LLMs

Transformers

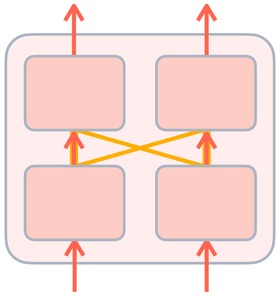
Janik Euskirchen

Part I

Recurrent Networks, Seq2Seq and Attention

Part II

Transformers



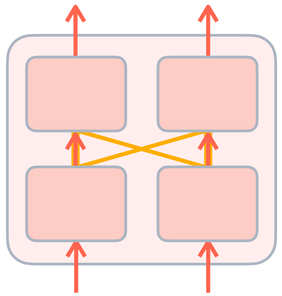
Part III

State Space Models

???

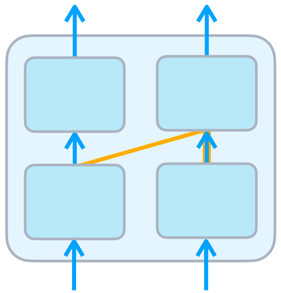
1

Transformer Encoder



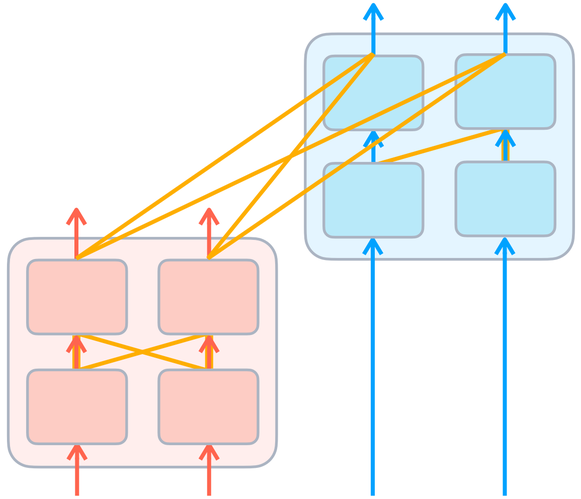
2

Transformer Decoder



3

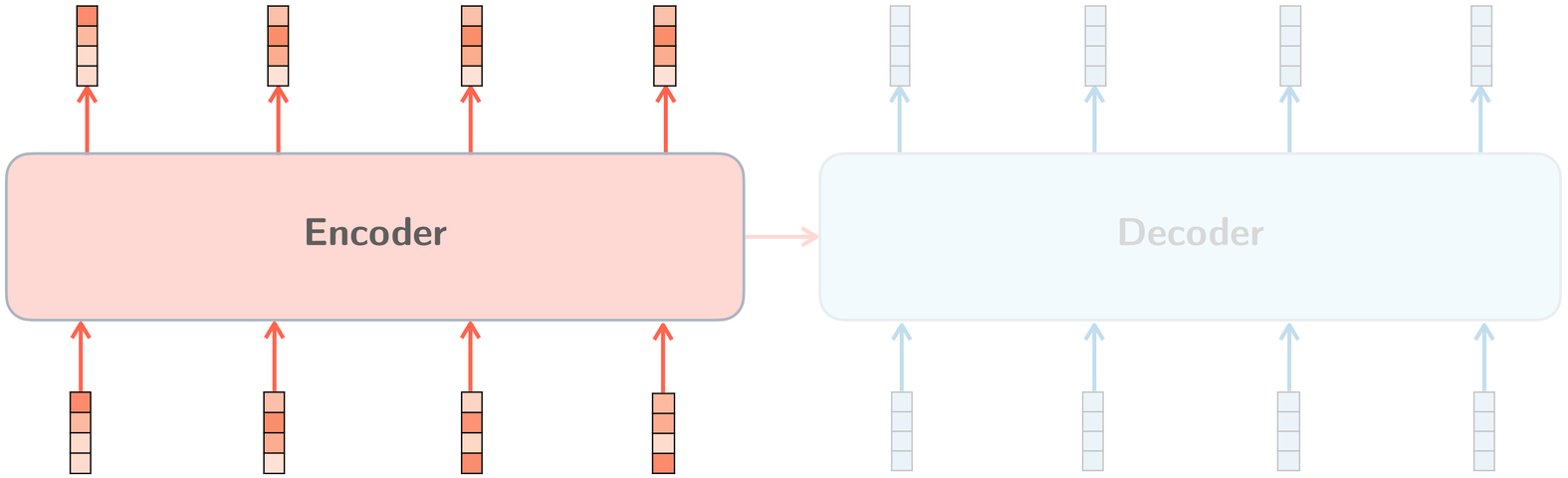
Transformer Encoder–Decoder



1

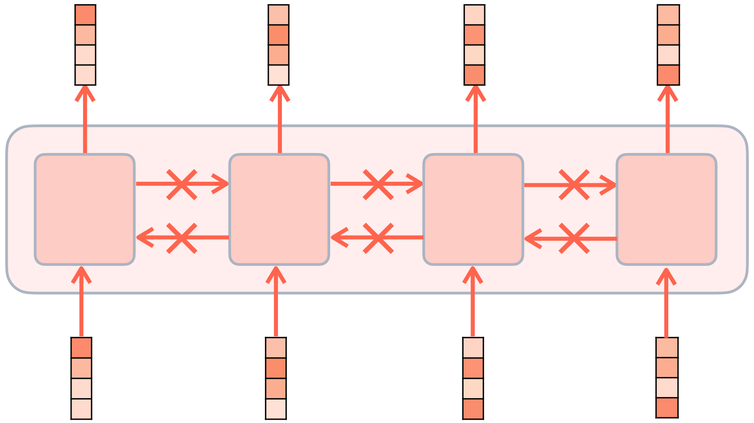
Transformer Encoder

(and BERT)


Encoder
Decoder

Getting rid of recurrence

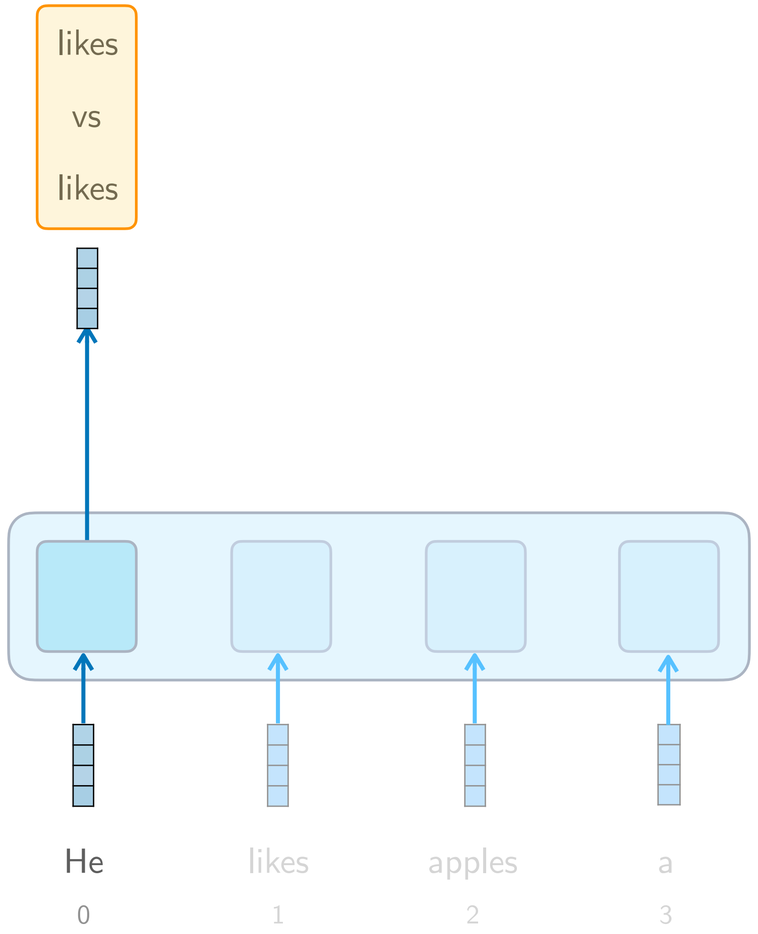
Think of ***N-to-N***



Getting rid of recurrence

Think of ***N-to-N***

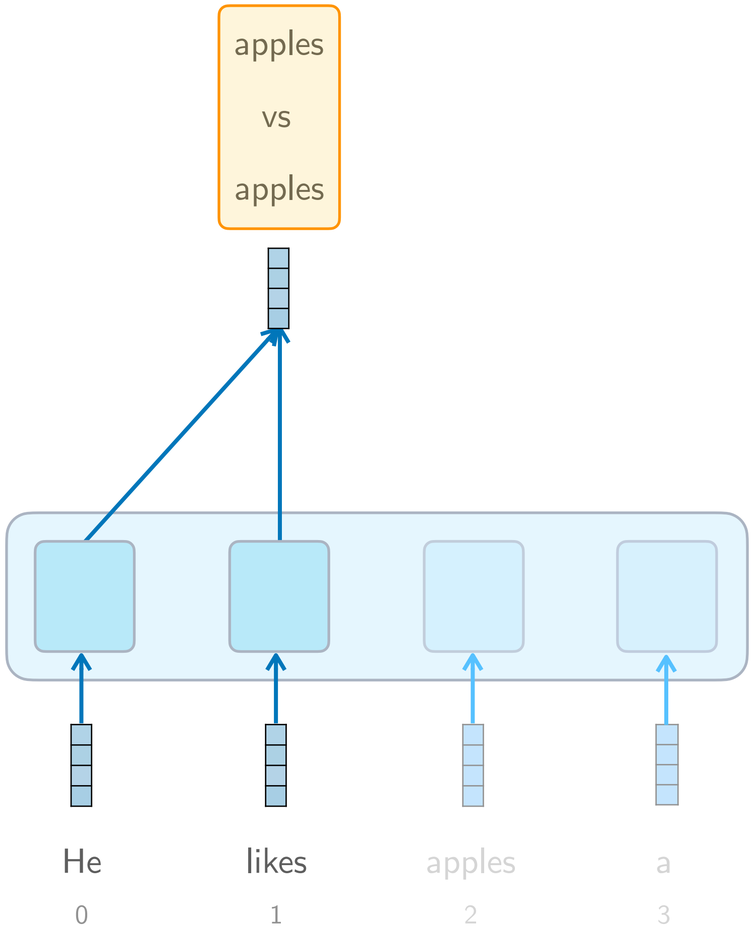
Why get rid of recurrence?



Getting rid of recurrence

Think of ***N-to-N***

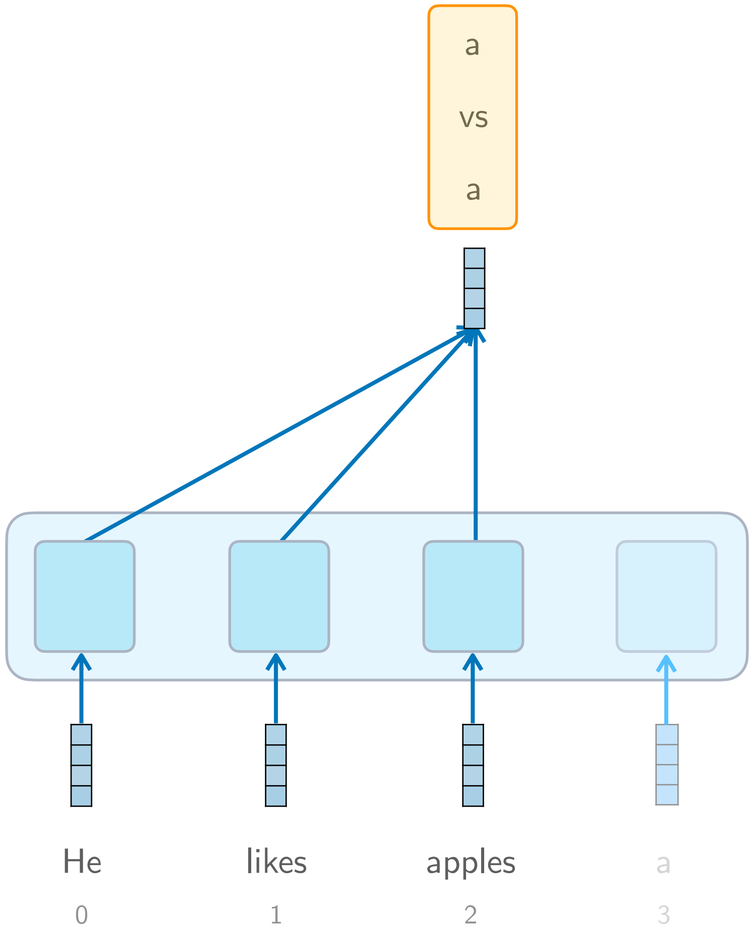
Why get rid of recurrence?



Getting rid of recurrence

Think of ***N-to-N***

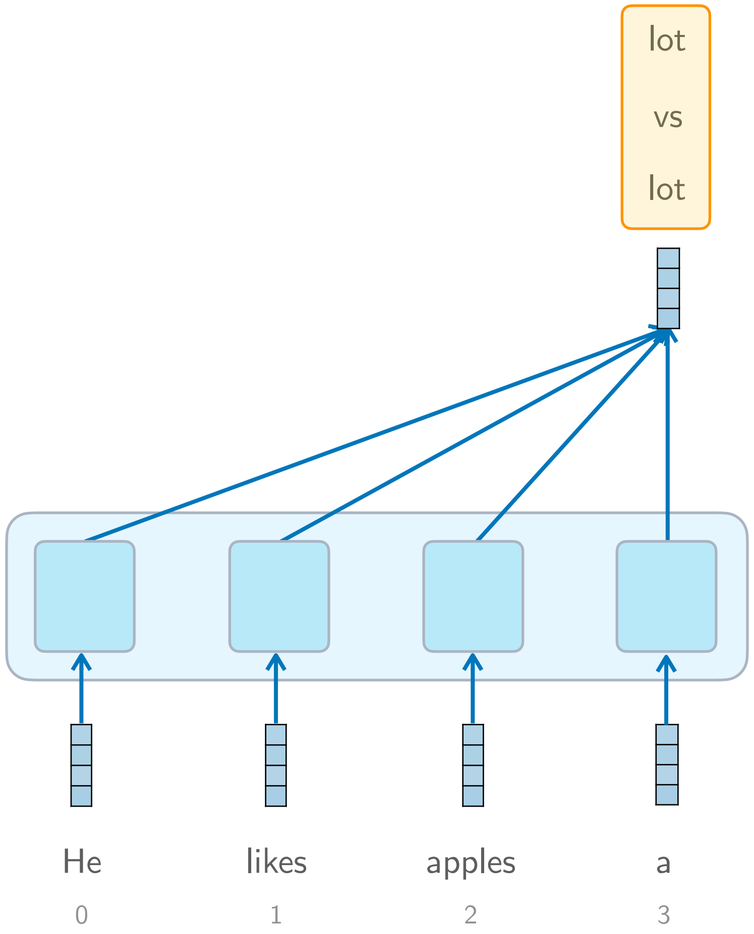
Why get rid of recurrence?



Getting rid of recurrence

Think of ***N-to-N***

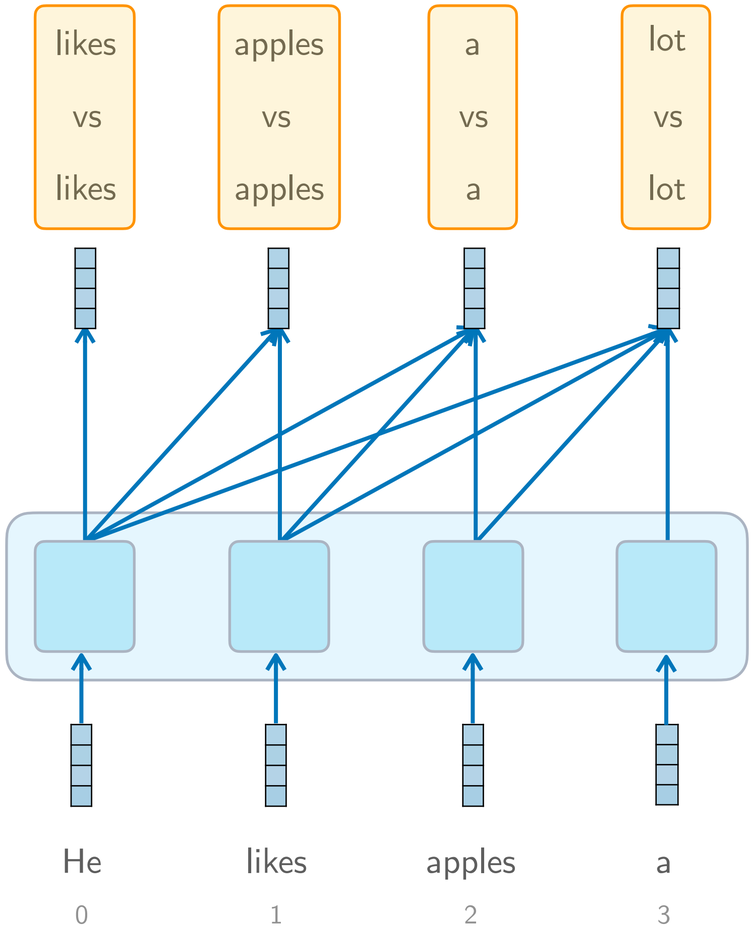
Why get rid of recurrence?



Getting rid of recurrence

Think of ***N-to-N***

Why get rid of recurrence?

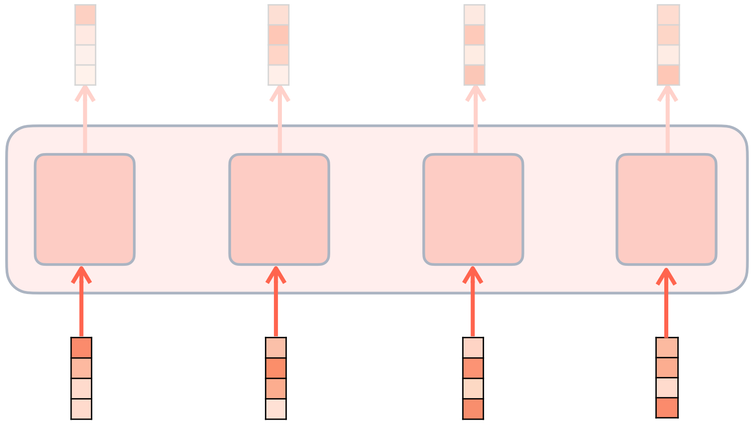


4 training examples at once!
⇒ training parallelism

Self-Attention

Think of ***N-to-N***

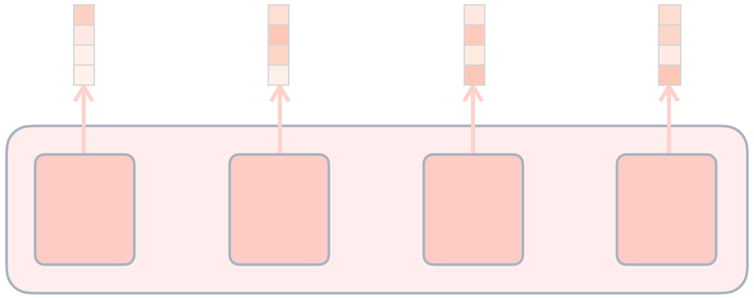
Feed **encoder** information into **encoder**



Self-Attention

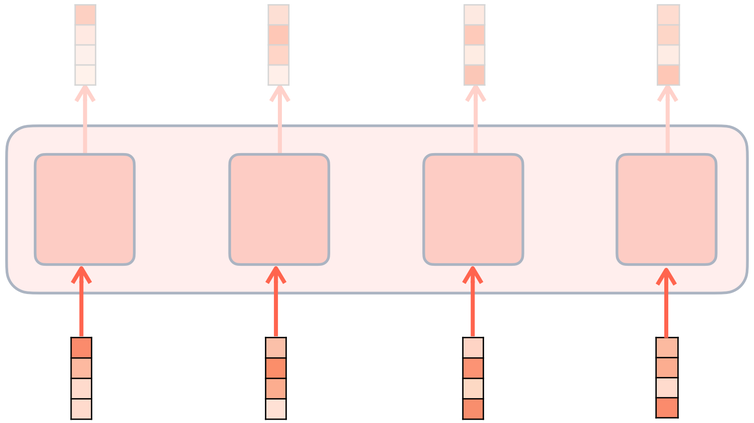
Think of ***N-to-N***

Feed **encoder** information into **encoder**



Each **encoder** token attends to all the other **encoder** tokens

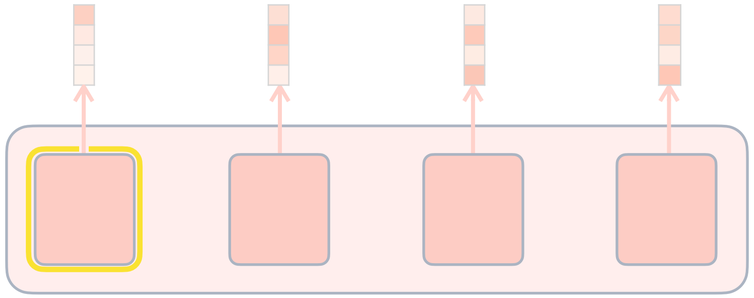
The way that, before, each **encoder** token attended to all the **decoder** tokens



Self-Attention

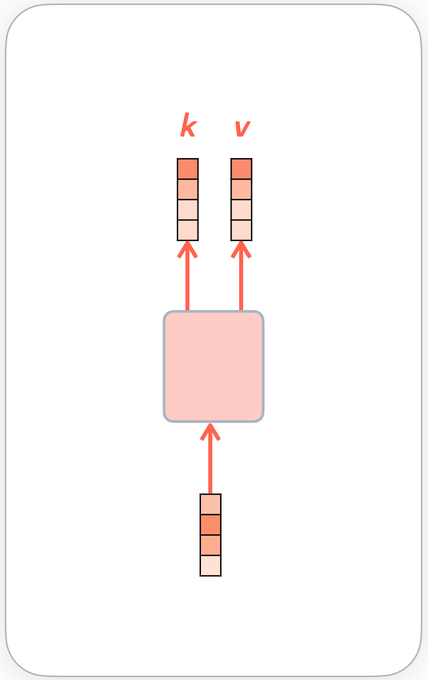
Think of ***N-to-N***

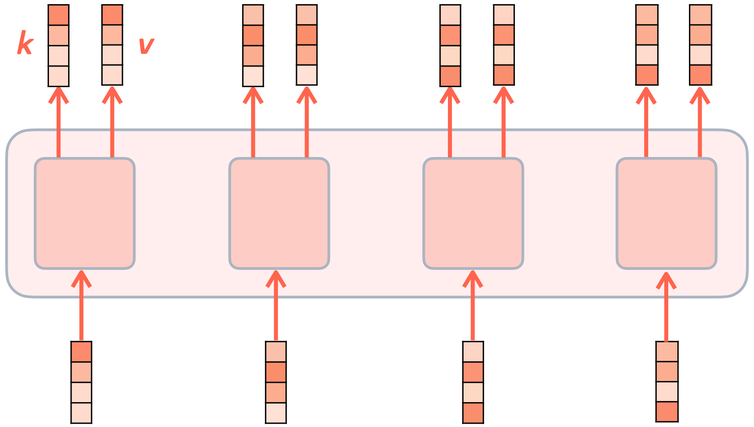
Feed **encoder** information into **encoder**



Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens





key and value vectors

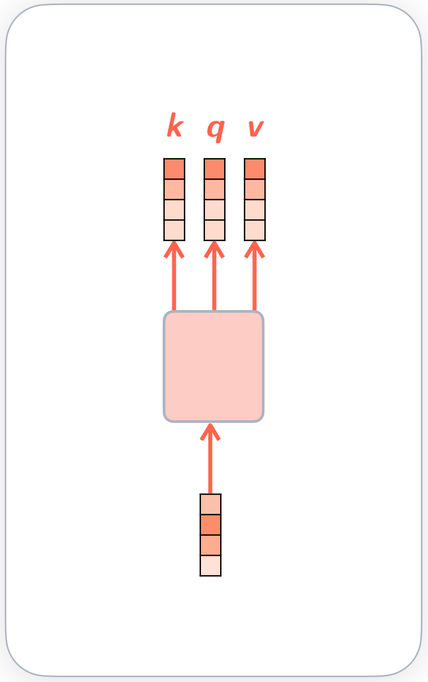
Self-Attention

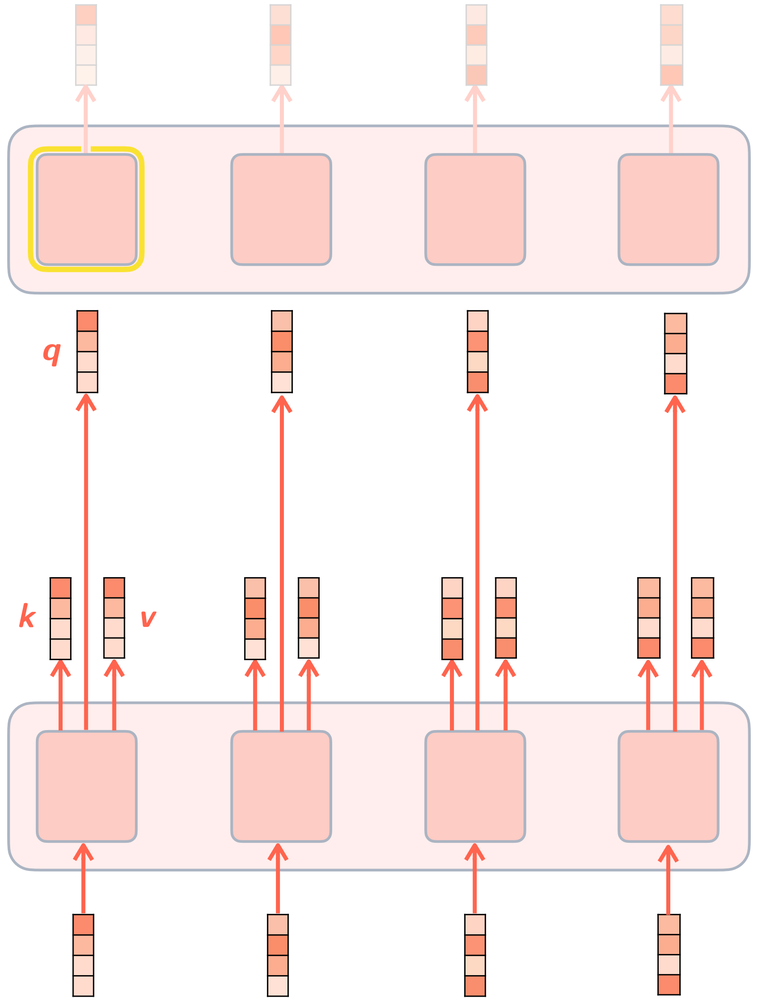
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**





Each **encoder** token attends to all the other **encoder** tokens

e.g. pronoun "it" might ask "What am I referring to?"
verb "runs" might ask "Who is running?"

and then incorporate that in its own representation

key and value vectors

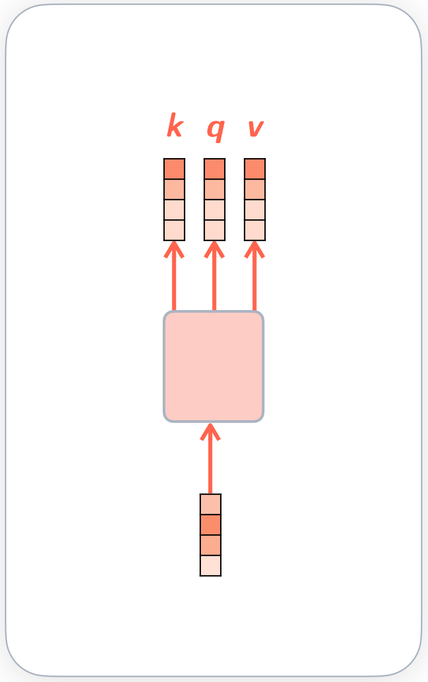
Self-Attention

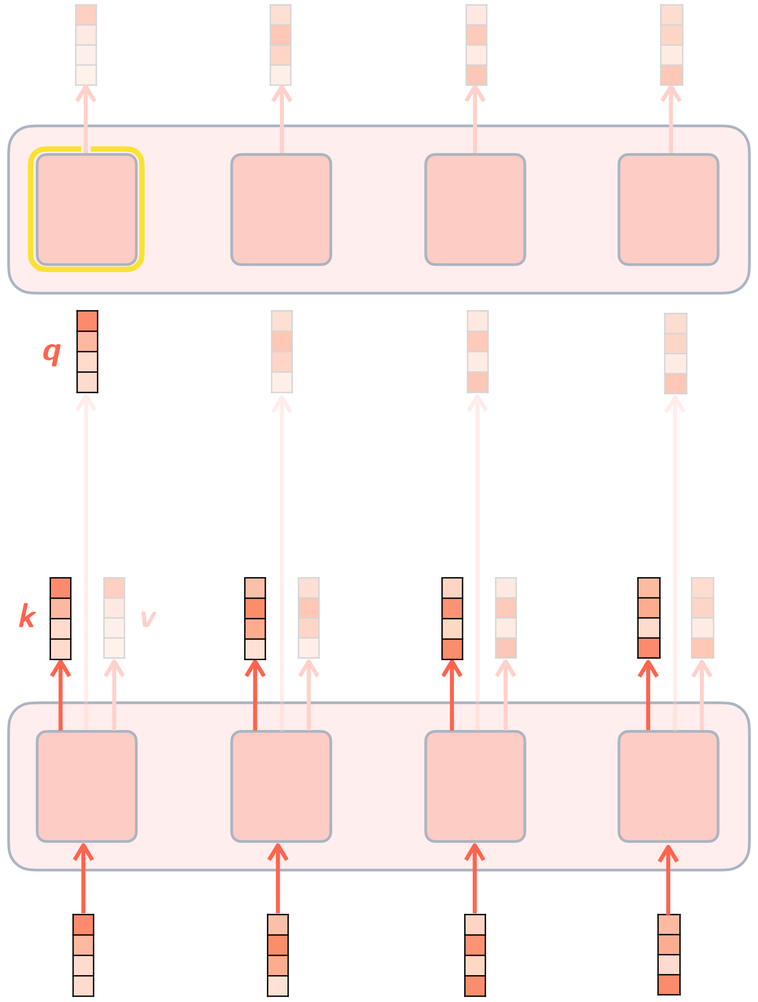
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**





Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors

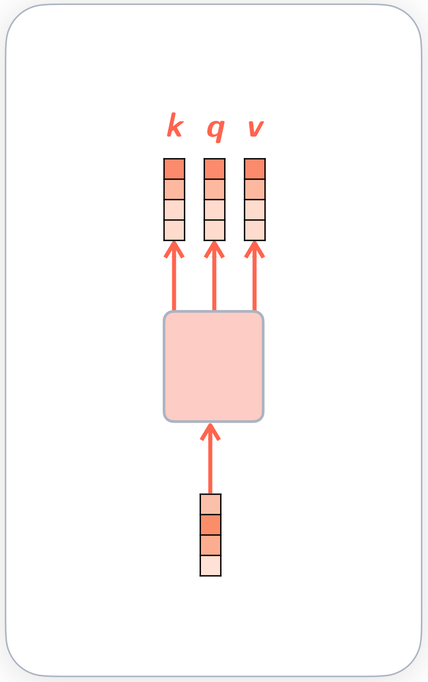
Self-Attention

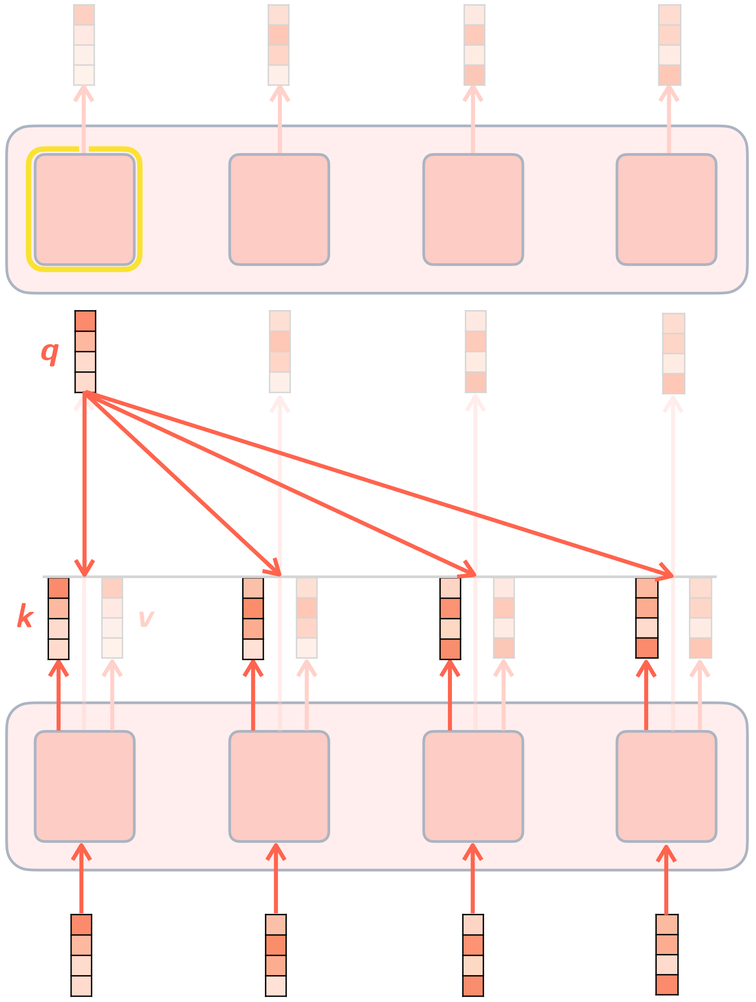
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

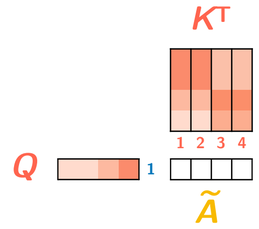




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



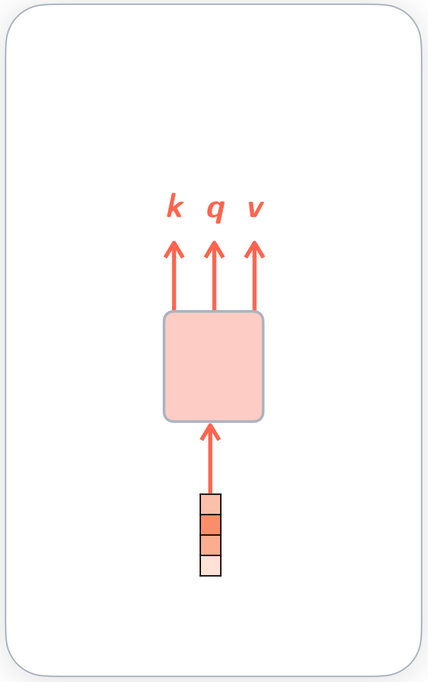
Self-Attention

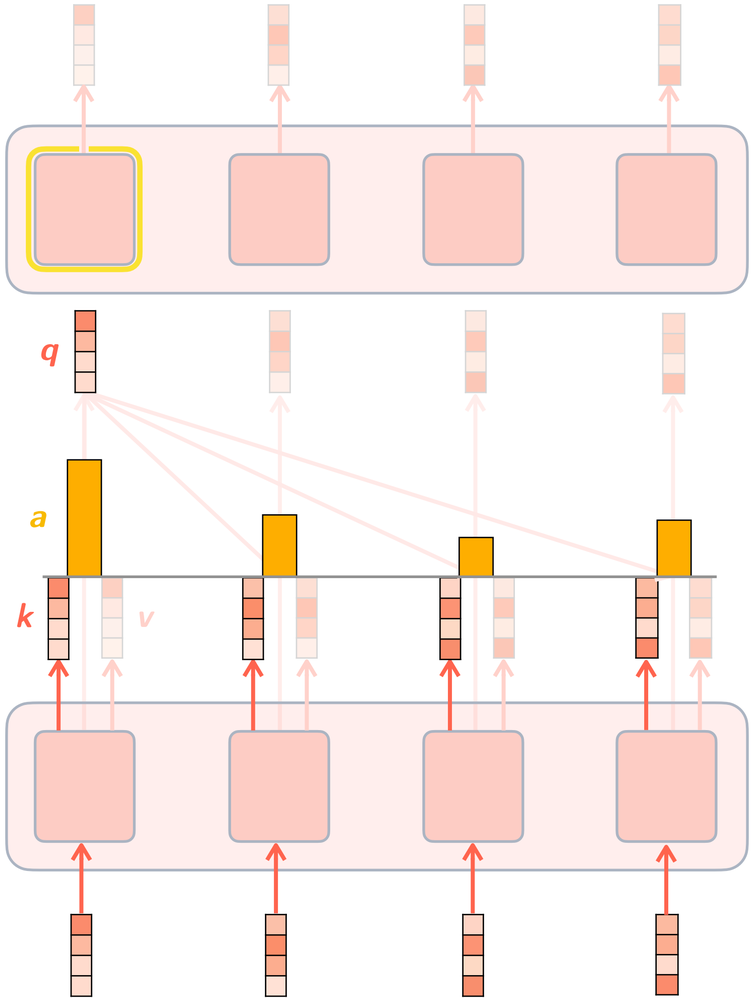
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

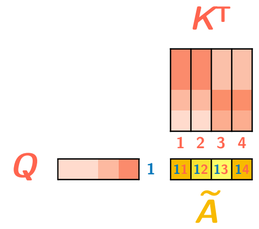




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



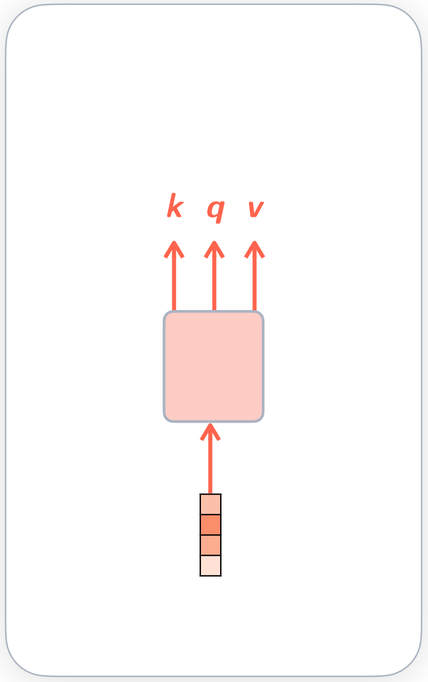
Self-Attention

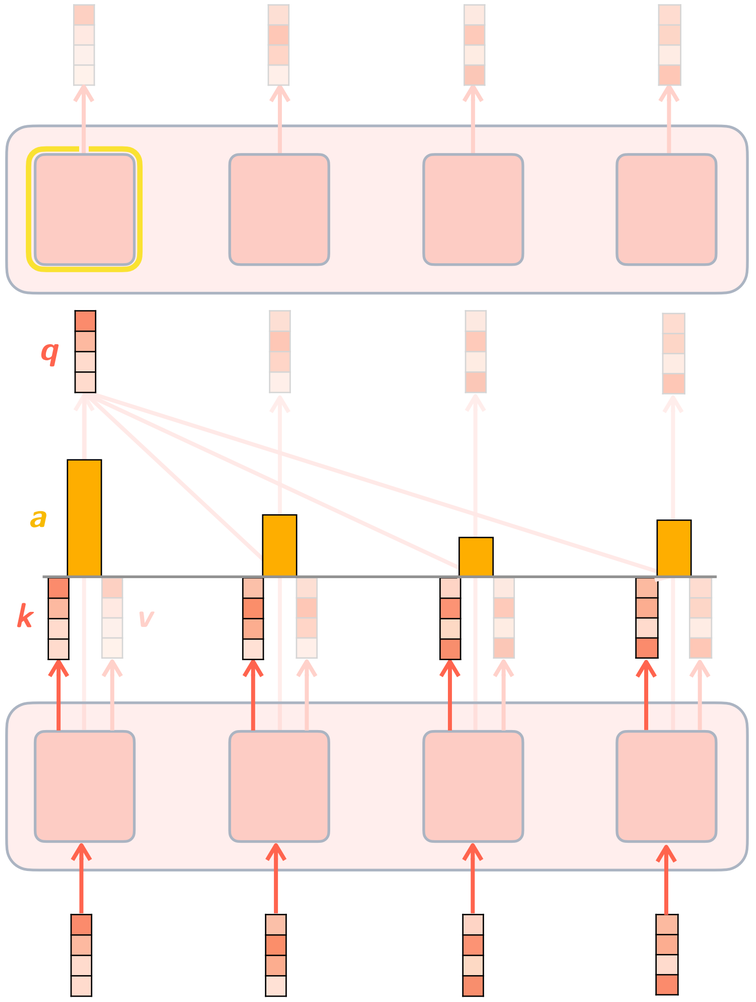
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

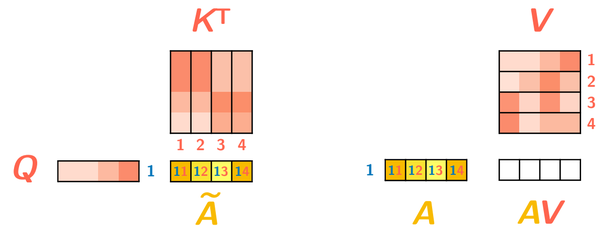




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



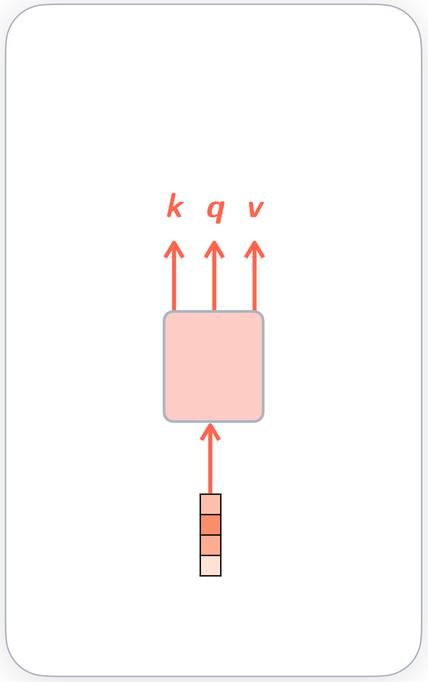
Self-Attention

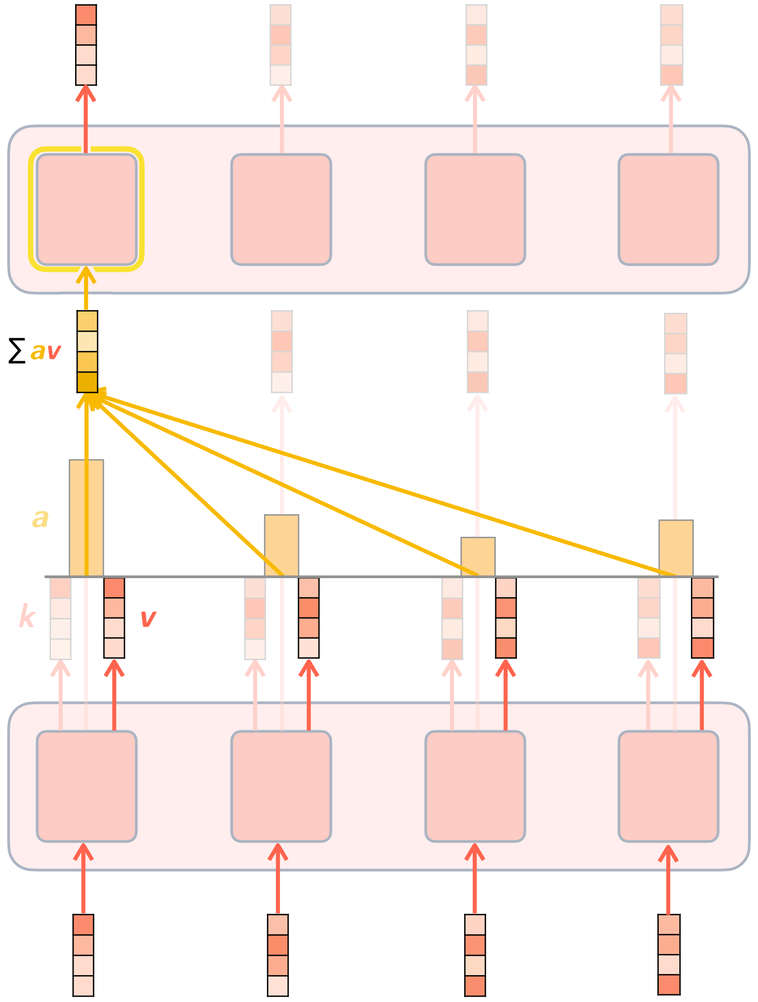
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

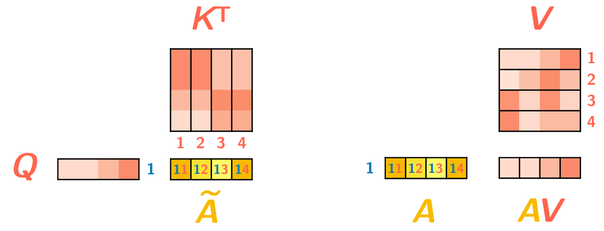




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



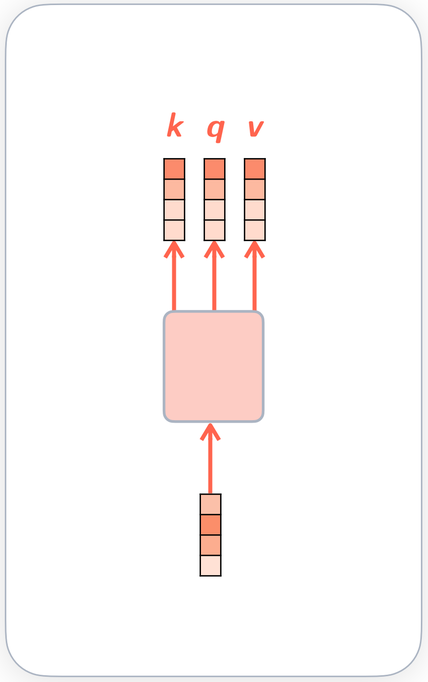
Self-Attention

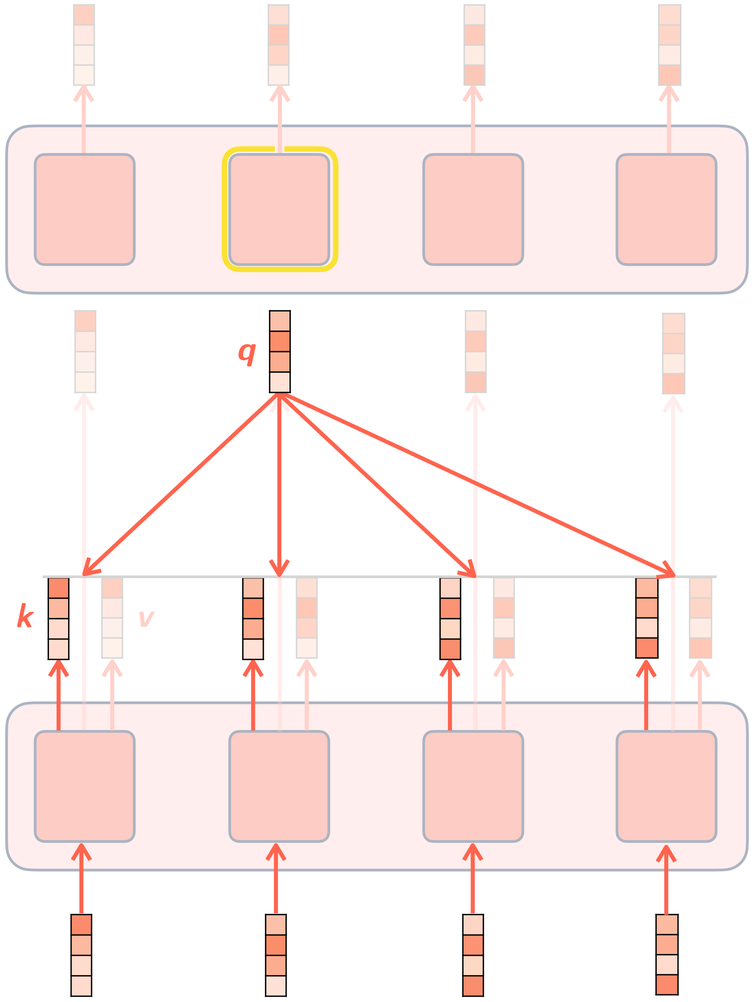
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

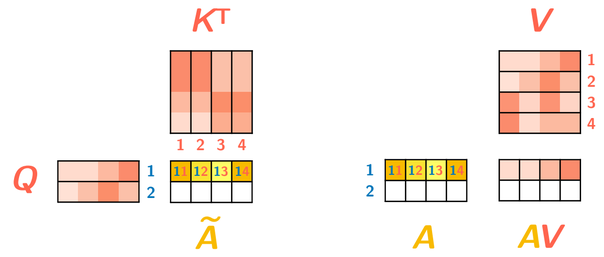




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



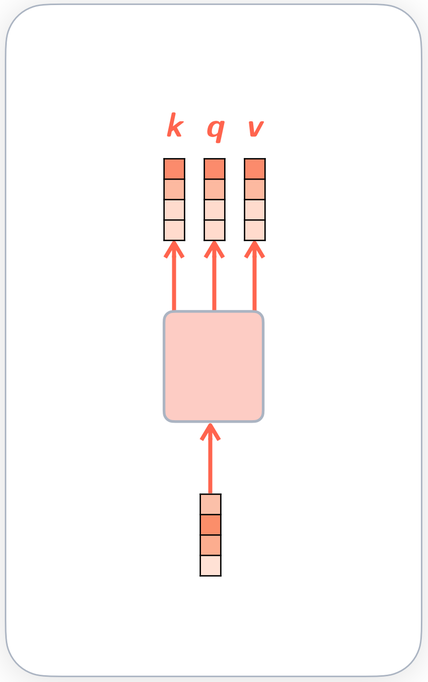
Self-Attention

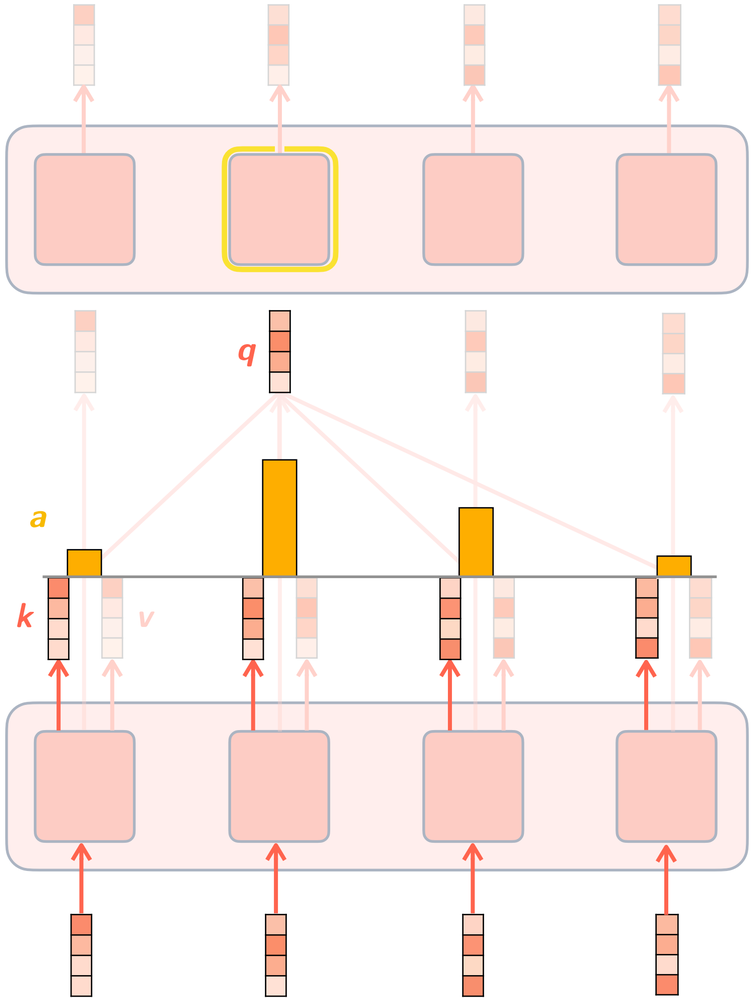
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

