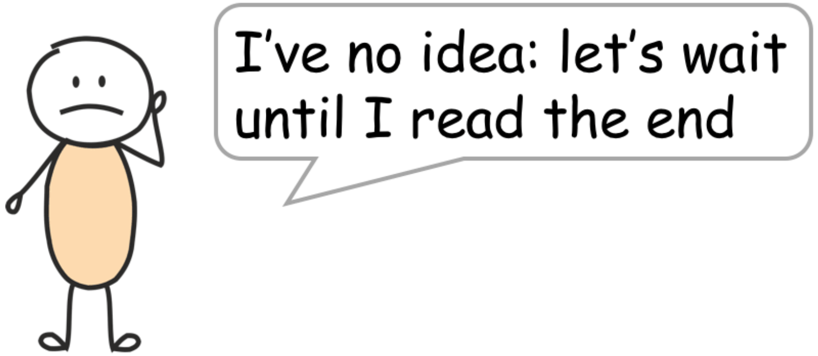
https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer vs RNN (Intuition)

I arrived at the bank after crossing the …

…street? …river?

What does **bank** mean in this sentence? Meaning depends on other input words



RNNs

O(N) steps to process a sentence with length N

Transformer

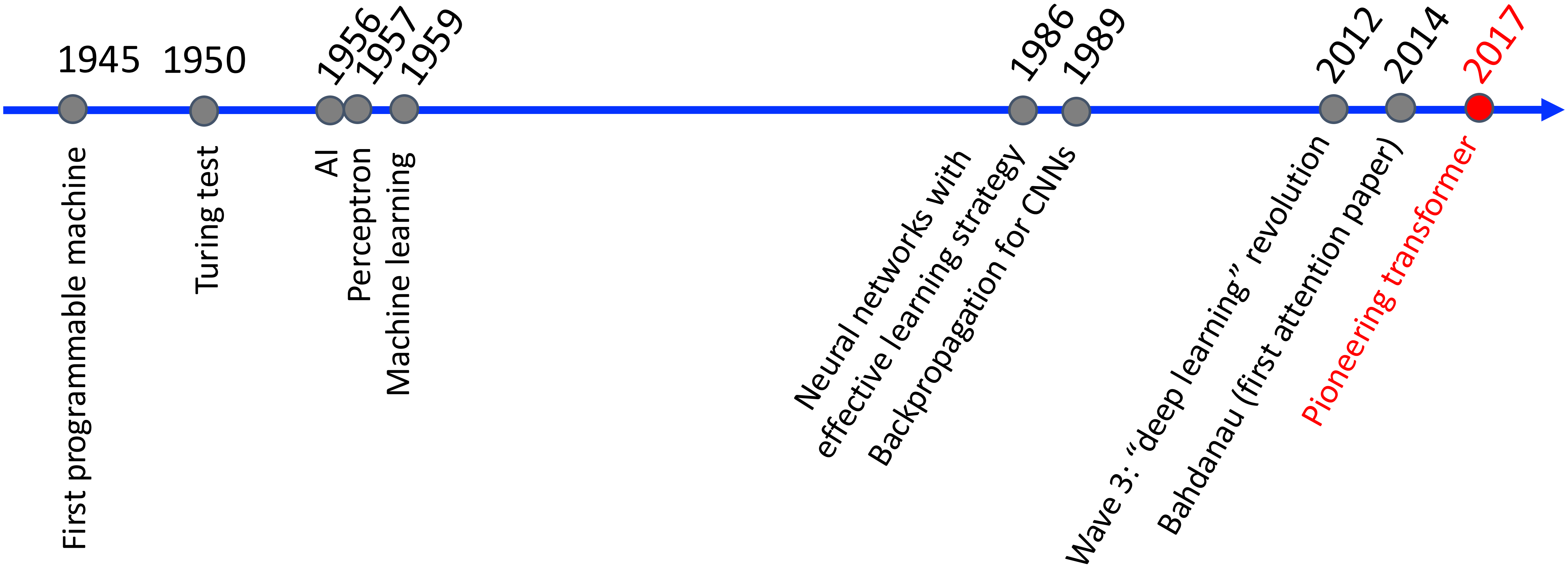
Constant number of steps to process any sentence

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

Transformer: A Suggested Definition

"Any architecture designed to process a connected set of units—such as the tokens in a sequence or the pixels in an image—where the only interaction between units is through self-attention."

Historical Context: Pioneering Transformer

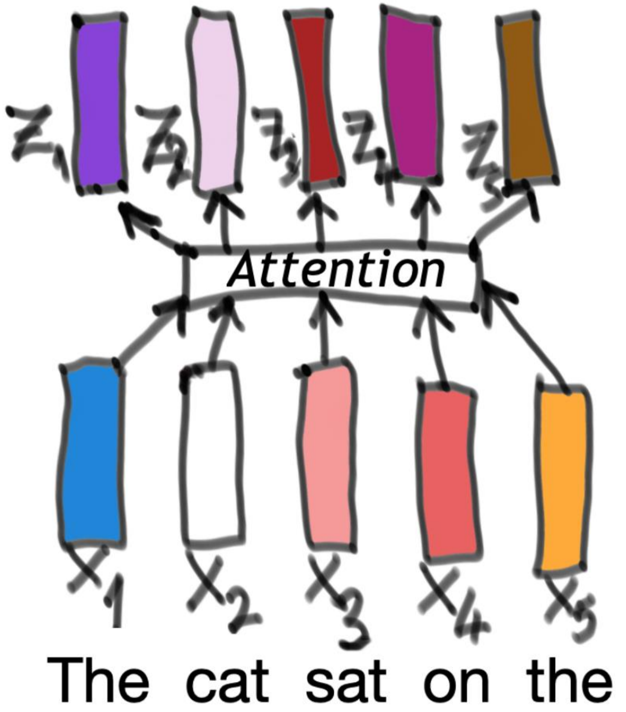


Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

Self-Attention: Outcome

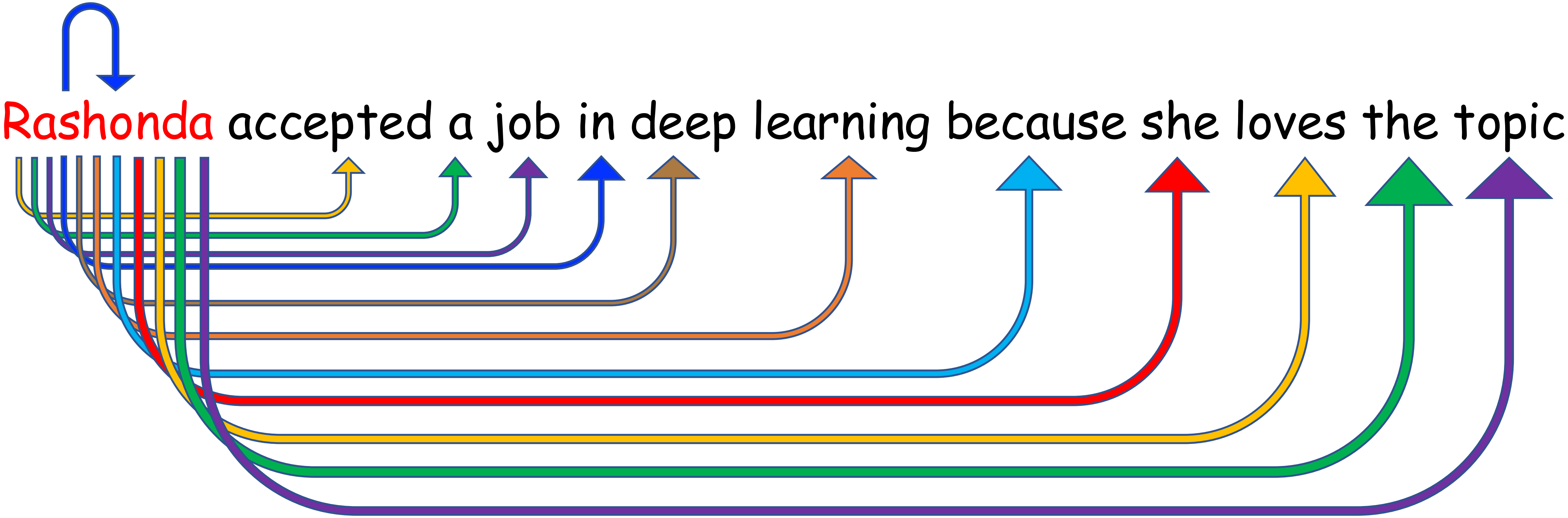
New representation of each token in a sequence showing its relationship to all tokens



https://towardsdatascience.com/self-attention-5b95ea164f61

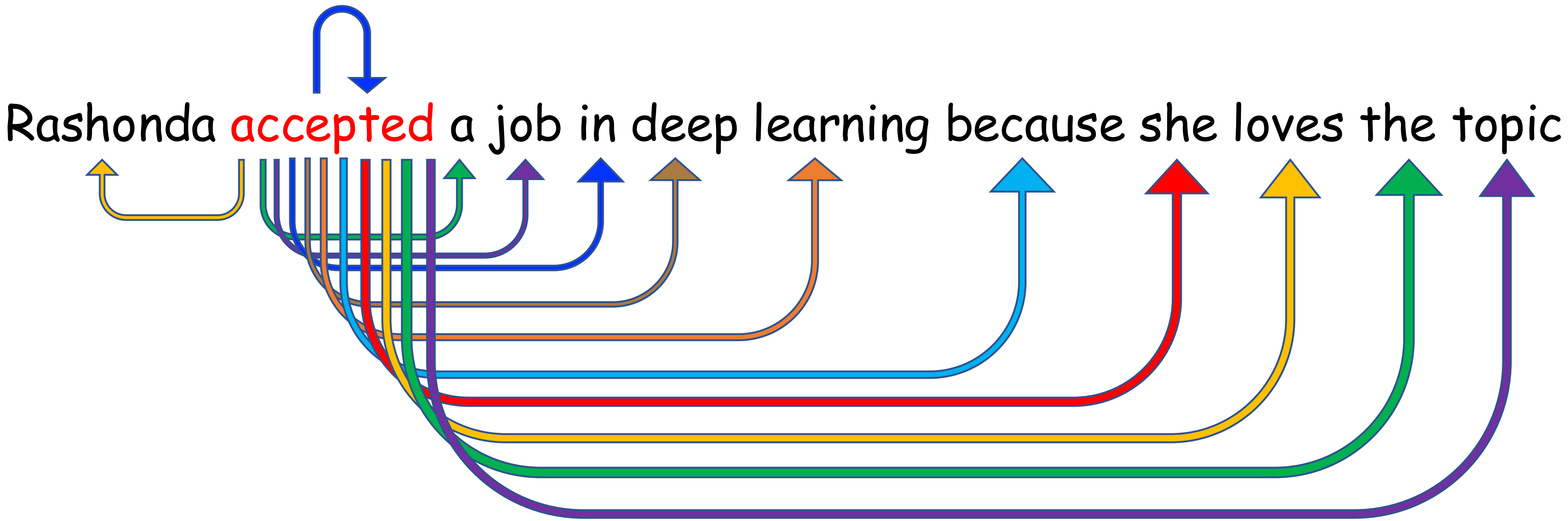
Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,



Self-Attention: Outcome

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

And so on for remaining words…

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

Rashonda accepted a job in deep learning because she loves the topic

A better representation of “she” would
encode information about “Rashonda”

Self-Attention: Disambiguates Word Meanings

New representation of each token in a sequence showing its relationship to all tokens; e.g.,

I arrived at the bank across the river

Recall: a better representation of “bank” would encode information about “river”

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html#transformer_intro

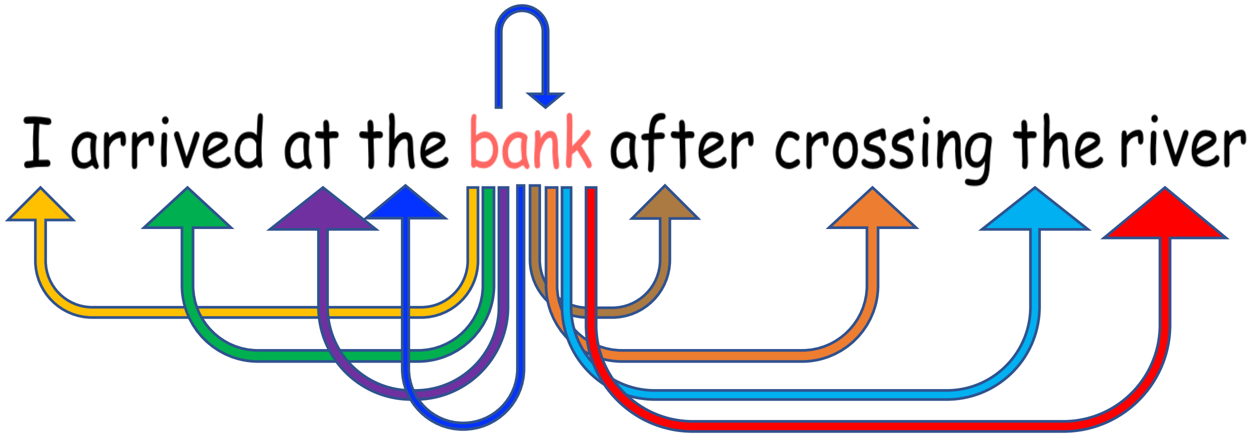
Self-Attention vs General Attention

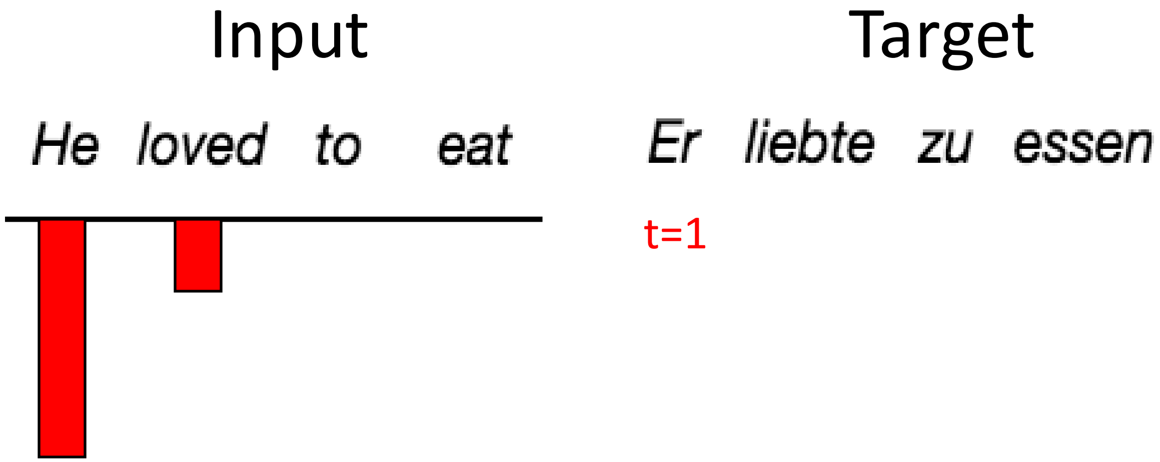
Self-attention

Relates tokens from the same source

General attention

Relates tokens from different sources





Computing Self-Attention: Similar Approach to How We Compute General Attention

Attention output

$$c^{(t)} = a_1^{(t)} s_1 + a_2^{(t)} s_2 + \dots + a_m^{(t)} s_m = \sum_{k=1}^{m} a_k^{(t)} s_k$$

"source context for decoder step t"

(weighted sum)

Attention weights

$$a_k^{(t)} = \frac{\exp(\text{score}(h_t, s_k))}{\sum_{i=1}^{m} \exp(\text{score}(h_t, s_i))}, \mathrm{k} = 1..\mathrm{m}$$

"attention weight for source token k at decoder step t"

(softmax)

Attention scores

$$\text{score}(h_t, s_k), \mathrm{k} = 1..\mathrm{m}$$

"How relevant is source token k for target step t?"

Attention input

$s_1, s_2, \dots, s_m$ — all encoder states

h_t — one decoder state

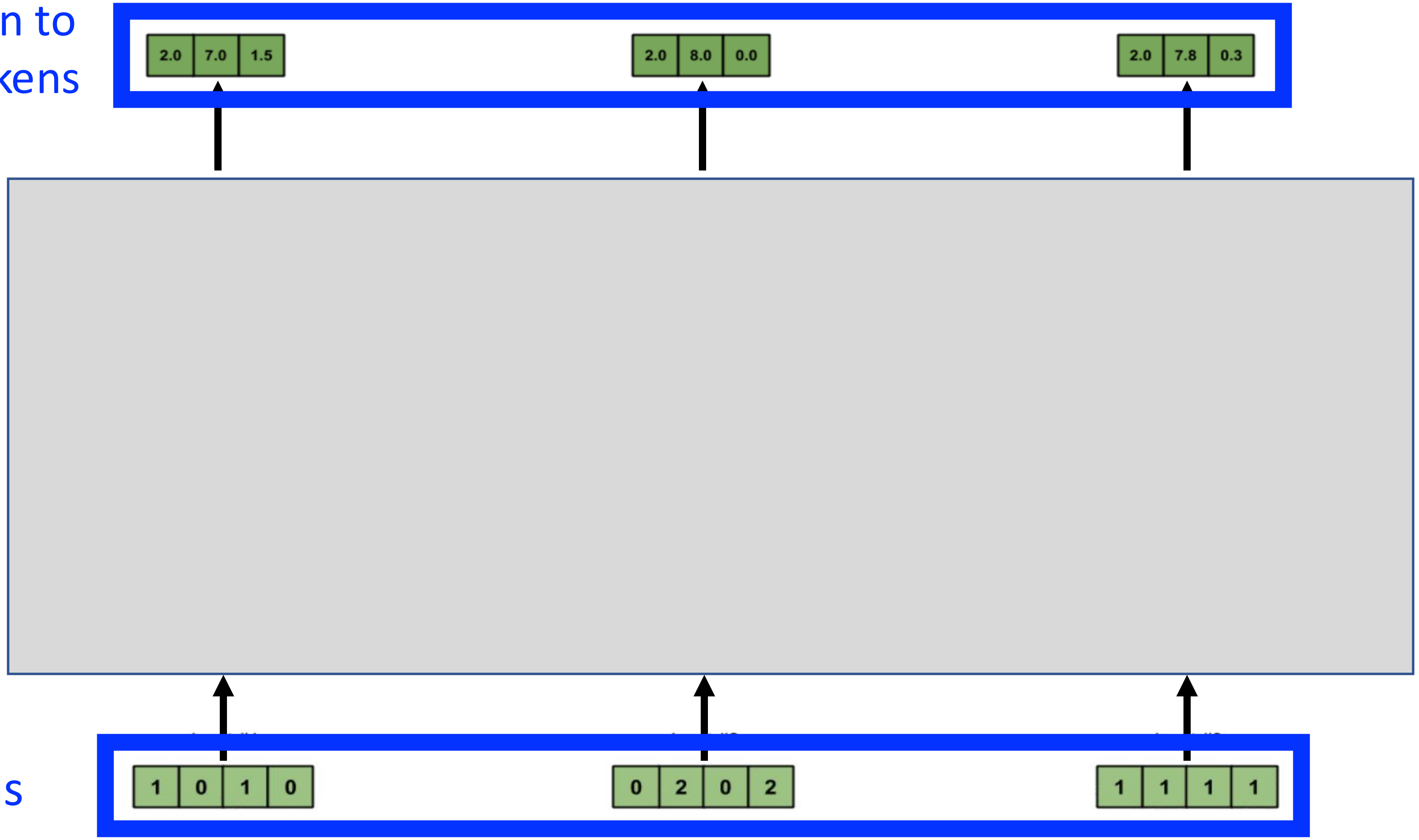
Key difference 2: attention score multiplied with a value derived from the input

Key difference 1: input for self-attention

https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html

Computing Self-Attention: Example

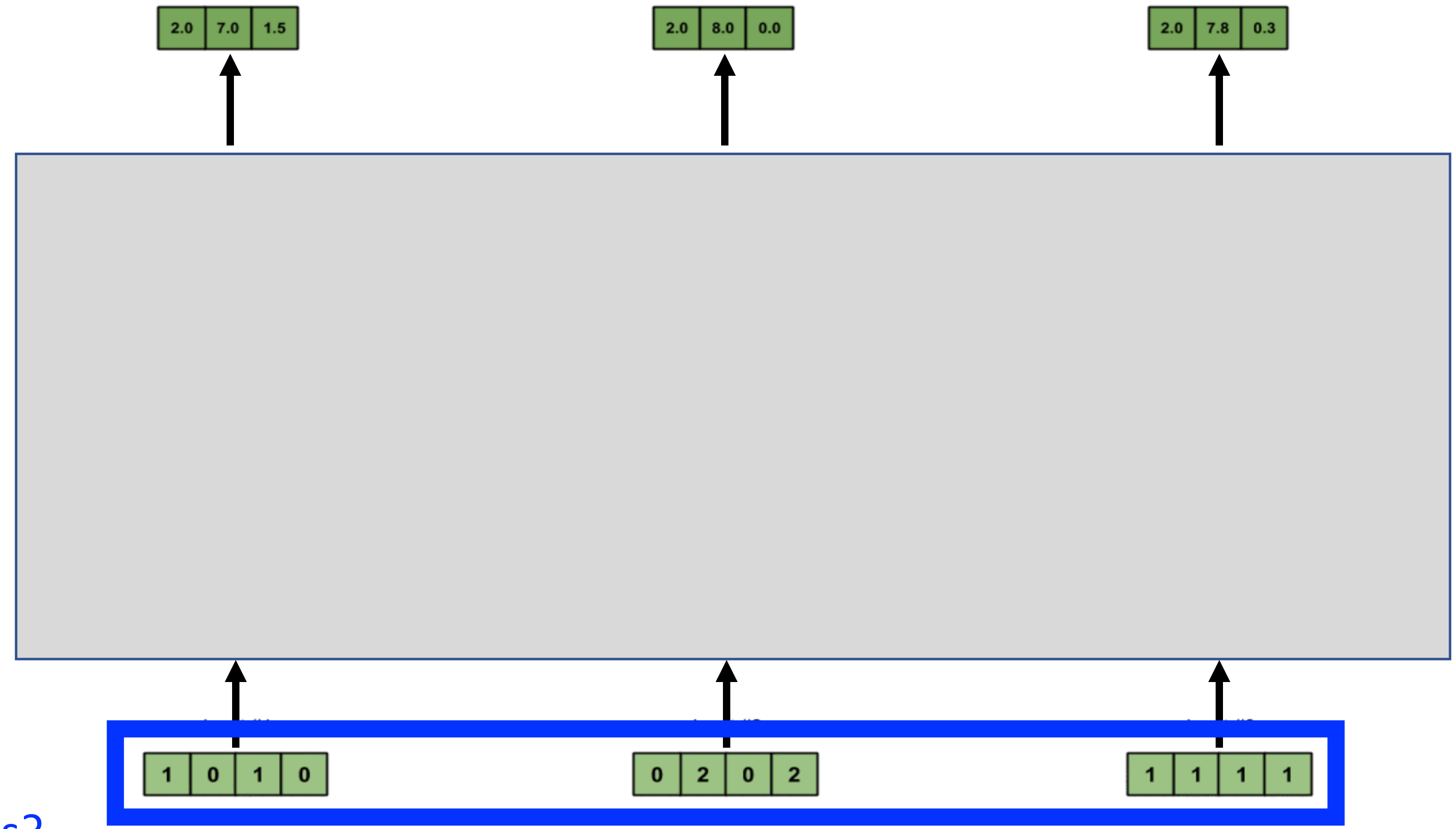
New representation of each input token to reflect each one's relationship to all tokens



