Transformers

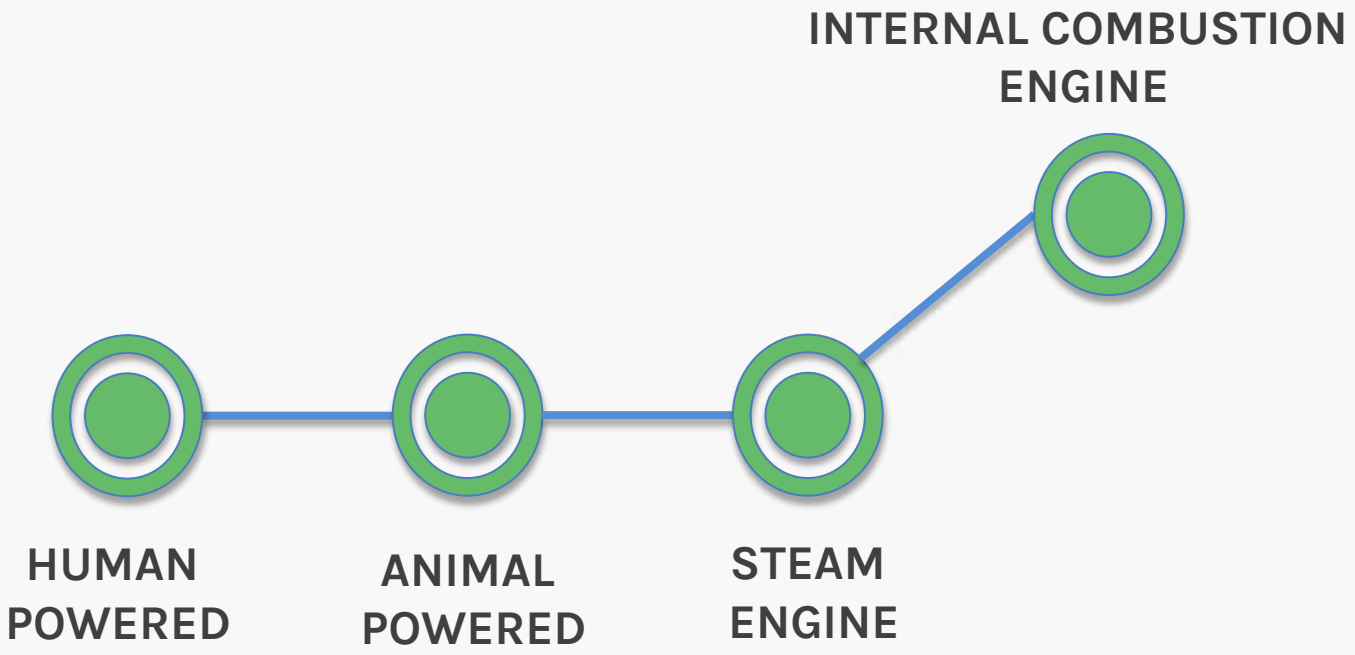
Pavlos Protopapas

Photo: Amanda Wasserman

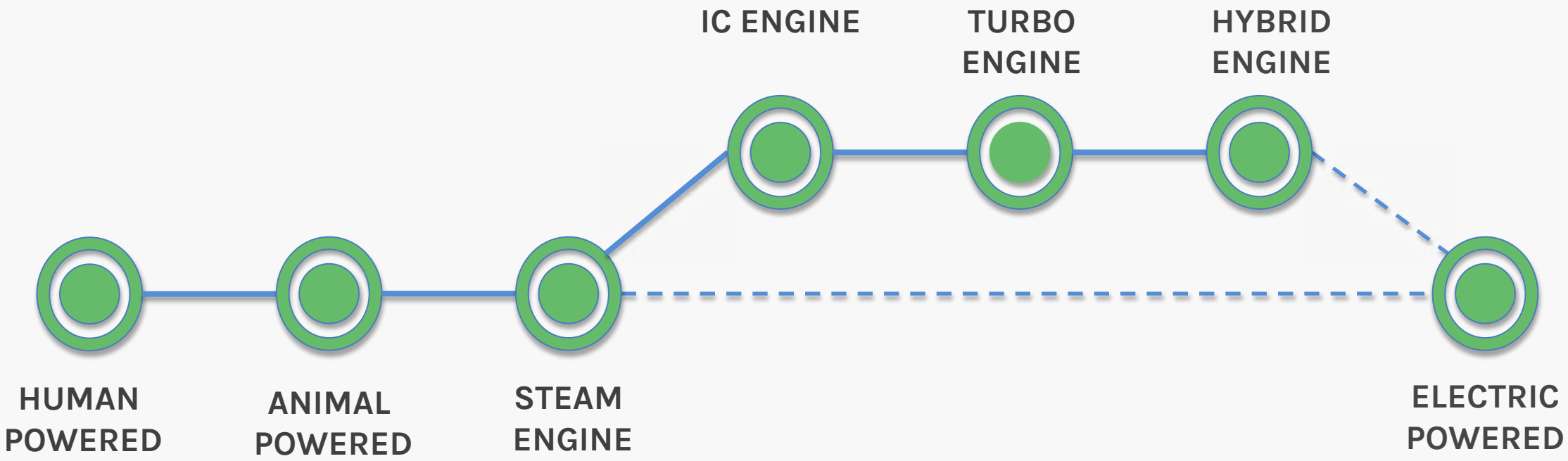
Outline

- Motivation for Attention
  - Limitations of RNNs
- Attention Basics
  - Issues with spatial attention models
  - 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

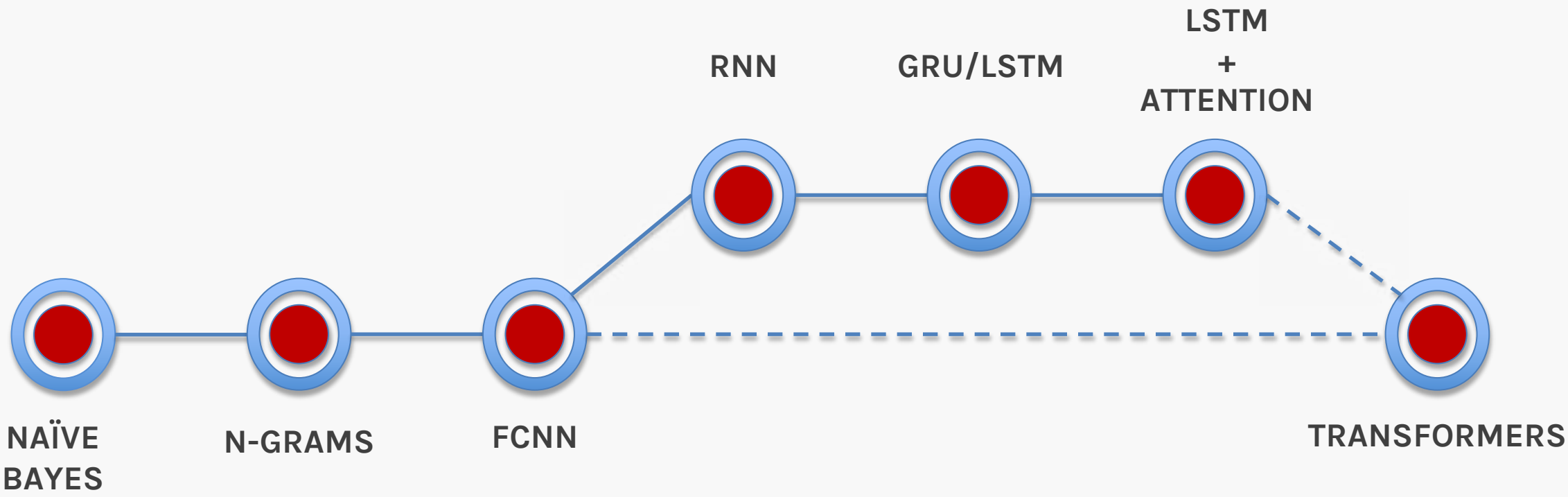
Electricity is all you need



Electricity is all you need



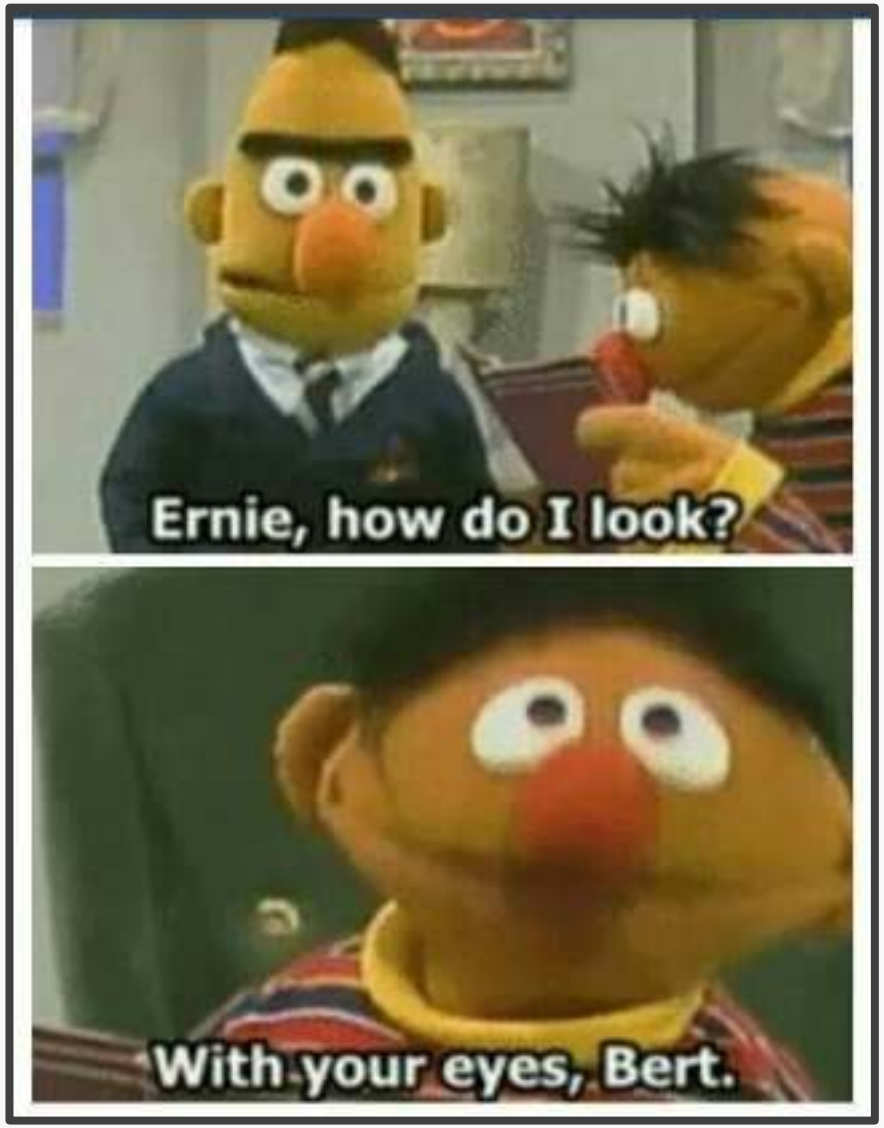
Language modeling



What we want

Language Model Wishlist?

- 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)



We've already seen an approach of relative importance

Attention

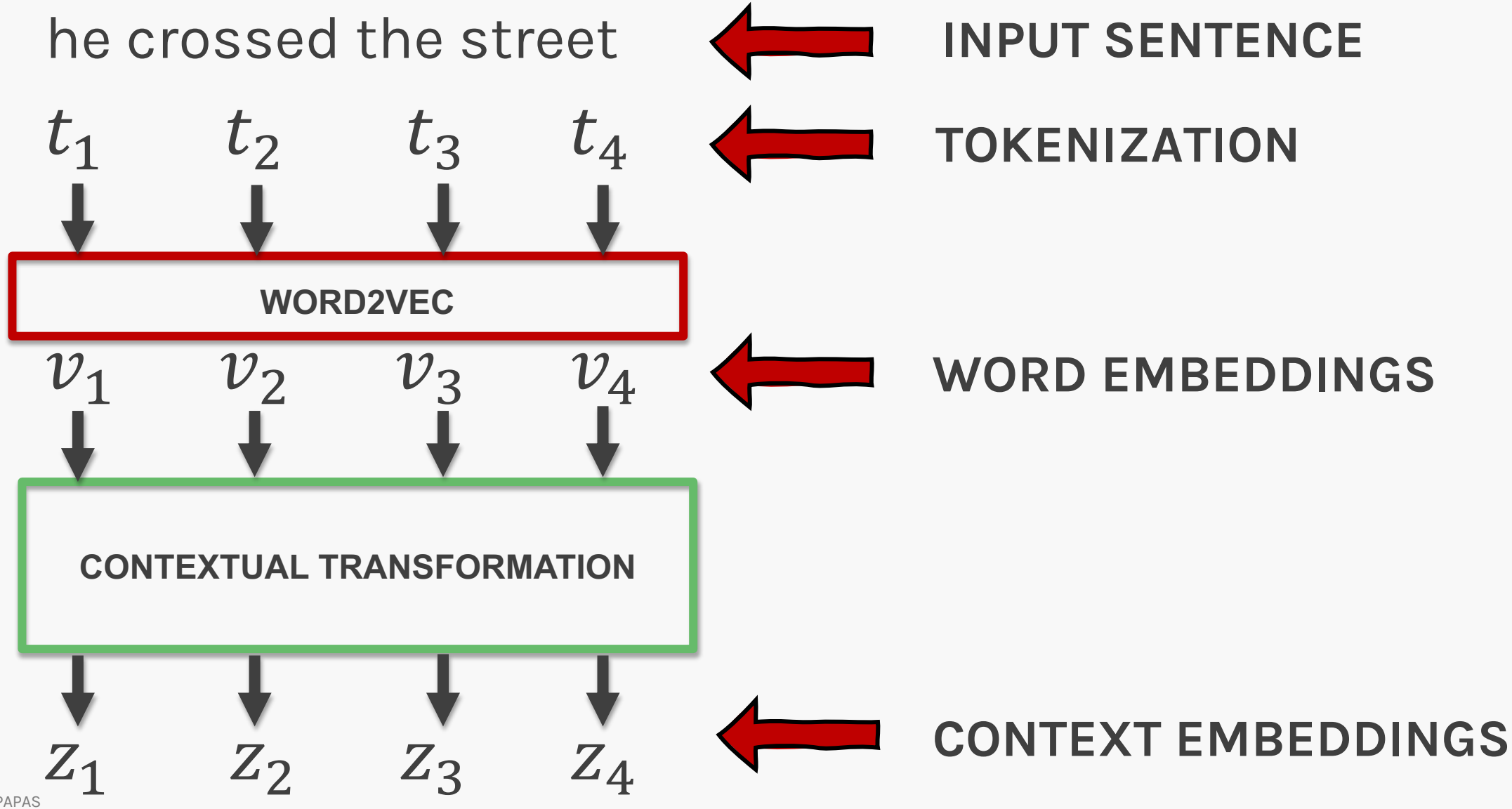
Attention: Example sentence

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

Ignacio was hit by a bus because he crossed the street

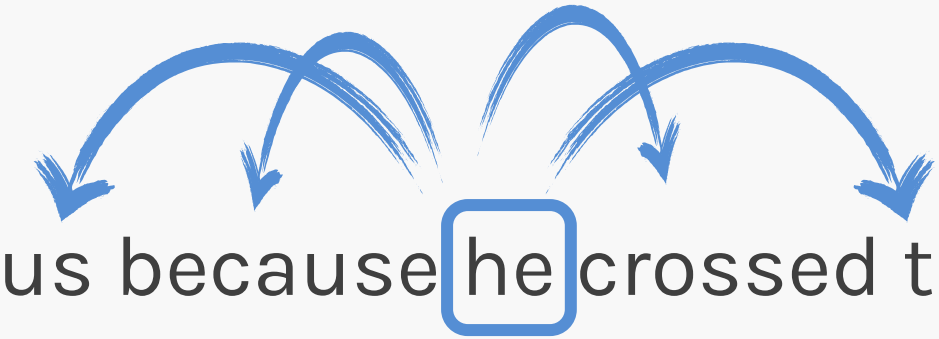
No teaching staff were harmed during the production of this lecture. This is a work of fiction. Names, characters, places and incidents either are products of the professor’s imagination or are used fictitiously. Any resemblance to actual events or locales or persons, living or dead, is entirely coincidental.

Attention – Where to add weights?



Attention: Example sentence

💡 **IDEA #1:** Positional relationship



Attention: Example sentence

💡 **IDEA #1:** Positional relationship

Ignacio was hit by a bus because he crossed the street

True context

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

Attention – The basics

- We can still use the idea of transforming the word embeddings to get more context

$$x = [x_1, x_2, x_3, \dots, x_n]$$

- However, the transformation matrix *A* must place some importance to the relative importance of words

$$z = A\, x^T$$

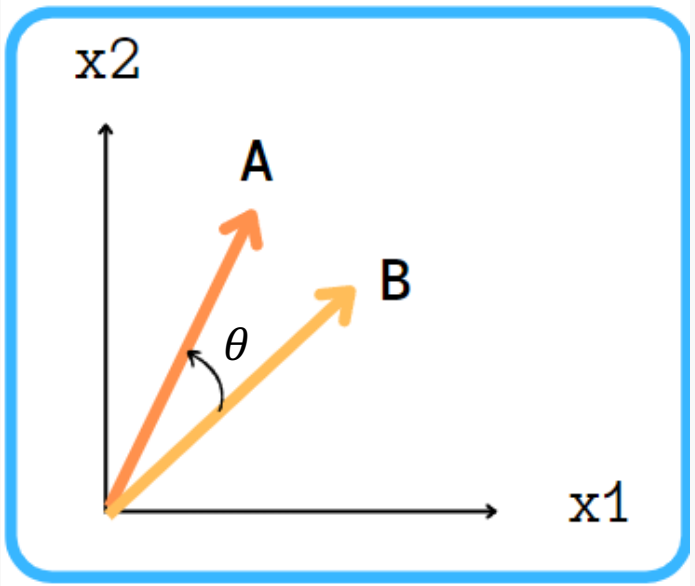
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & a_{ij} & \dots & a_{in} \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

a_{ij} must account for relative importance between word i & j

Attention – Text Data

But how can we do it?

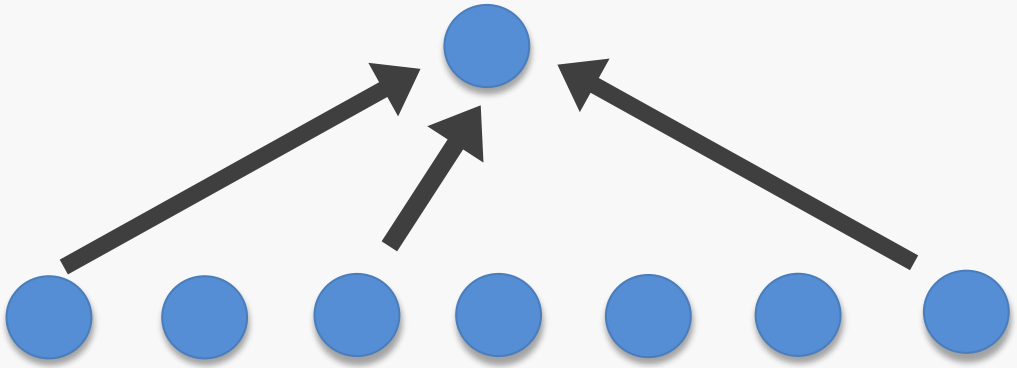
Yes! Let's use cosine similarity between words

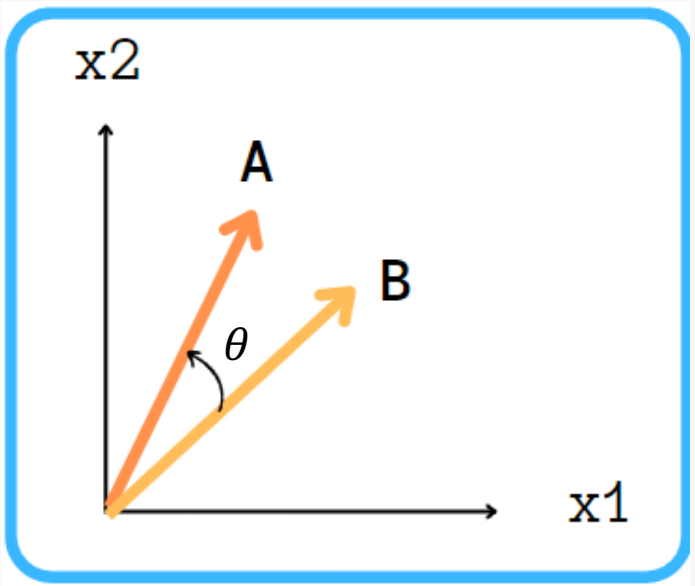


$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Recap: Cosine Similarity

For context, we could just take the cosine similarity of the target word with respect to every other word in the sentence





$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

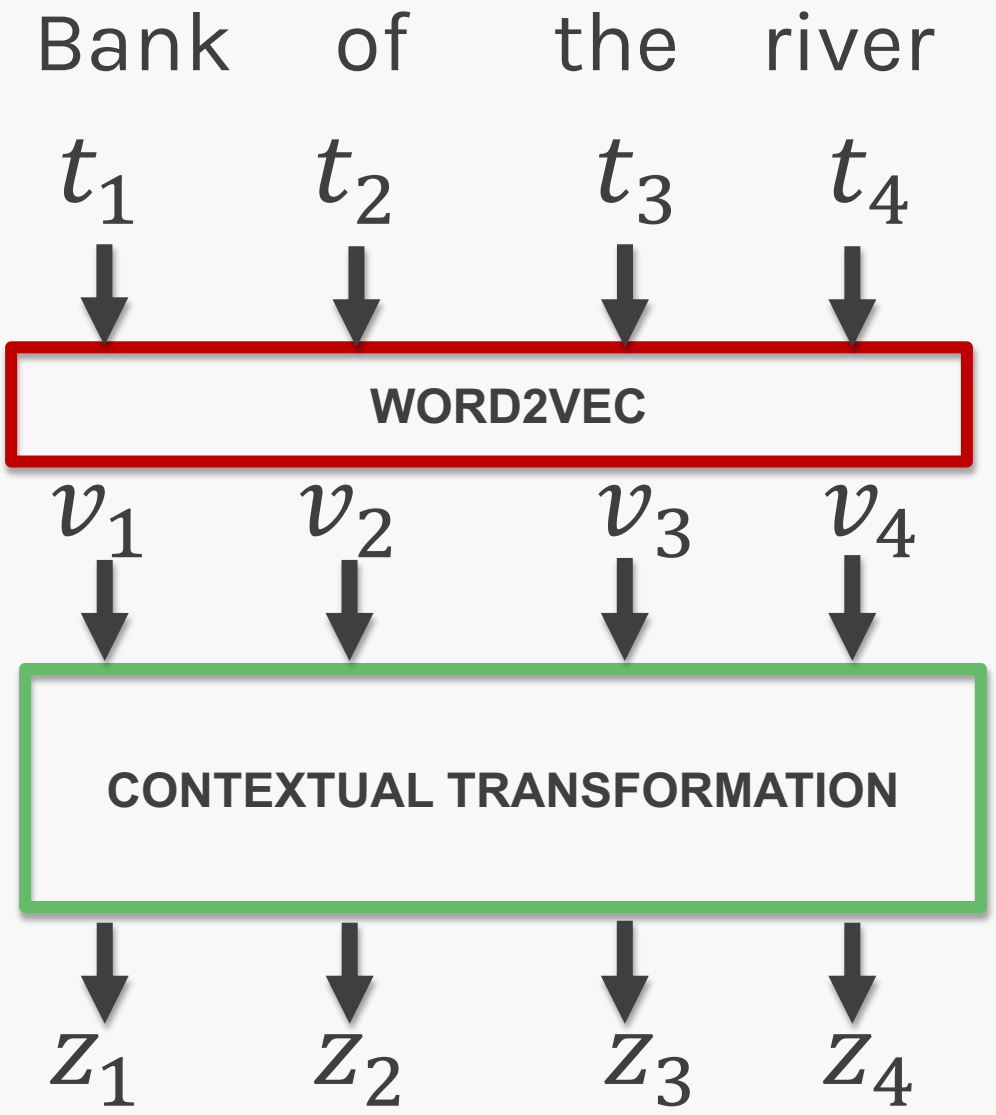
Where A_i & B_i are components of vector A & B respectively

How do we do it?

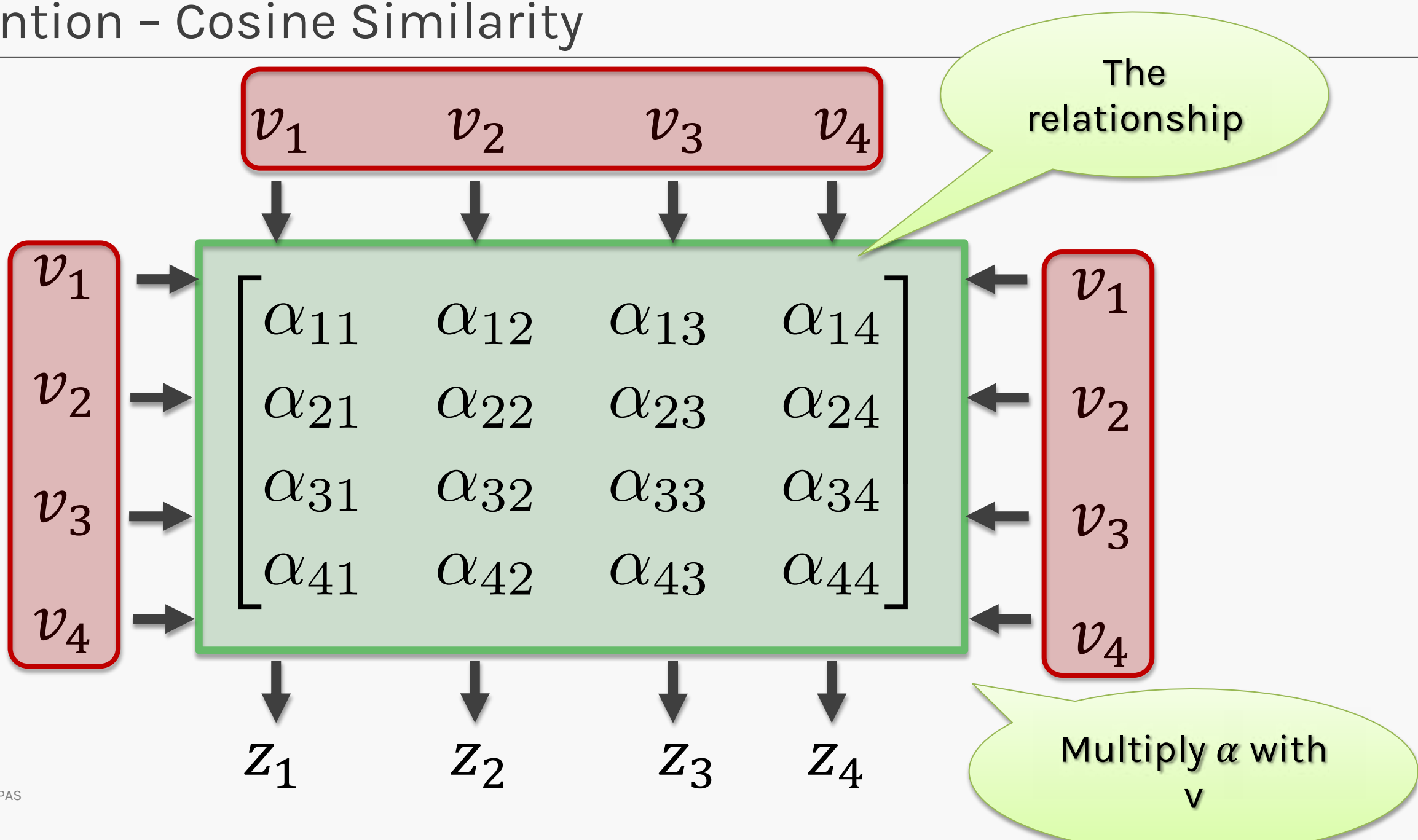
Attention: Example sentence #2

Ignacio was standing next to the bank of the river

Attention: Example sentence #2



Attention – Cosine Similarity



Attention: Finding α

$$v_1 \circ v_1 = a_{11}$$
$$v_1 \circ v_2 = a_{12}$$
$$v_1 \circ v_3 = a_{13}$$
$$v_1 \circ v_4 = a_{14}$$

softmax(.)

