scai

SORBONNE CLUSTER FOR
ARTIFICIAL INTELLIGENCE

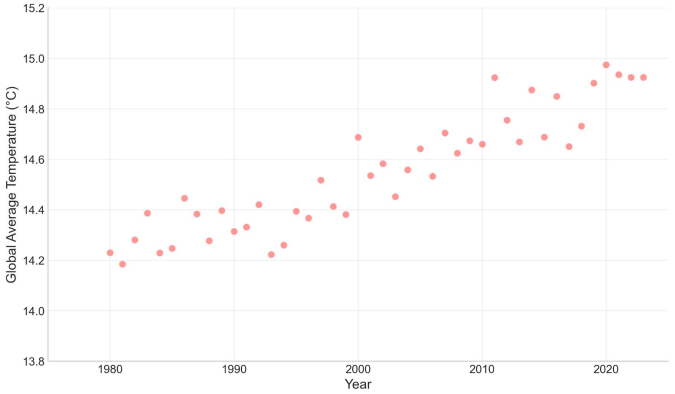
AI Introduction: From Linear Regression to Large Language Generative Models (part 2/2)

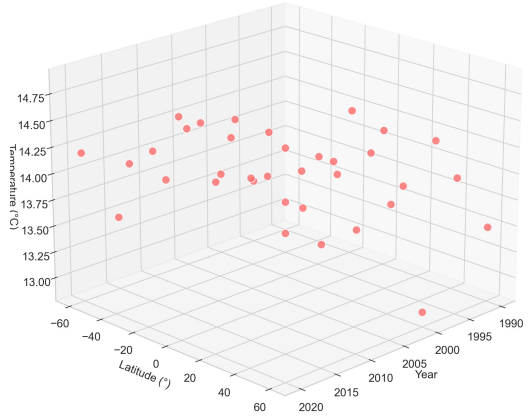
Raphael Cousin
Data Scientist
raphael.cousin@sorbonne-universite.fr

raphaelcousin.com

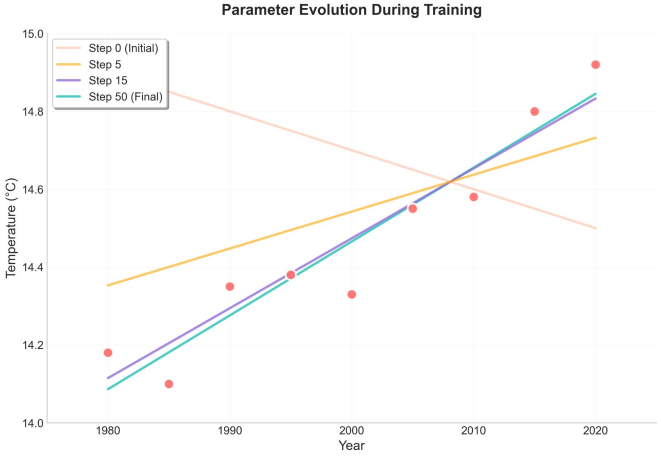
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Warm-up





1. **What are the two dimensions that characterize data in machine learning?**



2. **How are a model's parameter values initialized?**

3. **How do we train the model's parameters?**

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Plan

Natural Language Data:
- Sequential Nature
- Diversity
- Beyond Natural Language
- History

Duration : ~5 min

Represent text as numerical values:
- Tokenization
- Embedding

Duration : ~20 min

Models Essential:
- MLP
- Recurrent
- Attention

Duration : ~20 min

NLP application:
- Classification
- Translation
- Generative

Duration : ~5 min

LLM:
- Data
- Time
- Ressources

Duration : ~5 min

Foundation Model:
- Transfer Learning
- Fine Tuning
- Embeddings

Duration : ~5 min

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

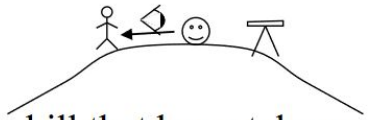
Natural Language Data

Sequential Nature of Text

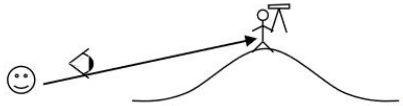
Word order impacts meaning: "Dog bites man" ≠ "Man bites dog"

Dependencies can span across **long distances in a sequence**

Context is critical for disambiguation (e.g., "bank" can mean financial institution or river edge)

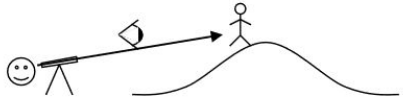


"I was on the hill that has a telescope when I saw a man."