Input tokens

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

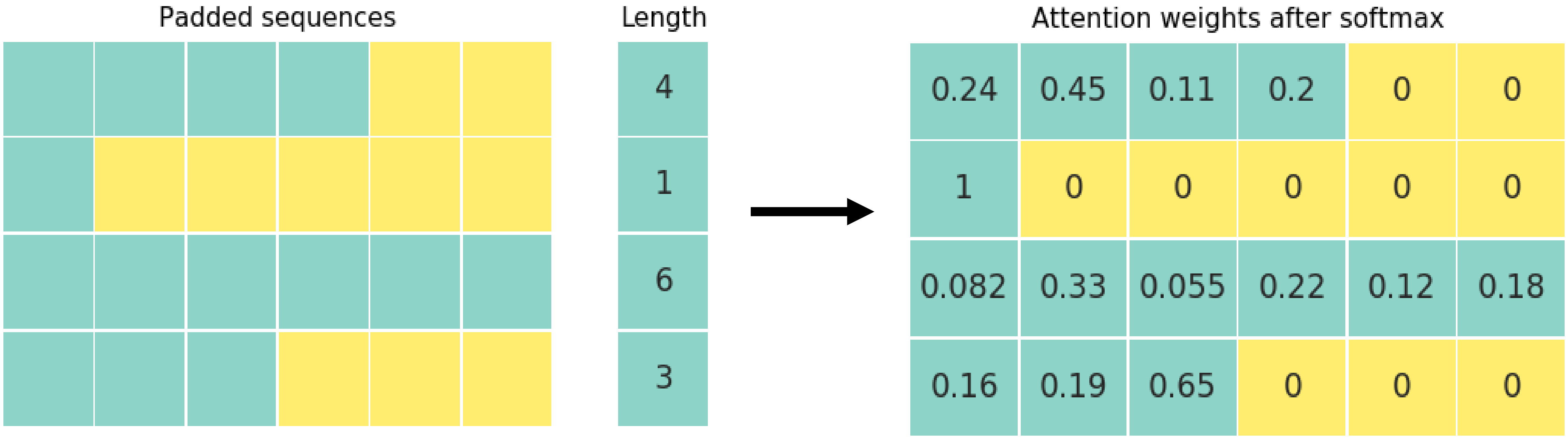


- How many input tokens are there?
- What is each token's dimensionality?
- How to support arbitrary length inputs?
    * Input length is a hyperparameter: pad shorter sequences with zeros and truncate longer sequences

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

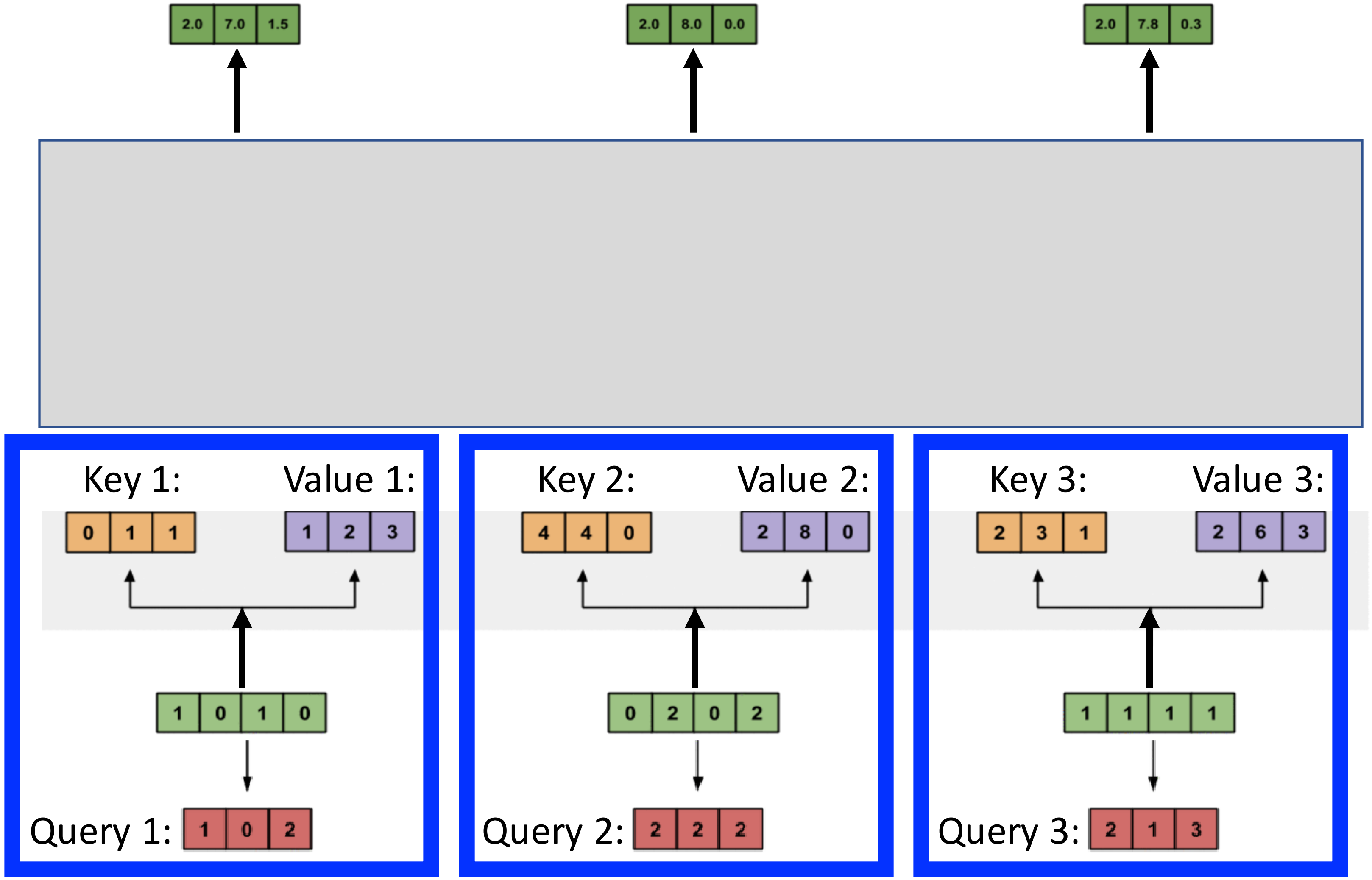
Input Length: Implementation

- [PAD] tokens with attention set to 0 enables variable input length



http://juditacs.github.io/2018/12/27/masked-attention.html

Computing Self-Attention: Example



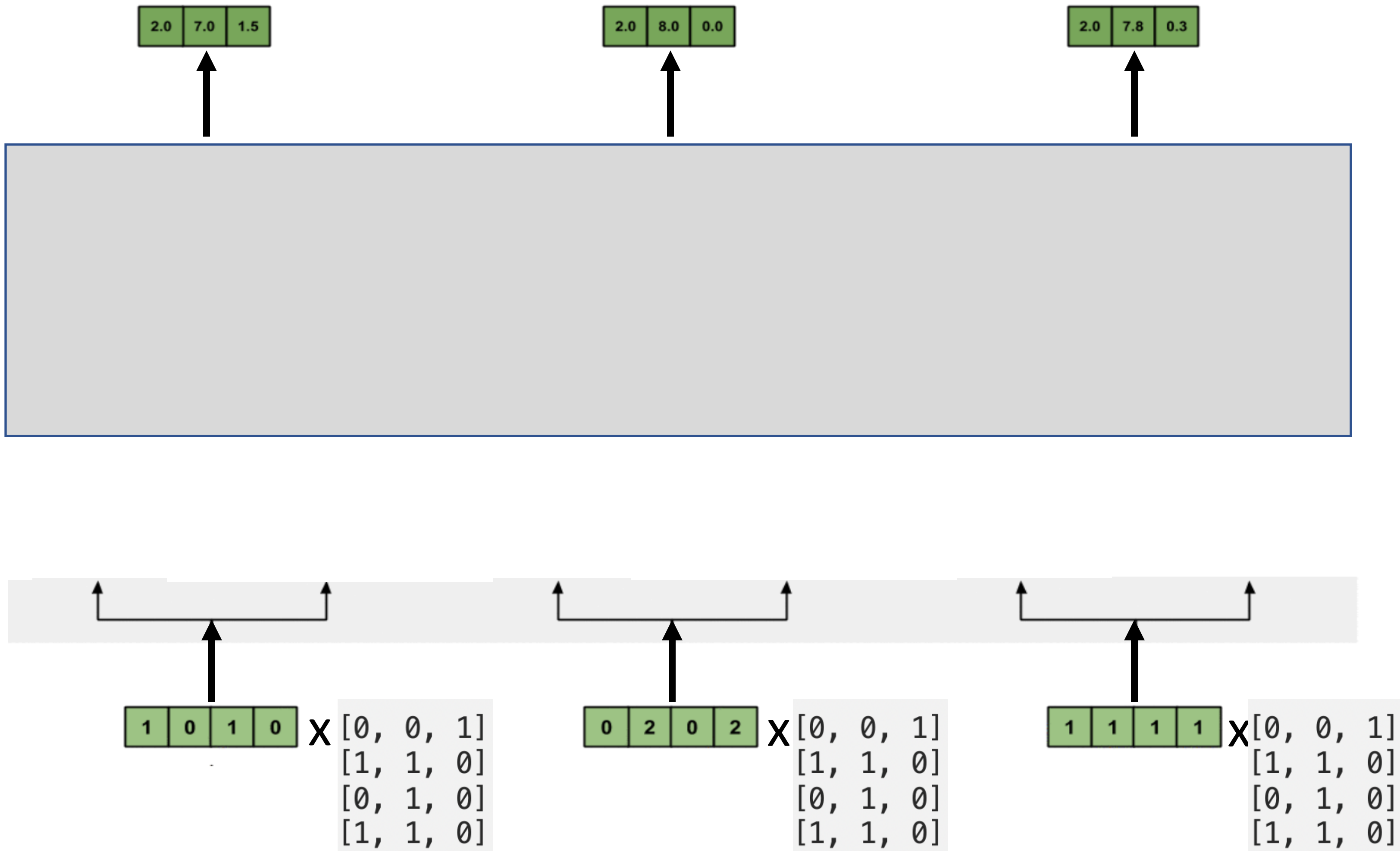
Three vectors are derived for each input by multiplying with three weight matrices (learned during training): query, key, and value

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., key weights

```
[0, 0, 1]
[1, 1, 0]
[0, 1, 0]
[1, 1, 0]
```

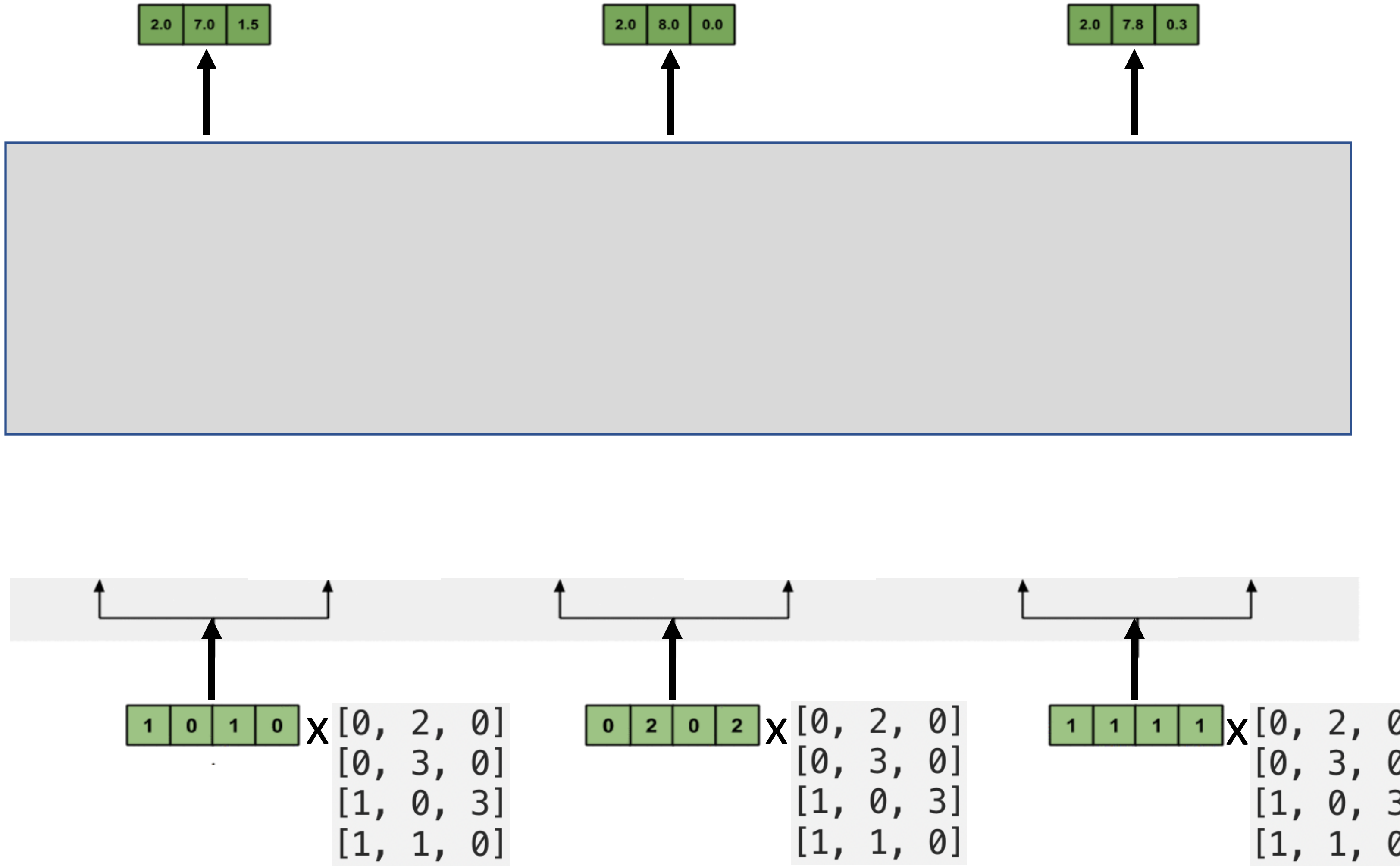


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

e.g., value weights

```
[0, 2, 0]
[0, 3, 0]
[1, 0, 3]
[1, 1, 0]
```

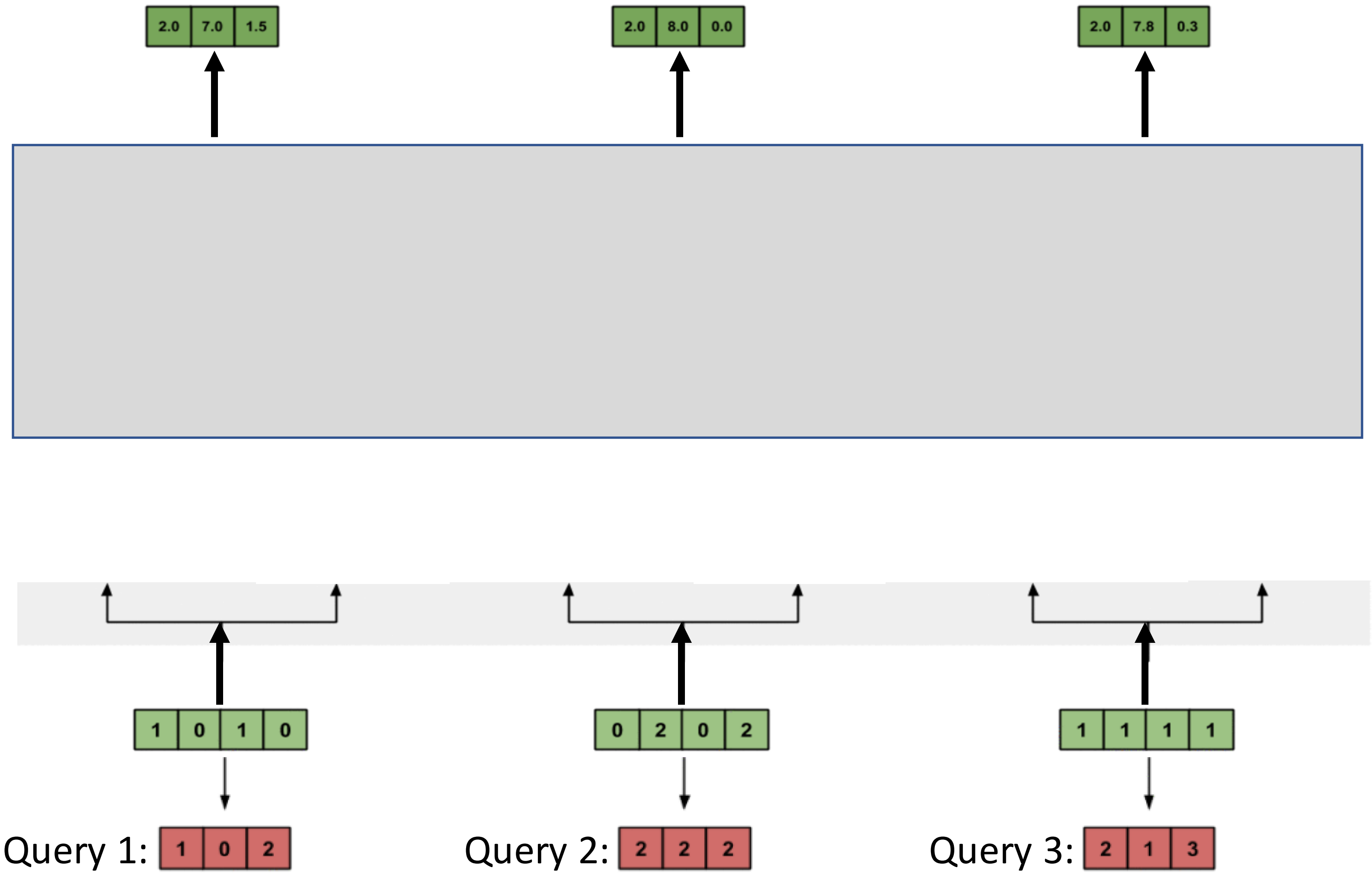


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

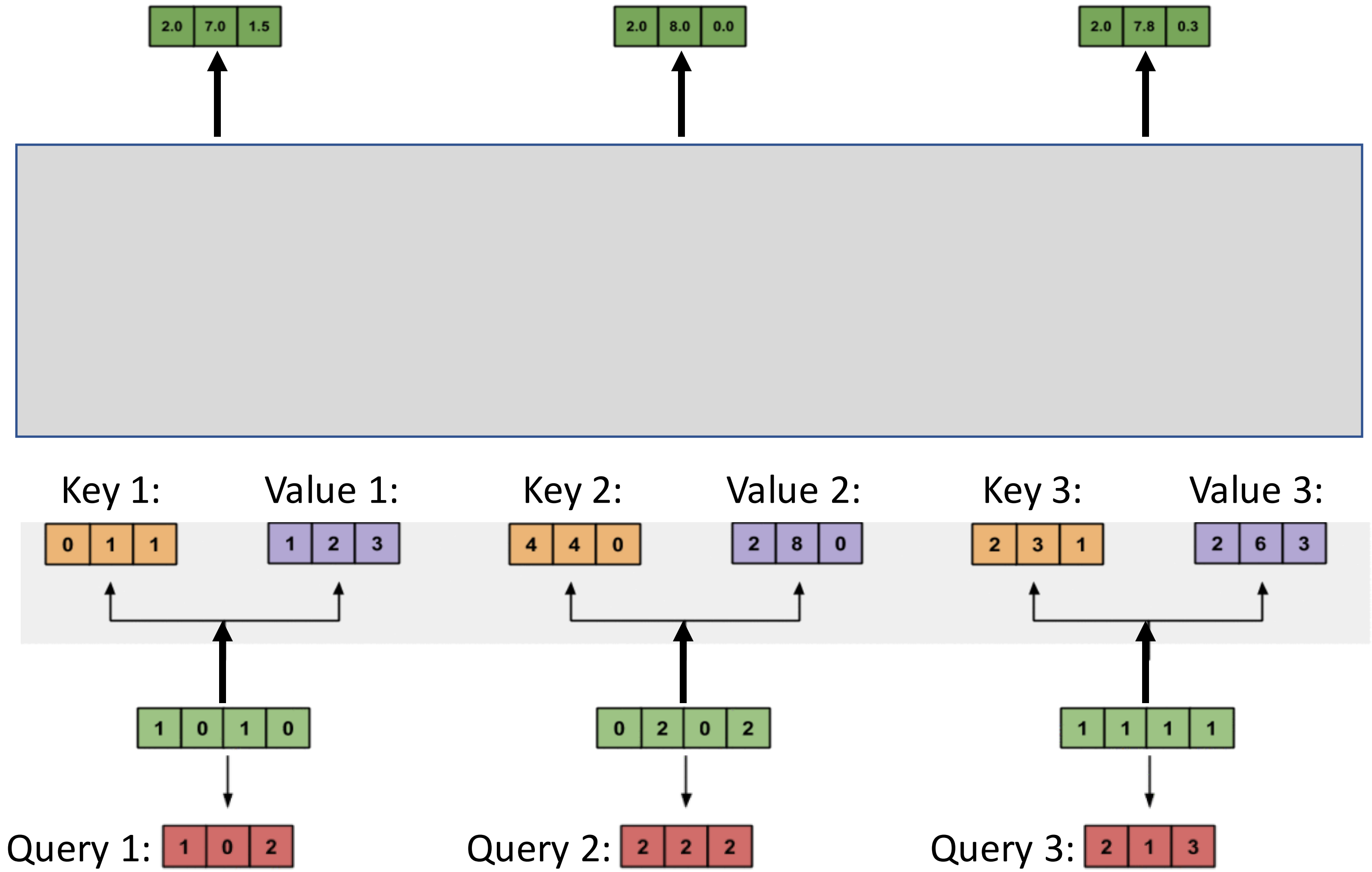
e.g., query weights

```
[1, 0, 1]
[1, 0, 0]
[0, 0, 1]
[0, 1, 1]
```



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



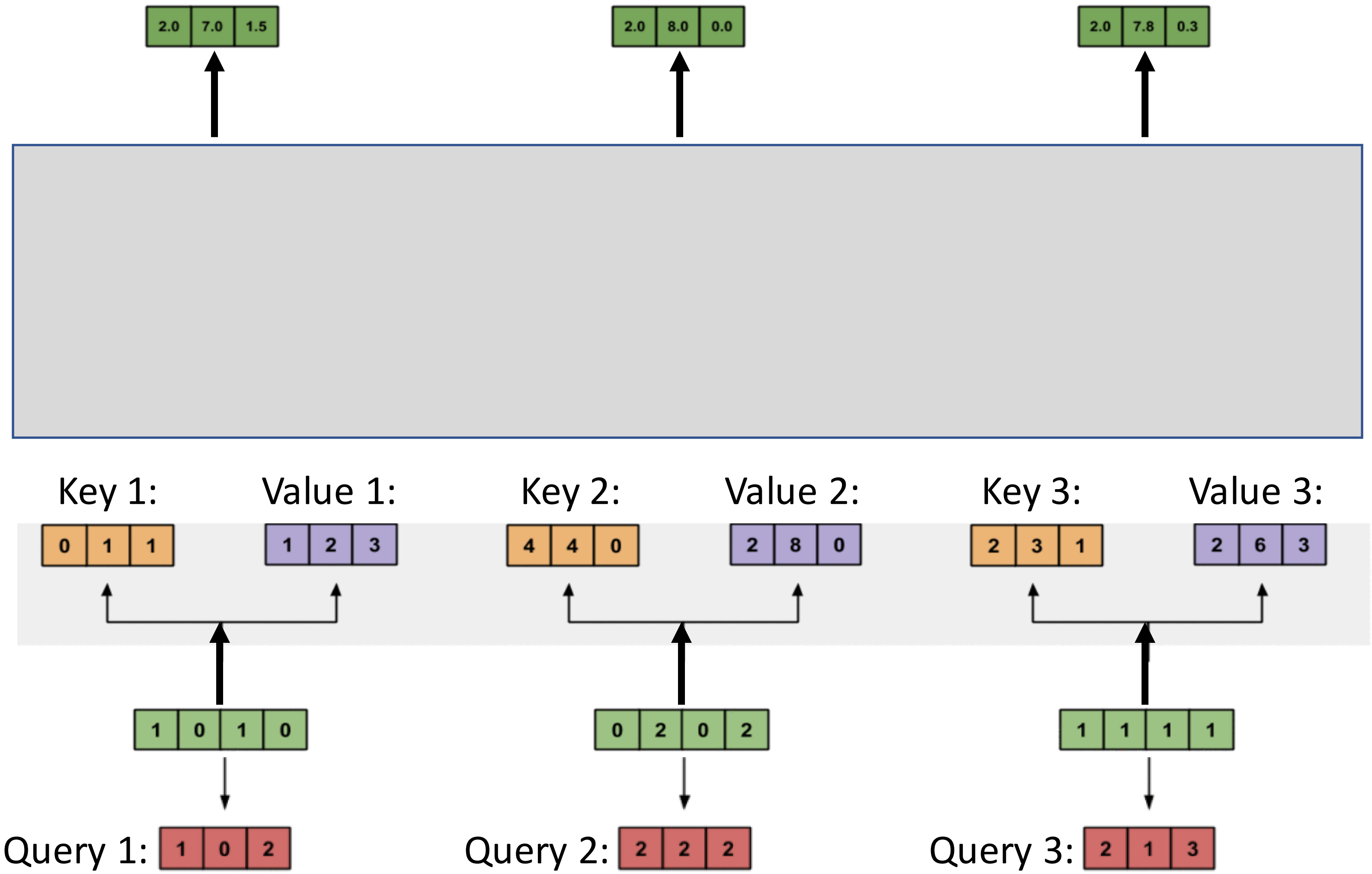
How many weight matrices are learned in this example?

