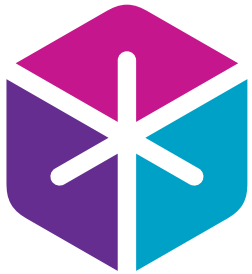
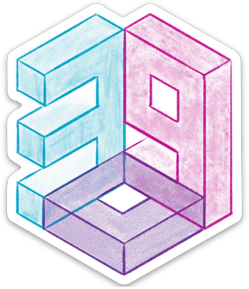




https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 9: Transformers

Shen Shen

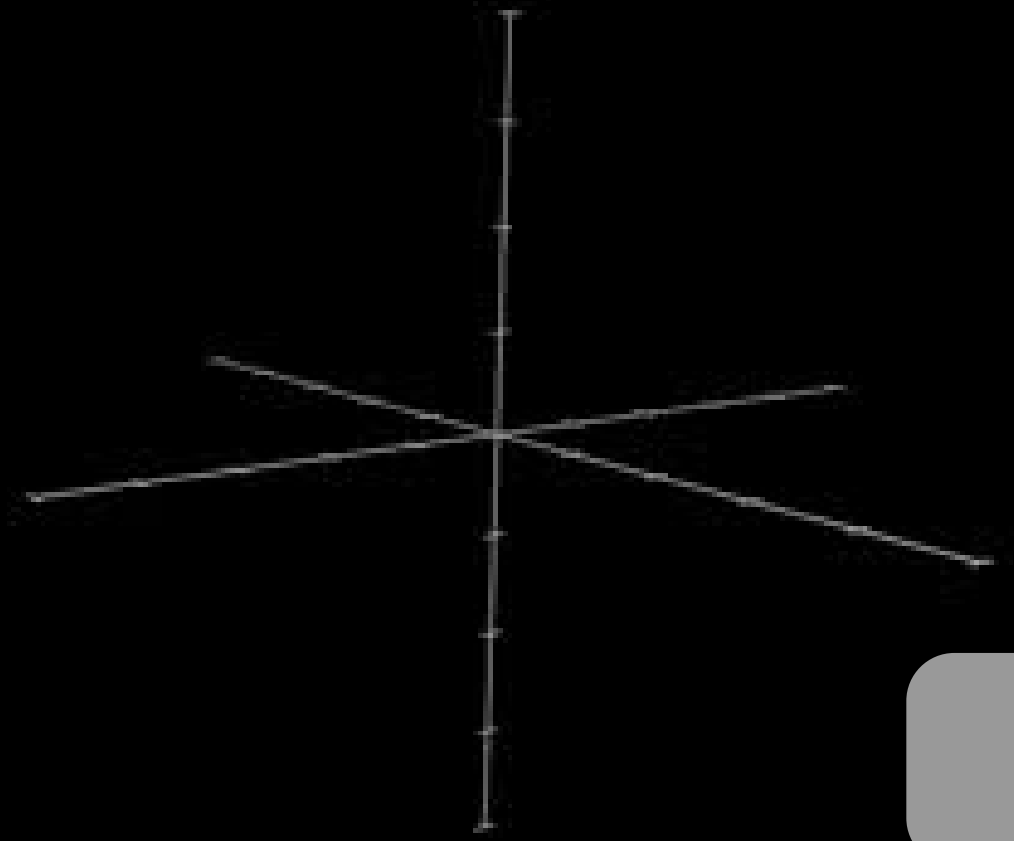
Apr 6, 2026

3pm, Room 10-250

Slides and Lecture Recording

Recap: **Word embedding**

Words → Vectors



[video edited from 3b1b]

Good word-embeddings space is equipped with *semantically meaningful vector arithmetic*

this enables "soft" dictionary look-up:

```
dict_en2fr = {
  "apple" : "pomme",
  "banana": "banane",
  "lemon" : "citron"}
```

Key Value

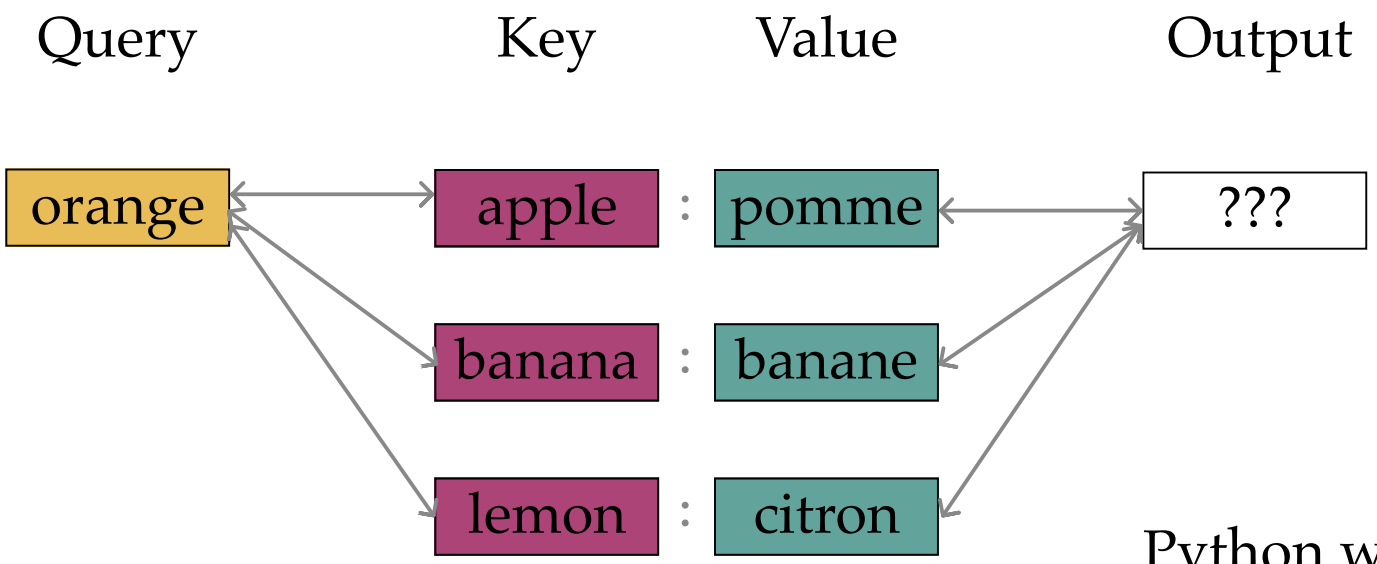
apple : pomme

banana : banane

lemon : citron

```python
dict_en2fr = {
  "apple" : "pomme",
  "banana": "banane",
  "lemon" : "citron"}

query = "orange"
output = dict_en2fr[query]
```

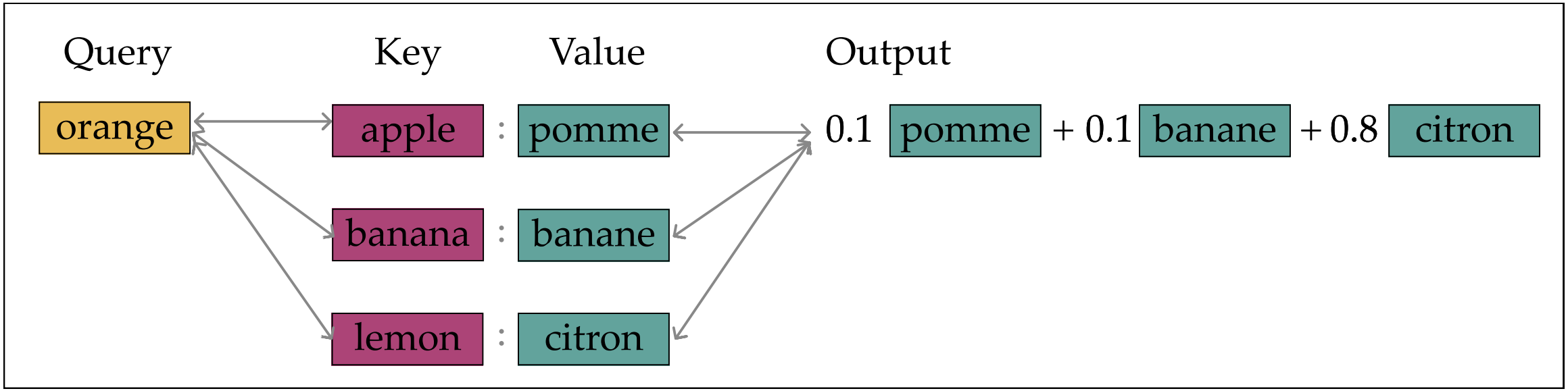


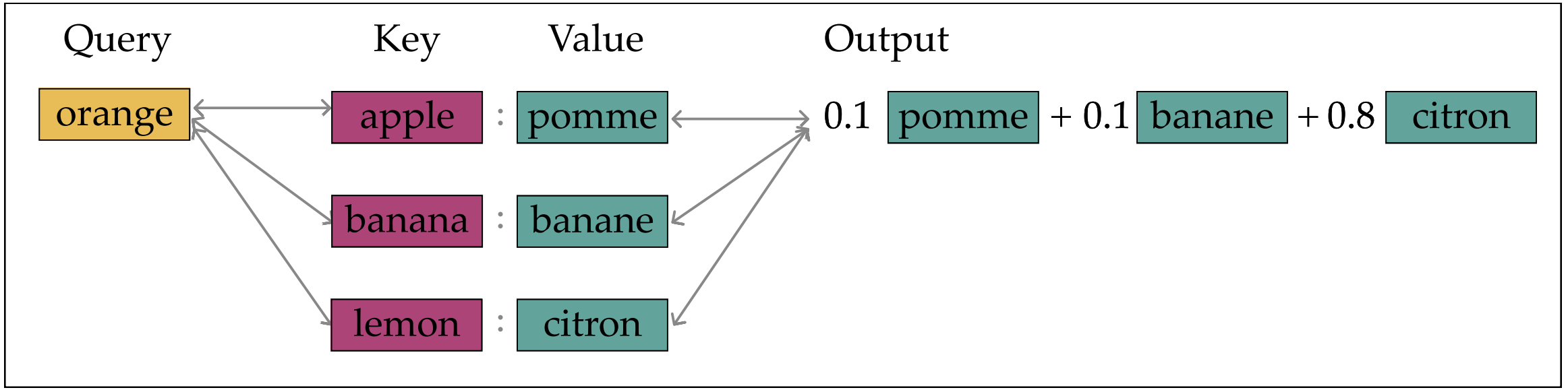
Python would complain. 🤯

```python
dict_en2fr = {
  "apple" : "pomme",
  "banana": "banane",
  "lemon" : "citron"}

query = "orange"
output = dict_en2fr[query]
```

But we can probably see the rationale behind something like this:



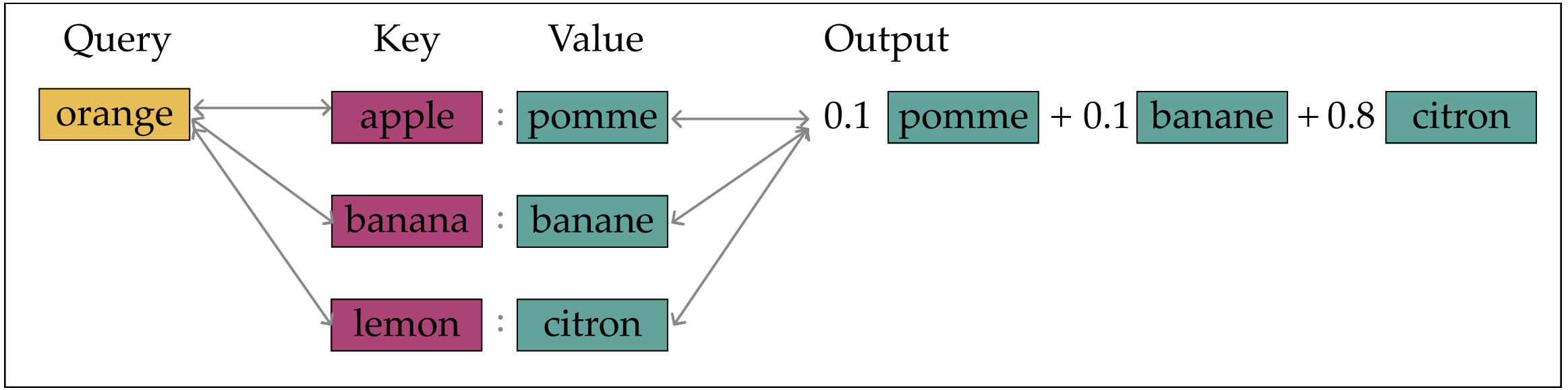


We put (query, key, value) in "good" embeddings in our human brain

such that mixing the values 0.1 pomme + 0.1 banane + 0.8 citron

via these mixing percentages [0.1 0.1 0.8] made sense

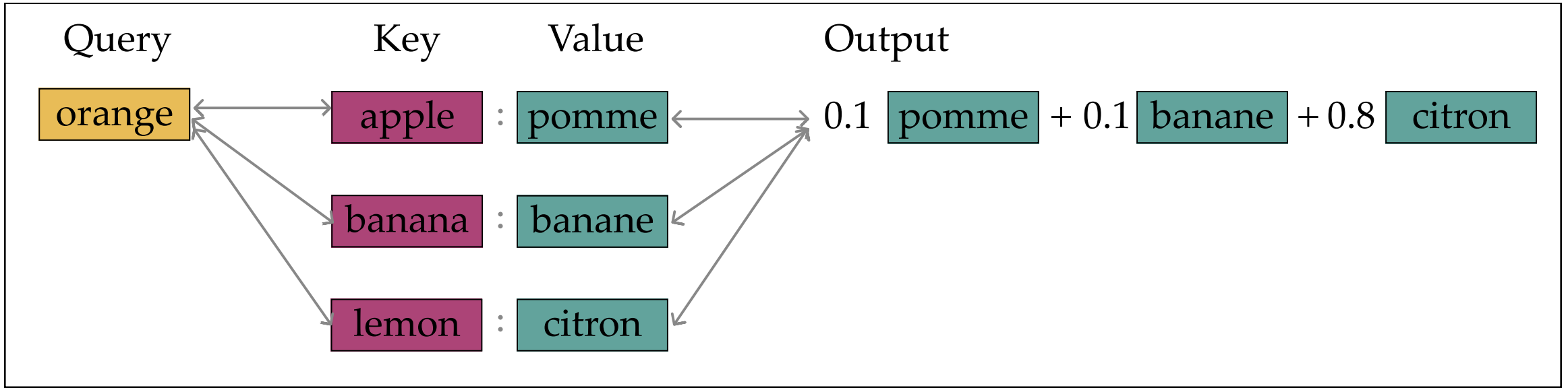
very roughly, the attention mechanism in transformers automates this process.



a. compare query and key for merging percentages:

$$\text{softmax}\left(\text{orange}\,\text{apple}, \text{orange}\,\text{banana}, \text{orange}\,\text{lemon}\right) = \begin{bmatrix} 0.1 & 0.1 & 0.8 \end{bmatrix}$$

dot-product similarity



a. compare query and key for merging percentages:

$$\text{softmax}\left(\text{orange}\cdot\text{apple},\ \text{orange}\cdot\text{banana},\ \text{orange}\cdot\text{lemon}\right) = \begin{bmatrix}0.1 & 0.1 & 0.8\end{bmatrix}$$

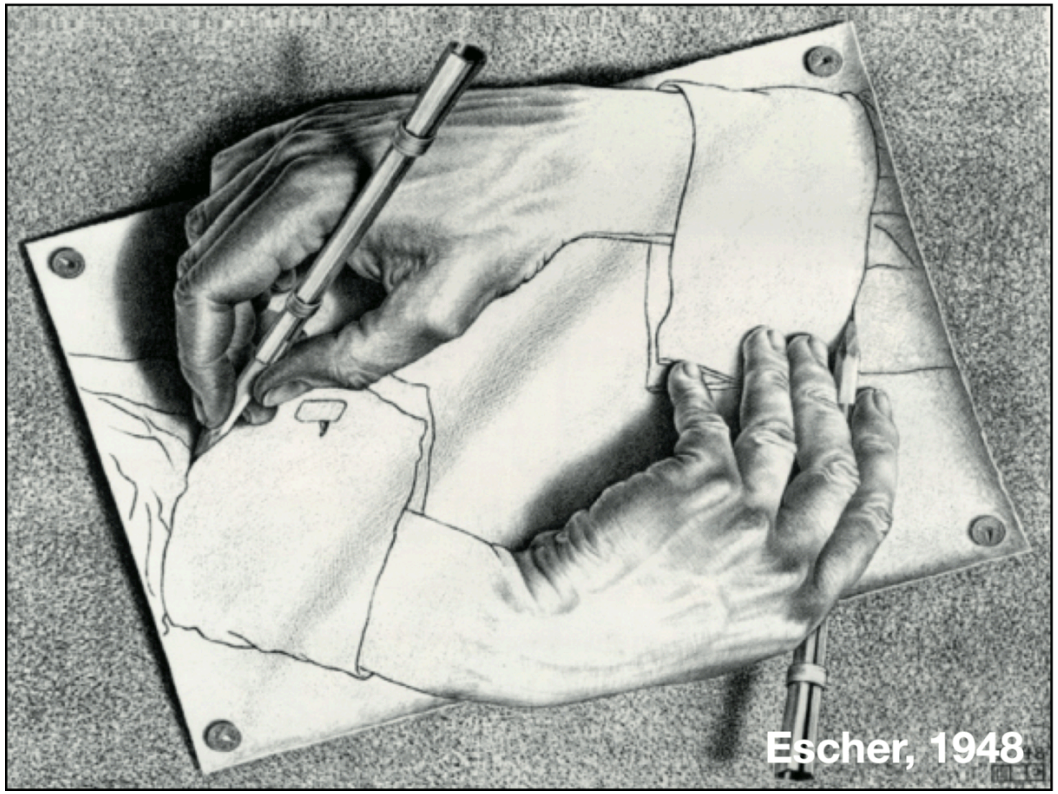
b. then output mixed values $0.1\,\text{pomme} + 0.1\,\text{banane} + 0.8\,\text{citron}$

Let's see how this intuition becomes a trainable mechanism.

Outline

- Transformers high-level intuition and architecture
- Attention mechanism
- Multi-head attention
- (Applications)

Large Language Models (LLMs) are trained in this self-supervised way

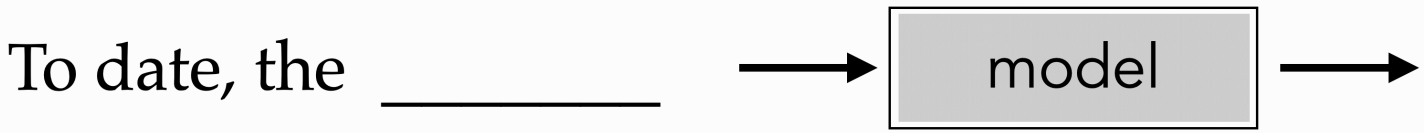


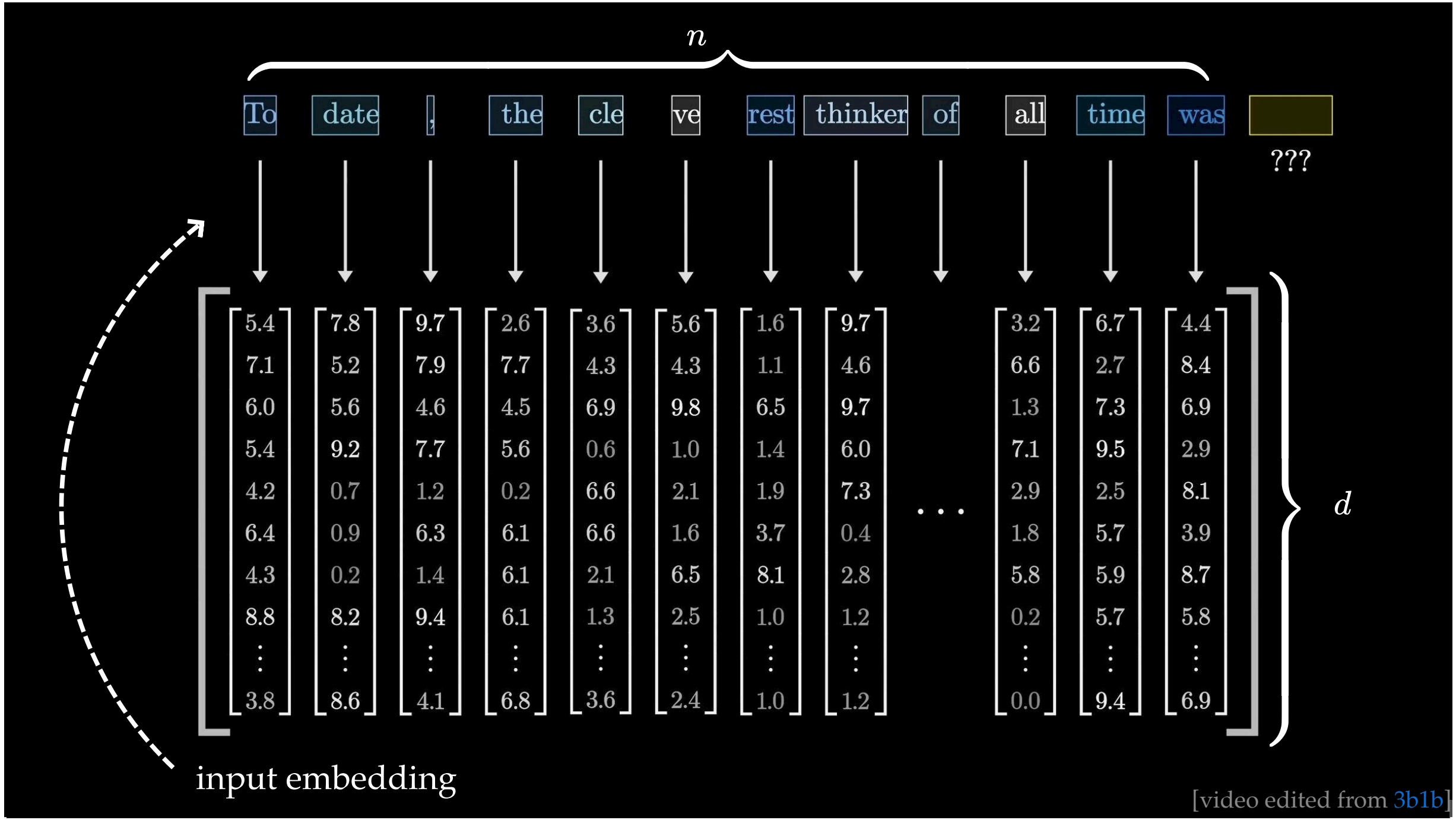
- Scrape the internet for plain texts.
- Cook up “labels” (prediction targets) from these texts.
- Convert “unsupervised” problem into “supervised” setup.

"To date, the cleverest thinker of all time was Issac. "

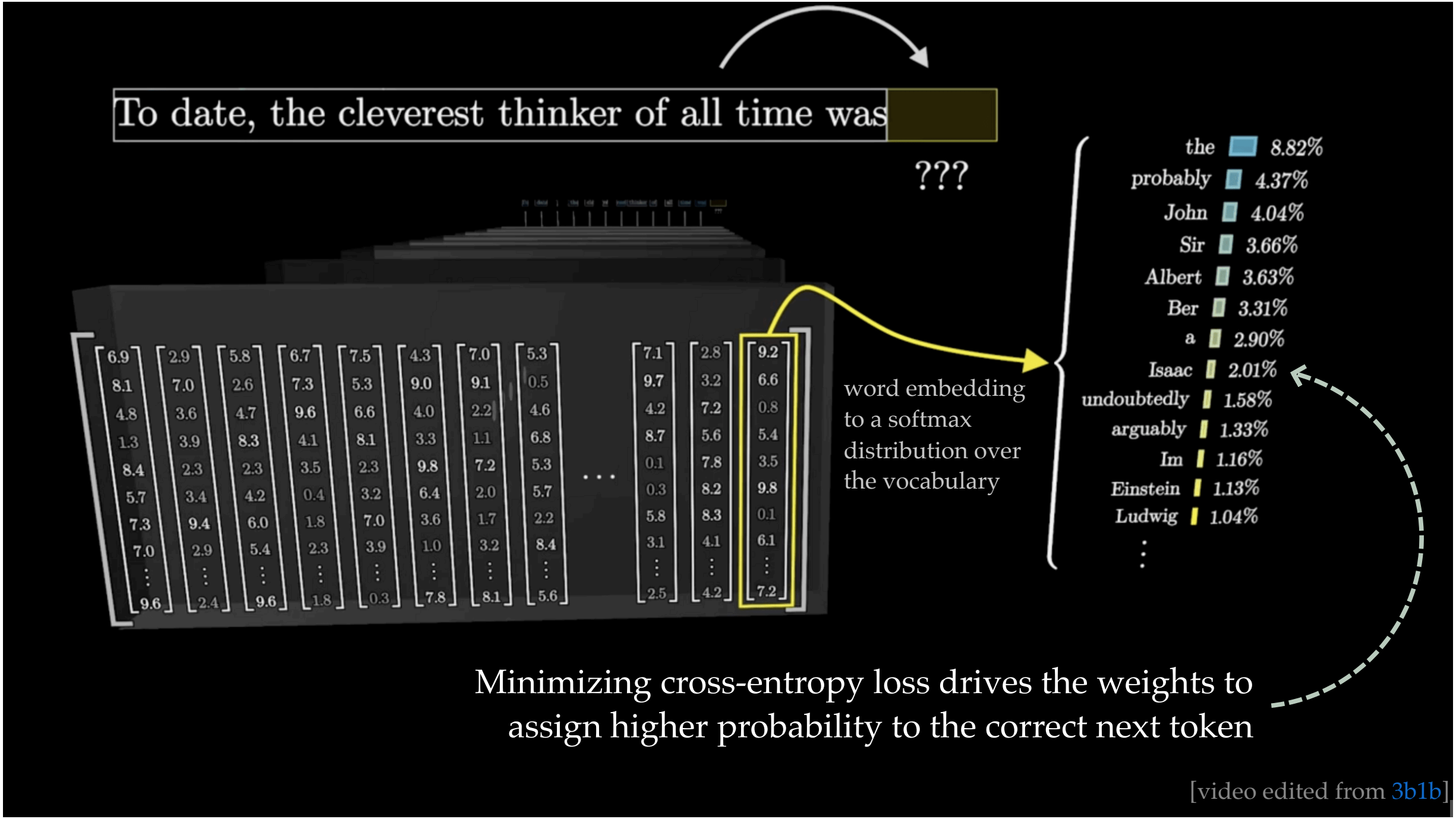
feature	label
To date, the	cleverest
To date, the cleverest	thinker
To date, the cleverest thinker	was
⋮	⋮
To date, the cleverest thinker of all time was	Issac

