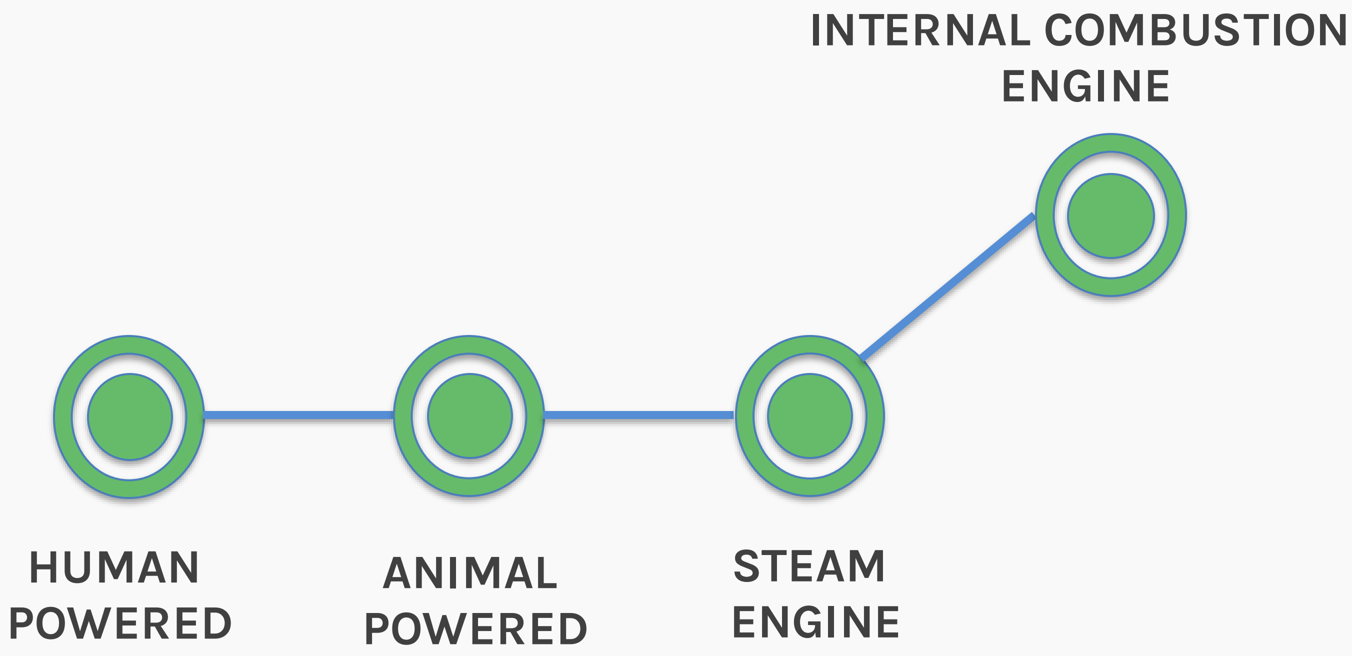
Transformer

Pavlos Protopapas

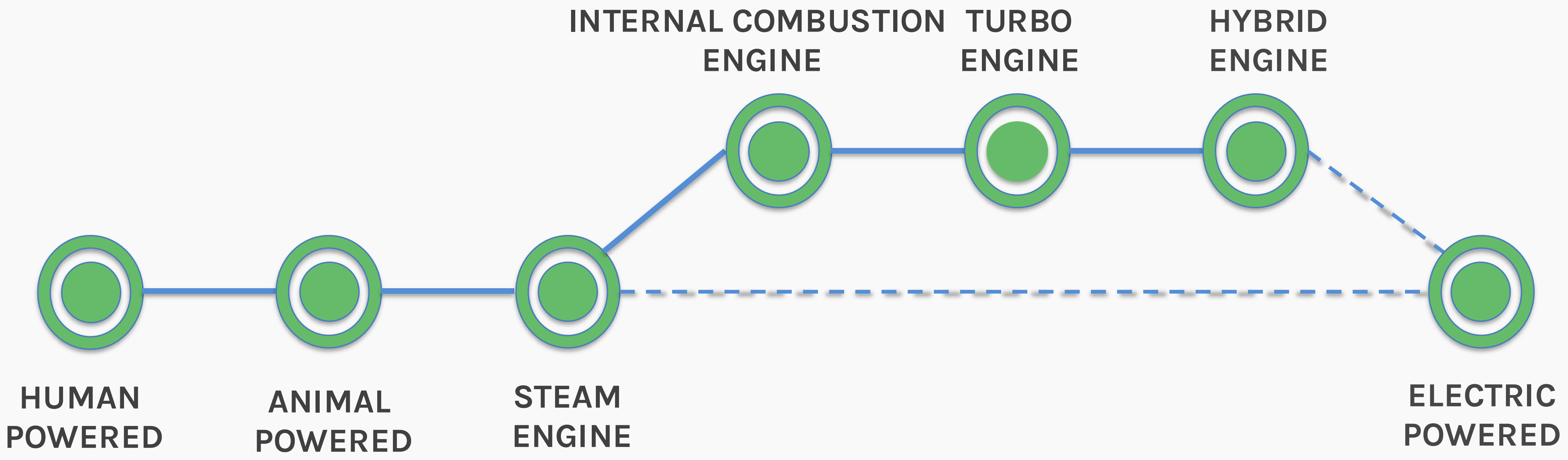
Outline

- **Motivation for Attention**
- Attention Basics
  - Using cosine similarity as a tool for contextual relations
  - Self-Attention
- Building blocks of Transformers and BERT
  - Multi-head attention block
  - Positional Encoding
  - Bringing it all together

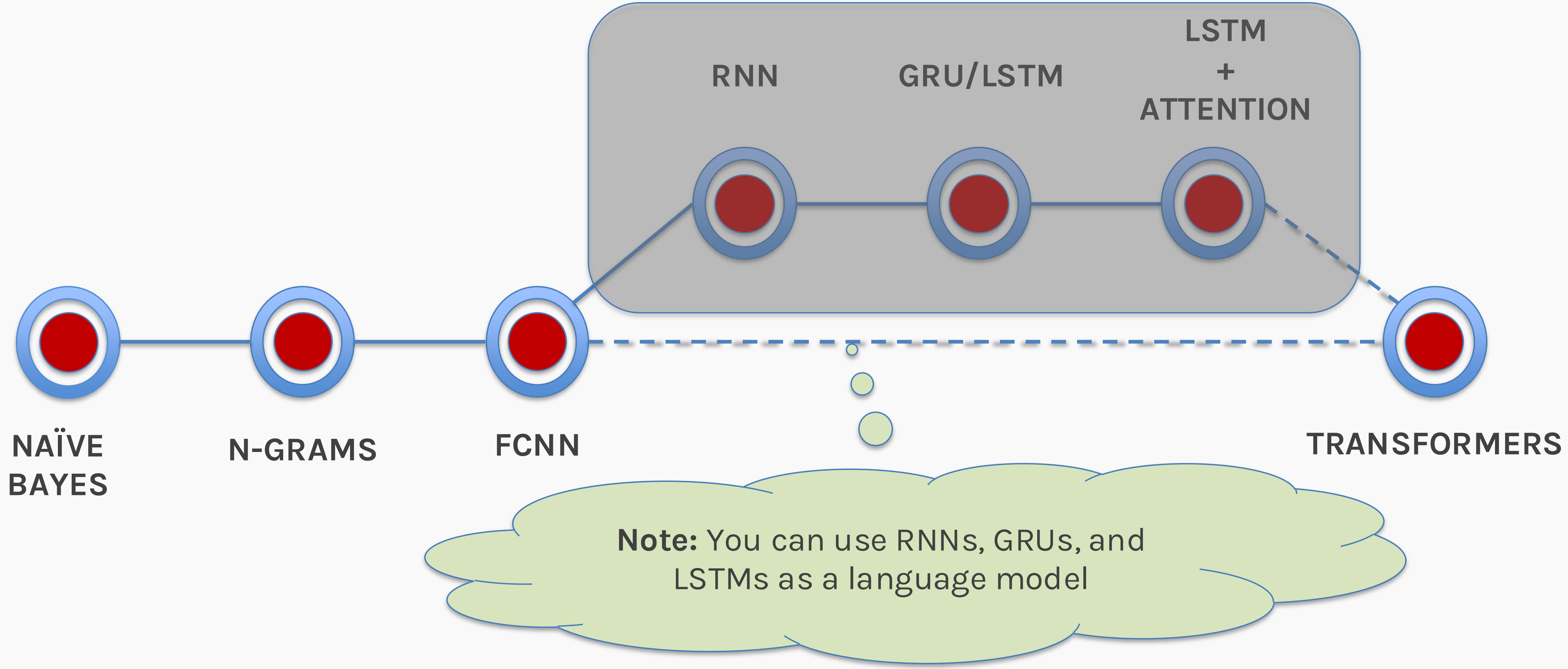
Electricity is all you need



Electricity is all you need



Language modeling



Limitations

Language Model Wishlist?

- Need long context
- We want to have strong contextual relations between words

FCNN have fixed context.

Word2Vec gives us one embedding per word

Ambiguity in static embeddings

The **bank** is open on Fridays.

I went to the **bank** to take a walk by the river.

Limitations

Language Model Wishlist?

- Need long context
- We want to have strong contextual relations between words
  - We want words to have sequential information
  - We need an architecture that can be trained in parallel (non-Markovian property)

Wo

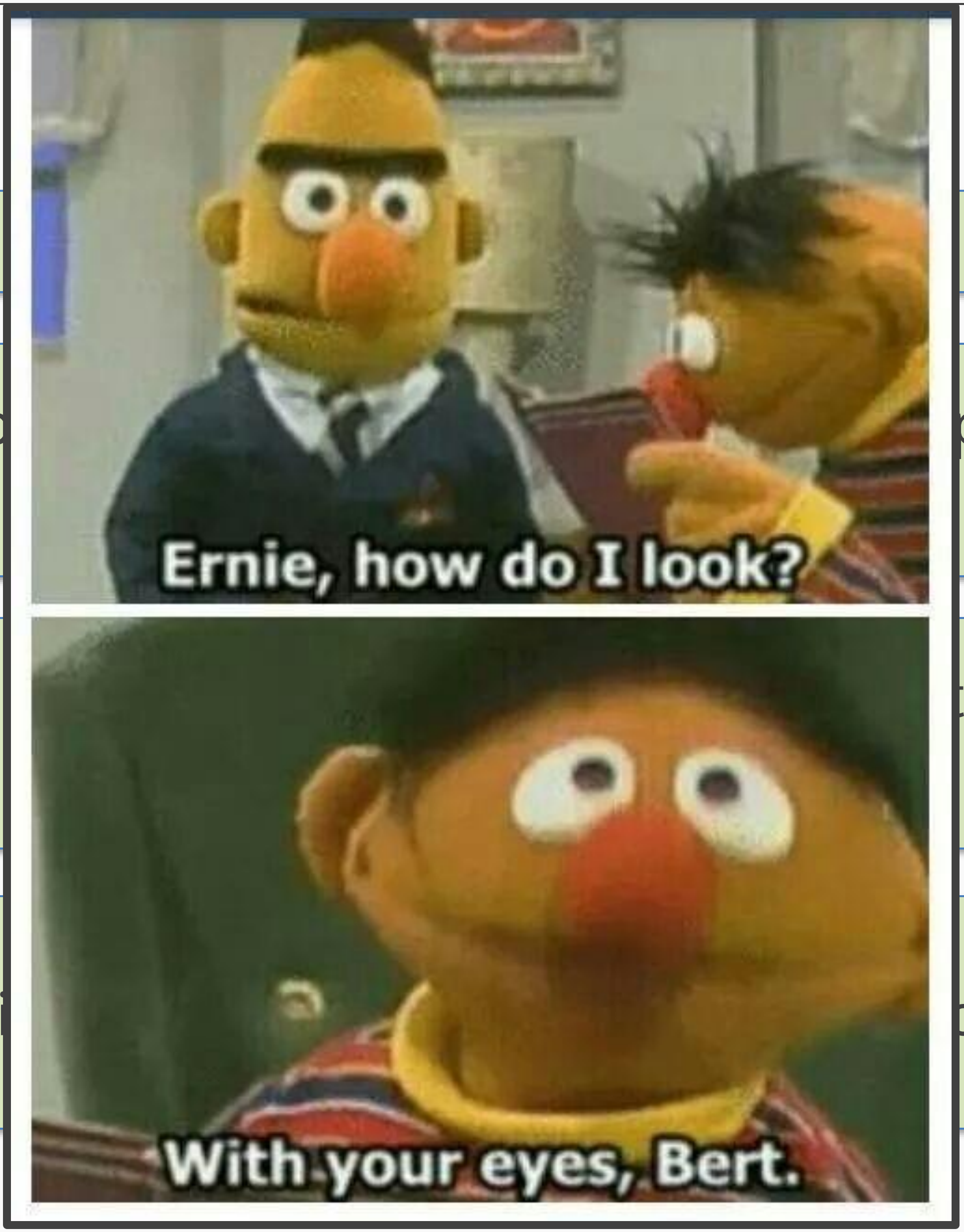
per

FC

tial

trai

ous



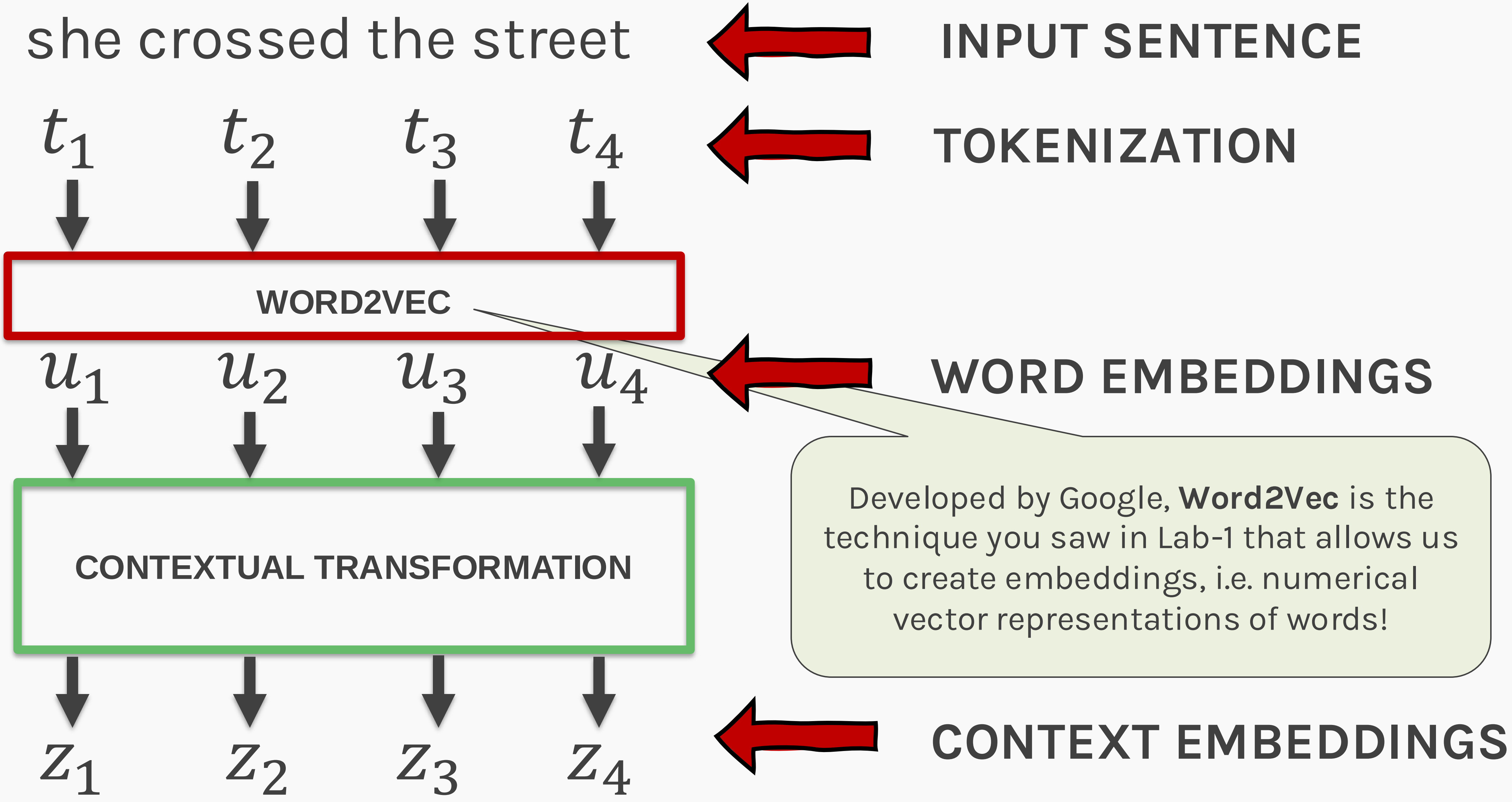
Attention: Example

Amelia was hit by a bus because $\boxed{\text{she}}$ crossed the street

How do we find the context of the word ‘she’ in the sentence?

No teaching staff were harmed during the production of this lecture. This is a work of fiction. Names, characters, places and incidents either are products of the professor’s imagination or are used fictitiously. Any resemblance to actual events or locales or persons, living or dead, is entirely coincidental.