WEIGHTS NORMALIZATION

$$\alpha_{11}$$
$$\alpha_{12}$$
$$\alpha_{13}$$
$$\alpha_{14}$$

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

Cosine similarity

Attention

$$z_1 = v_1\alpha_{11} + v_2\alpha_{12} + v_3\alpha_{13} + v_4\alpha_{14}$$

New contextual
word embedding

Attention: Simple with cosine similarity

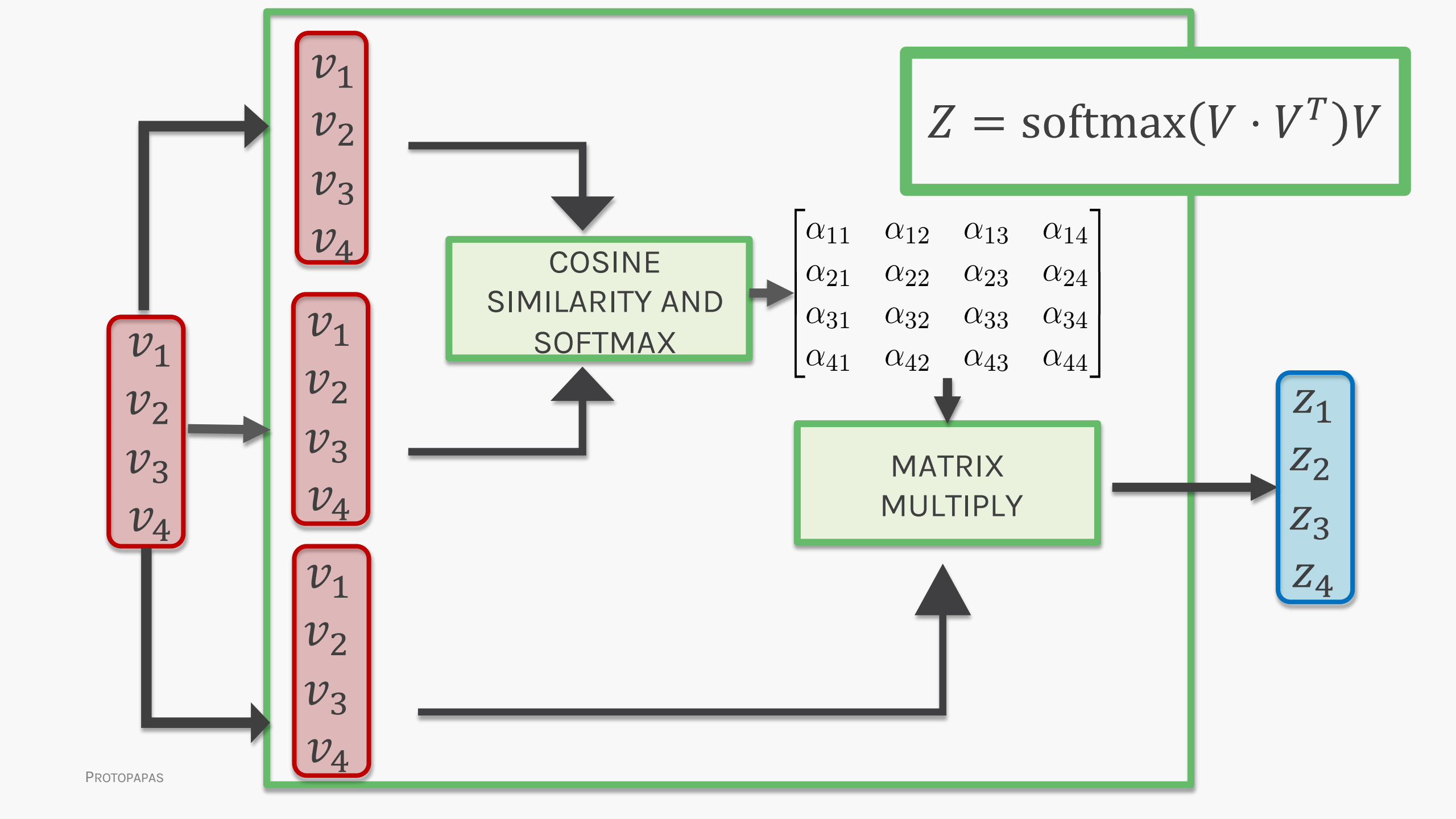
Similarly…

$$z_1 = v_1\alpha_{11} + v_2\alpha_{12} + v_3\alpha_{13} + v_4\alpha_{14}$$

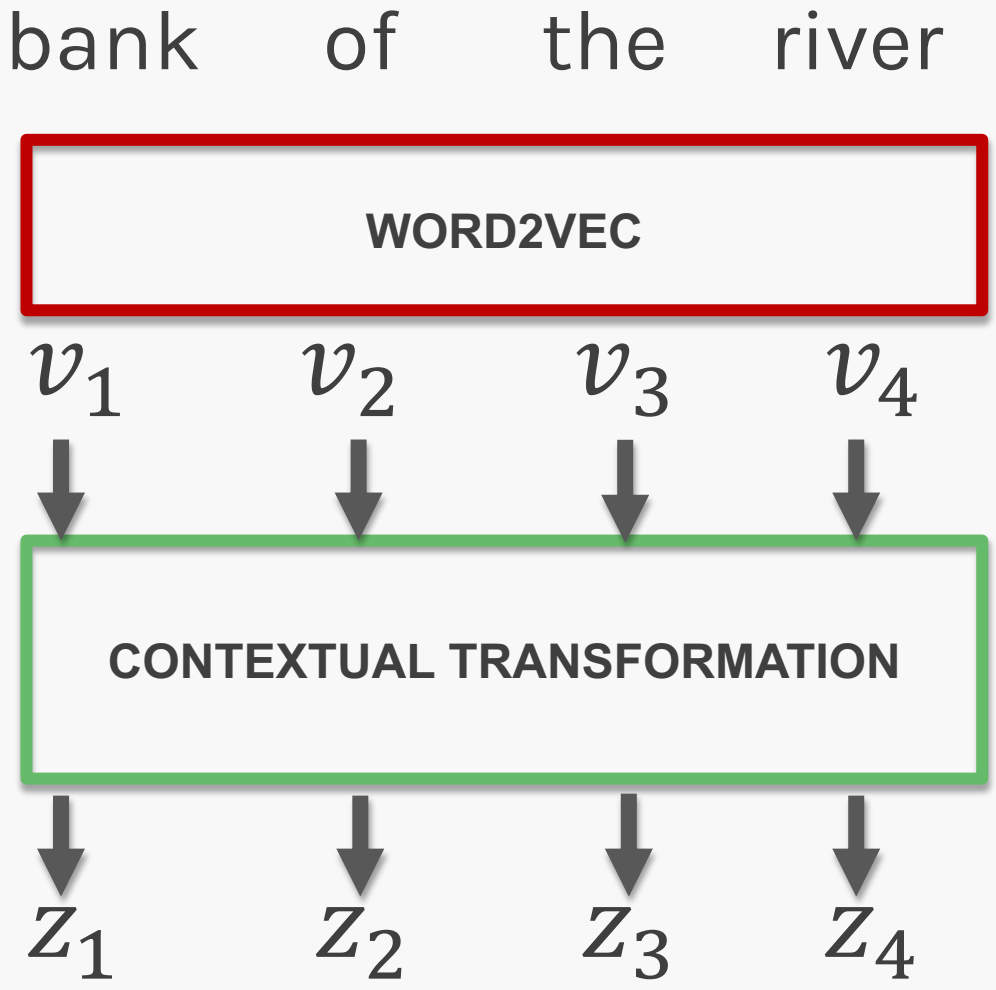
$$z_2 = v_1\alpha_{21} + v_2\alpha_{22} + v_3\alpha_{23} + v_4\alpha_{24}$$

$$z_3 = v_1\alpha_{31} + v_2\alpha_{32} + v_3\alpha_{33} + v_4\alpha_{34}$$

$$z_4 = v_1\alpha_{41} + v_2\alpha_{42} + v_3\alpha_{43} + v_4\alpha_{44}$$


$Z = \text{softmax}(V \cdot V^T)V$
v_1 v_2 v_3 v_4
v_1 v_2 v_3 v_4
v_1 v_2 v_3 v_4
v_1 v_2 v_3 v_4
COSINE SIMILARITY AND SOFTMAX
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4

Attention: Example sentence #2

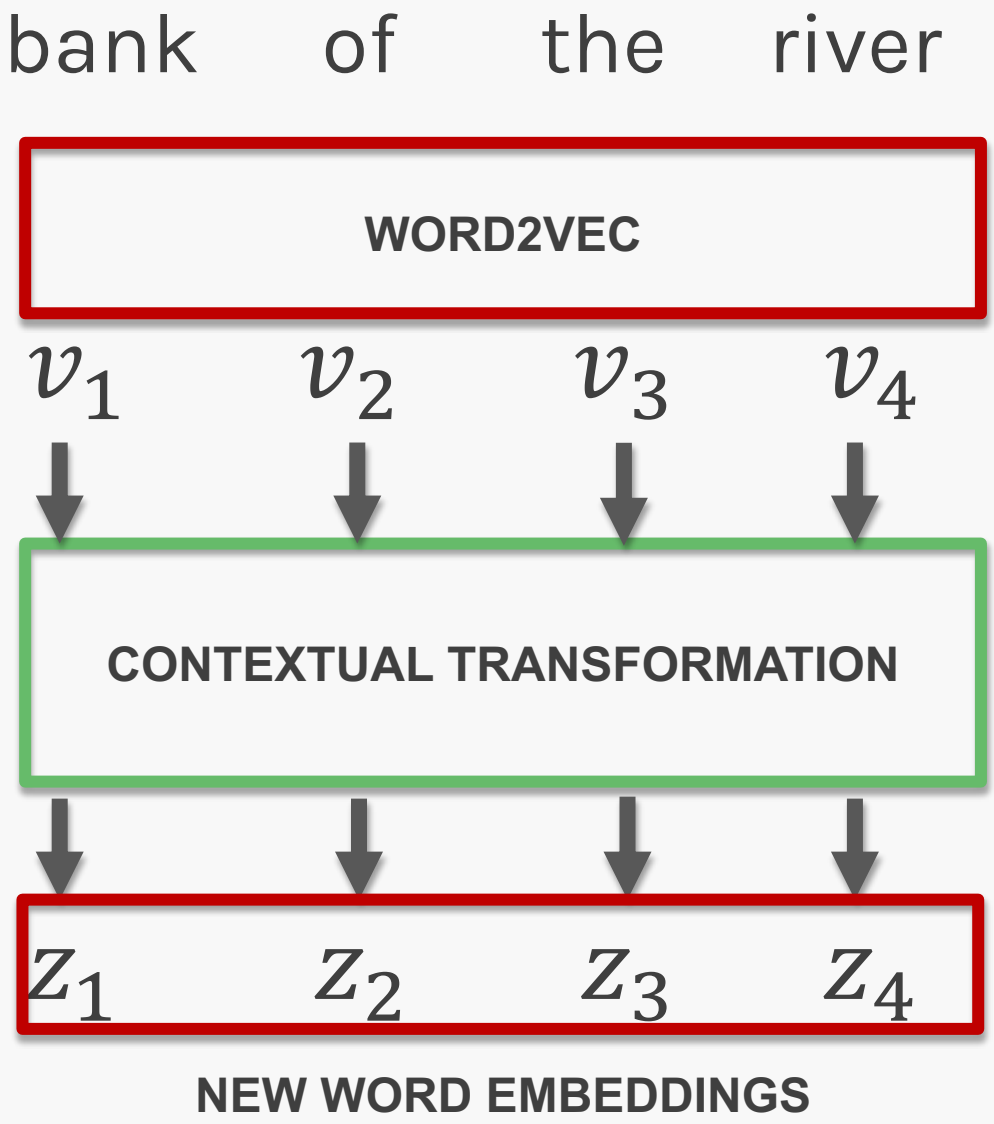


Attention

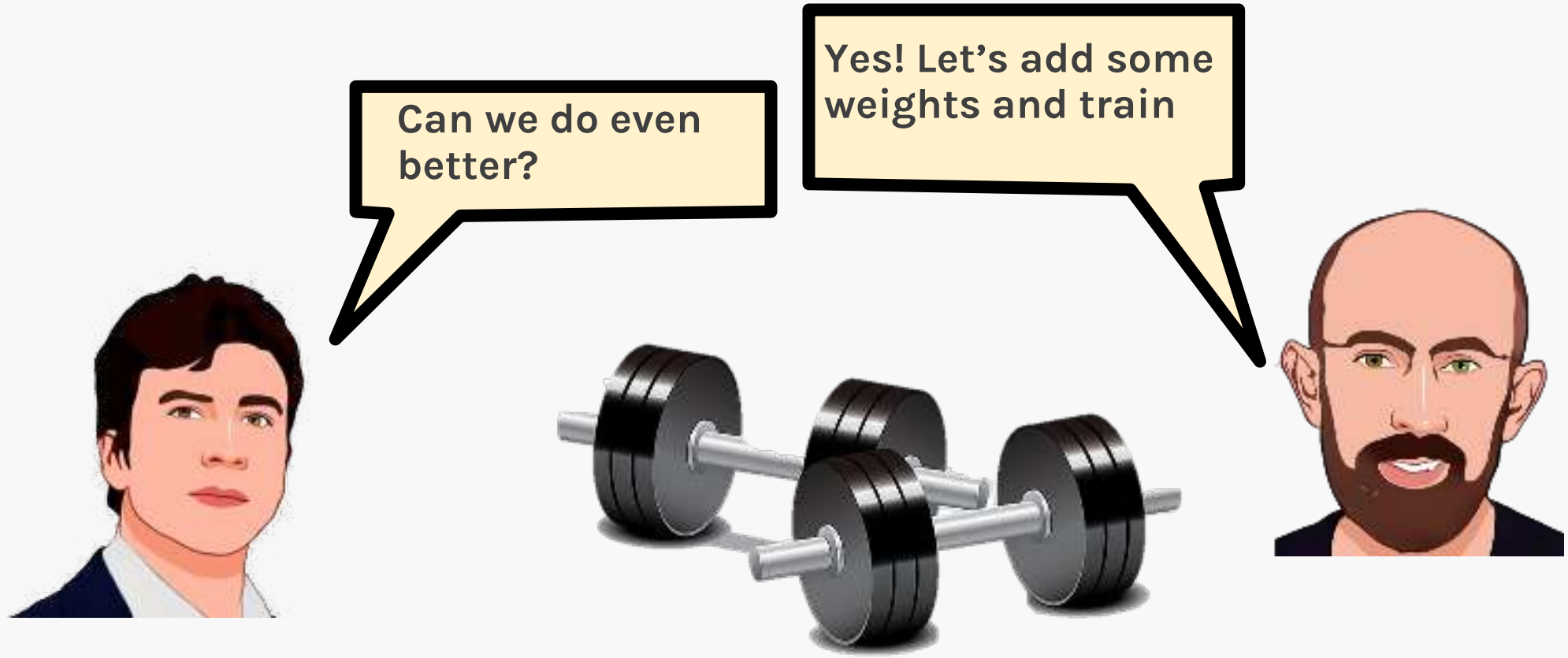
ATTENTION ISSUES?

This process gives us new embeddings with some context. However, we still have the following issues:

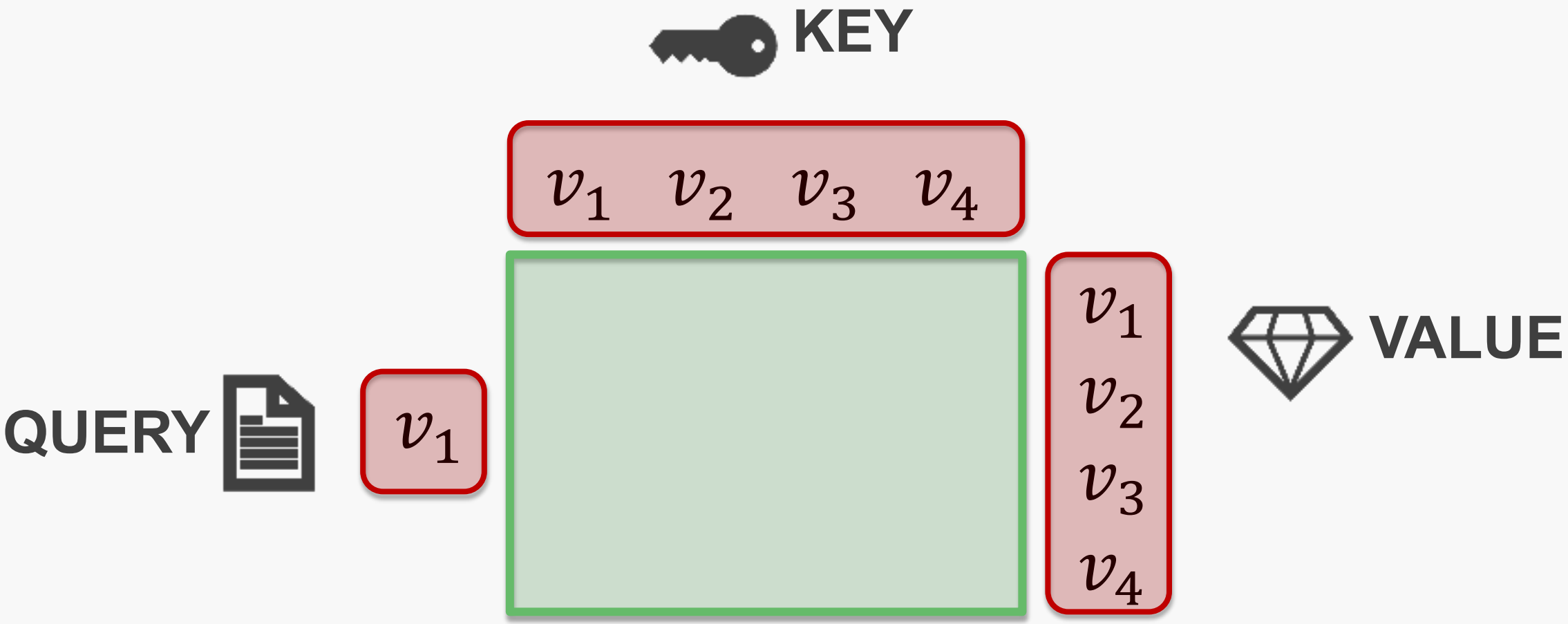
- No weights are trained in the process
- Attention as defined to be cosine similarity leads to fixed contextual mapping: two words will always have the same cosine similarity irrespective of the context
- There is no positional information encoded



Attention – Text Data

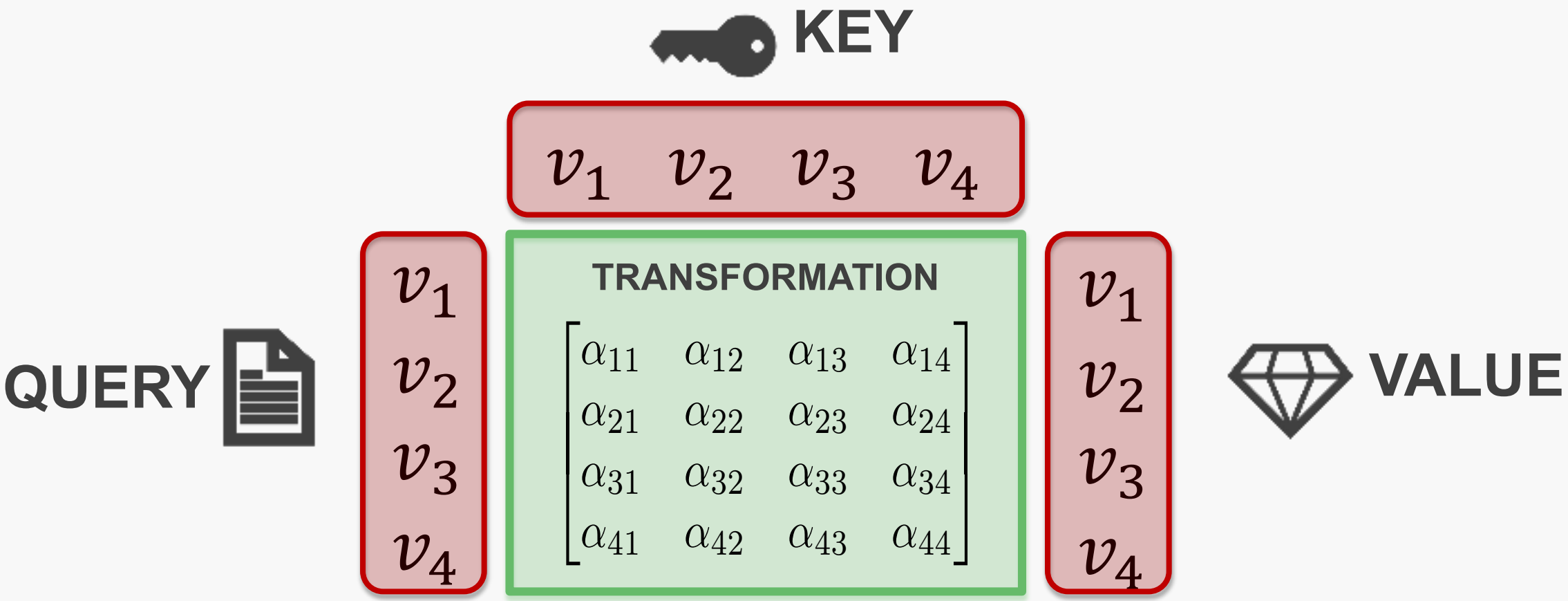


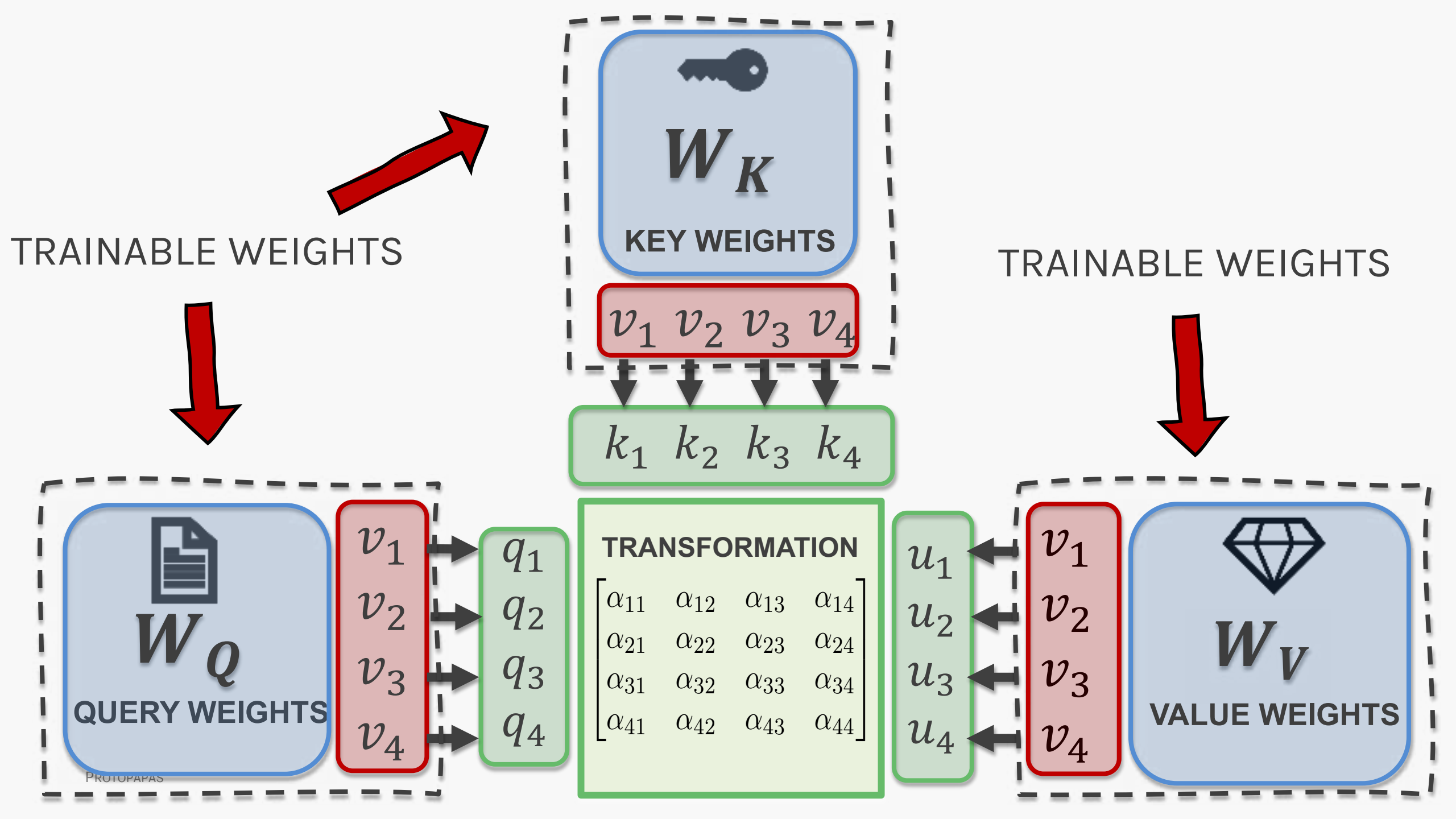
Database Analogy



Database Analogy

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




TRAINABLE WEIGHTS
TRAINABLE WEIGHTS
W_K
KEY WEIGHTS
v_1 v_2 v_3 v_4
k_1 k_2 k_3 k_4
W_Q
QUERY WEIGHTS
v_1 v_2 v_3 v_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}
u_1 u_2 u_3 u_4
v_1 v_2 v_3 v_4
W_V
VALUE WEIGHTS

TRAINABLE WEIGHTS

TRAINABLE WEIGHTS

By adding trainable weights, we allow our transformations to be flexible to the task

