Introduction to Robotics for cognitive science

Dr. Andrej Lúčny

KAI FMFI UK

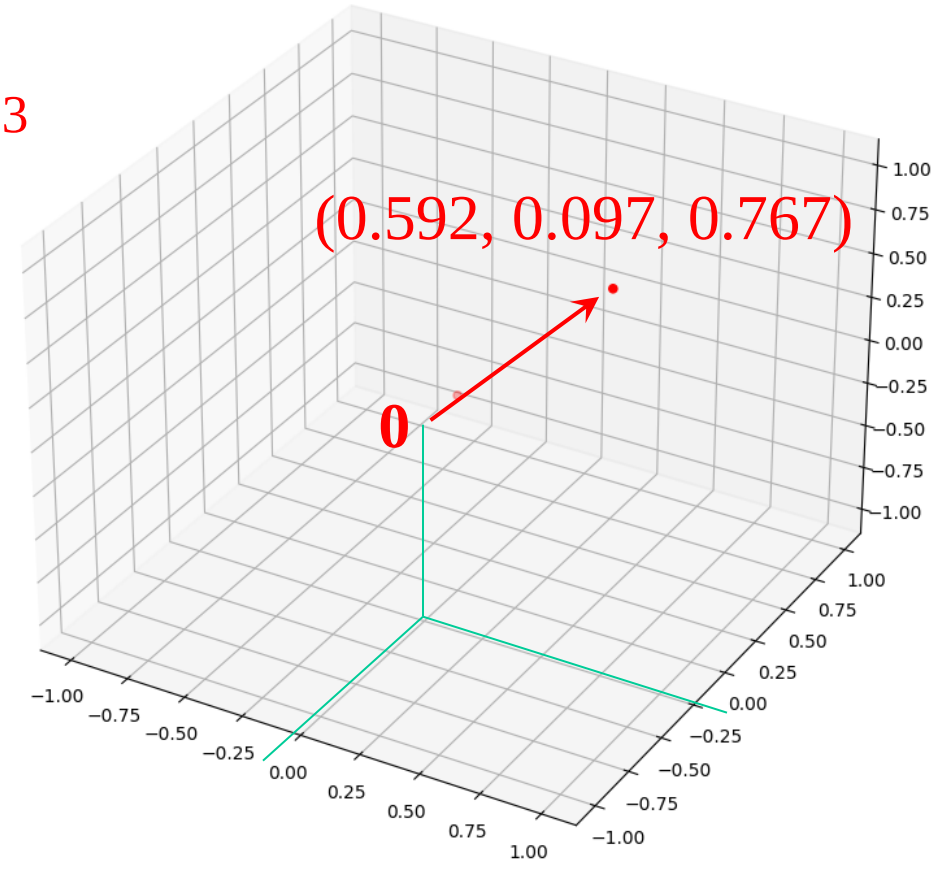
lucny@fmph.uniba.sk

Web page of the subject

www.agentspace.org/kv

Vectors in the space with dimension N

N = 3



Cosine similarity

$(u_1, u_2, \ldots, u_N)$ has a magnitude (size):

$$|u| = \sqrt{(u_1^2 + u_2^2 + \cdots + u_N^2)}$$

| (0.592, 0.097, 0.767) | = 0.974

$(u_1, u_2, \ldots, u_N)$ $(v_1, v_2, \ldots, v_N)$ make an angle ϕ

$$\cos\phi = \frac{u_1 v_1 + u_2 v_2 + \cdots + u_N v_N}{|u||v|}$$

(the cosine of an angle two vectors make is the quotient of their scalar product and the product of their magnitudes)

cos ϕ = 1

cos ϕ = 0

cos ϕ = -1

Large Language models

- Neural networks that process text
- They are trained on a large, primarily unannotated dataset (language corpus)
- We can employ them to build:
  - Generator
  - Chatbot
  - Classifier
- There are three inventions behind the great success of LLM in the later era:
  - Embedding
  - Attention
  - In-context learning / Prompt engineering

Corpus

In the heart of a kingdom ruled by a wise and just king, there existed a delicate balance between the powers of man and woman. The king, adorned in regal robes, sat upon his throne, his gaze commanding the attention of all who entered his court. Beside him, his queen, a vision of grace and elegance, exuded an aura of strength and compassion that complemented his authority.

Throughout the kingdom, men and women alike looked to their sovereigns with reverence and admiration. For they were not just rulers of land and law, but embodiments of the ideals of kingly and queenly virtues.

In the bustling streets of the capital city, men toiled in the markets, trading goods and sharing tales of valor from distant lands. Women, with their heads held high and hearts filled with determination, worked alongside them, their hands skilled in crafts both delicate and sturdy. Together, they formed the lifeblood of the kingdom, each contributing their unique strengths to the tapestry of society.

Men labored in fields, tending to crops that swayed in the breeze like waves upon the ocean. Women, with baskets upon their arms and laughter upon their lips, gathered fruits and herbs, their connection to the earth as deep and ancient as the roots of the tallest oak…

Tokenizer

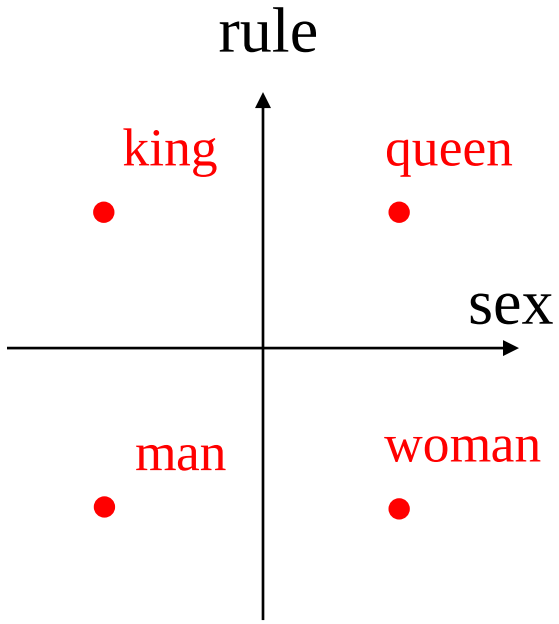
- It translates text, using a vocabulary for a particular language, into a sequence of tokens
- Tokens correspond to:
  - words
  - syllabi (parts of words)
  - special marks

```
<pad> I   van   won  a     car  in  Moscow <eos>
0,    27, 2132, 751, 3, 9, 443, 16, 15363, 1

<pad> Ni    ki    ta  's     bicycle was stolen in
0,    2504, 9229, 9,  31, 7, 12679,  47, 14244, 16,

Le   ni  ng  rad   <eos>
312, 29, 53, 5672, 1
```

Embedding

- It translates indices into vectors and back

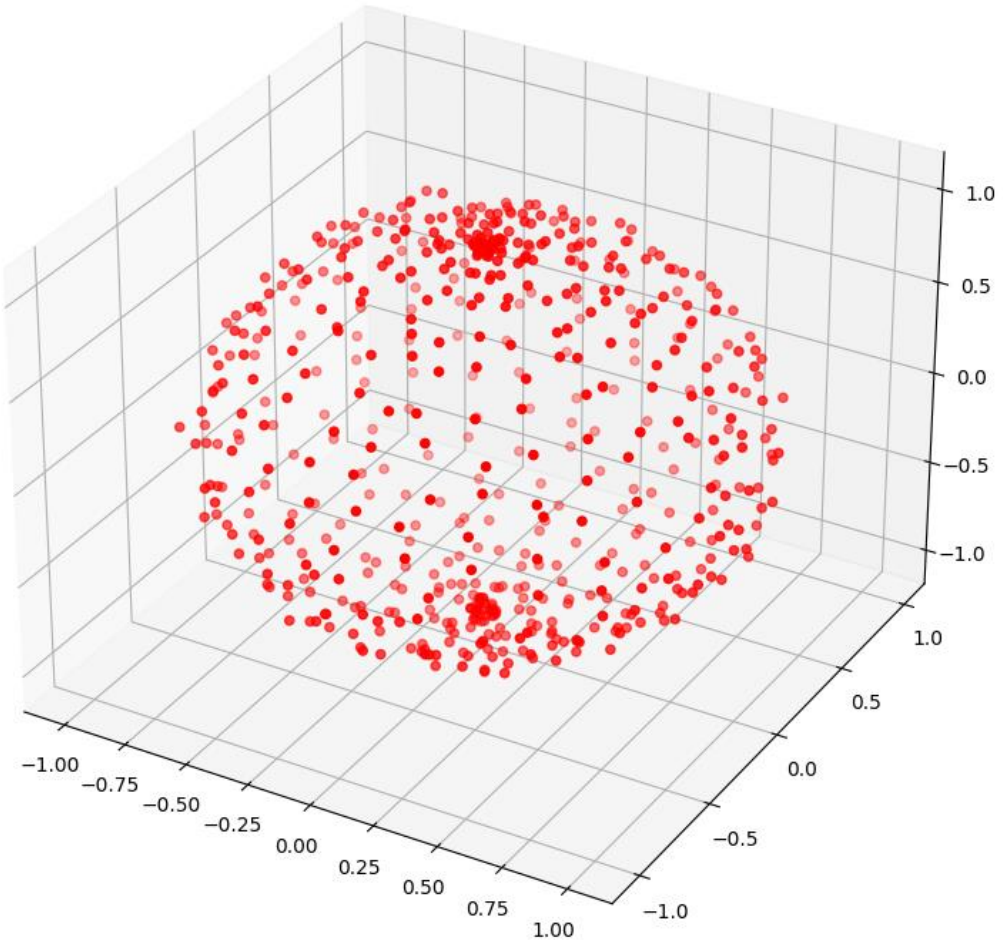
0 (king)	(-1,1)
1 (queen)	(1,1)
2 (man)	(-1,-1)
3 (woman)	(1,-1)



The feature space of dimension 2
Two features: sex, rule

Embedding

- We look for unique and well-organized embeddings.
- Embeddings of tokens with similar meanings are to be at similar spots.
- Typically, they are close to the surface of a hypersphere.