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

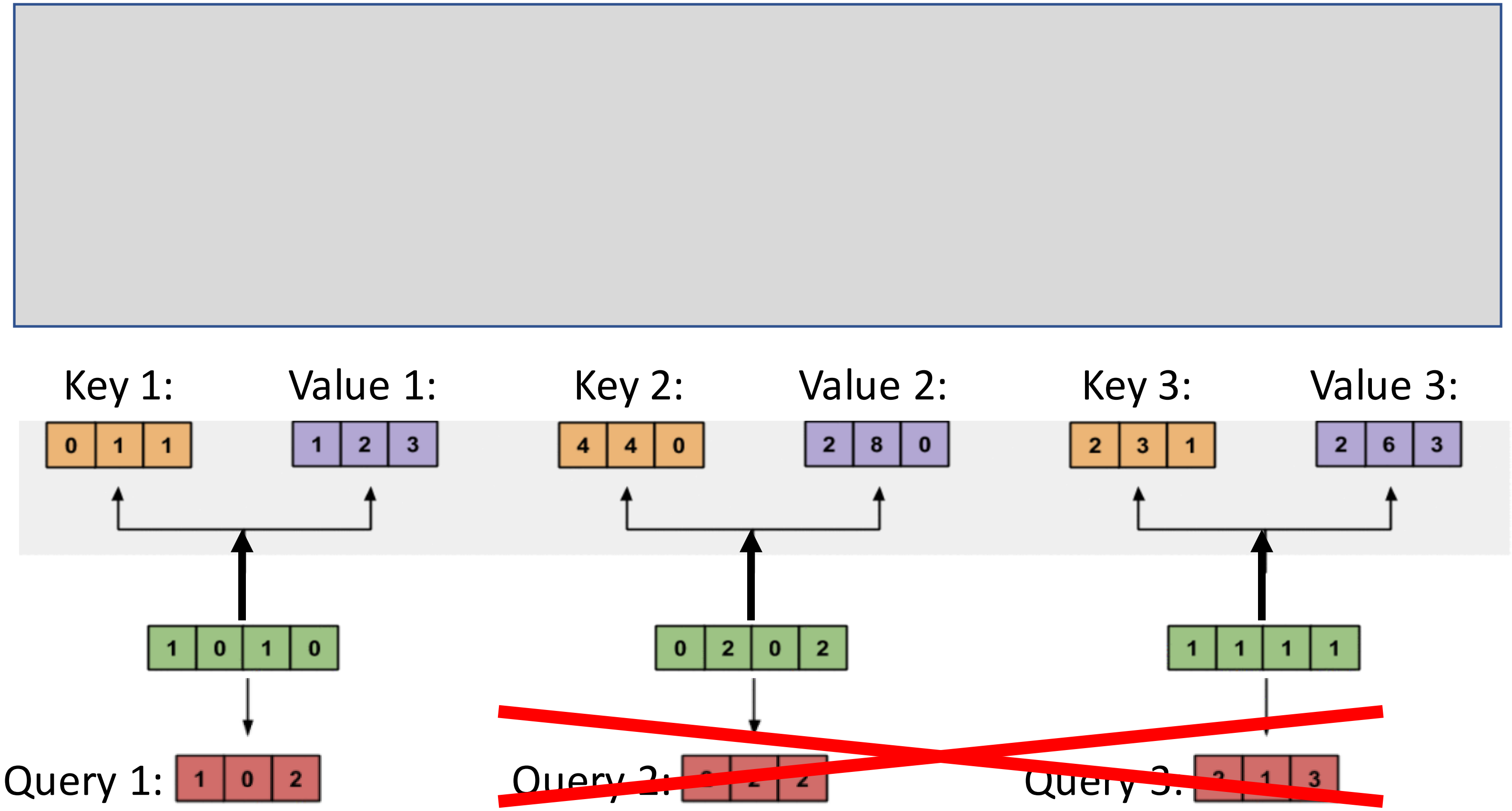
What is the purpose of the three weight matrices?

For each input, 2 of the derived vectors are used to compute **attention weights** (query and key) and the 3rd is **information** passed on for the new representation (value)



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



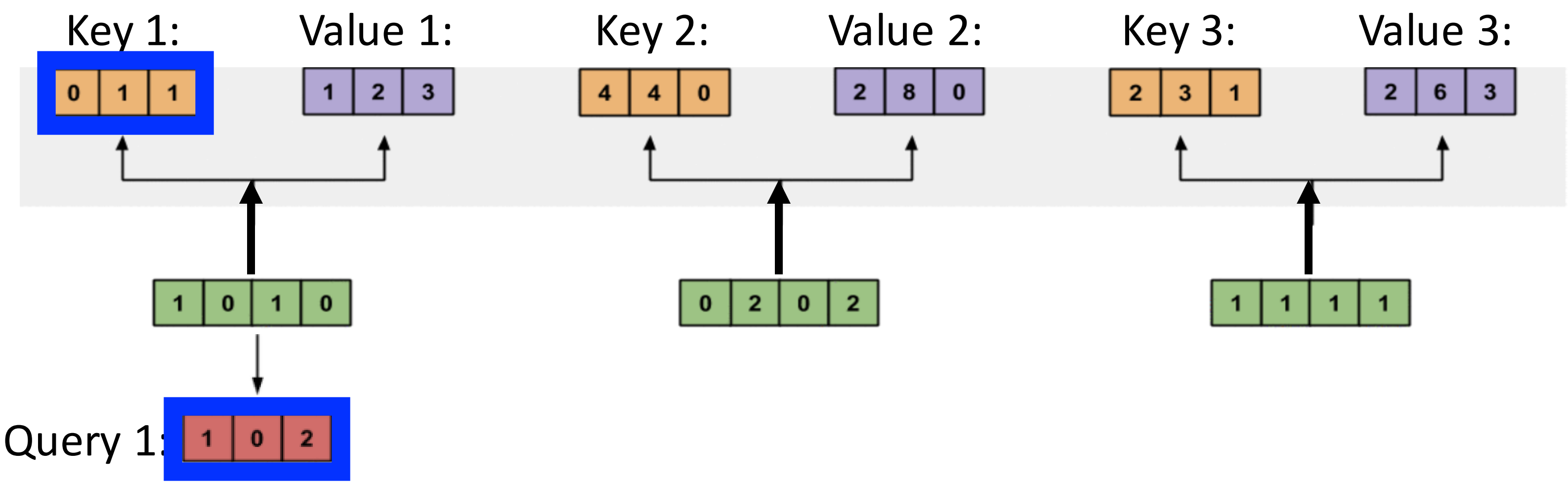
Let's compute the new representation for the inputs…

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query ("what am I looking for") with all keys ("what I have") to identify relevant tokens (higher scores are better matches); e.g.,

$$\begin{bmatrix}1 & 0 & 2\end{bmatrix} \times \begin{bmatrix}0\\1\\1\end{bmatrix} = ?$$

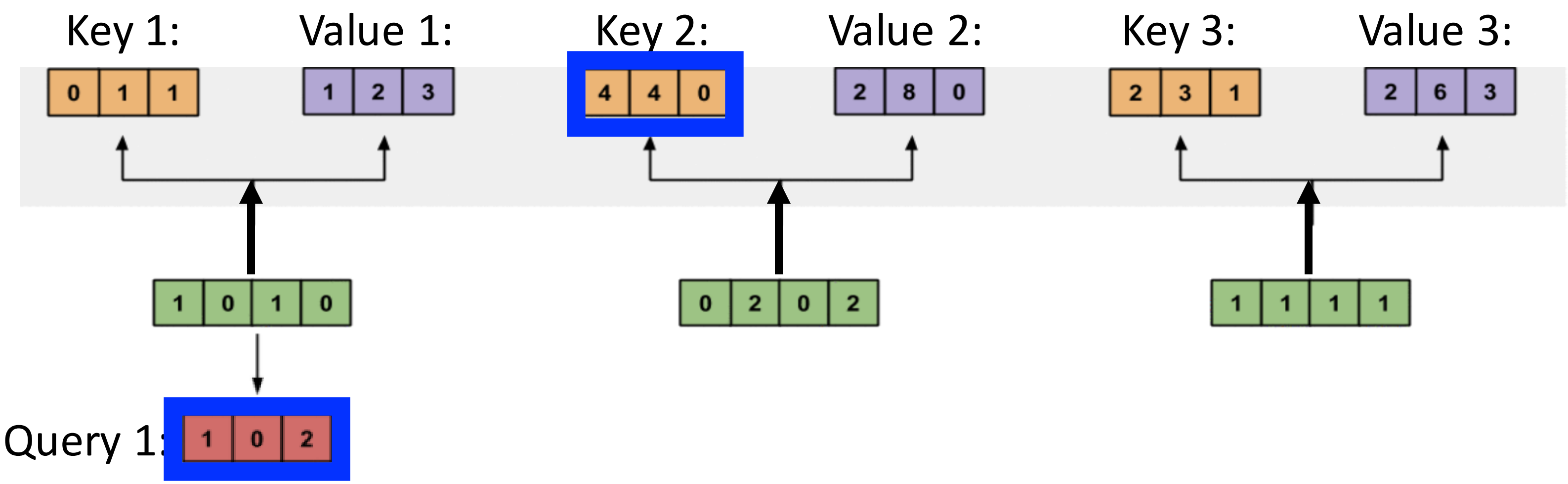


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query ("what am I looking for") with all keys ("what I have") to identify relevant tokens (higher scores are better matches); e.g.,

$$\begin{bmatrix} 1 & 0 & 2 \end{bmatrix} \times \begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} = ?$$

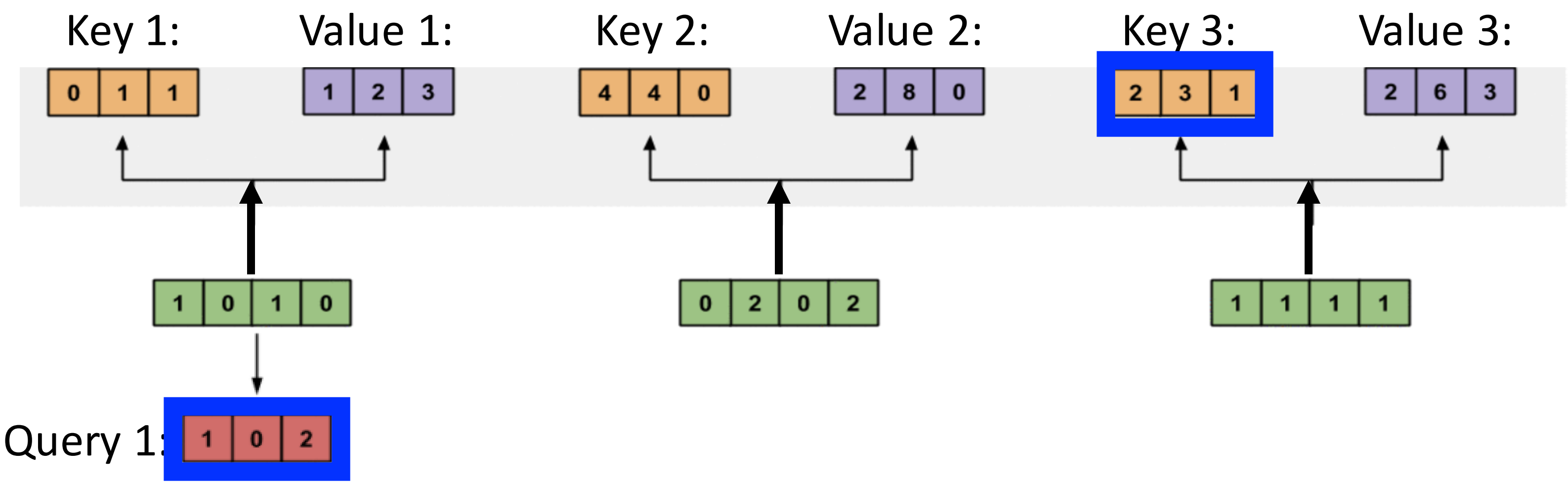


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention score: dot product of query ("what am I looking for") with all keys ("what I have") to identify relevant tokens (higher scores are better matches); e.g.,

$$\begin{bmatrix}1 & 0 & 2\end{bmatrix} \times \begin{bmatrix}2\\3\\1\end{bmatrix} = ?$$

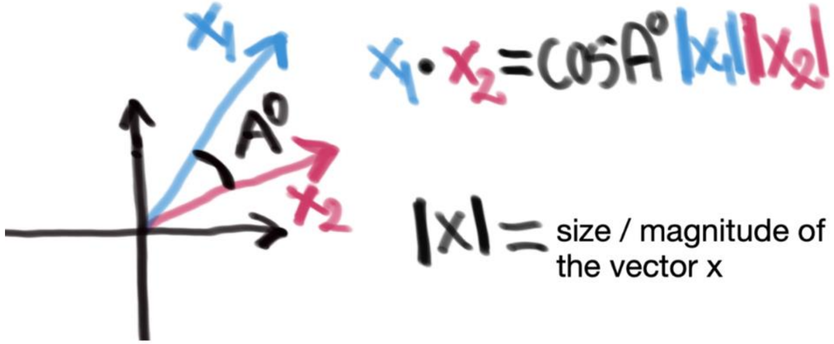


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

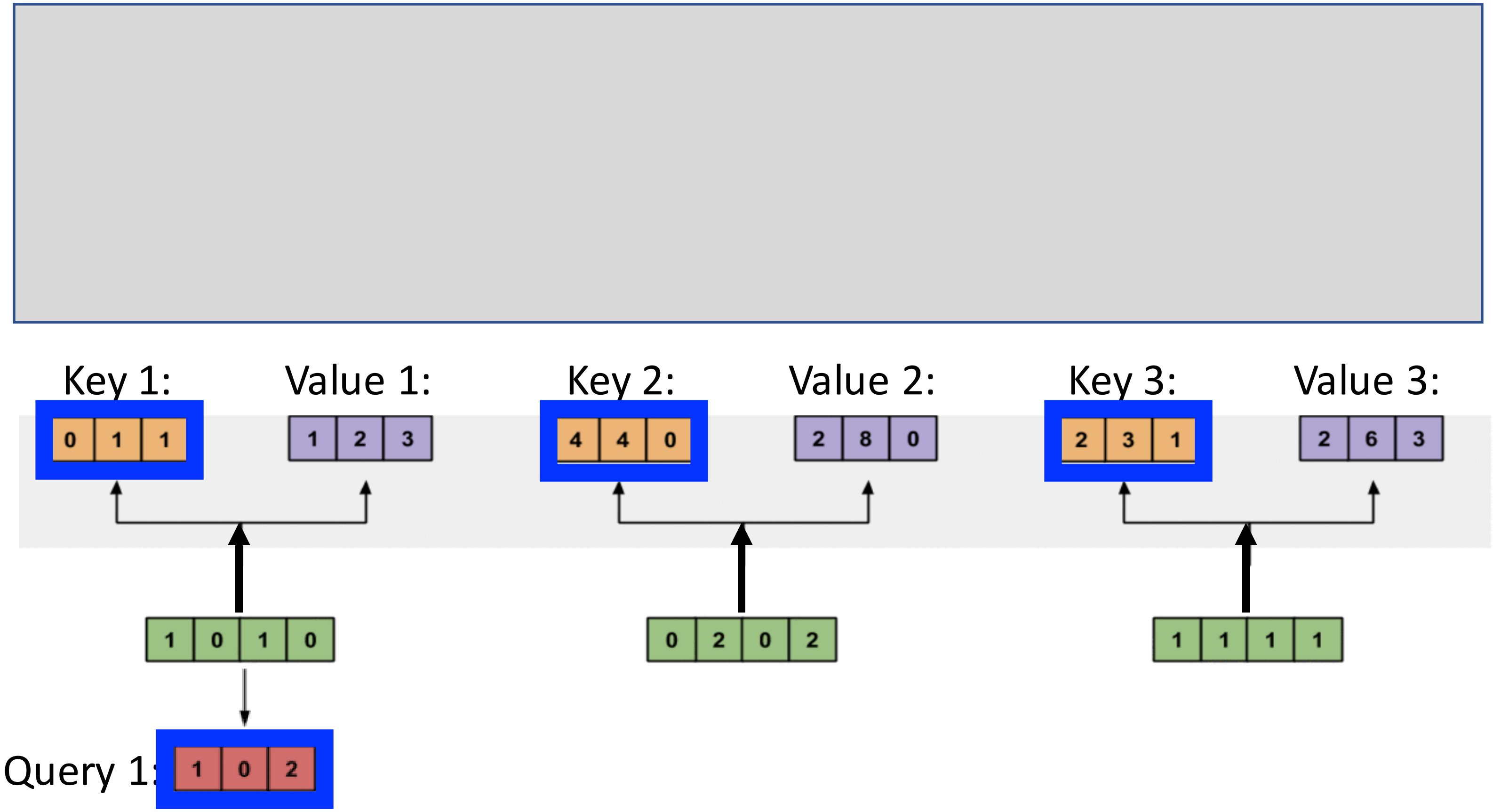
Computing Self-Attention: Example

Why dot product? Indicates similarity of two vectors

- Match = 1 (i.e., cos(0))
- Opposites = -1 (i.e., cos(180))



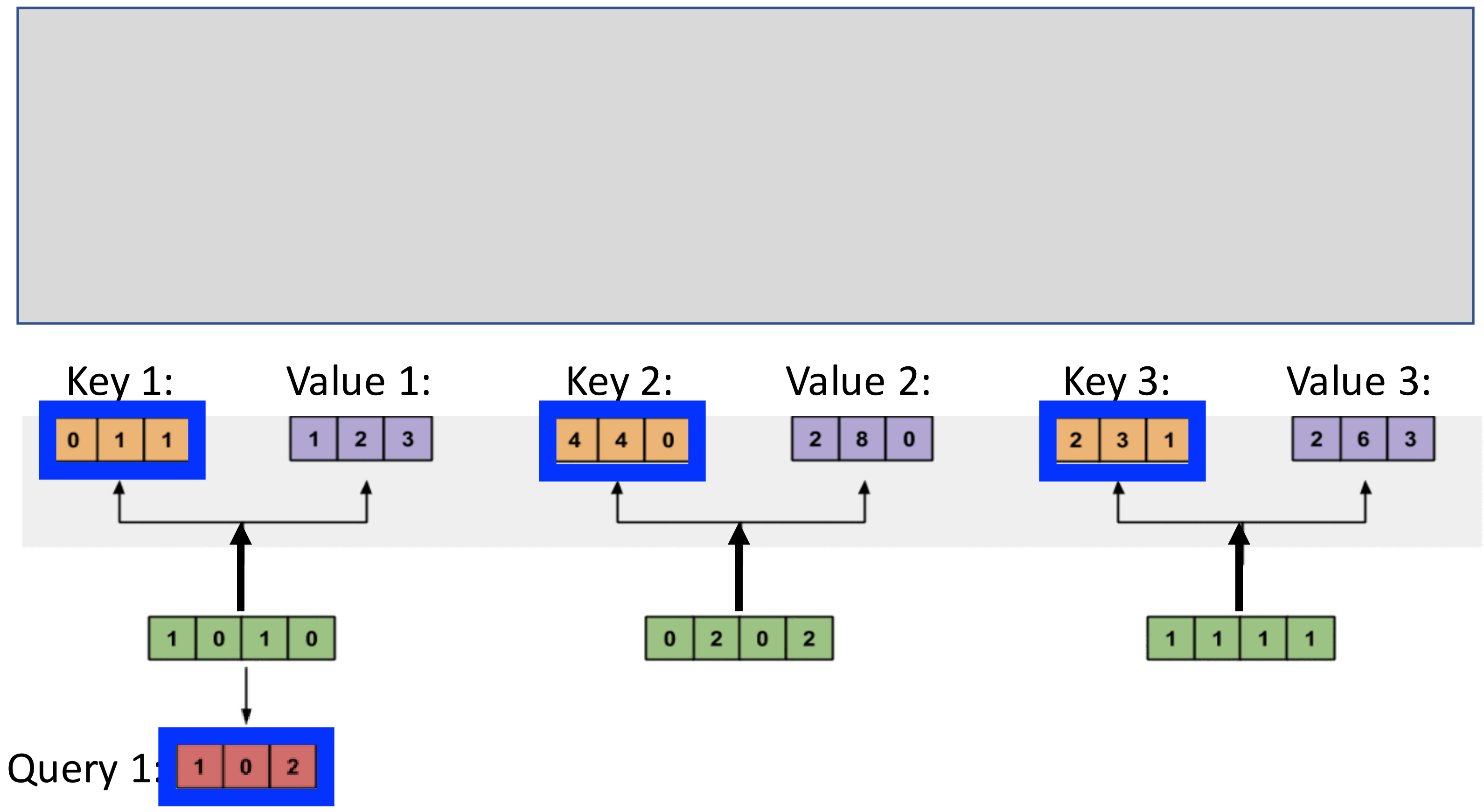
https://towardsdatascience.com/self-attention-5b95ea164f61