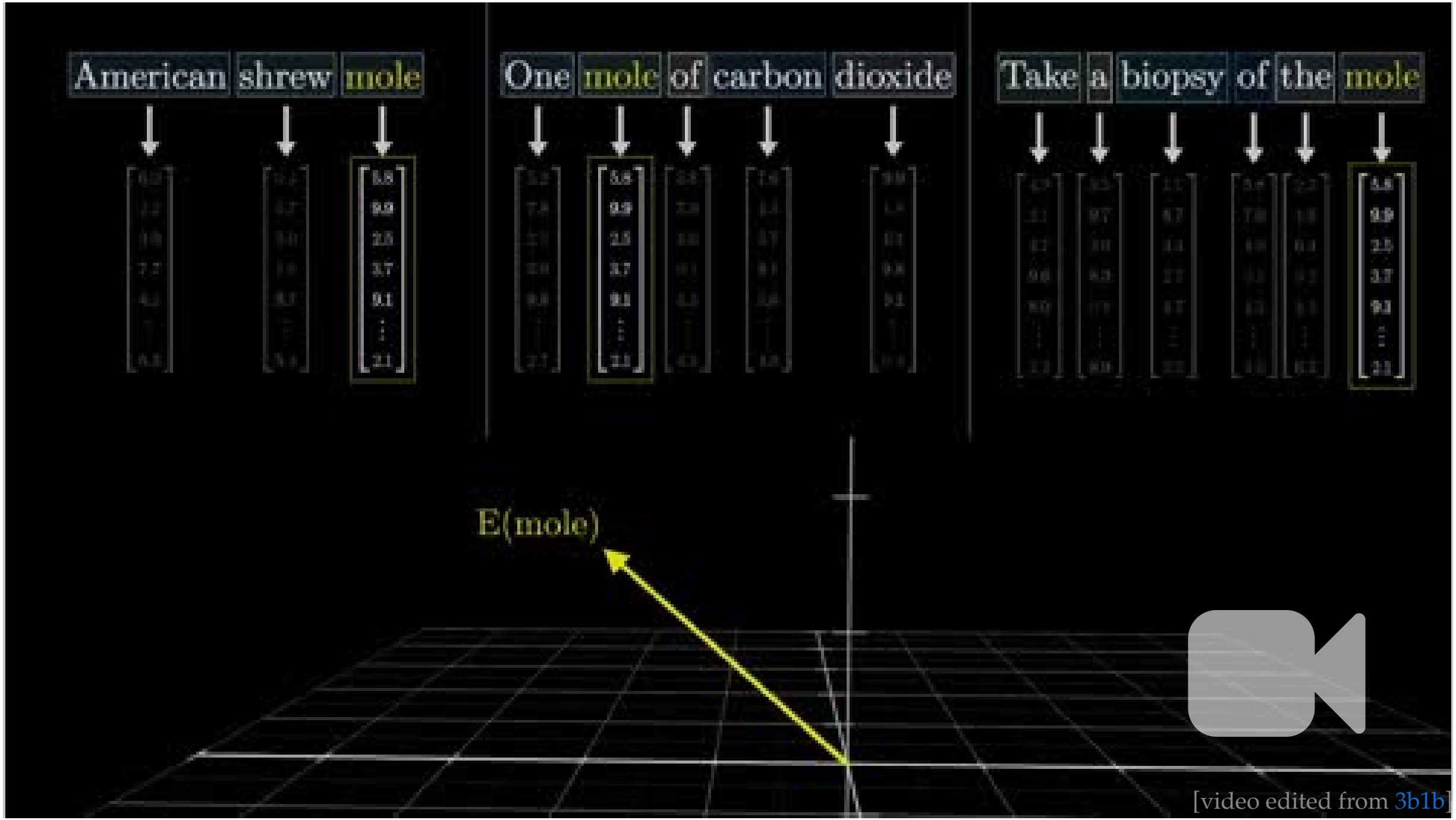
auto-regressive prediction





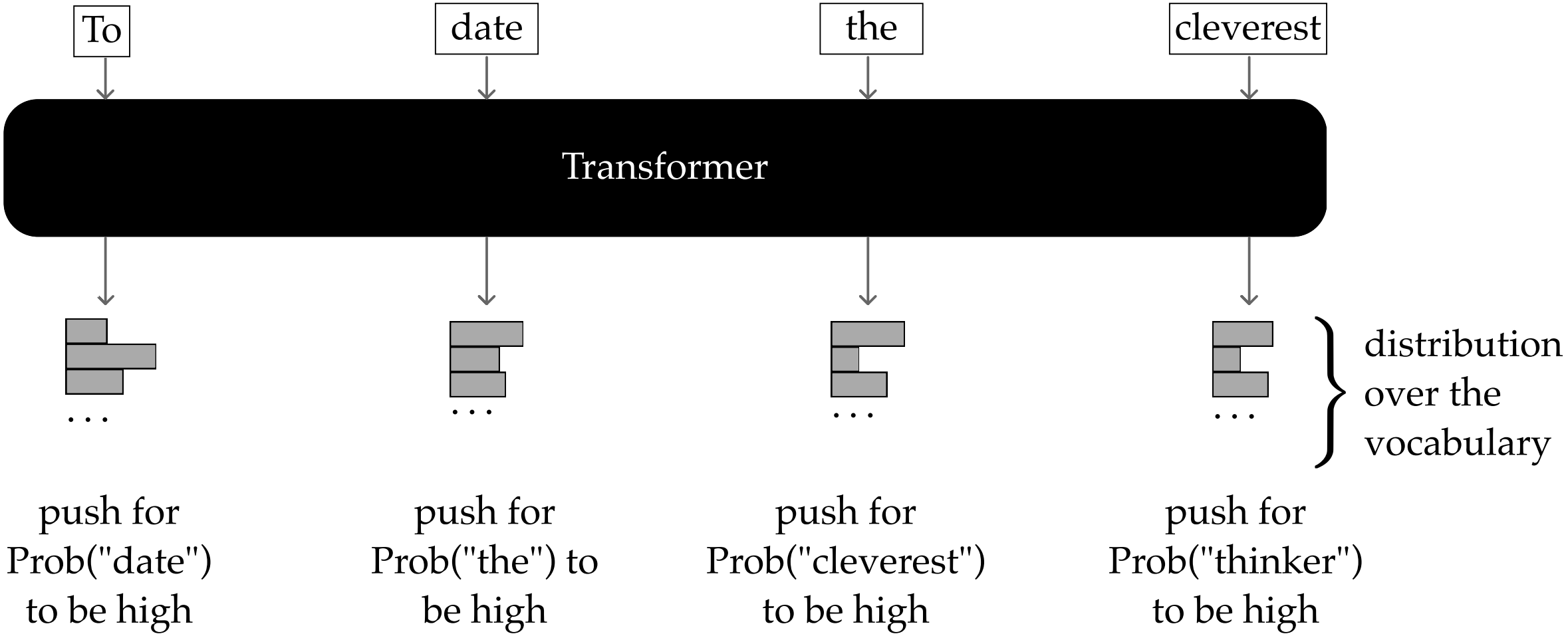
[video edited from 3b1b]

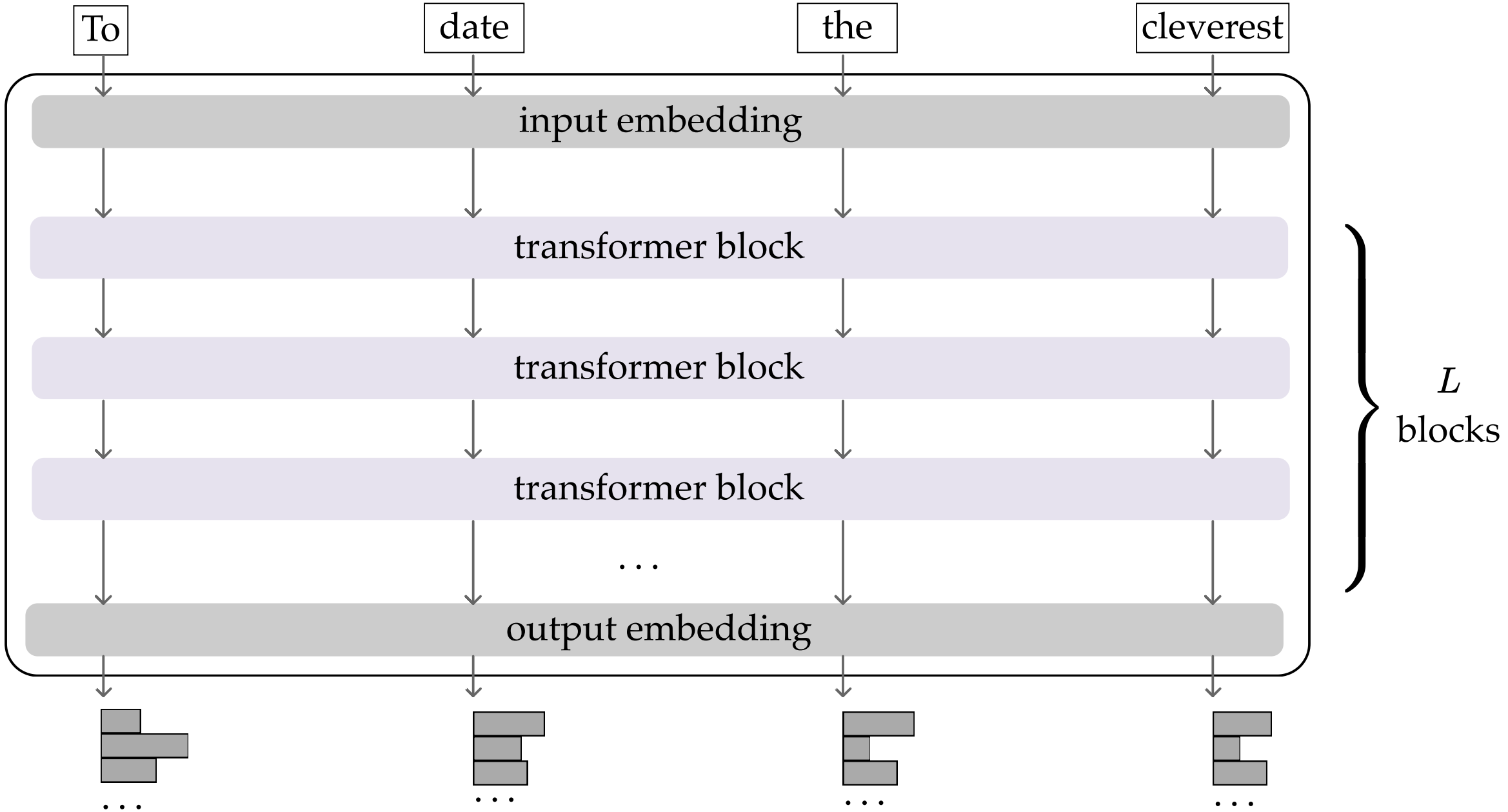

To date, the cleverest thinker of all time was
???
word embedding to a softmax distribution over the vocabulary
the 8.82%
probably 4.37%
John 4.04%
Sir 3.66%
Albert 3.63%
Ber 3.31%
a 2.90%
Isaac 2.01%
undoubtedly 1.58%
arguably 1.33%
Im 1.16%
Einstein 1.13%
Ludwig 1.04%
Minimizing cross-entropy loss drives the weights to assign higher probability to the correct next token
[video edited from 3b1b]

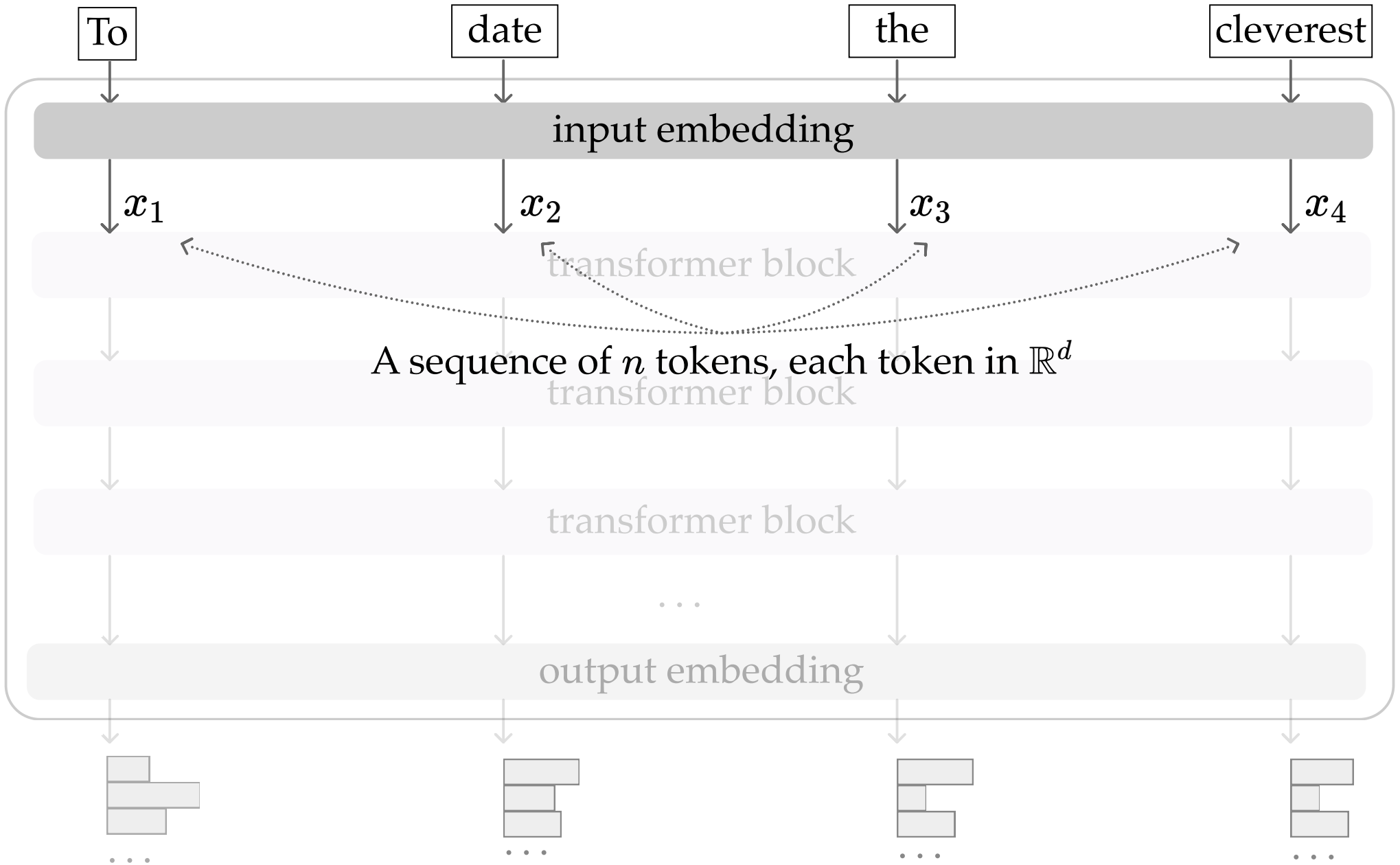


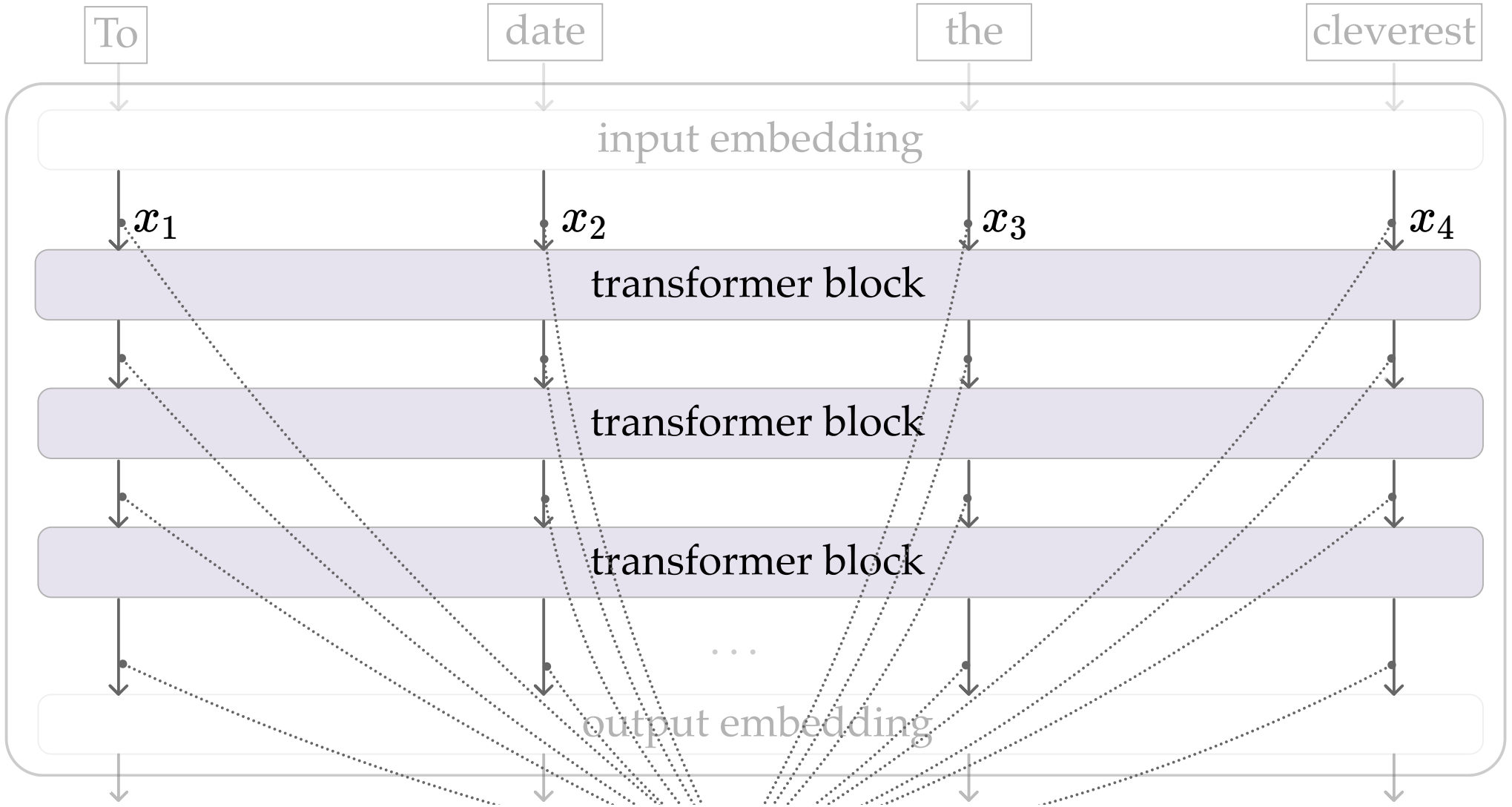
[video edited from 3b1b]

"To date, the cleverest [thinker] of all time was Issac.



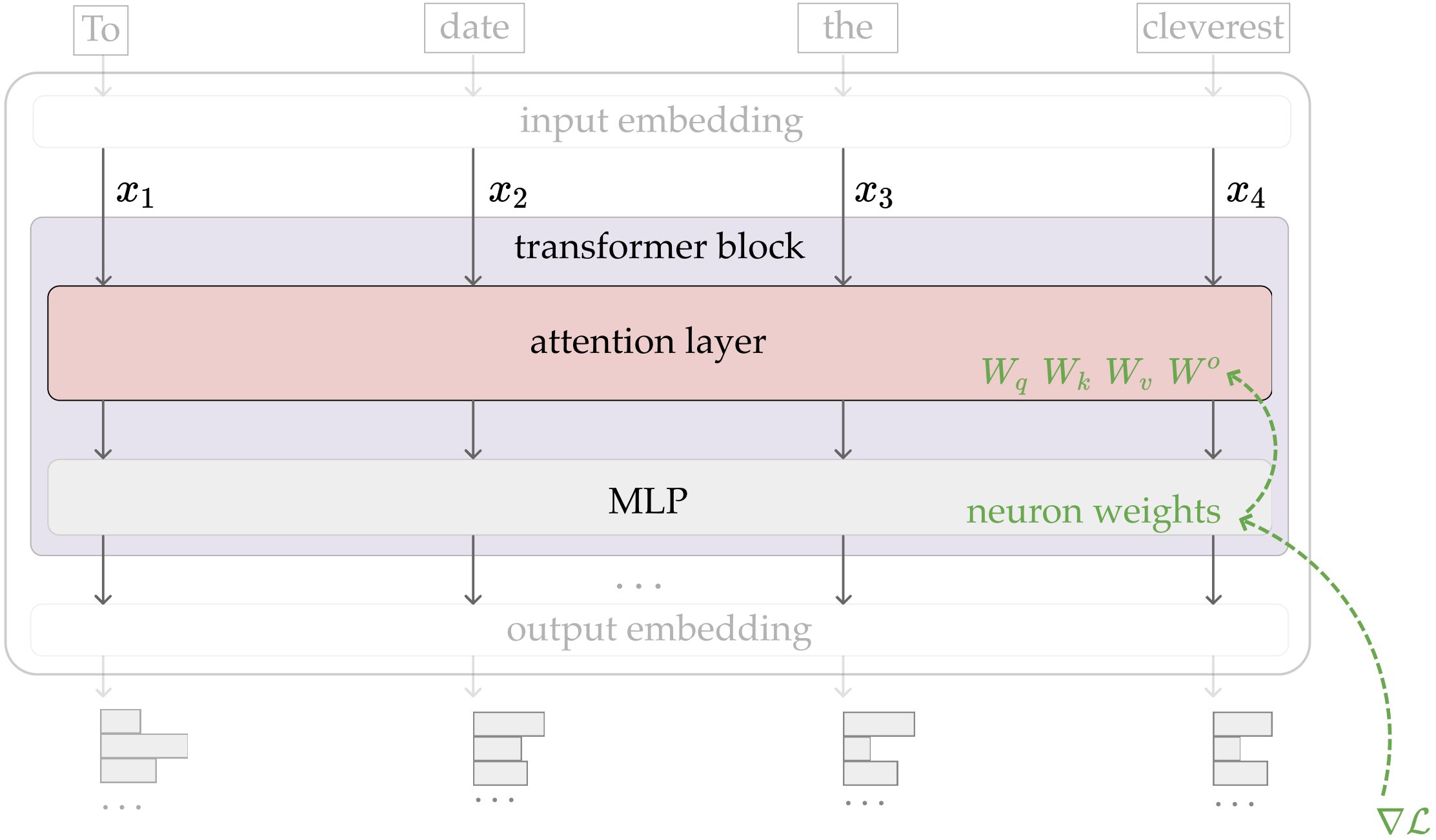

To
date
the
cleverest
input embedding
transformer block
transformer block
transformer block
. . .
output embedding
L
blocks

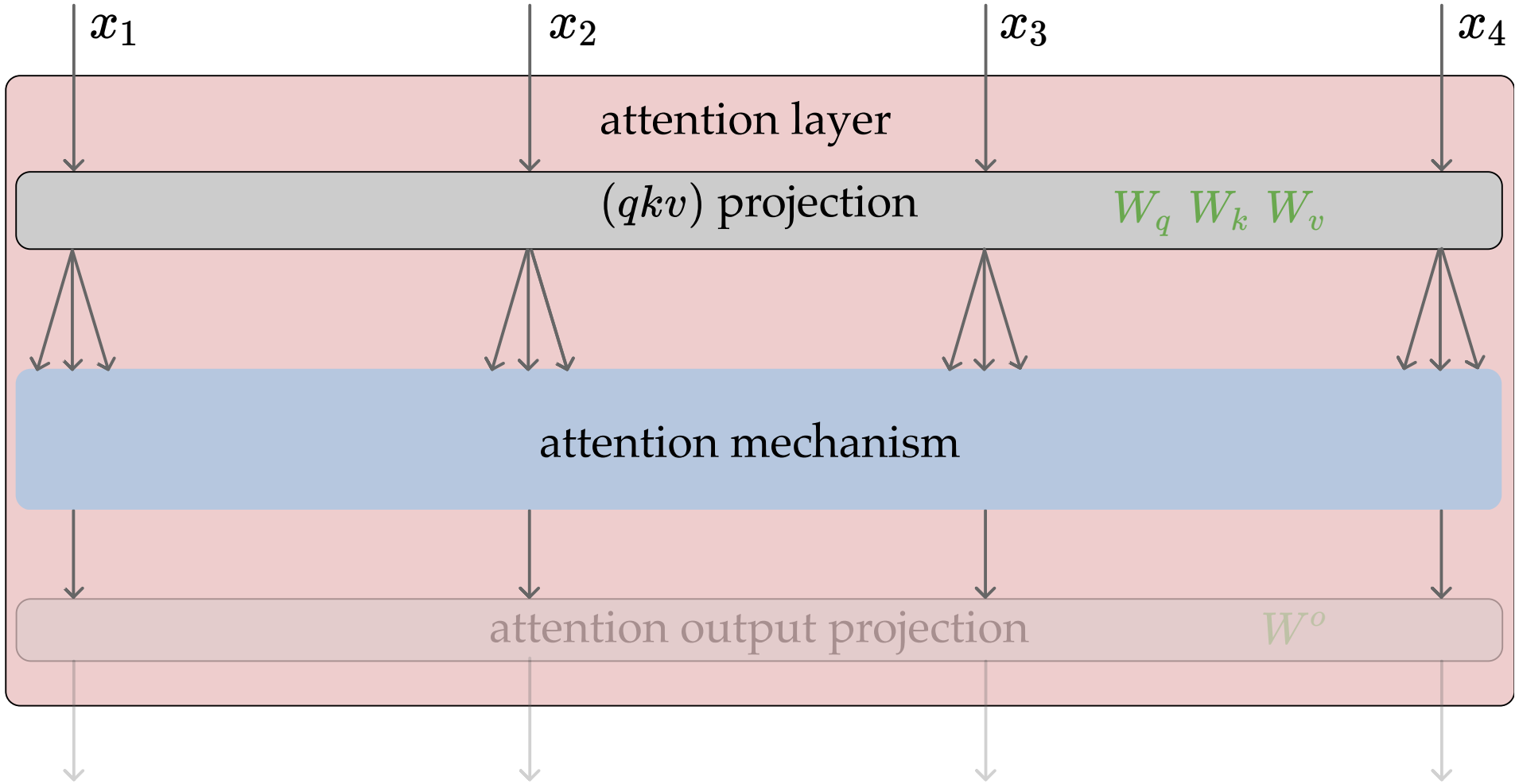

To
date
the
cleverest
input embedding
x_1
x_2
x_3
x_4
transformer block
A sequence of n tokens, each token in $\mathbb{R}^d$
transformer block
transformer block
. . .
output embedding



each of the n tokens transformed, block by block

within a shared d-dimensional word-embedding space.