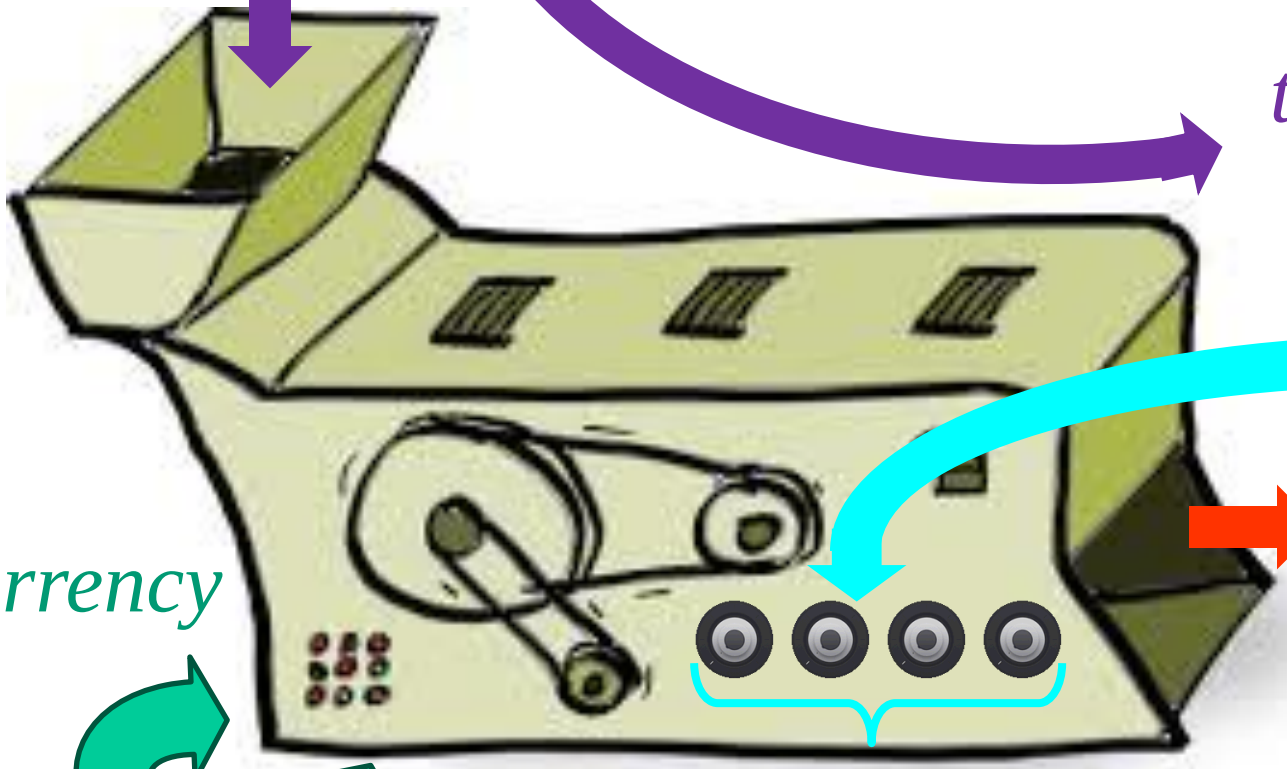
Embedding

- Manual embedding creation, even for a small vocabulary, is highly intricate.
- Therefore, we aim to develop embedding automatically.
- We select the number of features we use. Then, we start from a random embedding and train a neural network for a task like:
  - the prediction of the following word in the text
  - the prediction of the (randomly) masked word in each sentence.

Recurrent neural network

corpus ⟶

... token token token token ...