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Recall: there are many compatibility measures



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Attention weights: softmax scores for all inputs to quantify each token's relevance; e.g.,

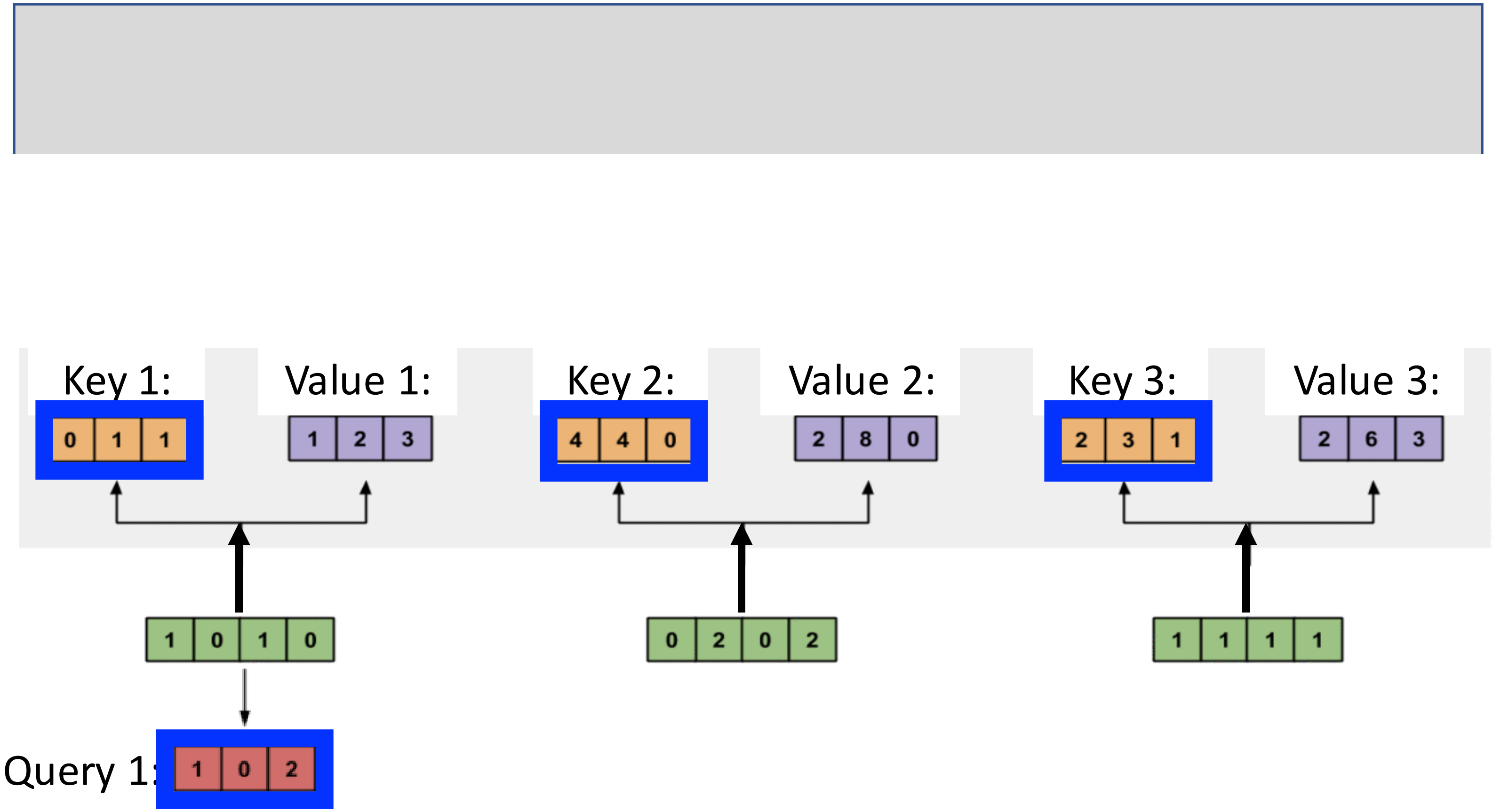
= softmax([2, 4, 4])

= [0.0, 0.5, 0.5])

Note: 0 from softmax can arise from rounding

To which input(s) is input 1 **least** related?

To which input(s) is input 1 **most** related?

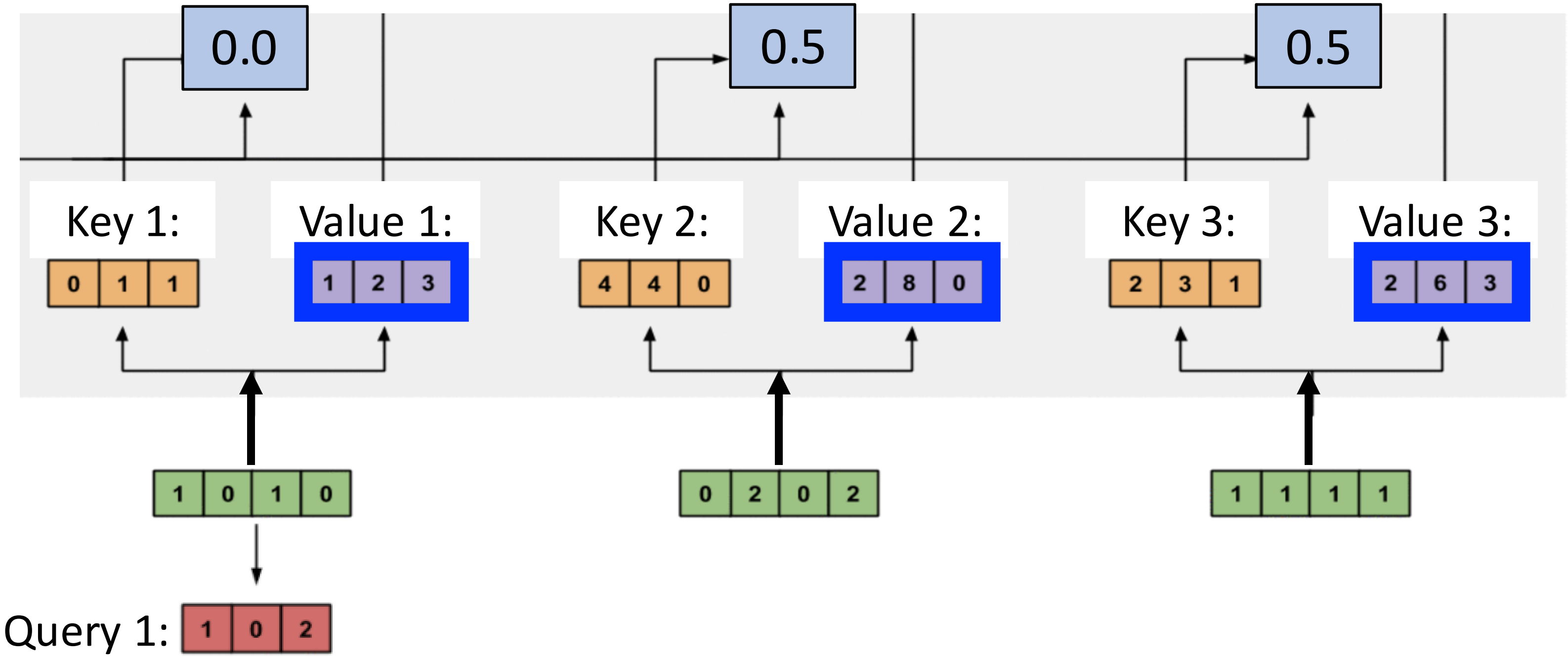


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values



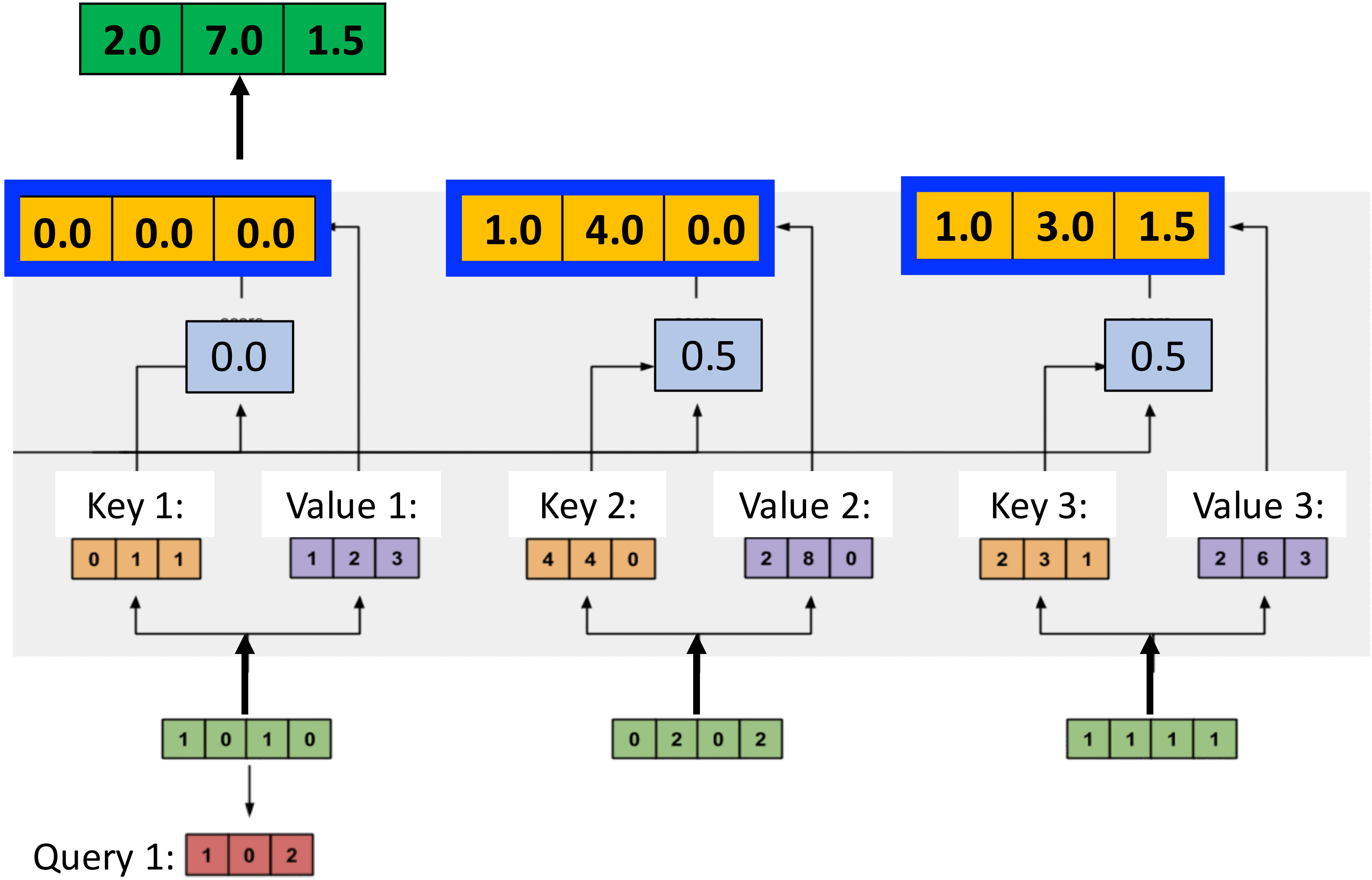
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Compute new representation of input token that reflects entire input:

1. Attention weights x Values

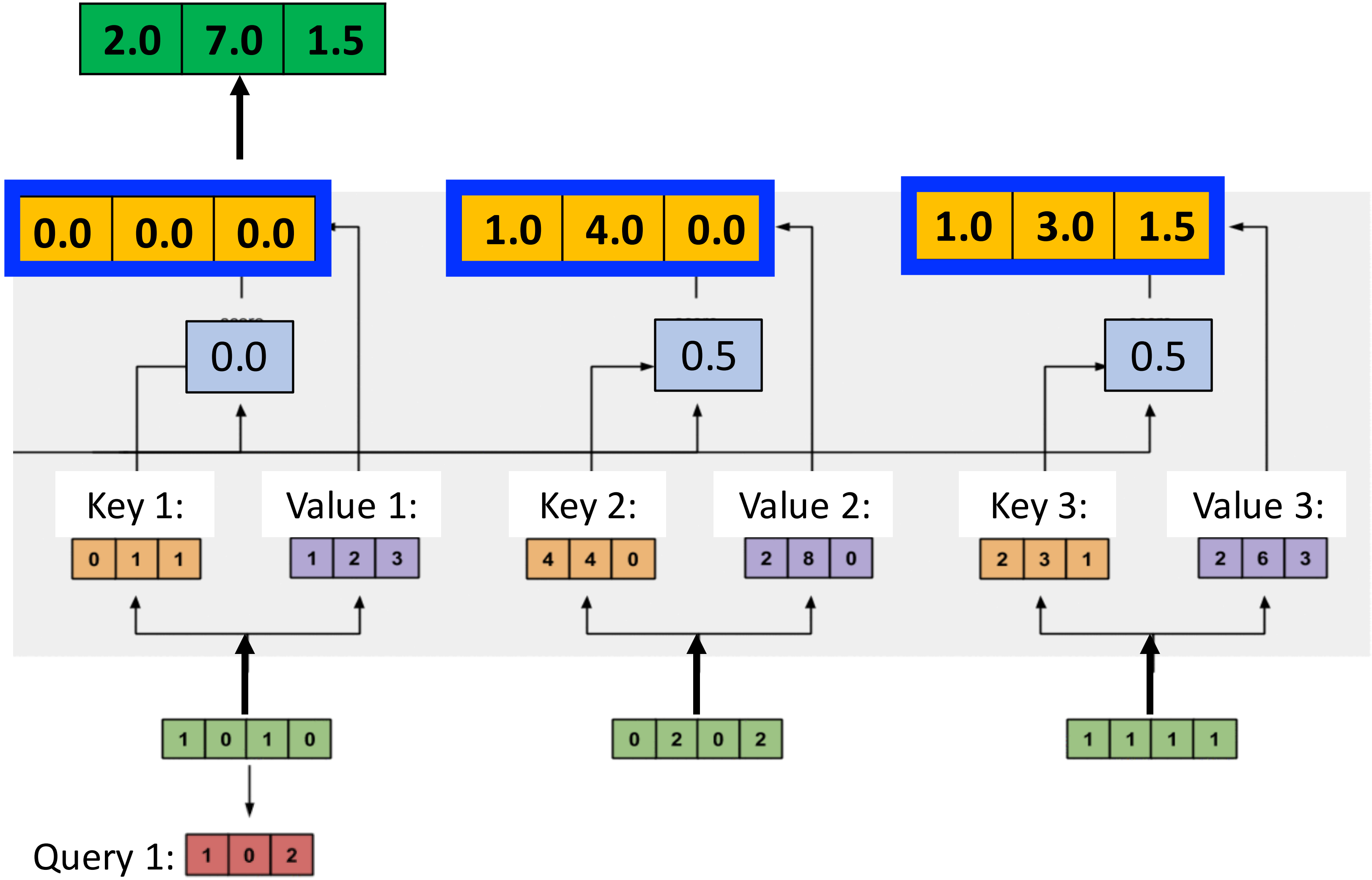
2. Sum all weighted vectors



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

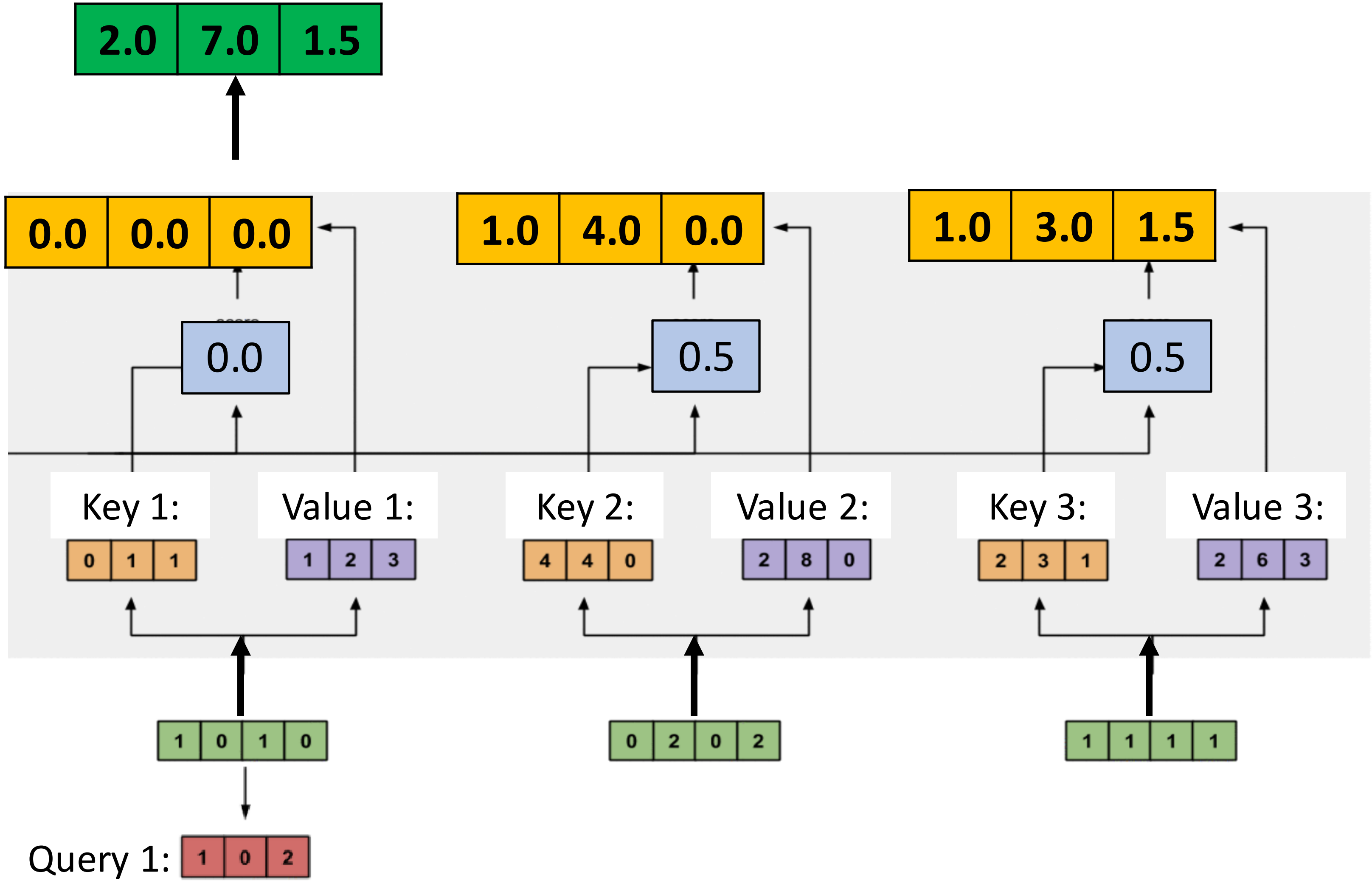
Attention weights amplify input representations (values) that we want to pay attention to and repress the rest



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

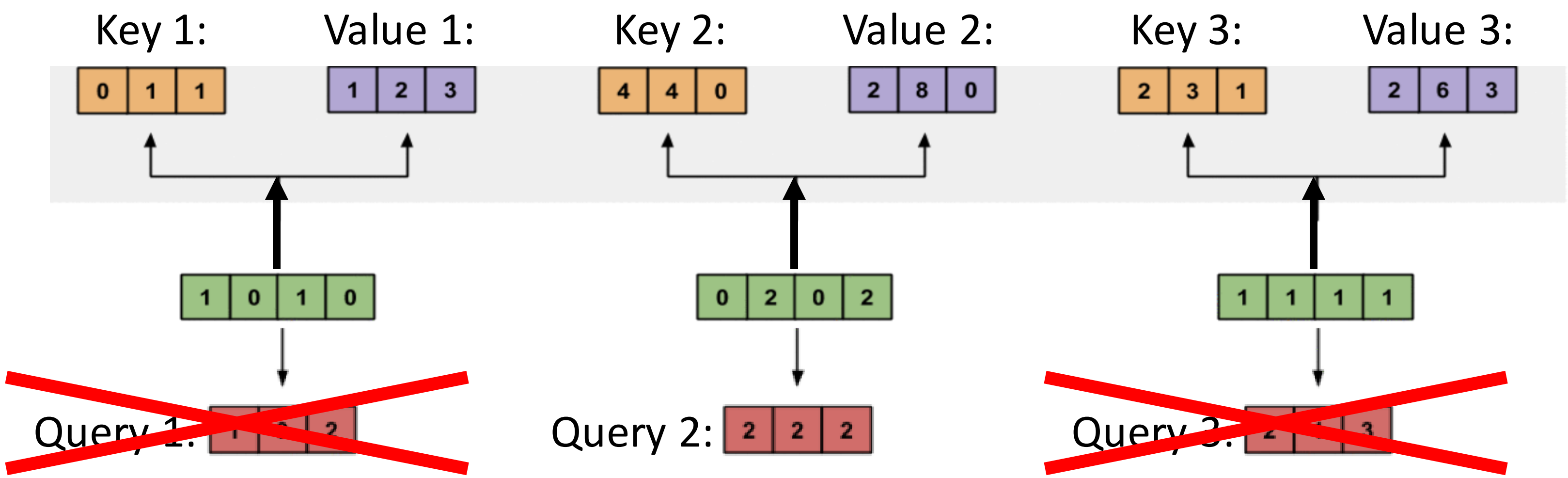
Attention weights amplify input representations (values) that we want to pay attention to and repress the rest



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

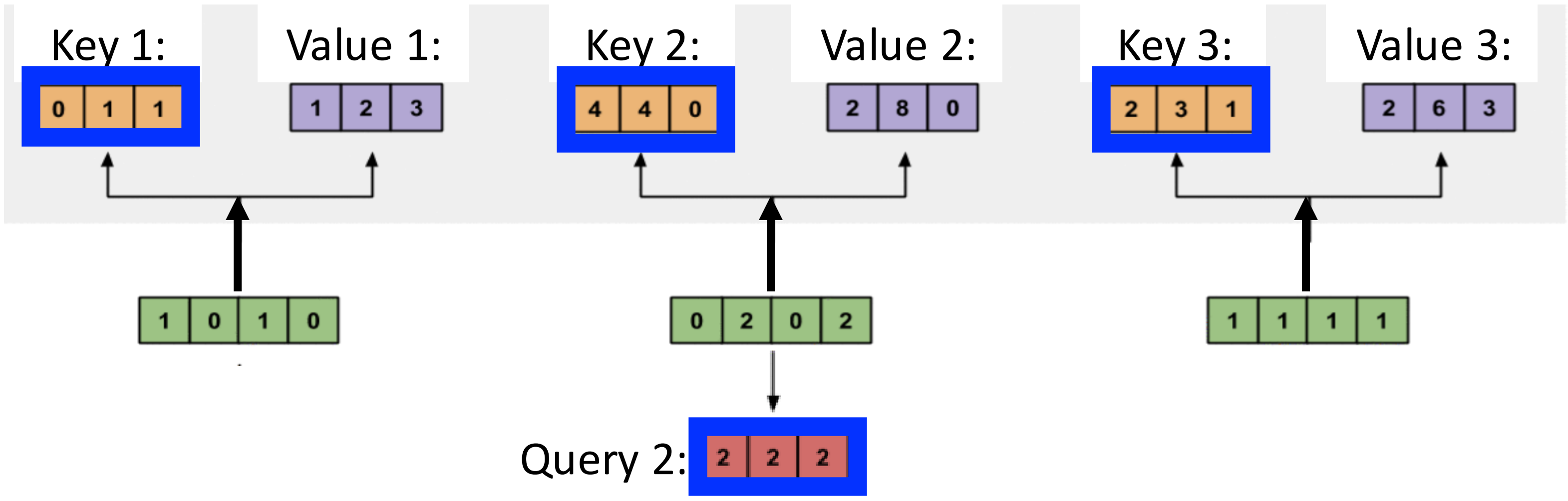


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 2 most related?