W_K

KEY WEIGHTS

v_1 v_2 v_3 v_4

k_1 k_2 k_3 k_4

W_Q

QUERY WEIGHTS

v_1 v_2 v_3 v_4

q_1 q_2 q_3 q_4

TRANSFORMATION

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

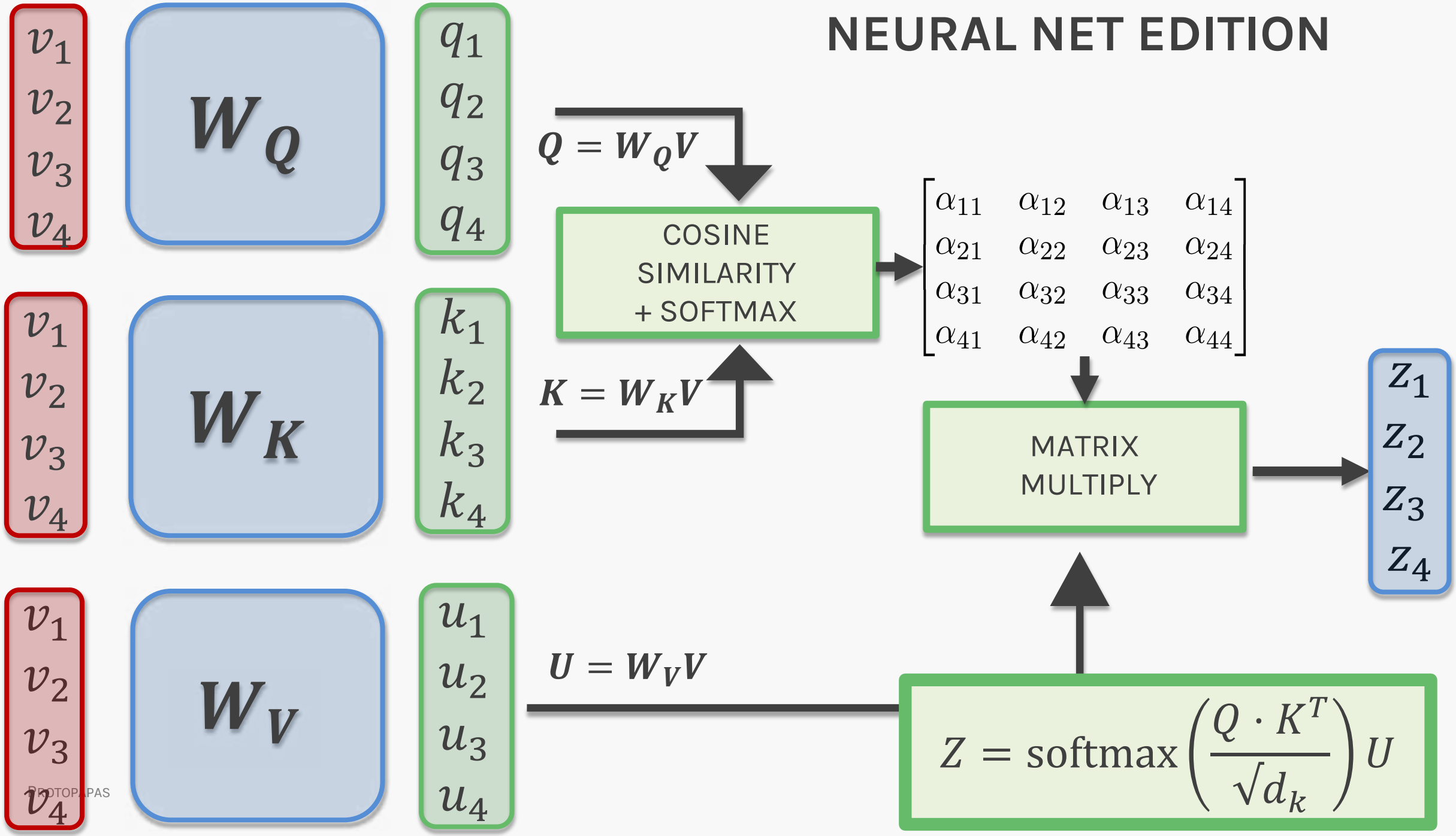
u_1 u_2 u_3 u_4

v_1 v_2 v_3 v_4

W_V

VALUE WEIGHTS

NEURAL NET EDITION



NEURAL NET EDITION

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

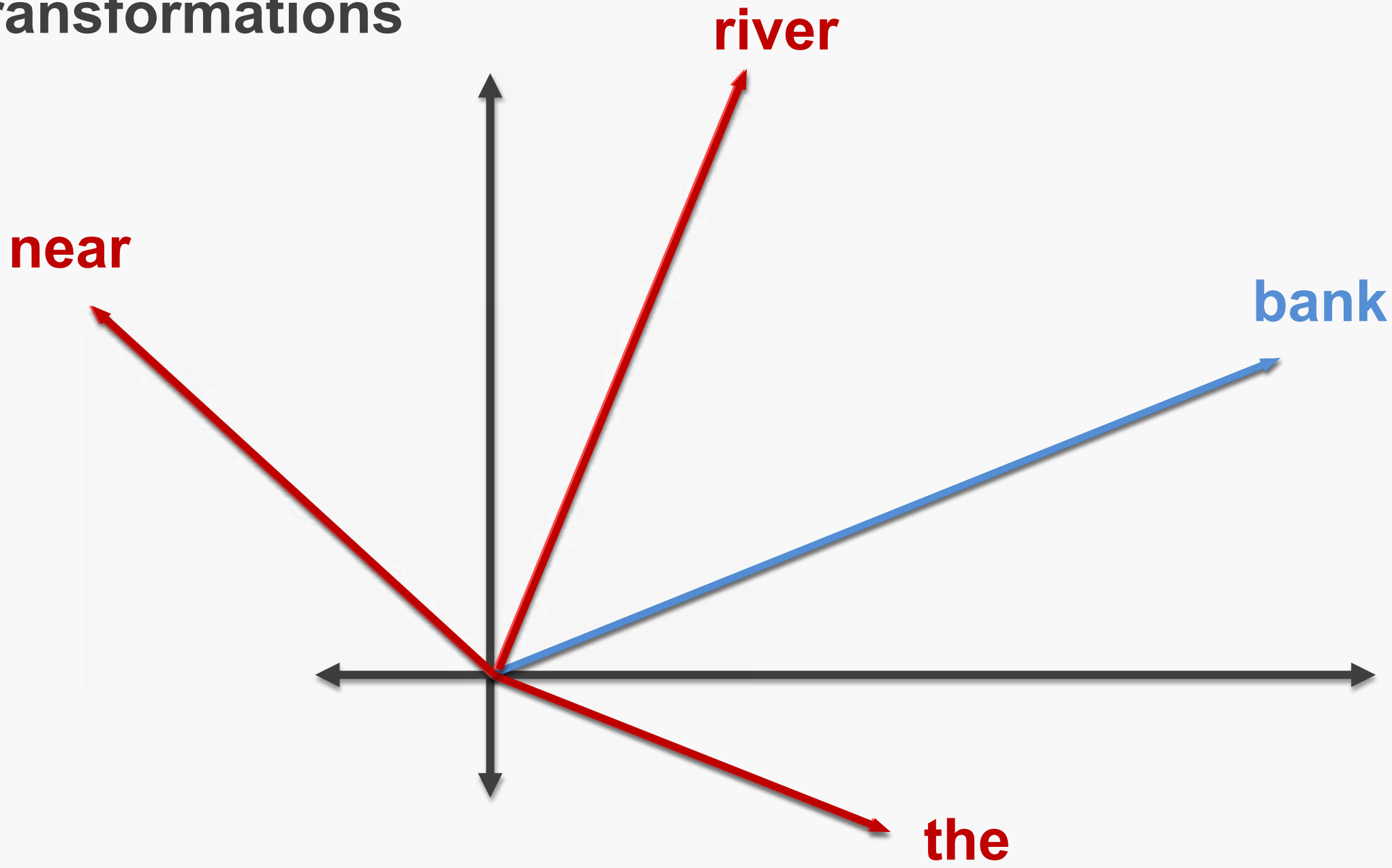
MATRIX

z_1 z_2

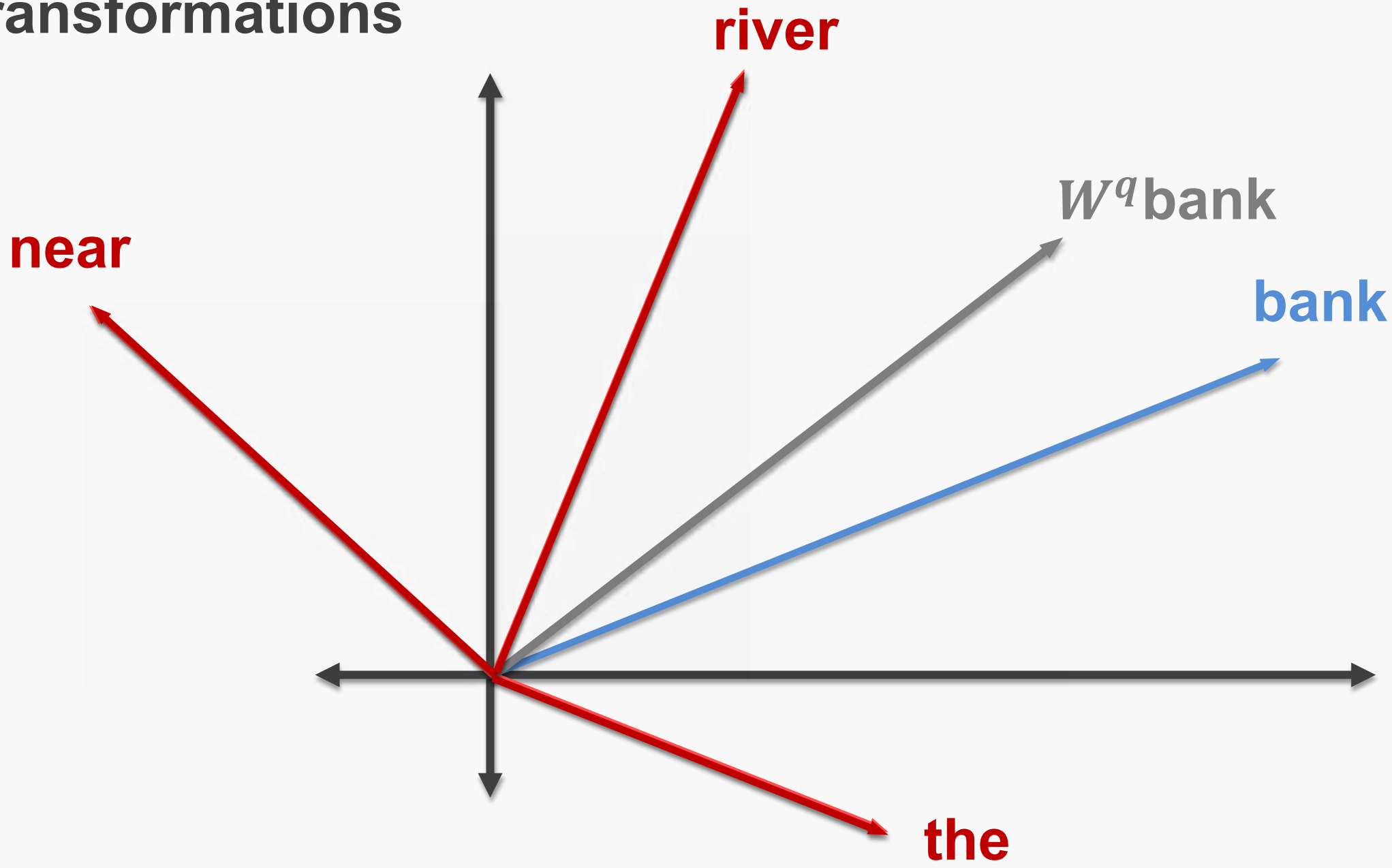
Original paper "Attention Is All You Need" calls it scaled dot product. d_k is the dimension of the key and query vectors

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

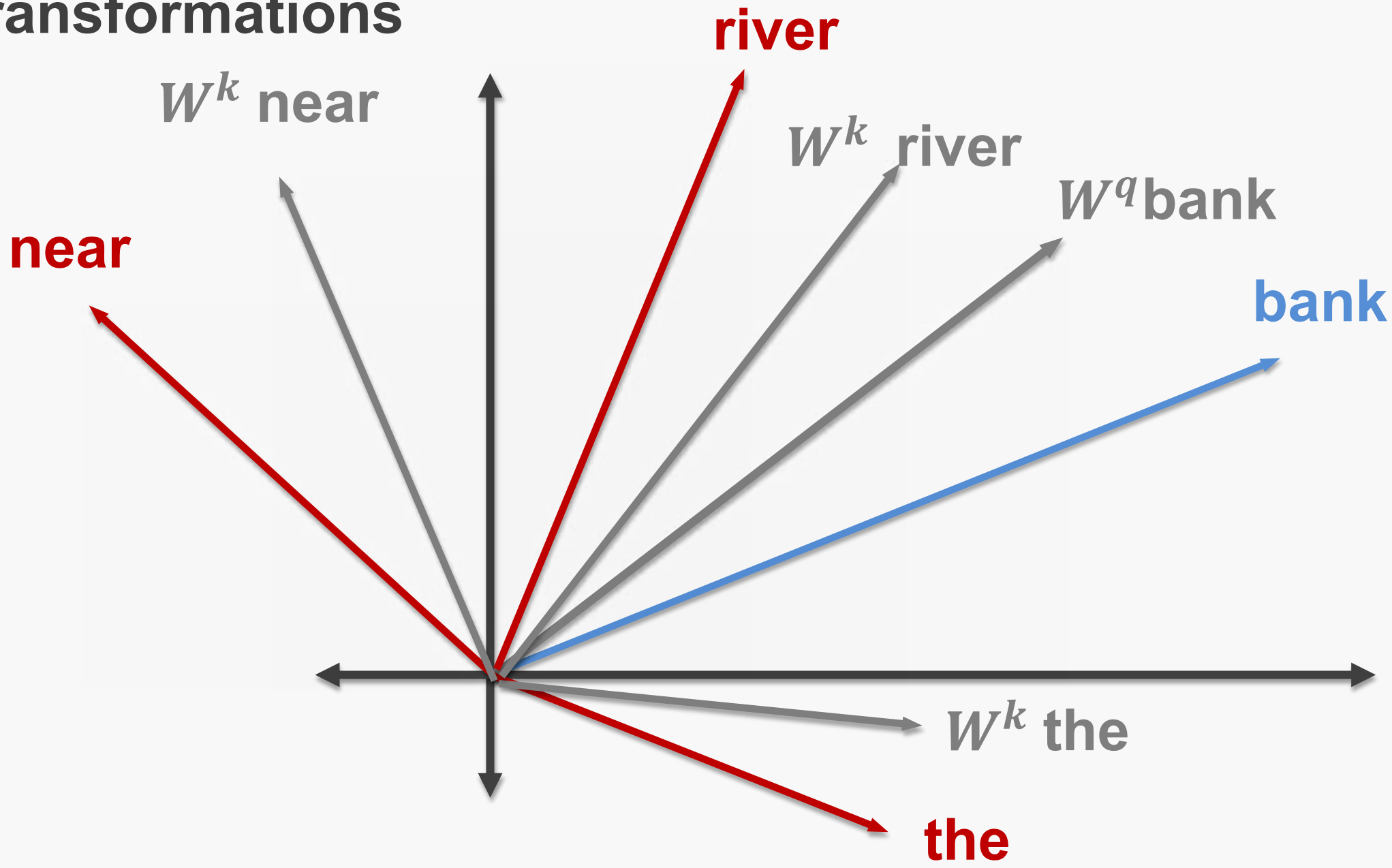
Transformations

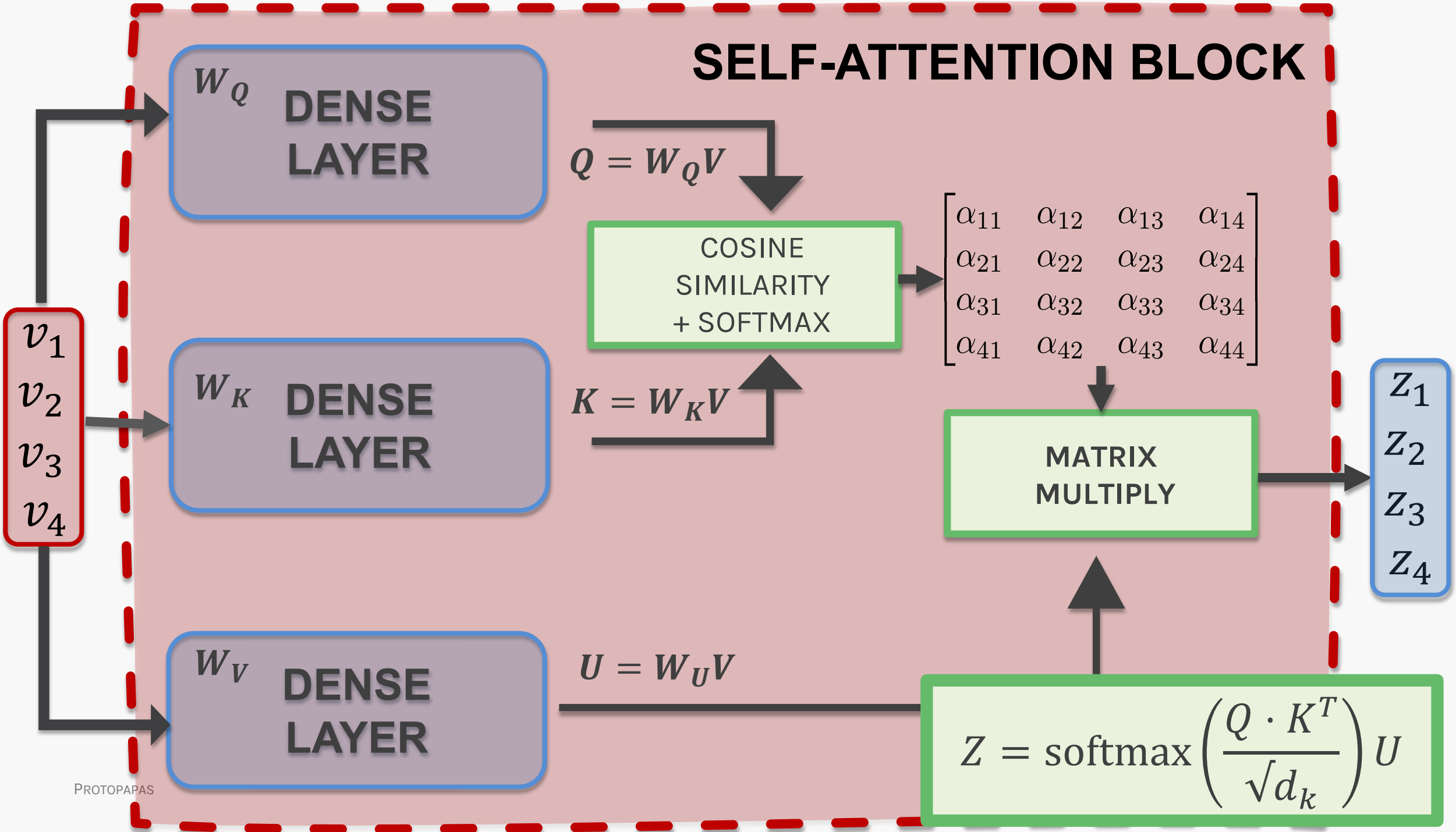


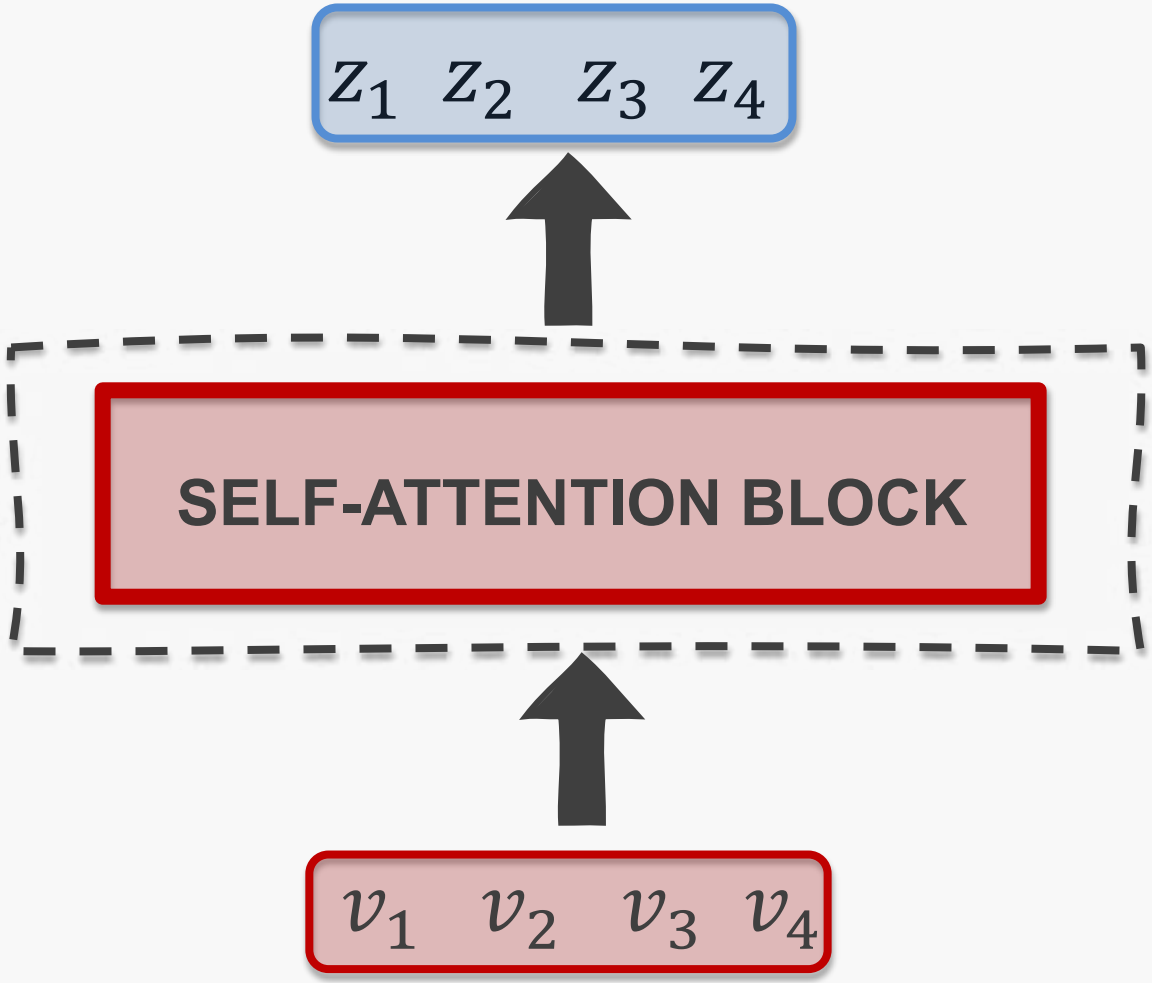
Transformations

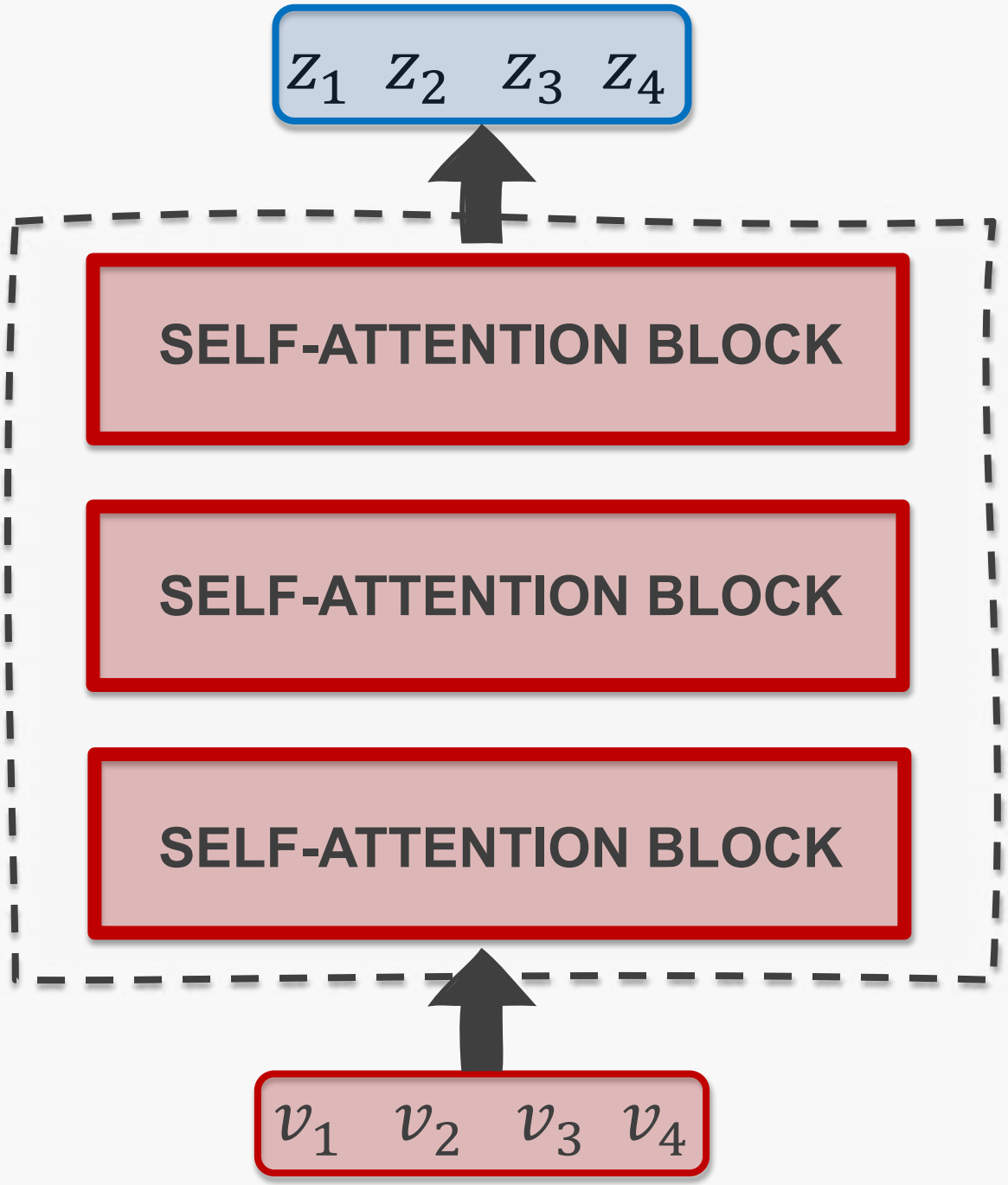


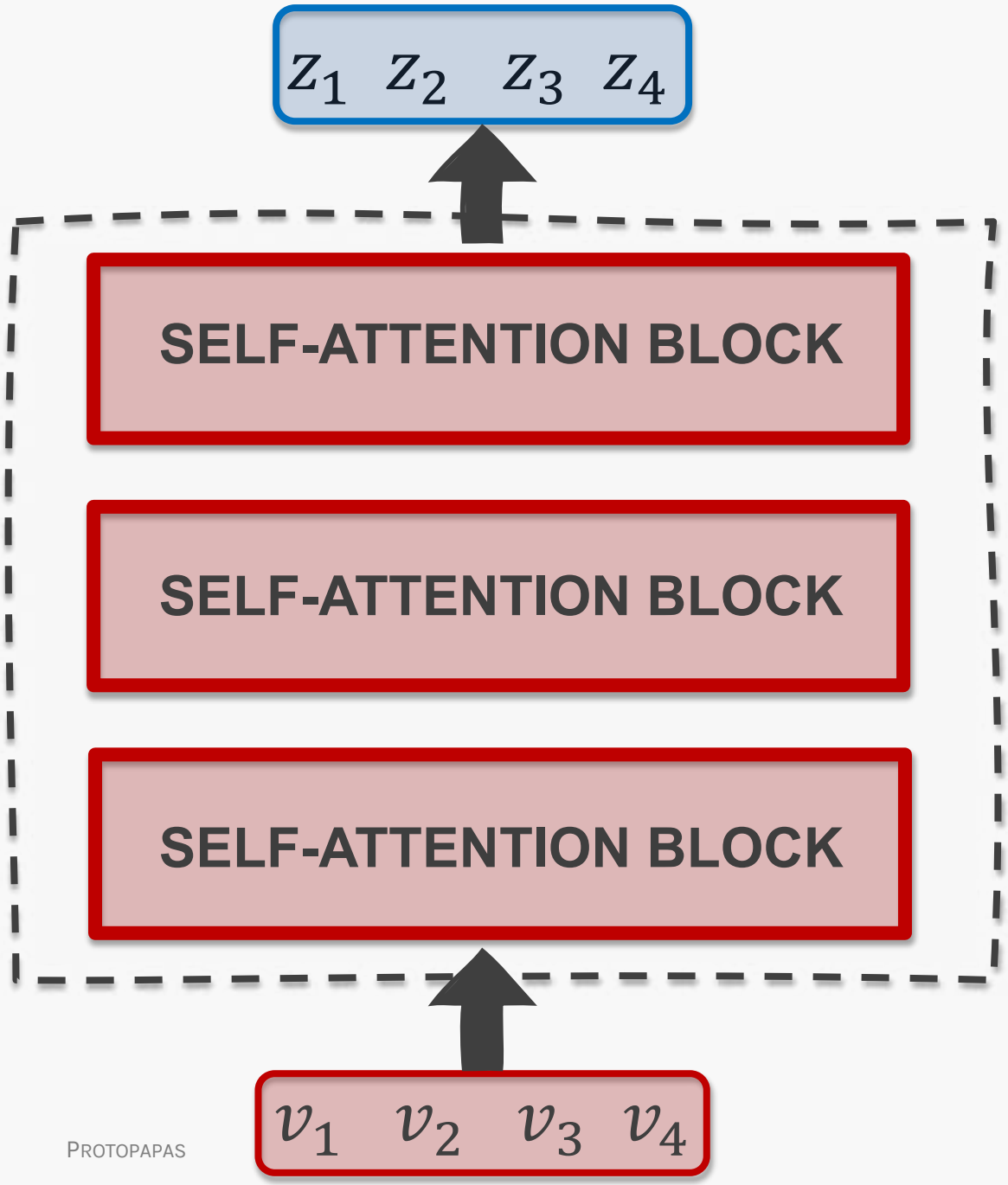
Transformations




SELF-ATTENTION BLOCK
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
v_1
v_2
v_3
v_4
$Q = W_Q V$
$K = W_K V$
$U = W_U V$
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
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) U$


z_1 z_2 z_3 z_4
SELF-ATTENTION BLOCK
v_1 v_2 v_3 v_4

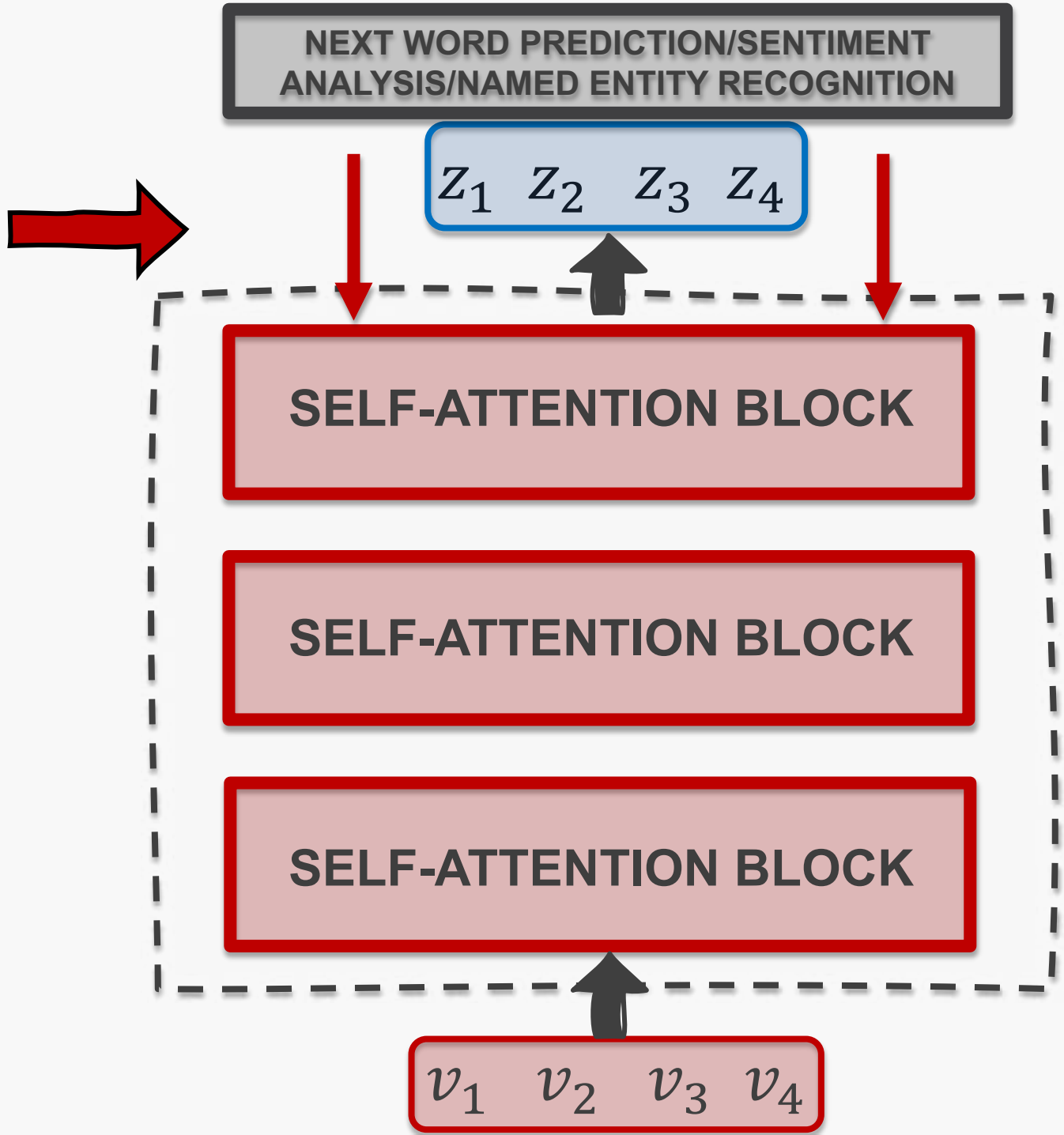

z_1 z_2 z_3 z_4
SELF-ATTENTION BLOCK
SELF-ATTENTION BLOCK
SELF-ATTENTION BLOCK
v_1 v_2 v_3 v_4

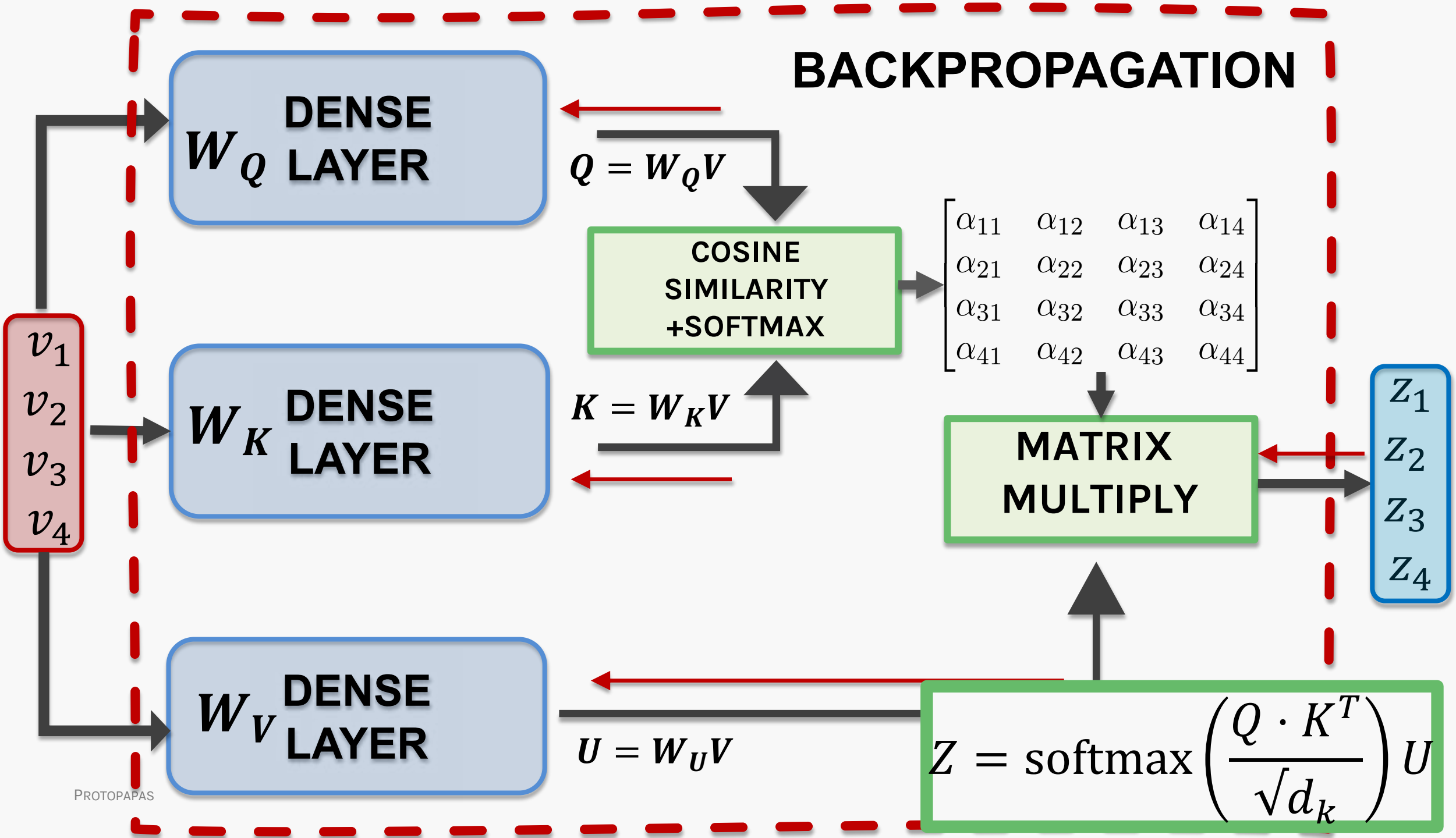


We could stack up these self-attention blocks to get richer and more complex contextual relationships

Flow of gradients

We could train these weights by using backpropagation using masked words or a task like sentiment analysis




BACKPROPAGATION
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
v_1
v_2
v_3
v_4
$Q = W_Q V$
$K = W_K V$
$U = W_U V$
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
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) U$

Attention

ATTENTION ISSUES?