Contextual Embeddings



Attention: Example

💡IDEA #1: Distance relationship

Amelia was hit by a bus because she crossed the street

True context

🚫This idea does not work because context can be unevenly spread out in a sentence

Attention: Example

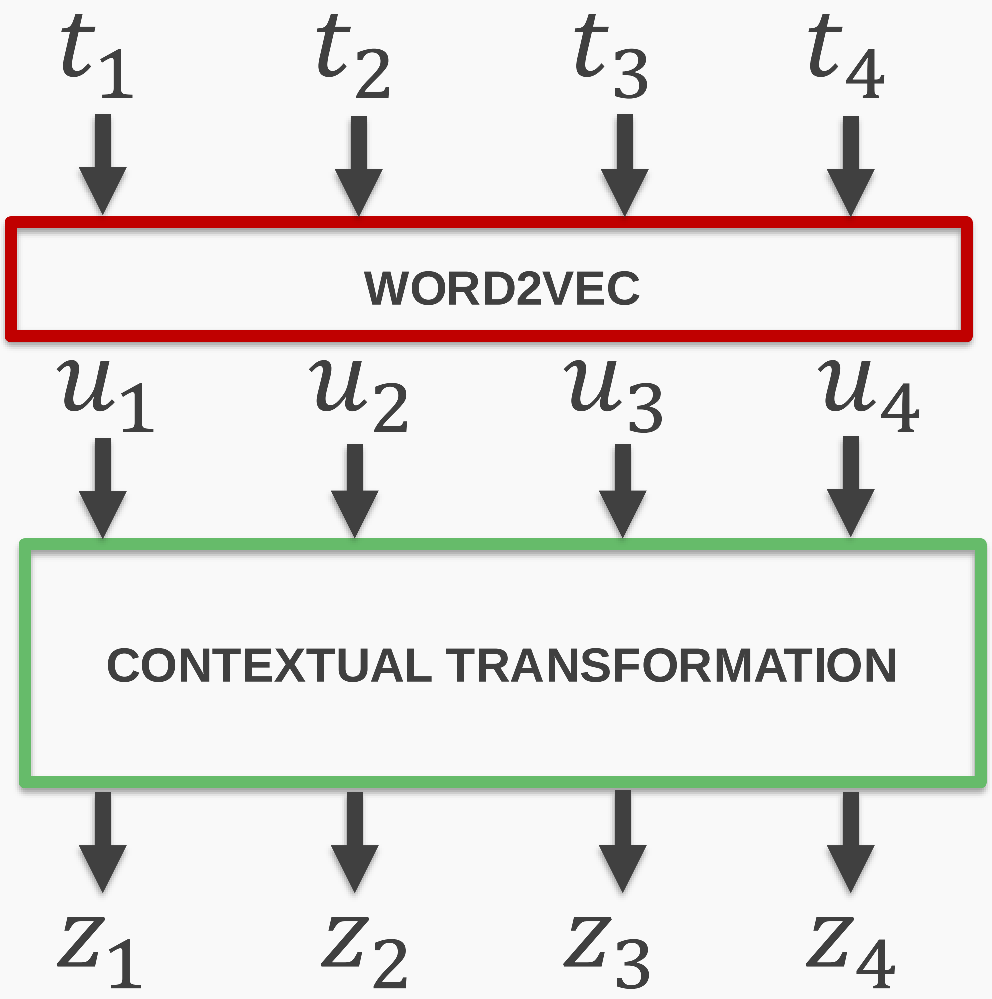
Amelia was hit by a bus because $\boxed{\text{she}}$ crossed the street

We would want the word ‘she’ to be related to every other word in the sentence, irrespective of the distance.

One way to do that would be to make the contextual embedding of the word ‘she’ to be a linear combination of the other word embeddings.

Contextual Embeddings

she crossed the street



Contextual embeddings, z, can be obtained by applying a linear transformation to the word embeddings, u.

So, we want

$$z_1 = \alpha_{11}u_1 + \alpha_{12}u_2 + \alpha_{13}u_3 + \alpha_{14}u_4$$

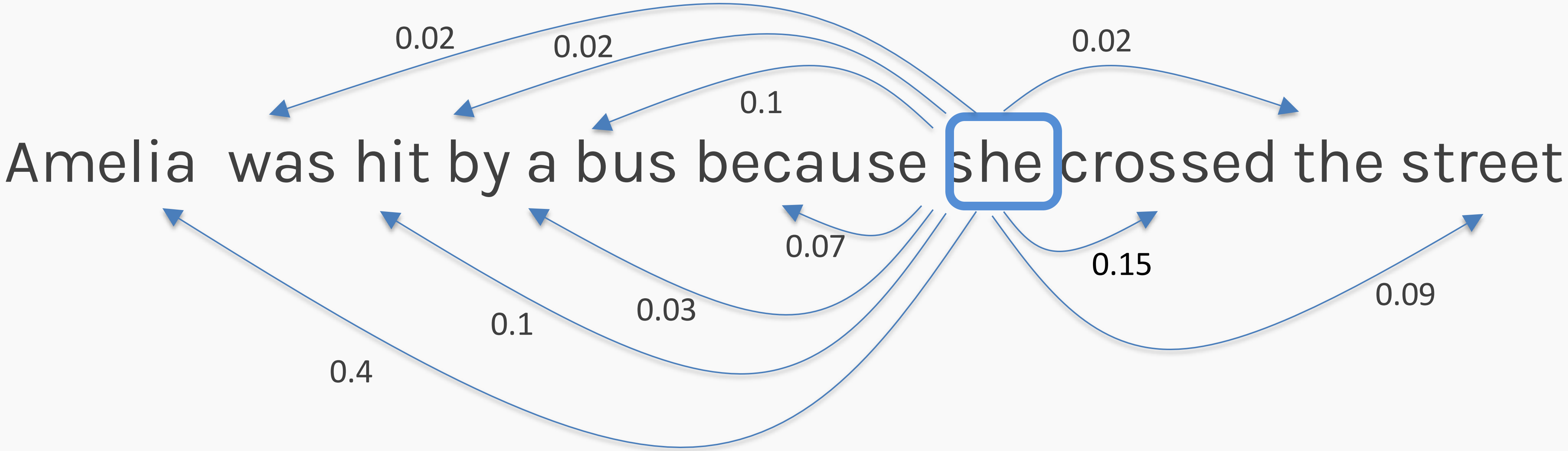
$$z_2 = \alpha_{21}u_1 + \alpha_{22}u_2 + \alpha_{23}u_3 + \alpha_{24}u_4$$

$$z_3 = \alpha_{31}u_1 + \alpha_{32}u_2 + \alpha_{33}u_3 + \alpha_{34}u_4$$

$$z_4 = \alpha_{41}u_1 + \alpha_{42}u_2 + \alpha_{43}u_3 + \alpha_{44}u_4$$

Contextual Embeddings: Coefficients

The coefficients are weights that encapsulates the degree to which new contextual embeddings should incorporate the influence of the other tokens



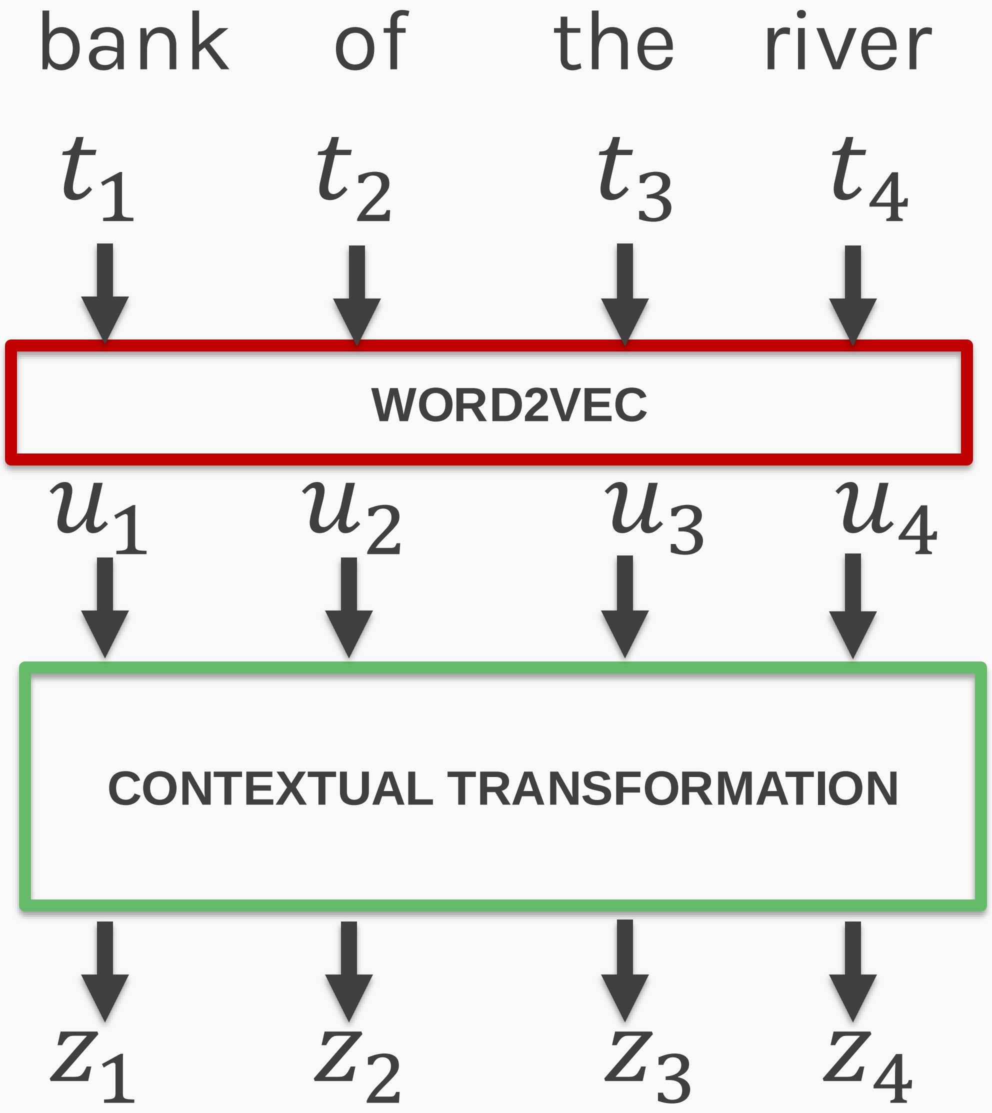
Contextual Embeddings: Coefficients

How do we calculate these coefficients?