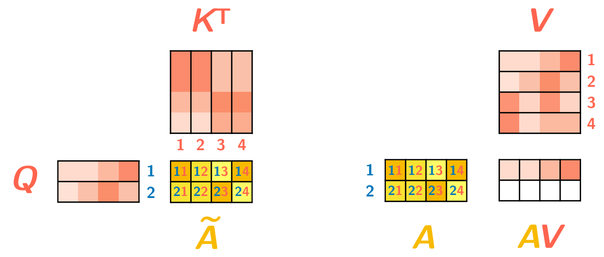




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



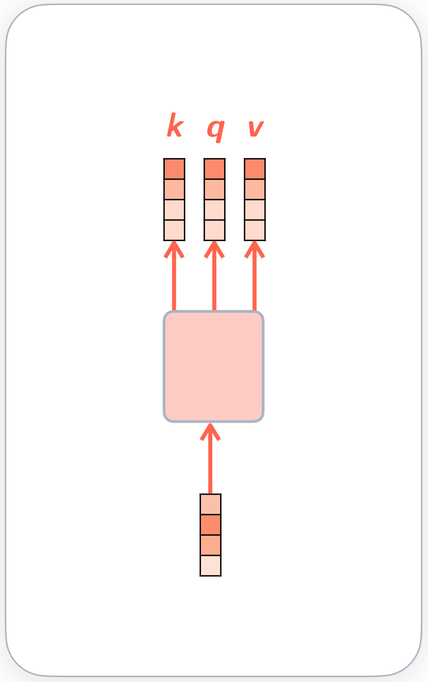
Self-Attention

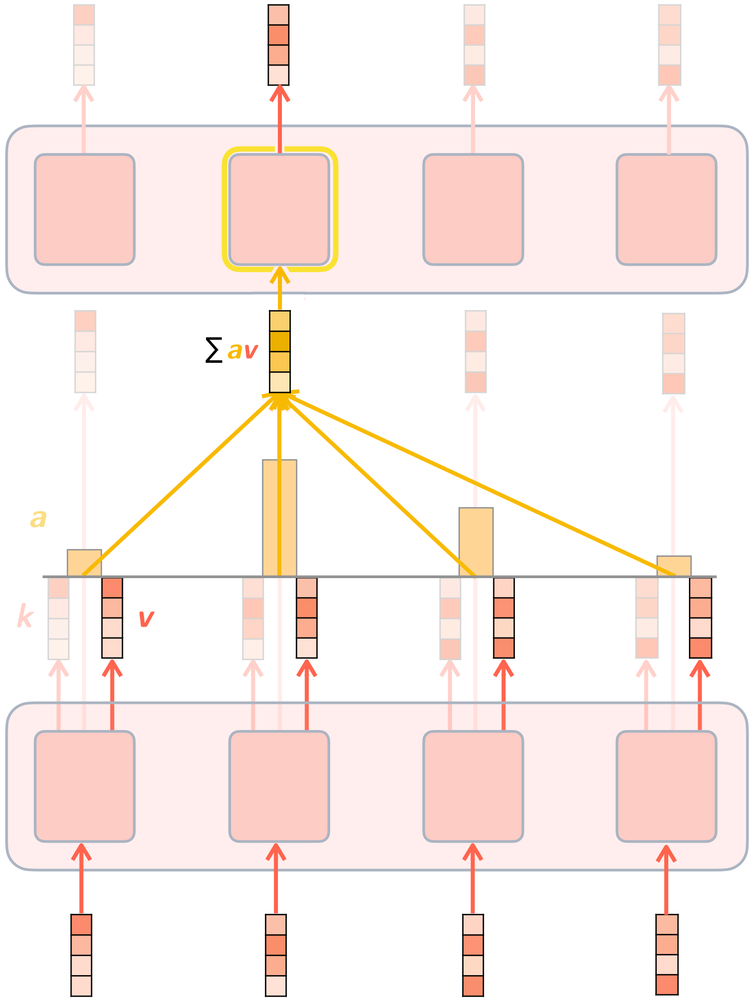
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

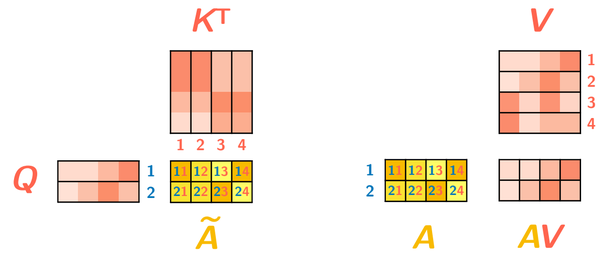




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



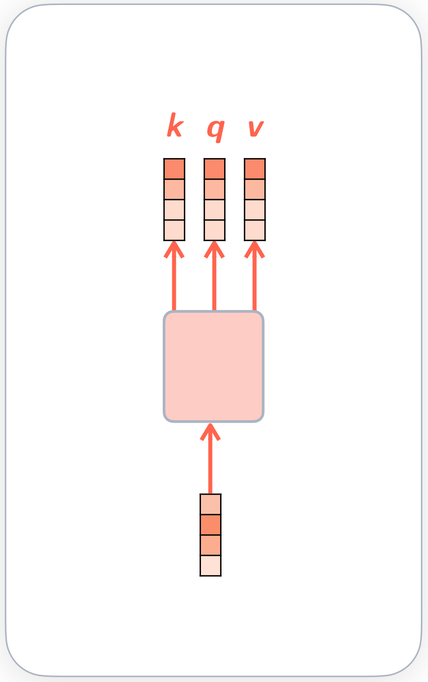
Self-Attention

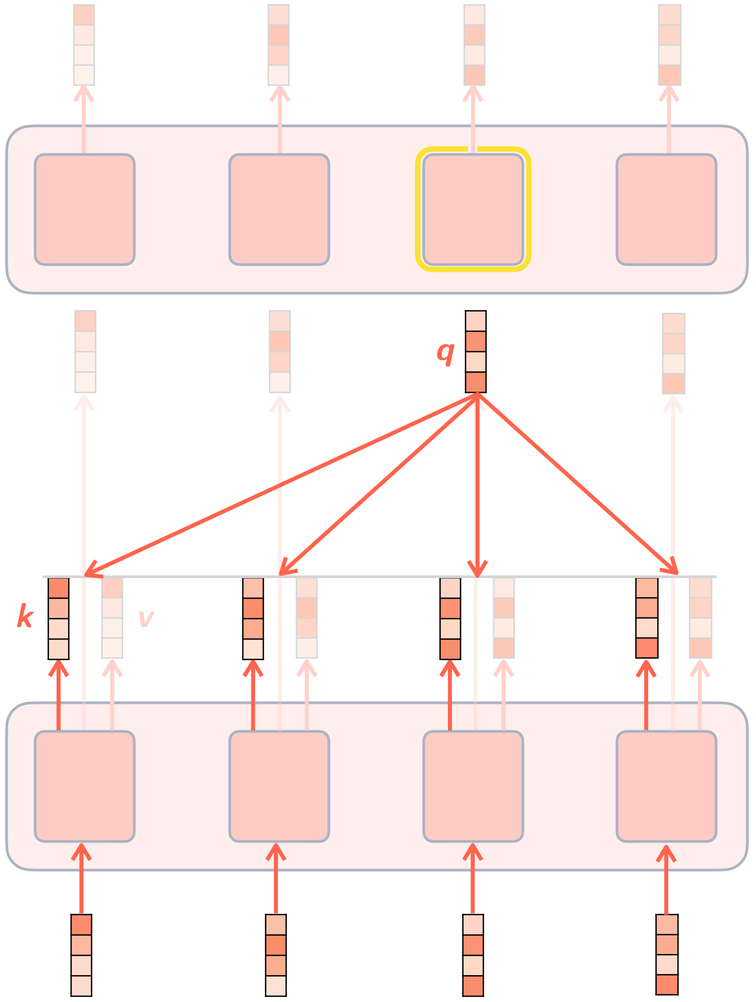
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

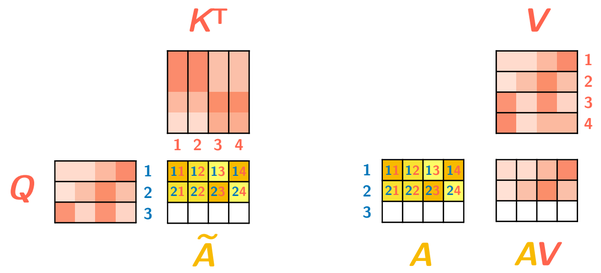




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



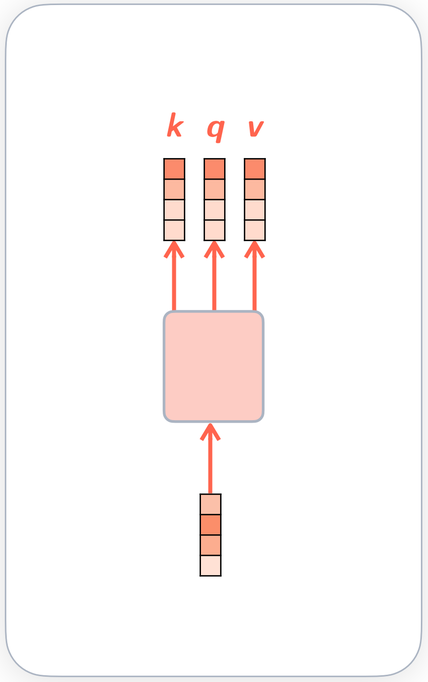
Self-Attention

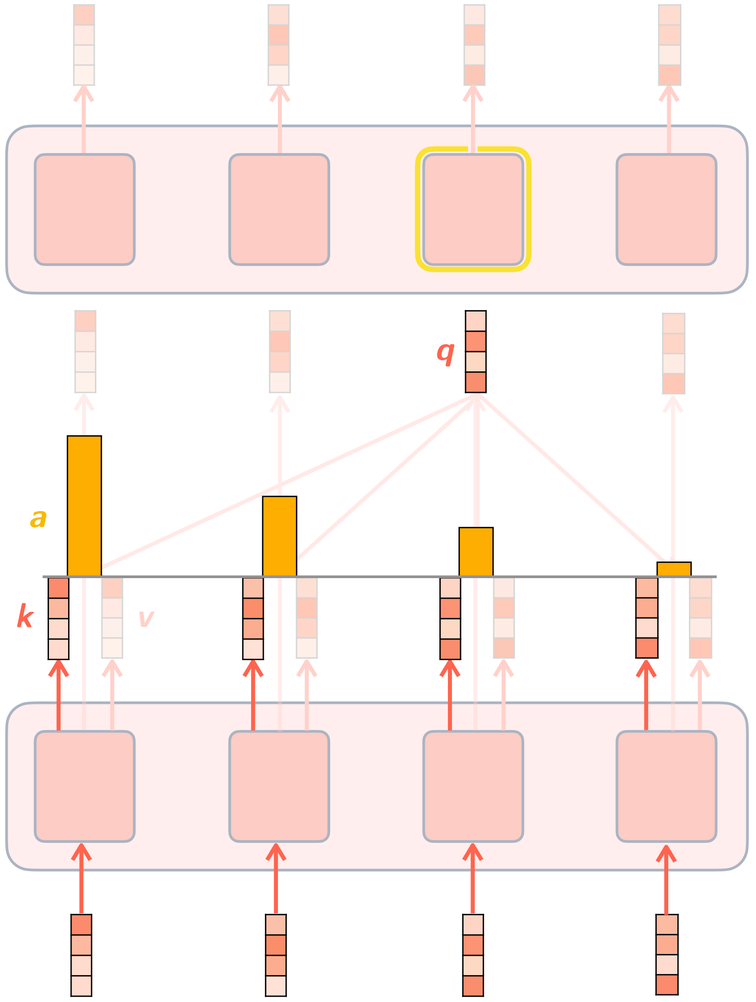
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

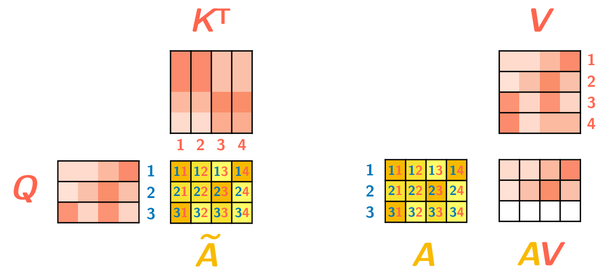




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



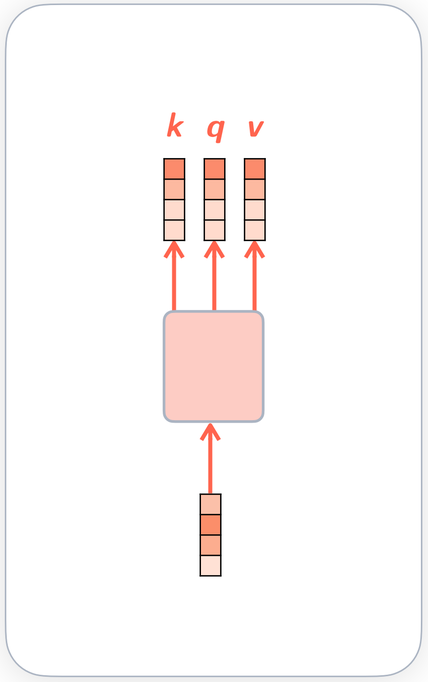
Self-Attention

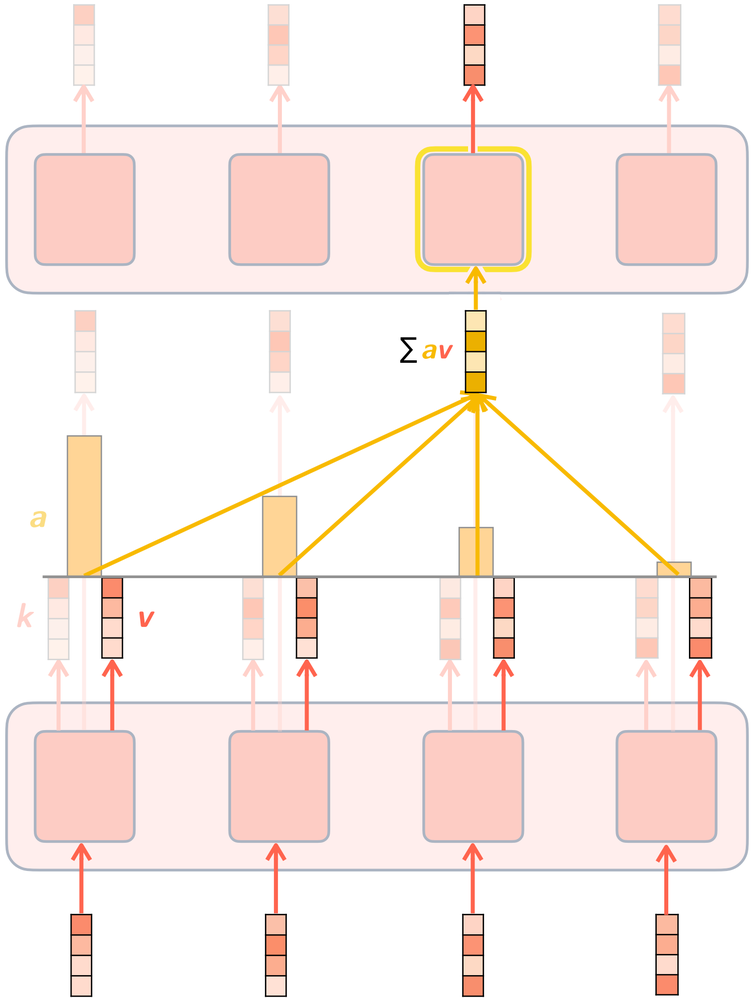
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

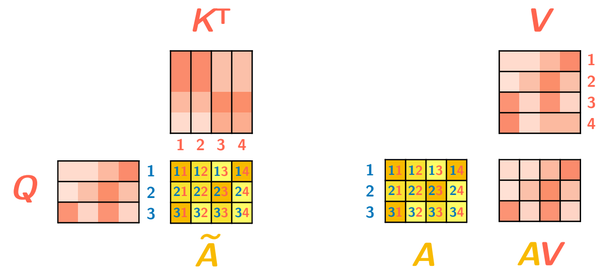




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



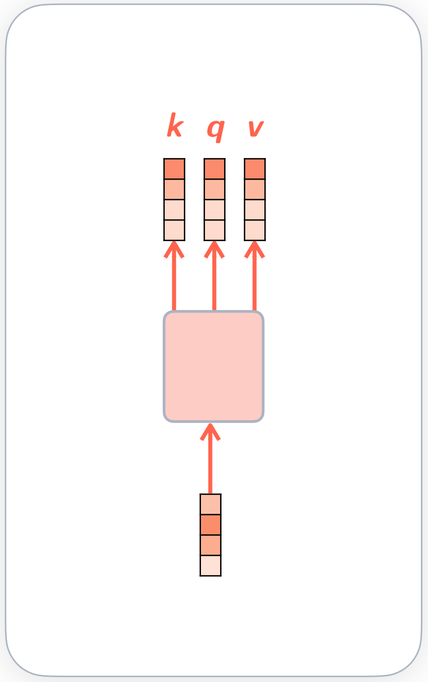
Self-Attention

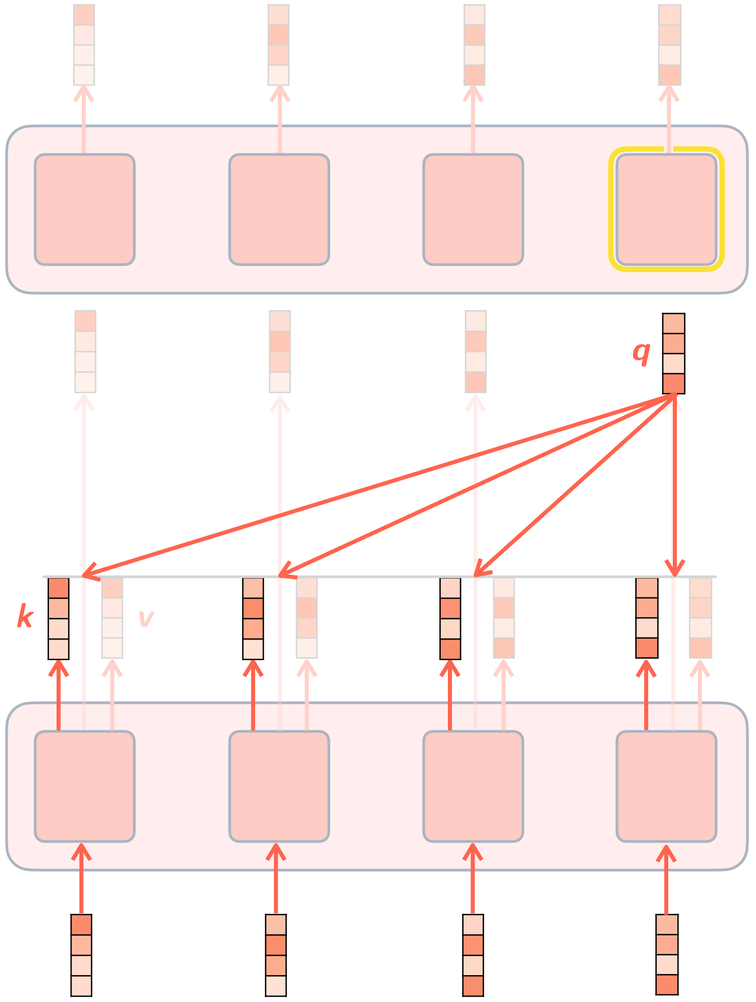
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

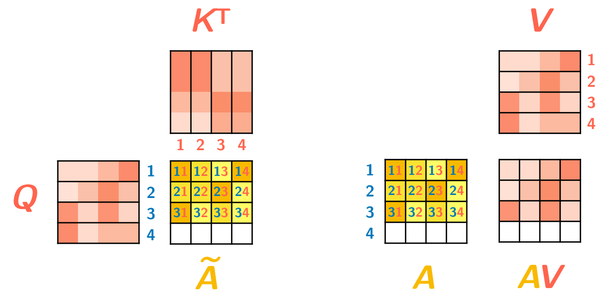




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



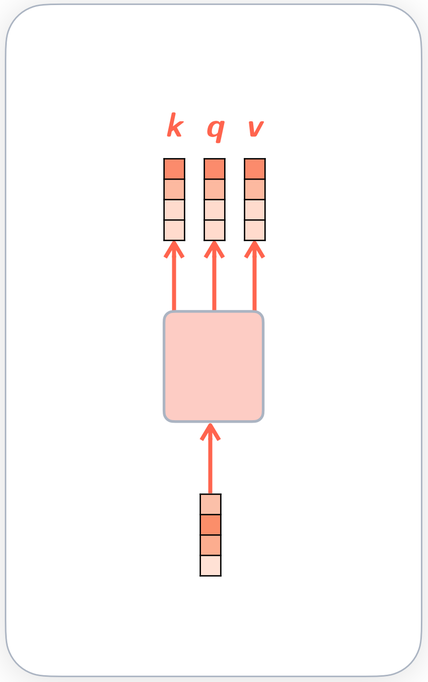
Self-Attention

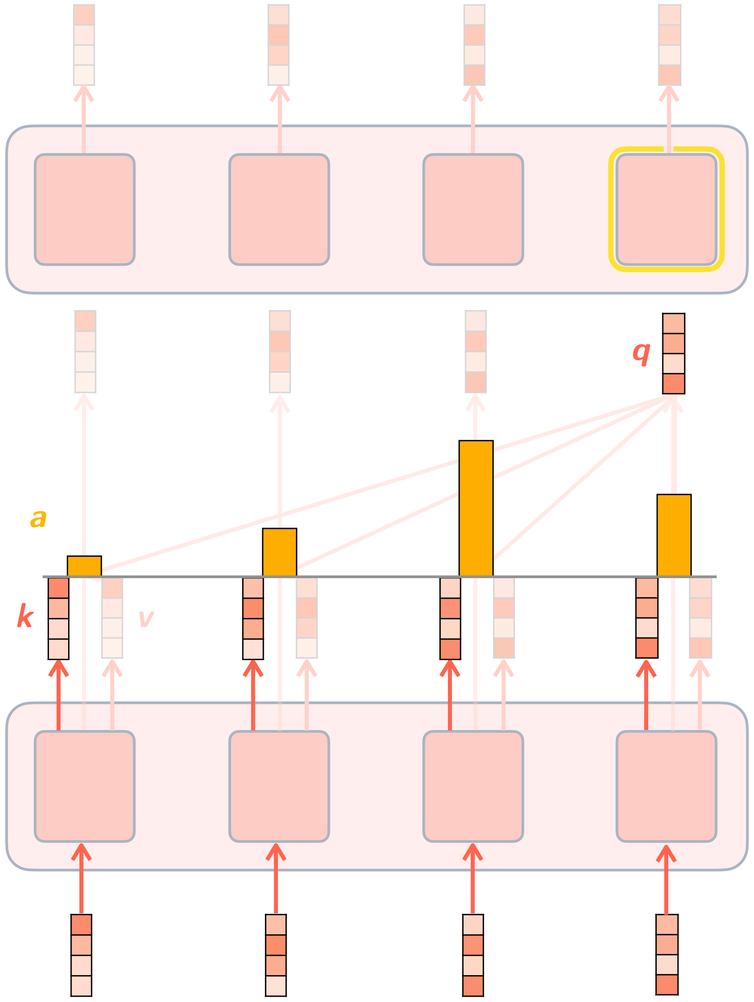
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

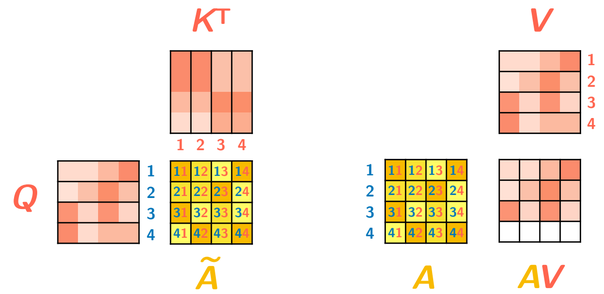




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



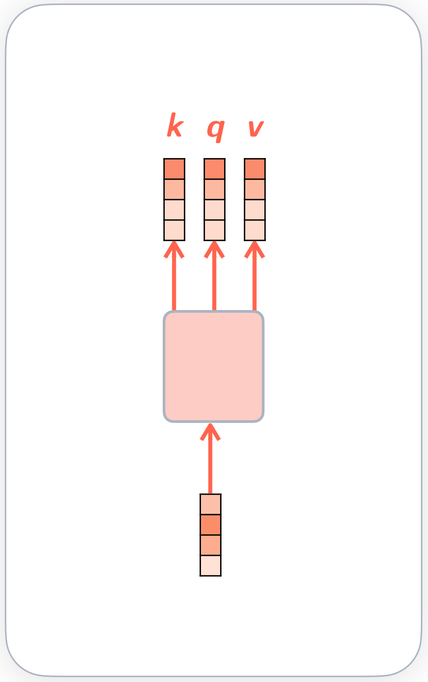
Self-Attention

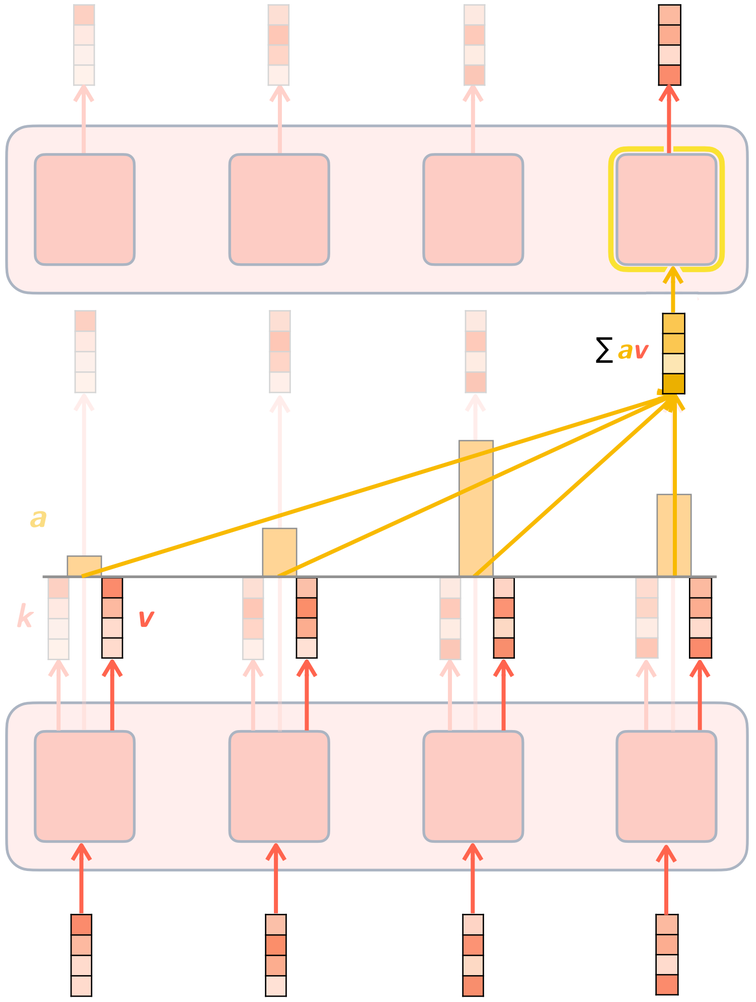
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

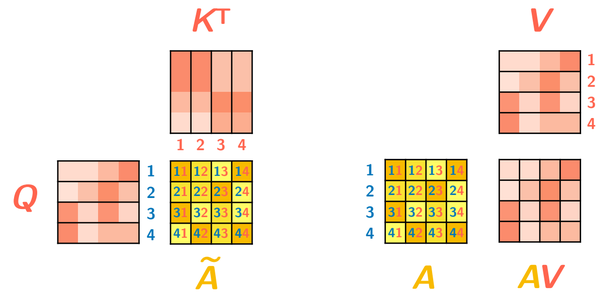




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



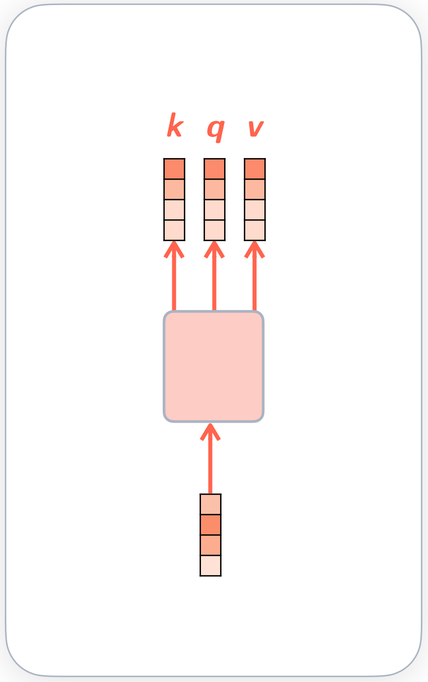
Self-Attention

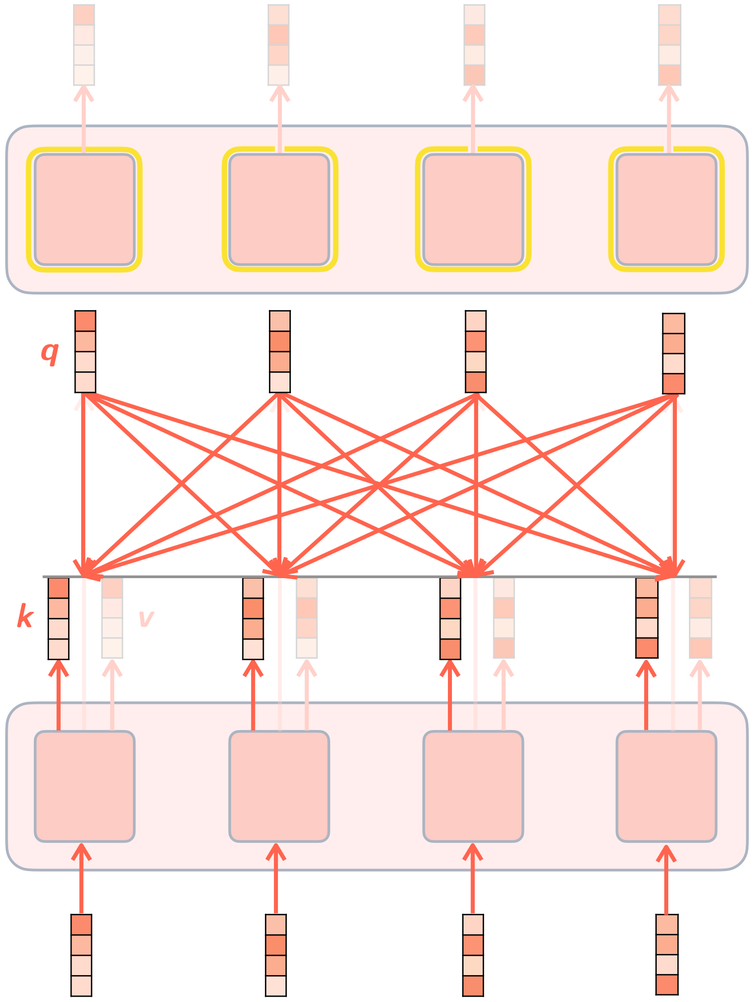
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**

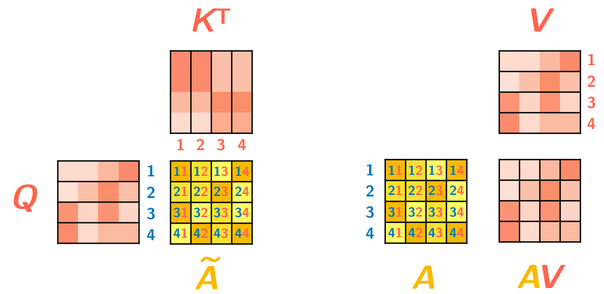




Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors



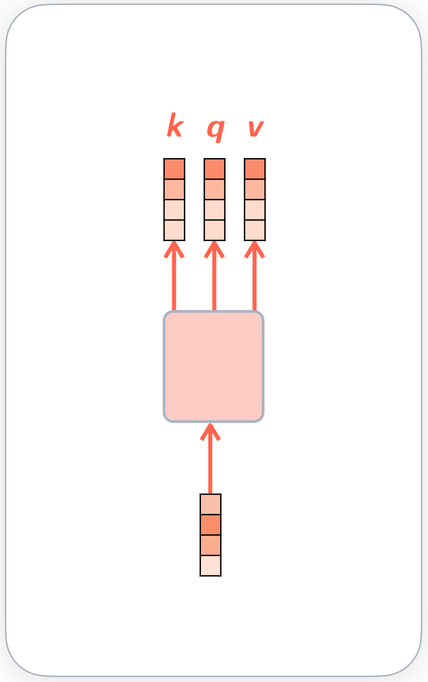
Self-Attention

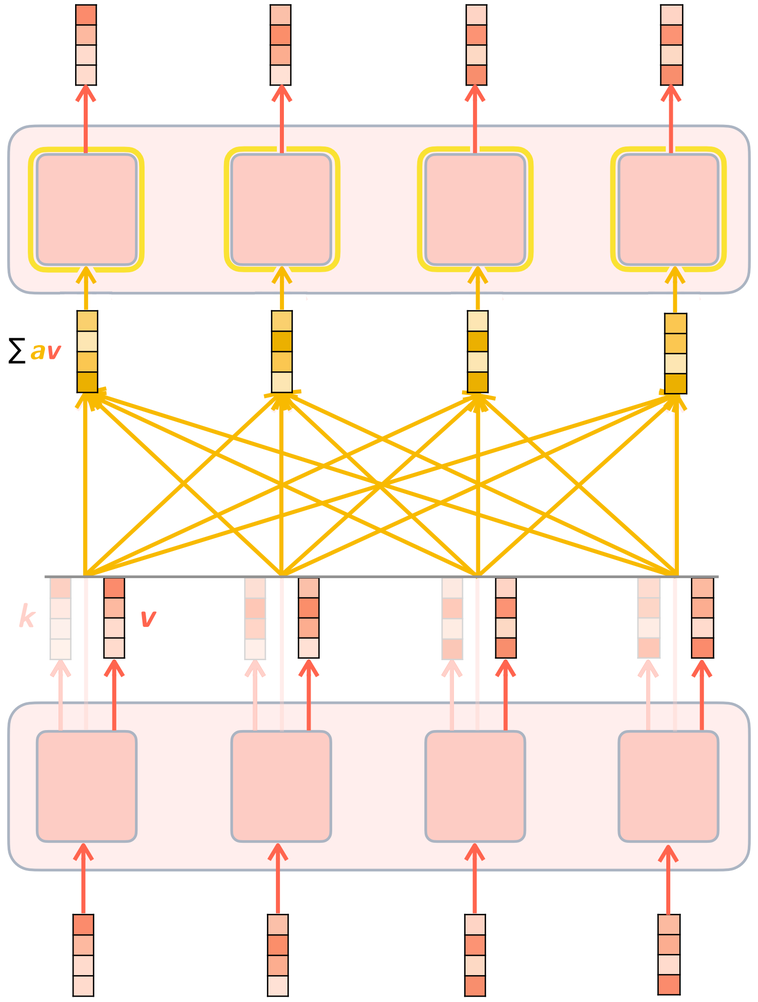
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**





self-attention network instead of **recurrent network**

Each **encoder** token attends to all the other **encoder** tokens

The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors

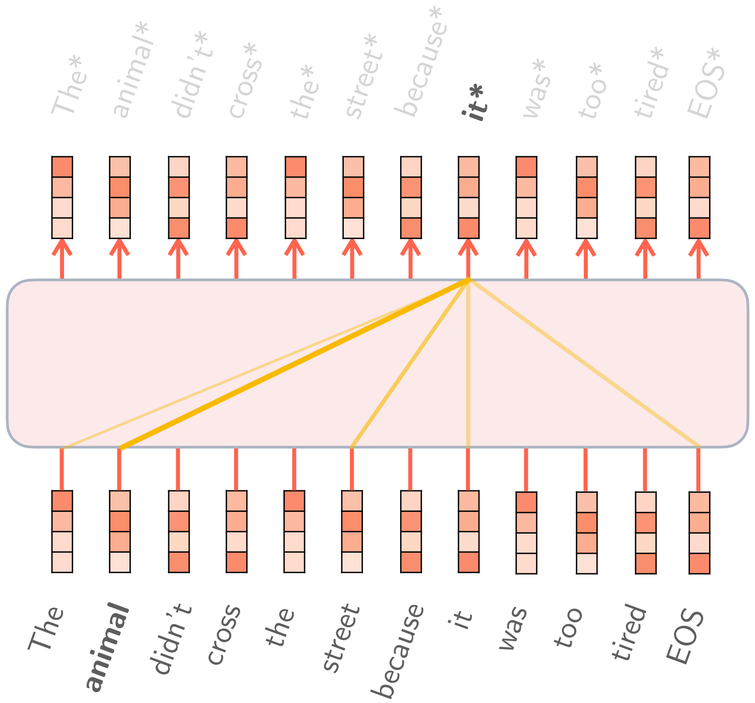
Notice that since the attention is computed in the forward pass, it technically works for any context size. I.e. in theory, transformers can take as input an arbitrary number of tokens.

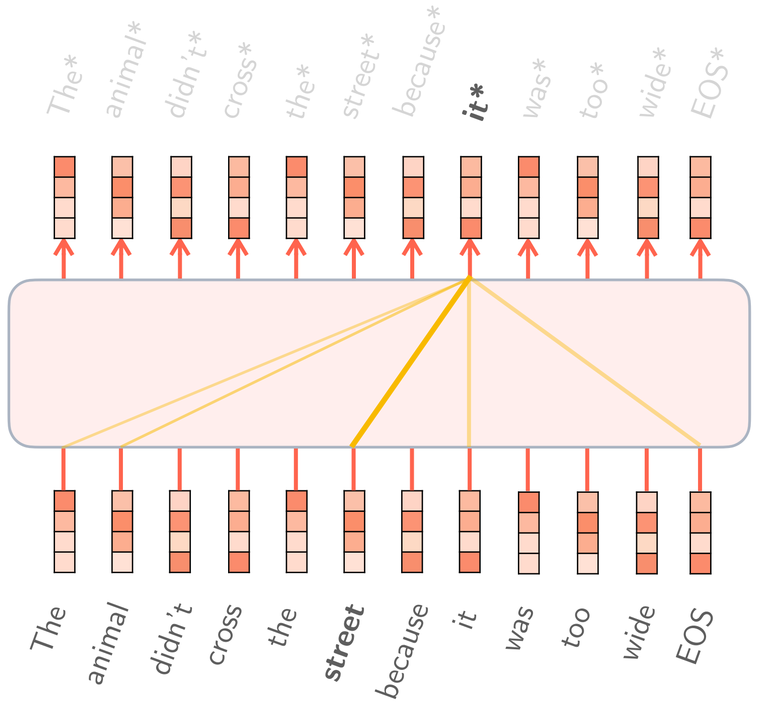
Self-Attention

Think of ***N-to-N***

Feed **encoder** information into **encoder**

Contextualized representation, like in bidirectional RNN





https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

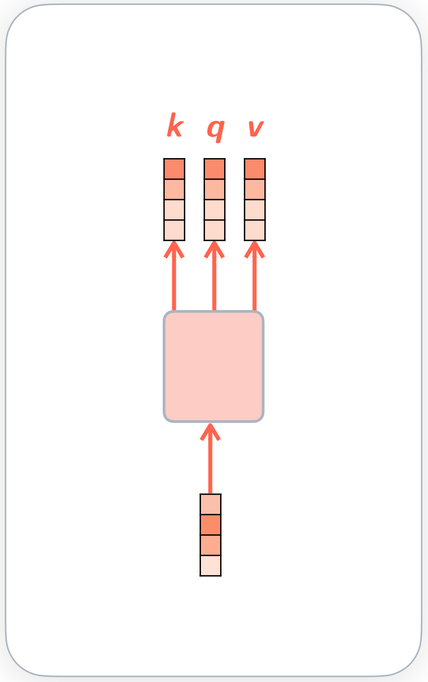
Self-Attention

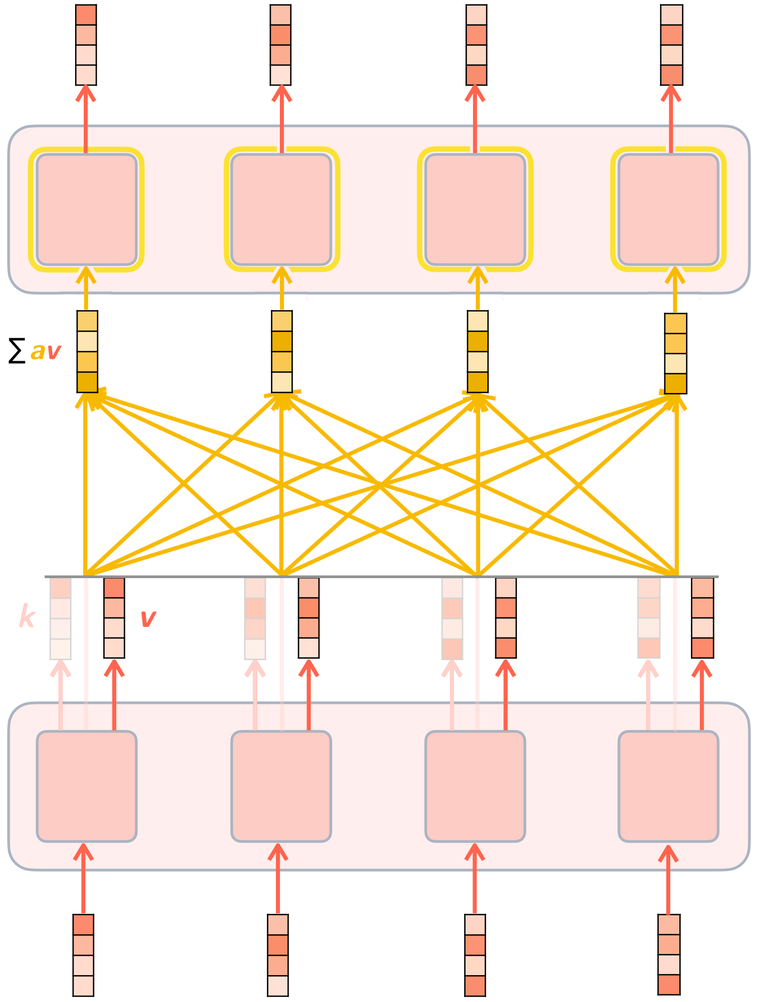
Think of ***N-to-N***

Feed **encoder** information into **encoder**

come up with a **query**
for *this* **encoder** time step

search for this **query**
in the same **encoder sequence**,
by comparing it to each **key**





Each **encoder** token attends to all the other **encoder** tokens

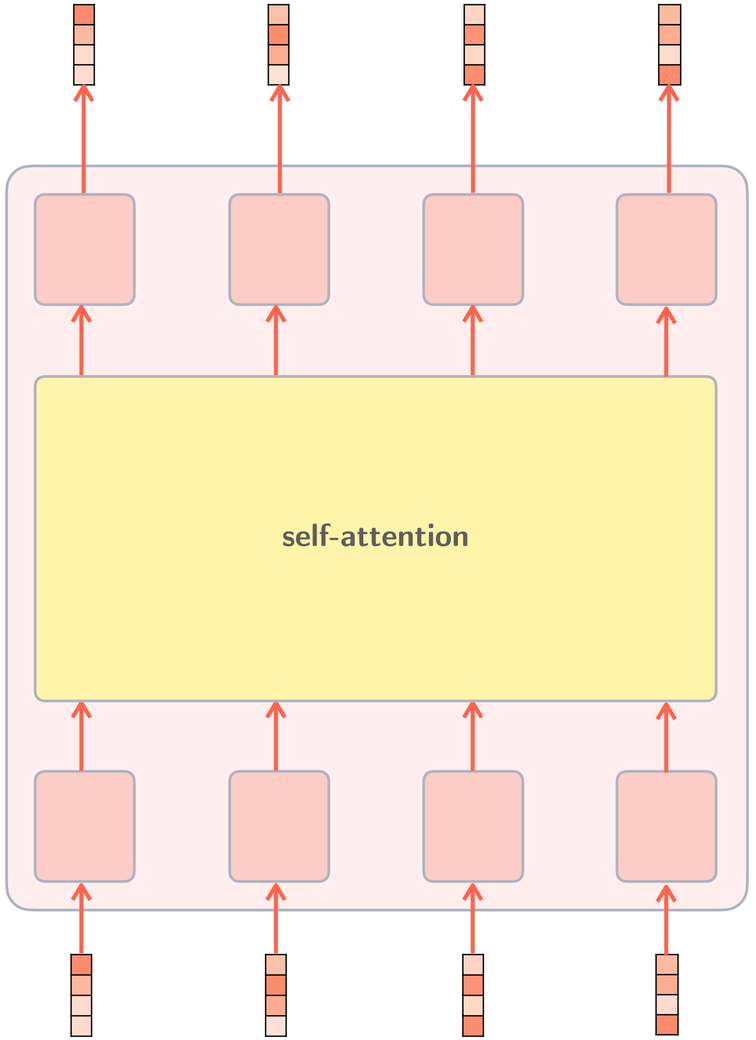
The way that, before, each **encoder** token attended to all the **decoder** tokens

key and value vectors

Notice that since the attention is computed in the forward pass, it technically works for any context size. I.e. in theory, transformers can take as input an arbitrary number of tokens.

Self-Attention

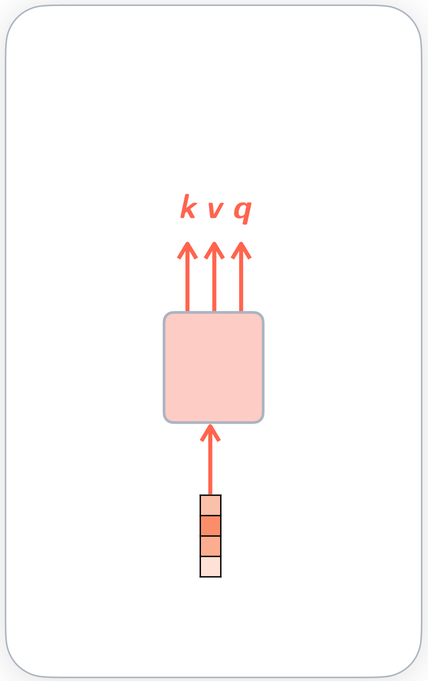
Feed **encoder** information into **encoder**



refined **encoder sequence *X'***

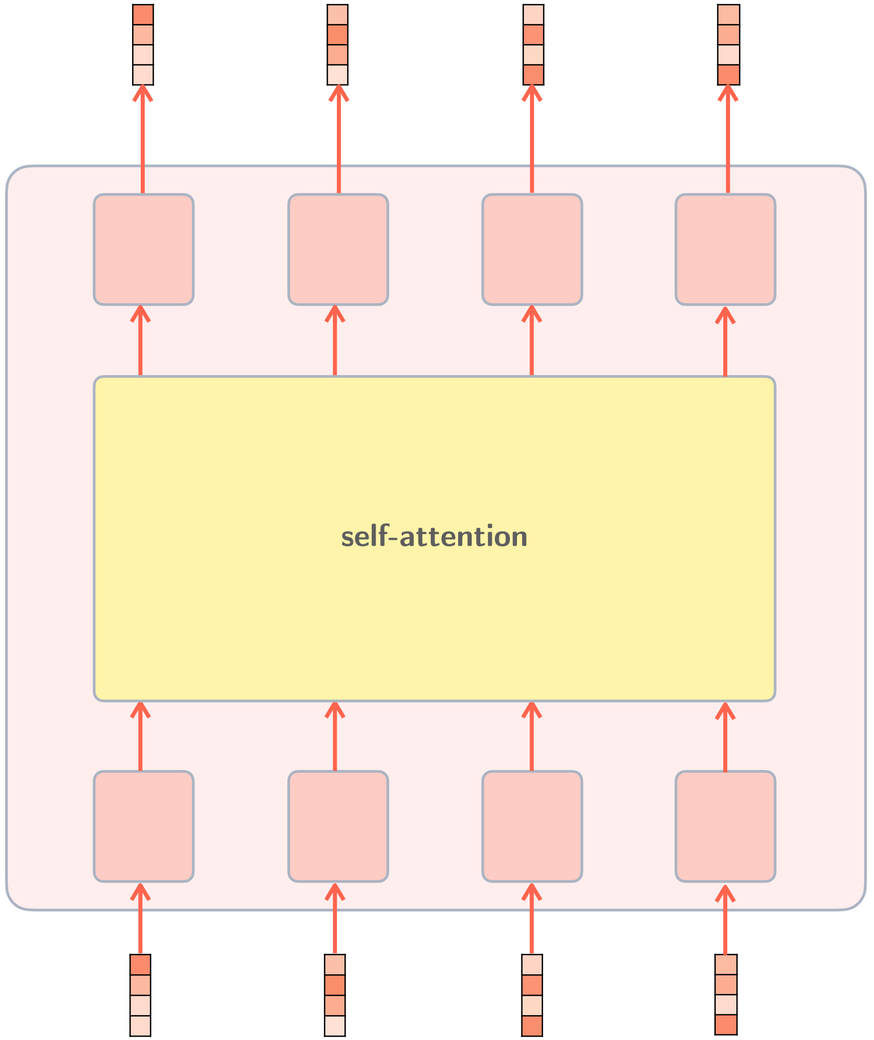
refines and ***contextualizes*** **encoder sequence**

encoder sequence *X*



Self-Attention

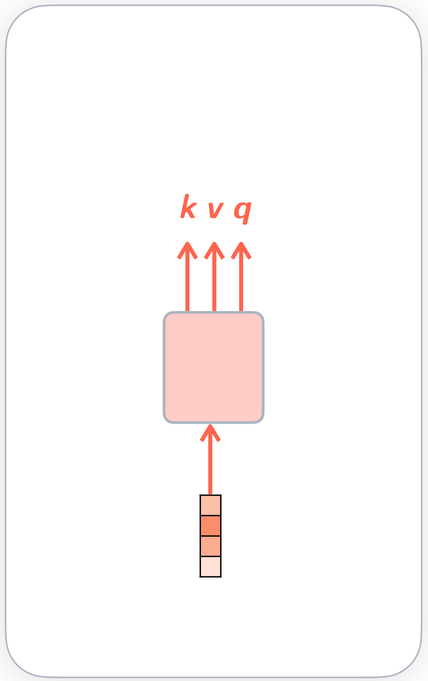
Feed **encoder** information into **encoder**



