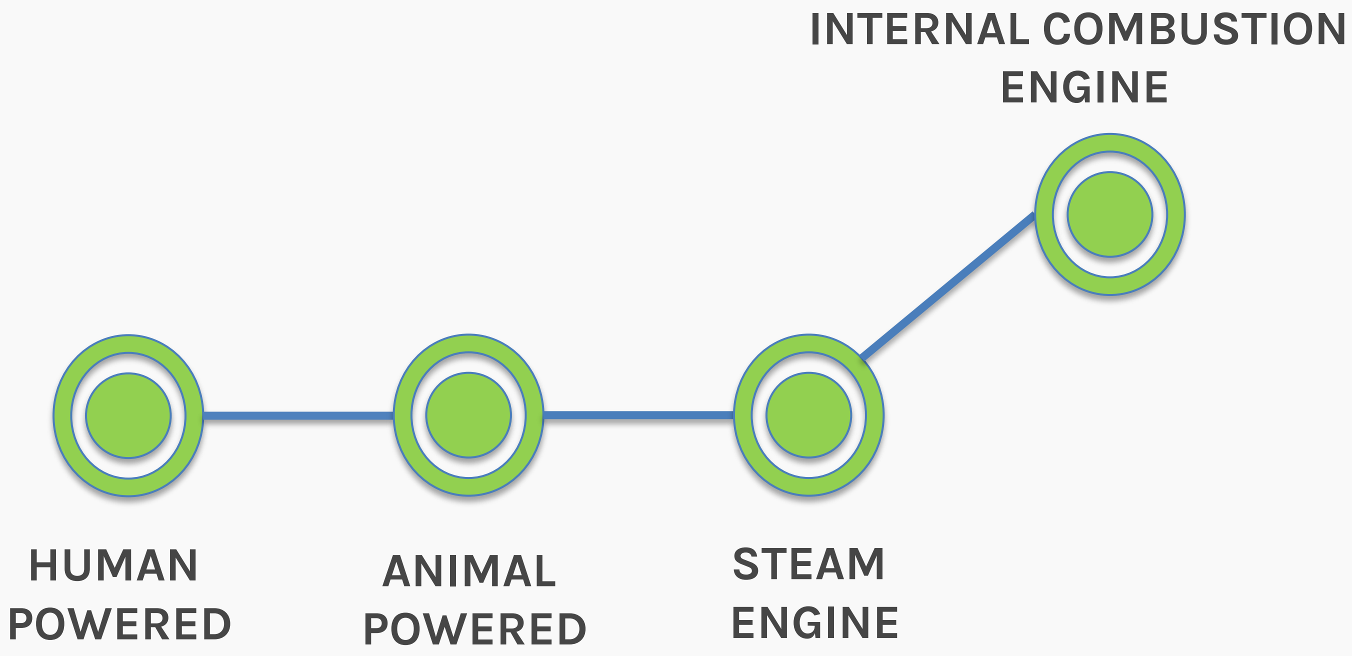
Self-Attention, Transformers, BERT

Pavlos Protopapas

Outline

- Motivation for Attention
  - Recap Seq2Seq
  - 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



A brief history of engines

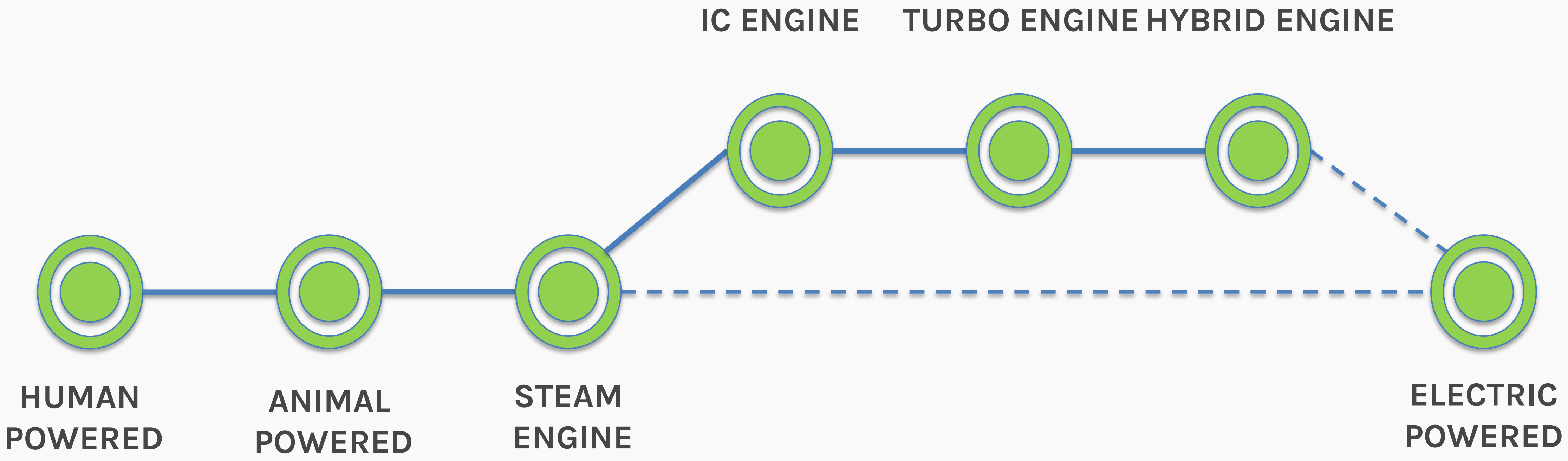
FLYWHEEL

IC COMBUSTION ENGINE

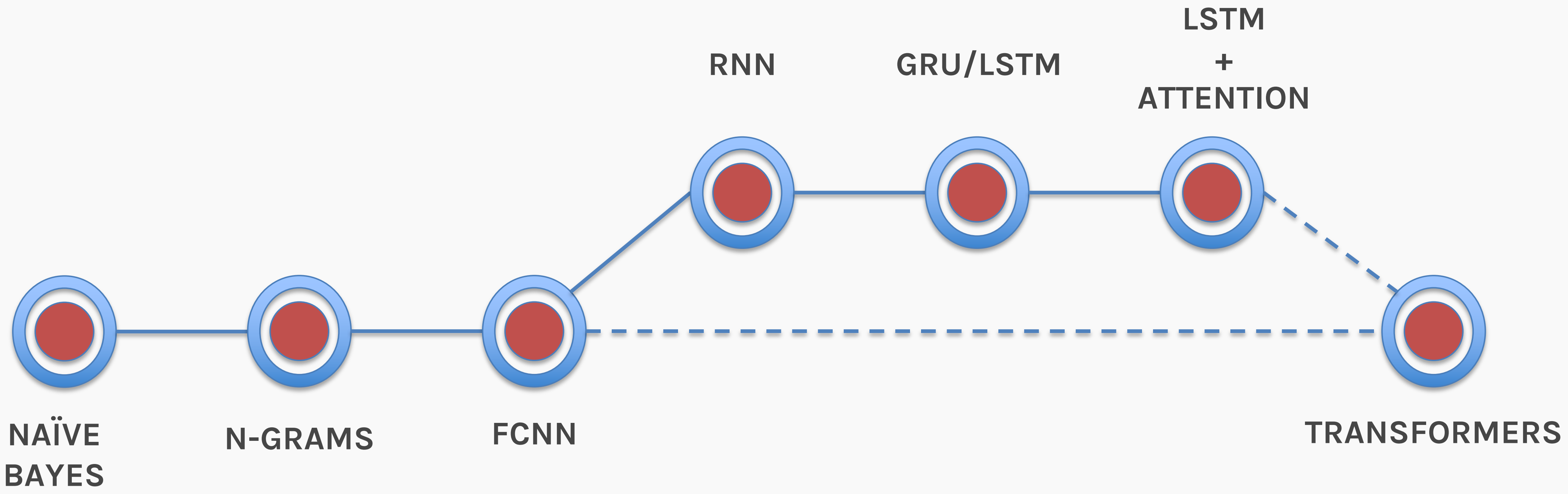
DC MOTOR

ALTERNATOR

Electricity is all you need



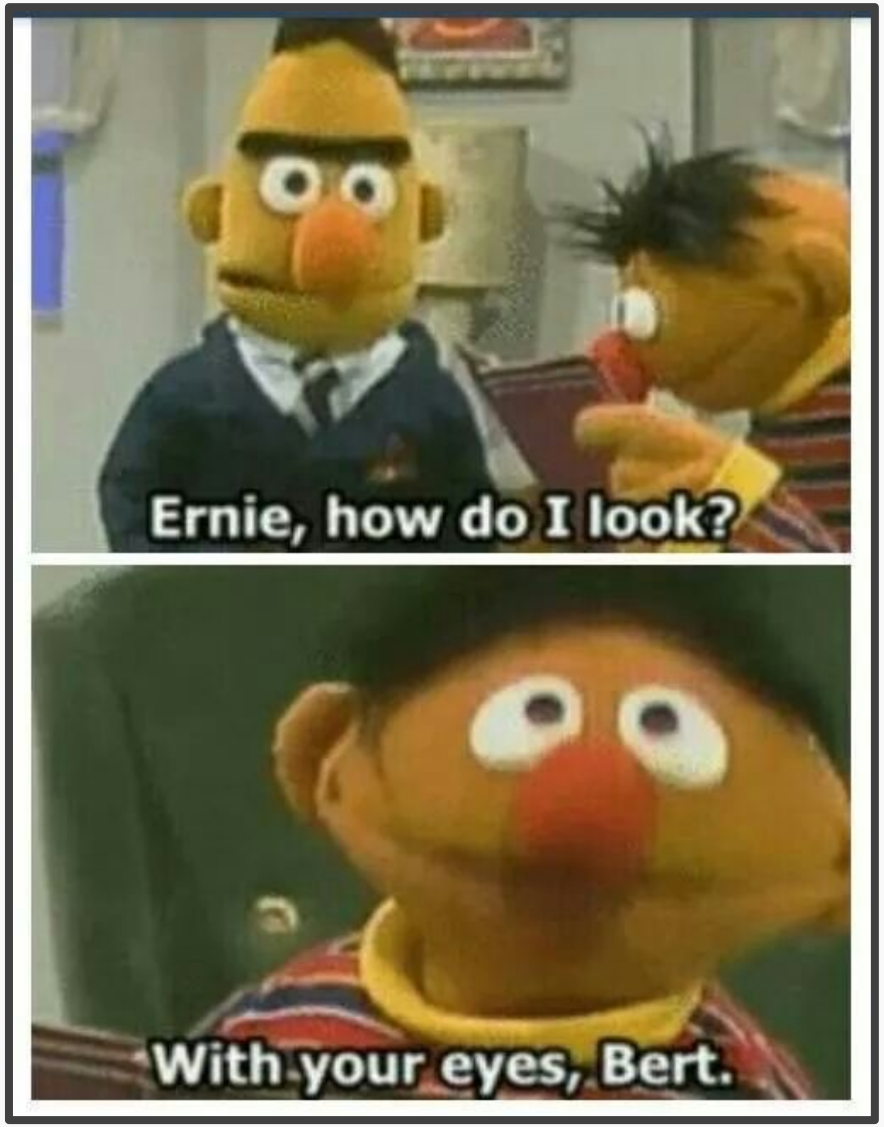
Language modeling



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

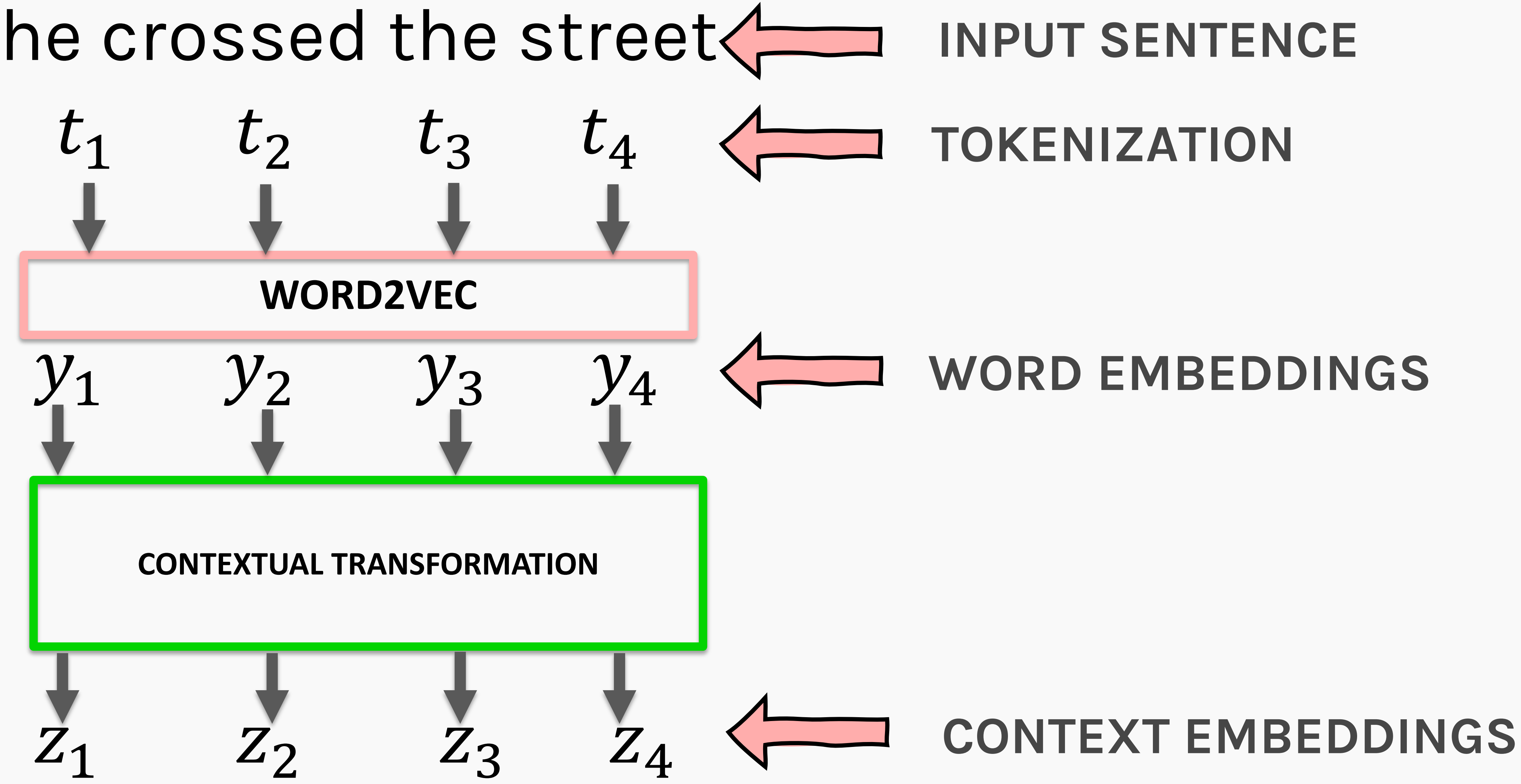
Attention

Attention: Example sentence

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

Shivas was hit by a bus because $\boxed{\text{he}}$ crossed the street

Attention – Where to add weights?



💡 **IDEA #1:** Positional relationship

Shivas was hit by a bus because $\boxed{\text{he}}$ crossed the street

Attention: Example sentence

IDEA #1: Positional transformation

Shivas 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

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

$$y = [y_1, y_2, y_3, \ldots, y_n]$$

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

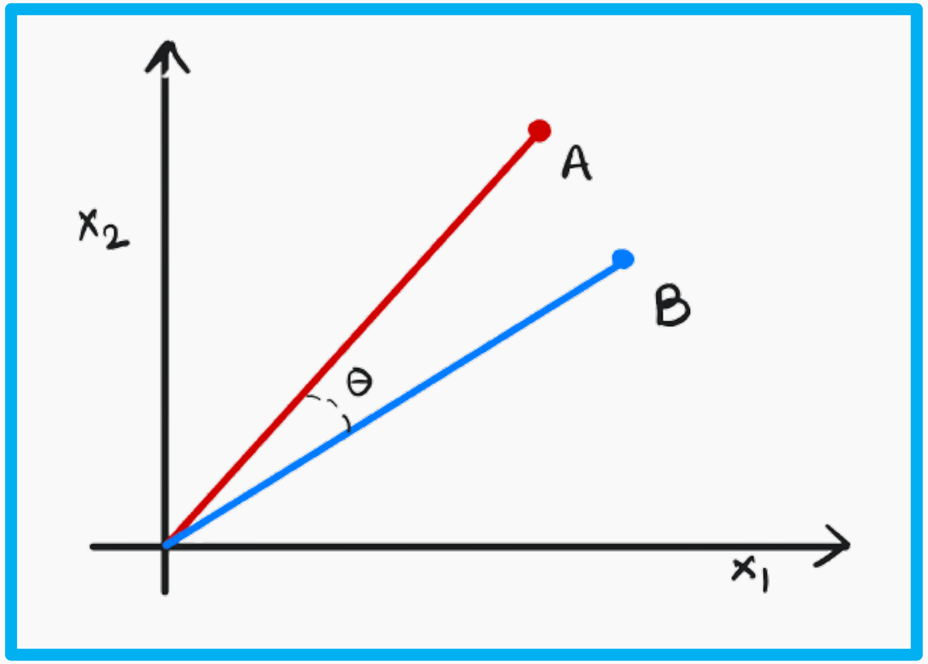
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots a_{ij} & \cdots & a_{in} \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}$$

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

Attention – Text Data

But how can we do it?

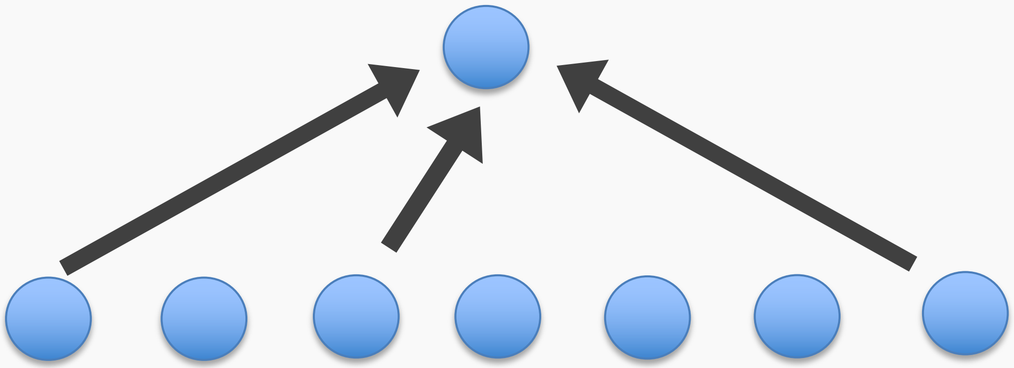
Yes! Let's use cosine similarity between words

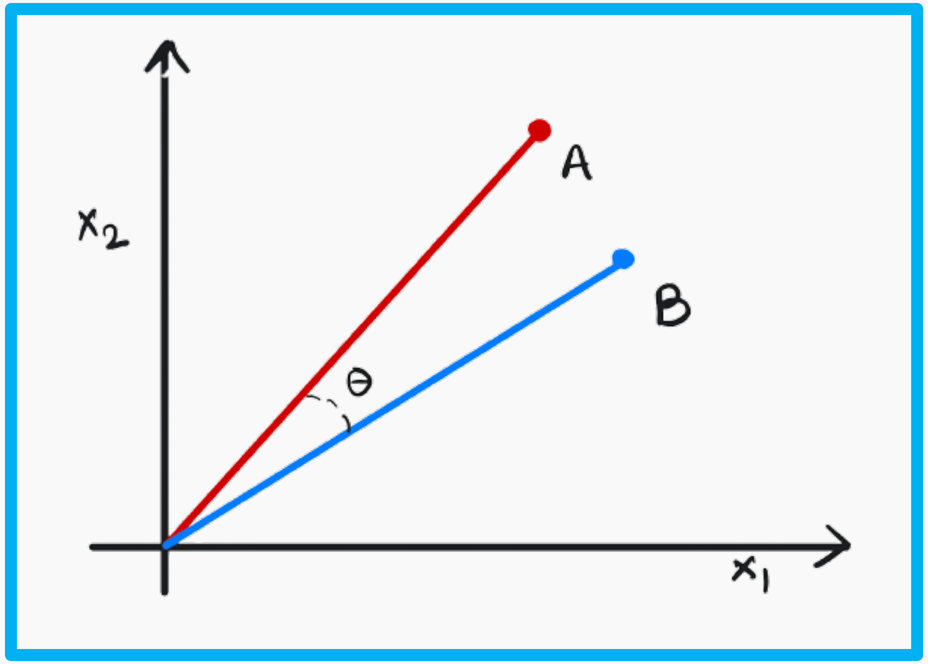


$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

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

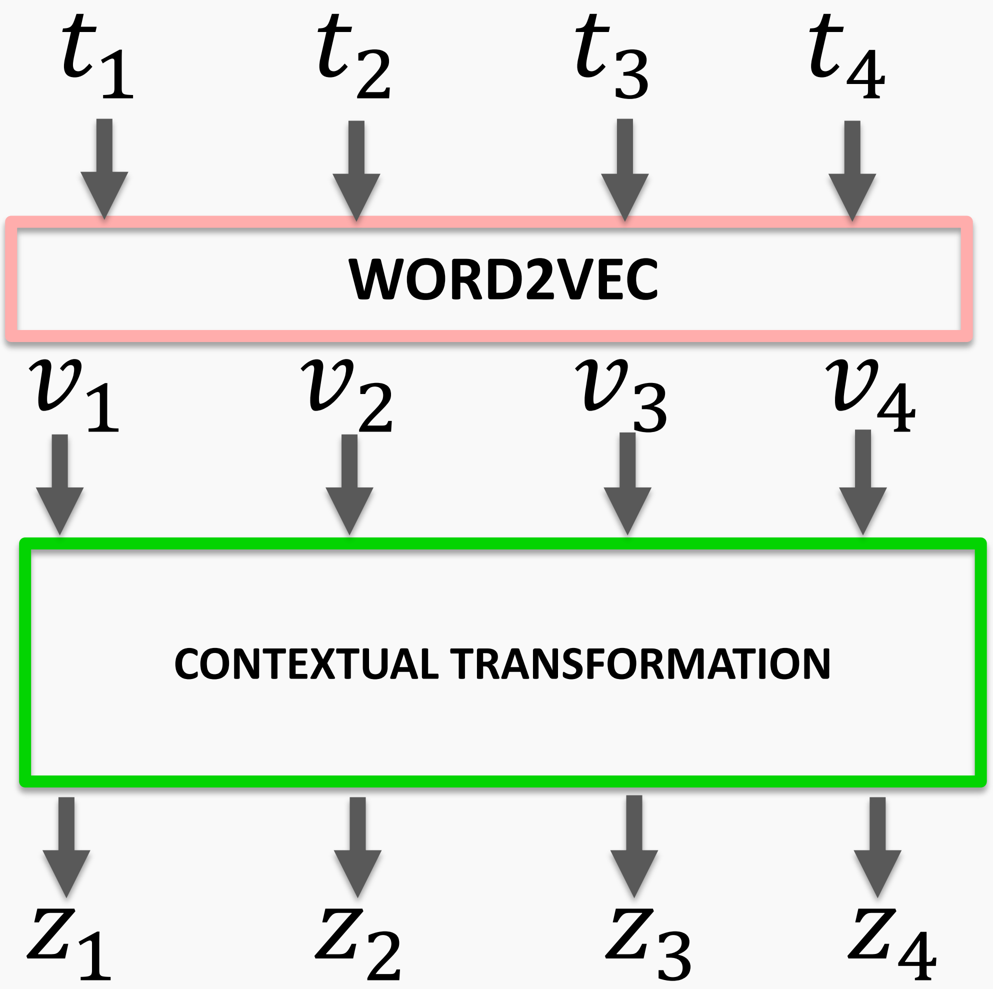
Ignacio was standing next to the bank of the river

Attention: Example sentence #2

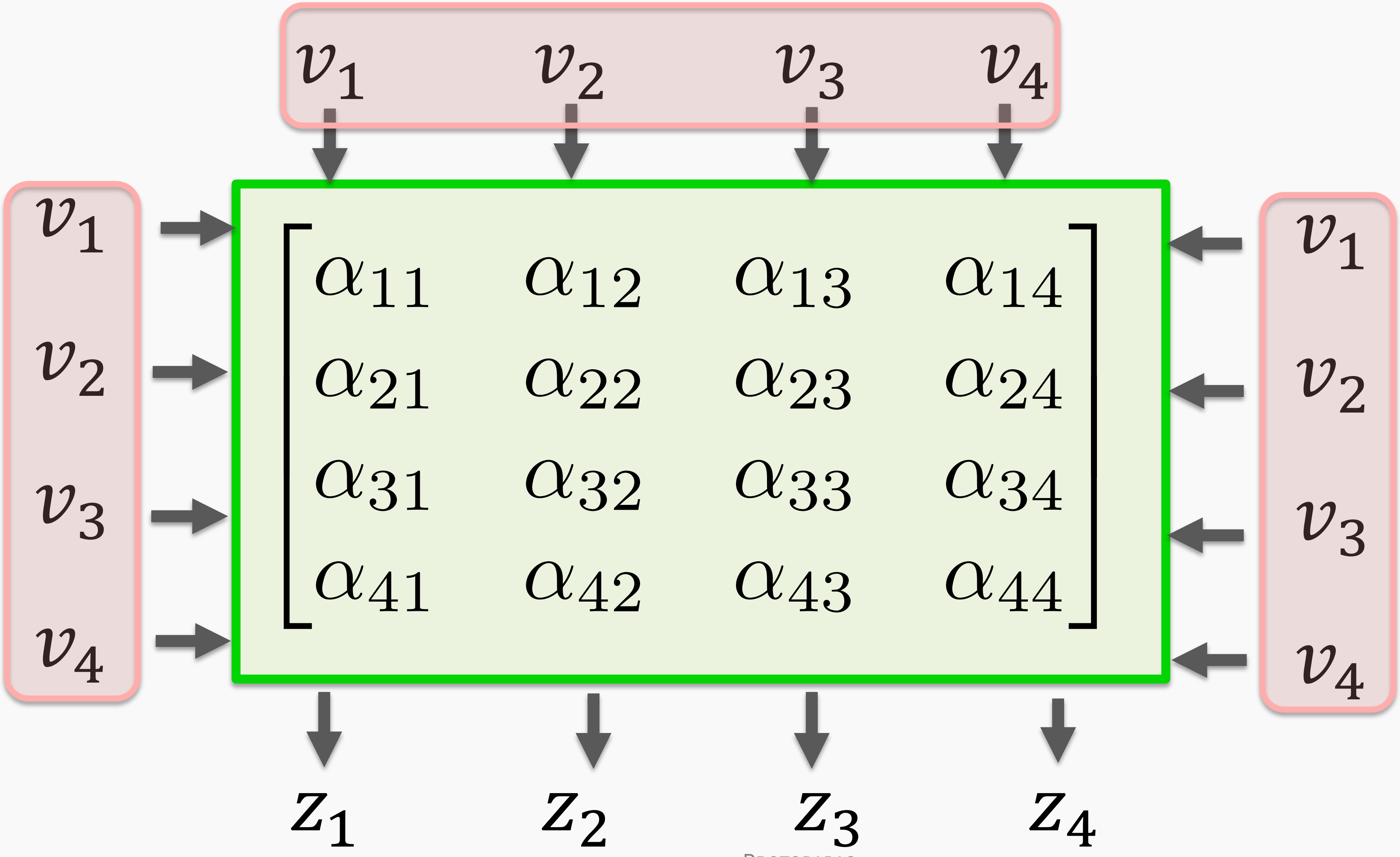
bank of the river

Attention – Cosine Similarity

bank of the river



Attention – Cosine Similarity



Attention: Example sentence #2

$v_1 \circ v_1 = a_{11}$

$v_1 \circ v_2 = a_{12}$

$v_1 \circ v_3 = a_{13}$

$v_1 \circ v_4 = a_{14}$

$\xrightarrow{\text{WEIGHTS NORMALIZATION}}$

α_{11}

α_{12}

α_{13}

α_{14}

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

Attention: Example sentence #2

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

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

New contextual
word embedding

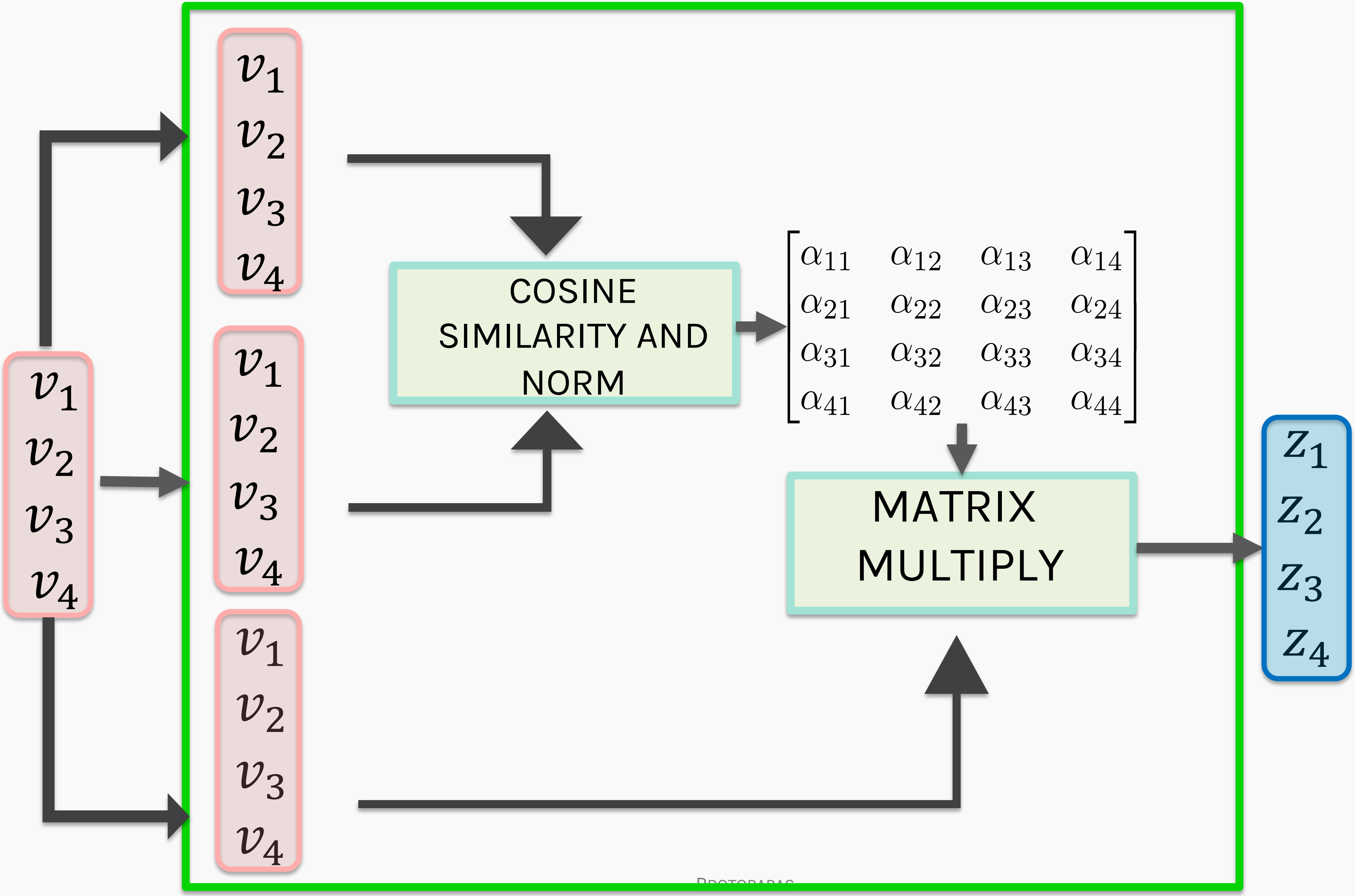
Similarly…

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

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

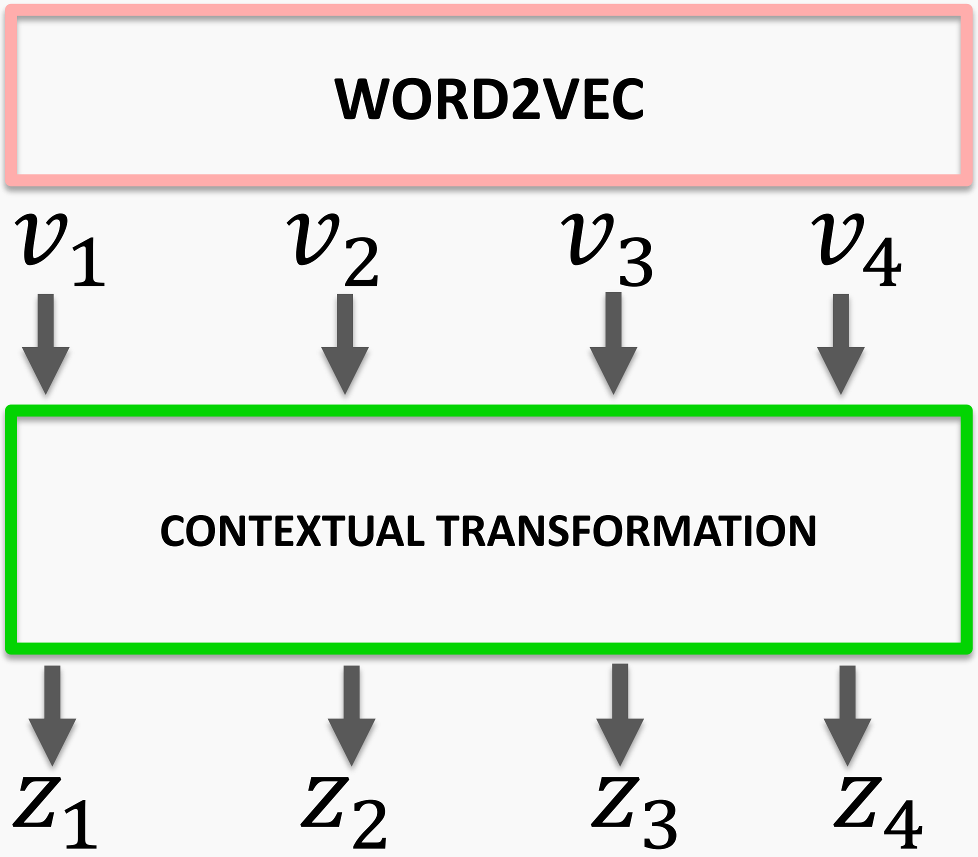
$$z_3 = v_1\alpha_{31} + v_2\alpha_{32} + v_3\alpha_{33} + v_4\alpha_{34}$$

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


v_1 v_2 v_3 v_4
COSINE SIMILARITY AND NORM
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4

Attention: Example sentence #2

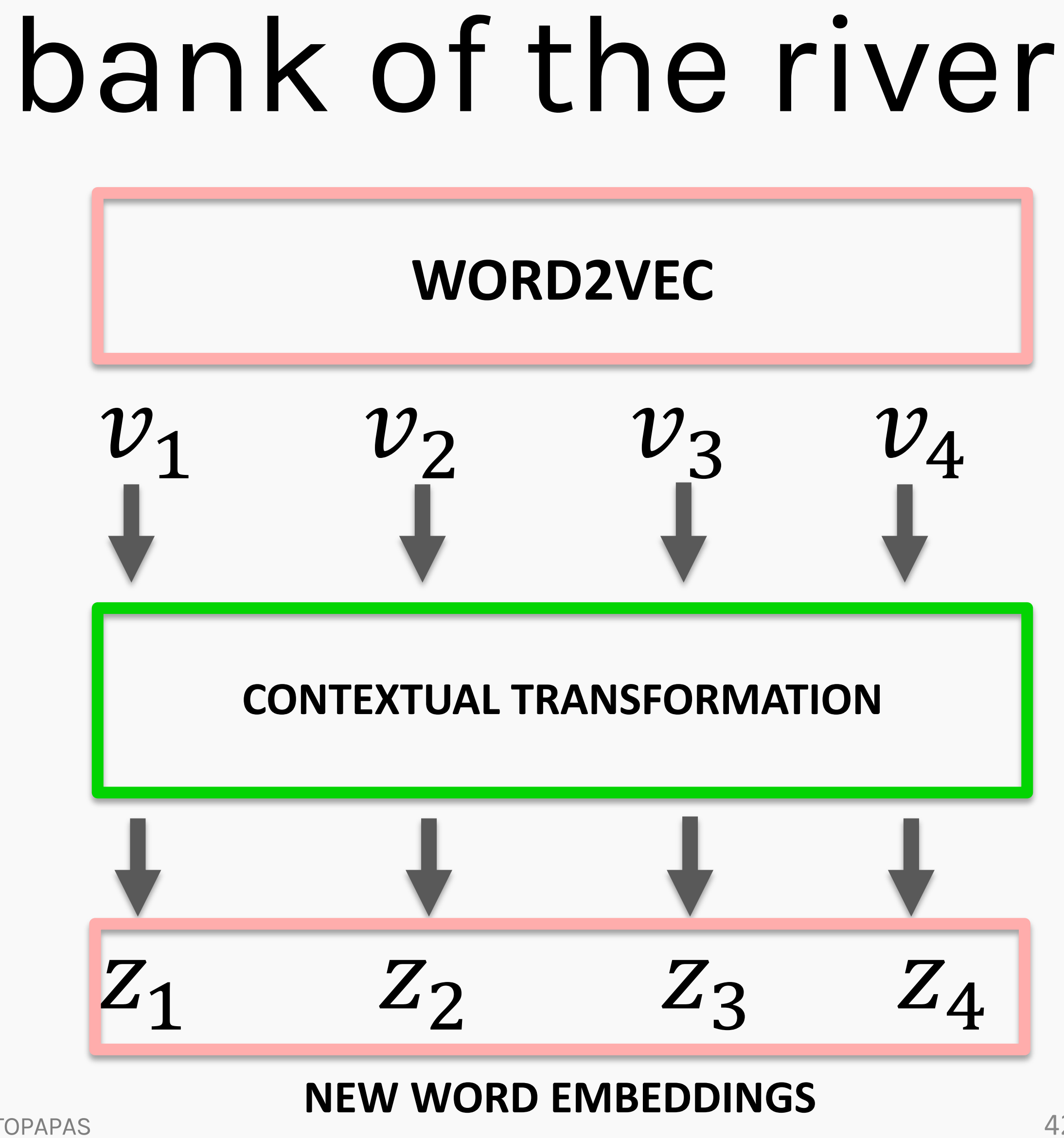
bank of the river



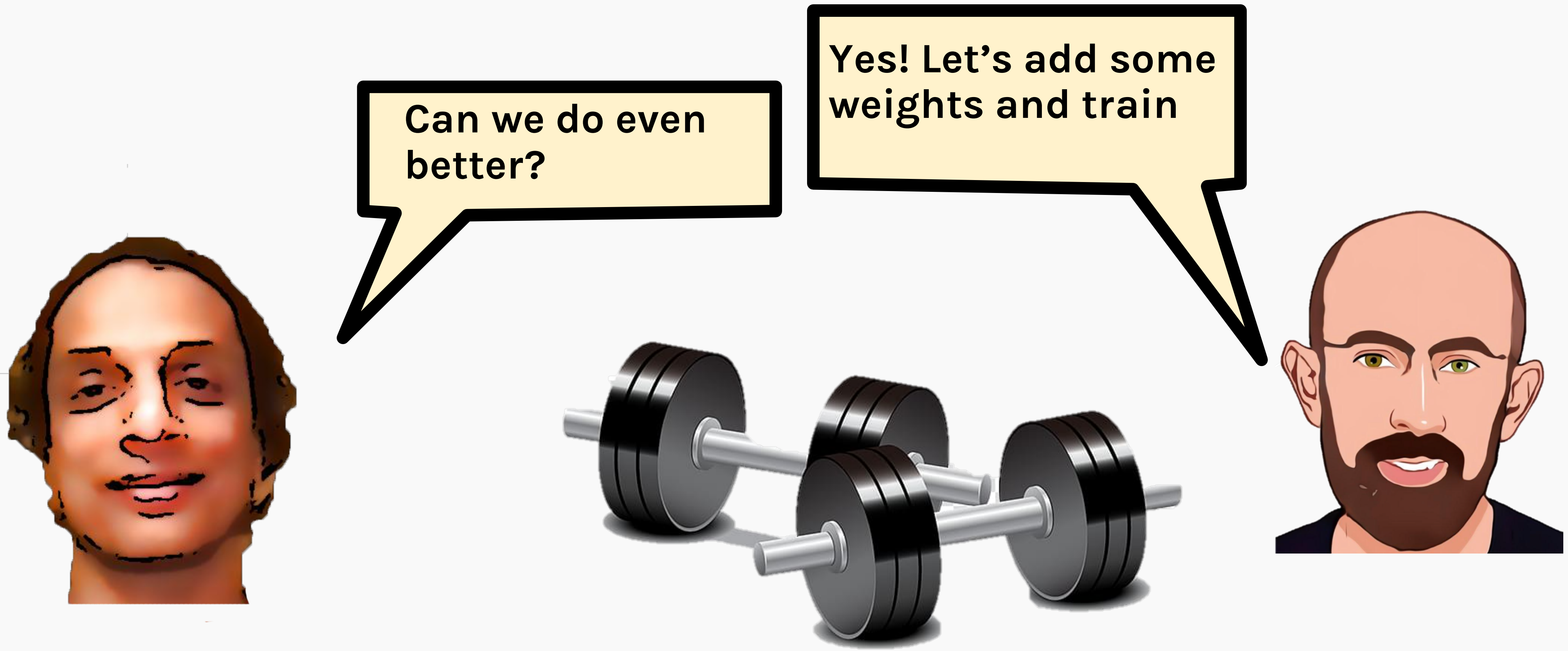
ATTENTION ISSUES?

