Large Language models (LLM)

- 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

Čo sa LLM učia:

- Vstupom je reprezentácia postupnosti tokenov (slov a/alebo kúskov slov)

- Výstupom je vektor pravdepodobností pre každý token, ktoré určujú buď
  - aký token bude nasledovať

  alebo
  - aký token je na vstupe zamaskovaný (nahradený špeciálnym tokenom pre masku)

Tokenizer: text to token ids

- It translates text, using a vocabulary for a particular language, into a sequence of token ids
- Token ids are integers from e.g. 0 to 32767 and correspond to words, syllabi (parts of words) or special marks

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

Embedding

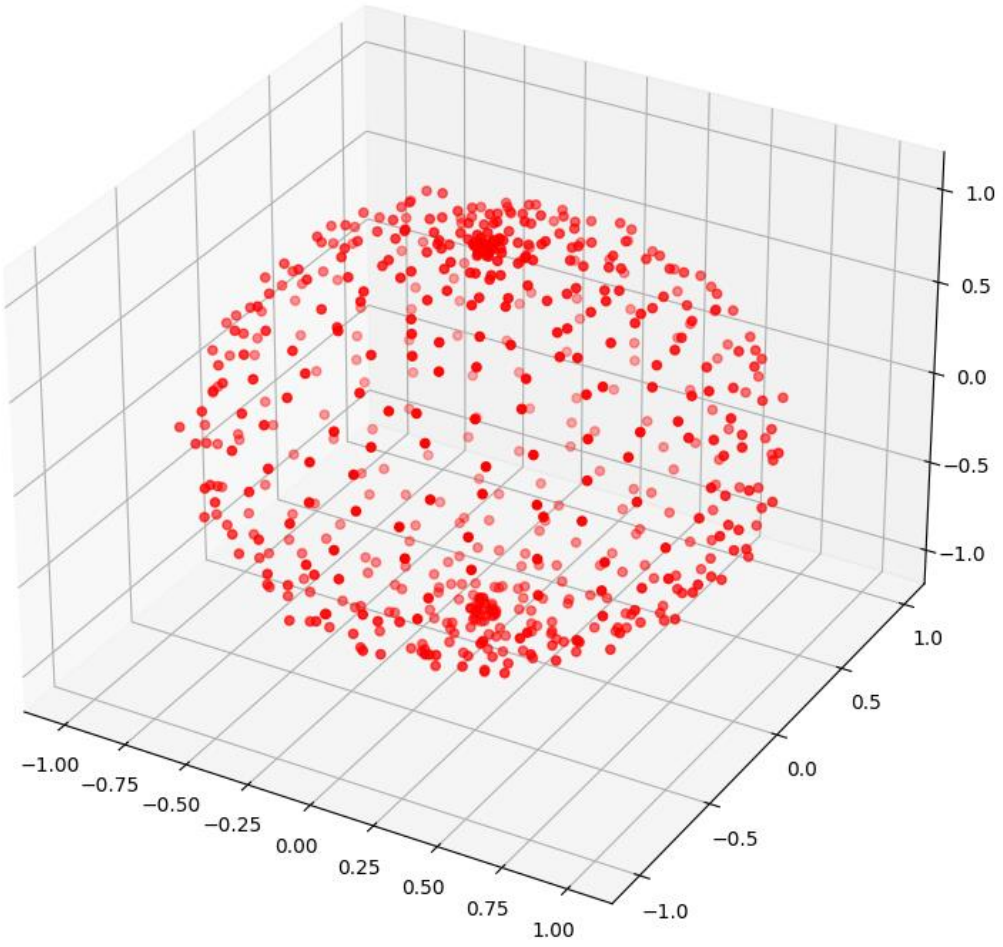
- It is a vocabulary that turns token ids into vectors (token embeddings)

```
    0  →  [-0.42, 1.21, -0.11, ..., 0.04]   <pad>
   27  →  [1.58, 4.22, 1.14, ..., -4.89]       I
 2132  →  [6.86, -13.95, -5.79, ..., -15.07]  van
  751  →  [4.69, 1.24, 9.22, ..., -6.04]      won
    3  →  [0.22, -0.28, -0.56, ..., -5.43]     a
    9  →  [1.34, -1.18, -2.81, ..., -5.98]
  443  →  [-3.48, -2.12, -24.10, ..., 1.49]   car
   16  →  [-7.46, 2.17, 0.29, ..., 4.52]       in
15363  →  [18.84, 8.61, 4.89, ..., -3.76]   Moscow
    1  →  [15.78, 7.15, 15.10, ..., 11.51]  <eos>
```

dimension: 768 for LaMini, 12288 for GPT3.5

Embedding

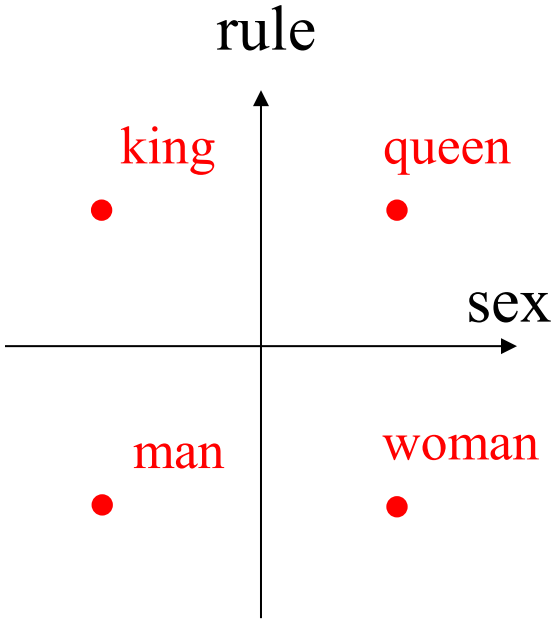
- Embeddings are well-organized: tokens with similar meanings are to be at similar spots.
- Typically, they are close to the surface of a hypersphere.



Embedding

We can design embedding manually for a small vocabulary

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

But!
LaMini: 768
GPT3.5: 12288
⟶ automation

Corpus

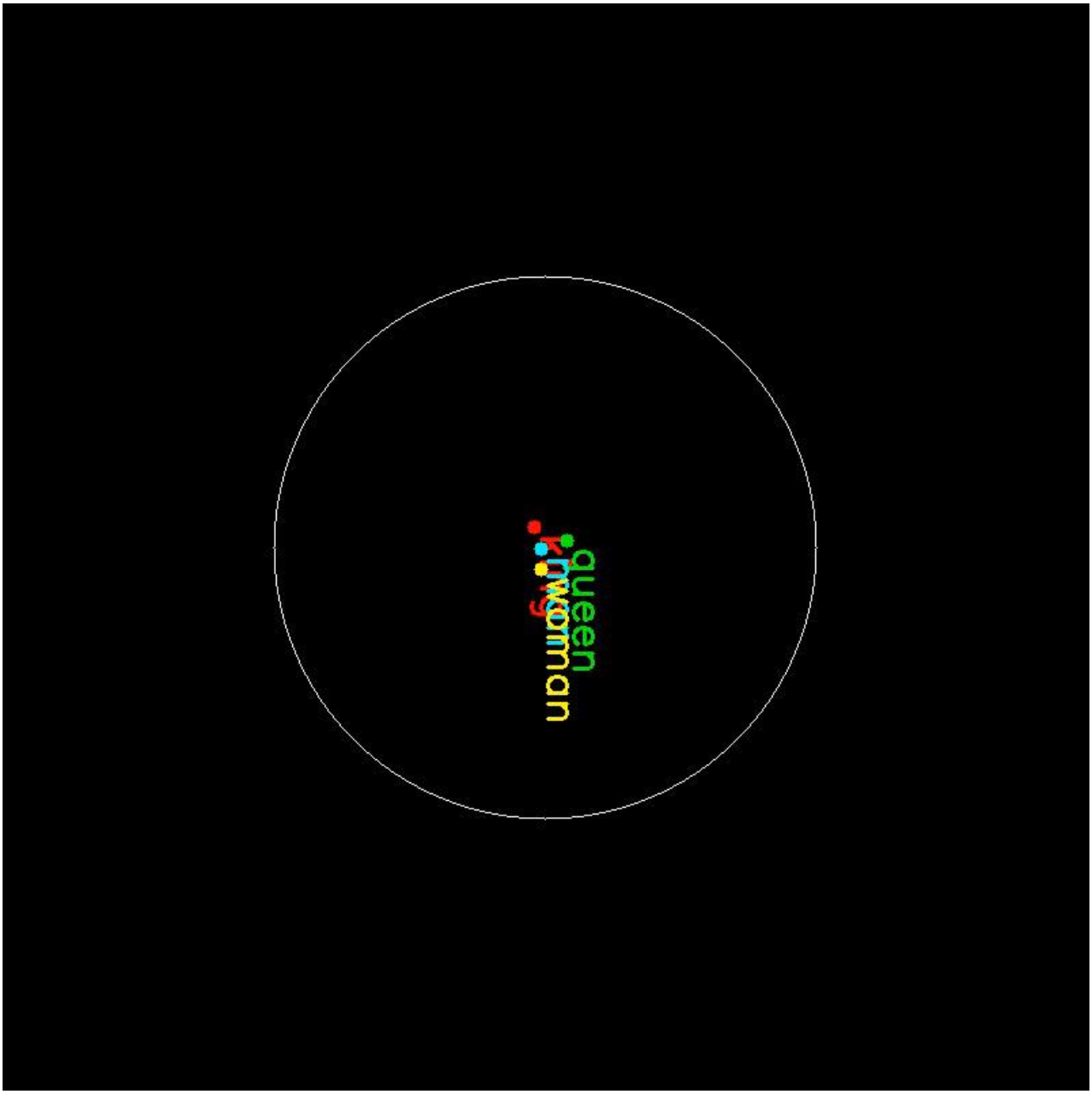
In the heart of a kingdom ruled by a wise and just king, there existed a delicate balance between the powers of man and woman. The king, adorned in regal robes, sat upon his throne, his gaze commanding the attention of all who entered his court. Beside him, his queen, a vision of grace and elegance,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

