CINECA

Introduction to Large Language Models

Focus of the Talk:

1. AI, DeepLearning, GenAI and LLMs

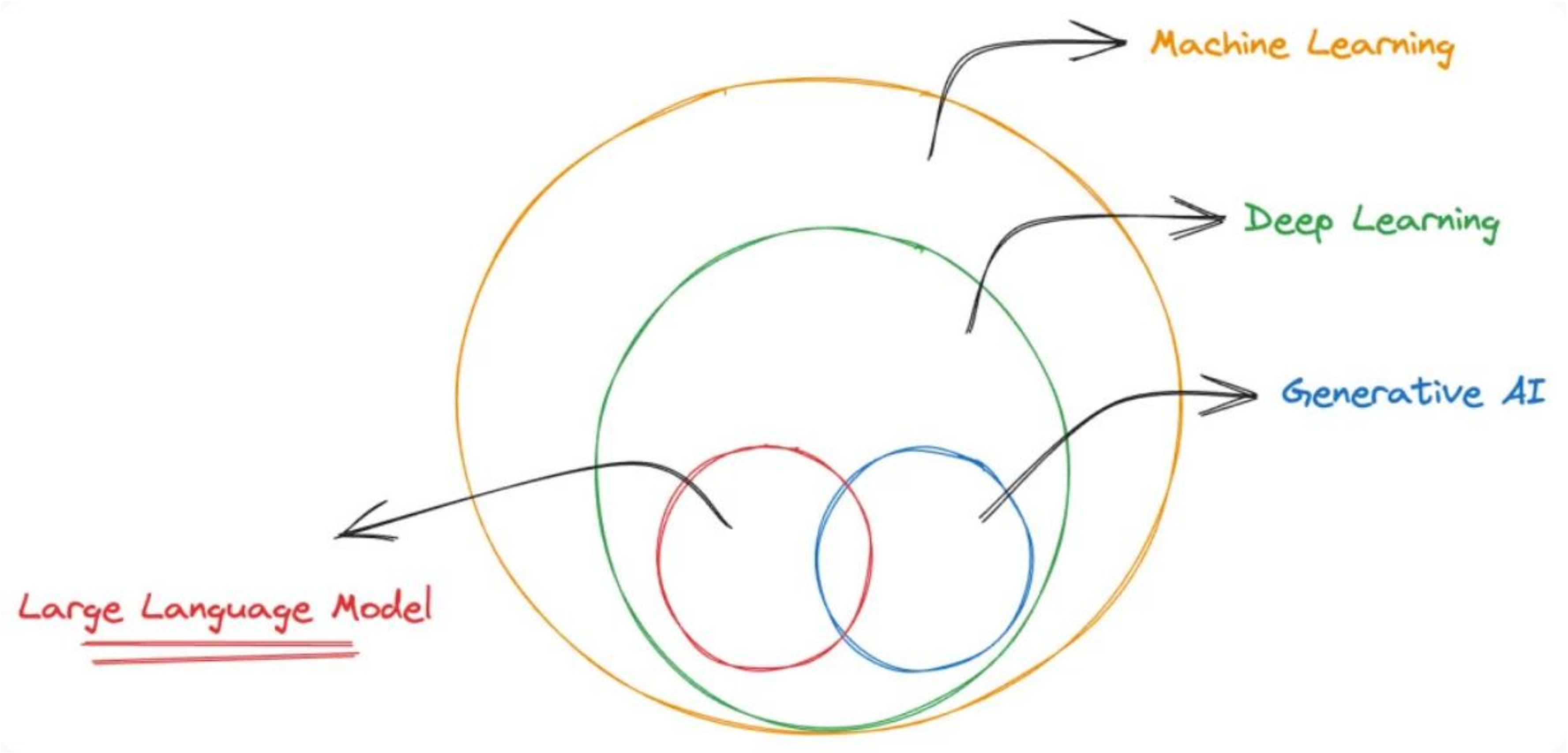
2. LLMs core concepts

    - Tokens
    - Embedding
    - Transformers
    - Attention Mechanism

3. Tailor LLMs to specific application

    - Foundation Models
    - Retrieval-Augmented Generation (RAG)
    - Fine-tuning

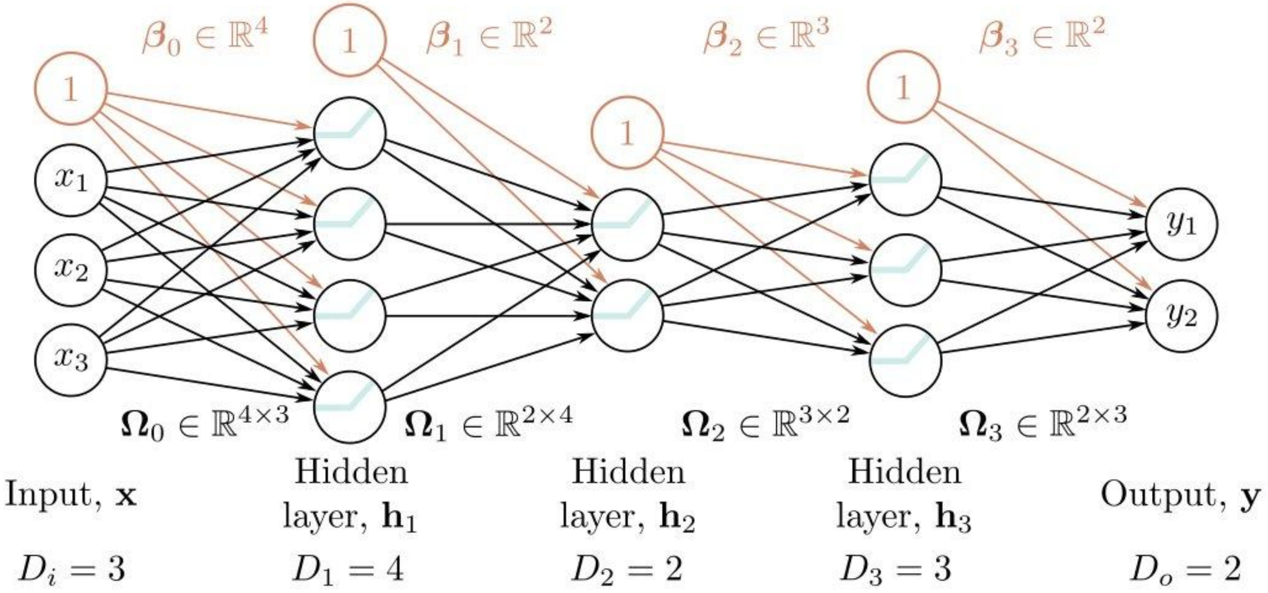
Relation between LLMs and AI



•

https://shurutech.com/innovating-with-generative-ai/

Deep Learning: Neural Networks

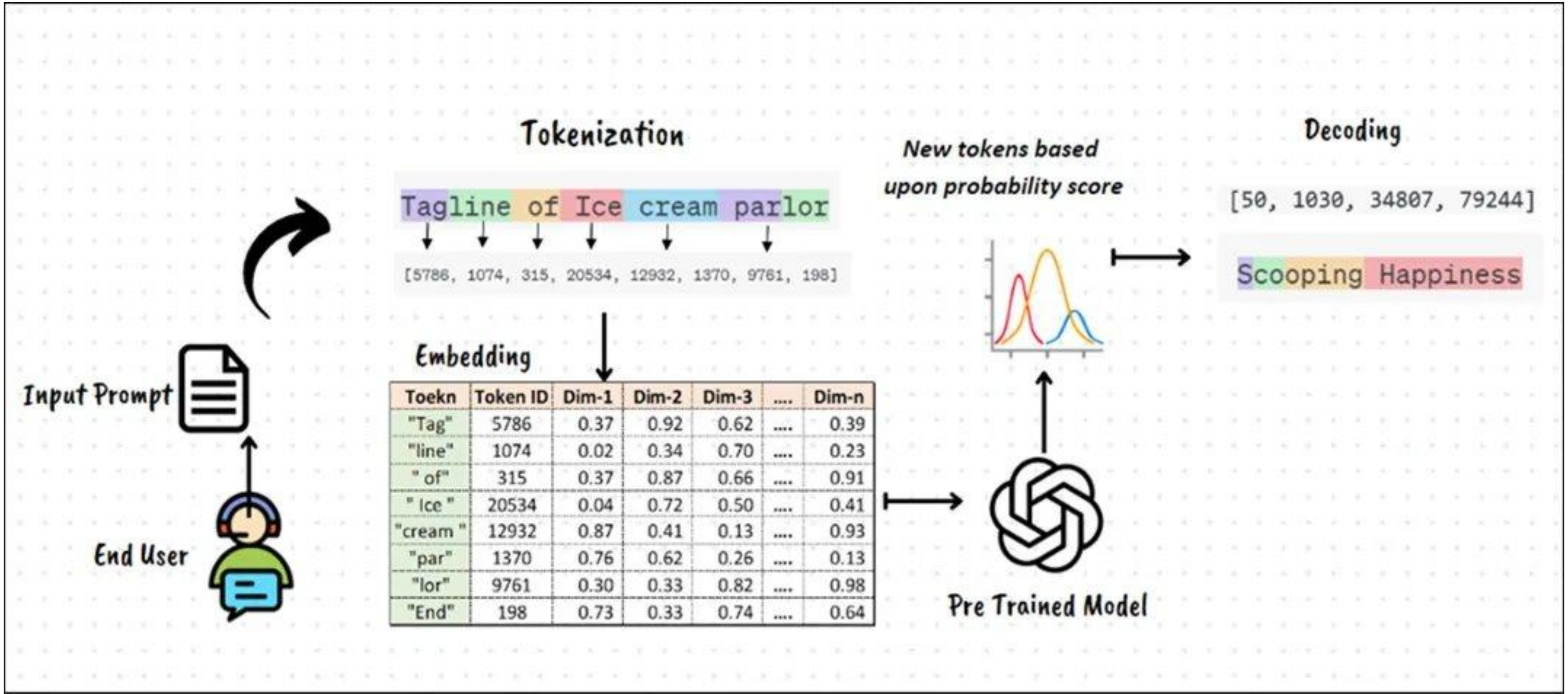