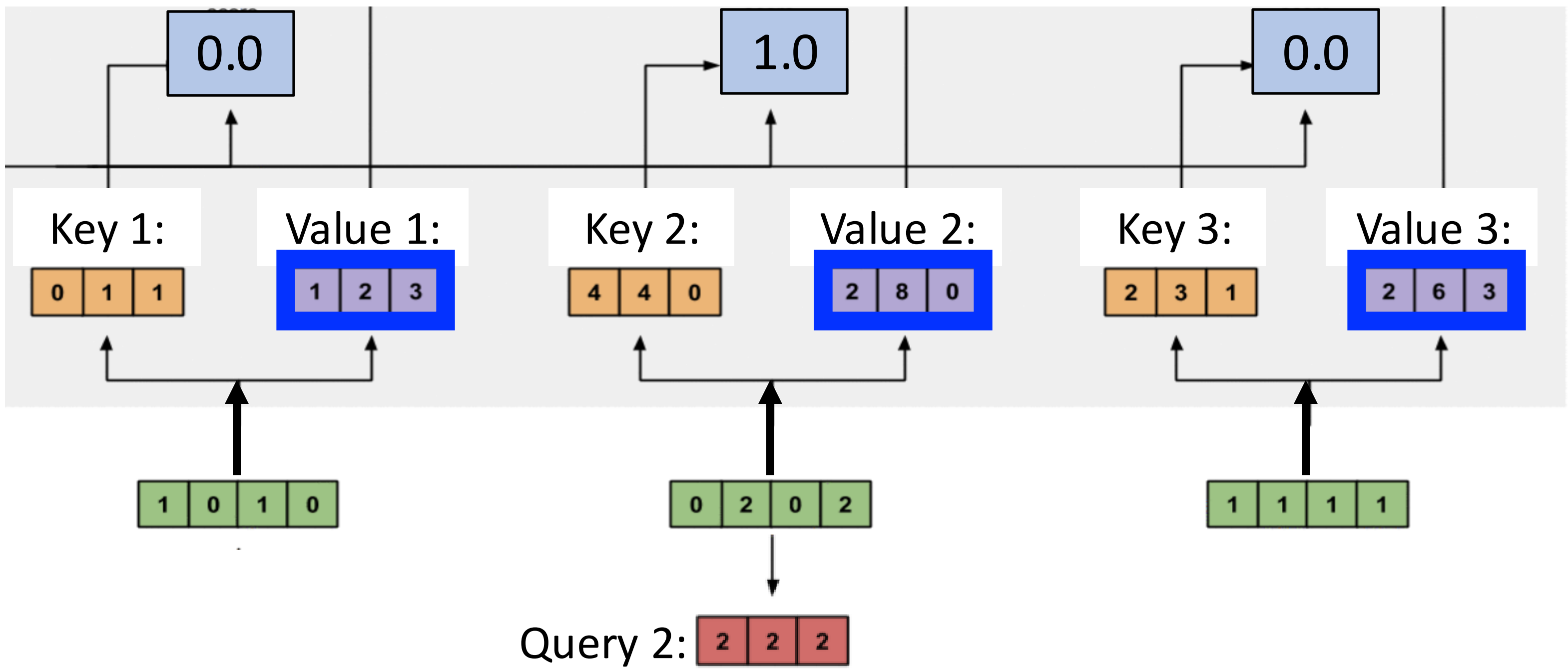


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

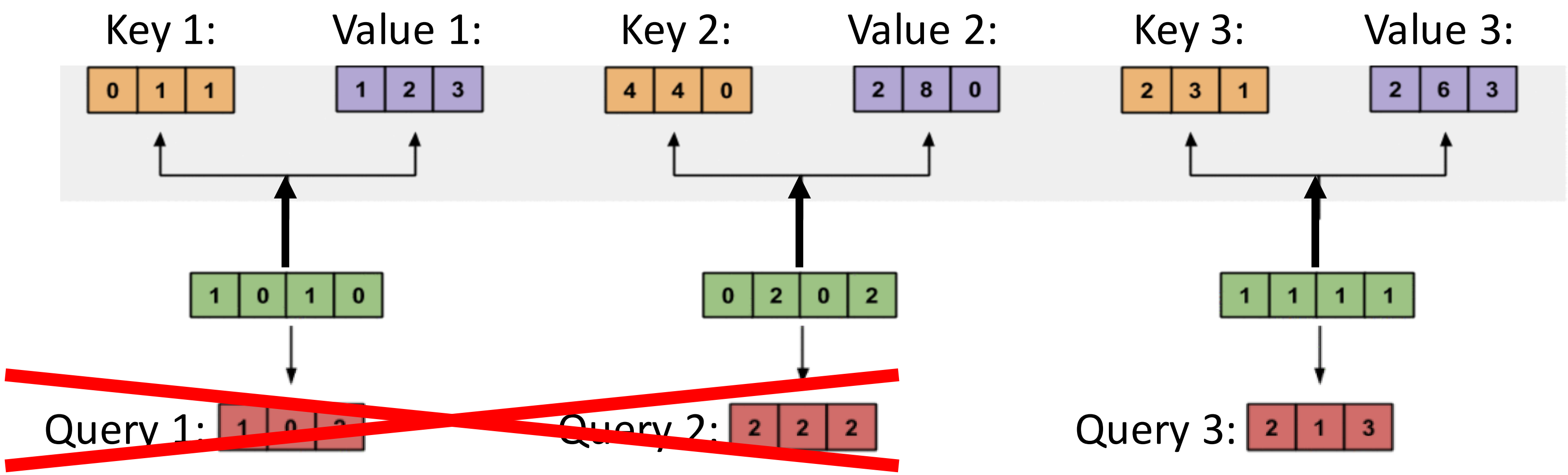
2. Compute weighted sum of values using attention scores



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

Repeat the same process for each remaining input token

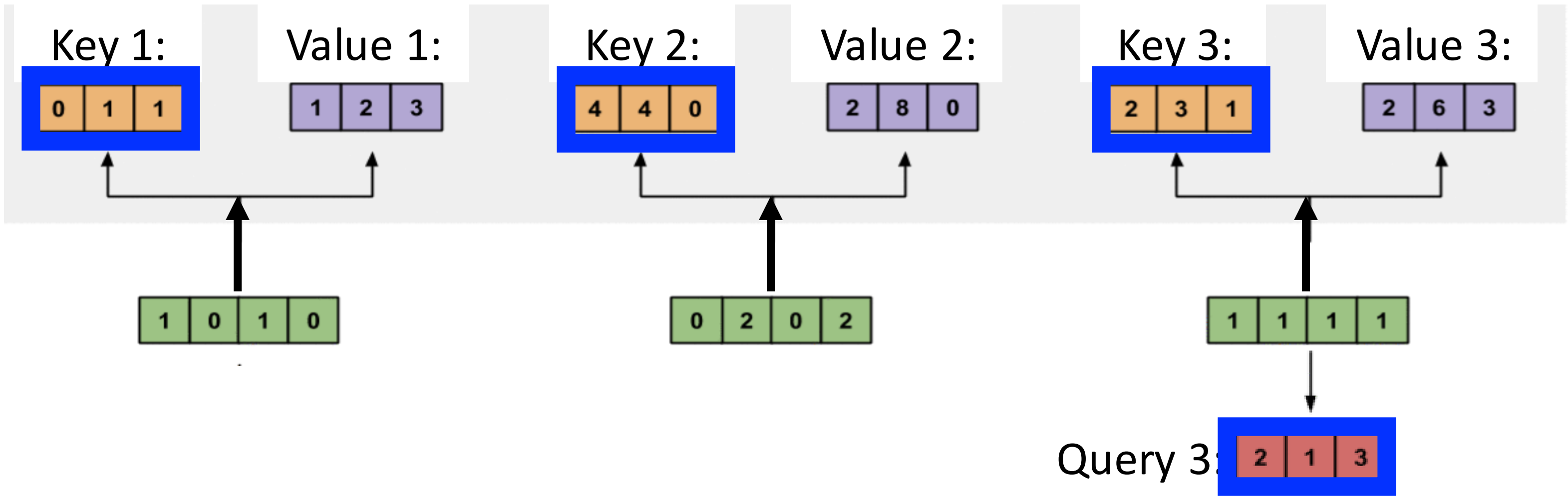


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

1. Compute attention weights
- Softmax resulting 3 scores from query x keys

To which input(s) is input 3 most related?

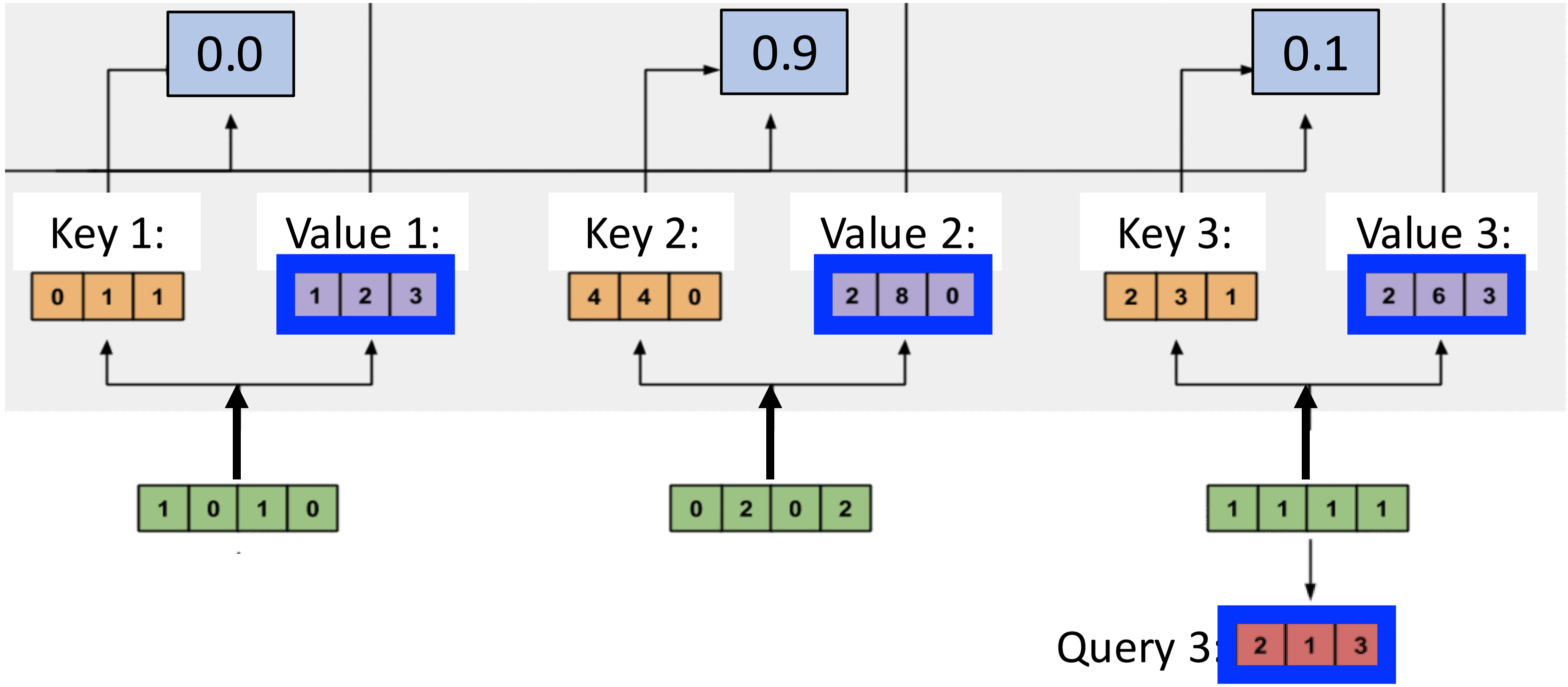


https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example

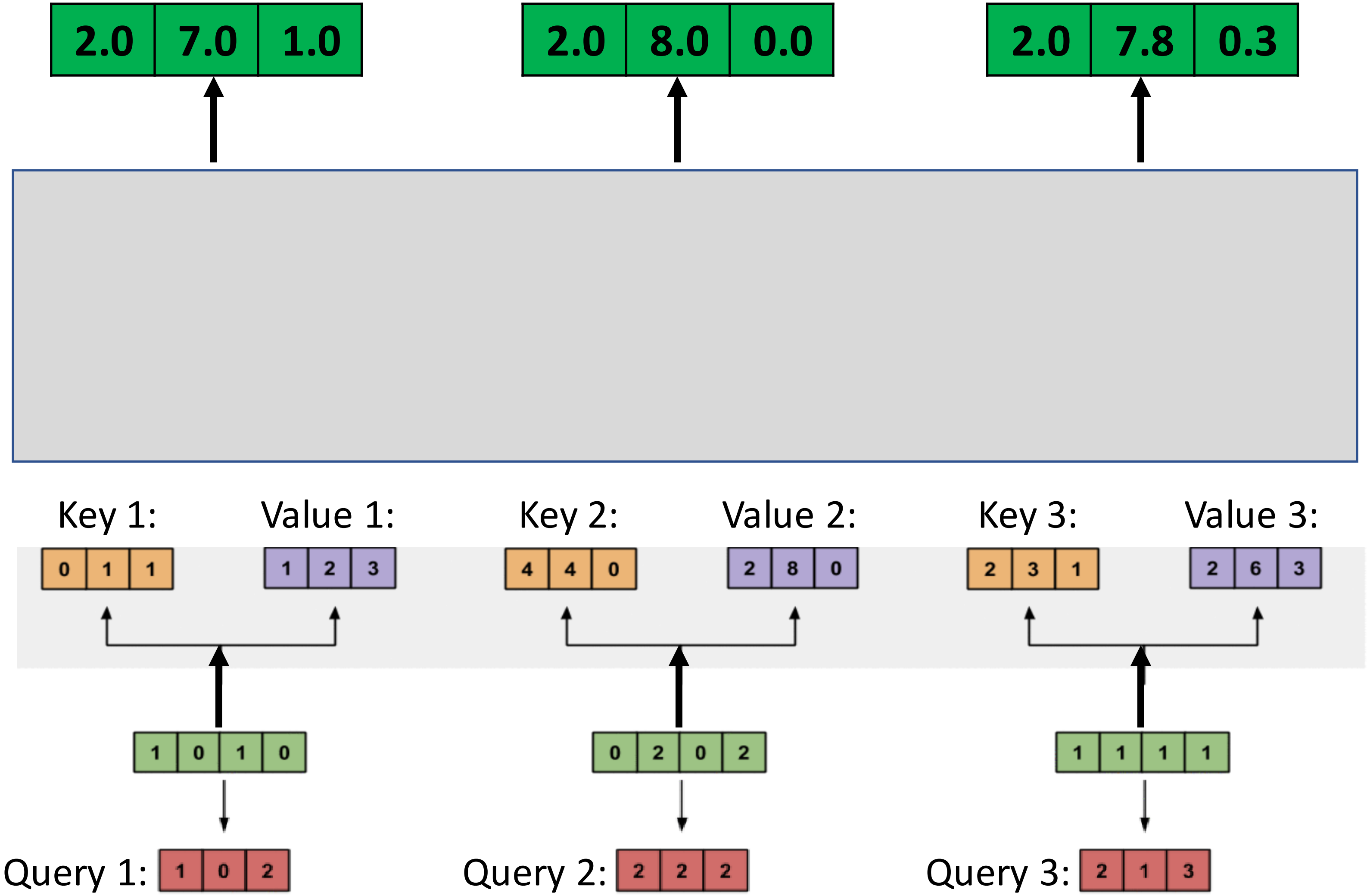
1. Compute attention weights
- Softmax resulting 3 scores from query x keys

2. Compute weighted sum of values using attention scores



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Computing Self-Attention: Example



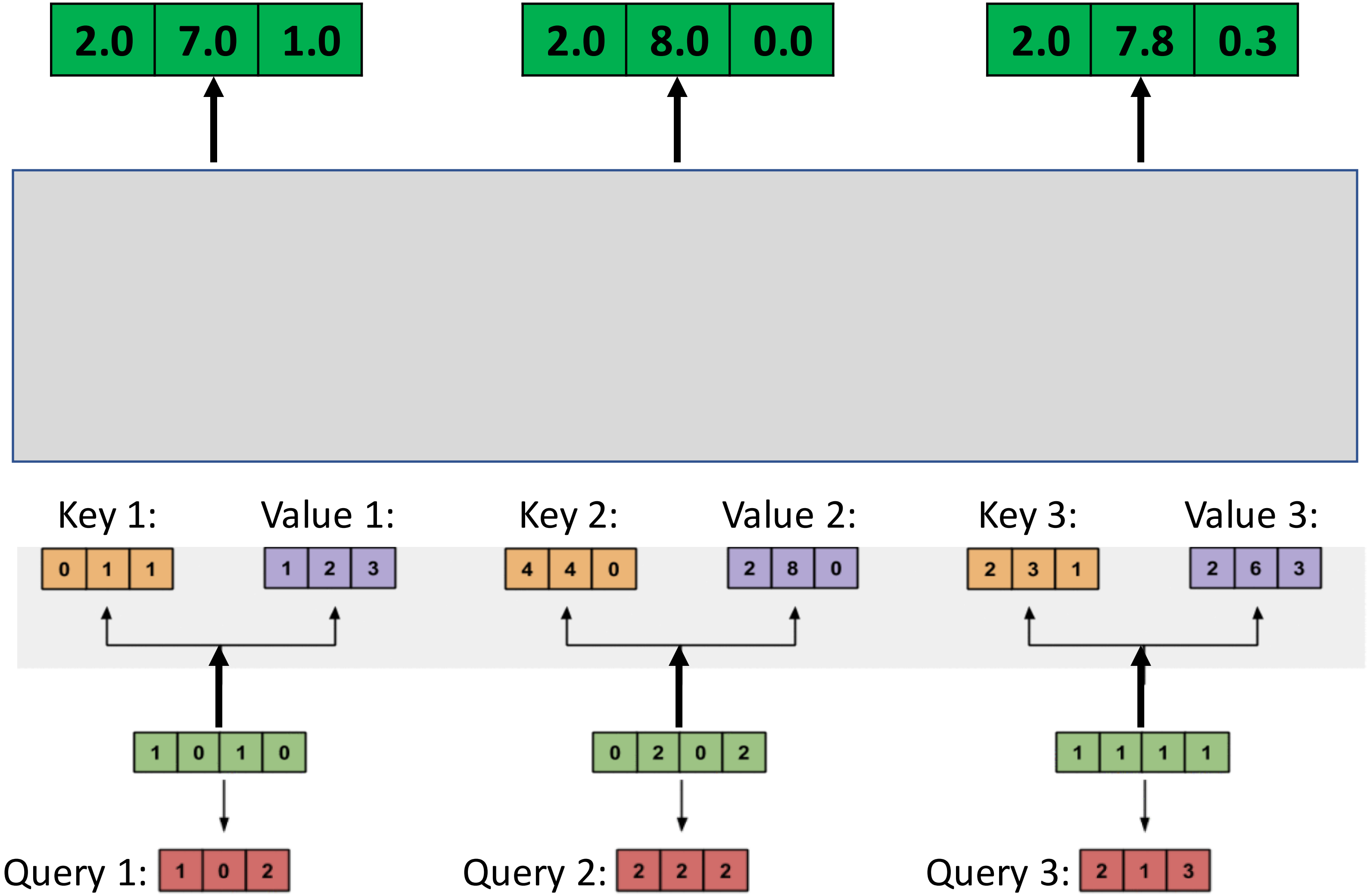
https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Hyperparameters

A developer chooses **input token length** and **number of matrices' columns** (and thus vector sizes)

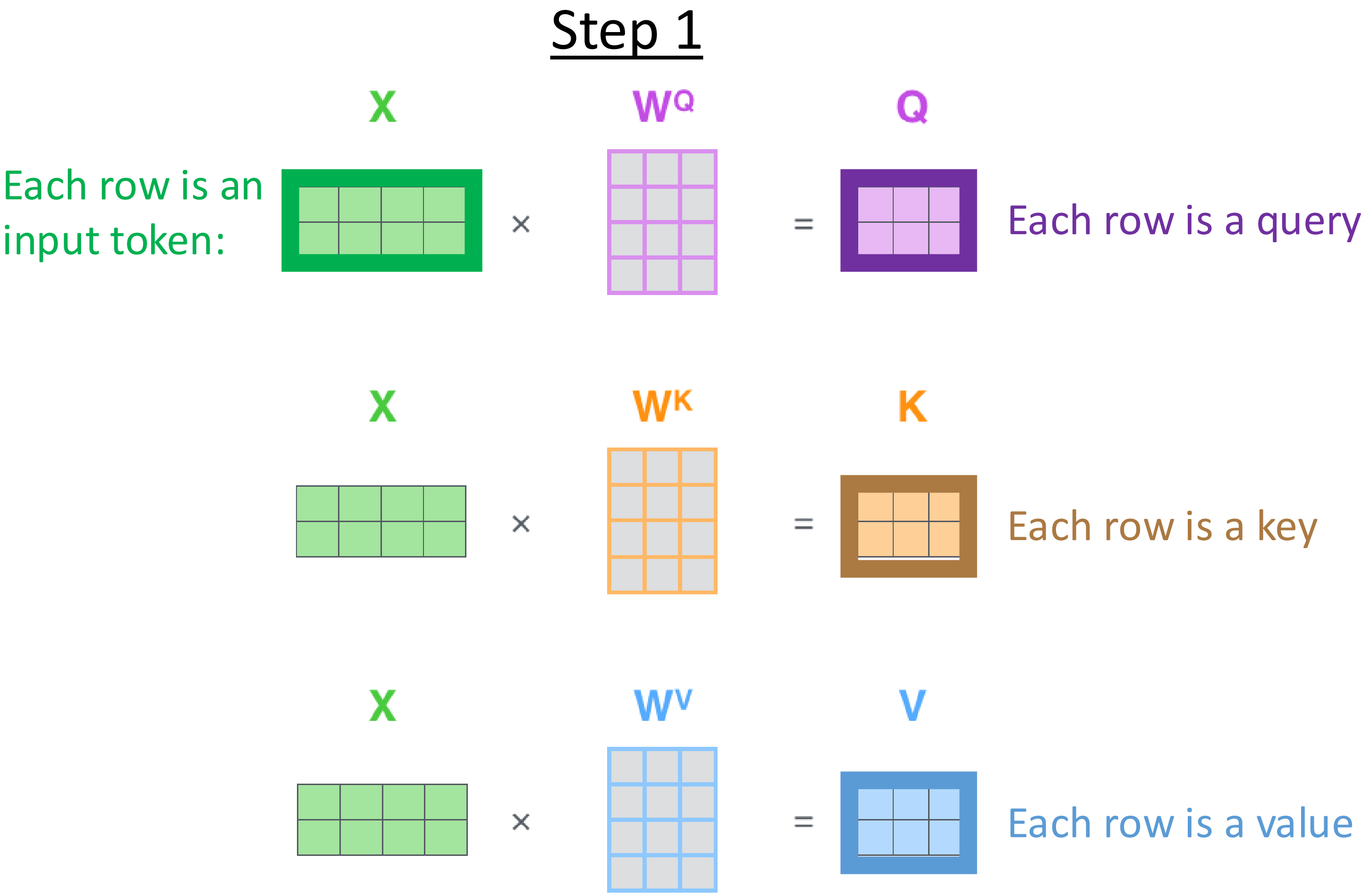
Dimension of **query** and **key** must match to assess similarity (e.g., dot product).