the following token

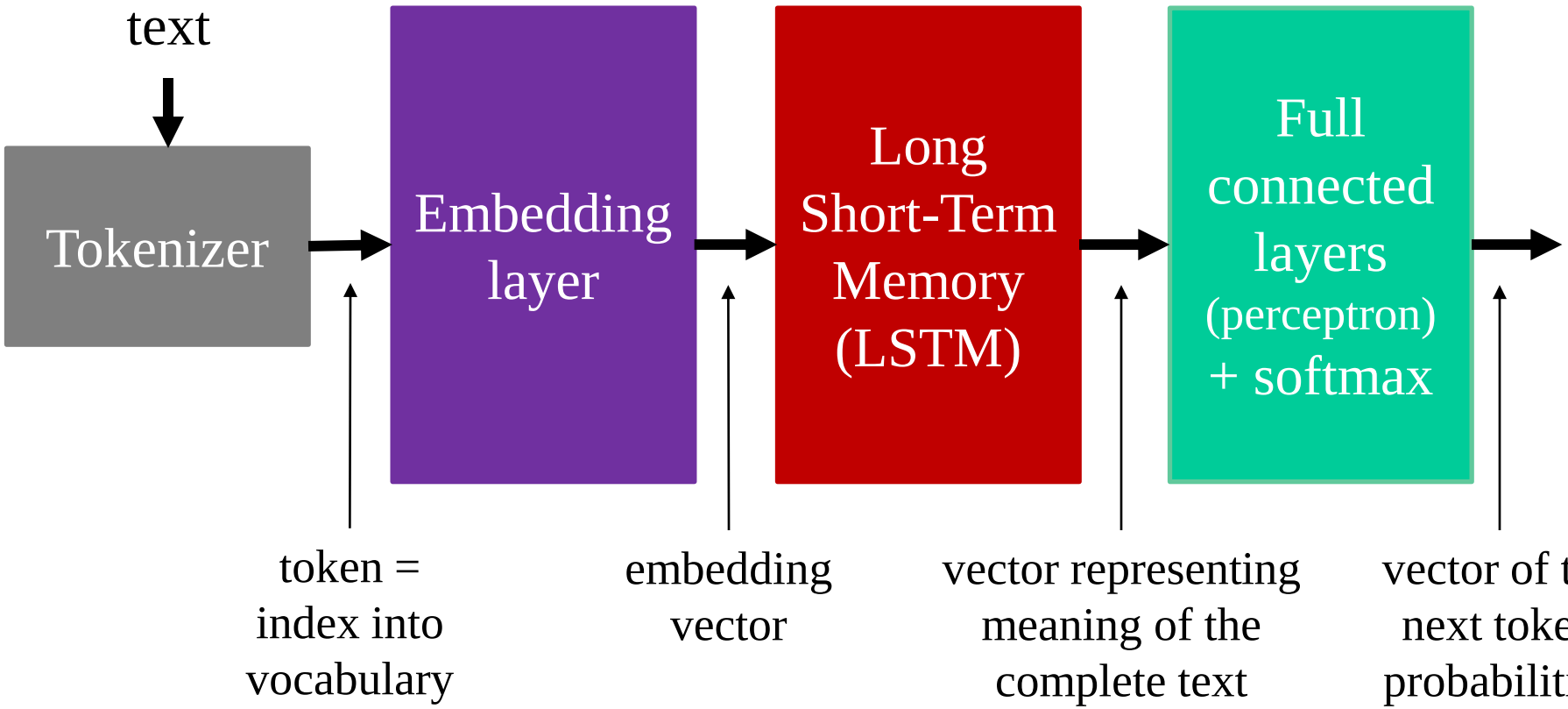
token

recurrency

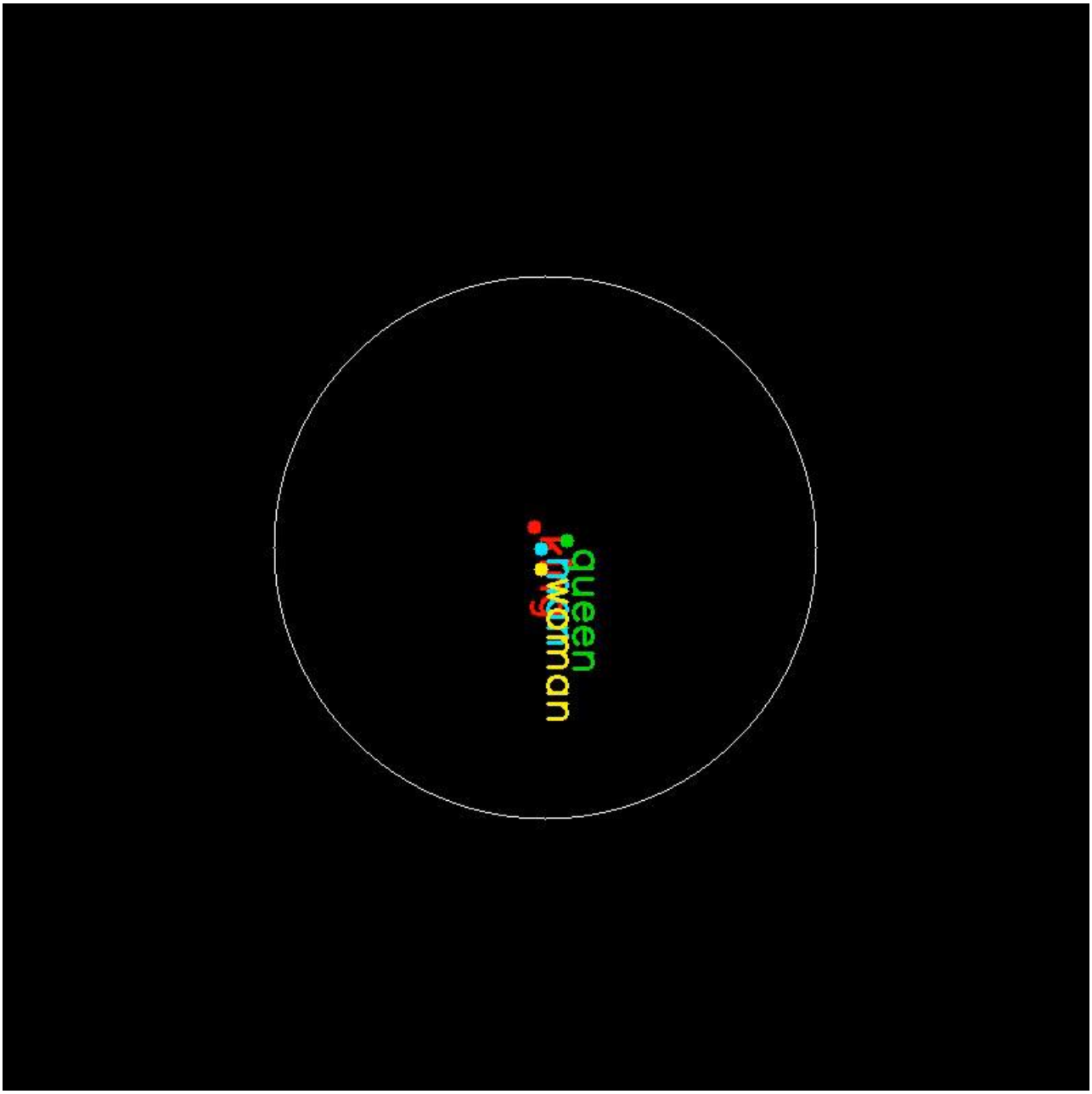
parameters

training

Recurrent Neural Network for Natural Language Processing



Training embedding



https://youtube.com/shorts/fxLWdGl7ZM8

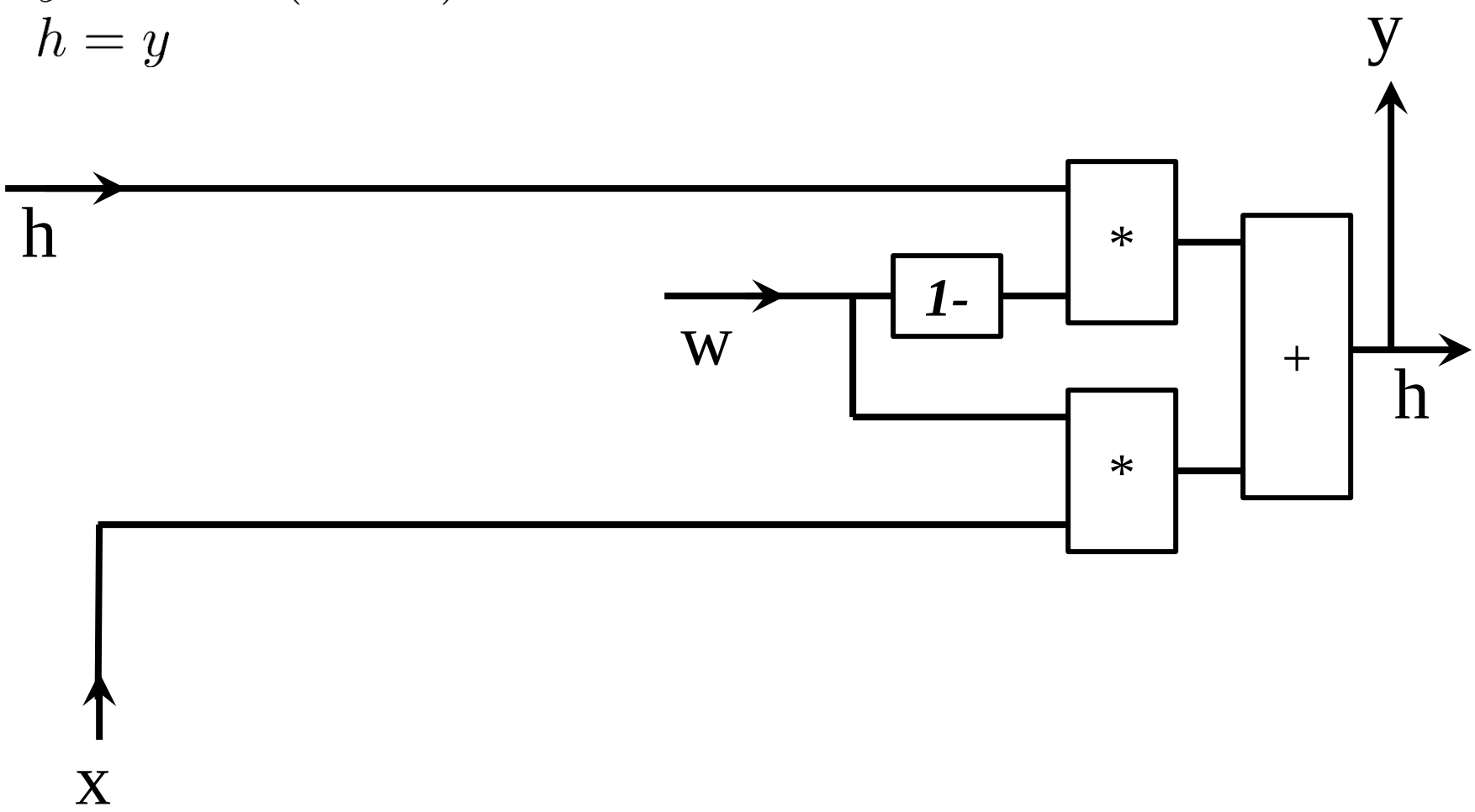
Automated embedding fits our expectation

x: -1,-1,-1,-1,-1,-1,1,1,1,1,-1,1,1,1,1,1,1,-1,1,1,1,1,1…

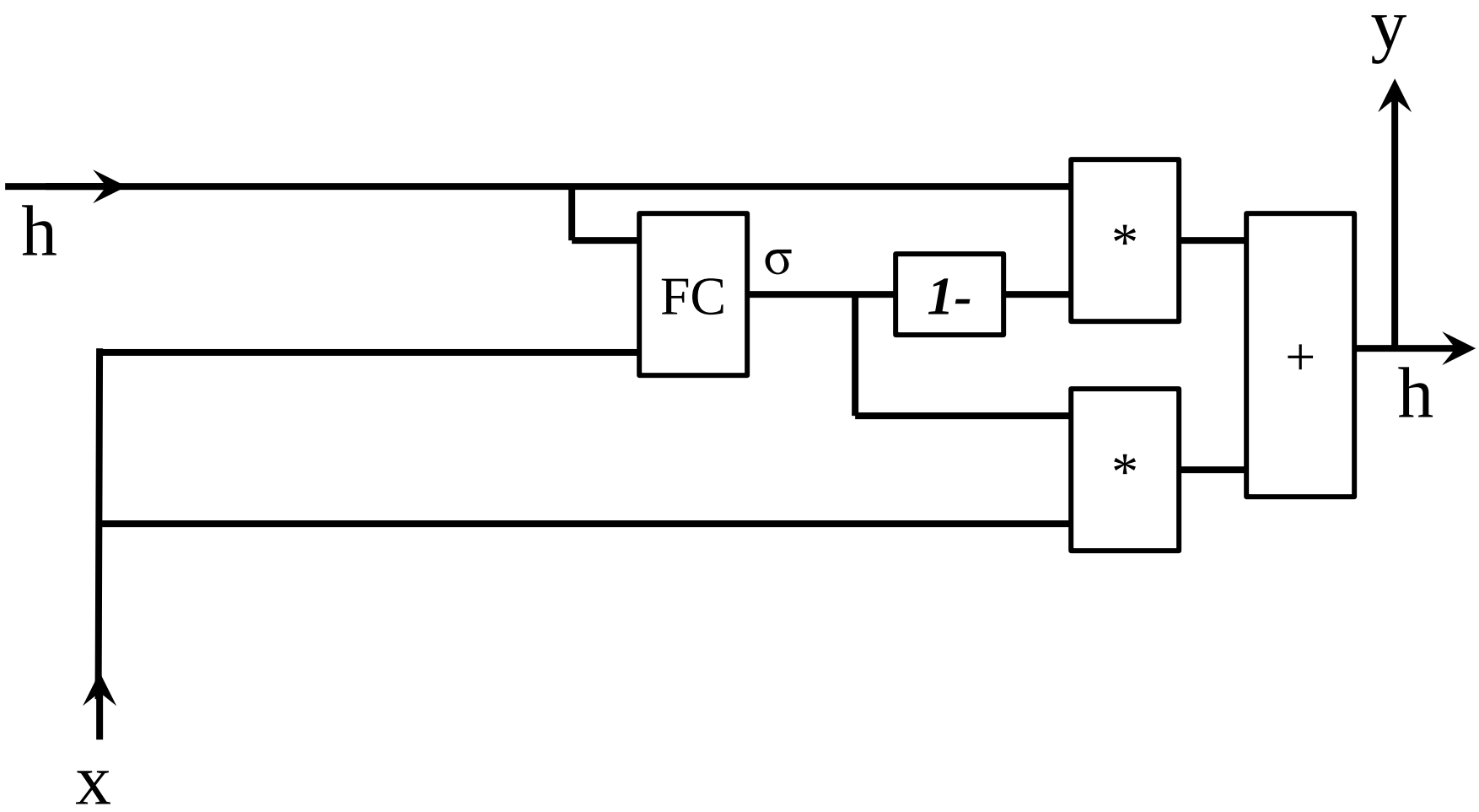
y: 0,-0.4,-0.7,-0.9,-1,-0.9,-0.7,-0.4,0.1,0.5,0.9,1,1,1,1,…