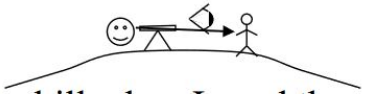


"I saw a man who was on a hill and who had a telescope."

"I saw a man who was on the hill that has a telescope on it."



"Using a telescope, I saw a man who was on a hill."



"I was on the hill when I used the telescope to see a man."

. . .

I saw the man on the hill with the telescope

Language Diversity at Scale

Languages and Writing Systems

~7,000 living languages worldwide (Ethnologue 2023)

~300 writing systems across history, with ~150 currently in use

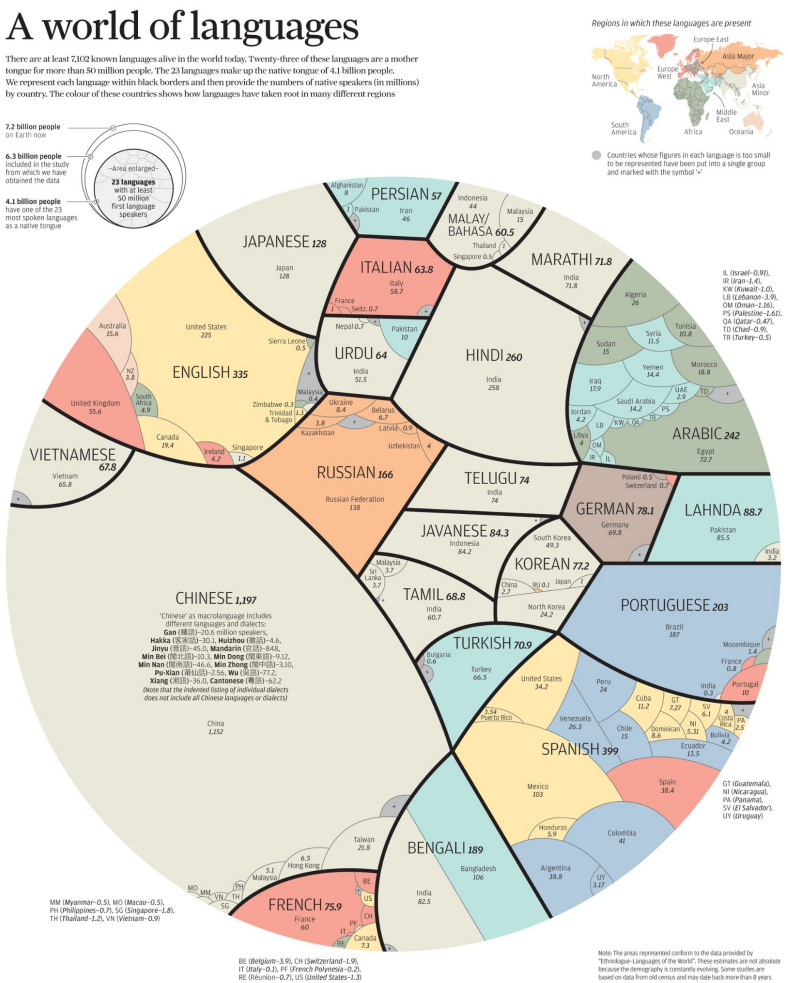
Vocabulary and Text Volume

170,000+ words in current English usage (Oxford English Dictionary)

~130 million books published in all languages throughout history

2.5 million+ books published annually worldwide

Billions of web pages containing trillions of words across languages



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Sequence Representations Beyond Natural Language

Biological Sequences: DNA sequences use four nucleotide bases: A (Adenine), T (Thymine), G (Guanine), C (Cytosine).