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

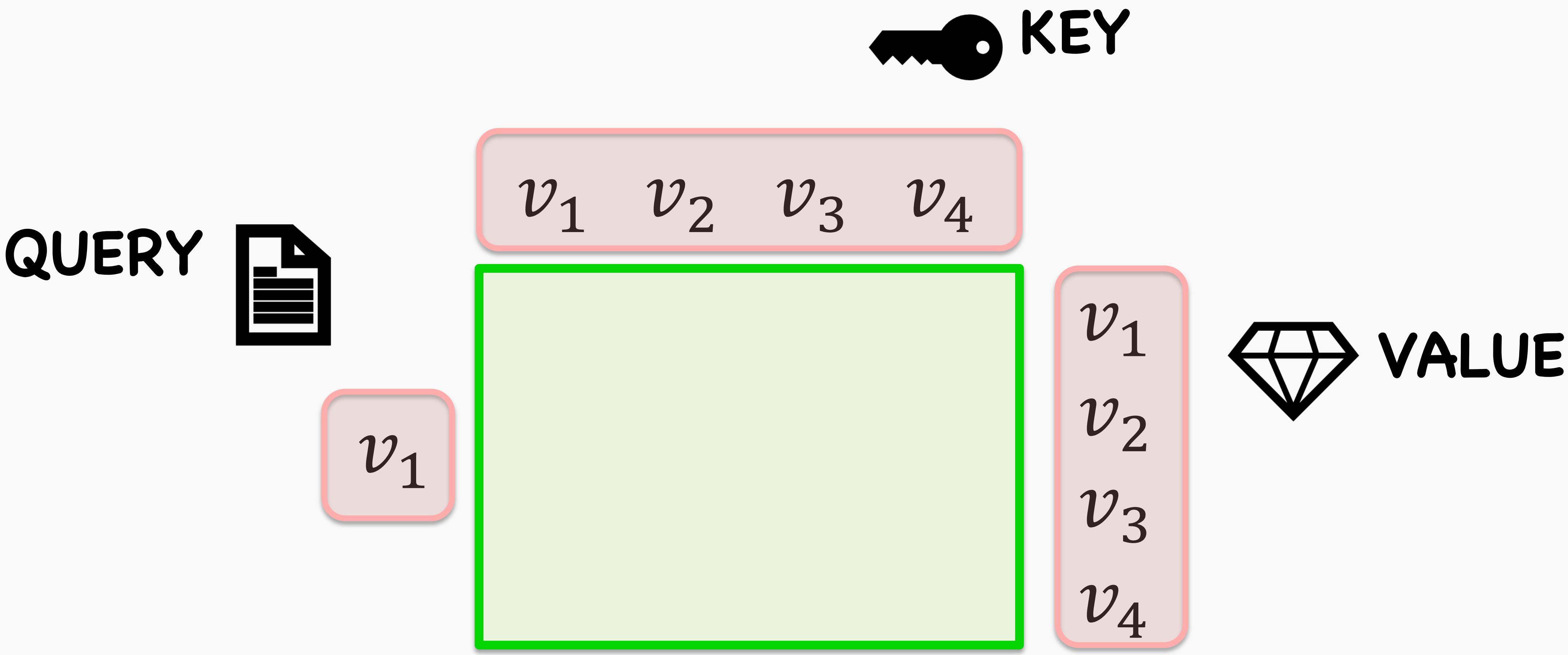
- No weights are trained in the process
- Attention as defined to be cosine similarity leads to fixed contextual mapping (two words will always have the same cosine similarity irrespective of the context)
- There is no positional information encoded



Attention – Text Data



Database Analogy



Database Analogy

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

KEY

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

QUERY

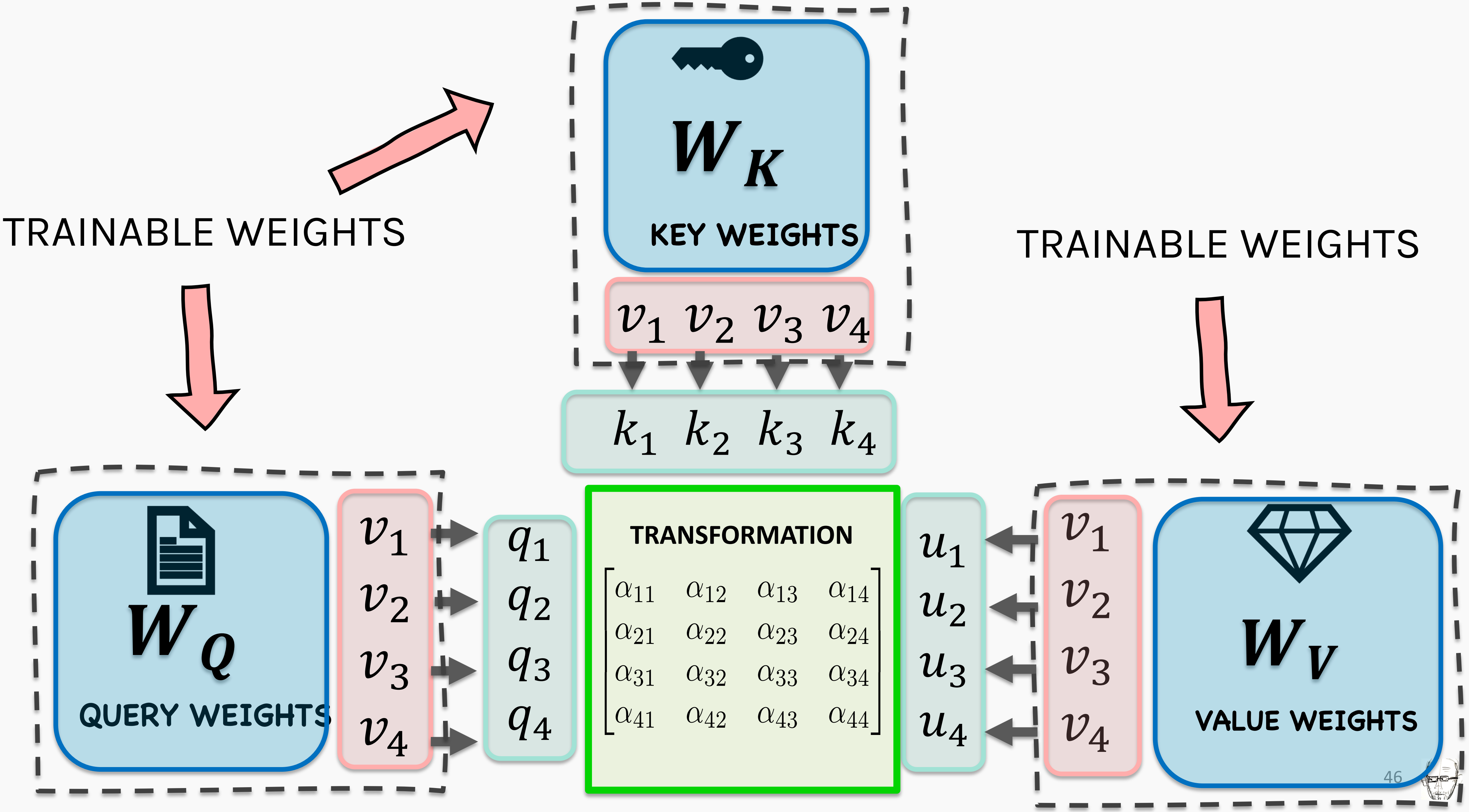
v_1
v_2
v_3
v_4

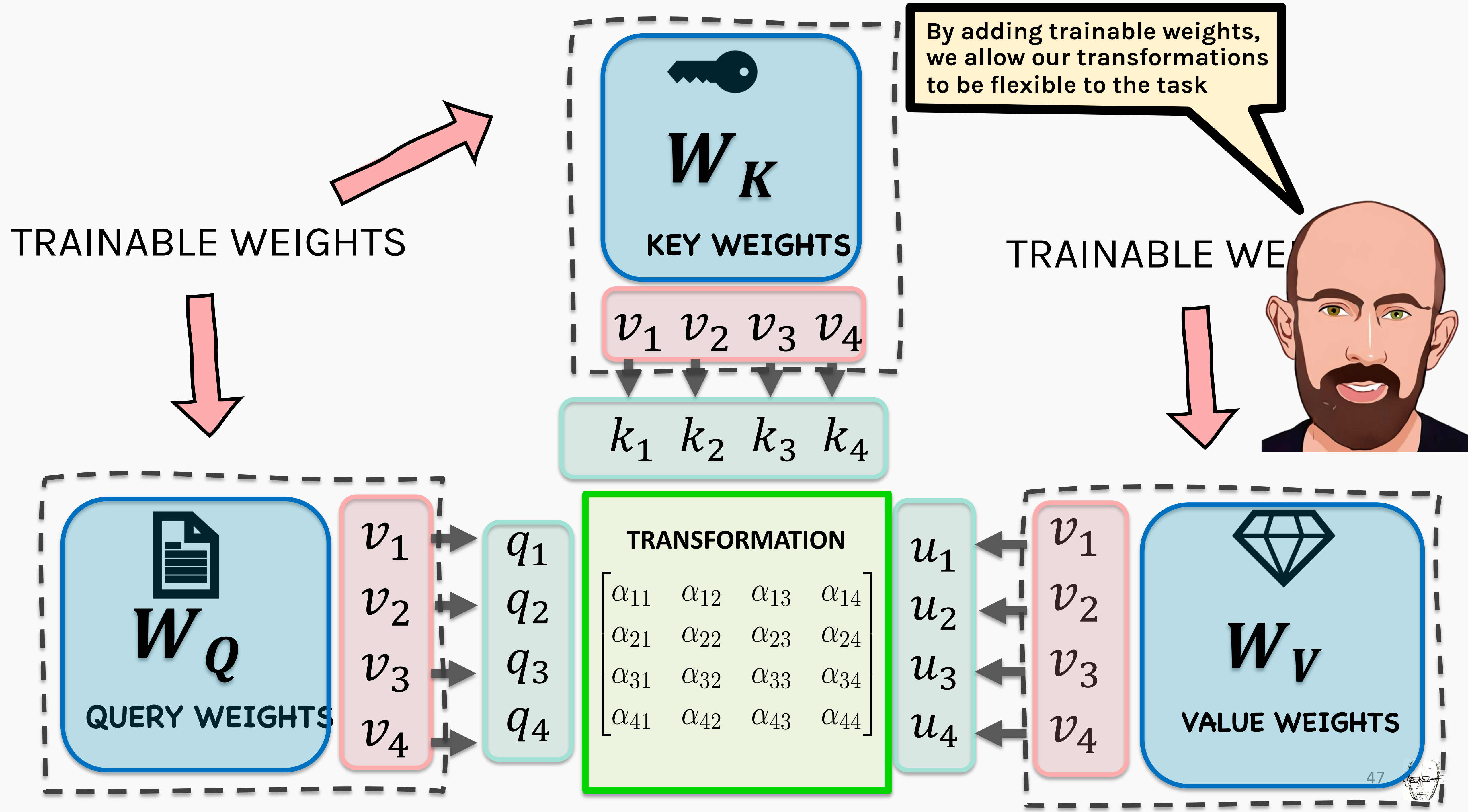
TRANSFORMATION

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

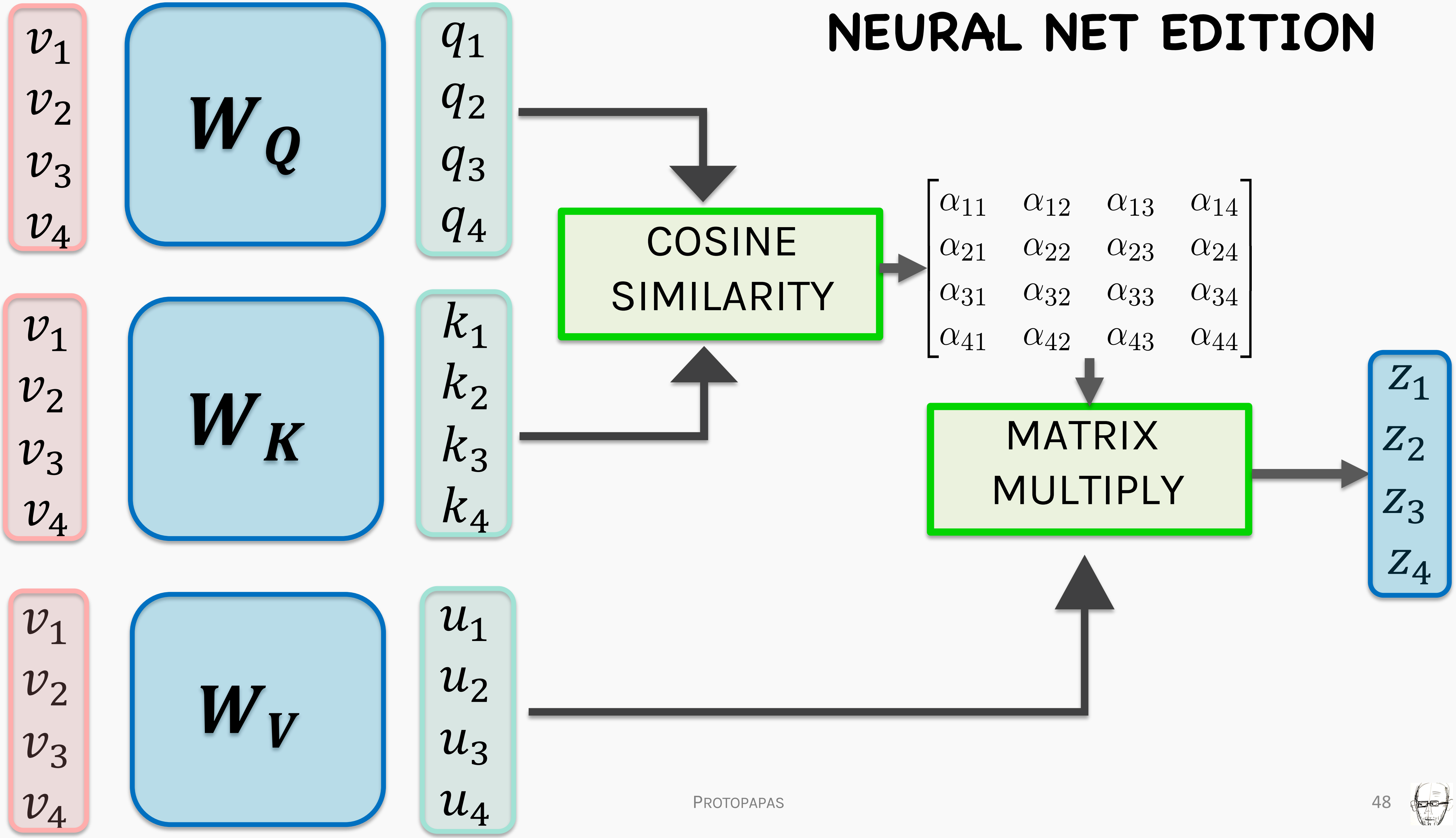
v_1
v_2
v_3
v_4

VALUE

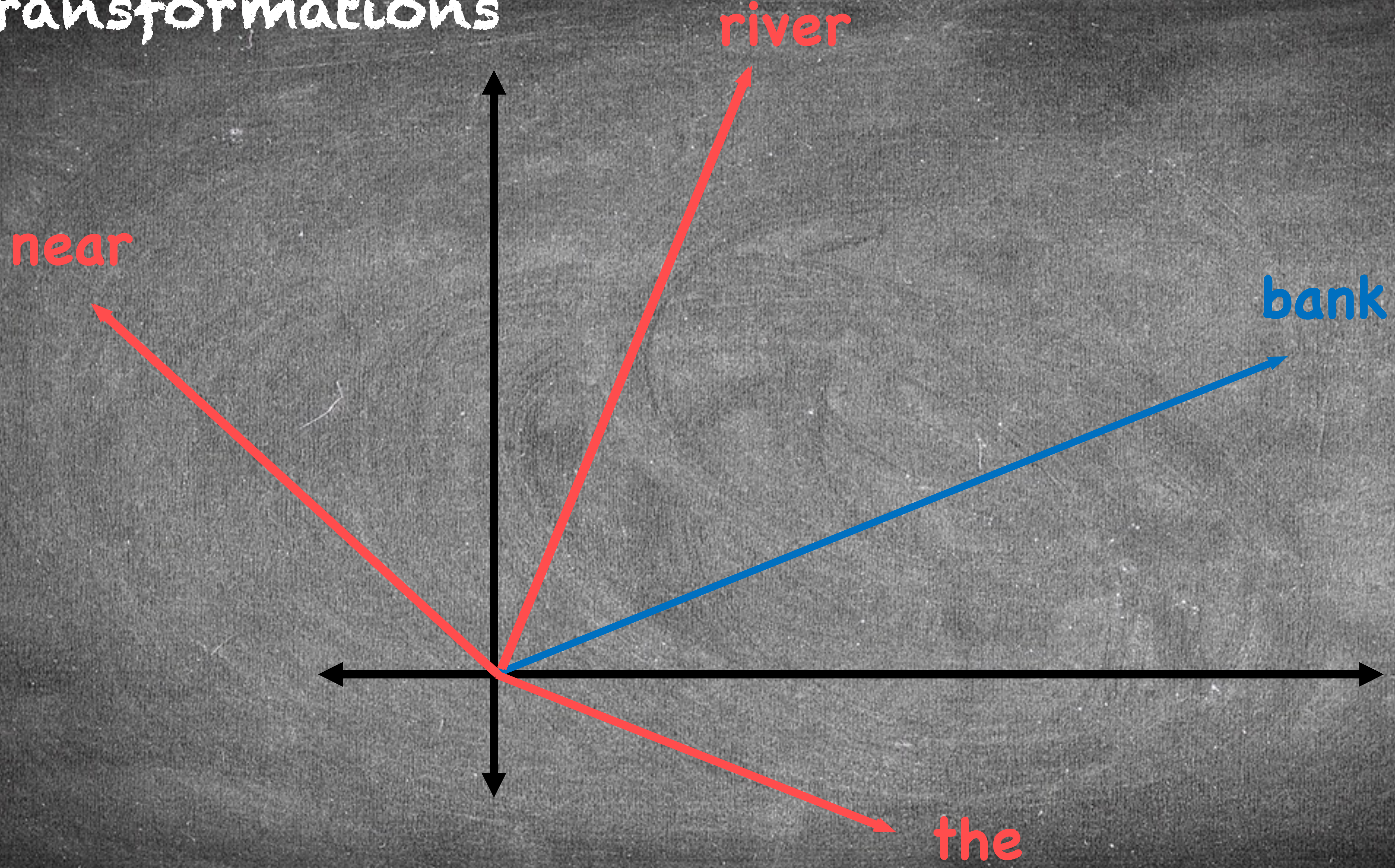

TRAINABLE WEIGHTS
W_K
KEY WEIGHTS
v_1 v_2 v_3 v_4
k_1 k_2 k_3 k_4
TRAINABLE WEIGHTS
W_Q
QUERY WEIGHTS
v_1 v_2 v_3 v_4
q_1 q_2 q_3 q_4
TRANSFORMATION
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32}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


By adding trainable weights, we allow our transformations to be flexible to the task
TRAINABLE WEIGHTS
W_K
KEY WEIGHTS
v_1 v_2 v_3 v_4
k_1 k_2 k_3 k_4
TRAINABLE WE
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