Dimension of **value** can differ from that of **query** and **key** and is output dimension.



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

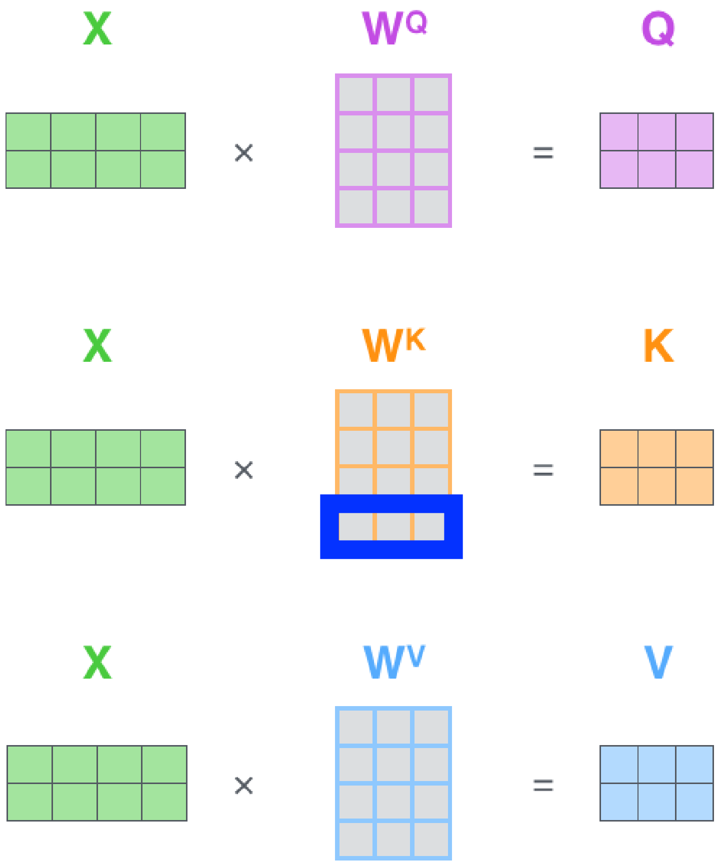
Efficient Computation for Self-Attention



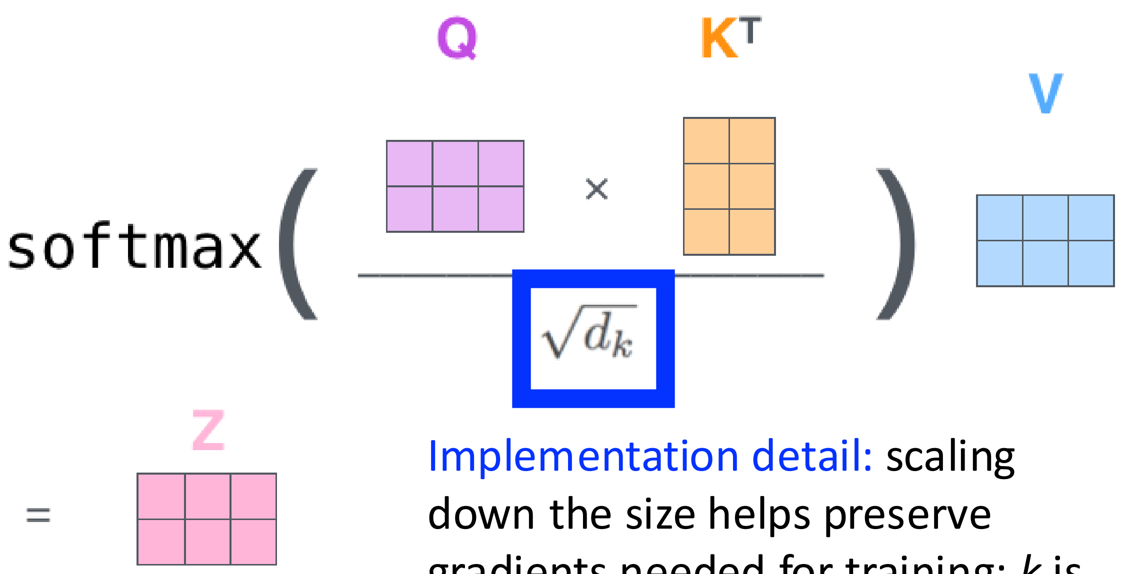
http://jalammar.github.io/illustrated-transformer/

Efficient Computation for Self-Attention

Step 1



Step 2

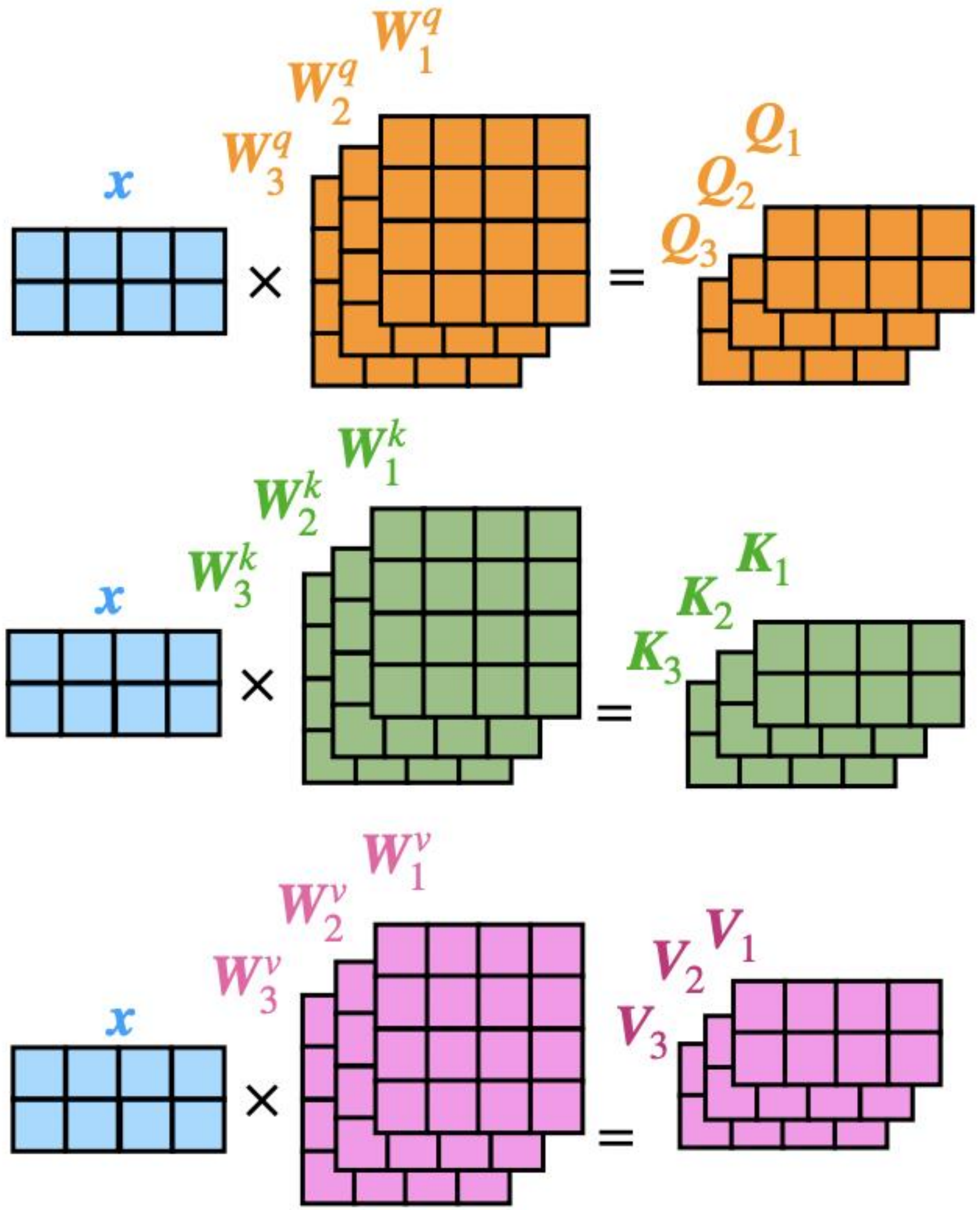


Implementation detail: scaling down the size helps preserve gradients needed for training; *k* is dimensionality of the key vector

http://jalammar.github.io/illustrated-transformer/

Multi-head Attention

- **Goal**: enable each token to relate to other tokens in multiple ways
- **Key idea**: multiple self-attention mechanisms, each with their own key, value and query matrices



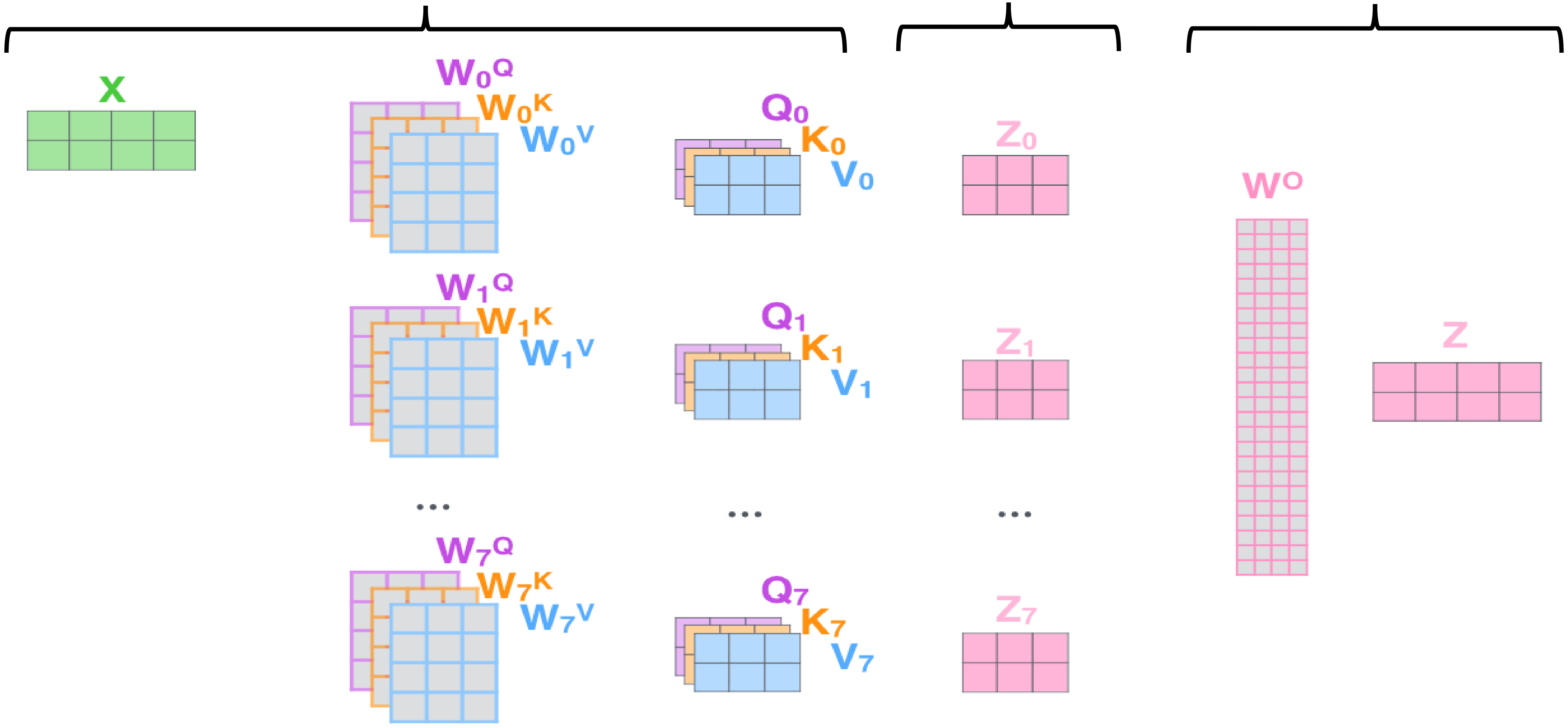
https://sebastianraschka.com/pdf/lecture-notes/stat453ss21/L19_seq2seq_rnn-transformers__slides.pdf

Multi-head Attention

1) Create query, key, and value vectors for all attentions heads

2) Compute new input representations

3) Condense all representations into a single representation by concatenating **z**-s and multiplying by a weight matrix



http://jalammar.github.io/illustrated-transformer/

Trained Multi-head Attention Examples

Figure shows two columns of attention weights for the first two attention heads

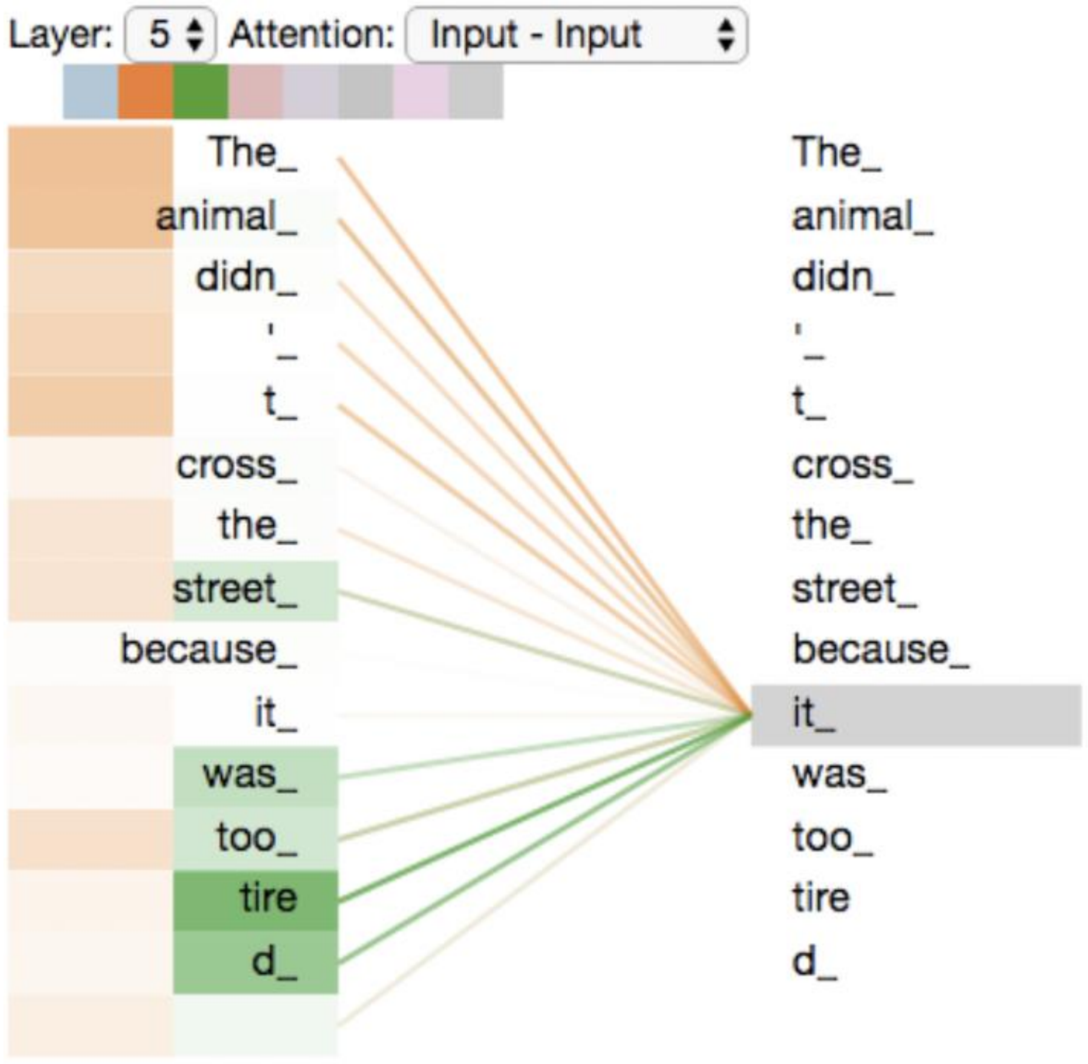
- Darker values signify larger attention scores

What does “it” focus on most in the first attention head?

- The animal (e.g., represents what is “it”)

What does “it” focus on most in the second attention head?

- tired (e.g., represents how “it” feels); note, a tokenizer was used that separates “tire” and “d”



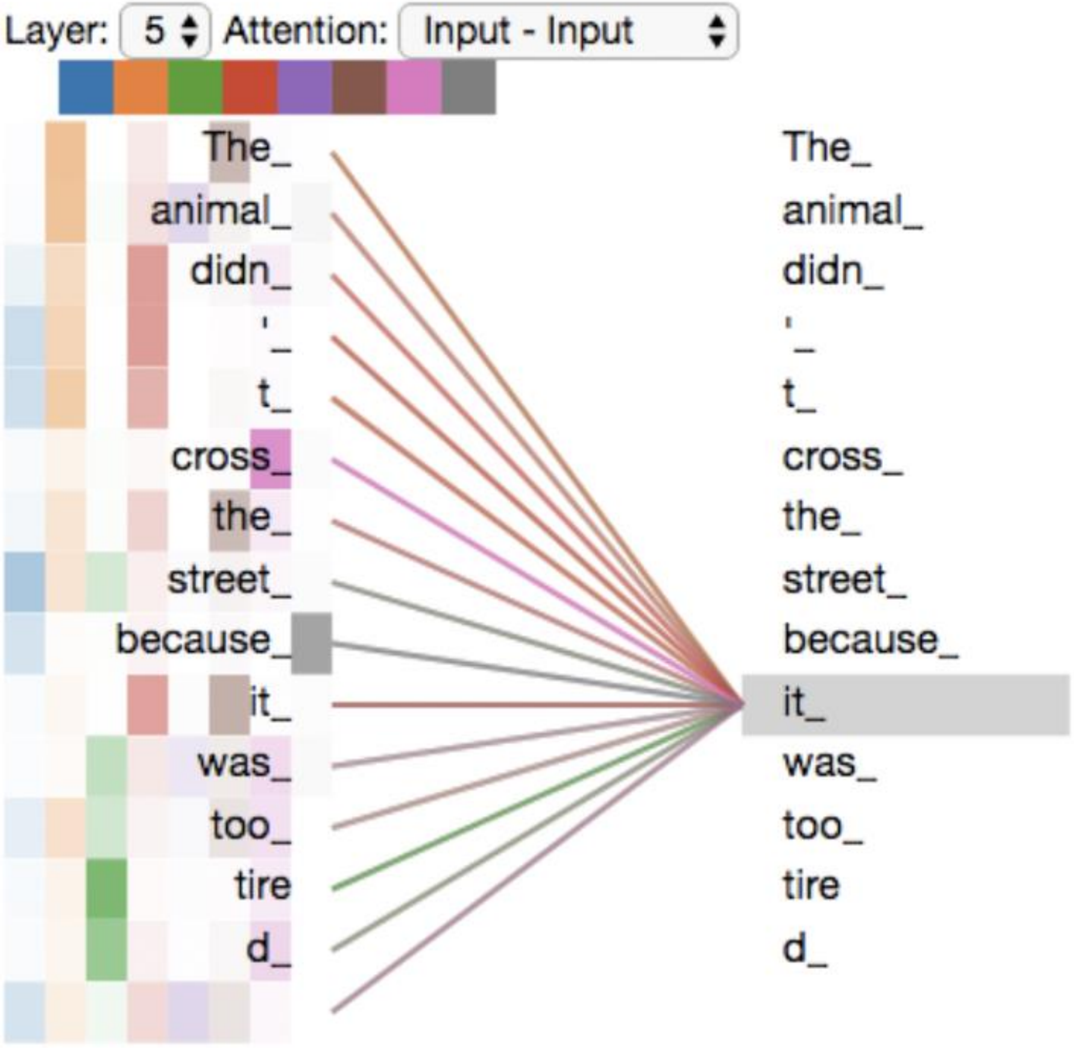
http://jalammar.github.io/illustrated-transformer/

Trained Multi-head Attention Examples

Figure shows five columns of attention weights for five attention heads

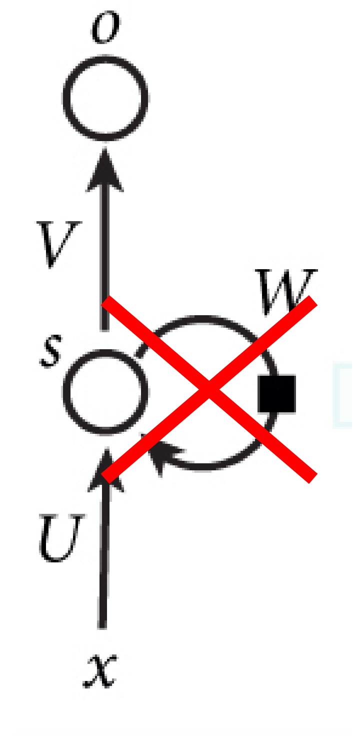
- Darker values signify larger attention scores