To
date
the
cleverest
input embedding
x_1
x_2
x_3
x_4
transformer block
attention layer
W_q W_k W_v W^o
MLP
neuron weights
output embedding
$\nabla \mathcal{L}$



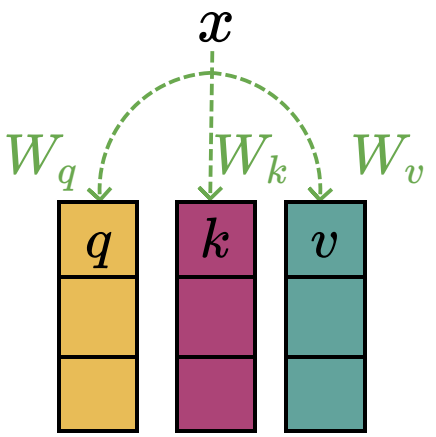
Most important bits in an attention layer:

1. (query, key, value) projection
2. attention mechanism

1. (query, key, value) projection

With *learned* projections, we frame x into:

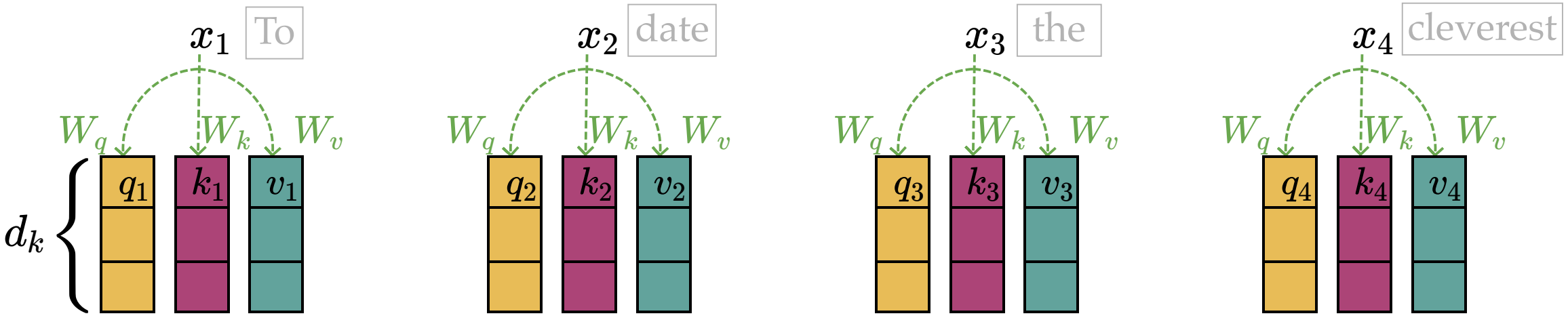
- a query to be the questions
- a key to be compared
- a value to contribute



Why learning these projections:

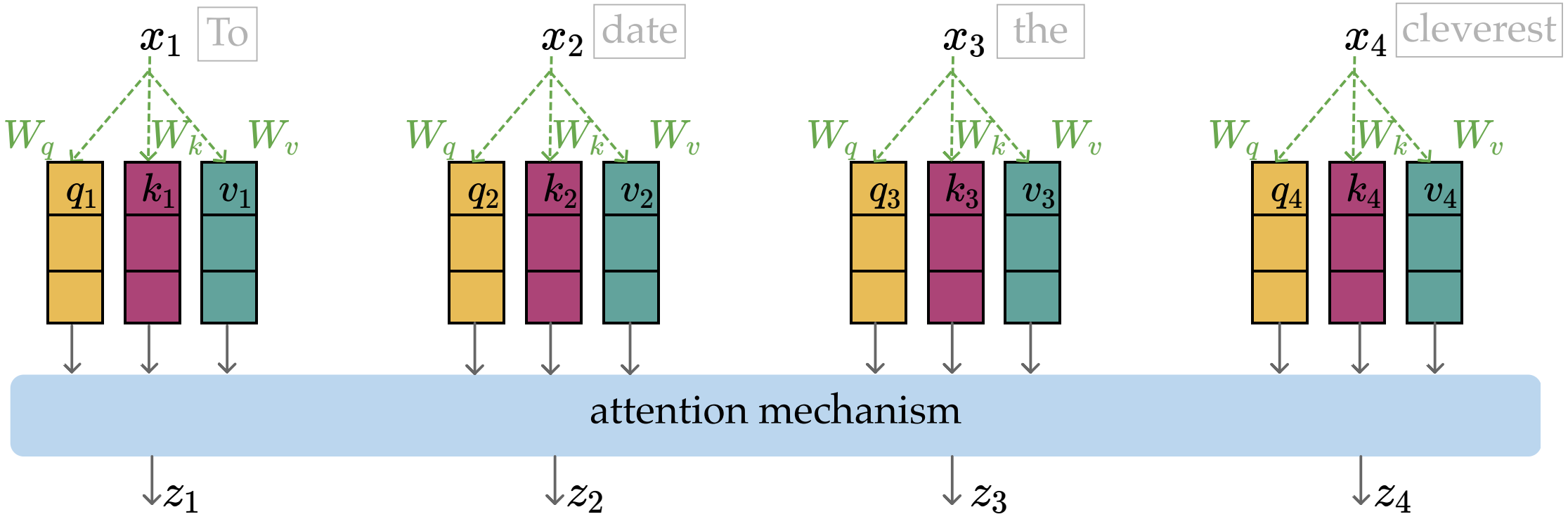
- W_q learns *how* to ask
- W_k learns *how* to listen
- W_v learns *how* to speak

1. (query, key, value) projection



- W_q, W_k, W_v, all in $\mathbb{R}^{d \times d_k}$
- project word embedding x from d-dimensional space to d_k-dimensional (q, k, v) spaces (typically $d_k < d$)
- $q_i = W_q^T x_i,\ k_i = W_k^T x_i,\ v_i = W_v^T x_i,\ \forall i$ — ***weight sharing*** across positions
- ***parallel*** and ***structurally identical*** processing

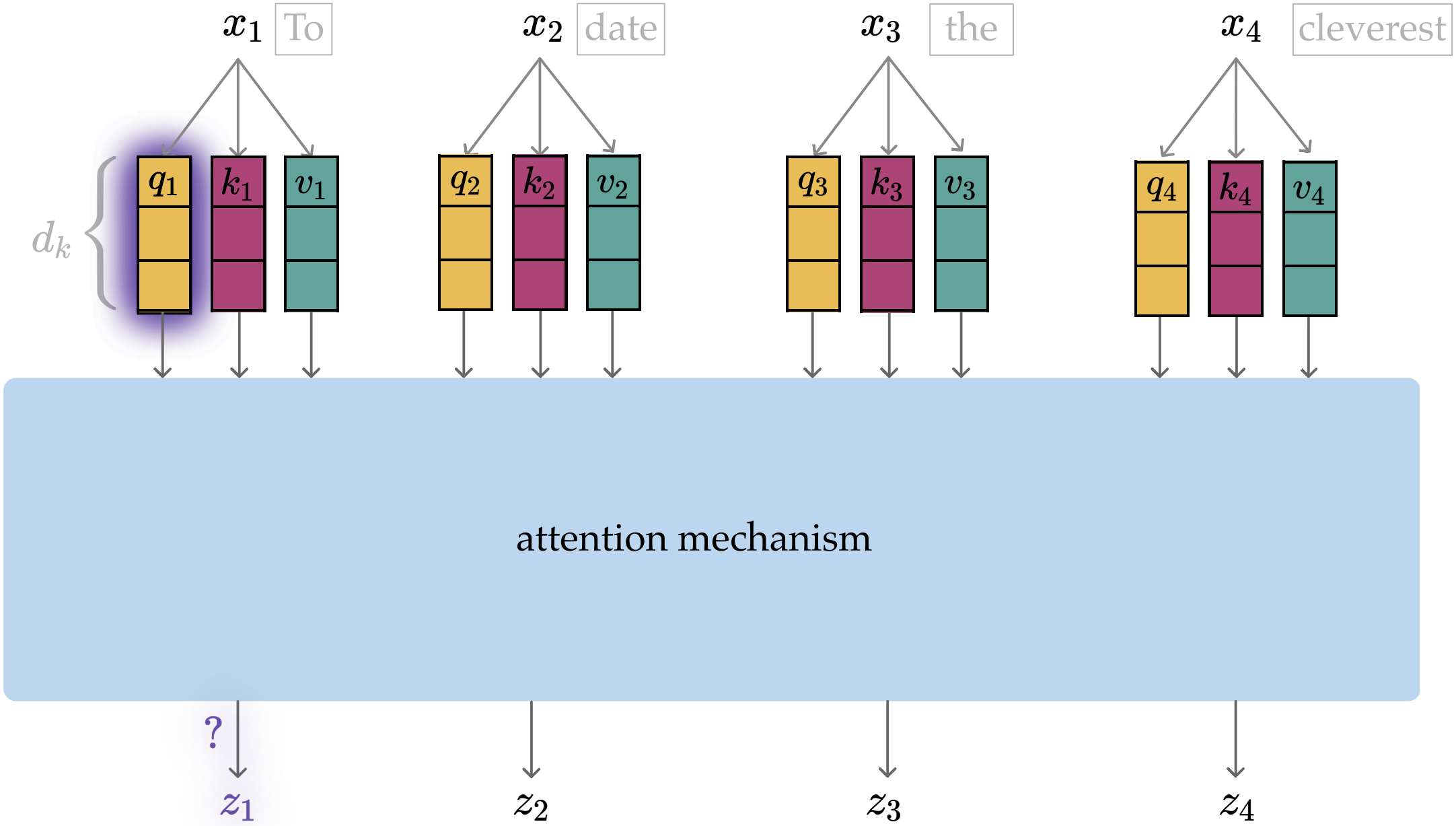
2. Attention mechanism

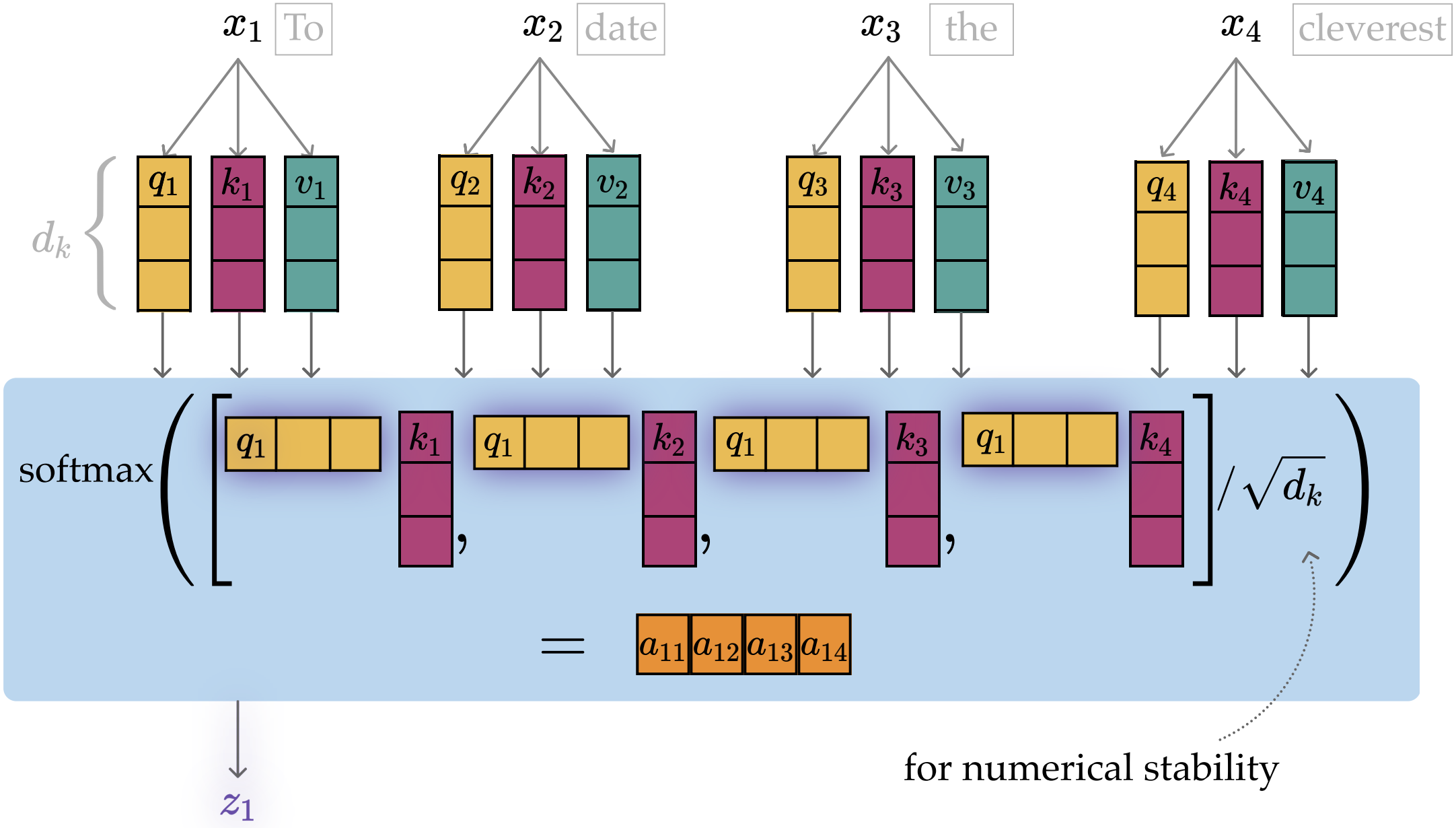


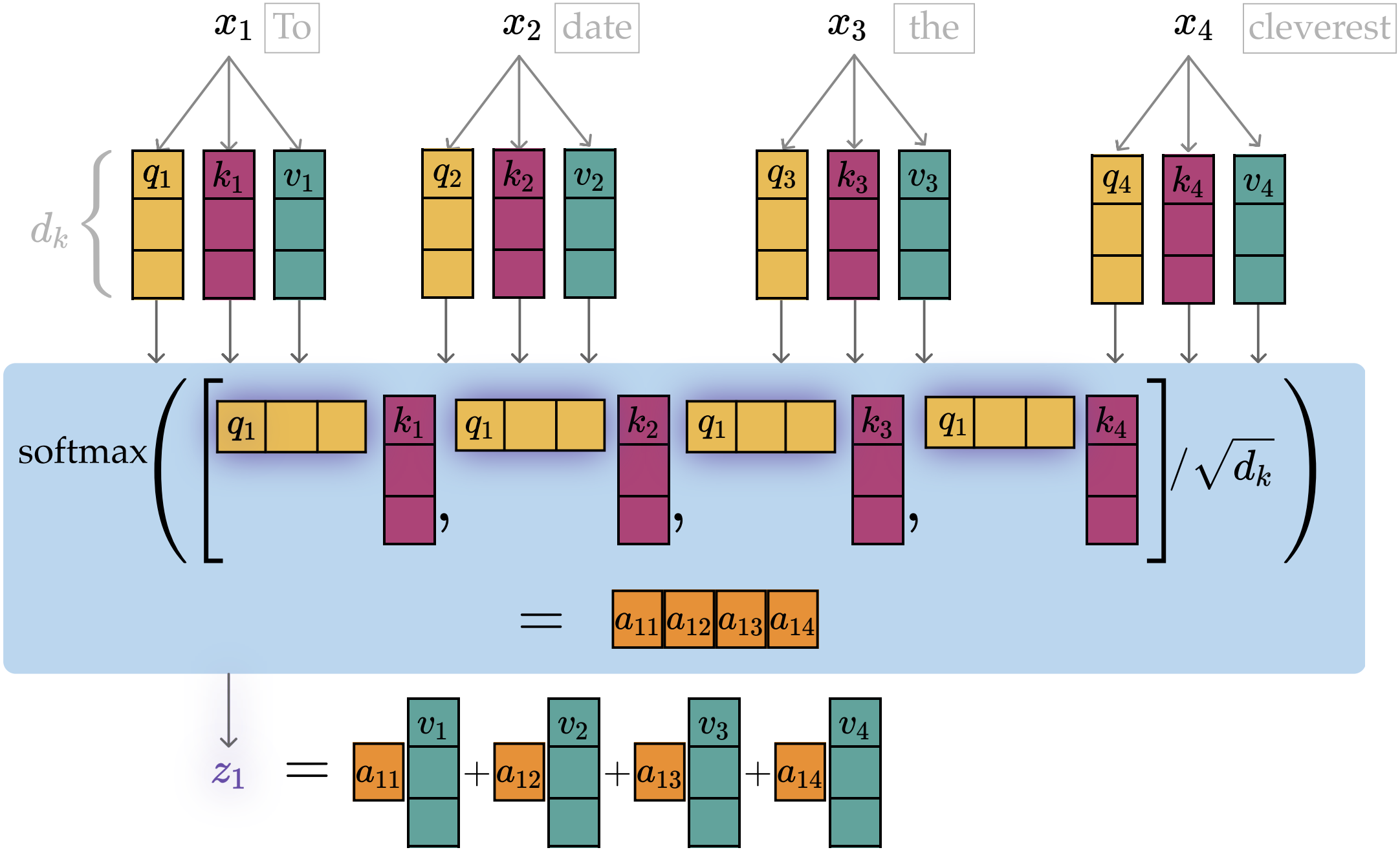
- Attention mechanism turns the projected (q, k, v) into z
- Each z is context-aware: a mixture of everyone's values, weighted by relevance

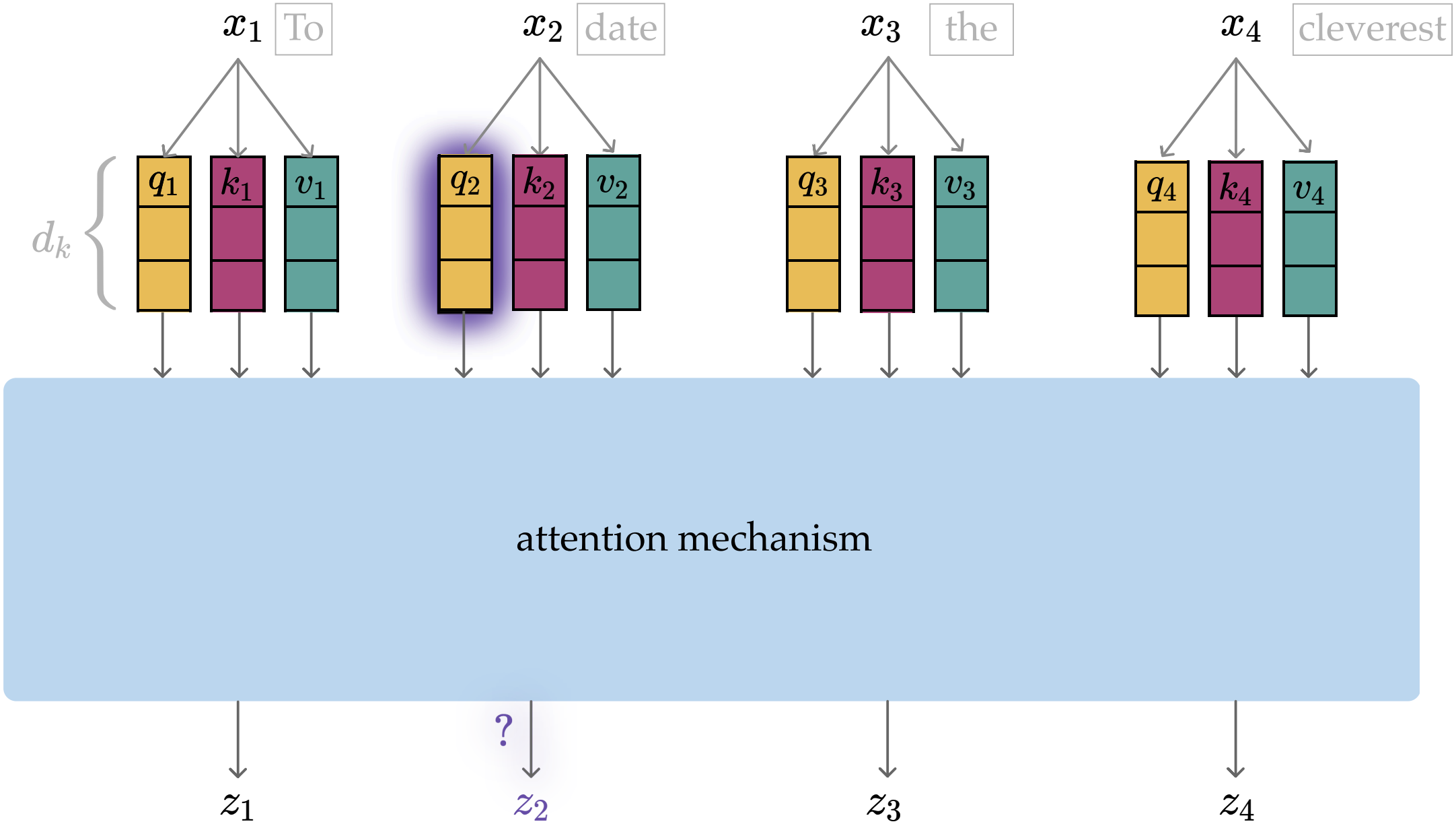
Outline

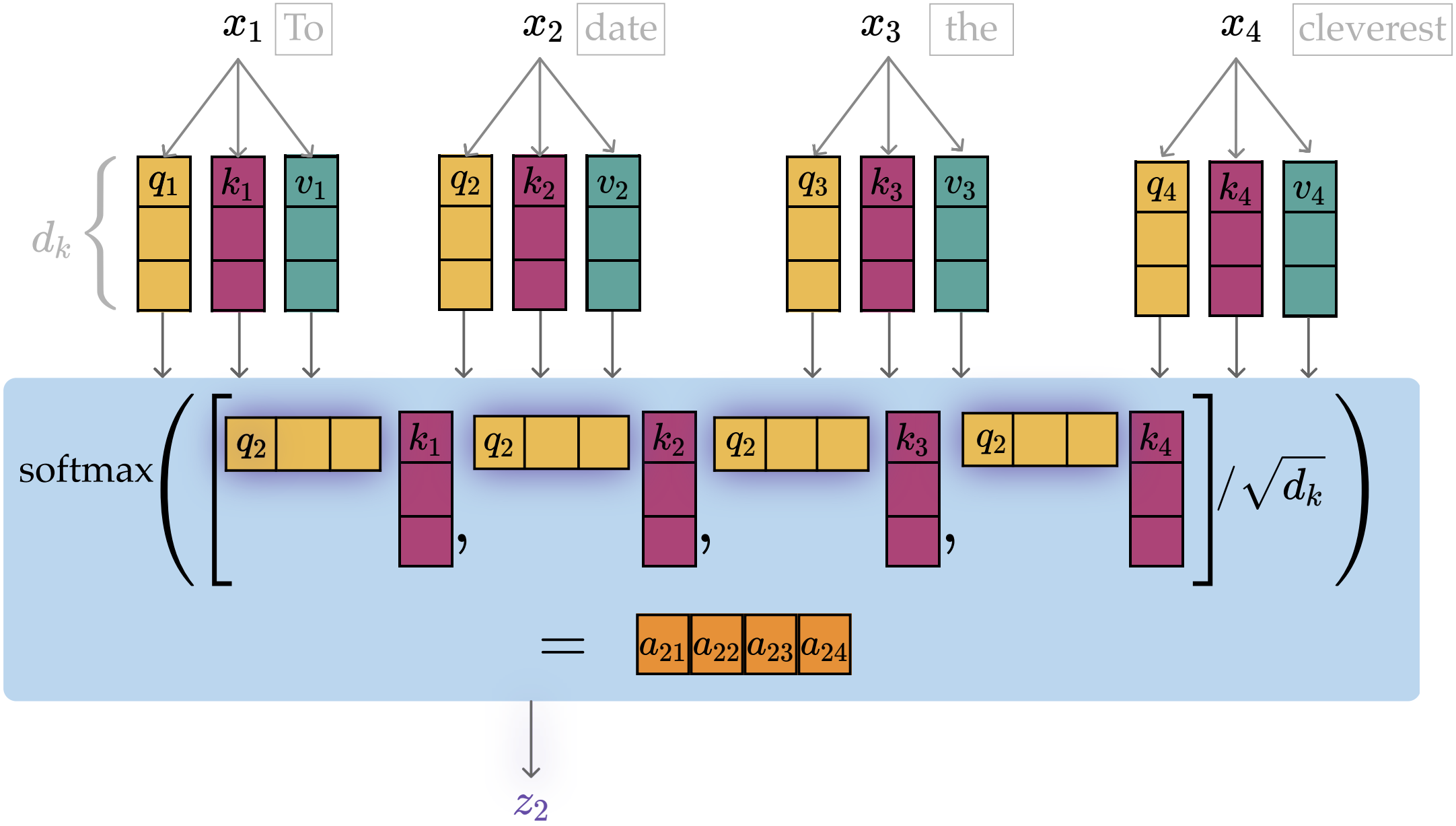
- Transformers high-level intuition and architecture
- Attention mechanism
- Multi-head attention
- (Applications)

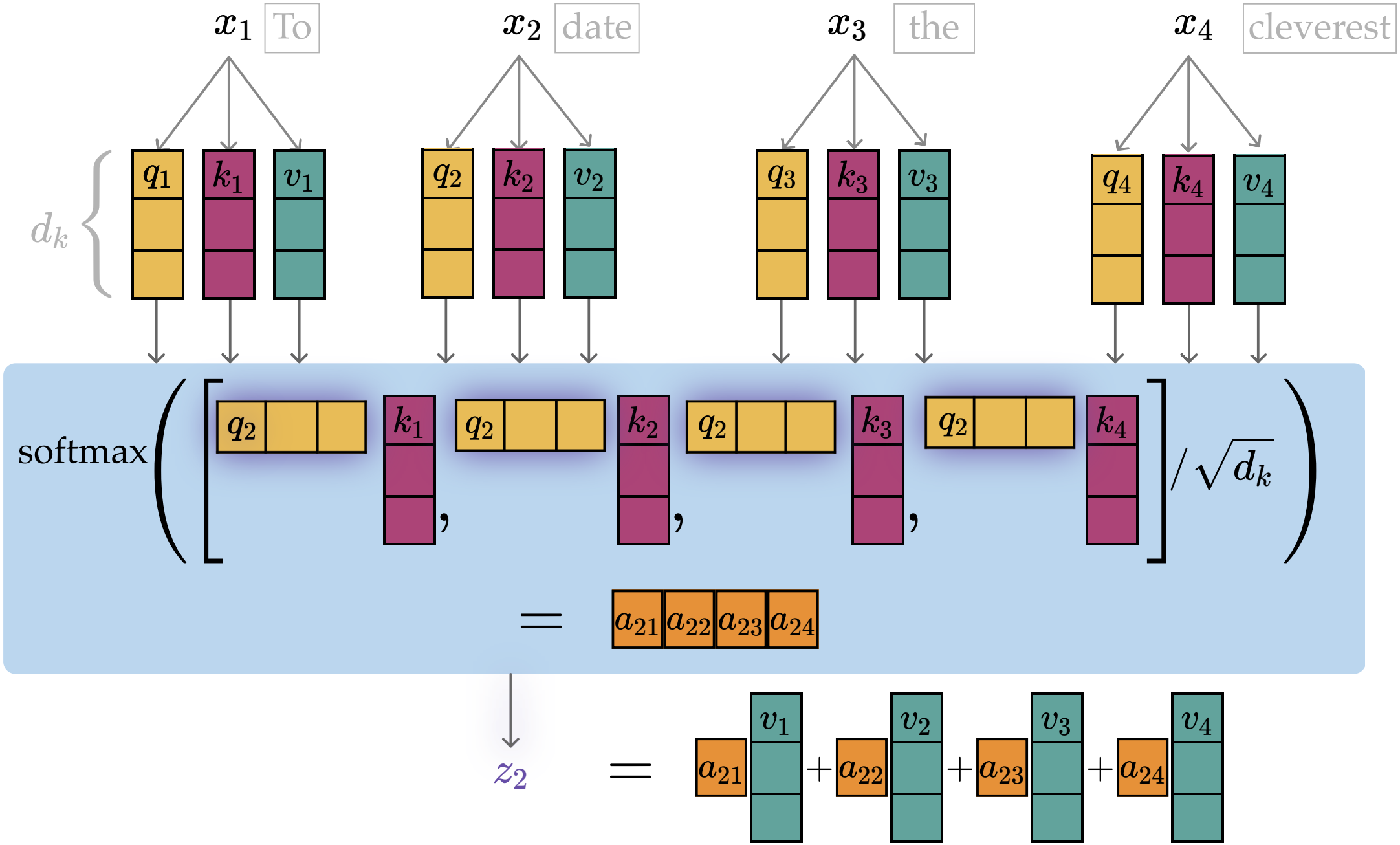

x_1 To
x_2 date
x_3 the
x_4 cleverest
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
attention mechanism
?
z_1
z_2
z_3
z_4