refined **encoder sequence *X'***

refines and ***contextualizes*** **encoder sequence**

encoder sequence *X*

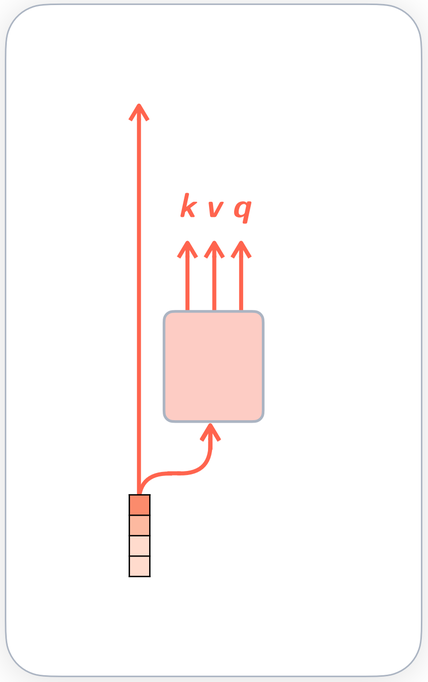


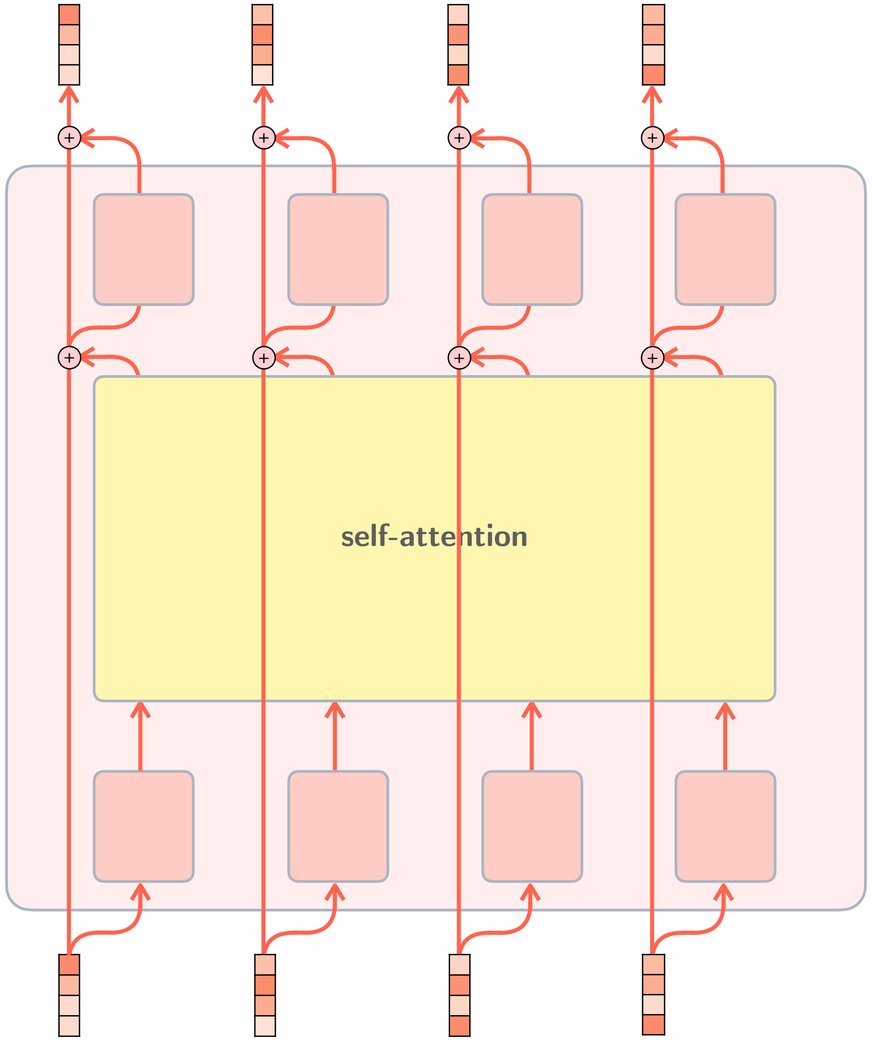
Self-Attention

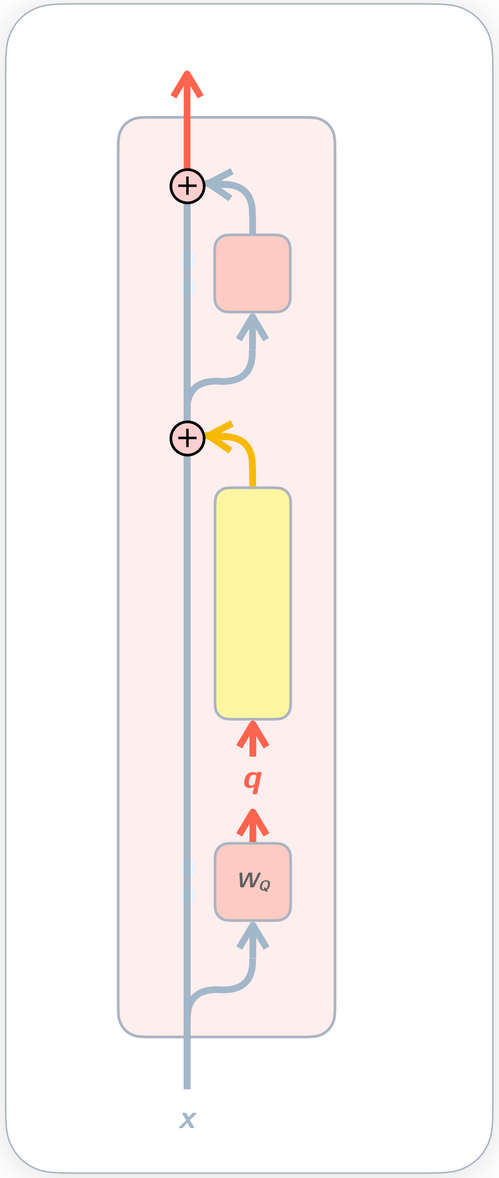
The ***residual stream***

Each token has its own ***residual stream***

Attention reads and writes information to the residual stream



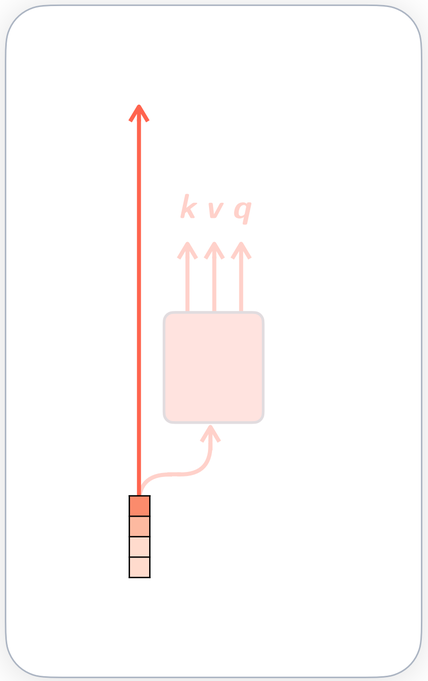


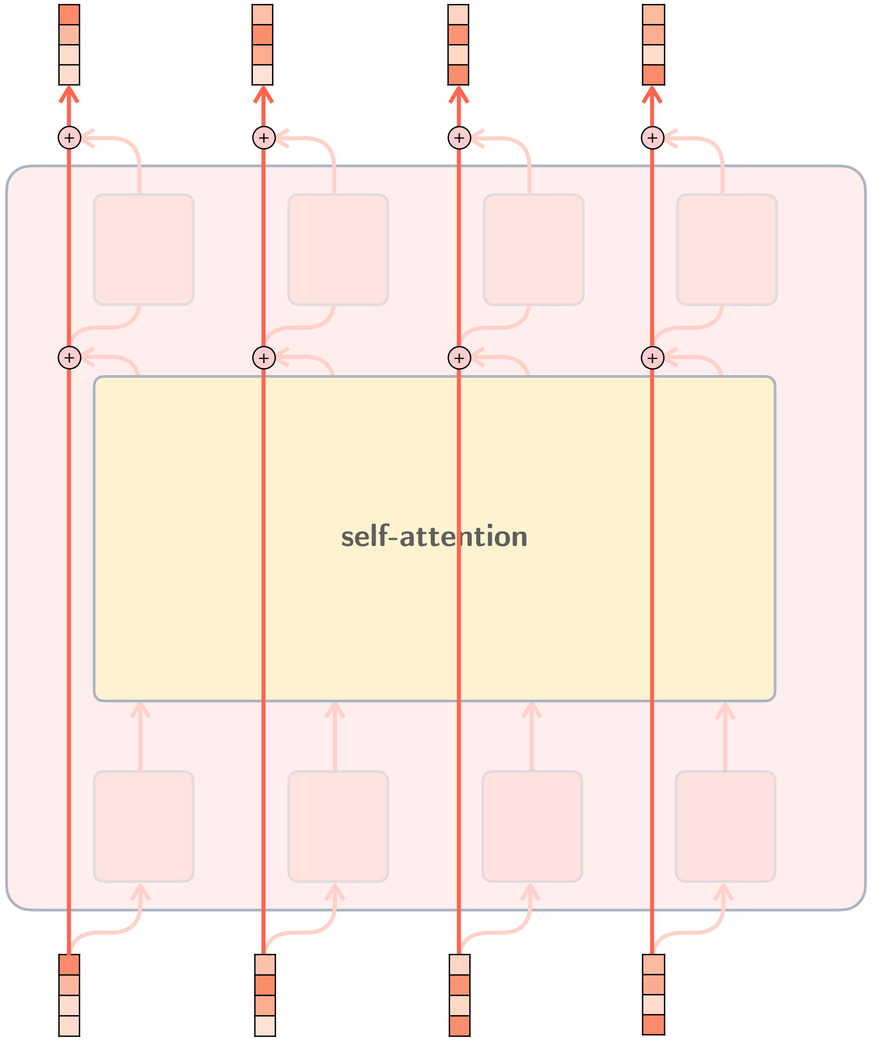


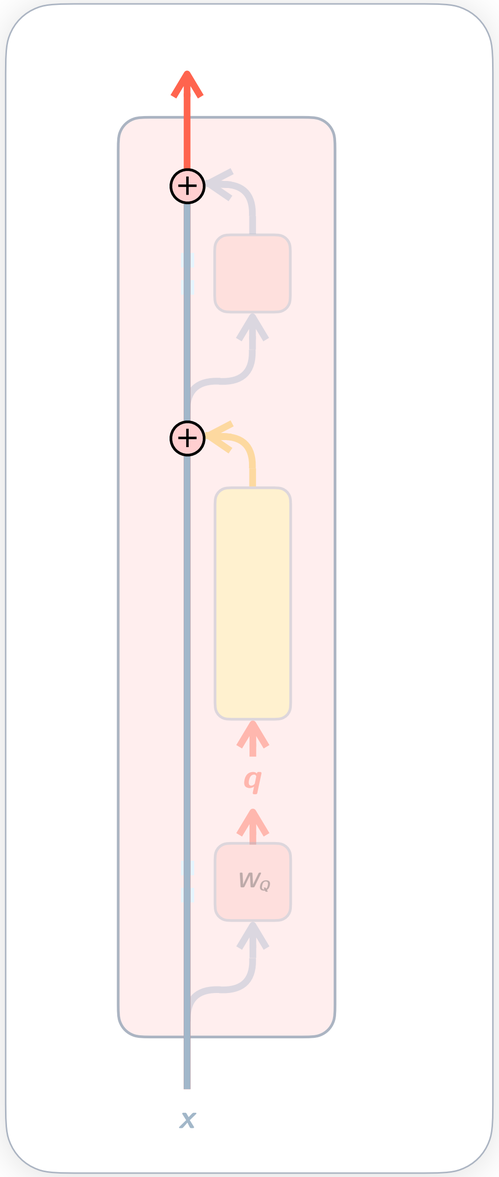
Self-Attention

The ***residual stream***

Each token has its own ***residual stream***



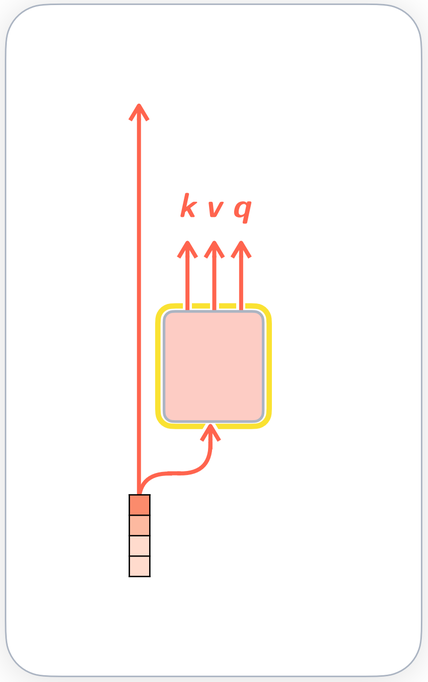


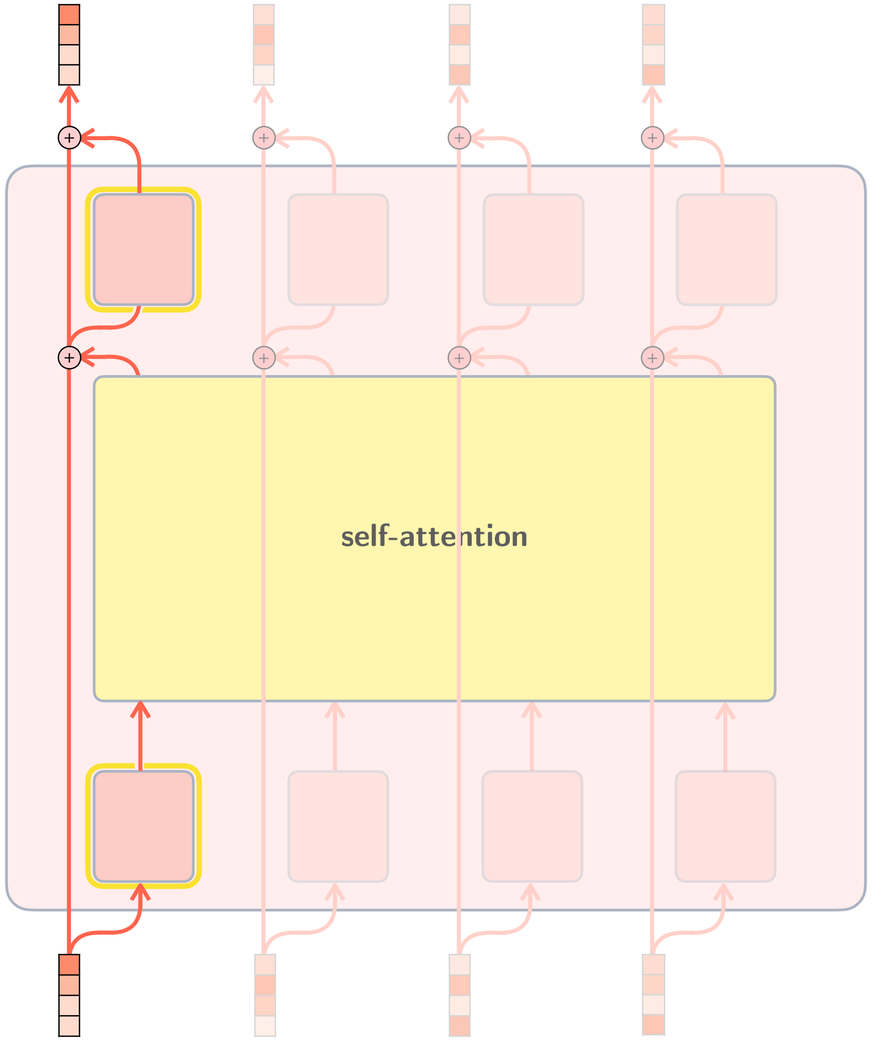


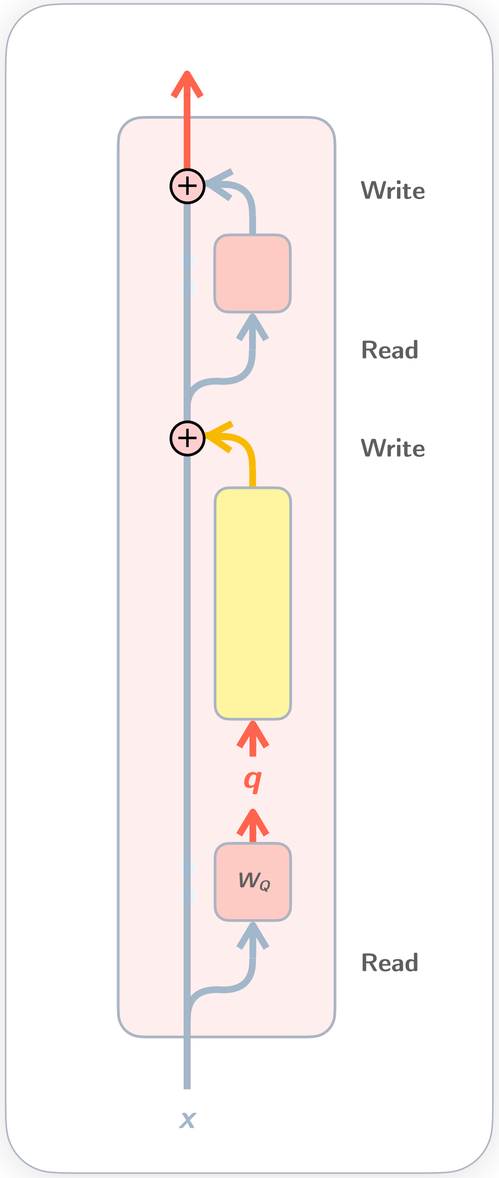
The ***residual stream***

Each token has its own ***residual stream***

"**Reading**" and "**writing**" to the residual stream





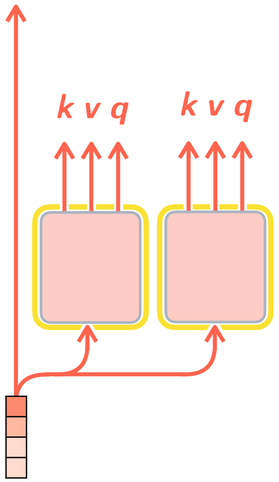


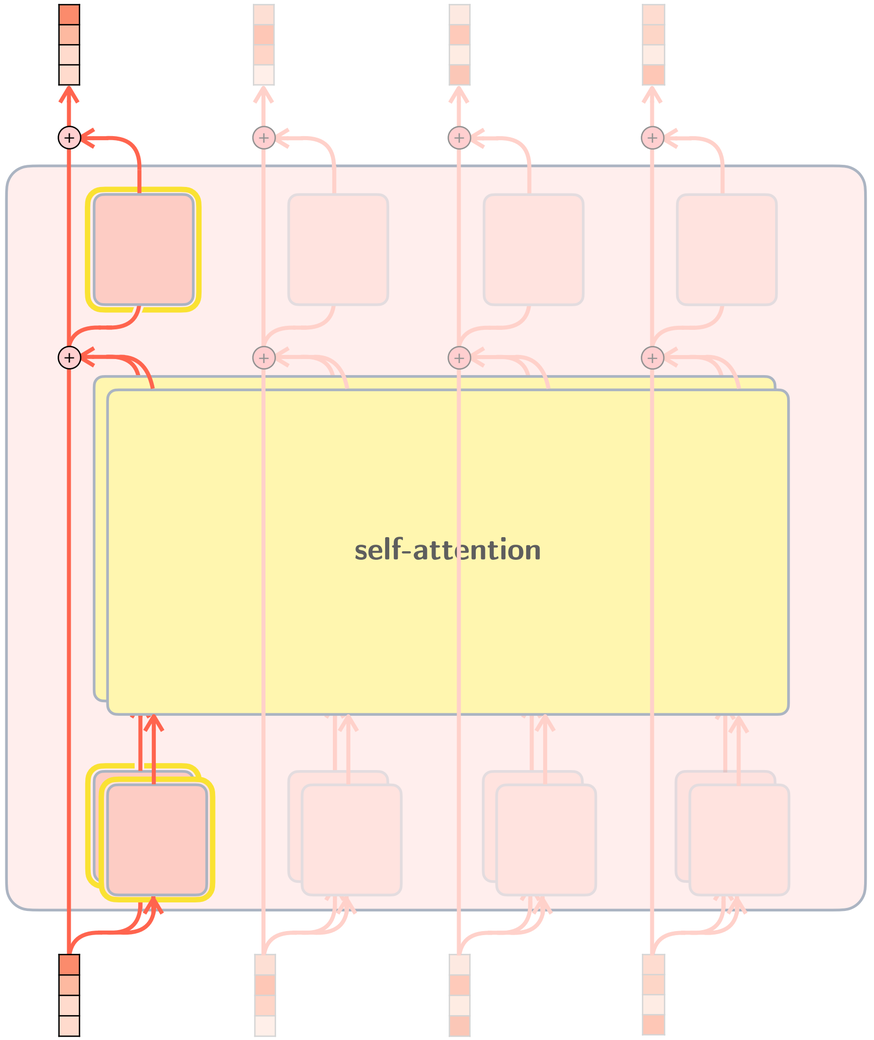
Self-Attention

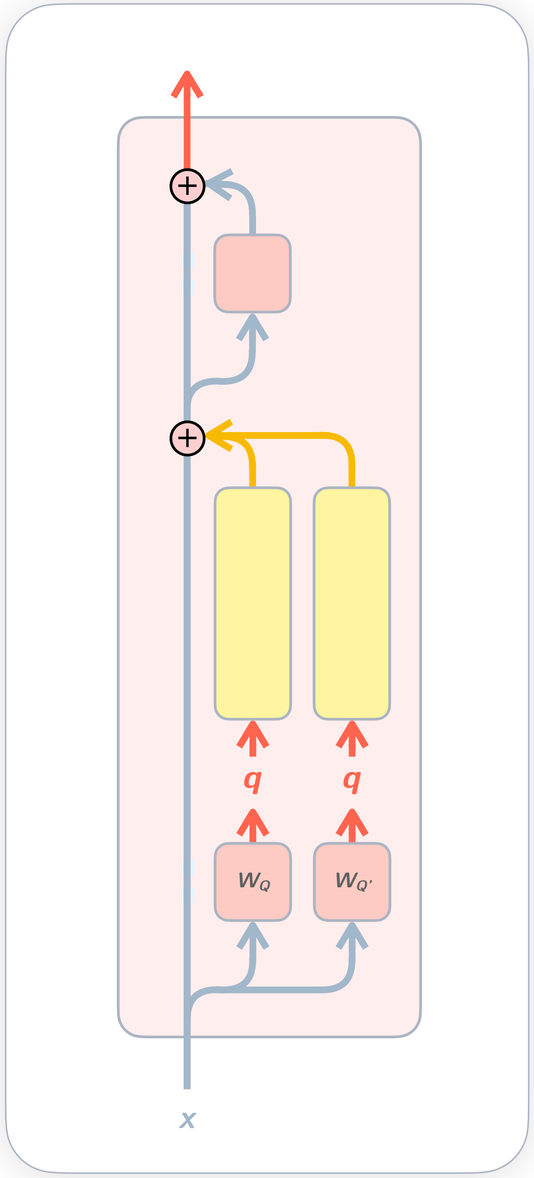
The ***residual stream***

Each token has its own ***residual stream***

"**Reading**" and "**writing**" to the residual stream

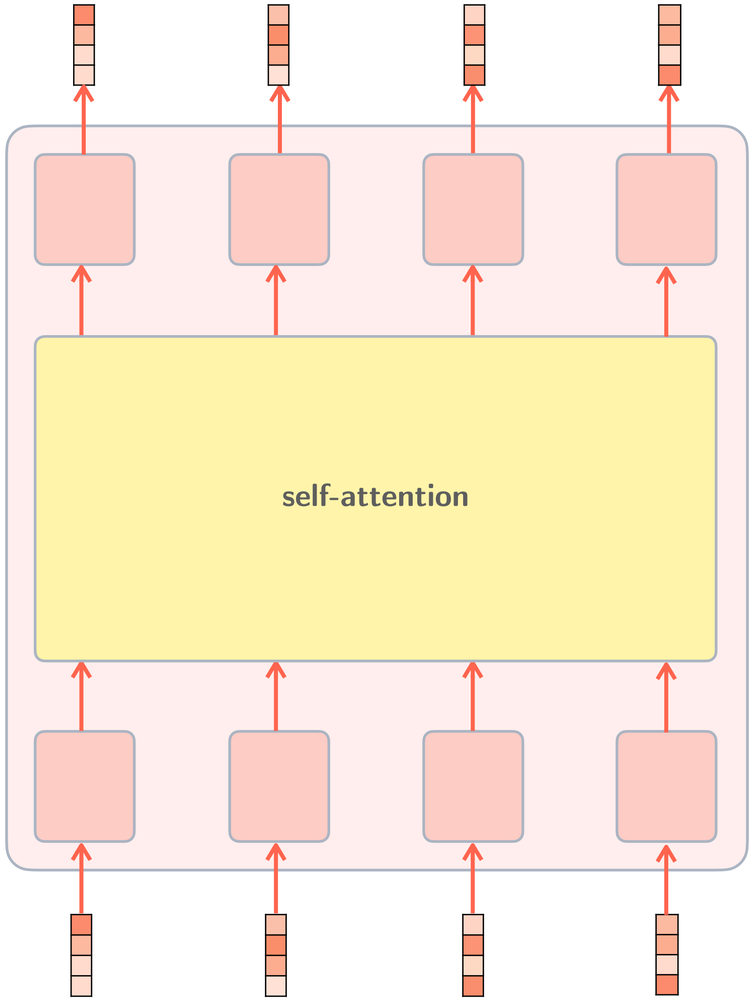






Self-Attention

Feed **encoder** information into **encoder**



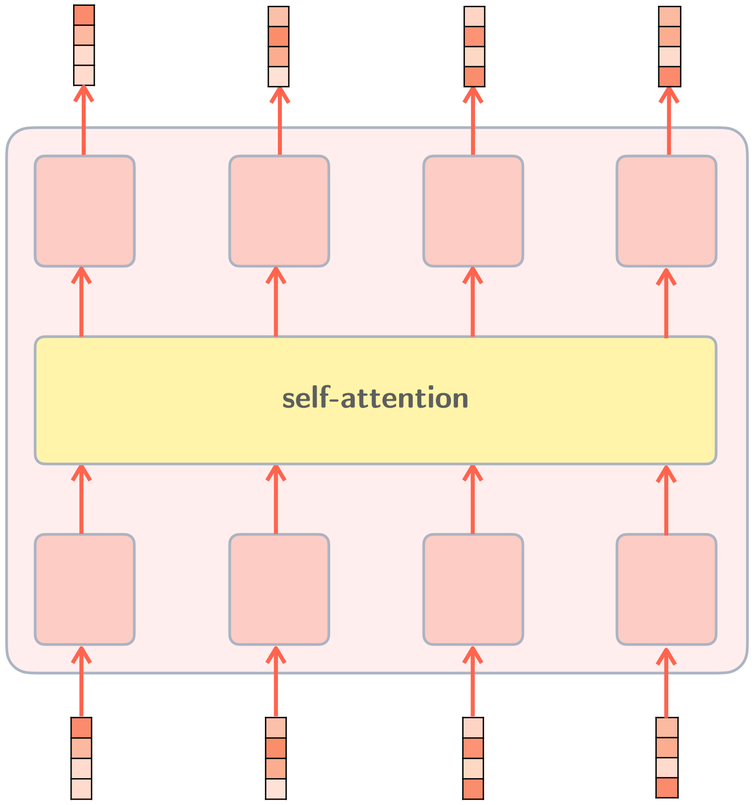
refined **encoder sequence *X'***

refines and ***contextualizes*** **encoder sequence**

encoder sequence *X*

Self-Attention

Feed **encoder** information into **encoder**



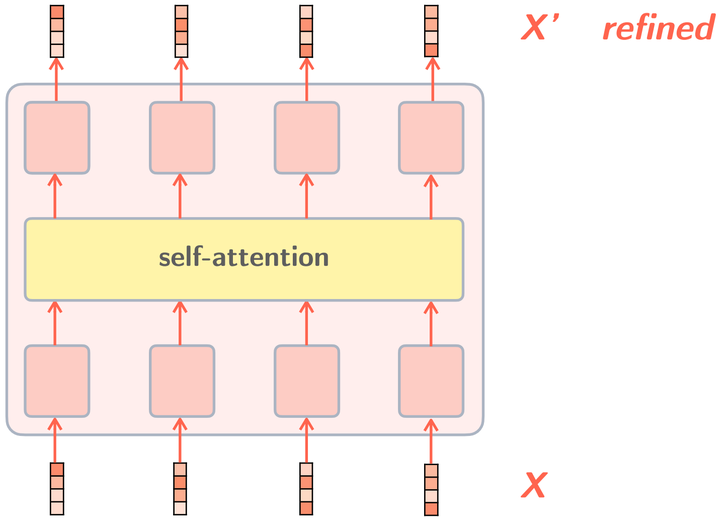
refined **encoder sequence** ***X'***

refines and ***contextualizes*** **encoder sequence**

encoder sequence ***X***

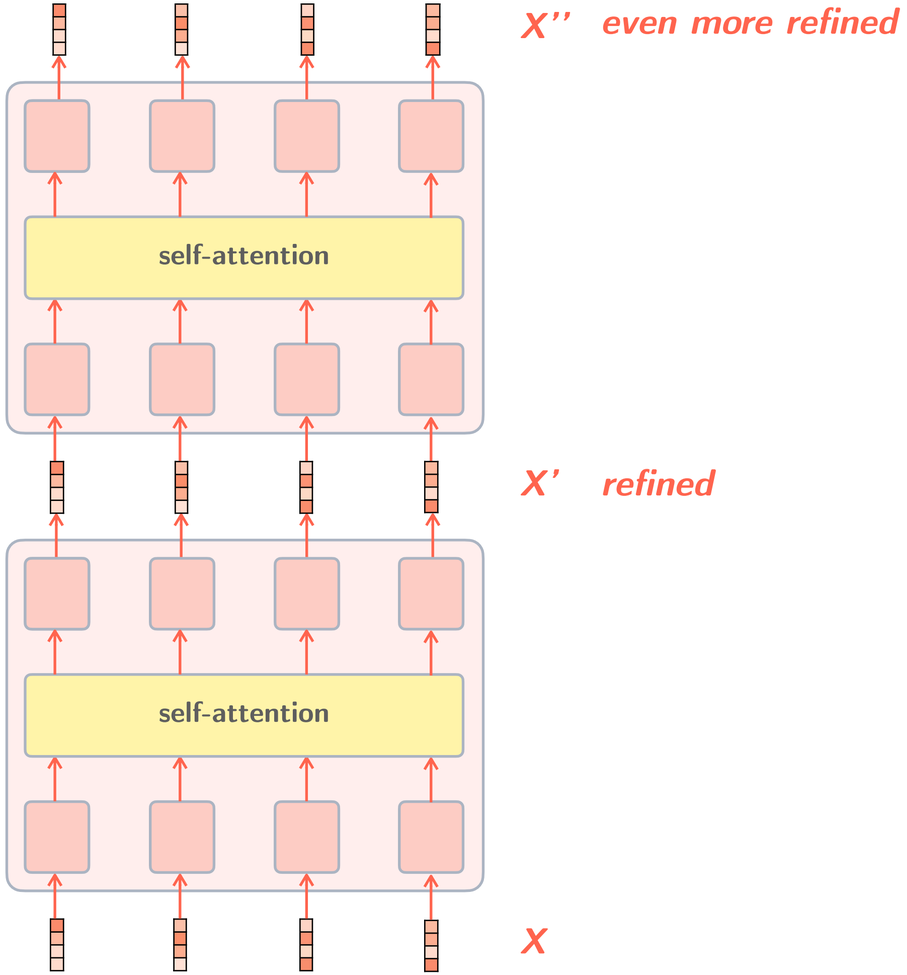
Self-Attention

Feed **encoder** information into **encoder**

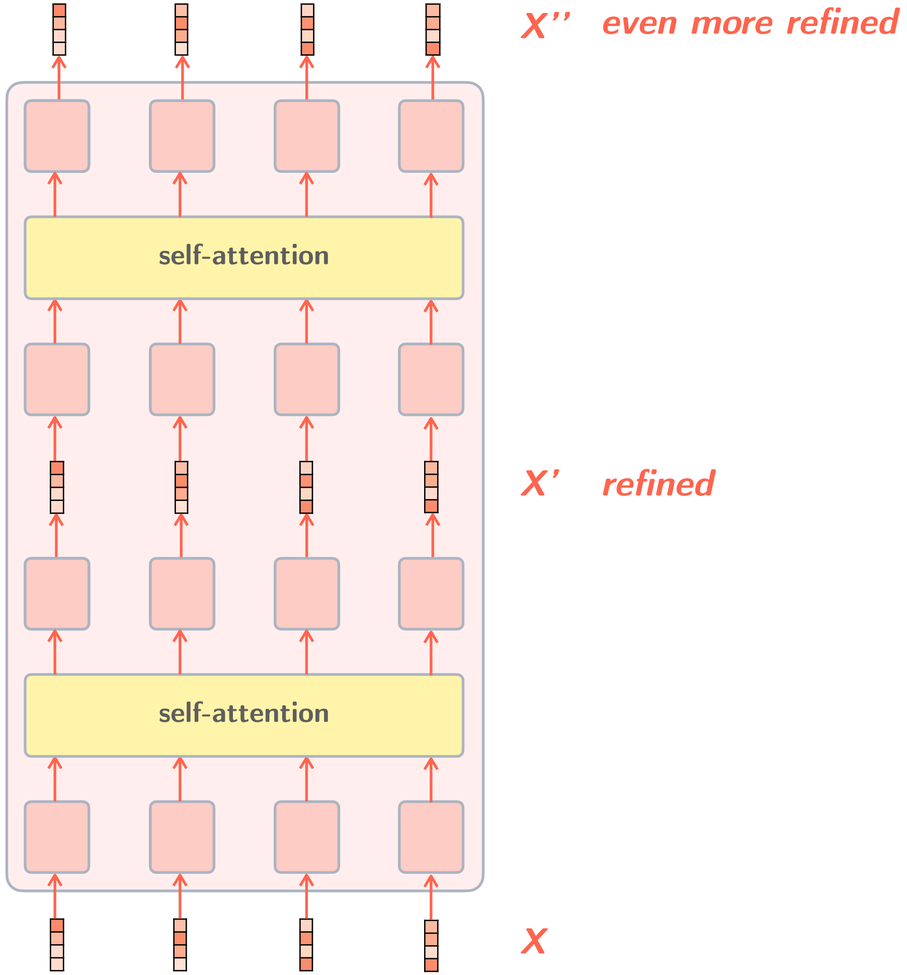


Self-Attention

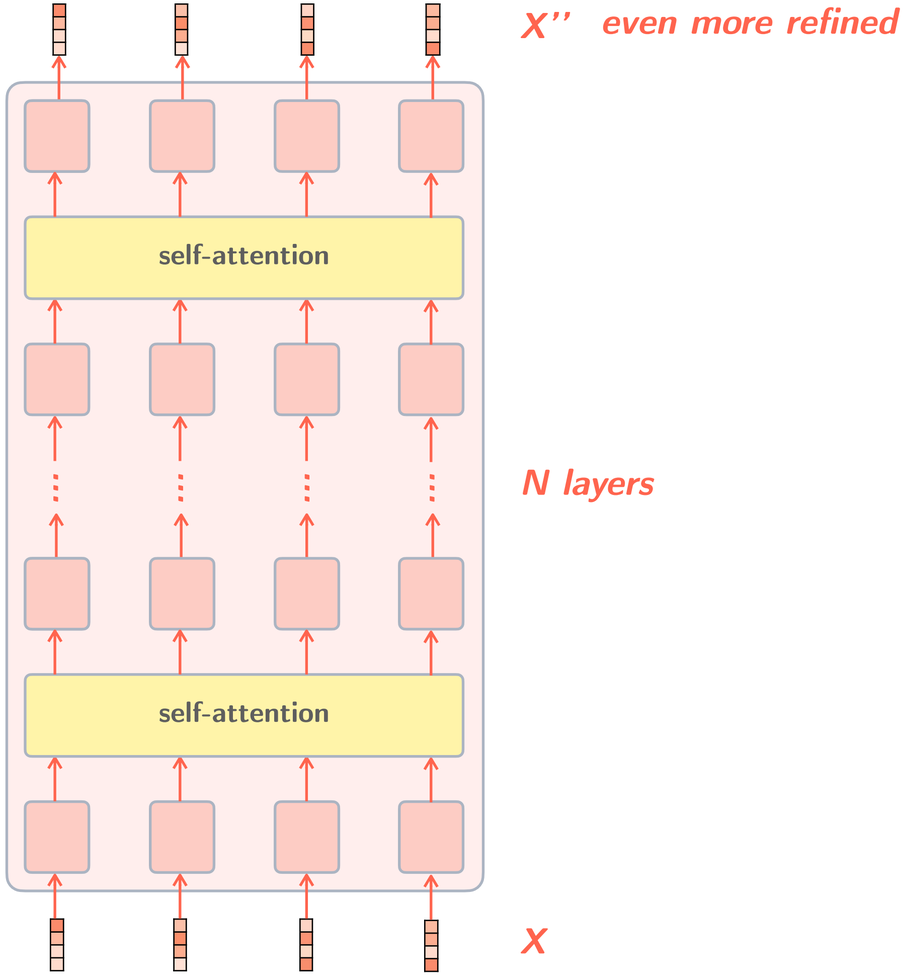
Feed **encoder** information into **encoder**



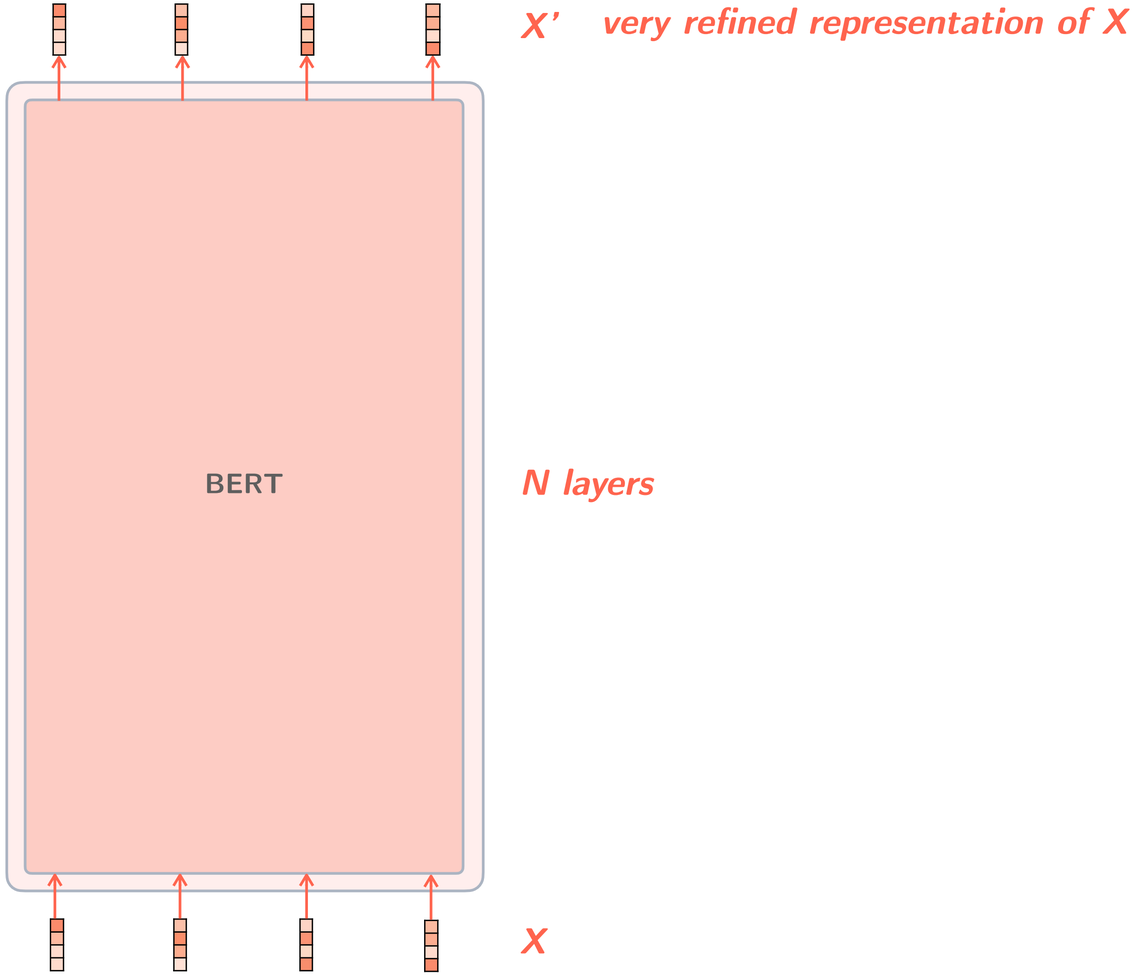
Feed **encoder** information into **encoder**



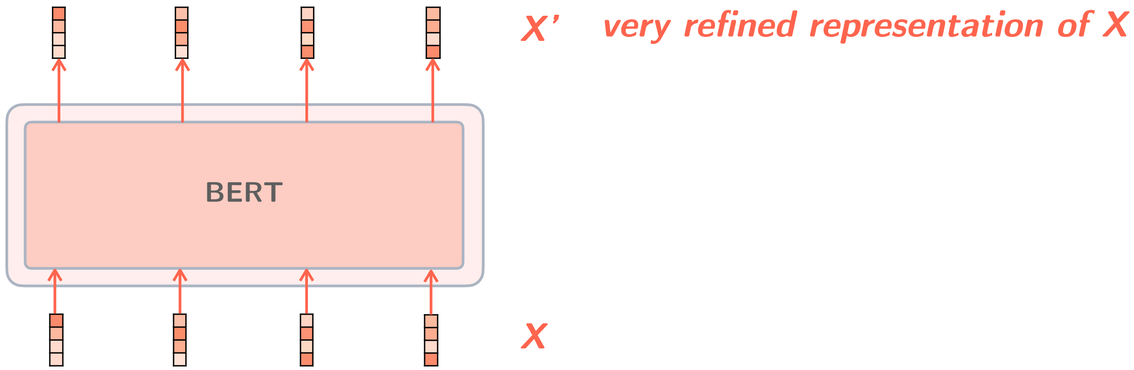
Feed **encoder** information into **encoder**



BERT = Bidirectional Encoder Representations from Transformers



BERT = Bidirectional Encoder Representations from Transformers

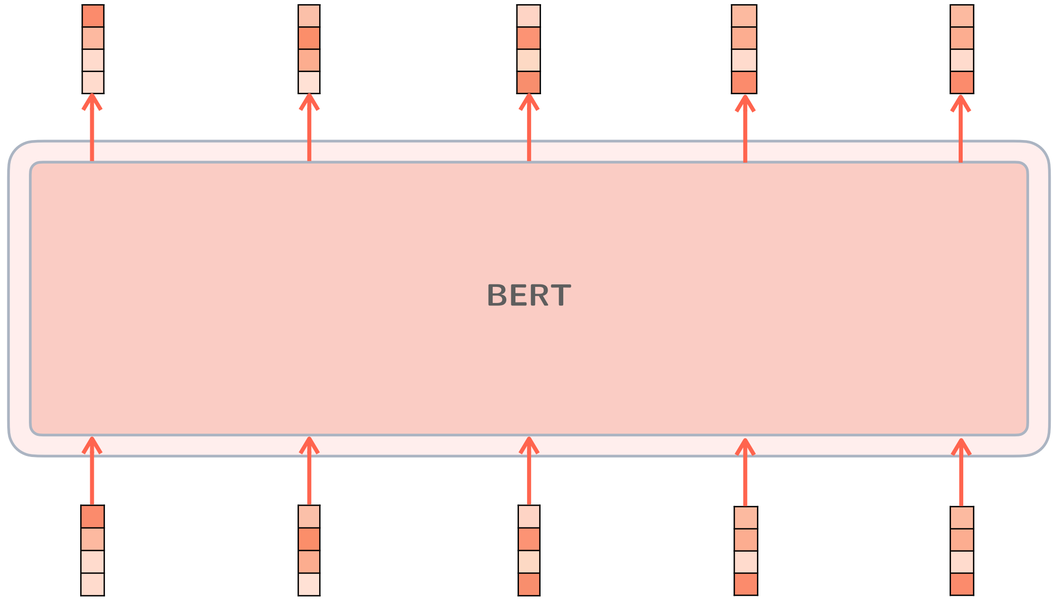


BERT

BERT = Bidirectional Encoder Representations from Transformers

BERT is trained using **masked language modeling (MLM)**

and **next sentence prediction (NSP)**



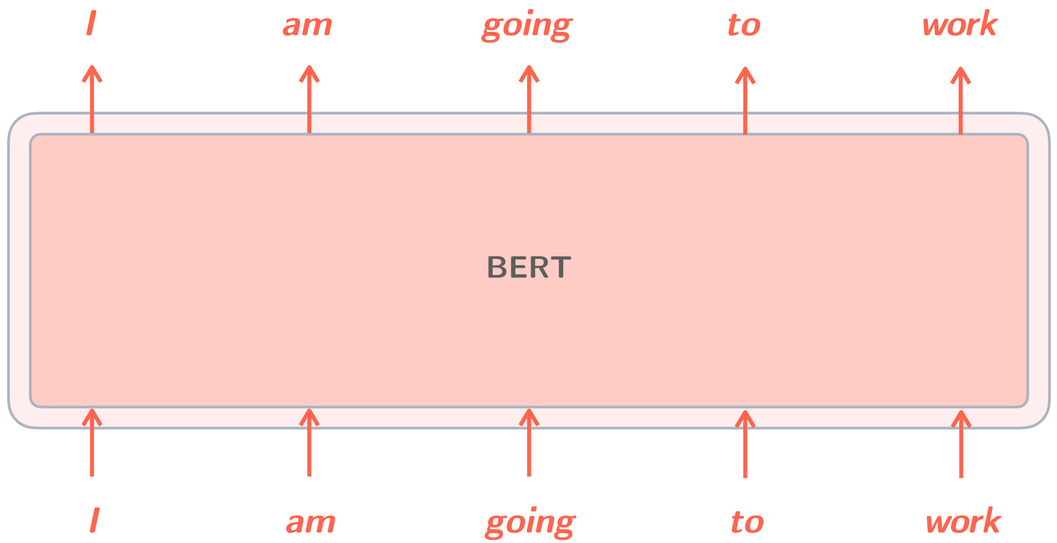
Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

BERT is trained using **masked language modeling (MLM)**

and **next sentence prediction (NSP)**



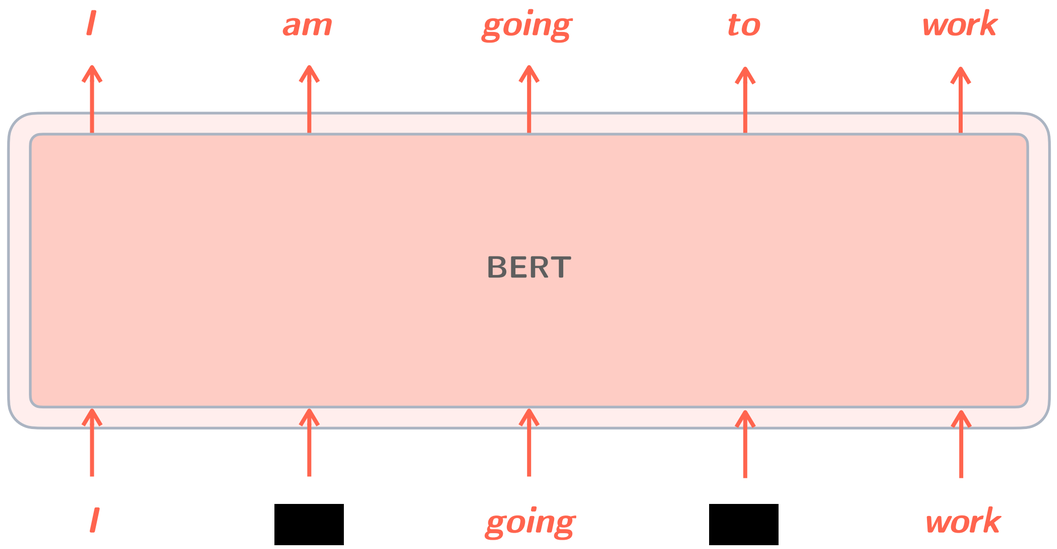
Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

BERT is trained using **masked language modeling (MLM)**

and **next sentence prediction (NSP)**



Randomly mask 15% of tokens

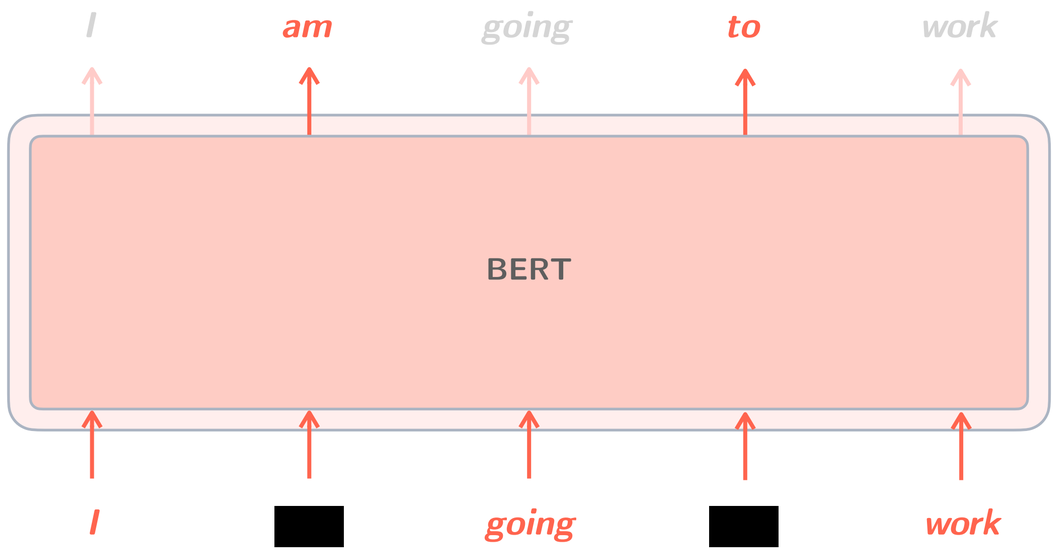
Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

BERT is trained using **masked language modeling (MLM)**

and **next sentence prediction (NSP)**



We know the correct answer!

Randomly mask 15% of tokens

Based on "The Illustrated Transformer" by Jay Alammar

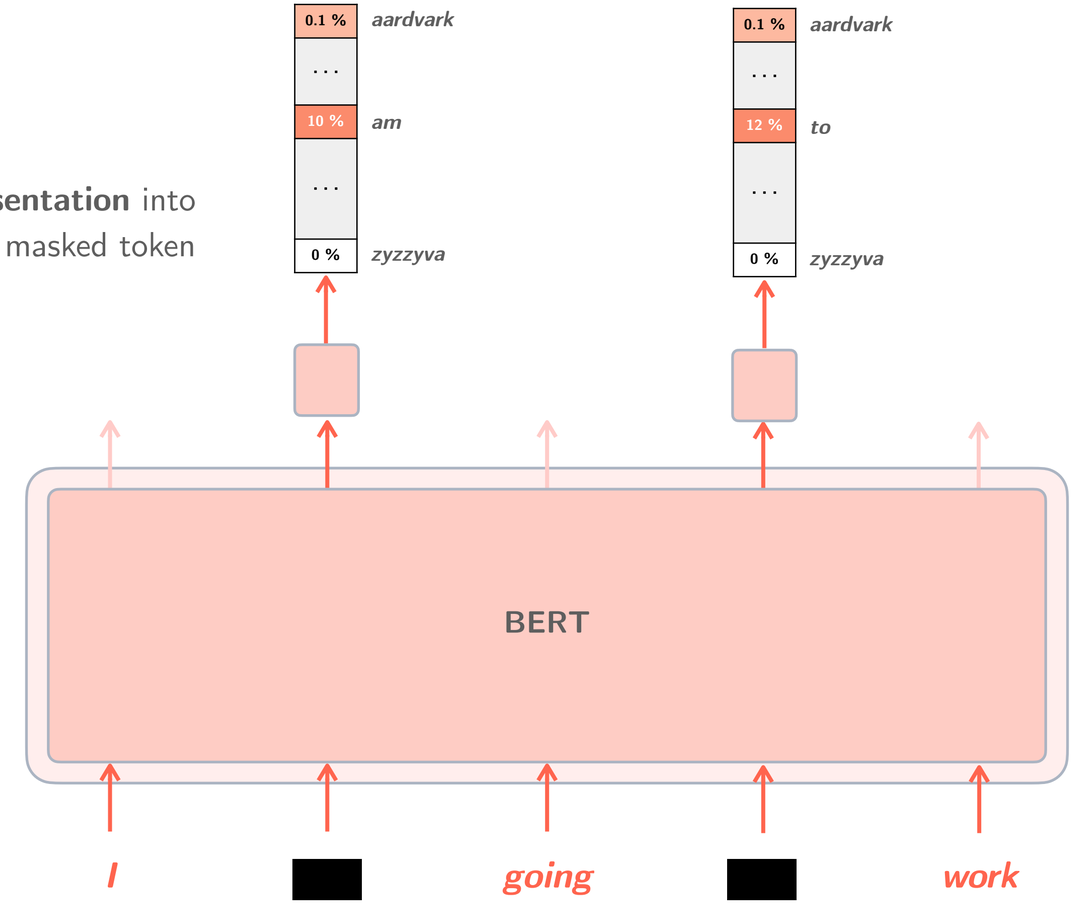
BERT

BERT = Bidirectional Encoder Representations from Transformers

BERT is trained using **masked language modeling (MLM)**

and **next sentence prediction (NSP)**

Turn **refined representation** into prediction of the masked token



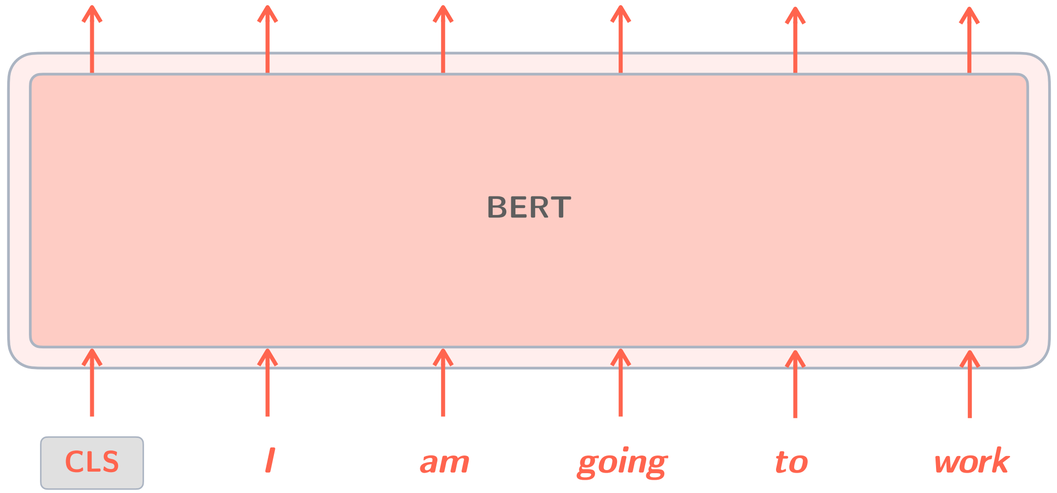
Randomly mask 15% of tokens

Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

Once trained, **BERT** can be ***fine-tuned*** on specific language tasks



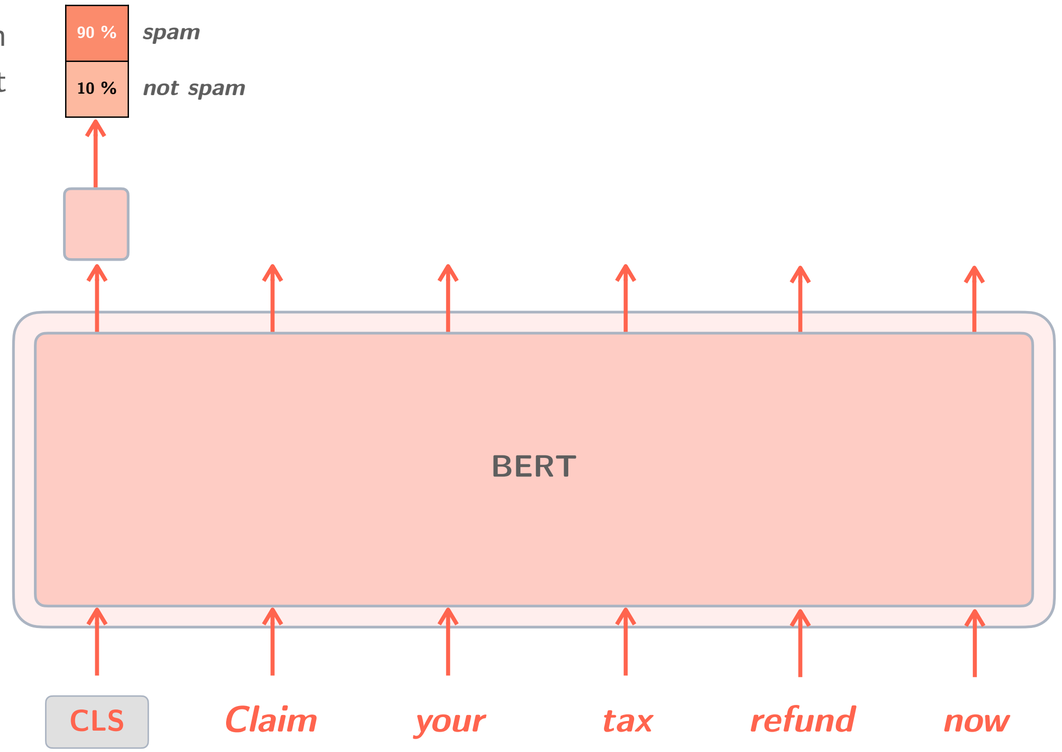
Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

Once trained, **BERT** can be ***fine-tuned*** on specific language tasks

Turn **refined representation** of CLS token into task-specific output



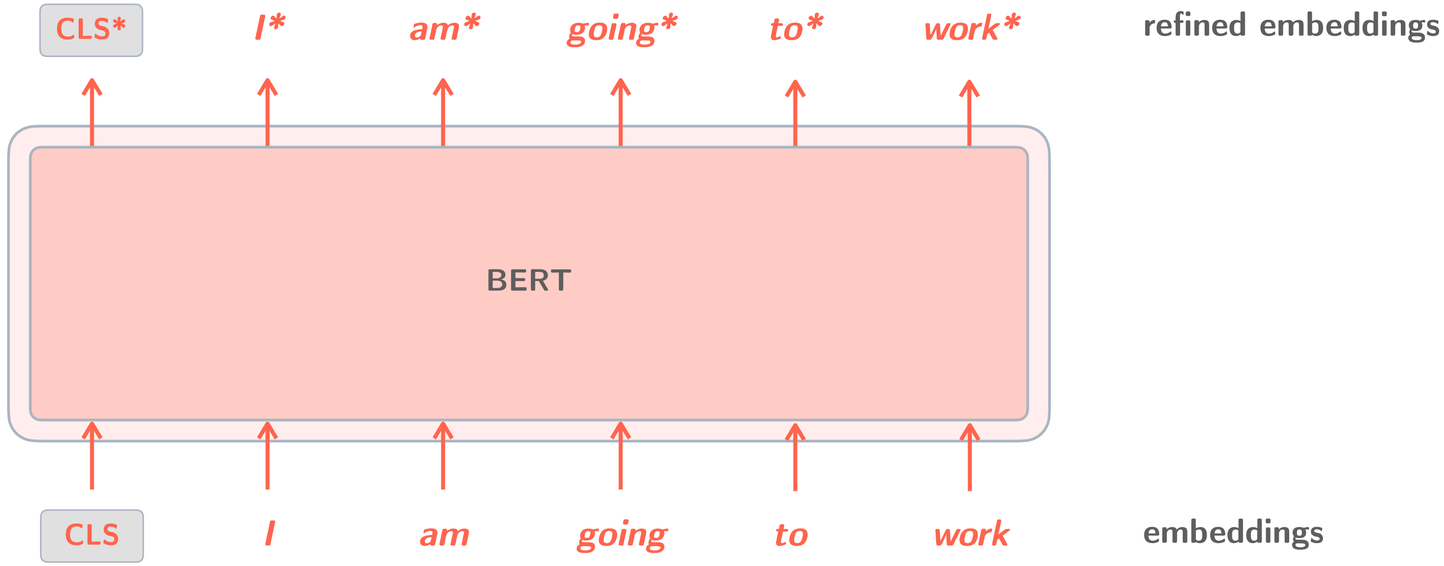
Based on "The Illustrated Transformer" by Jay Alammar

BERT

BERT = Bidirectional Encoder Representations from Transformers

Once trained, **BERT** can be ***fine-tuned*** on specific language tasks

We can think of Transformer encoders as **refining** (contextualizing) token representations (embeddings).