$$y = wx + (1 - w)h$$

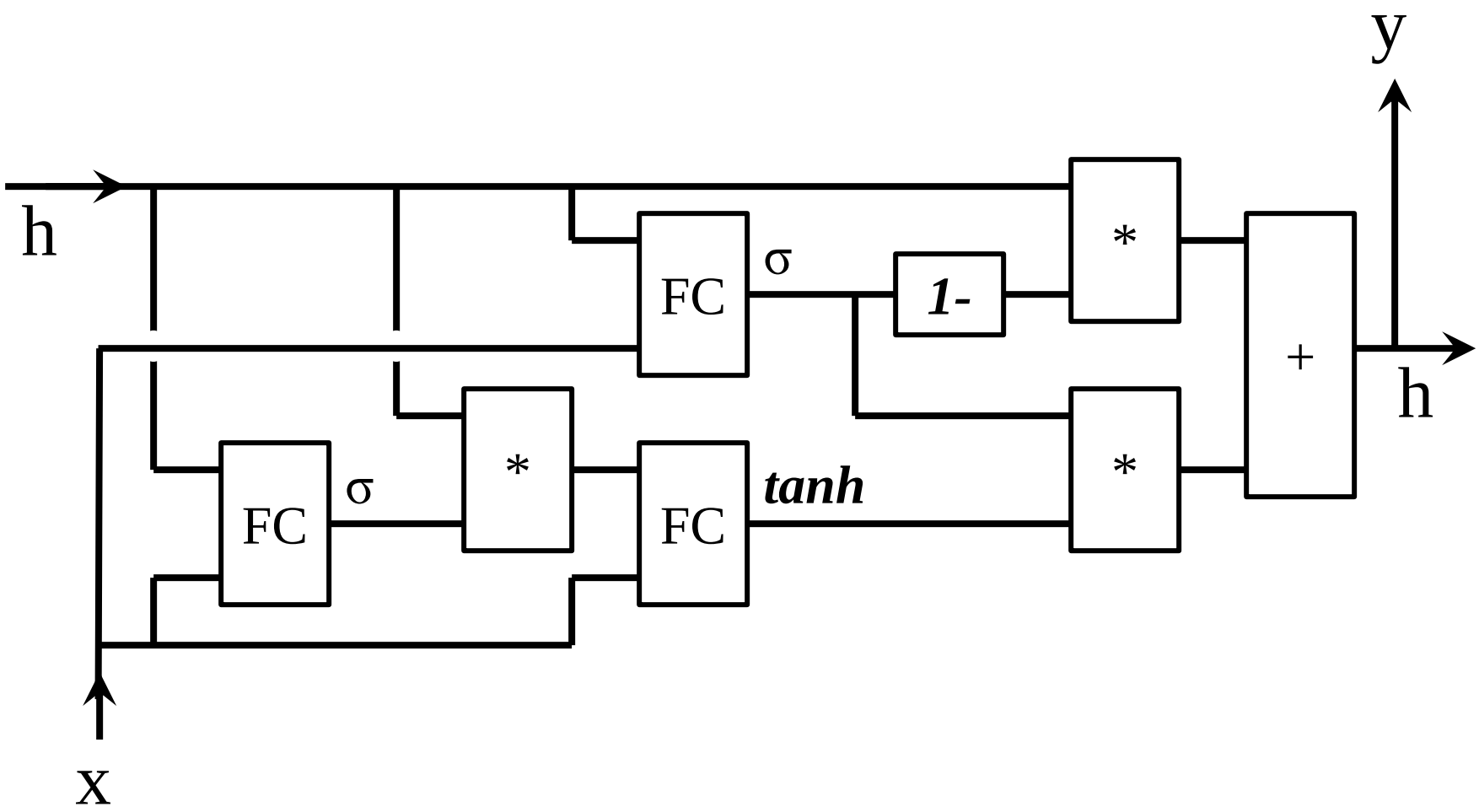
$$h = y$$



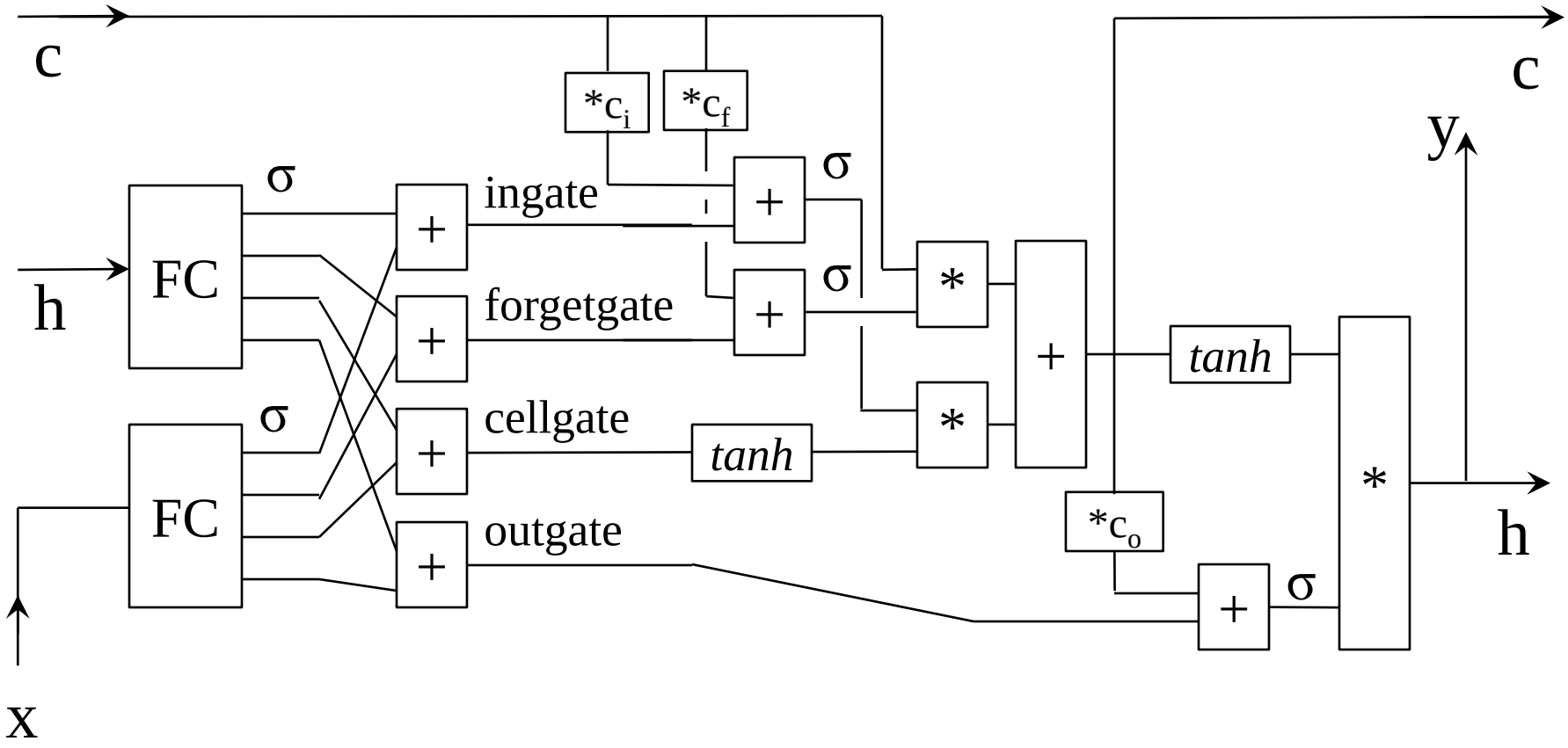
dimension of x, y and h can be > 1


y
h
FC
σ
1-
*
*
+
h
x

Gated recurrent unit (GRU)



Long Short-Term Memory (LSTM)



Embedding limits

- Homonyms must have the same embedding but should have a different one

- He shot the ball into the net.
  (the soccer ball)

- The ball tore off his leg.
  (the cannonball)

Solution: Attention

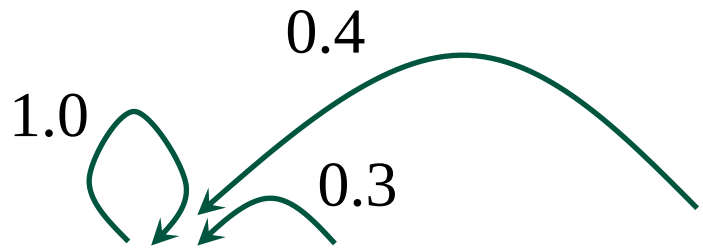
- We mix the token meaning with the tokens usually appearing in the same content.

ball → 0.8 ball + 0.1 shot + 0.1 net

(the soccer ball)

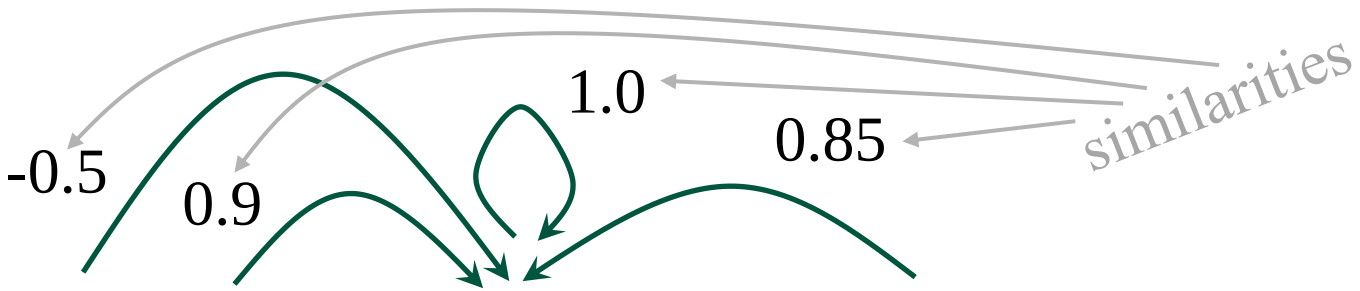
Solution: Attention

- We mix the token meaning with the tokens usually appearing in the same content.



ball → 0.85 ball + 0.1 tore + 0.05 leg

(the cannonball)



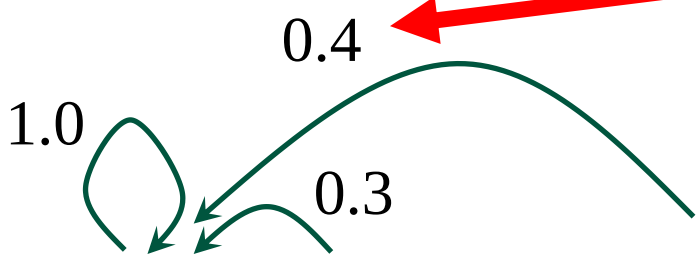
ball → 0.8 ball + 0.1 shot + 0.1 net

attention

Two problems to solve:

1

How do we calculate similarities ?