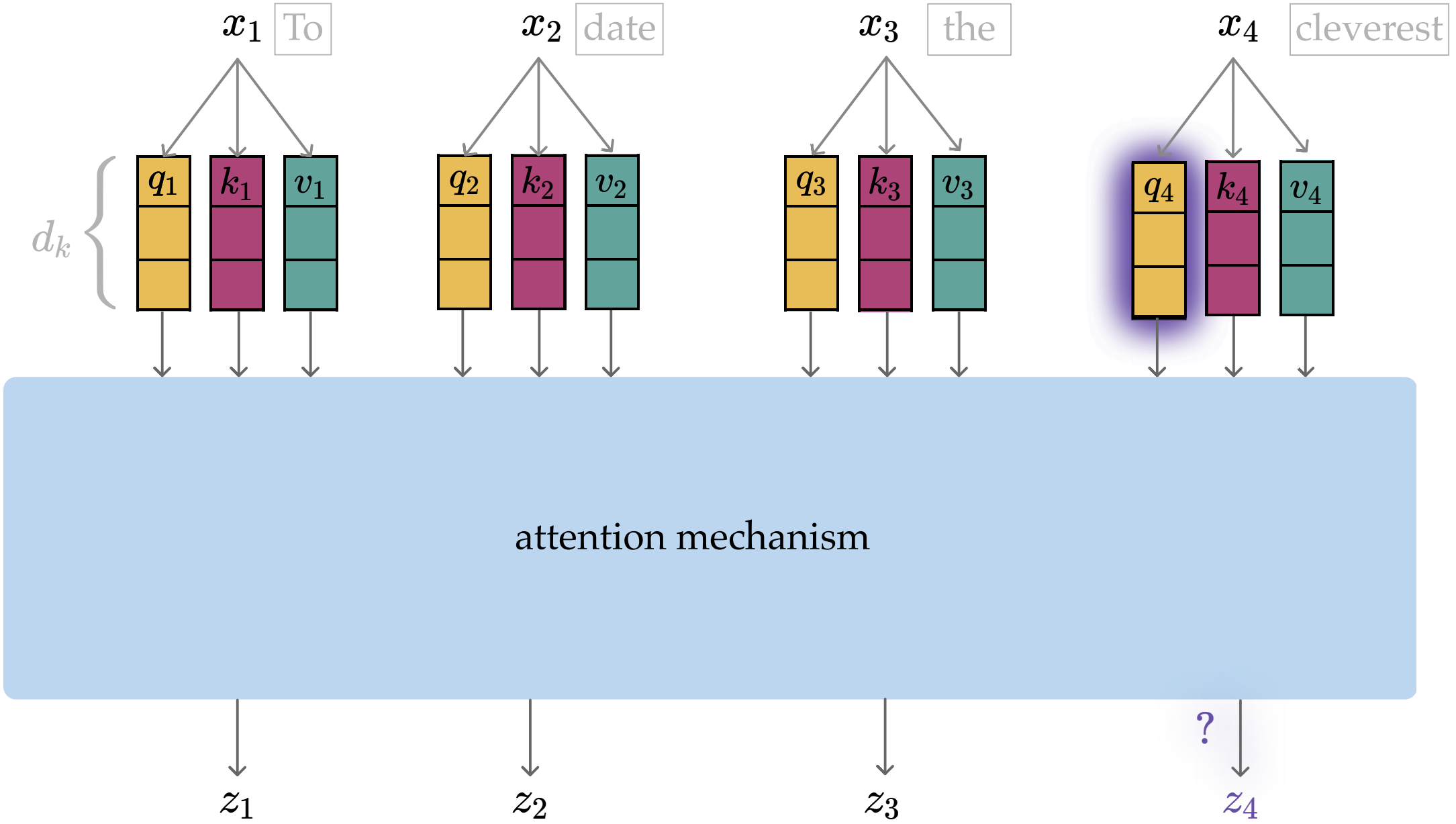

x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
$/\sqrt{d_k}$
a_{11} a_{12} a_{13} a_{14}
z_1
for numerical stability

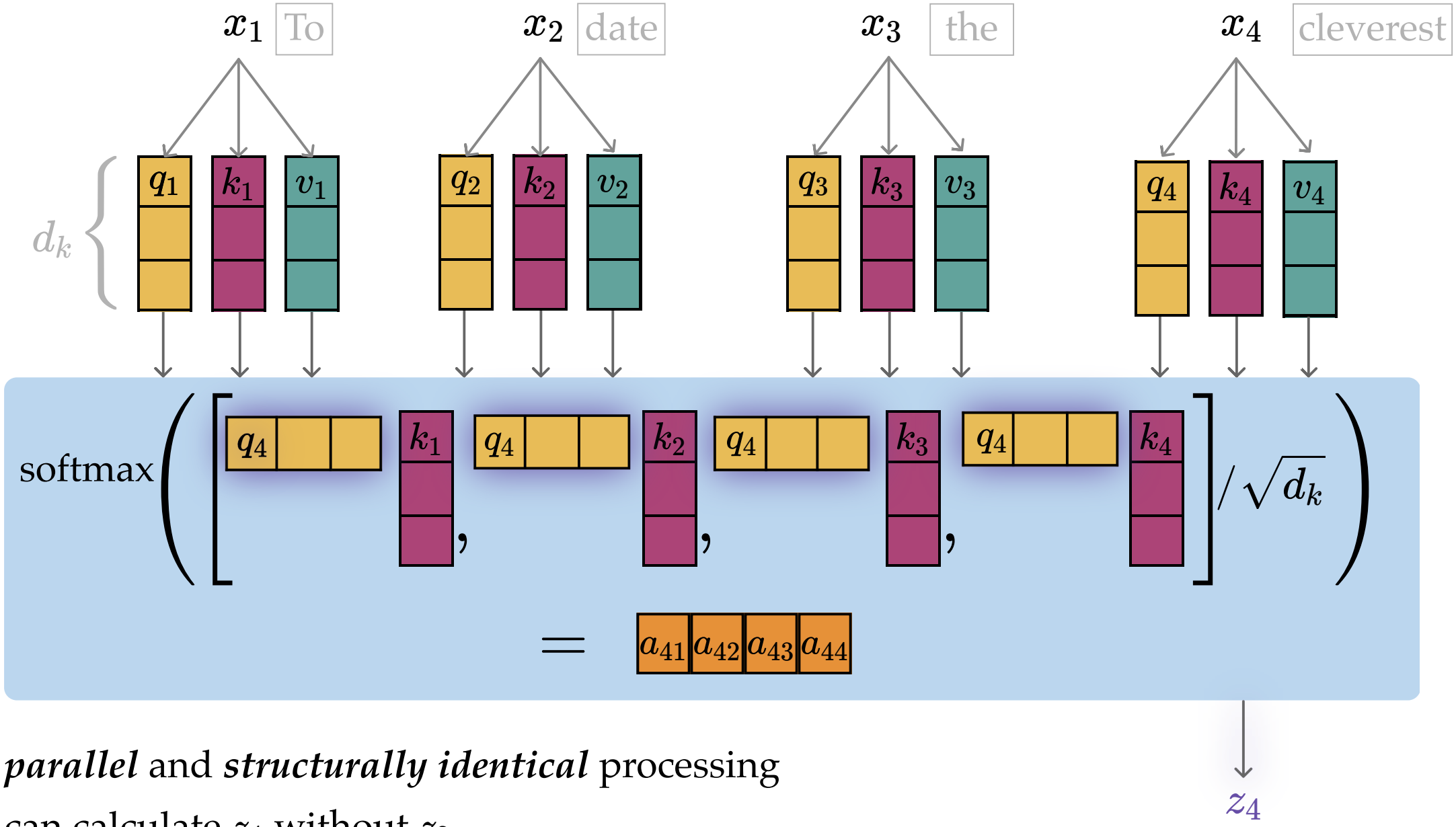

x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
$/\sqrt{d_k}$
$=$
a_{11} a_{12} a_{13} a_{14}
z_1 $=$
a_{11} v_1 + a_{12} v_2 + a_{13} v_3 + a_{14} v_4


x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
?
z_1
z_2
z_3
z_4


x_1 To
x_2 date
x_3 the
x_4 cleverest
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
q_2 k_1 q_2 k_2 q_2 k_3 q_2 k_4
$/\sqrt{d_k}$
=
a_{21} a_{22} a_{23} a_{24}
z_2

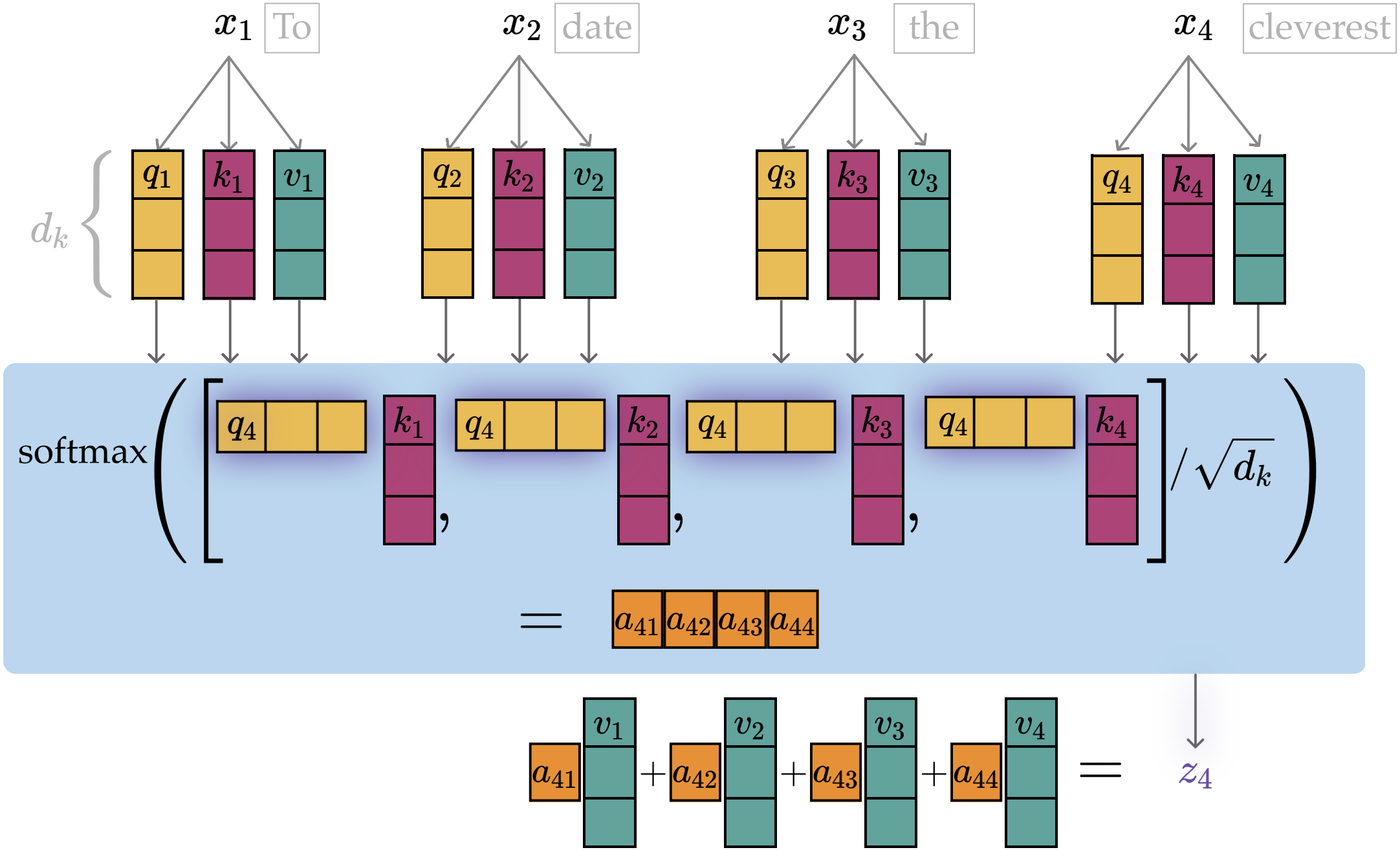

x_1 To
x_2 date
x_3 the
x_4 cleverest
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
$/\sqrt{d_k}$
$=$
a_{21} a_{22} a_{23} a_{24}
z_2
$=$
$+$ $+$ $+$

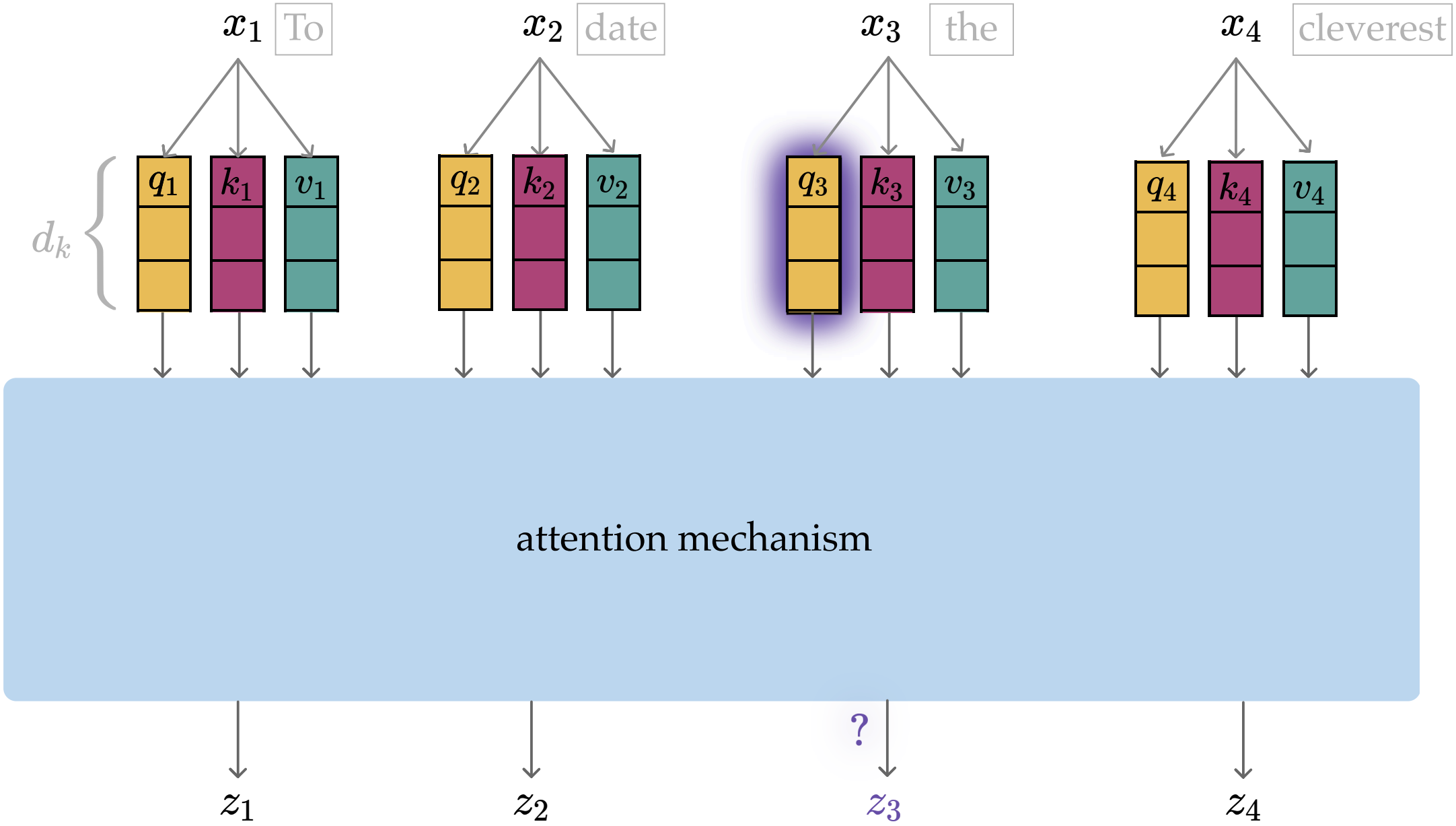

x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
?
z_1
z_2
z_3
z_4

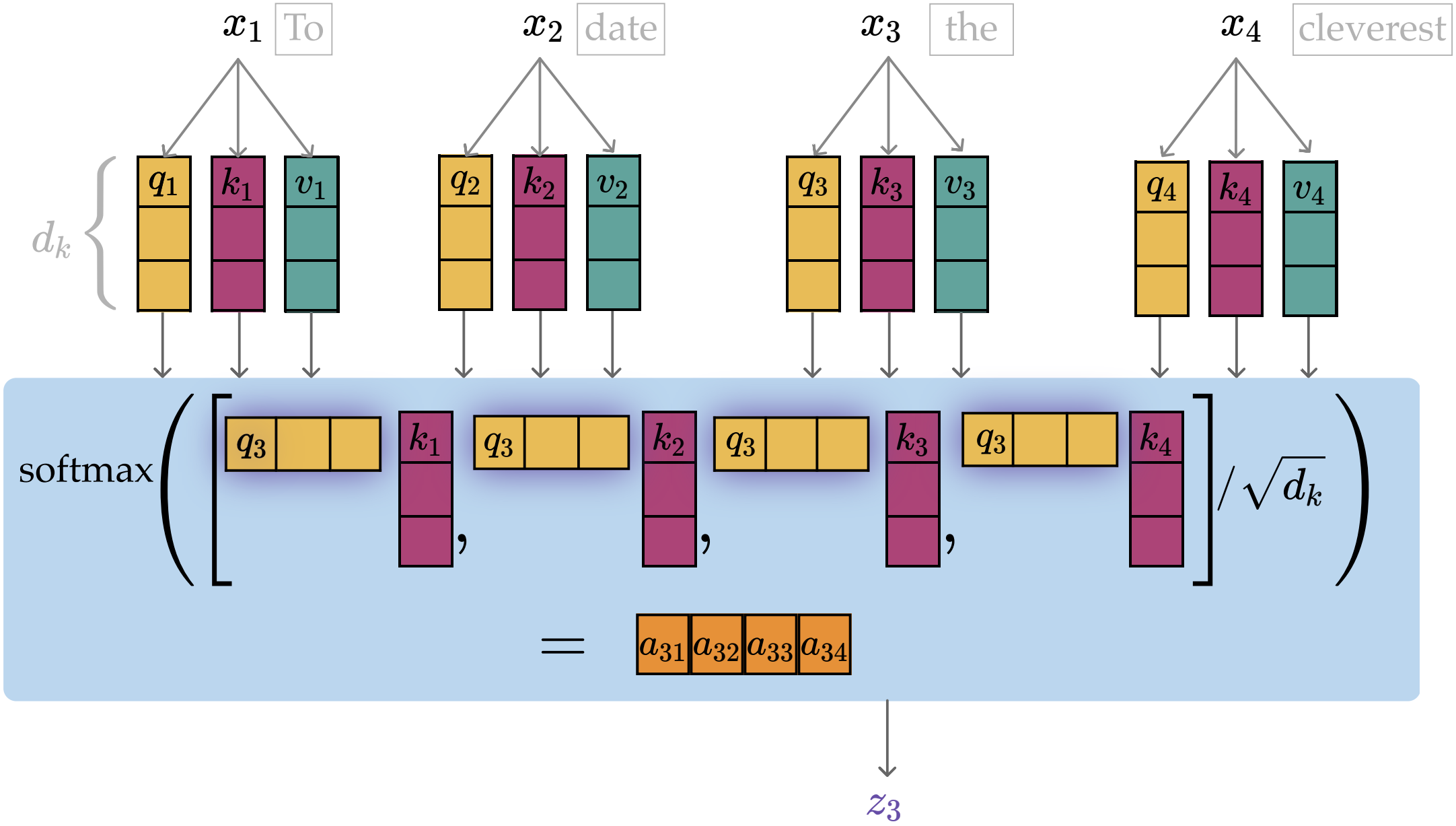


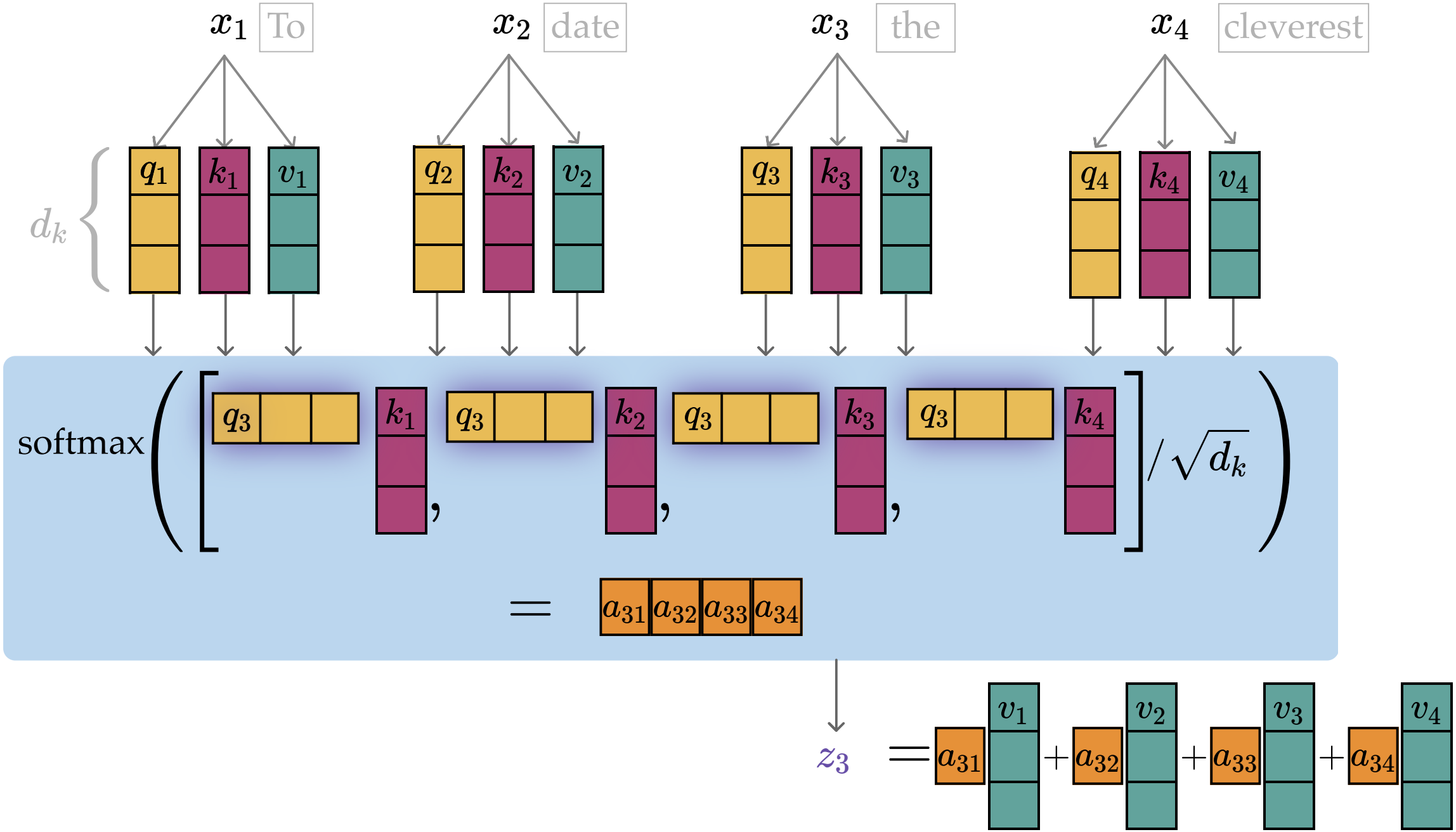
parallel and ***structurally identical*** processing

can calculate z_4 without z_3

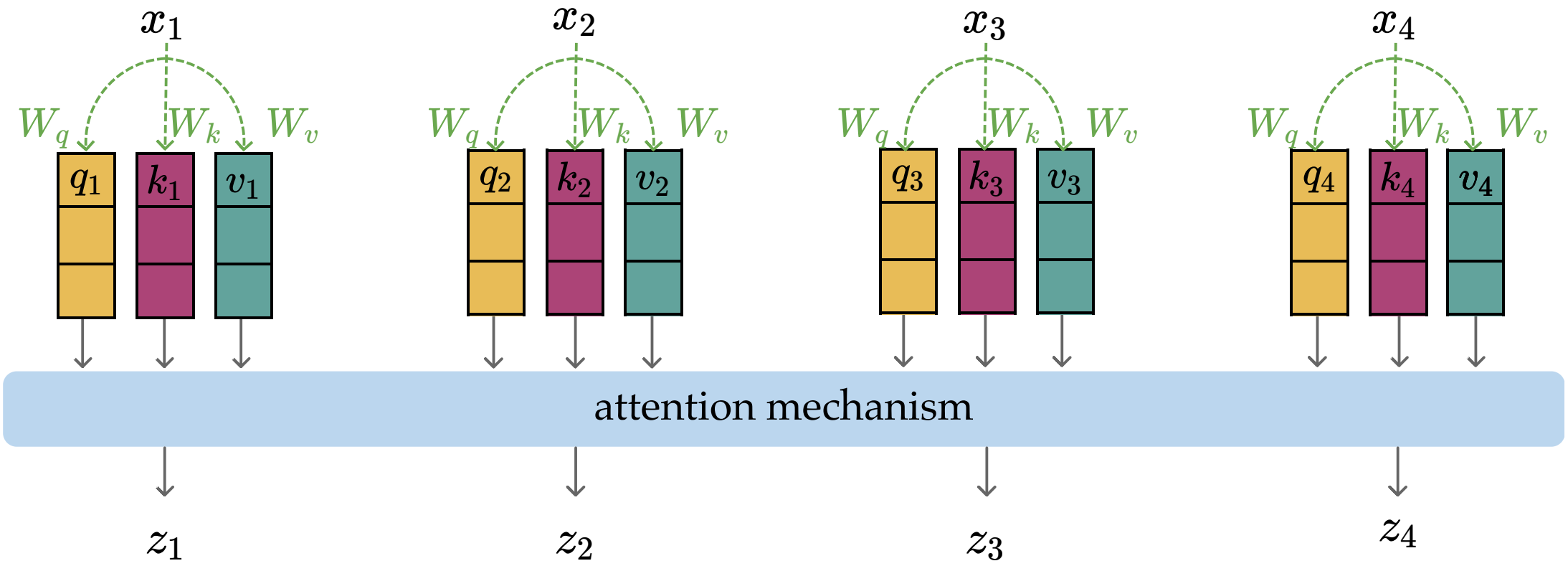

x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
$/\sqrt{d_k}$
$=$
a_{41} a_{42} a_{43} a_{44}
$+$ $+$ $+$ $=$
z_4


x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
?
z_1 z_2 z_3 z_4


x_1 To
x_2 date
x_3 the
x_4 cleverest
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
d_k
softmax
q_3 k_1 q_3 k_2 q_3 k_3 q_3 k_4
$/\sqrt{d_k}$
=
a_{31} a_{32} a_{33} a_{34}
z_3


x_1 To
x_2 date
x_3 the
x_4 cleverest
d_k
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
softmax
q_3 k_1 q_3 k_2 q_3 k_3 q_3 k_4
$/\sqrt{d_k}$
=
a_{31} a_{32} a_{33} a_{34}
z_3 = a_{31} v_1 + a_{32} v_2 + a_{33} v_3 + a_{34} v_4

Attention head

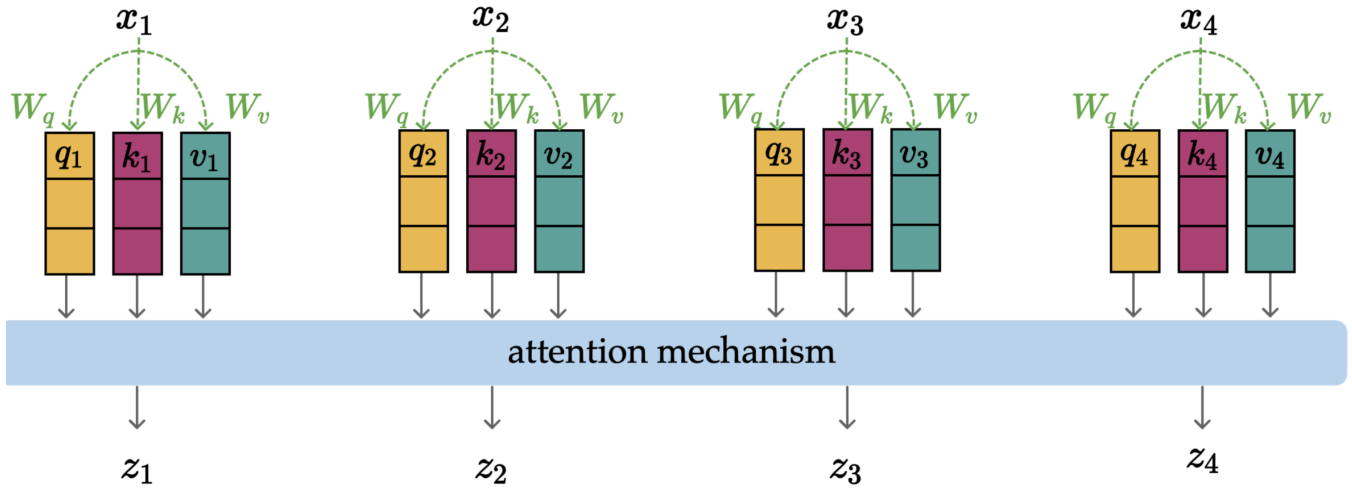


maps sequence of x to sequence of z:

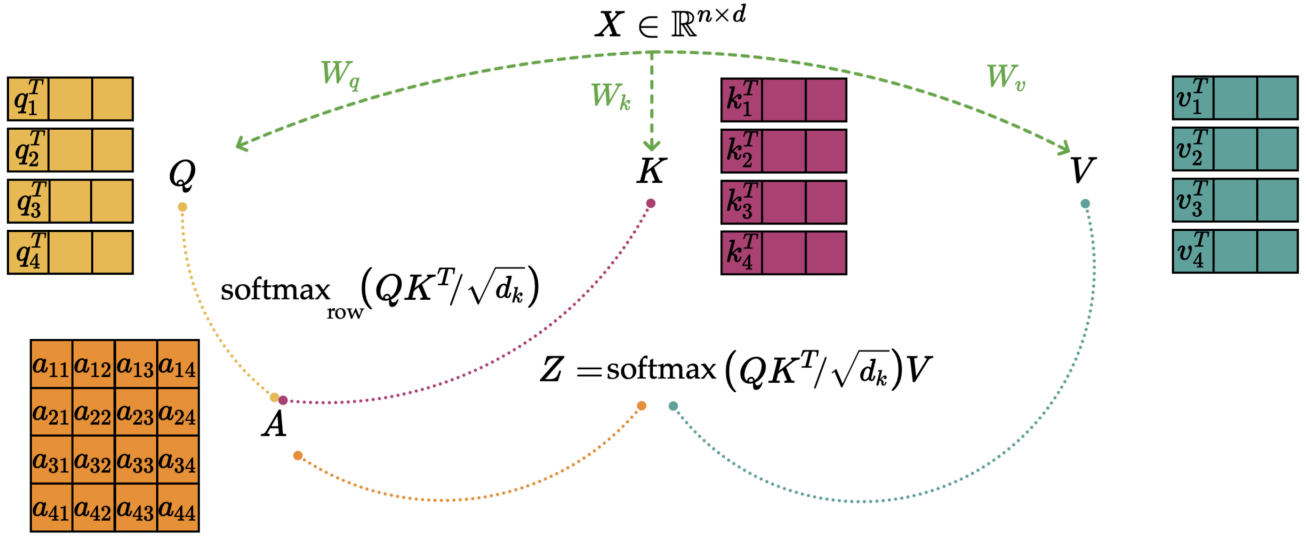
1. (query, key, value) projection
2. attention mechanism

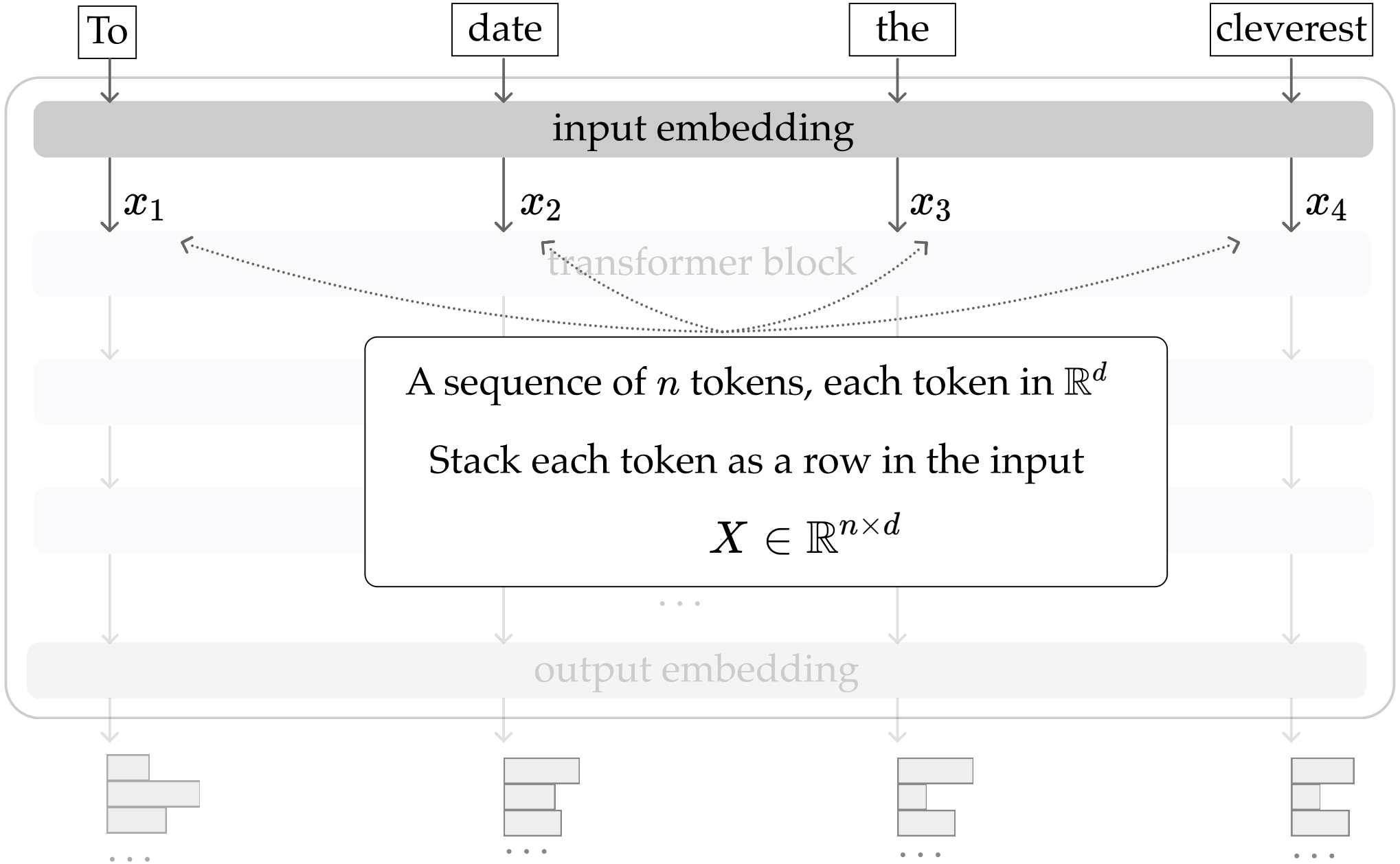
parallel and ***structurally identical*** processing

Attention head - compact matrix form

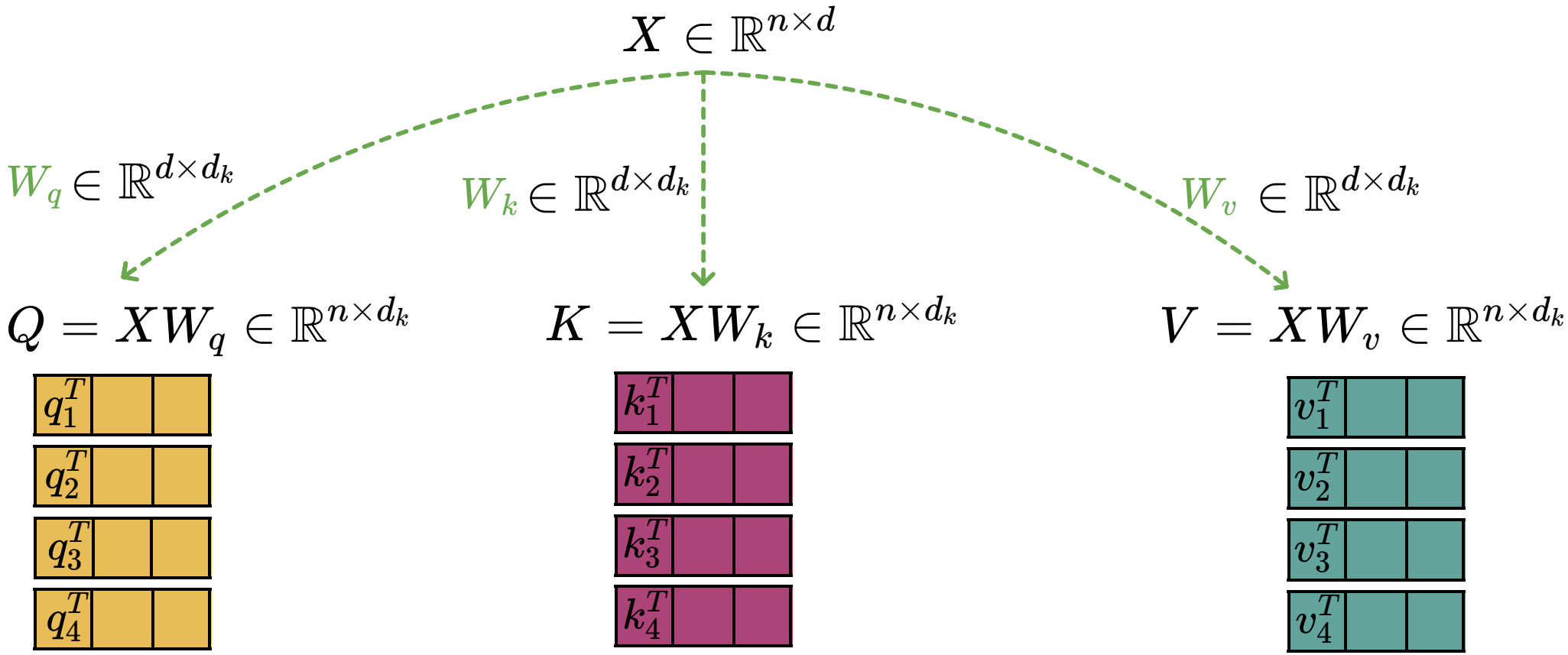


By stacking each individual vector in the sequence as a row




To
date
the
cleverest
input embedding
x_1
x_2
x_3
x_4
transformer block
A sequence of n tokens, each token in $\mathbb{R}^d$
Stack each token as a row in the input
$X \in \mathbb{R}^{n \times d}$
output embedding

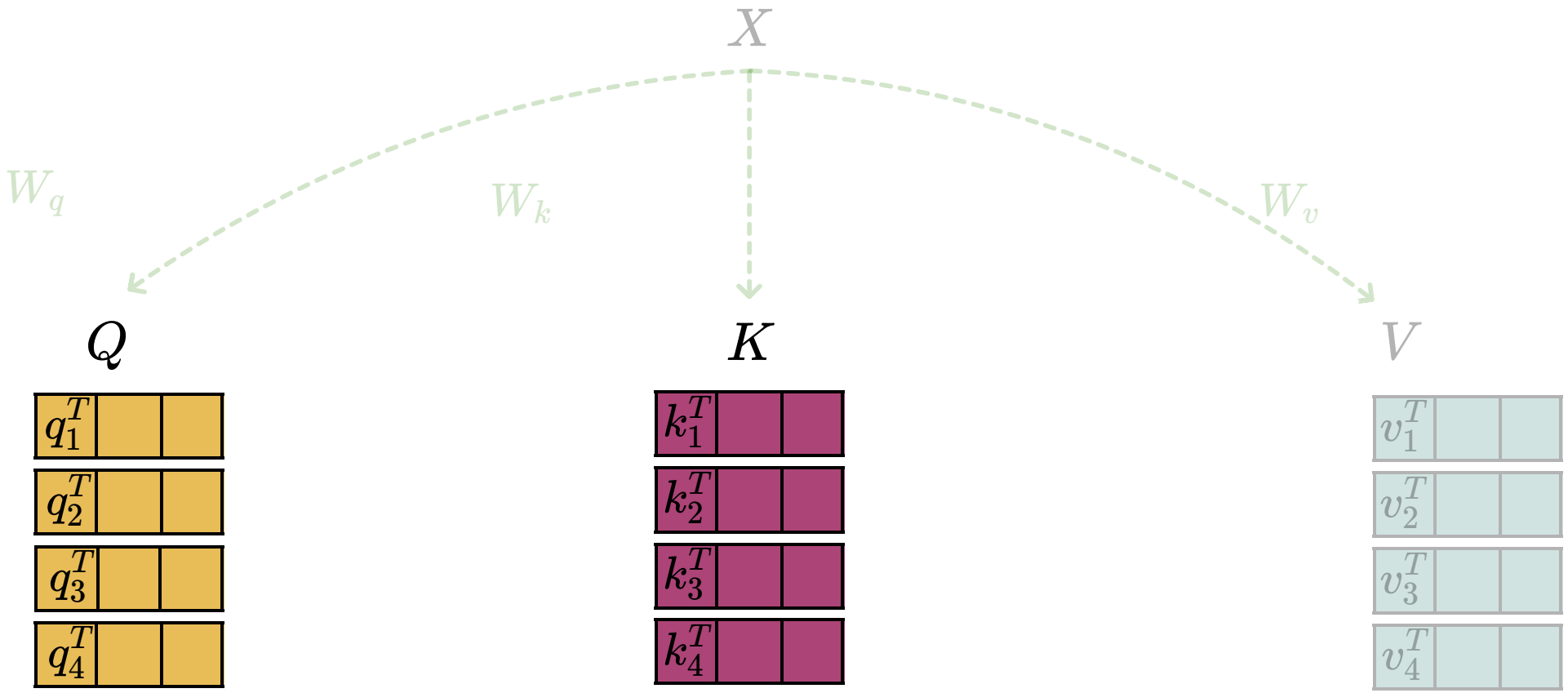
1. (query, key, value) projection



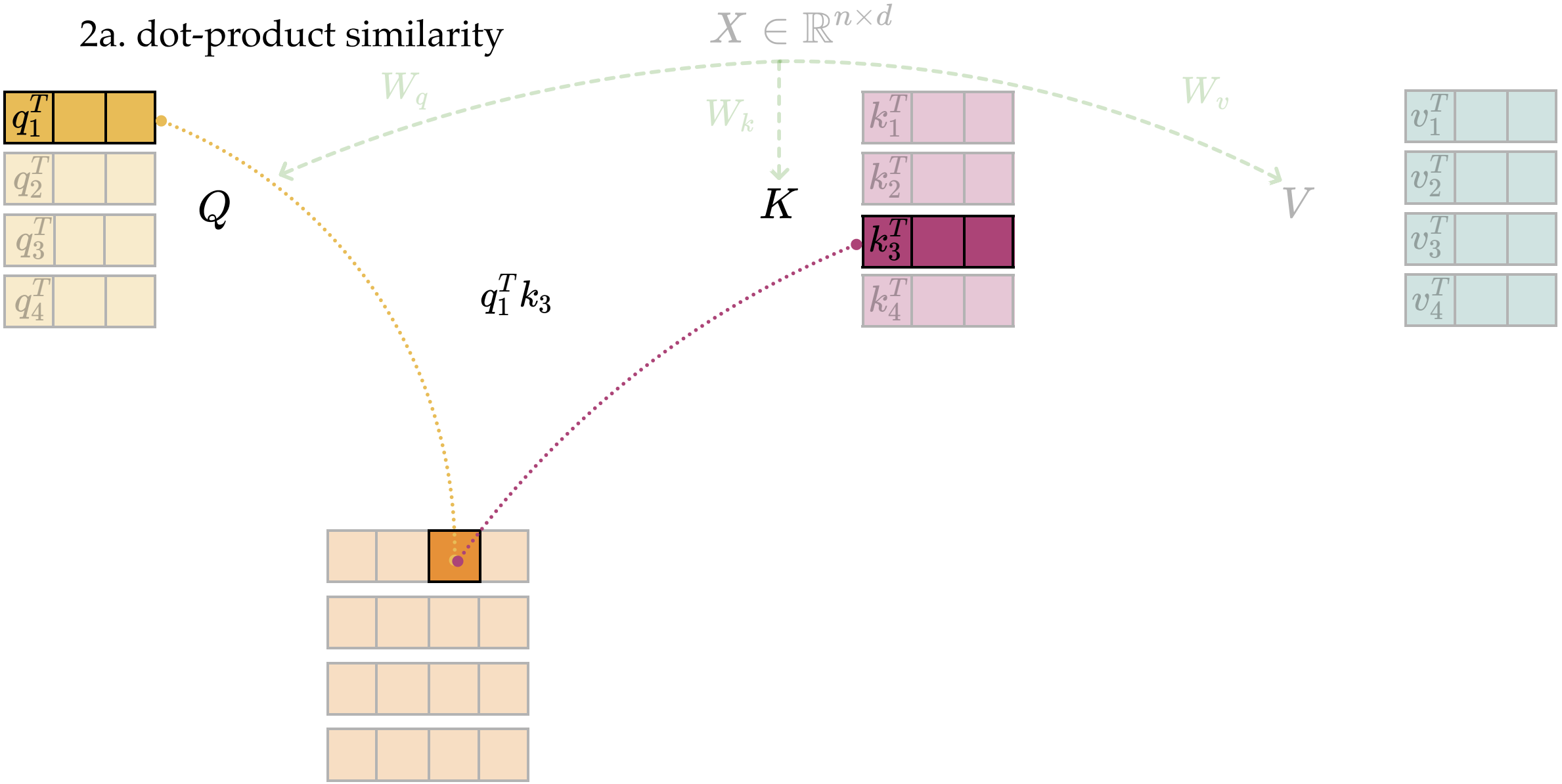
2a. dot-product similarity

compare q_i and k_j

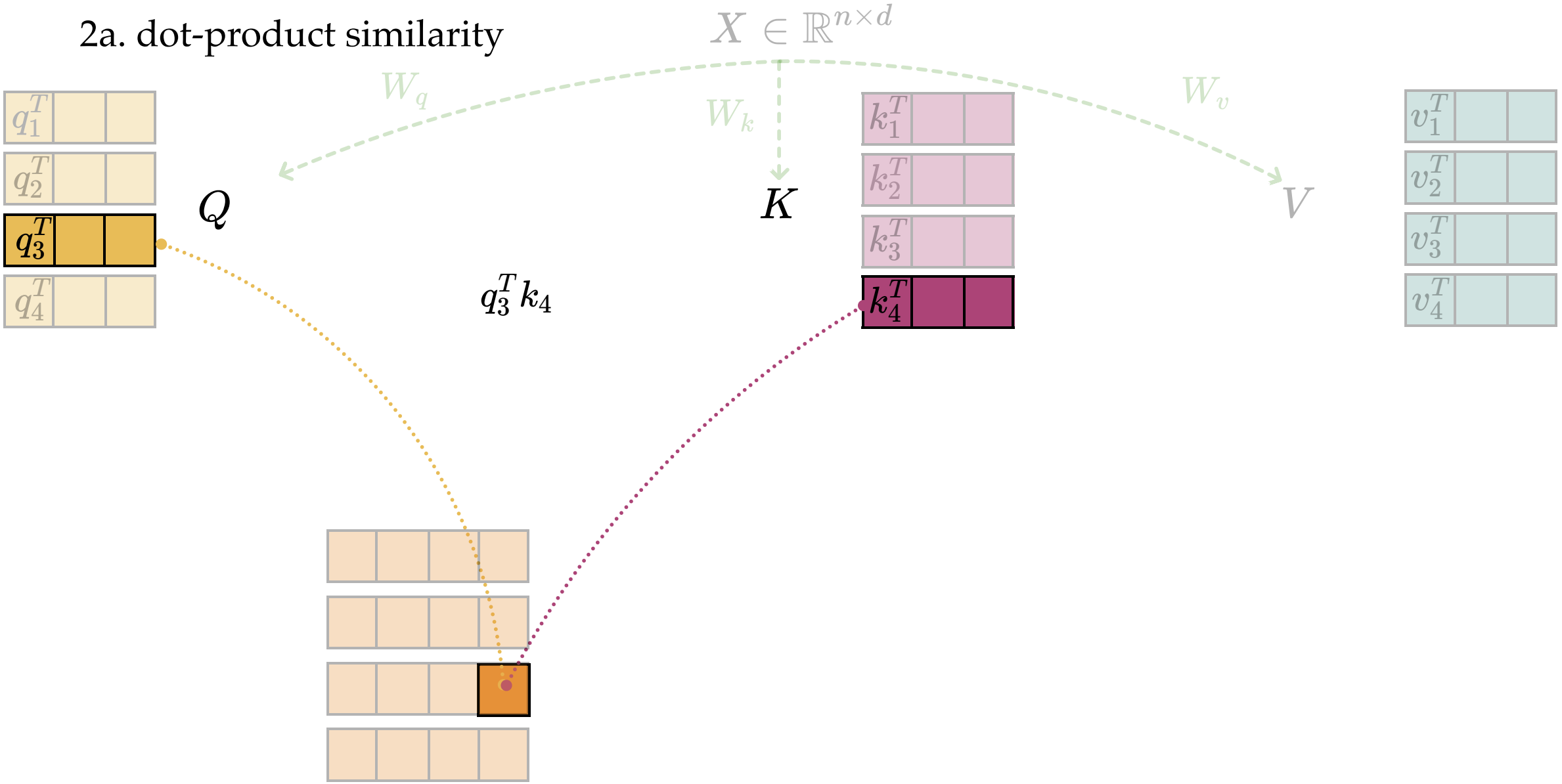
assemble the $n \times n$ similarities so rows correspond to query



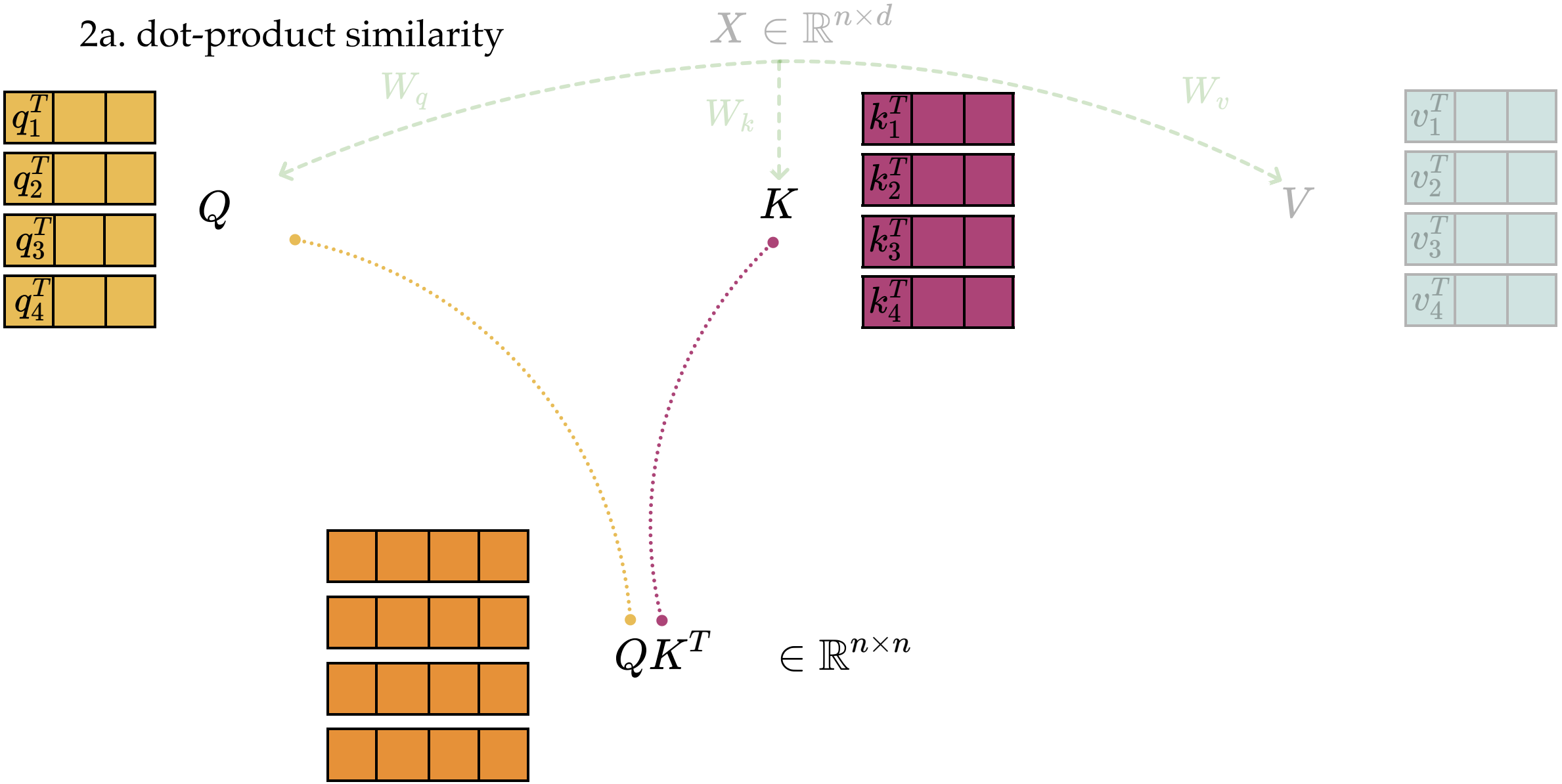
2a. dot-product similarity



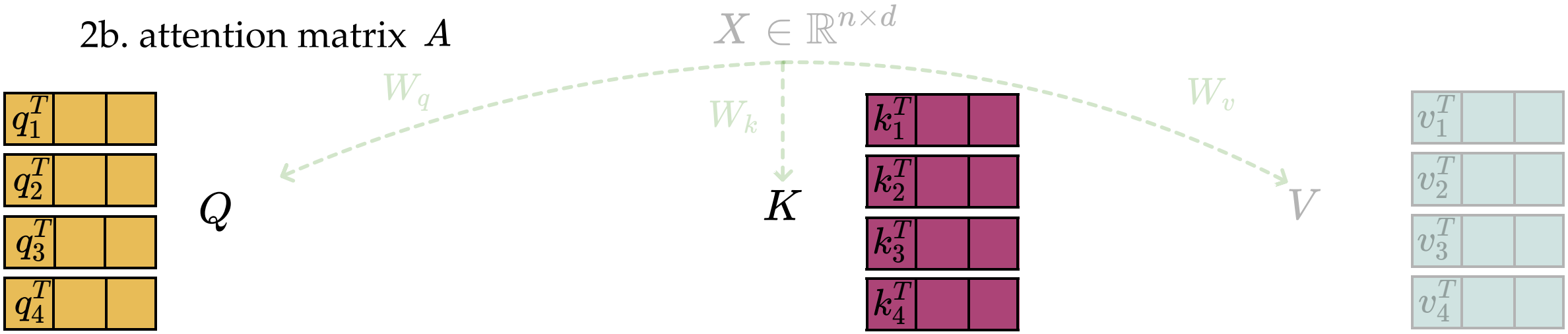
2a. dot-product similarity



2a. dot-product similarity



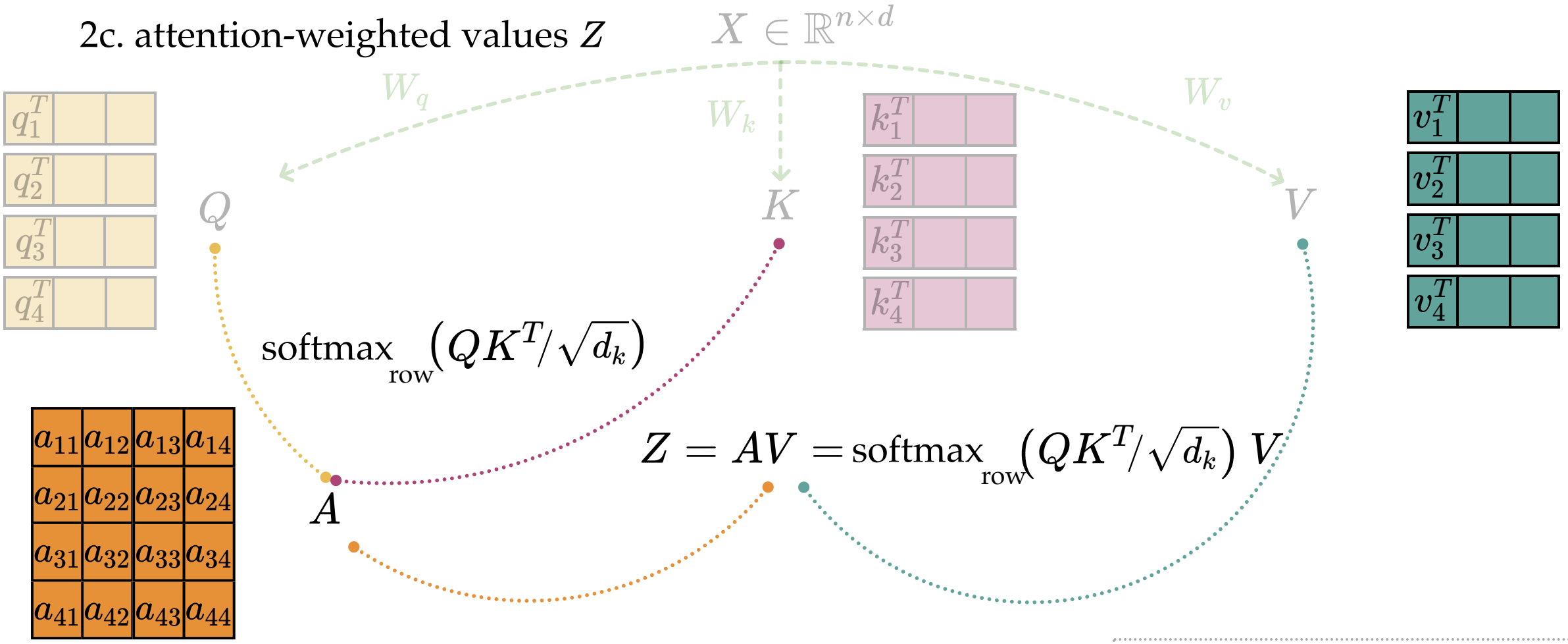
2b. attention matrix A



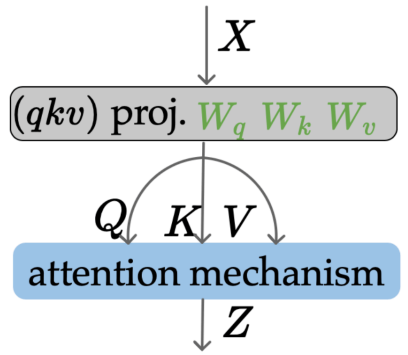
$$A = \begin{bmatrix} \text{softmax}(\quad / \sqrt{d_k}) \\ \text{softmax}(\quad / \sqrt{d_k}) \\ \text{softmax}(\quad / \sqrt{d_k}) \\ \text{softmax}(\quad / \sqrt{d_k}) \end{bmatrix} = \text{softmax}_{\text{row}}(QK^T/\sqrt{d_k}) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