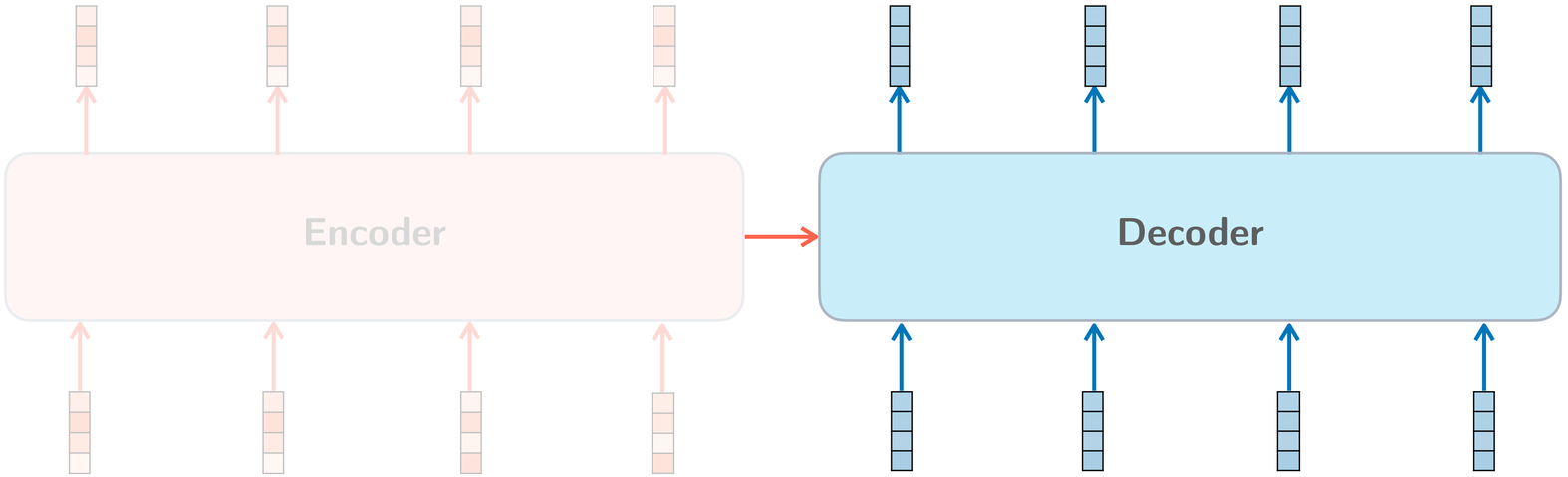


Based on "The Illustrated Transformer" by Jay Alammar

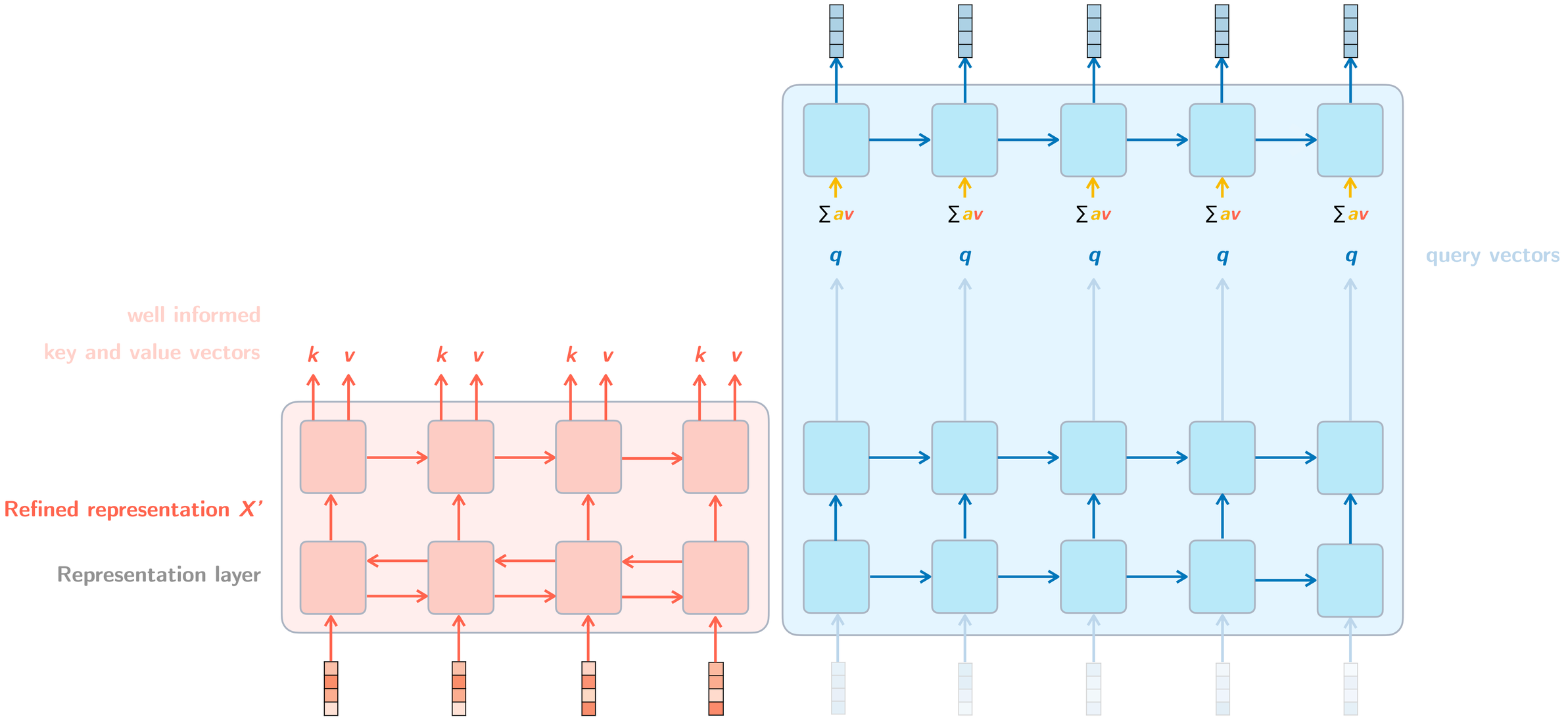
2

Transformer Decoder

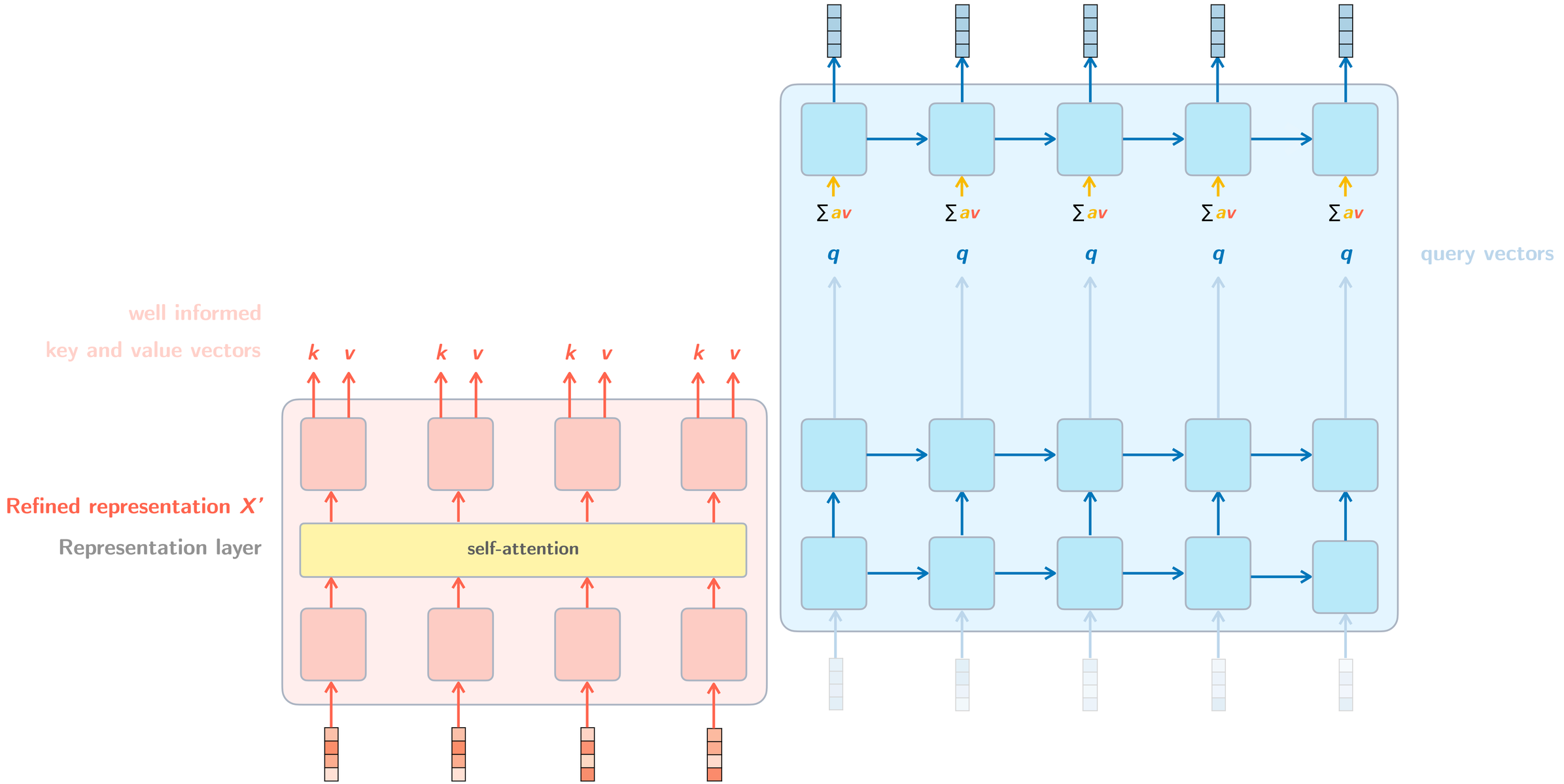
(and GPT)


Encoder
Decoder

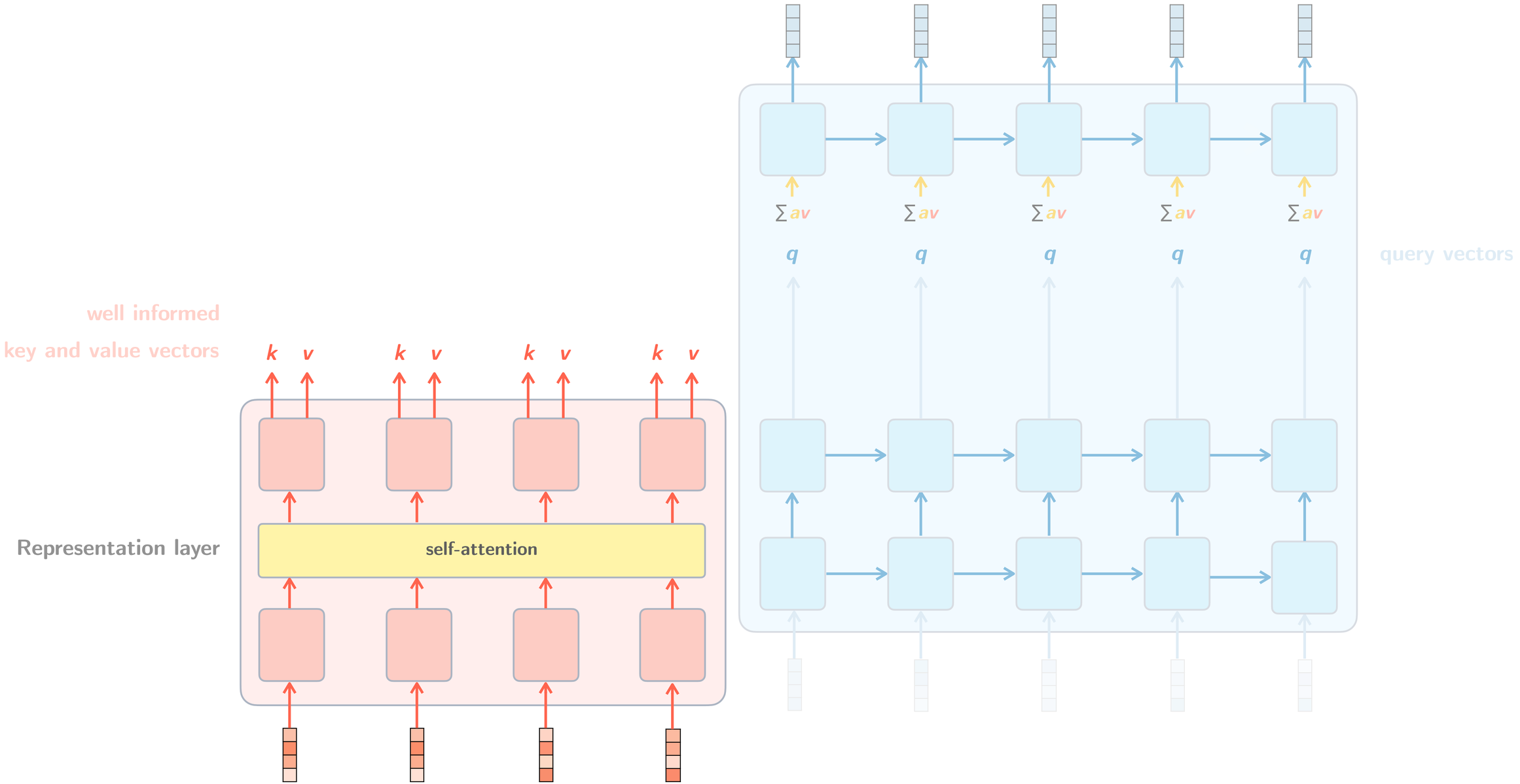
Reminder: Seq2Seq RNN with Attention



Replacing recurrence with self-attention in the encoder

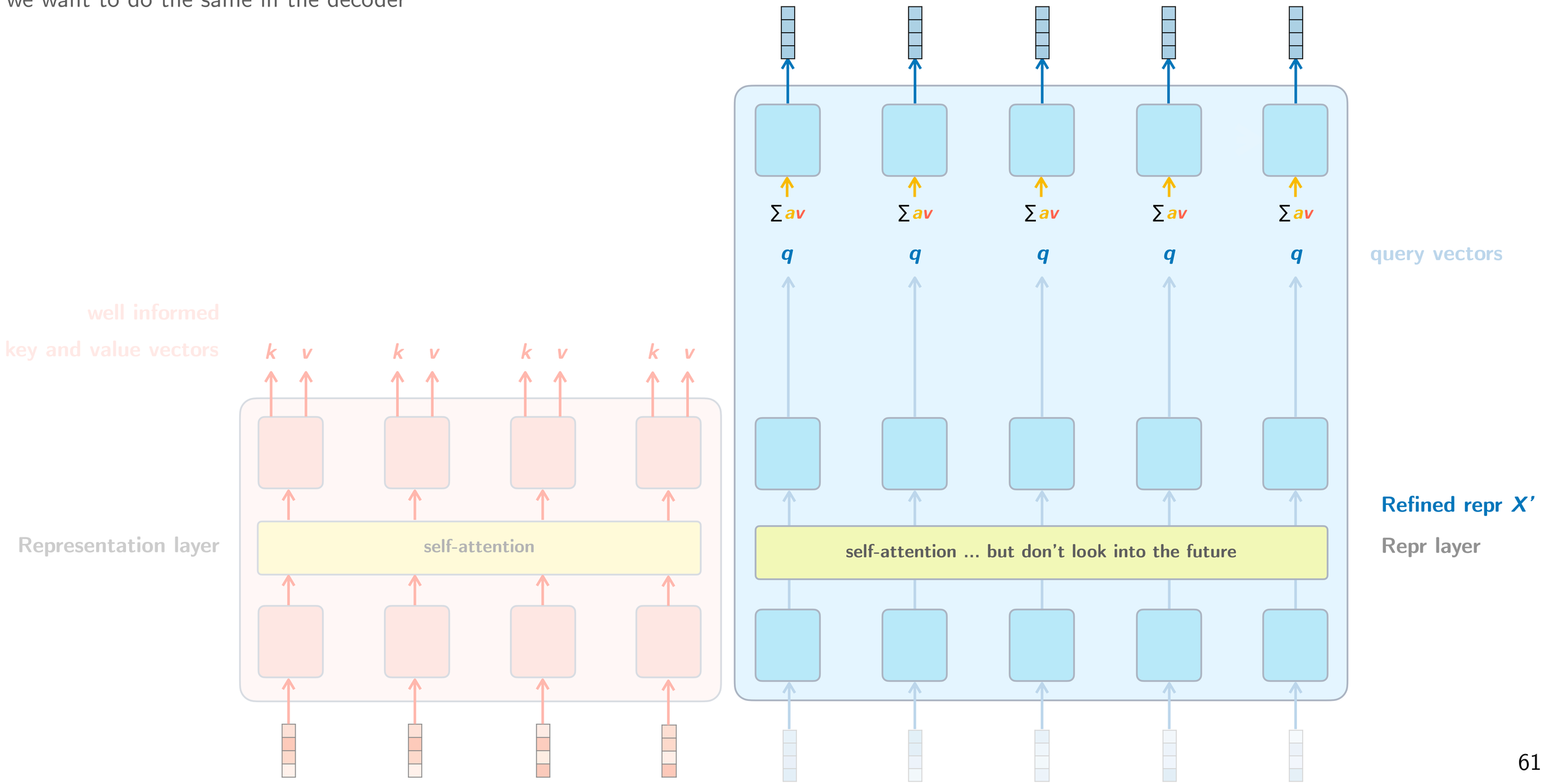


Replacing recurrence with self-attention in the encoder



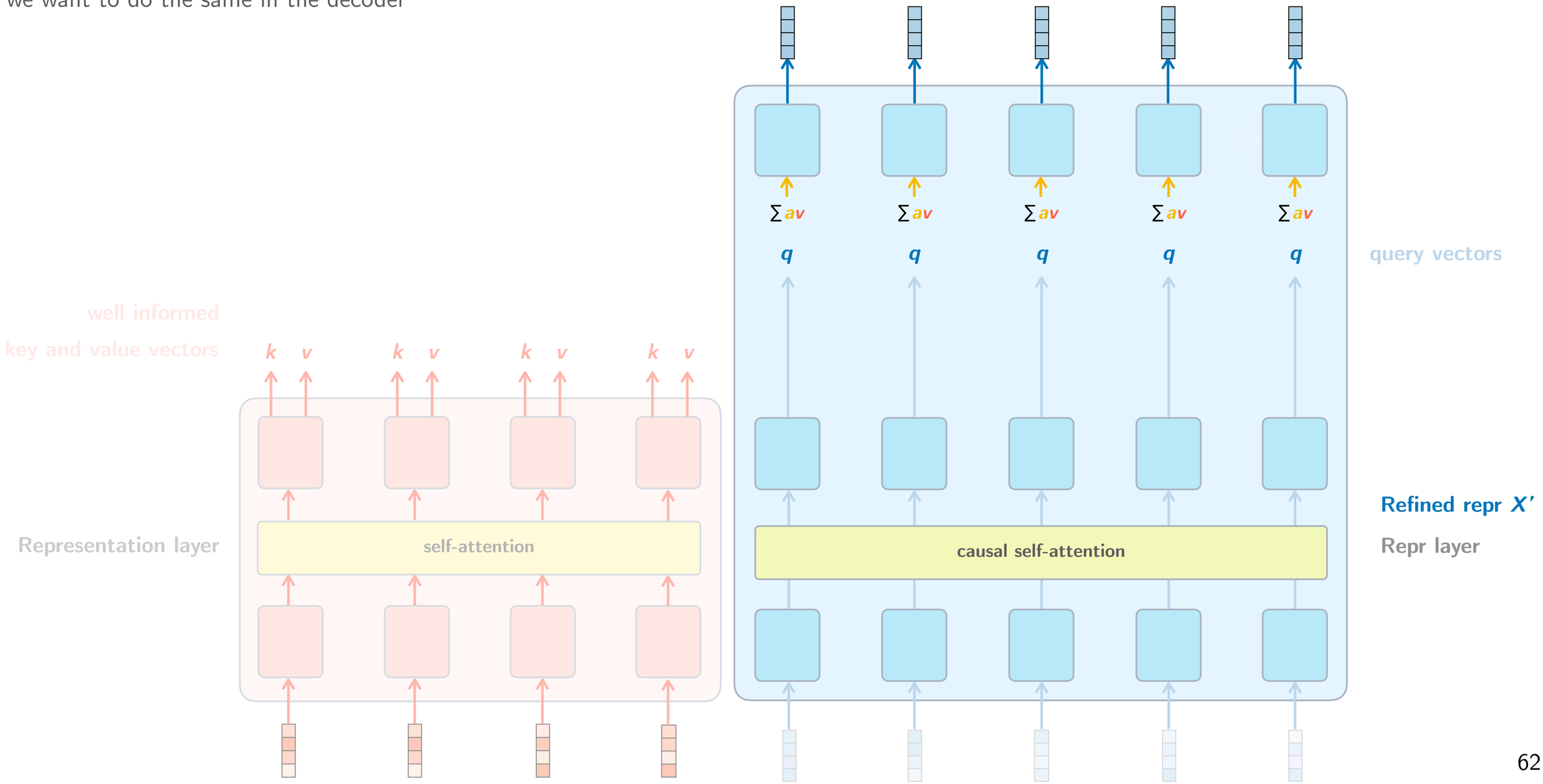
Replacing recurrence with self-attention in the encoder

Now we want to do the same in the decoder



Replacing recurrence with self-attention in the encoder

Now we want to do the same in the decoder



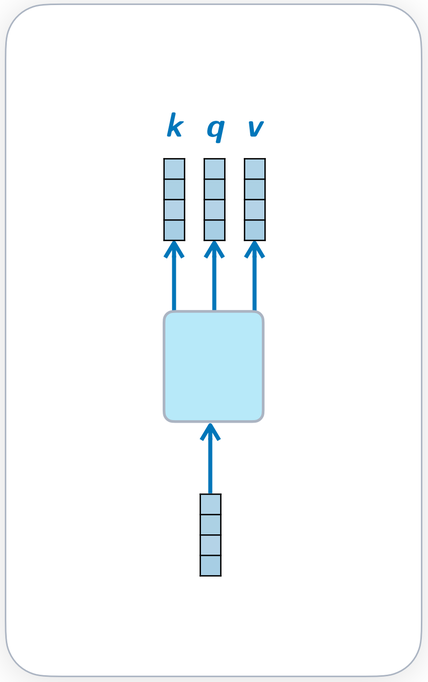
Causal Self-Attention

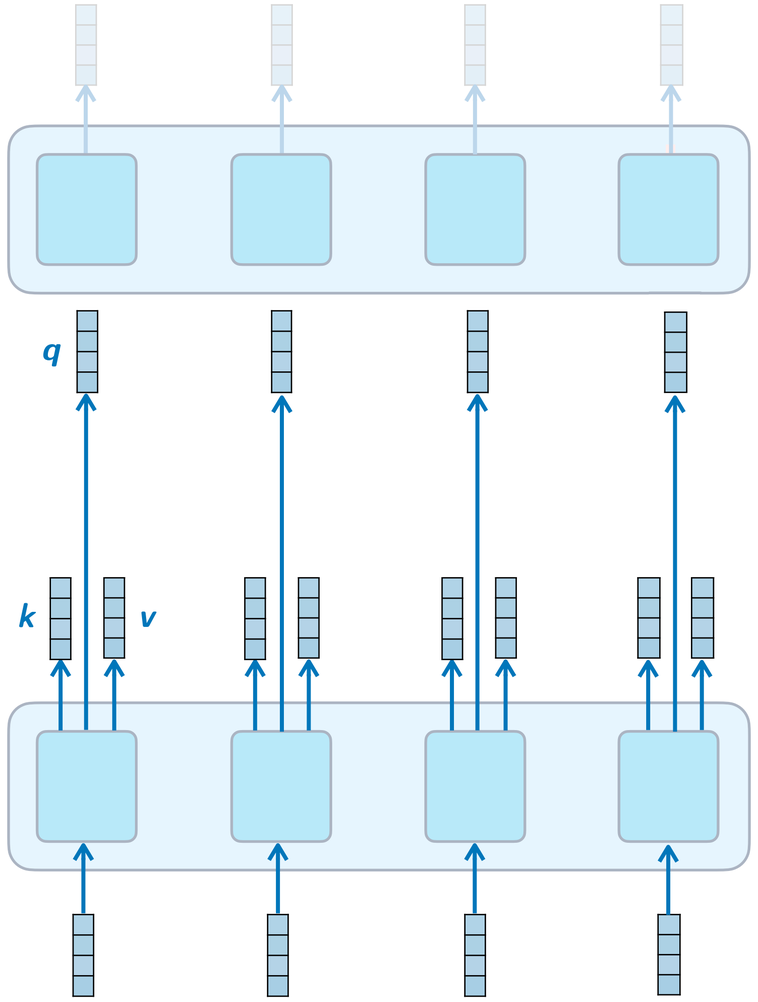
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

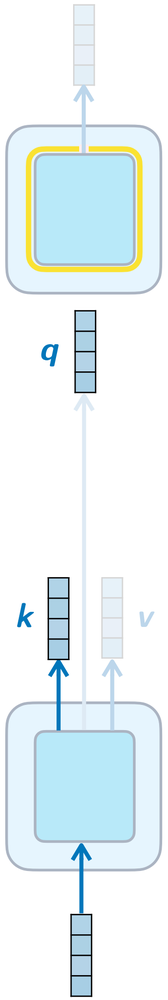
Causal Self-Attention

Think of ***N-to-N***

Feed **decoder** information into **decoder**

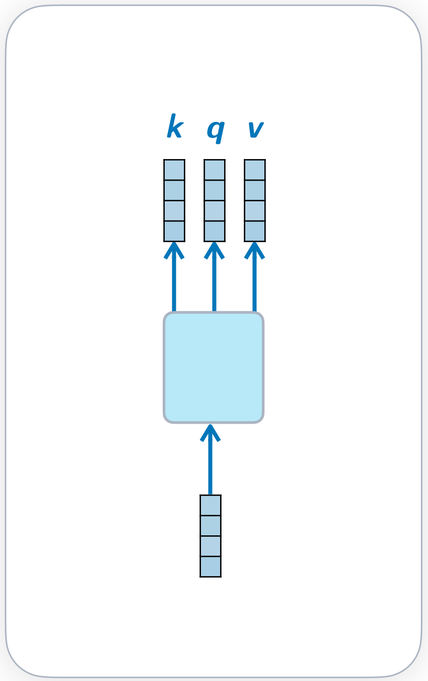
come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors



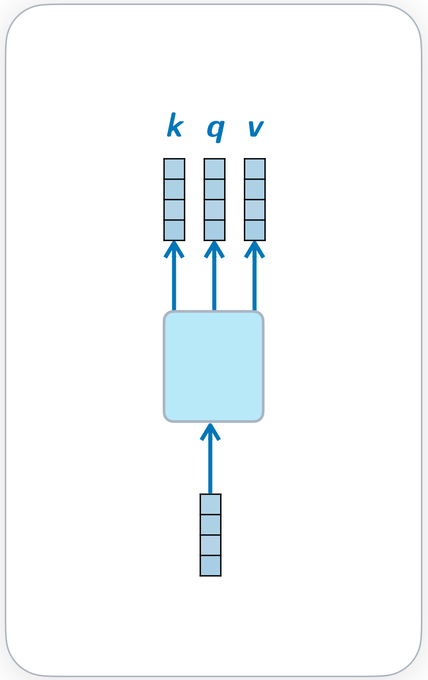
Causal Self-Attention

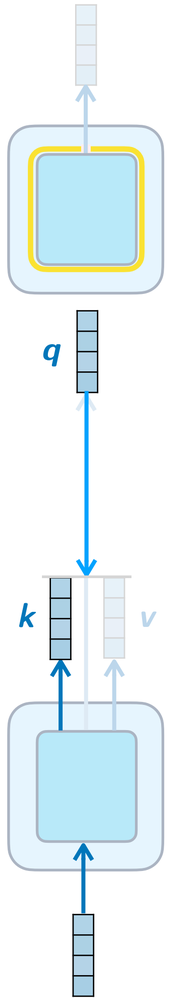
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

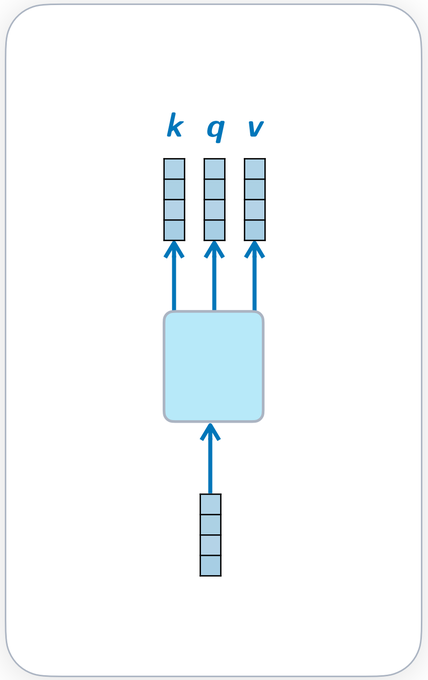
Causal Self-Attention

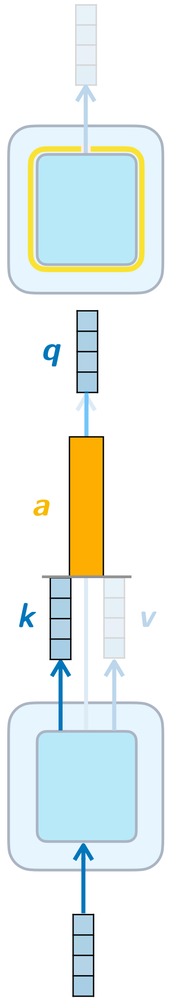
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

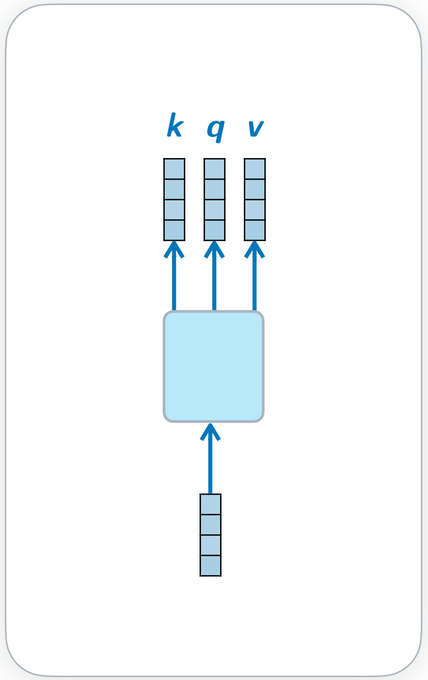
Causal Self-Attention

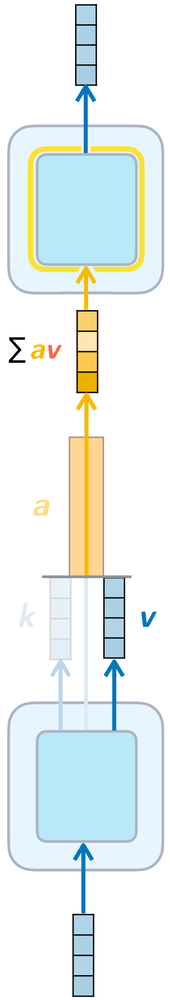
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

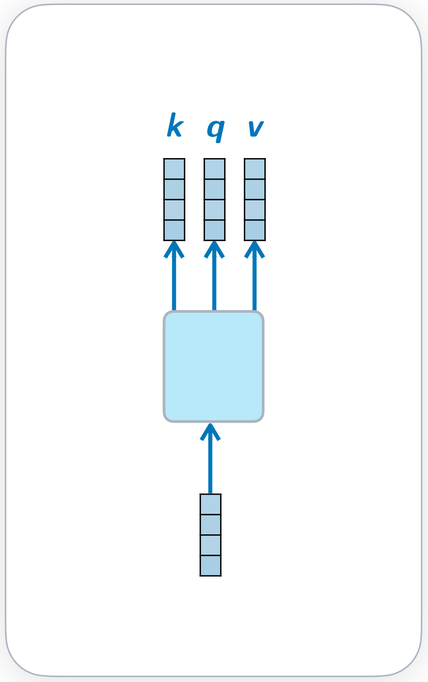
Causal Self-Attention

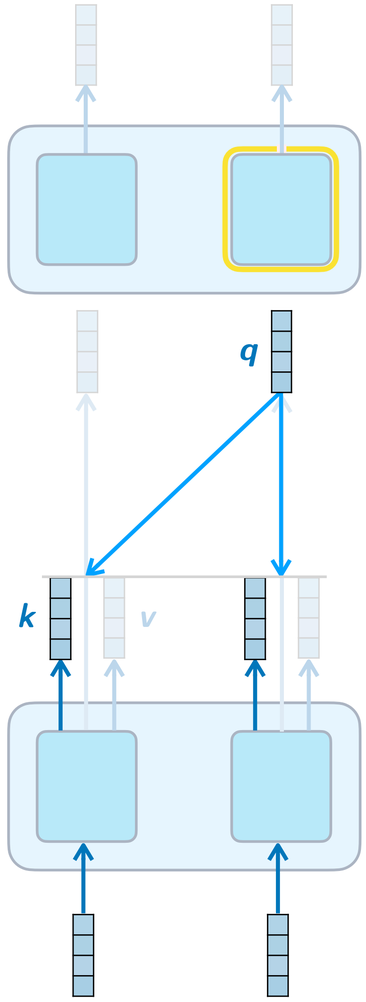
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

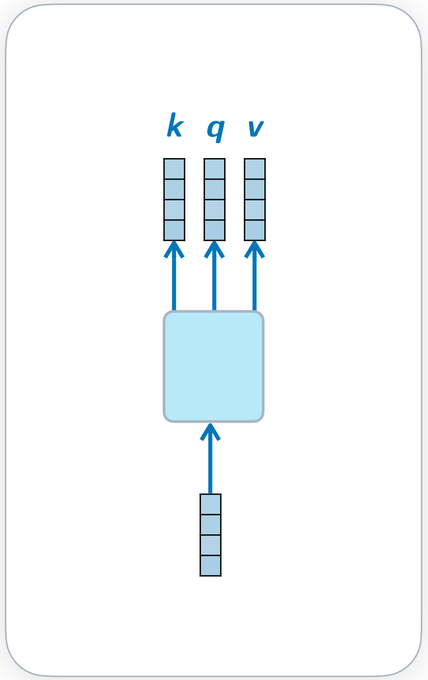
Causal Self-Attention

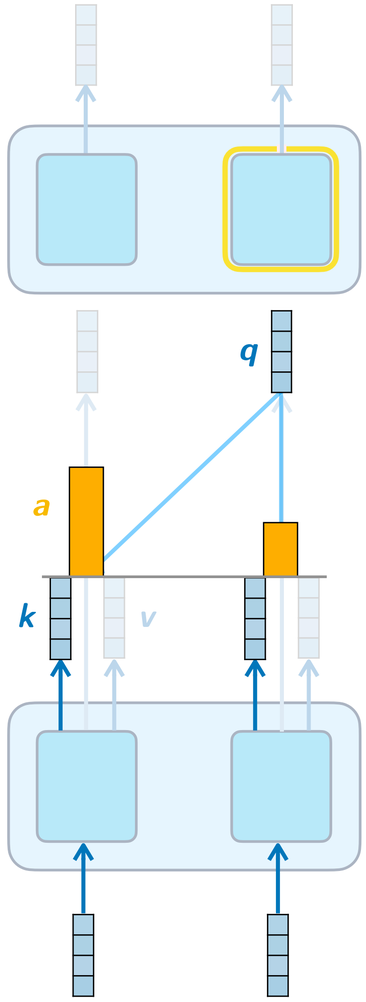
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

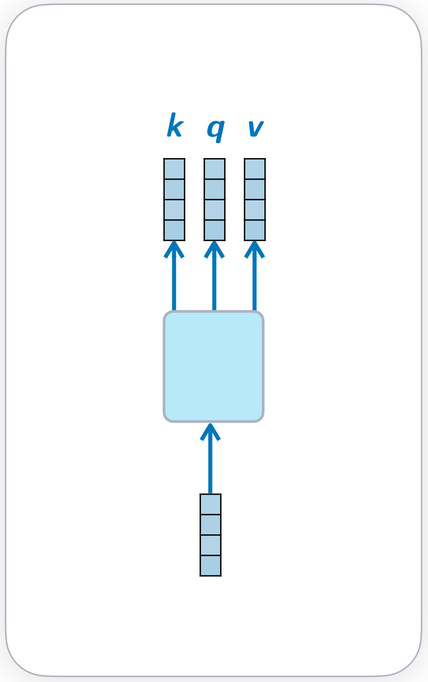
Causal Self-Attention

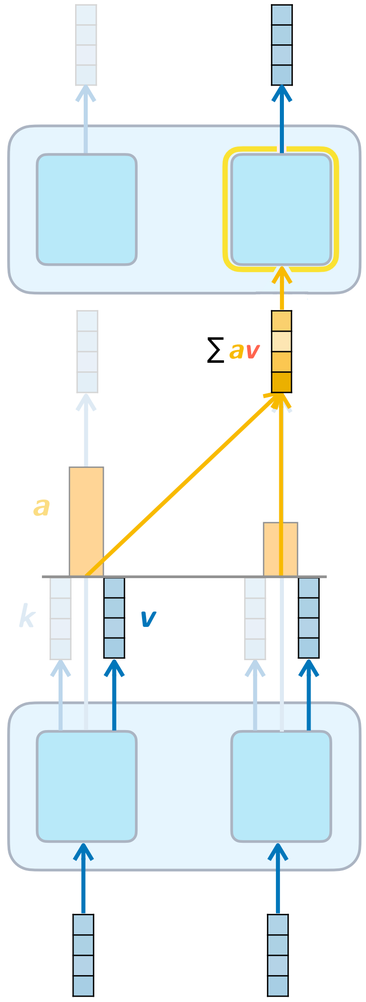
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

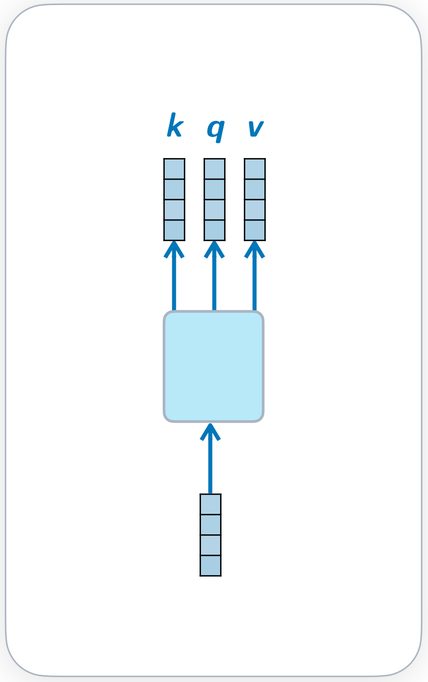
Causal Self-Attention

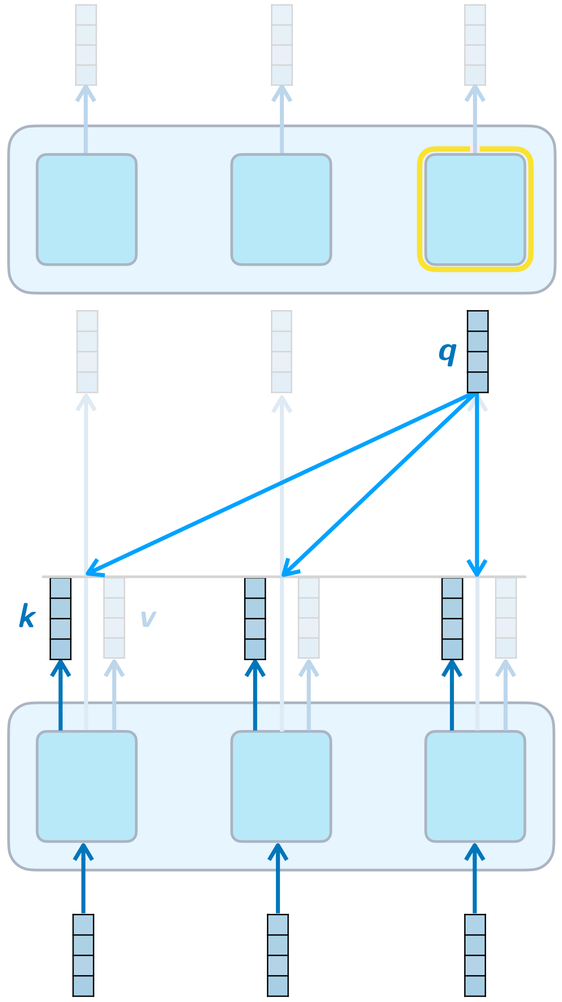
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

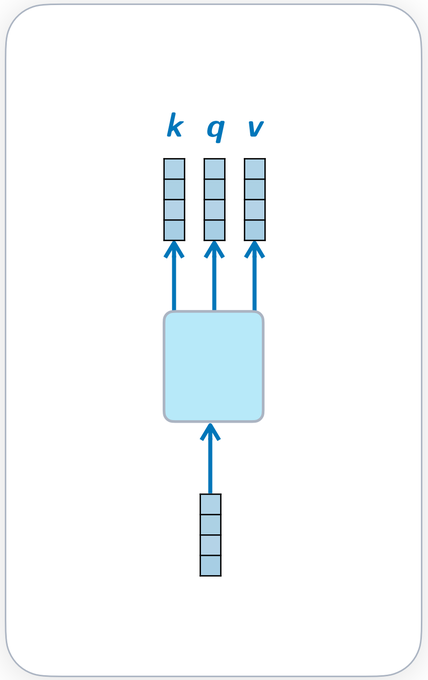
Causal Self-Attention

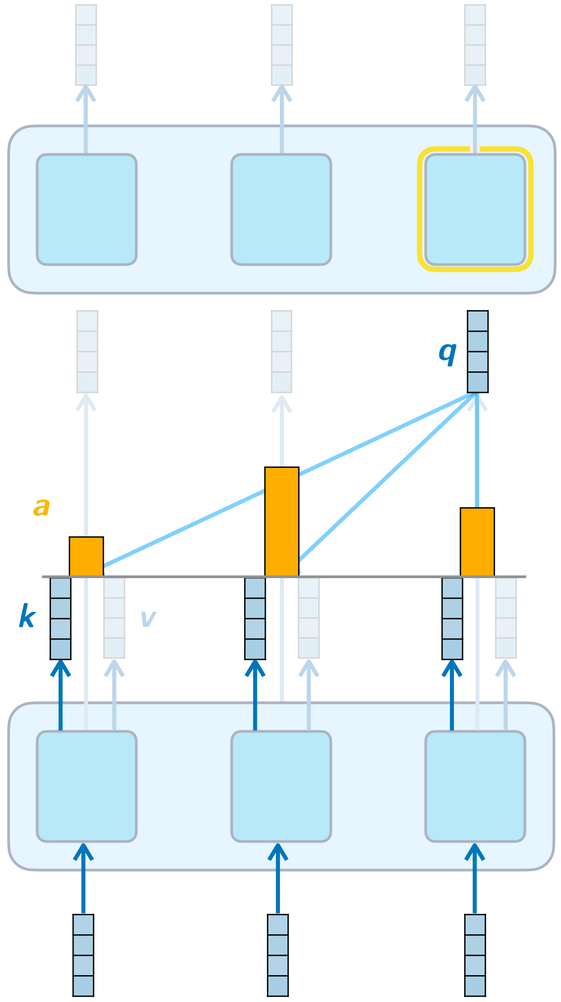
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

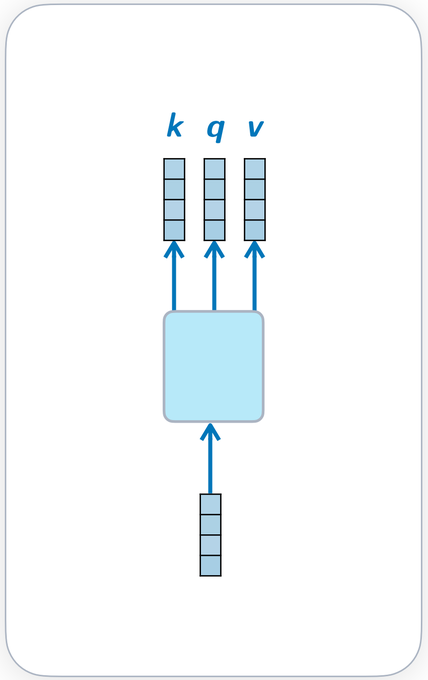
Causal Self-Attention

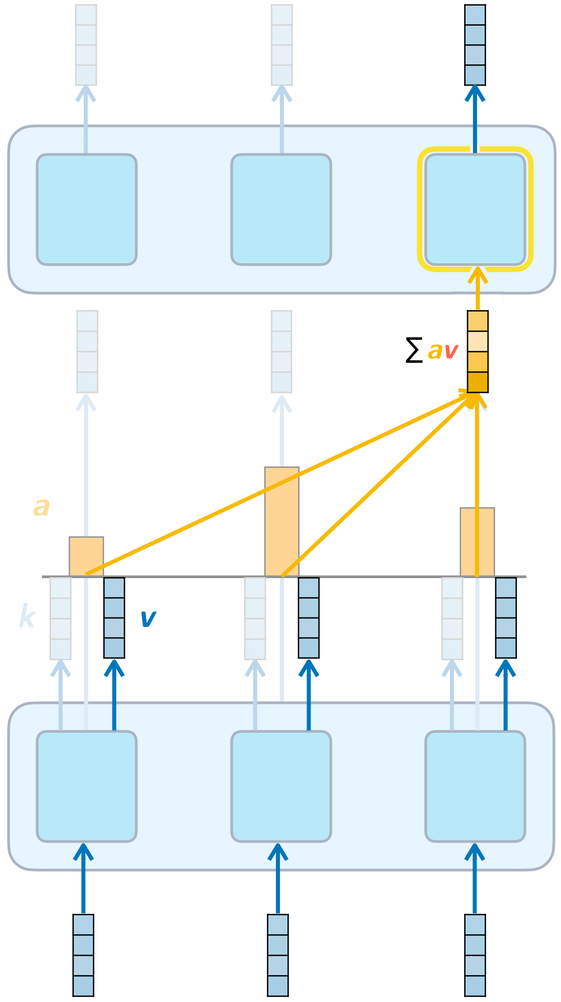
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

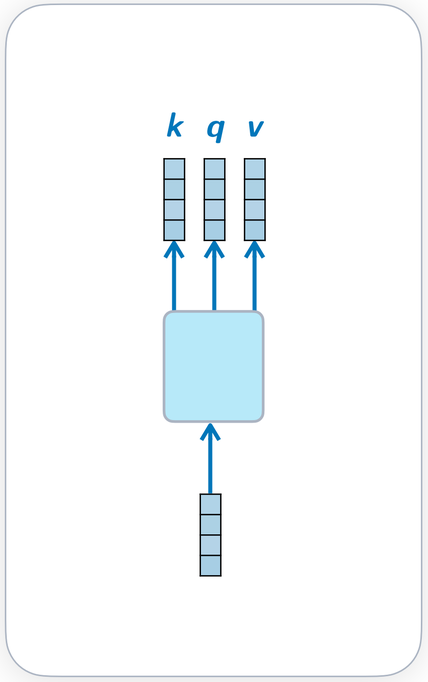
Causal Self-Attention

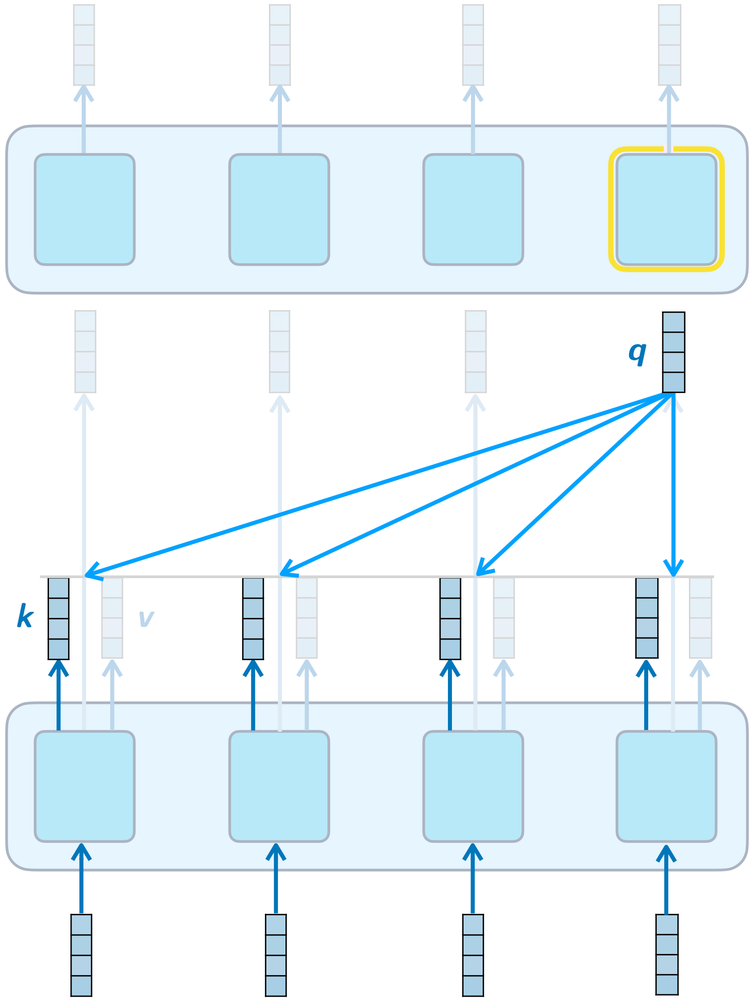
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