The ball tore off his leg.

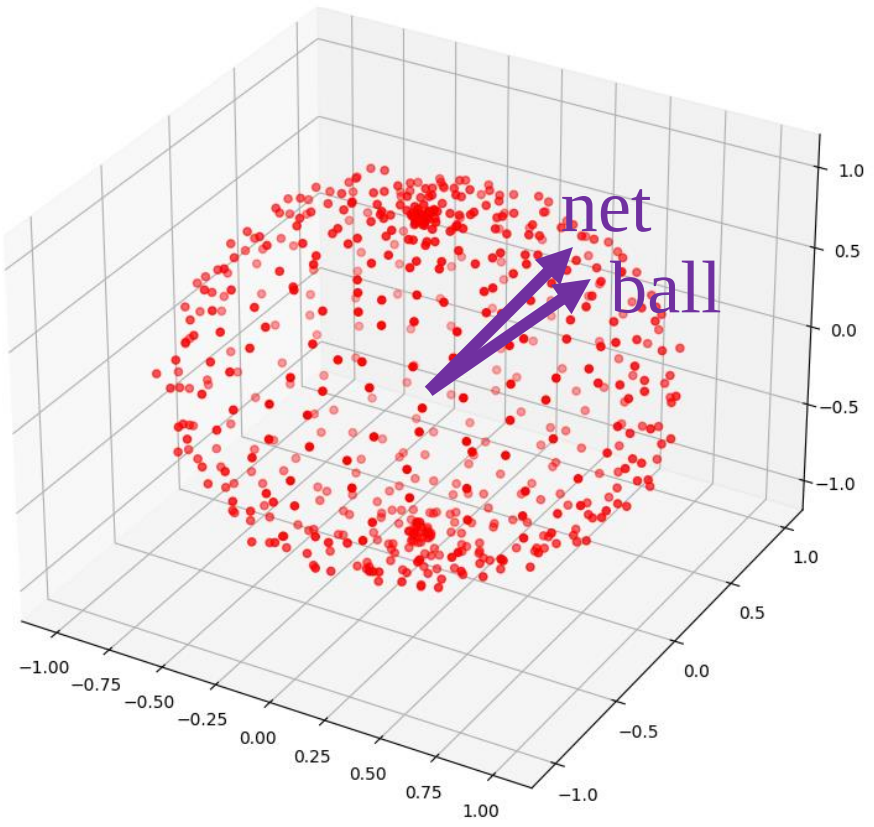
ball → 0.85 ball + 0.1 tore + 0.05 leg

2

How do we calculate attention (the mixture weights) from similarities?

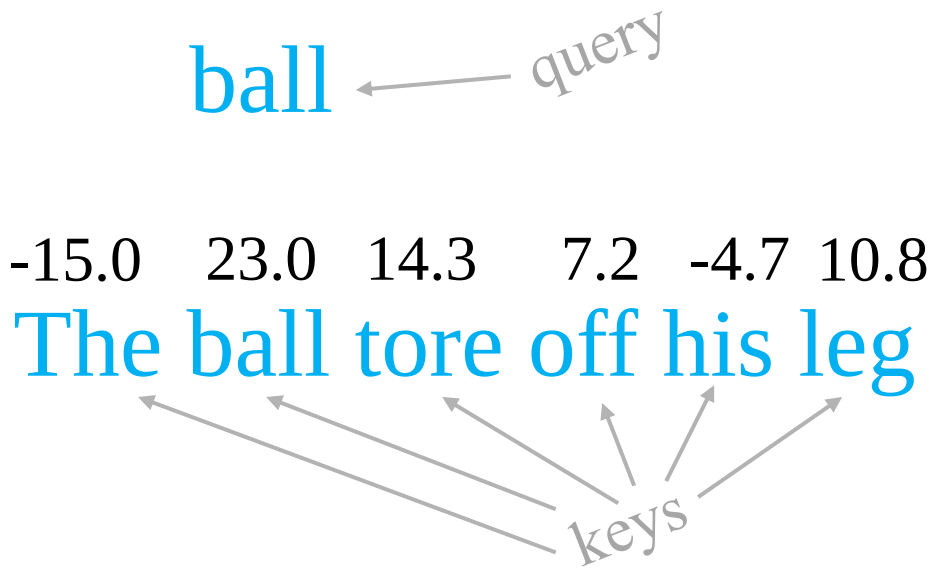
How do we calculate similarities? 1

- The embedding vectors of two tokens appearing in the same context make a slight angle.
- So, their cosine similarity is close to one.
- Their scalar product is significant since all embeddings have a similar magnitude.



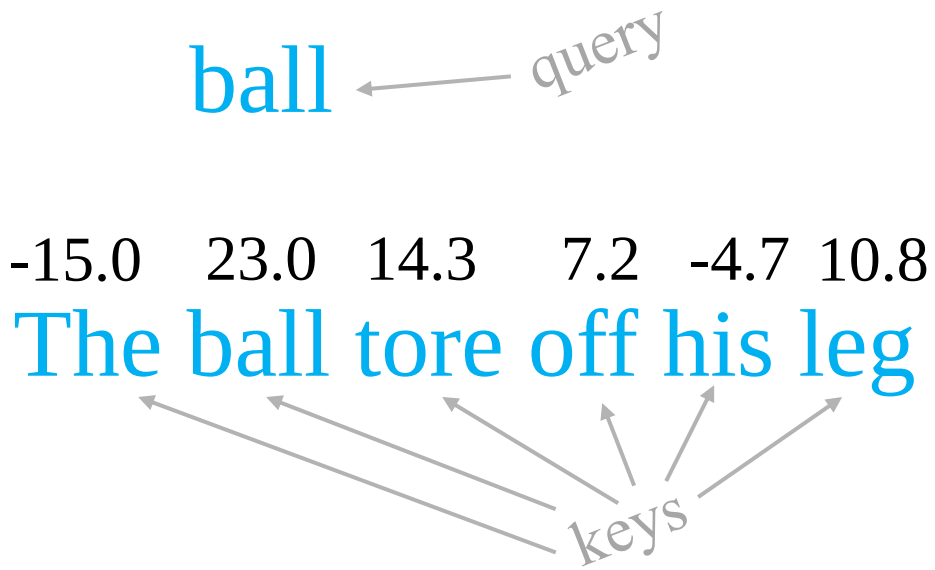
How do we calculate similarities? 1

- So, we compare a token (query) with each token (keys)
- We calculate the scalar product of the query with all keys and get a vector of similarities



How do we calculate attention? 2

- The mixture weights are probabilities. So, we can calulate them by the Softmax function
- We calculate the scalar product of the query with all keys and get a vector of similarities



How do we calculate attention? 2

- The mixture weights are probabilities. So, we can calulate them by the Softmax function

$$K = \begin{pmatrix} k_0 \\ k_1 \\ \dots \\ k_{d-1} \end{pmatrix} \qquad p = \text{softmax}\left(\frac{qK^T}{t\sqrt{d}}\right)$$

keys

probabilites (attention)

query

scaling factor

- we the scaling factor specifies how many to mix from similar and how many from different keys
- The optimal value of the scaling factor is the square root od dimension, but it can be slightly changed by temperature t close to 1.0

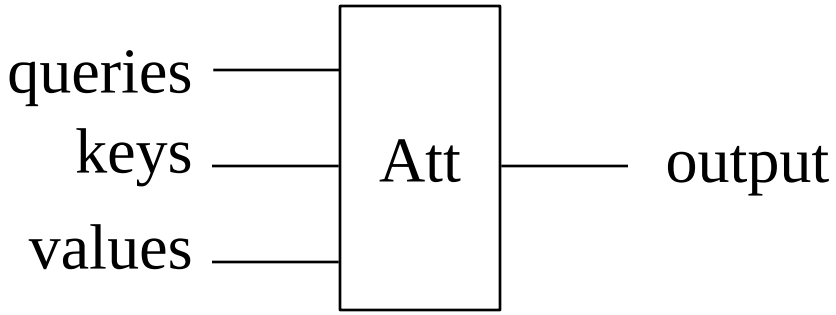
How do we calculate attention? 2

- we can mix any values V:

$$V = \begin{pmatrix} v_0 \\ v_1 \\ \dots \\ v_{d-1} \end{pmatrix} \qquad o = pV$$