$\beta_0 \in \mathbb{R}^4$
$\beta_1 \in \mathbb{R}^2$
$\beta_2 \in \mathbb{R}^3$
$\beta_3 \in \mathbb{R}^2$
1
1
1
1
x_1
x_2
x_3
y_1
y_2
$\mathbf{\Omega}_0 \in \mathbb{R}^{4\times3}$
$\mathbf{\Omega}_1 \in \mathbb{R}^{2\times4}$
$\mathbf{\Omega}_2 \in \mathbb{R}^{3\times2}$
$\mathbf{\Omega}_3 \in \mathbb{R}^{2\times3}$
Input, $\mathbf{x}$
Hidden layer, $\mathbf{h}_1$
Hidden layer, $\mathbf{h}_2$
Hidden layer, $\mathbf{h}_3$
Output, $\mathbf{y}$
$D_i = 3$
$D_1 = 4$
$D_2 = 2$
$D_3 = 3$
$D_o = 2$

CAT

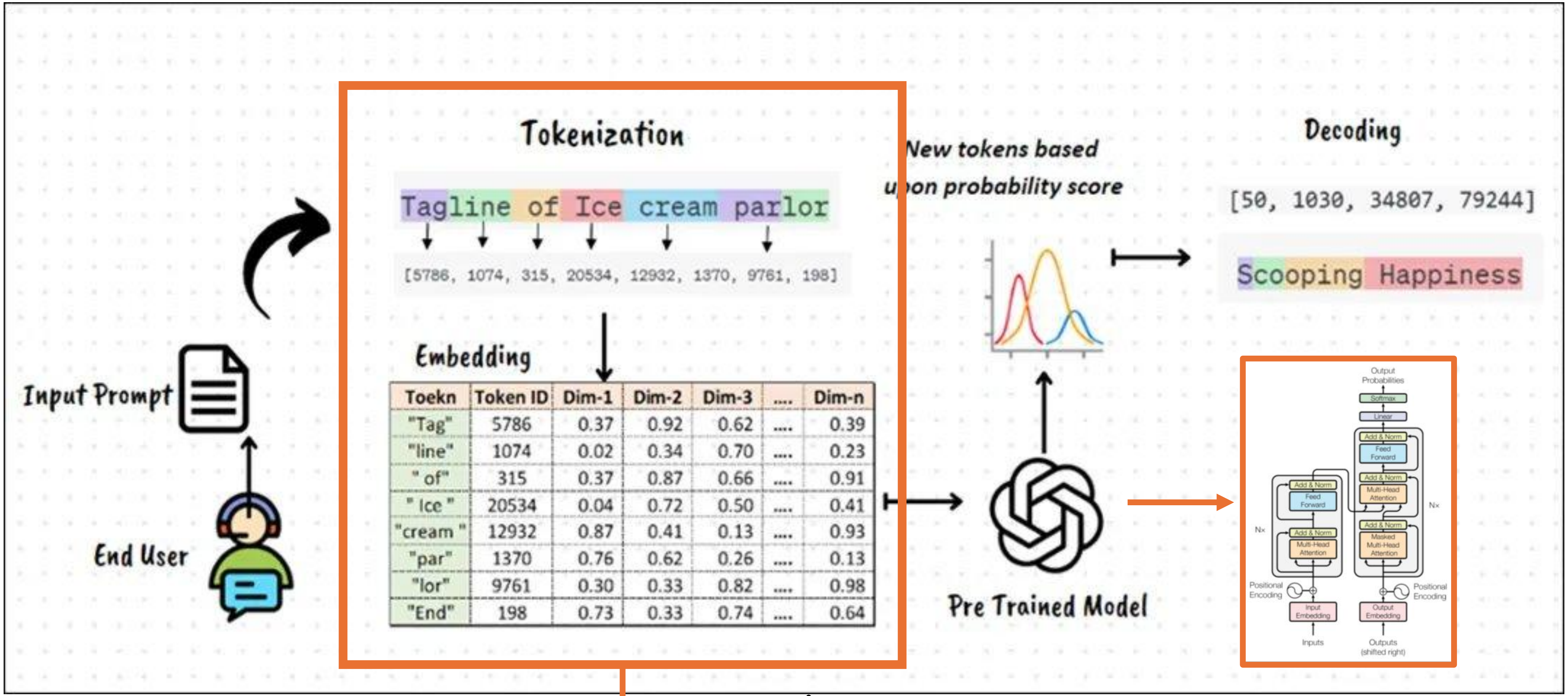
$$Y = a(\dots a(\Omega_0 x + \beta_0))$$