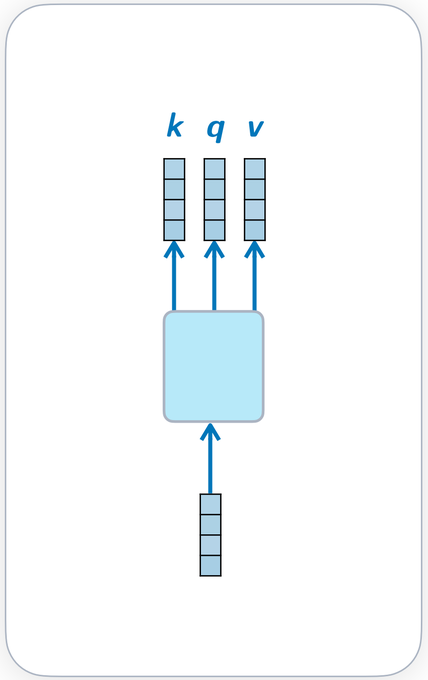
Causal Self-Attention

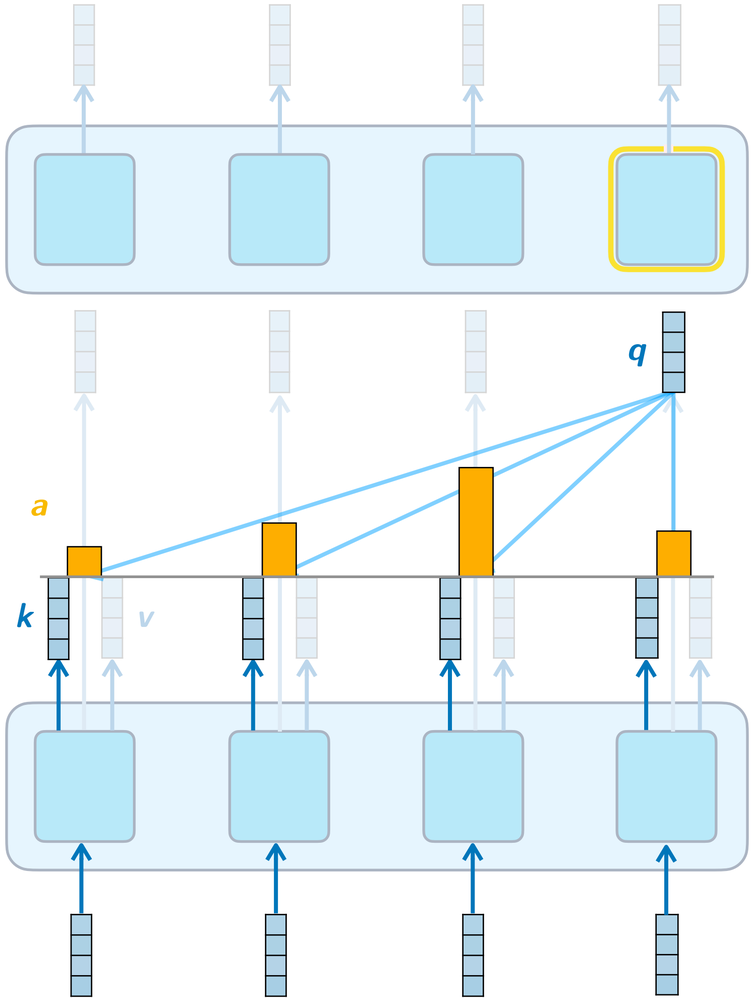
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

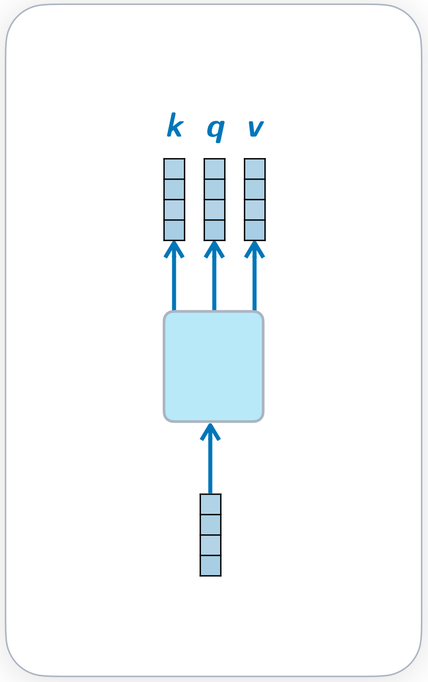
Causal Self-Attention

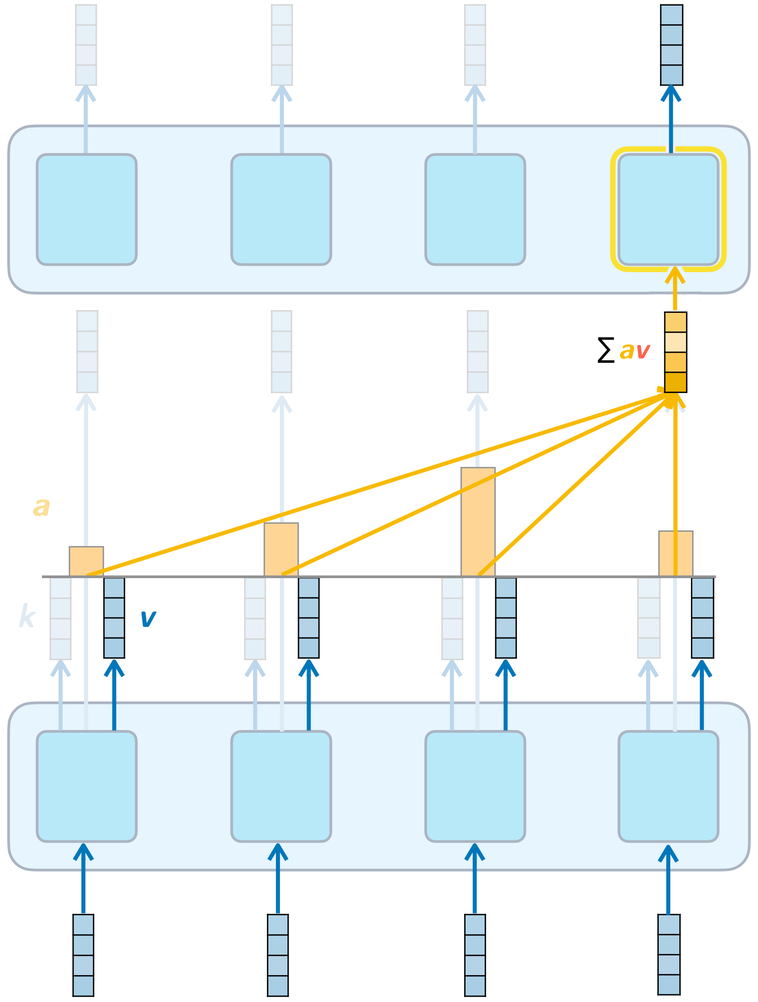
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

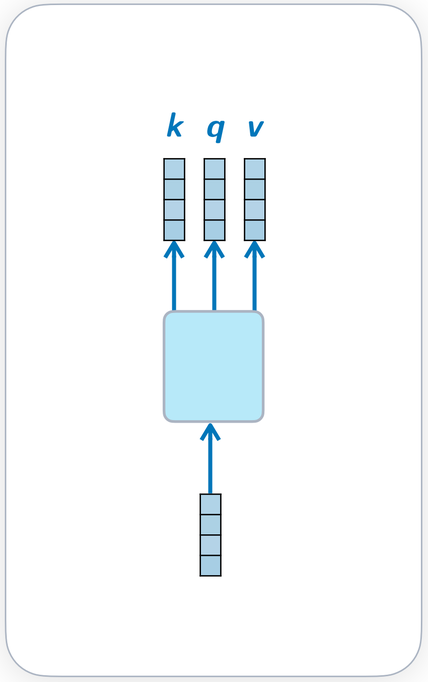
Causal Self-Attention

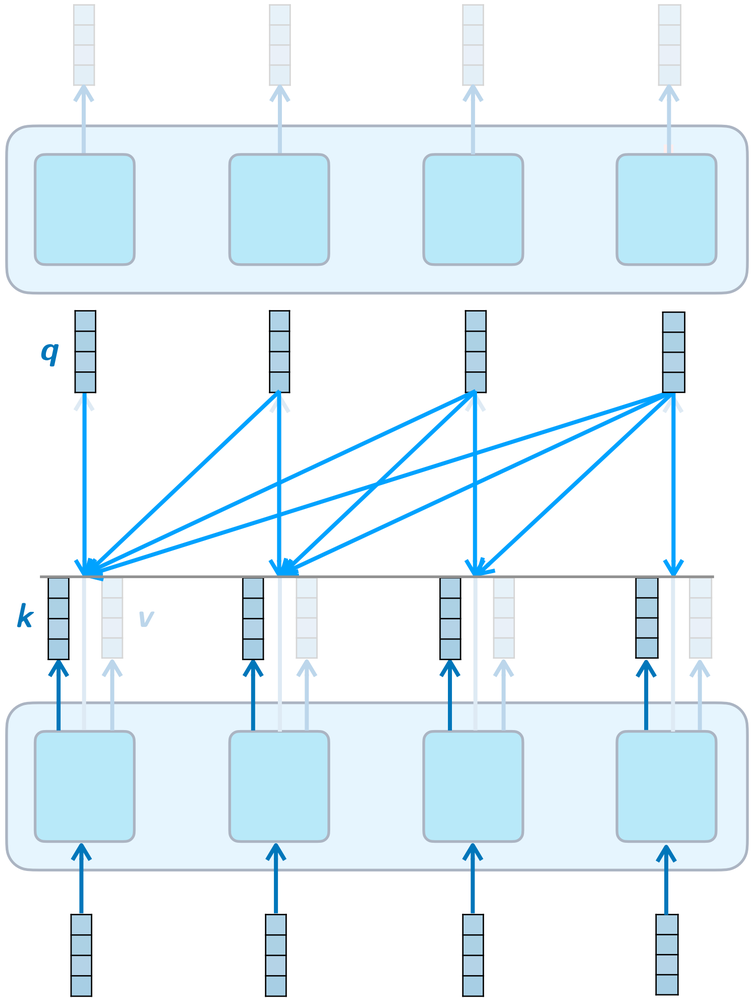
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

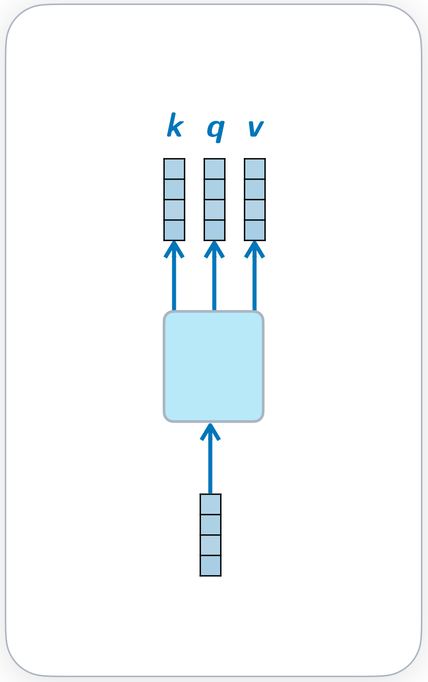
Causal Self-Attention

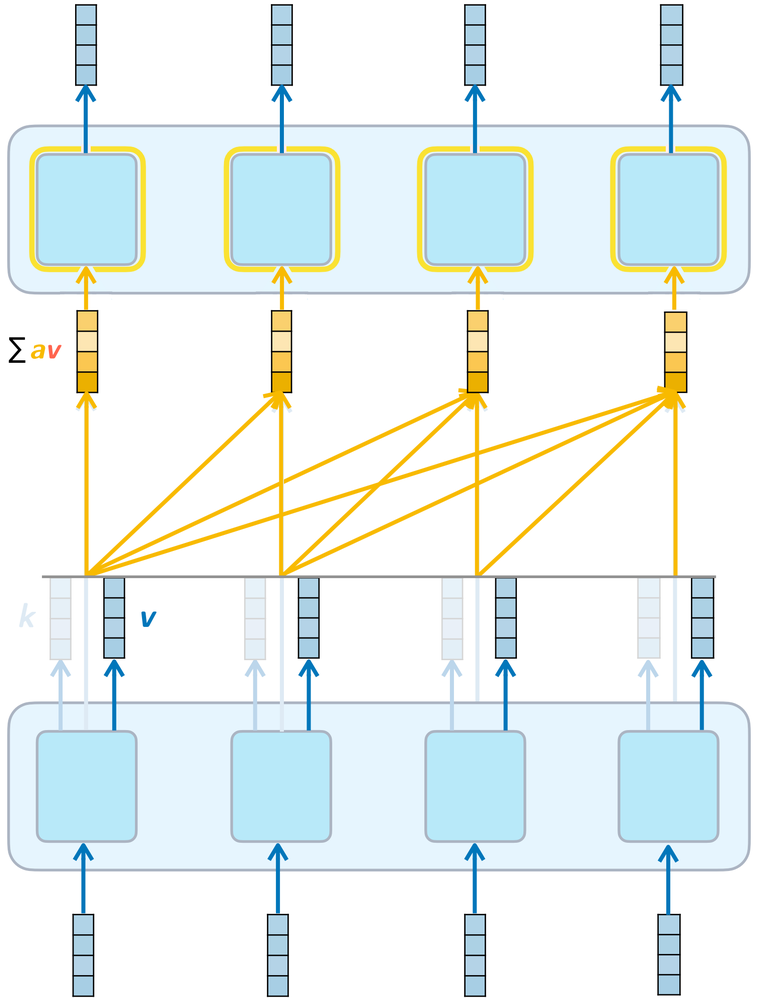
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**





Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

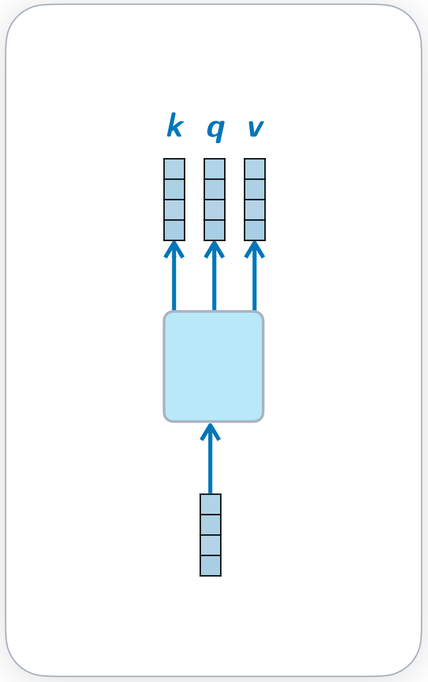
Causal Self-Attention

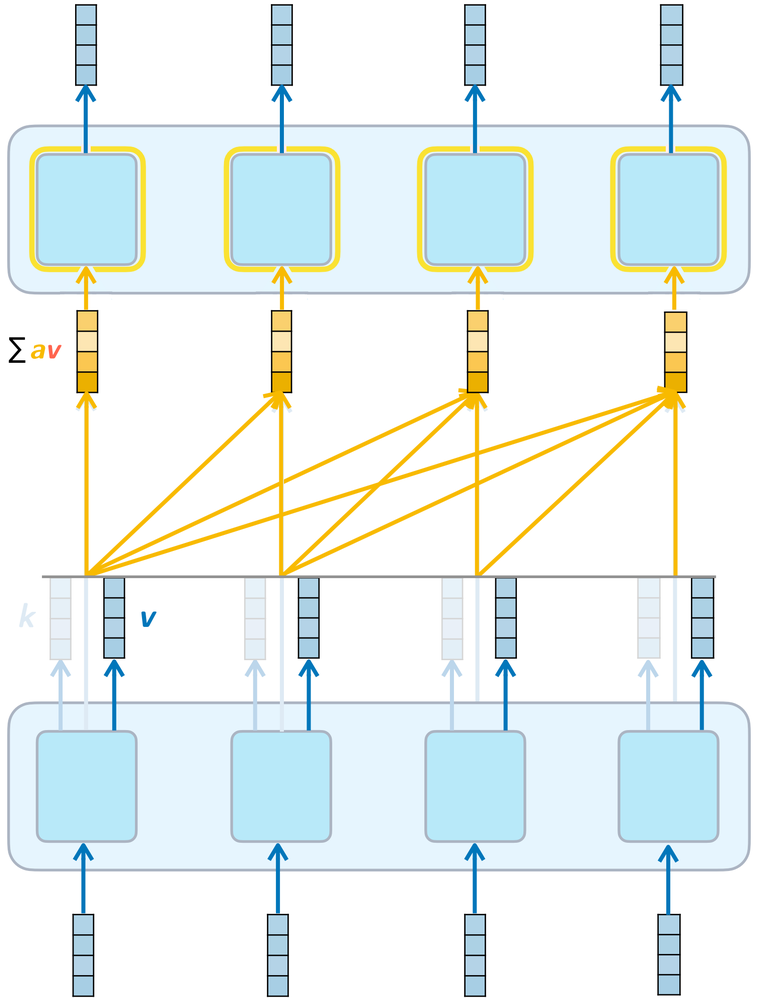
Think of ***N-to-N***

Feed **decoder** information into **decoder**

come up with a **query**
for *this* **decoder** time step

search for this **query**
in the same **decoder sequence**,
by comparing it to each **key**



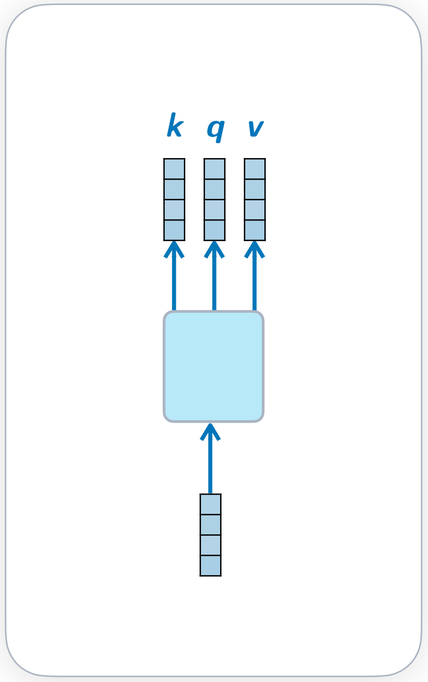


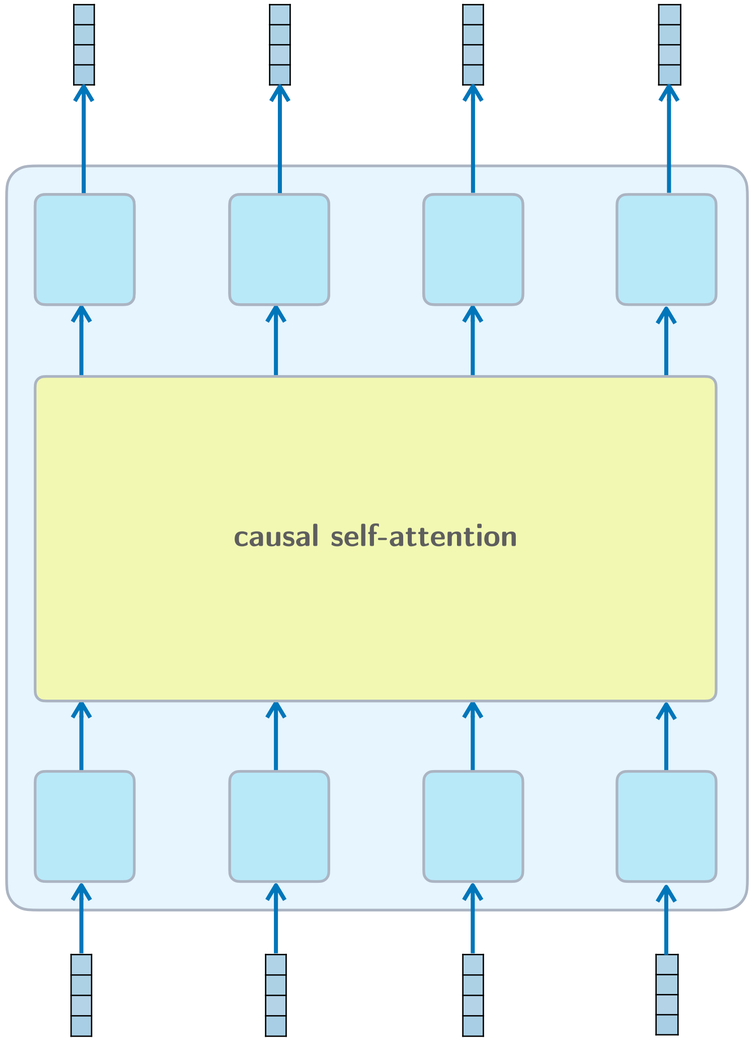
Each **decoder** token attends to all the *previous* **decoder** tokens

key and value vectors

Causal Self-Attention

Feed **decoder** information into **decoder**





refined **decoder sequence *X'***

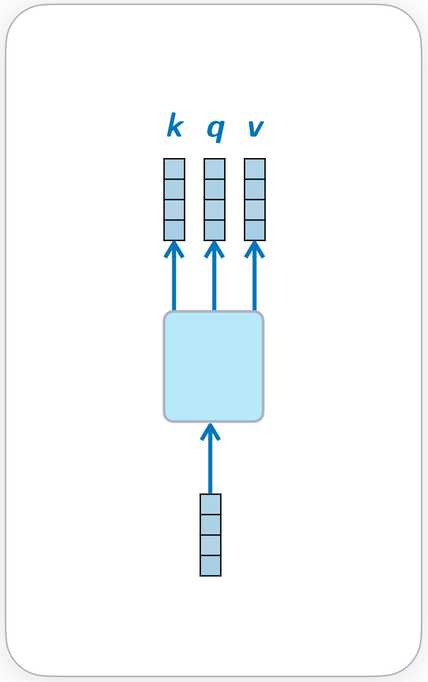
such that if you apply **unembed** you get **logits** over the vocabulary for next token

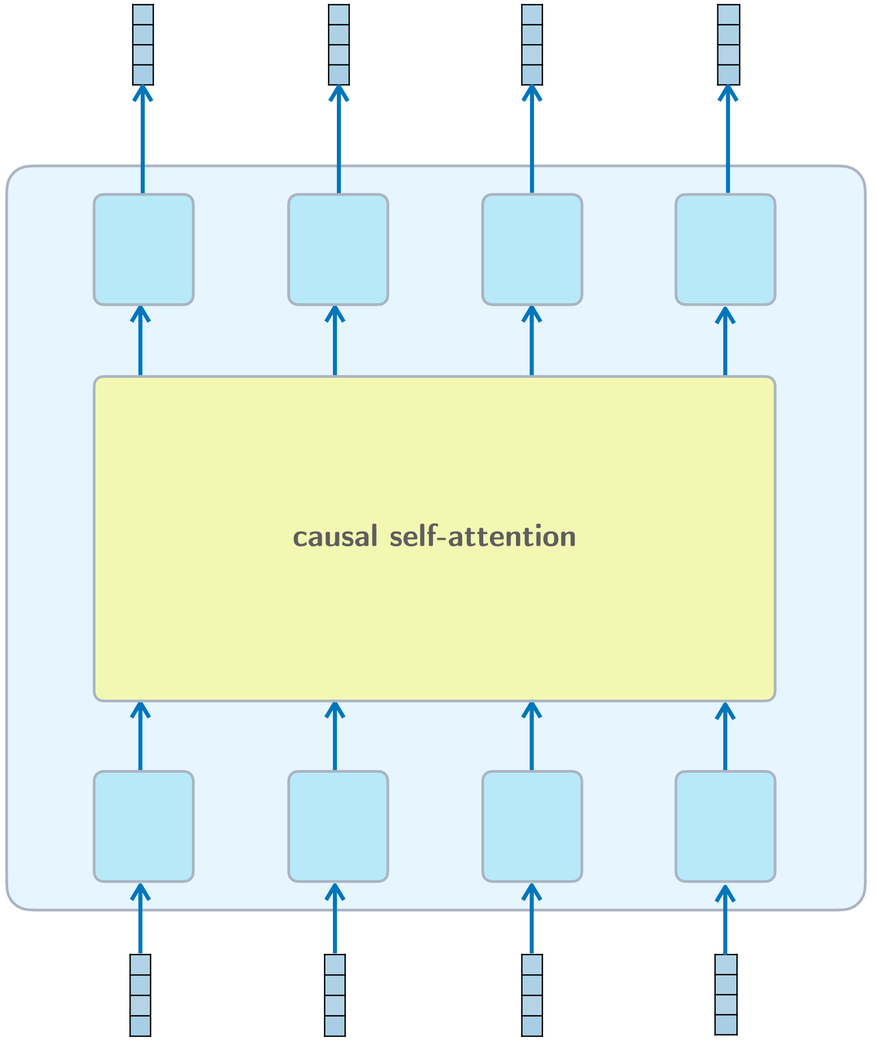
refines and ***contextualizes*** **decoder sequence**

decoder sequence *X*

Causal Self-Attention

Feed **decoder** information into **decoder**





refined **decoder sequence *X'***

such that if you apply **unembed** you get **logits** over the vocabulary for next token

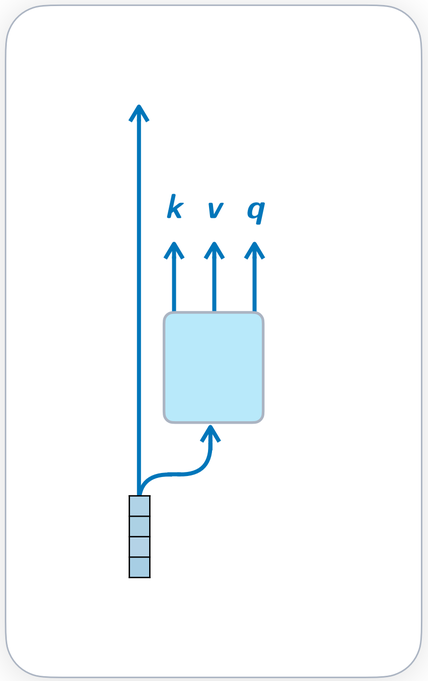
refines and ***contextualizes*** **decoder sequence**

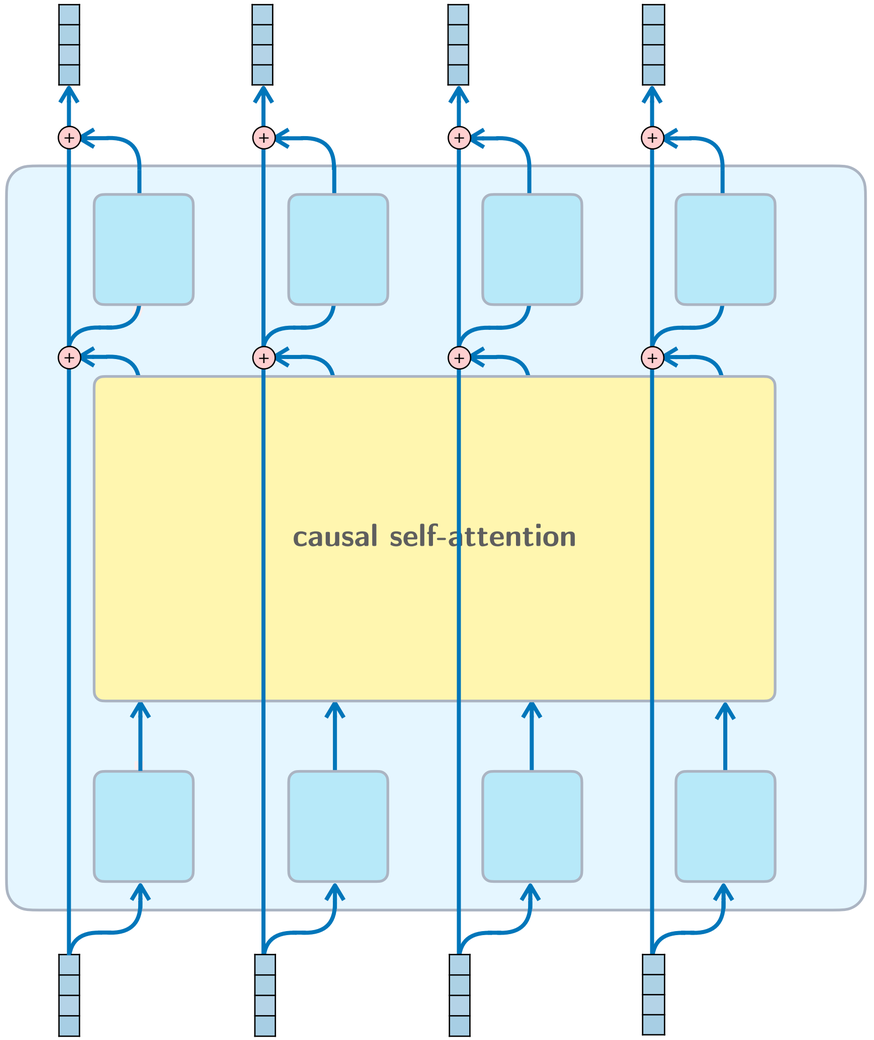
decoder sequence *X*

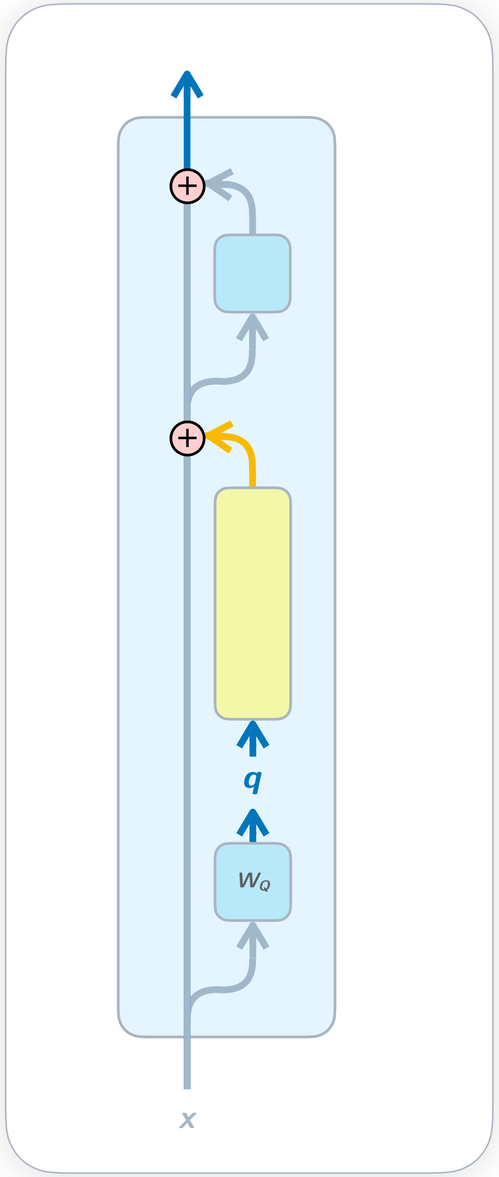
Causal Self-Attention

The ***residual stream***

Each token has its own ***residual stream***



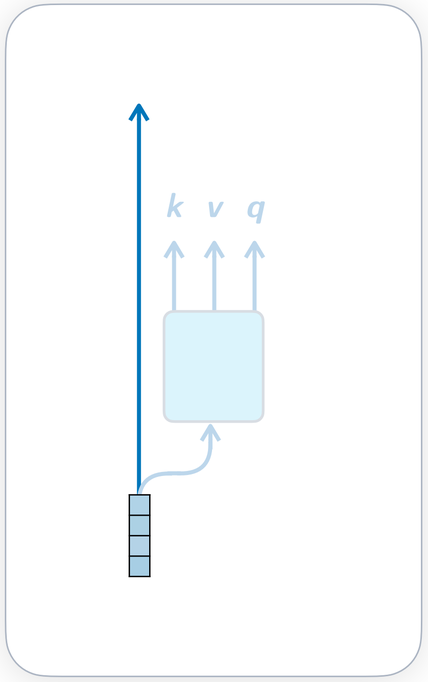


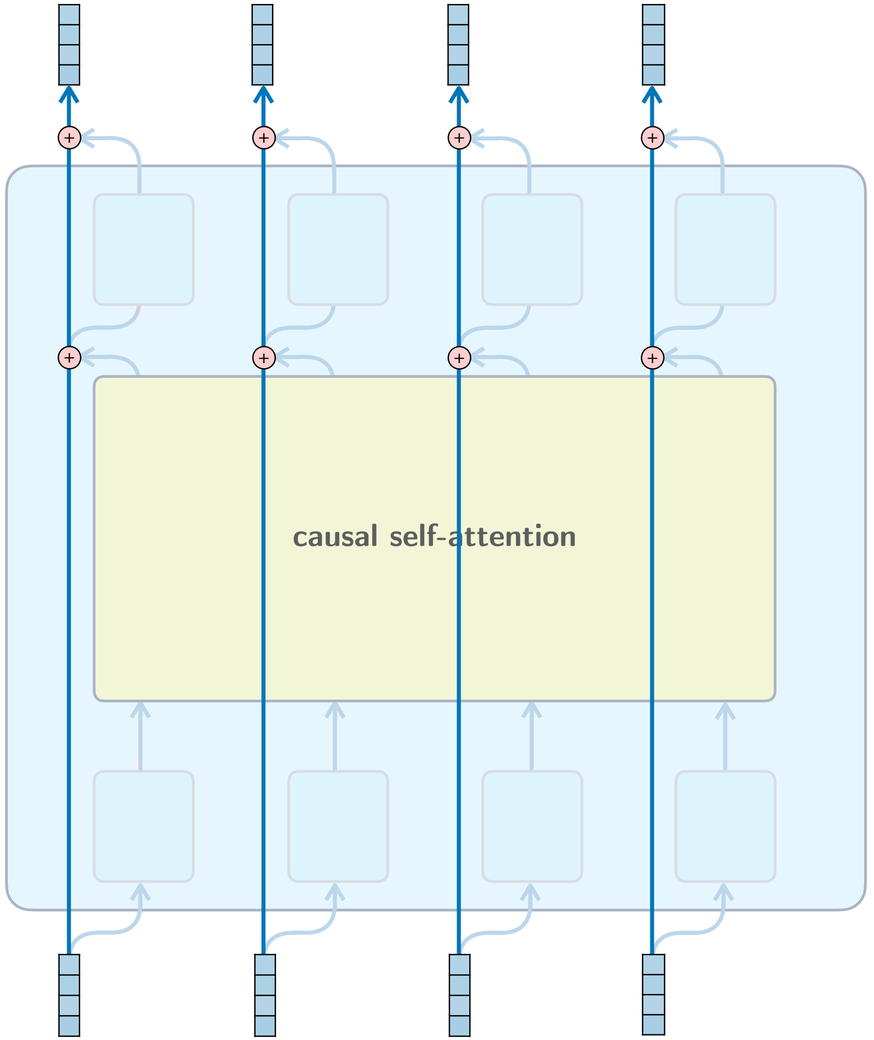


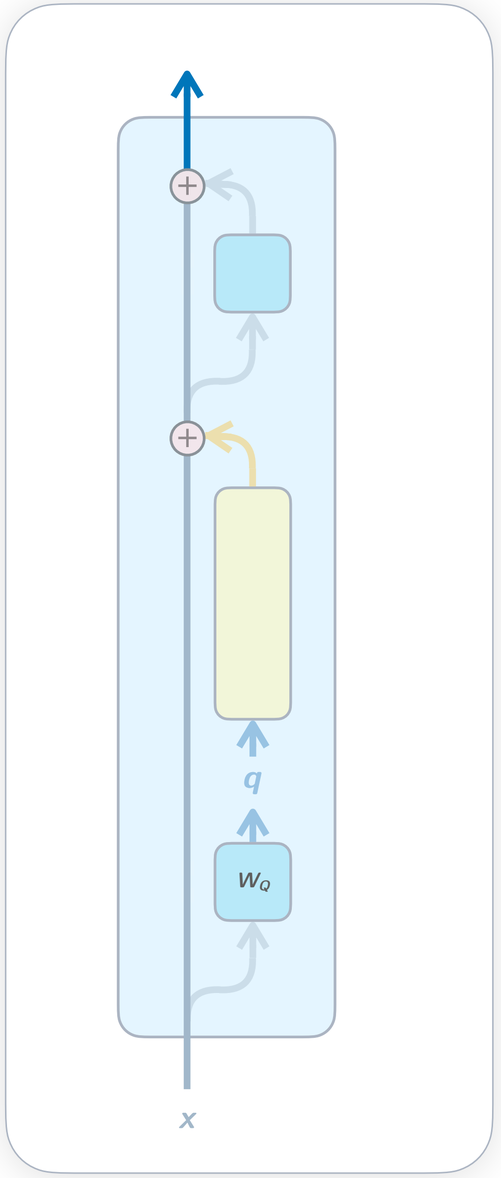
Causal Self-Attention

The ***residual stream***

Each token has its own ***residual stream***





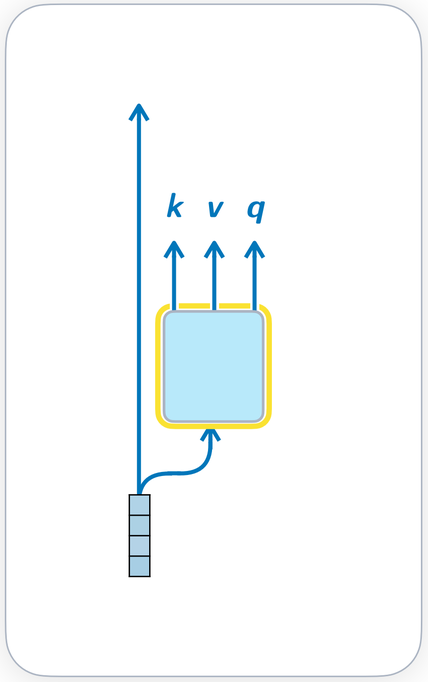


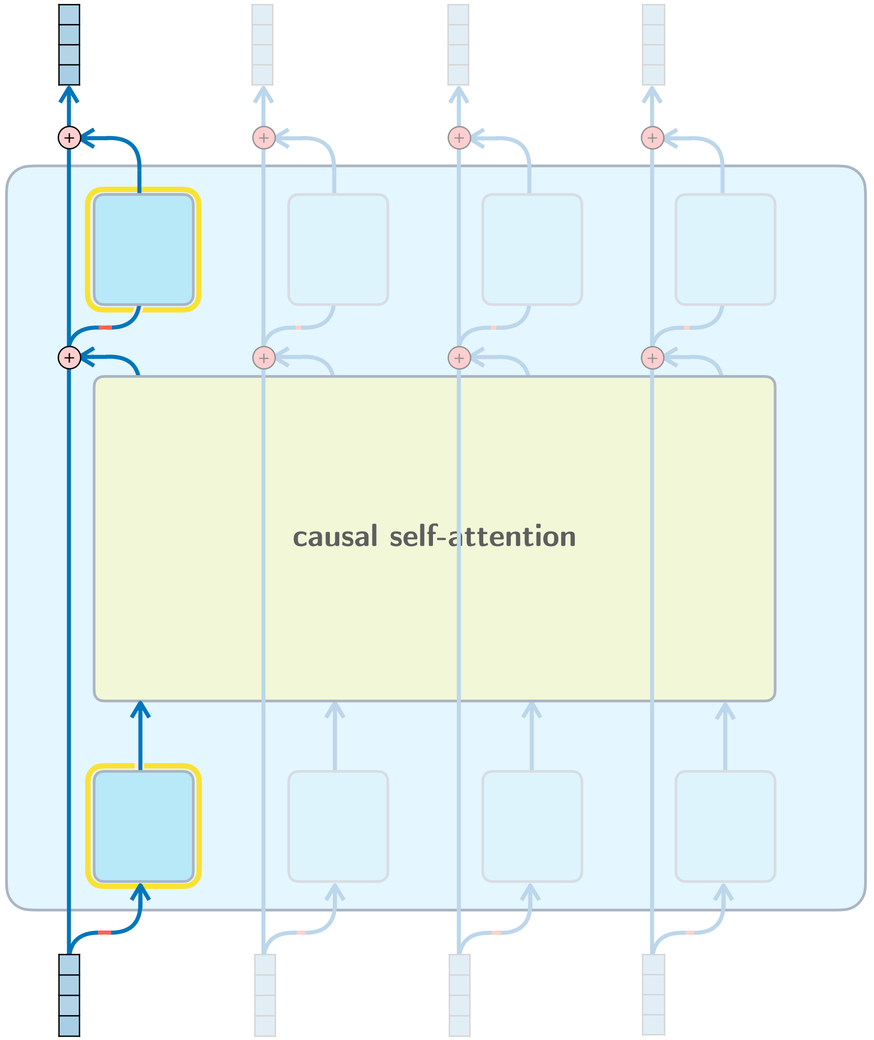
Self-Attention

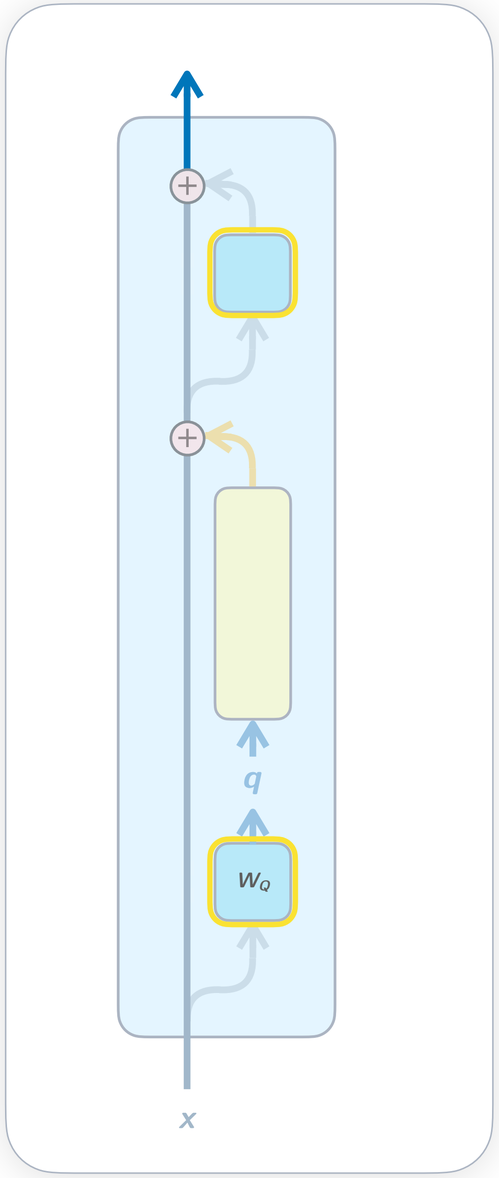
The ***residual stream***

Each token has its own ***residual stream***

"**Reading**" and "**writing**" to the residual stream





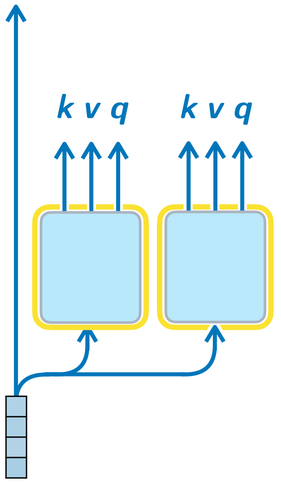


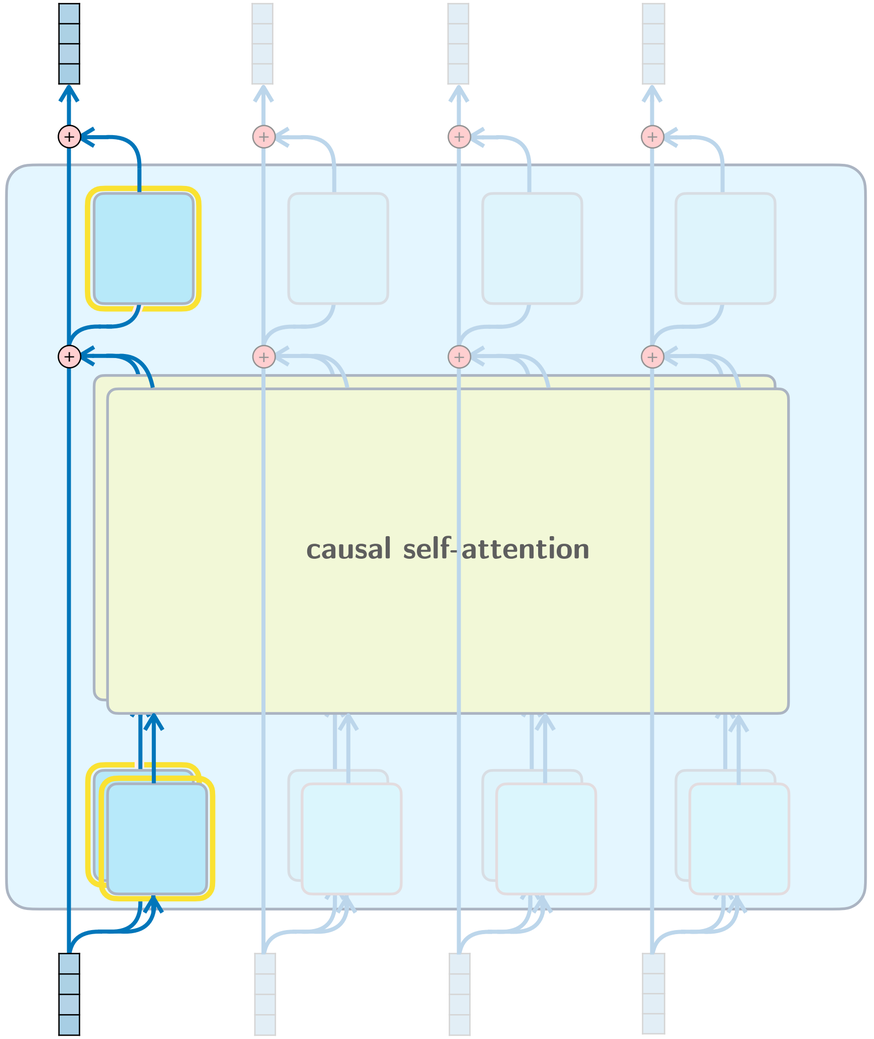
Self-Attention

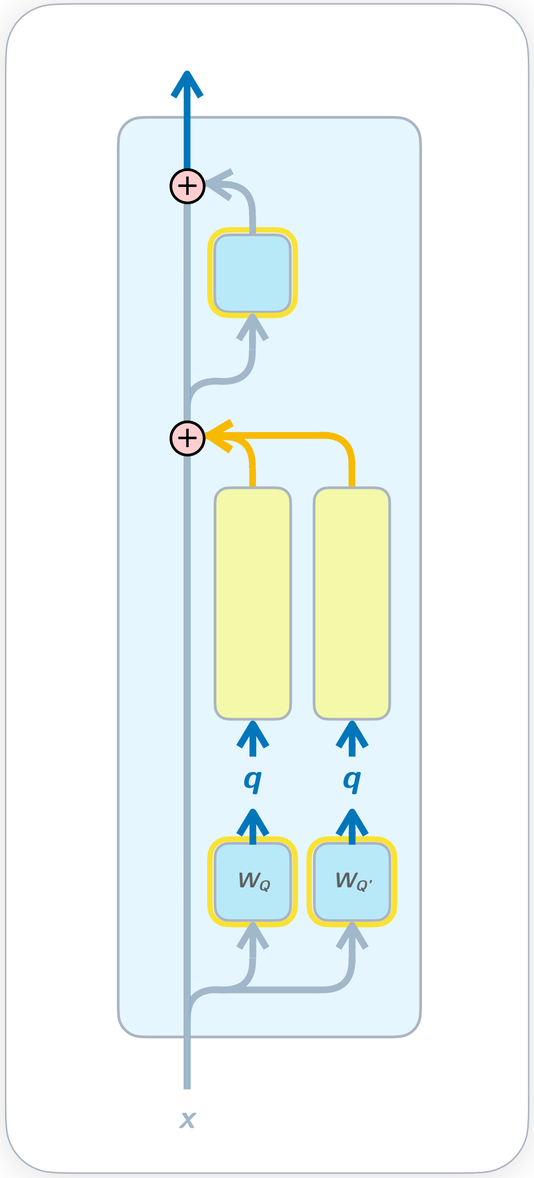
The ***residual stream***

Each token has its own ***residual stream***

"**Reading**" and "**writing**" to the residual stream

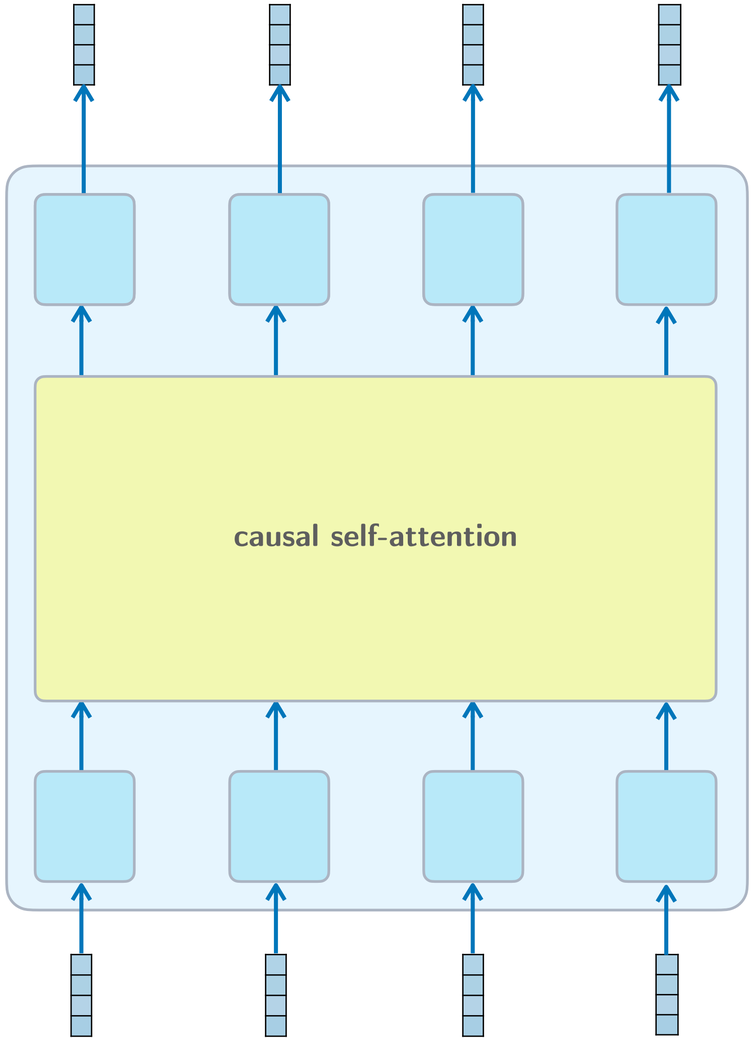






Causal Self-Attention

Feed **decoder** information into **decoder**



refined **decoder sequence *X'***

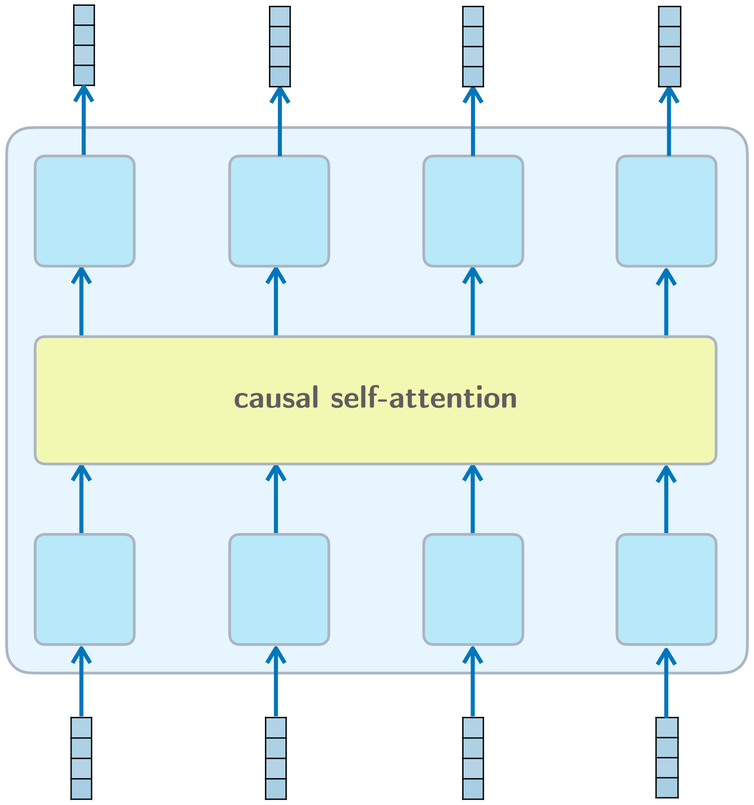
such that if you apply **unembed** you get **logits** over the vocabulary for next token