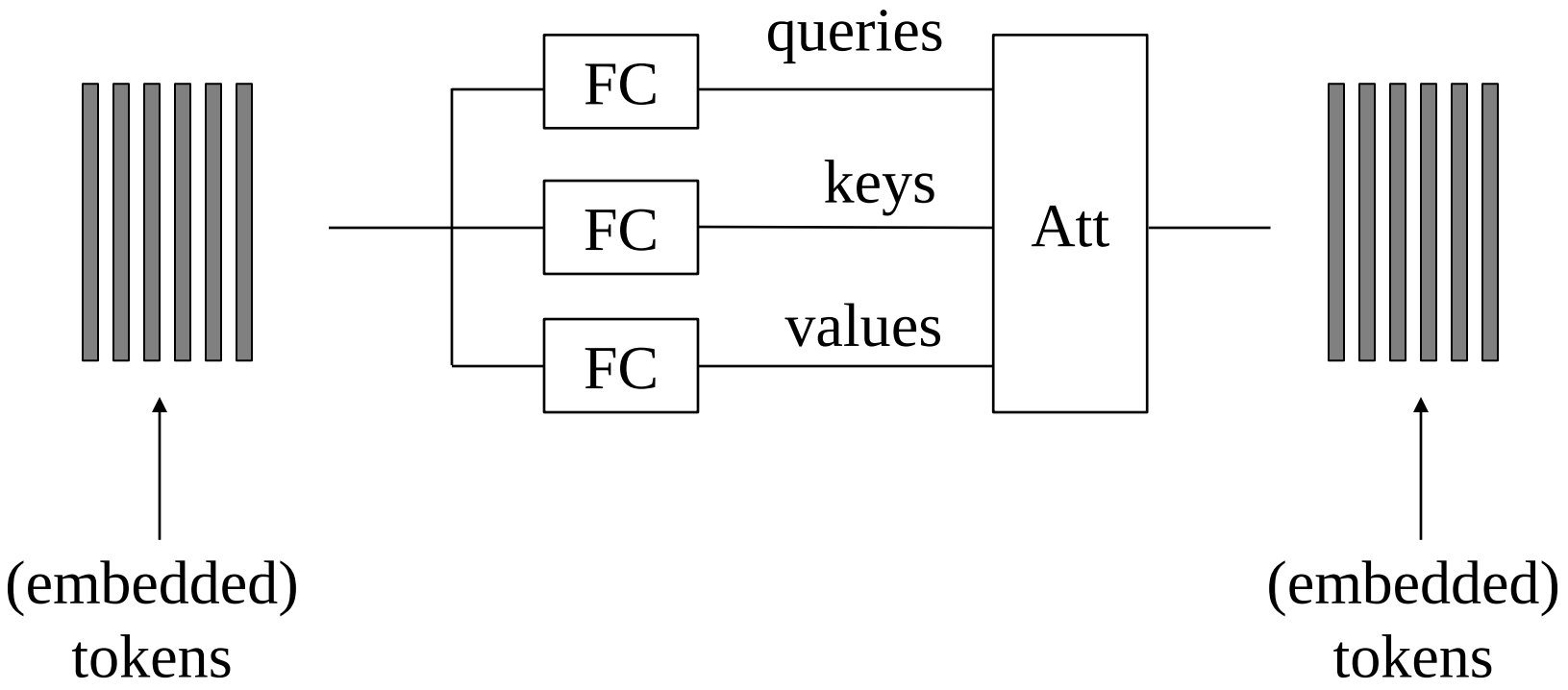
- we do it for all queries: $Q = \begin{pmatrix} q_0 \\ q_1 \\ \dots \\ q_{d-1} \end{pmatrix}$

Attention mechanism

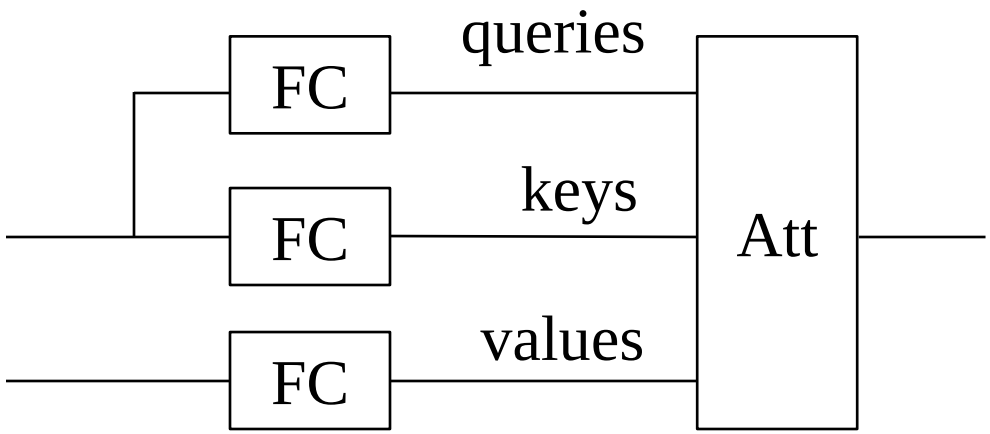


$$Att(Q, K, V) = \text{softmax}\left(\frac{QK^T}{t\sqrt{d}}\right)V$$

Self-Attention

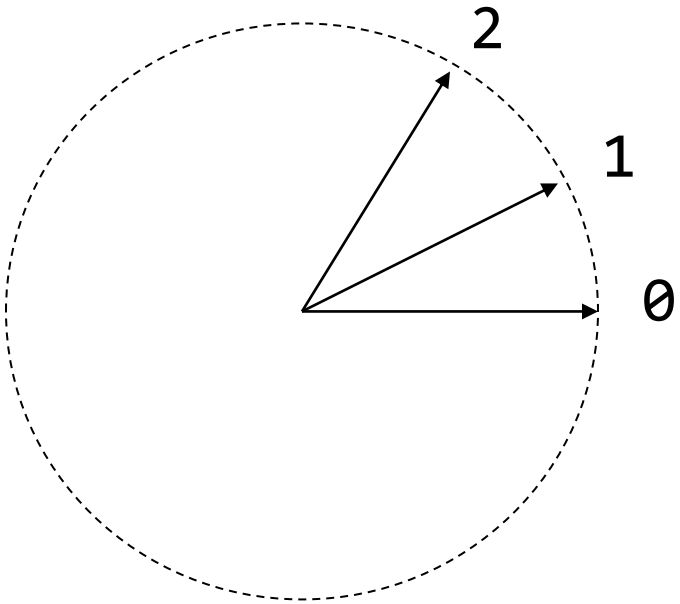


Cross-Attention



Positional Encoding

$$position(i) := [cos(i\phi), sin(i\phi)]$$



- The tokens' meaning depends on the position in the sentence
- Therefore, we can extend embedding by two features coding the position

Multi-head Attention

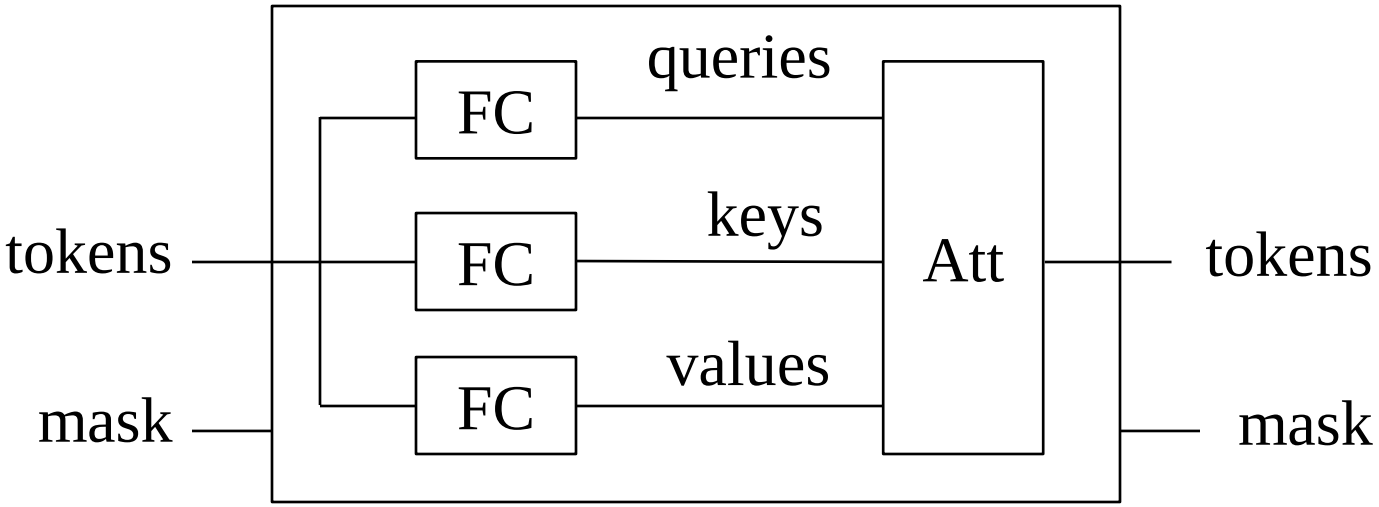
- We can use more attention blocks in parallel and concatenate the output
- In this way, we let individual heads to specialize