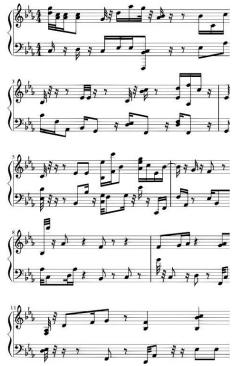
$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$

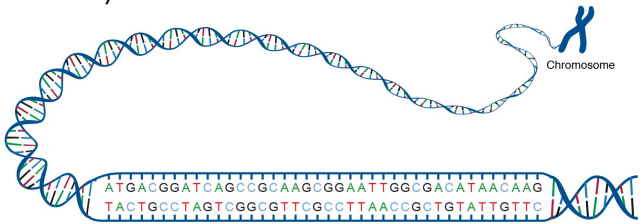
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Mathematical: Mathematical notation forms a symbolic language with operators (+, -, ×, ÷, =), variables (x, y, z), functions (sin, cos, log), Greek letters (α, β, γ), and special symbols (∫, ∑, ∂)

Programming Code: Programming languages like Python, Java, and C++ have finite token vocabularies including keywords (if, while, return), operators (=, +, ==), and punctuation (braces, parentheses, semicolons).



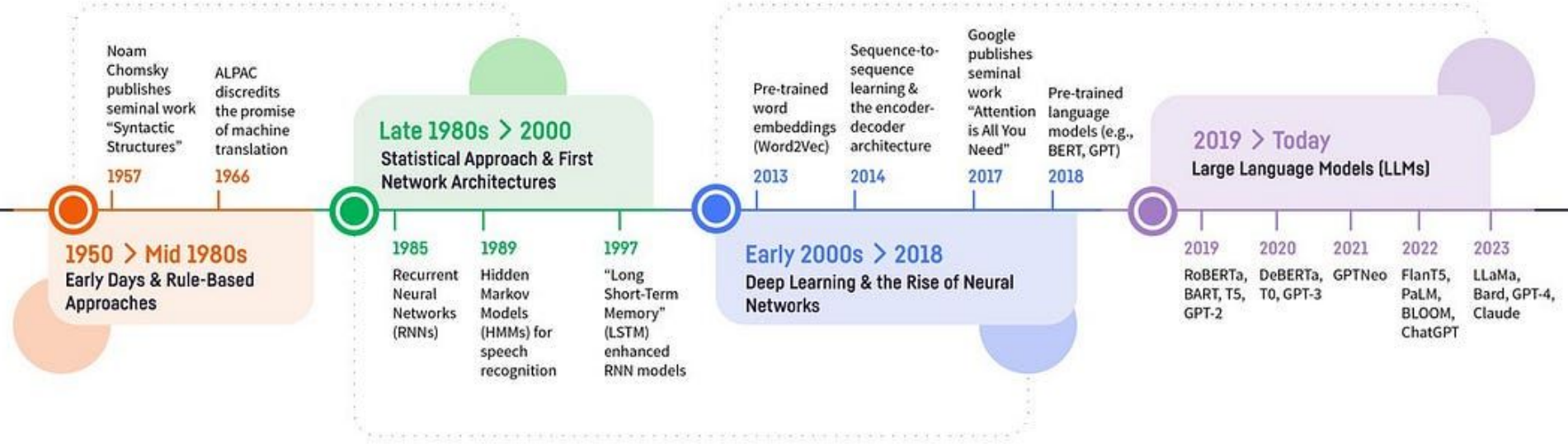
Musical Sequences: MIDI note numbers (0-127), pitch classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B), duration values (whole, half, quarter notes)



Chemical Representations: SMILES (Simplified Molecular Input Line Entry System) notation represents molecular structures as text strings using characters for atoms (C, N, O, S), bonds (-, =, #), and branches (parentheses)