Training embedding



https://youtube.com/shorts/fxLWdGl7ZM8

Automated embedding fits our expectation

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

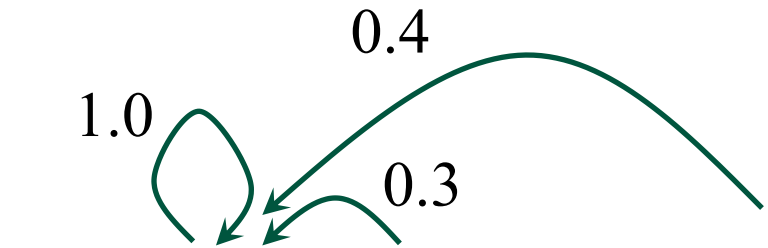
- We mix the token meaning with the tokens usually appearing in the same content.

ball → 0.8 ball + 0.1 shot + 0.1 net

(the soccer ball)

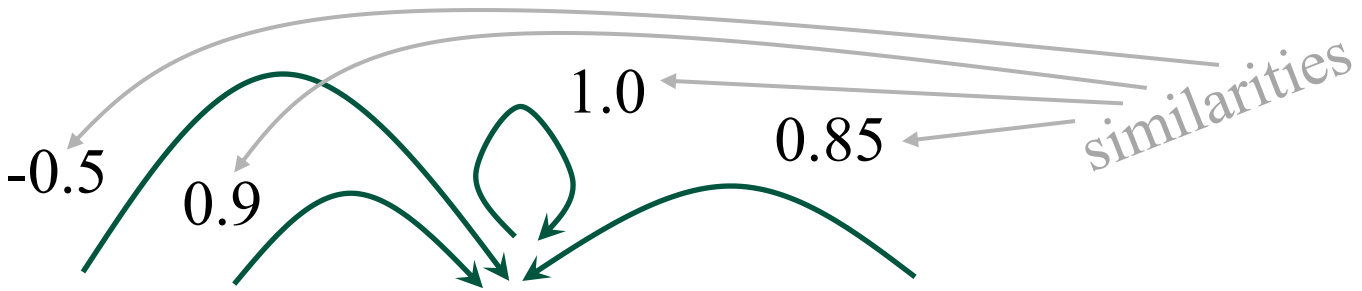
Solution: Attention

- 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)



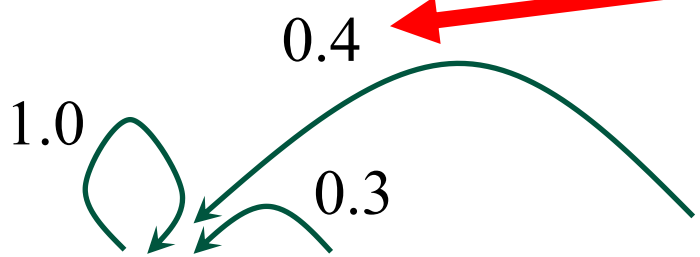
ball → 0.8 ball + 0.1 shot + 0.1 net

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

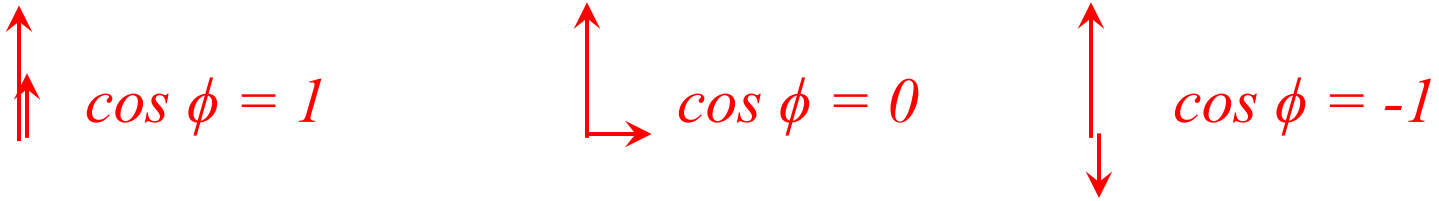
How do we calculate attention (the mixture weights) from similarities?

Cosine similarity

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

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

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

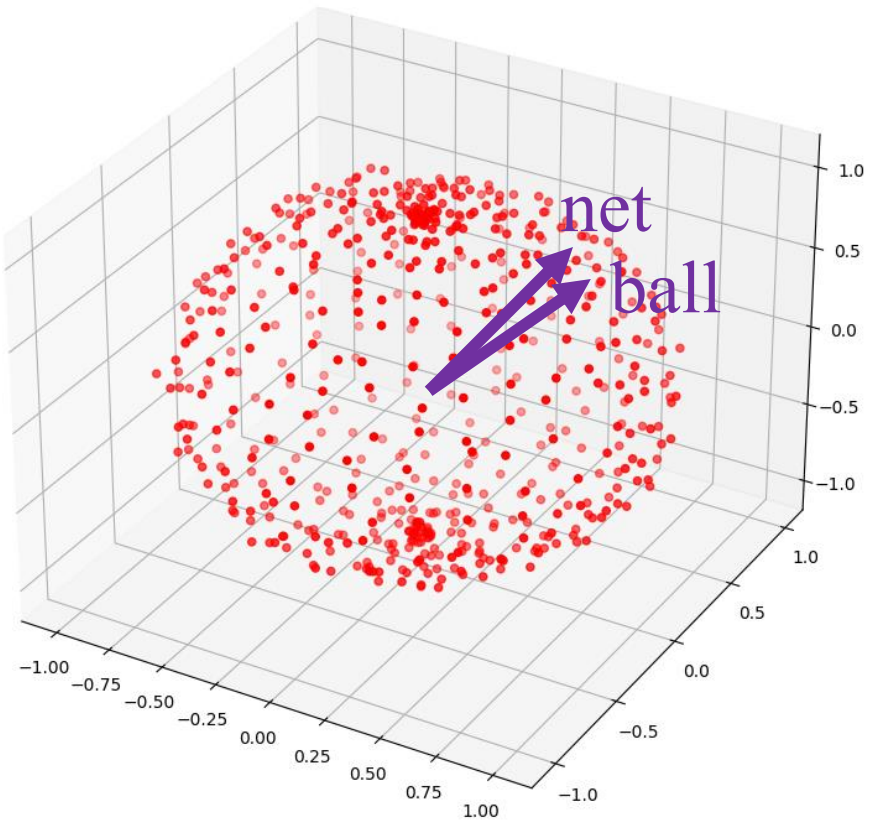


$(u_1, u_2, \ldots, u_N)$ has a magnitude (size):