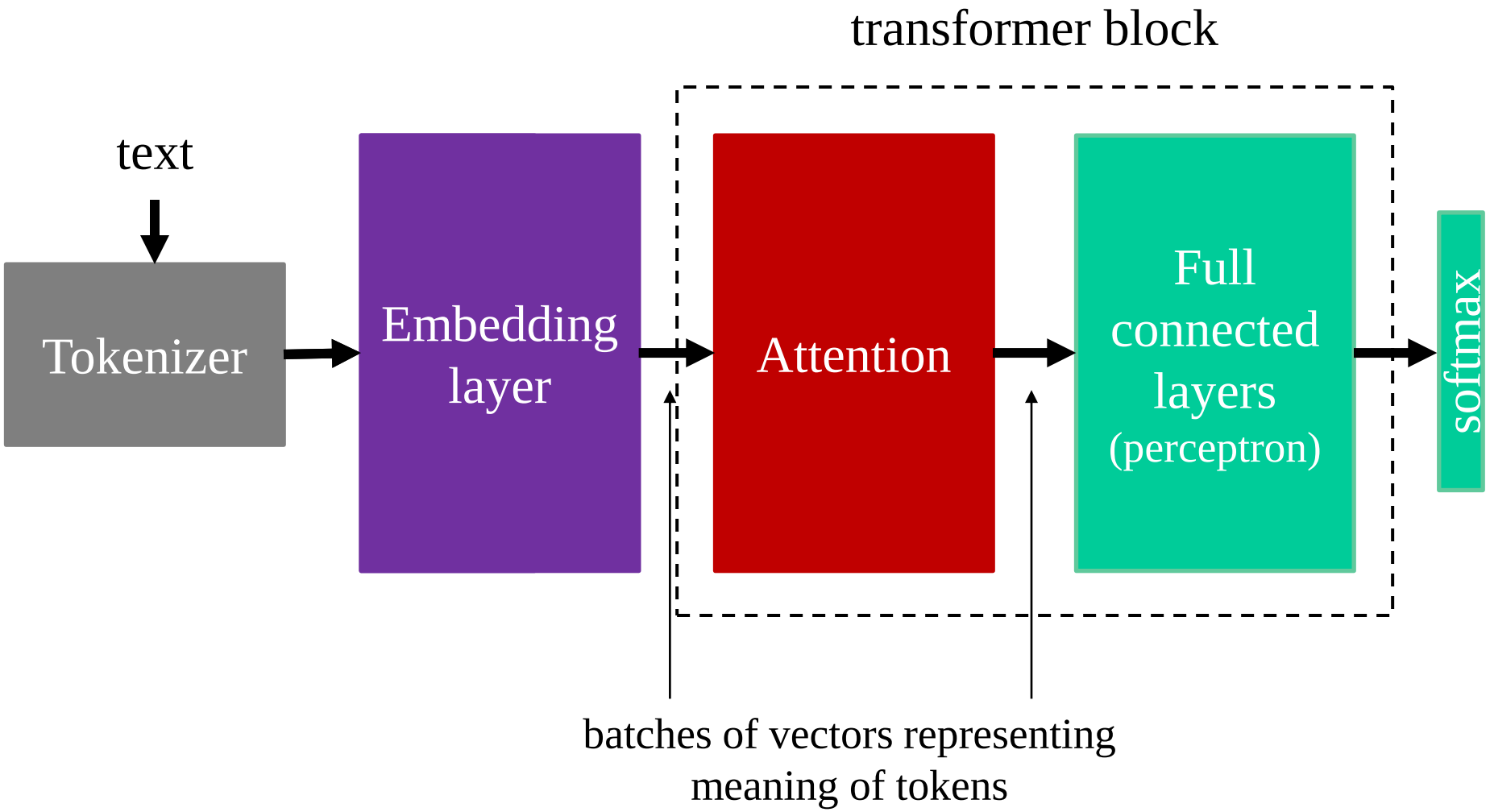
Masked Attention

- Attention is relatively slow since its computational complexity is the quadratic of the number of tokens
- We can decrease the number of the calculated scalar products by a mask

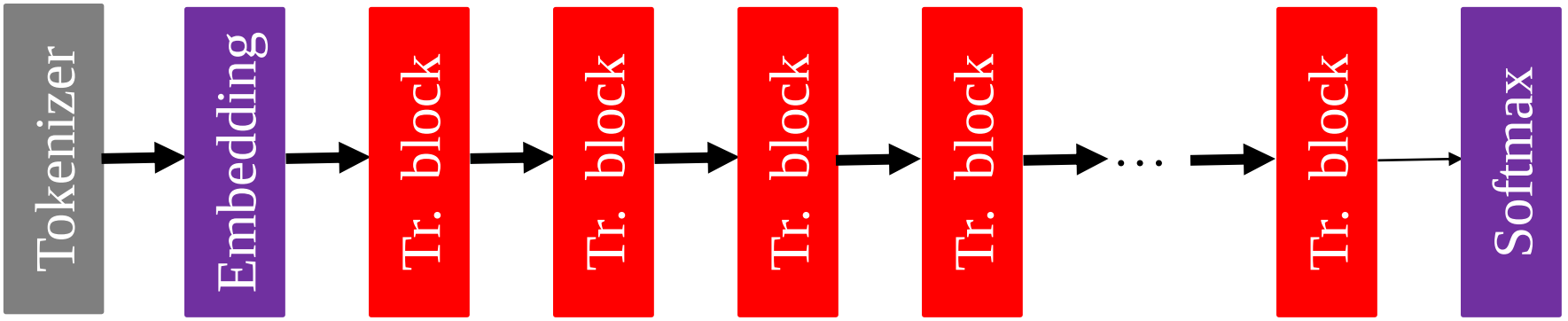
Deformed Attention

- The mask can be fixed or we can calculate it dynamically
- In this case, we call it deformed attention



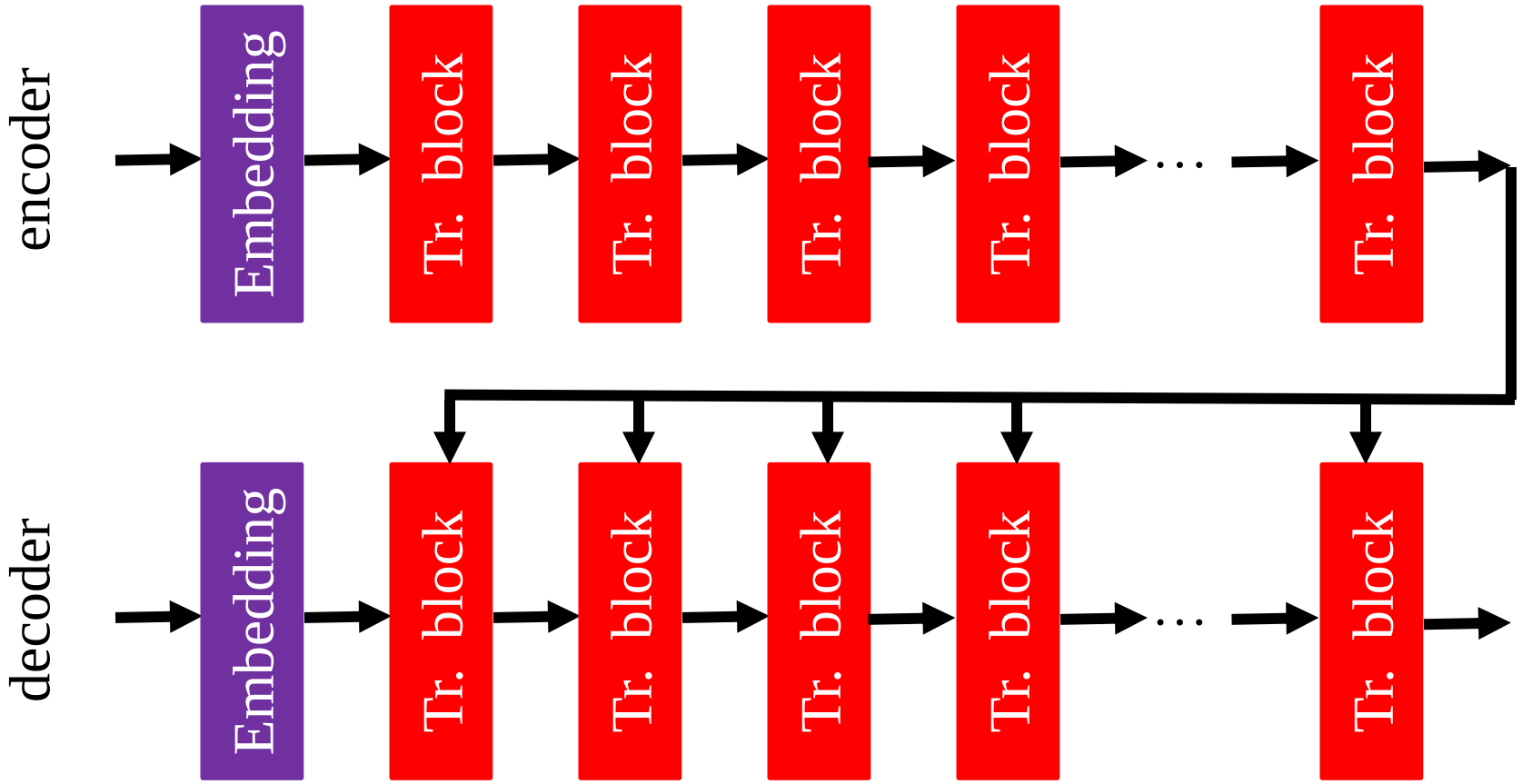

transformer block
text
Tokenizer
Embedding layer
Attention
Full connected layers (perceptron)
softmax
batches of vectors representing meaning of tokens

