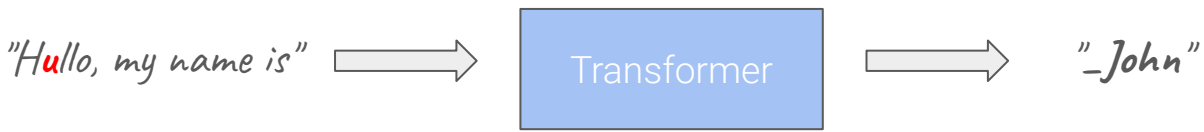
AI – Under the hood

Mihály Bárász, 2024-09-17

The core of a chat bot is a **module**, which **continues whatever input**.



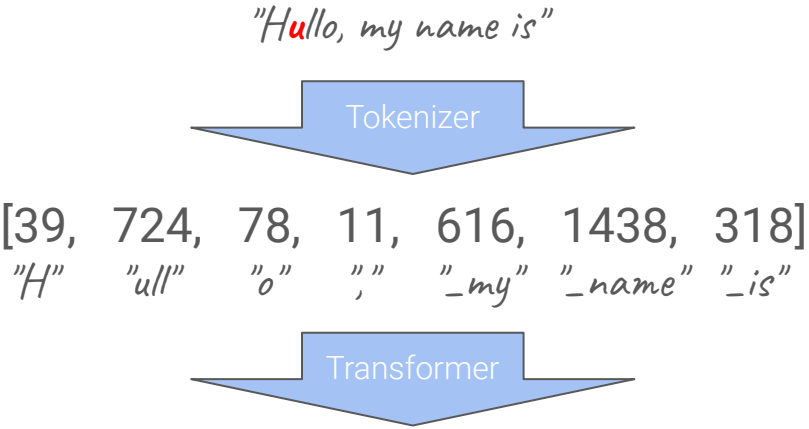
The transformer eats **tokens**, and outputs **predictions**

A fix set of tokens: ~50000 short combinations of characters often found in human text

The tokens are not words, nor characters. They are statistically frequently occurring combinations of characters, typically 1..10 characters long, e.g.

"_name", "_is", "ull", "o"...

Execution of one step:



Predictions:

Token	Likelihood	Value
0	0.00...	
1	0.00...	
...		
367	0.018	"_H"
...		
1757	0.023	"_John"
...		
500257		

Tokens are **converted to vectors**, which flow through the transformer

Input:

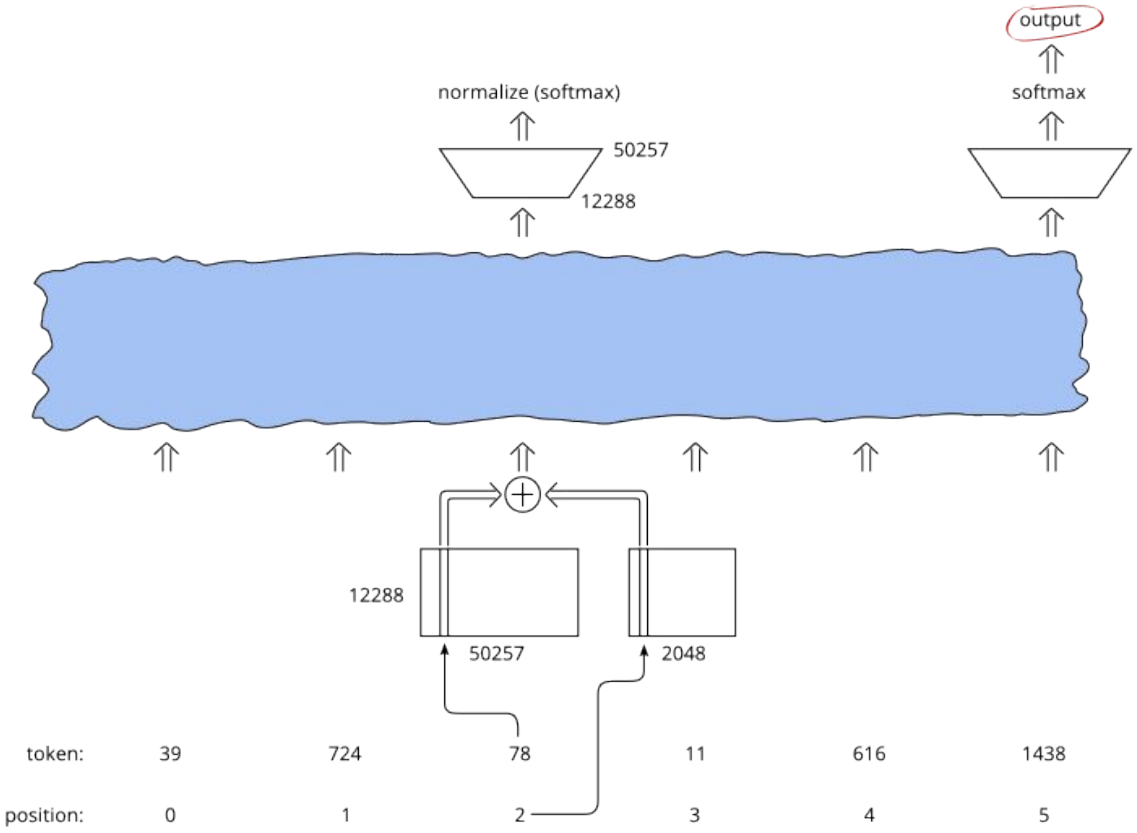
“Embedding” of each token

For each (token, position) pair, two vectors are picked from *embedding* matrices (lookup tables), and added together. The resulting vectors of 12288 elements are the starting point of the computation.

Output:

Prediction vector per stream (per input token)

Final value in each stream is multiplied by a 50257×12288 *unembedding* matrix to get a 50257 element vector: a relative log-probability for every possible token. These can be normalized by the “softmax” function to get probabilities, and the last one is a prediction for *next* token.



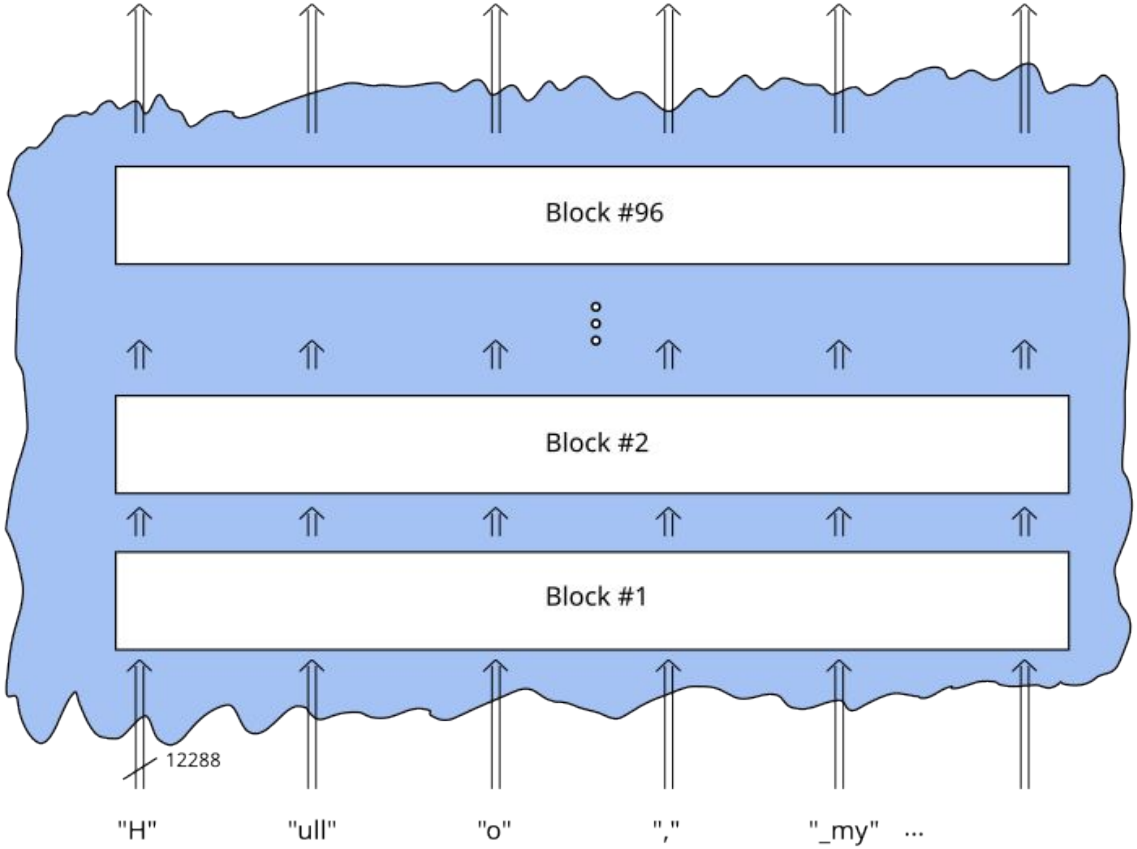
As **vectors** travel from block to block, they **gather** more and **more information**