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

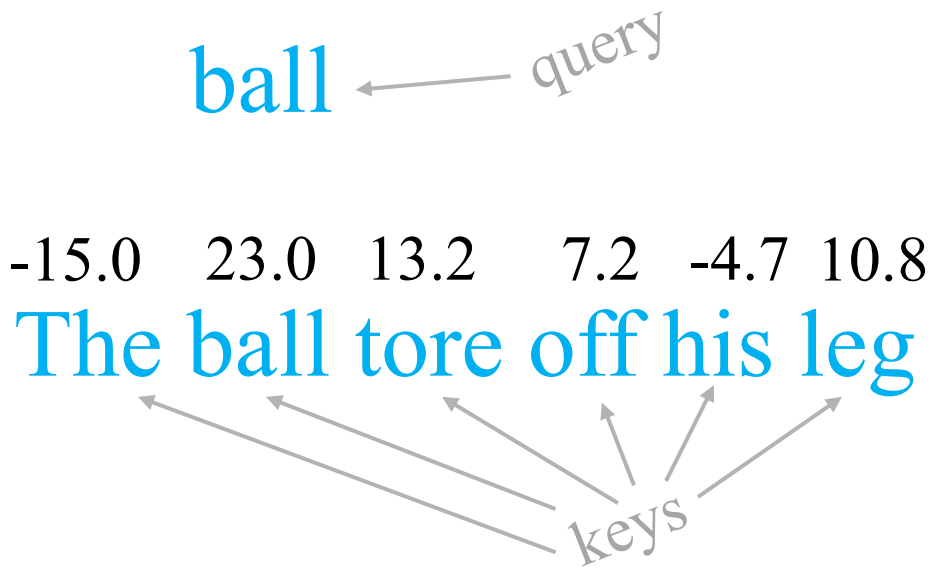
How do we calculate similarities? 1

- 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

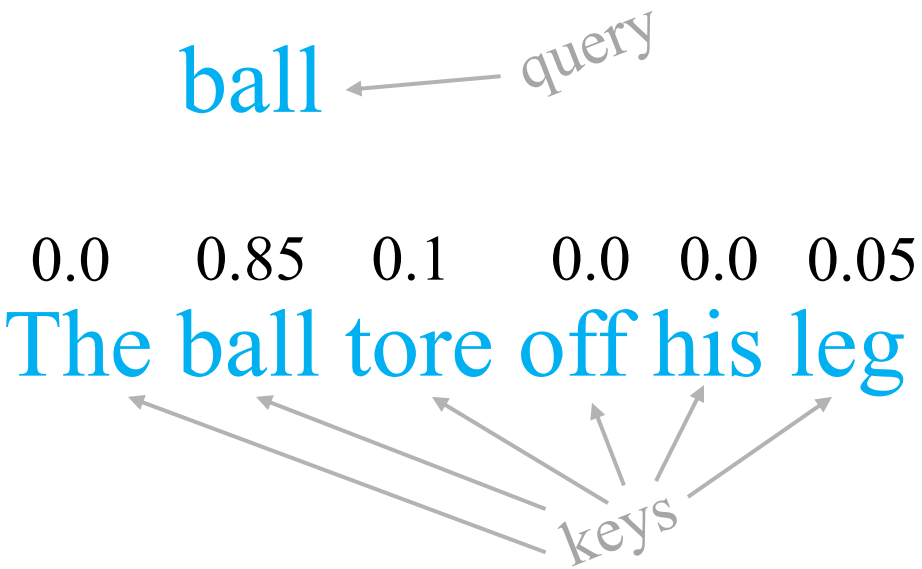
- So, we compare a token (query) with each token (keys)
- We calculate the scalar product of the query with all keys and get a vector of similarities



How do we calculate attention? 2

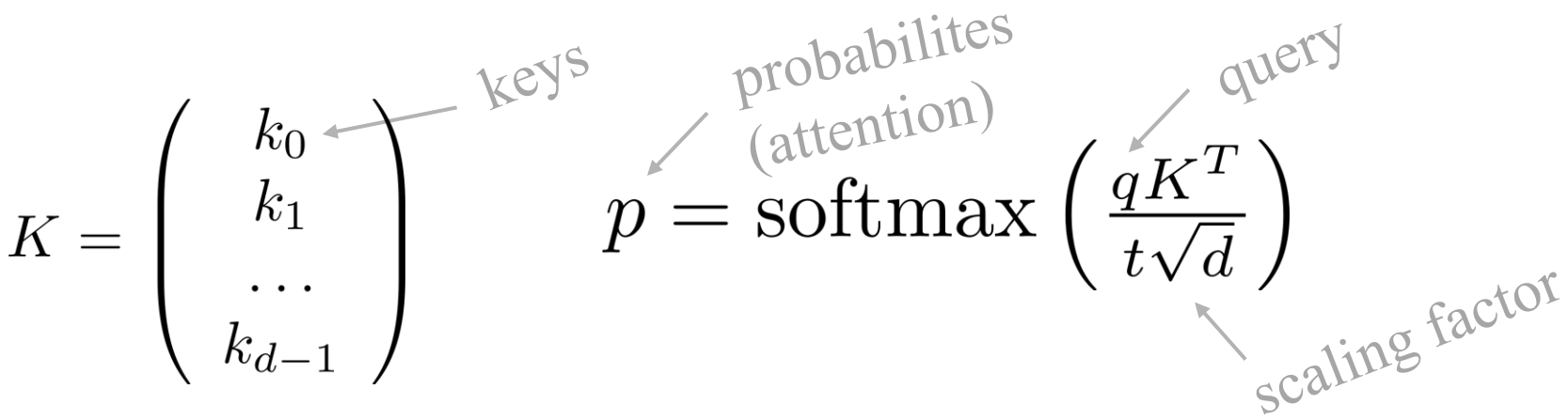
- The mixture weights are probabilities. So, we can calulate them by the Softmax function

$$softmax_i(x) = \frac{e^{x_i}}{\sum_k e^{x_k}}$$



How do we calculate attention? 2

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



- 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 of the keys' dimension *d*, but it can be slightly changed by temperature t close to 1.0

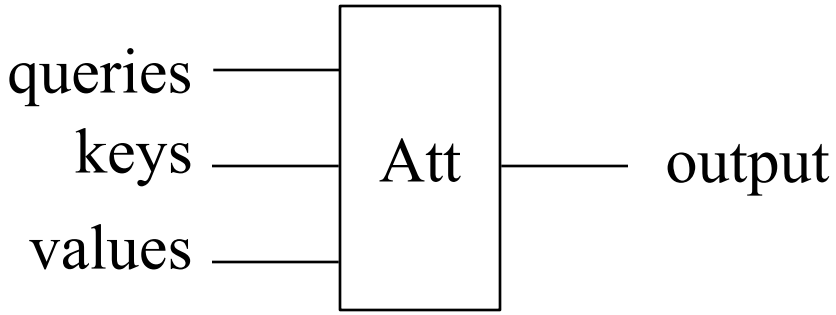
How do we calculate attention? 2

- 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: $Q = \begin{pmatrix} q_0 \\ q_1 \\ \dots \\ q_{d-1} \end{pmatrix}$