refines and ***contextualizes*** **decoder sequence**

decoder sequence *X*

Causal Self-Attention

Feed **decoder** information into **decoder**



refined **decoder sequence *X'***

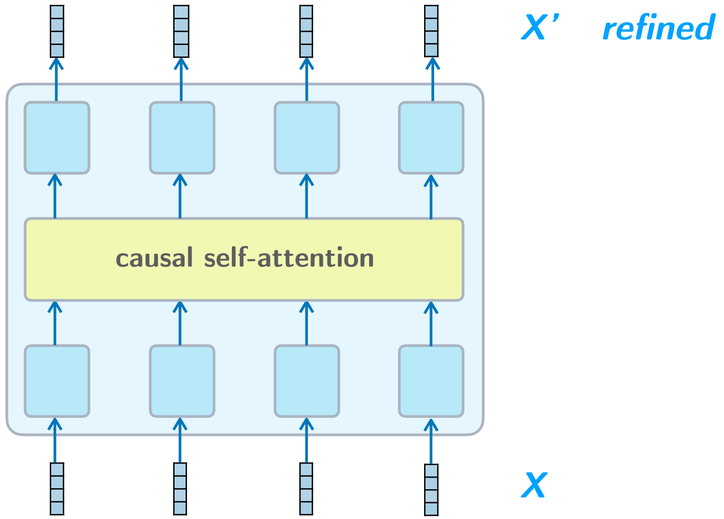
such that if you apply **unembed** you get **logits** over the vocabulary for next token

refines and ***contextualizes*** **decoder sequence**

decoder sequence *X*

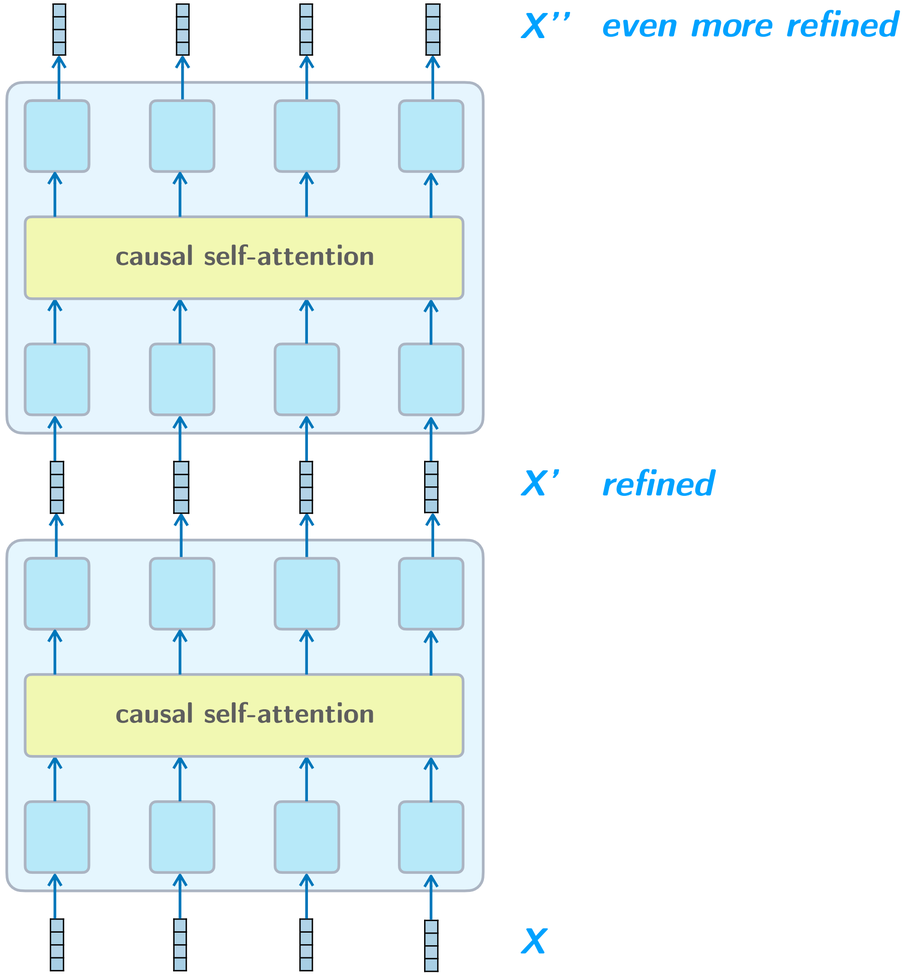
Causal Self-Attention

Feed **decoder** information into **decoder**



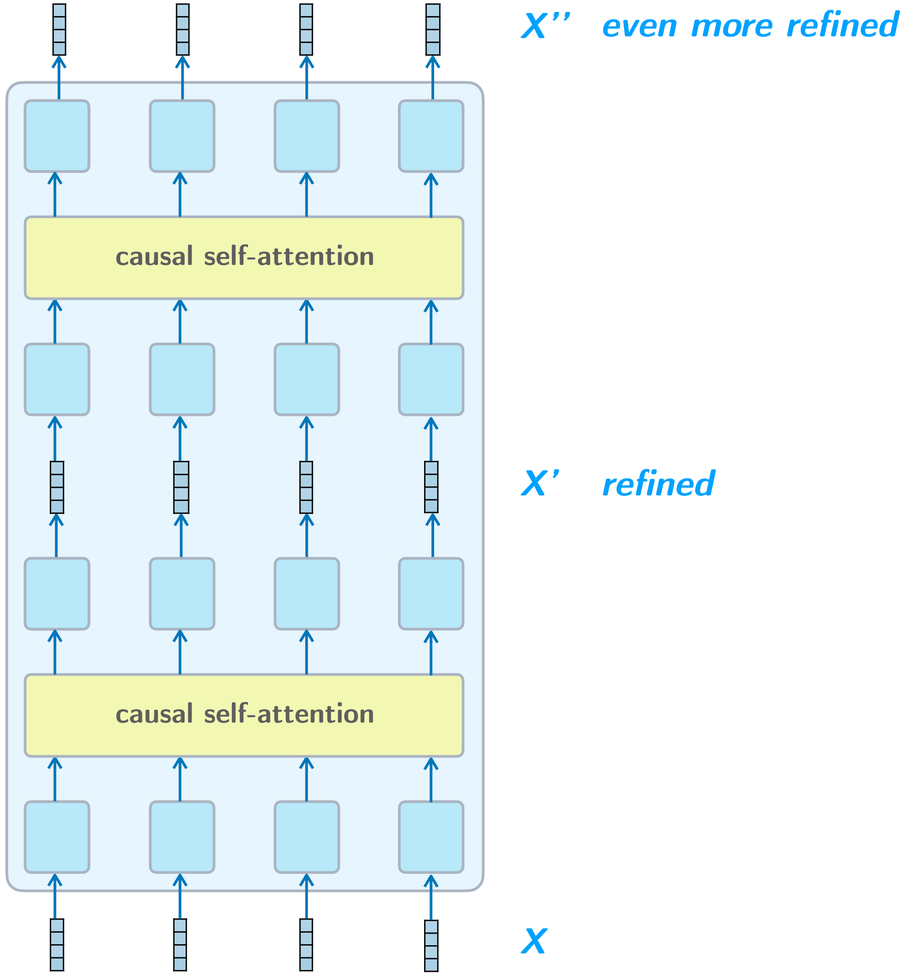
Causal Self-Attention

Feed **decoder** information into **decoder**



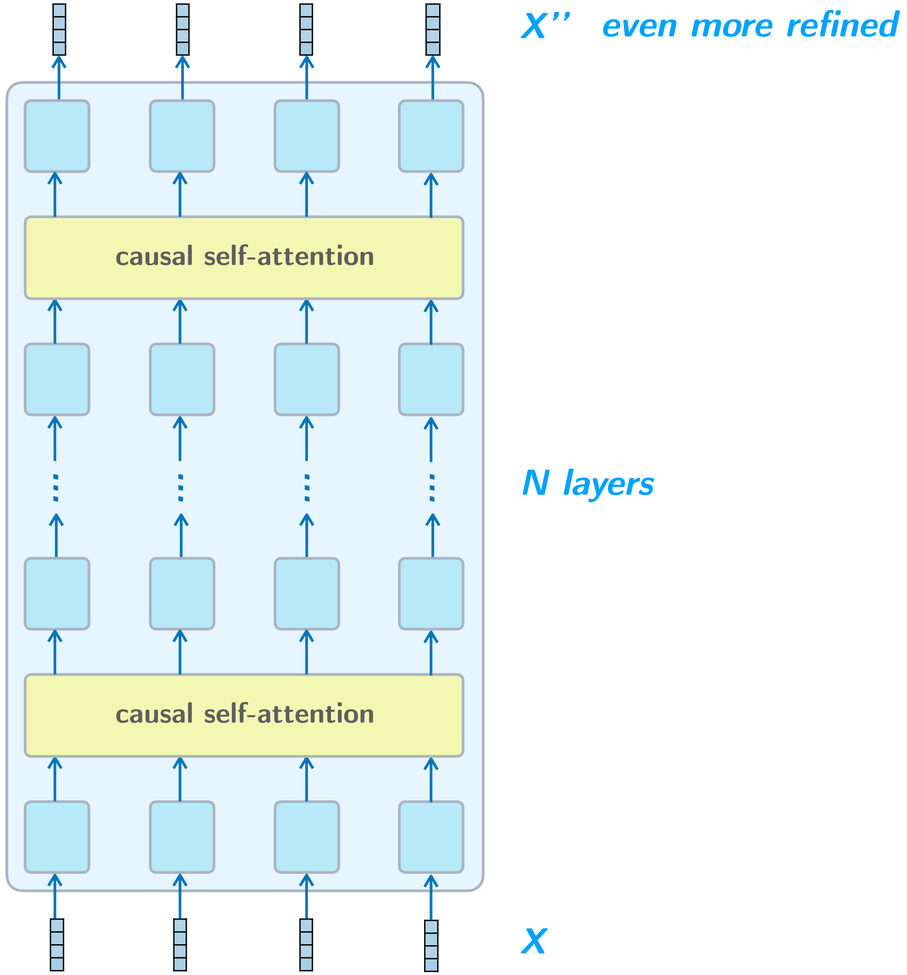
Causal Self-Attention

Feed **decoder** information into **decoder**

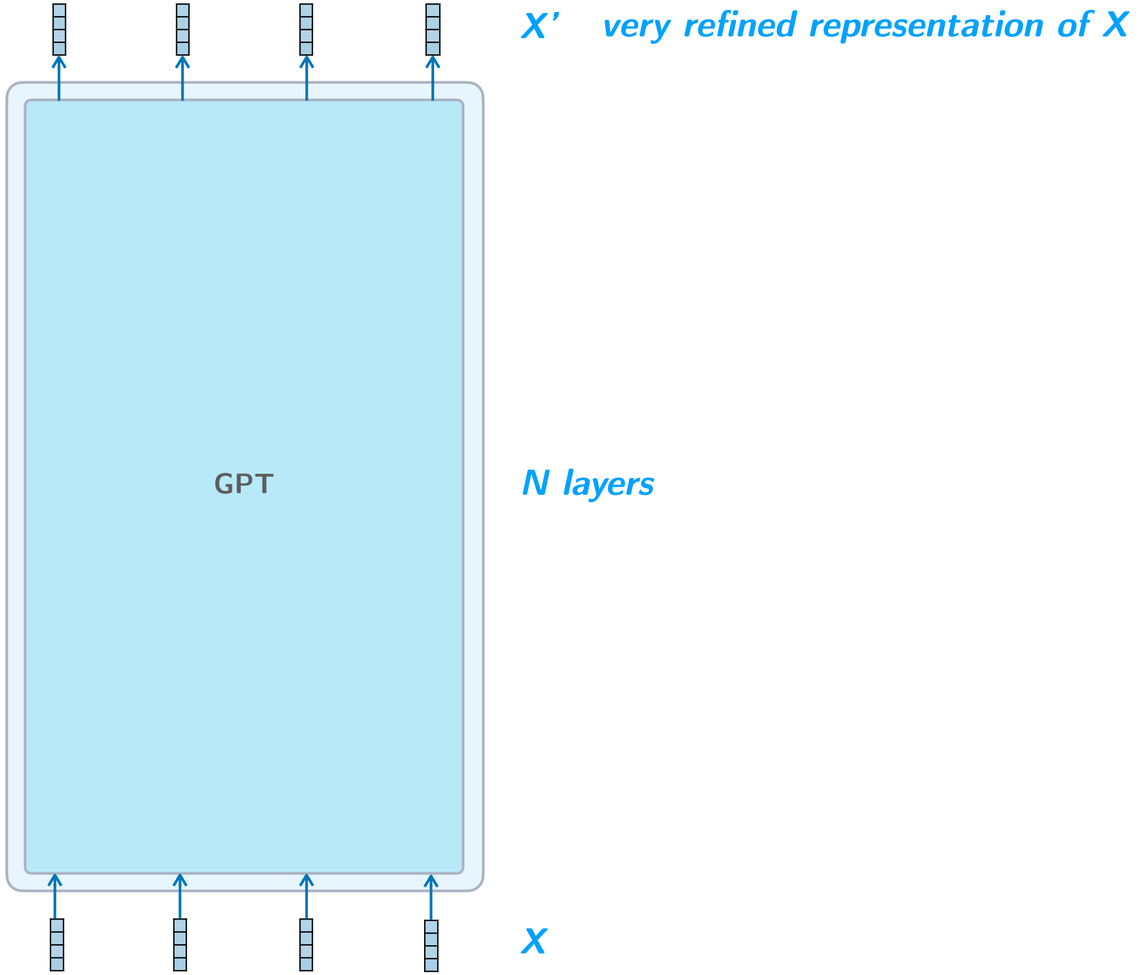


Causal Self-Attention

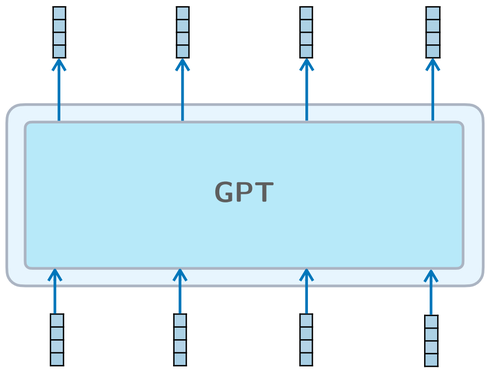
Feed **decoder** information into **decoder**



GPT = Generative Pre-trained Transformer



GPT = Generative Pre-trained Transformer



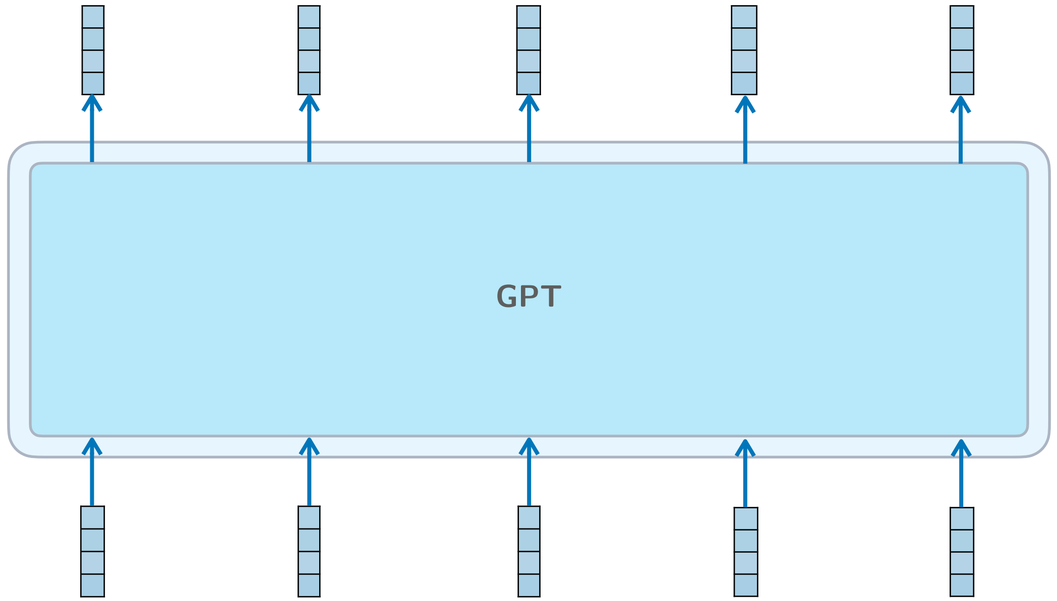
X' ***very refined representation of X***

X

GPT

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**

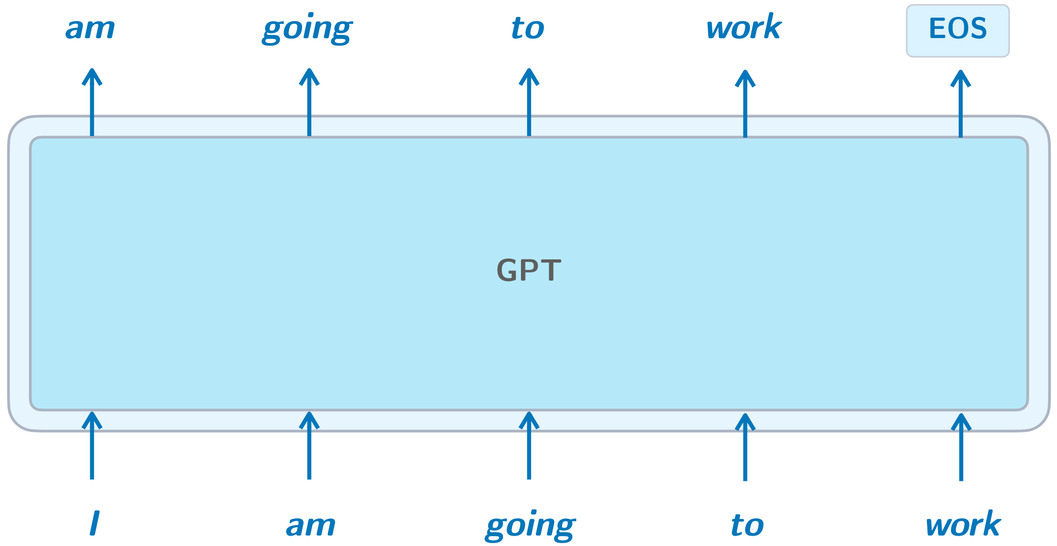


Based on "The Illustrated Transformer" by Jay Alammar

GPT

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**



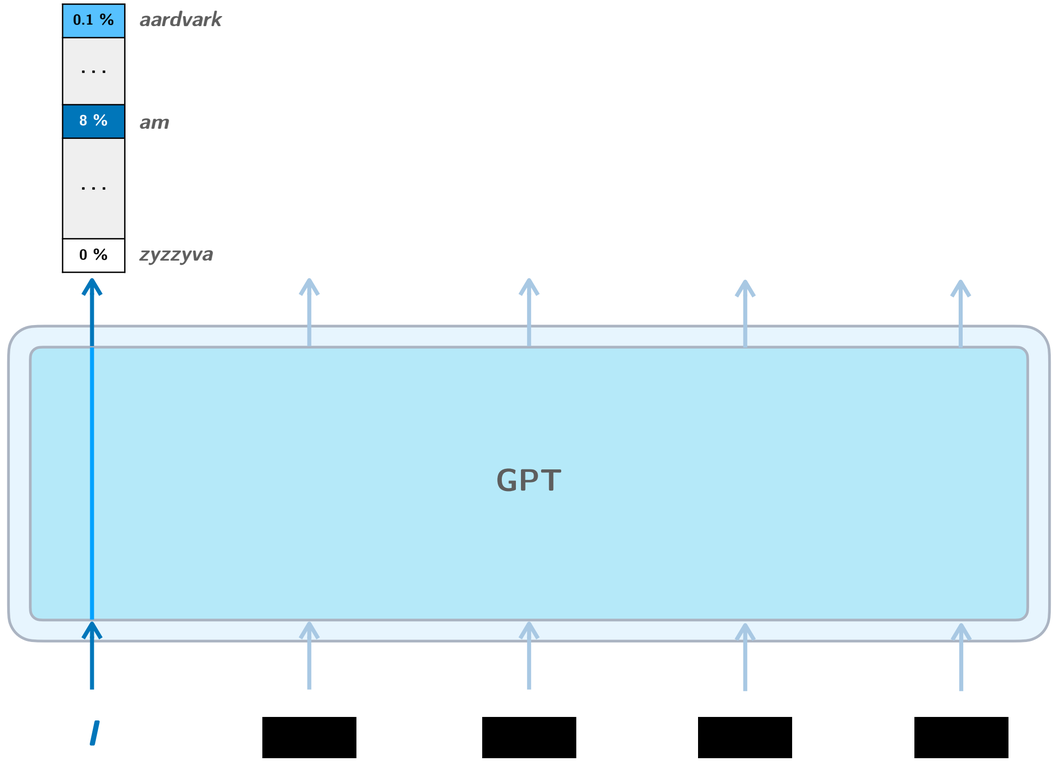
Based on "The Illustrated Transformer" by Jay Alammar

GPT

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**

Turn **refined representation** into prediction of the next token



We know the correct answer!

am

Mask future tokens

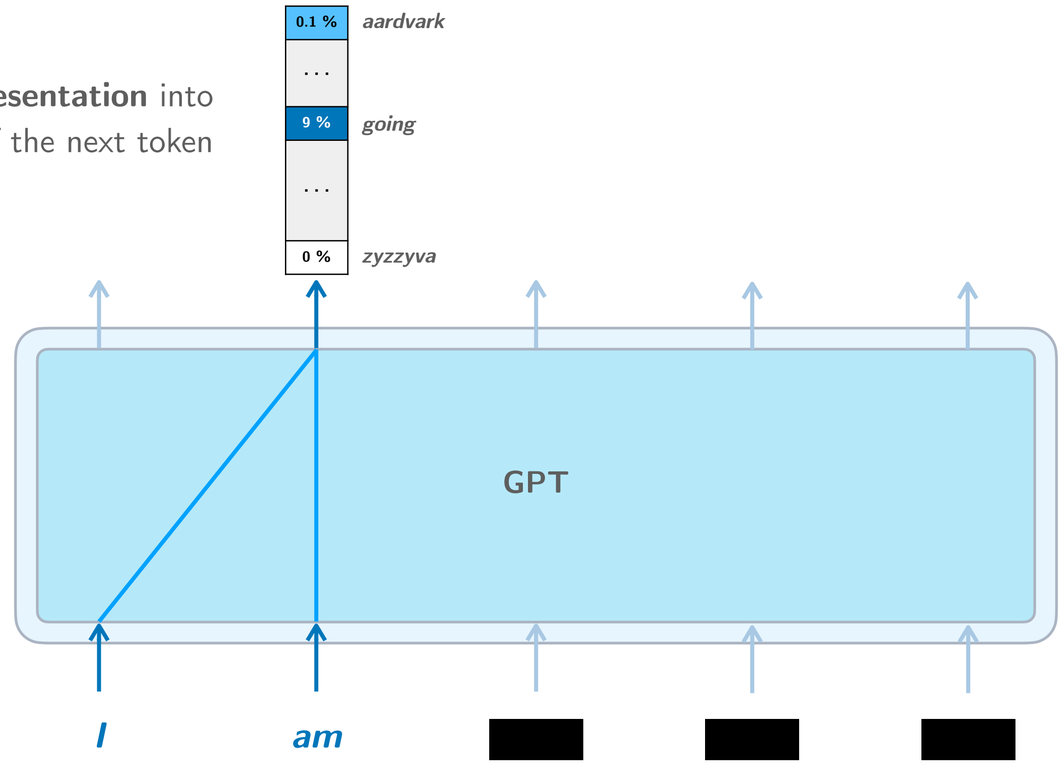
Based on "The Illustrated Transformer" by Jay Alammar

GPT

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**

Turn **refined representation** into prediction of the next token



We know the correct answer!

going

Mask future tokens

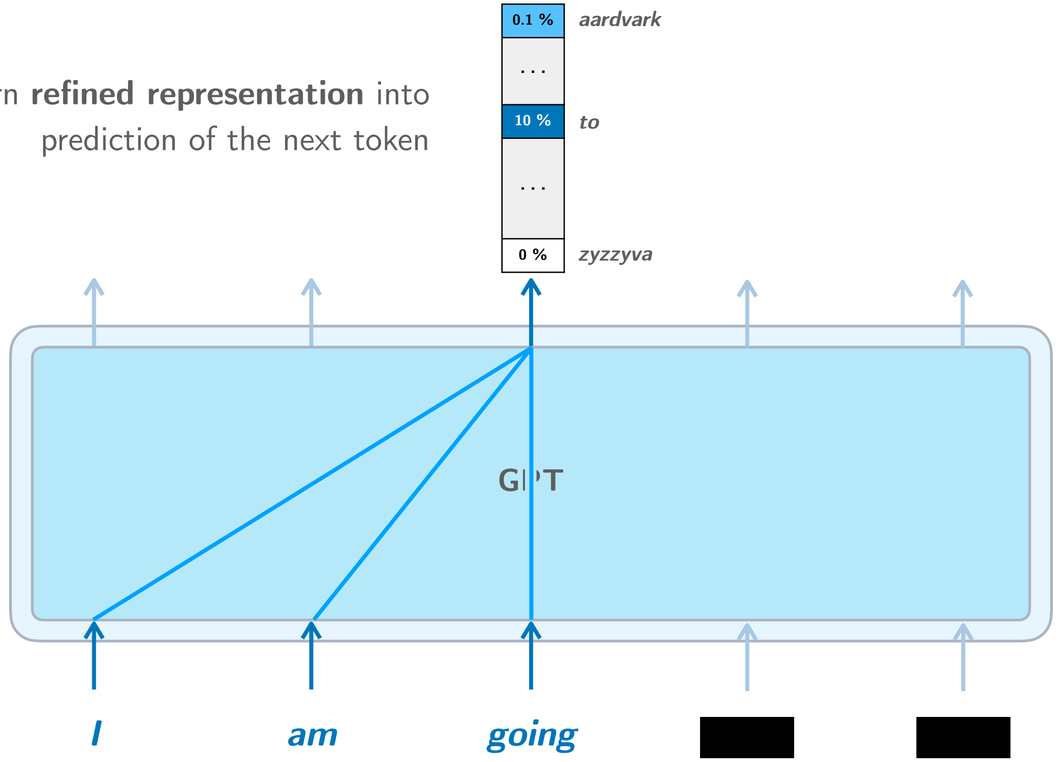
output so far

Based on "The Illustrated Transformer" by Jay Alammar

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**

Turn **refined representation** into prediction of the next token



We know the correct answer!

to

Mask future tokens

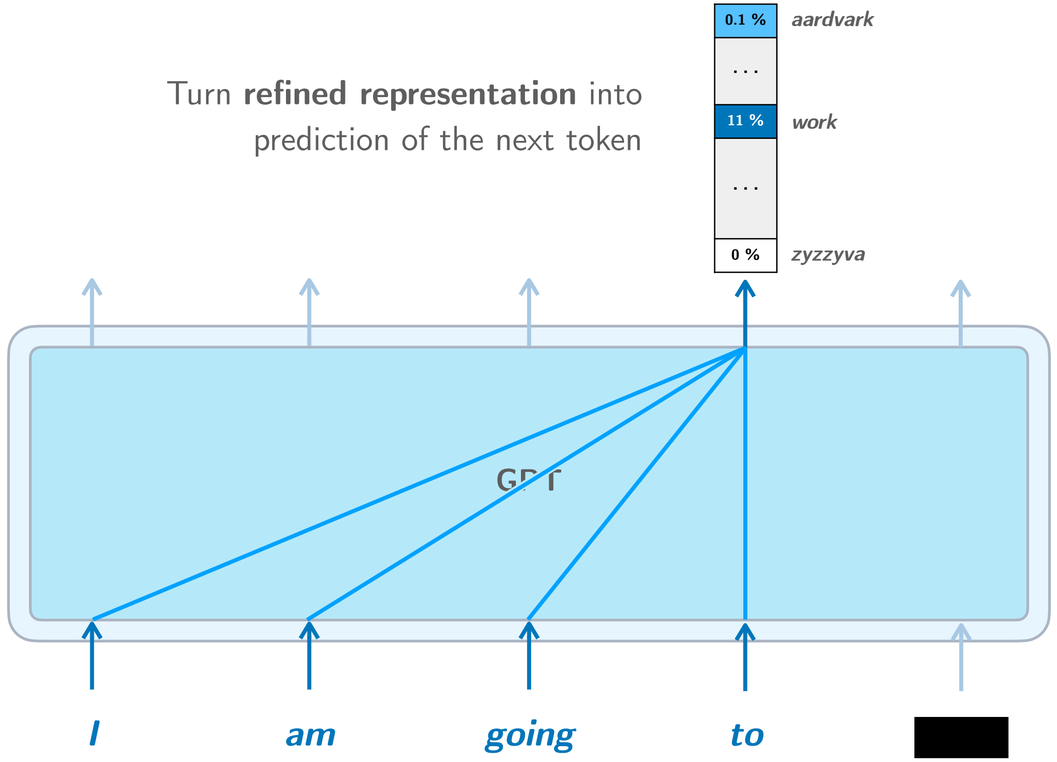
output so far

Based on "The Illustrated Transformer" by Jay Alammar

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**

Turn **refined representation** into prediction of the next token



output so far

We know the correct answer!

work

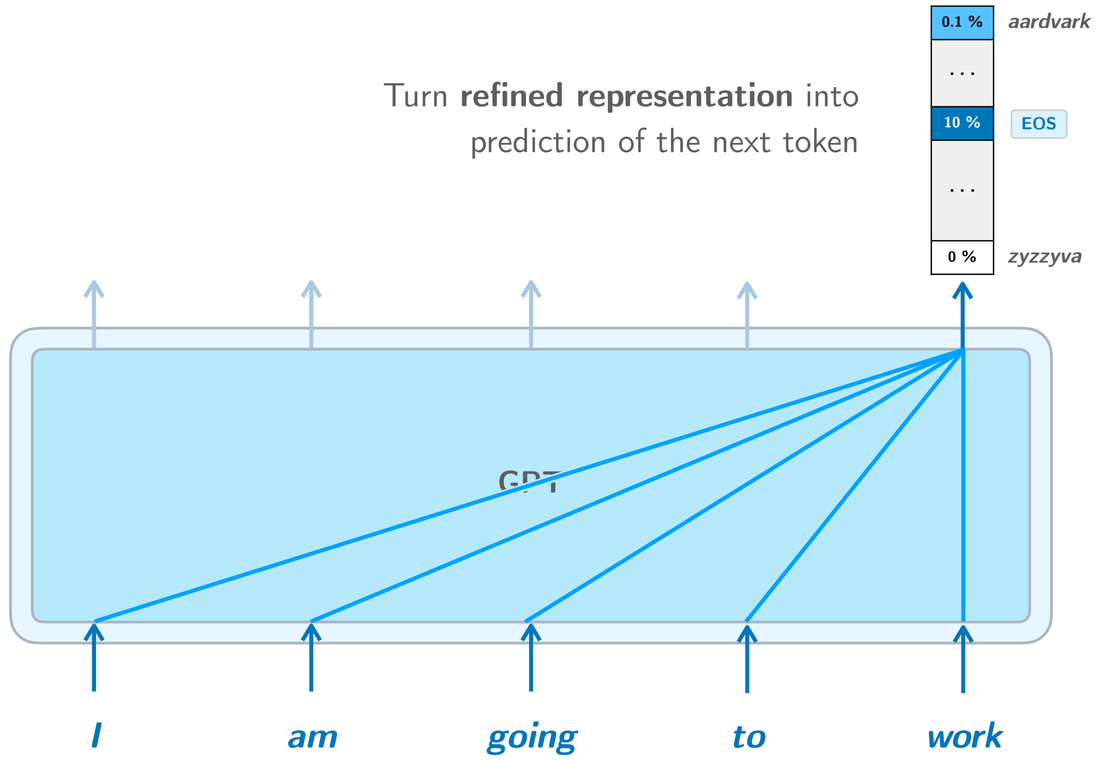
Mask future tokens

Based on "The Illustrated Transformer" by Jay Alammar

GPT

GPT = Generative Pre-trained Transformer

GPT is trained using **language modeling (LM)** aka **next-token prediction**



We know the correct answer!

EOS

Mask future tokens

There's also a SOS (start of sentence) token, but we'll ignore that.

Based on "The Illustrated Transformer" by Jay Alammar

3

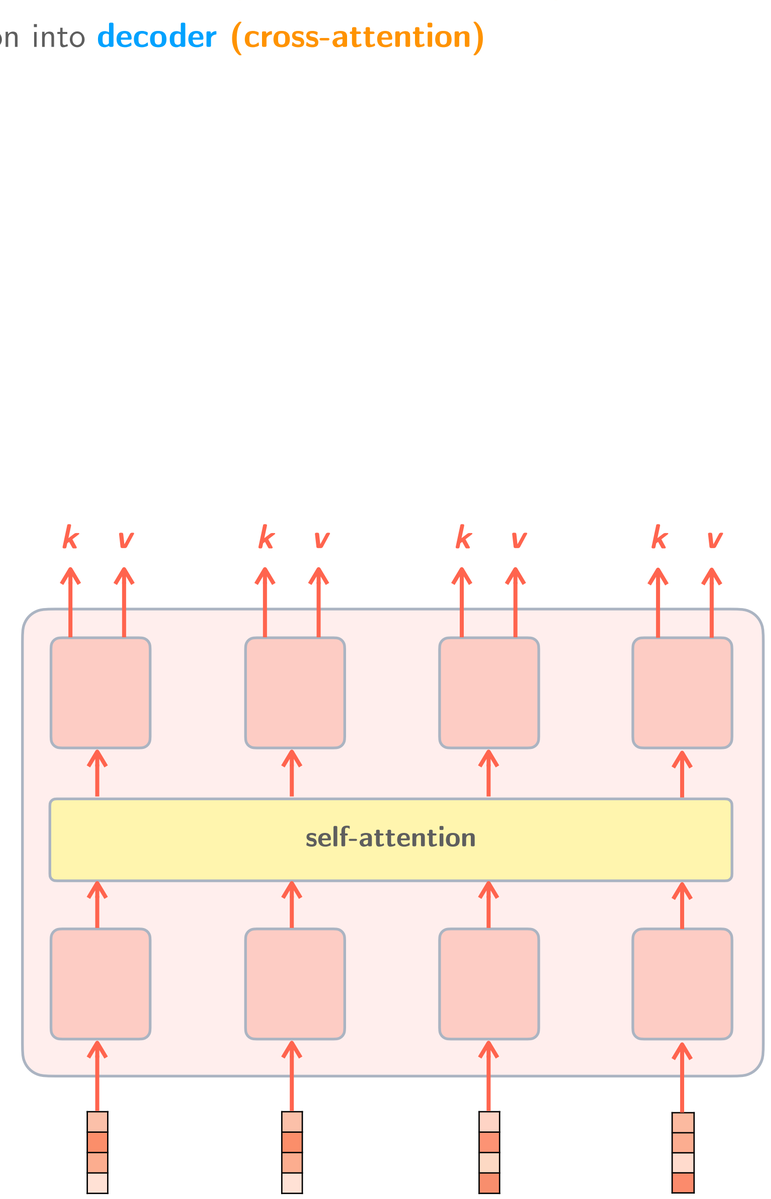
Transformer Encoder–Decoder

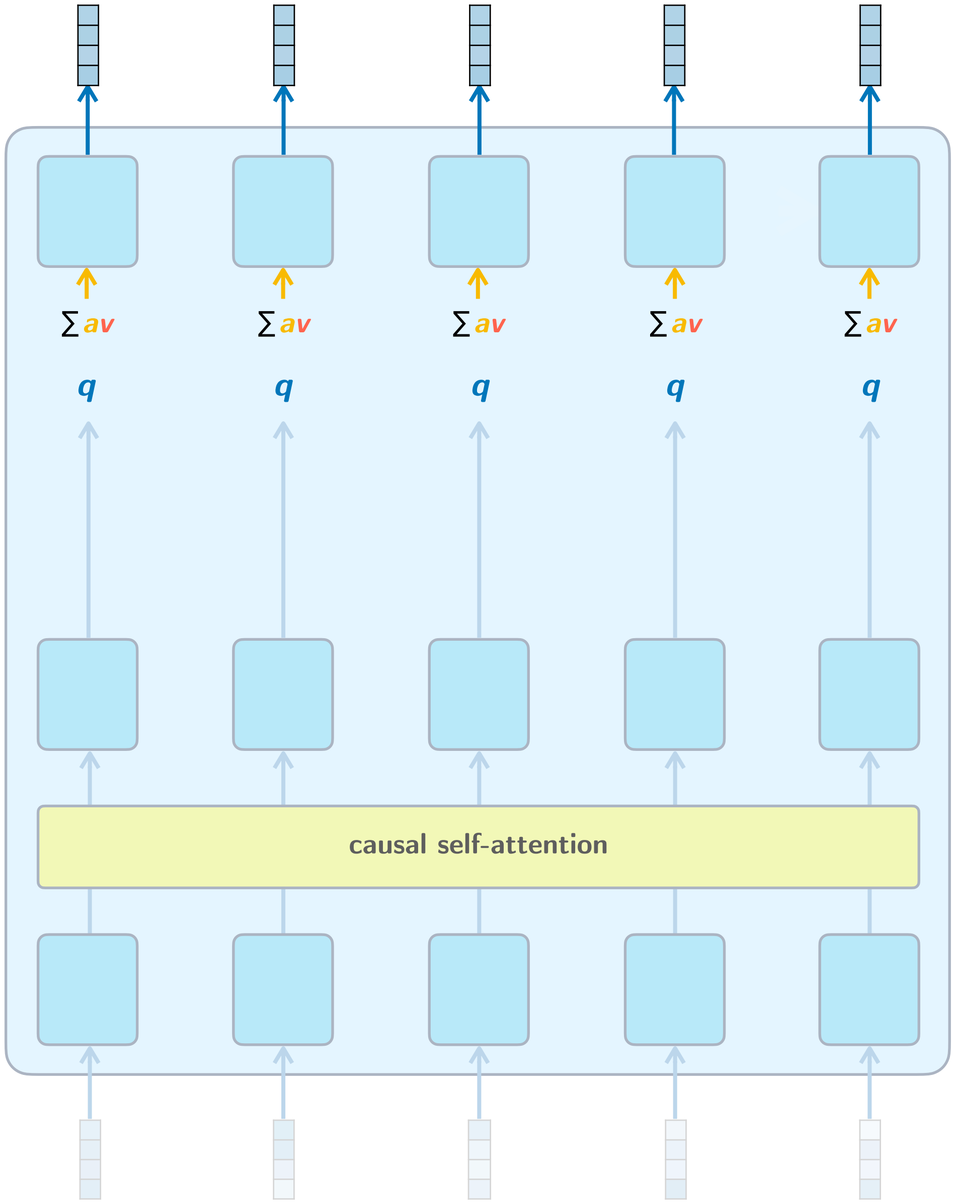
(The original Transformer)

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

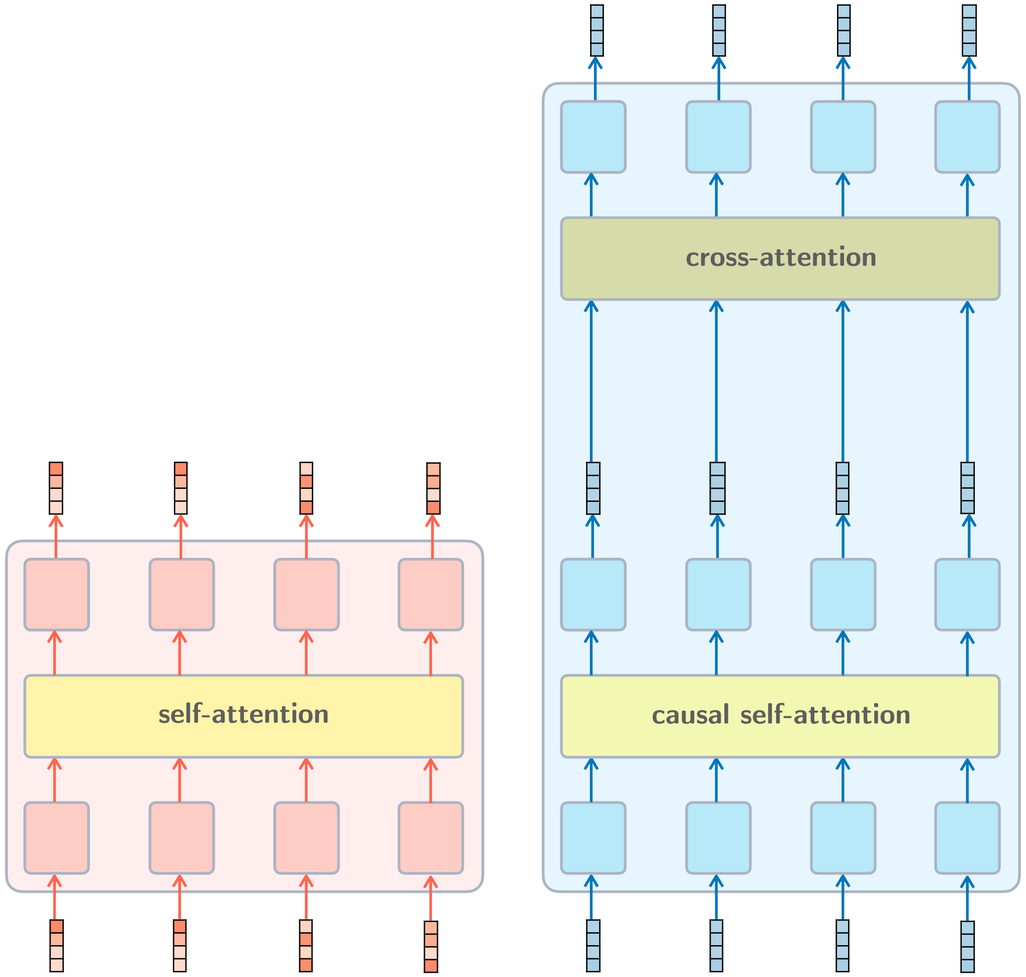




Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

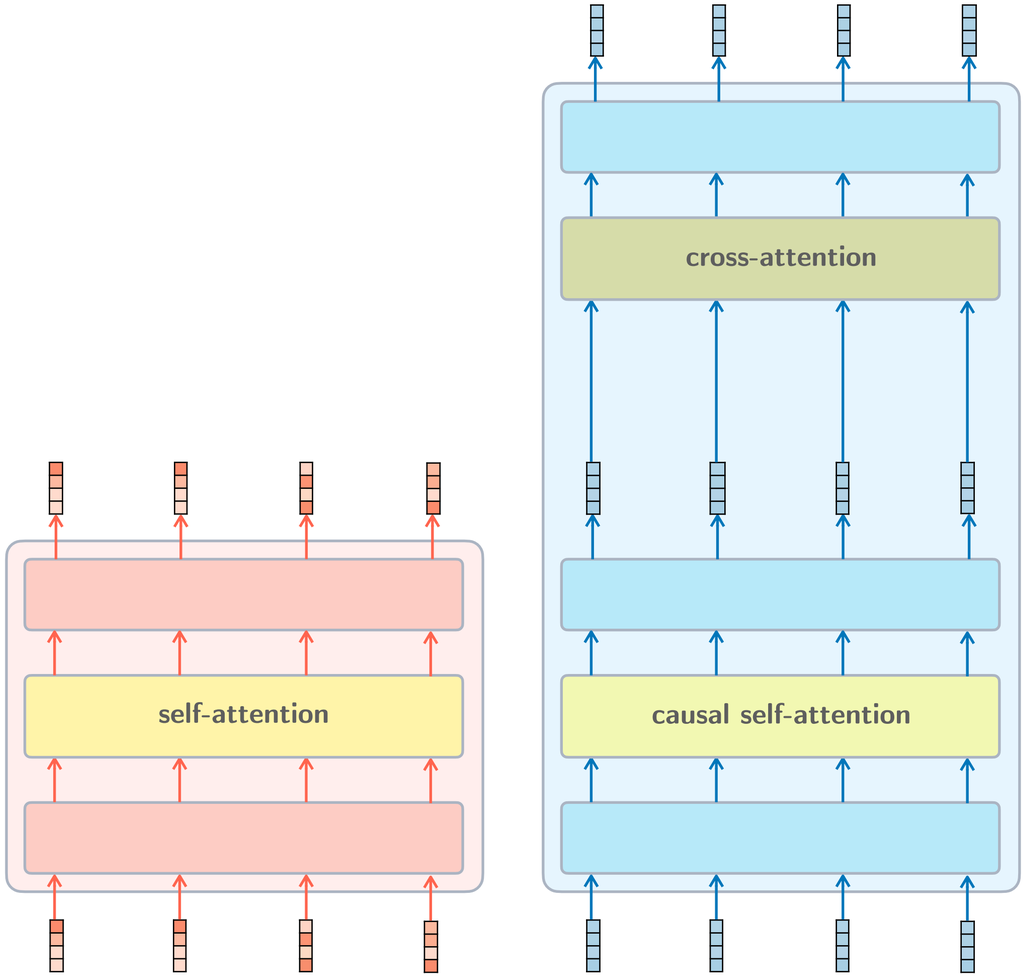
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

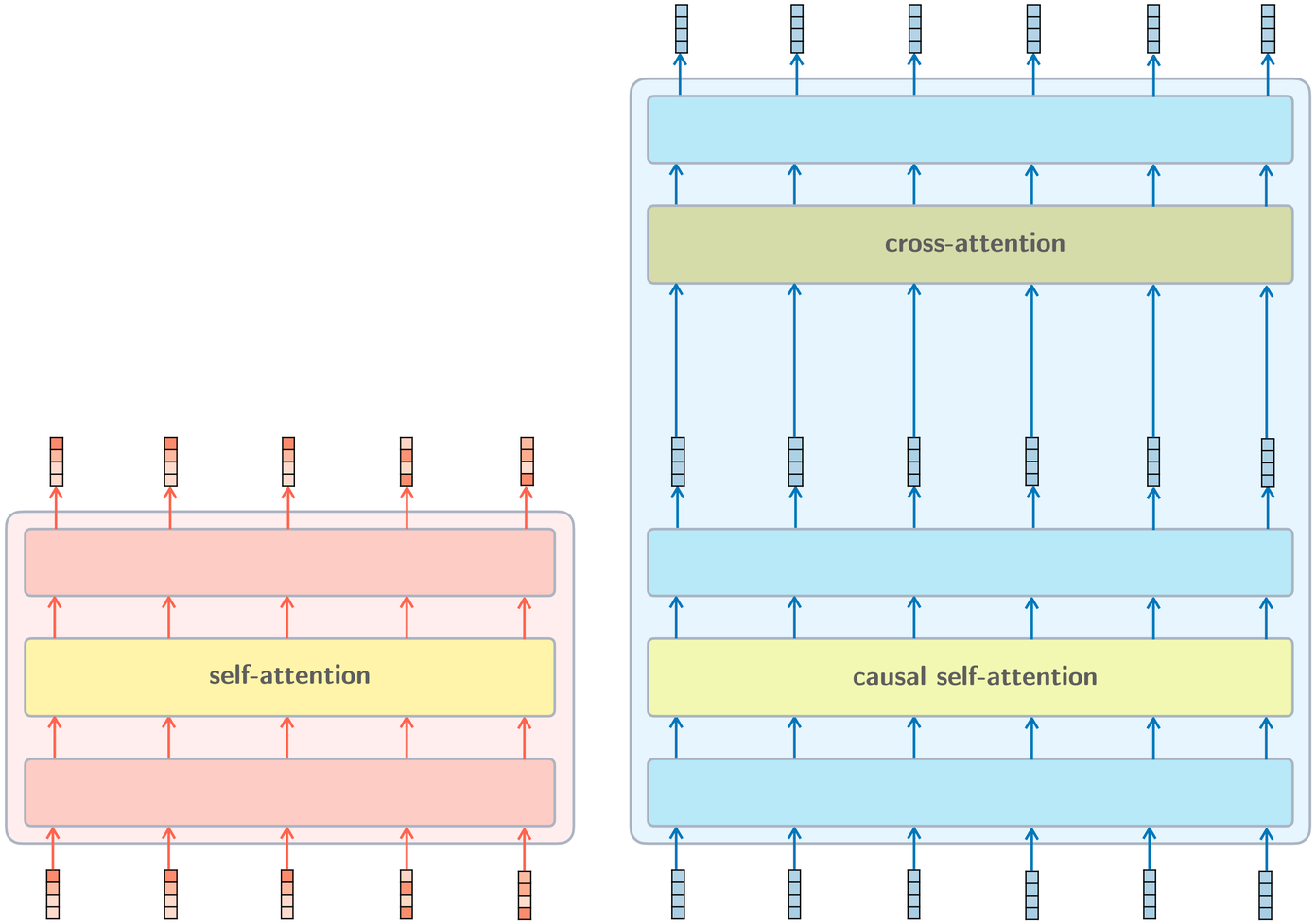
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

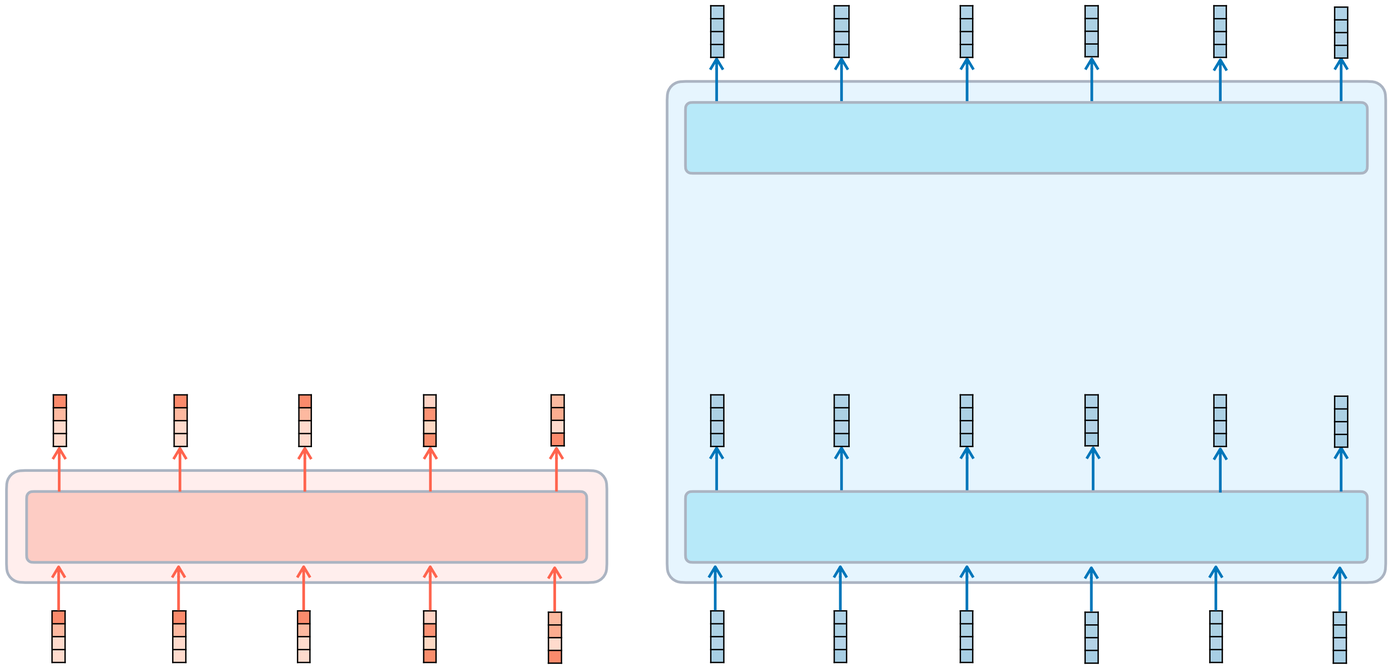
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

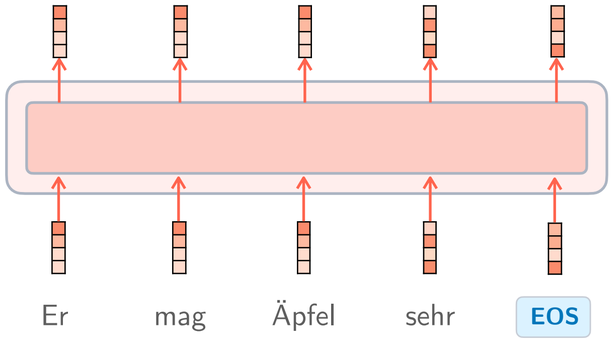
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

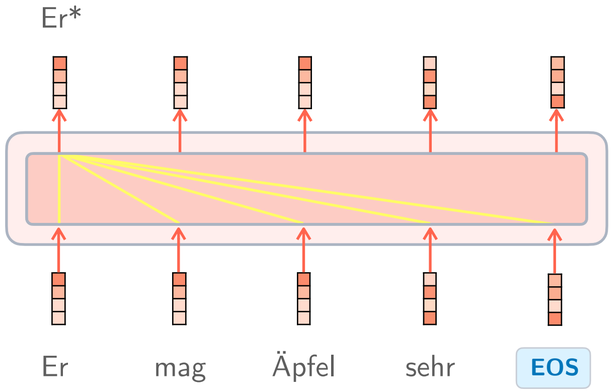
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

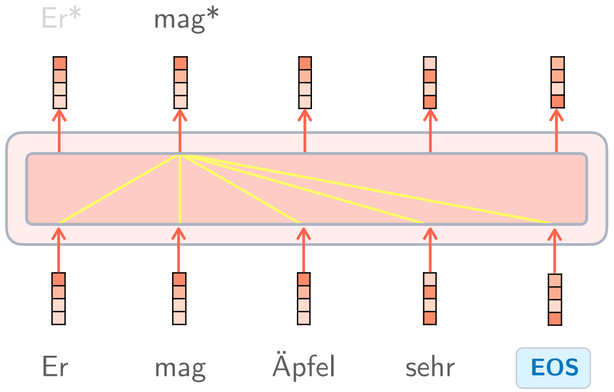
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

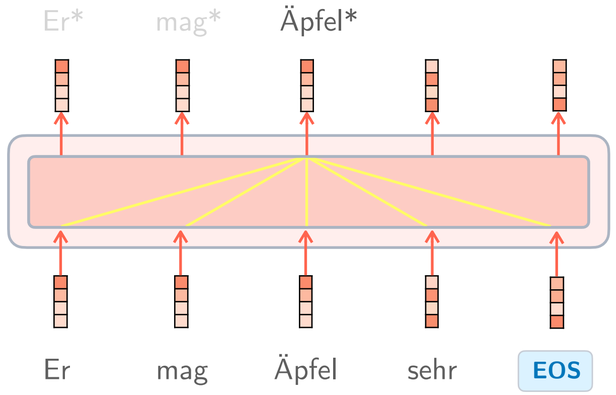
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

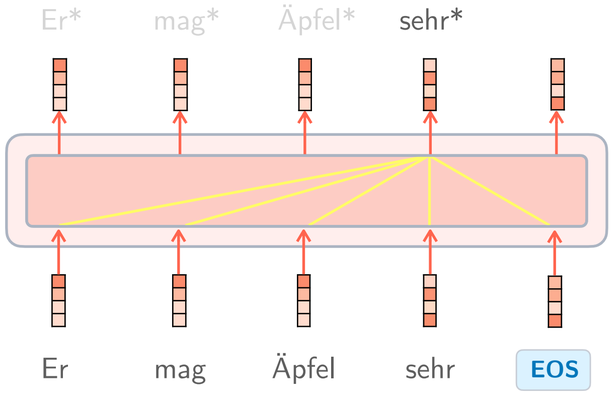
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

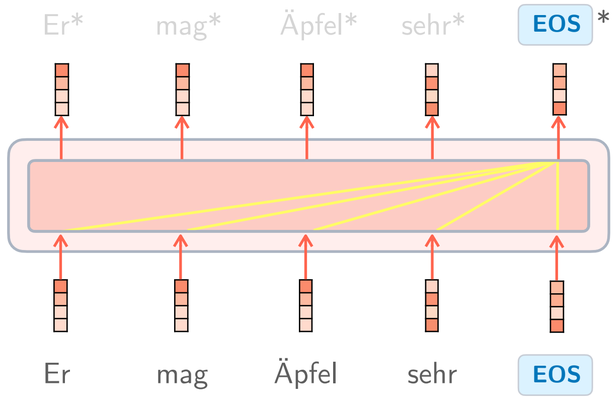
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

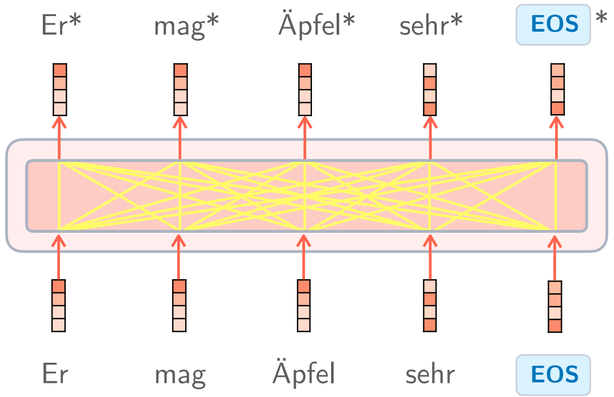


Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

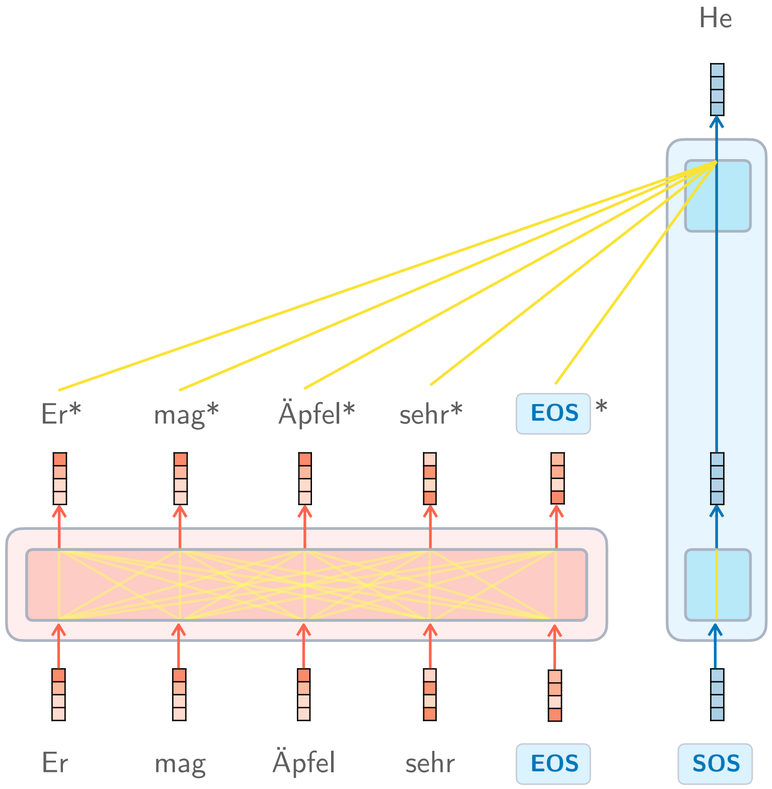
Notice that this happens as a single forward pass.