Tokens are now vectors…

After embedding, each token becomes a vector of 12288 real numbers.

…transformed by each block as they travel upwards.

Each block transforms the vector of each stream, taking input from values in the channels to the left of the given channel.



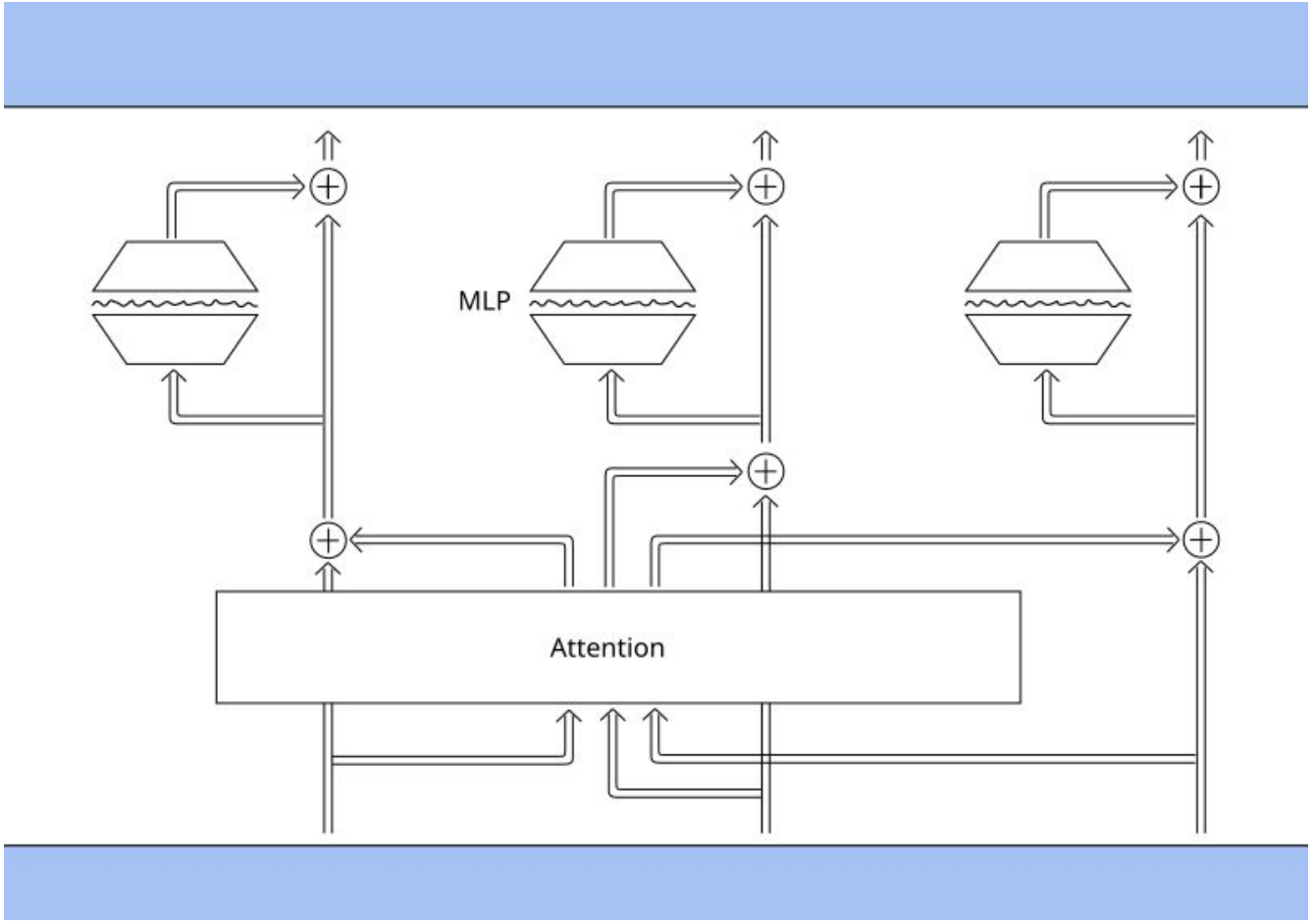
A **block** consists of **information routing**, and **information enhancing**