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

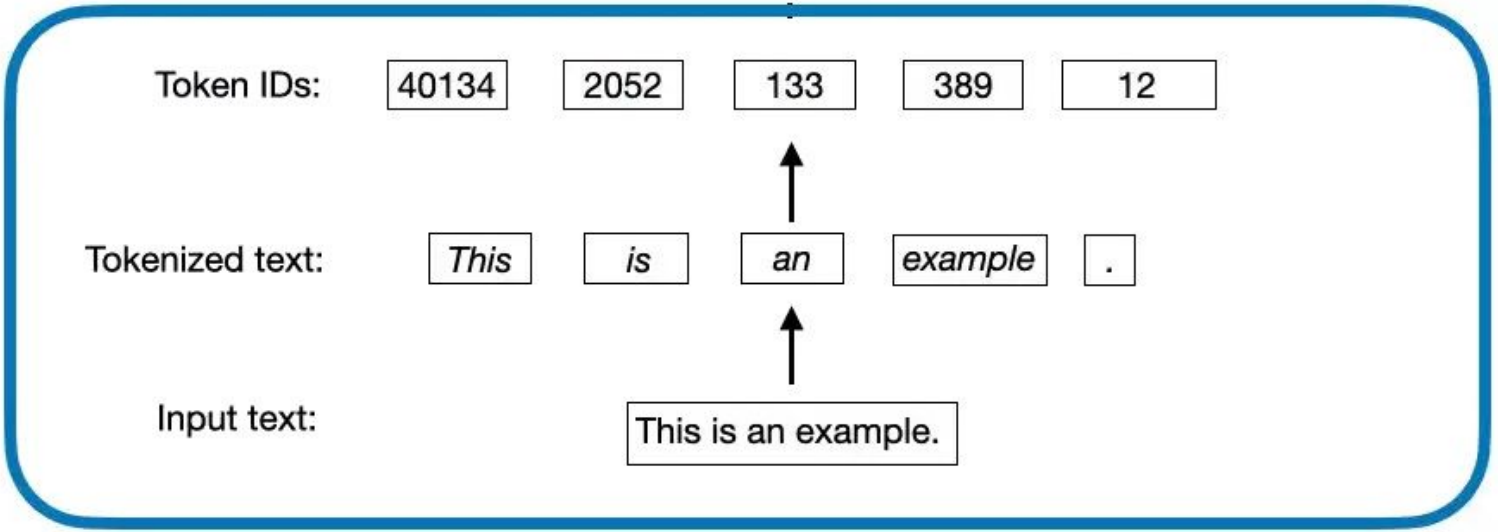
Natural Language processing - History



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Tokenization - Represent text as numerical values

Tokenization



Tokenization = The process of breaking text into smaller units (tokens) that serve as the basic elements for numerical representation.

Vocabulary **=** A fixed collection of all possible tokens that the tokenizer recognizes.

Character-Level Tokenization

Character-Level Tokenization

Character tokenization breaks text into individual characters, offering a very small vocabulary but requiring longer sequences.

Vocabulary: Alphabet + special characters (~30 tokens)

Example :

Token	ID	Token	ID
"a"	1	"z"	26
"b"	2	" "	27
"c"	3	"."	28

vocabulary

"The cats are running" → [T,h,e, ,c,a,t,s, ,a,r,e, ,r,u,n,n,i,n,g] → [20,8,5,27,3,1,20,19,27,1,18,5,27,18,21,14,14,9,14,7]

❌ **Problems:** Very long sequences + loss of word structure

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Word-Level Tokenization

Word-Level Tokenization
Word tokenization splits text at word boundaries, typically using spaces and punctuation as delimiters.

Vocabulary: All the words in the corpus (example: English Wikipedia ~13M tokens)

Example :

Token	ID
"The"	1
"cats"	856,432
"running"	2,341,567

vocabulary

"The cats are running" → [The, cats, are, running] → [1, 856432, 15, 2347]