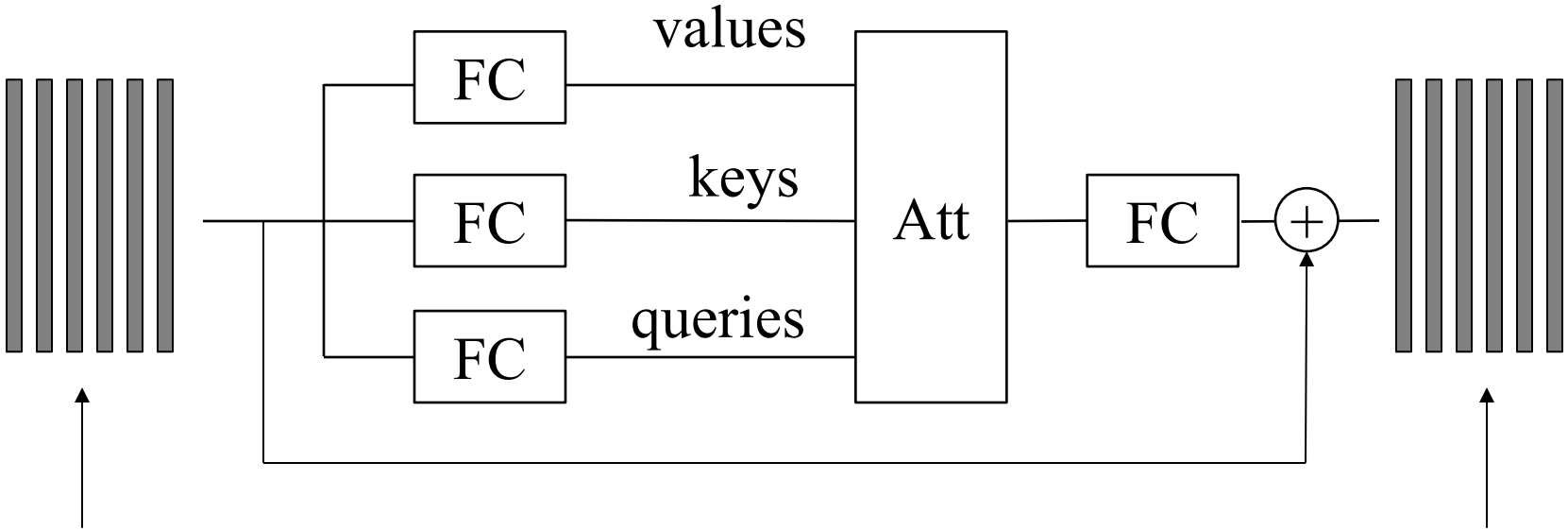
Attention mechanism



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

temperature $1.0 \pm \varepsilon$

Self-Attention

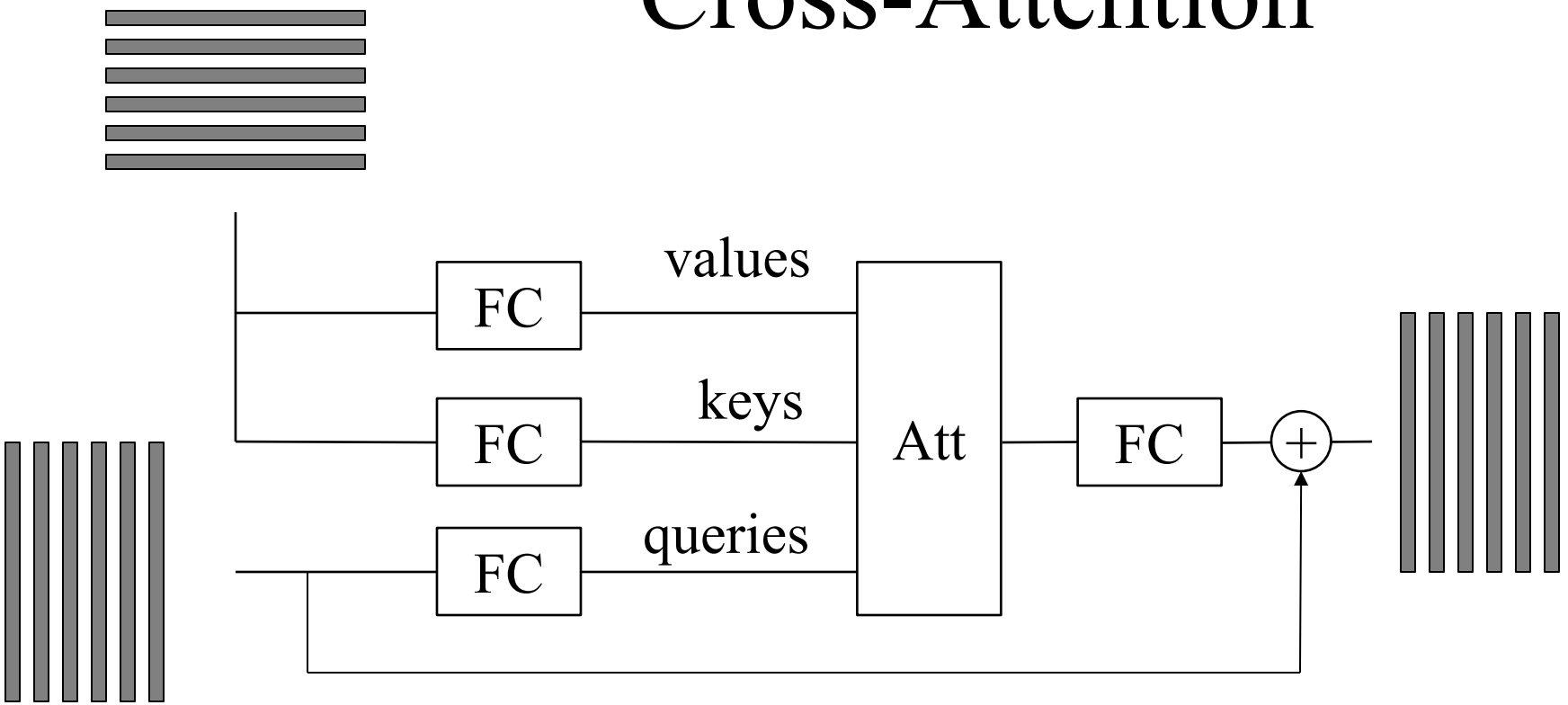


(embedded)
tokens

počíta ako sa mení
reprezentácia tokenu v kontexte
ostatných tokenov

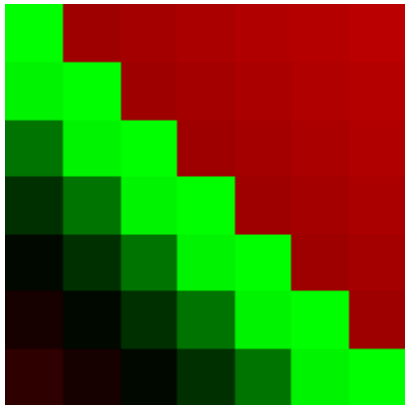
(embedded)
tokens

Cross-Attention

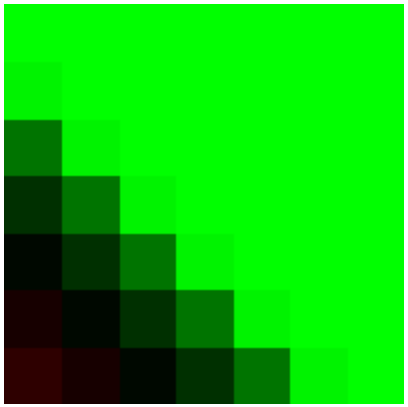


počíta ako sa mení
reprezentácia tokenu v kontexte
sady iných tokenov

Masked Attention



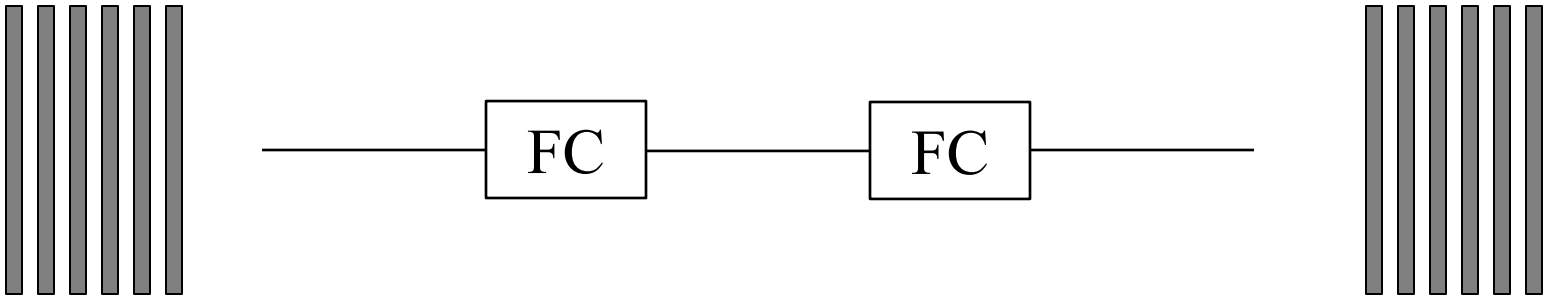
bidirectional
(encoder)