***Attention* layer**

Moves information from earlier streams to later streams. Consists of 96 independent *Attention Heads*.

***MLP* or *Feedforward* layer**

Every stream processed independently, in parallel. This is where the “knowledge” of the transformer is encoded.



Attention head: what information to get **from where**

KQ-circuit

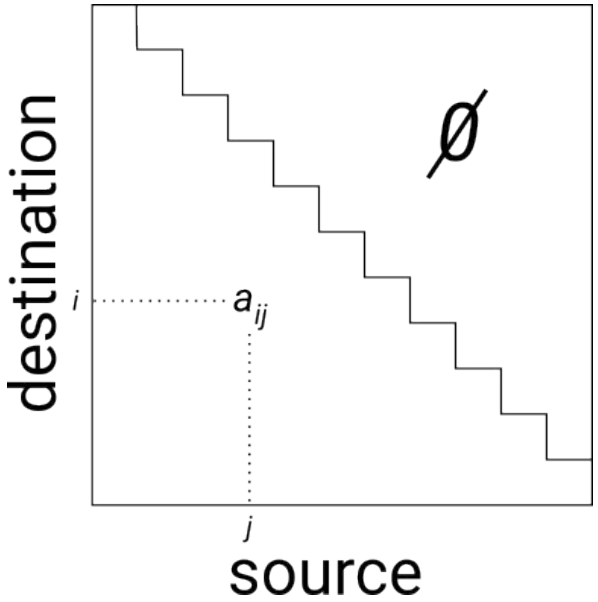
For every source–destination pair computes a *matching* (or *attention*) score; removes sources that come after the destination; *softmax*-normalizes the scores.

OV-circuit

For every source computes what info to send: a linear function of the current vector.

Head output

Weighted average of source output according to the matching scores.



$$a_{ij} = x_j \cdot KQx_i$$

$$o_j = OVx_j$$

Playing with LLMs

- `pip install llm`
  and write your own RAG as a shell script!
- Fine-tune your own model
- Explore SAEs and steer models on Neuronpedia
- Open up and look inside a model on Colab

Thank you!

Slides and notebook at https://mihaly.barasz.com/town-hall-2024/