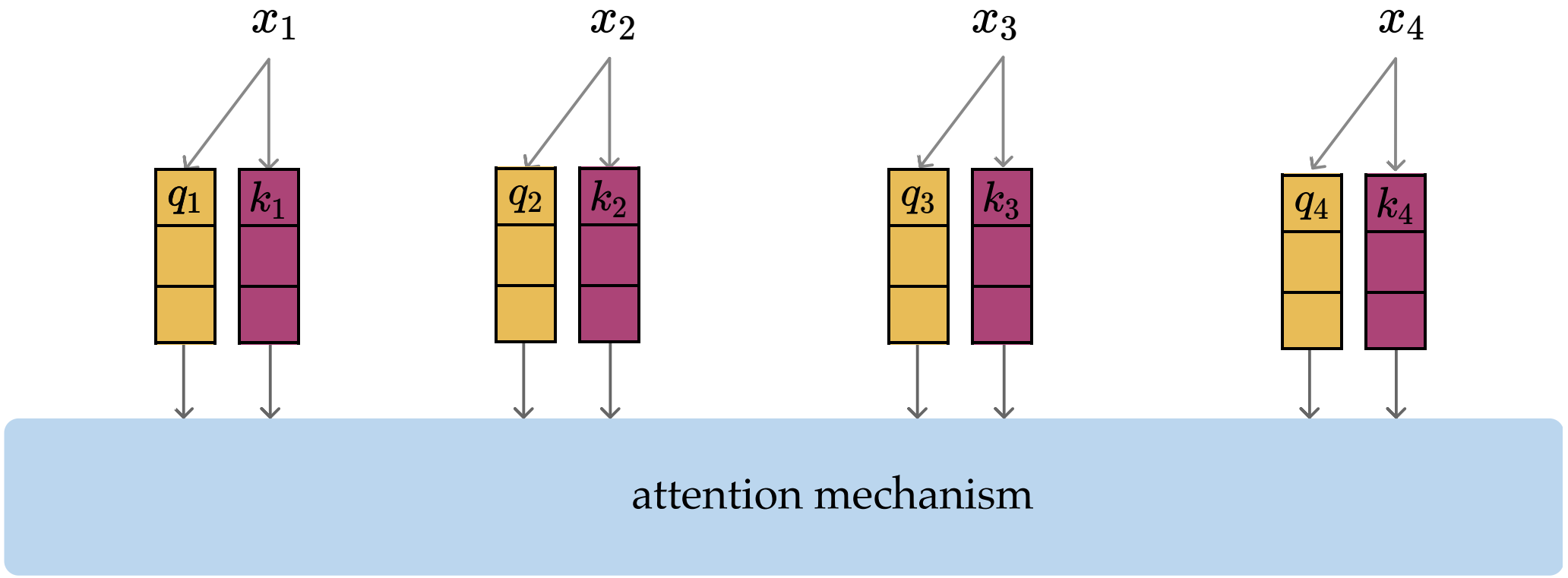
each row sums up to 1

2c. attention-weighted values Z



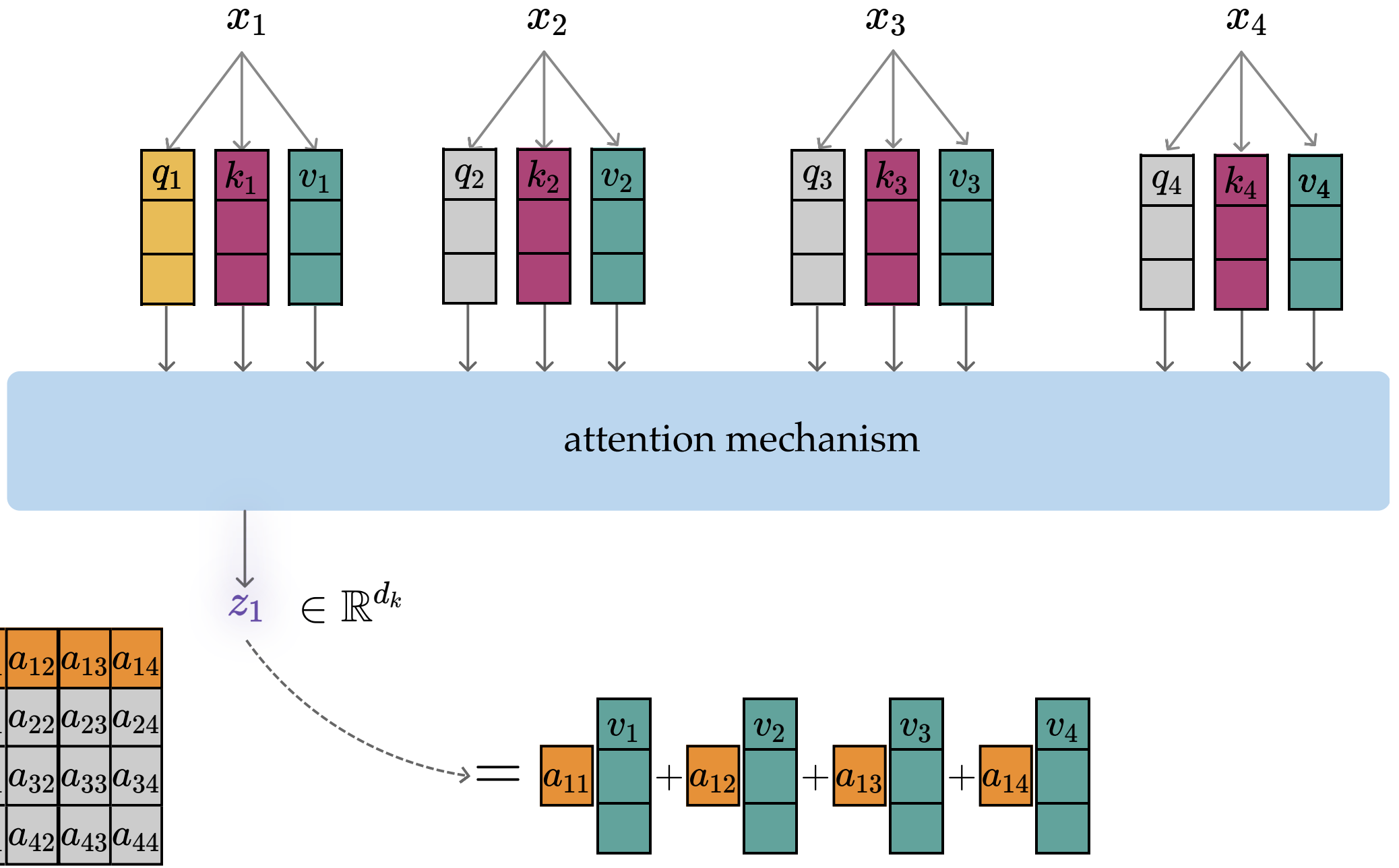
or, often even more compactly

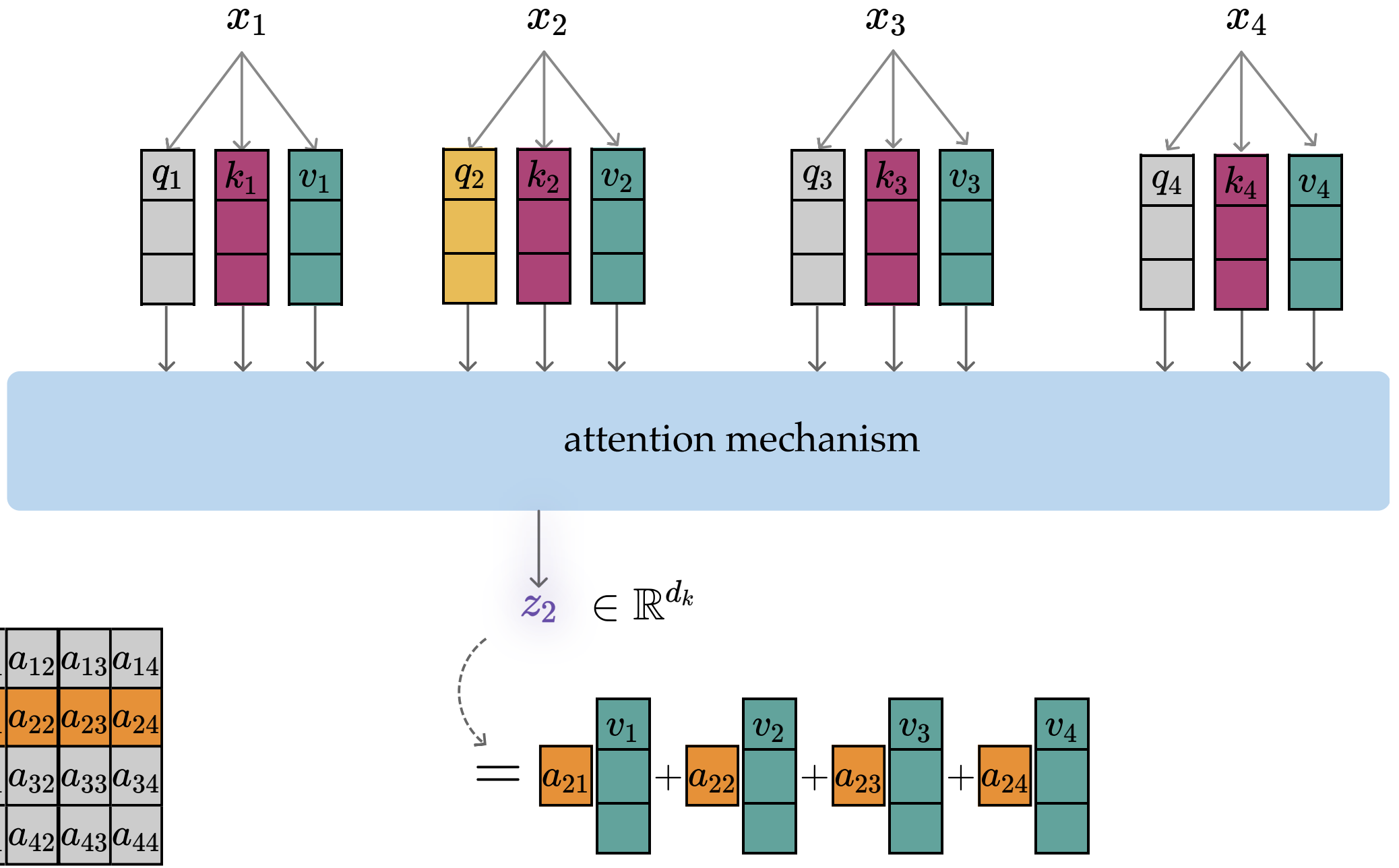


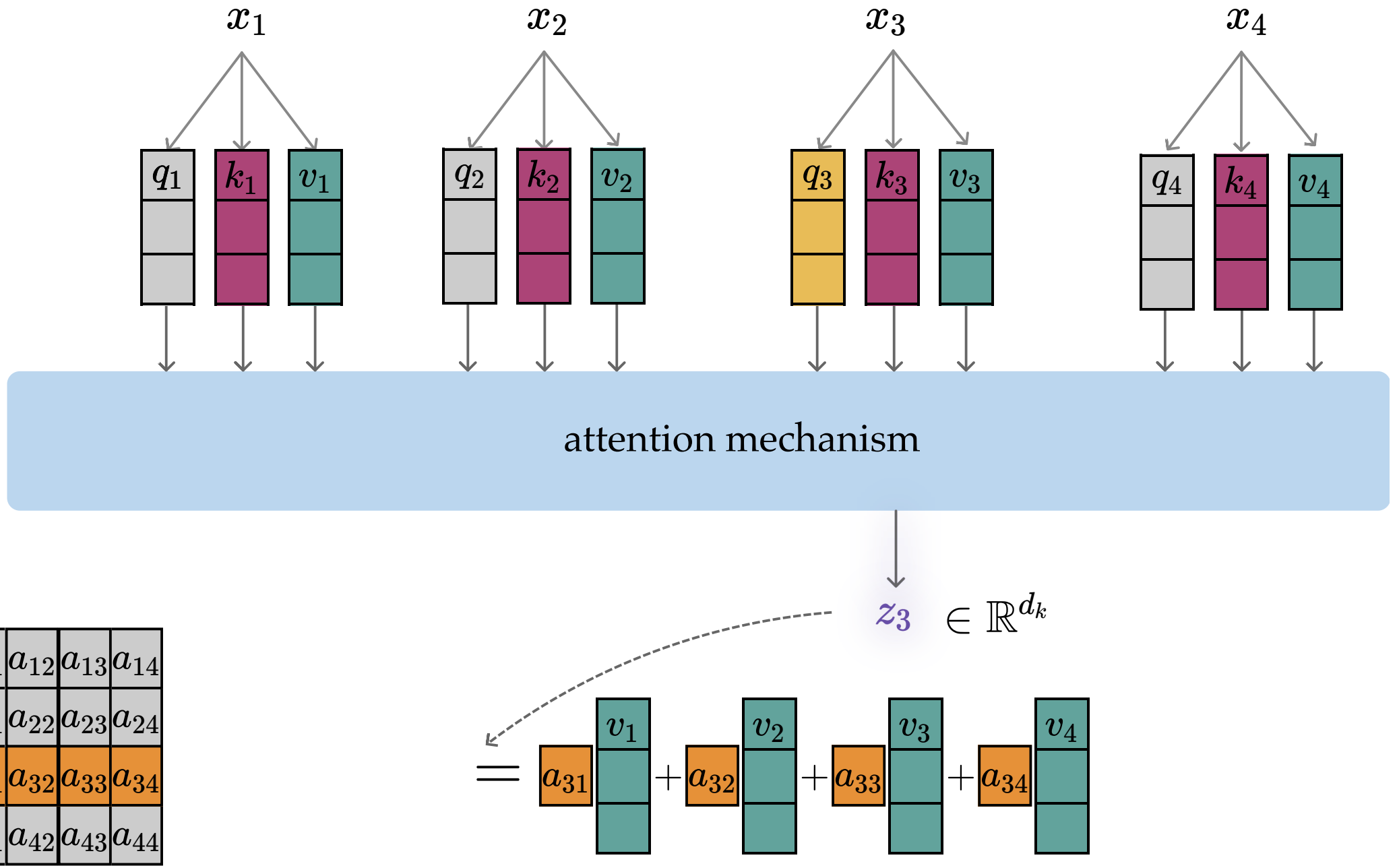


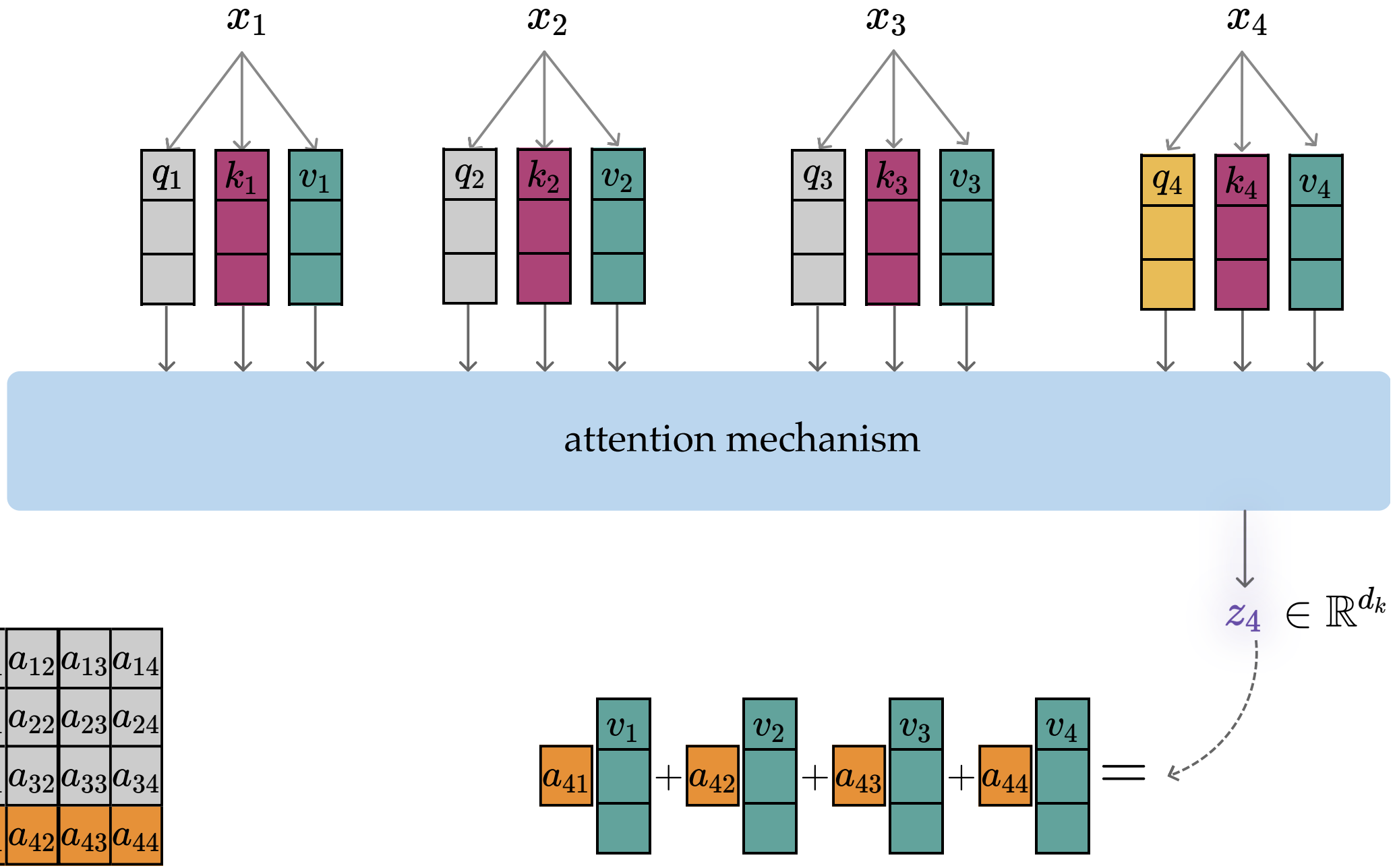
a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

attention scores depend on the (query, key) only


x_1
x_2
x_3
x_4
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
$z_1 \in \mathbb{R}^{d_k}$
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
$= a_{11} v_1 + a_{12} v_2 + a_{13} v_3 + a_{14} v_4$

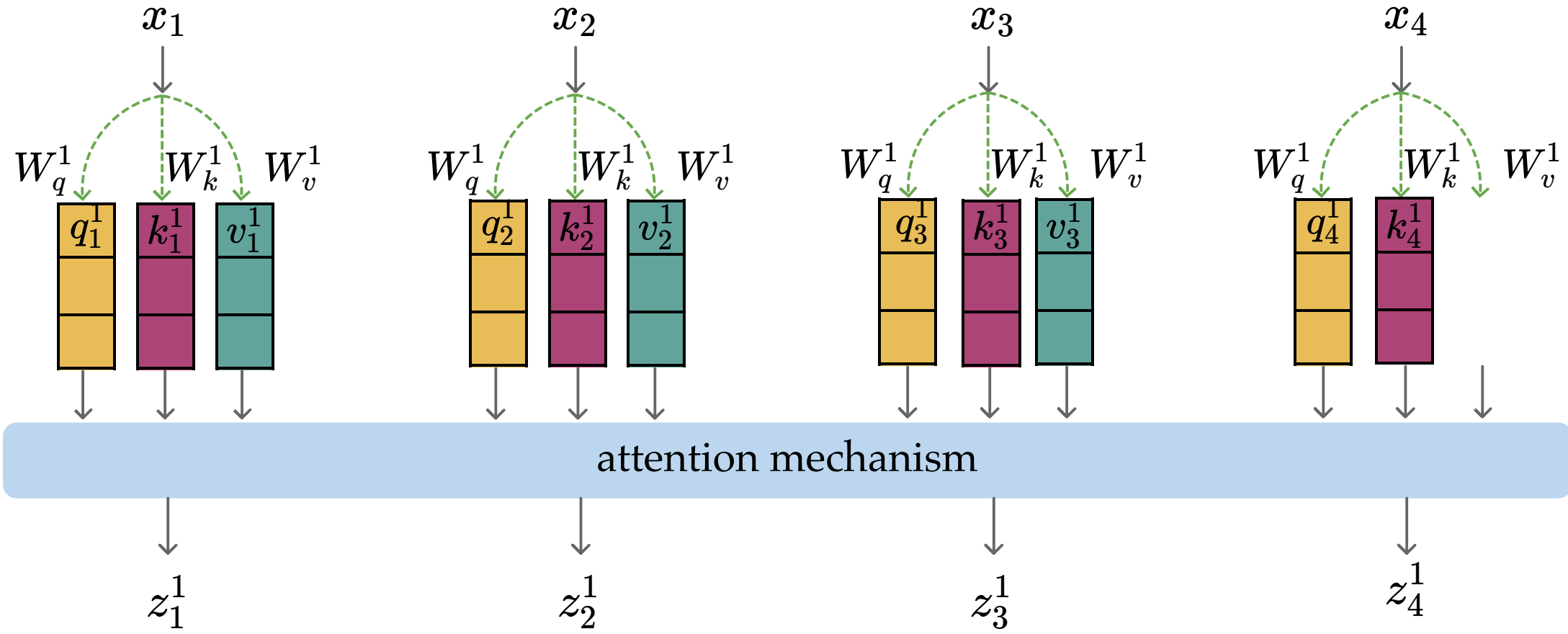

x_1
x_2
x_3
x_4
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
$z_2 \in \mathbb{R}^{d_k}$
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
$= a_{21} v_1 + a_{22} v_2 + a_{23} v_3 + a_{24} v_4$

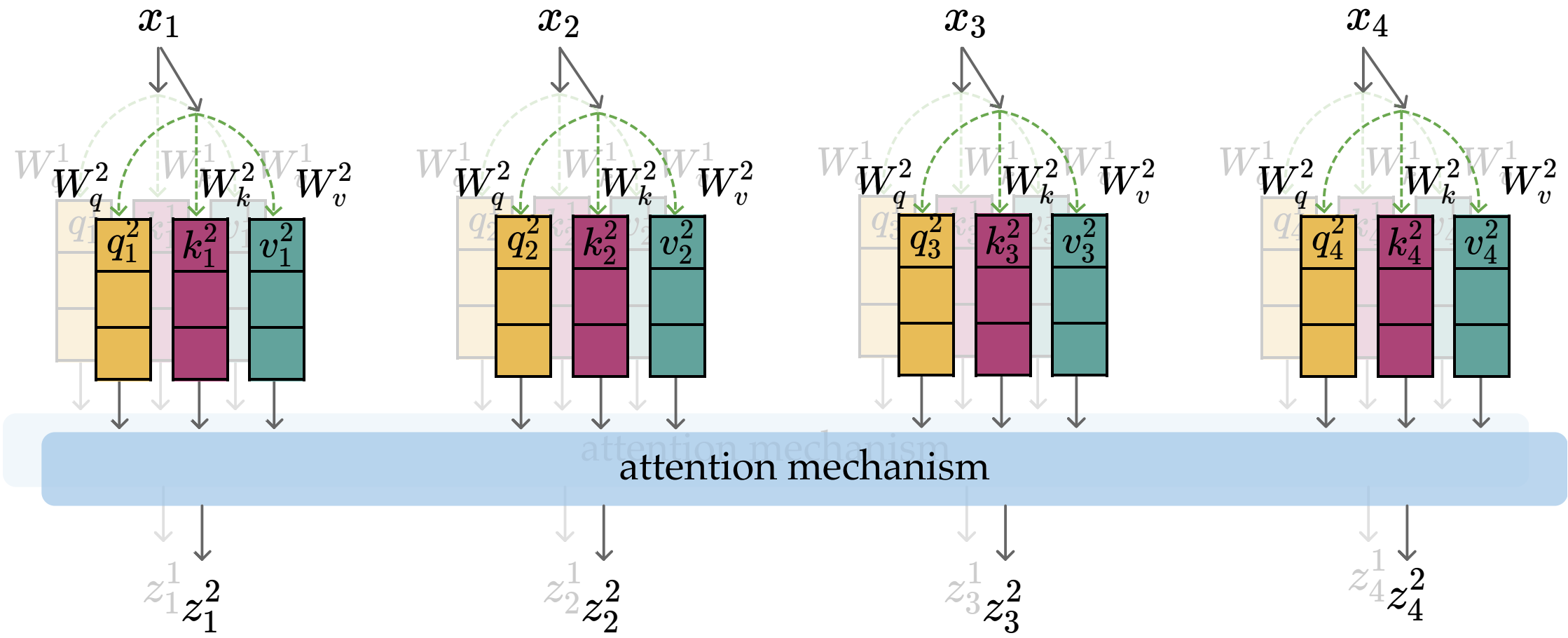

x_1
x_2
x_3
x_4
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
$z_3 \in \mathbb{R}^{d_k}$
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
$= a_{31} v_1 + a_{32} v_2 + a_{33} v_3 + a_{34} v_4$