- **Attention leads to limited contextual mapping.**
- There is no positional information encoded.

Attention: Do we have enough attention?

ATTENTION MECHANISM #1

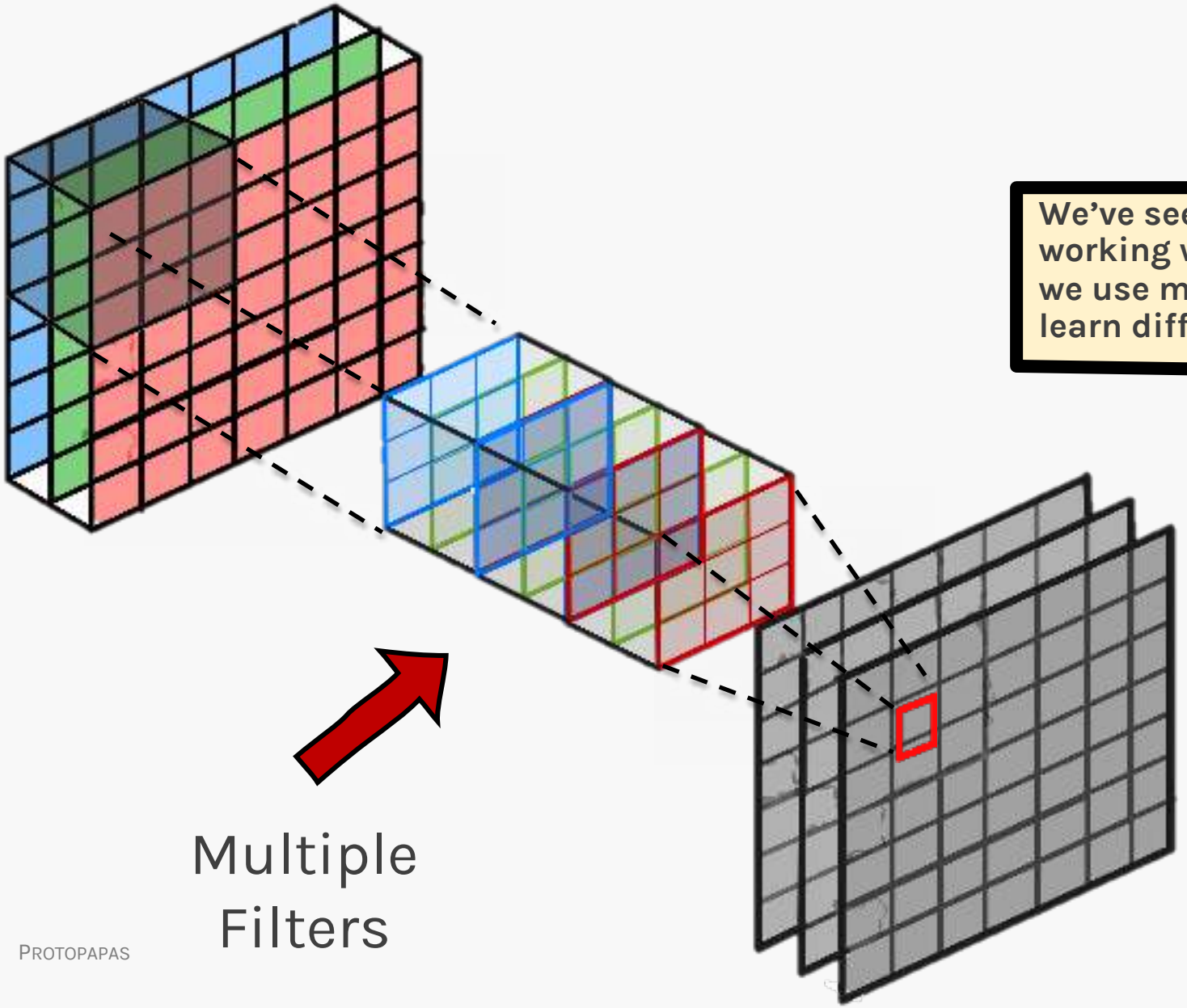
ATTENTION MECHANISM #2

Ignacio spoke to Pavlos about attention

ATTENTION MECHANISM #3

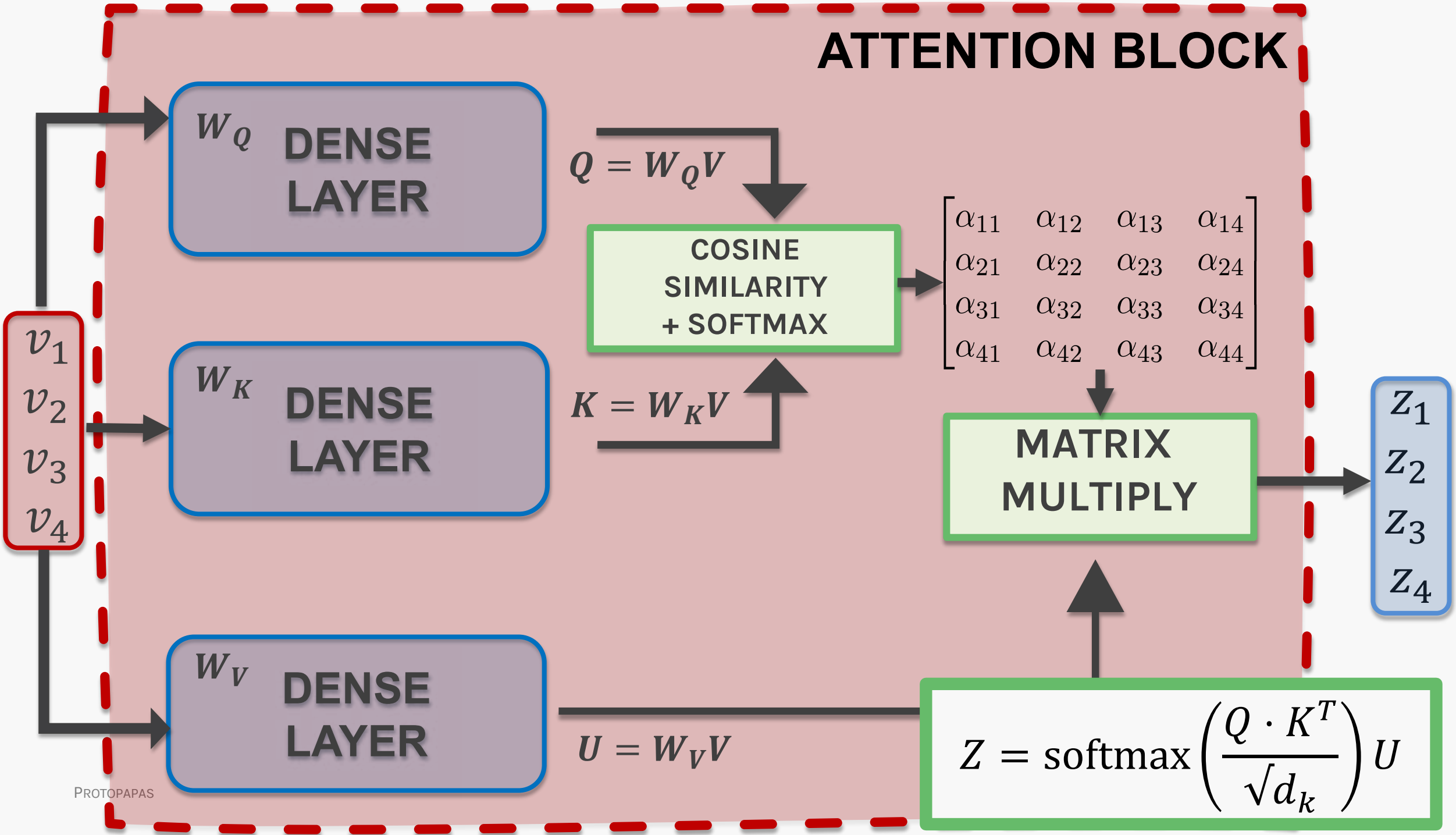
We need to have multiple attention mechanisms to look for different relations.

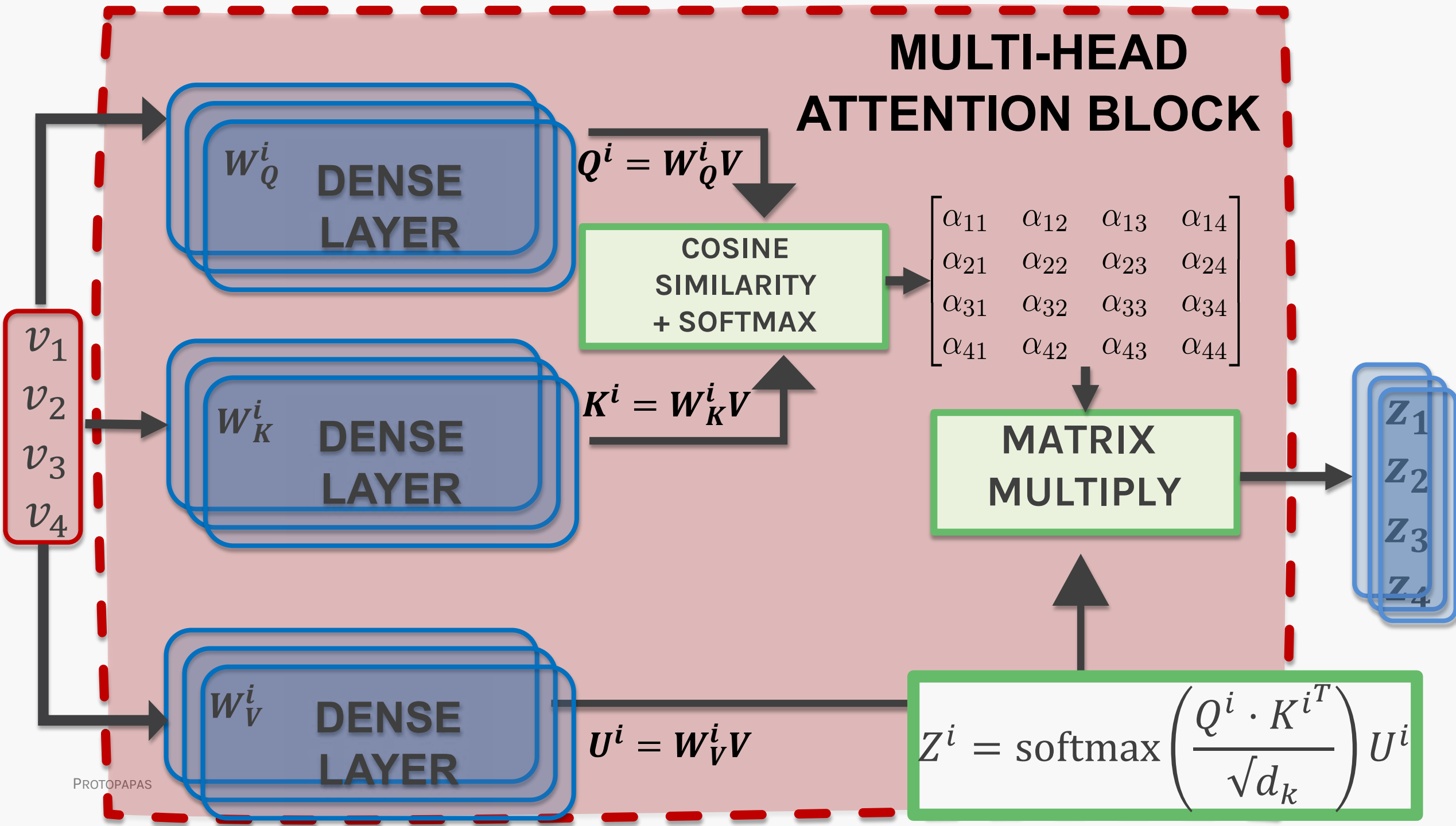
Analogy - CNN Filters

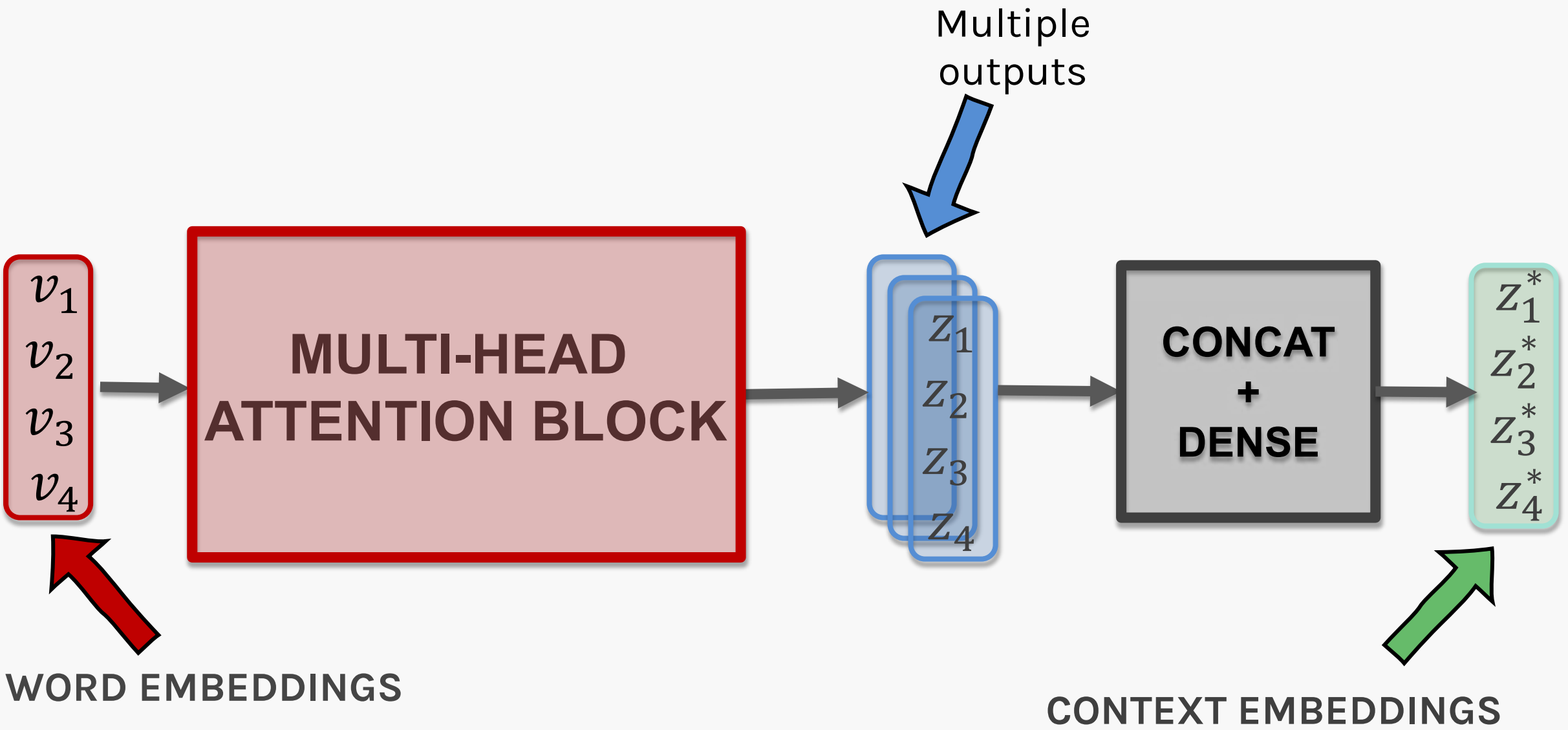


Multiple Filters

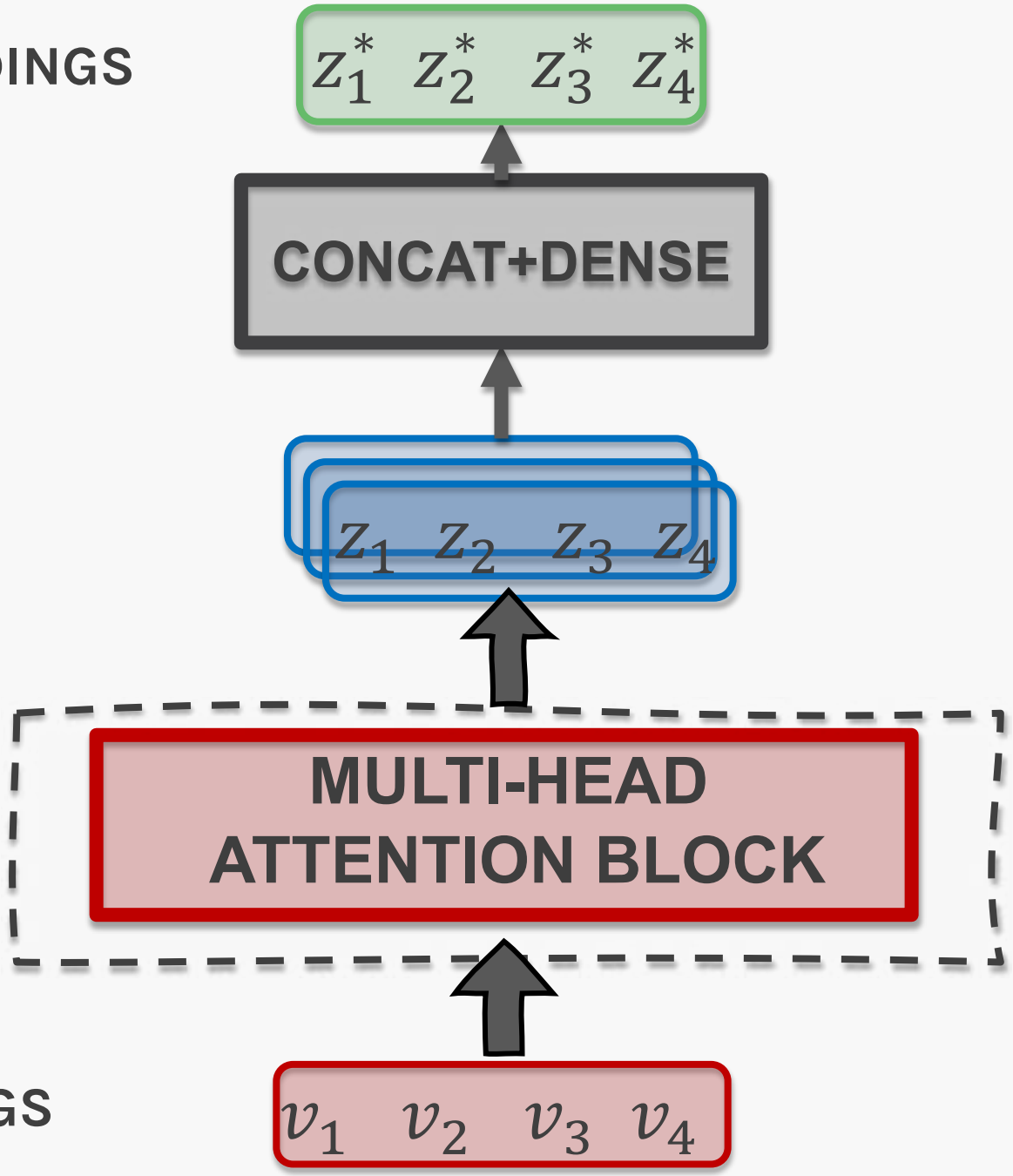
We've seen this before while working with CNNs, where we use multiple filters to learn different features


ATTENTION BLOCK
W_Q DENSE LAYER
W_K DENSE LAYER
W_V DENSE LAYER
v_1
v_2
v_3
v_4
$Q = W_Q V$
$K = W_K V$
$U = W_V V$
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
$Z = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) U$


MULTI-HEAD ATTENTION BLOCK
v_1
v_2
v_3
v_4
W_Q^i
DENSE LAYER
W_K^i
DENSE LAYER
W_V^i
DENSE LAYER
$Q^i = W_Q^i V$
$K^i = W_K^i V$
$U^i = W_V^i V$
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
$Z^i = \text{softmax}\left(\frac{Q^i \cdot K^{iT}}{\sqrt{d_k}}\right) U^i$
z_1
z_2
z_3
z_4
PROTOPAPAS


Multiple outputs
v_1
v_2
v_3
v_4
MULTI-HEAD ATTENTION BLOCK
z_1
z_2
z_3
z_4
CONCAT + DENSE
z_1^*
z_2^*
z_3^*
z_4^*
WORD EMBEDDINGS
CONTEXT EMBEDDINGS

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

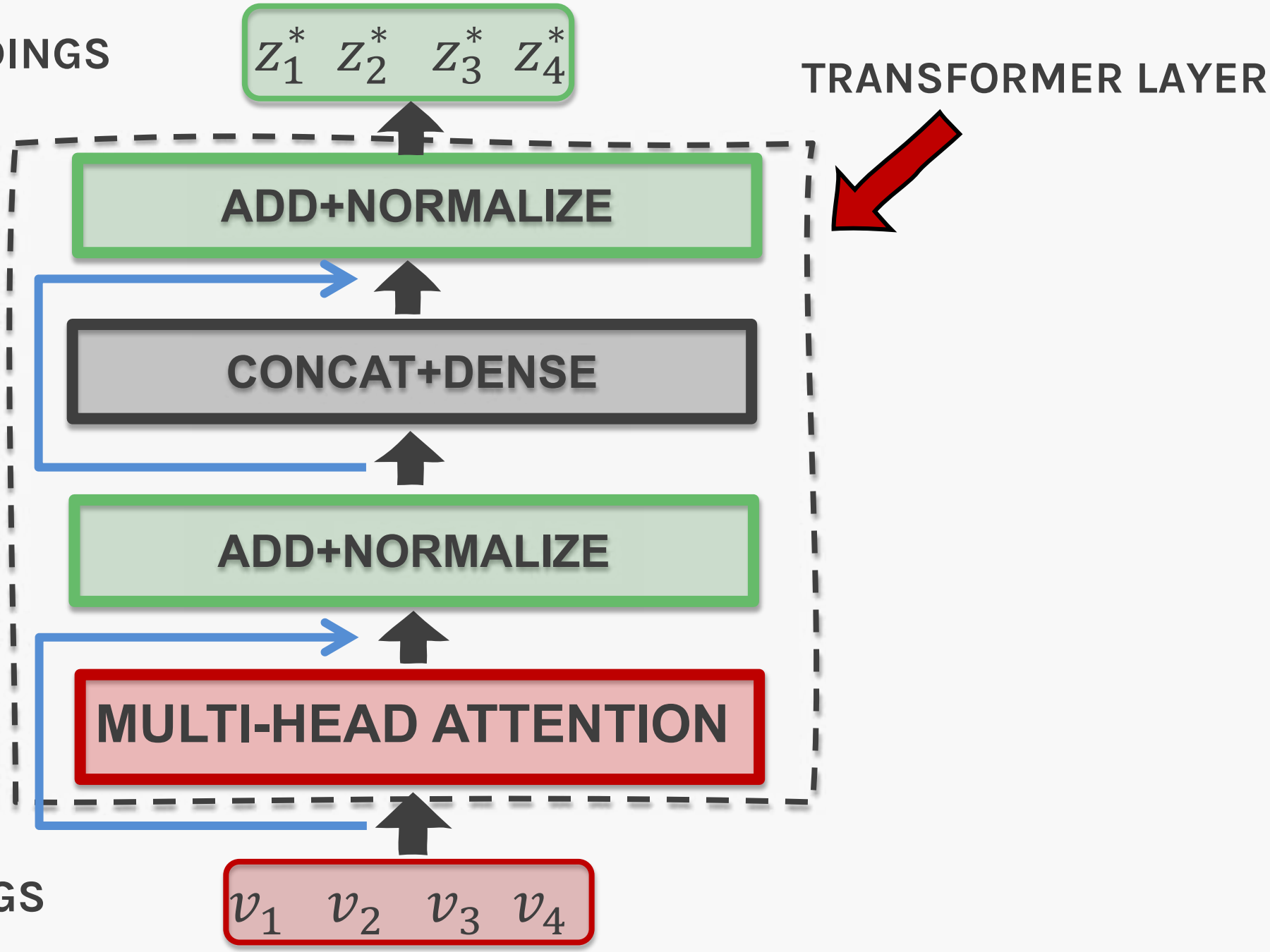
$z_1^* \quad z_2^* \quad z_3^* \quad z_4^*$