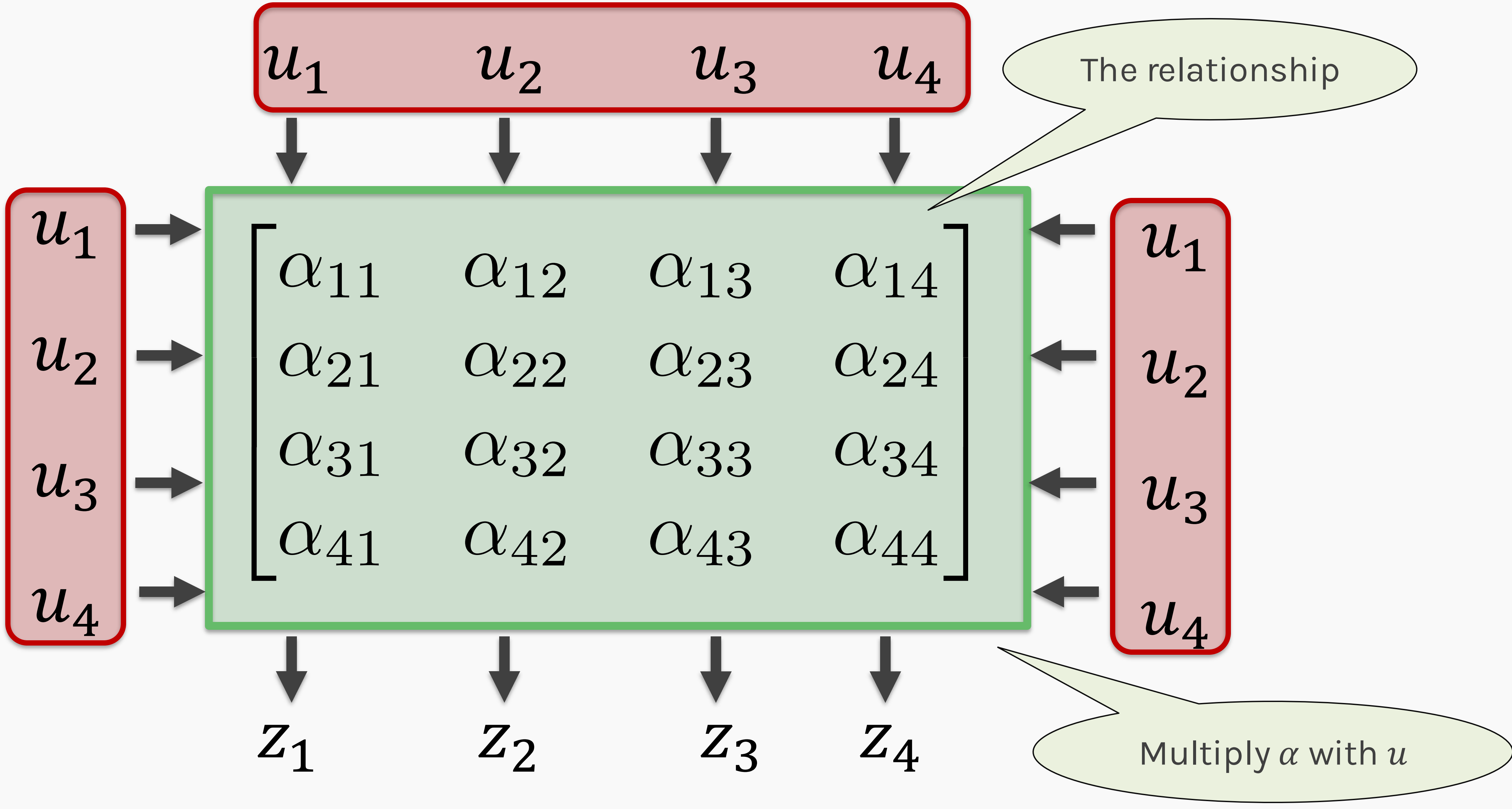
Attention: Example #2

Amelia was standing next to the $\boxed{\text{bank of the river}}$

Attention: Example #2

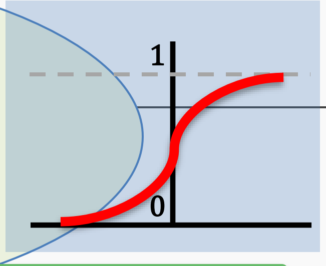


Attention: Behind the scenes



Attention: Behind the scenes

Softmax is a function that scales numbers (logits) into probabilities



$$u_1 \circ u_1 = a_{11}$$

$$u_1 \circ u_2 = a_{12}$$

$$u_1 \circ u_3 = a_{13}$$

$$u_1 \circ u_4 = a_{14}$$

$$\xrightarrow{\mathbf{softmax}(.)}$$

WEIGHTS NORMALIZATION

$$\alpha_{11}$$

$$\alpha_{12}$$

$$\alpha_{13}$$

$$\alpha_{14}$$

$$\boxed{\Sigma\alpha_{1i} = 1}$$

Cosine similarity

Attention: Behind the scenes

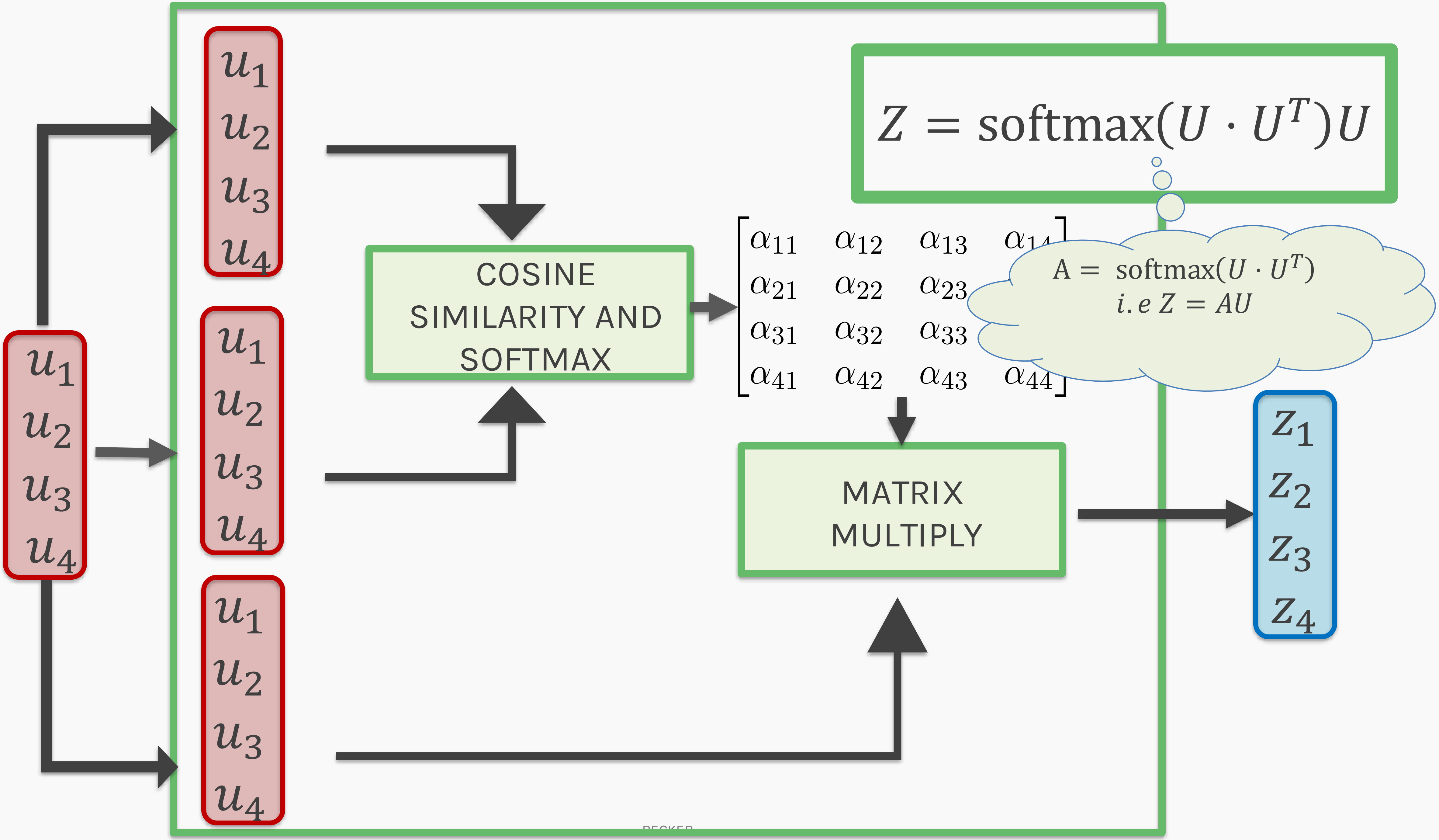
Similarly…

$$z_1 = \alpha_{11}u_1 + \alpha_{12}u_2 + \alpha_{13}u_3 + \alpha_{14}u_4$$

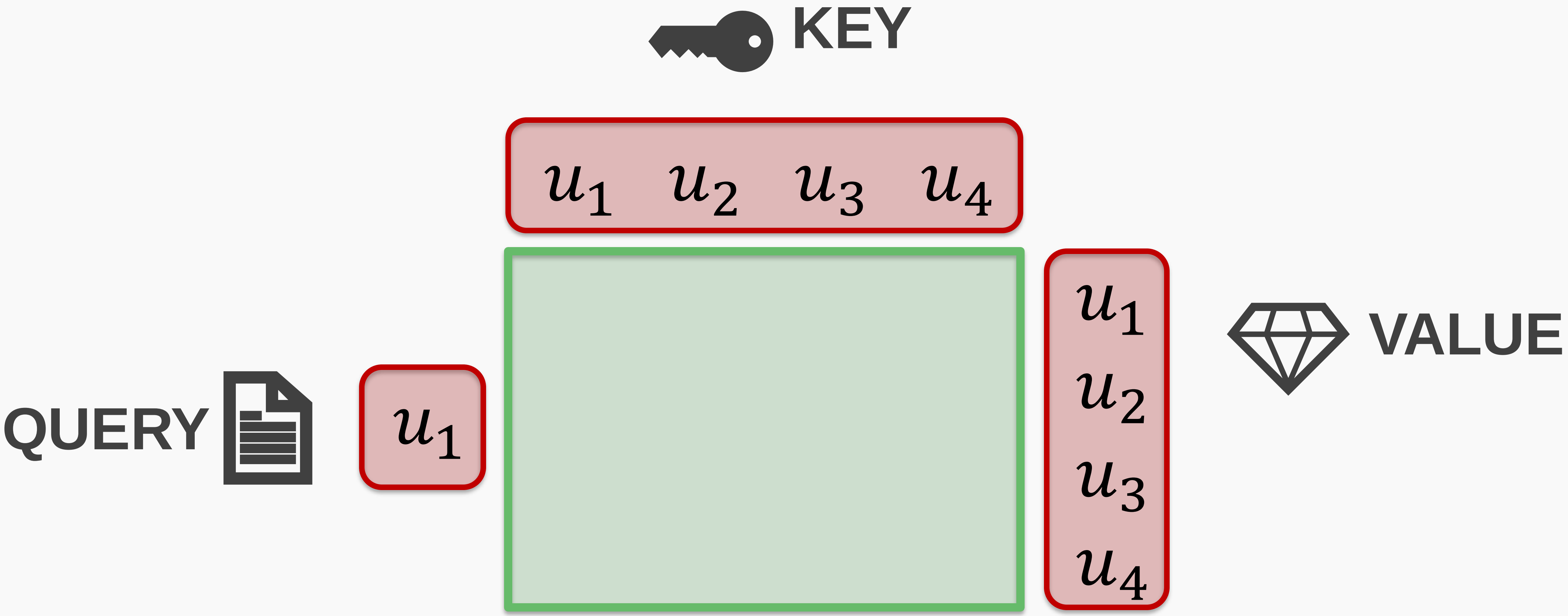
$$z_2 = \alpha_{21}u_1 + \alpha_{22}u_2 + \alpha_{23}u_3 + \alpha_{24}u_4$$

$$z_3 = \alpha_{31}u_1 + \alpha_{32}u_2 + \alpha_{33}u_3 + \alpha_{34}u_4$$

$$z_4 = \alpha_{41}u_1 + \alpha_{42}u_2 + \alpha_{43}u_3 + \alpha_{44}u_4$$


$Z = \mathrm{softmax}(U \cdot U^T)U$
u_1 u_2 u_3 u_4
u_1 u_2 u_3 u_4
u_1 u_2 u_3 u_4
u_1 u_2 u_3 u_4
COSINE SIMILARITY AND SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
$A = \mathrm{softmax}(U \cdot U^T)$
$i.e\ Z = AU$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4

Attention: Database Analogy



Attention: Database Analogy

For simplicity, we stick to our database analogy of **QUERY, KEY & VALUE**

KEY

$u_1 \quad u_2 \quad u_3 \quad u_4$

QUERY

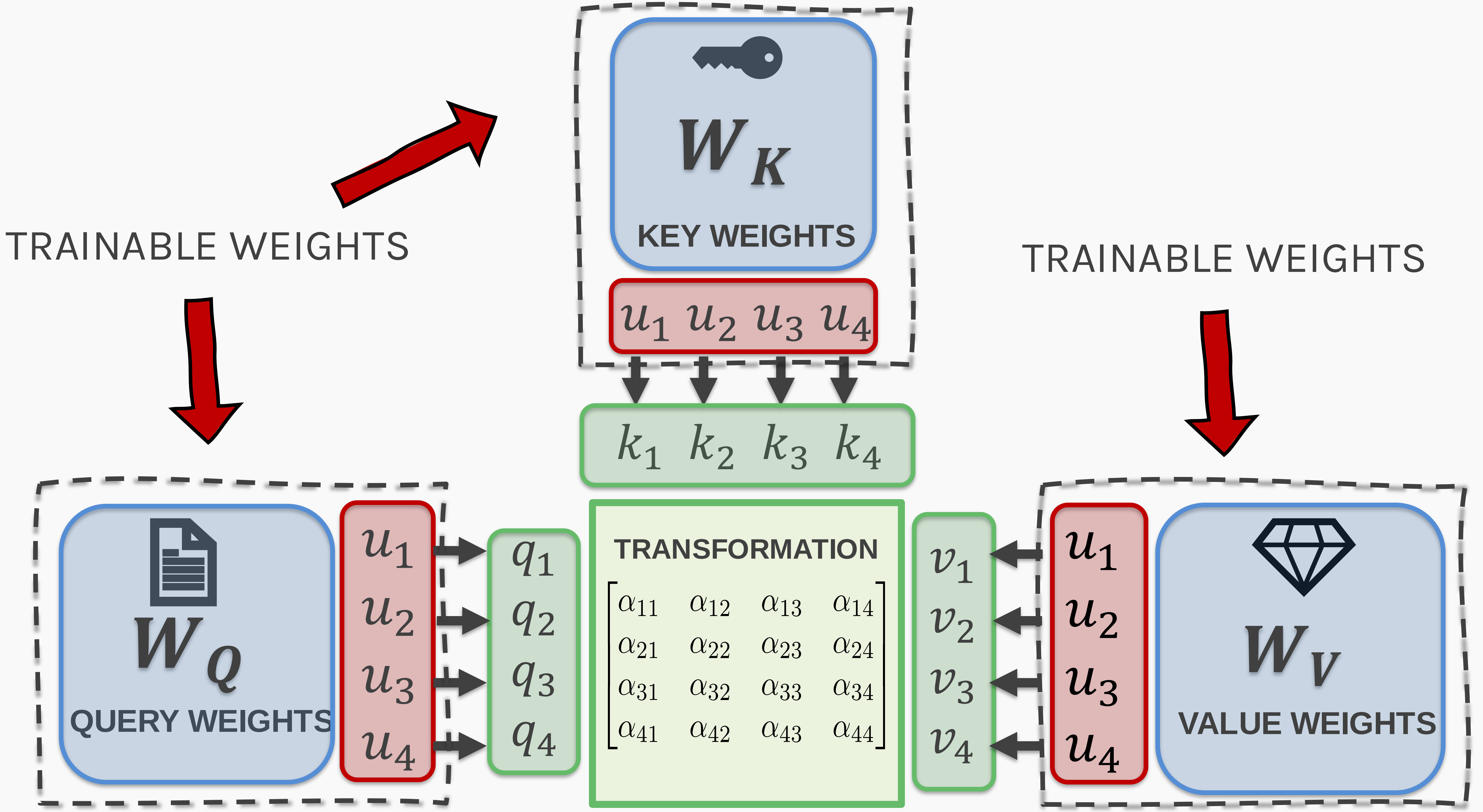
$$\begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{matrix}$$

TRANSFORMATION

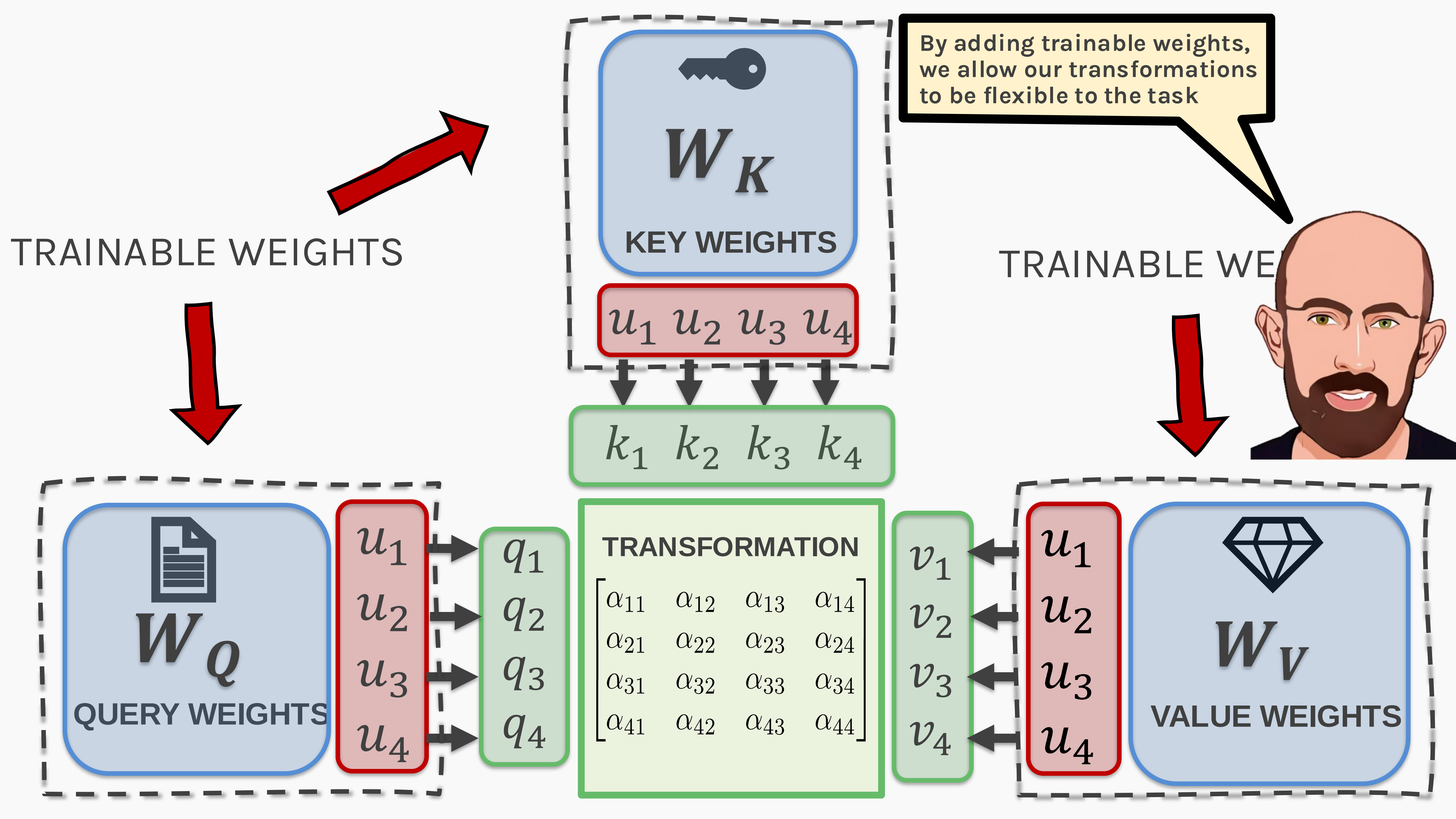
$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$$

VALUE

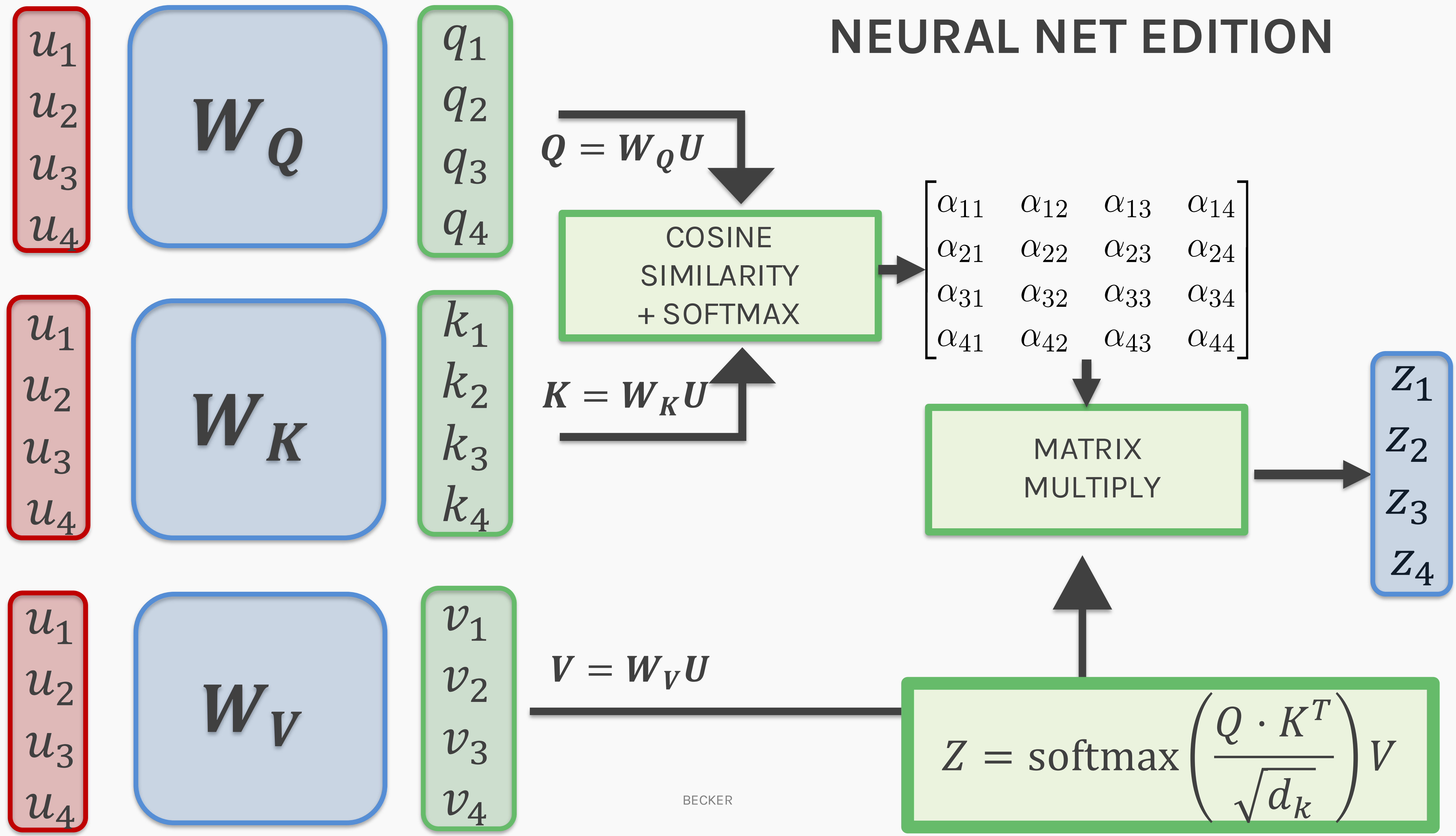
$$\begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{matrix}$$