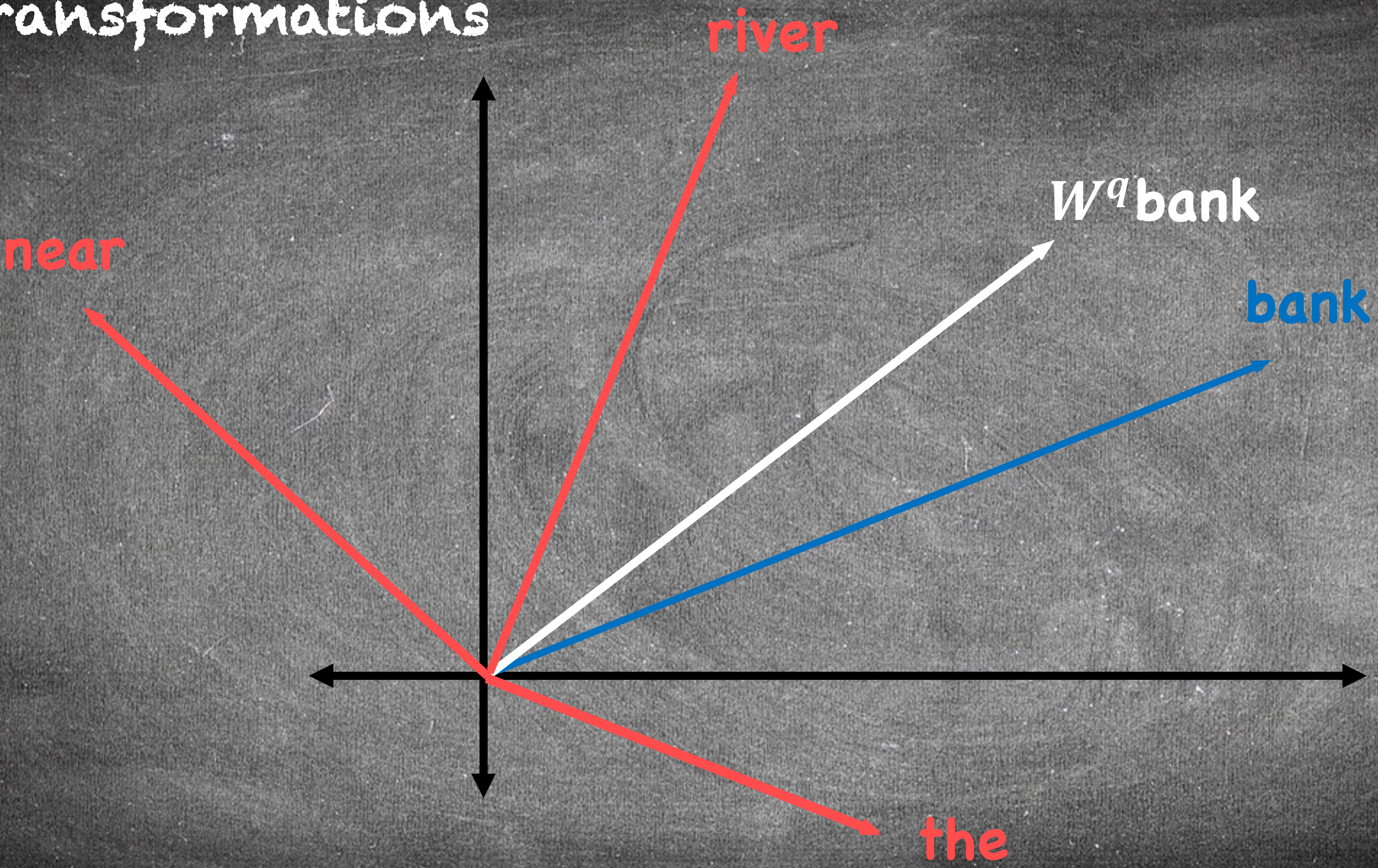
NEURAL NET EDITION



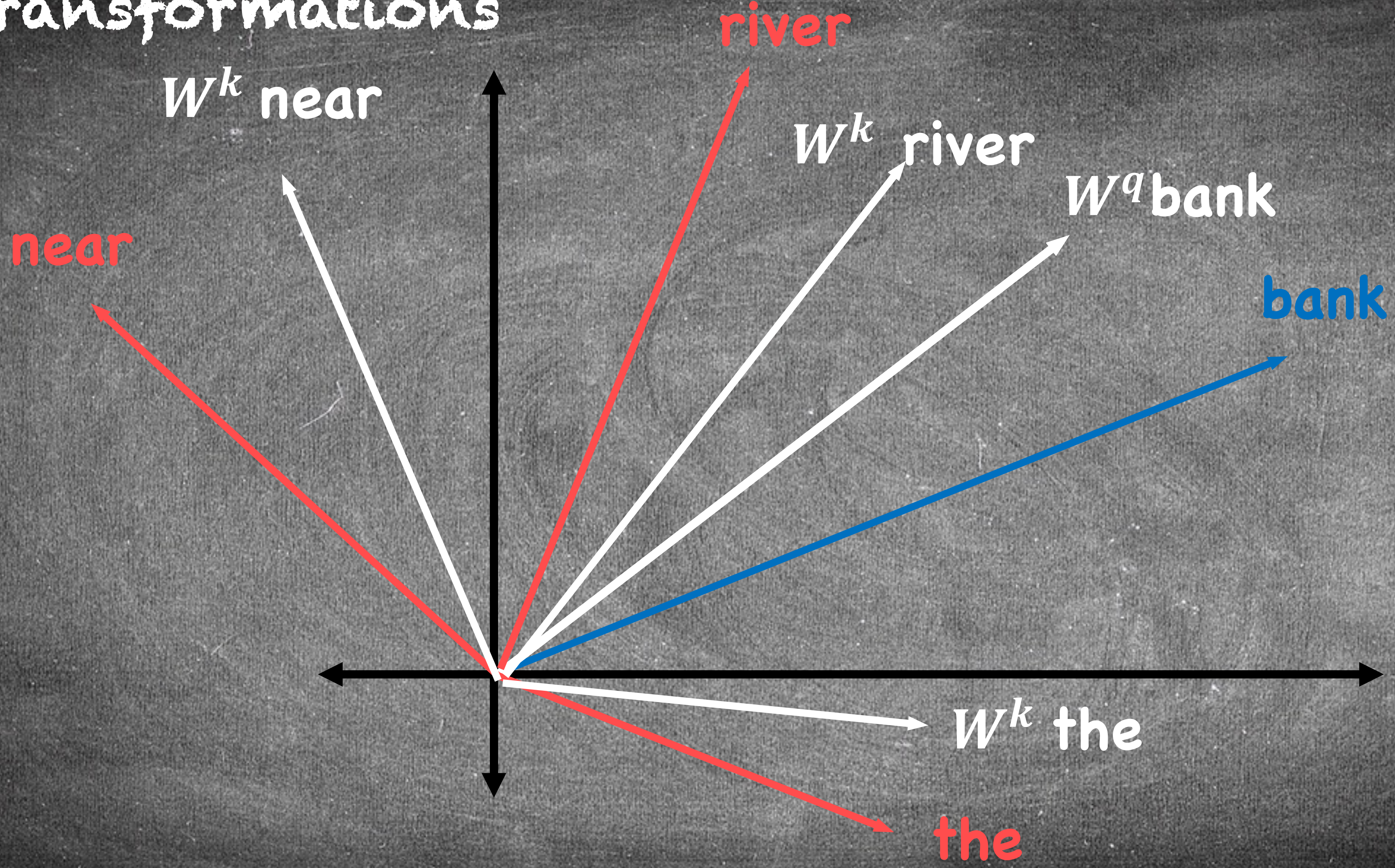
Transformations

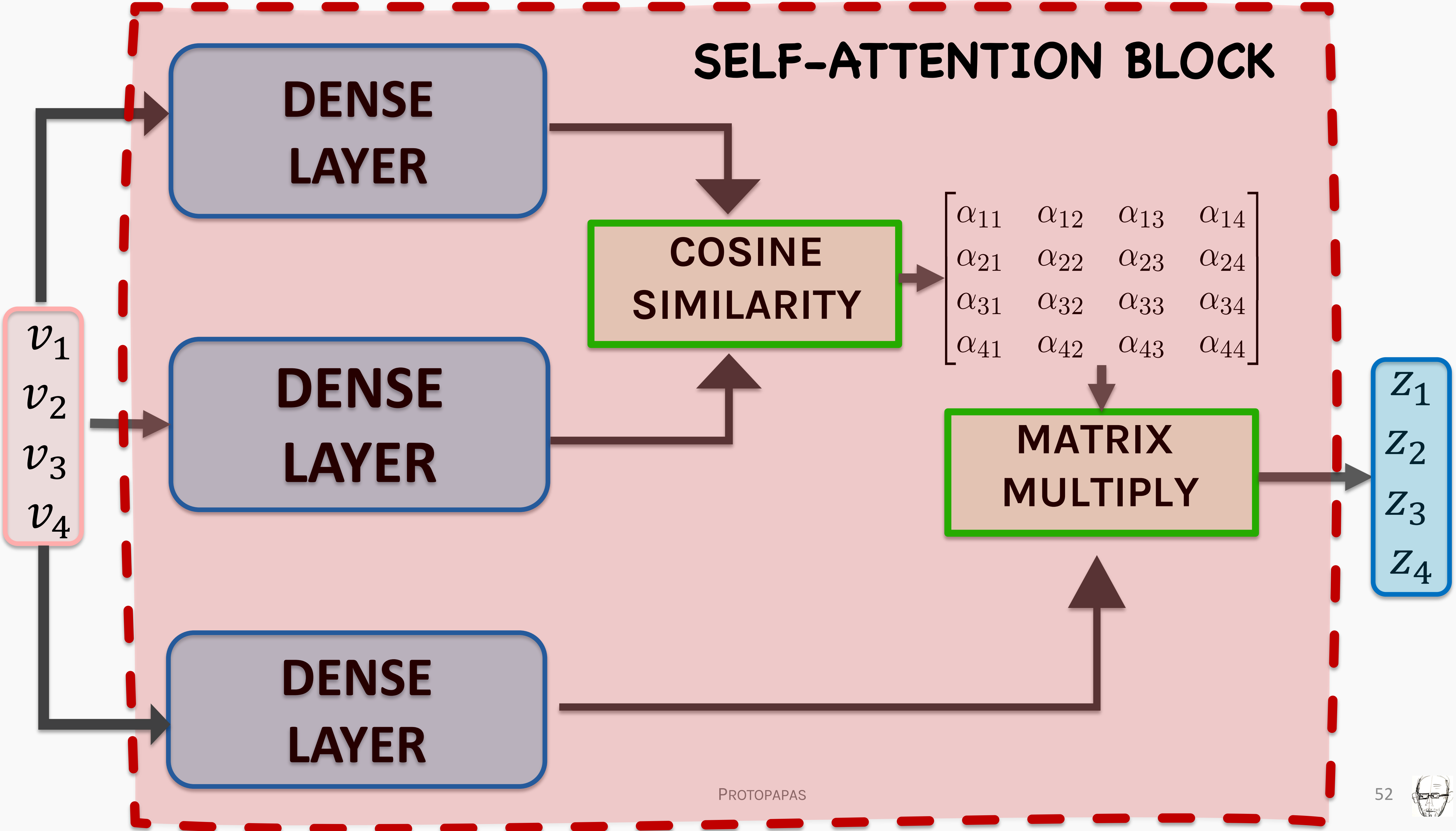


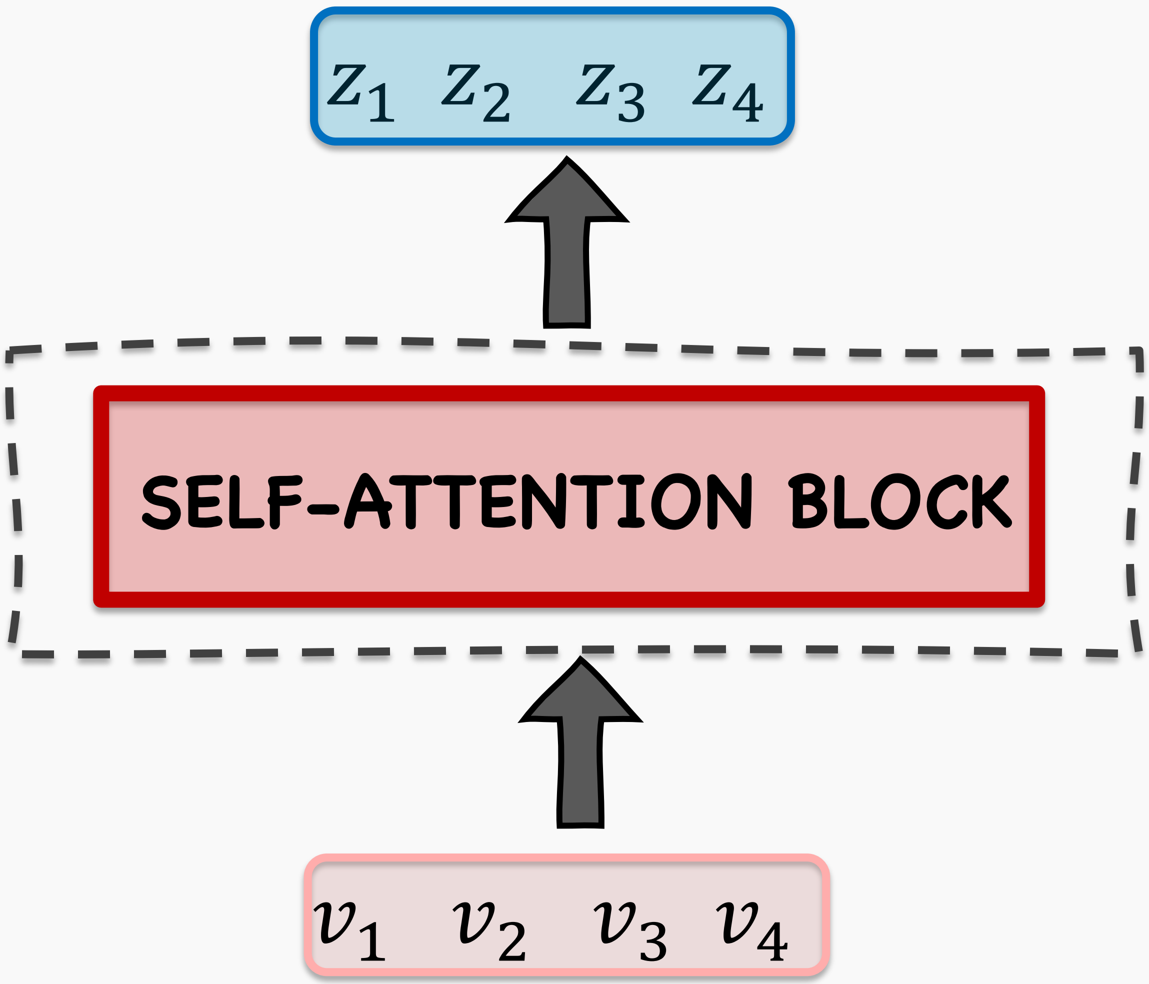
Transformations



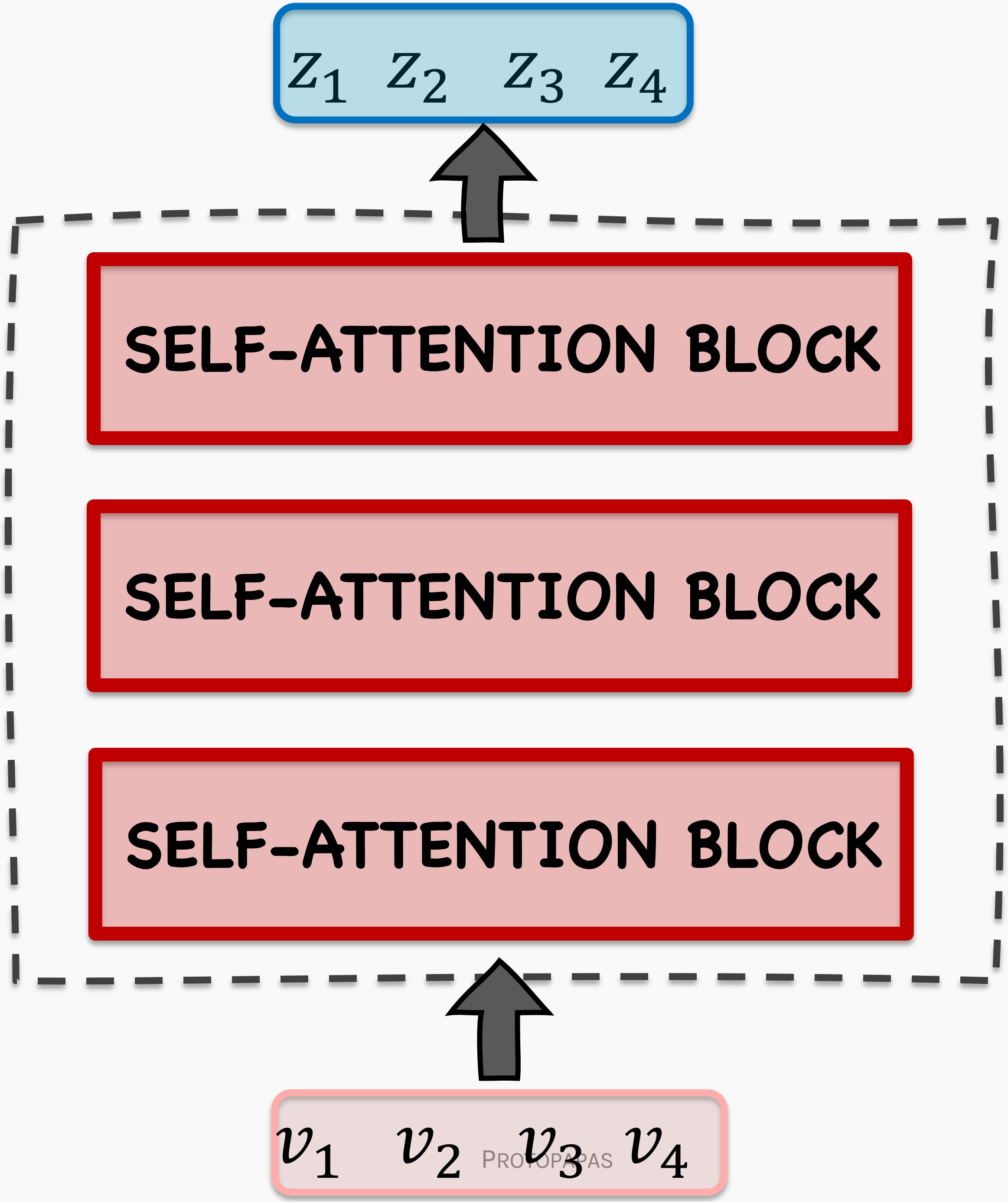
Transformations

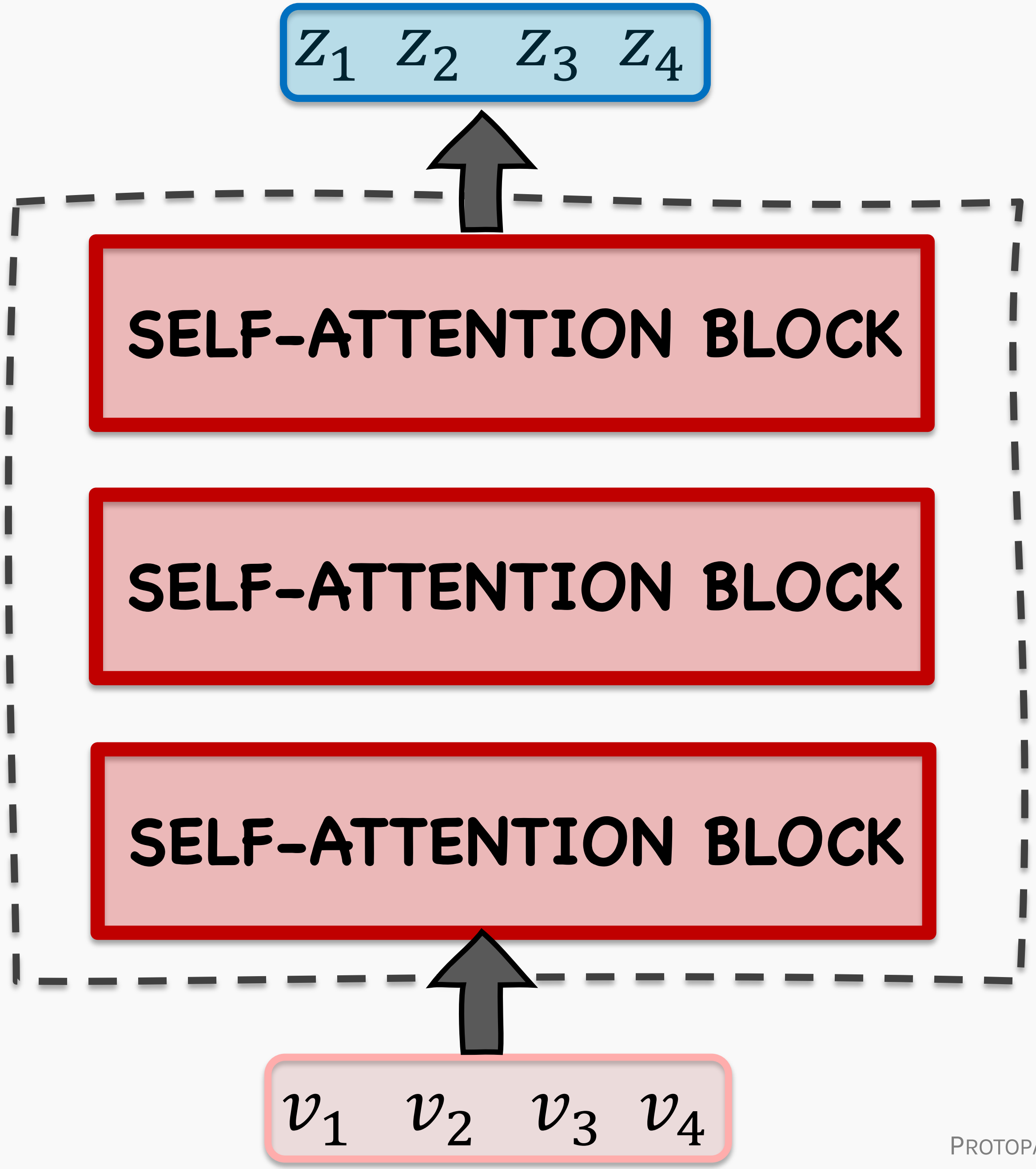


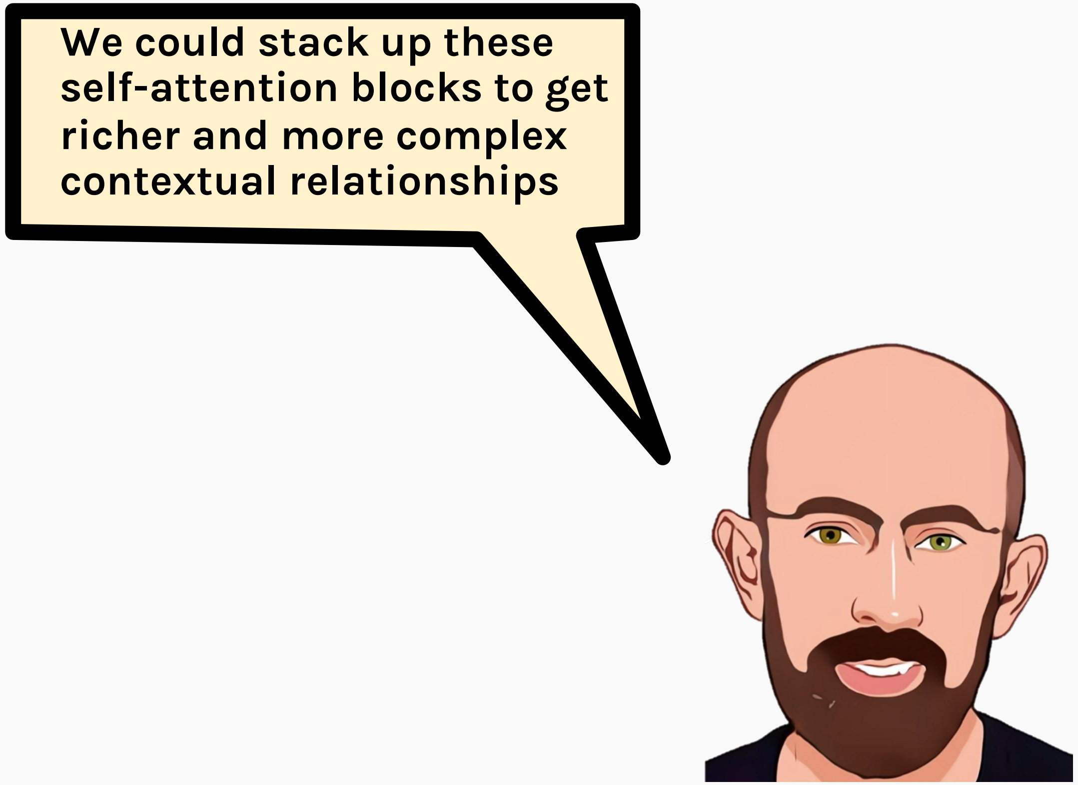

SELF-ATTENTION BLOCK
v_1 v_2 v_3 v_4
DENSE LAYER
DENSE LAYER
DENSE LAYER
COSINE SIMILARITY
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4



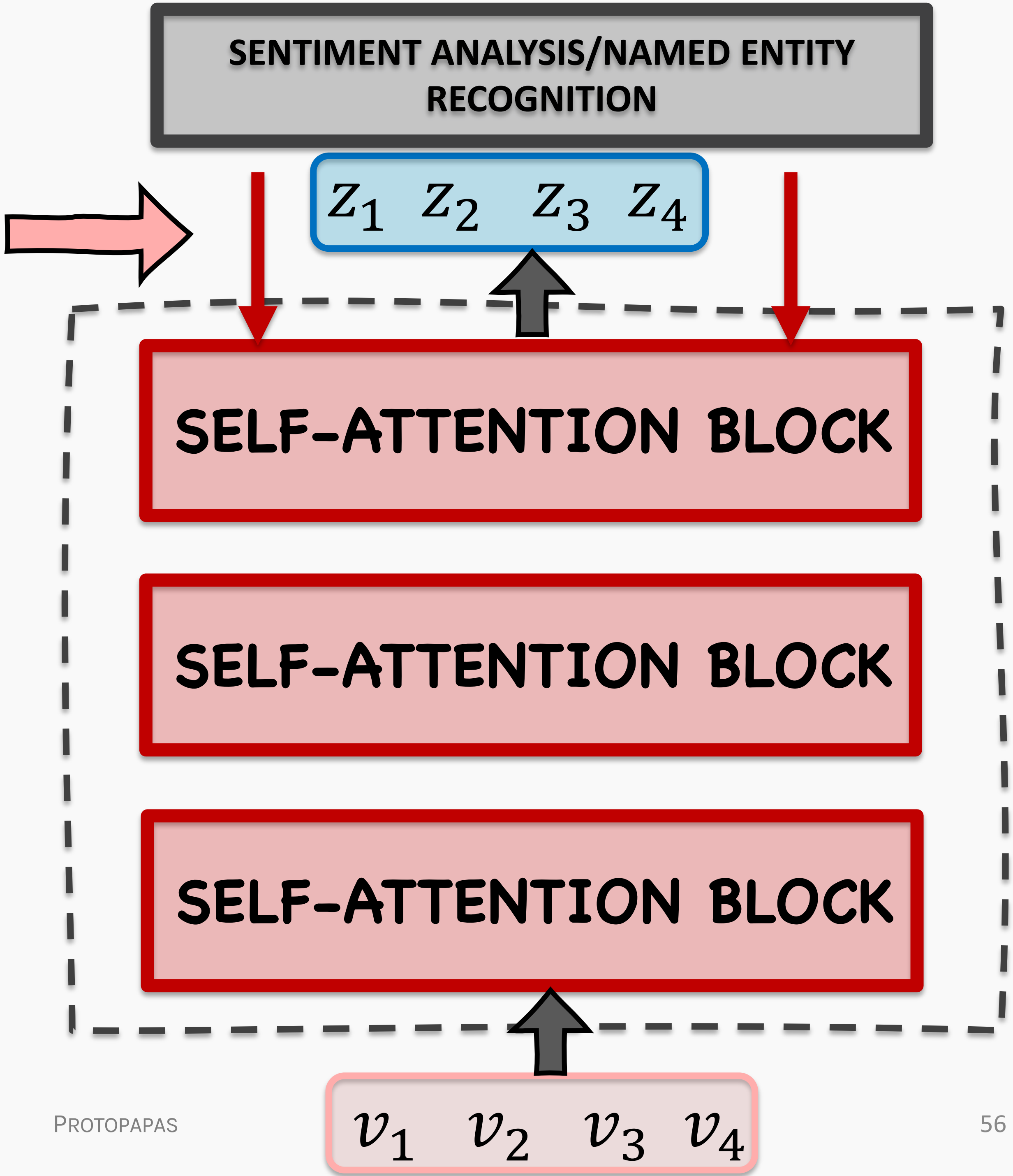
z_1 z_2 z_3 z_4
SELF-ATTENTION BLOCK
v_1 v_2 v_3 v_4


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

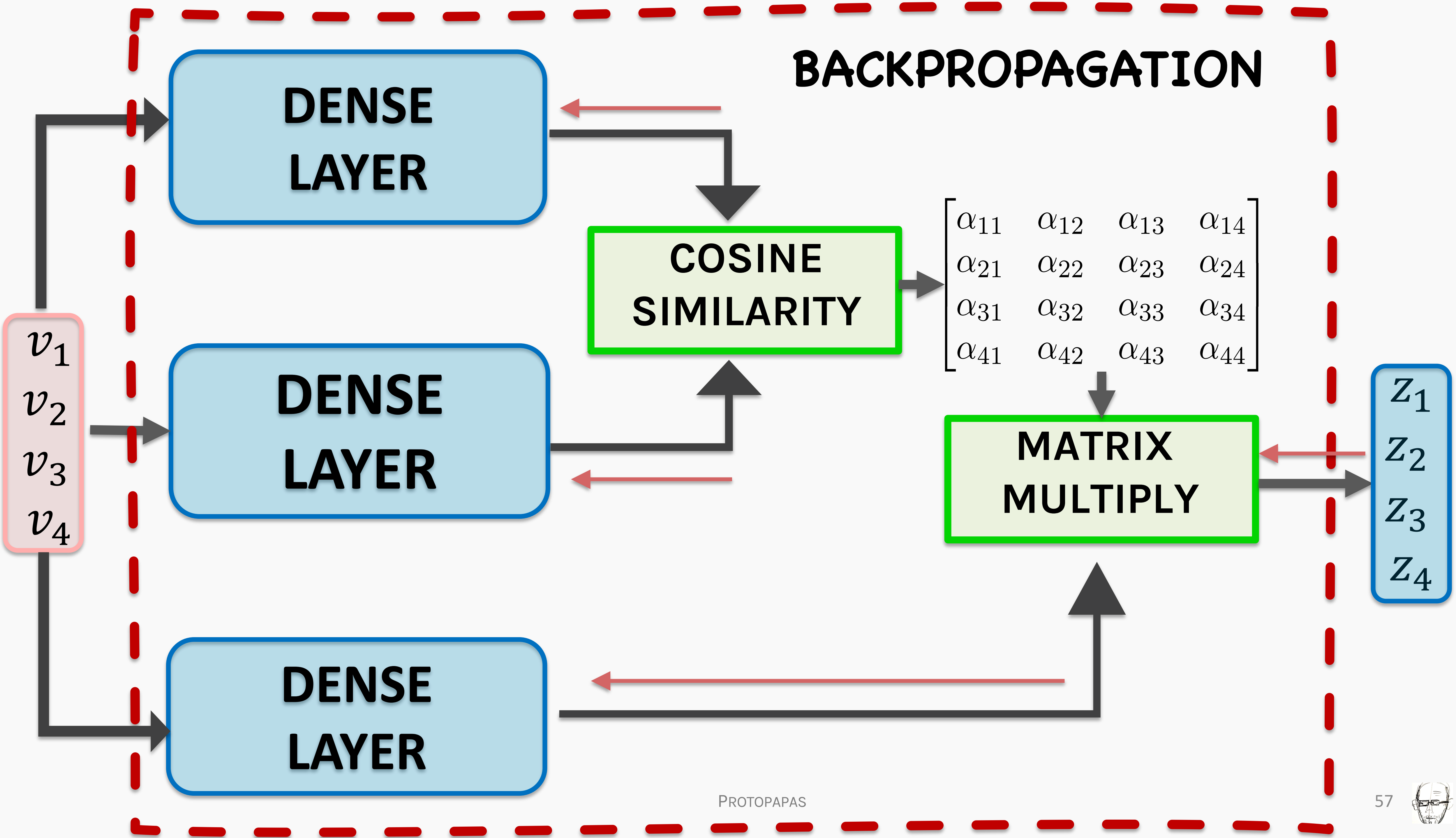

$z_1 \quad z_2 \quad z_3 \quad z_4$
SELF-ATTENTION BLOCK
SELF-ATTENTION BLOCK
SELF-ATTENTION BLOCK
$v_1 \quad v_2 \quad v_3 \quad v_4$


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

Flow of gradients

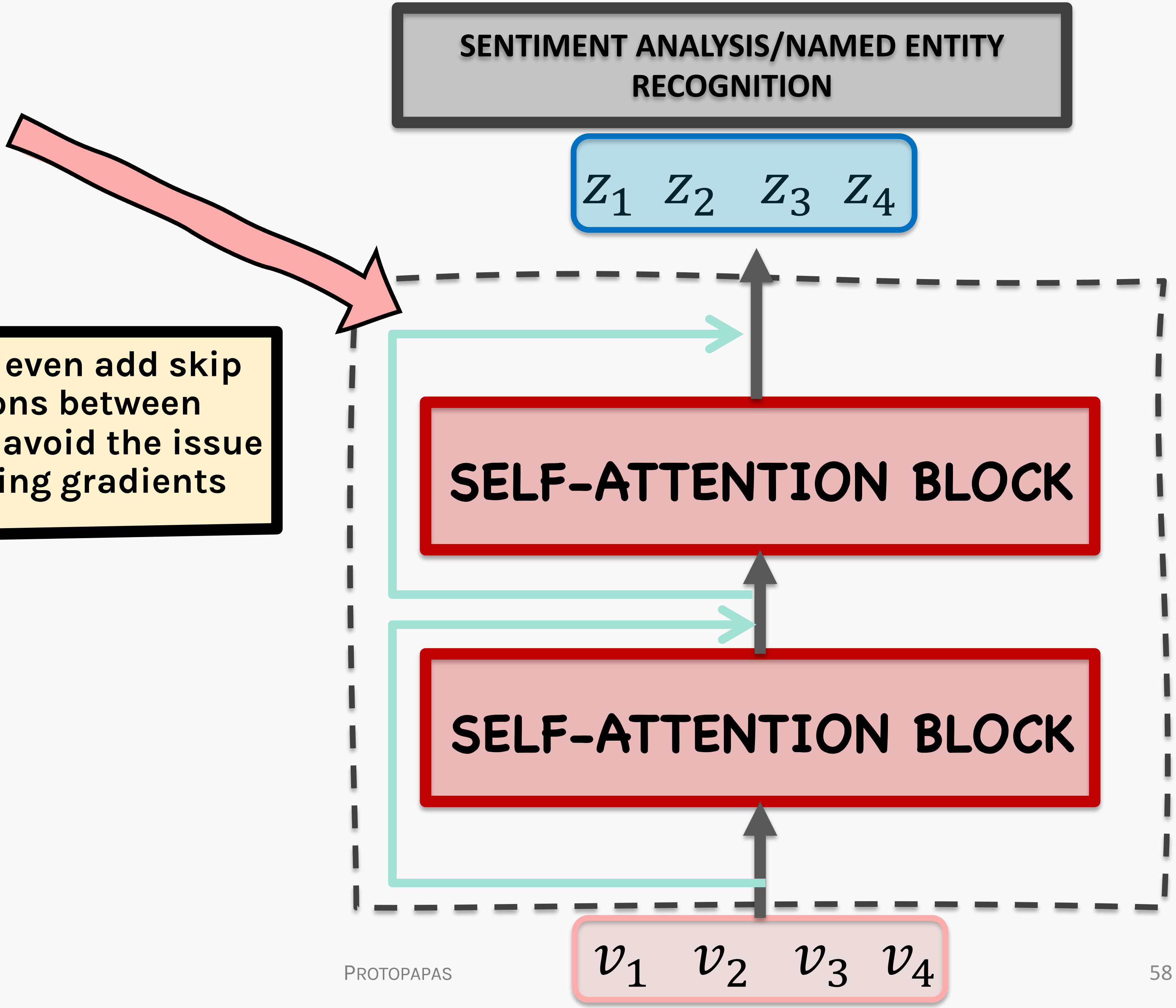


And we could train these weights by using backpropagation using masked words or a task like sentiment analysis


BACKPROPAGATION
DENSE LAYER
DENSE LAYER
DENSE LAYER
COSINE SIMILARITY
MATRIX MULTIPLY
v_1 v_2 v_3 v_4
z_1 z_2 z_3 z_4
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$

Skip connections

We could even add skip connections between blocks to avoid the issue of vanishing gradients



Attention

ATTENTION STRENGTHS?

- Unlike RNNs, each input is in direct context of every other input $O(1)$, hence vanishing gradients are not a significant issue with the attention block

- Unlike RNNs, the operations are not sequentially dependent (Non-Markovian)

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$

d: Dimension of embedding
n: Length of input sequence

Non-Markovian

Full context

Attention

ATTENTION ISSUES?

- **Optimization using attention leads to limited contextual mapping**
- There is no positional information encoded

Shivas spoke to Pavlos about attention

Attention: Do we have enough attention?

ATTENTION MECHANISM #1

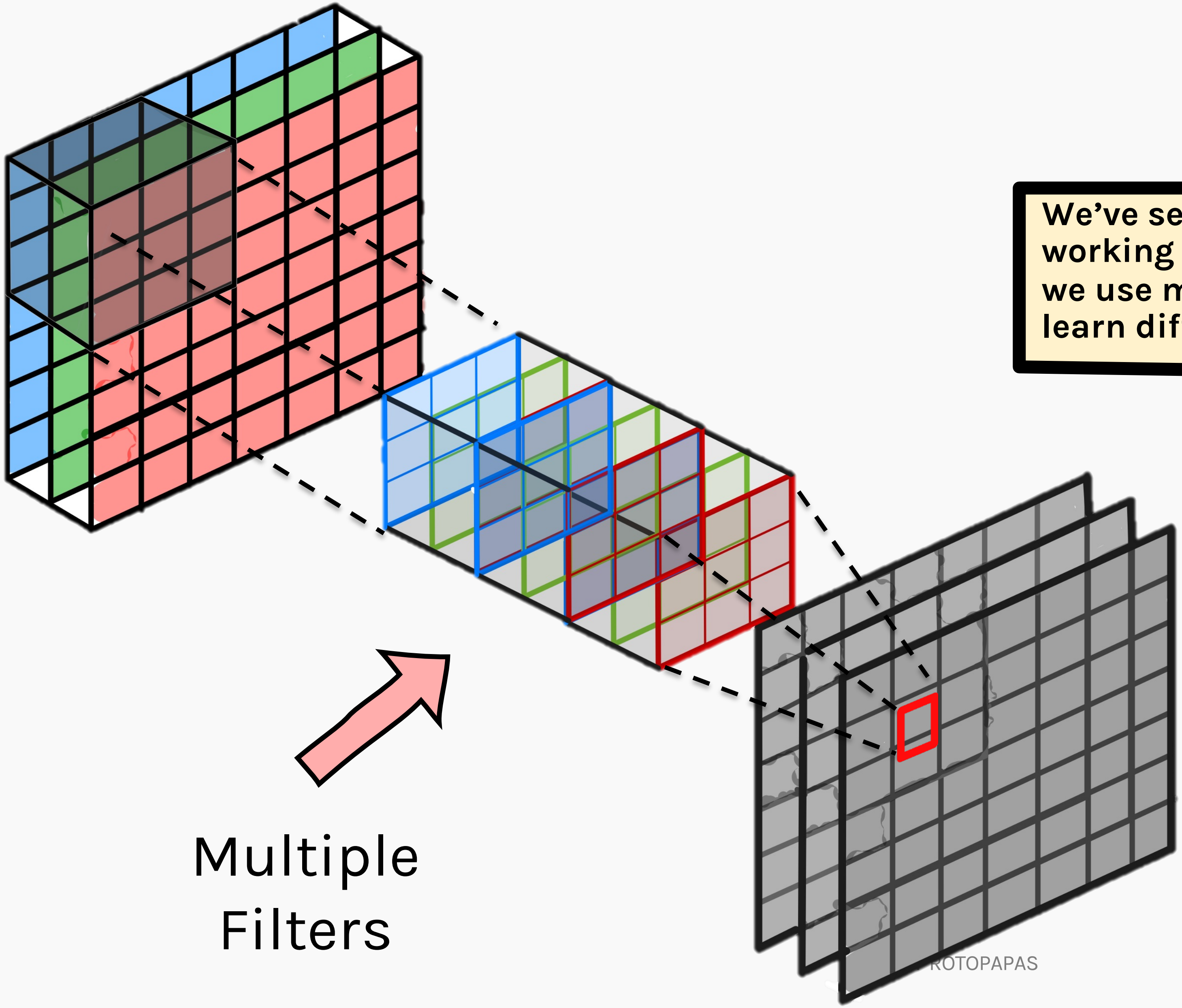
ATTENTION MECHANISM #2

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

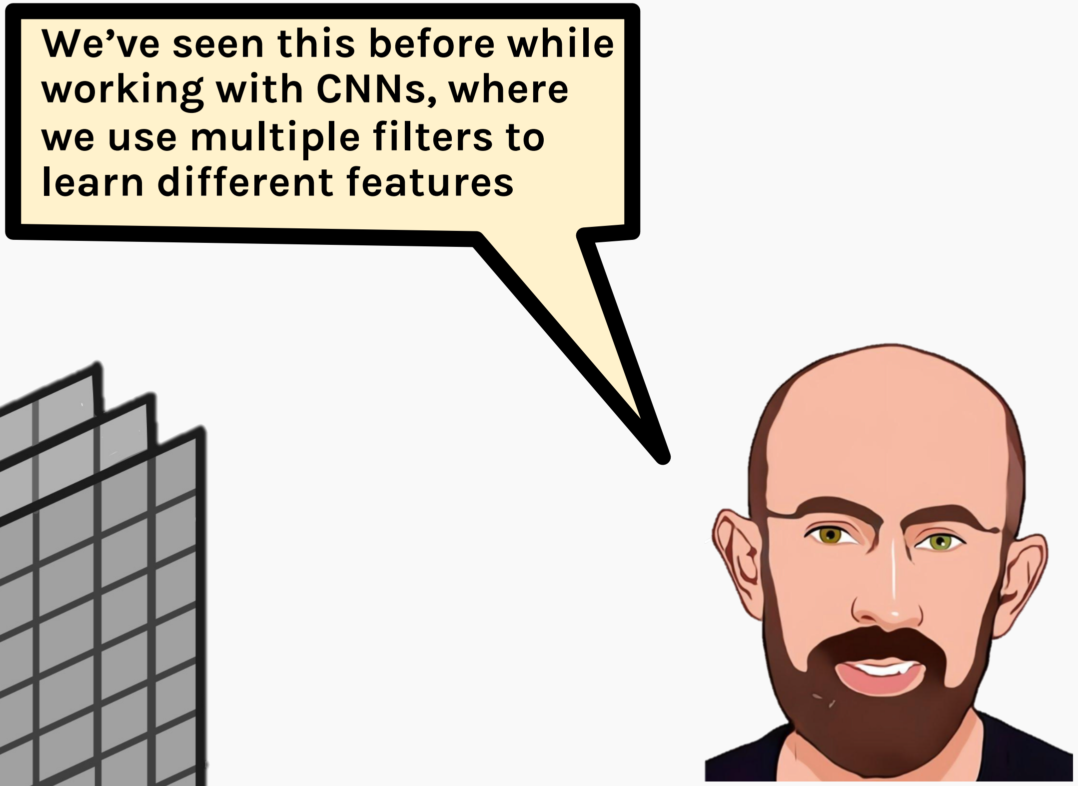
ATTENTION MECHANISM #3

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

Analogy - CNN Filters



Multiple Filters



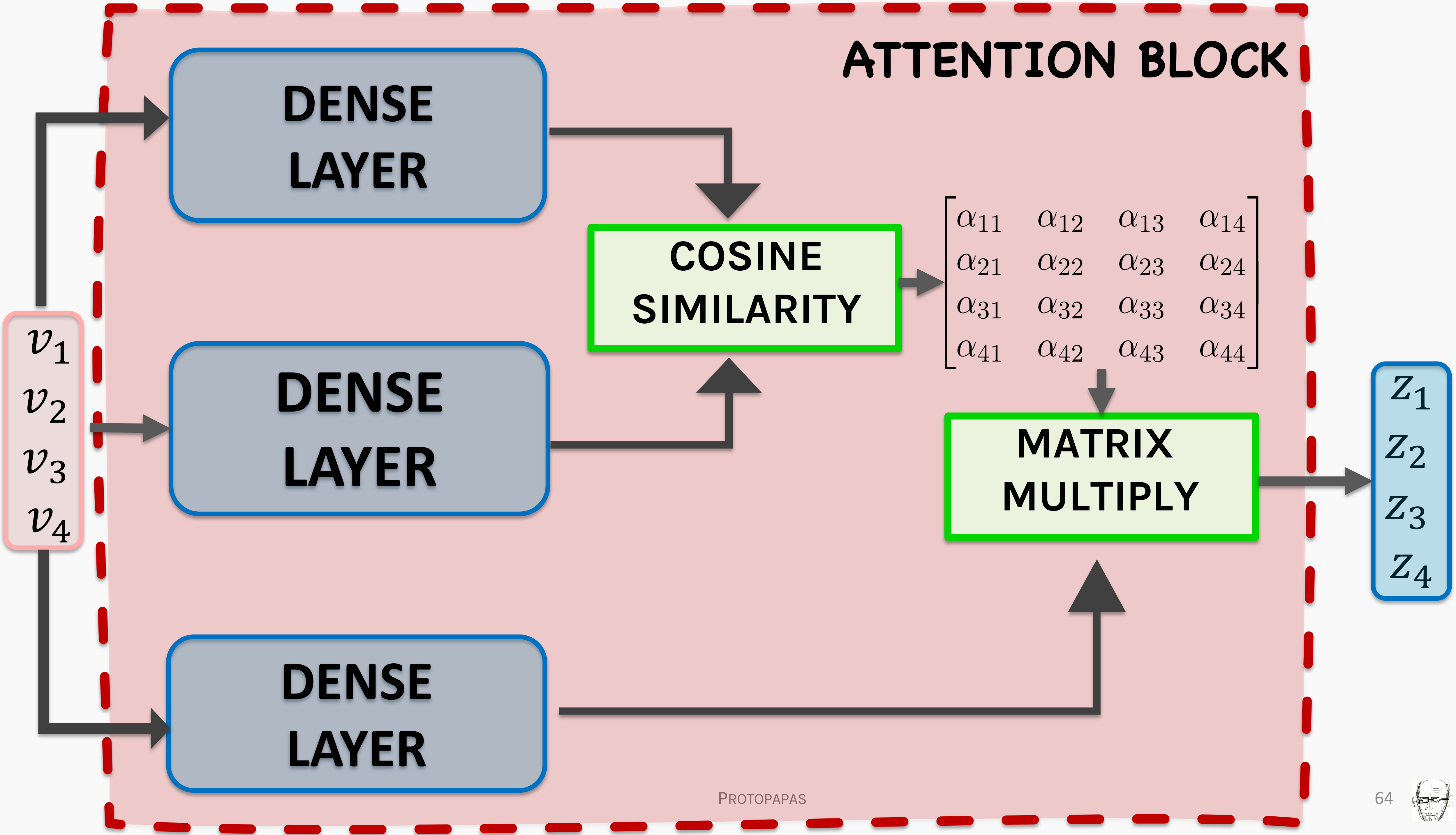
RECAP: CNNs

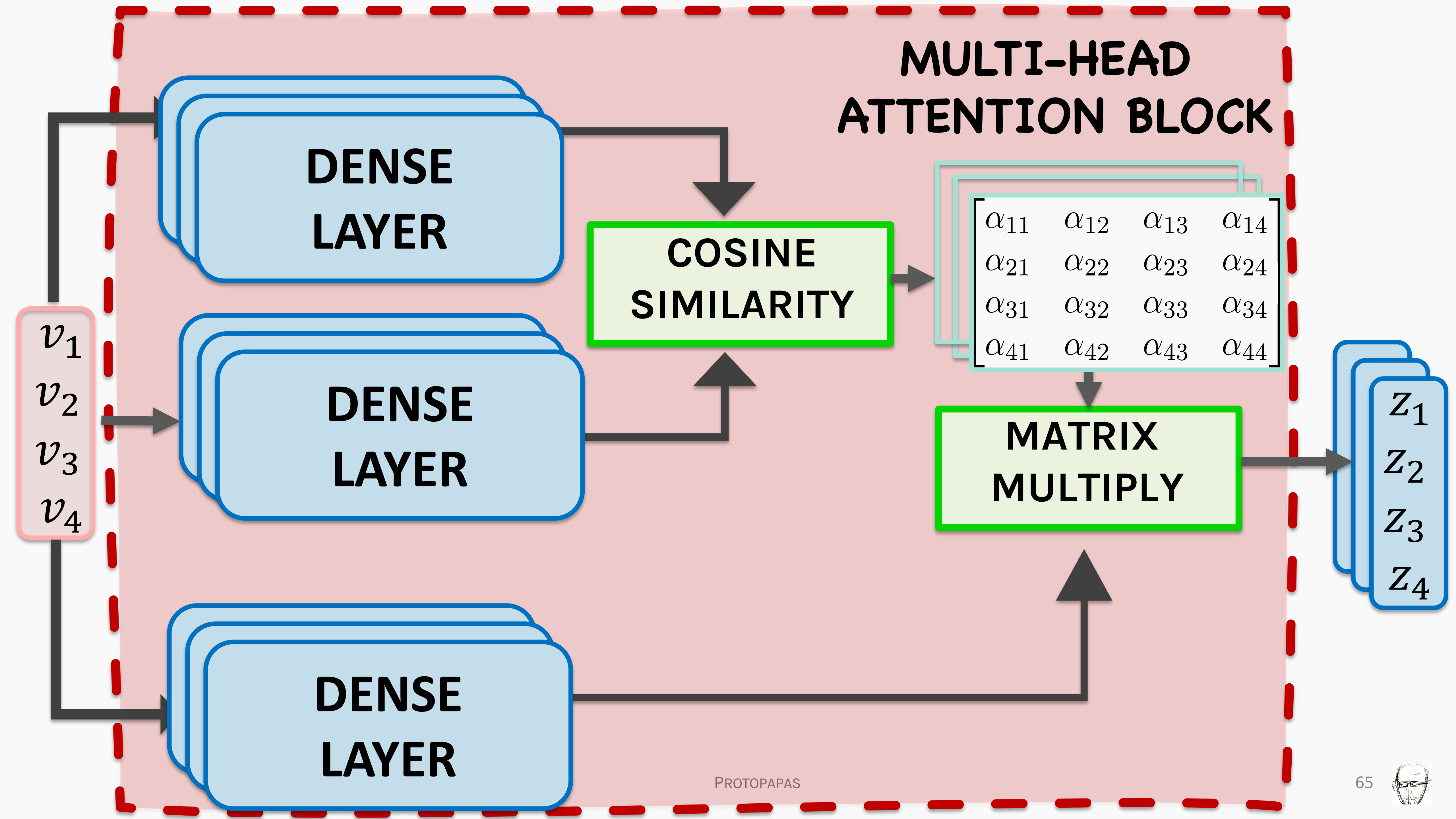
Imagine that we want to recognize swans in an image:

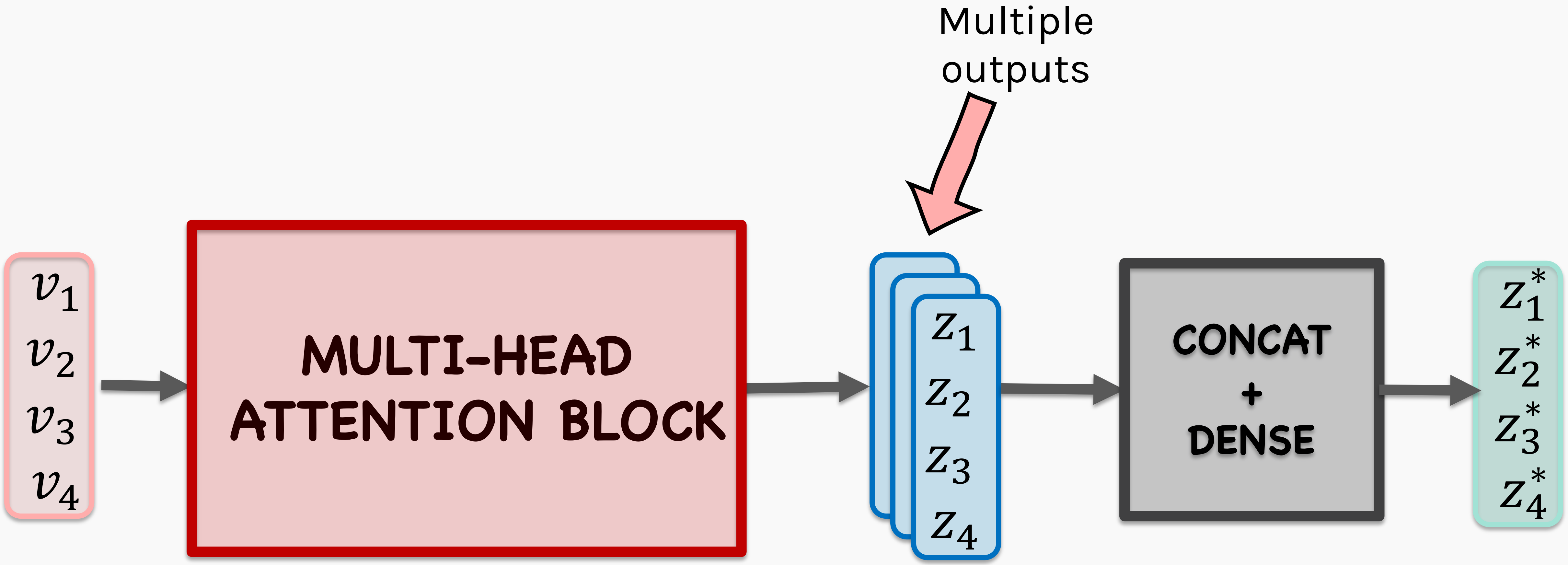
Oval-shaped white blob (body)

Round, elongated oval with orange protuberance

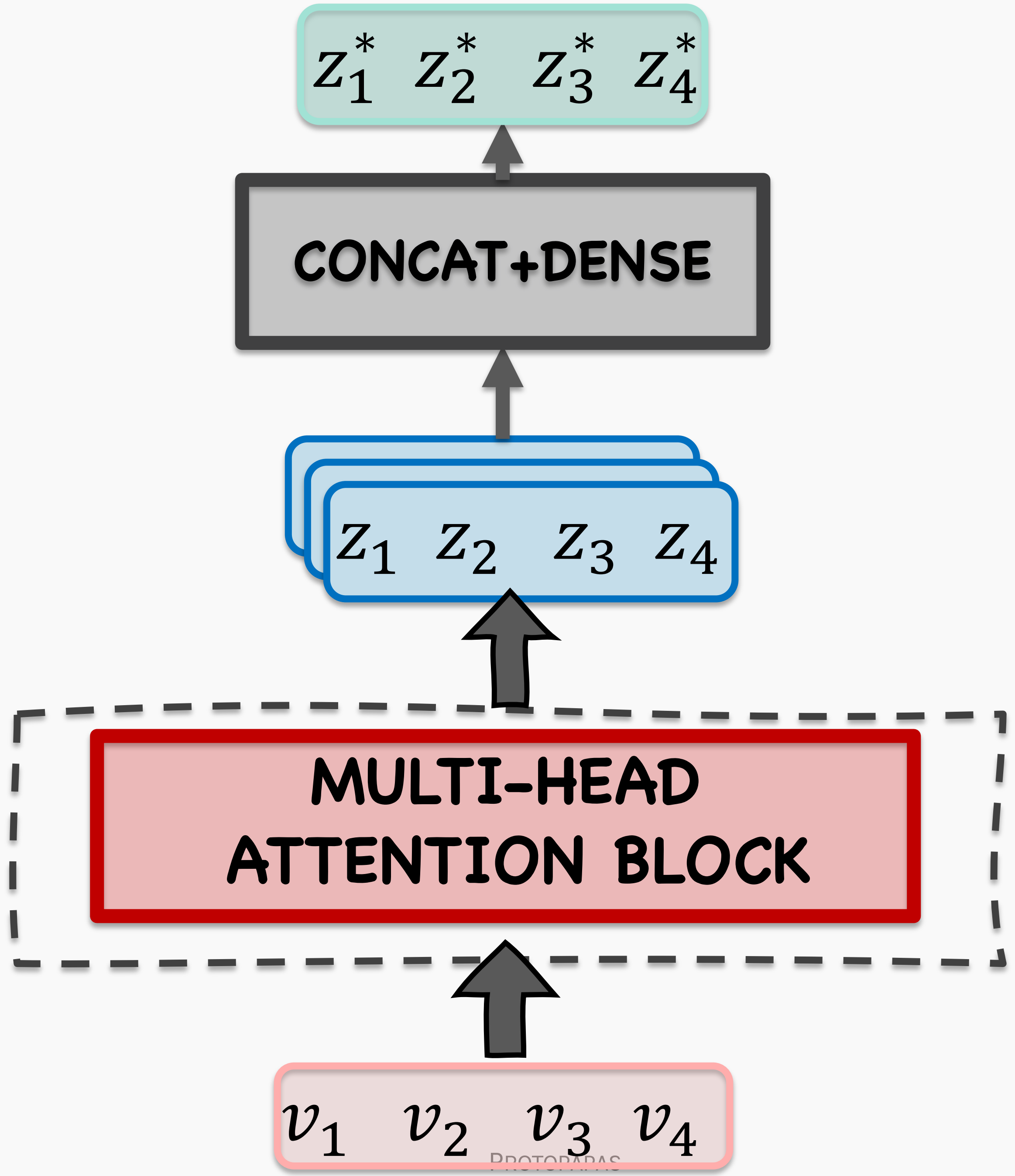
Long white rectangular shape (neck)

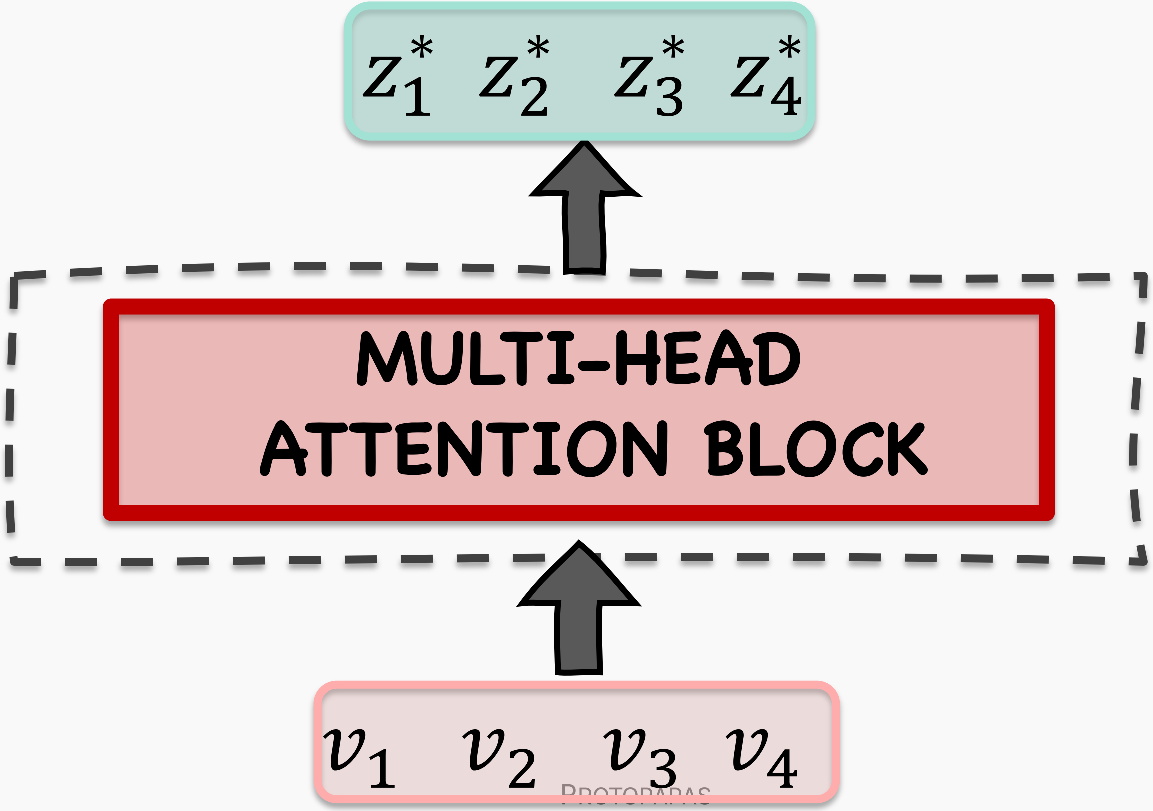

ATTENTION BLOCK
v_1 v_2 v_3 v_4
DENSE LAYER
DENSE LAYER
DENSE LAYER
COSINE SIMILARITY
$\begin{bmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{bmatrix}$
MATRIX MULTIPLY
z_1 z_2 z_3 z_4

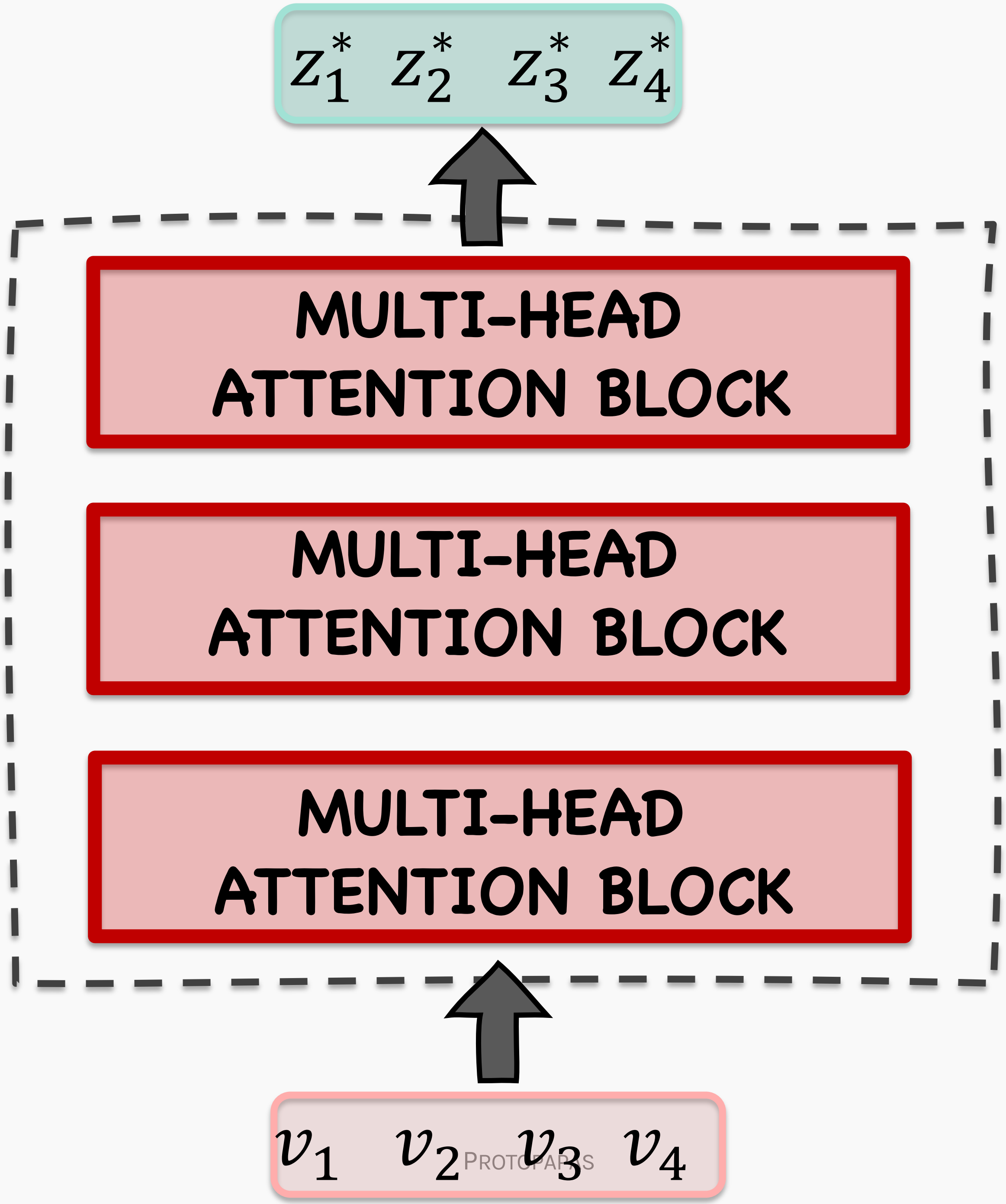

MULTI-HEAD ATTENTION BLOCK
v_1 v_2 v_3 v_4
DENSE LAYER
DENSE LAYER
DENSE LAYER
COSINE SIMILARITY
α_{11} α_{12} α_{13} α_{14}
α_{21} α_{22} α_{23} α_{24}
α_{31} α_{32} α_{33} α_{34}
α_{41} α_{42} α_{43} α_{44}
MATRIX MULTIPLY
z_1 z_2 z_3 z_4

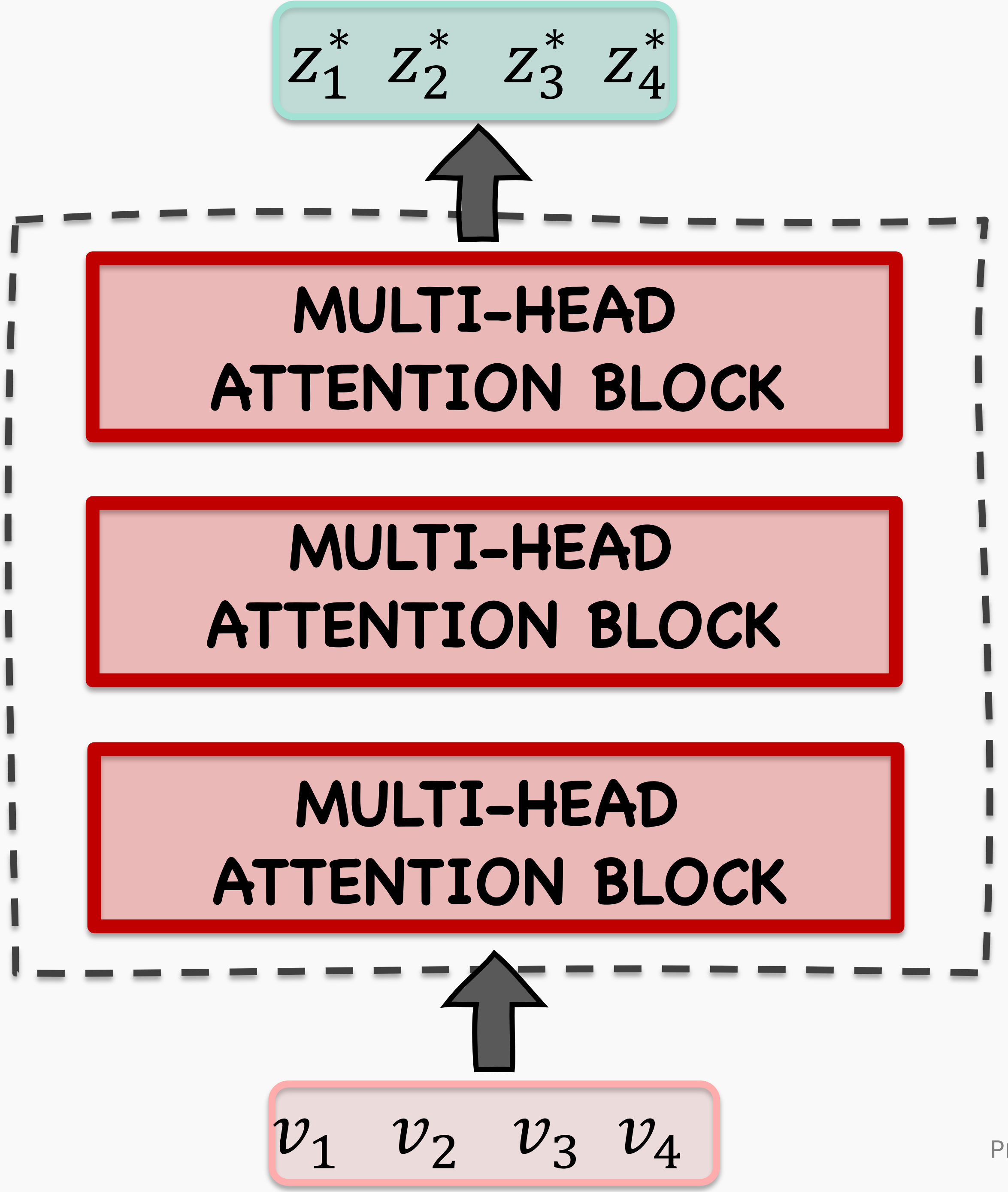


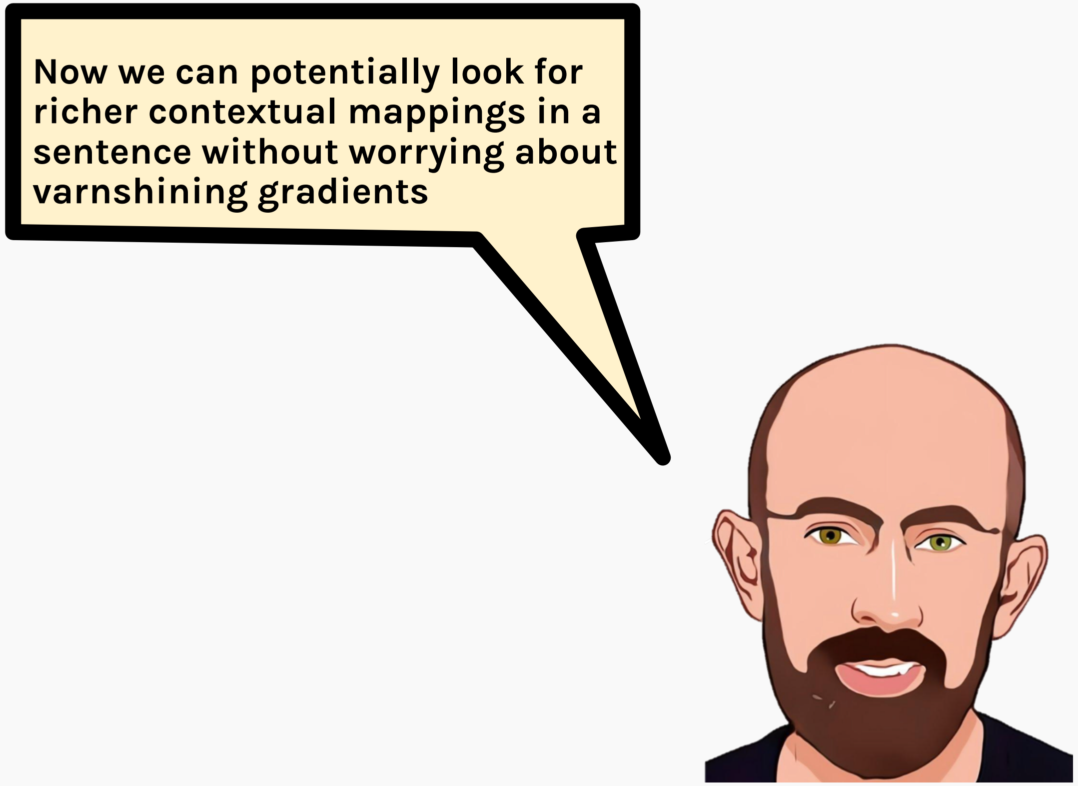
Multiple outputs
v_1
v_2
v_3
v_4
MULTI-HEAD ATTENTION BLOCK
z_1
z_2
z_3
z_4
CONCAT + DENSE
z_1^*
z_2^*
z_3^*
z_4^*


z_1^* z_2^* z_3^* z_4^*
CONCAT+DENSE
z_1 z_2 z_3 z_4
MULTI-HEAD ATTENTION BLOCK
v_1 v_2 v_3 v_4
PROTOPAPAS


z_1^* z_2^* z_3^* z_4^*
MULTI-HEAD
ATTENTION BLOCK
v_1 v_2 v_3 v_4


$z_1^* \quad z_2^* \quad z_3^* \quad z_4^*$
MULTI-HEAD ATTENTION BLOCK
MULTI-HEAD ATTENTION BLOCK
MULTI-HEAD ATTENTION BLOCK
$v_1 \quad v_2 \quad v_3 \quad v_4$


$z_1^* \quad z_2^* \quad z_3^* \quad z_4^*$
MULTI-HEAD ATTENTION BLOCK
MULTI-HEAD ATTENTION BLOCK
MULTI-HEAD ATTENTION BLOCK
$v_1 \quad v_2 \quad v_3 \quad v_4$


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

Attention

MULTI-HEAD ATTENTION ISSUES?

- ~~No weights trained in the process~~
- ~~Optimization using attention leads to limited contextual mapping~~
- There is no positional information encoded

Shivas spoke to Pavlos about attention

Attention

MULTI-HEAD ATTENTION ISSUES?

- ~~No weights trained in the process~~
- ~~Optimization using attention leads to limited contextual mapping~~
- There is no positional information encoded

Pavlos spoke to Shivas about attention

What we want

LANGUAGE MODEL WISHLIST

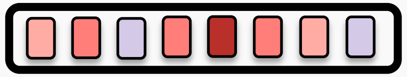
- Position and order of words are the essential parts of any language
- Recurrent Neural Networks (RNNs) inherently take the order of word into account
- Multi-head attention blocks do not take such an order by design, so there's the need to incorporate the order of the words separately

💡 **IDEA #255:** Positional Encoding

Positional Encoding

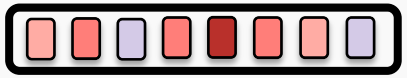
Assume an input embedding v_i of some dimension for the word 'Shivas' at the position *i* in the sentence

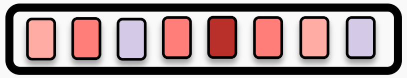
v_i

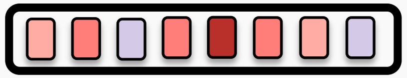


This input embedding will be the same for any position in the sentence

v_1 v_2 v_3





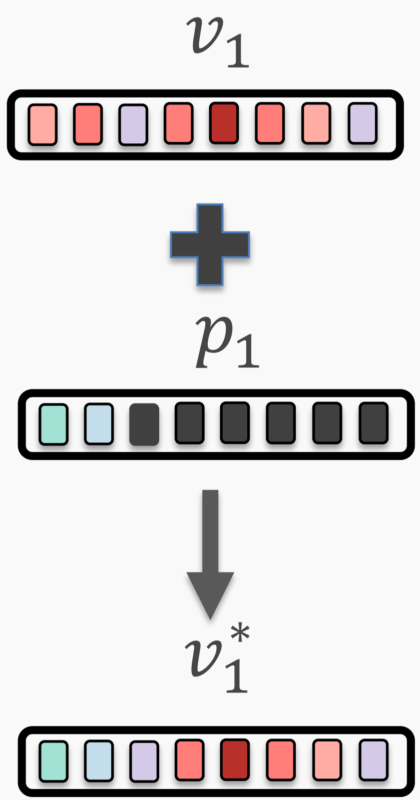


We hope to modify the embedding with some positional information

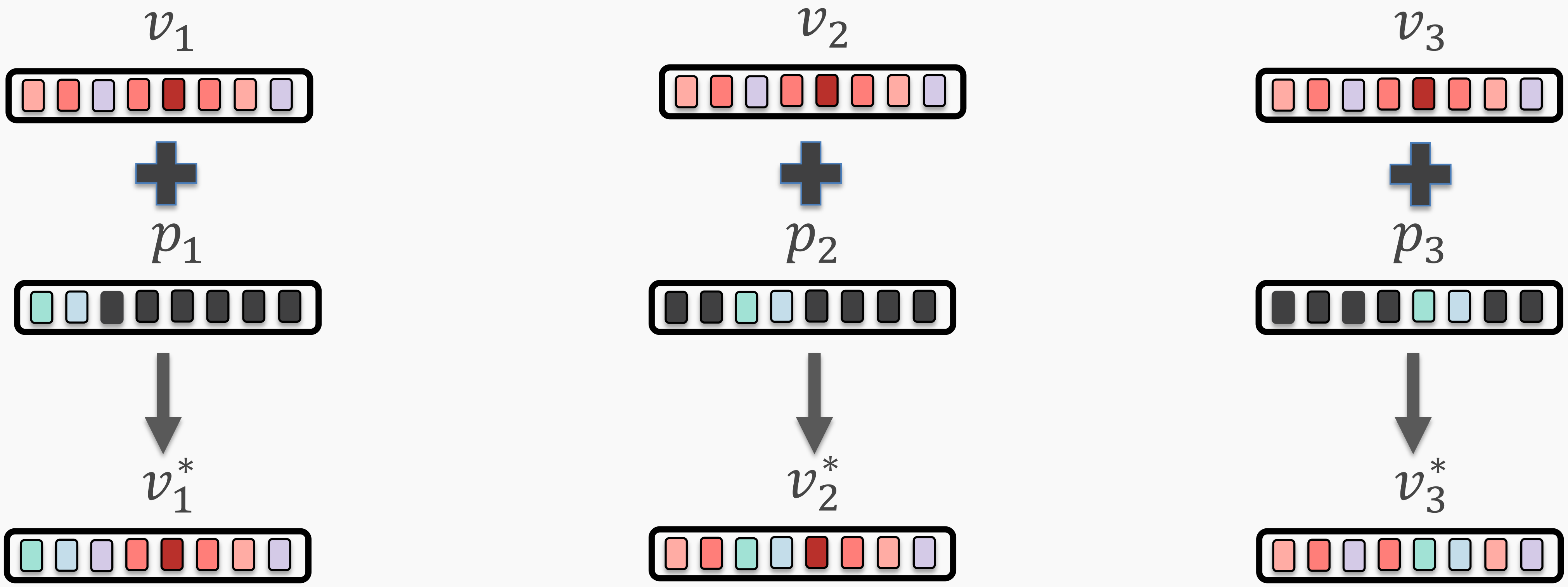
Positional Encoding

- 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 be a different value v_i^* depending on the position of the embedding in the sentence

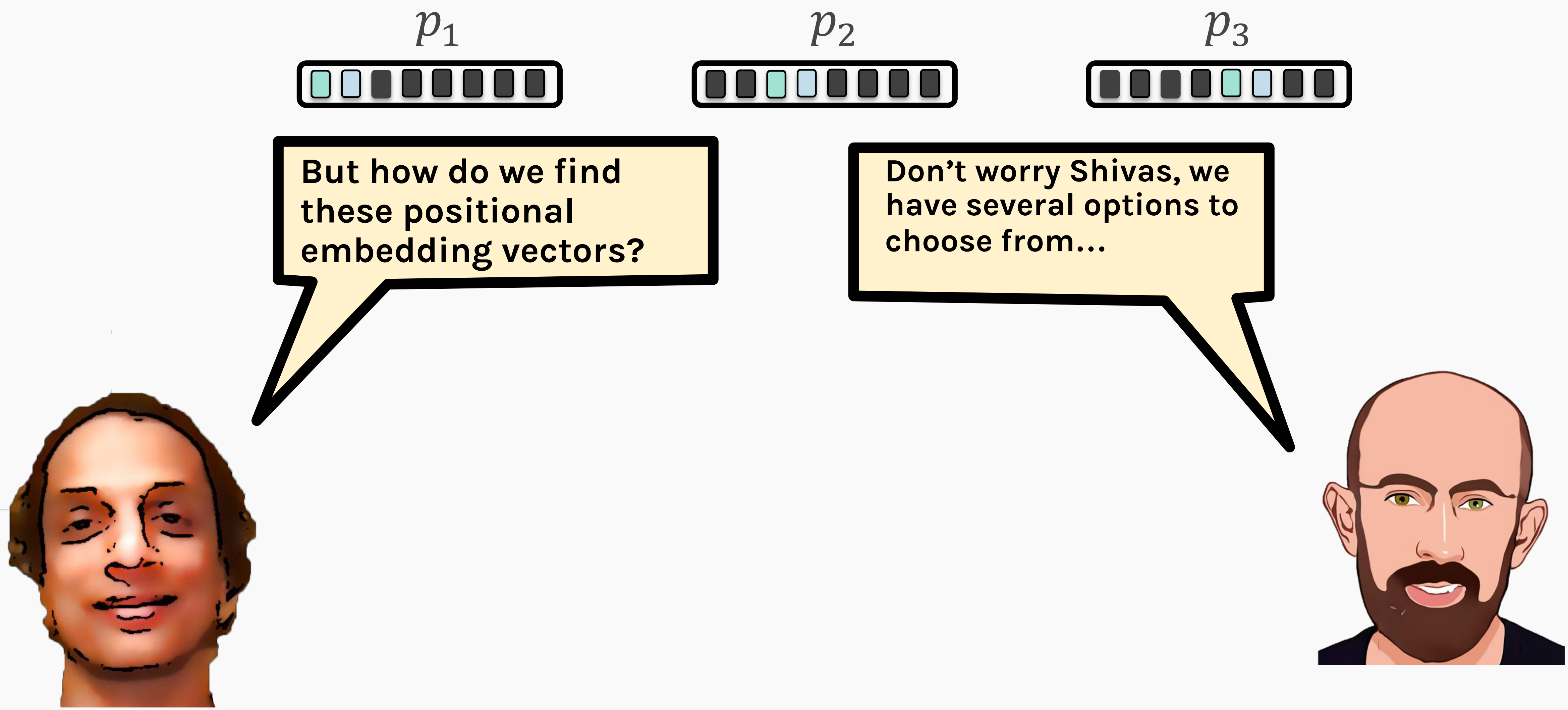


Positional Encoding



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

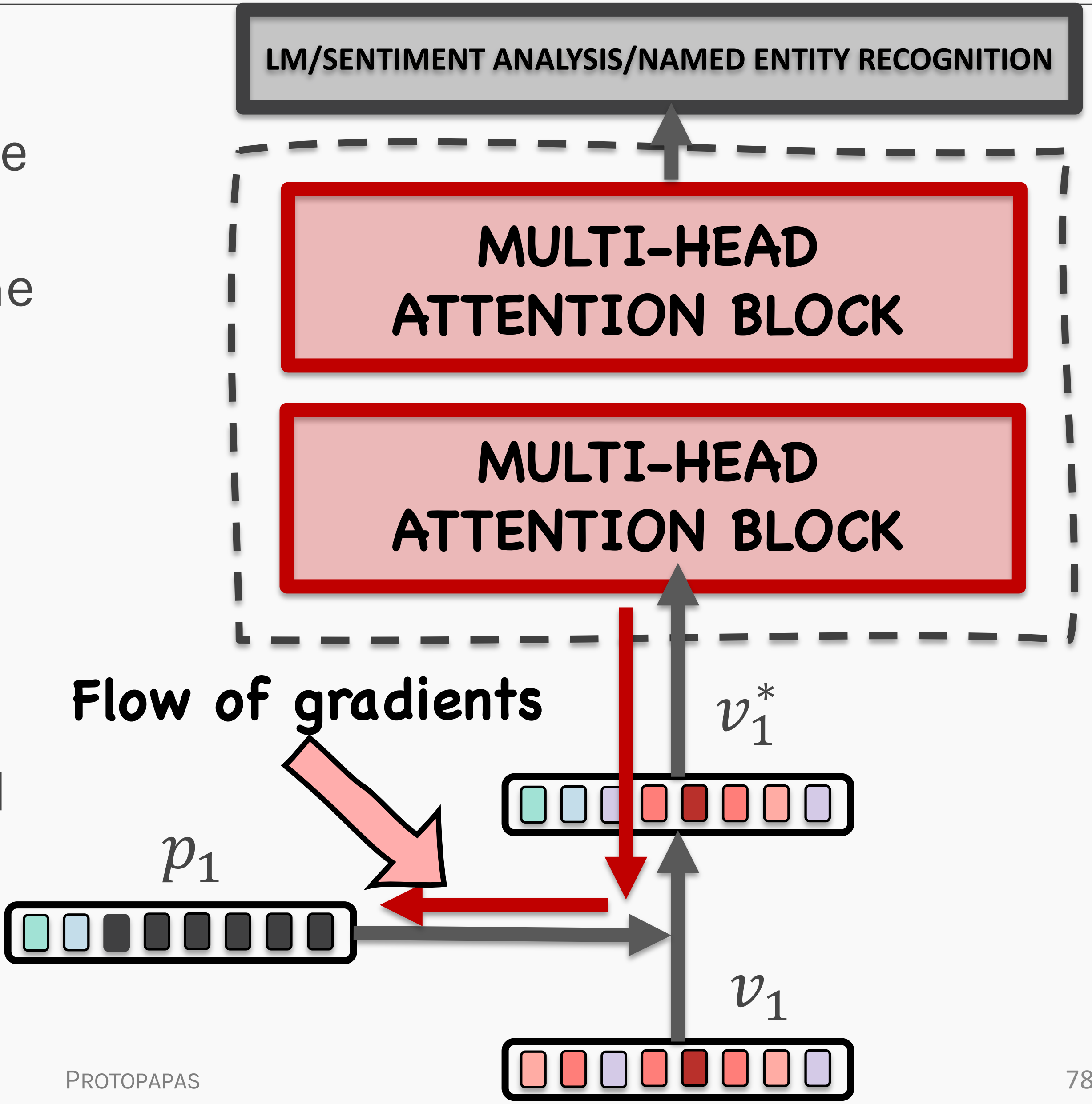
Positional Encoding



Positional Encoding

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

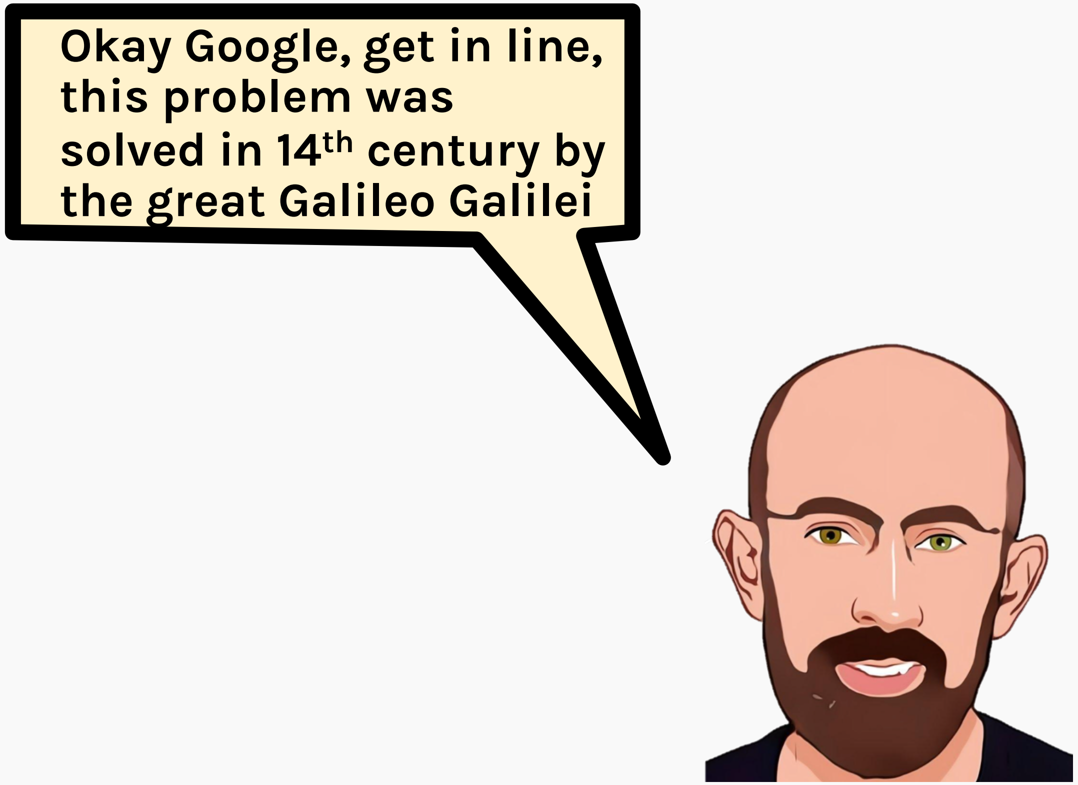
- But the number of learnable parameters in a stacked multi-head attention block can massively grow, and this can lead to poor training