LARGE LANGUAGE MODELS pipeline



Toekn	Token ID	Dim-1	Dim-2	Dim-3	....	Dim-n
"Tag"	5786	0.37	0.92	0.62	....	0.39
"line"	1074	0.02	0.34	0.70	....	0.23
" of"	315	0.37	0.87	0.66	....	0.91
" Ice "	20534	0.04	0.72	0.50	....	0.41
"cream "	12932	0.87	0.41	0.13	....	0.93
"par"	1370	0.76	0.62	0.26	....	0.13
"lor"	9761	0.30	0.33	0.82	....	0.98
"End"	198	0.73	0.33	0.74	....	0.64

LARGE LANGUAGE MODELS pipeline



Toekn	Token ID	Dim-1	Dim-2	Dim-3	....	Dim-n
"Tag"	5786	0.37	0.92	0.62	....	0.39
"line"	1074	0.02	0.34	0.70	....	0.23
" of"	315	0.37	0.87	0.66	....	0.91
" Ice "	20534	0.04	0.72	0.50	....	0.41
"cream "	12932	0.87	0.41	0.13	....	0.93
"par"	1370	0.76	0.62	0.26	....	0.13
"lor"	9761	0.30	0.33	0.82	....	0.98
"End"	198	0.73	0.33	0.74	....	0.64

From text to numbers

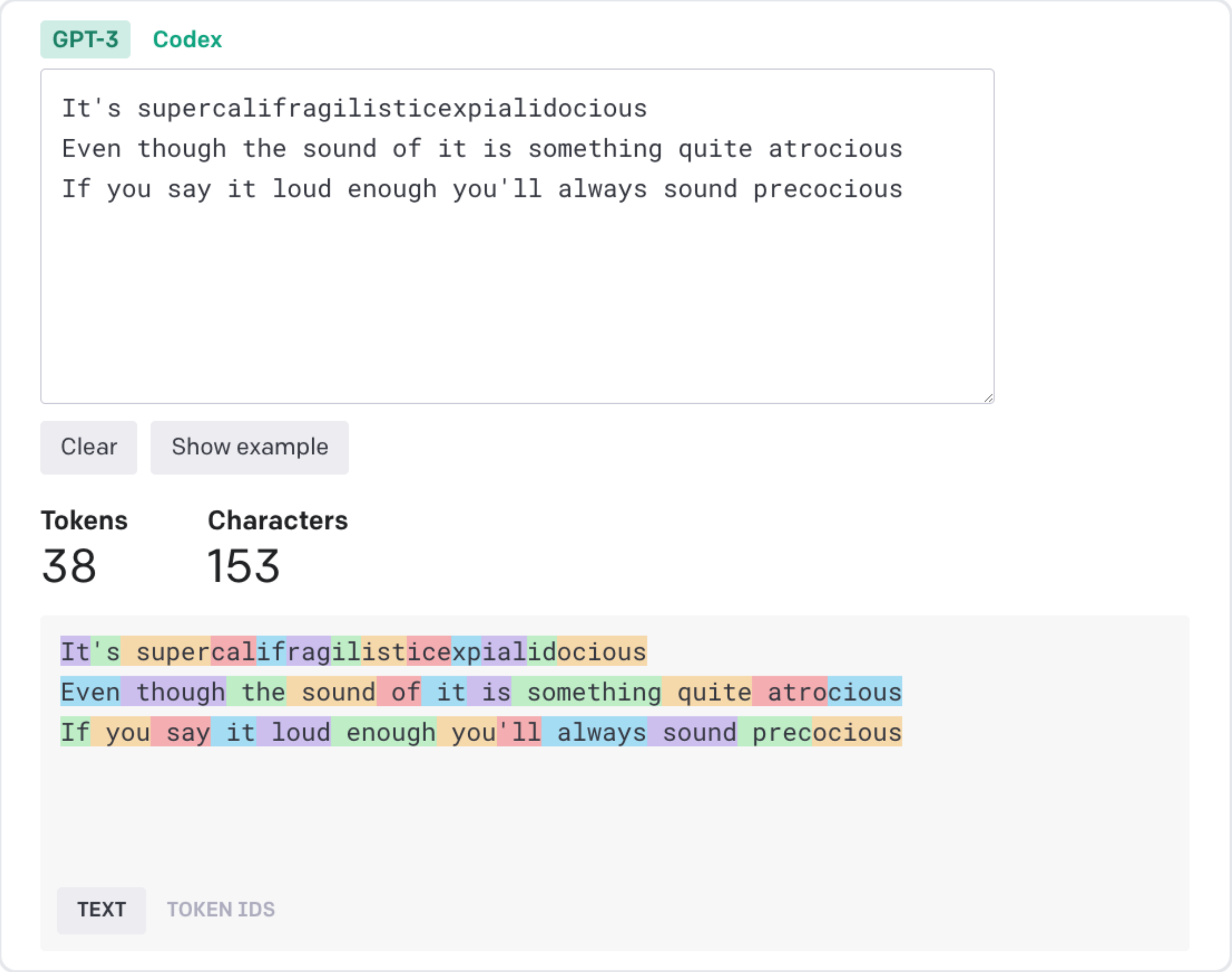
1. Tokenization

A token is a basic unit of text or code that an LLM can understand and process.

They can range from entire words down to single letters.

Nowadays, a token is a part of words.

Then, to each token in the vocabulary is assigned a token ID, a unique numerical identifier.