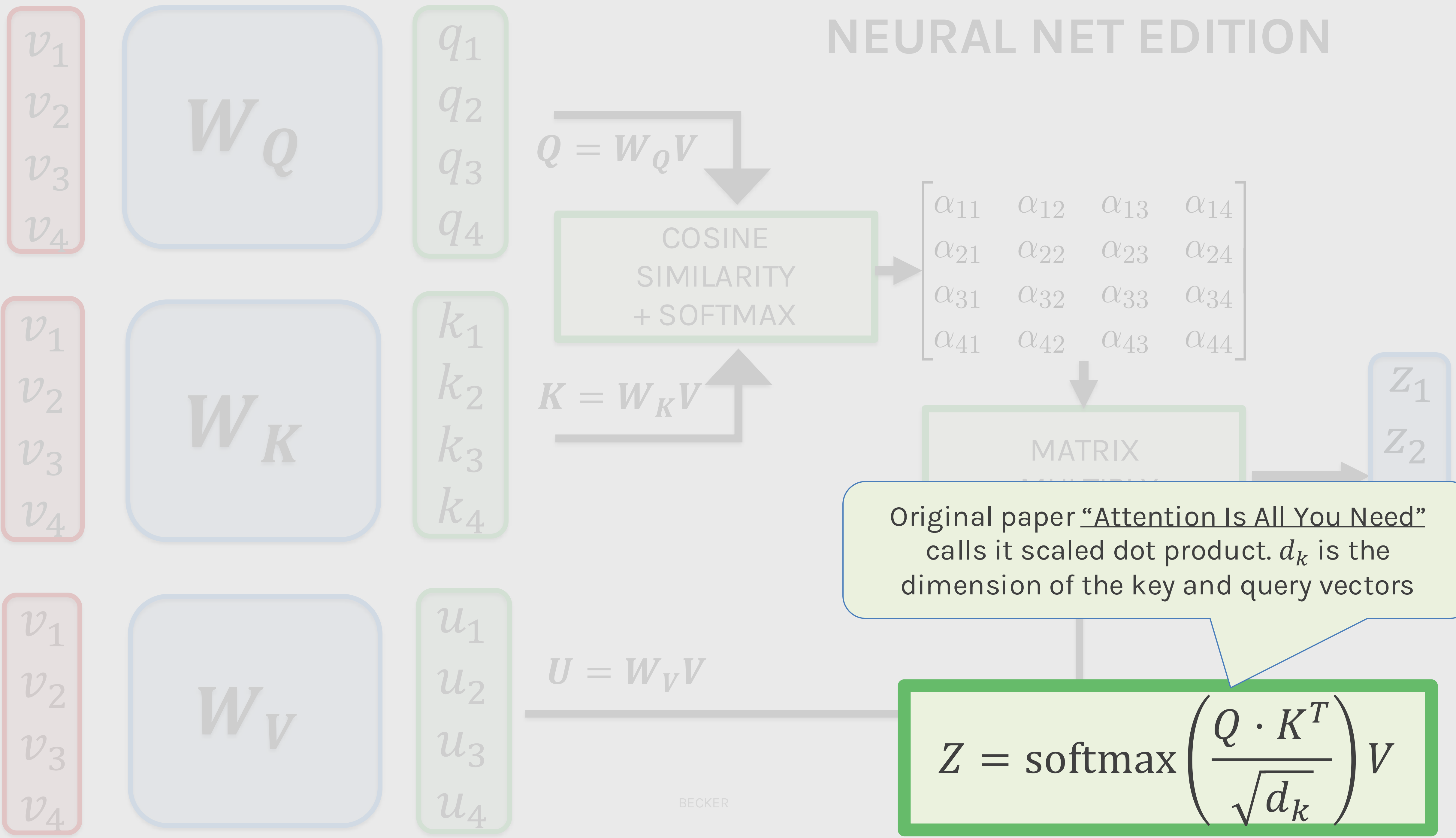
TRAINABLE WEIGHTS
W_K
KEY WEIGHTS
u_1 u_2 u_3 u_4
k_1 k_2 k_3 k_4
TRAINABLE WEIGHTS
W_Q
QUERY WEIGHTS
u_1 u_2 u_3 u_4
q_1 q_2 q_3 q_4
TRANSFORMATION
α_{11} α_{12} α_{13} α_{14}
α_{21} α_{22} α_{23} α_{24}
α_{31} α_{32} α_{33} α_{34}
α_{41} α_{42} α_{43} α_{44}
v_1 v_2 v_3 v_4
u_1 u_2 u_3 u_4
W_V
VALUE WEIGHTS



By adding trainable weights, we allow our transformations to be flexible to the task
TRAINABLE WEIGHTS
TRAINABLE WE
W_K
KEY WEIGHTS
u_1 u_2 u_3 u_4
k_1 k_2 k_3 k_4
W_Q
QUERY WEIGHTS
q_1 q_2 q_3 q_4
TRANSFORMATION
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
v_1 v_2 v_3 v_4
W_V
VALUE WEIGHTS

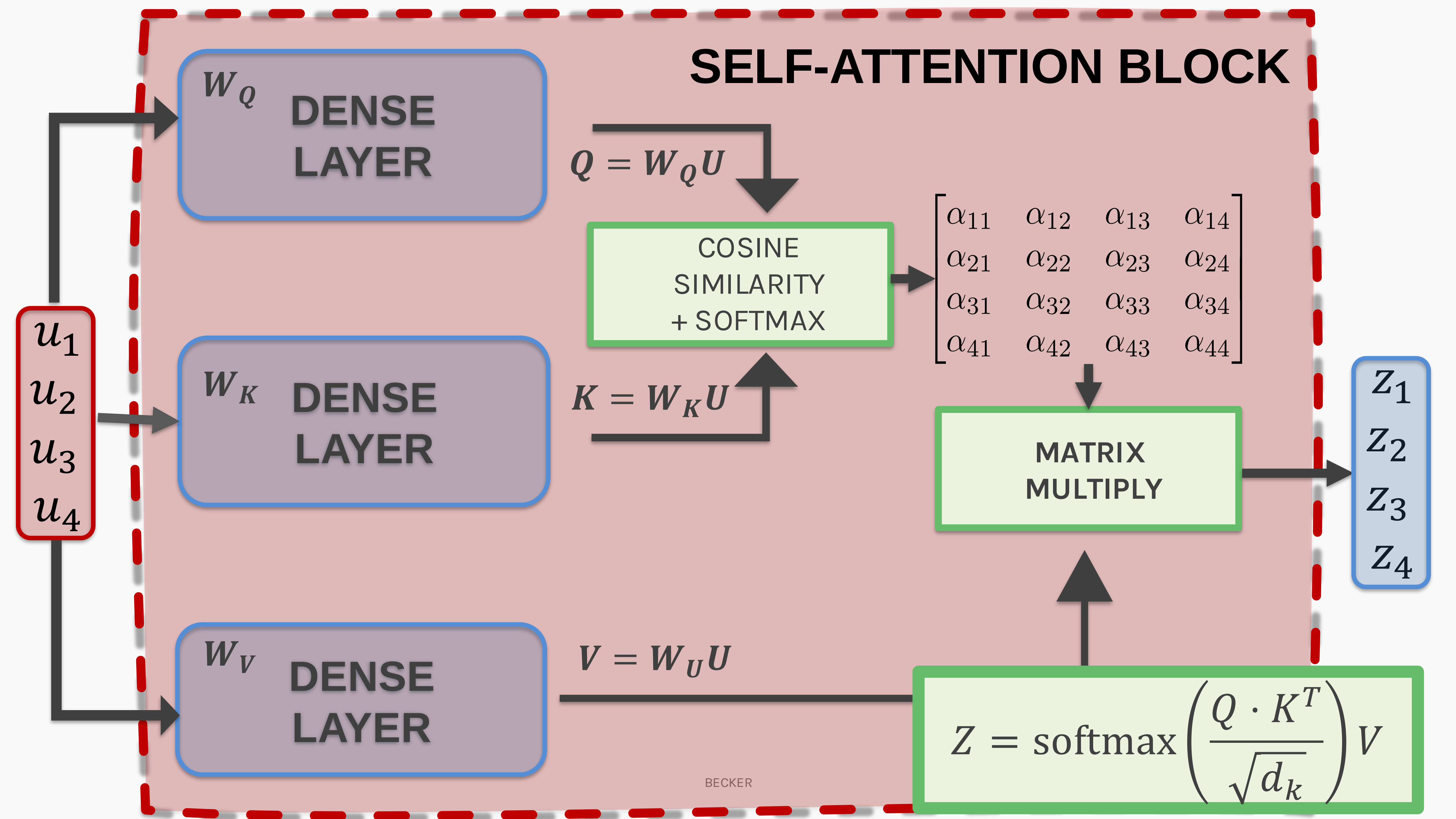

NEURAL NET EDITION
u_1 u_2 u_3 u_4
W_Q
q_1 q_2 q_3 q_4
u_1 u_2 u_3 u_4
W_K
k_1 k_2 k_3 k_4
u_1 u_2 u_3 u_4
W_V
v_1 v_2 v_3 v_4
$Q = W_Q U$
$K = W_K U$
$V = W_V U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V$
BECKER

NEURAL NET EDITION



Original paper "Attention Is All You Need" calls it scaled dot product. d_k is the dimension of the key and query vectors

$$Z = \mathrm{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) V$$


SELF-ATTENTION BLOCK
W_Q
DENSE LAYER
W_K
DENSE LAYER
W_V
DENSE LAYER
u_1 u_2 u_3 u_4
$Q = W_Q U$
$K = W_K U$
$V = W_U U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V$
BECKER

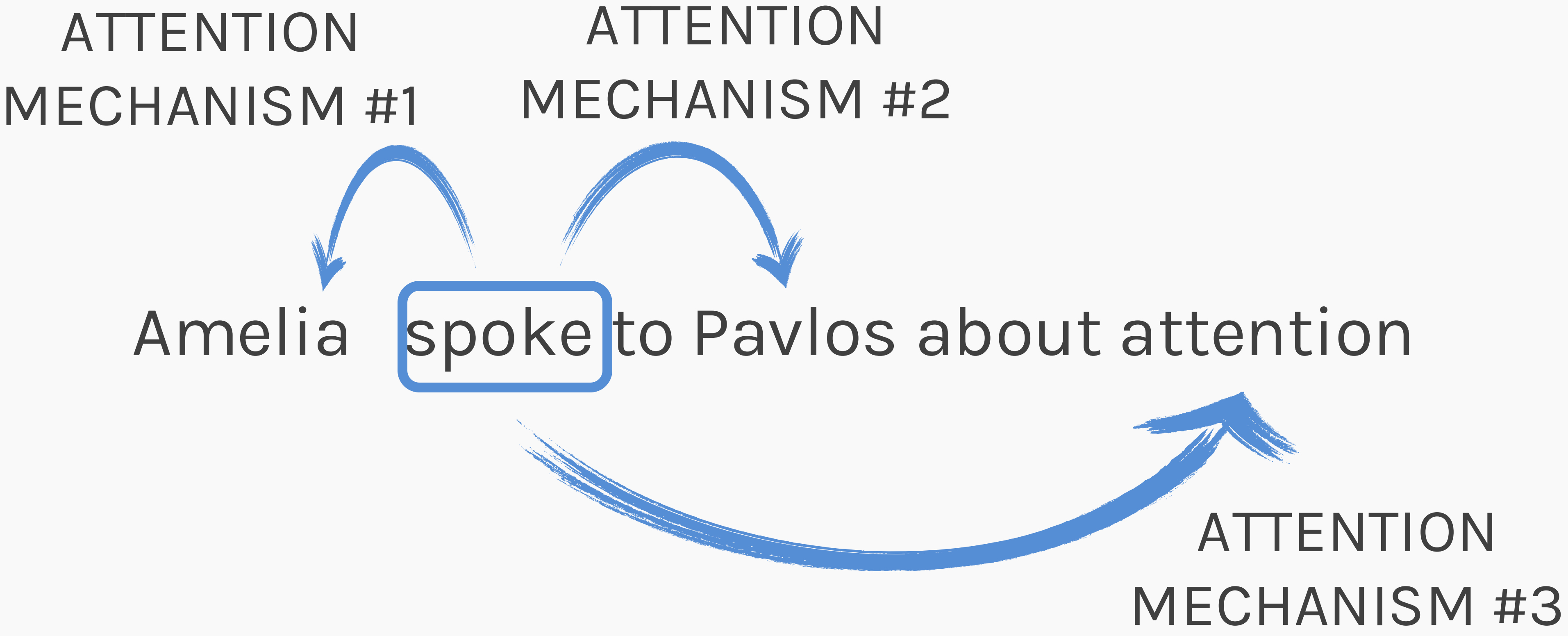
Attention

ATTENTION ISSUES?

- ~~No weights trained in the process~~
- **Attention leads to limited contextual mapping.**
- There is no positional information encoded.

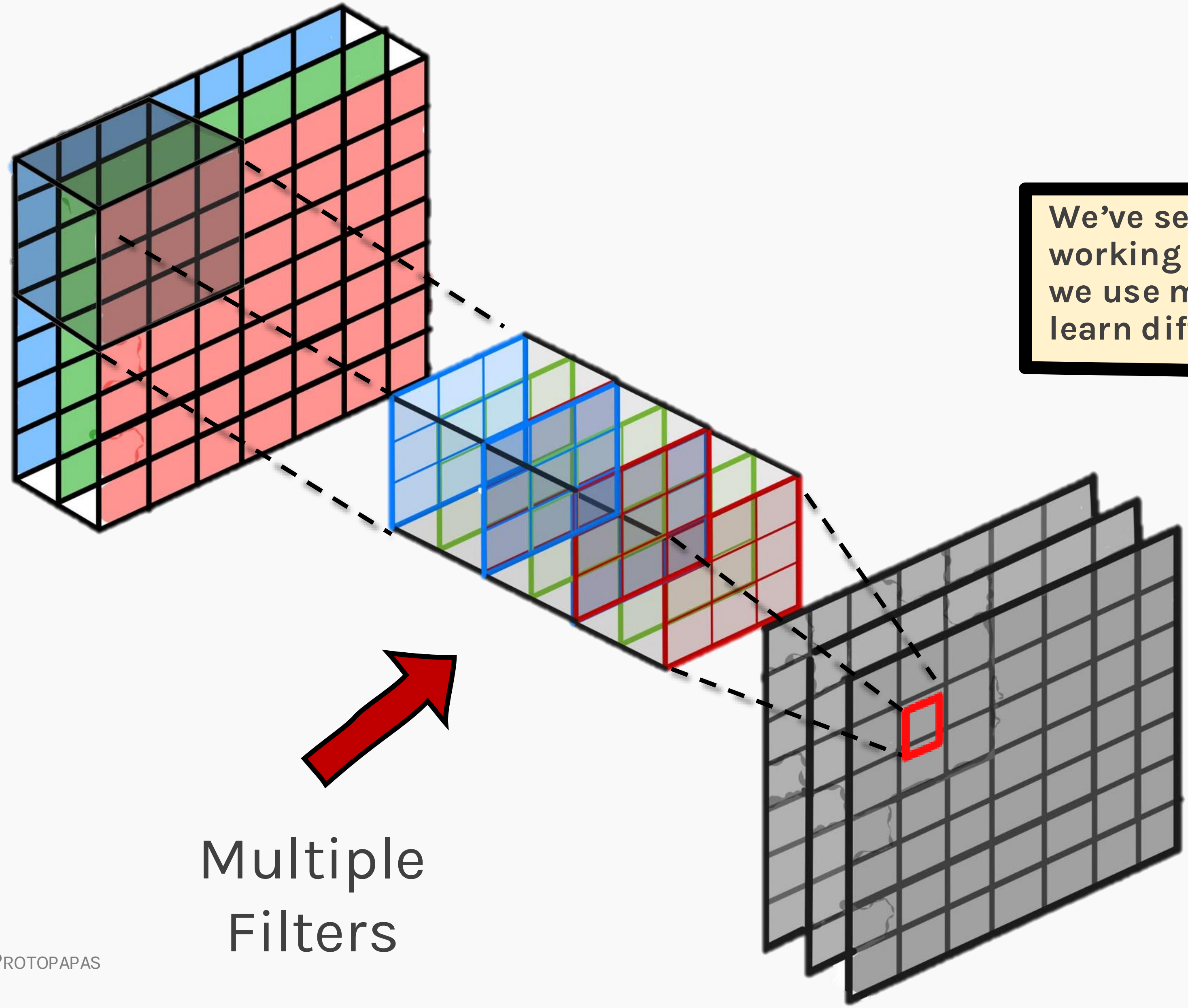
Amelia $\boxed{\text{spoke}}$ to Pavlos about attention

Attention: Do we have enough attention?