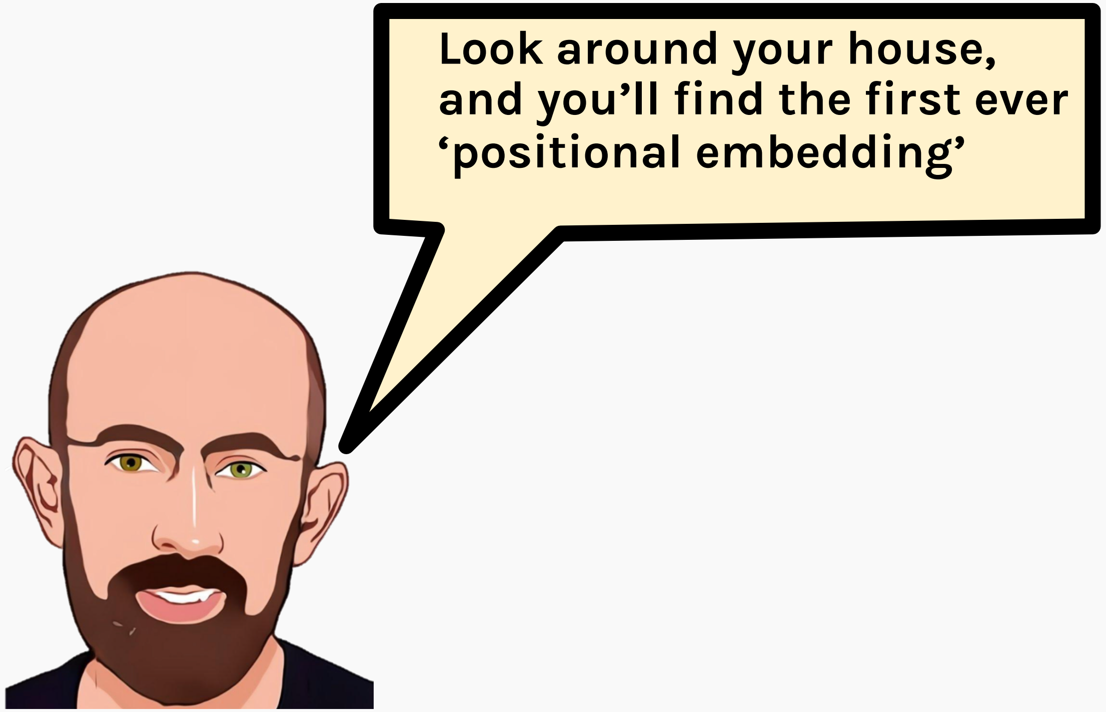
Positional Encoding

POSITIONAL EMBEDDING WISHLIST

- It should be a unique embedding for each timestep
- Relative encodings 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

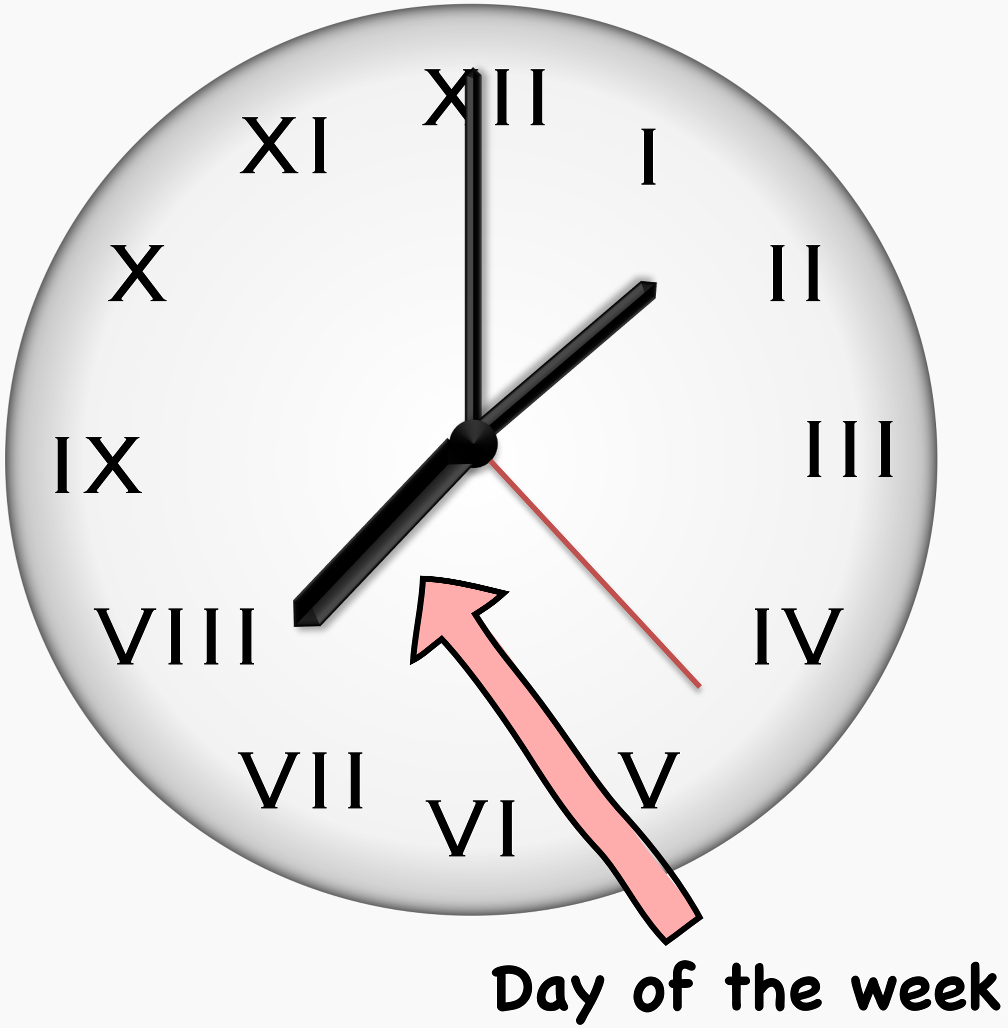
The mechanical clock

Positional Embedding – Motivation

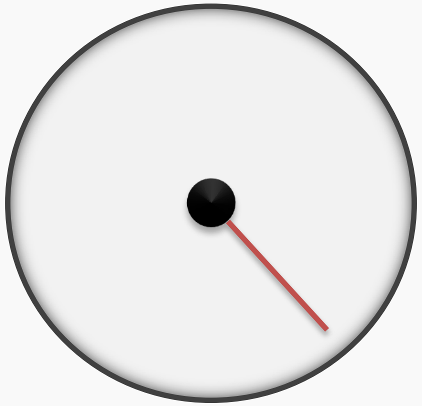
Even though the *seconds* hand repeats every minute, the overall tuple is unique

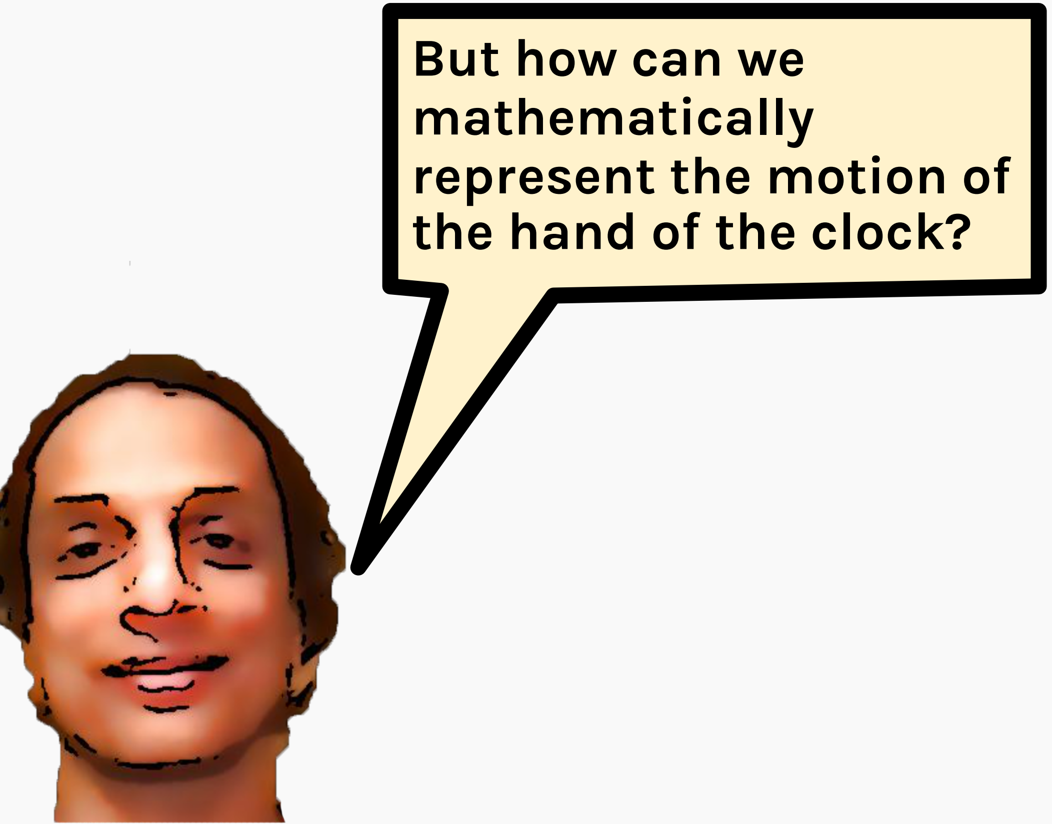
Positional Embedding – Motivation

- 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



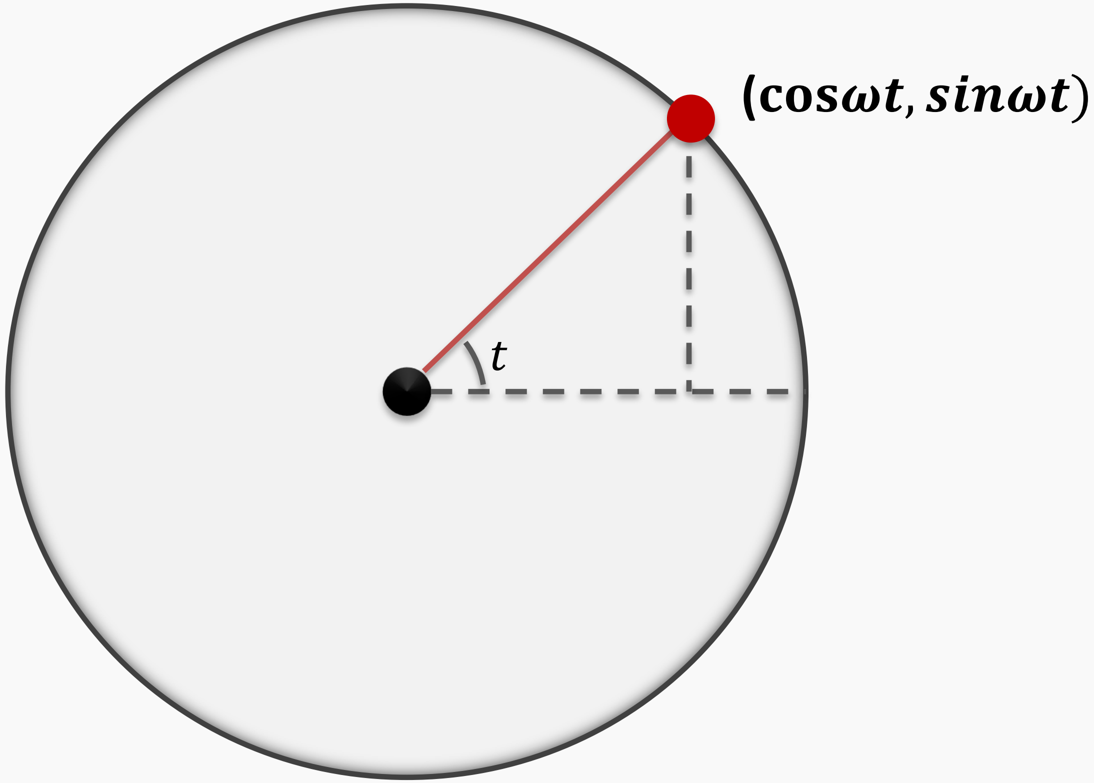
Positional Embedding - Motivation



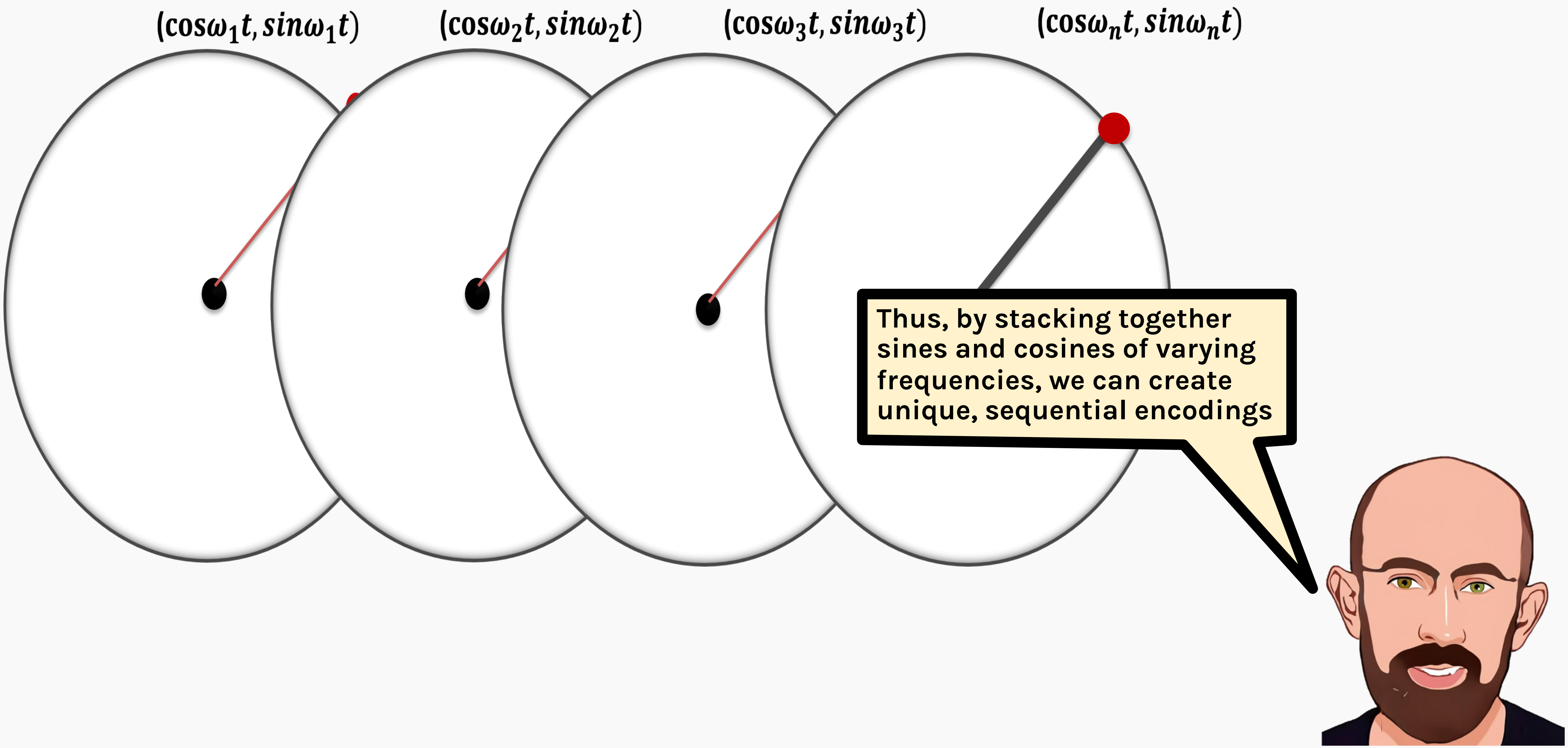


Positional Embedding - Motivation

- 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



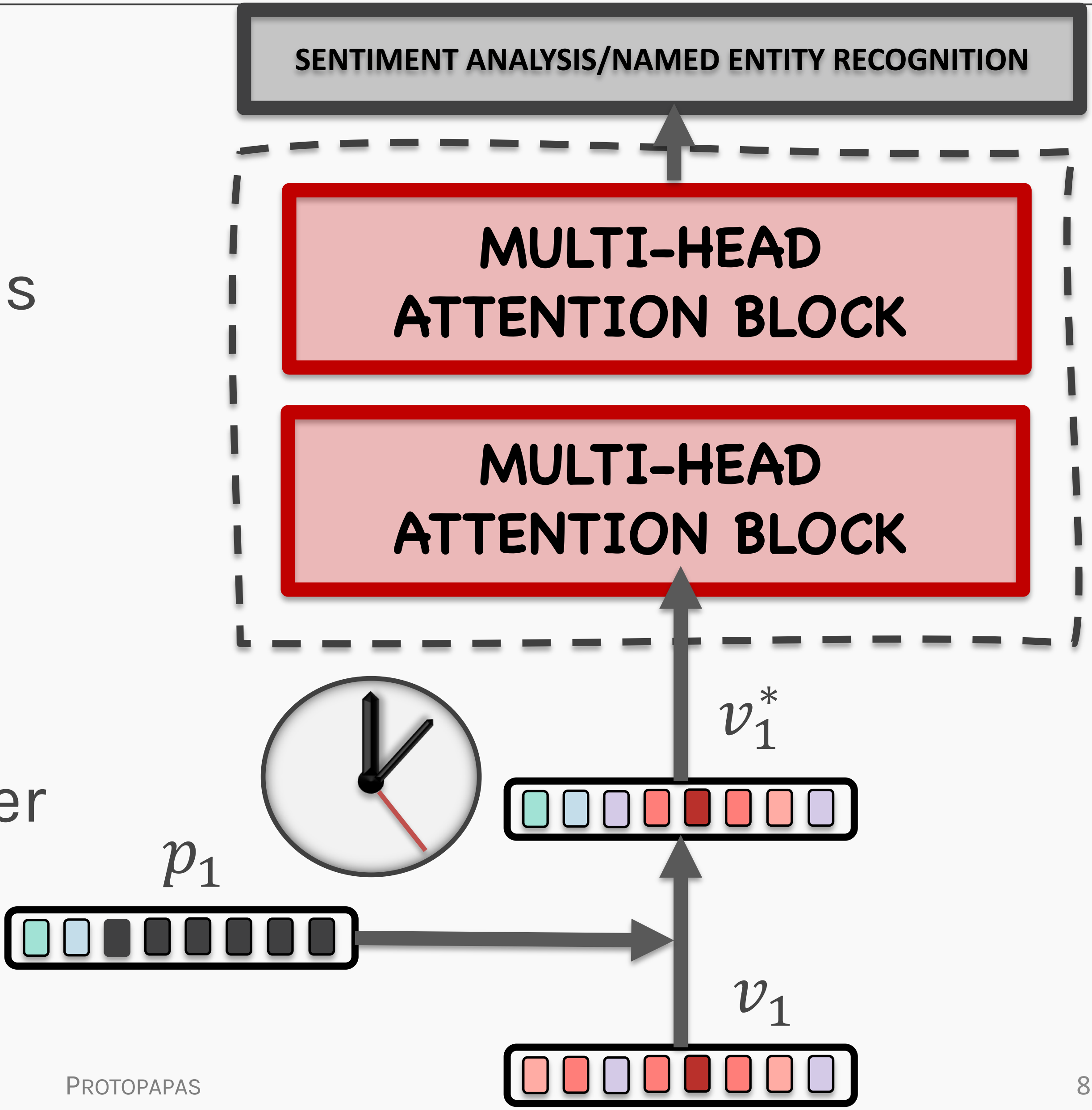
Positional Embedding - Motivation



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 more general and much faster to implement, it is usually preferred



Positional Embedding

TECHNICAL DETAILS

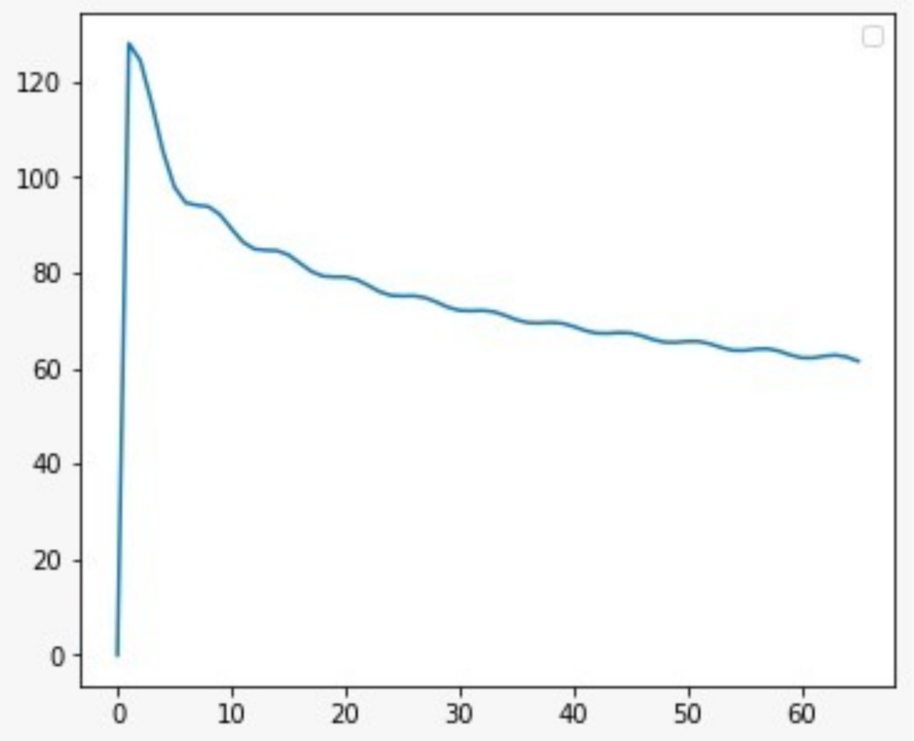
Let t be the desired position in an input sentence, $\vec{\mathrm{p}}_t \in \mathbb{R}^d$ be it's corresponding encoding and d be the encoding dimension.

Then $f : \mathbb{N} \rightarrow \mathbb{R}^d$ is the function that produces $\vec{\mathrm{p}}_t$ and is defined as:

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

Where,

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

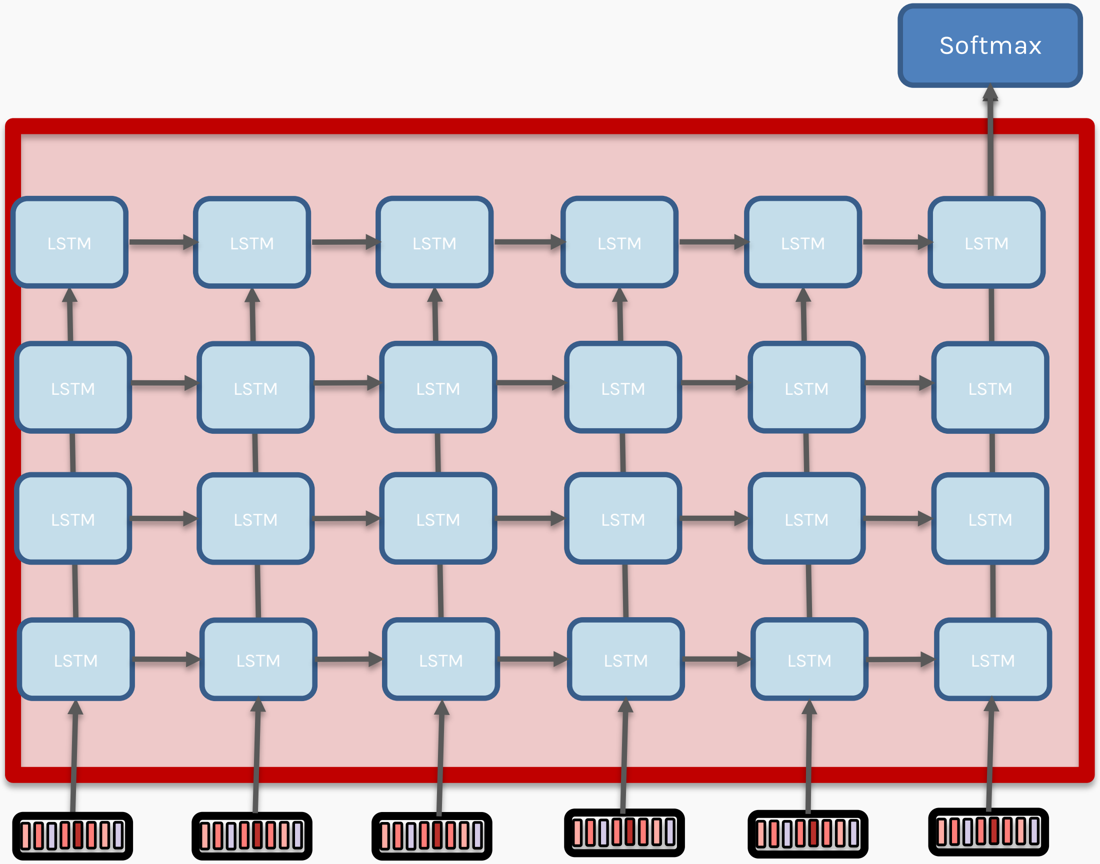


X-axis is time (position), y-axis is w_k

Bringing it all together

Comparison – RNN v/s Transformer

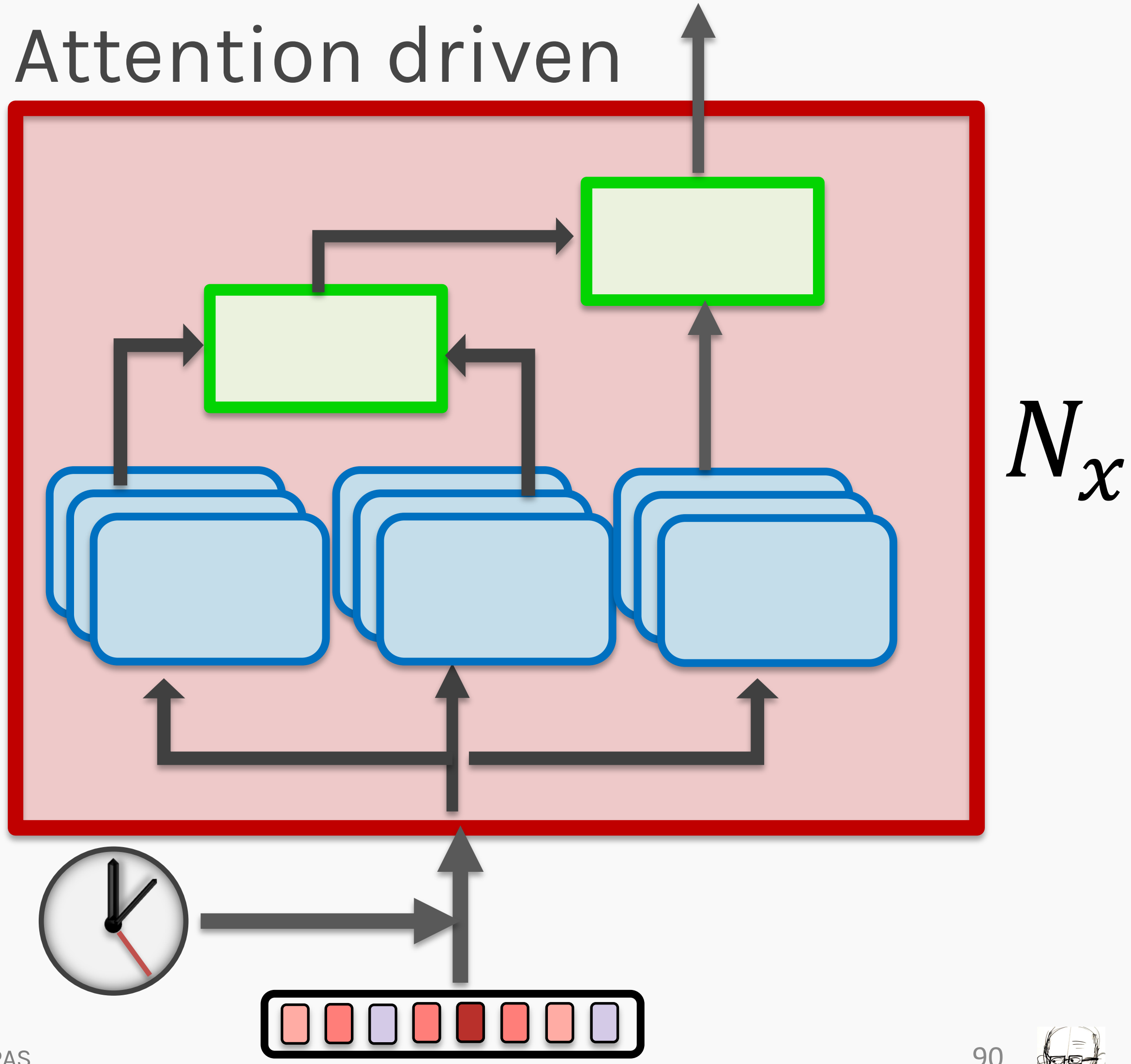
BEFORE



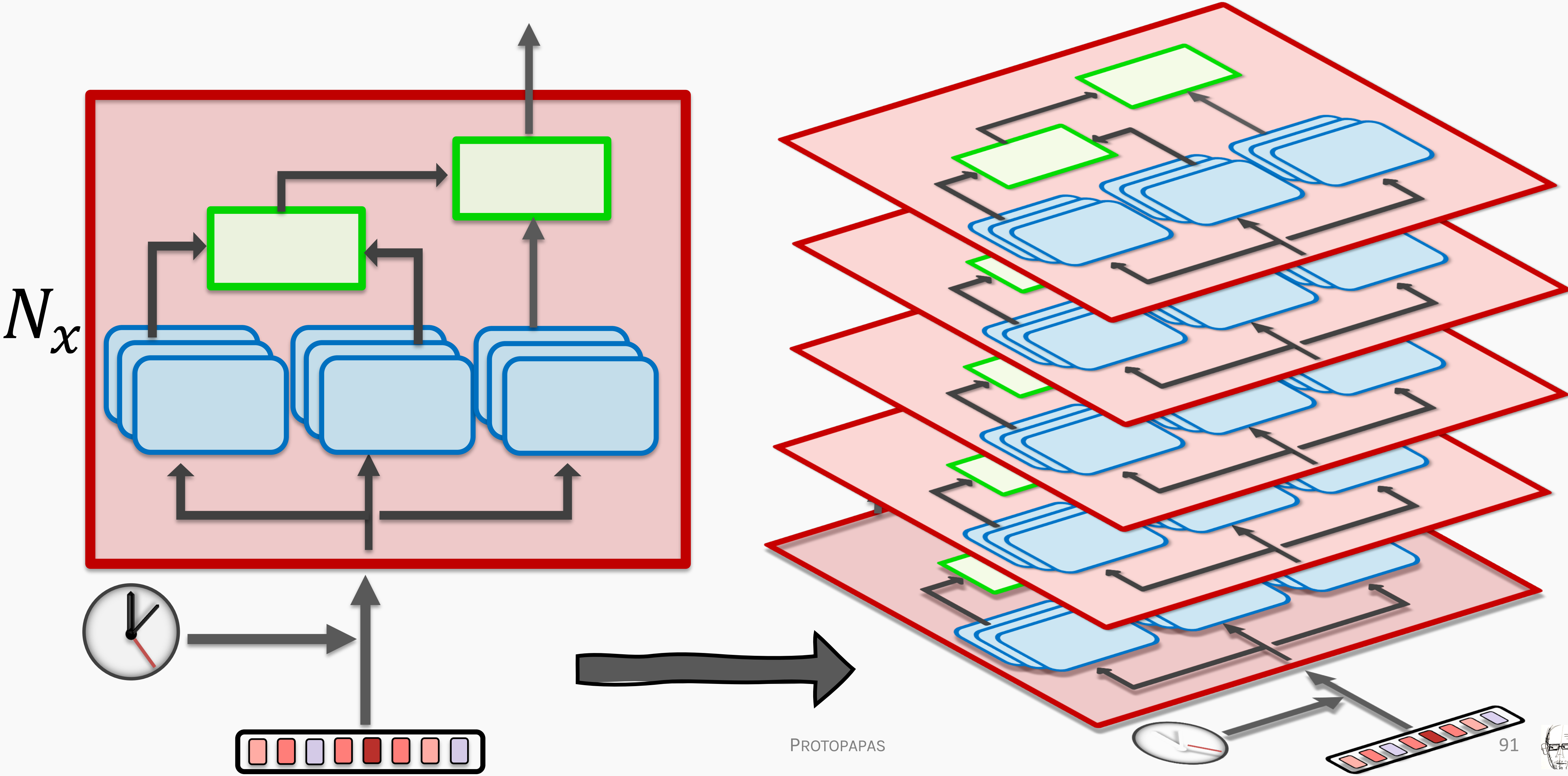
Recurrent units

AFTER

Attention driven



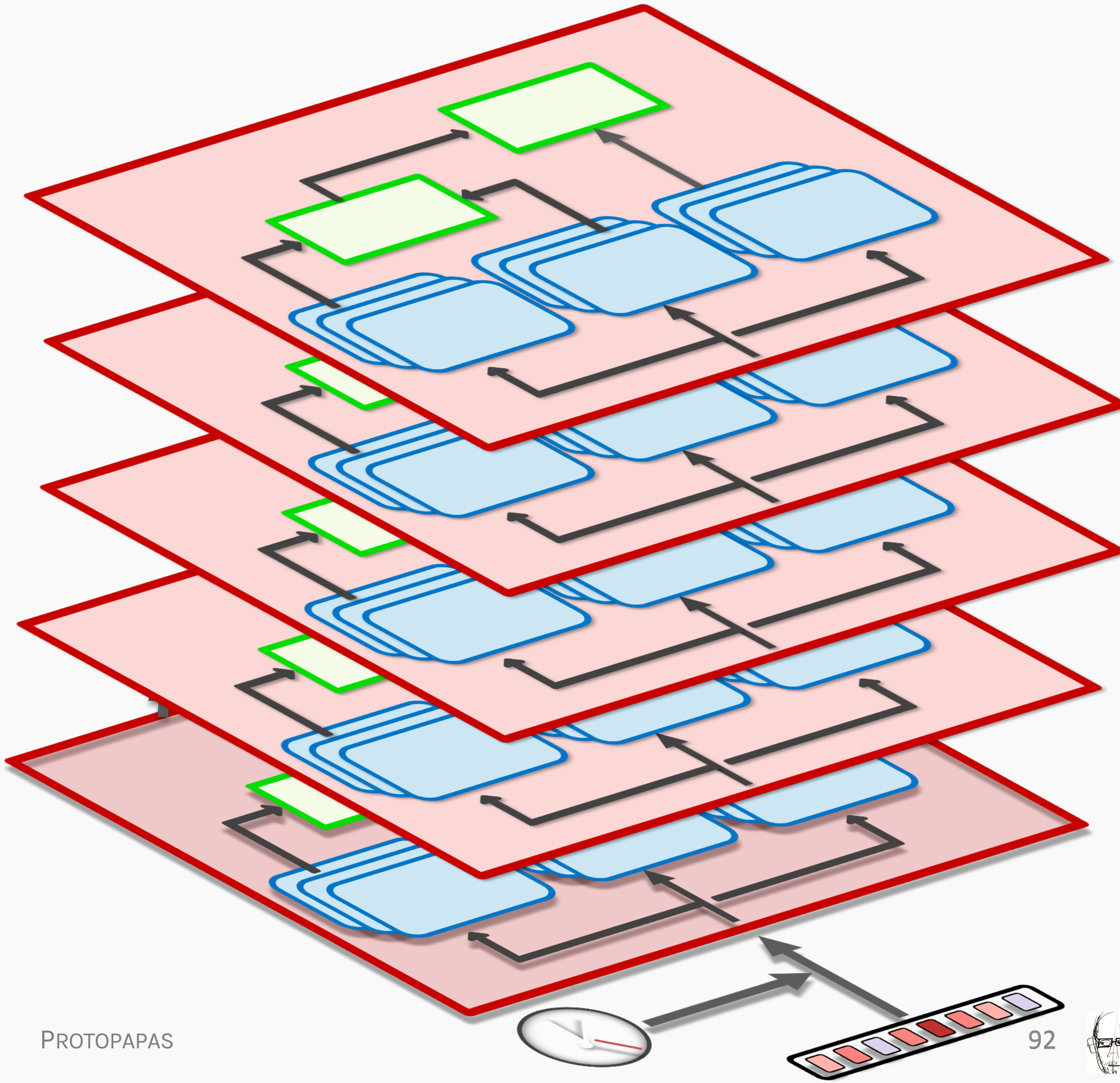
Transformer as a 3-D model



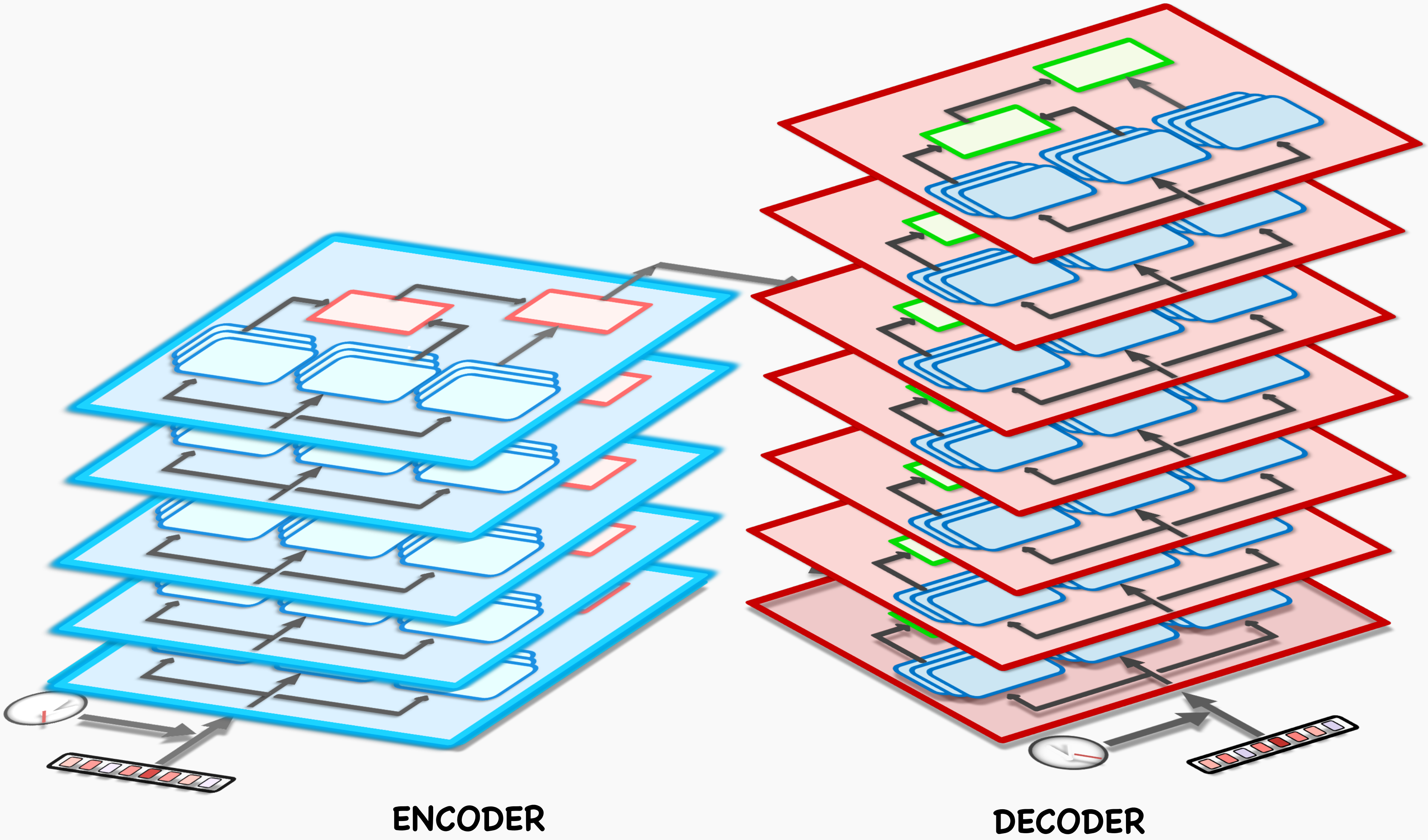
Transformers - Summary

Language Model Wishlist?