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**



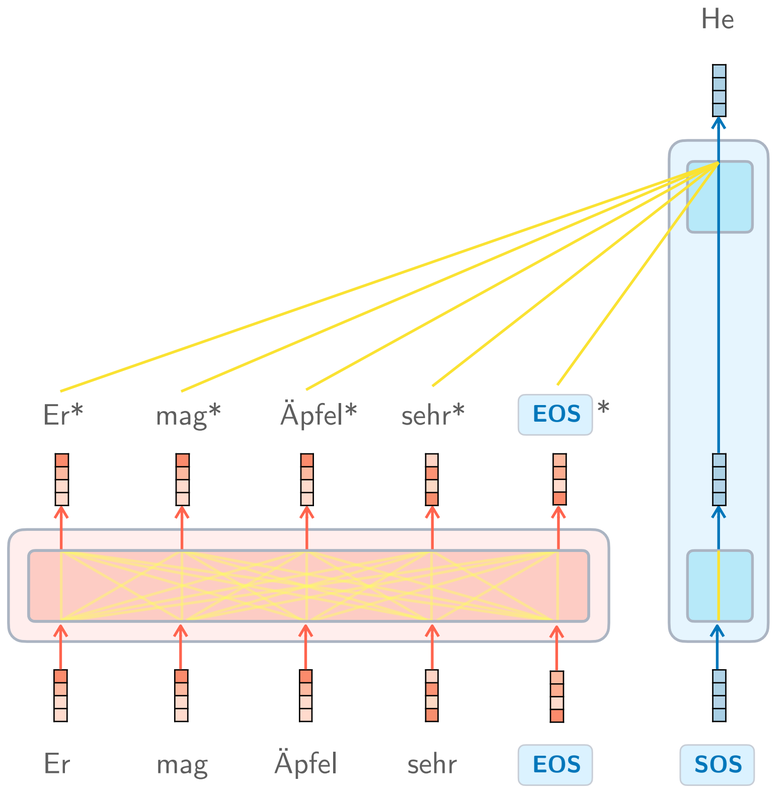
Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

He likes apples a lot EOS

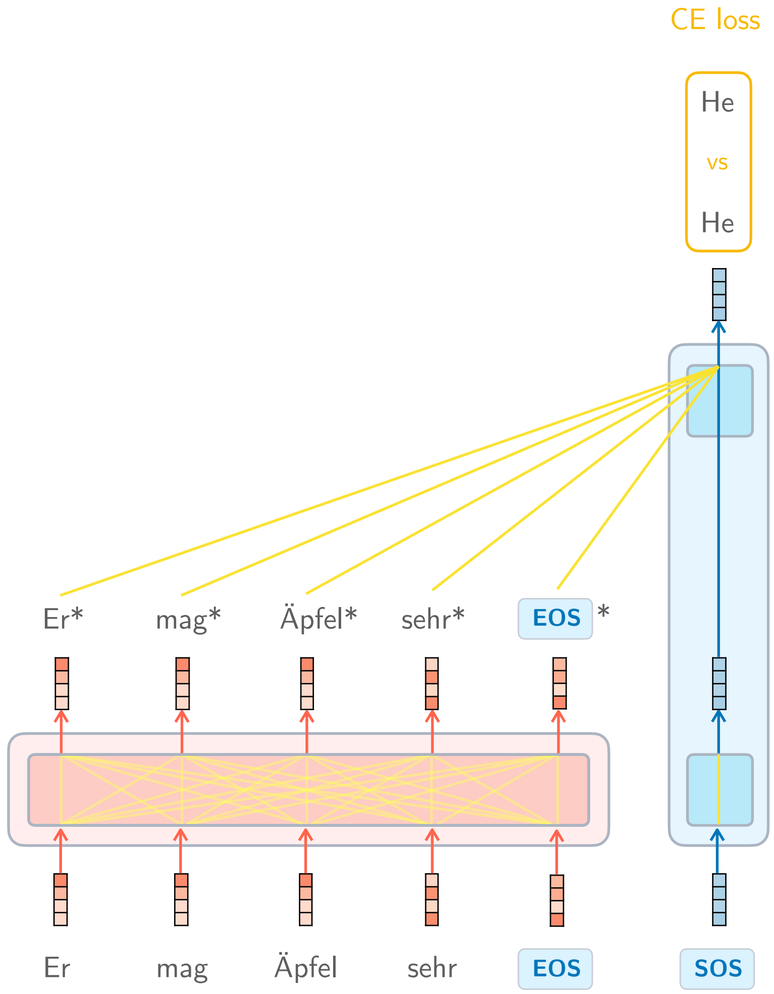
During training,
we know the correct answer!



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

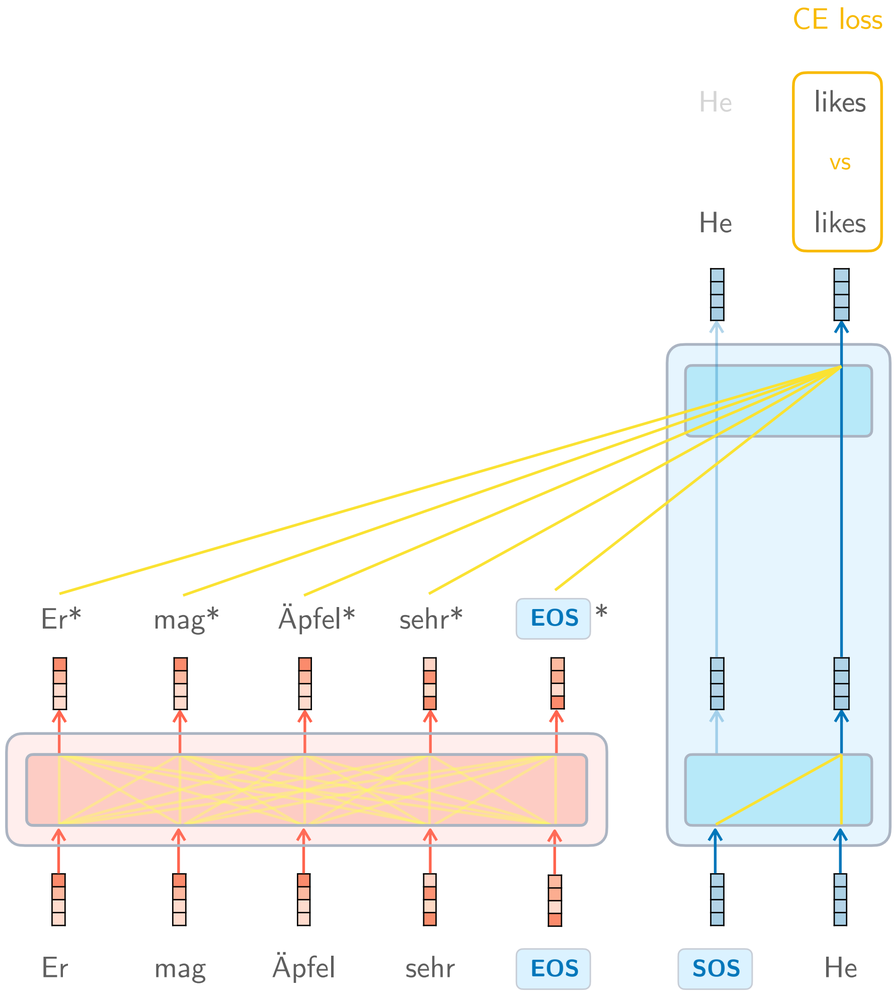


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

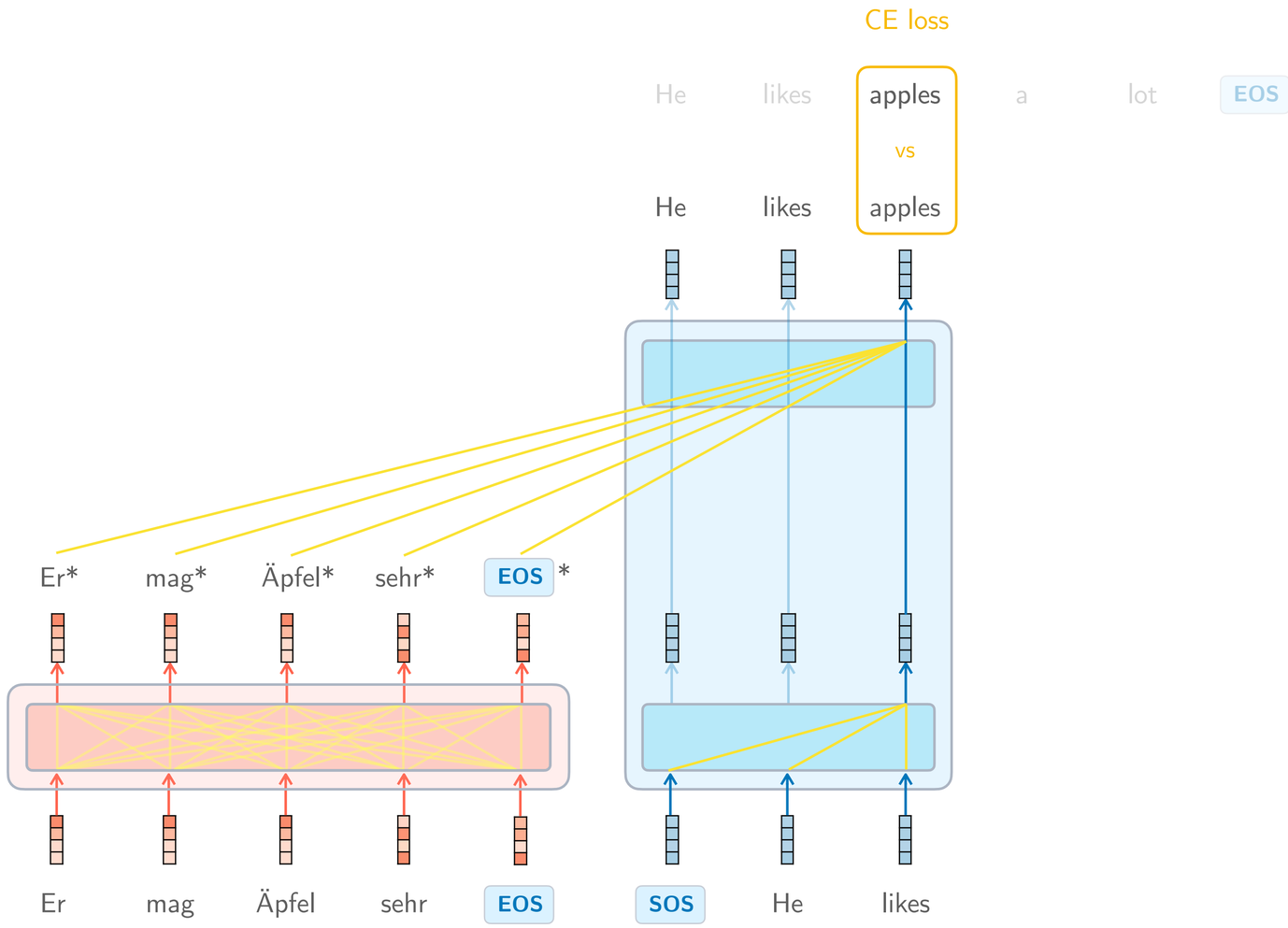


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

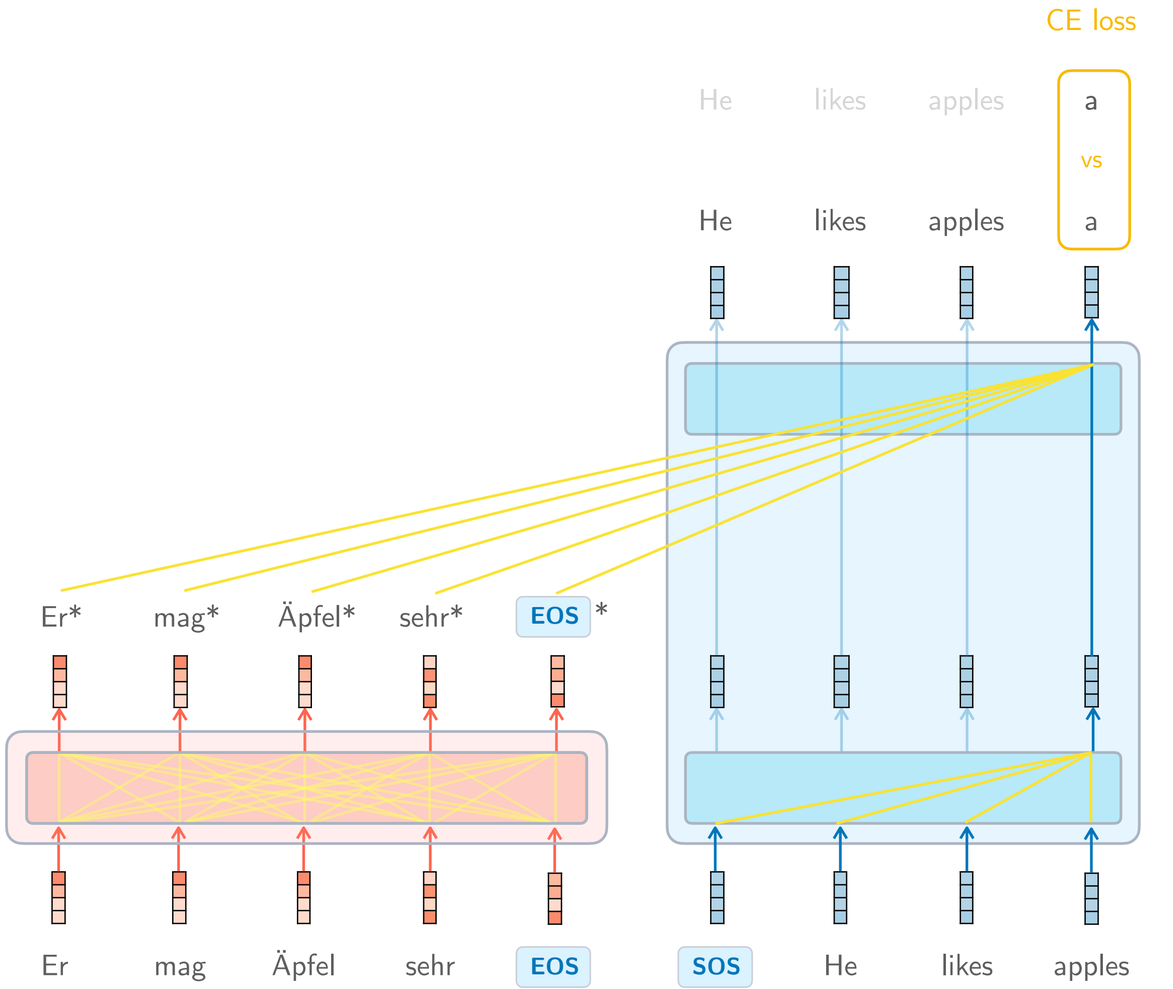


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

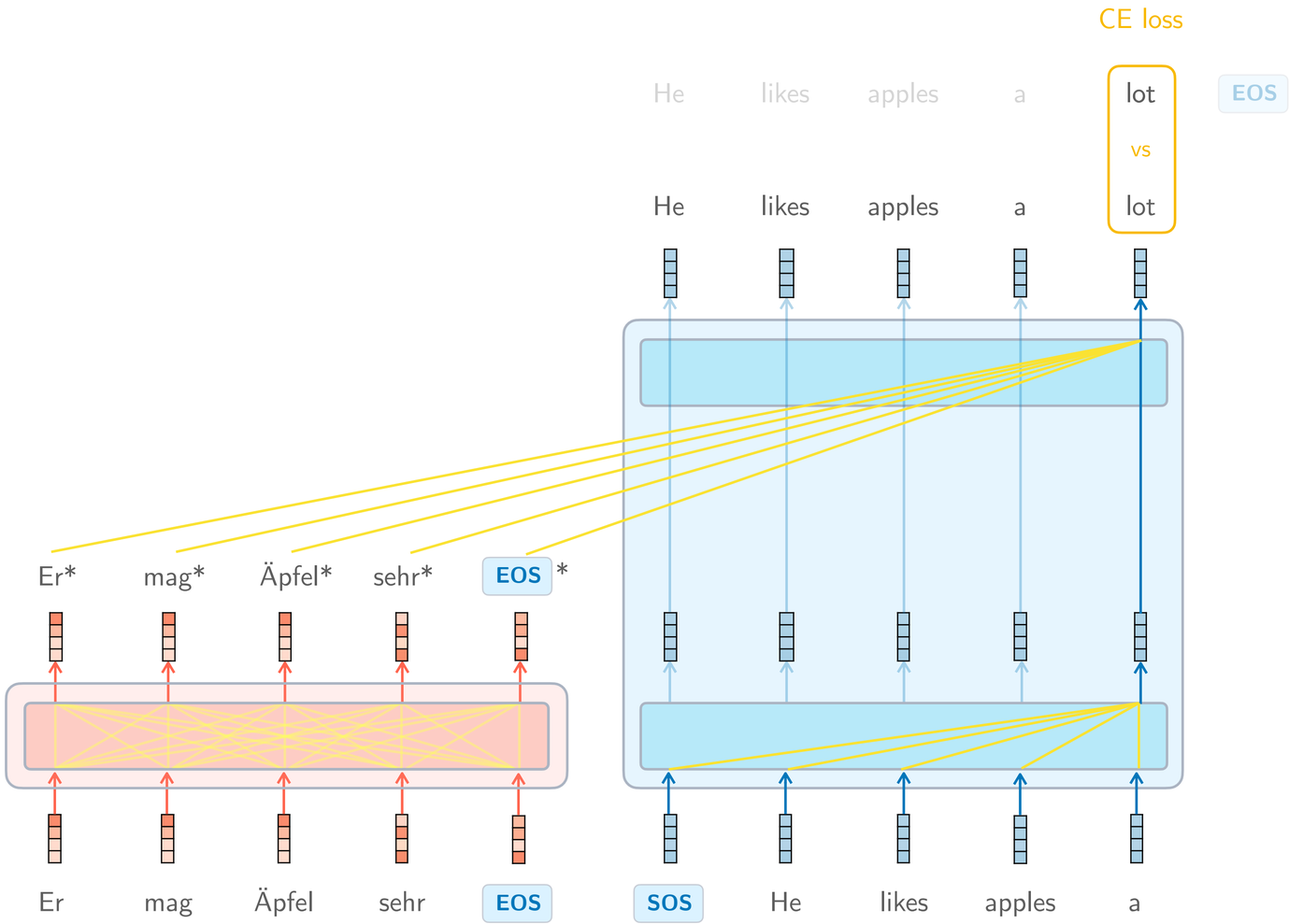


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

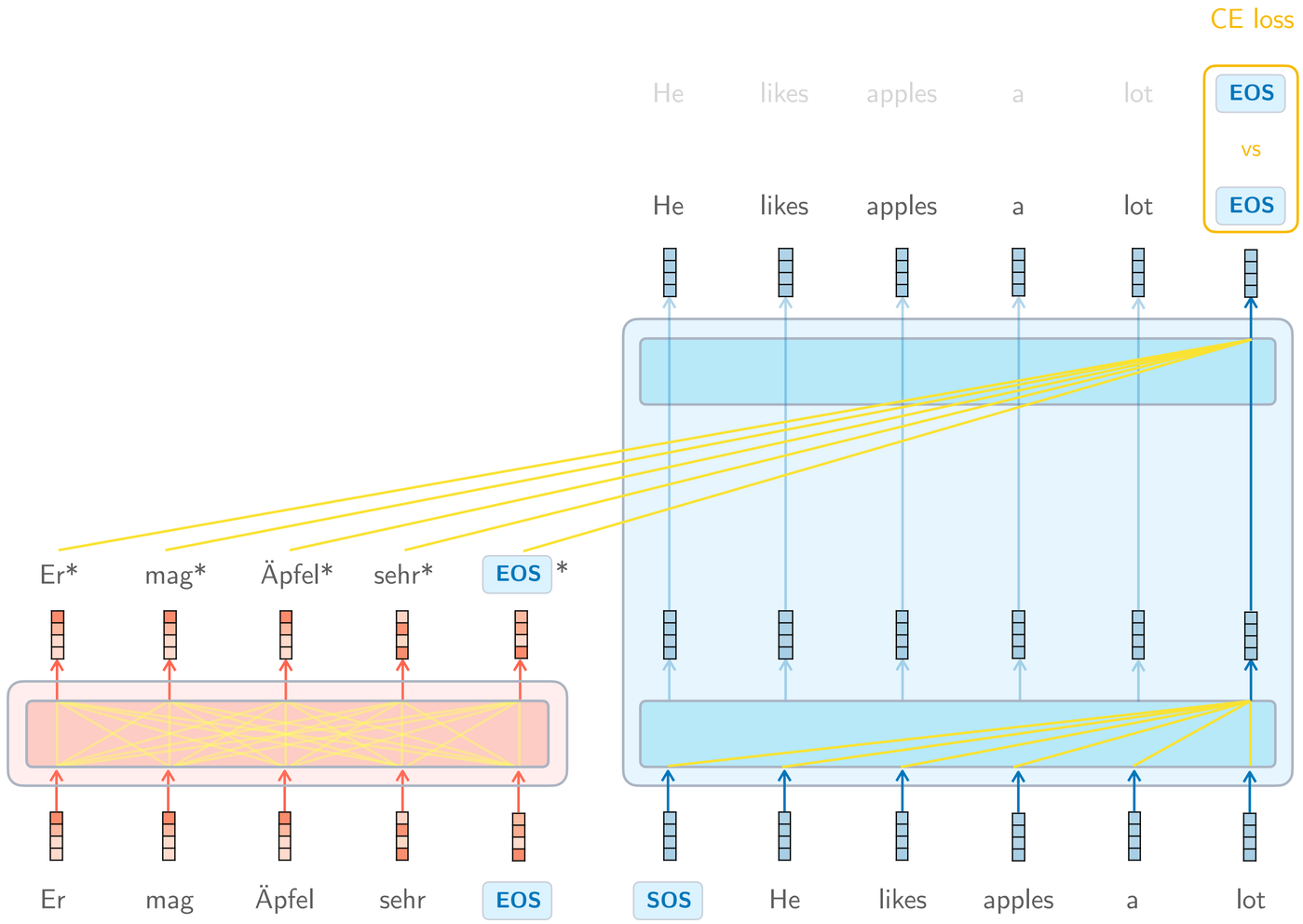


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**

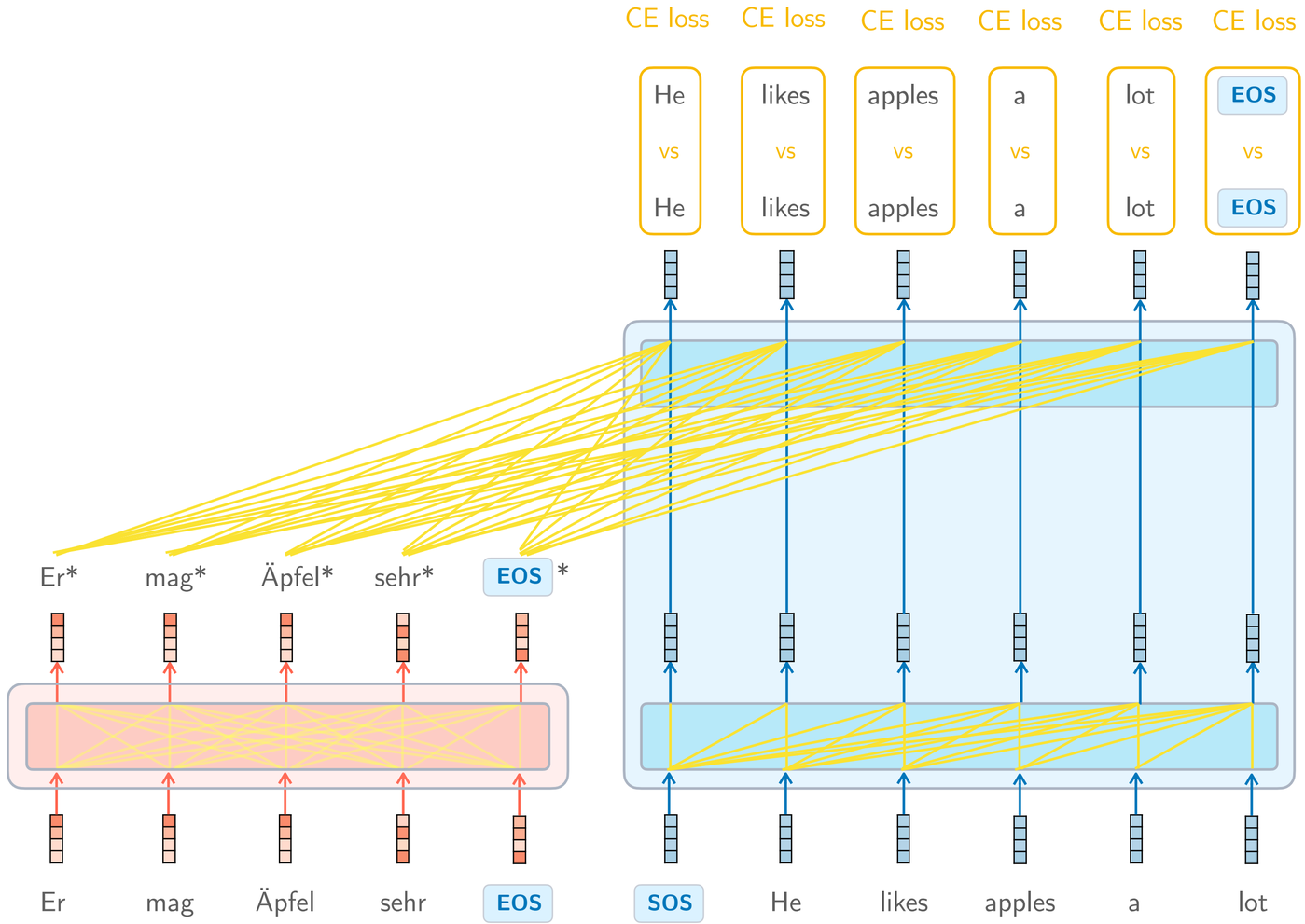


During training,
we know the correct answer!

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**



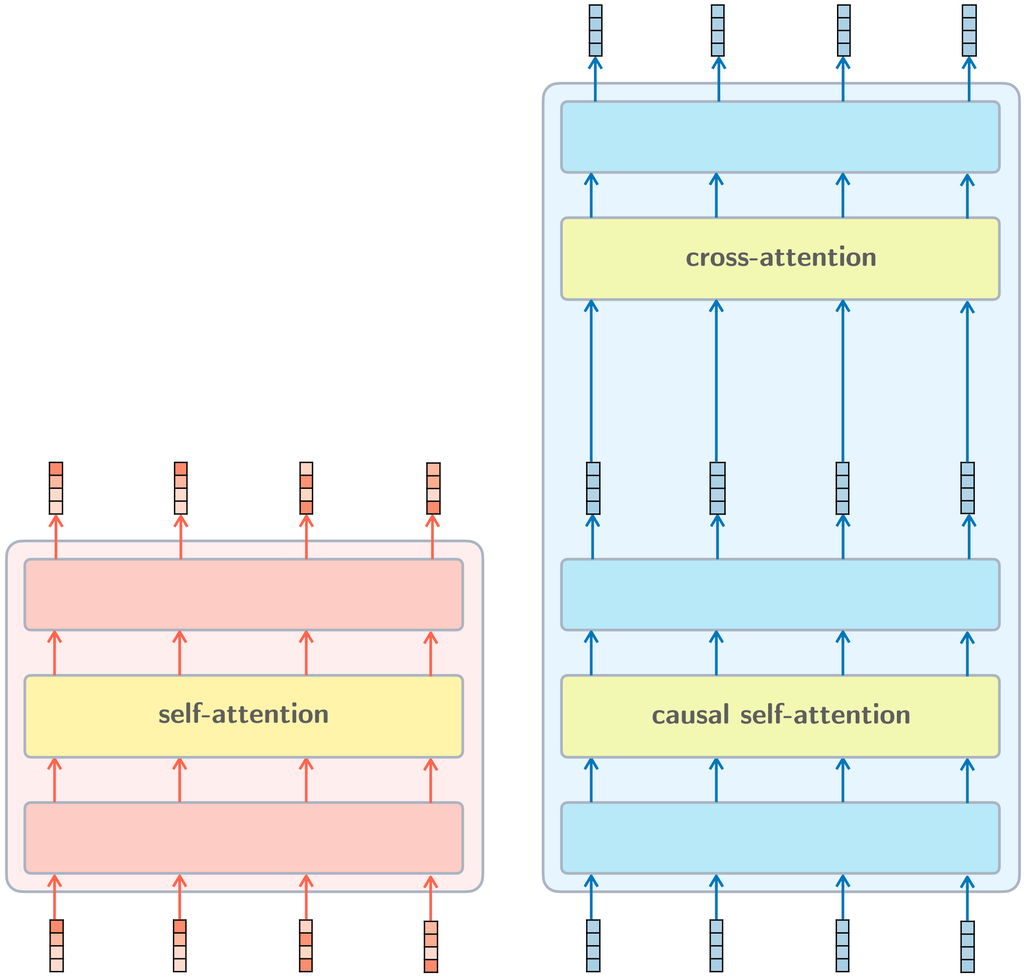
During training,
we know the correct answer!

During training,
we can do all *in parallel*.

Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

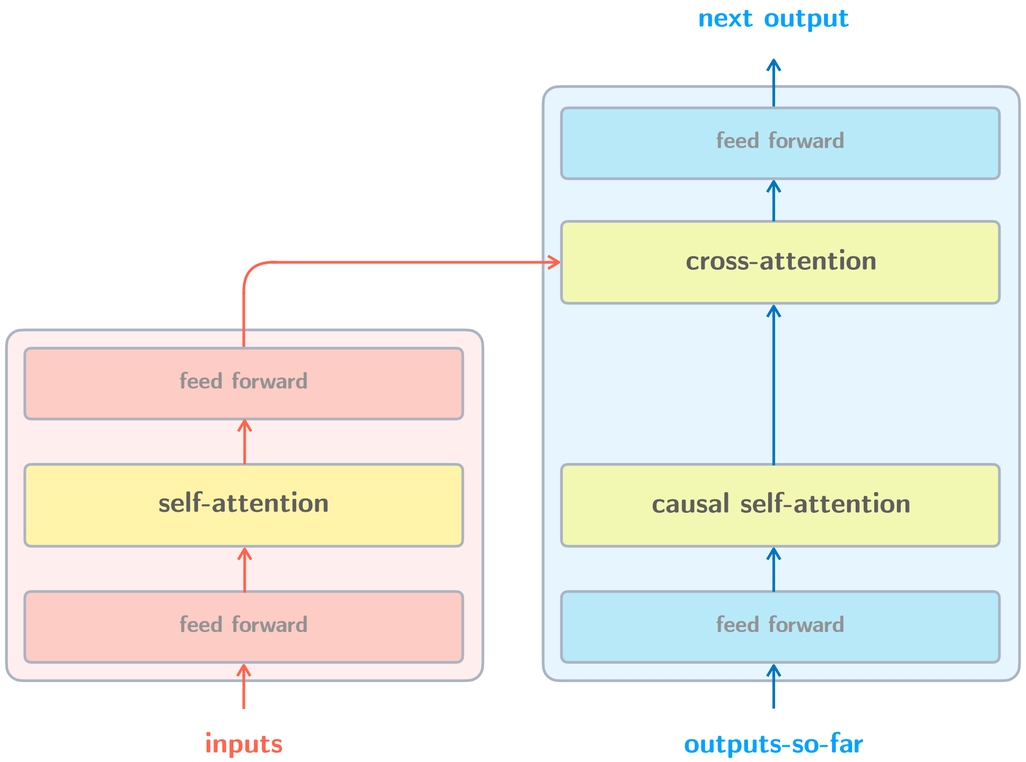
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

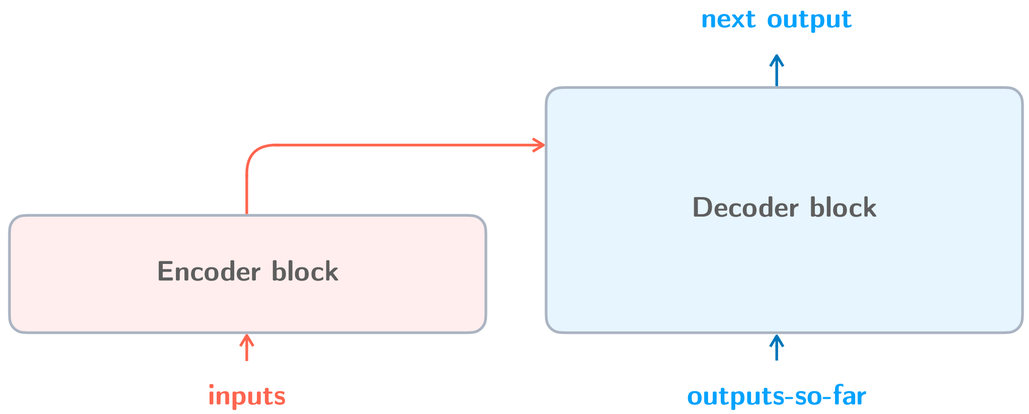
Feed **encoder** information into **decoder** **(cross-attention)**



Feed **encoder** information into **encoder** **(self-attention)**

Feed **decoder** information into **decoder** **(causal self-attention)**

Feed **encoder** information into **decoder** **(cross-attention)**



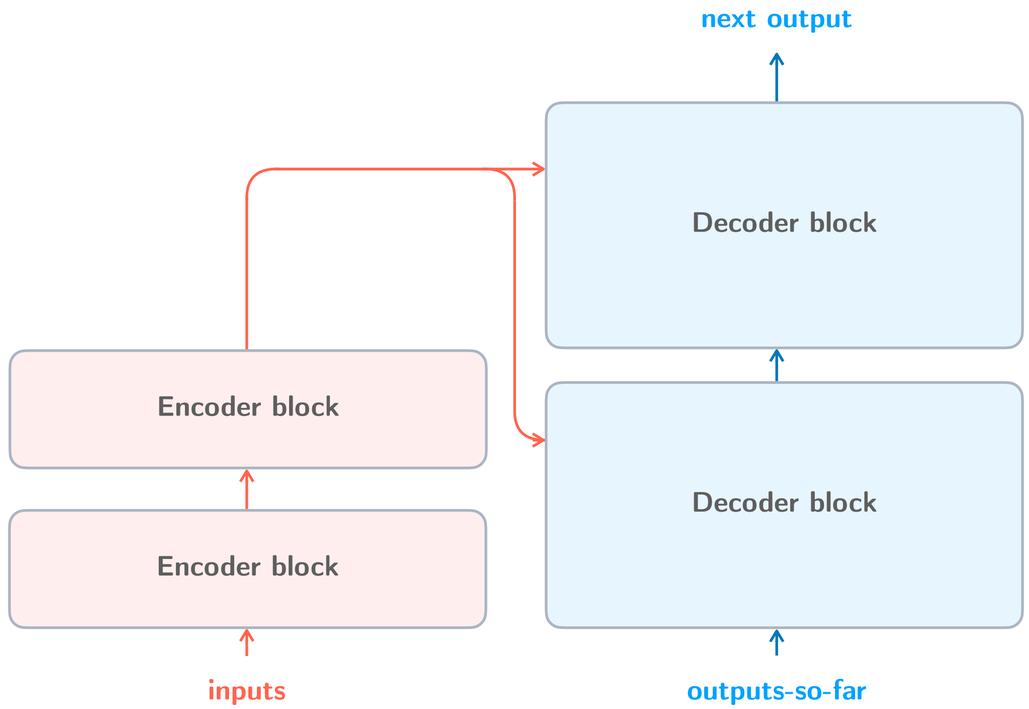
Feed **encoder** information into **encoder (self-attention)**

Feed **decoder** information into **decoder (causal self-attention)**

Feed **encoder** information into **decoder (cross-attention)**

No use in having **decoder** attend to half-baked **encoder representation**.

Run the full **encoder stack** and then have each **decoder block** attend over the same fully refined **encoder representation** (keys and values).



queries get more detailed and more informed by each layer

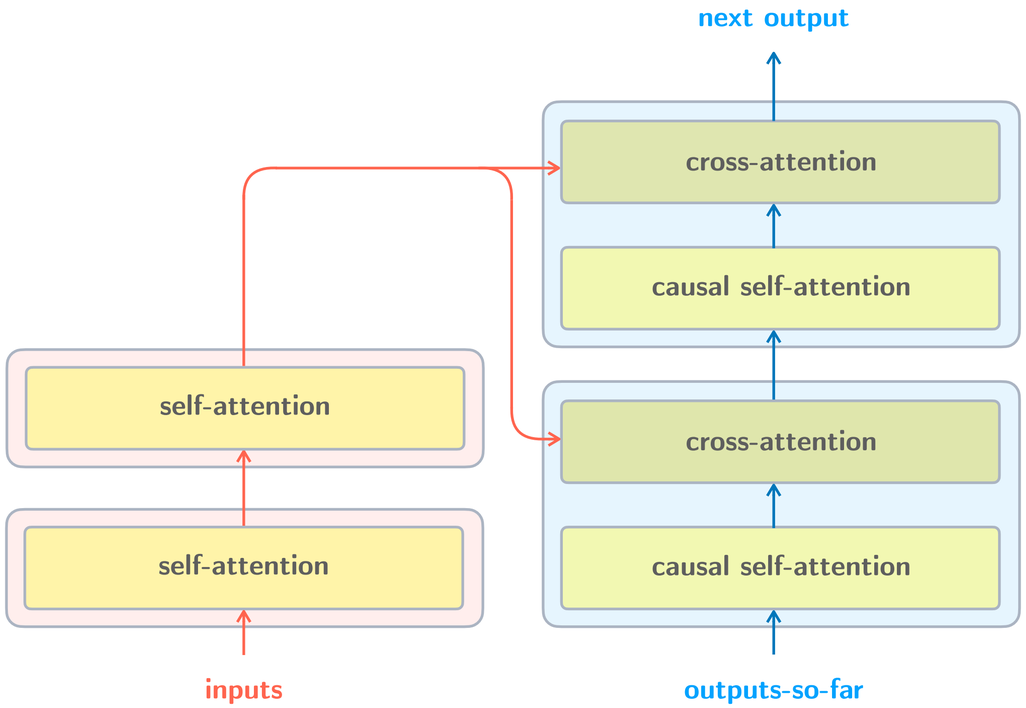
queries are informed by encoder now!

queries are only informed by decoder (contextualized outputs-so-far)

Feed **encoder** information into **encoder (self-attention)**

Feed **decoder** information into **decoder (causal self-attention)**

Feed **encoder** information into **decoder (cross-attention)**



queries get more detailed and more informed by each layer

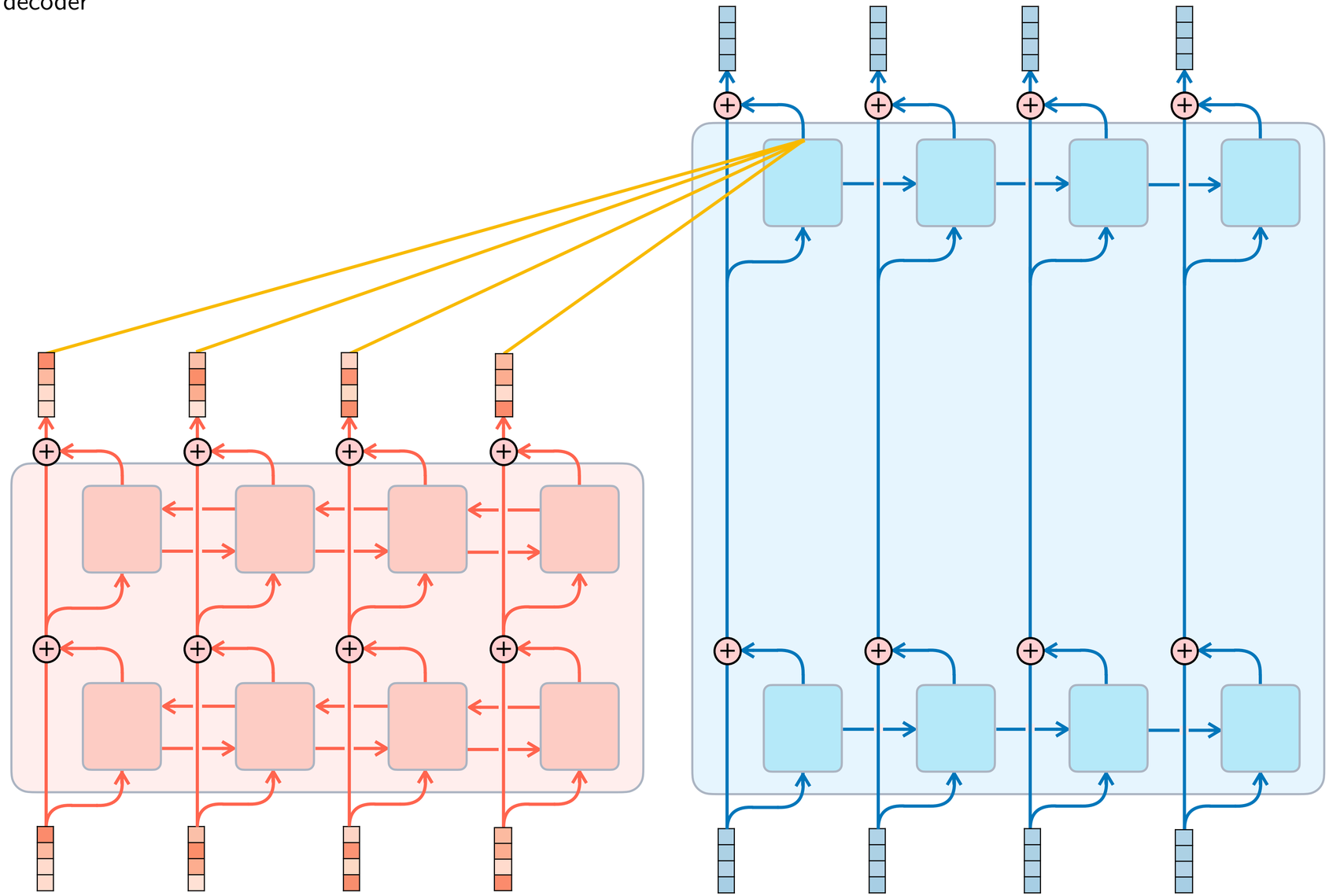
queries are informed by encoder now!

queries are only informed by decoder (contextualized outputs-so-far)

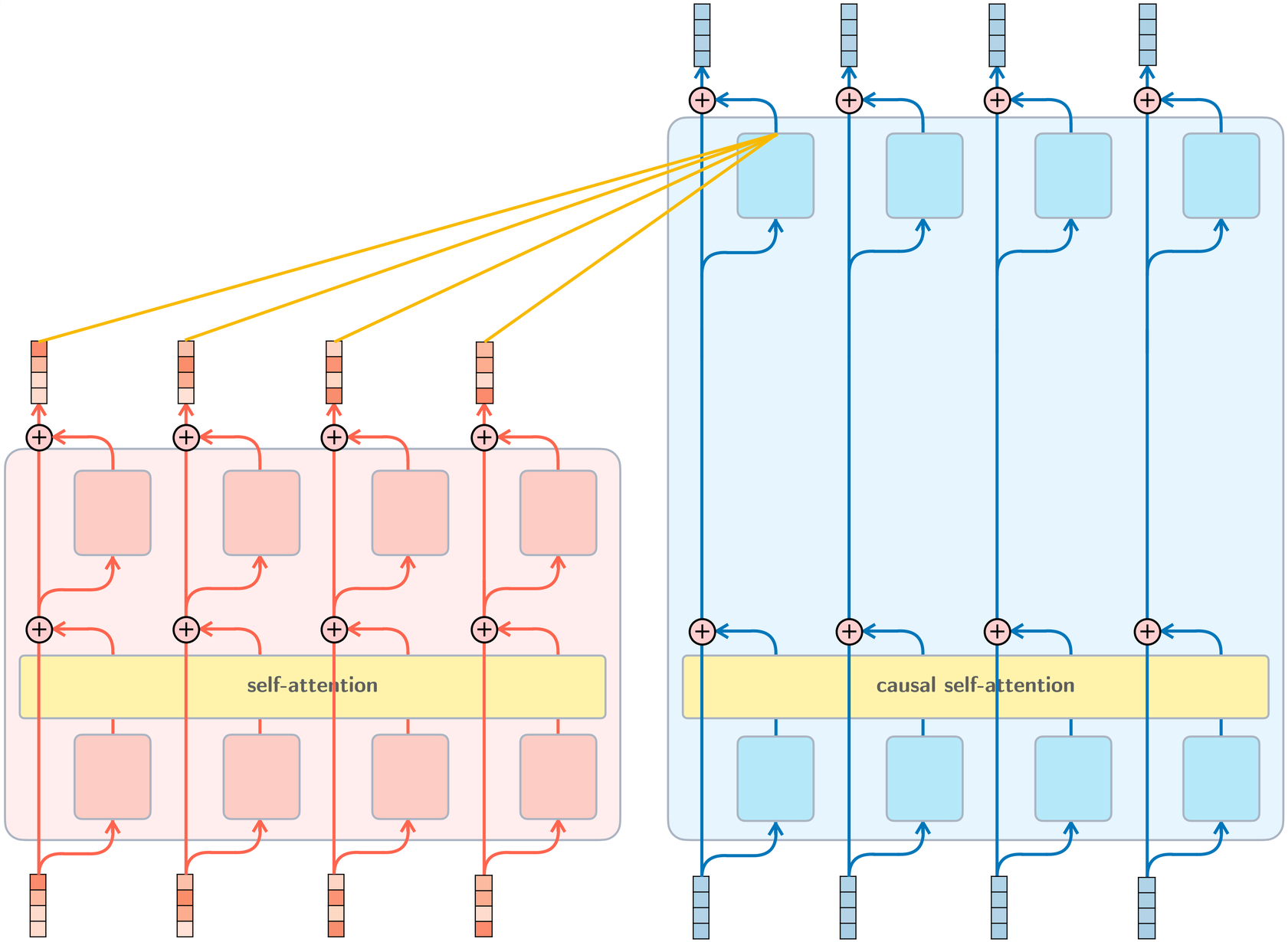
No use in having **decoder** attend to half-baked **encoder representation**.

Run the full **encoder stack** and then have each **decoder block** attend over the same fully refined **encoder representation** (keys and values).