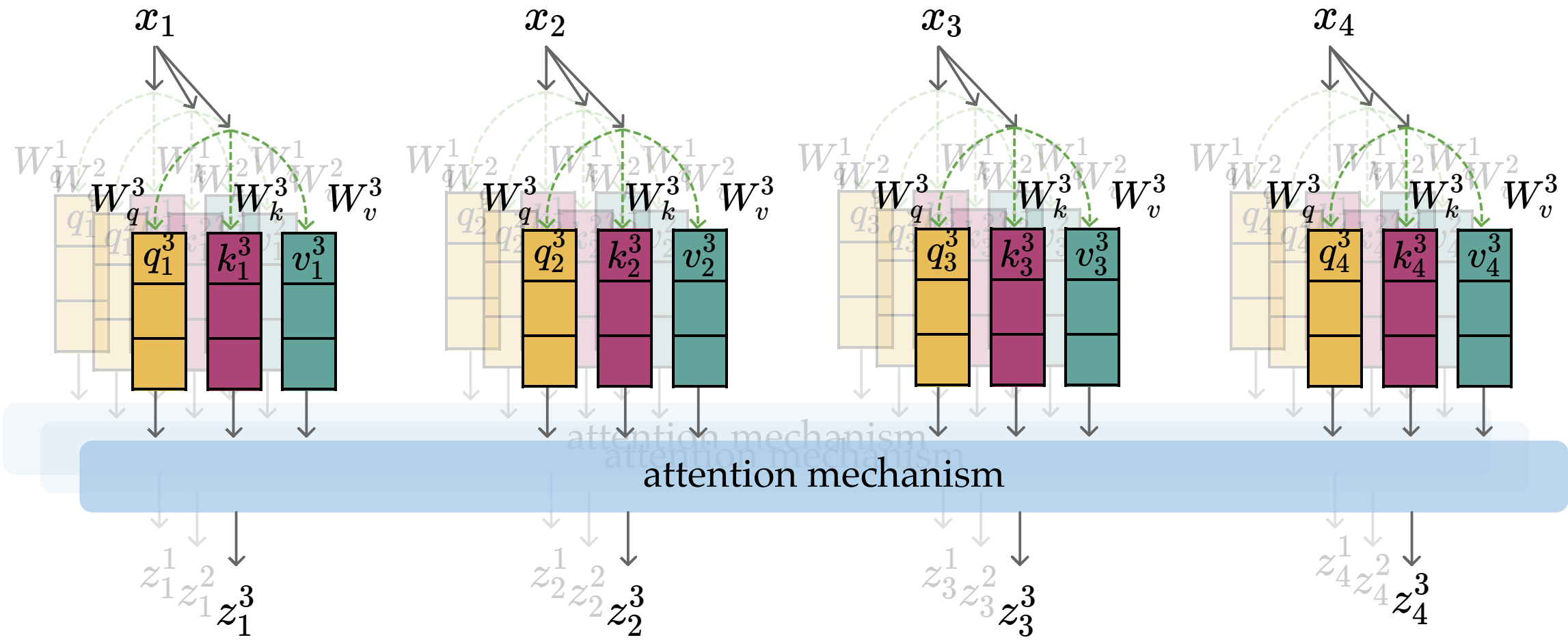

x_1
x_2
x_3
x_4
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
q_4 k_4 v_4
attention mechanism
$z_4 \in \mathbb{R}^{d_k}$
a_{11} a_{12} a_{13} a_{14}
a_{21} a_{22} a_{23} a_{24}
a_{31} a_{32} a_{33} a_{34}
a_{41} a_{42} a_{43} a_{44}
a_{41} v_1 + a_{42} v_2 + a_{43} v_3 + a_{44} v_4 =

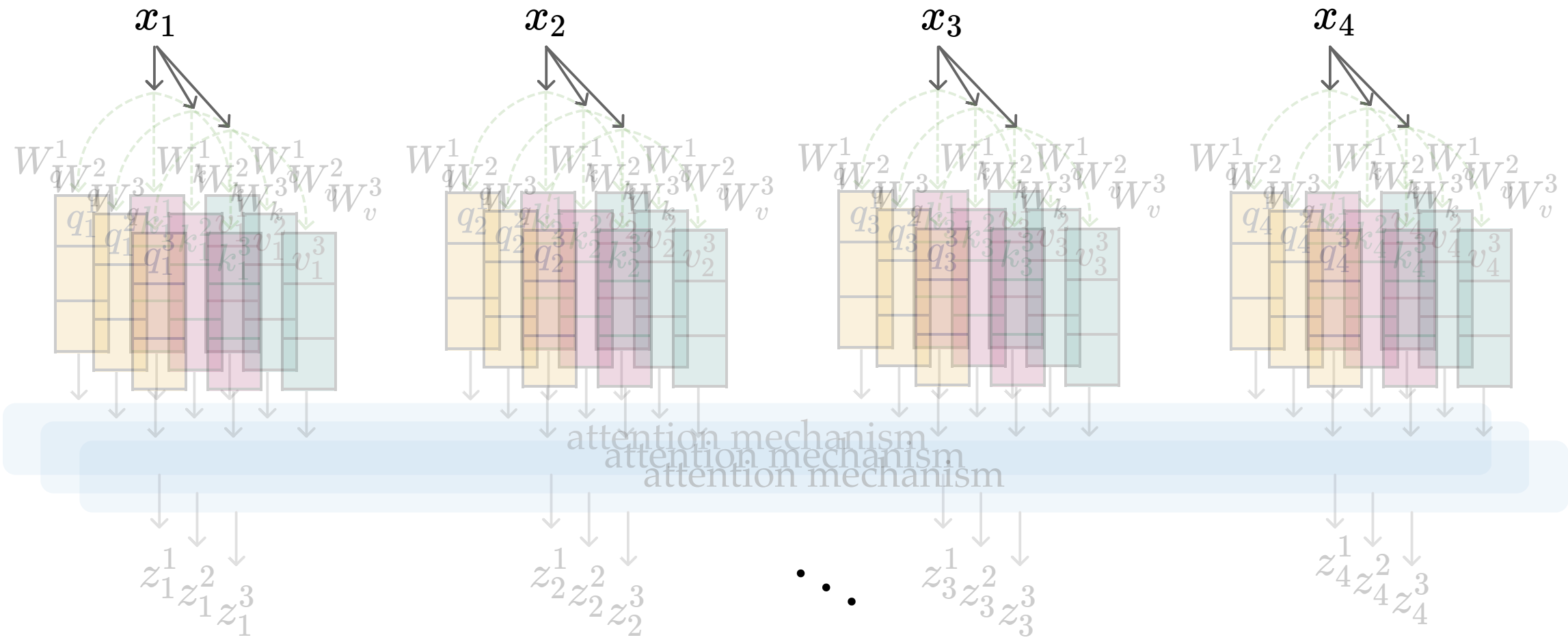
Outline

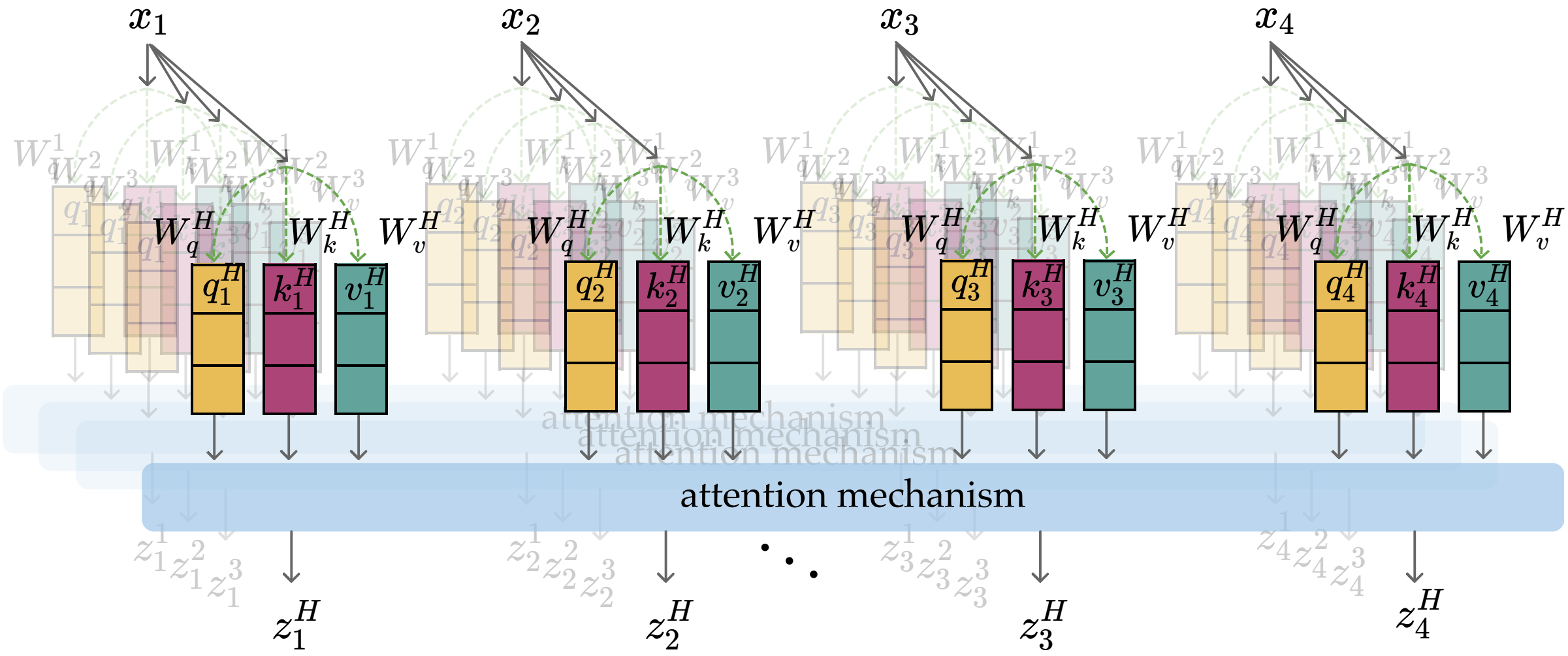
- Transformers high-level intuition and architecture
- Attention mechanism
- Multi-head attention
- (Applications)


x_1
x_2
x_3
x_4
W_q^1
W_k^1
W_v^1
q_1^1
k_1^1
v_1^1
q_2^1
k_2^1
v_2^1
q_3^1
k_3^1
v_3^1
q_4^1
k_4^1
attention mechanism
z_1^1
z_2^1
z_3^1
z_4^1


x_1
x_2
x_3
x_4
W_q^2
W_k^2
W_v^2
q_1^2
k_1^2
v_1^2
q_2^2
k_2^2
v_2^2
q_3^2
k_3^2
v_3^2
q_4^2
k_4^2
v_4^2
attention mechanism
z_1^2
z_2^2
z_3^2
z_4^2


x_1
x_2
x_3
x_4
W_q^3
W_k^3
W_v^3
q_1^3
k_1^3
v_1^3
q_2^3
k_2^3
v_2^3
q_3^3
k_3^3
v_3^3
q_4^3
k_4^3
v_4^3
attention mechanism
z_1^3
z_2^3
z_3^3
z_4^3


x_1
x_2
x_3
x_4
attention mechanism
attention mechanism
attention mechanism

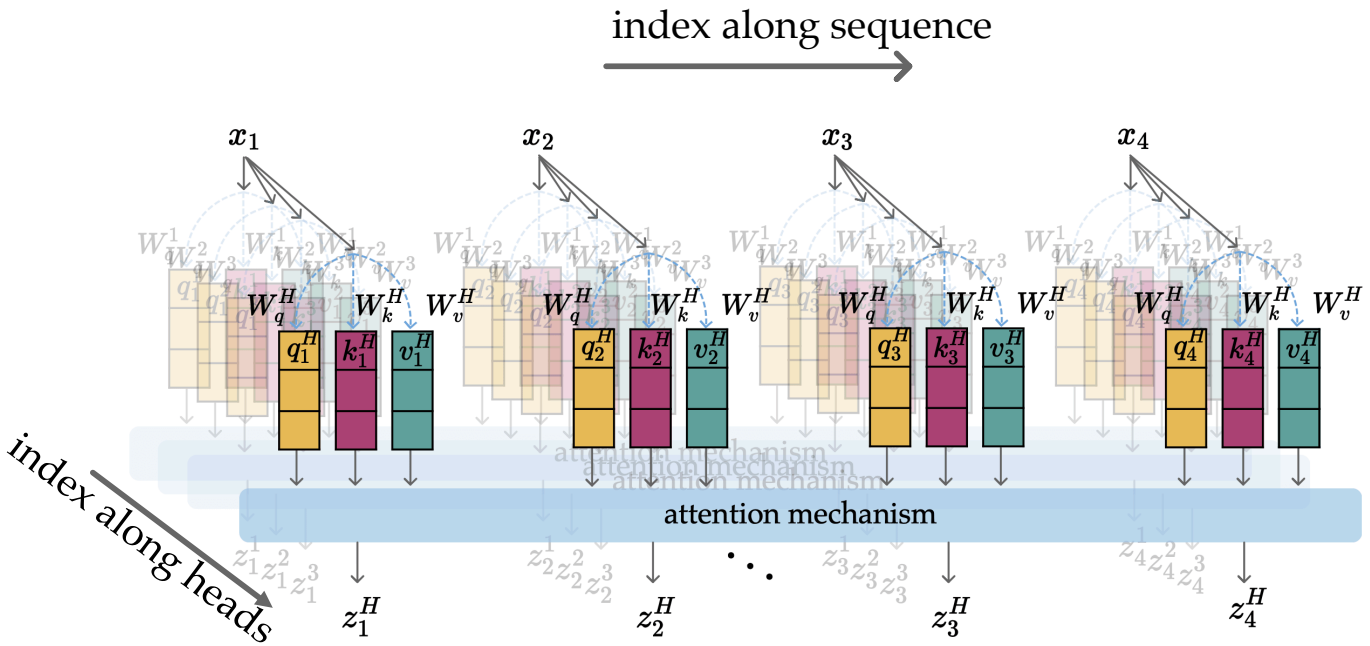

x_1
x_2
x_3
x_4
W_q^H
W_k^H
W_v^H
q_1^H
k_1^H
v_1^H
q_2^H
k_2^H
v_2^H
q_3^H
k_3^H
v_3^H
q_4^H
k_4^H
v_4^H
attention mechanism
z_1^H
z_2^H
z_3^H
z_4^H

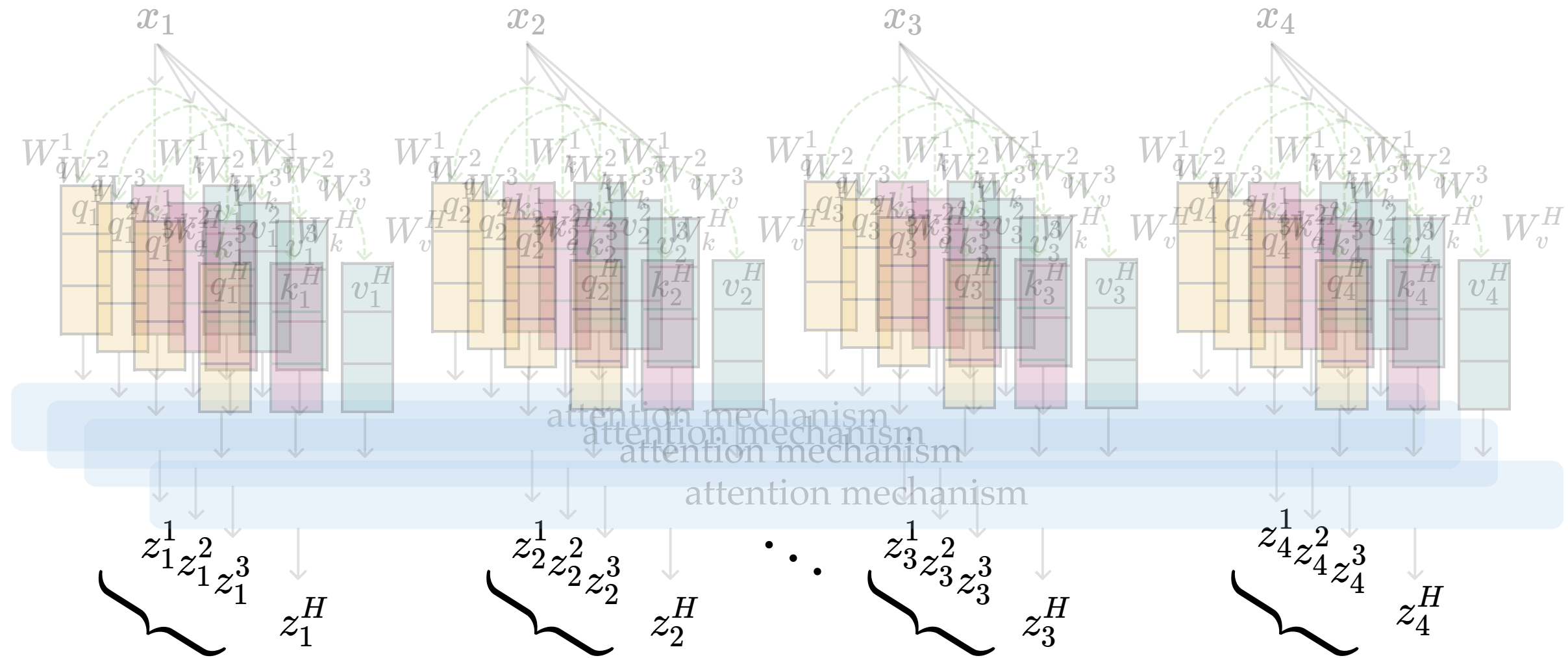
Multi-head Attention

Parallel, and ***structurally identical*** processing across all heads and tokens.

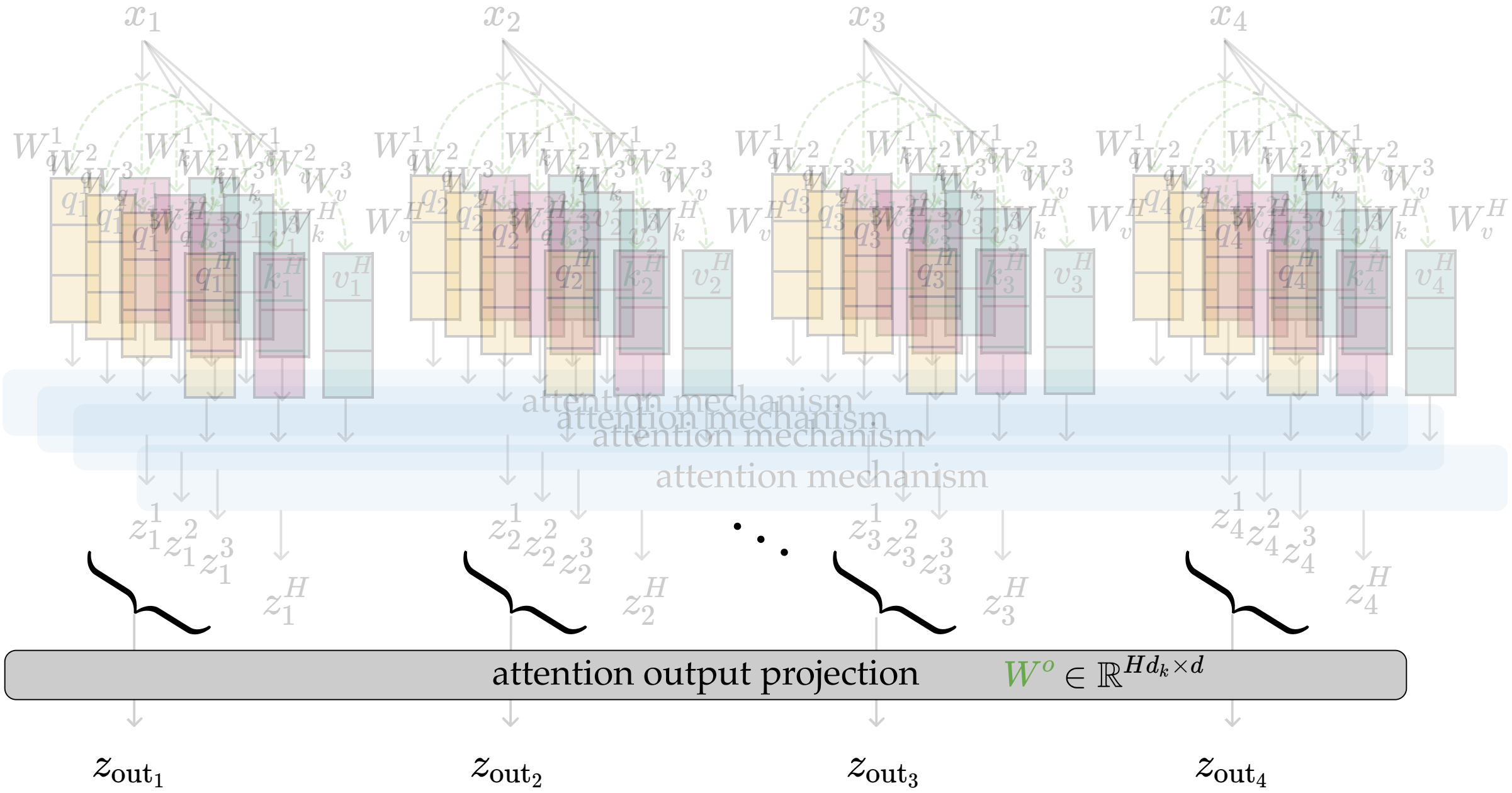
In particular, each head:

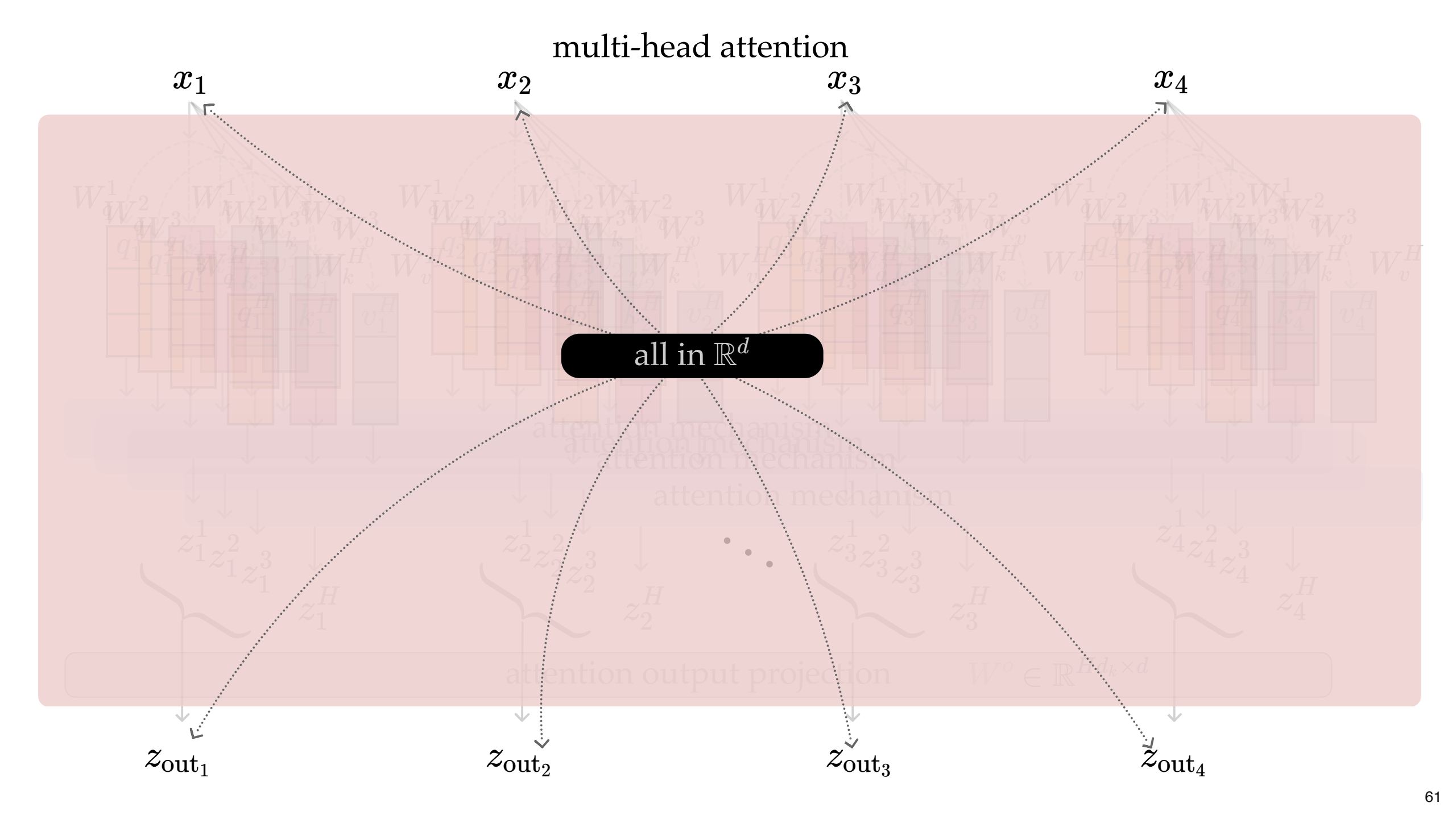
- learns its own set of W_q, W_k, W_v
- creates its own projected sequence of (q, k, v)
- computes its own sequence of z
- structurally identical processing
- for each token in the sequence:
  - structurally identical processing





each concatenated $z_i \in \mathbb{R}^{Hd_k}$


attention mechanism
attention mechanism
attention mechanism
attention mechanism
attention output projection
$W^o \in \mathbb{R}^{Hd_k \times d}$
z_{out_1}
z_{out_2}
z_{out_3}
z_{out_4}


multi-head attention
x_1
x_2
x_3
x_4
all in $\mathbb{R}^d$
z_{out_1}
z_{out_2}
z_{out_3}
z_{out_4}

Shape Example:

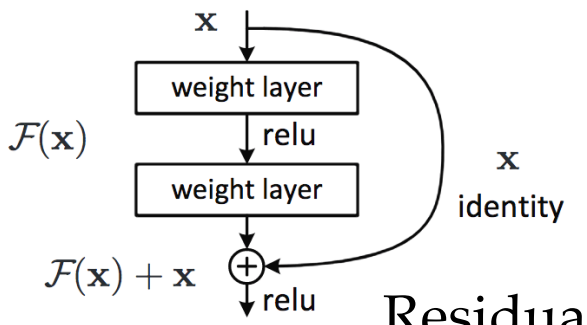
n	num tokens	5
d	word-embedding dim	6
d_k	(qkv) embedding dim	3
H	num heads	2

for a single attention head

X	**input**	$n \times d$	5×6
W_q^h	**query proj**	$d \times d_k$	6×3
W_k^h	**key proj**	$d \times d_k$	6×3
W_v^h	**value proj**	$d \times d_k$	6×3
Q^h	**query**	$n \times d_k$	5×3
K^h	**key**	$n \times d_k$	5×3
V^h	**value**	$n \times d_k$	5×3
A^h	**attn matrix**	$n \times n$	5×5
Z^h	**attn head out**	$n \times d_k$	5×3
$\text{concat}(Z^1 \dots Z^H)$	**multi-head out**	$n \times Hd_k$	5×6
W^o	**output proj**	$Hd_k \times d$	6×6
Z_{out}	**attn layer out**	$n \times d$	5×6

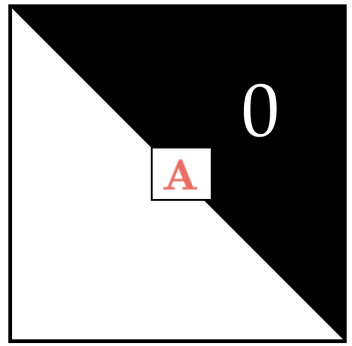
Ws are the learned weights

Some practical techniques commonly needed when training auto-regressive transformers:

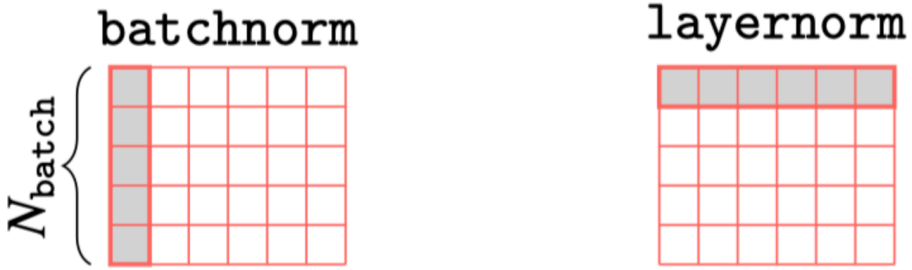


Residual connection

masking



Positional encoding



Layer normalization

https://poloclub.github.io/transformer-explainer/

Outline

- Transformers high-level intuition and architecture
- Attention mechanism
- Multi-head attention
- (Applications)

Generative Boba

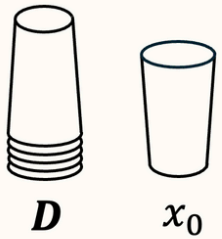
By Boyuan Chen
boyuanc@mit.edu

Intro:

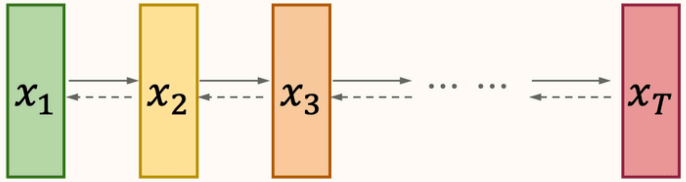
Boba tea is an essential component of Asian students' well-being and productivity.
Prior studies [1] have suggested the lack of accessible boba shop hurts MIT's reputation as a top CS school.
To address this problem, we open a novel boba shop at MIT - Generative Boba.
By providing PhD students with free boba every afternoon, Generative Boba boosts the research productivity of the floor by TBD%.
Generative Boba is also looking for Chief Boba Scientists to scale up! Contact me to be a co-author.