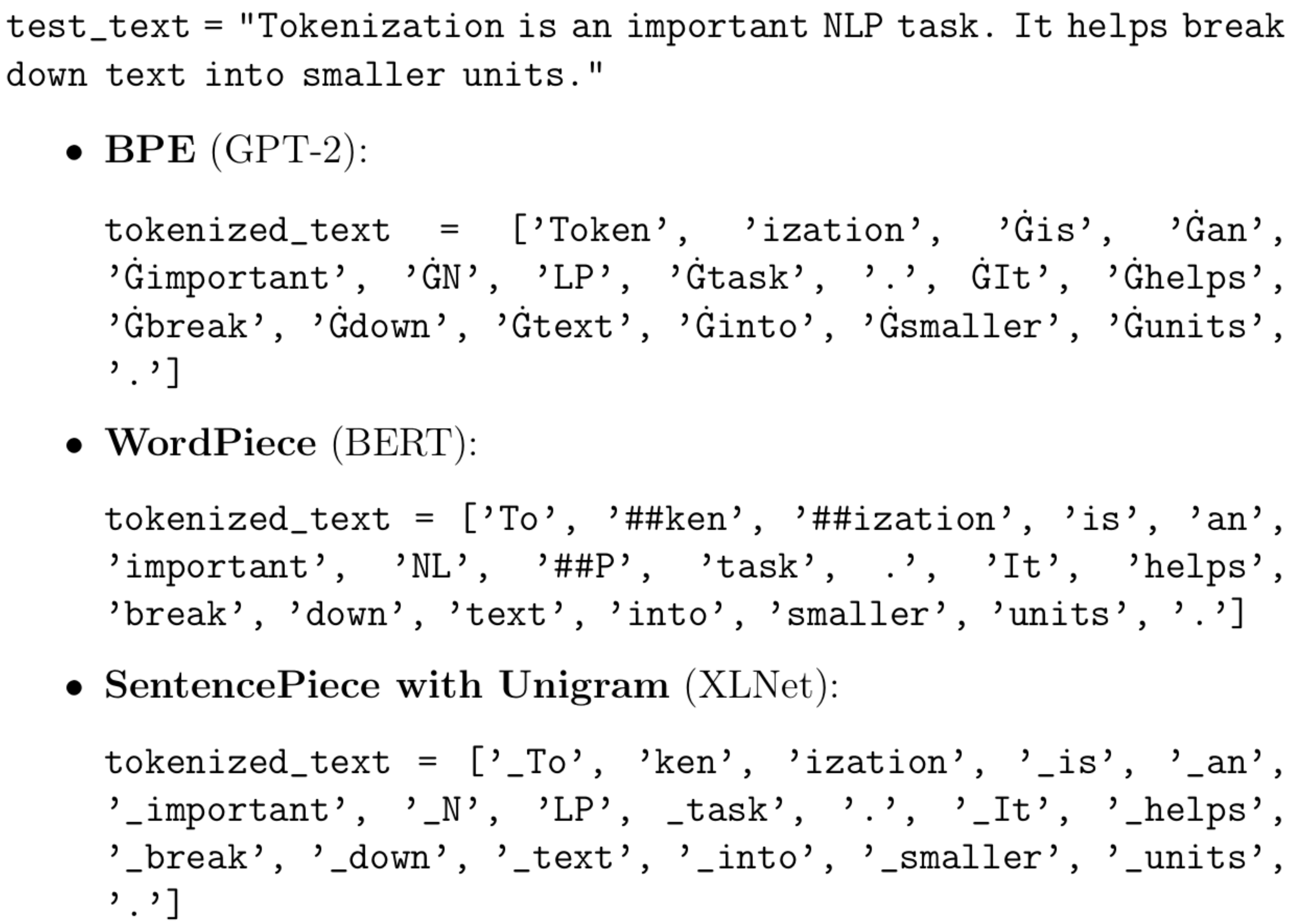
```
It's supercalifragilisticexpialidocious
Even though the sound of it is something quite atrocious
If you say it loud enough you'll always sound precocious
```

Tokens
38

Characters
153

```
It's supercalifragilisticexpialidocious
Even though the sound of it is something quite atrocious
If you say it loud enough you'll always sound precocious
```

An example of Tokenization



```
test_text = "Tokenization is an important NLP task. It helps break
down text into smaller units."
```

- **BPE** (GPT-2):

```
tokenized_text = [’Token’, ’ization’, ’Ġis’, ’Ġan’,
’Ġimportant’, ’ĠN’, ’LP’, ’Ġtask’, ’.’, ĠIt’, ’Ġhelps’,
’Ġbreak’, ’Ġdown’, ’Ġtext’, ’Ġinto’, ’Ġsmaller’, ’Ġunits’,
’.’]
```

- **WordPiece** (BERT):

```
tokenized_text = [’To’, ’##ken’, ’##ization’, ’is’, ’an’,
’important’, ’NL’, ’##P’, ’task’, .’, ’It’, ’helps’,
’break’, ’down’, ’text’, ’into’, ’smaller’, ’units’, ’.’]
```

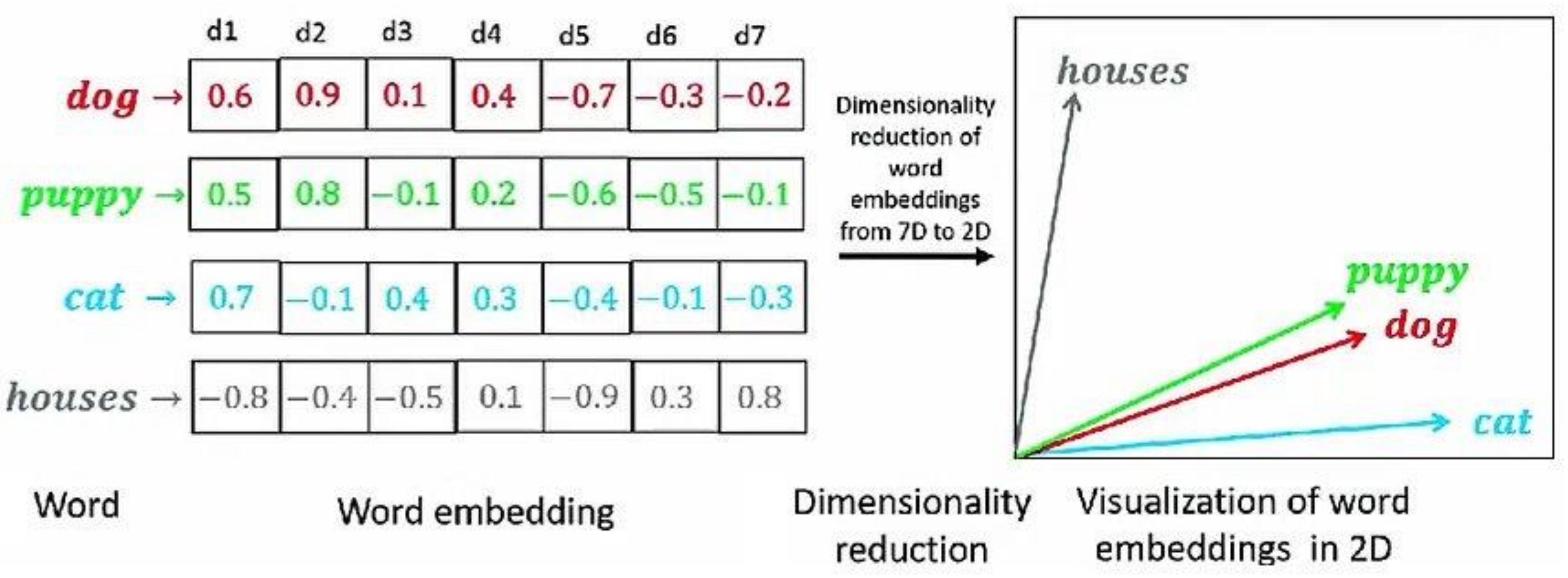
- **SentencePiece with Unigram** (XLNet):

```
tokenized_text = [’_To’, ’ken’, ’ization’, ’_is’, ’_an’,
’_important’, ’_N’, ’LP’, _task’, ’.’, ’_It’, ’_helps’,
’_break’, ’_down’, ’_text’, ’_into’, ’_smaller’, ’_units’,
’.’]
```

2. Embedding

from tokens ID to vectors

Initially (and randomly), tokens get assigned vectors in an n-dimensional space (**embeddings**).