CONCAT+DENSE

MULTI-HEAD ATTENTION

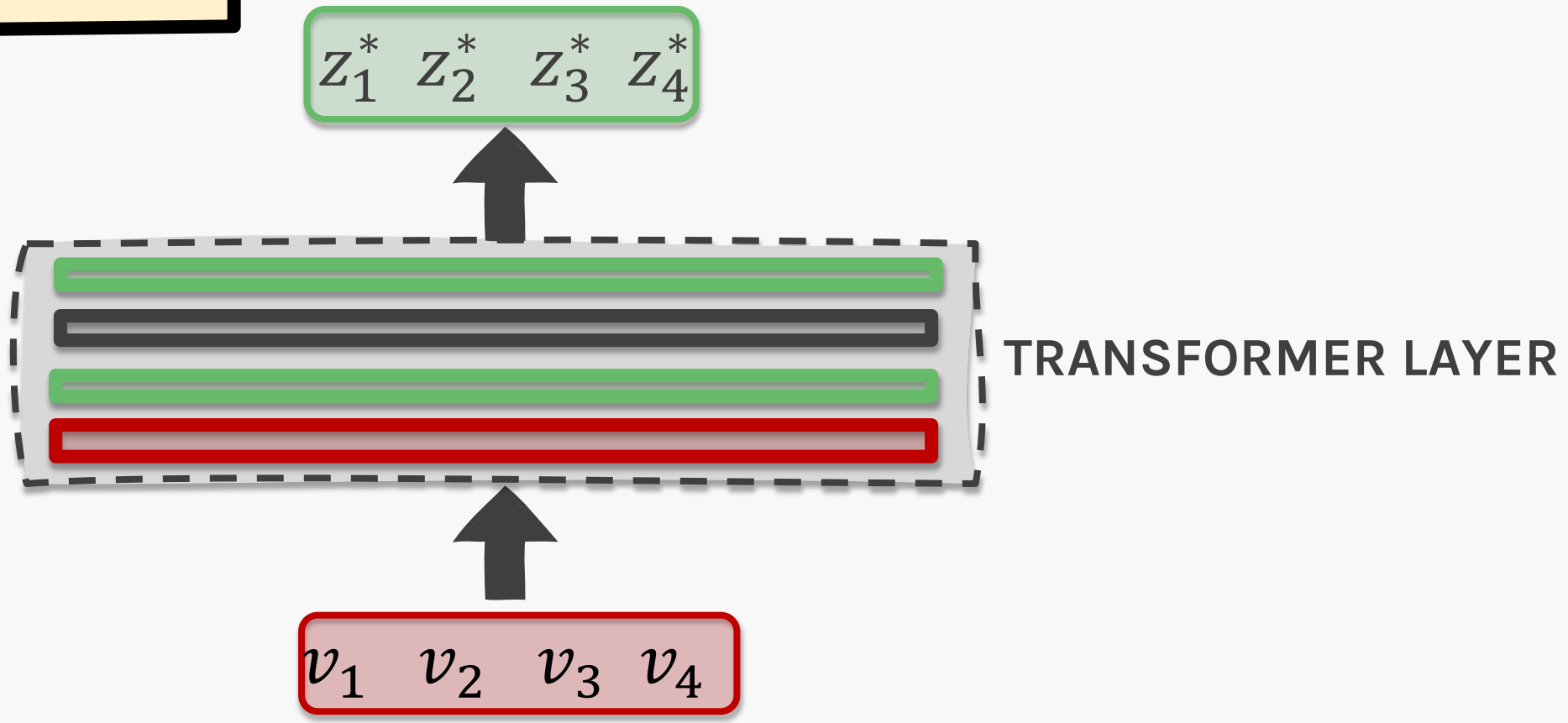
$v_1 \quad v_2 \quad v_3 \quad v_4$

CONTEXT EMBEDDINGS

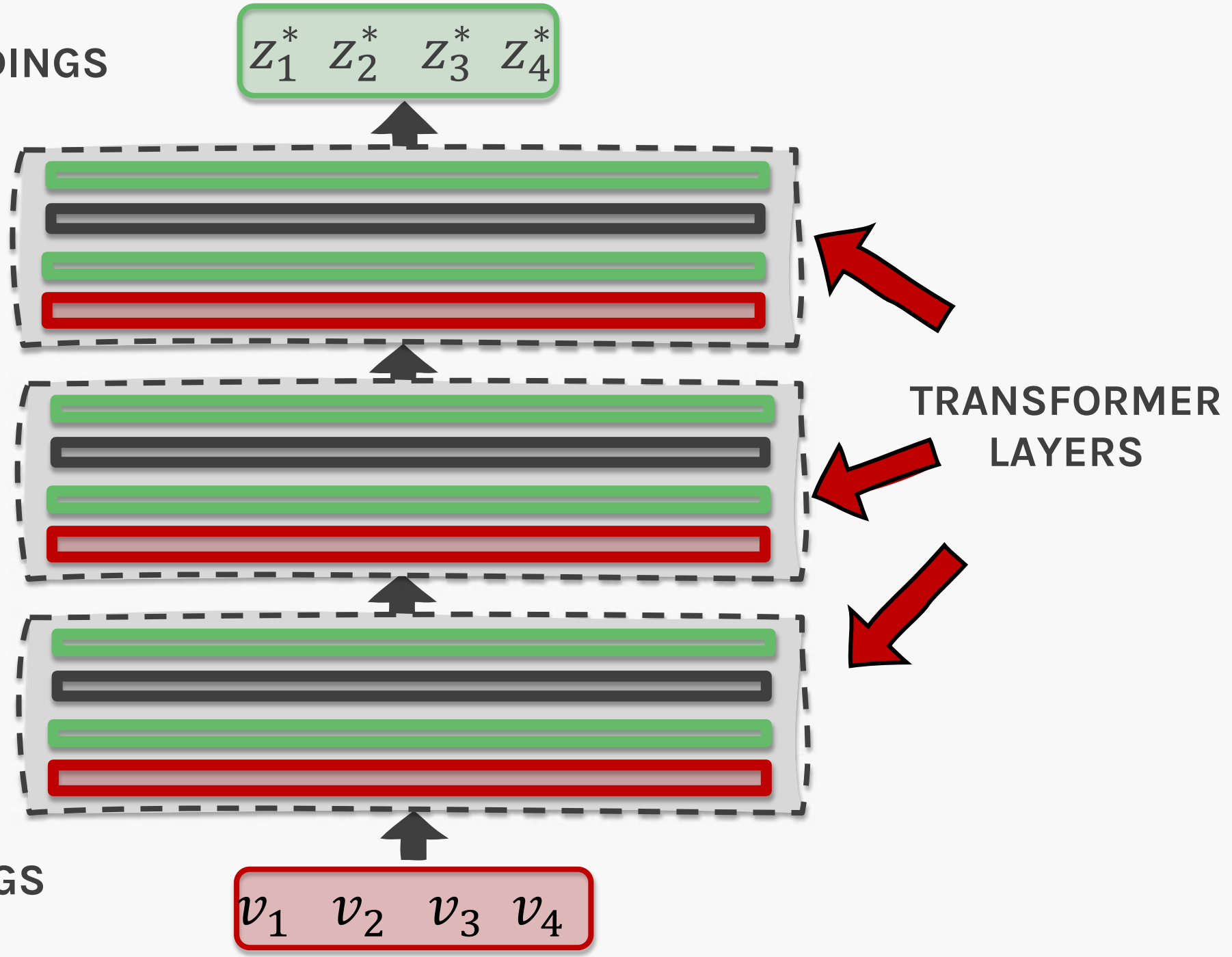


WORD EMBEDDINGS

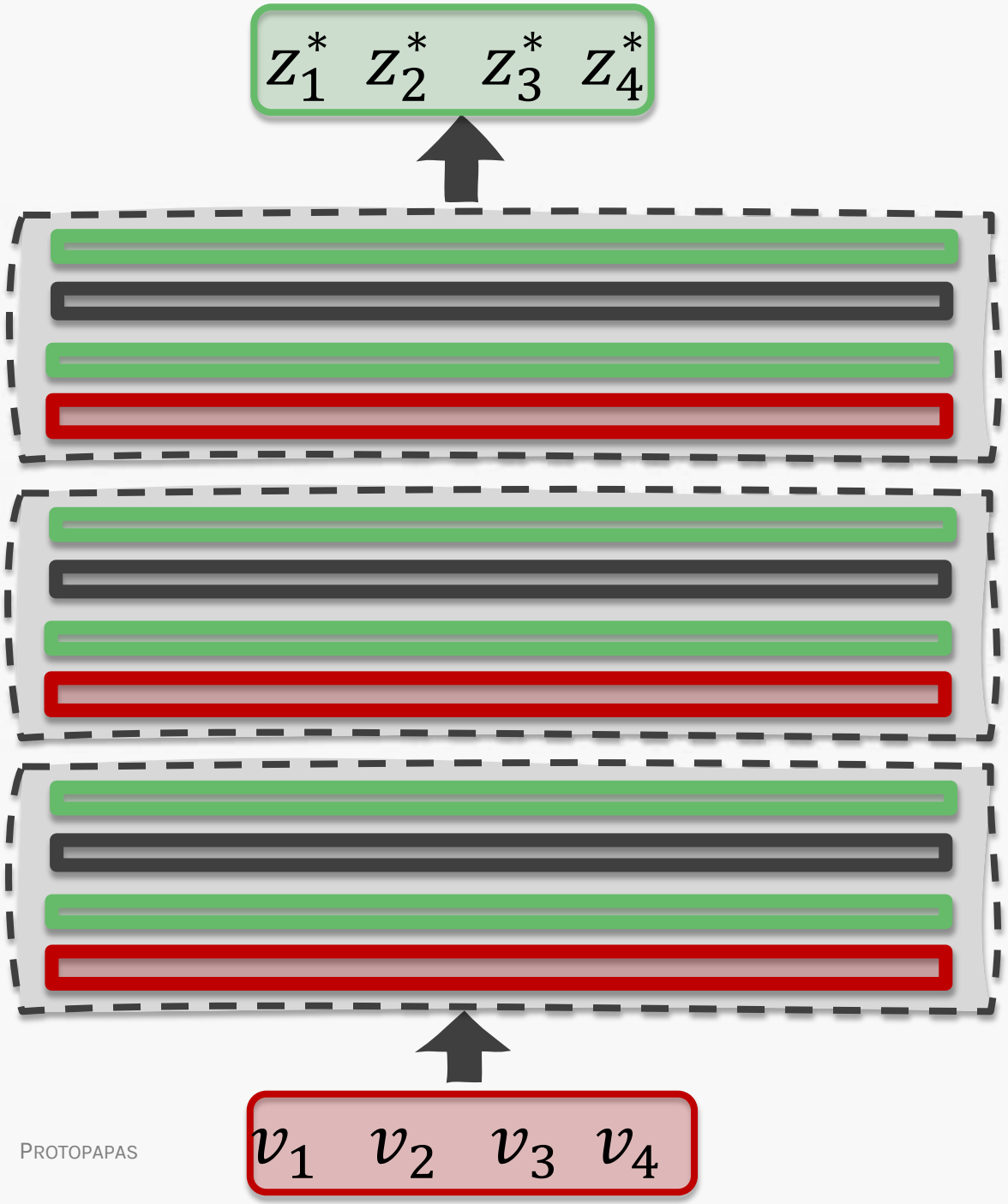
Why not stack multiple transformer layers?



CONTEXT EMBEDDINGS



WORD EMBEDDINGS


z_1^* z_2^* z_3^* z_4^*
v_1 v_2 v_3 v_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**

Ignacio spoke to Pavlos about attention

Attention

MULTI-HEAD ATTENTION ISSUES?

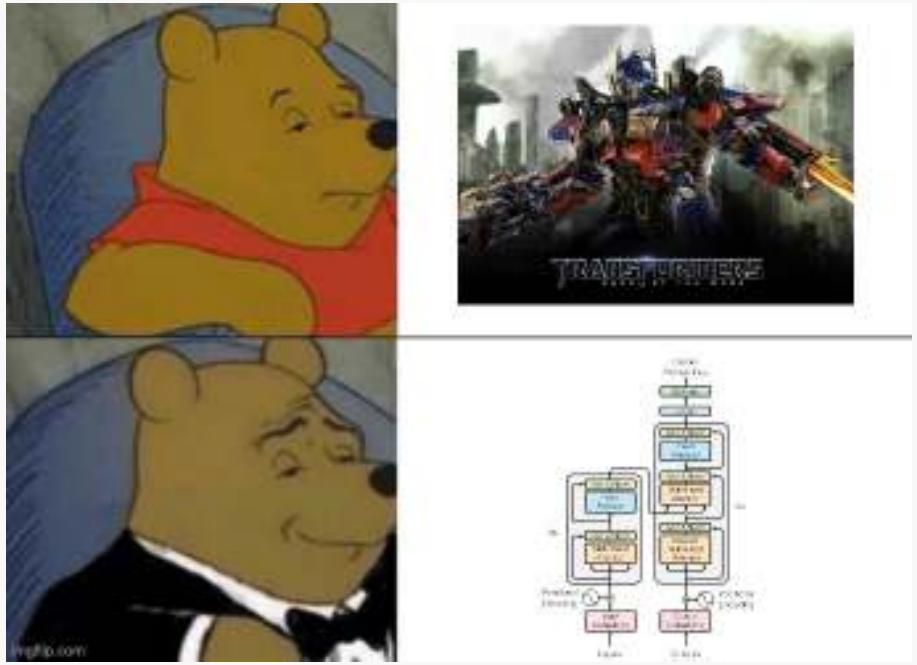
- ~~No weights trained in the process~~
- ~~Attention leads to limited contextual mapping~~
- **There is no positional information encoded**

Pavlos spoke to Ignacio 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

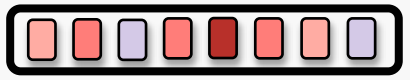


💡 **IDEA #255:** Positional Encoding

Positional Encoding

Assume an input embedding v_i of some dimension for the word 'Ignacio' at the position i in the sentence

v_i



This input embedding will be the same for any position in the sentence

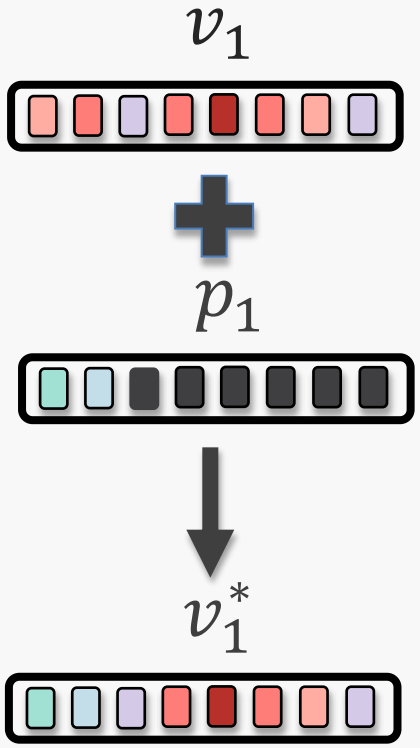
v_1 v_1 v_1

We hope to modify the embedding with some positional information.

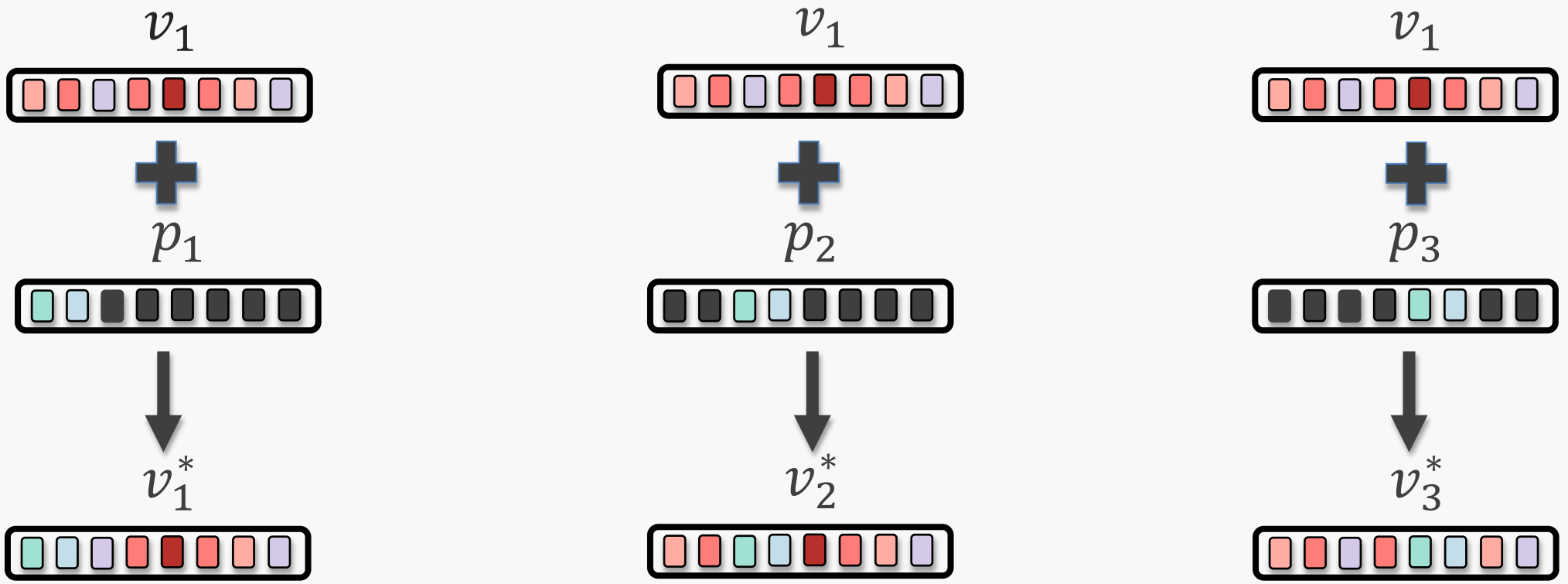
Positional Encoding

- A very simple way to do this would be to modify the input embedding v_i with a positional vector p_i which encodes some information of the position of the input embedding

- Thus, the same input embedding v_i will have a different value v_i^* depending on the position of the embedding in the sentence



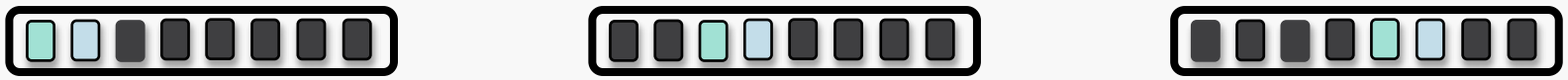
Positional Encoding



In the above case, $v_1^* \neq v_2^* \neq v_3^*$

Positional Encoding

p_1 p_2 p_3



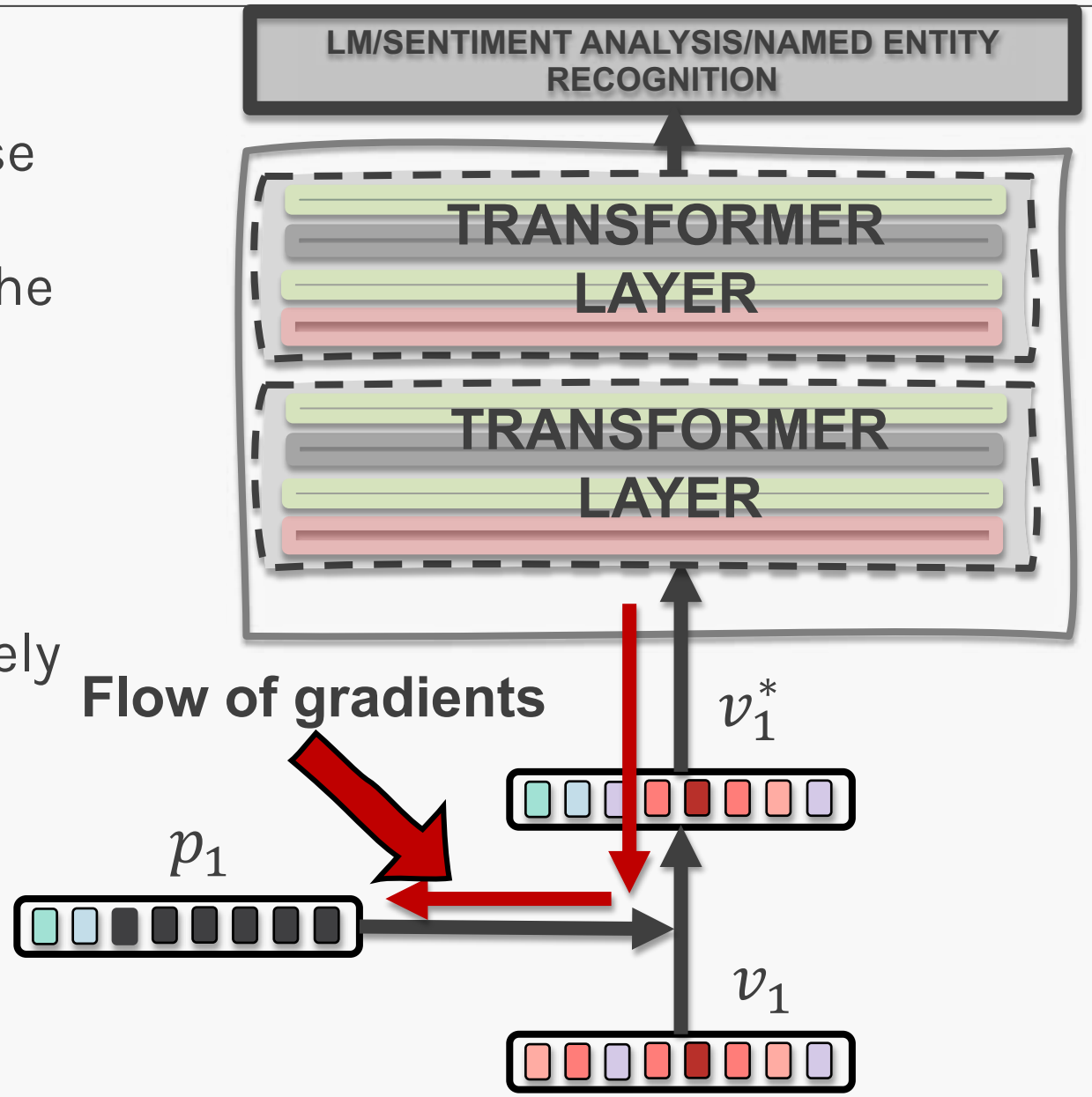
But how do we find these positional embedding vectors?

Don't worry Ignacio, we have several options to choose from...

Positional Encoding

- The no-nonsense way to find these positional encodings would be to learn them during training (like the Query, Key & Value matrices)

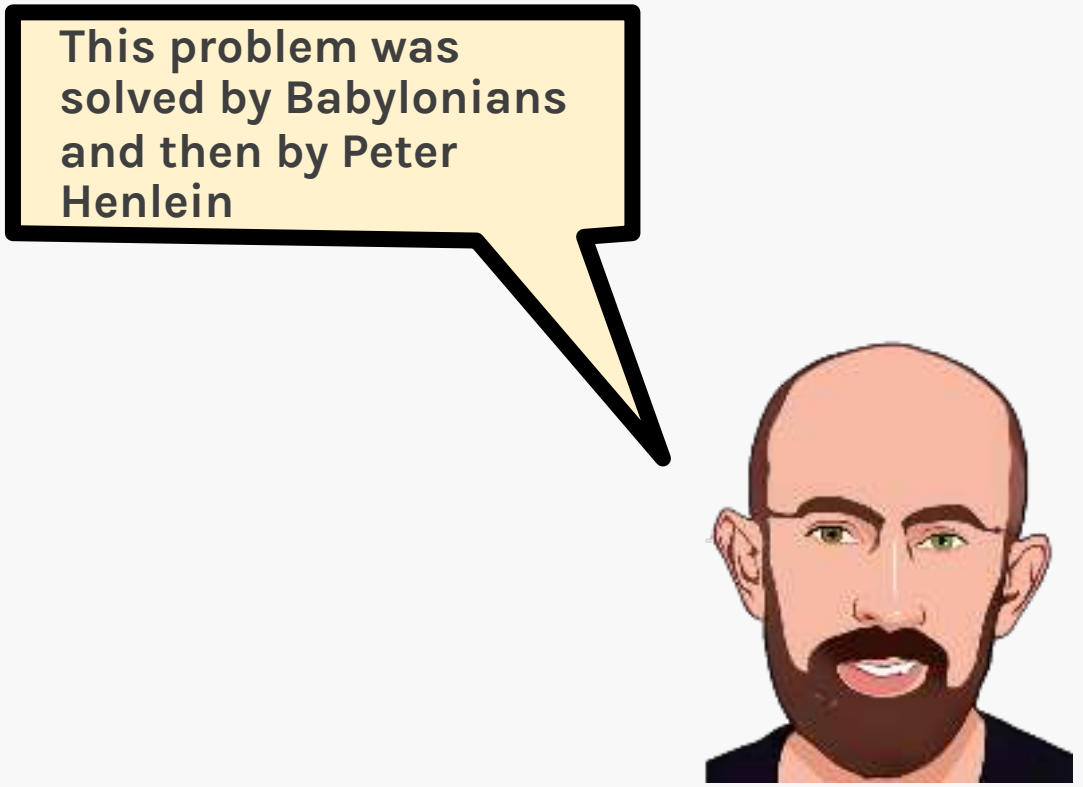
- But the number of learnable parameters in a stacked multi-head attention block can massively grow, and this can lead to poor training



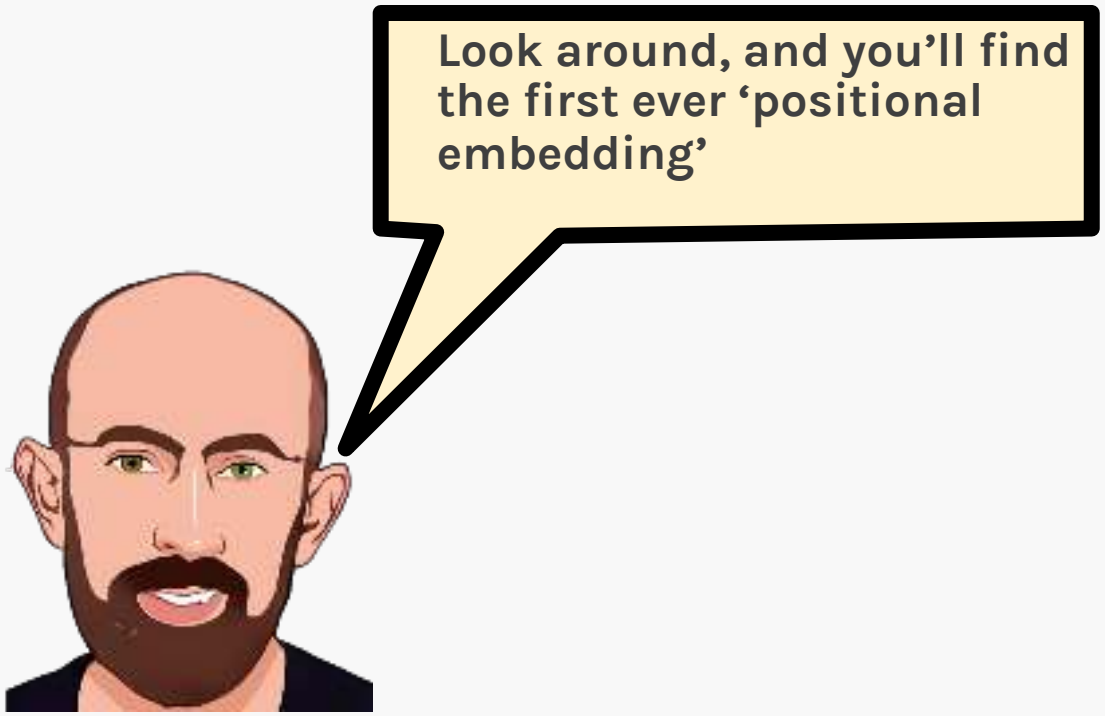
Positional Encoding

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 Embedding – Motivation



Positional Embedding – Motivation

- The clock was an ingenious, compact solution to keep time
- Using the H:M:S system for hour, minute and second, we actually read a tuple of three numbers (H,M,S), with each dimension representing a **unique** time of the day
- We achieve this by setting separate frequencies for the hour, minute & second

The mechanical clock

Positional Embedding – Motivation

Even though the *seconds* hand repeats every minute, the overall tuple is unique

Positional Embedding - Motivation

- In principle, we could add more *hands* to the clock, for days of the week, week of the month, month of the year etc.
- With enough *hands,* we could uniquely represent any time series from the start of the universe to the end
- This powerful idea can be used to generate scale independent positional encodings

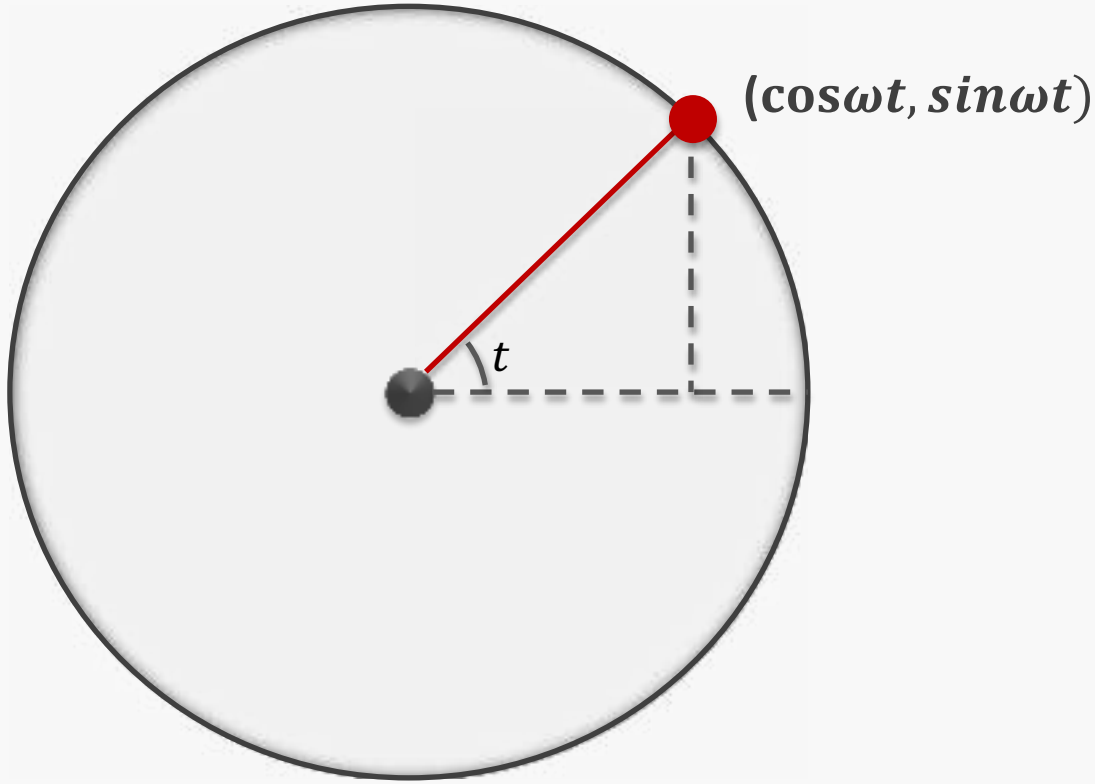
Positional Embedding - Motivation

But how can we mathematically represent the motion of the hand of the clock?

Simple geometry!