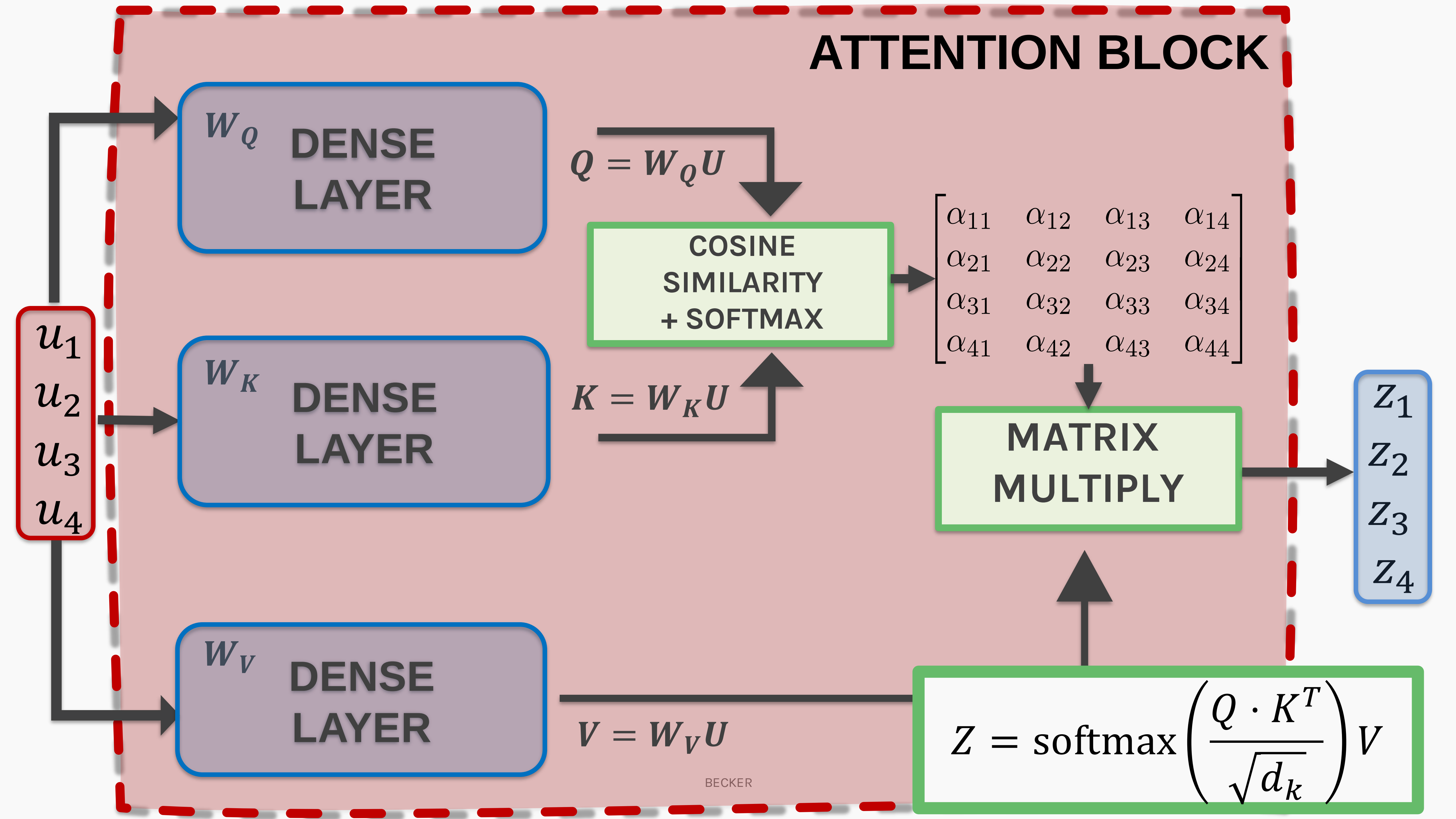
We need to have multiple attention mechanisms to look for different relations.

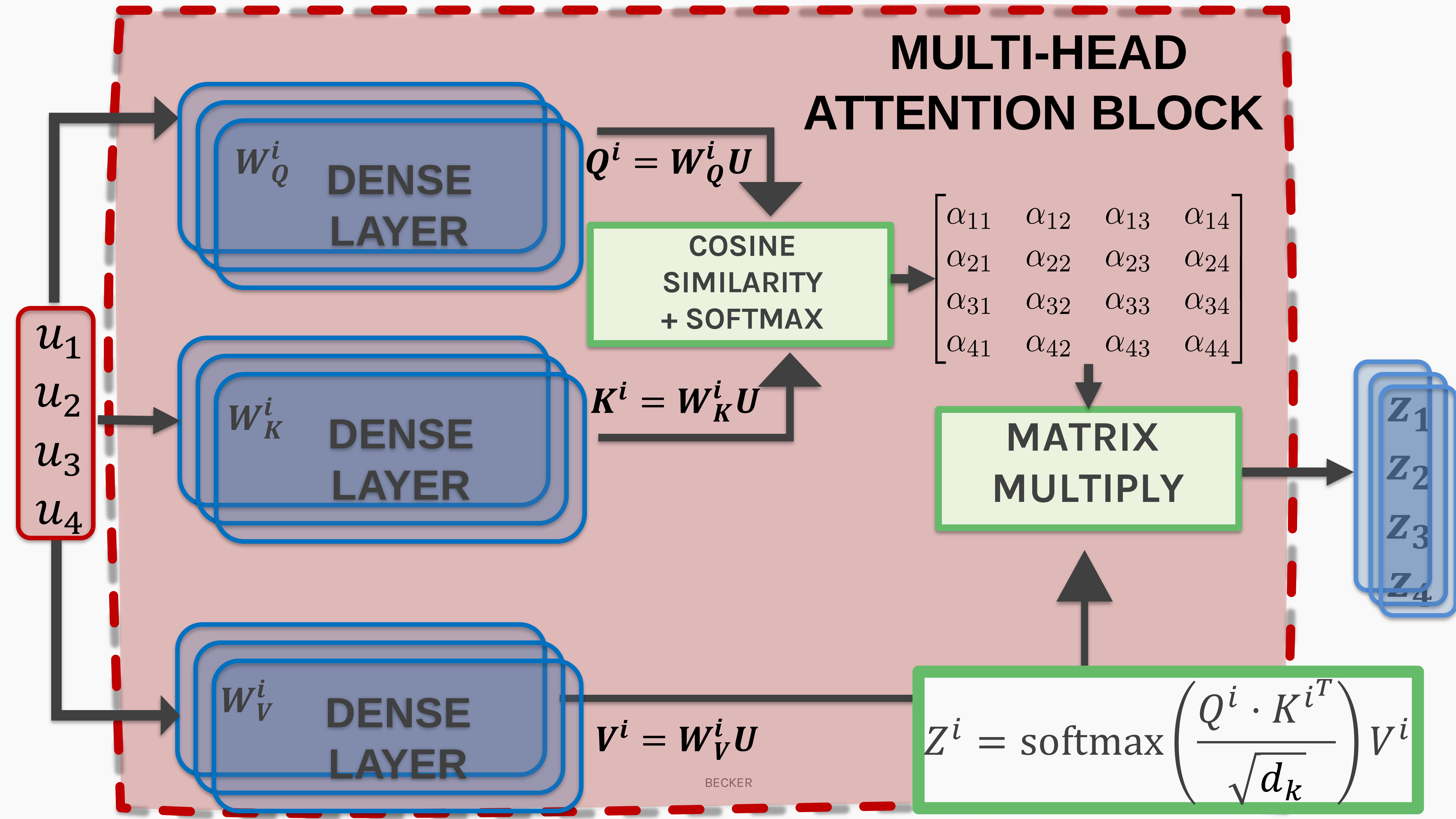
Analogy - CNN Filters



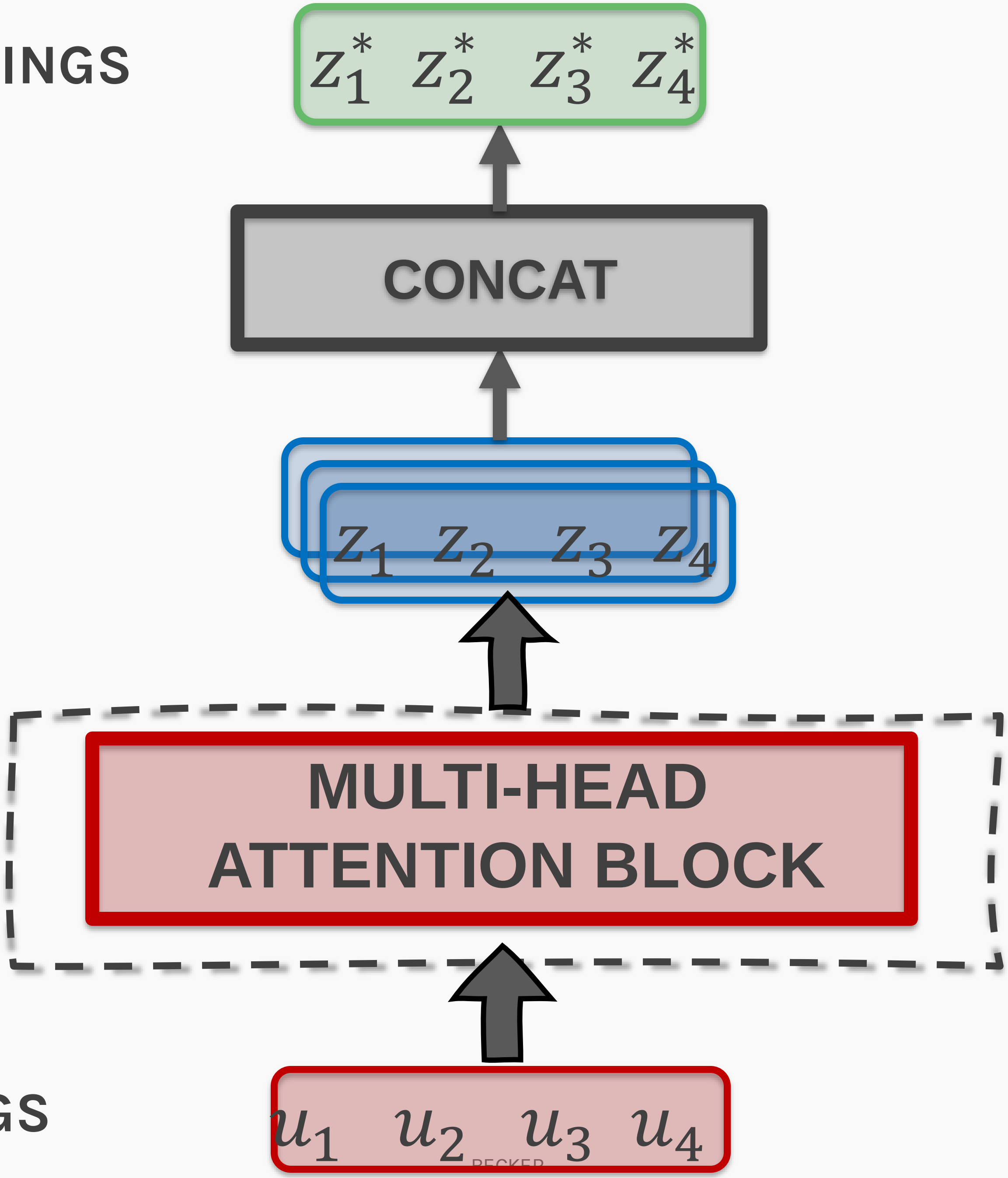
Multiple Filters

We've seen this before while working with CNNs, where we use multiple filters to learn different features

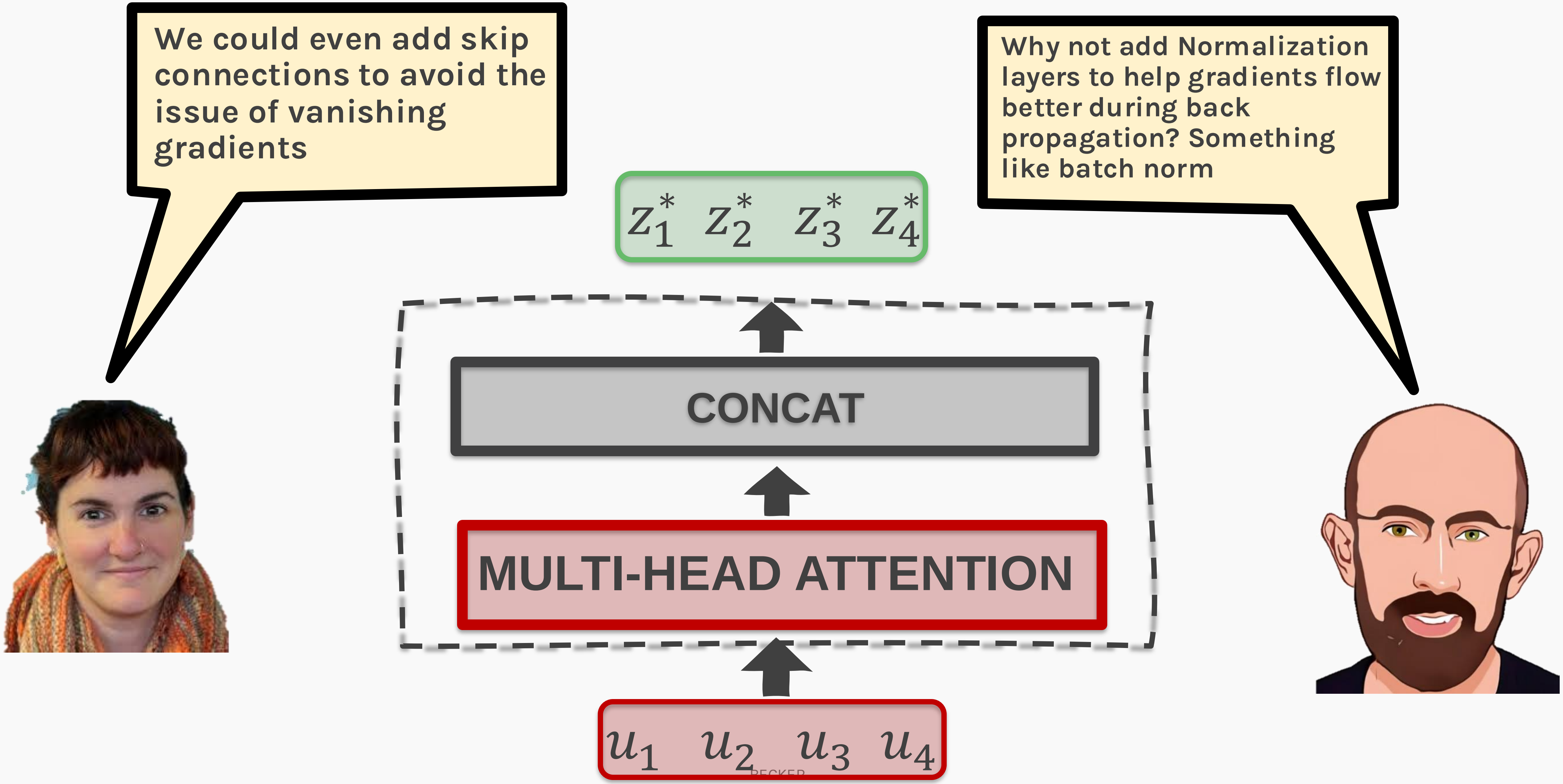

ATTENTION BLOCK
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
u_1 u_2 u_3 u_4
$Q = W_Q U$
$K = W_K U$
$V = W_V U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4
$Z = \mathrm{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V$
BECKER

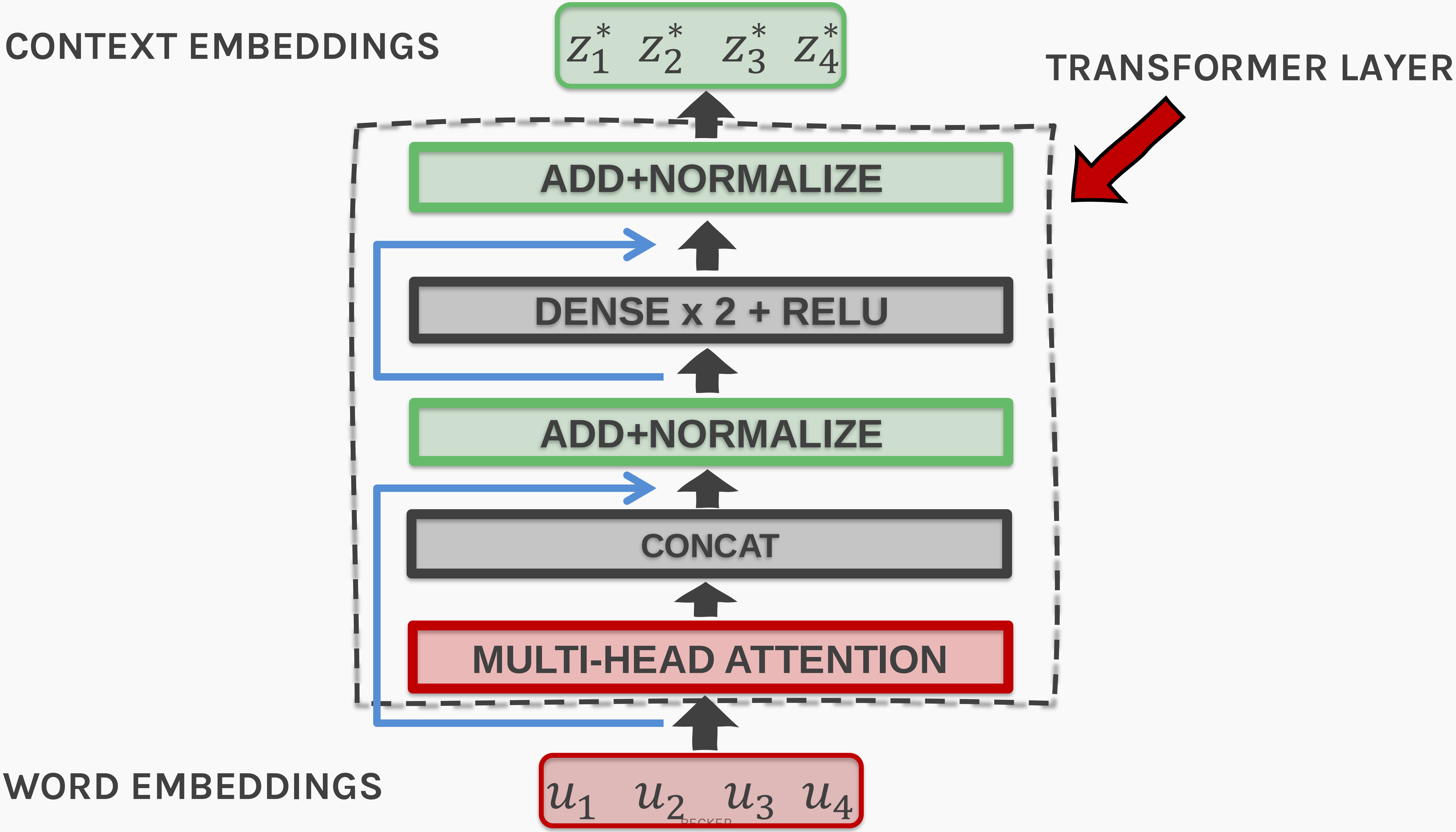

MULTI-HEAD
ATTENTION BLOCK
u_1 u_2 u_3 u_4
W_Q^i DENSE LAYER
W_K^i DENSE LAYER
W_V^i DENSE LAYER
$Q^i = W_Q^i U$
$K^i = W_K^i U$
$V^i = W_V^i U$
COSINE SIMILARITY + SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
$Z^i = \mathrm{softmax}\left(\frac{Q^i \cdot {K^i}^T}{\sqrt{d_k}}\right) V^i$
z_1 z_2 z_3 z_4
BECKER