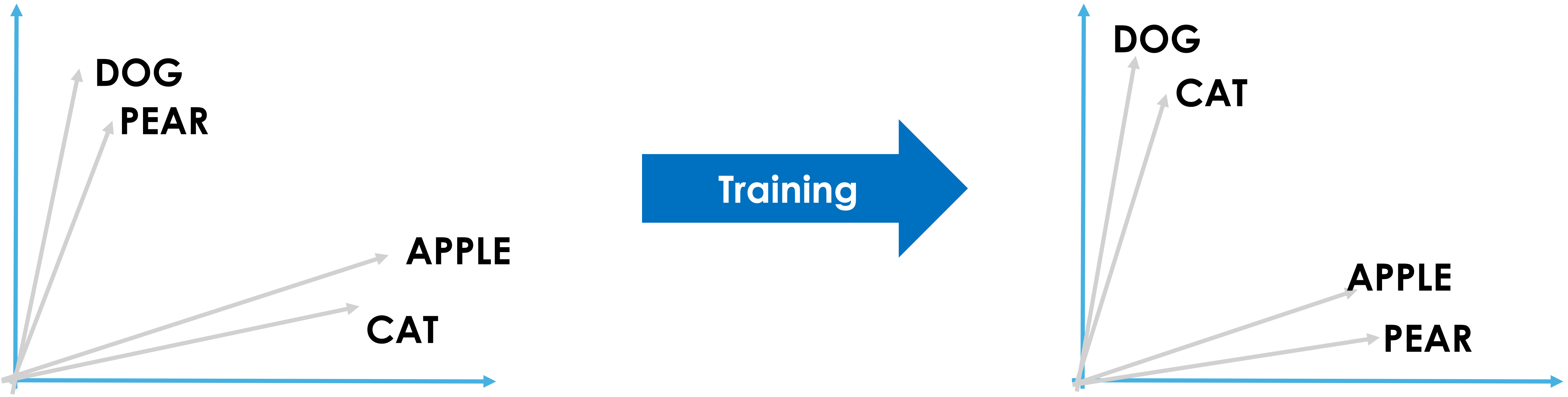
	d1	d2	d3	d4	d5	d6	d7
dog →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
puppy →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
cat →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8

Dimensionality reduction of word embeddings from 7D to 2D
houses
puppy
dog
cat
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

2. Embedding

from tokens to vectors

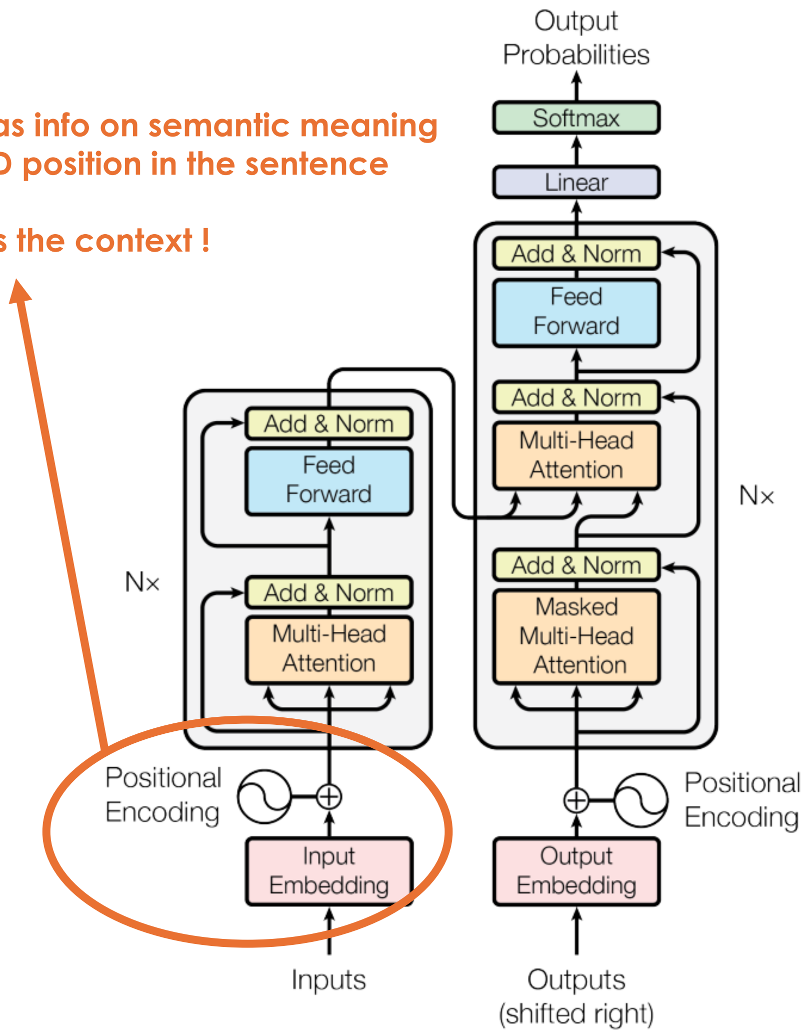
Training a model does it so that words that are semantically "close to each other tend also to have vector representation that are close in the N-dimensional space.



3. Core of LLMs : Transformers

It has info on semantic meaning AND position in the sentence

Miss the context !

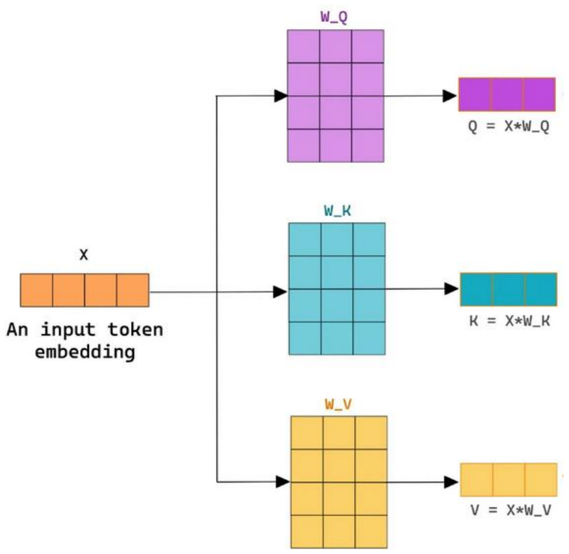


Transformers are neural networks that learn context and understanding through sequential data analysis.