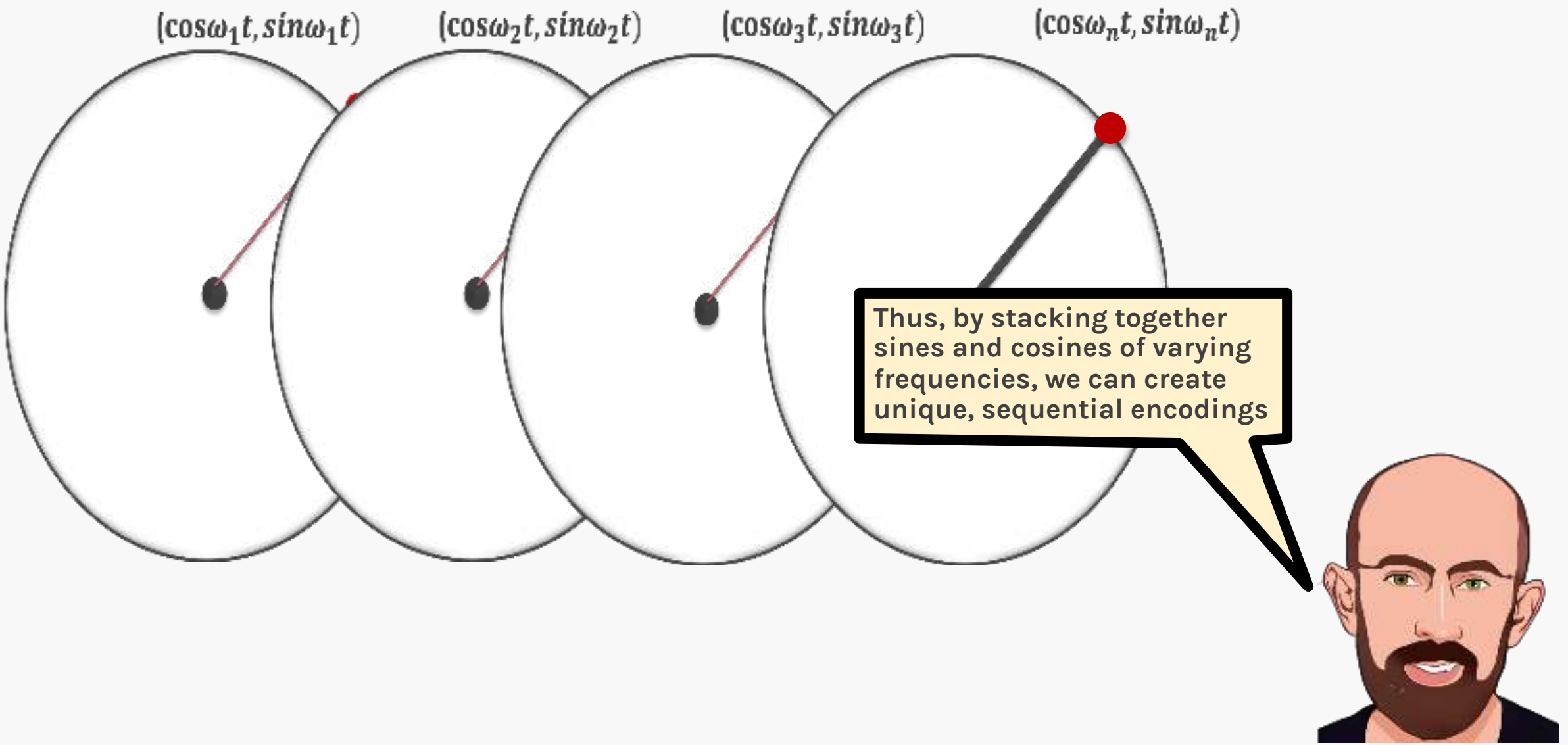
Positional Embedding - Motivation

- We can sufficiently define the path of a unit length around a circle by two hyperparameters:
  - Angle of rotation $\boldsymbol{t}$
  - Frequency of rotation $\boldsymbol{\omega}$



- The combination the sine and the cosine together can completely define the unit circle

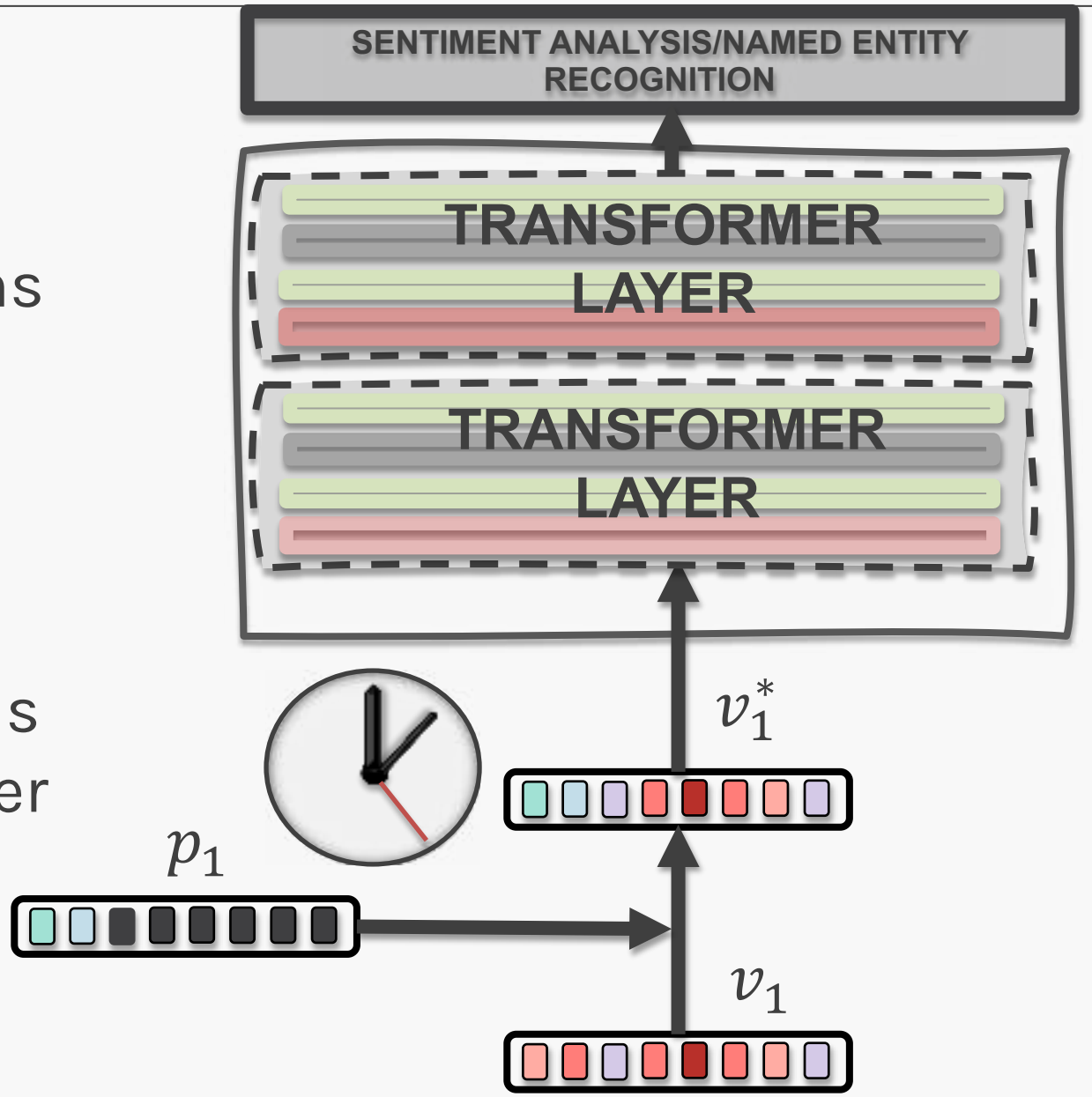
Positional Embedding - Motivation



Positional Encoding

- In the experimental results, the positional encodings using deterministic functions were found to be as effective as those learned using backpropagation.

- However, since this method is more general and much faster to implement, it is usually preferred.



Positional Embedding

TECHNICAL DETAILS

Let t be the desired position in an input sentence, $\vec{p}_t \in \mathbb{R}^d$ be it's corresponding enco
and d be the encoding dimension.

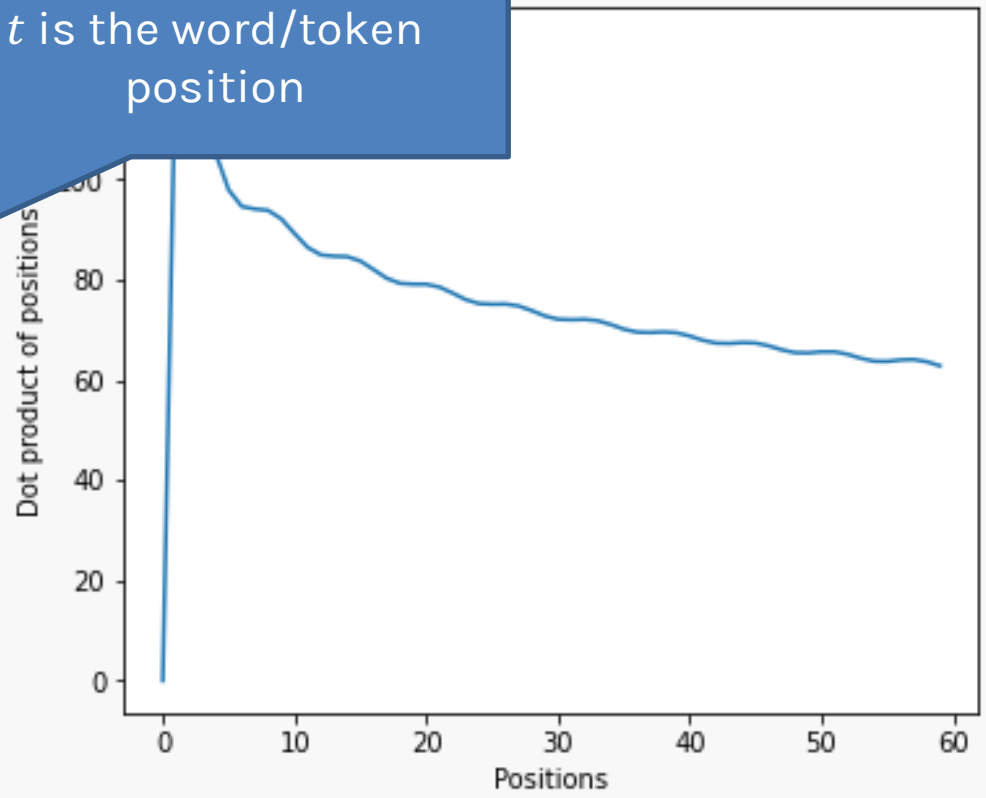
Then $f: \mathbb{N} \to \mathbb{R}^d$ is the function that produces
and is defined as:

- i is the positional embedding vector index
- d is the positional embedding dimension
- t is the word/token position

$$f(t)^{(i)} := \begin{cases} \sin(\omega_k t) \ , if \ i = 2k \\ \cos(\omega_k t) \ , if \ i = 2k+1 \end{cases}$$

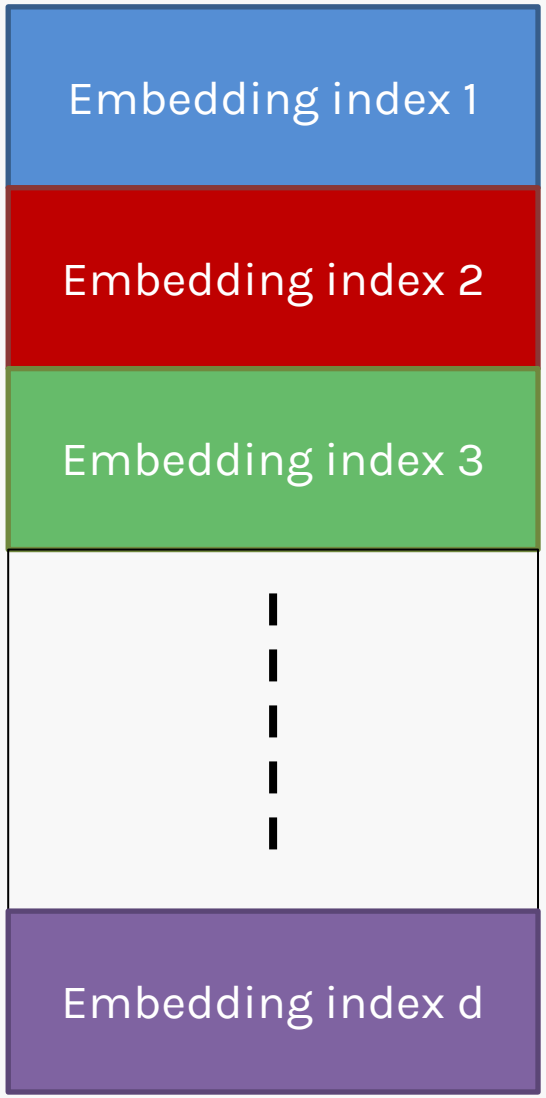
Where,

$$\omega_k = \frac{1}{10000^{\frac{2k}{d}}}$$

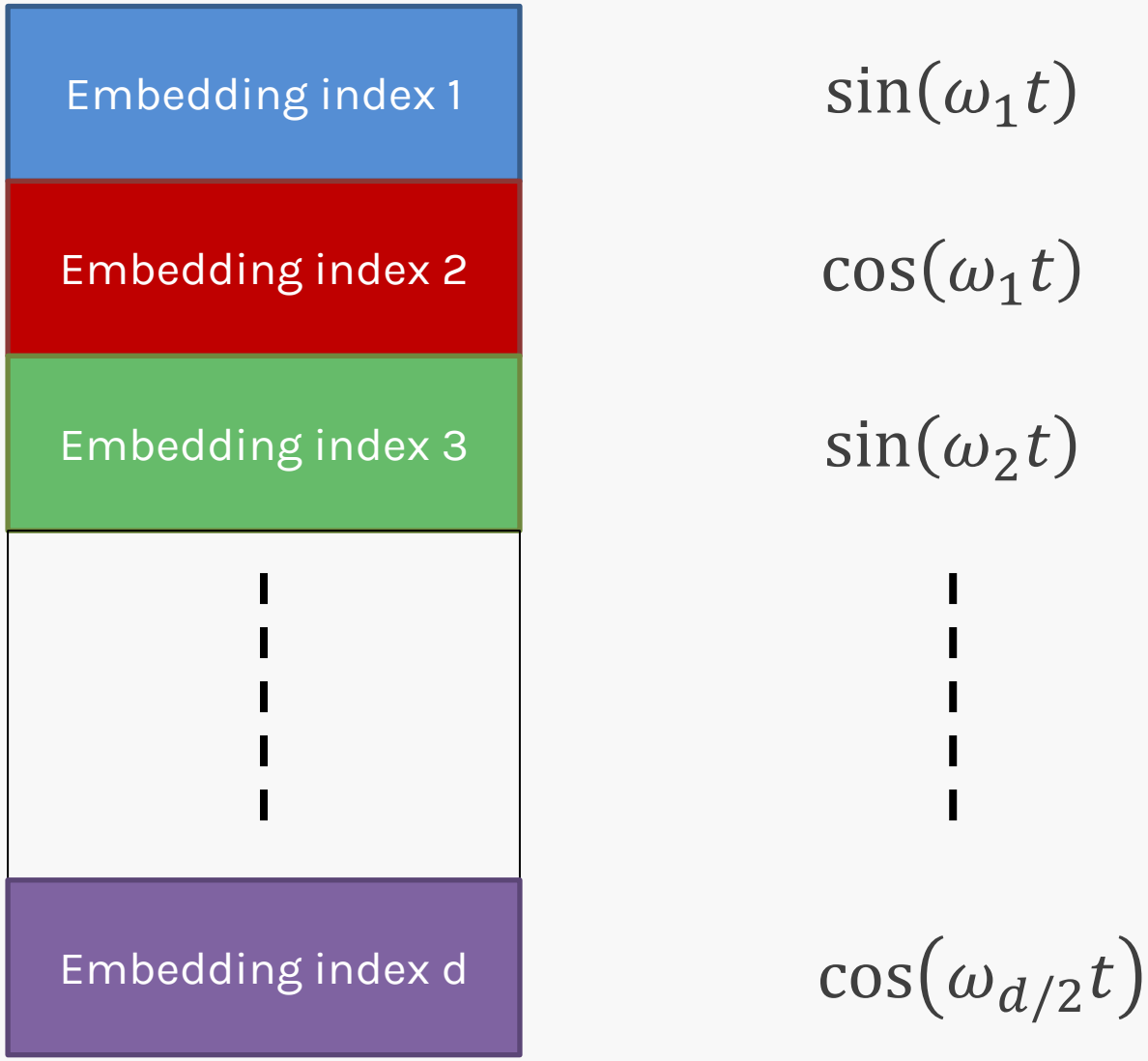


Positional Encoding

We have our word embedding vector



Positional Encoding

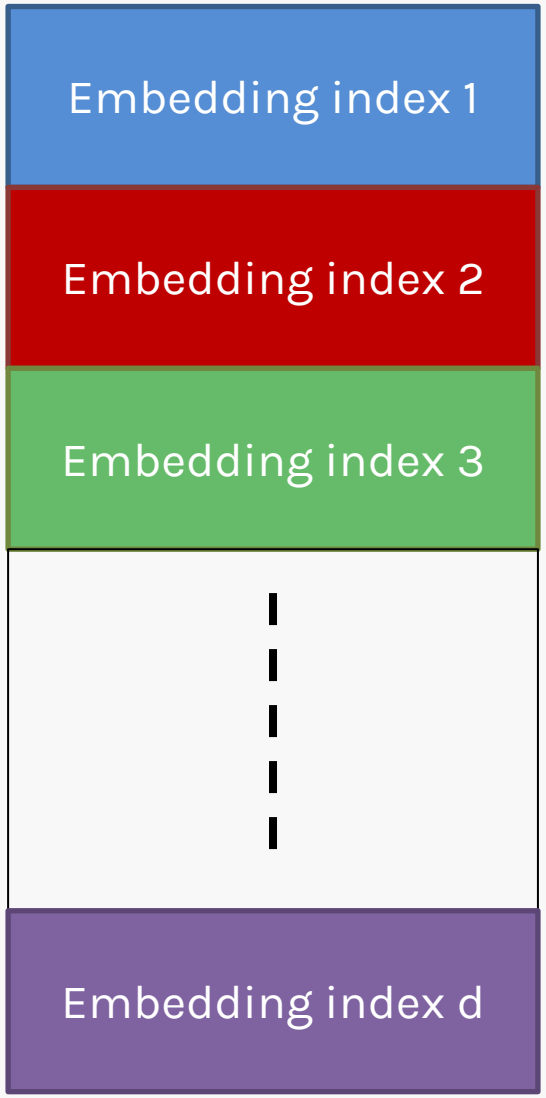


Now remember,

$$\omega_k = \frac{1}{10000^{\frac{2k}{d}}}$$

So, as k increases, the frequency of the sinusoidal wave will decrease.