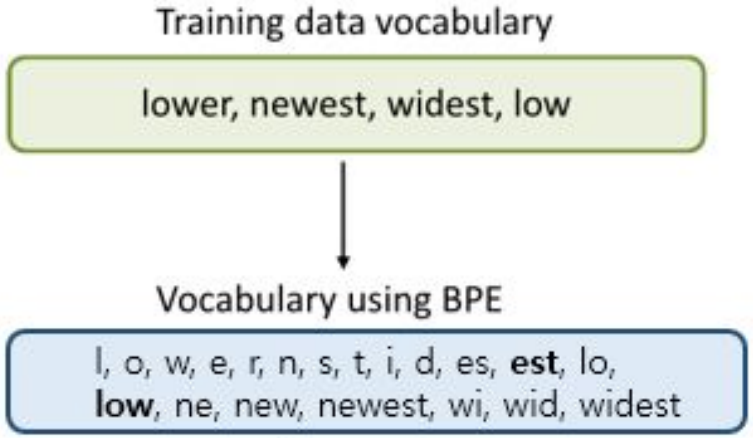
❌ **Problems:** Gigantic vocabulary + rare words underrepresented + "eat" ≠ "eats"

SubWord - Level Tokenization

Byte-Pair Encoding

BPE iteratively merges the most frequent adjacent byte pairs or character pairs to form new subword tokens.

1. Start with **character level vocabulary**
2. Identify the **most frequent pair** of adjacent tokens
3. **Add it to the vocabulary**
4. Iterate and stop at ~10k-100k tokens



Example :

Token	ID	Fréquence
"a"	1	
"s"	19	
"are"	38	100M
"ing"	40	80M
"The"	37	50M
"cat"	901	2M
"runn"	1517	500k

vocabulary

"The cats are running" → [The, cat, s, are, runn, ing] → [37, 901, 19, 38, 1517, 40]

✅ **Advantages:** Manageable vocabulary + preserved structure + semantic proximity

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

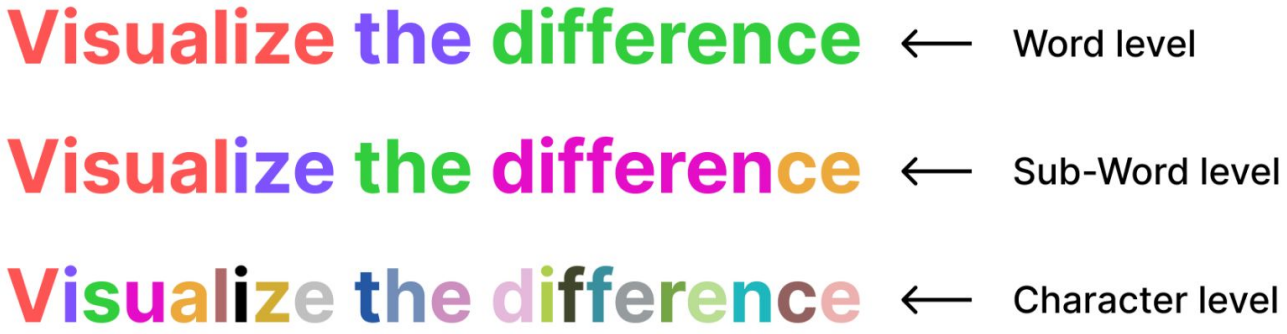
Different tokenization approaches

Special Tokens

Most tokenizers include special tokens in their vocabulary for specific purposes:

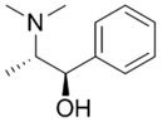
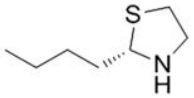
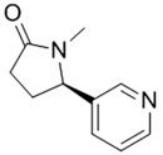
- **<PAD>**: Padding token to make sequences uniform length
- **<UNK>**: Unknown token for out-of-vocabulary words
- **<CLS>**: Beginning of sequence markers

Different tokenization approaches balance trade-offs between vocabulary size, semantic granularity, and handling of unseen words