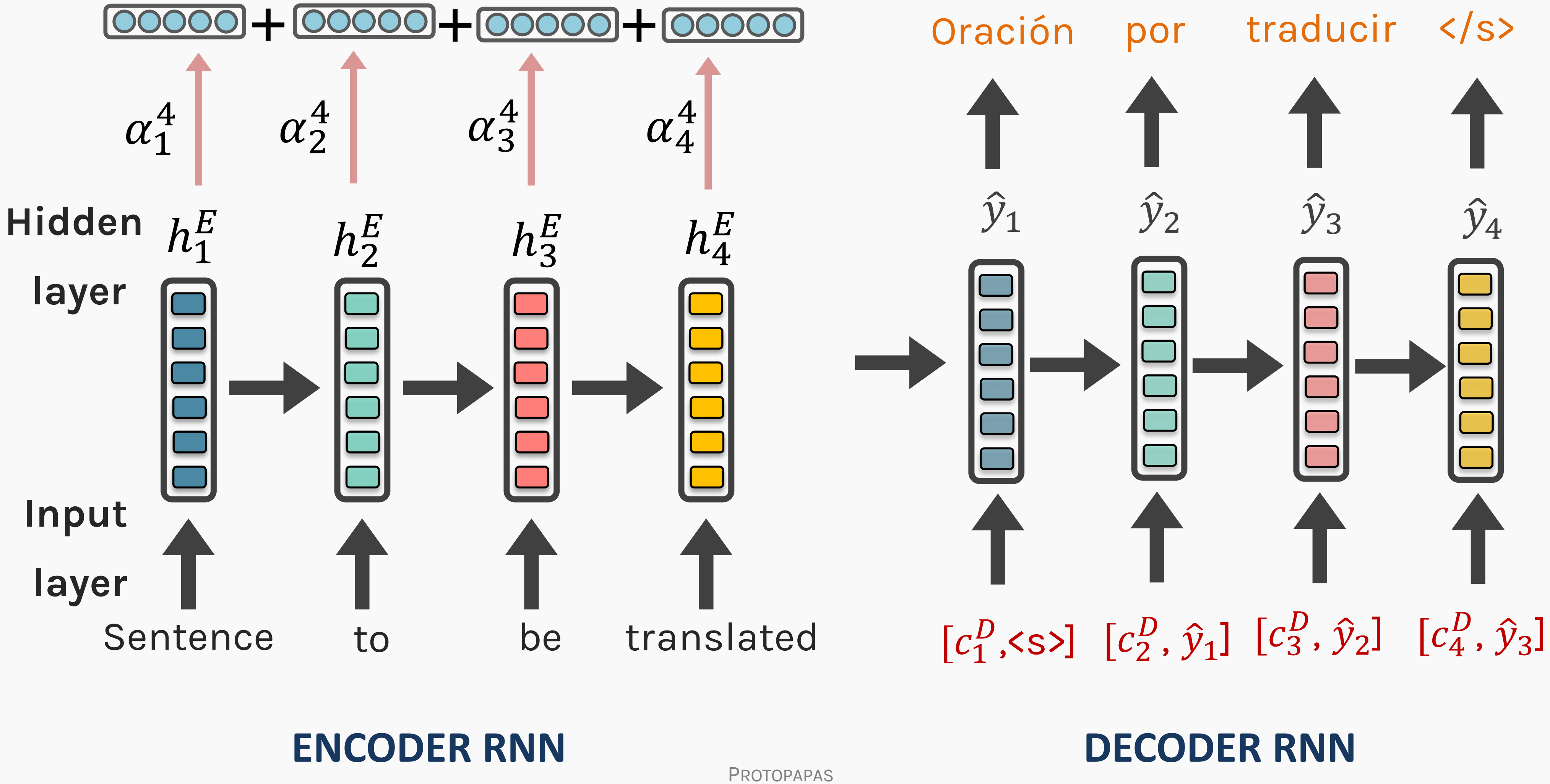
- ☑ We want to have strong contextual relations between words - **DONE**
- ☑ We want words to have sequential information - **DONE**
- ☑ We need an architecture that can be trained in parallel (non-Markovian property) - **DONE**



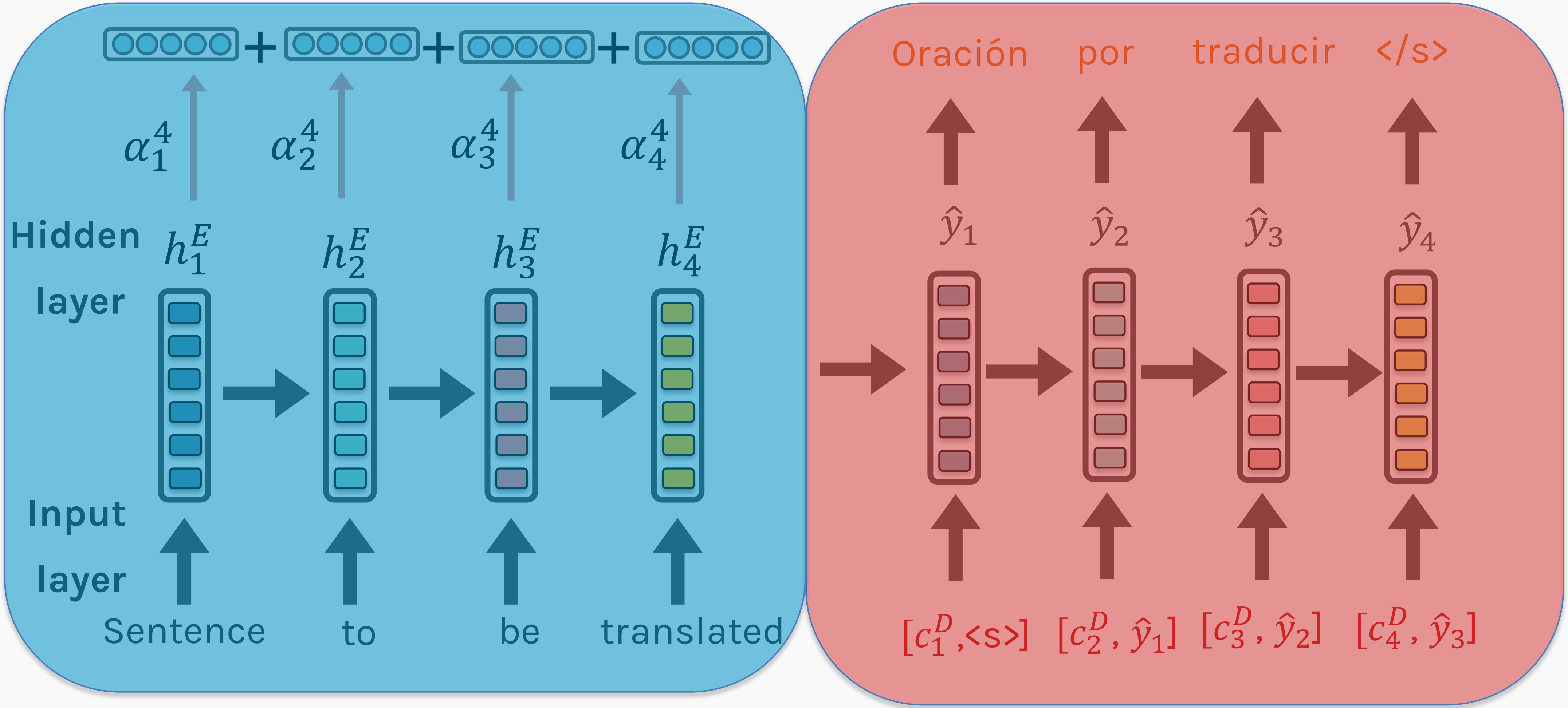
Transformers - Summary



RECAP: Seq2Seq + Attention



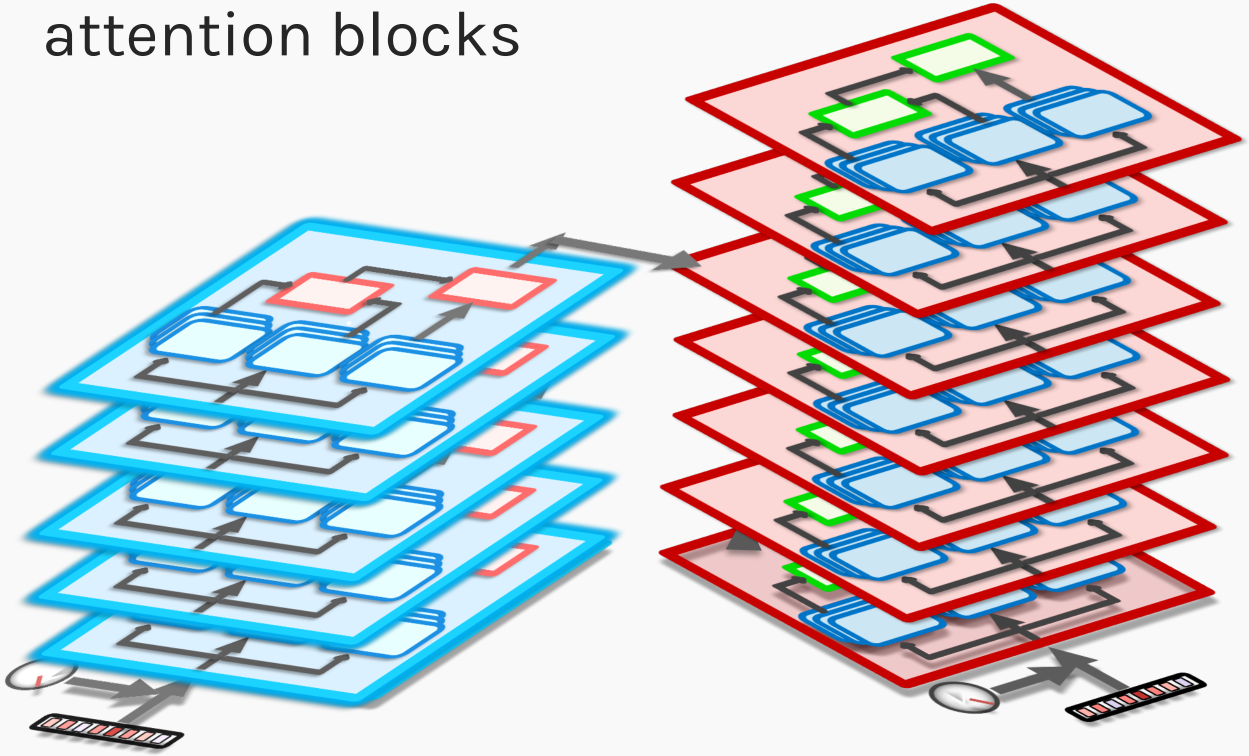
RECAP: Seq2Seq + Attention

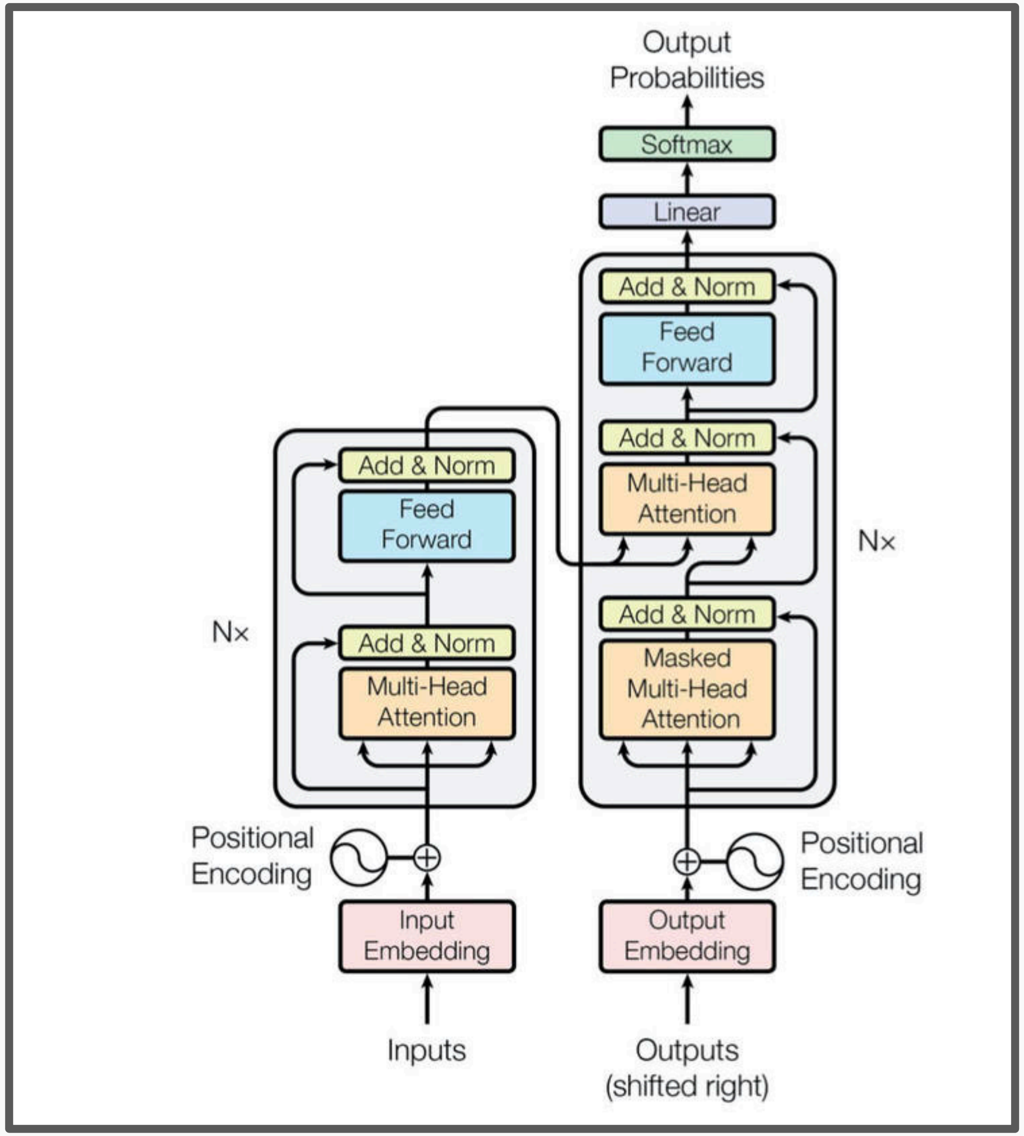


ENCODER RNN **DECODER RNN**

Transformers - Summary

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

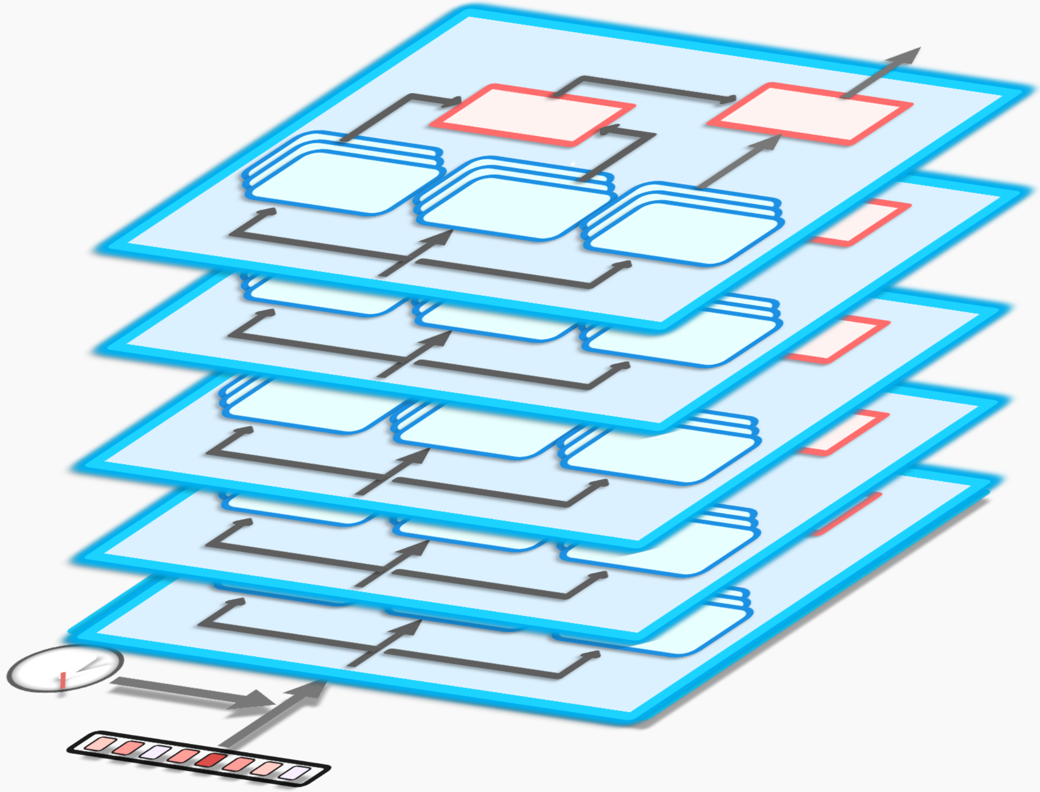




From Transformers to BERT

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

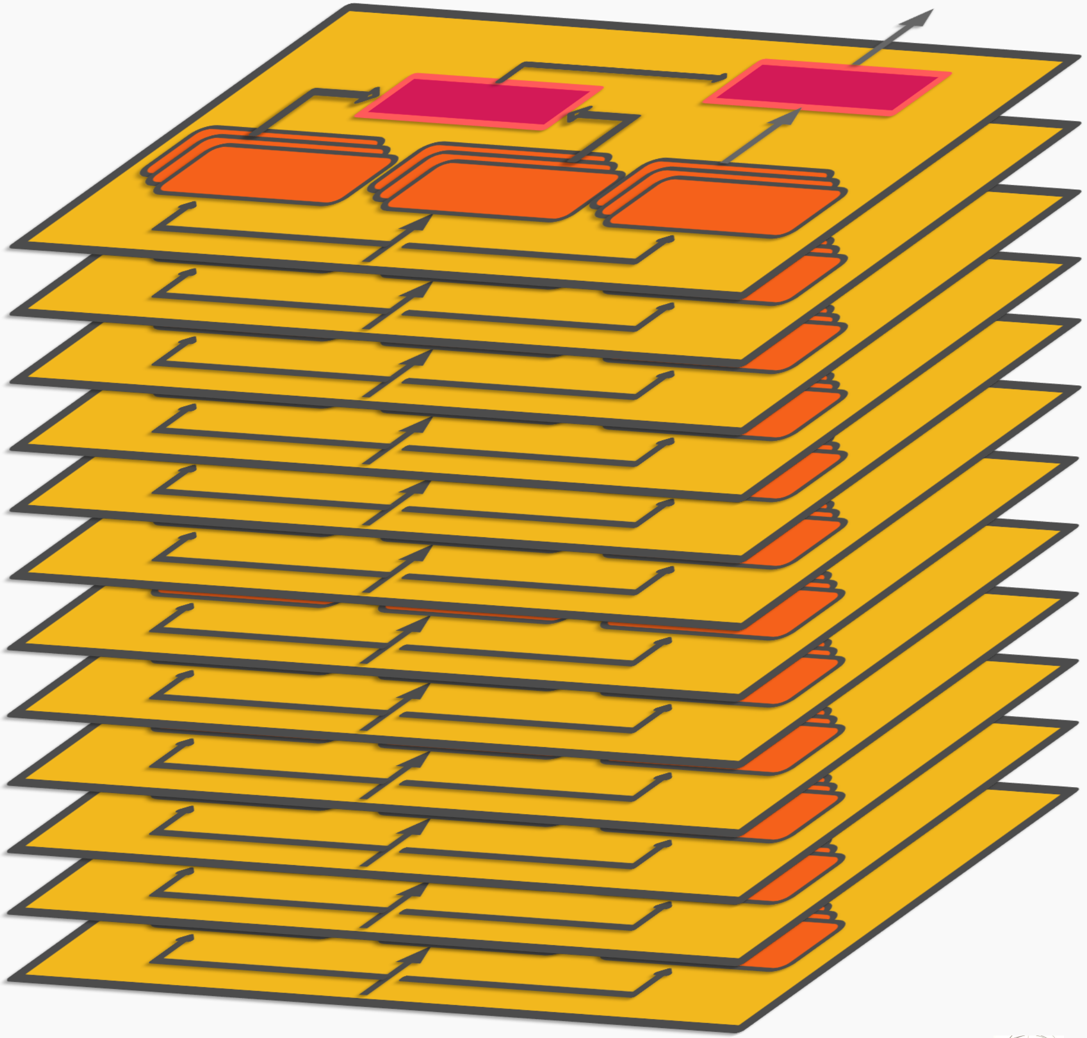
- This led to the new architecture called Bidirectional Encoder Representations from Transformers, or more commonly known as BERT



Bidirectional Encoder Representations from Transformers (BERT)

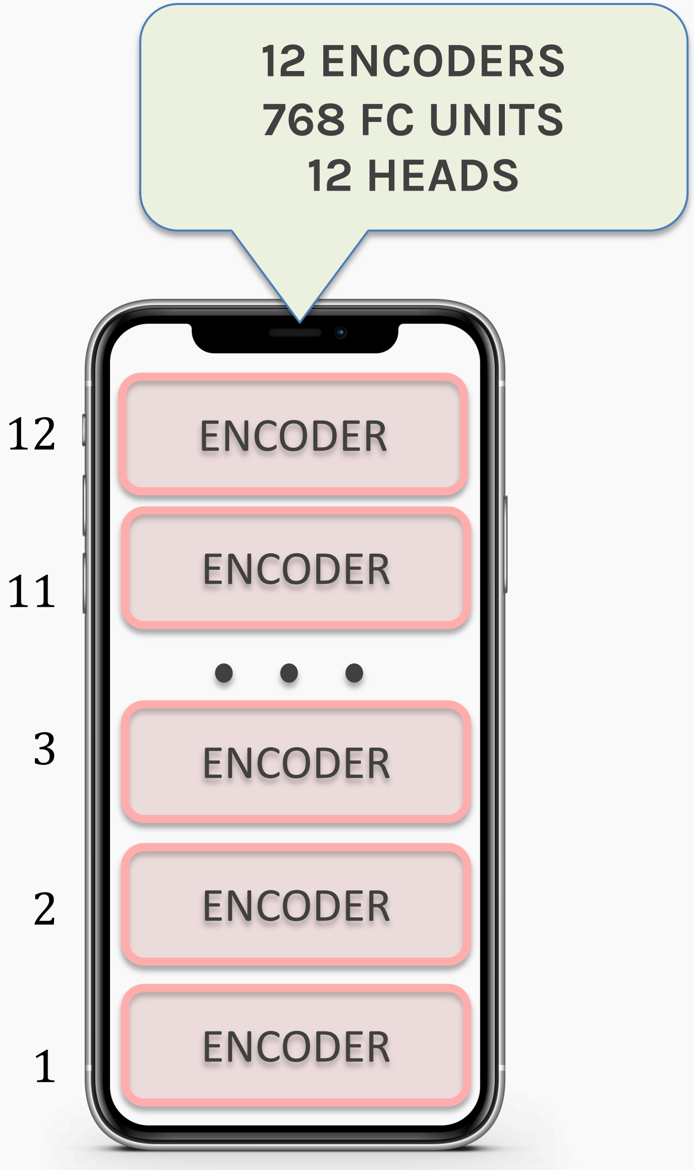
BERT

- The BERT model broke several records across various language model tasks
- The source-code and pre-trained BERT models are all open-source and can be easily adapted to individual tasks
- This makes BERT a prime candidate for transfer learning applications

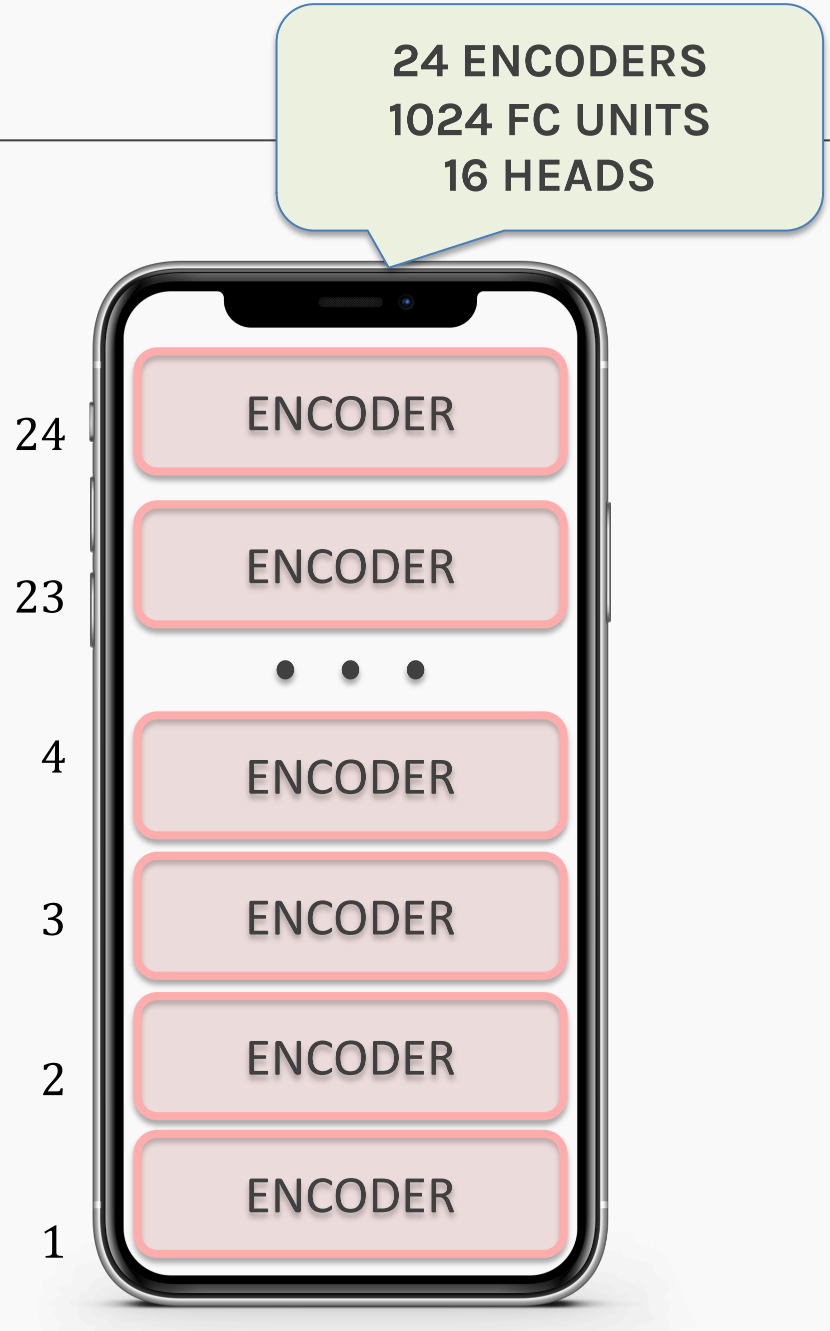


Flavors of BERT

Flavors of BERT



BERT BASE



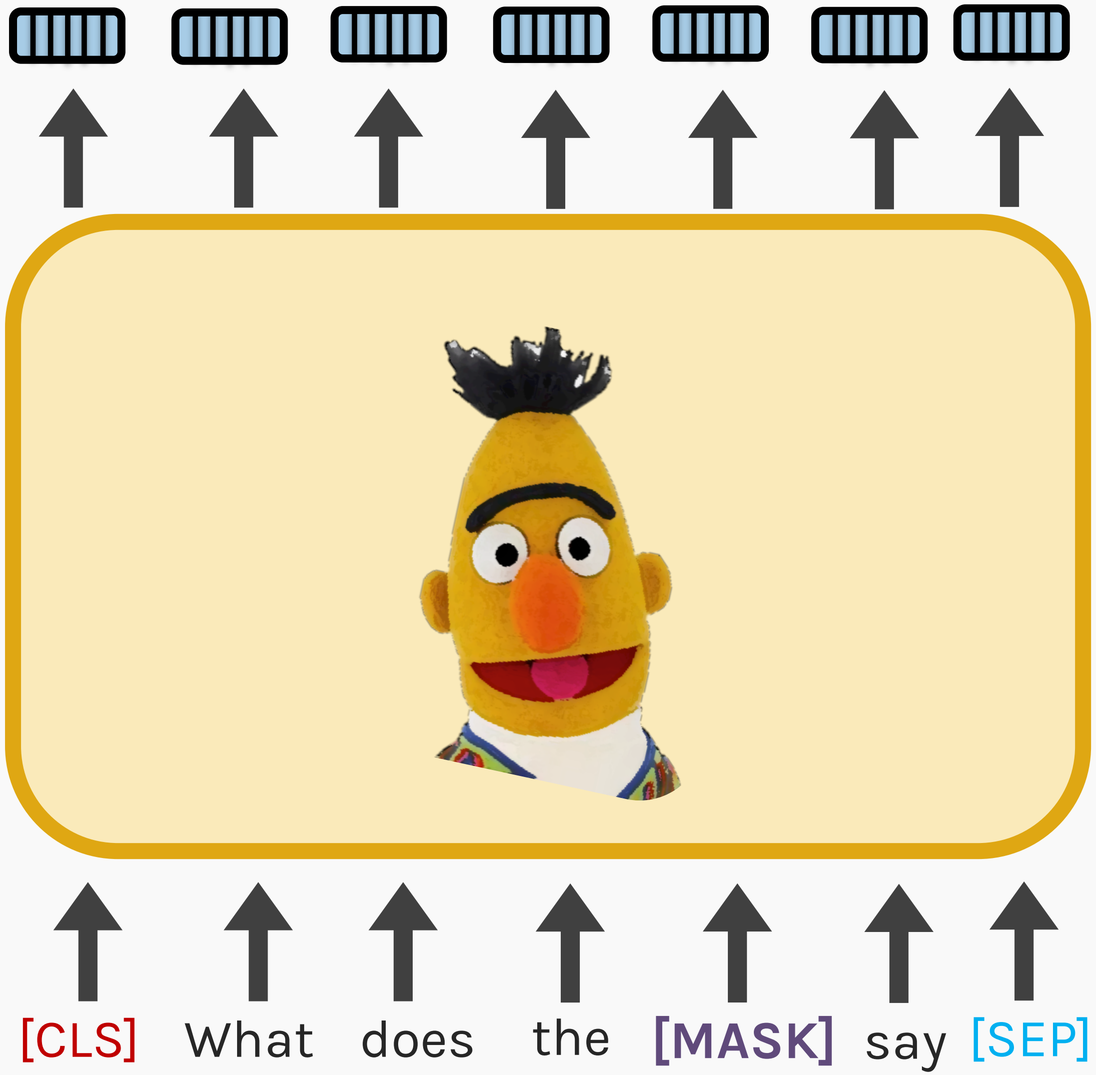
BERT LARGE

BERT

TECHNICAL DETAILS

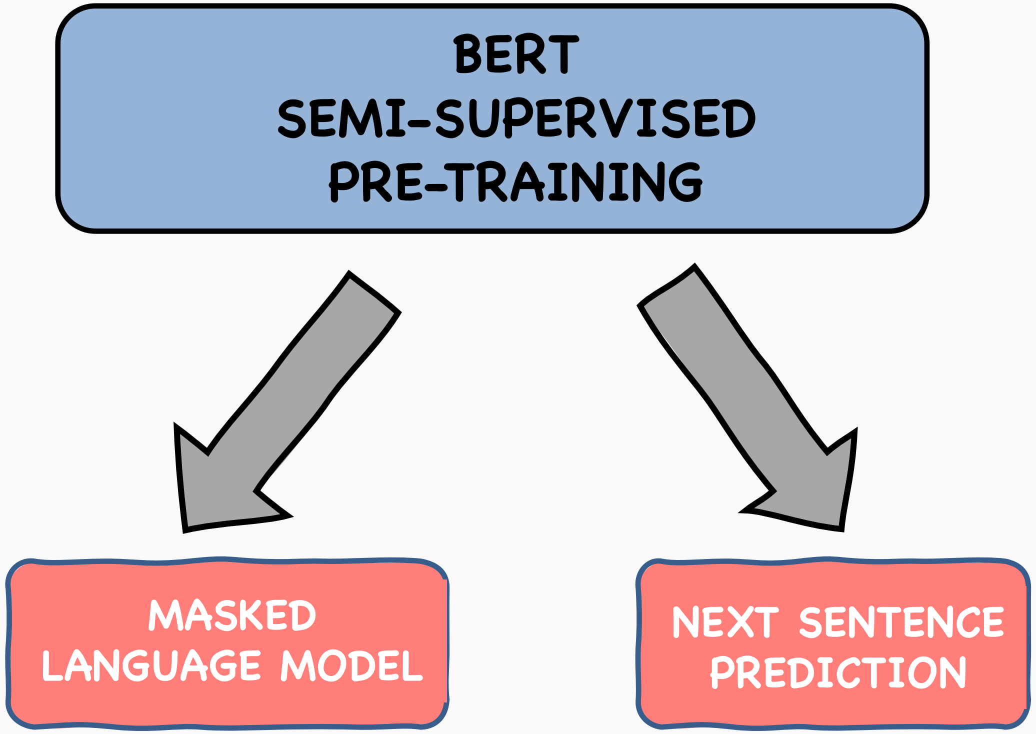
- Input sequences to BERT must have the following special tokens:
  - [MASK] – Masking token
  - [SEP] – Separator token
  - [CLS] – Classifier Token

What are these new tokens you ask? Hold on!



How to train a BERT?

How to train a BERT



How to train a BERT?

TRAINING ISSUES?

- If BERT takes the entire sequence as an input, how can we train it as a language model?

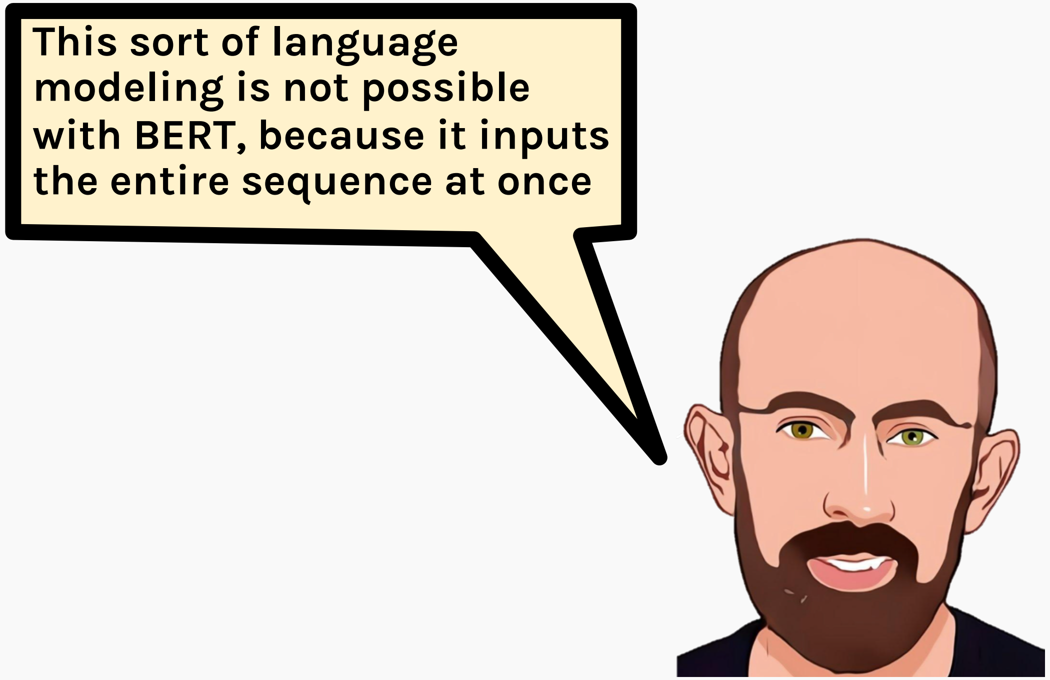
- How could we classify if two chosen sentences follow each other or are out of place if the output is also a sequence of tokens?