CONTEXT EMBEDDINGS

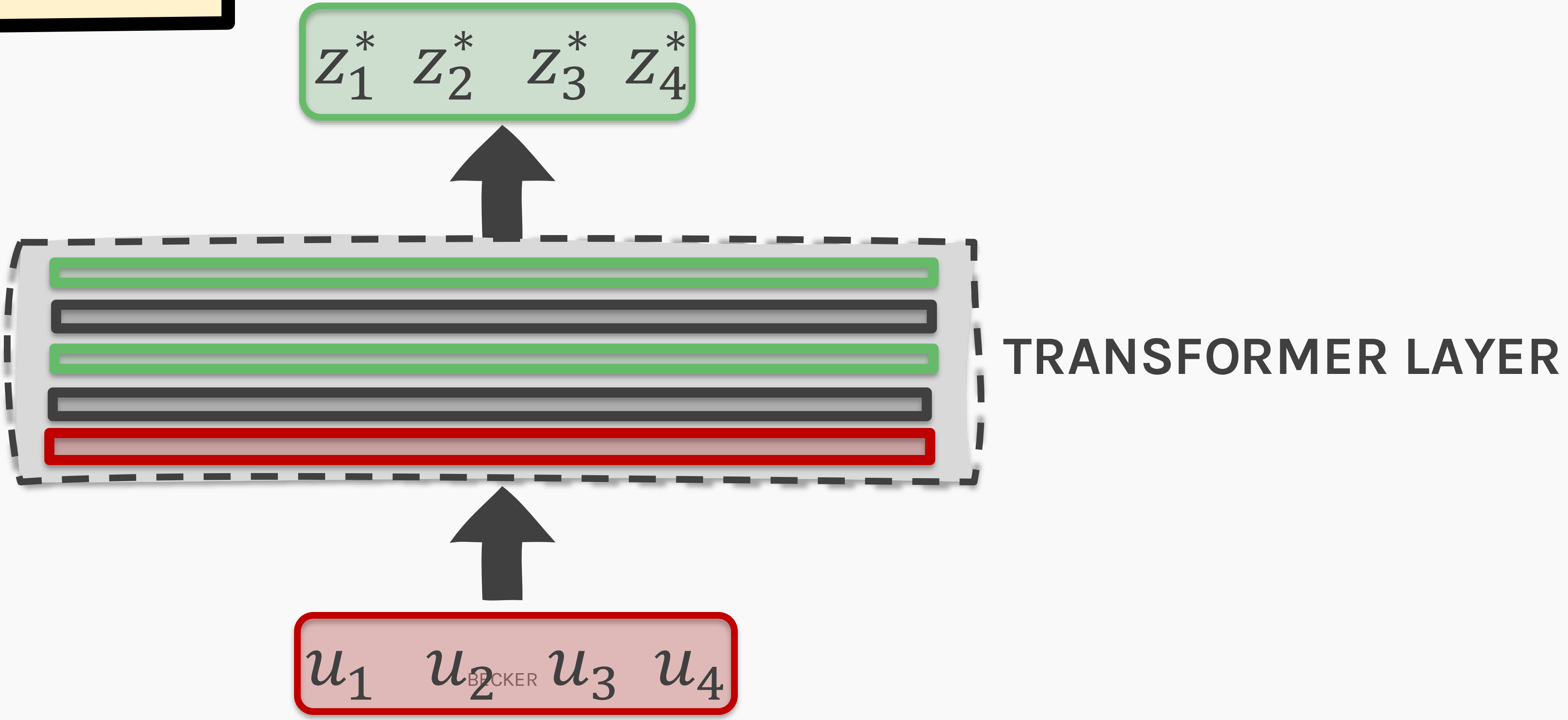


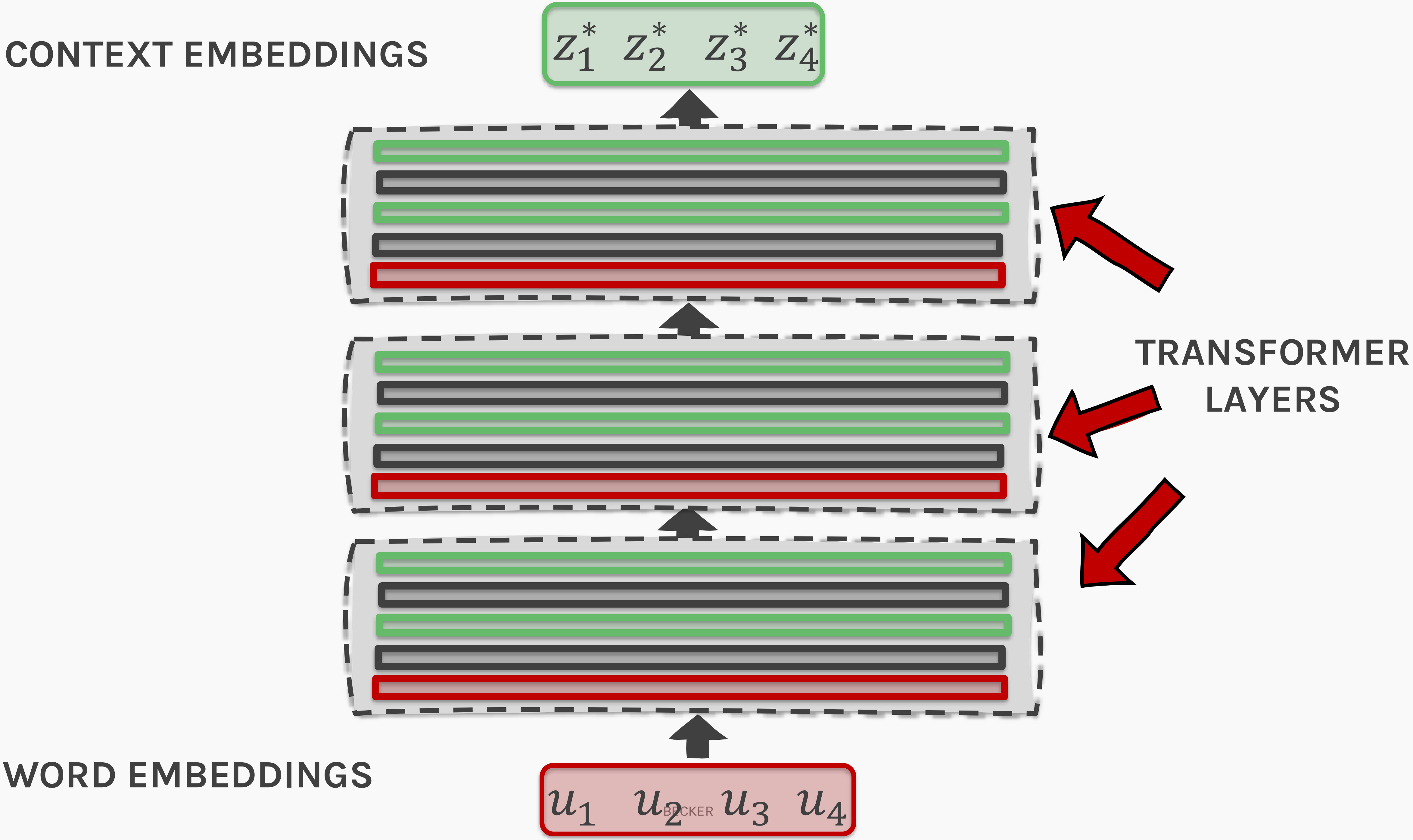
WORD EMBEDDINGS

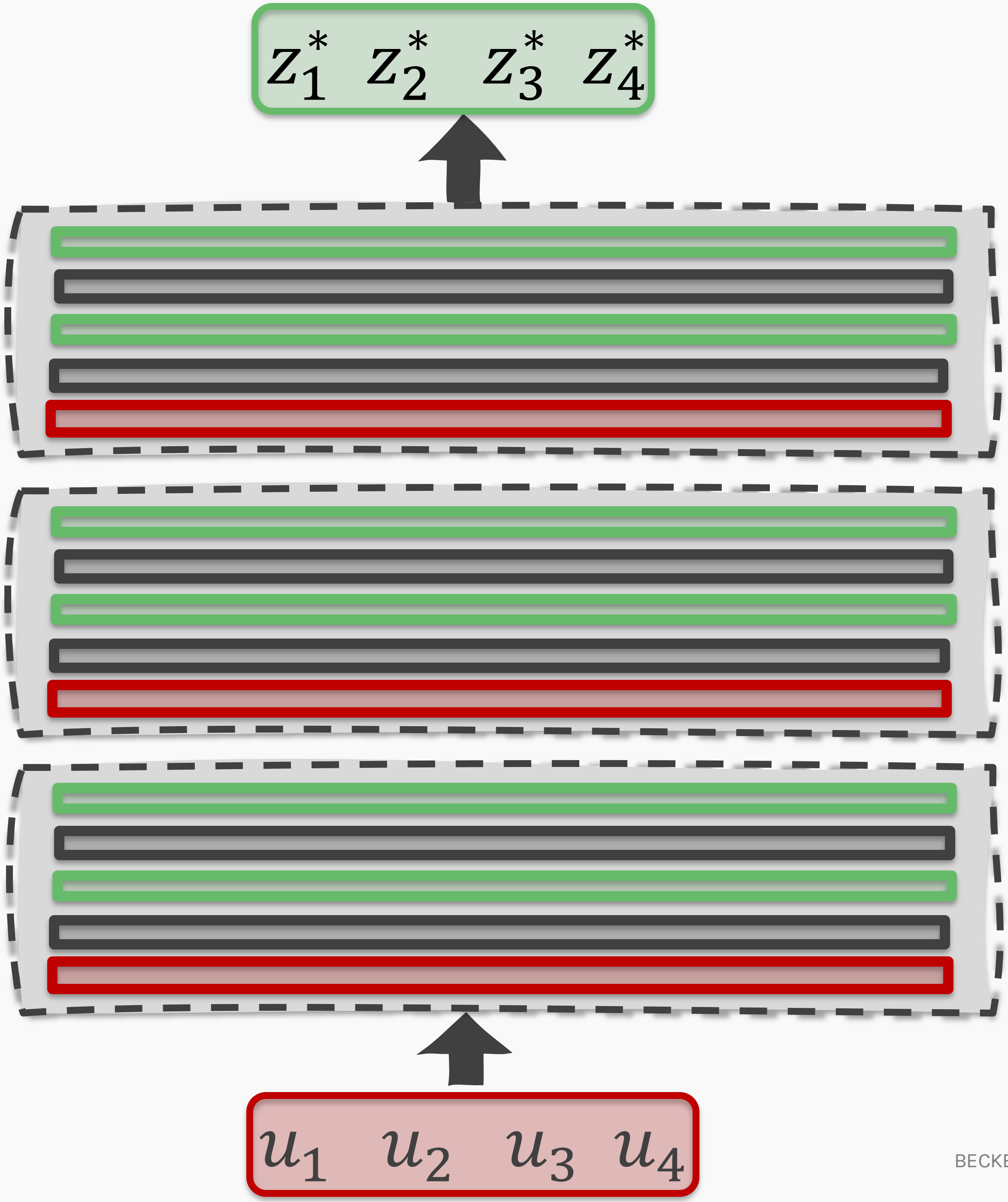

We could even add skip connections to avoid the issue of vanishing gradients
Why not add Normalization layers to help gradients flow better during back propagation? Something like batch norm
z_1^* z_2^* z_3^* z_4^*
CONCAT
MULTI-HEAD ATTENTION
u_1 u_2 u_3 u_4

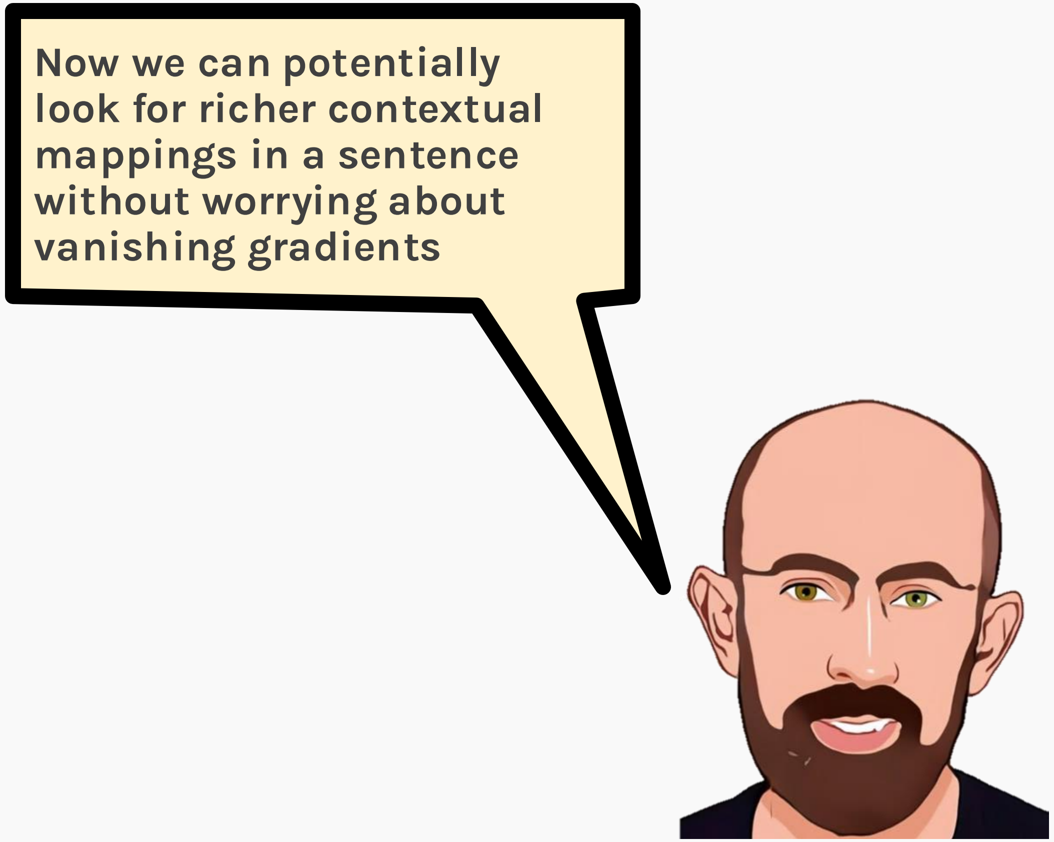

CONTEXT EMBEDDINGS
z_1^* z_2^* z_3^* z_4^*
TRANSFORMER LAYER
ADD+NORMALIZE
DENSE x 2 + RELU
ADD+NORMALIZE
CONCAT
MULTI-HEAD ATTENTION
WORD EMBEDDINGS
u_1 u_2 u_3 u_4

Why not stack multiple transformer layers?




CONTEXT EMBEDDINGS
$z_1^* \quad z_2^* \quad z_3^* \quad z_4^*$
TRANSFORMER LAYERS
WORD EMBEDDINGS
$u_1 \quad u_2 \quad u_3 \quad u_4$
BECKER


z_1^* z_2^* z_3^* z_4^*
u_1 u_2 u_3 u_4


Now we can potentially look for richer contextual mappings in a sentence without worrying about vanishing gradients

Attention

MULTI-HEAD ATTENTION ISSUES?

- ~~No weights trained in the process~~
- ~~Attention leads to limited contextual mapping~~
- **There is no positional information encoded**

Amelia spoke to Pavlos about attention

Attention

MULTI-HEAD ATTENTION ISSUES?