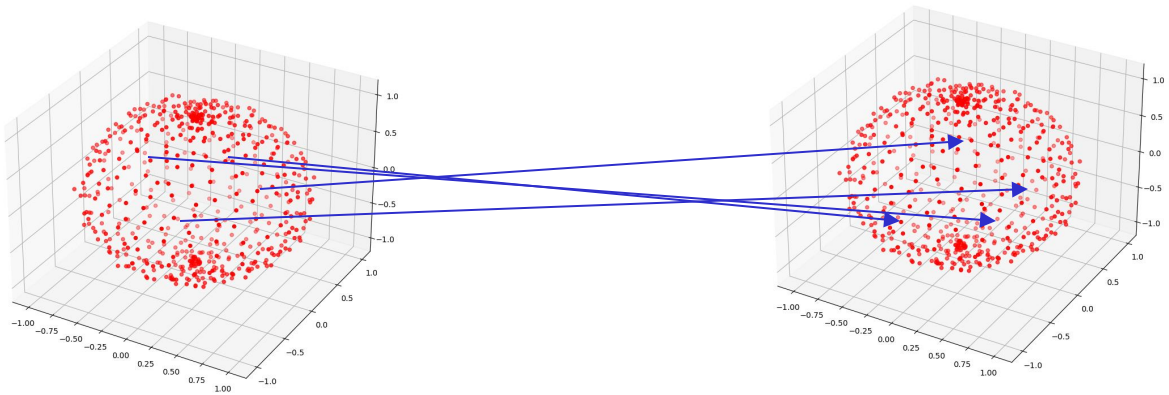


unidirectional
(decoder)

reprezentuje vplyv relatívnej polohy vplývajúceho slova na slovo

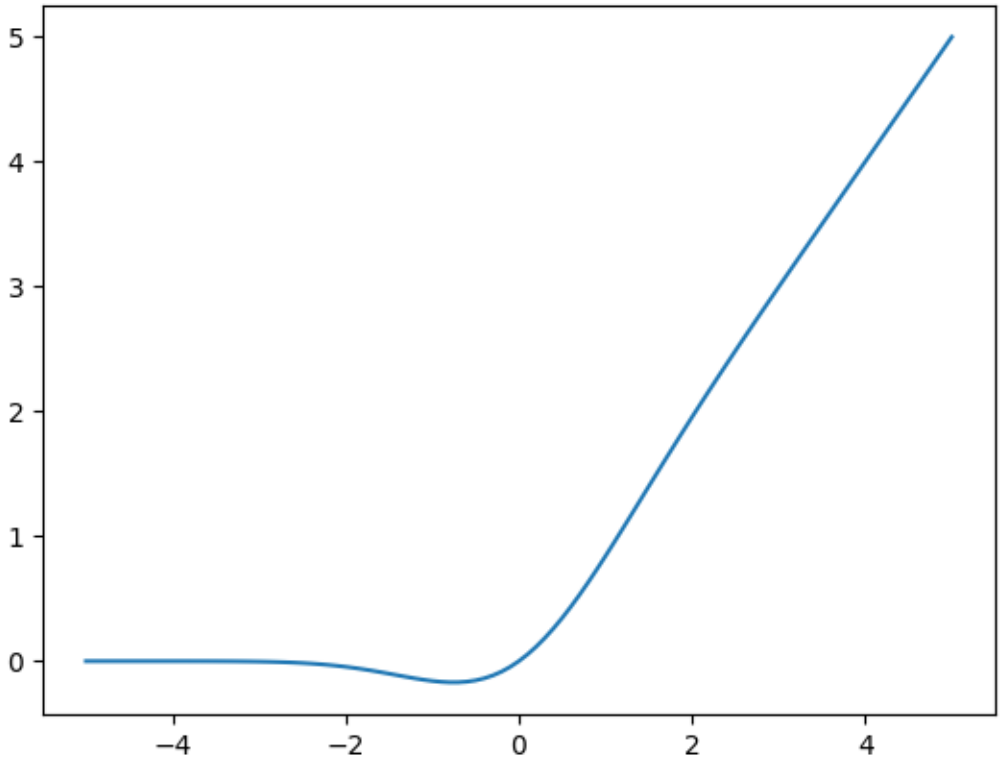
Feed Forward Network





GELU (Gaussian Error Linear Unit)

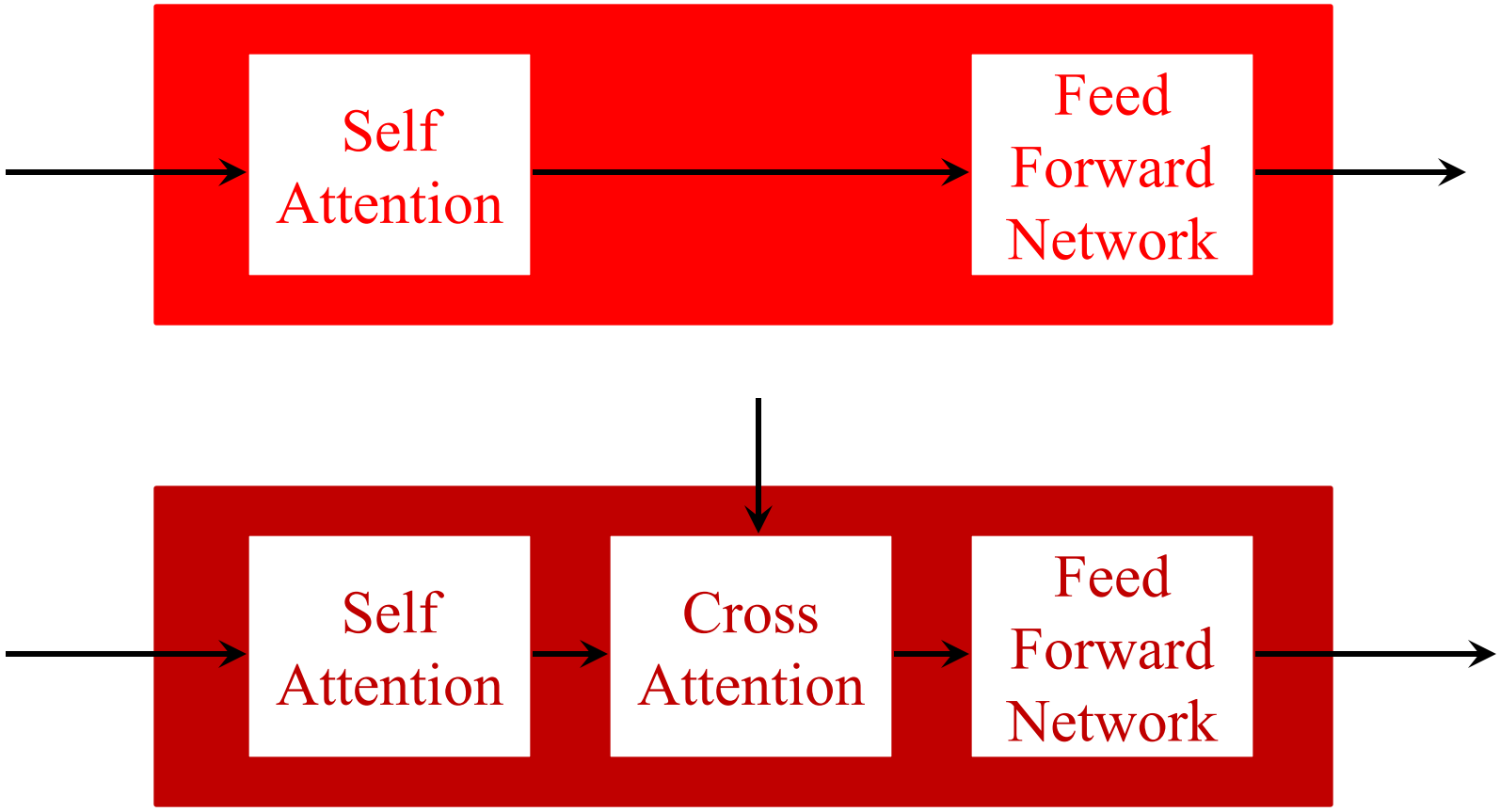
$$\frac{1}{2}x\left(1+\tanh\left(\sqrt{\frac{2}{\pi}}(x+0.044715x^3)\right)\right)$$