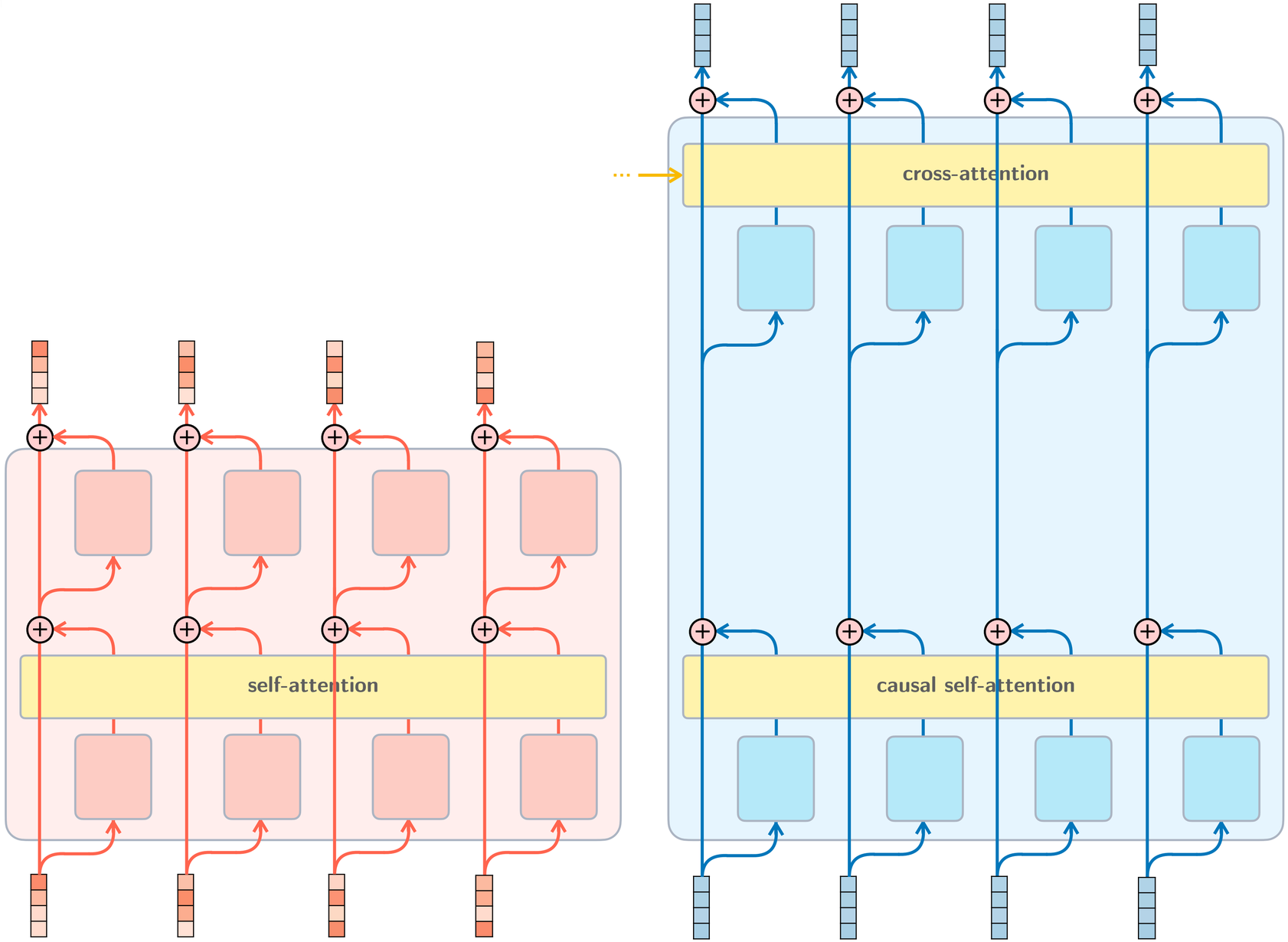
GNMT (2016), encoder and decoder



Transformer encoder–decoder



Transformer encoder–decoder



Alternative visualization

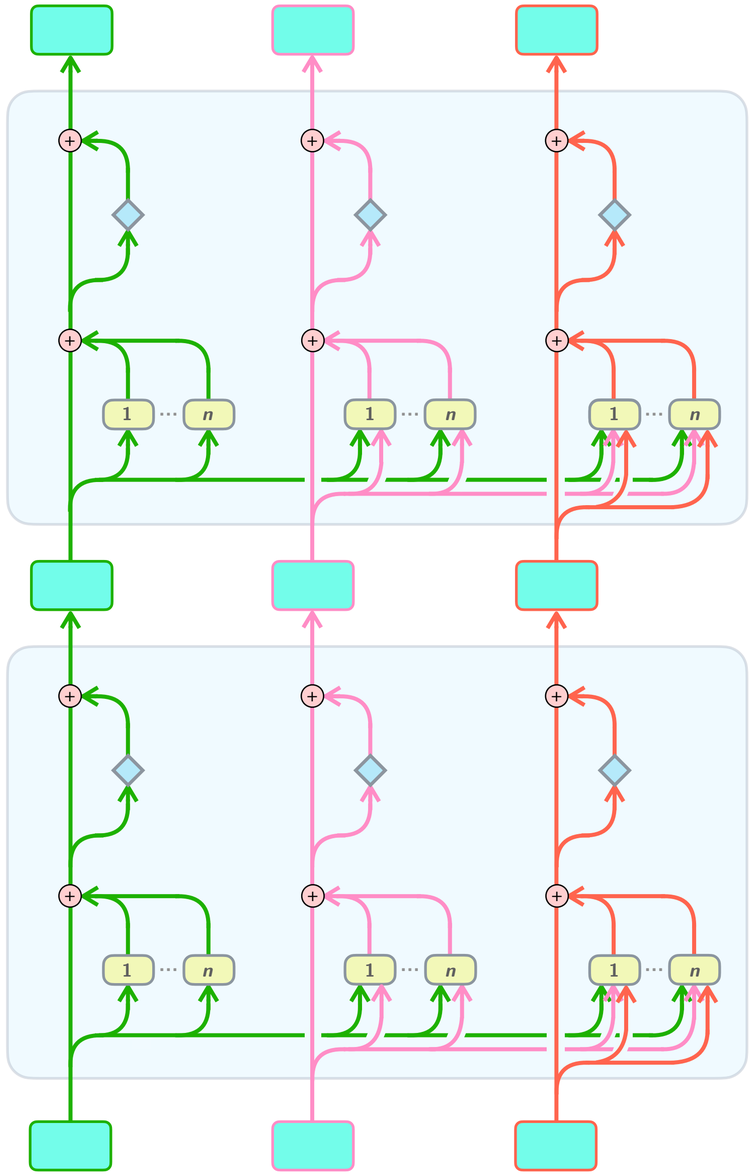


Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

Alternative visualization

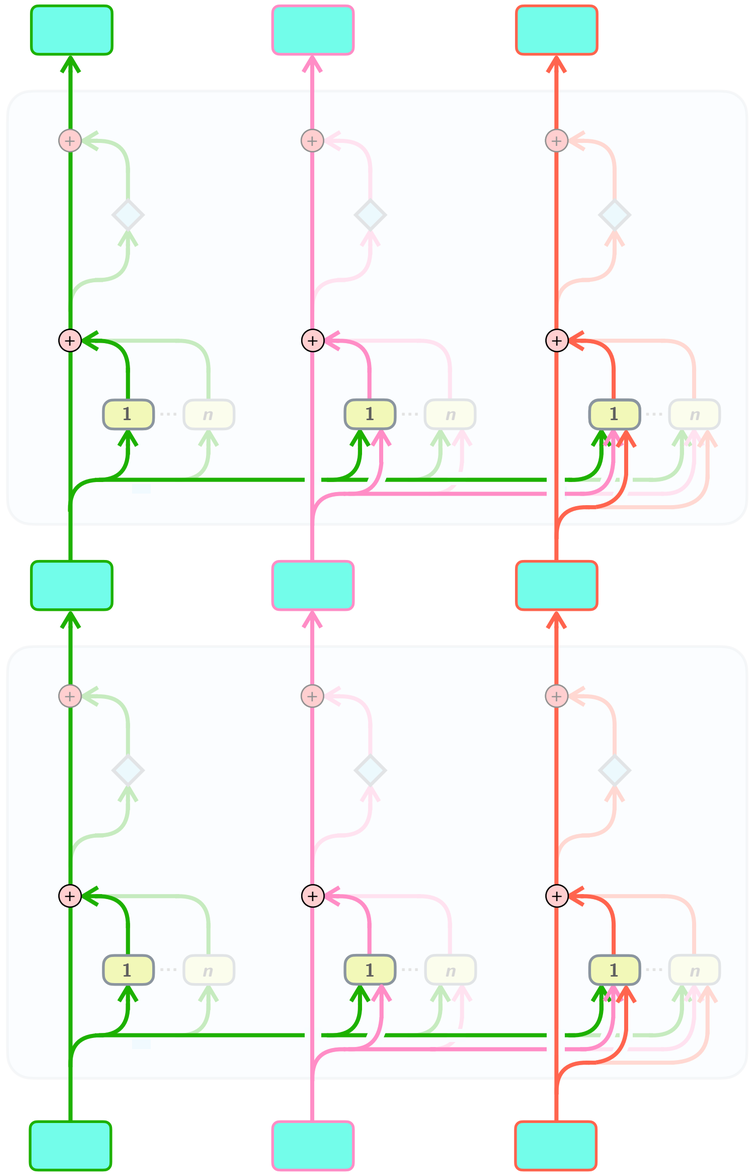
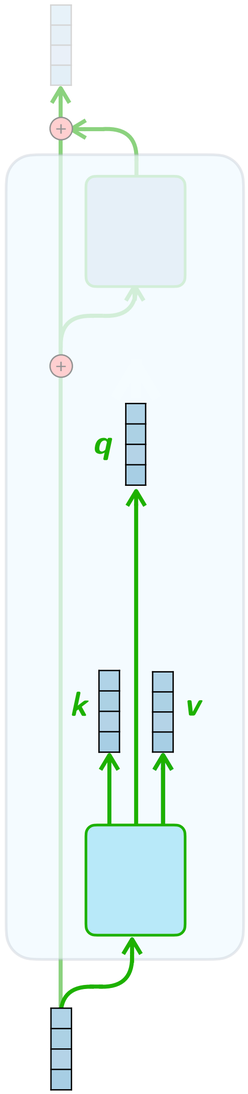
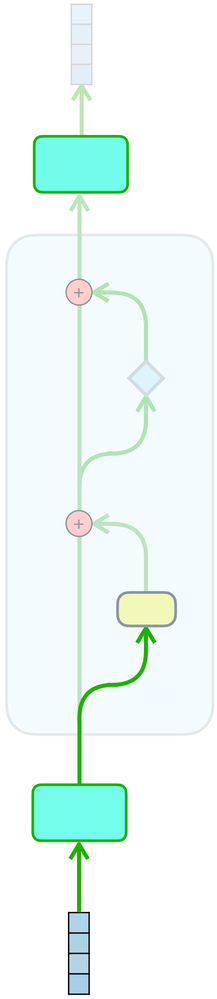


Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

Alternative visualization

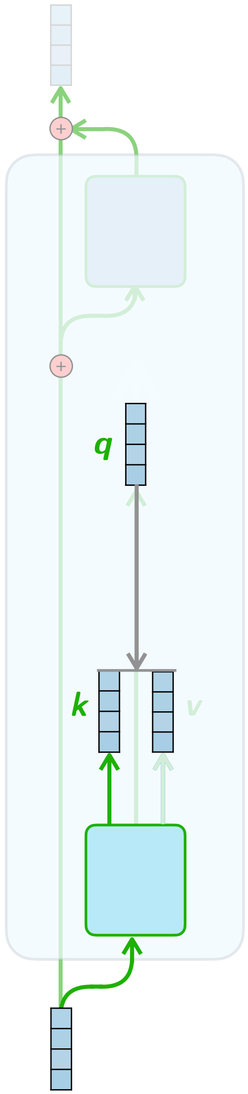
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

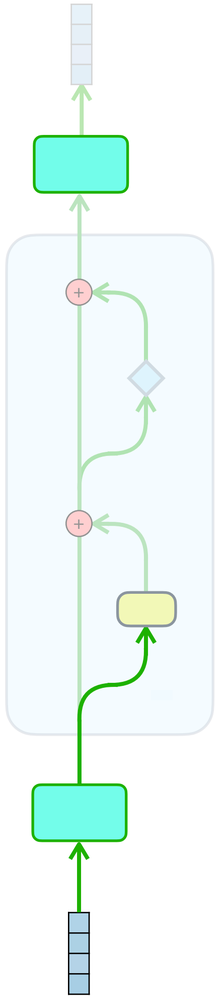




Alternative visualization

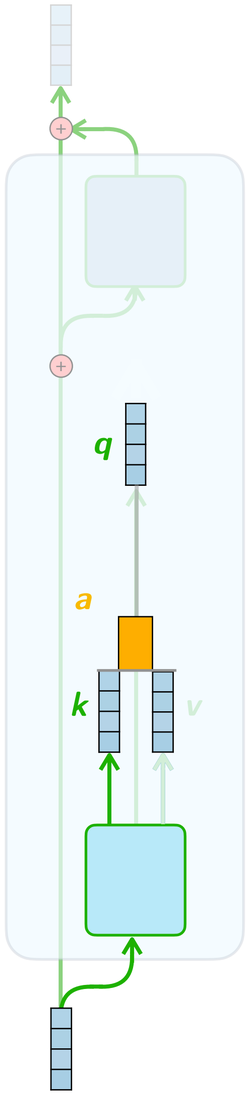
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

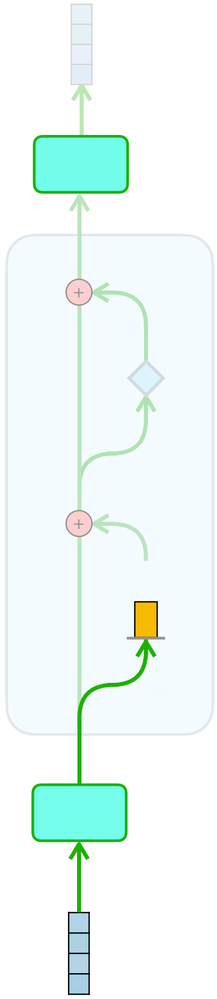




Alternative visualization

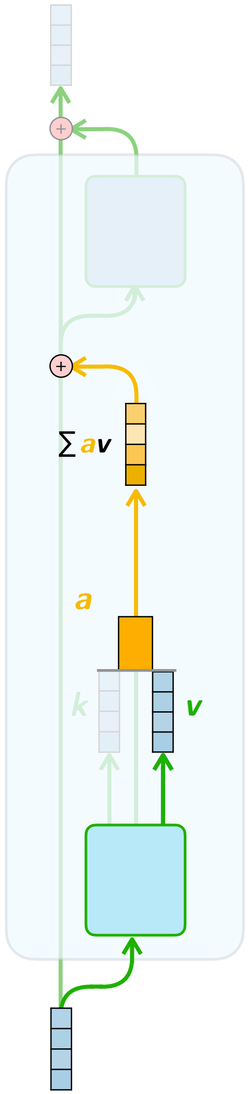
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

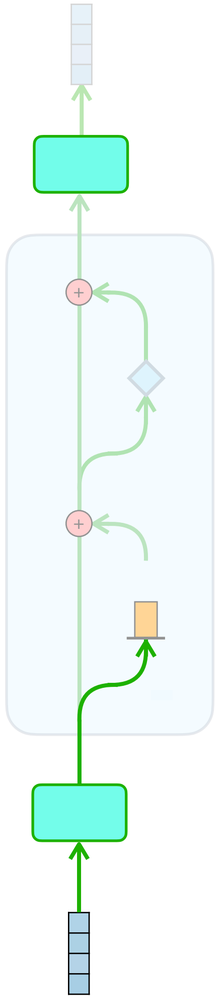




Alternative visualization

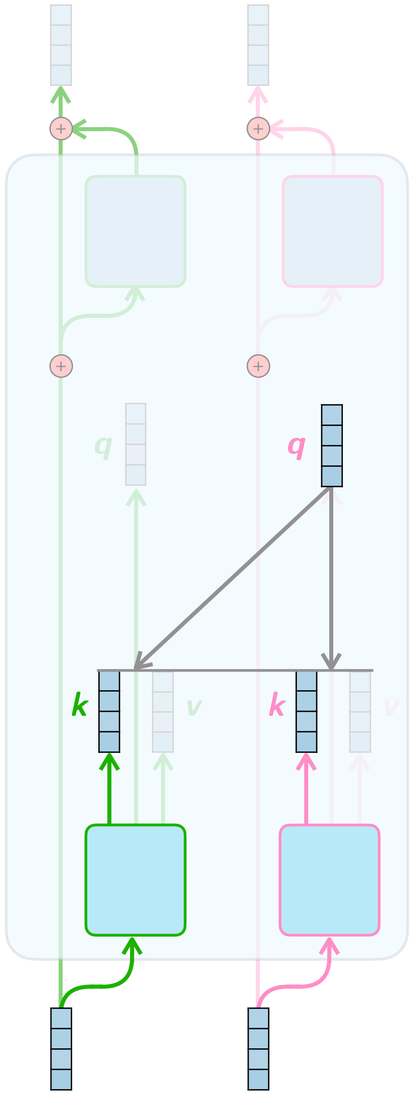
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

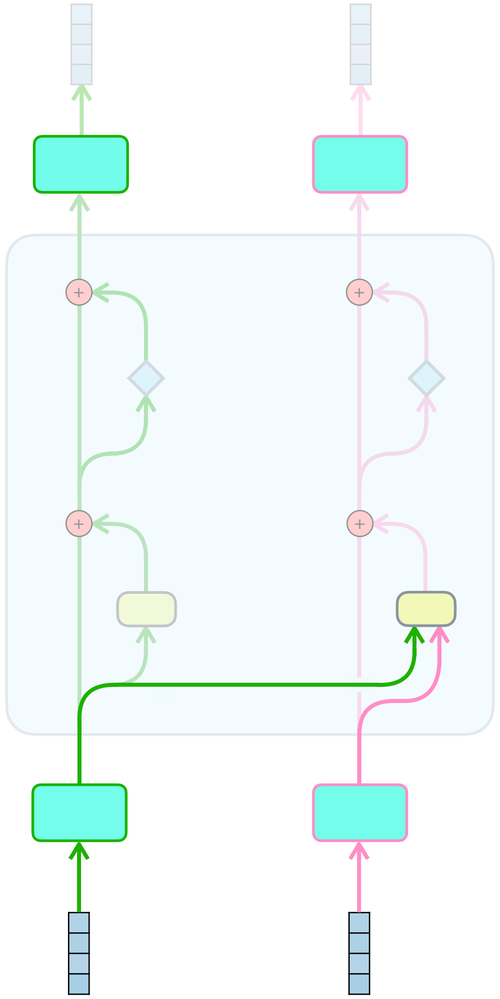




Alternative visualization

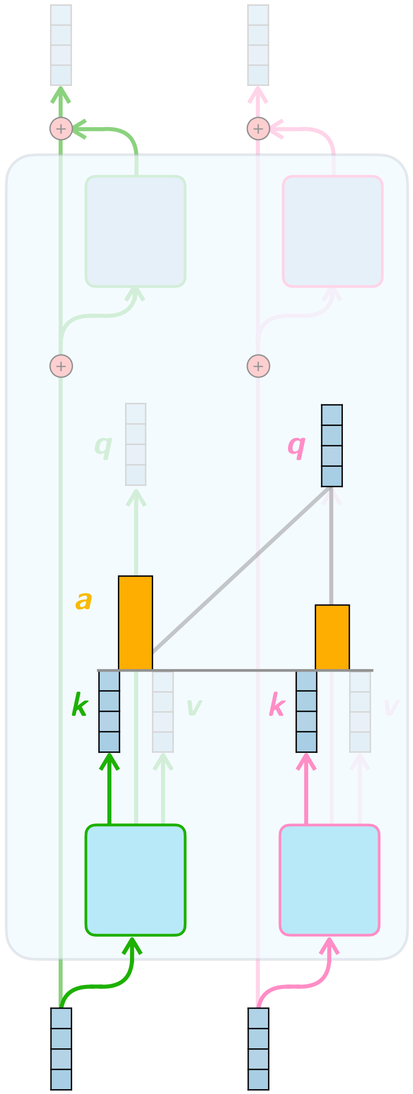
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

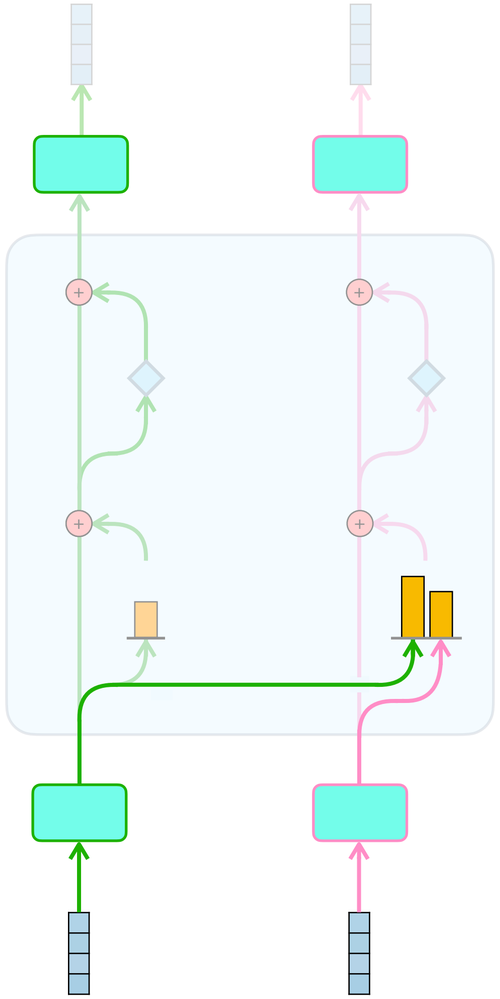




Alternative visualization

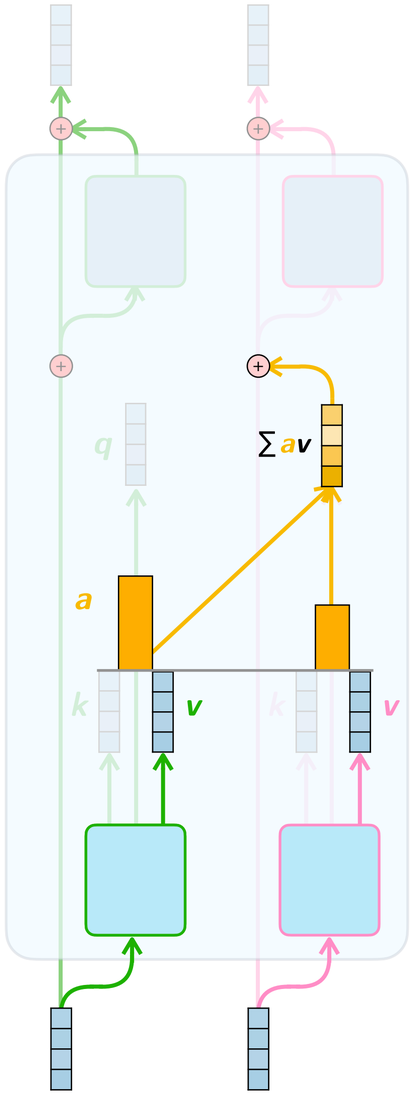
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

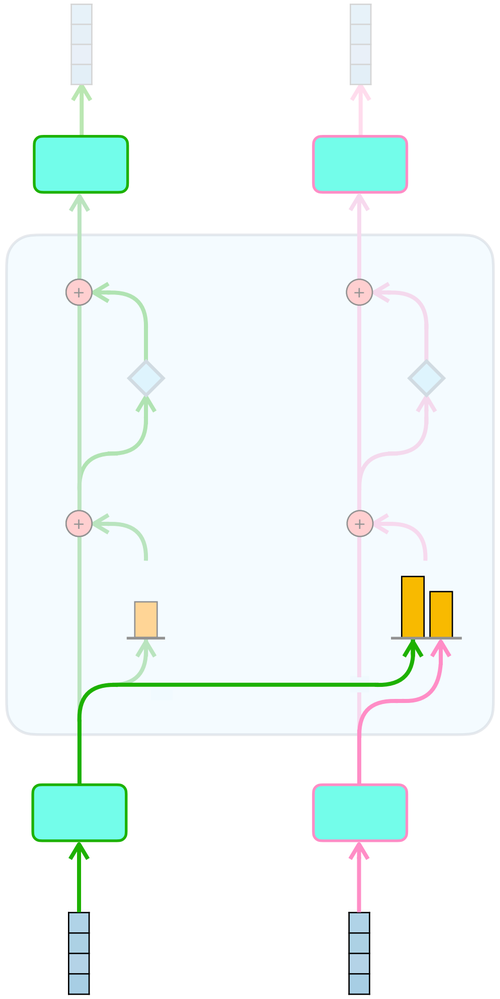




Alternative visualization

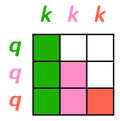
Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

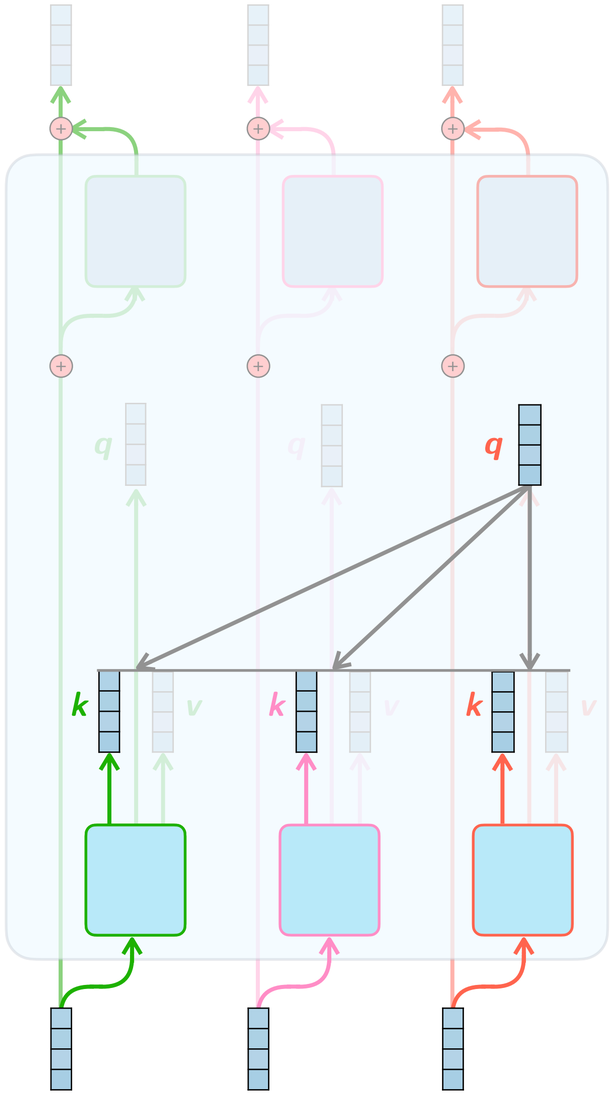


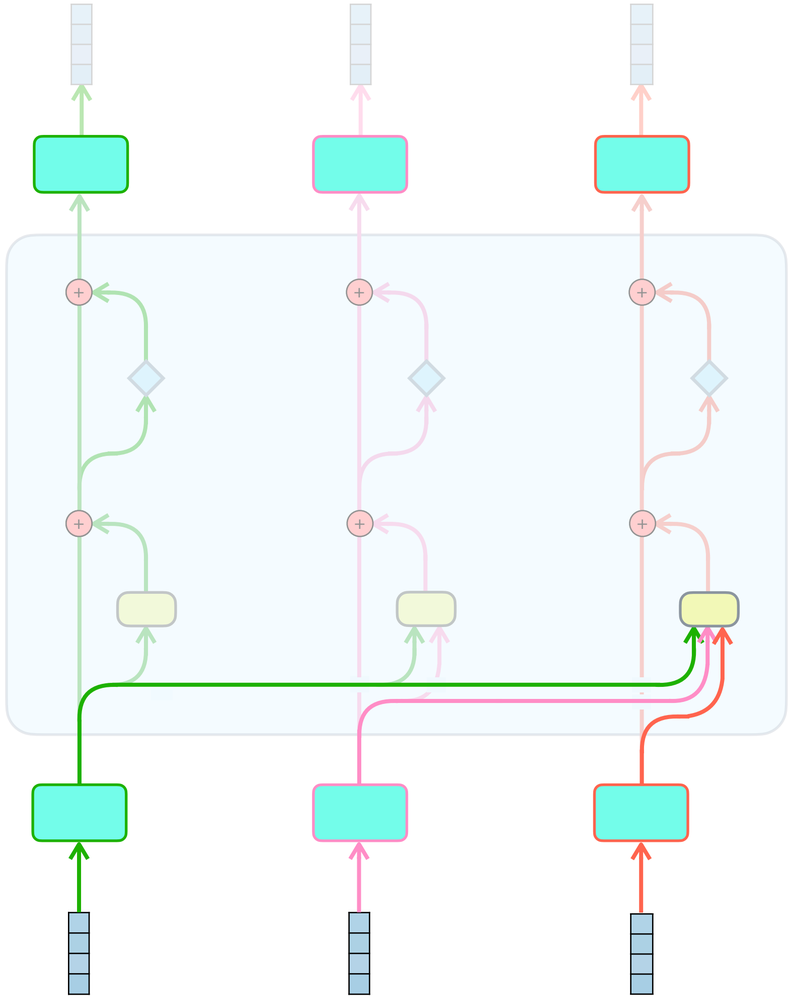


Alternative visualization

Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

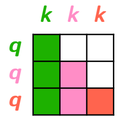


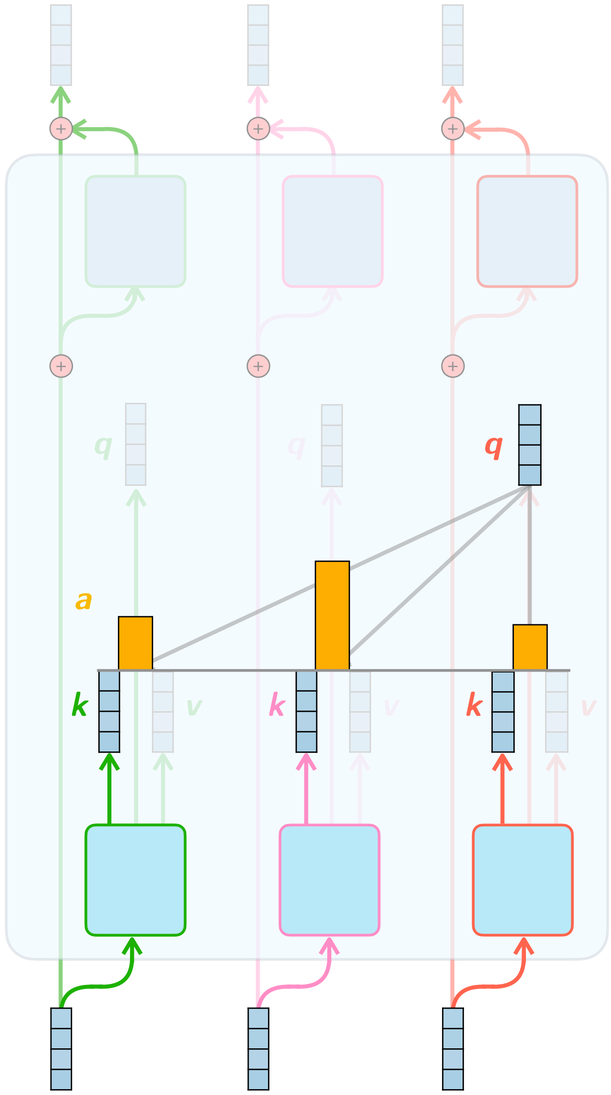


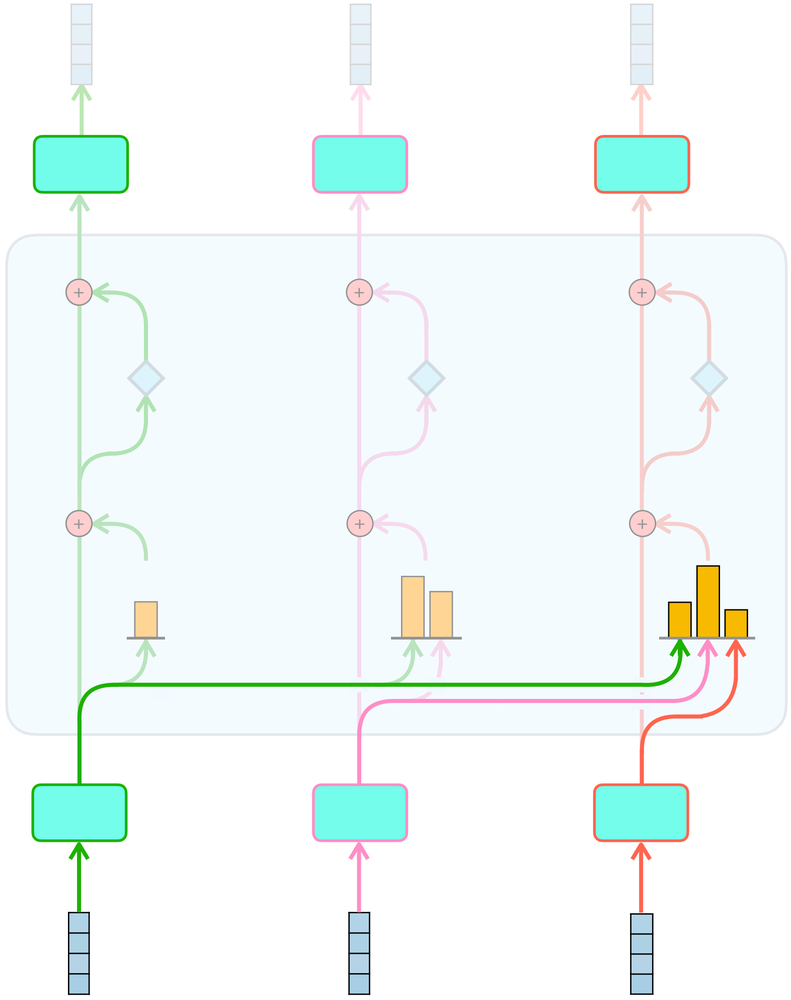


Alternative visualization

Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)

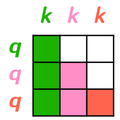


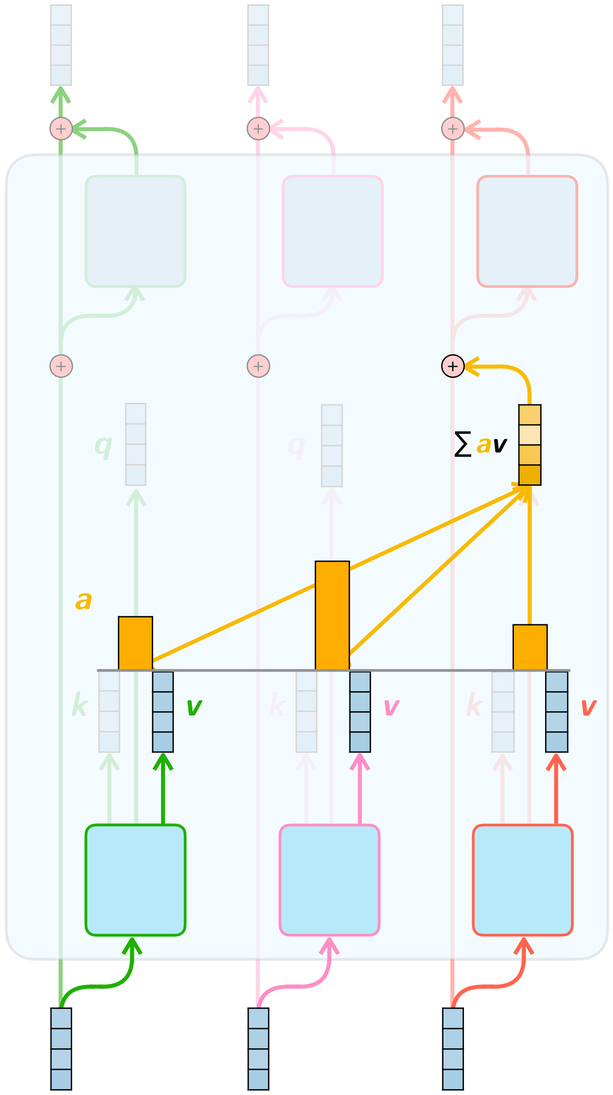


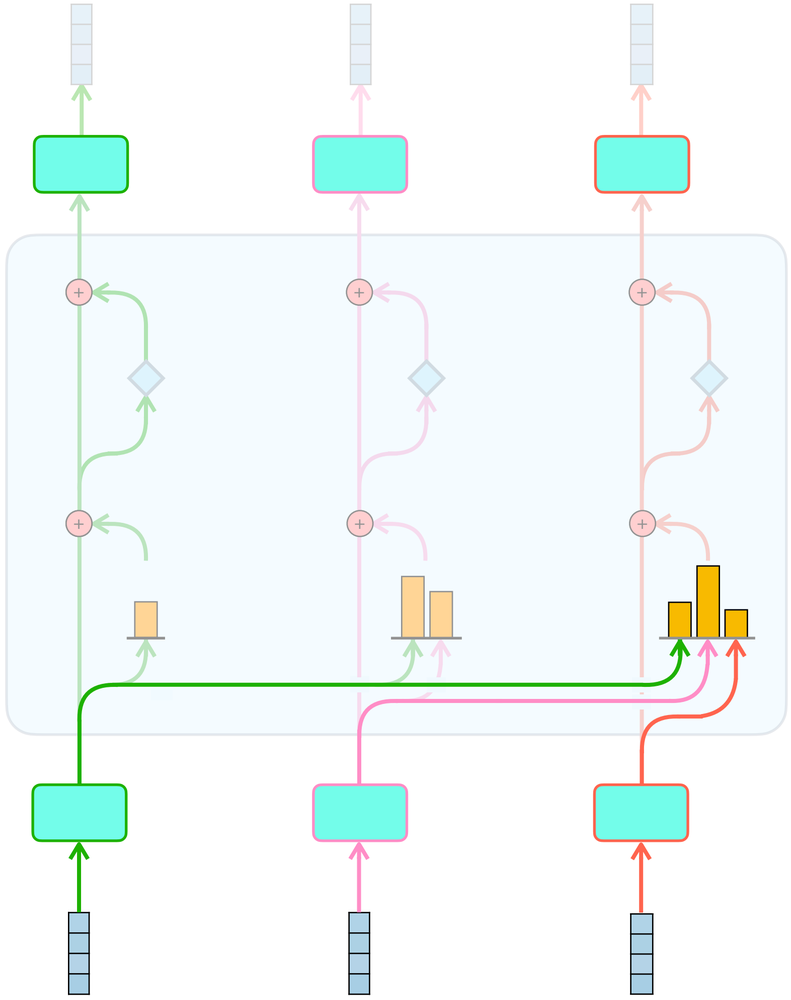


Alternative visualization

Diagram inspired by "A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models" by Rai et al. (2024)



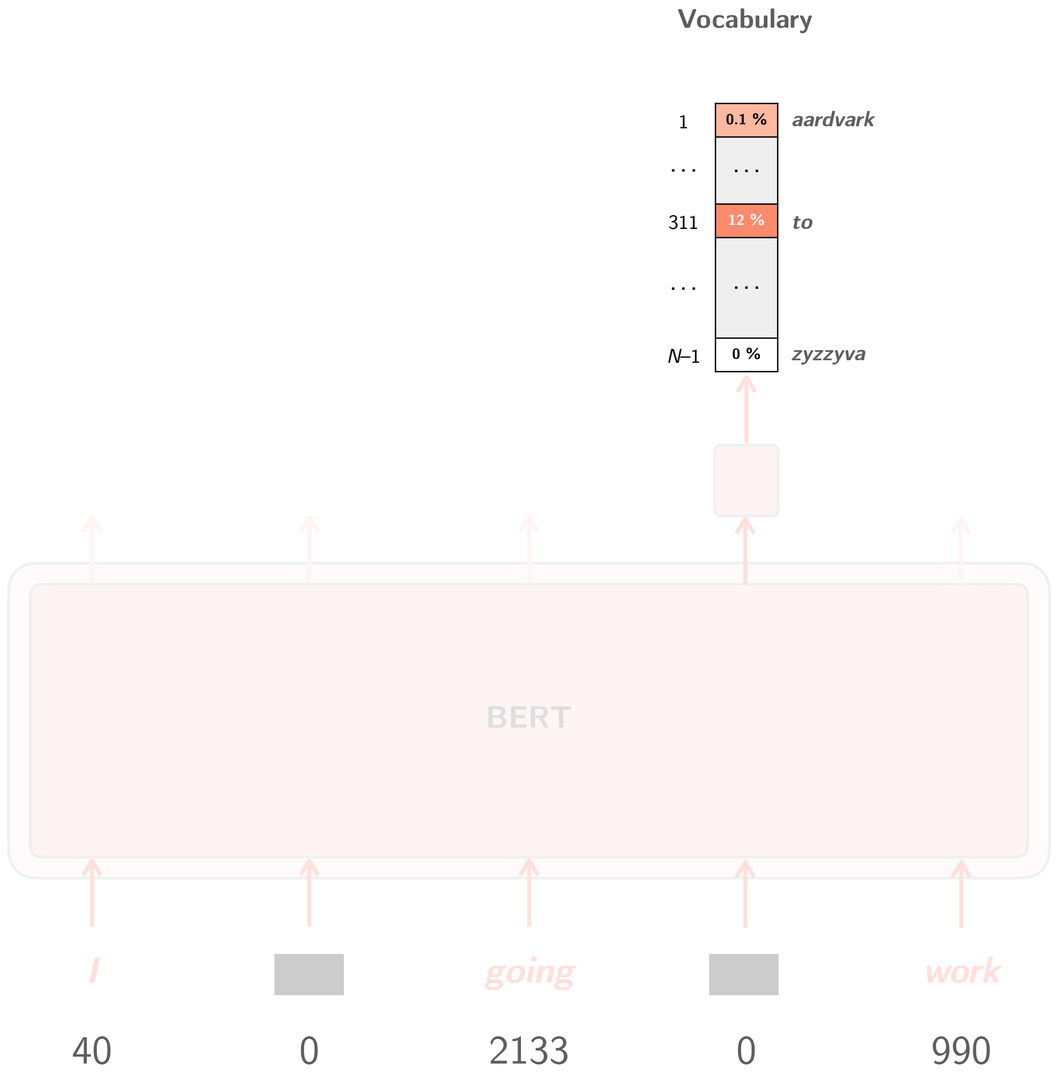




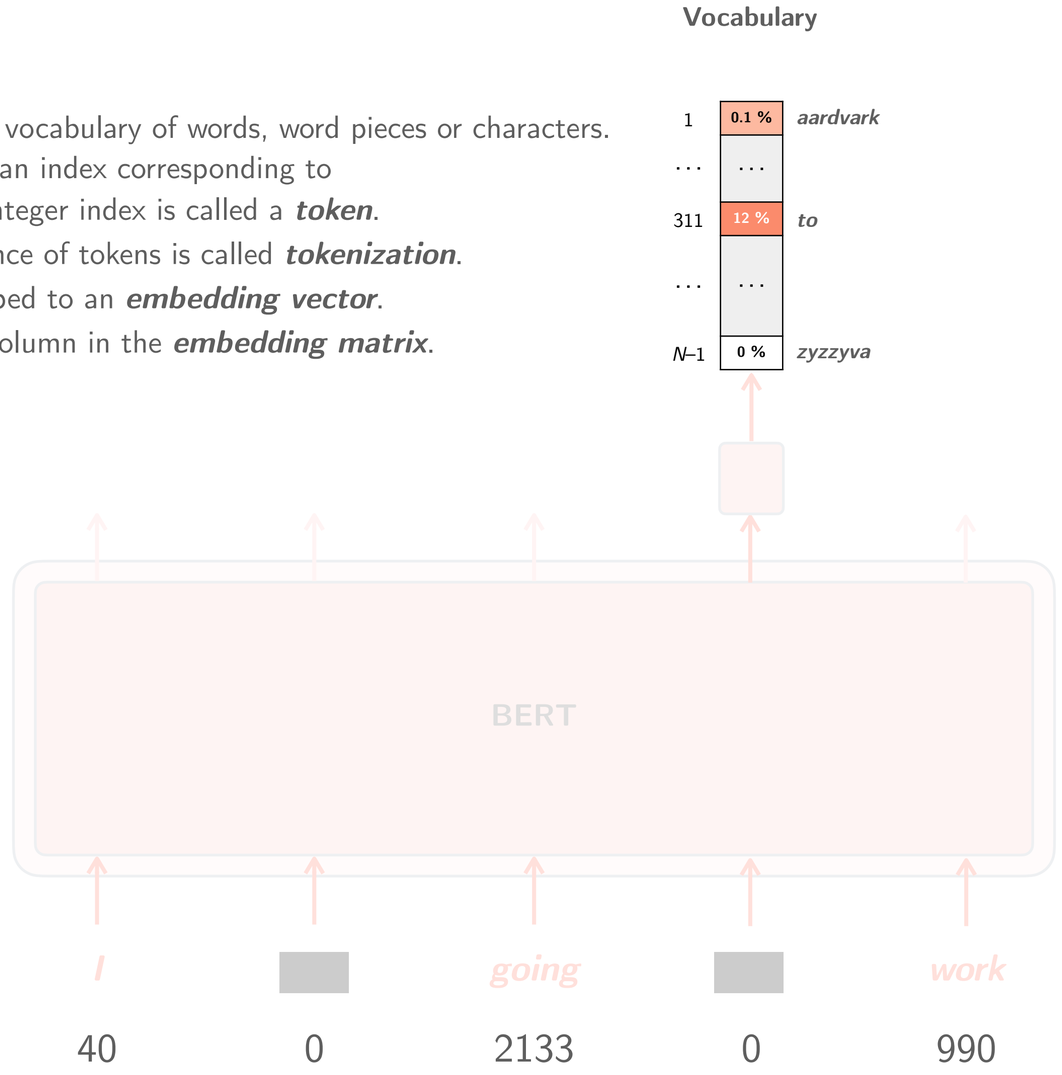
Part II

4

Tokenization


Vocabulary
1
0.1 %
aardvark
. . .
. . .
311
12 %
to
. . .
. . .
V–1
0 %
zyzzyva
BERT
I
going
work
40
0
2133
0
990

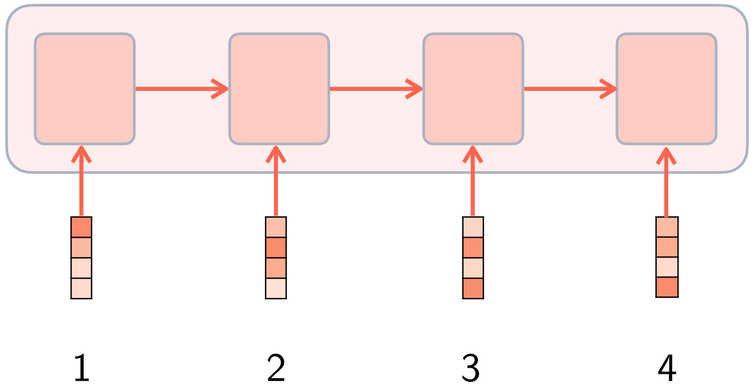
When training a transformer (or RNN), you define a vocabulary of words, word pieces or characters.
If the vocabulary has *N* entries, then each entry has an index corresponding to its position in the vocabulary, from 0 to *N*–1. This integer index is called a ***token***.
The process of mapping a body of text into a sequence of tokens is called ***tokenization***.
In a given transformer (or RNN), each token is mapped to an ***embedding vector***.
For token *i*, its embedding vector is simply the *i*-th column in the ***embedding matrix***.



5

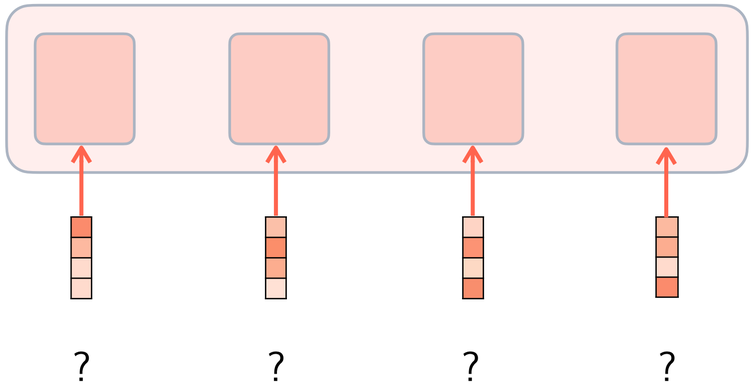
Positional encoding

Recurrent networks implicitly deal with positions



Transformers do not.

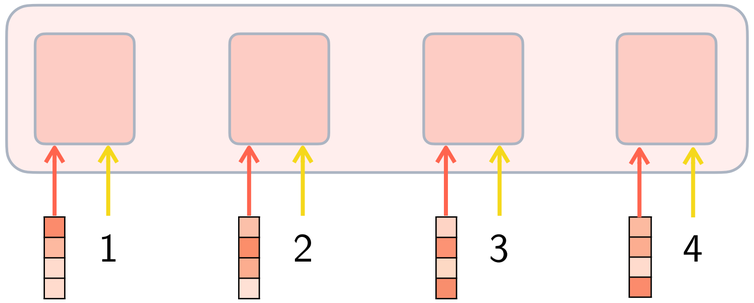
Attention is a *weighted sum* of vectors, hence the ordering of these vectors does not change the result.



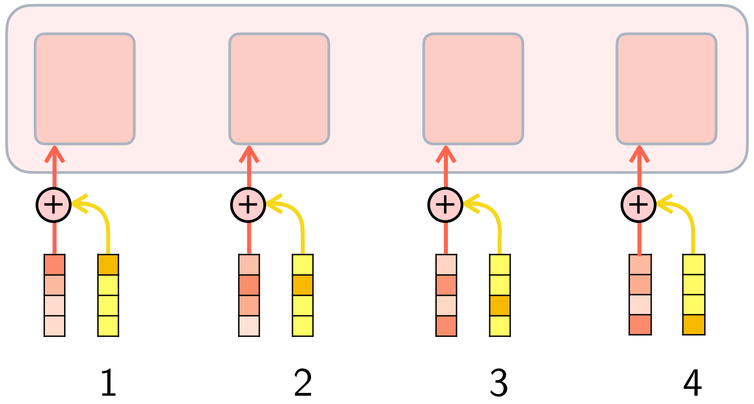
Transformers do not.

Attention is a *weighted sum* of vectors, hence the ordering of these vectors does not change the result.

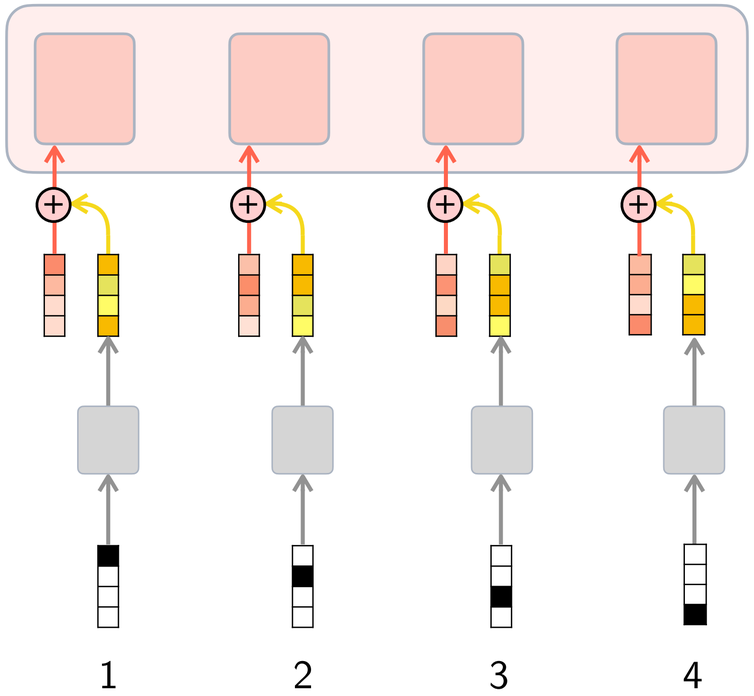
We need to tell the transformer what each input's position in the sequence is.



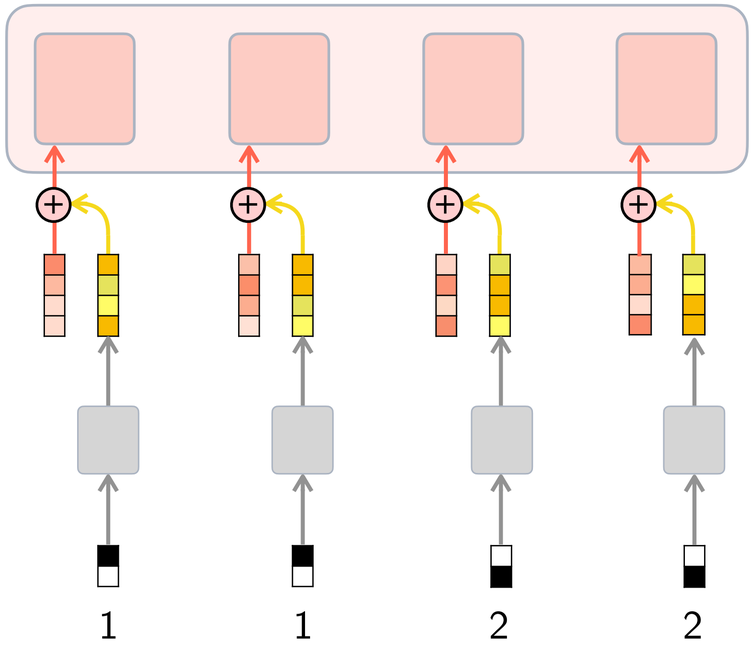
Sinusoidal encodings



Learned encodings



Learned encodings



e.g. modality encoding
(here: 2 possible modalities)

Thanks for your *attention*.

Blog posts:

- https://colah.github.io/posts/2015-08-Understanding-LSTMs
- https://jalammar.github.io/illustrated-transformer
- https://jalammar.github.io/illustrated-bert
- https://jalammar.github.io/illustrated-gpt2
- https://dugas.ch/artificial_curiosity/GPT_architecture.html

Papers:

- *Sequence-to-Sequence Learning with Neural Networks*
- *Neural Machine Translation by Jointly Learning to Align and Translate* (2016, Bahdanau, Cho & Bengio)
- *Attention Is All You Need* (2017, Vaswani et al.)
- *A Primer in BERTology* [...] (2020, Rogers et al.)
- *Google's Neural Machine Translation System* [...] (2016, Wu et al.)
- *Improving Language Understanding by Generative Pre-Training* (2017, Radford et al.)
- *Language Models are Unsupervised Multitask Learners* (2018, Radford et al.)
- *Language Models are Few-Shot Learners* (2020, Brown et al.)

Videos:

- *Illustrated Guide to Transformers Neural Network: A step by step explanation* (by Michael Phi), https://www.youtube.com/watch?v=4Bdc55j80l8
- *What are Transformer Neural Networks?* (by Ari Seff), https://www.youtube.com/watch?v=XSSTuhyAmnI
- *Attention Is All You Need* (by Yannic Kilcher), https://www.youtube.com/watch?v=iDulhoQ2pro
- *The Transformer neural network architecture EXPLAINED. "Attention is all you need" (NLP)* (by AI Coffee Break with Letitia) https://www.youtube.com/watch?v=FWFA4DGuzSc