- ~~No weights trained in the process~~
- ~~Attention leads to limited contextual mapping~~
- **There is no positional information encoded**

Pavlos spoke to Amelia about attention

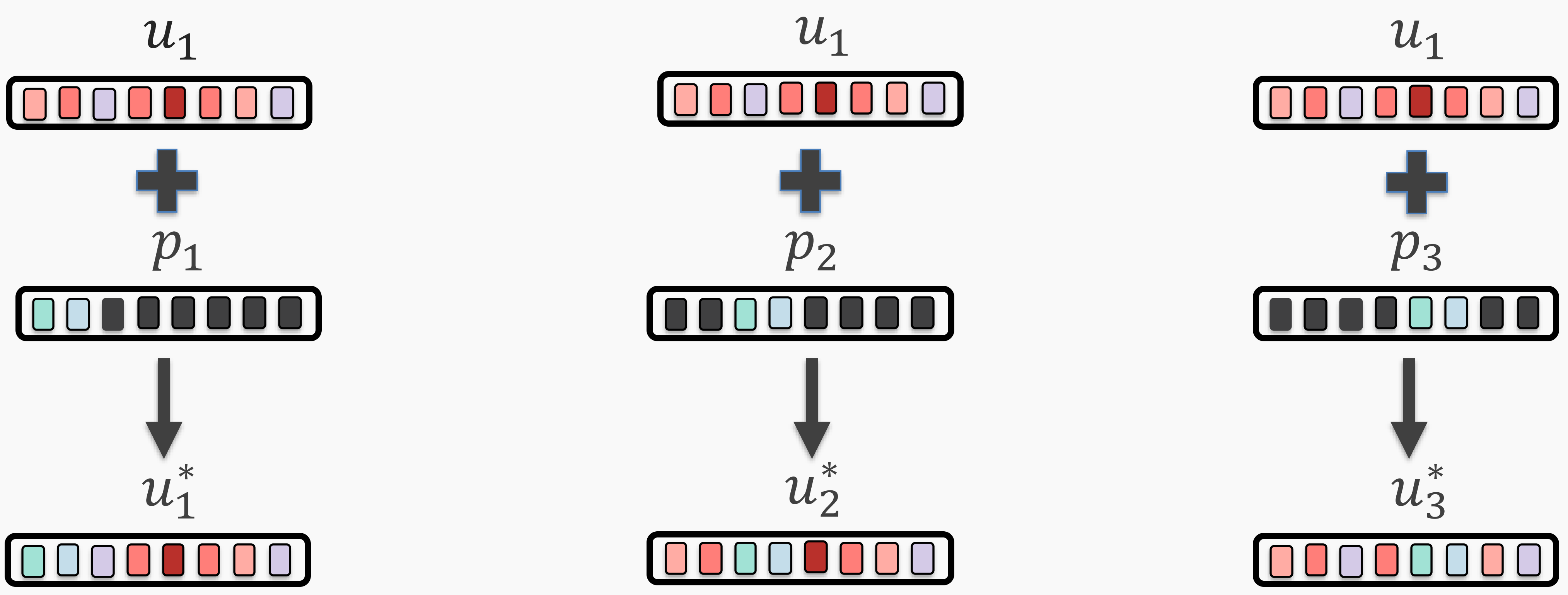
What we want

LANGUAGE MODEL WISHLIST

- Position and order of words are the essential parts of any language
- Recurrent Neural Networks (RNNs) inherently take the order of word into account
- Multi-head attention blocks do not take such an order by design, so there's the need to incorporate the order of the words separately

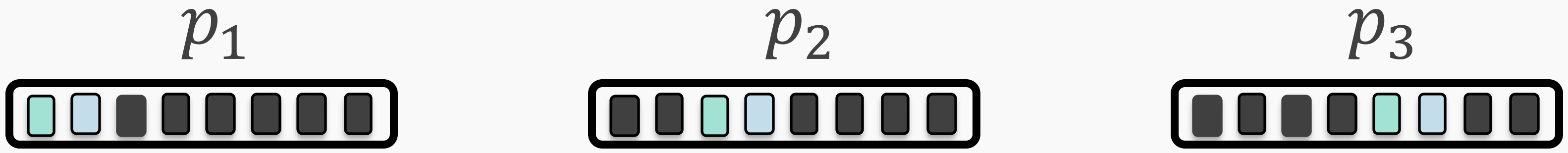
💡**IDEA #255:** Positional Encoding

Positional Encoding



In the above case, $u_1^* \neq u_2^* \neq u_3^*$

Positional Encoding



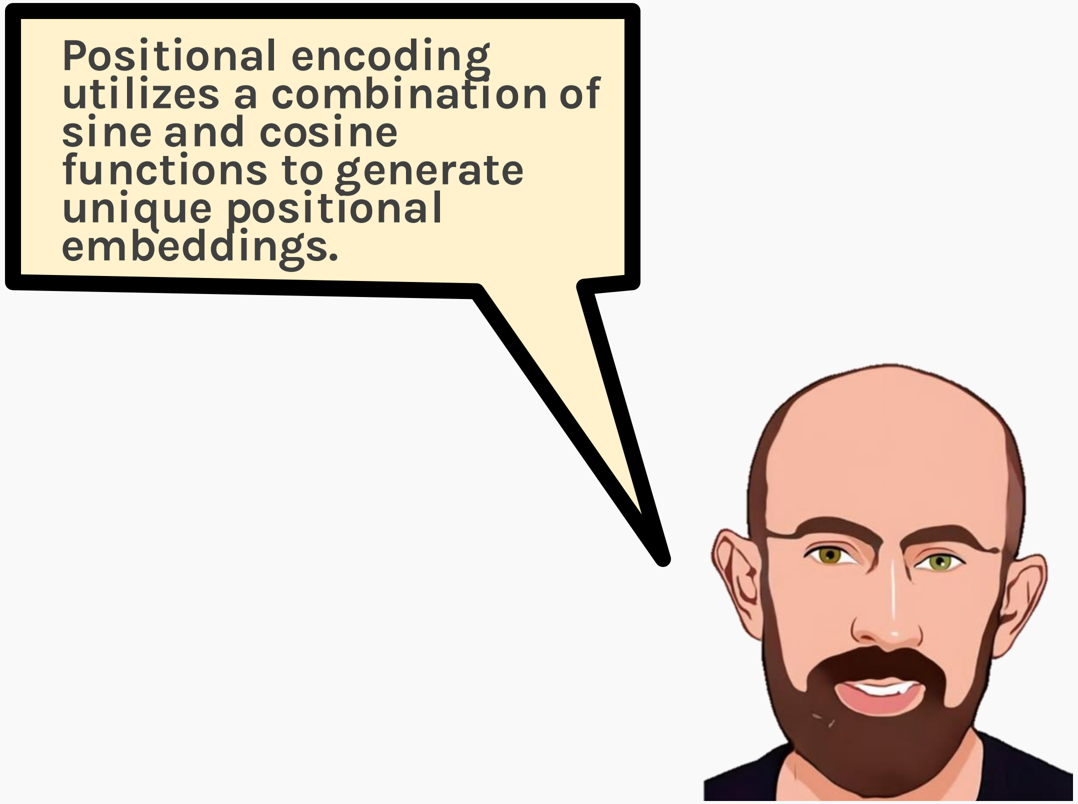
But how do we find these positional embedding vectors?

Don’t worry Ignacio, we have several options to choose from…

Positional Encoding

POSITIONAL EMBEDDING WISHLIST

- We should have a unique embedding for each timestep
- Relative embedding must remain consistent across sentences of different lengths
- It should generalize to longer sentences
- It should be bounded & deterministic

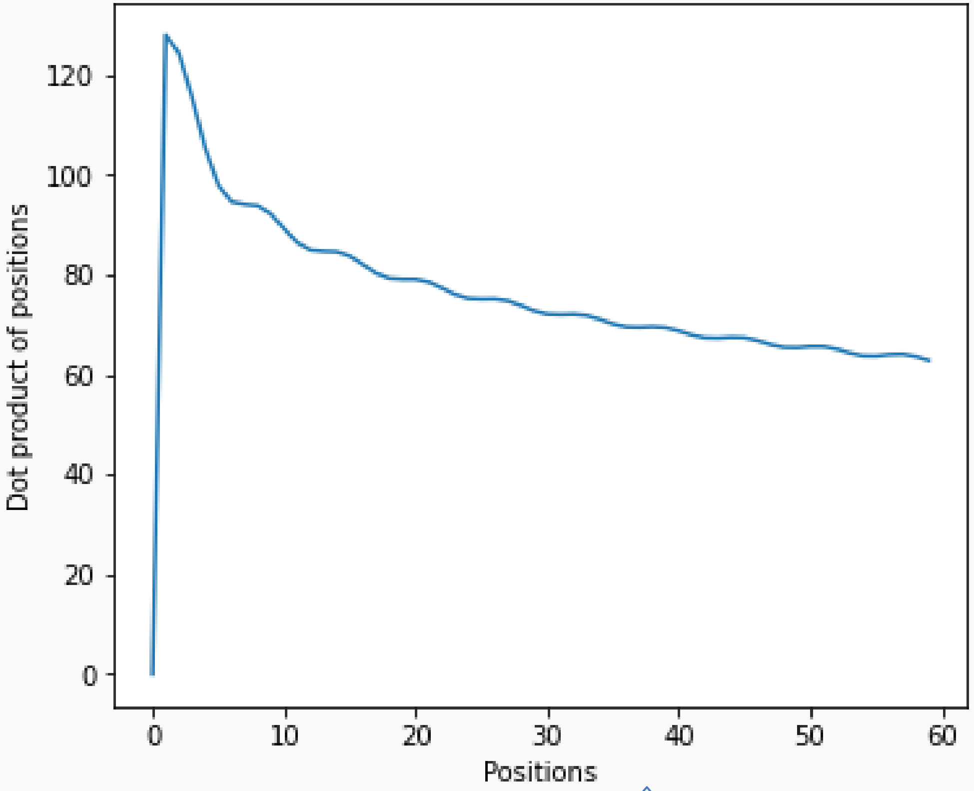


Positional Encoding

TECHNICAL DETAILS

Positional encoding is a method used to help models like transformers understand the order and relationship of words in a sentence.

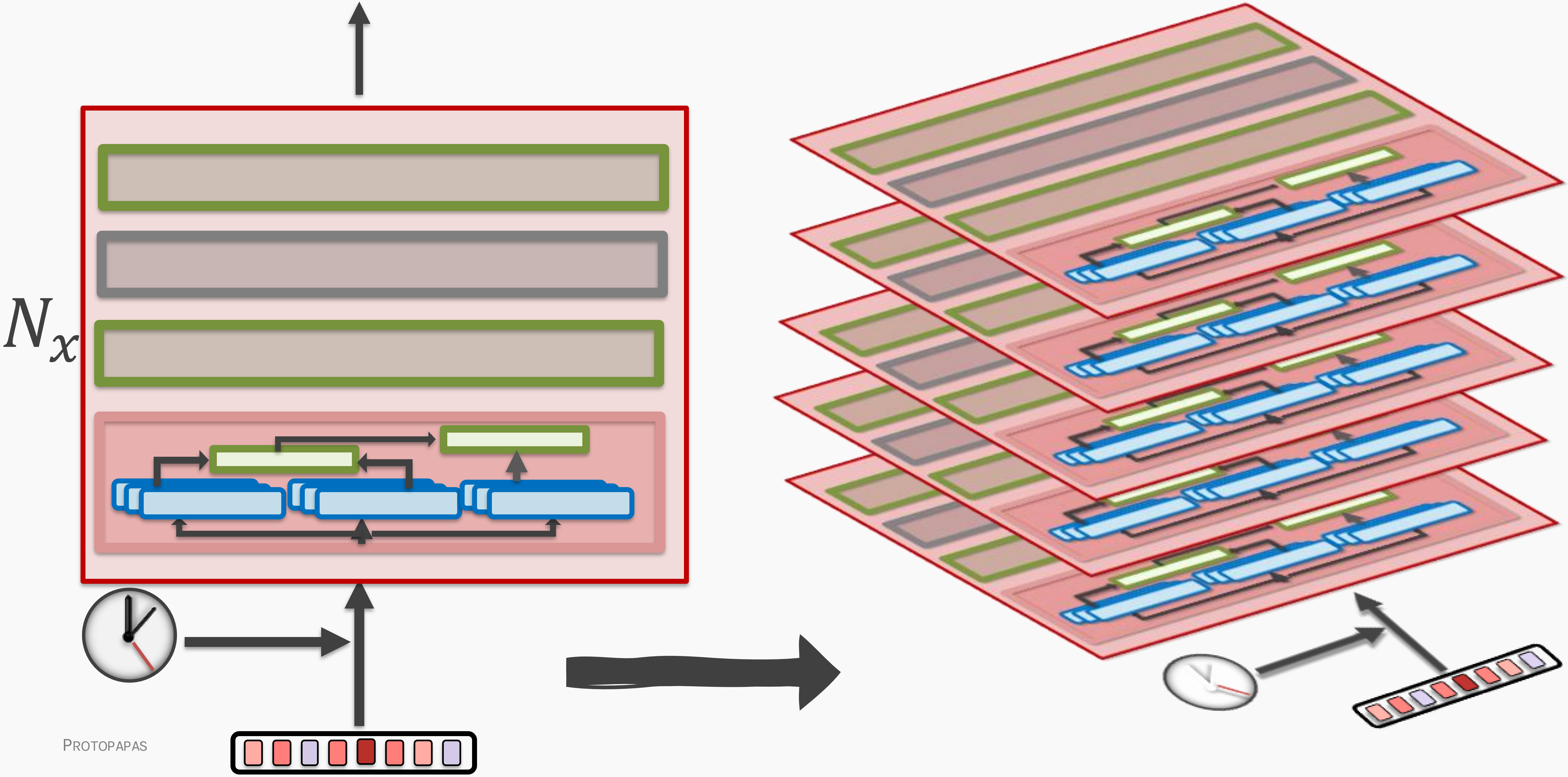
- Each position in a sentence is assigned a vector that encodes its position.
- The encoding uses a mix of sine and cosine functions to ensure each position has a unique, yet repeatable, pattern.



values of sine and cosine change with the position in the sequence

Bringing it all together

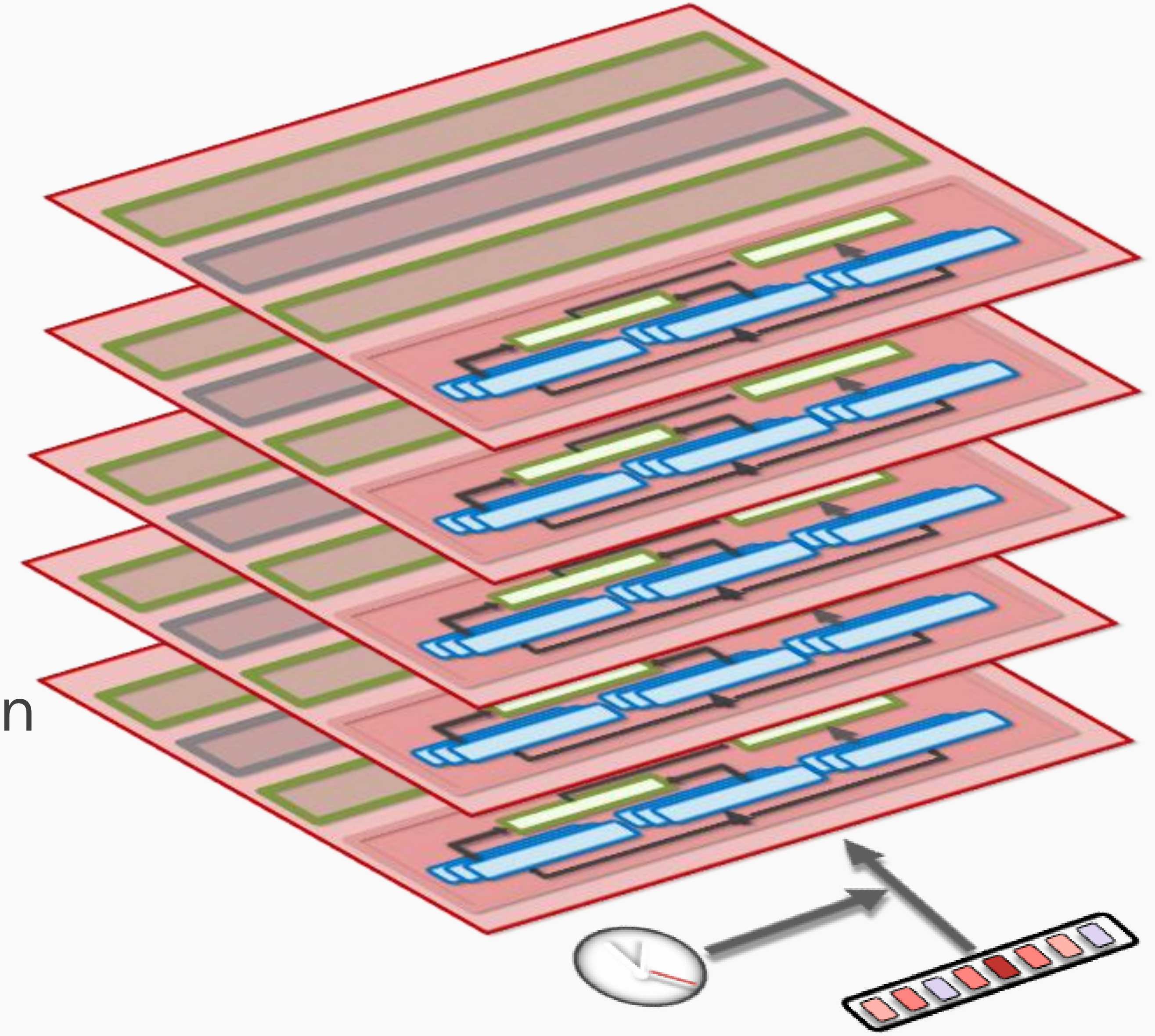
Transformer as a 3-D model



Transformers - Summary

Language Model Wishlist?