Positional Encoding



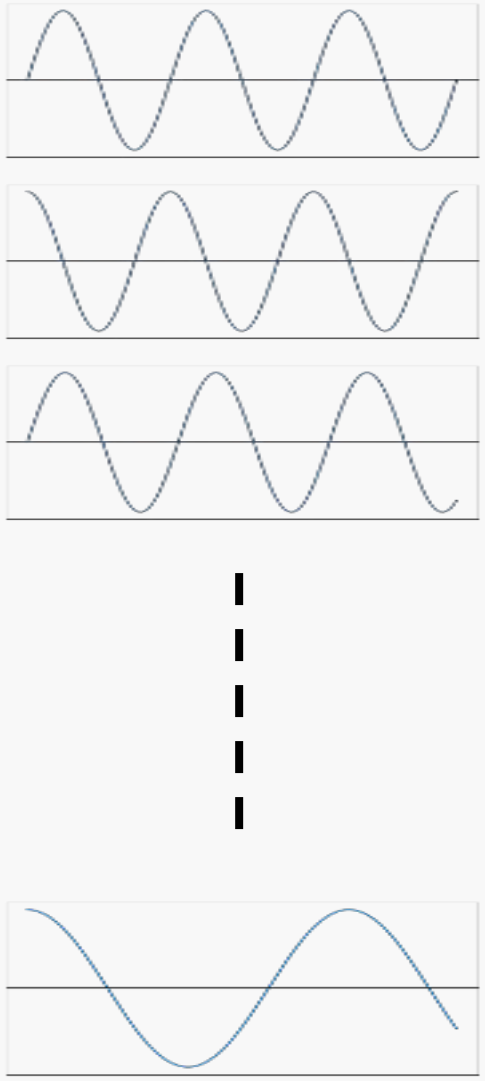
PE

$\sin(\omega_1 t)$

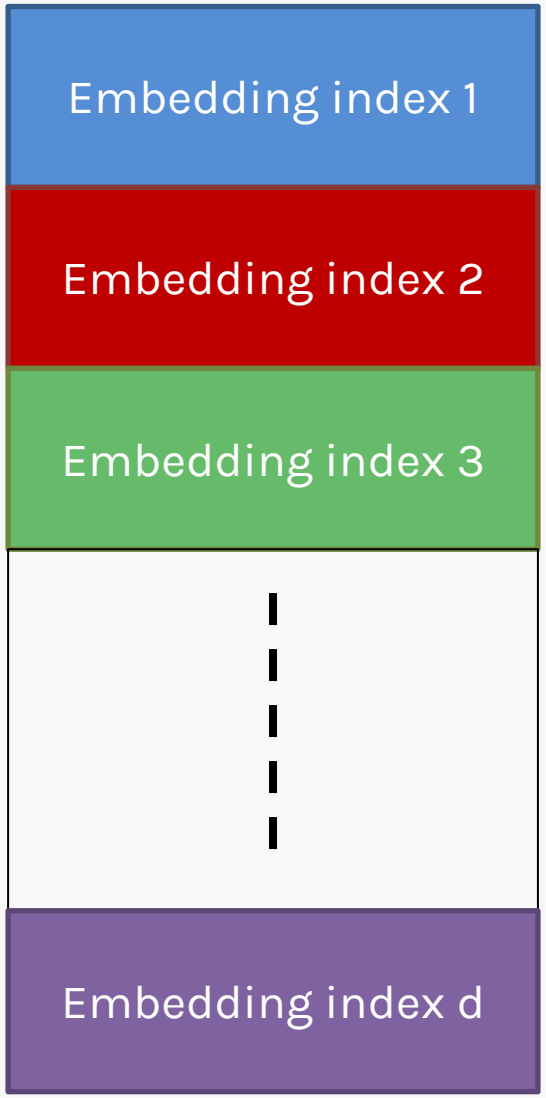
$\cos(\omega_1 t)$

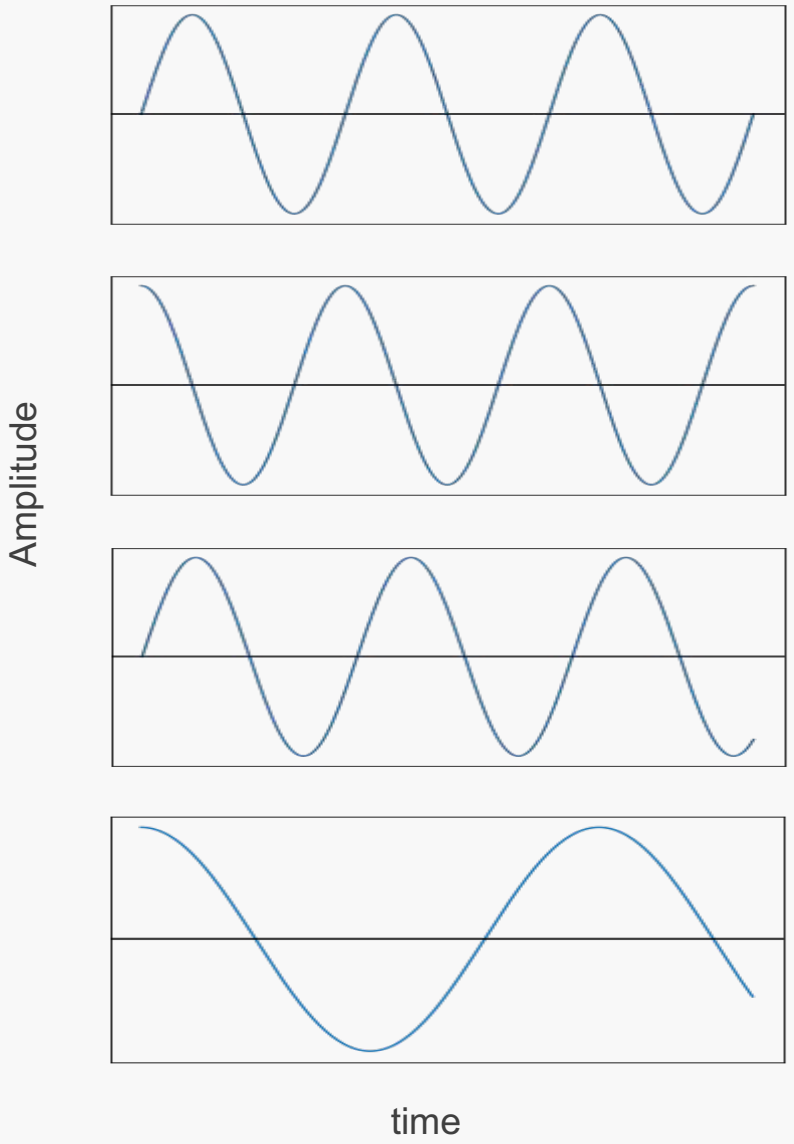
$\sin(\omega_2 t)$

$\cos(\omega_{d/2} t)$

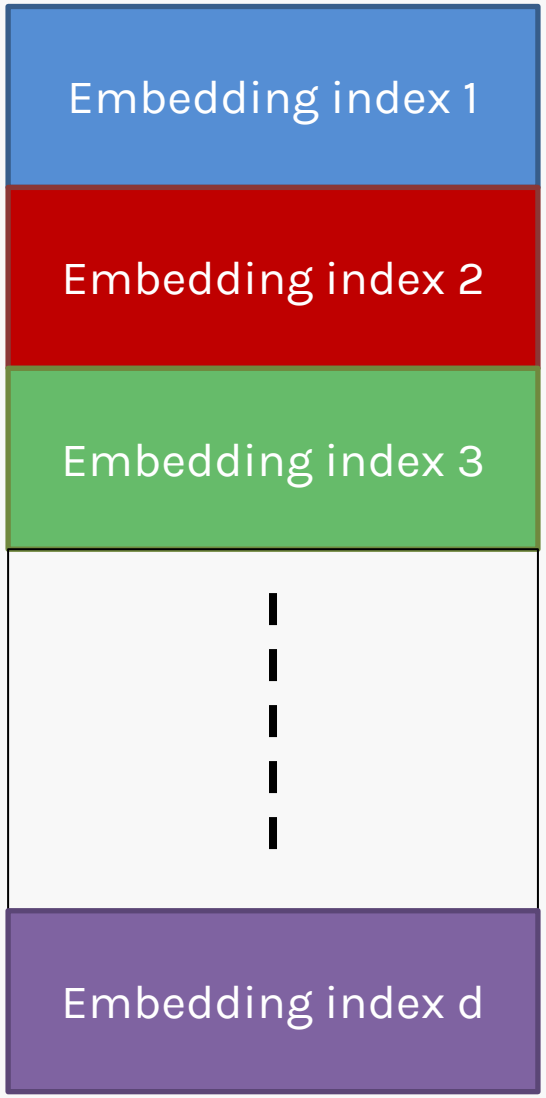


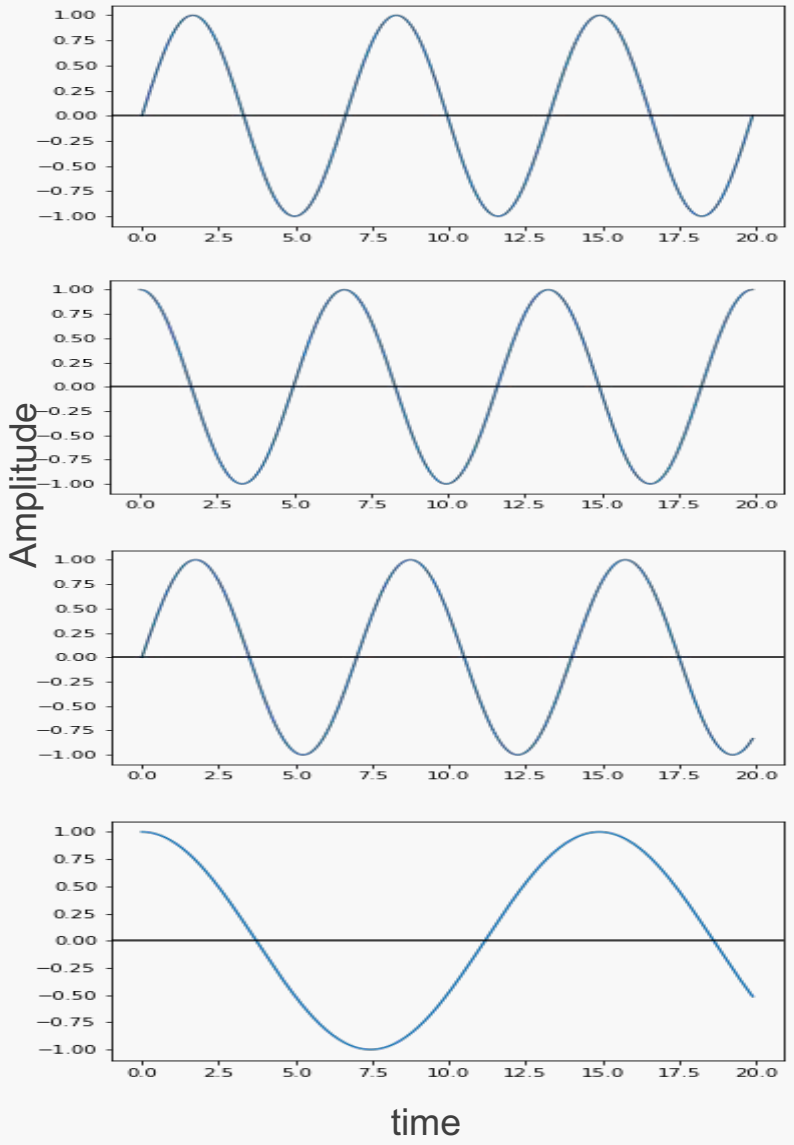
Positional Encoding



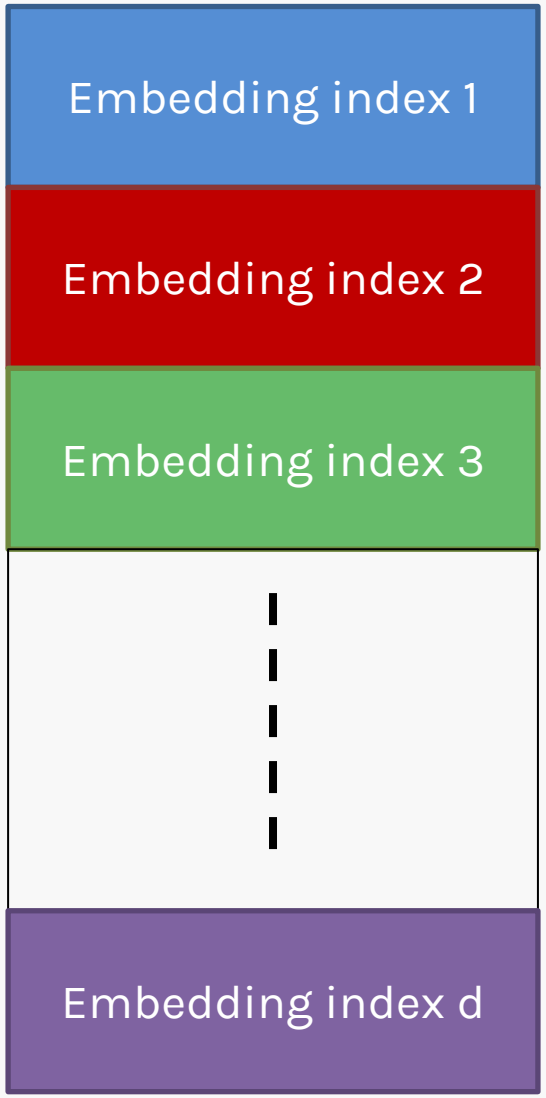


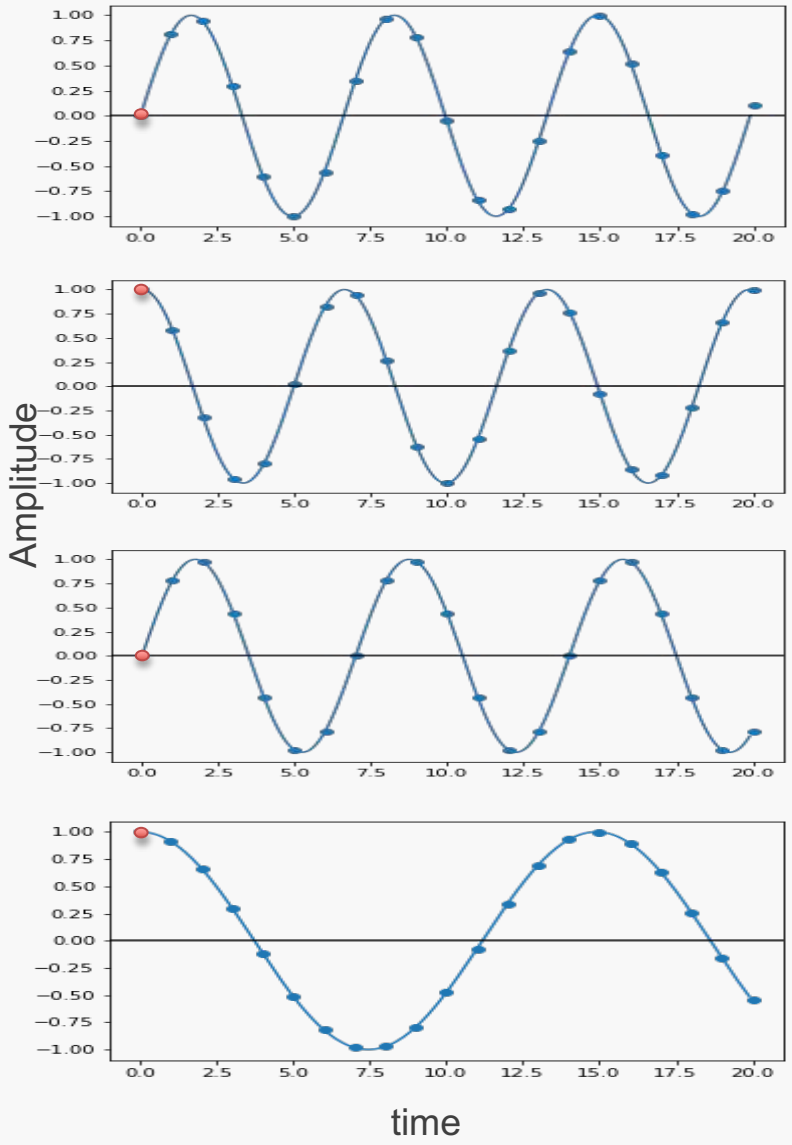
Positional Encoding





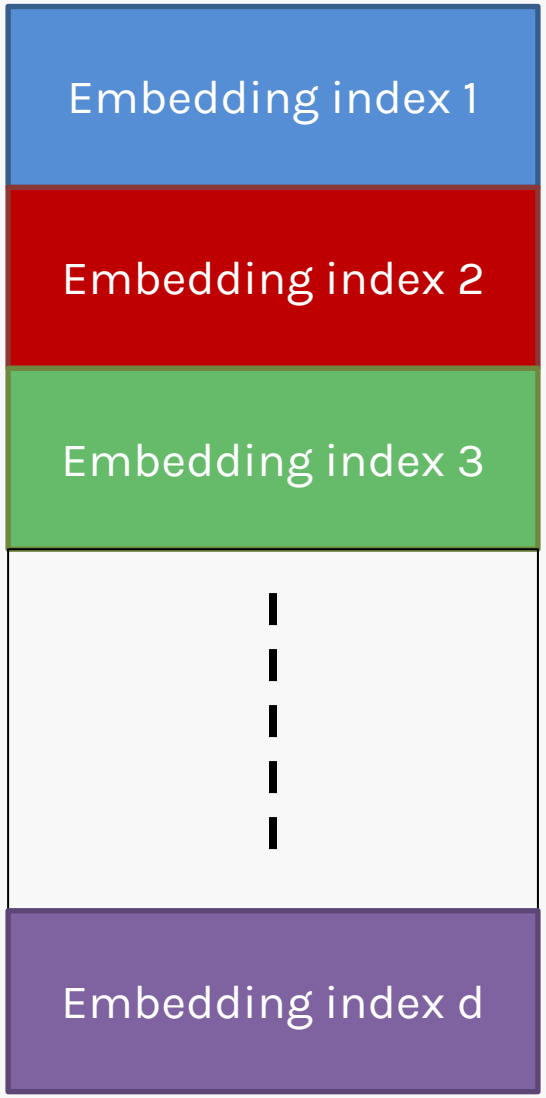
Positional Encoding

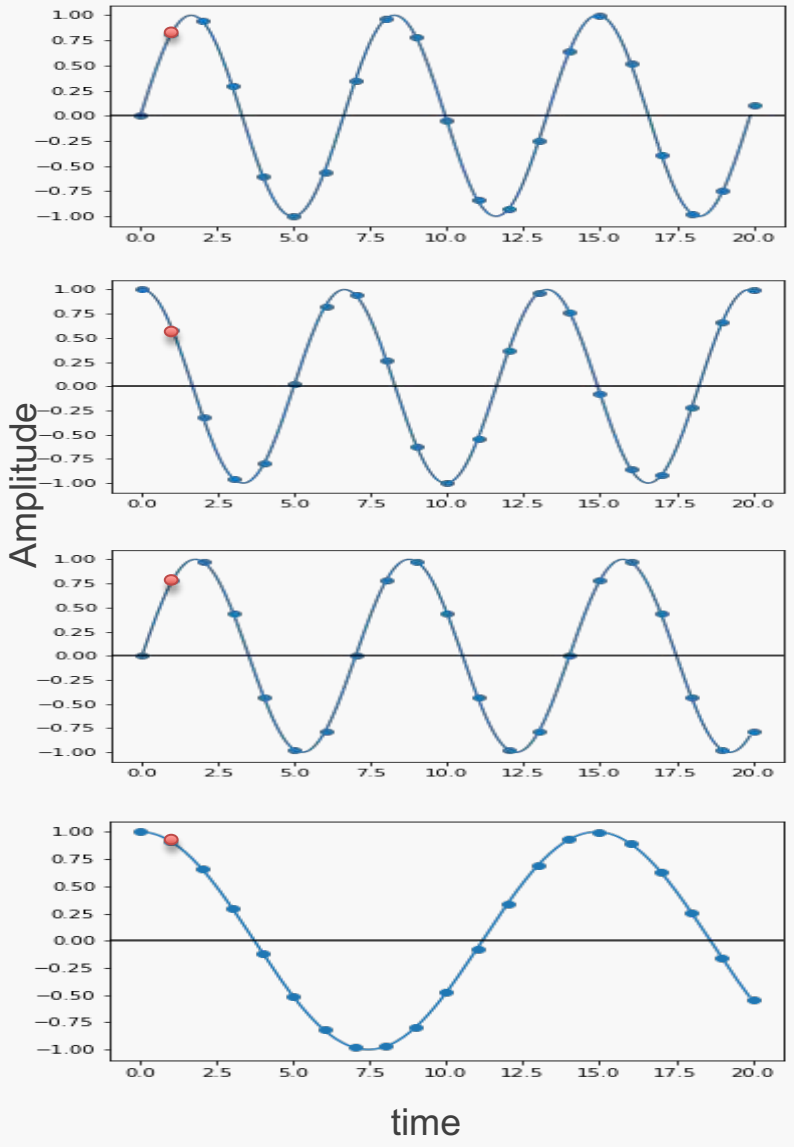




Now, If we want positional embeddings of the first word **(t=0)**

Positional Encoding



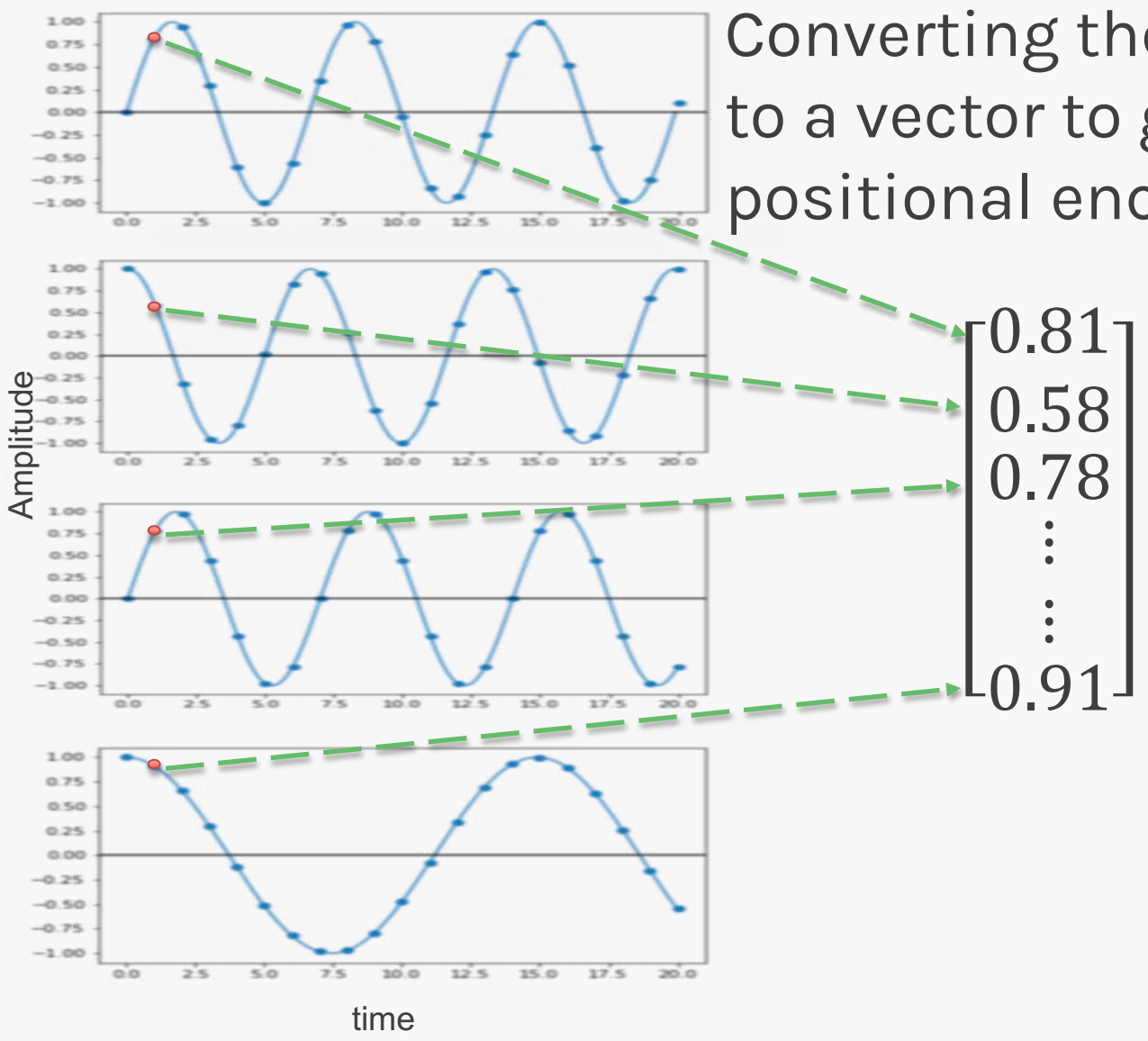


Now, If we want positional embeddings of the second word **(t=1)**

Positional Encoding

Let's suppose we have the word embedding for the second word:

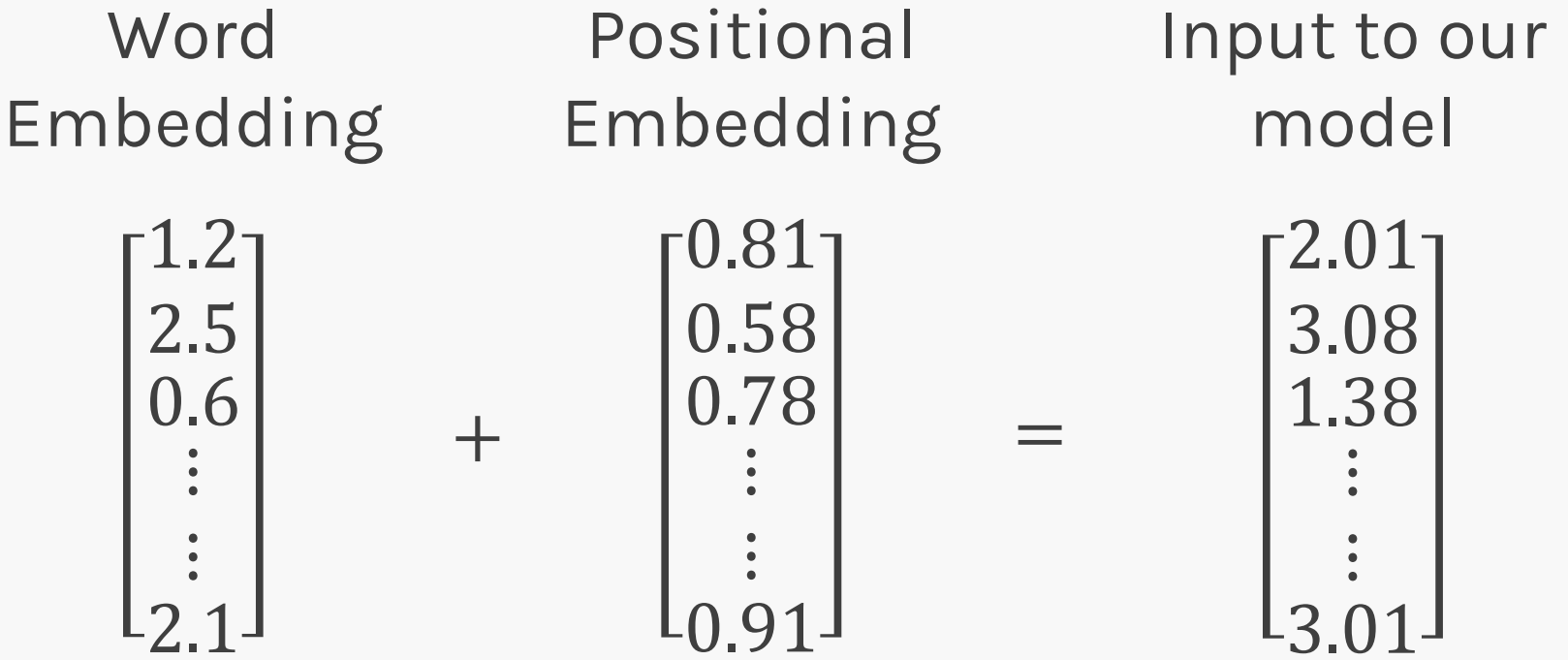
$$\begin{bmatrix} 1.2 \\ 2.5 \\ 0.6 \\ \vdots \\ \vdots \\ 2.1 \end{bmatrix}$$



Converting the points to a vector to get the positional encoding:

$$\begin{bmatrix} 0.81 \\ 0.58 \\ 0.78 \\ \vdots \\ \vdots \\ 0.91 \end{bmatrix}$$

Positional Encoding



Relative positioning

Another characteristic of sinusoidal positional encoding is that it allows the model to attend relative positions.

For every sine-cosine pair corresponding to frequency ω_k, there is a linear transformation $M \in \mathbb{R}^{2X2}$ (independent of t) where the following equation holds:

$$M \cdot \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t + \phi)) \\ \cos(\omega_k . (t + \phi)) \end{bmatrix}$$

Proof

Let M be a 2×2 matrix, we want to find u_1, v_1, u_2, v_2 so that:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \cdot \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t + \phi)) \\ \cos(\omega_k . (t + \phi)) \end{bmatrix}$$

By applying the addition theorem, we can expand the right hand side as follows:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} \cdot \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . t)\ \cos(\omega_k . \phi) + \cos(\omega_k . t)\ \sin(\omega_k . \phi) \\ \cos(\omega_k . t)\ \cos(\omega_k . \phi) - \sin(\omega_k . t)\ \sin(\omega_k . \phi) \end{bmatrix}$$

Proof

This results in following two equations:

$$u_1 \sin(\omega_k . t) + v_1 \cos(\omega_k . t) = \cos(\omega_k . \phi) \sin(\omega_k . t) + \sin(\omega_k . \phi) \cos(\omega_k . t)$$
$$u_2 \sin(\omega_k . t) + v_2 \cos(\omega_k . t) = -\sin(\omega_k . \phi) \sin(\omega_k . t) + \cos(\omega_k . \phi) \cos(\omega_k . t)$$

By solving above equations, we get:

$$u_1 = \cos(\omega_k . \phi) \quad v_1 = \sin(\omega_k . \phi)$$
$$u_2 = -\sin(\omega_k . \phi) \quad v_2 = \cos(\omega_k . \phi)$$

Proof

So the final transformation matrix M is:

$$M_{\phi,k} = \begin{bmatrix} \cos(\omega_k.\phi) & \sin(\omega_k.\phi) \\ -\sin(\omega_k.\phi) & \cos(\omega_k.\phi) \end{bmatrix}$$

As we can see, the final transformation does not depend on t. Note that one can find the matrix M very similar to the rotation matrix.

Relative Positional Embedding

- Even though in theory, it is possible to encode relative positions using the current formula, it doesn't work very well when it comes to implementation.

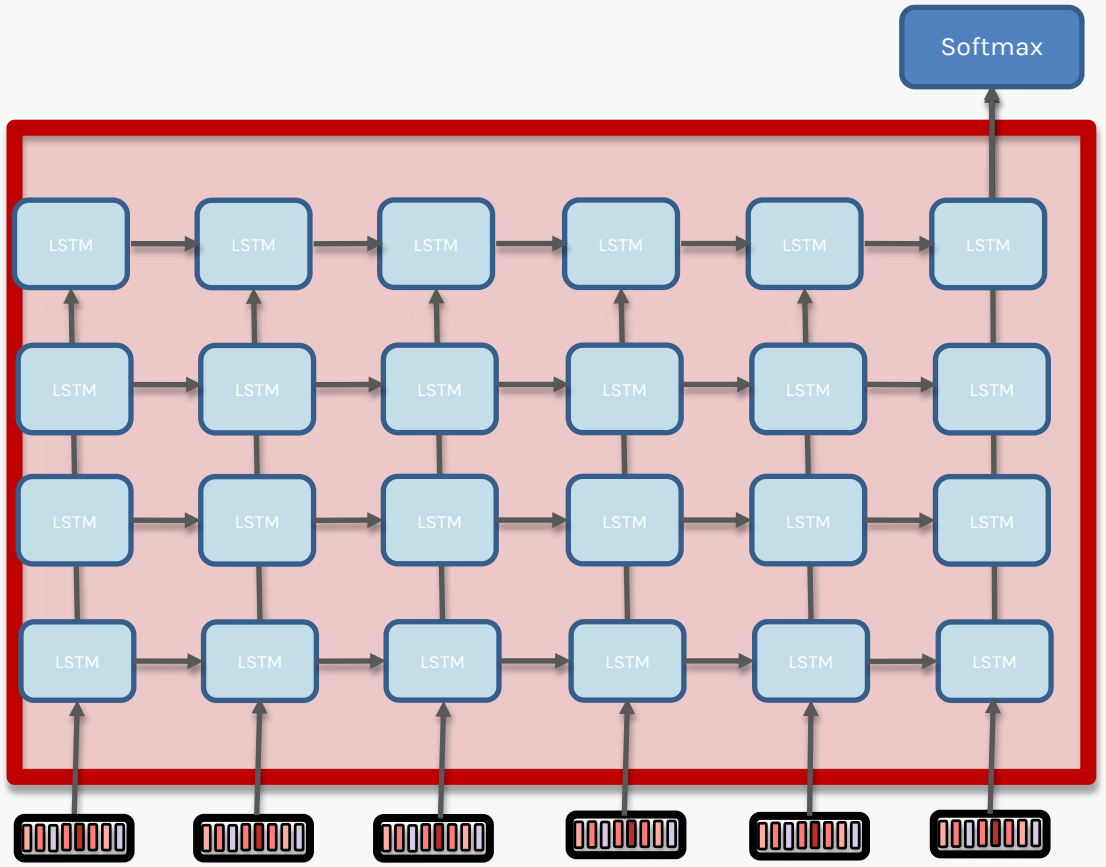
- To overcome this problem, a subsection of the authors of the transformers paper came up with an alternative way to do this in the paper "**Self-Attention with Relative Position Representations**".