Attention weights may be hard to interpret

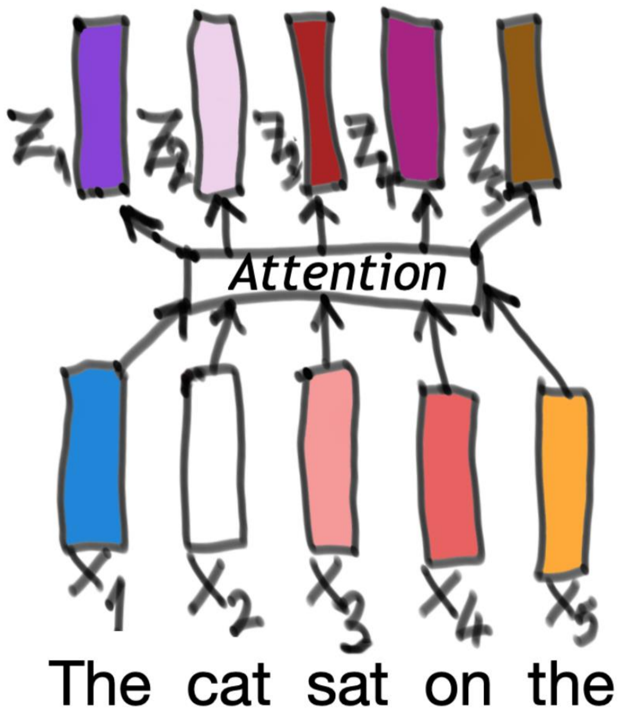


http://jalammar.github.io/illustrated-transformer/

Self-Attention vs RNN: Propagates Information About Other Inputs Without Recurrent Units



http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

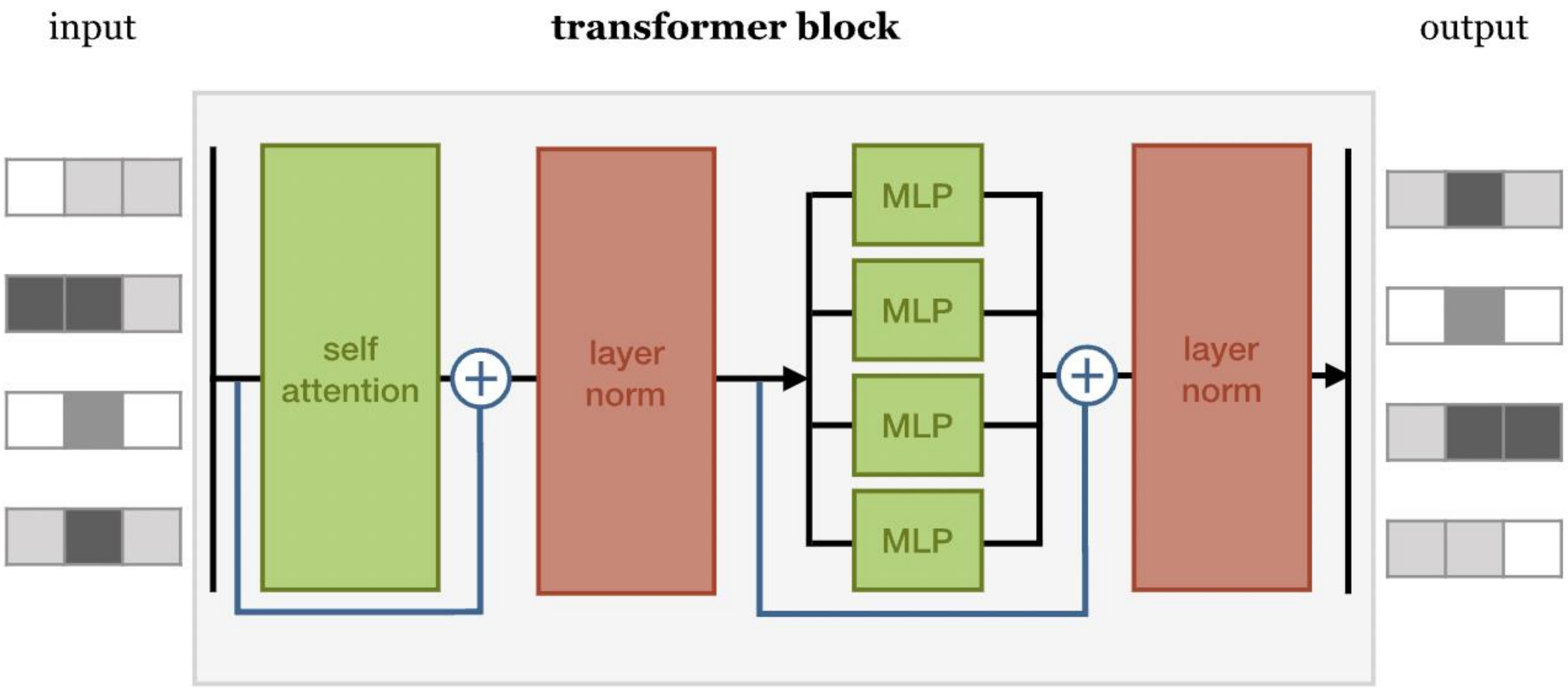


https://towardsdatascience.com/self-attention-5b95ea164f61

Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

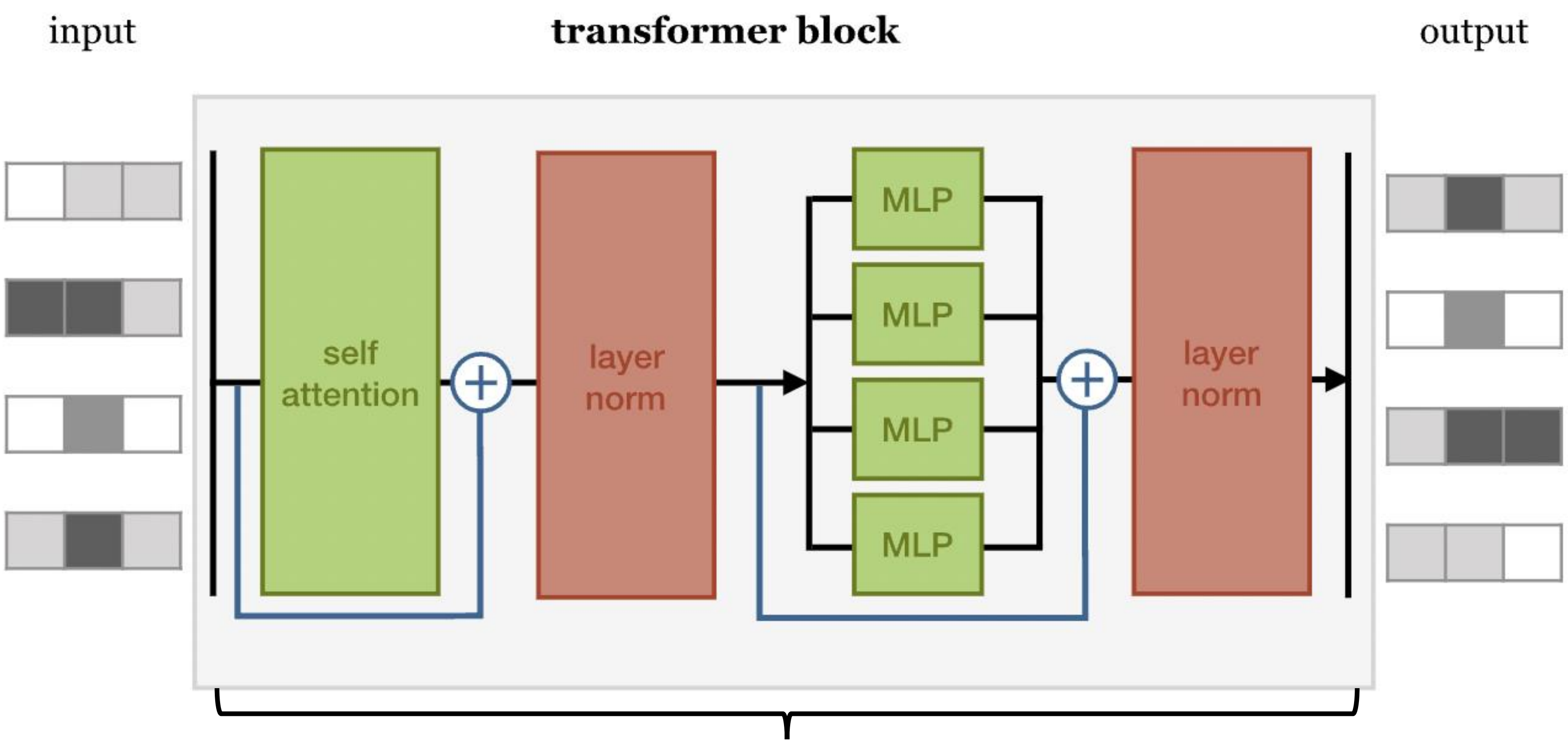
Typical Transformer Block



Architectures often chain together multiple transformer blocks, like that shown here

http://peterbloem.nl/blog/transformers

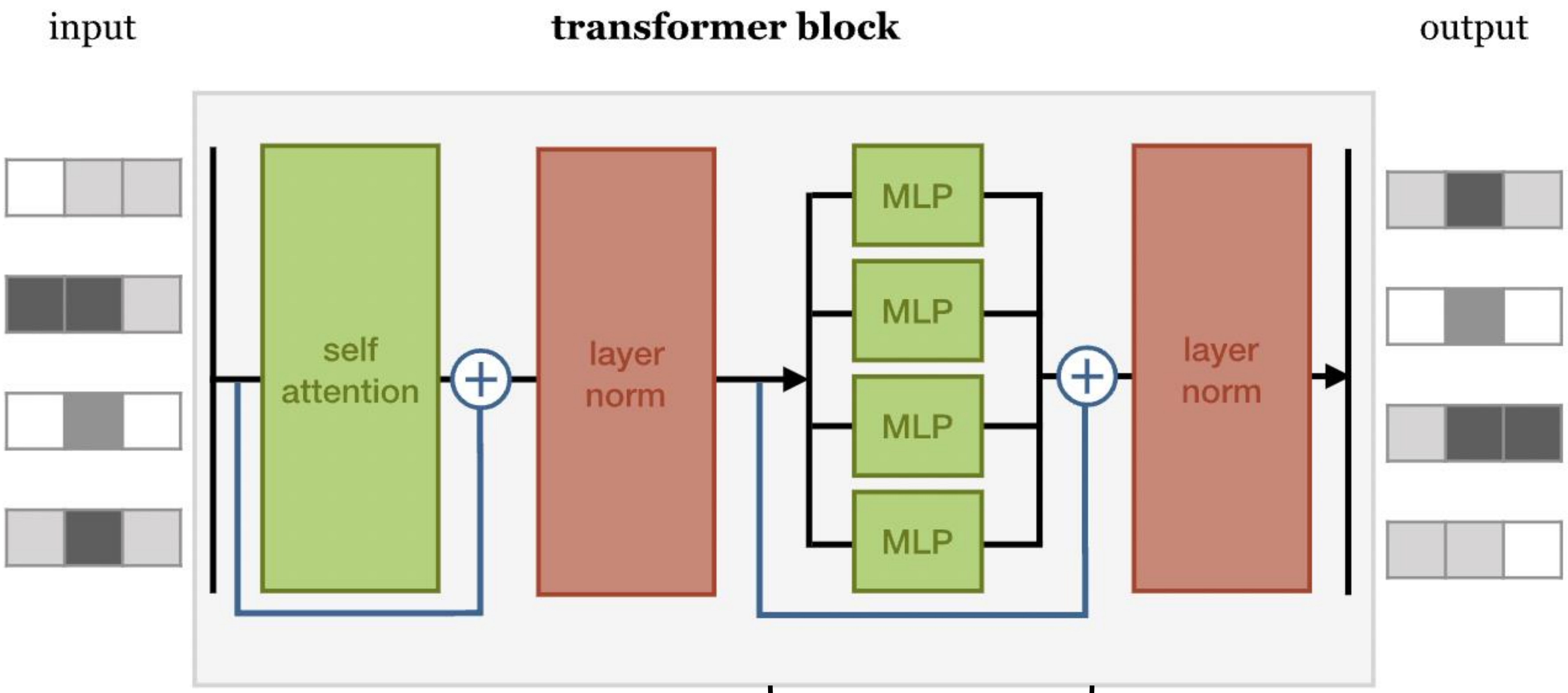
Typical Transformer Block



Layer normalization and residual connections improve training (i.e., faster and better results)

http://peterbloem.nl/blog/transformers

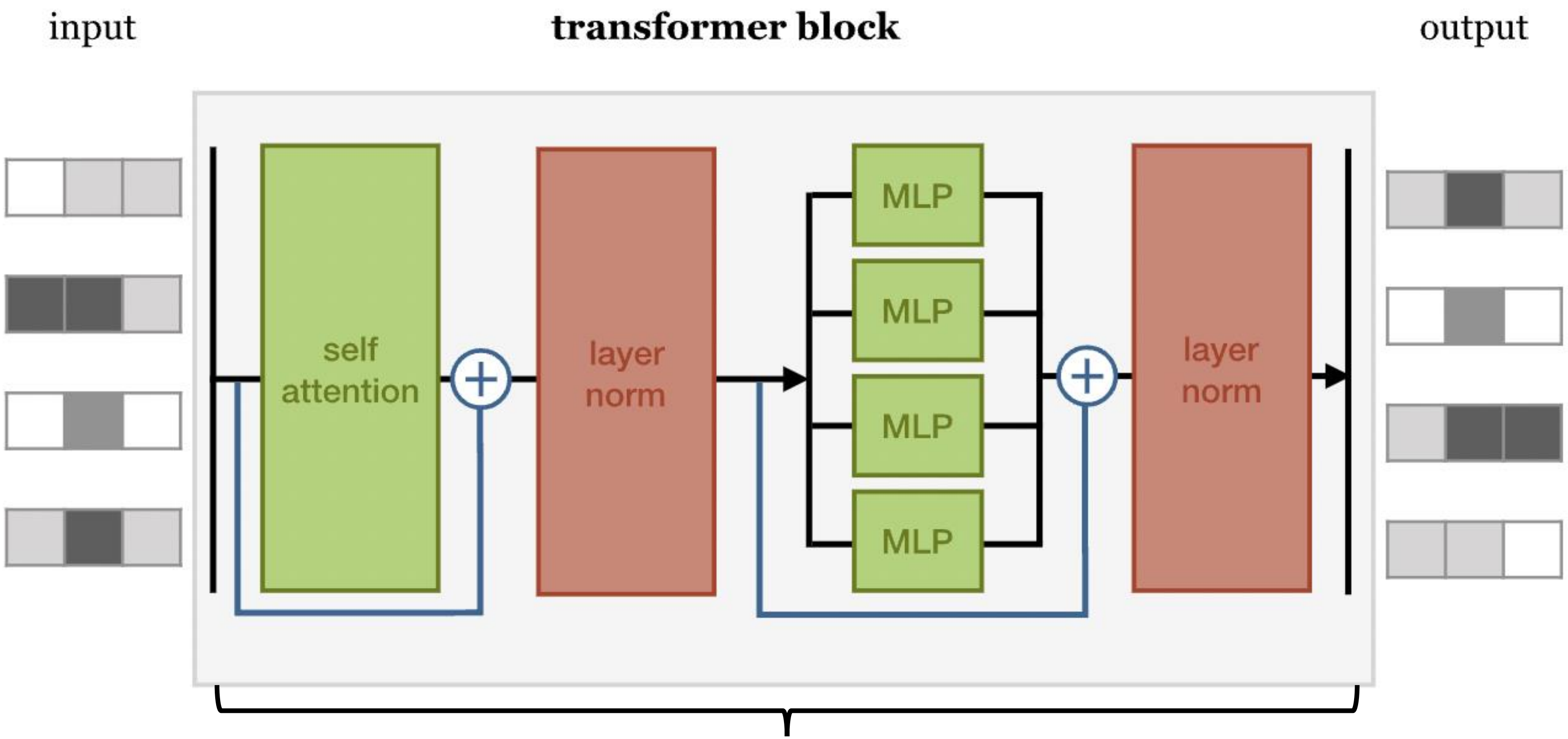
Typical Transformer Block



Feedforward layer per input

http://peterbloem.nl/blog/transformers

Typical Transformer Block

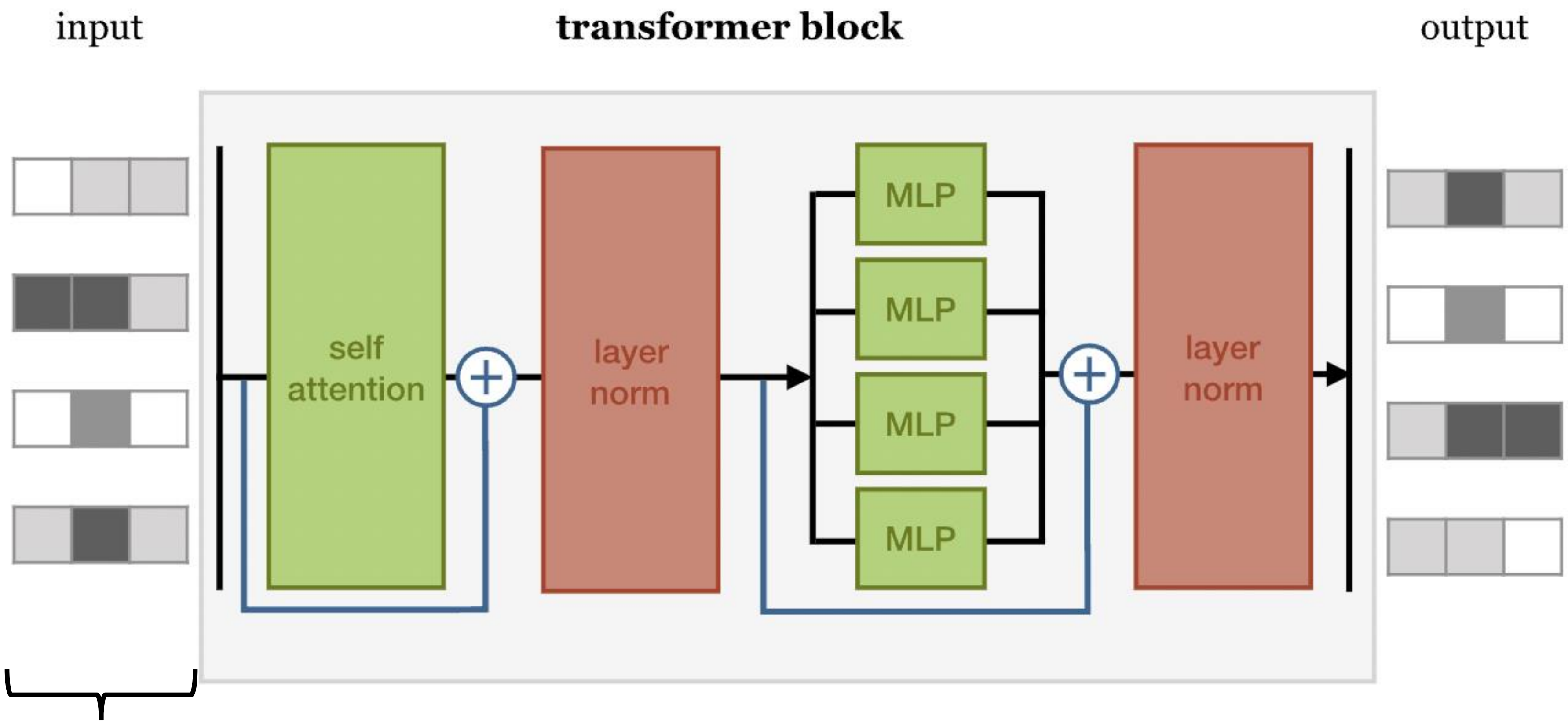


Where are non-linearities introduced in this block?

- self-attention's softmax, MLP's activation functions, layer norms

http://peterbloem.nl/blog/transformers

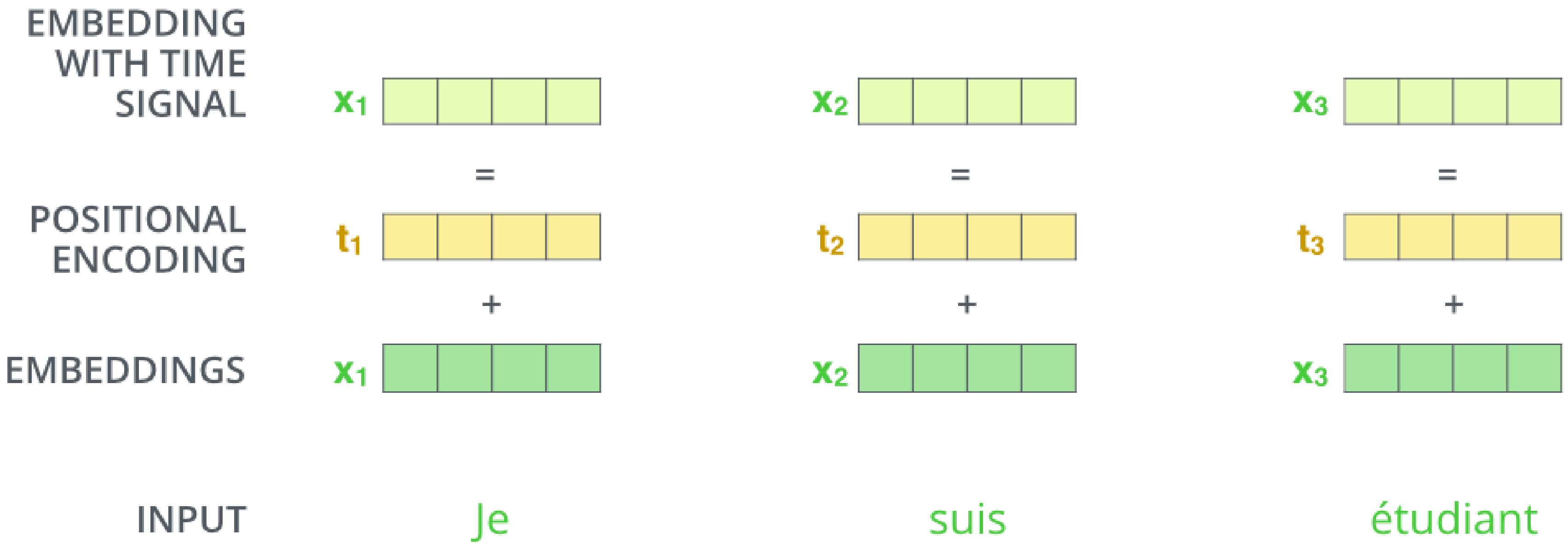
Challenge: Transformers Lack Sensitivity to the Order of the Input Tokens



Input: a *set* and so shuffling order of input tokens results yields same outputs except in the same shuffled order
(i.e. self-attention is *permutation equivariant*)

http://peterbloem.nl/blog/transformers

Solution: Add Position as Input to Transformer



- Options:
  - **Position embeddings**: created by training with sequences of every length during training
  - **Position encodings**: a function mapping positions to vectors that the network learns to interpret (enables generalization to lengths not observed during training)

http://jalammar.github.io/illustrated-transformer/

Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

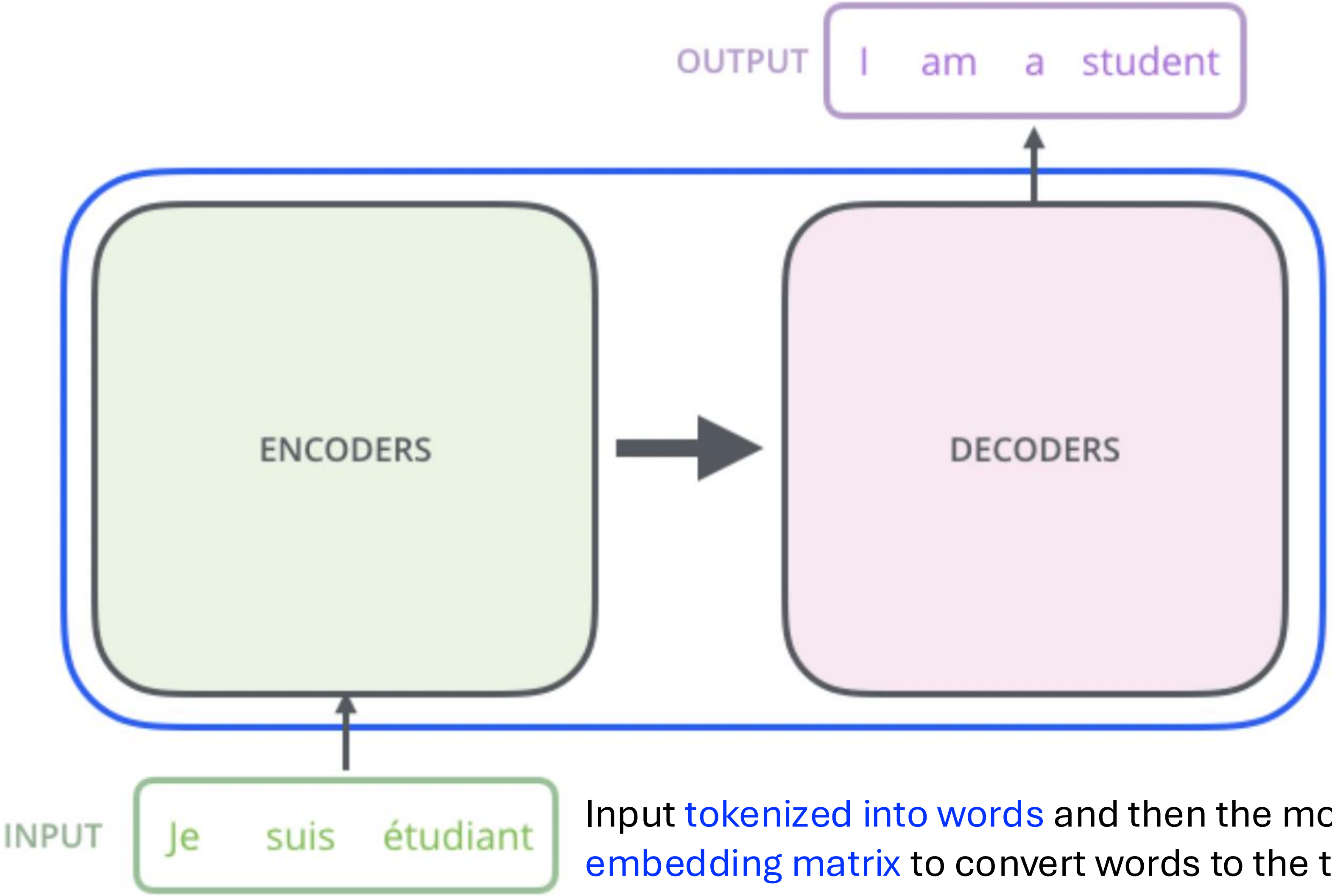
Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