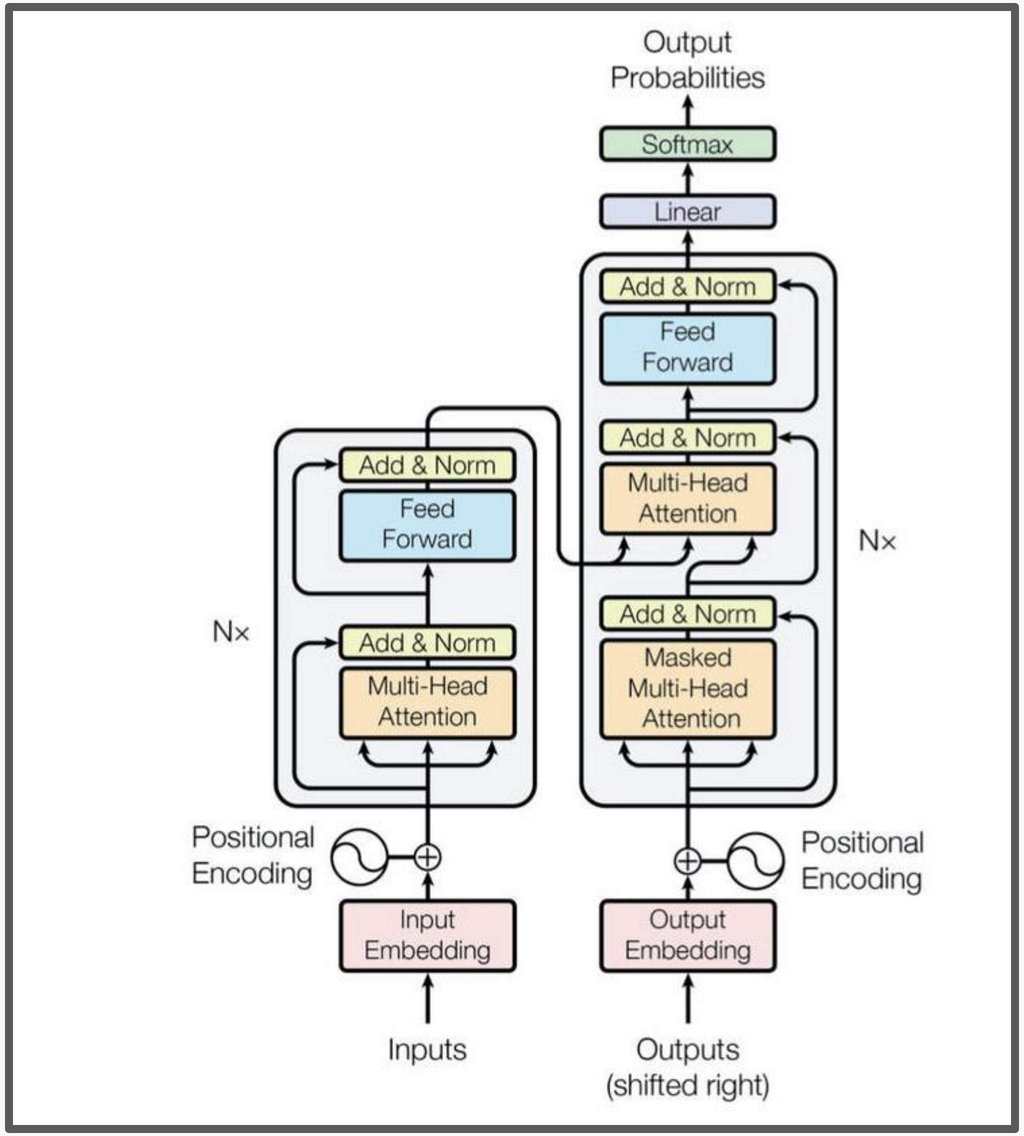
- ☑ We want to have strong contextual relations between words - **DONE**
- ☑ We want words to have sequential information - **DONE**
- ☑ We need an architecture that can be trained in parallel (non-Markovian property) - **DONE**



Transformers - Summary

Let’s look at the diagram of the transformer architecture from the original paper

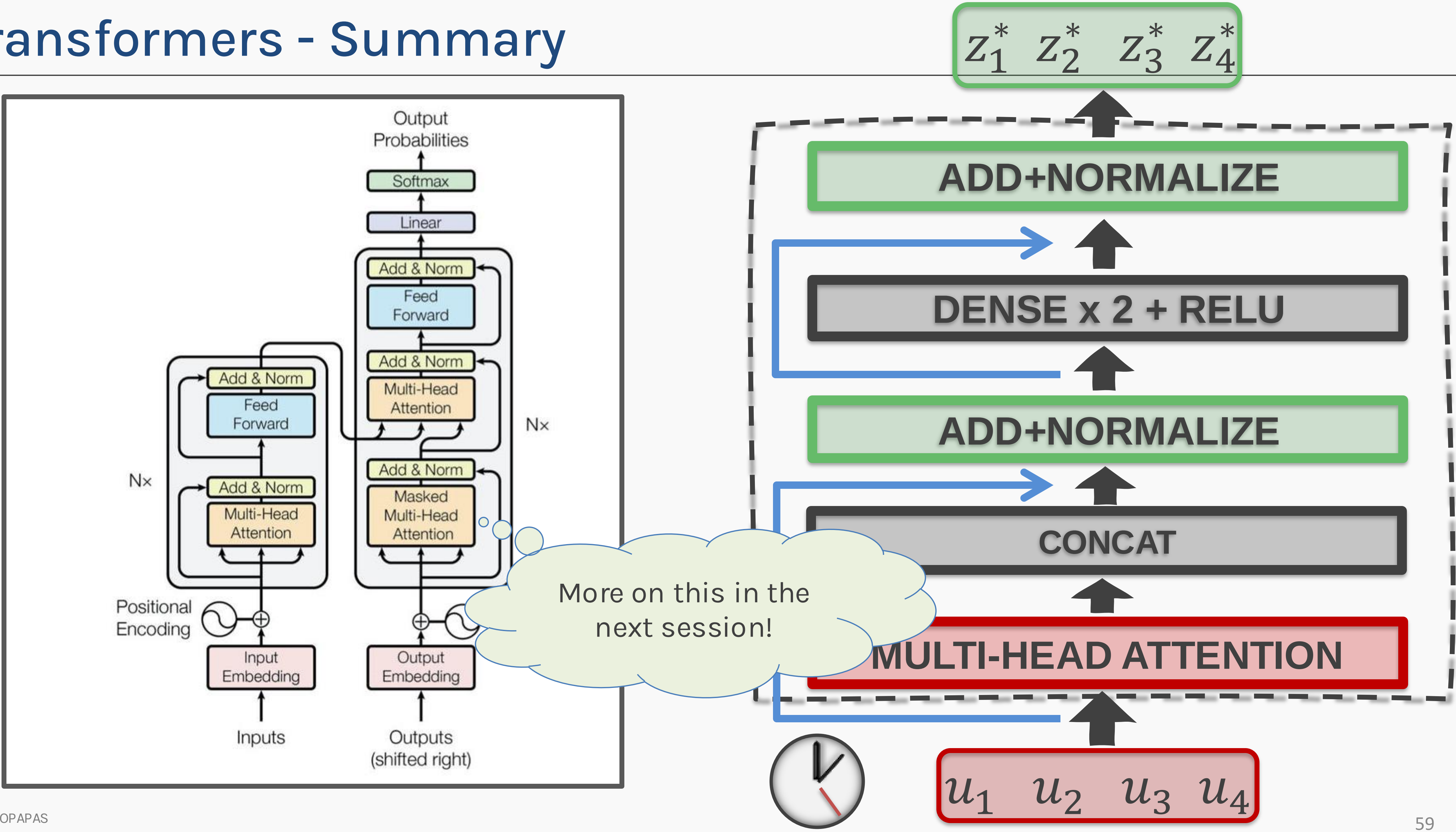
Transformers - Summary



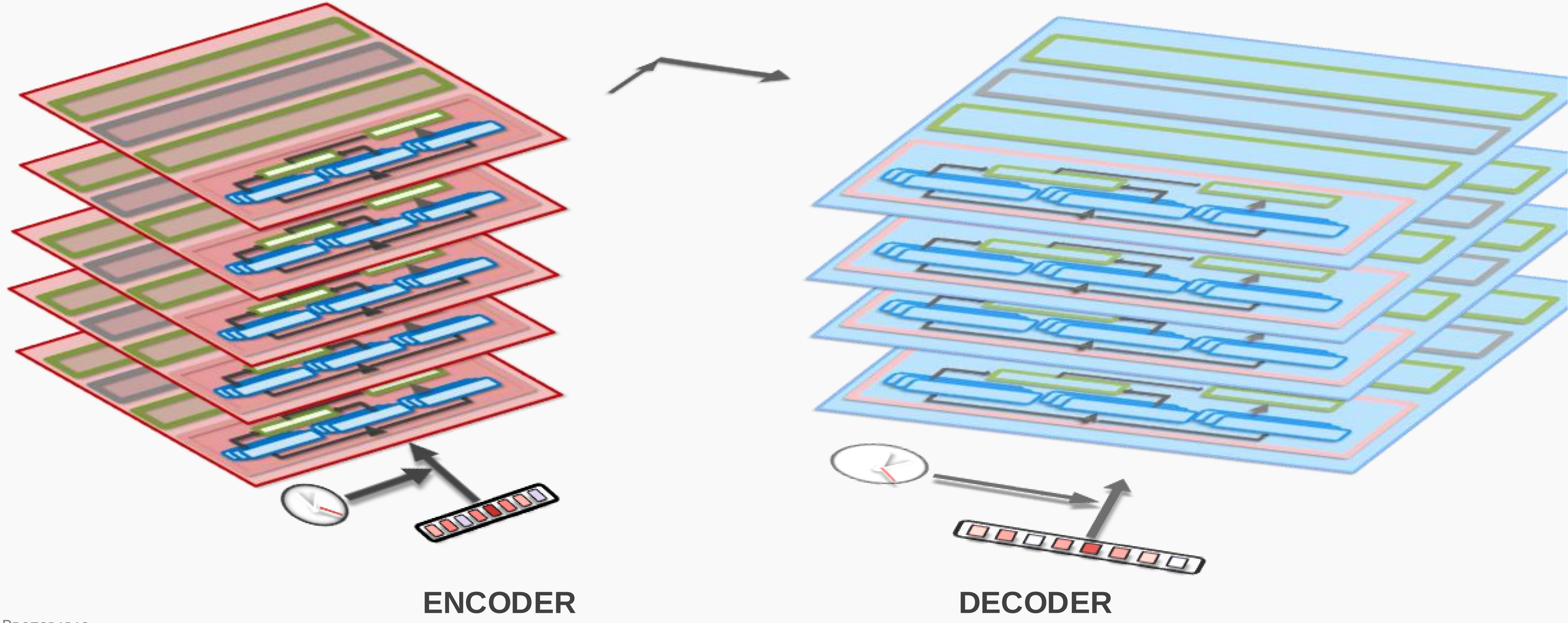
Let’s look at the diagram of the transformer architecture from the original paper

And now, compare it to what we have

Transformers - Summary

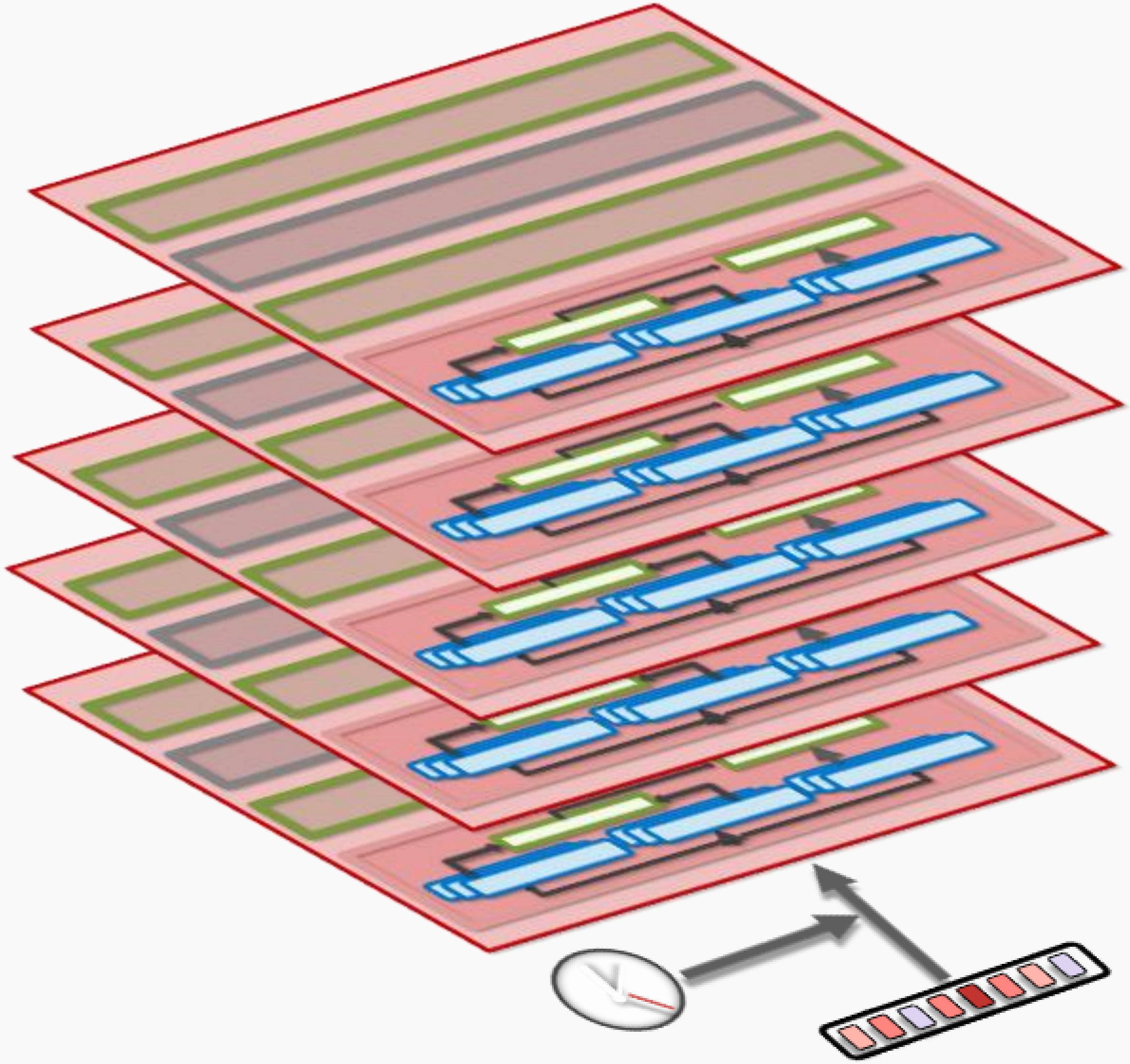


Transformers - Summary



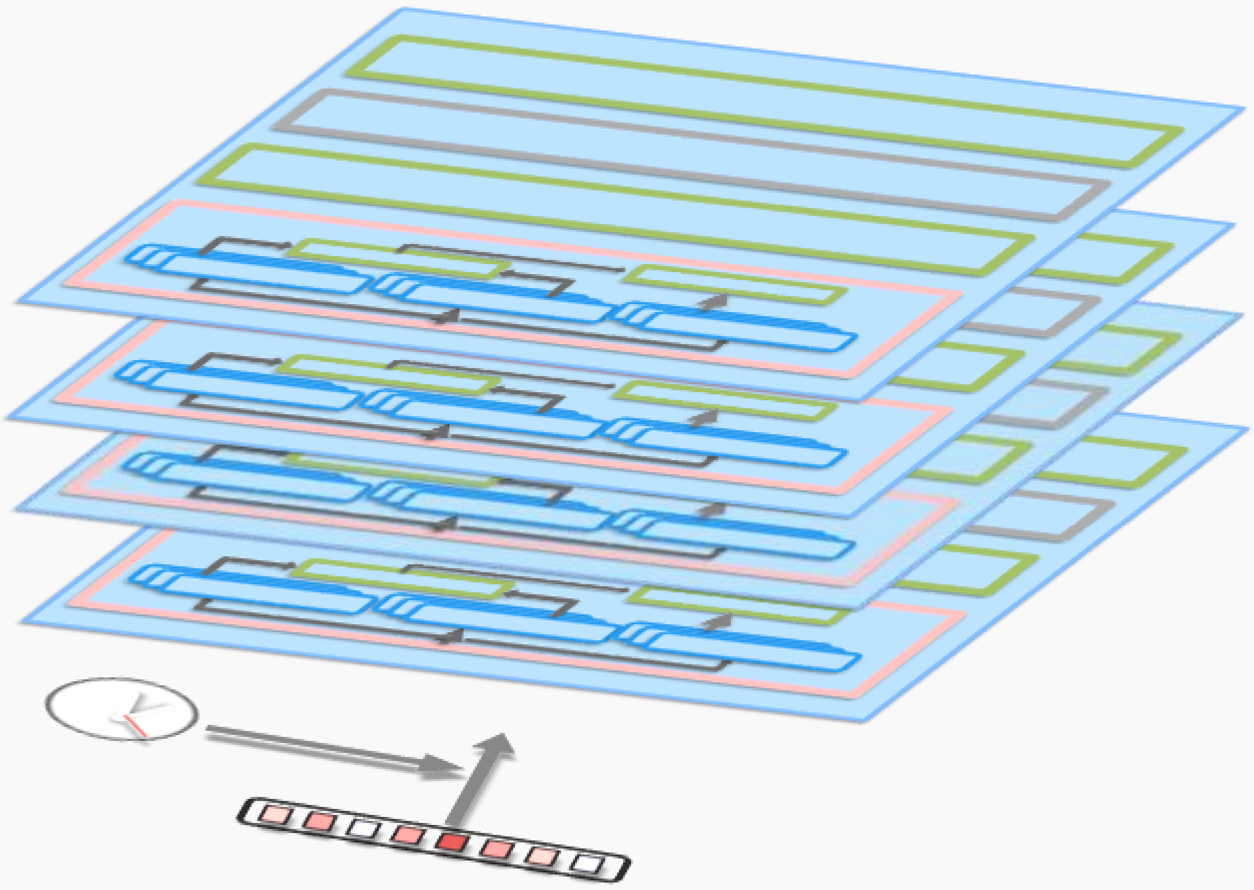
From Transformers to BERT

- Instead of an **Encoder-Decoder** architecture for machine translation, what if we just use the encoder for Language Model.
- This led to the new architecture called Bidirectional Encoder Representations from Transformers, or more commonly known as BERT.



From Transformers to GPT

- Now if we use just the decoder for Language Model
- We get a Causal Language Model (A word is predicted using words from its left context)
- Also known as autoregressive model
- Generative Pre-trained Transformer, or more commonly known as GPT



DECODER

Thank you!