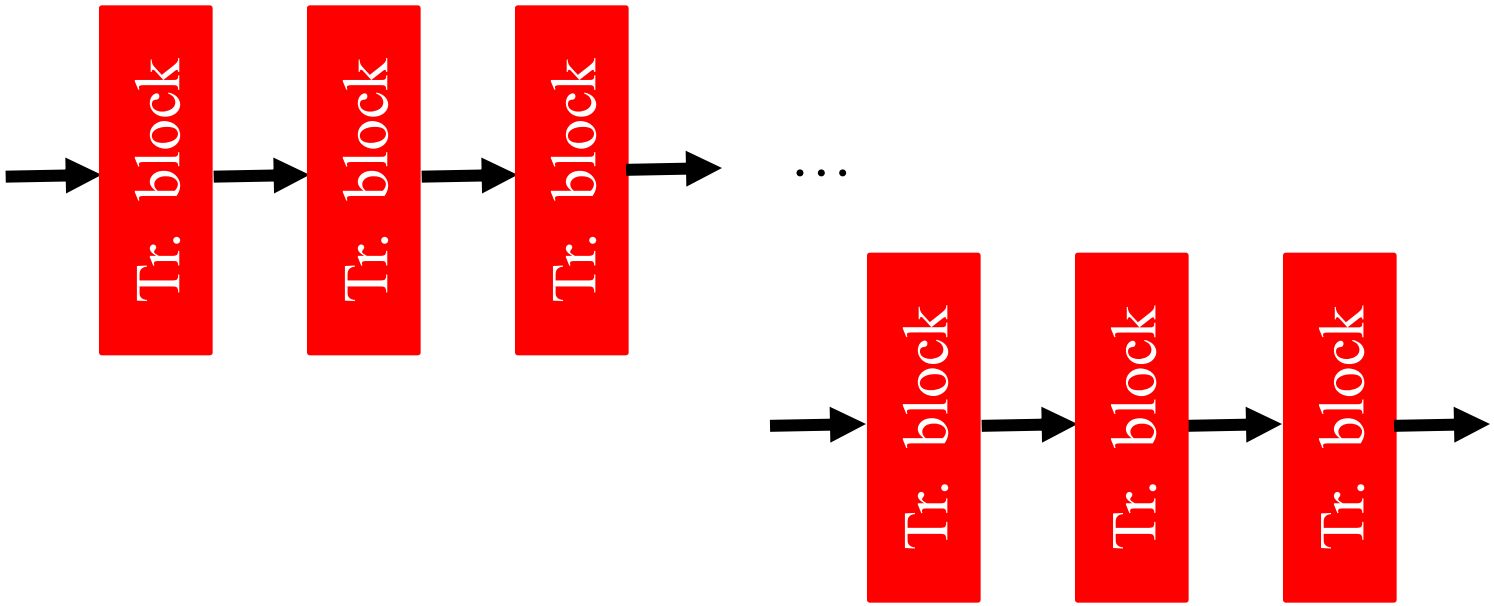
Multi-head

- Self či cross attention modulov môžeme postaviť vedľa seba viacero a ich výstup spojiť dohromady (concatenation)
- FFN potom musí výstup attention nielen spracovať ale aj zlúčiť (zredukovať dimenziu)
- napríklad LaMini má 12 hláv

Transformer blocks

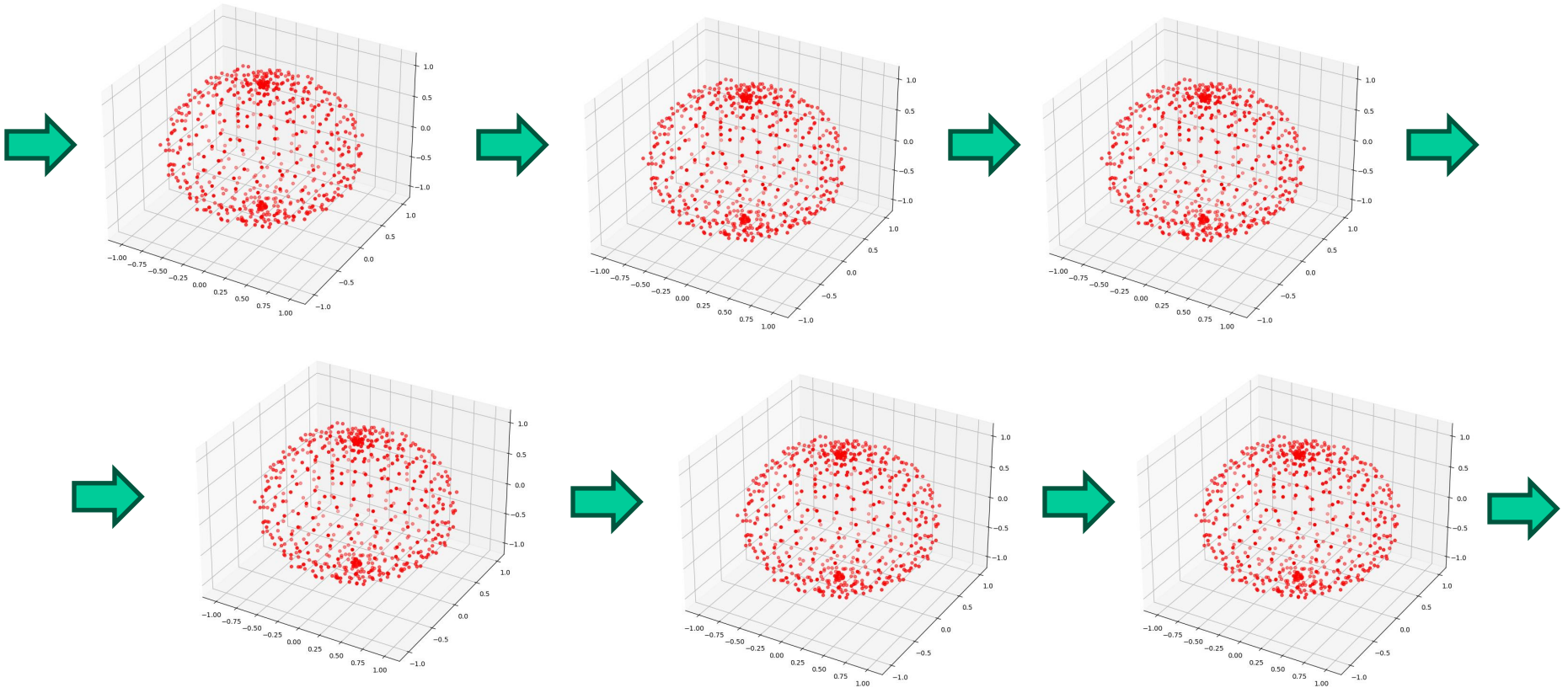


Transformer



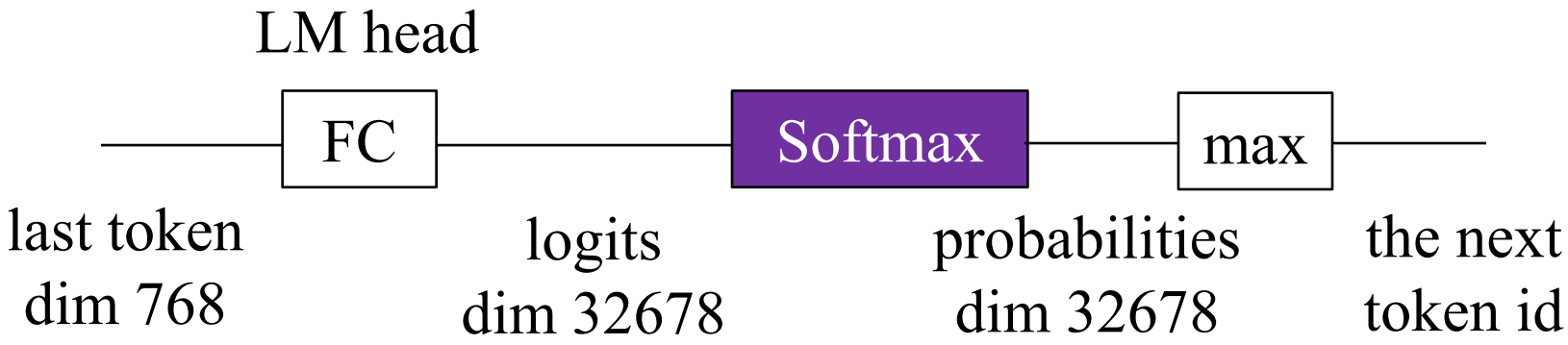
- Transformer is gradually processing a batch of tokens
- At the beginning they represent meaning of words or syllabi but gradually represent the global meaning
- Finally, (typically) the last one represents the prediction and the first one the classification