MASKED LANGUAGE MODELING

RECAP: Language Model

Shivas was hit by a bus as he was crossing the street

Shivas was hit by a ___

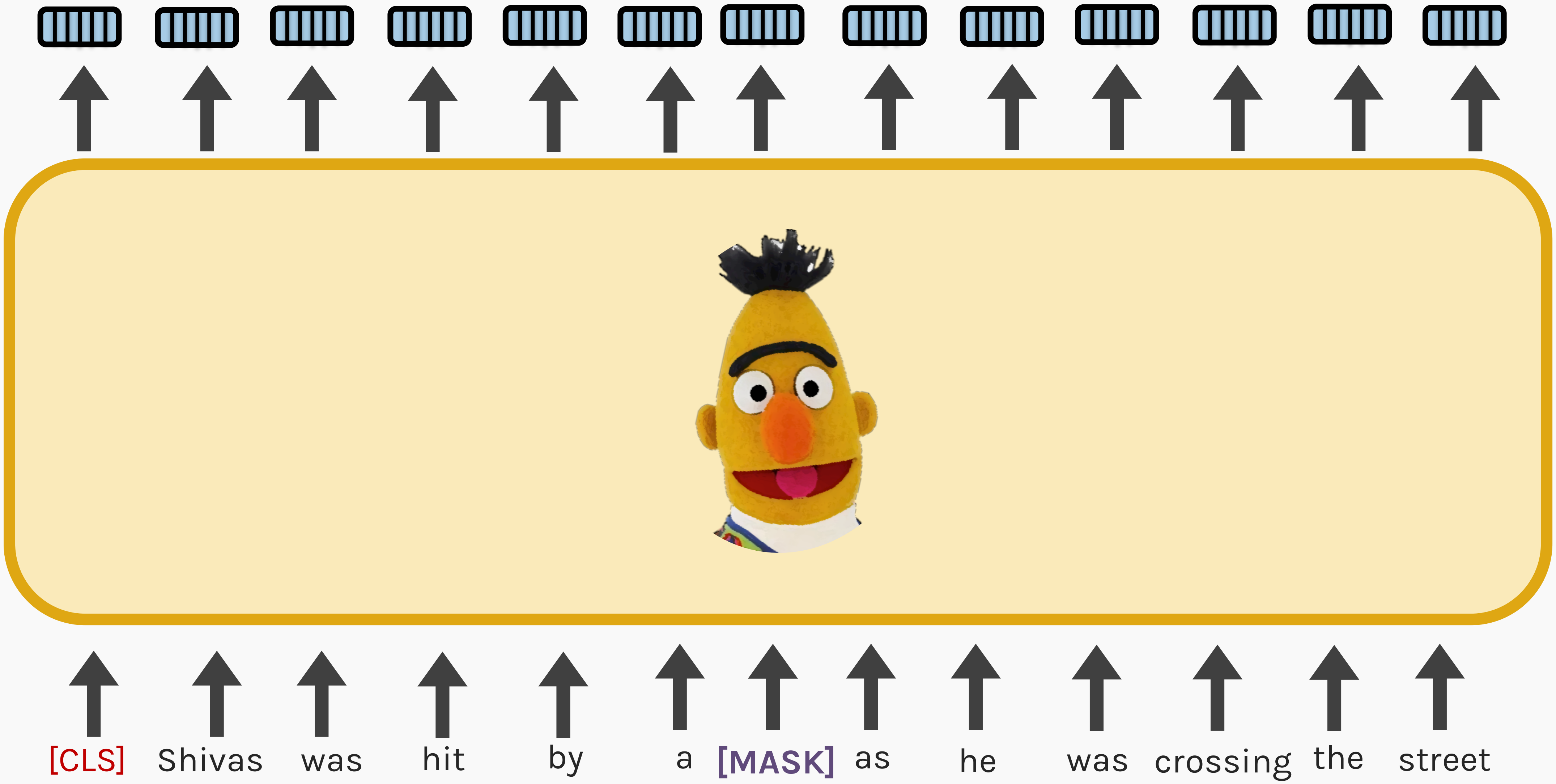


Masked Language Model

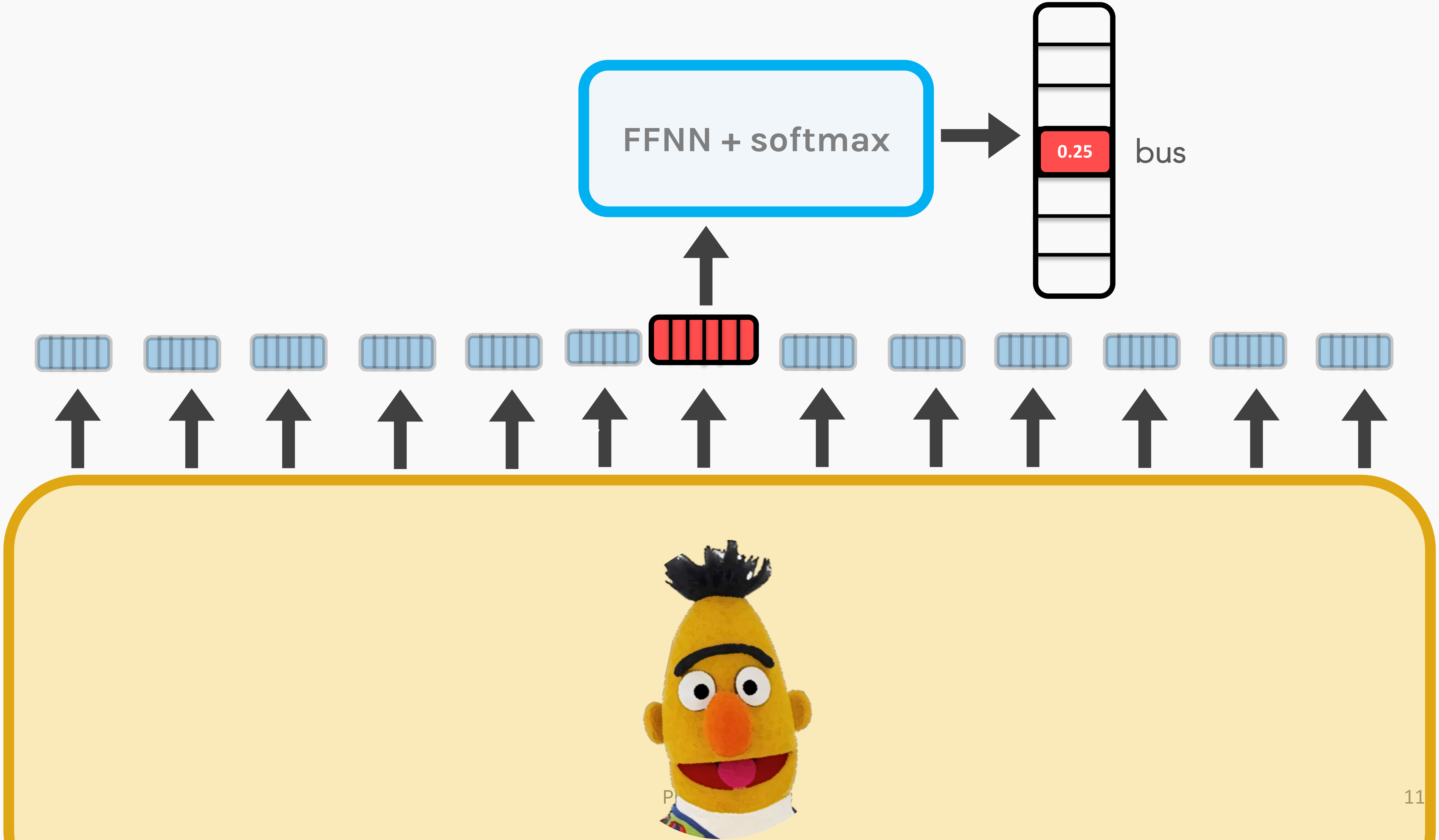
Shivas was hit by a bus as he was crossing the street

Shivas was hit by a [MASK] as he was ...

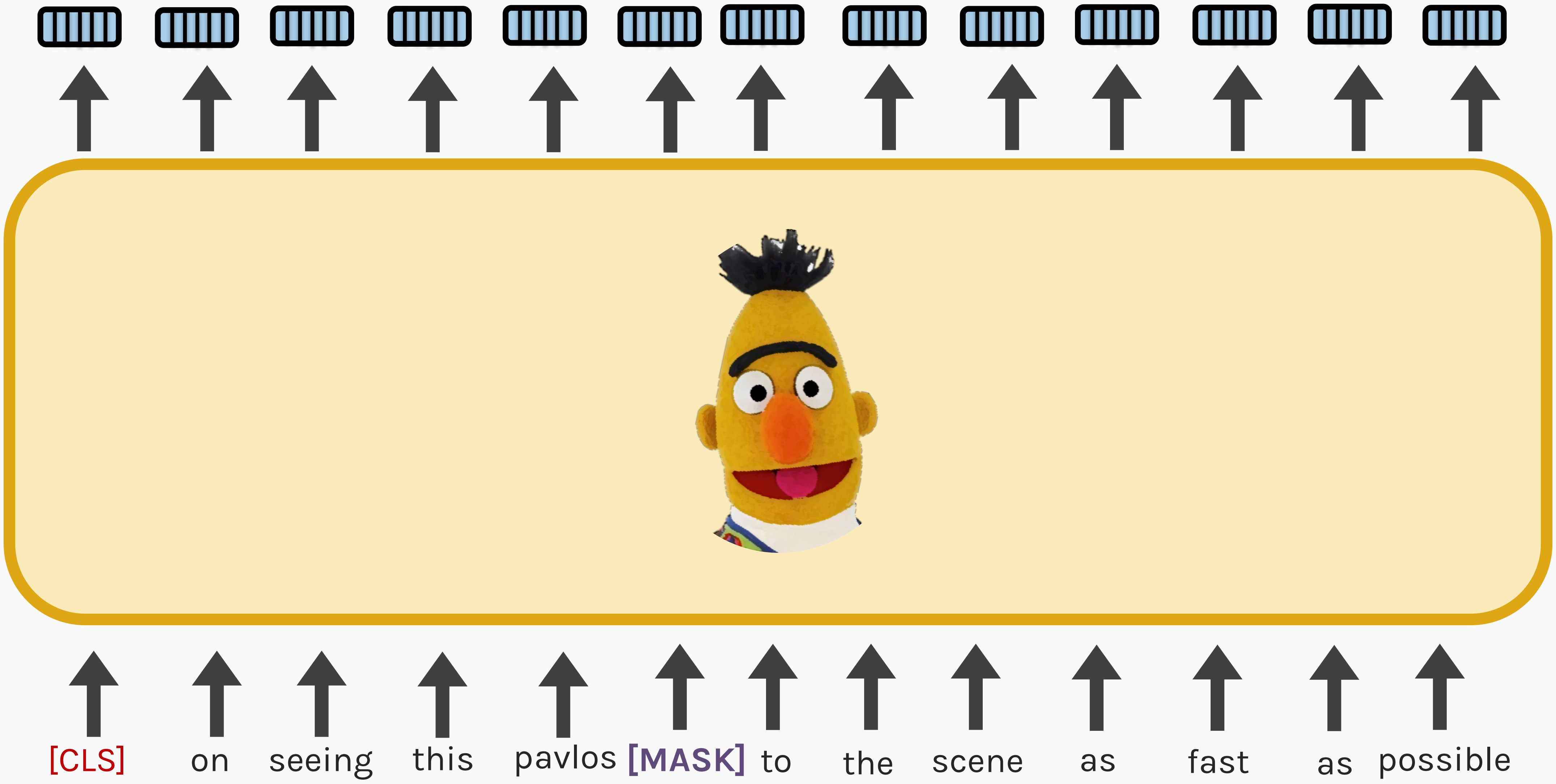
Shivas was hit by a **bus** as he was crossing the street



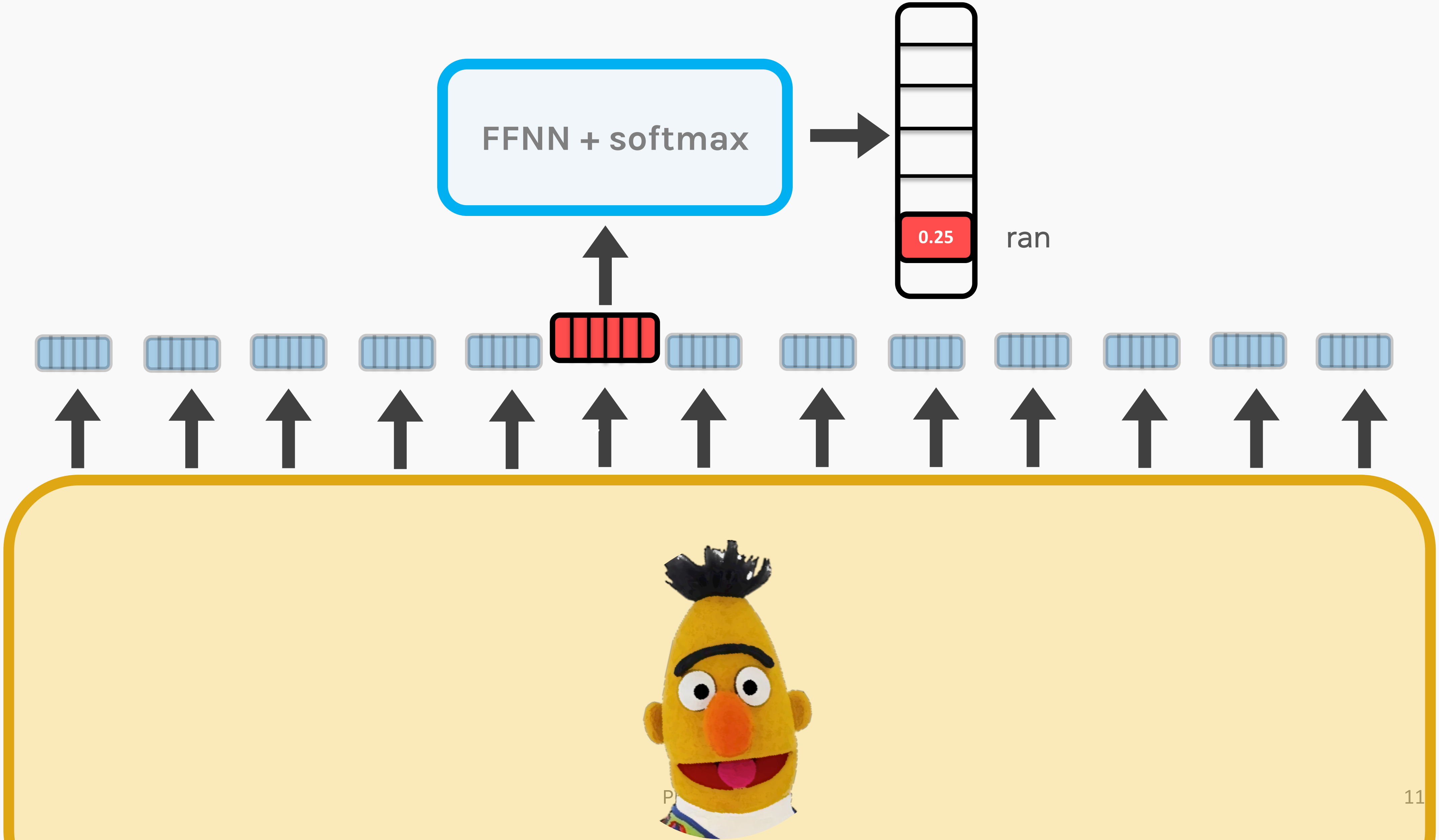
Shivas was hit by a bus as he was crossing the street



On seeing this, Pavlos ran to the scene as fast as possible



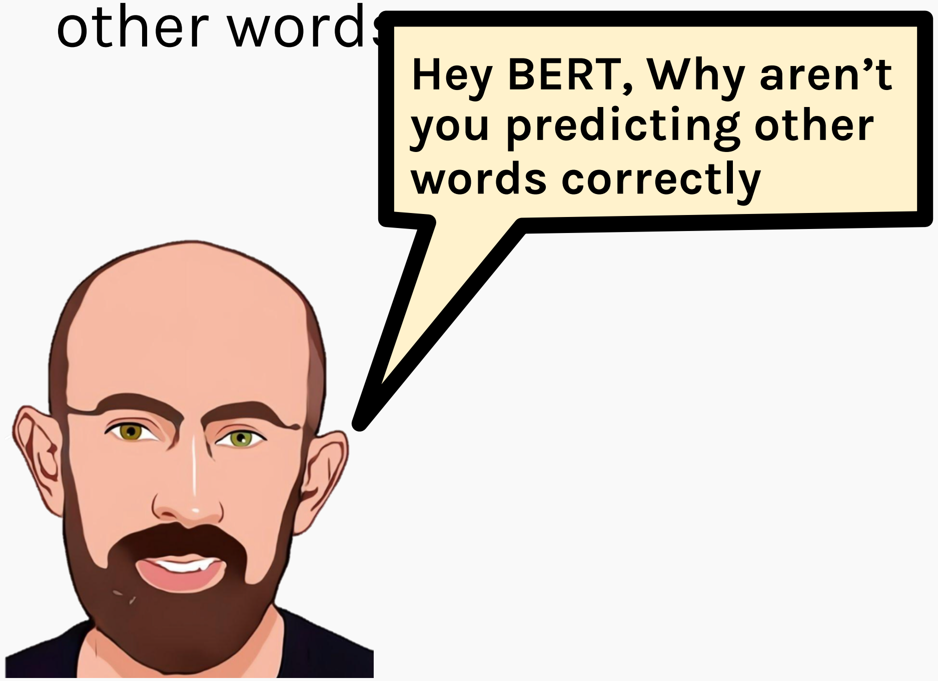
On seeing this, Pavlos ran to the scene as fast as possible

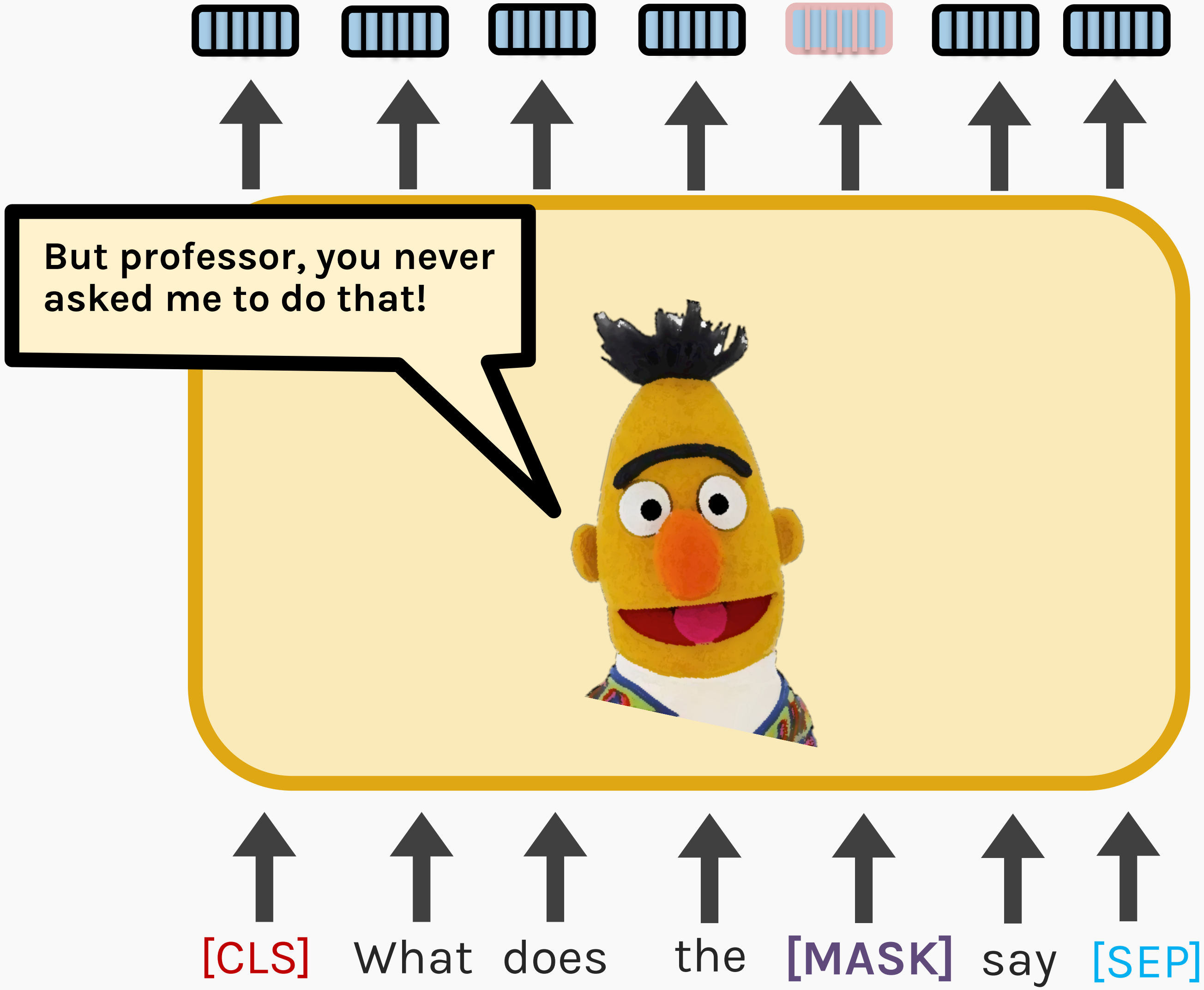


BERT

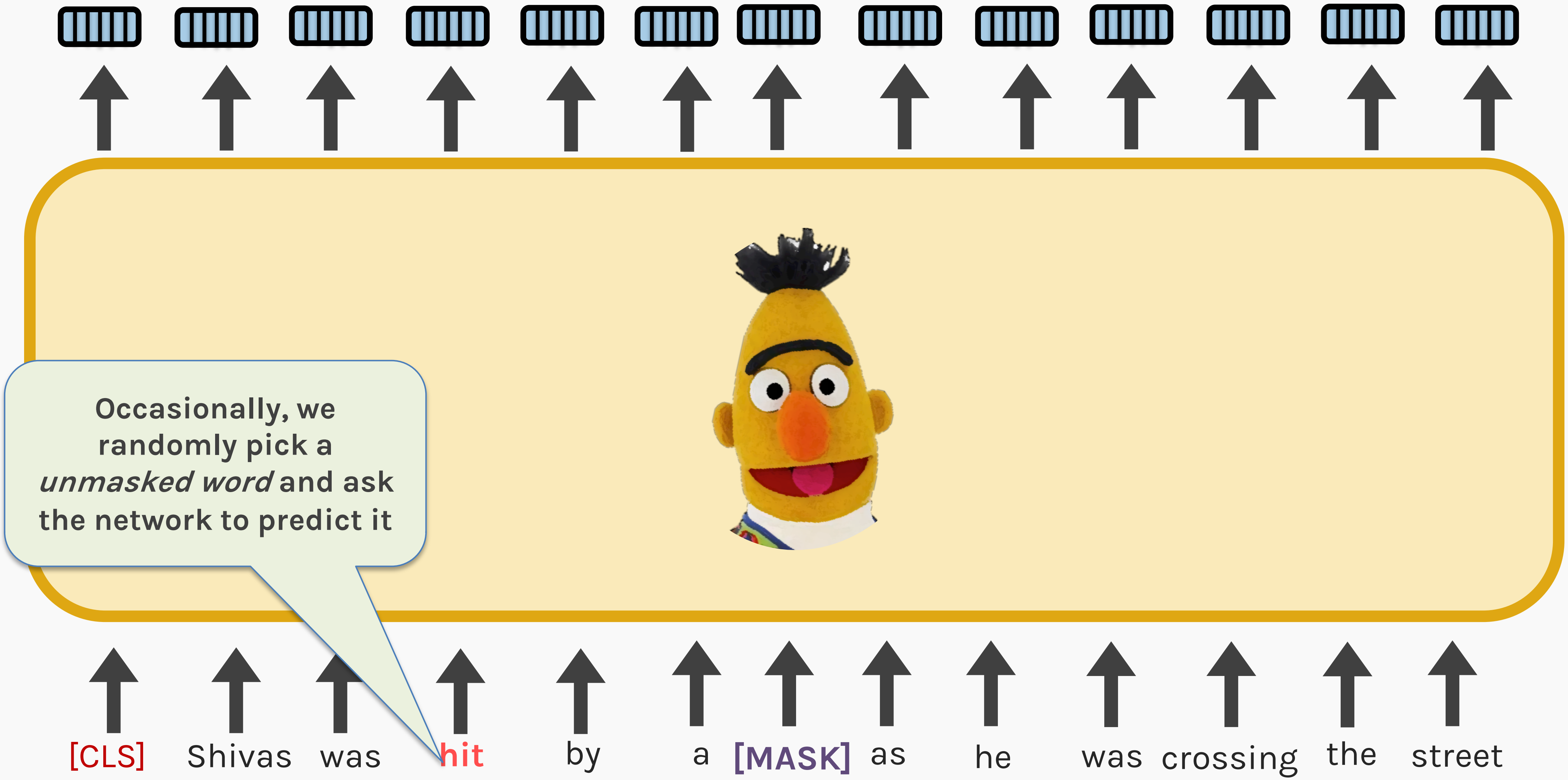
MASKING ISSUES?

- The model learns how to correctly predict the *masked* words but completely ignores other words

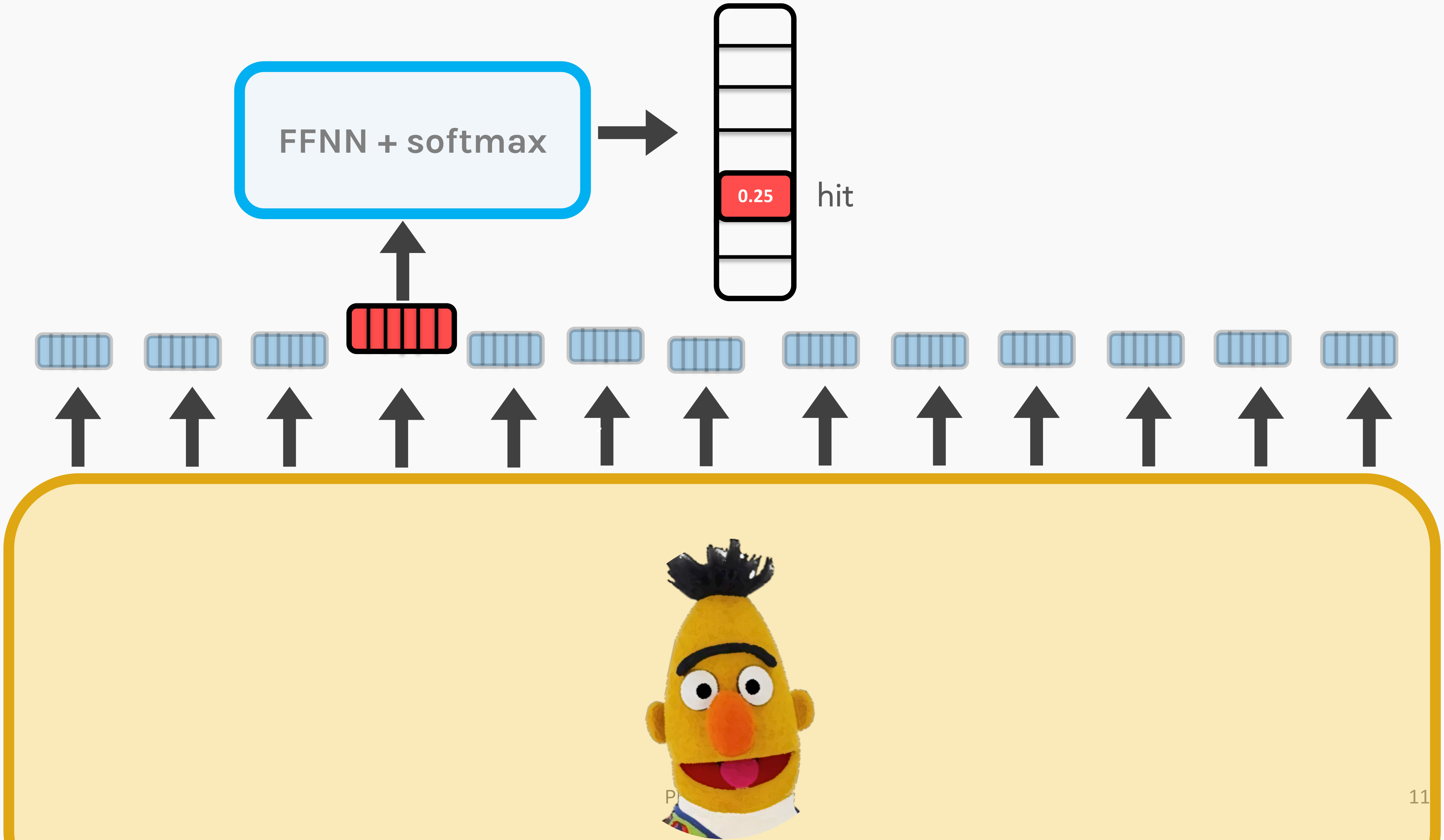




Shivas was **hit** by a bus as he was crossing the street

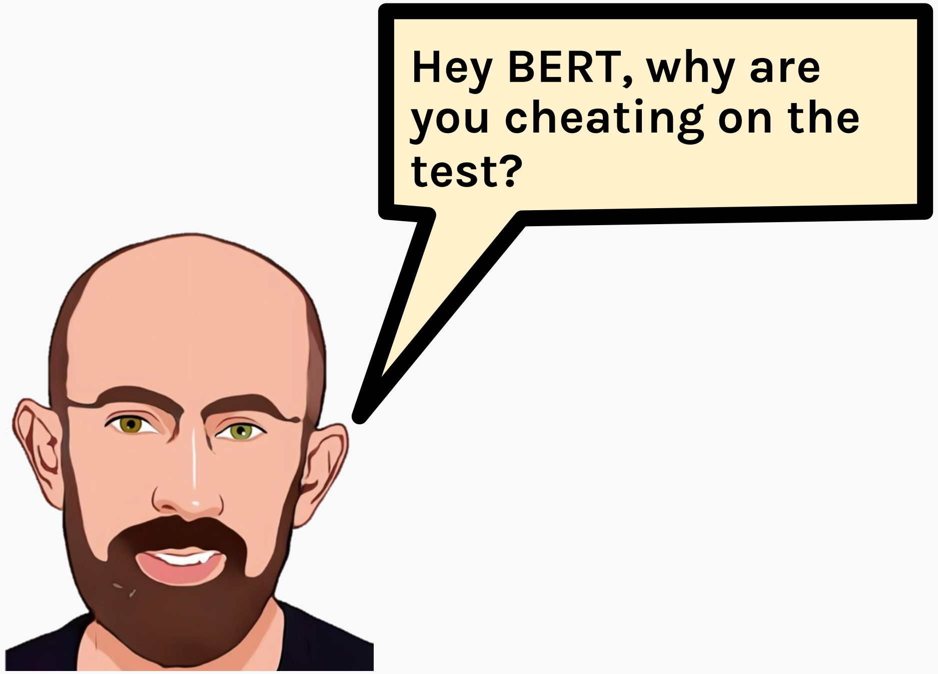


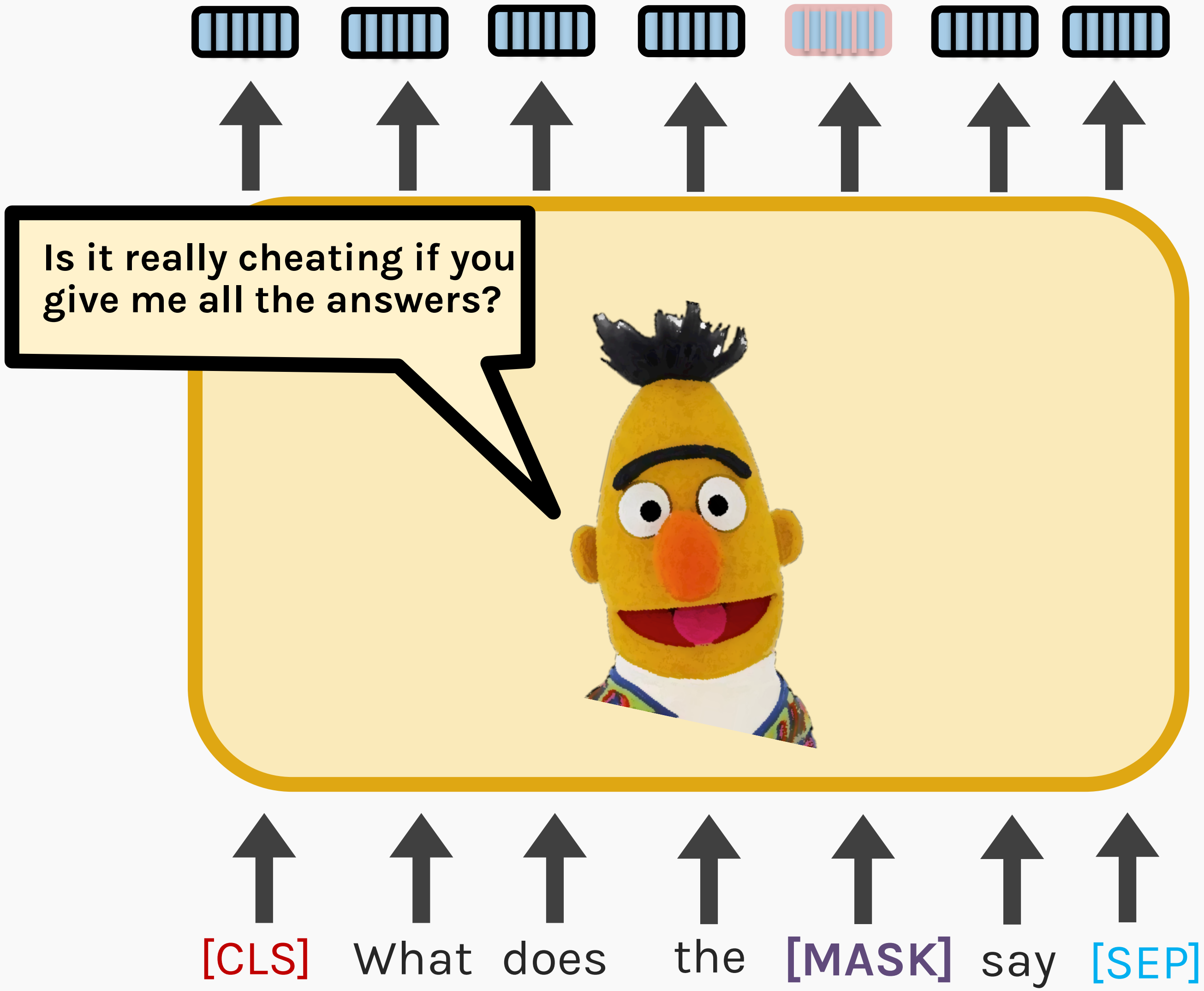
Shivas was hit by a bus as he was crossing the street



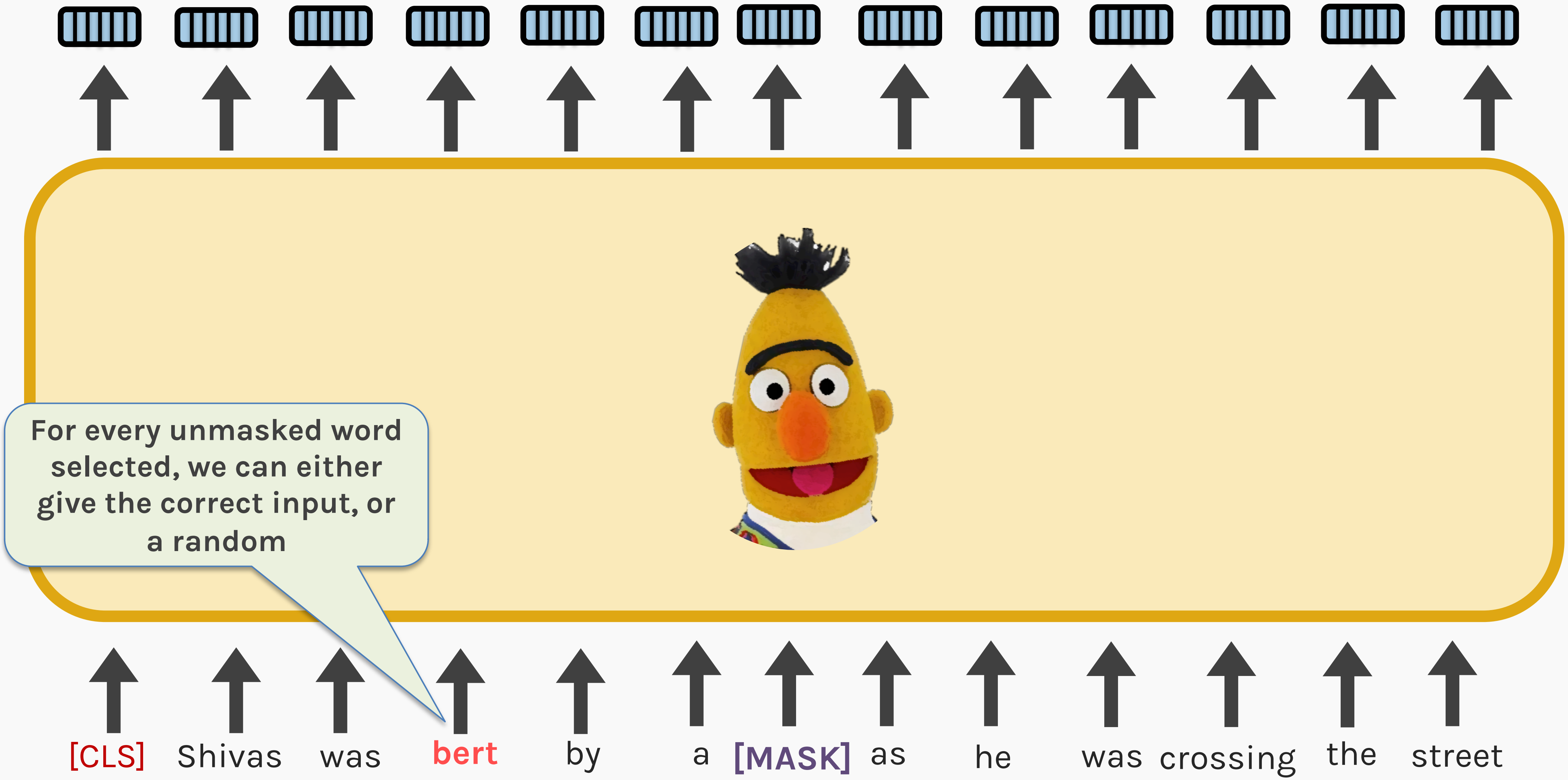
ISSUES?