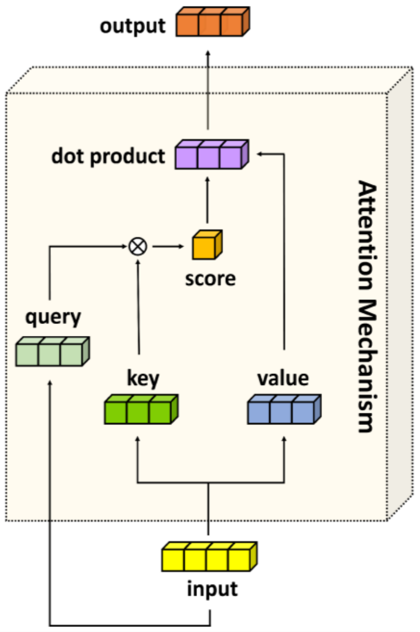
Transformer models use a technique known as **attention** or self-attention. This technique helps identify how distant data elements influence and depend on one another.

Transformers came into action in a 2017 Google paper as one of the most advanced models ever developed. This has resulted in a wave of advances called "**Transformer AI**" in machine learning.

3.1 Attention Mechanism



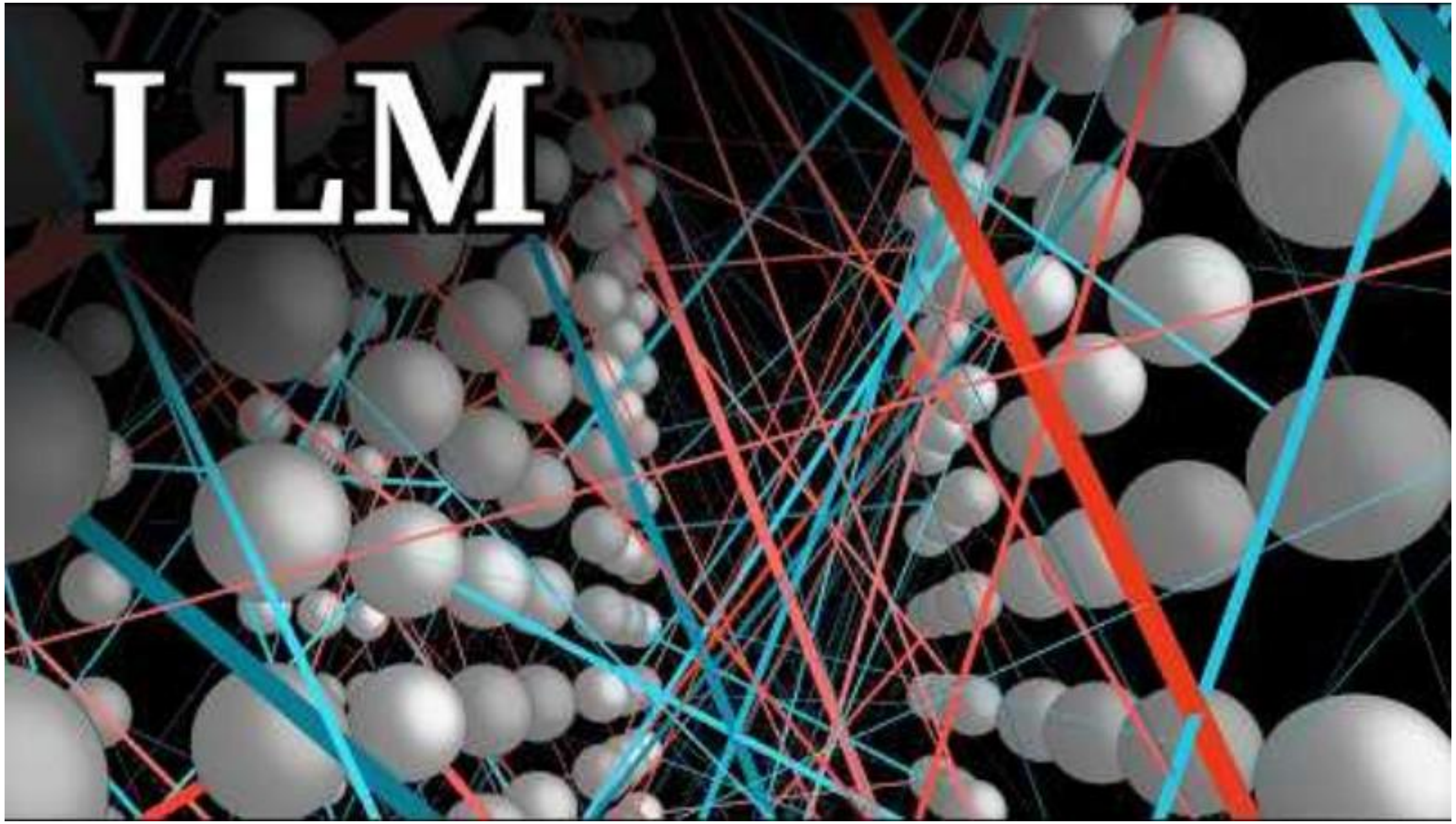
$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})\boldsymbol{V}$$



Q,K,V matrices are parameters that are learned during the training

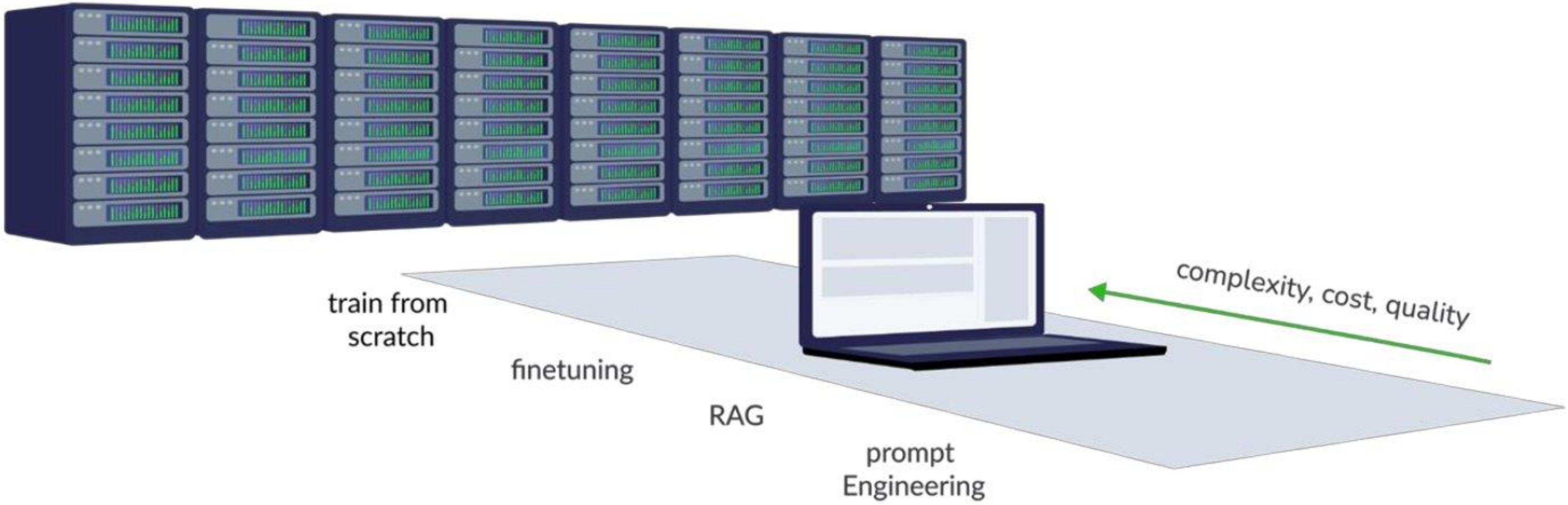
Q = represents the «query» i.e. the current token's perspective
K = represents the “key” or “label” of every past token
V = represents the “value” or “meaning” of every past token