Transformer



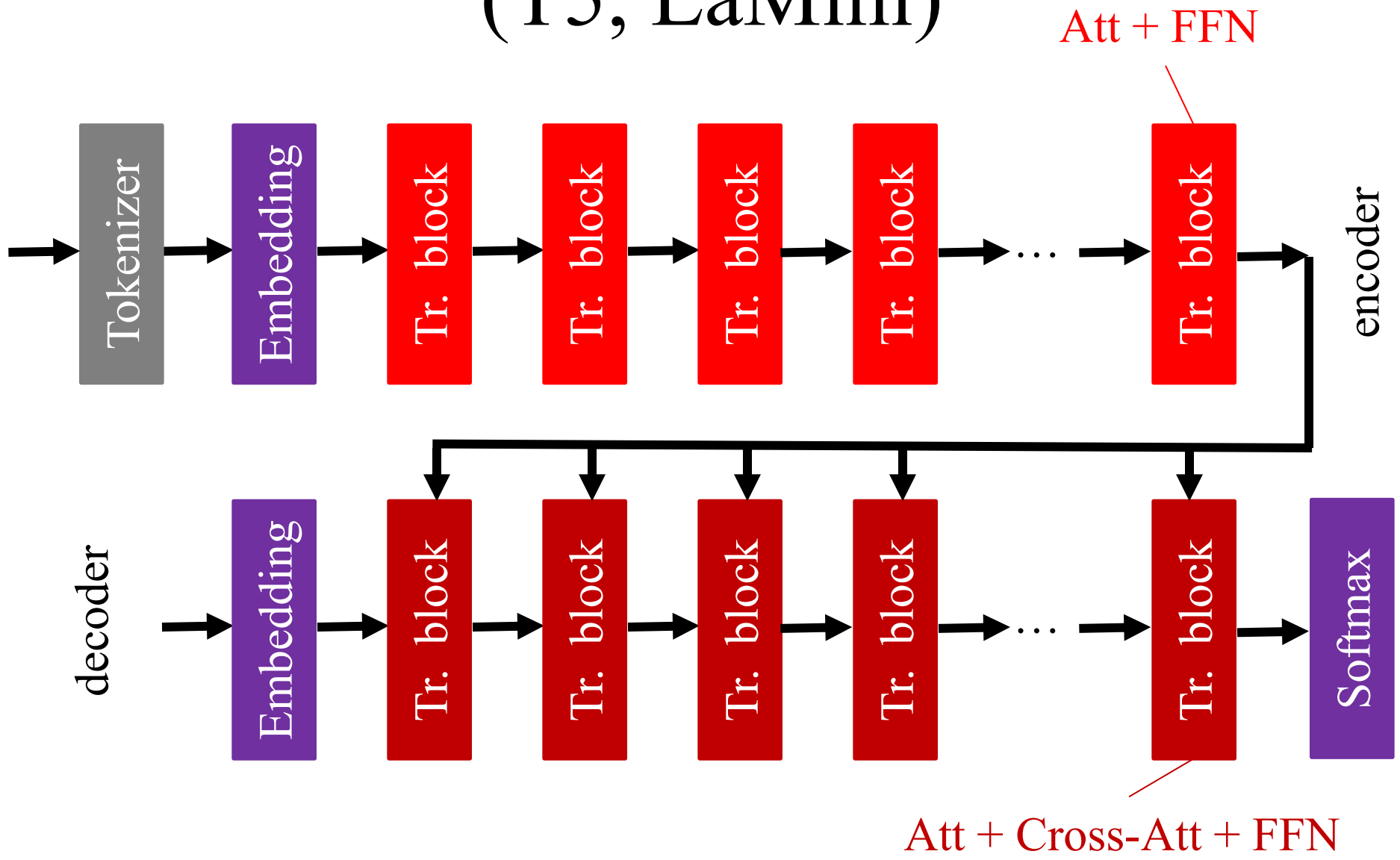
LM Head & Softmax (wipe-out)

- Text generation resides in output of probabilities for each possible token ids

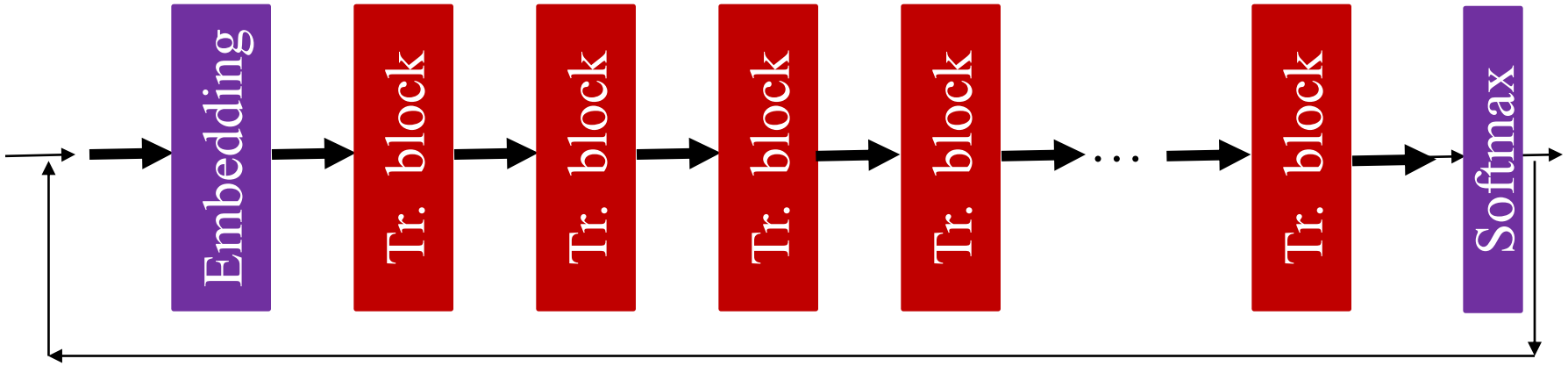


$$softmax_i(x) = \frac{e^{x_i}}{\sum_k e^{x_k}}$$

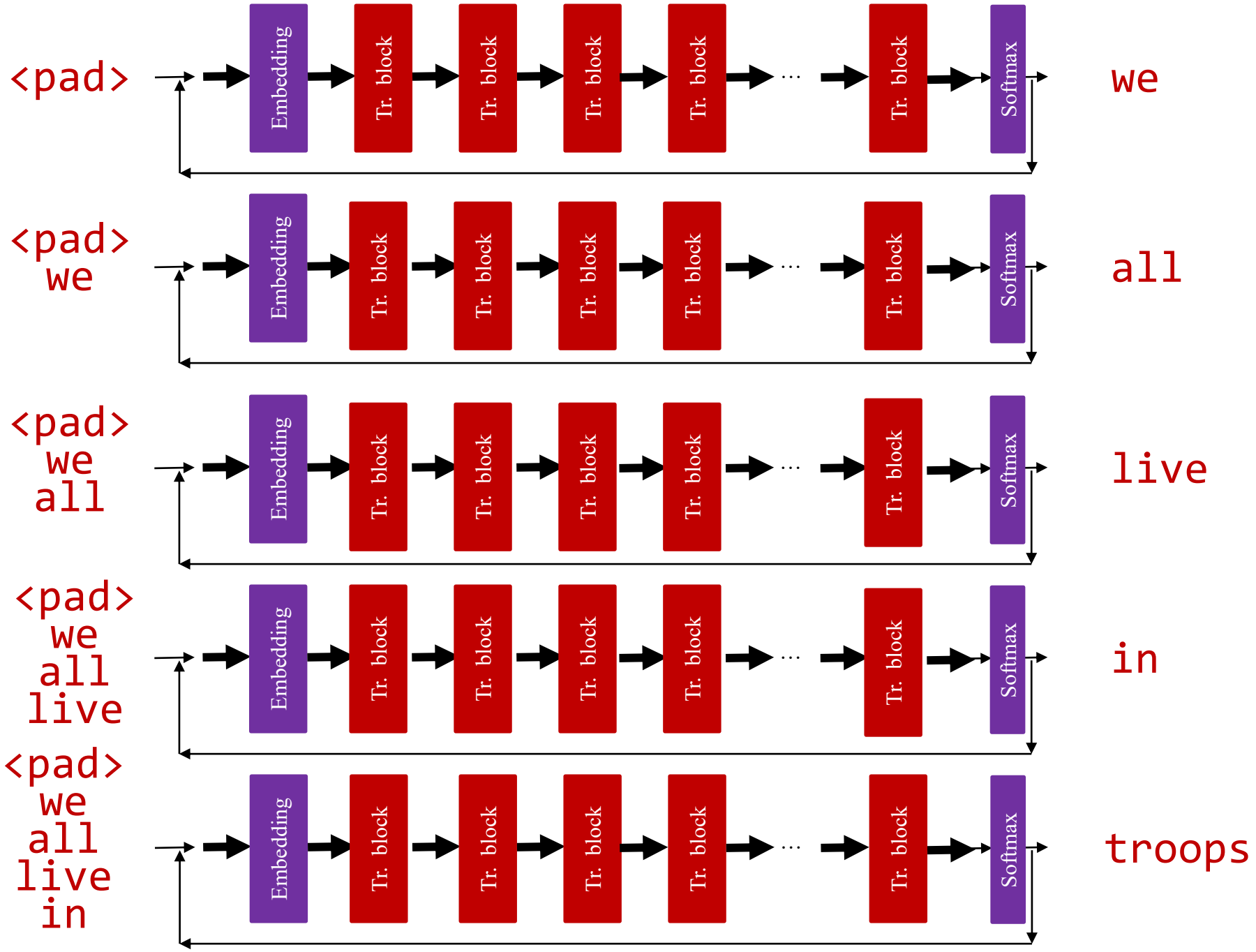
Encoder-Decoder architecture (T5, LaMini)