Text Data - Extended Sequence Application

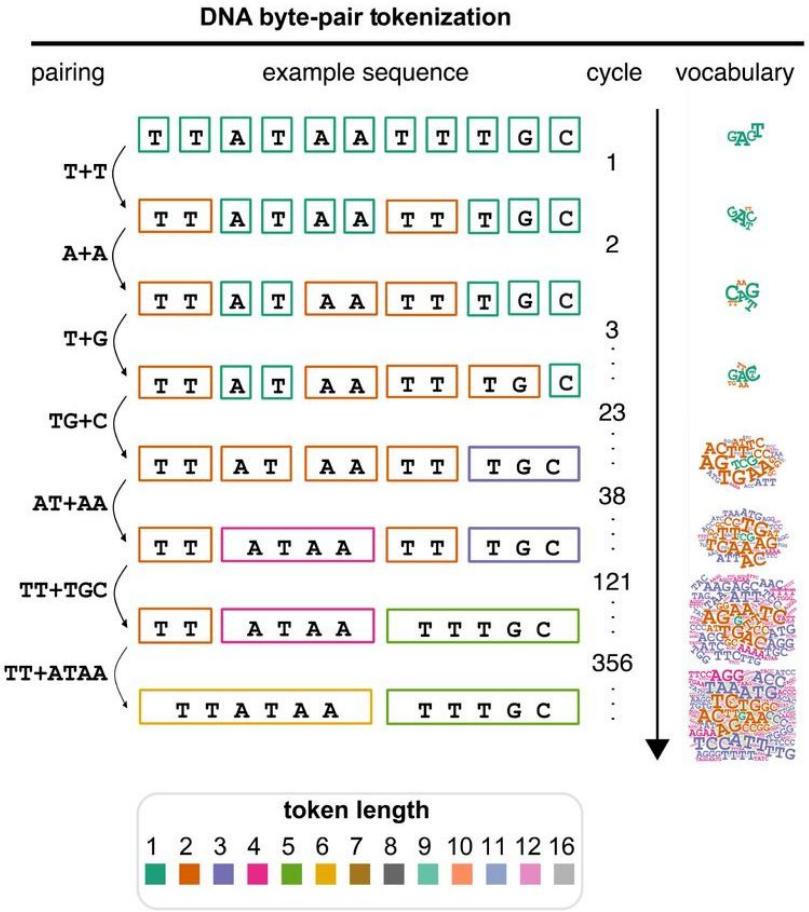
Biogenic Library

Structure	SMILES Notation
	C**[C@@H]**(**[C@H]**(O)c1ccccc1)N(C)C
	CCCC**[C@H]**1NCCS1
⋮	⋮
	CN1C(=O)CC**[C@@H]**1c1cccnc1

Tokenize

Vocabulary

Index	Token
1	C
2	[C@H]
3	[C@@H]
4	O
⋮	⋮
85	N
86	S
87	=



From Tokens to embeddings

Tokenizers Example

The tokenization of uncommon words like 'anticonstitutionnellement' and émojis 🤖 varies across tokenizers!

GPT2 Vocabulary size: **50,257**

Token count: **28**

Tokens:

THE TOKEN IZATION OF UNCOMMON WORDS LIKE ' ANT ICON ST ITUTION NEL LEMENT '

AND É MO J IS 🤖 VARIES ACROSS TOKEN IZERS !

Token IDs:

464 11241 1634 286 19185 2456 588 705 415 4749 301 2738 4954 1732 6 290

38251 5908 73 271 12520 97 244 17806 1973 11241 11341 0

GPT5 Vocabulary size: **200,006**

Token count: **25**

Tokens:

THE TOKEN IZATION OF UNCOMMON WORDS LIKE ' ANT ICON STITUTION NEL LEMENT '

AND É MO JIS 🤖 VARIES ACROSS TOKEN IZERS !

Token IDs:

976 6602 2860 328 69794 6391 1299 461 493 3679 20066 10085 1254 6 326 1212

3690 137286 93643 244 47245 5251 6602 24223 0

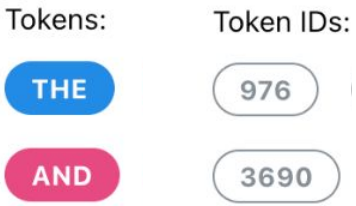
Breaking the token order

One hot encoding

Transforms a categorical variable into a binary vector.

One-hot vectors are sparse

No semantic similarity ("cat" and "dog" are equally distant from "car")



Is AND **Bigger than** THE **?**

NO!