Transformer



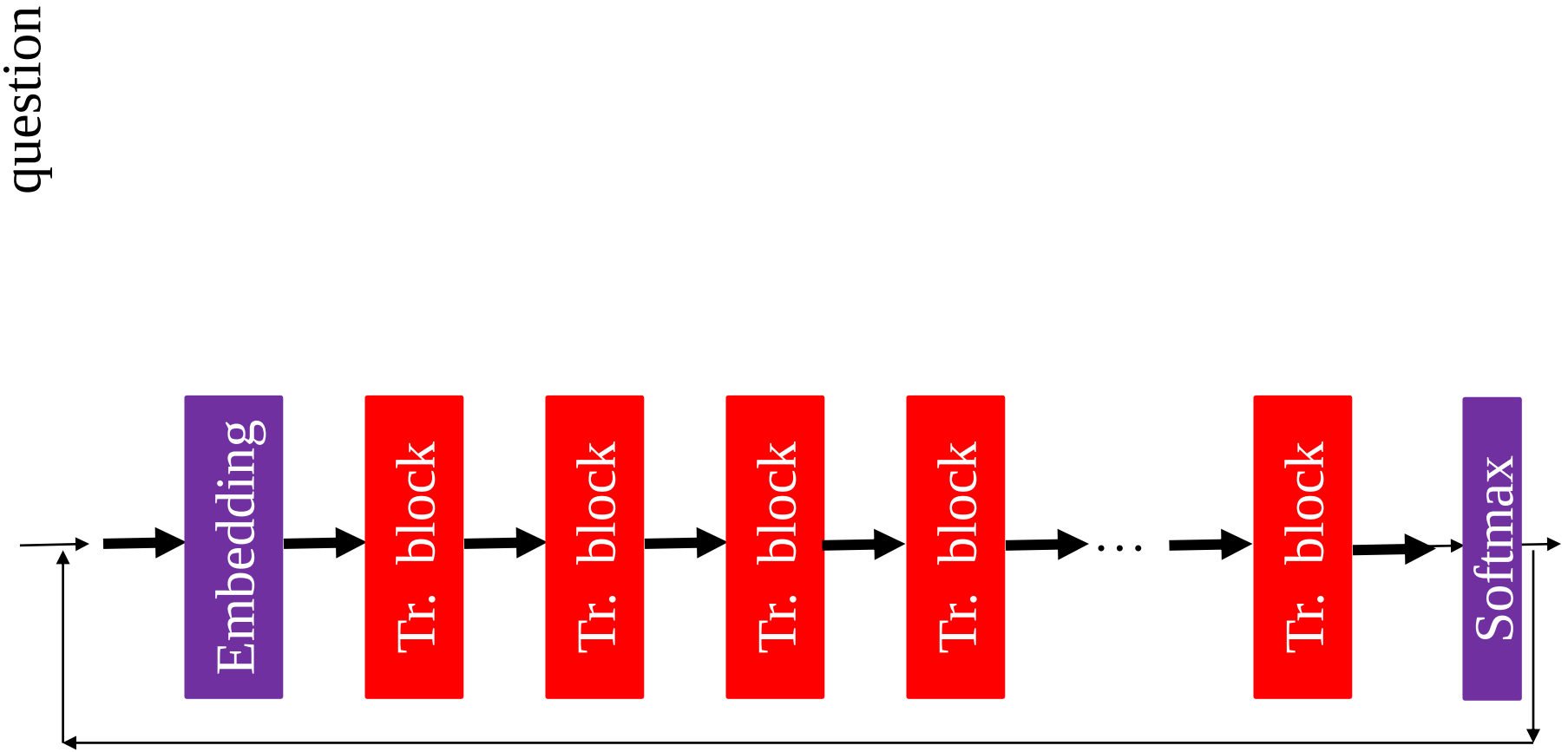
- Transformer is gradually processing a batch of tokens
- At the beginning they represent meaning of words or syllabi but gradually represent the global meaning
- Finally, (typically) the last one represents the prediction and the first one the classification

Encoder-Decoder architecture

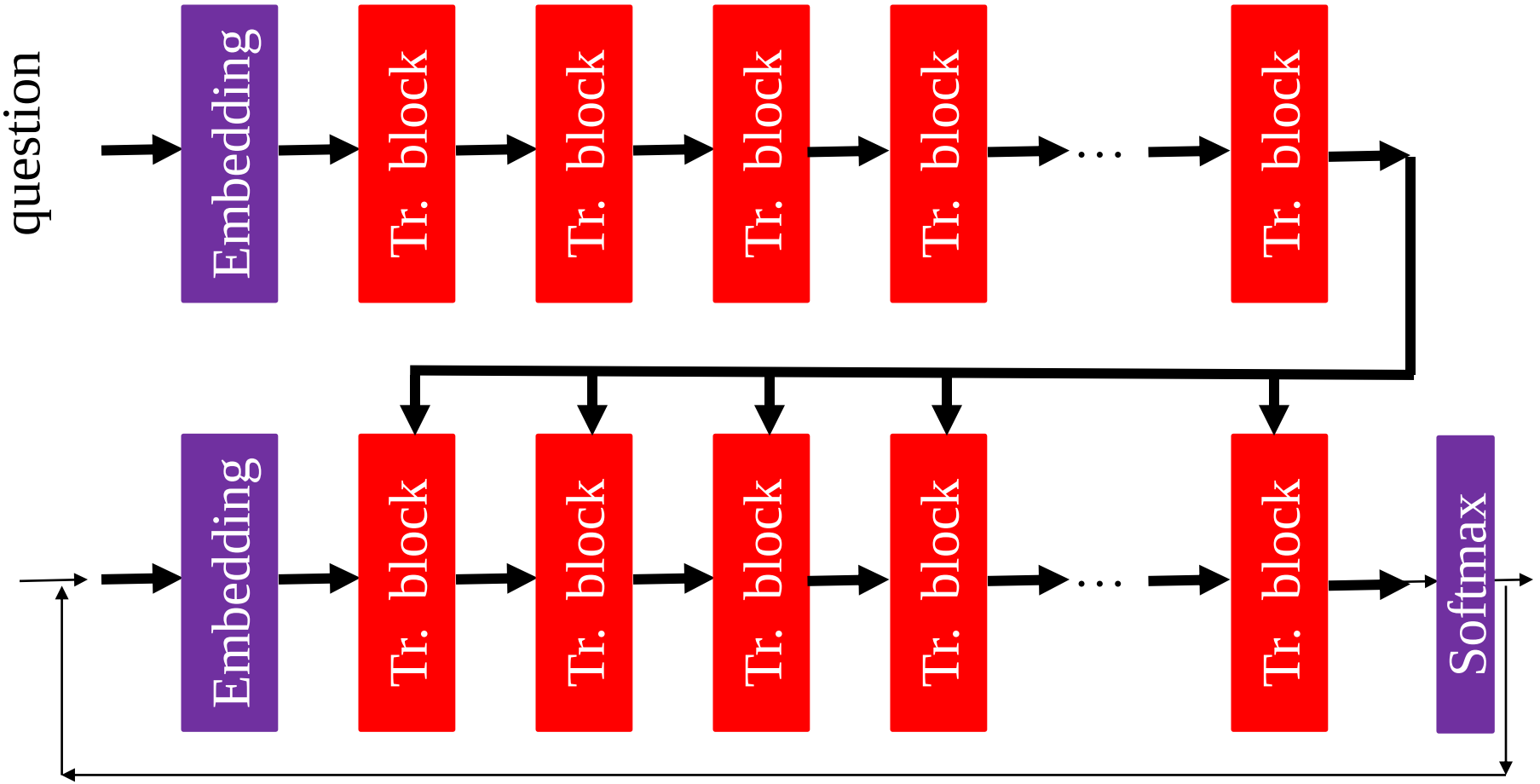


Encoder employs attention and Decoder emplys cross-attention

Text generator

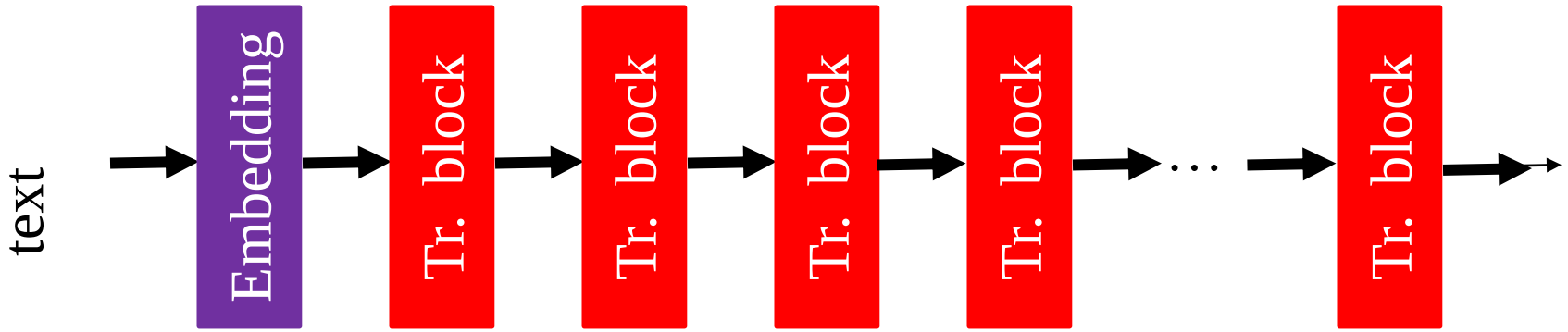


Chatbot



Chatbot is a regulated tare

Classifier



In-context learning

- The quality of first LLMs was not convincing. Therefore, the first thing that occurred to everyone who still needed an ideal answer from the model was whether it was possible to ask a better question to get a better answer.

- As a result, they discovered an interesting emergent phenomenon called **in-context learning**: the quality of the answer is positively affected when, together with the question, we enter additional information into the model

In-context learning

Human: "What is the capital of Slovakia?“
LLM: "Prague."

Human: "The capital of Slovakia is Bratislava. What is the capital of Slovakia?"
LLM: "Bratislava."

Of course, LLM does not learn anything, its parameters are fixed. However, the more precise context causes that more precise answer is generated.

Prompt engineering

In-context learning enables us to engineer the prompt.

Human to the robot: “Can you walk?“
We feed LLM with: “You are a robot with two hands, but no legs, … Can you walk?”
LLM: “No”

question

chain of thoughts