3.1 Attention Mechanism



From 4.35 to 6.35

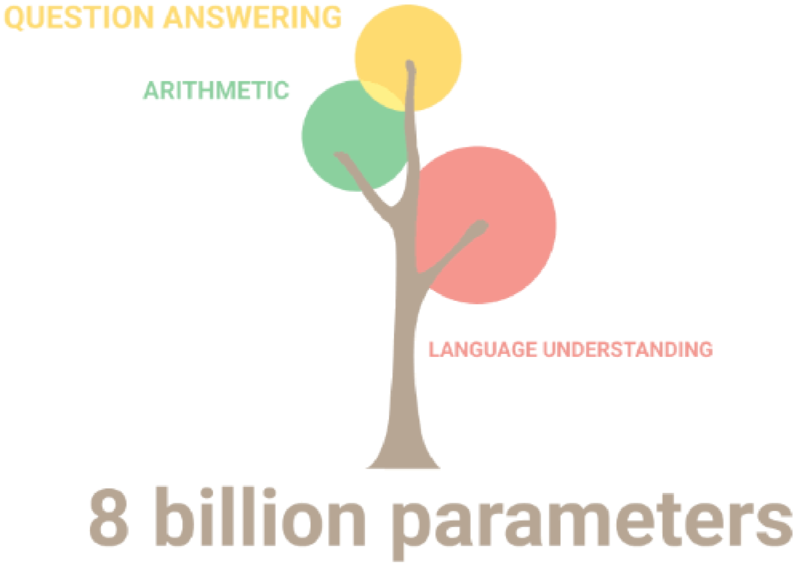
Different LLMs stages



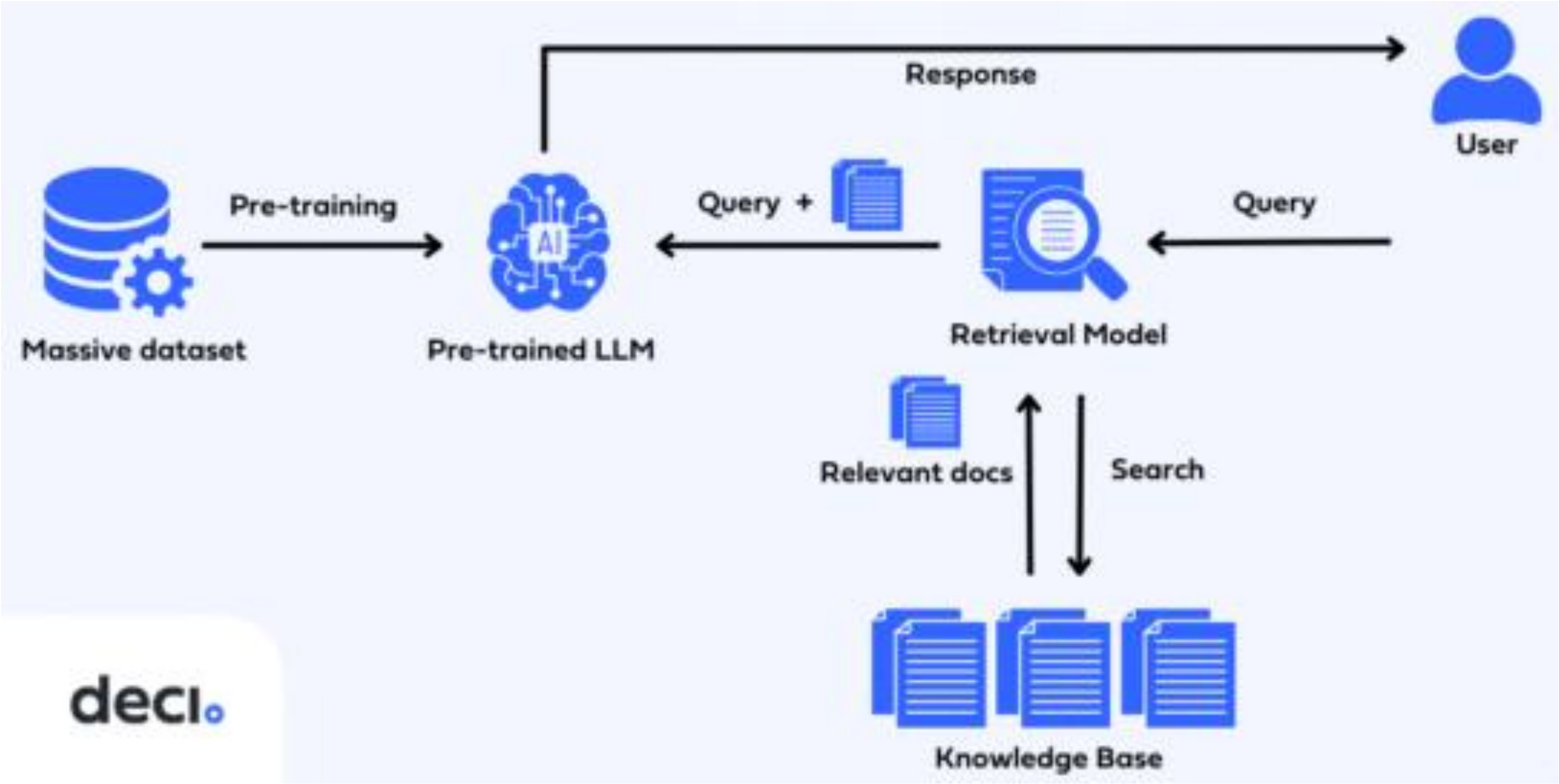
•

Foundation Models

from predicting next word to several gained task



Retrieval-Augmented Generation (RAG)