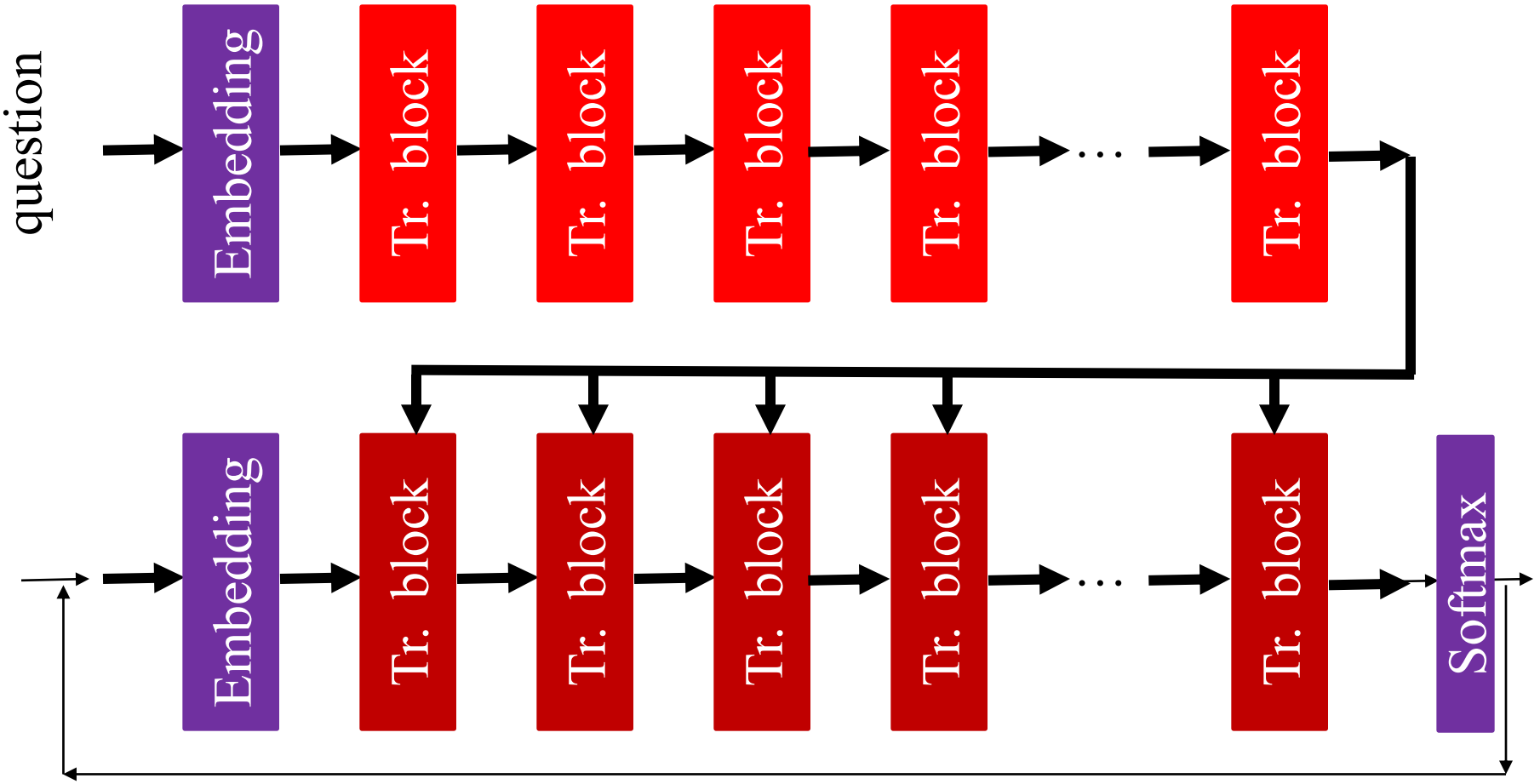
Text generator



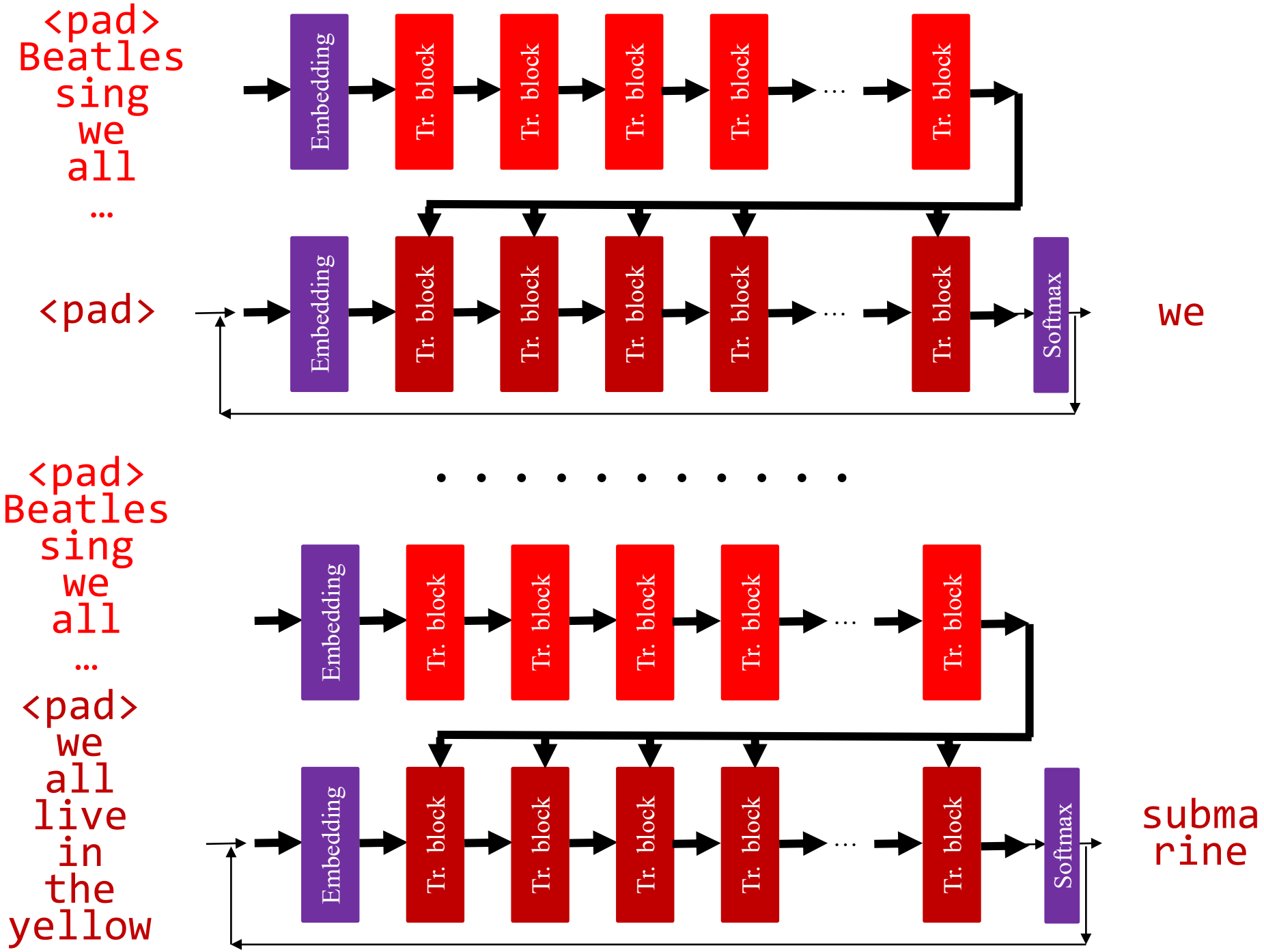
tare


<pad>
Embedding
Tr. block
Tr. block
Tr. block
Tr. block
...
Tr. block
Softmax
we
<pad>
we
Embedding
Tr. block
Tr. block
Tr. block
Tr. block
...
Tr. block
Softmax
all
<pad>
we
all
Embedding
Tr. block
Tr. block
Tr. block
Tr. block
...
Tr. block
Softmax
live
<pad>
we
all
live
Embedding
Tr. block
Tr. block
Tr. block
Tr. block
...
Tr. block
Softmax
in
<pad>
we
all
live
in
Embedding
Tr. block
Tr. block
Tr. block
Tr. block
...
Tr. block
Softmax
troops

Chatbot



Chatbot is a regulated tare


<pad>
Beatles
sing
we
all
…
<pad>
Embedding
Tr. block
Softmax
we
<pad>
Beatles
sing
we
all
…
<pad>
we
all
live
in
the
yellow
submarine

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