https://arxiv.org/abs/1803.02155

Bringing it all together

Comparison – RNN v/s Transformer

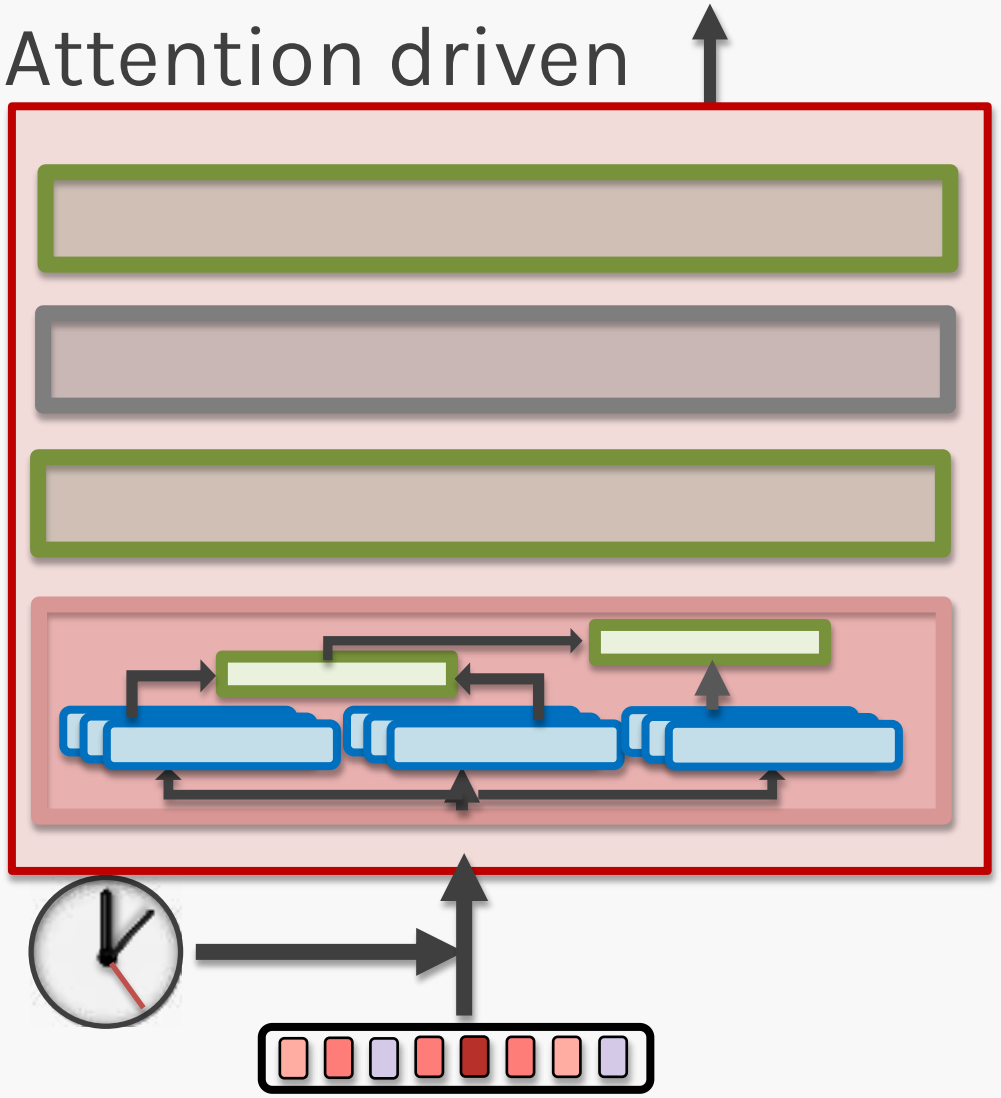
BEFORE



Recurrent units

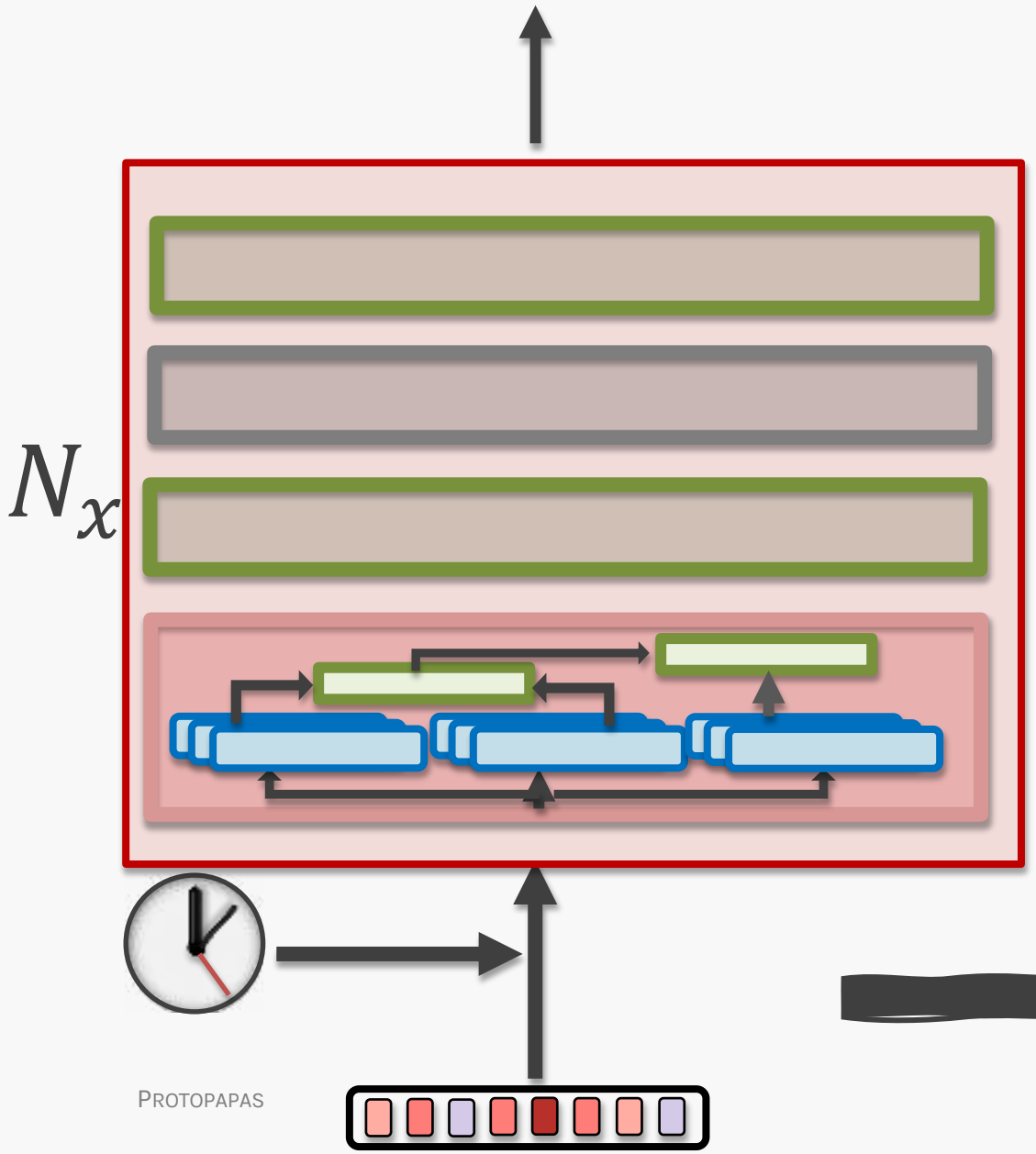
AFTER

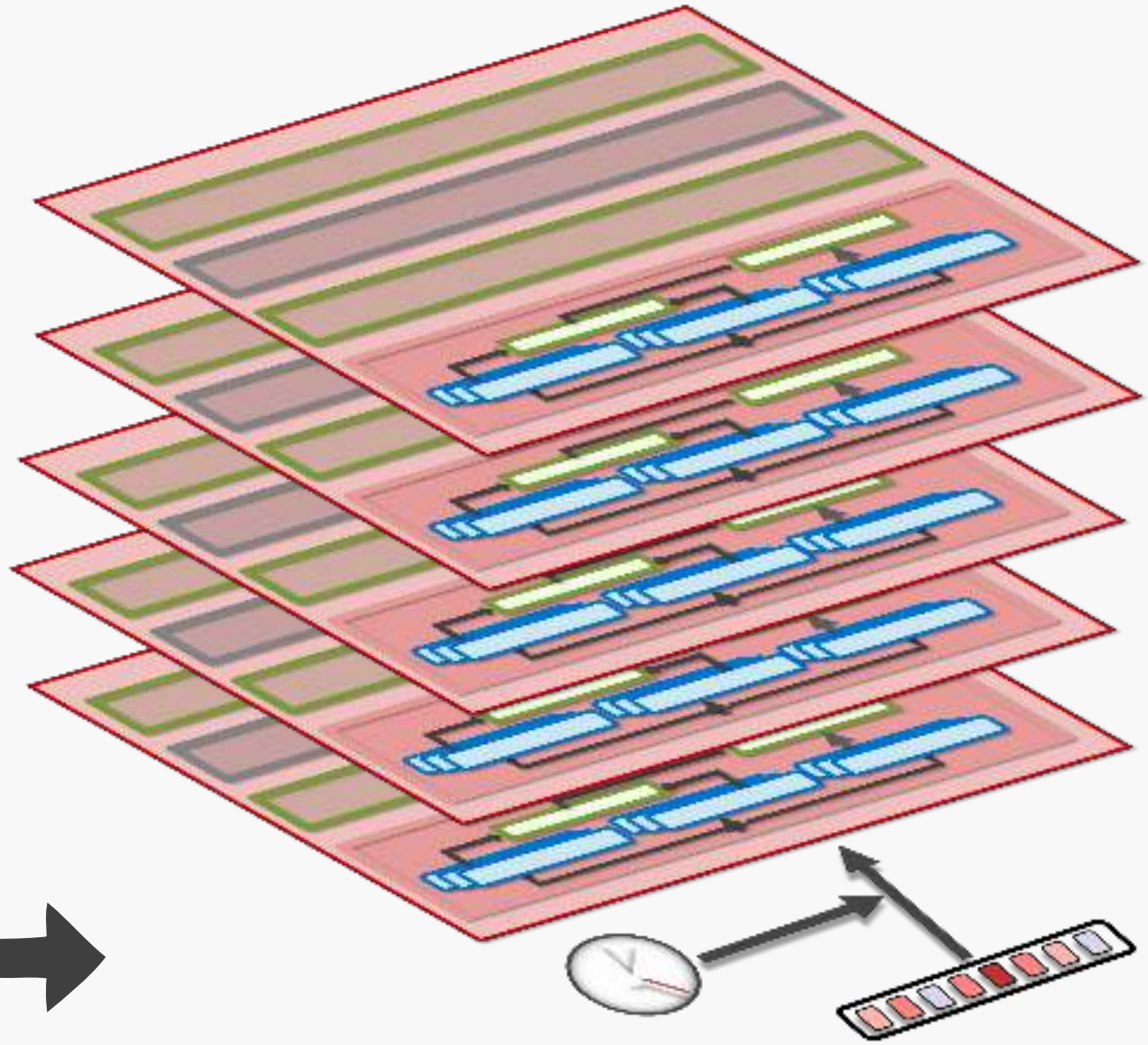
Attention driven



N_x

Transformer as a 3-D model

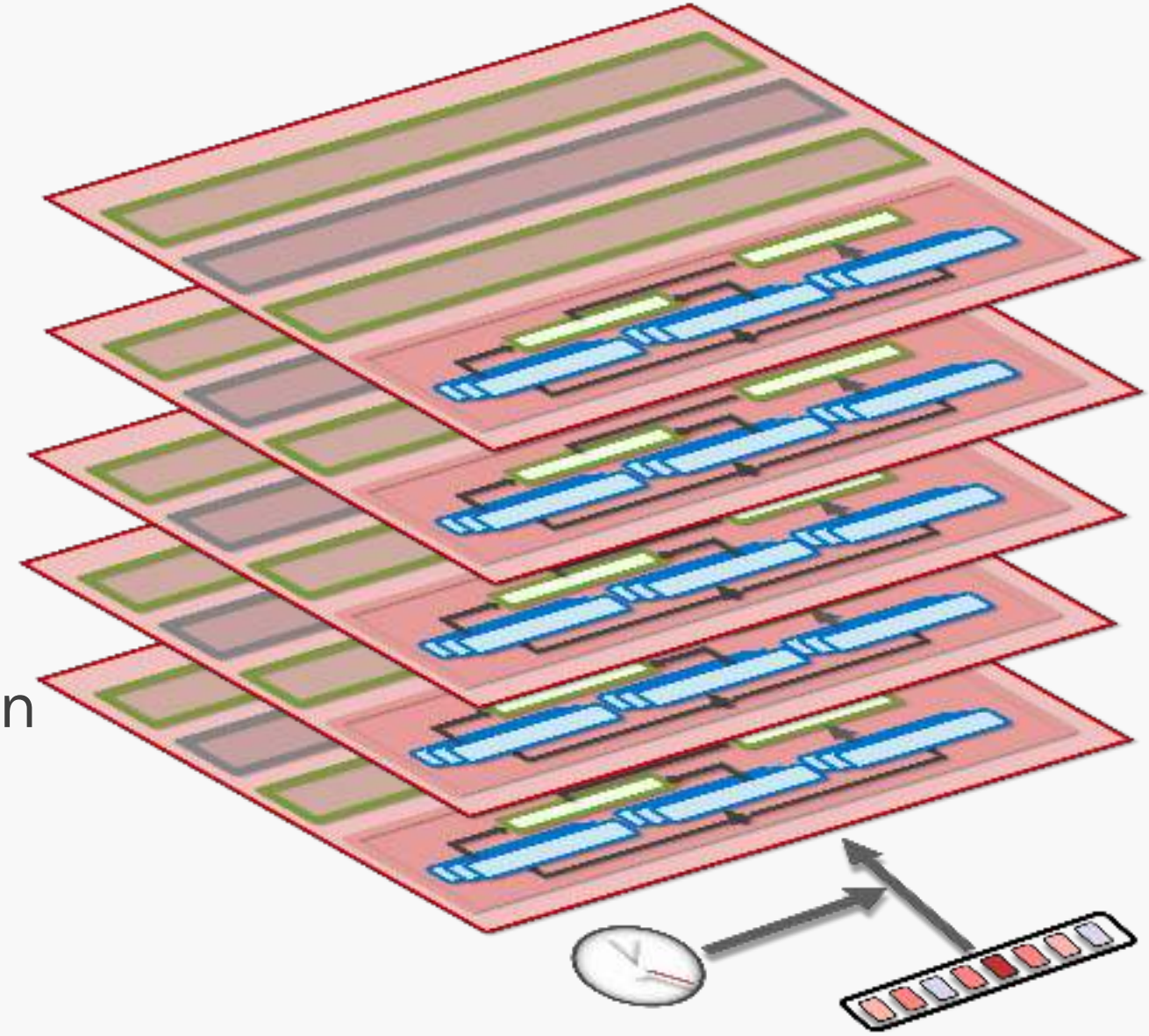




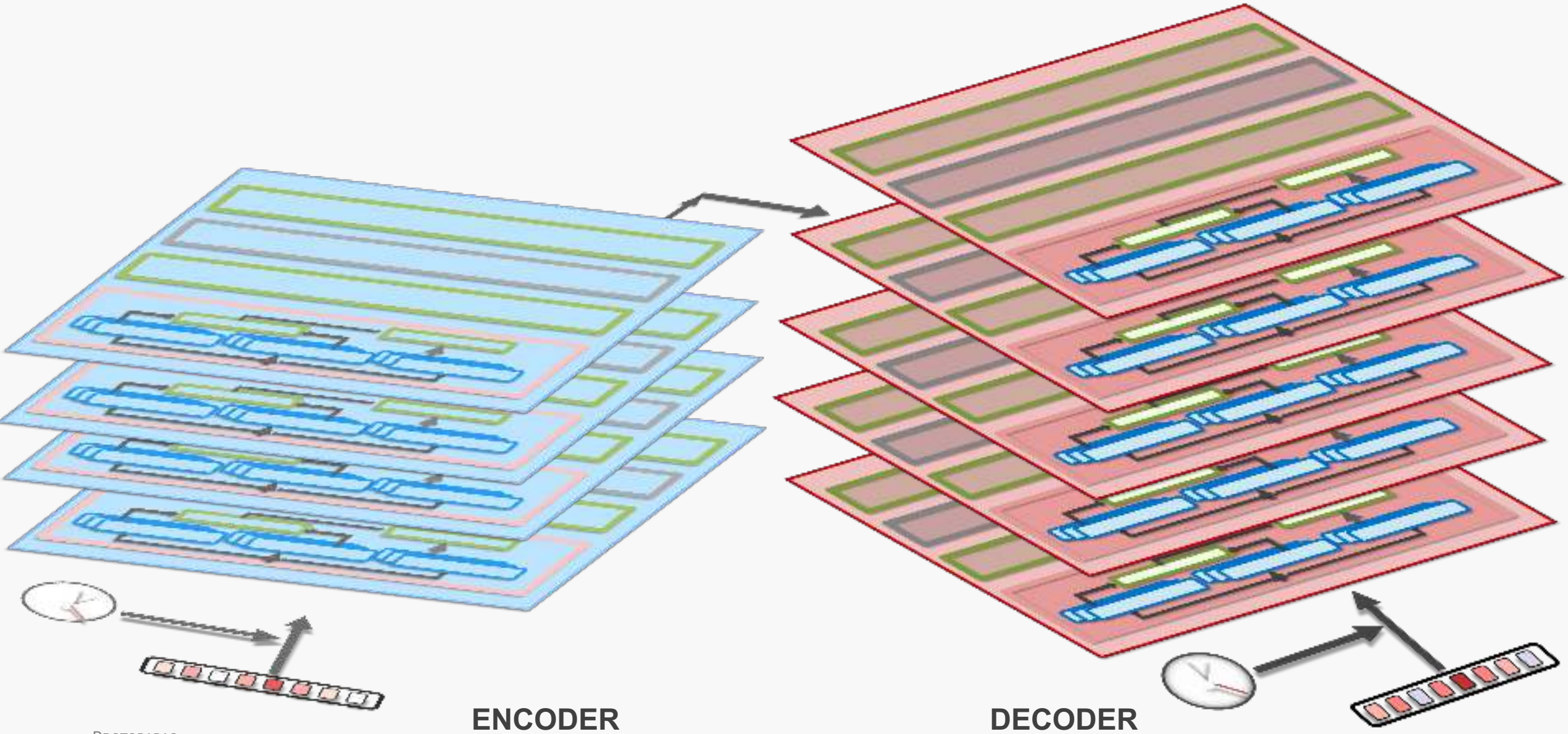
Transformers - Summary

Language Model Wishlist?

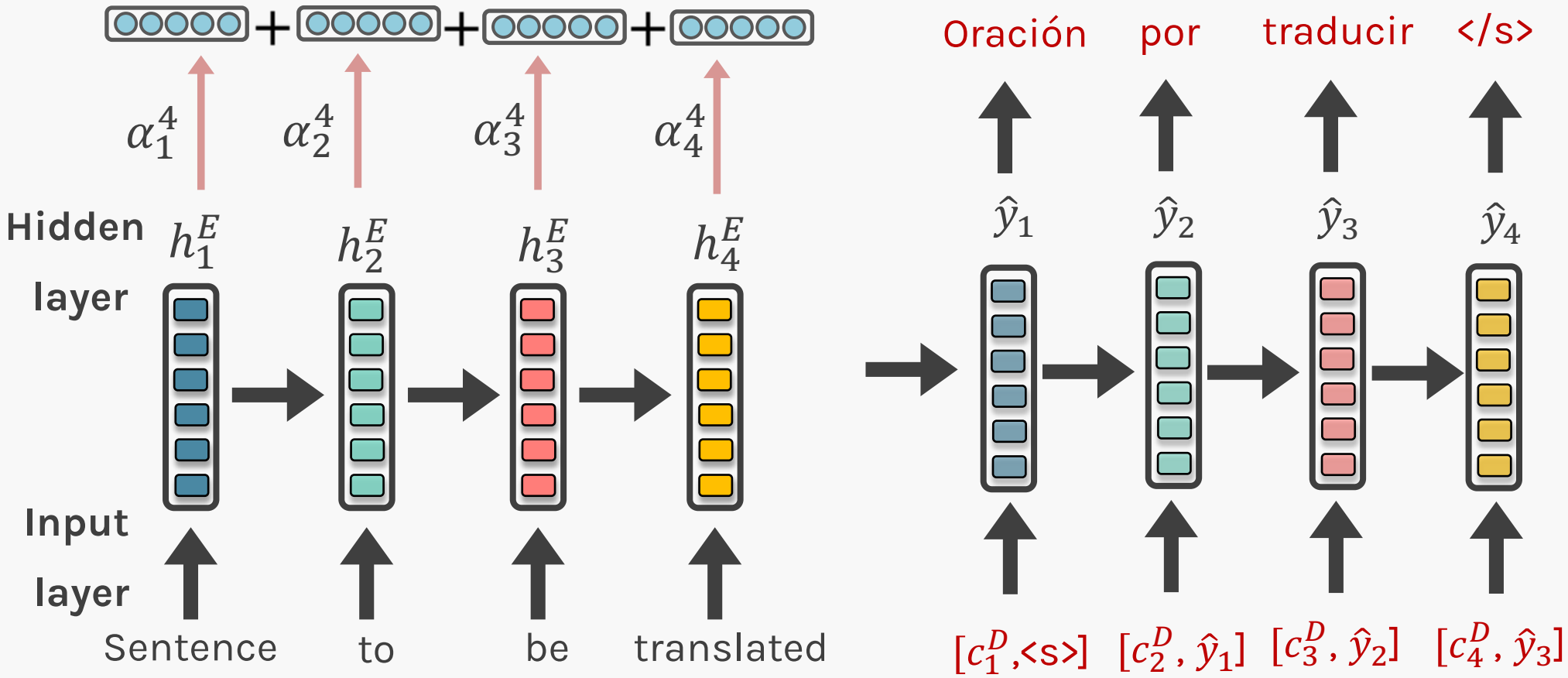
- ☑ We want to have strong contextual relations between words - **DONE**
- ☑ We want words to have sequential information - **DONE**
- ☑ We need an architecture that can be trained in parallel (non-Markovian property) - **DONE**



Transformers - Summary



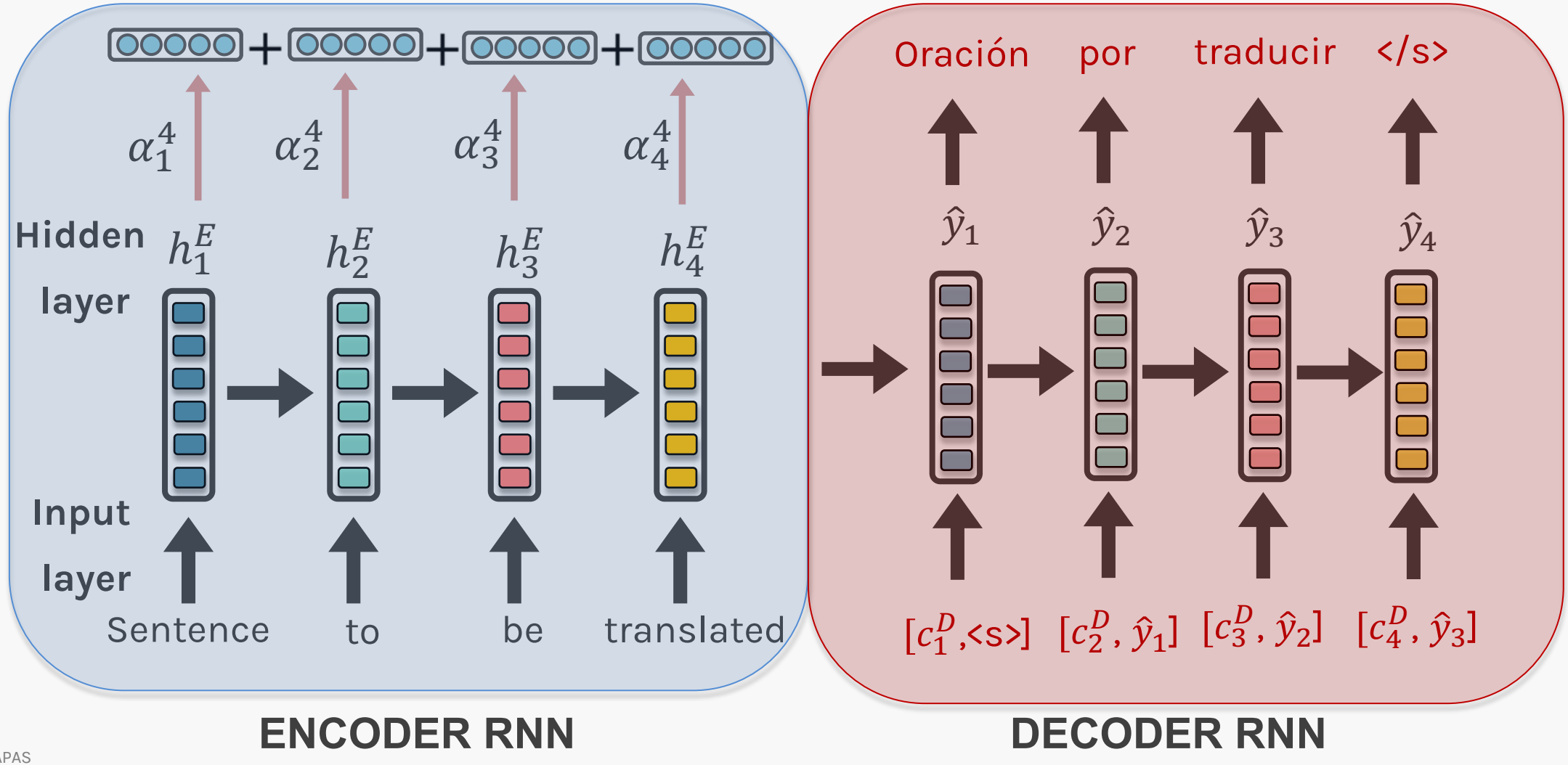
RECAP: Seq2Seq + Attention



ENCODER RNN

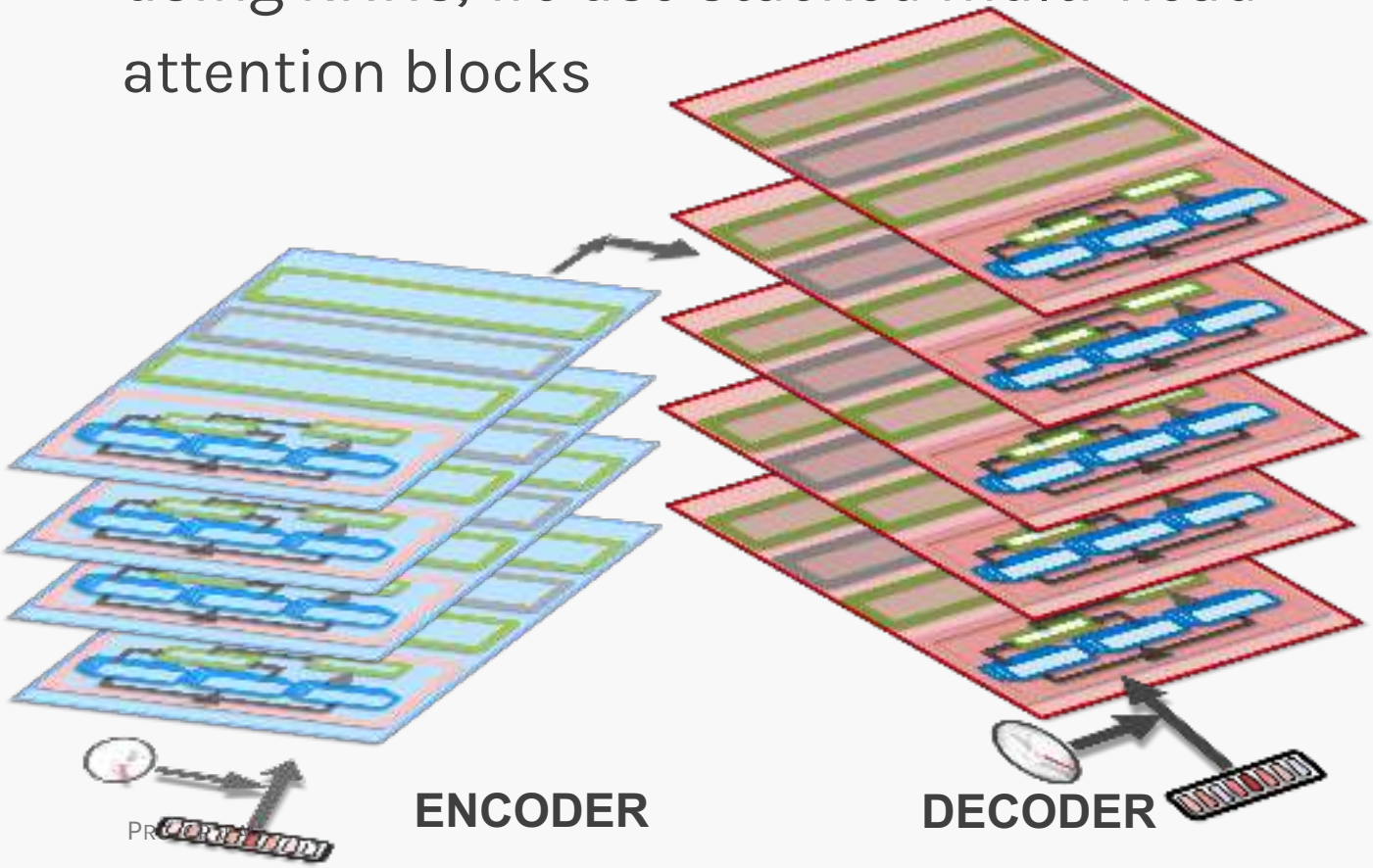
DECODER RNN

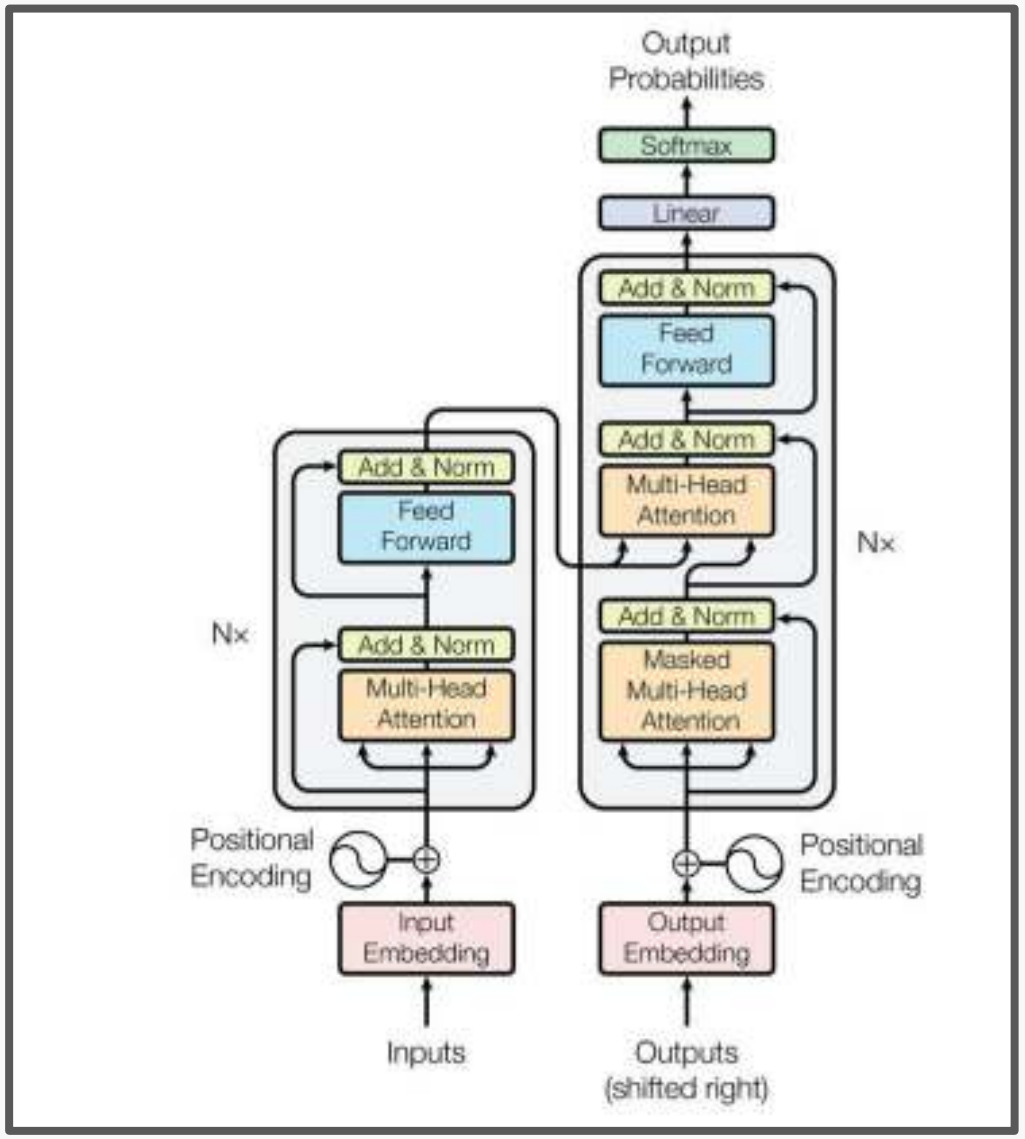
RECAP: Seq2Seq + Attention



Transformers - Summary

Transformers consist of an Encoder-Decoder architecture, but instead of using RNNs, we use stacked multi-head attention blocks

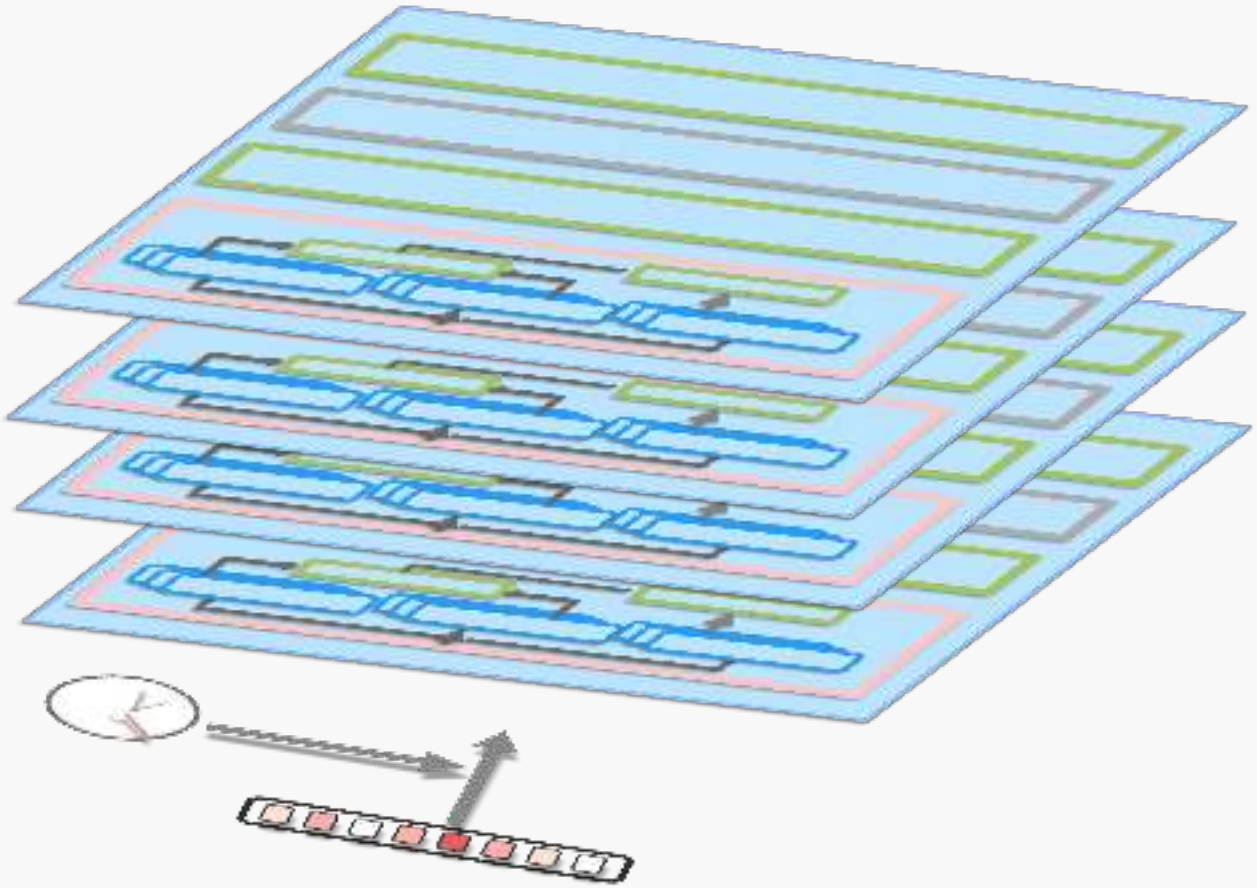




From Transformers to **BERT**

- Instead of an **Encoder-Decoder** architecture for machine translation, what if we just use the encoder for Language Model and other NLP tasks.

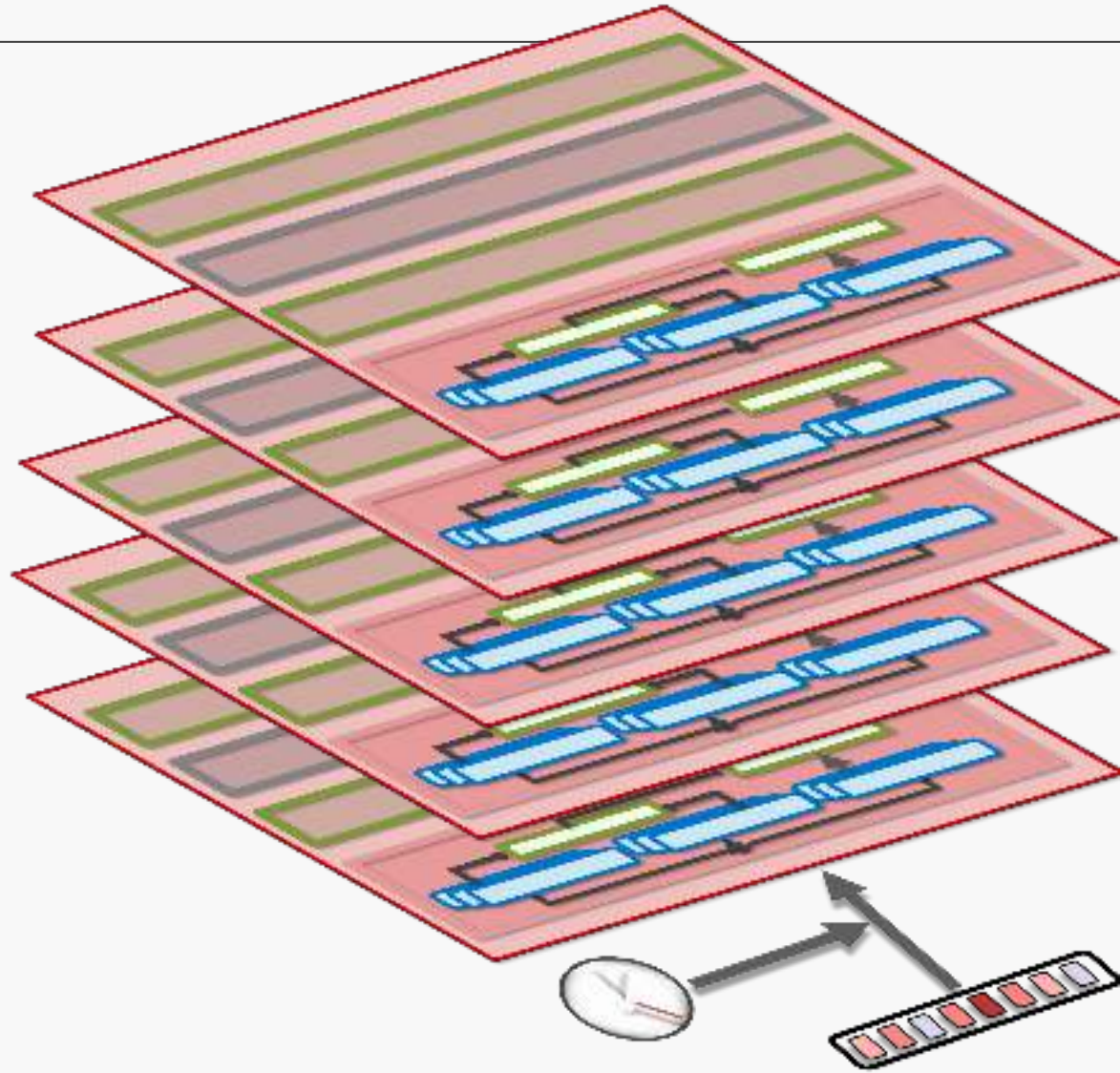
- This led to the new architecture called Bidirectional Encoder Representations from Transformers, or more commonly known as BERT.



ENCODER

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



BERT vs GPT2

BERT

- Masked Language Model
- Bidirectional language model
- Made up of only the Encoder with stacked transformer blocks.
- Vocabulary size = 30,000
- Good for a language model

GPT2

- Auto-regressive Language Model
- Unidirectional language model
- Made up of only the Decoder with stacked transformer blocks.
- Vocabulary size = 50,000
- Good for generating text