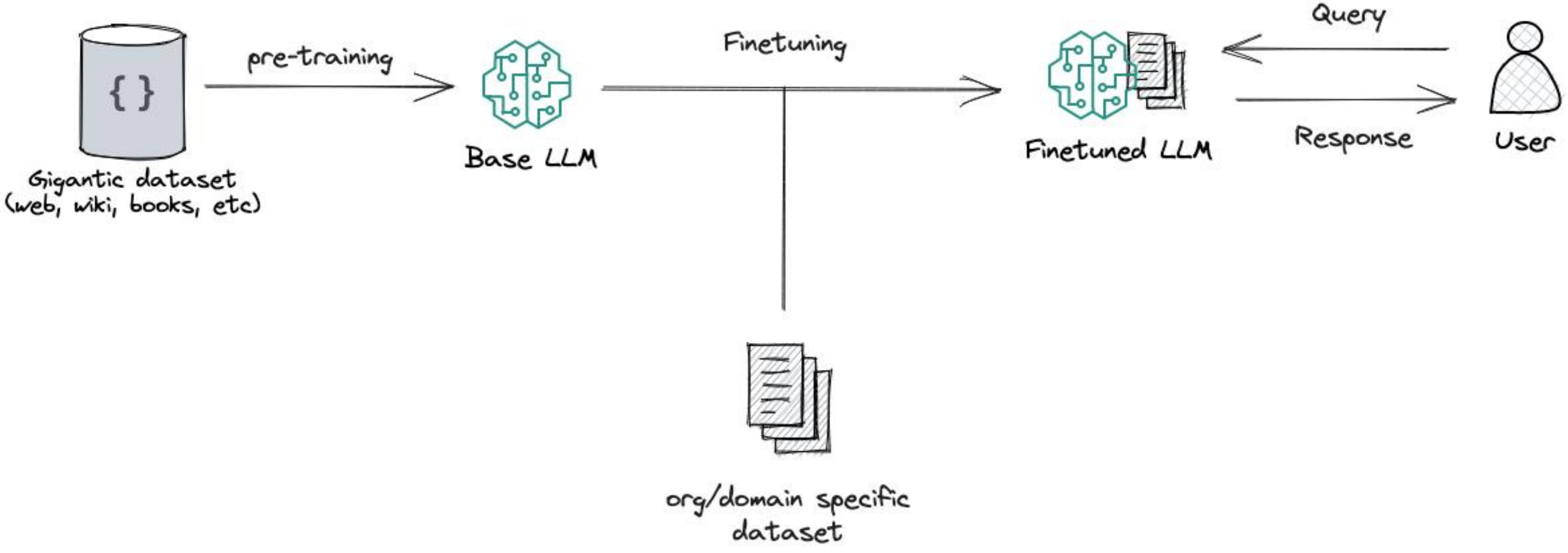


Technique for **enhancing the accuracy and reliability of LLMs with facts fetched from external sources.**

LLMs don't know, for example, company data, private PDFs from years of operations, specific knowledge about obscure topics, etc... In these cases, we can inject the prompt with context about the question asked on the specific topic.

Can be seen as an extension of prompt engineering

Fine-tuning

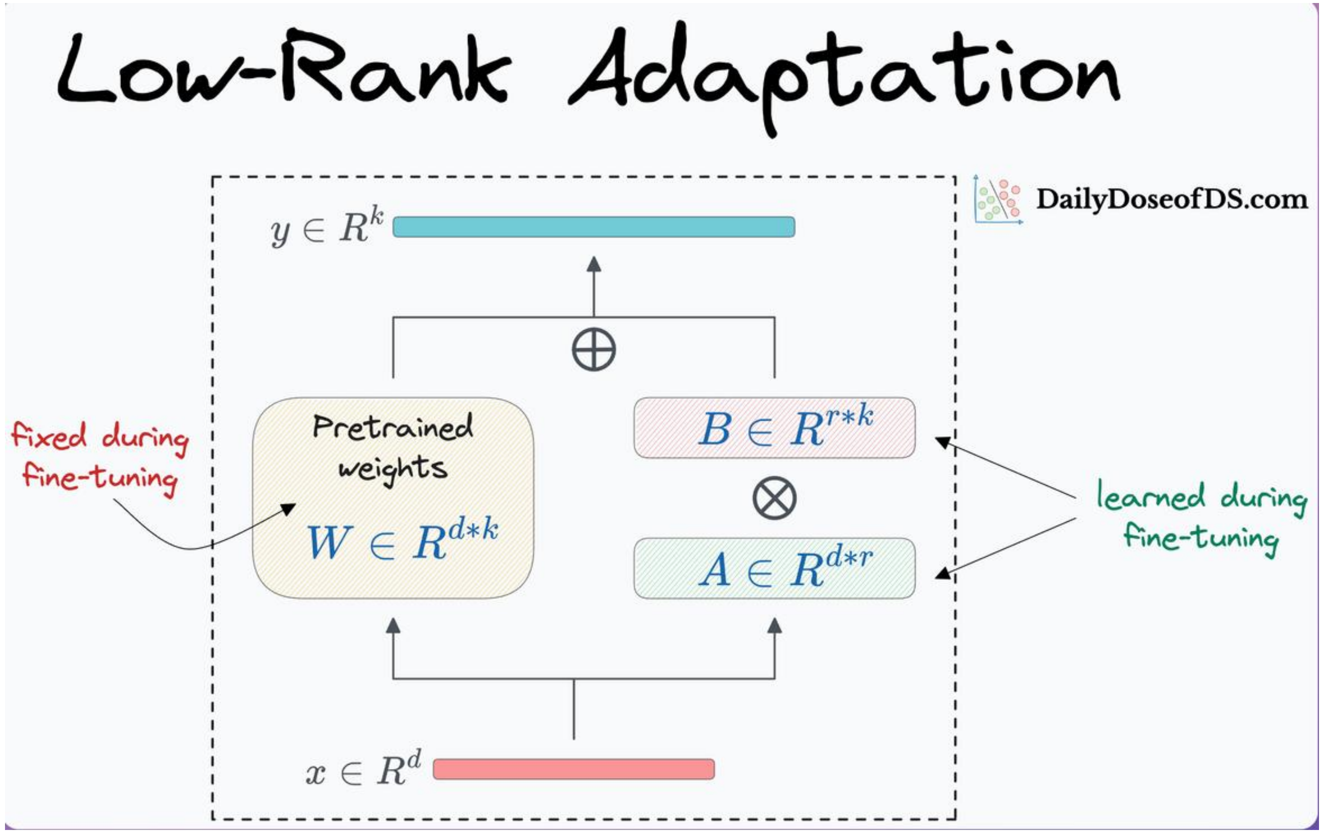


Supervised learning process of taking pre-trained models and **further training** them on smaller, specific datasets to refine their capabilities and improve performance in a particular task or domain.

Taking general-purpose models and turning them into **specialized models**.

Fine-tuning

- **Parameter efficient fine-tuning (PEFT)**:
  Fix the pretrained model , add some small layers at the "end" and train only these



Thanks!

Questions?