[1] Chen, B. (2022, June 20). *BEST COMPUTER SCIENCE SCHOOLS RANKED BY BOBA*. Boyuan's Blog. Retrieved March 27, 2024,.

Step 1:
Sample a cup from cup set



Step 2:
Gradually add t steps of ingredients



Step 3:
Graduate Student Descent

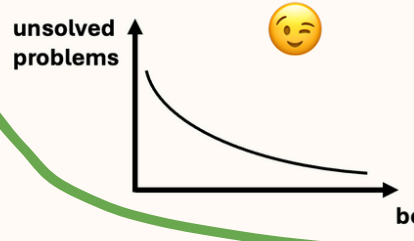


Fig 1: We propose boba diffusion, a novel generative model that boosts PhD student productivity

Today's special:

ε_1= *topping*

20%

open boba pot for topping
close pot lid to keep warm

ε_2= *milk*

30-60%

we serve lactose-free milk,
sometimes coconut milk

ε_3= *tea*

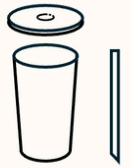
20-50%

use 20% for matcha
use 50% for black tea

ε_4 = *syrup*

any

hold black sugar bottle
upside down for 5-10s



add a lid and straw
our straws are compostable

Fig 2: Implementation details of boba diffusion. We follow an efficient ingredient schedule while "today's special" provides special recipes from time to time.

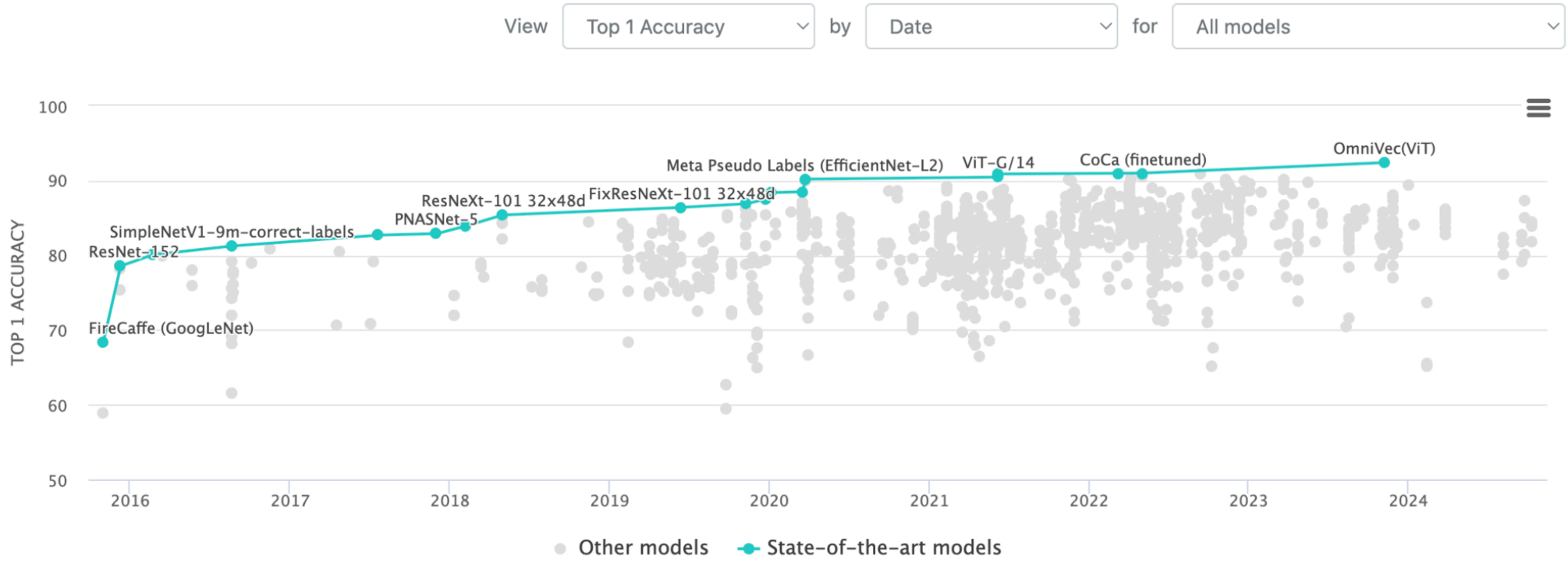
Generative Boba by Boyuan Chen in Bldg 45

image credit: Nicholas Pfaff

Image Classification on ImageNet

Leaderboard Dataset

Transformers in Action: Performance across domains



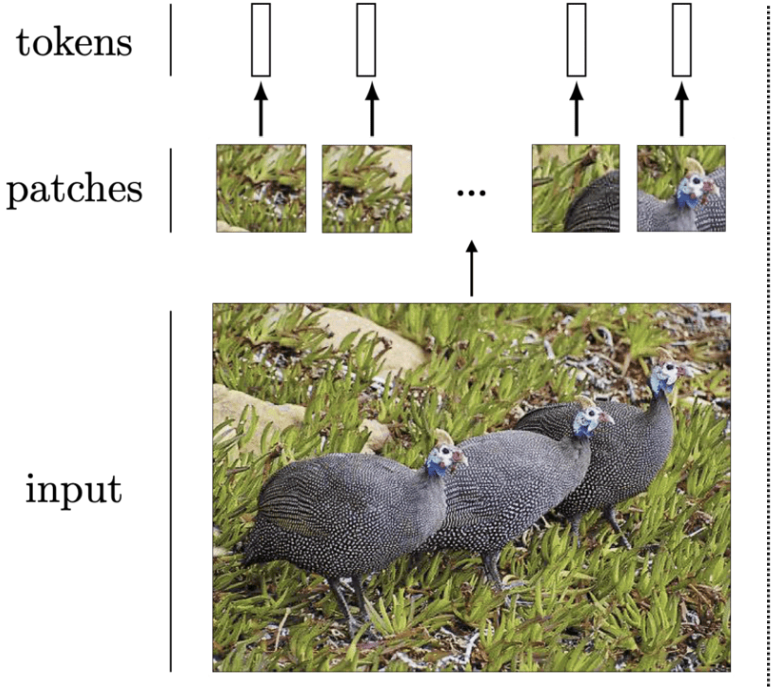
Filter:
ImageNet-1k only | Transformer | ResNet | CNN | ImageNet-22k | EfficientNet | JFT-300M | MLP | ResNeXt | Reversible
Neighborhood Attention | NAT Transformer | JFT-3B | PatchConvnet | FPN | MoE | Early Exit | Dynamic Model Arch
CNN+Transformer | ALIGN | Conv+Transformer | CLIP data | No Extra Data | SNN | IG-1B | Swin-Transformer | Teacher-22k
Vision Transformer | FLD-900M | Pure CNN | YFCC-15M | Laion-400M | Contrastive | ConvNeXt | Self-Supervised Learning
RegNet | Mixer | Memory-Centric | CLIP Pre-trained | CrossCovarianceAttention | untagged | Hardware Burden
Operations per network pass | Robustness reports

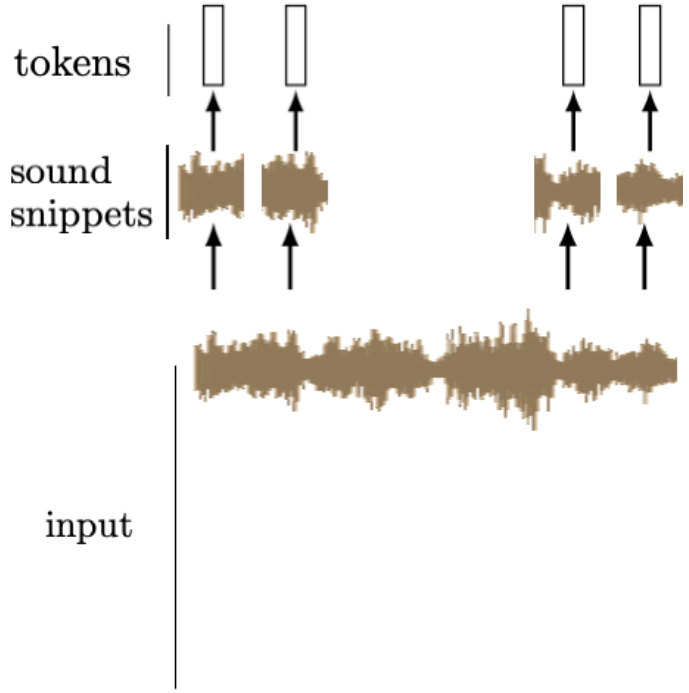
Edit Leaderboard

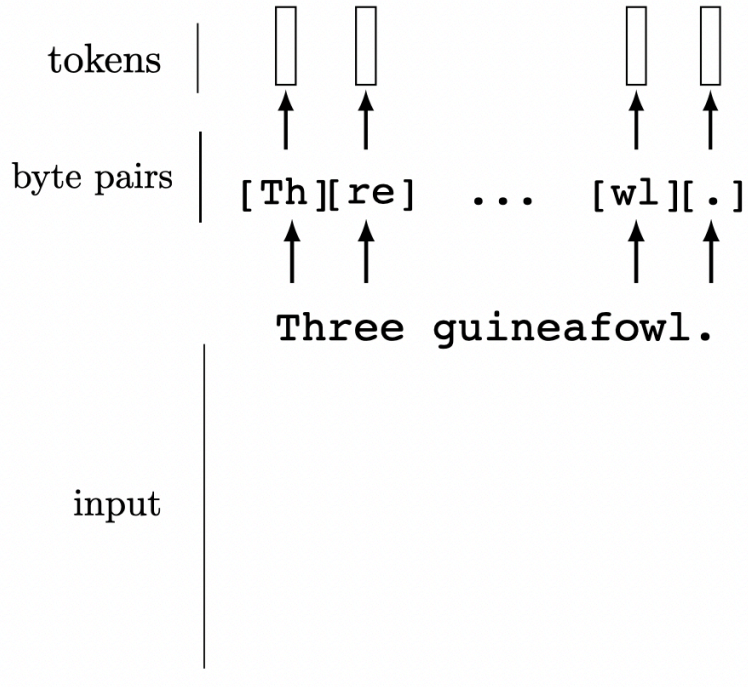
Paperswithcode

We can tokenize anything.

General strategy: chop the input up into chunks, *project* each chunk to an **embedding**



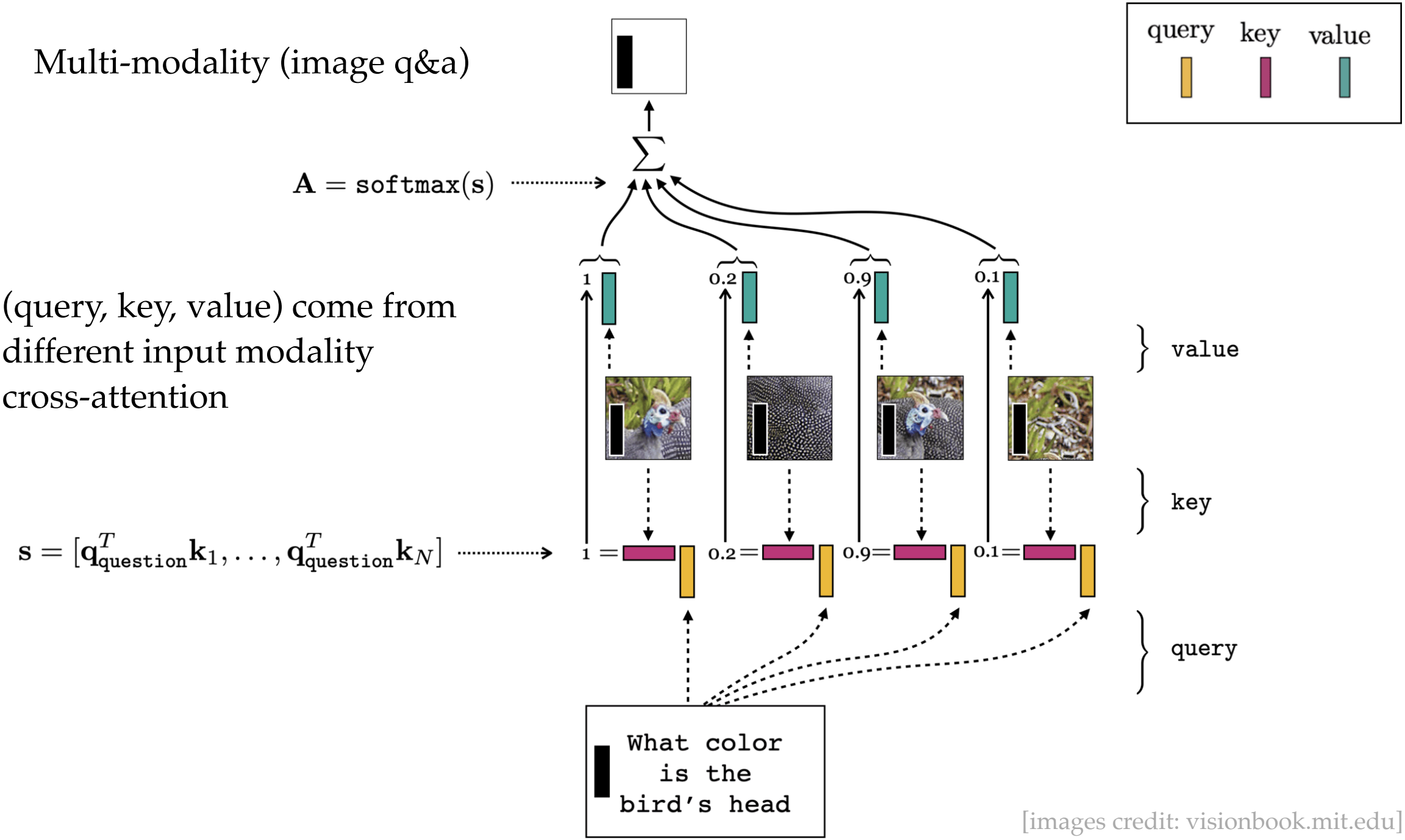




[images credit: visionbook.mit.edu]

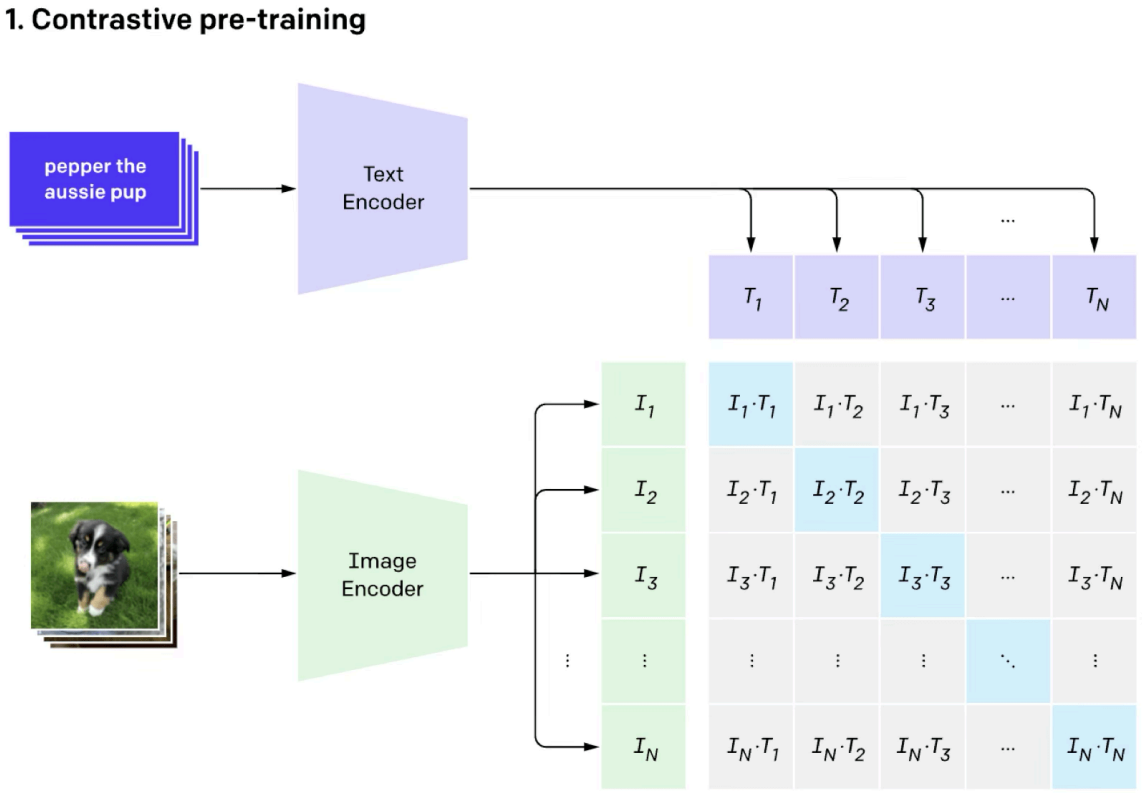
Multi-modality (image q&a)

- (query, key, value) come from different input modality
- cross-attention



[images credit: visionbook.mit.edu]

image classification (done in the contrastive way)




```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

[Radford et al, Learning Transferable Visual Models From Natural Language Supervision, ICML, 2011]

TRANSFORMER PROTEIN LANGUAGE MODELS ARE UNSUPERVISED STRUCTURE LEARNERS

Roshan Rao*
UC Berkeley
rmrao@berkeley.edu

Joshua Meier
Facebook AI Resea
jmeier@fb.com

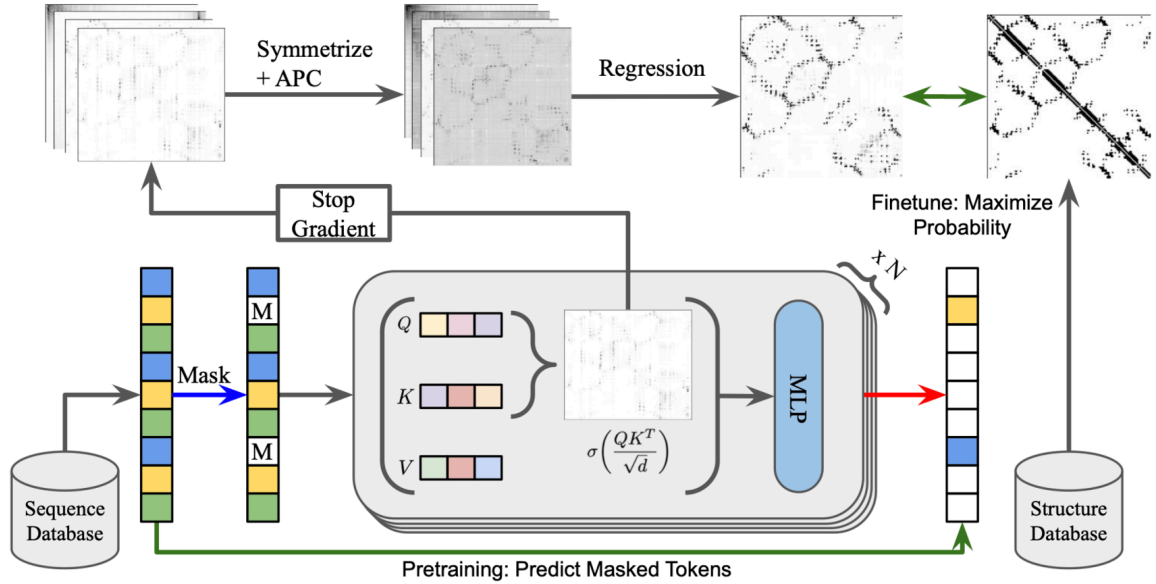


Figure 1: Contact prediction pipeline. The Transformer is first pretrained on sequences from a large database (Uniref50) via Masked Language Modeling. Once finished training, the attention maps are extracted, passed through symmetrization and average product correction, then into a regression. The regression is trained on a small number ($n \leq 20$) of proteins to determine which attention heads are informative. At test time, contact prediction from an input sequence can be done entirely on GPU in a single forward pass.

Unsupervised contact predic
functional constraints for pr
the predominant approach h
related sequences. In the pas
tential alternative, but perfo
in bioinformatics. In this pa
learn contacts from the unsupervised language modeling objective. We find the highest capacity models that have been trained to date already outperform a state-of-the-art unsupervised contact prediction pipeline, suggesting these pipelines can be replaced with a single forward pass of an end-to-end model.[1]

RESEARCH ARTICLE | STRUCTURE PREDICTION

Evolutionary-scale prediction of atomic-level protein structure with a language model

ZEMING LIN, HALIL AKIN, ROSHAN RAO, BRIAN HIE, ZHONGKAI ZHU, WENTING LU, NIKITA SMETANIN, ROBERT VERKUIL, ORI KABELI, [...], AN
ALEXANDER RIVES +5 authors Authors Info & Affiliations

Machine learning methods for protein structure prediction have taken advantage of the evolutionary information present in multiple sequence alignments to derive accurate structural information, but predicting structure accurately from a single sequence is much more difficult. Lin *et al*. trained transformer protein language models with up to 15 billion parameters on experimental and high-quality predicted structures and found that information about atomic-level structure emerged in the model as it was scaled up. They created ESMFold, a sequence-to-structure predictor that is nearly as accurate as alignment-based methods and considerably faster. The increased speed permitted the generation of a database, the ESM Metagenomic Atlas, containing more than 600 million metagenomic proteins. — MAF

nature machine intelligence

Article | Published: 09 June 2025

Human-like object concept representatio naturally in multimodal large language m

Changde Du, Kaicheng Fu, Bincheng Wen, Yi Sun, Jie Peng, Wei Wei, Ying Gao
Chuncheng Zhang, Jinpeng Li, Shuang Qiu, Le Chang & Huiguang He ✉

Nature Machine Intelligence **7**, 860–875 (2025) | Cite this article

Abstract

Understanding how humans conceptualize and categorize natural objects offers critical insights into perception and cognition. With the advent of large language models (LLMs), a key question arises: can these models develop human-like object representations from linguistic and multimodal data? Here we combined behavioural and neuroimaging analyses to explore the relationship between object concept representations in LLMs and human cognition. We collected 4.7 million triplet judgements from LLMs and multimodal LLMs to derive low-dimensional embeddings that capture the similarity structure of 1,854 natural objects. The resulting 66-dimensional embeddings were stable, predictive and exhibited semantic clustering similar to human mental representations. Remarkably, the dimensions underlying these embeddings were interpretable, suggesting that LLMs and multimodal LLMs develop human-like conceptual representations of objects. Further analysis showed strong alignment between model embeddings and neural activity patterns in brain regions such as the extrastriate body area, parahippocampal place area, retrosplenial cortex and fusiform face area. This provides compelling evidence that the object representations in LLMs, although not identical to human ones, share fundamental similarities that reflect key aspects of human conceptual knowledge. Our findings advance the understanding of machine intelligence and inform the development of more human-like artificial cognitive systems.

Success mode:

[“DINO”, Caron et all. 2021]

Success mode:

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)]

Failure mode:

A large white bird standing in a forest.

A woman holding a clock in her hand.

[Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Xu et al. CVPR (2016)]

Success or Failure? mode:

Giannis Daras ✈️ NeurIPS 2023
@giannis_daras

DALLE-2 has a secret language.
"Apoploe vesrreaitais" means birds.
"Contarra ccetnxniams luryca tanniounons" means bugs or pests.

The prompt: "Apoploe vesrreaitais eating Contarra ccetnxniams luryca tanniounons" gives images of birds eating bugs.

A thread (1/n)🧵

Another example: "Two whales talking about food, with subtitles". We get an image with the text "Wa ch zod rea" written on it. Apparently, the whales are actually talking about their food in the DALLE-2 language. (4/n)

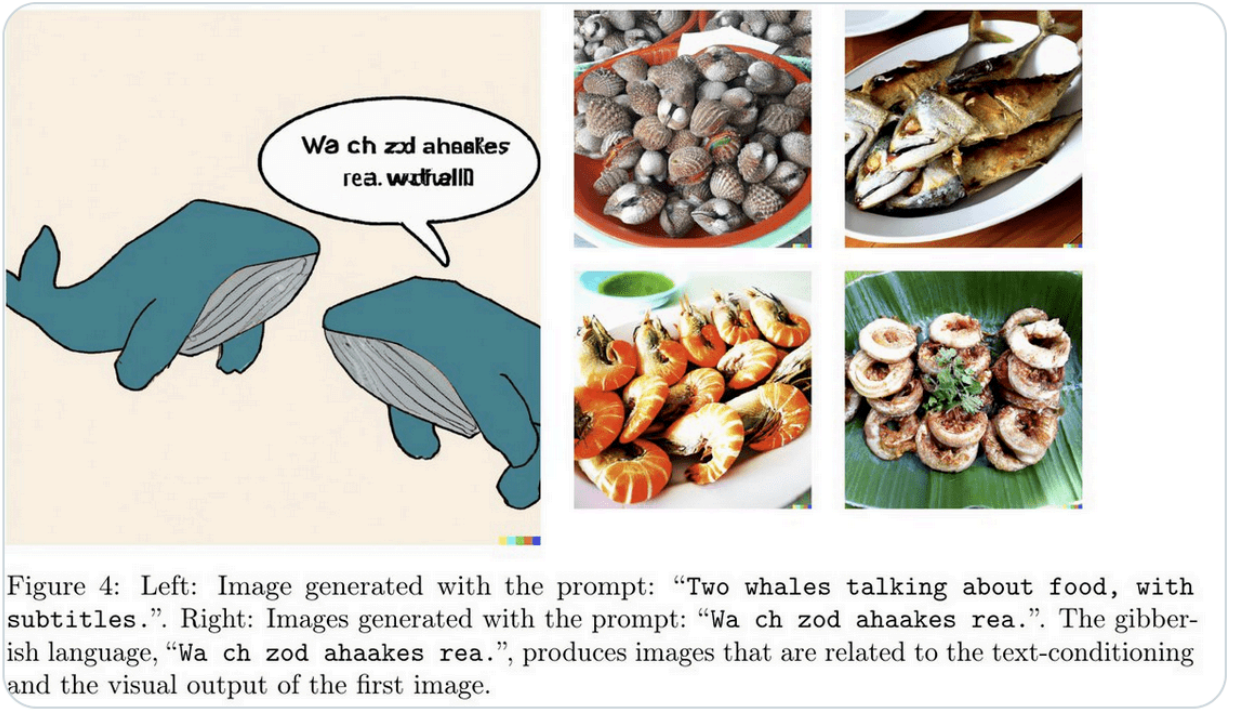


Figure 4: Left: Image generated with the prompt: "`Two whales talking about food, with subtitles.`". Right: Images generated with the prompt: "`Wa ch zod ahaakes rea.`". The gibberish language, "`Wa ch zod ahaakes rea.`", produces images that are related to the text-conditioning and the visual output of the first image.

Summary

- Transformers combine ideas from earlier architectures (convolution, ReLU, residual connections) with new innovations: embedding and attention layers.
- Transformers start with generic hard-coded **embeddings** and, block-by-block, create increasingly context-aware embeddings.
- **Attention** enables massive parallelism: each head, each q, k, v sequence, each attention score, and each output are all computed in parallel.
- (Because the architecture is input-agnostic — "attention is all you need" — transformers have become one model that rules across language, vision, and multi-modal tasks.)