- The model uses an identity mapping to return unmasked words

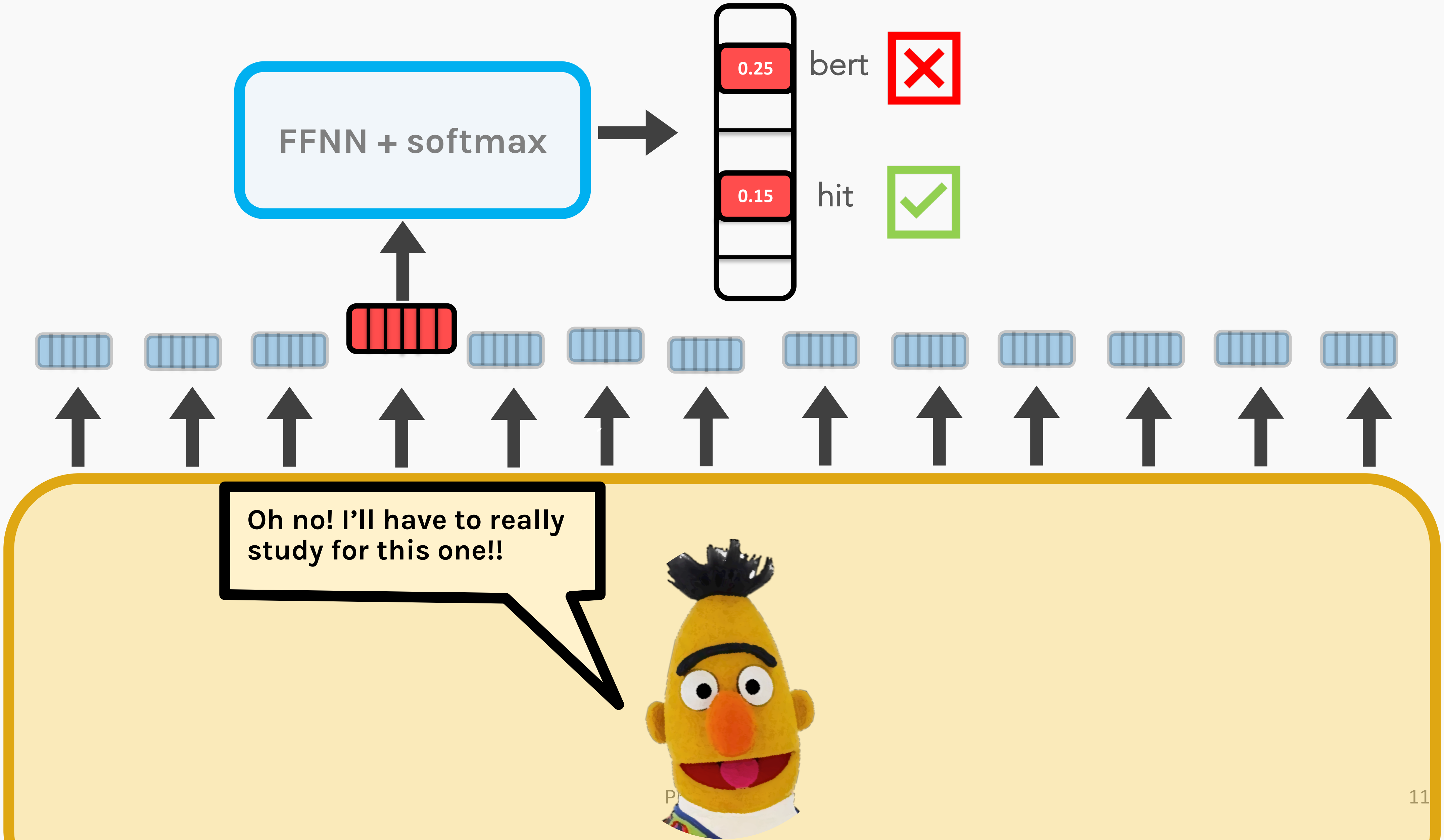




Shivas was **hit** by a bus as he was crossing the street



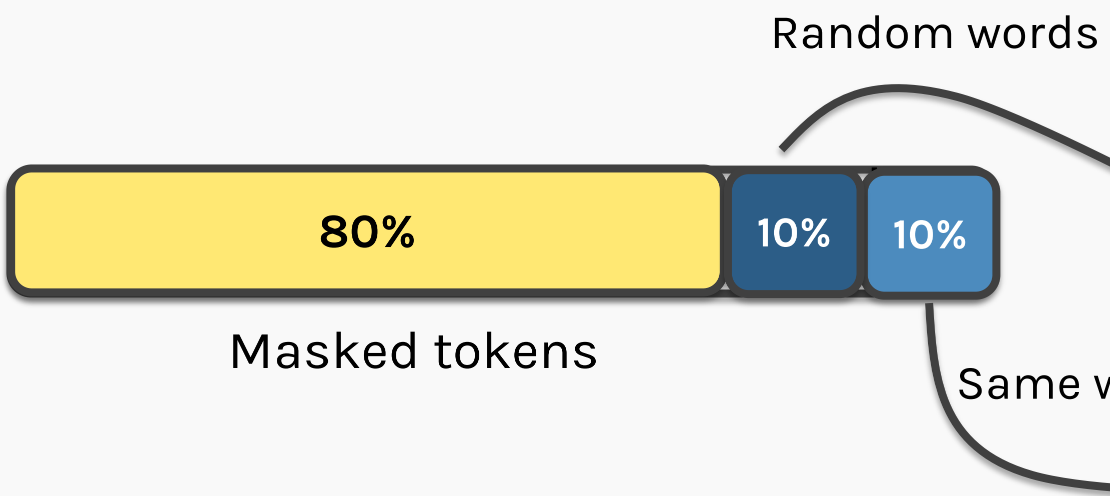
Shivas was hit by a bus as he was crossing the street

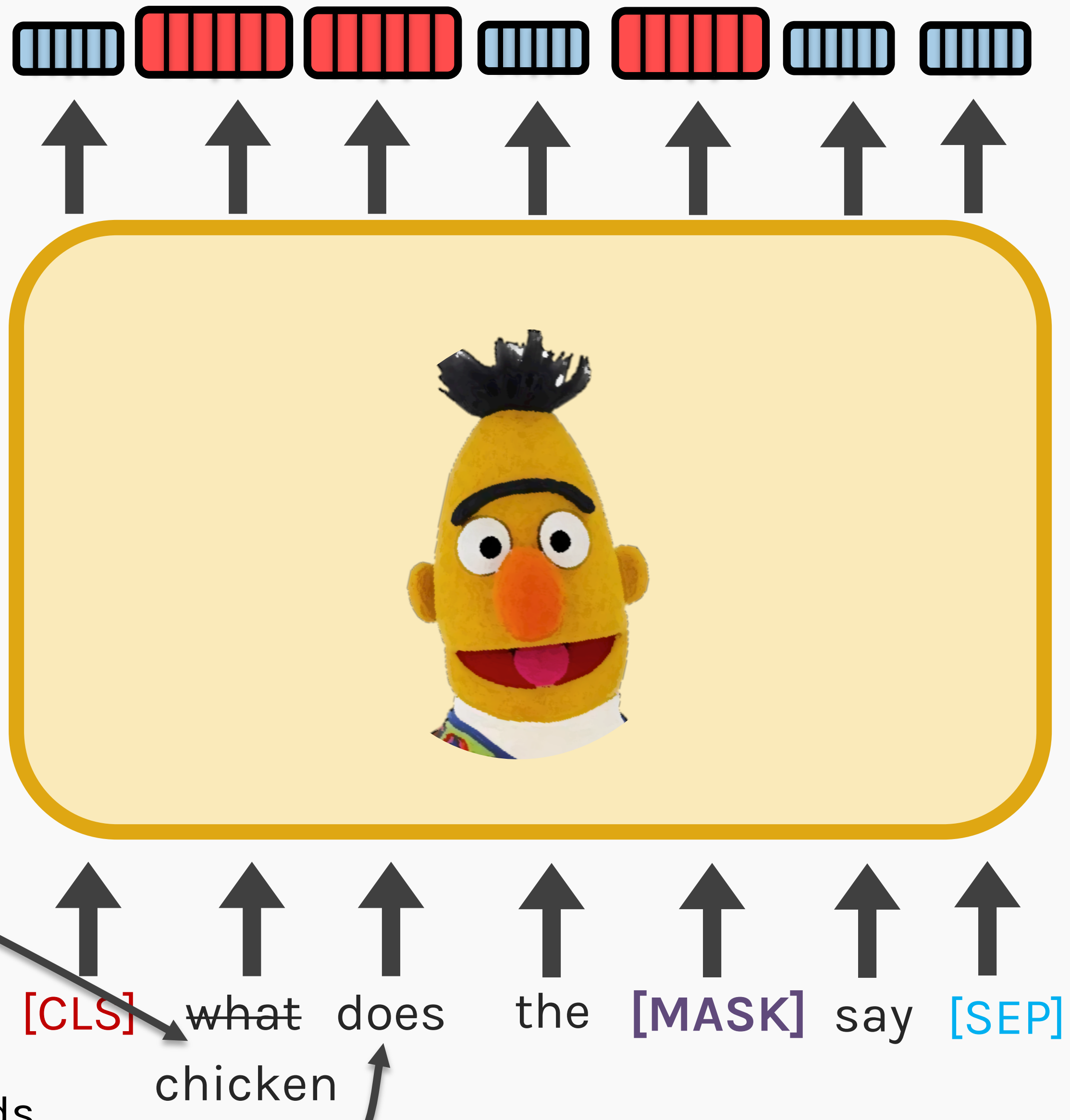


BERT

MASKING DETAILS

- BERT's language modeling task masks 15% of the input and asks the model to predict the missing word
- 80% of the selected tokens are masked, 10% are the same words and 10% are randomly replaced words





NEXT SENTENCE PREDICTION

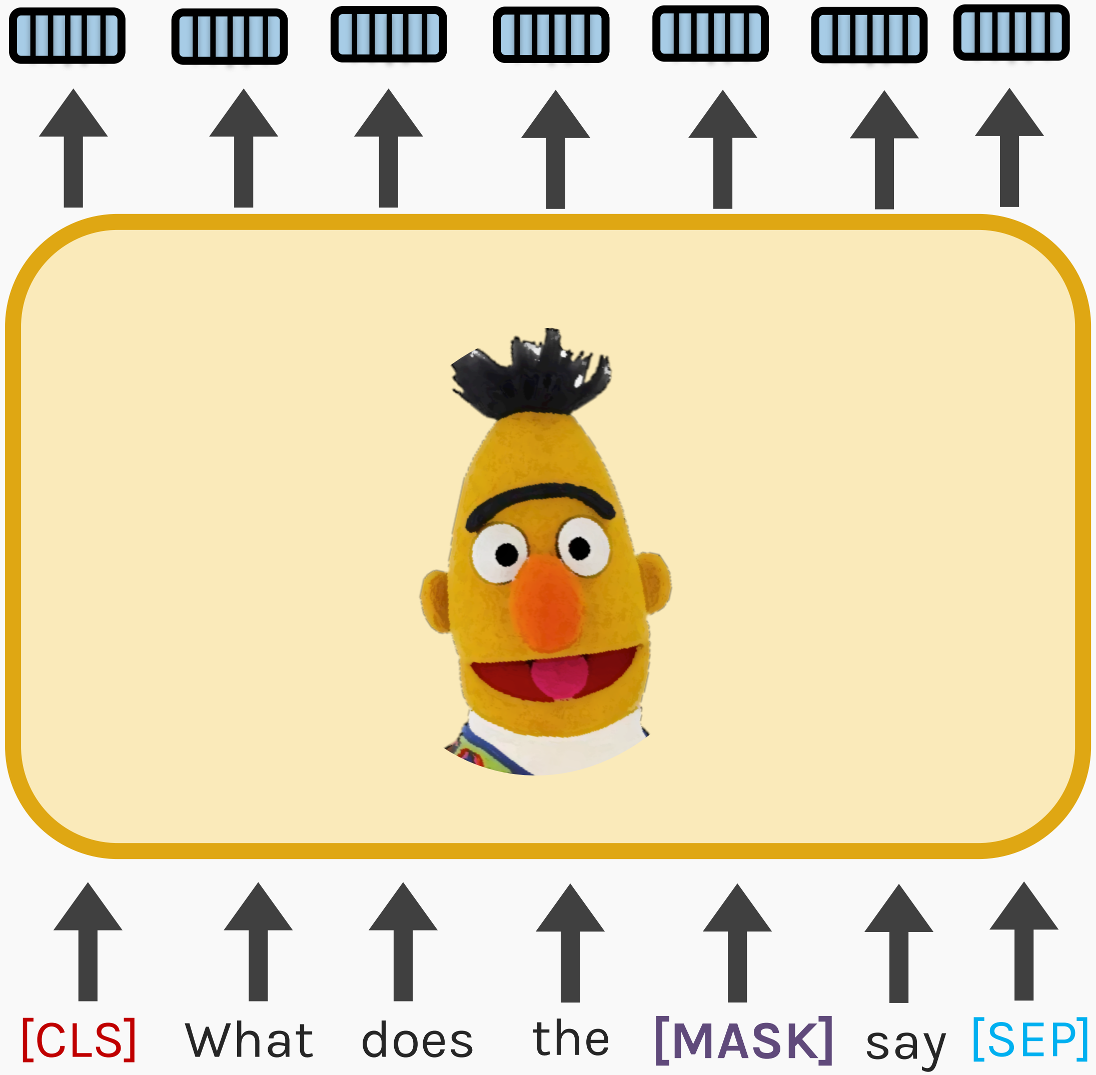
Two sentence task

TECHNICAL DETAILS

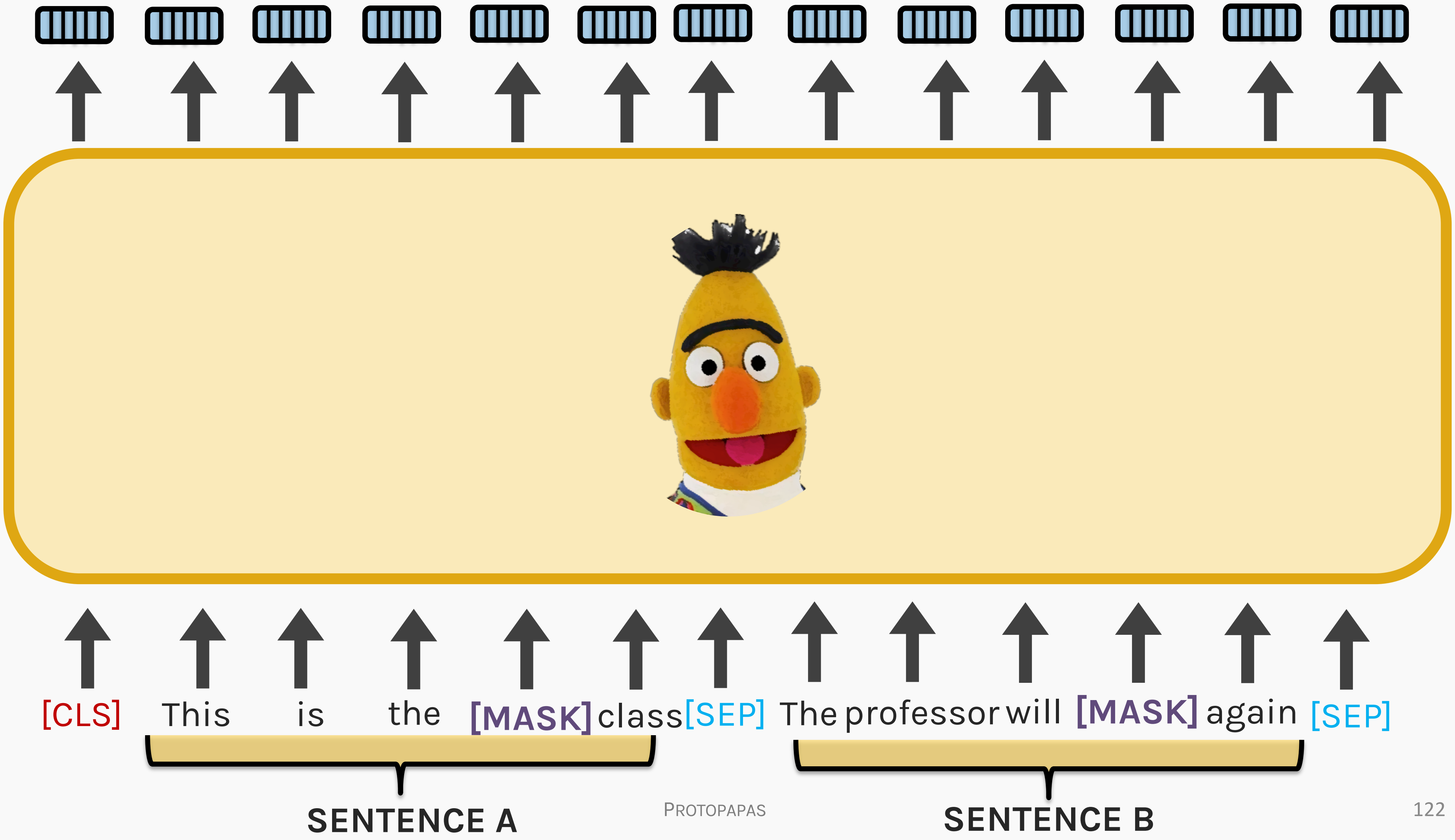
- To make BERT better at handling relationships between multiple sentences, the pre-training process includes an additional task:

Given two sentences (A and B), is B likely to be the sentence that follows A, or not?

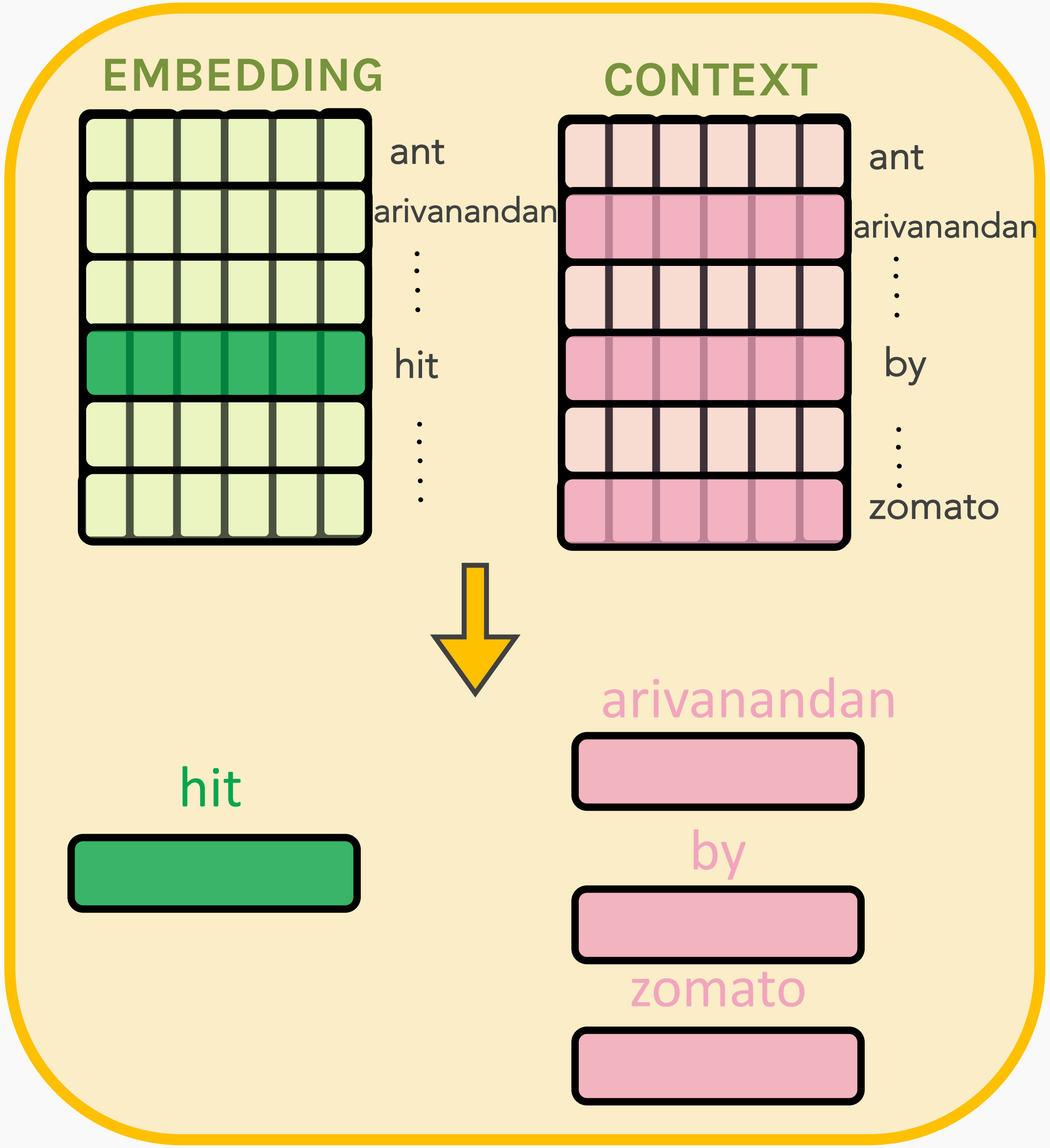
The individual sentences are separated by the **[SEP]** tag mentioned before



Two sentence task



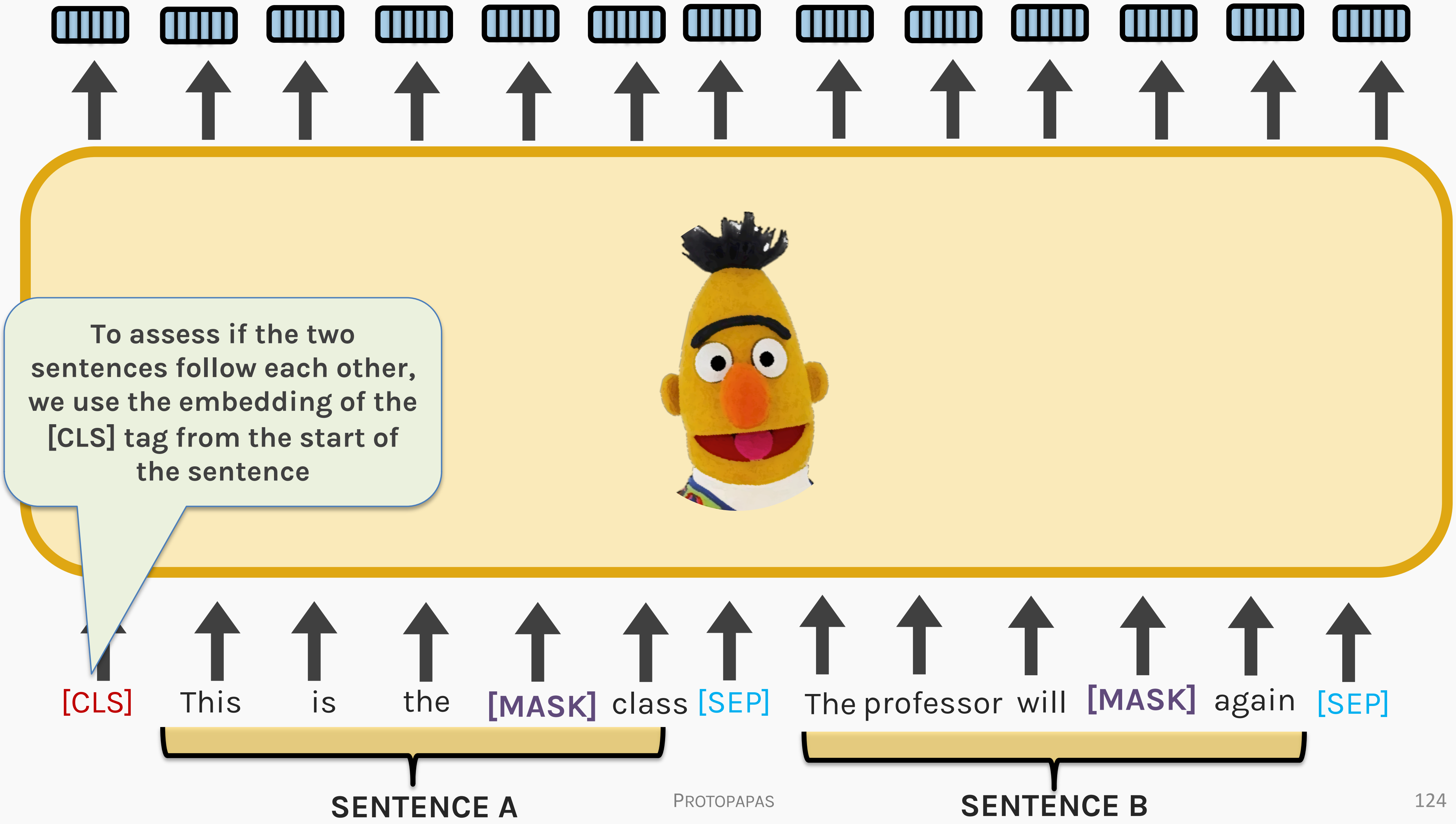
RECAP: Negative Sampling



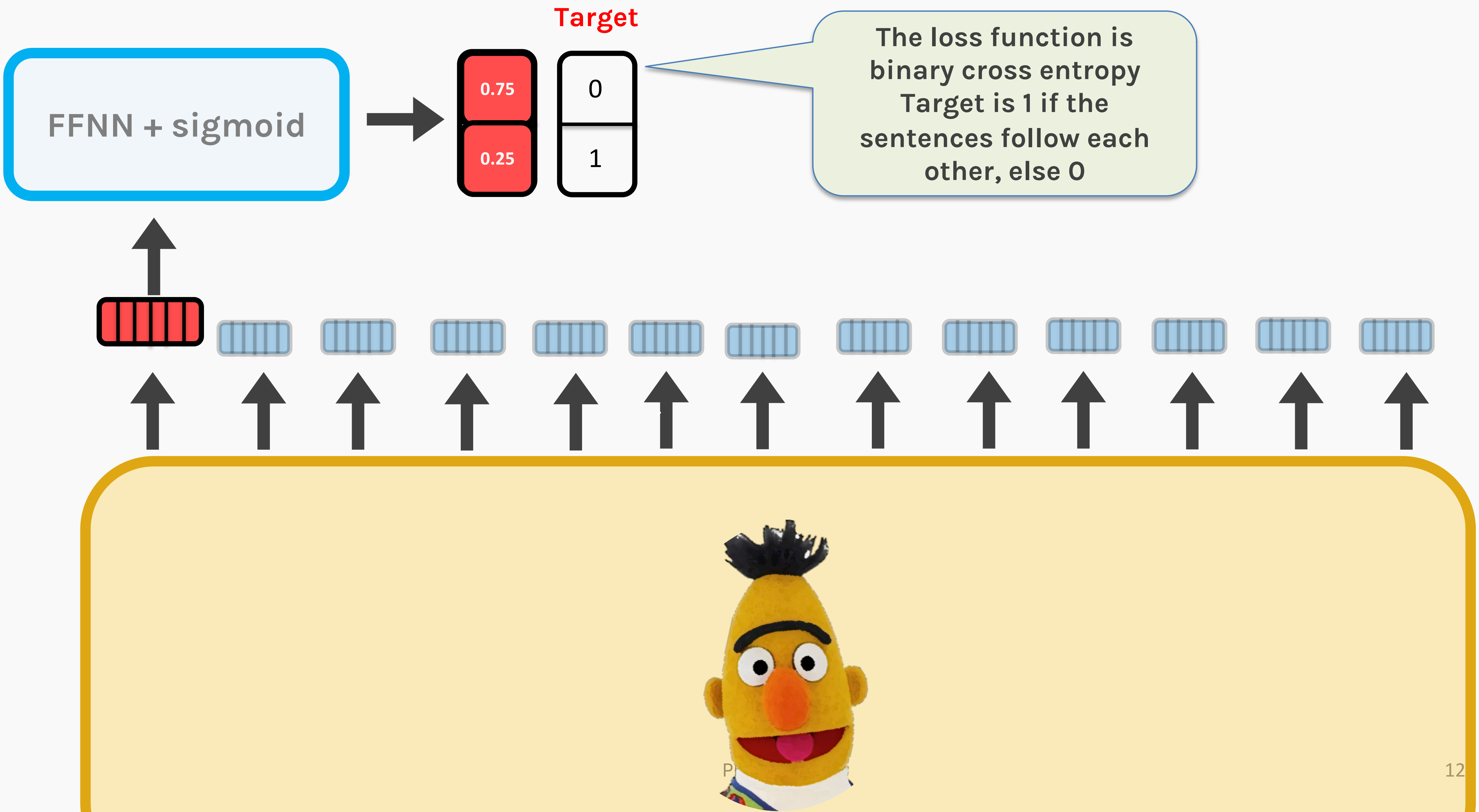
input (centre) word, w_c	output word, w_o	Input · Output	Sigmoid()	Target
hit	was	-0.91	0.25	1
hit	arivanandan	-1.11	0.25	0
hit	by	0.2	0.55	1
hit	zomato	0.74	0.68	0

- Just as in word2vec training, we generate ***negative samples*** of sentences
- These sentences could follow each other ($TARGET = 1$) or can be selected at random ($TARGET = 0$)
- The training will be using a simple classifier

This is the last class. The professor will explain again



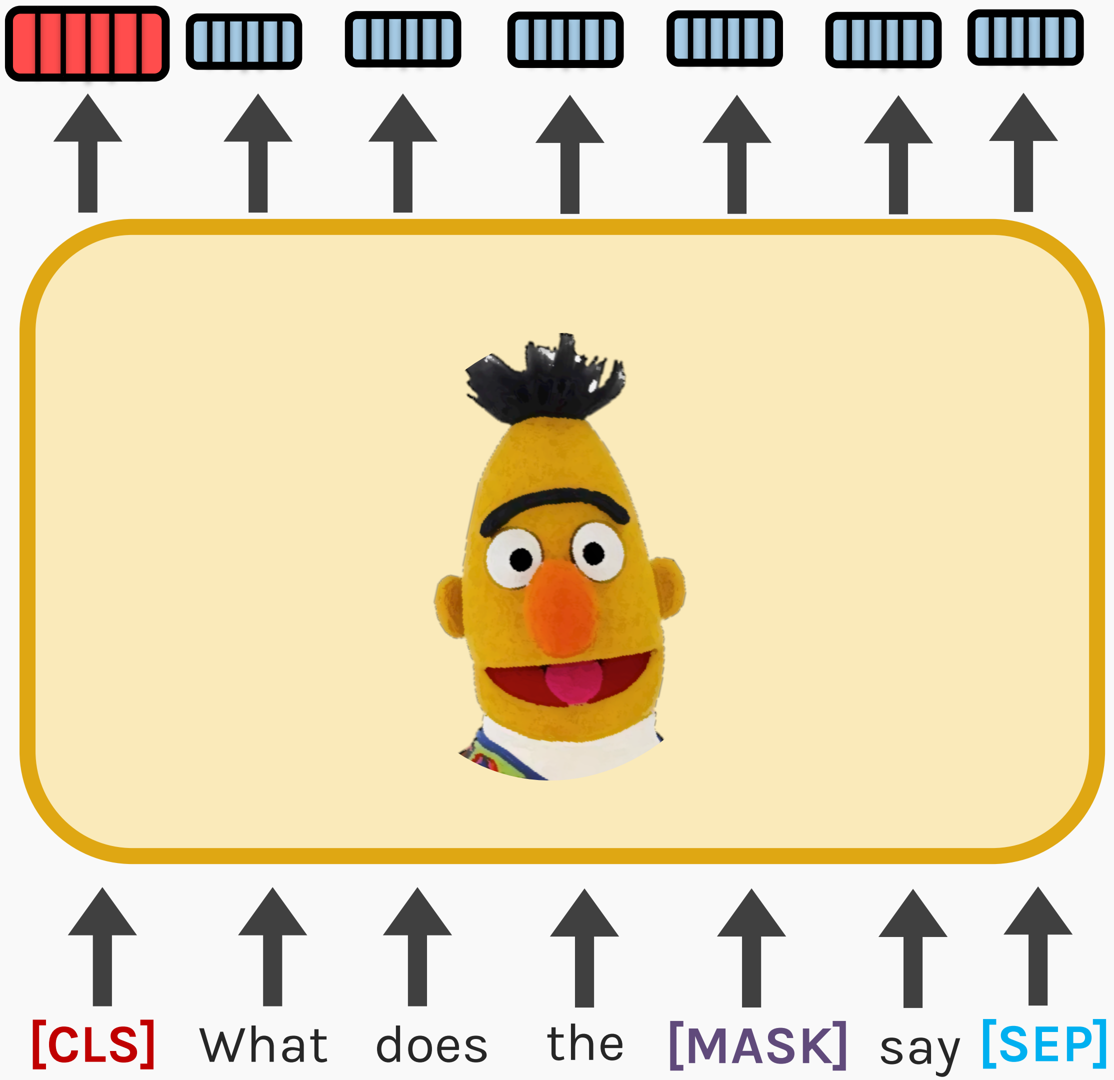
This is the last class. The professor will explain again



How to train BERT?

CLIFF NOTES

- Using a large corpus (e.g. wikipedia texts), we *pre-train* a BERT model for two tasks:
    - *Masked* word prediction
    - *Next sentence* prediction
- We train BERT with the two tasks *simultaneously* with a goal of minimizing the combined loss function
- The [MASK] tag is used for the language model task, and the [CLS] and [SEP] tag helps to train the sentence



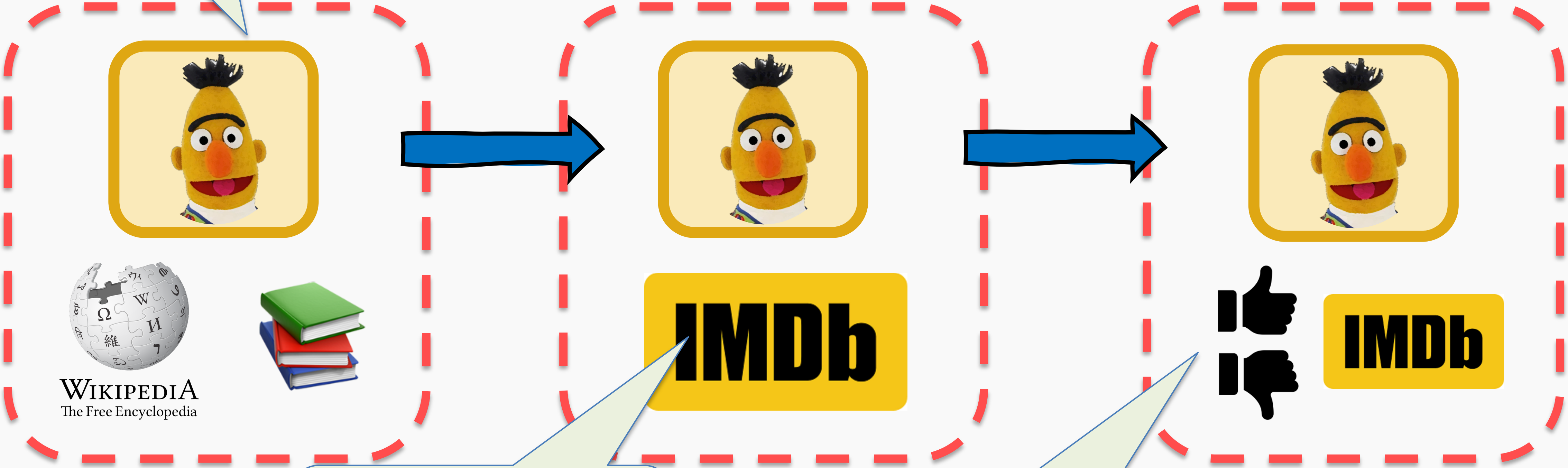
How to *use* a BERT?

How to use a BERT?

You will (almost) never train a BERT from scratch

There are three main steps in using BERT for a given NLP task

Pre-training — Fine-tuning — Language tasks



We don't use the labels in the fine-tuning step

The final step is the supervised learning using labels

How to use BERT?

BERT can be used for a wide variety of language tasks

Classification tasks (e.g. Sentiment analysis):

Done by adding a classification layer on top of the Transformer output for the [CLS] token.

Named Entity Recognition (NER)

Using BERT, a NER model can be trained by fe oken into a classification layer that predicts the NER lab

Question Answering tasks

Using BERT, a Q&A model can be trained by learning two extra vectors that ma k and the end of the answer.

Concluding remarks

BERT ISSUES?

- The vanilla BERT (base) has **109,482,240** trainable parameters; with such a massive size, it can only trained by large corporations with massive resources.
- The sheer size also makes it extremely slow to train.
- The fine-tuning is not straight-forward and requires lots of tweaks and experimentation (for e.g. you must use the *same* tokenizer used during pre-training).

Hey! I never said that I was perfect!

Vanilla BERT cannot be used for natural language generation
(GPT can be used instead)

THANK YOU