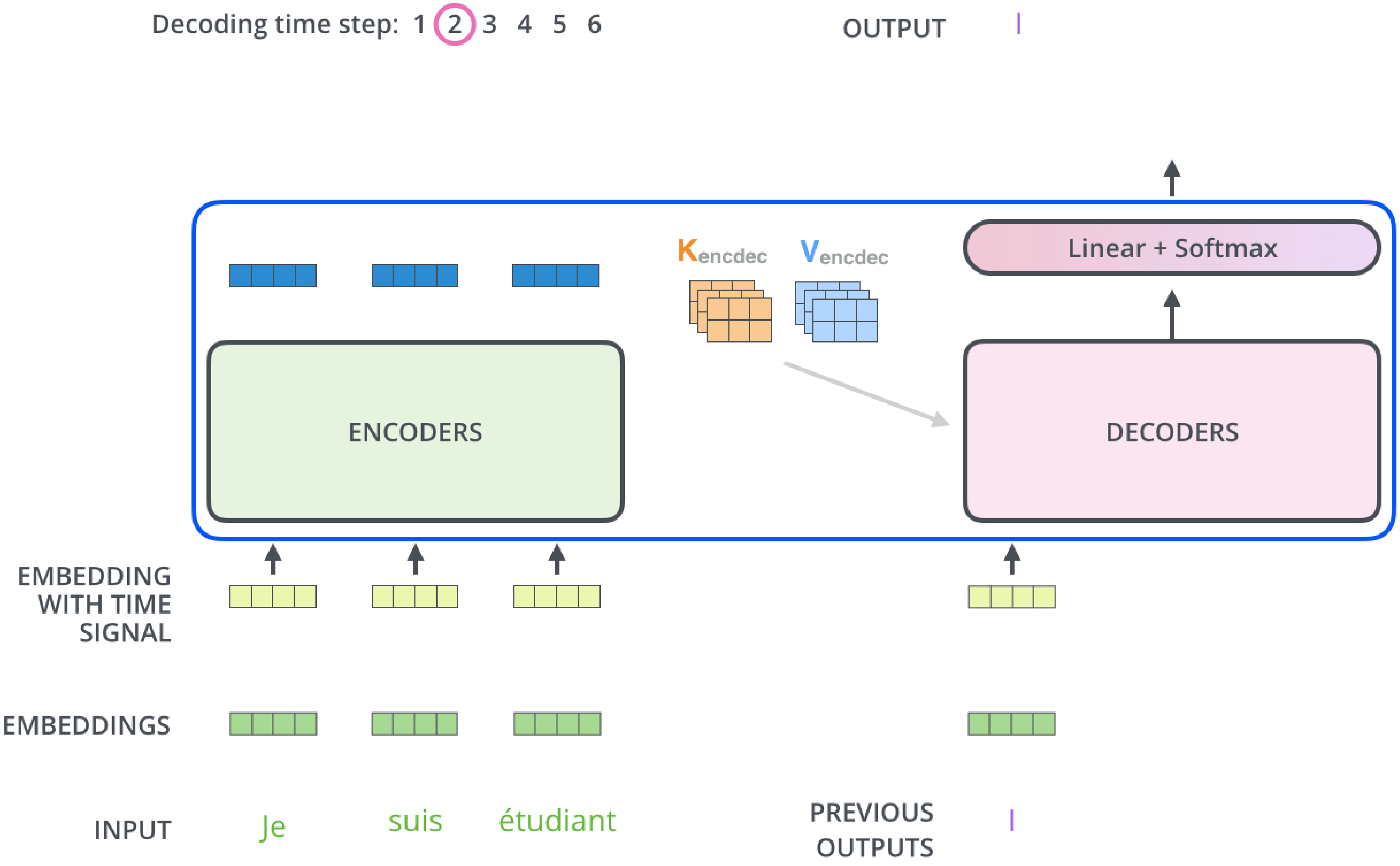
Target Application: Machine Translation



Input tokenized into words and then the model learns an embedding matrix to convert words to the target embedding size

http://jalammar.github.io/illustrated-transformer/

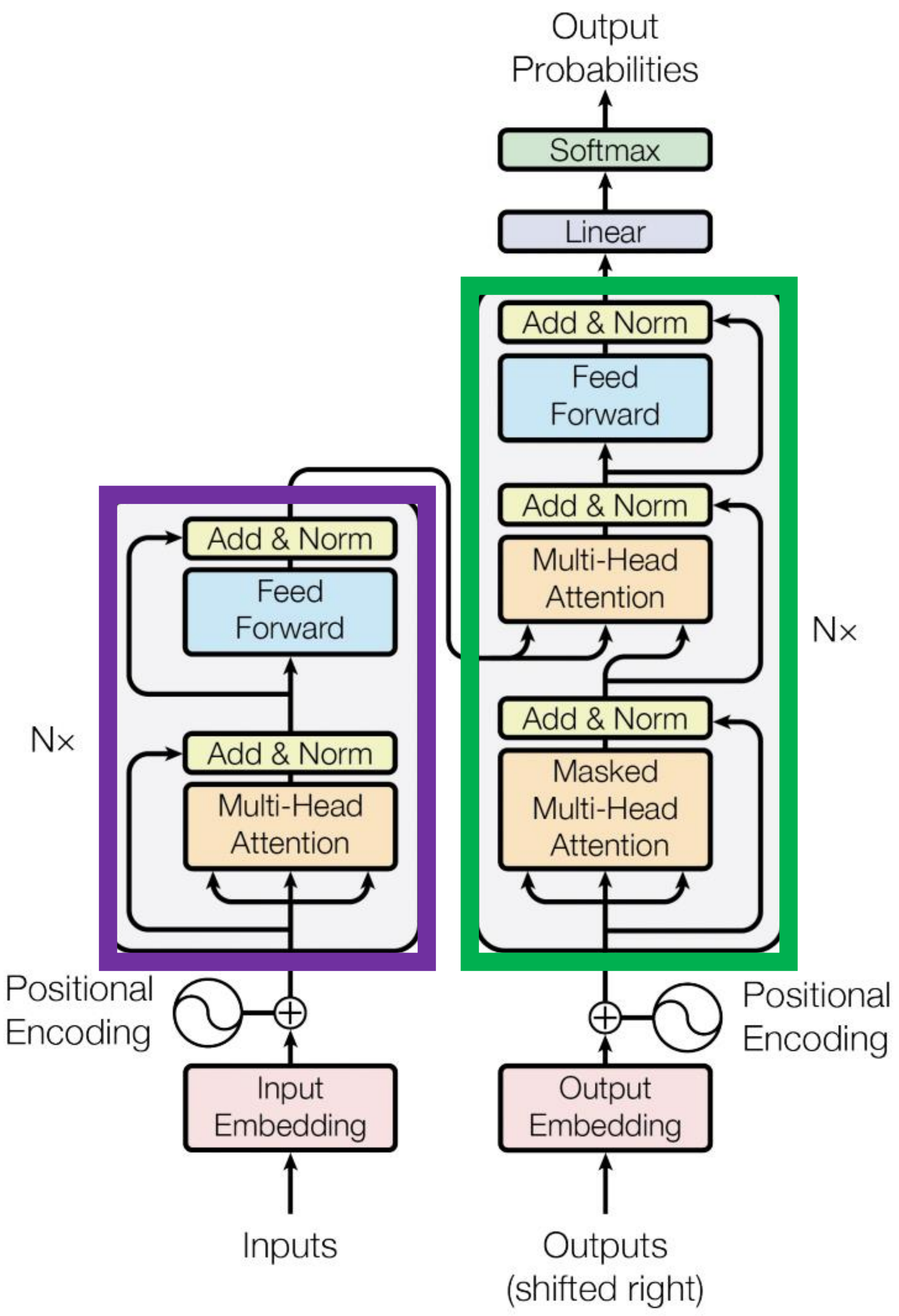
Example: Autoregressive Model



http://jalammar.github.io/illustrated-transformer/

Architecture

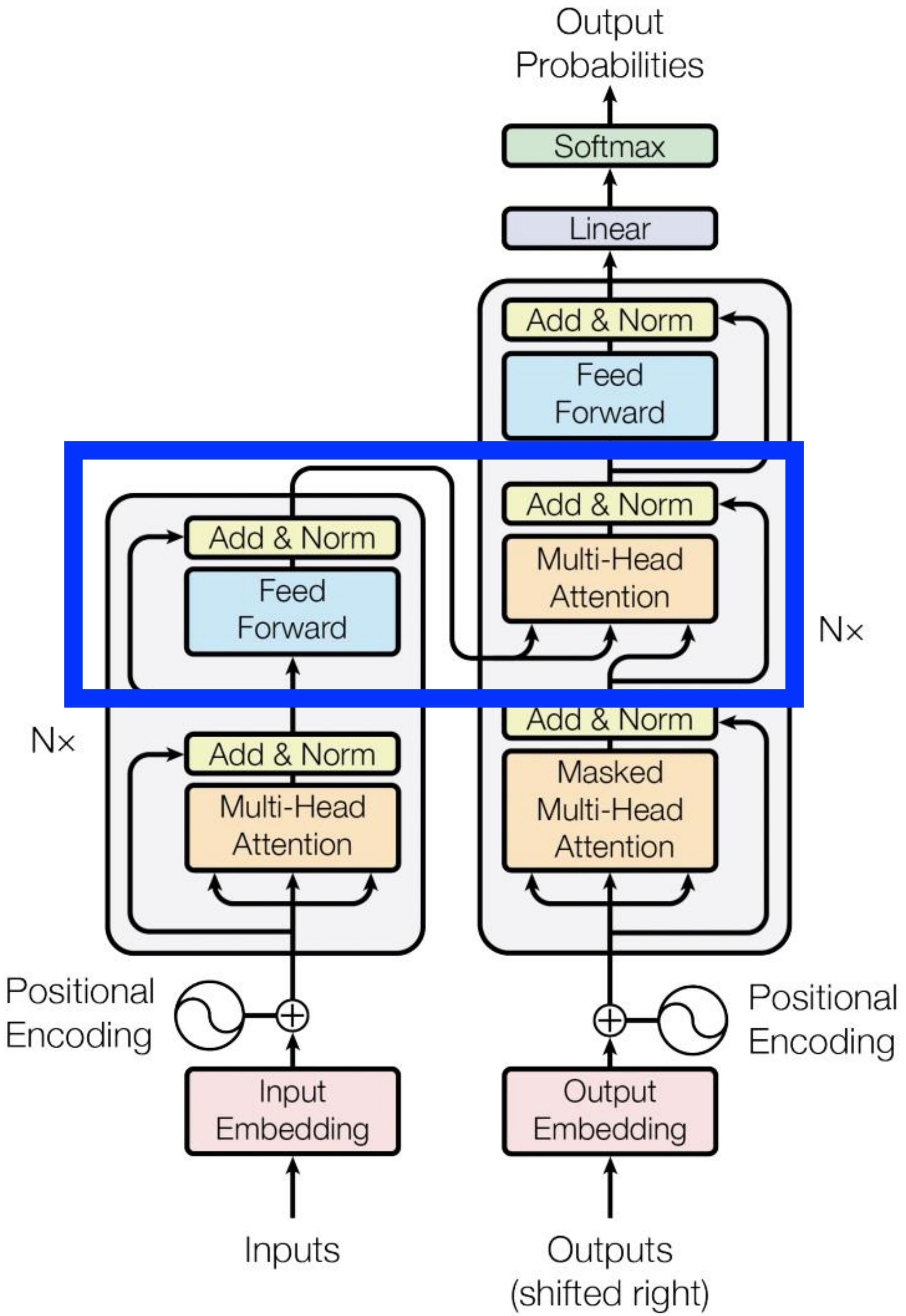
- Key Ingredient: Self-Attention
  - Used in both the encoder (provides context for translation) and decoder (translates)
- Other ingredients
  - Positional encoding
  - Layer normalization
  - Residual connections
  - Feed forward layers
- Nx = 6 chained blocks (encoder & decoder)



Vaswani et al. Attention Is All You Need. Neurips 2017

Architecture

Decoder can attend to all inputs via cross-attention (i.e., keys and values come from the encoder and query comes from the decoder)



Vaswani et al. Attention Is All You Need. Neurips 2017

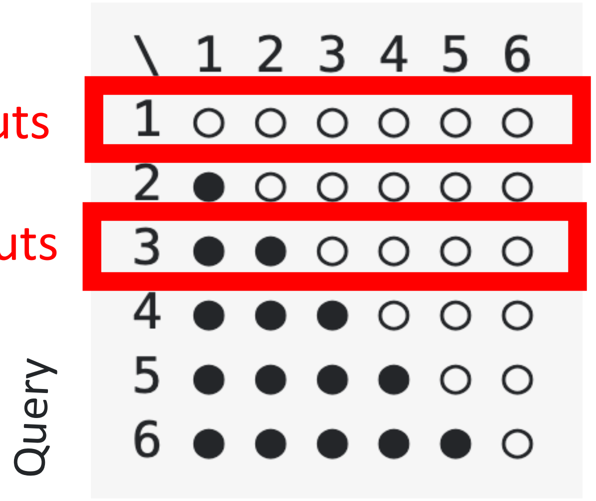
Architecture

Masking so decoder ONLY sees earlier predictions
(i.e., masked values set to –infinity so softmax output is 0)

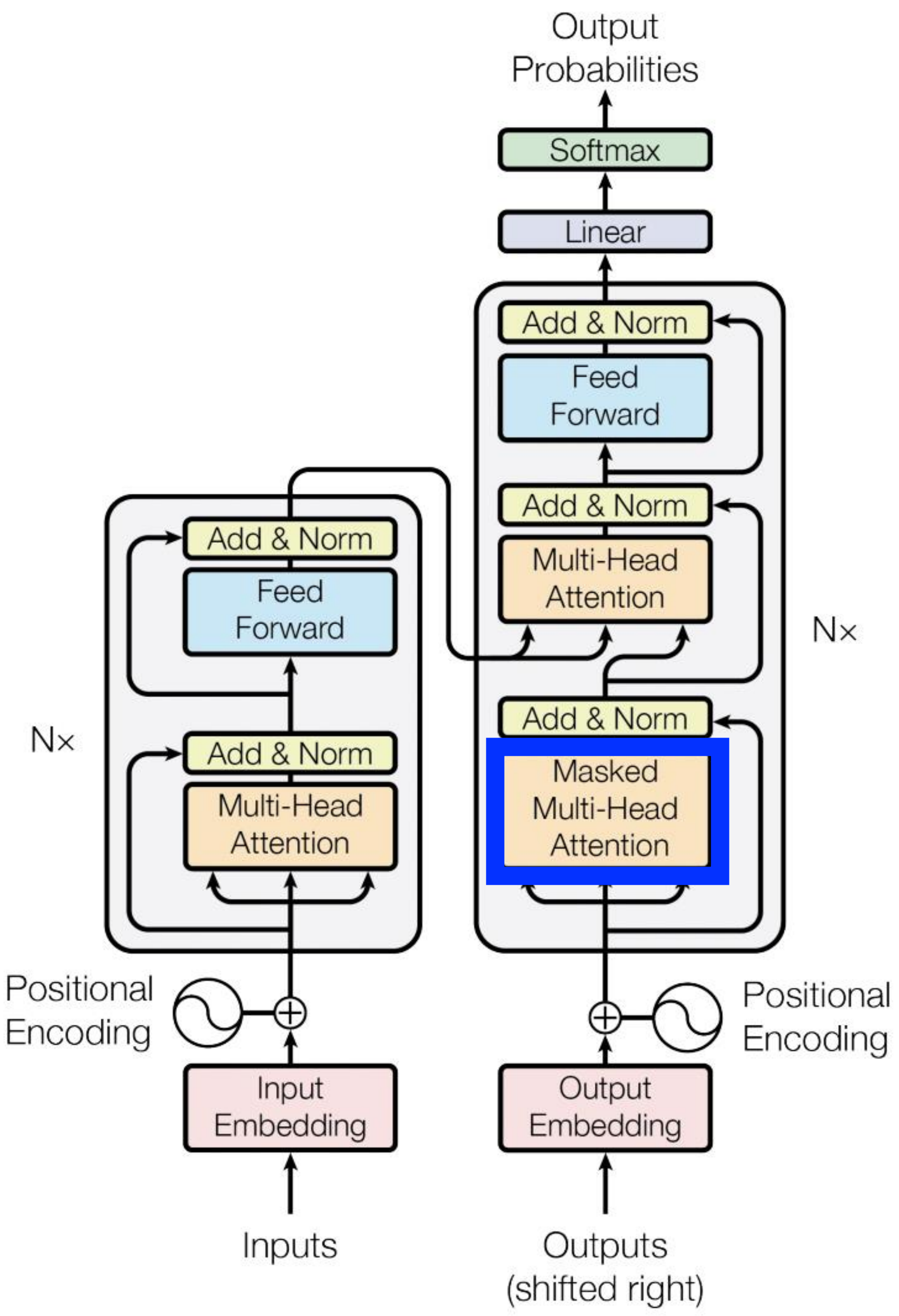
Query x Key

e.g., at start, no previous inputs

e.g., at 3rd step, two previous inputs

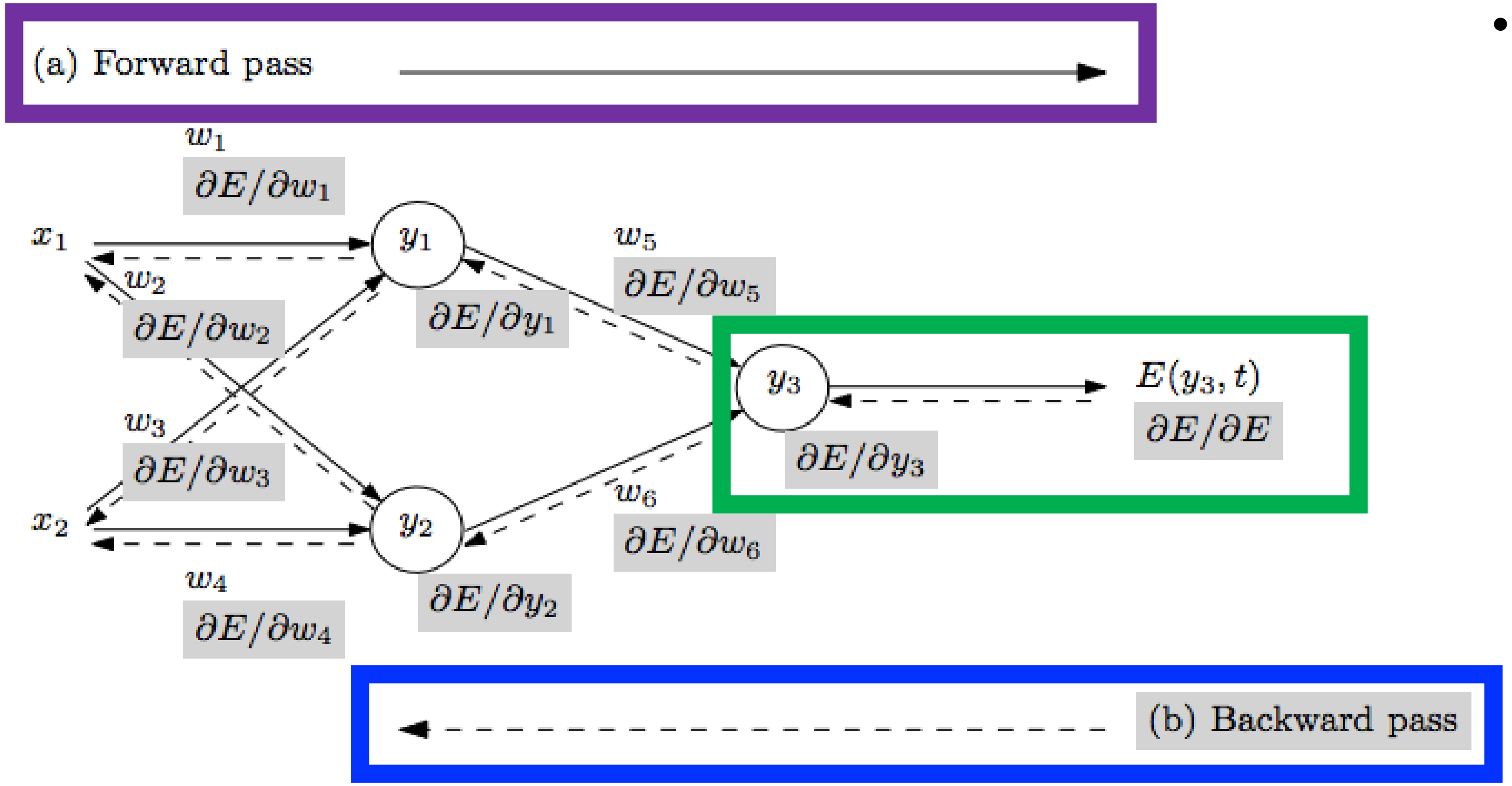


https://stackoverflow.com/questions/64799622/how-is-the-gpts-masked-self-attention-is-utilized-on-fine-tuning-inference



Vaswani et al. Attention Is All You Need. Neurips 2017

Training Procedure: 3.5 days on 8 NVIDIA P100s



- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make predictions
  2. **Error quantification**: measure error of the model's predictions on training data using a loss function
  3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
  4. Update each parameter using calculated gradients

For two tested datasets, achieved state-of-the-art performance:
(1) English-German, with ~4.5 million sentence pairs (byte-pair encoded)
(2) English-French, with 36M sentences

Baydin et al. Automatic Differentiation in Machine Learning: a Survey. 2018

Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

Today’s Topics

- Transformer overview
- Self-attention
- Common transformer ingredients
- Pioneering transformer: machine translation
- Programming tutorial

The End