Token IDs are arbitrary assignments, not meaningful rankings

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch

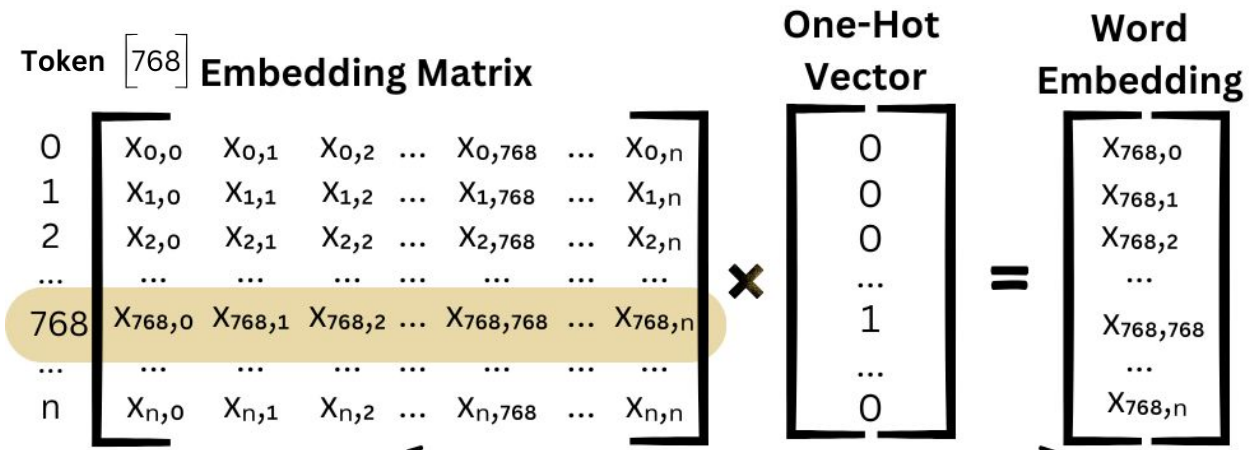
	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

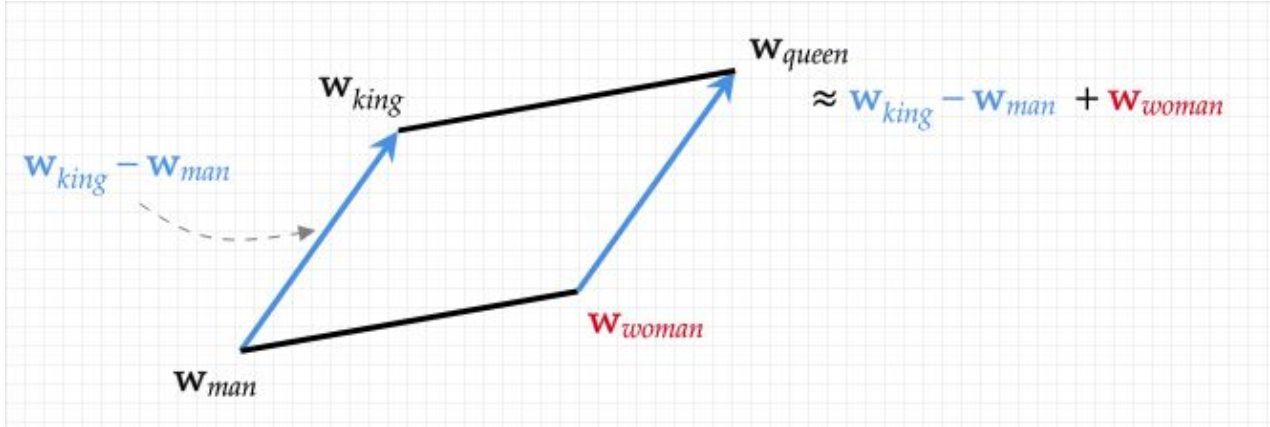
Each word gets a 1x9 vector representation

Embeddings

Embedding:
it is a learned dense vector representation that maps discrete objects into a continuous low-dimensional space where semantic similarity is captured by geometric proximity

Embeddings learn meaningful relationships during training





Models Essential

MLP Fails with Token Sequences

Fixed input size:

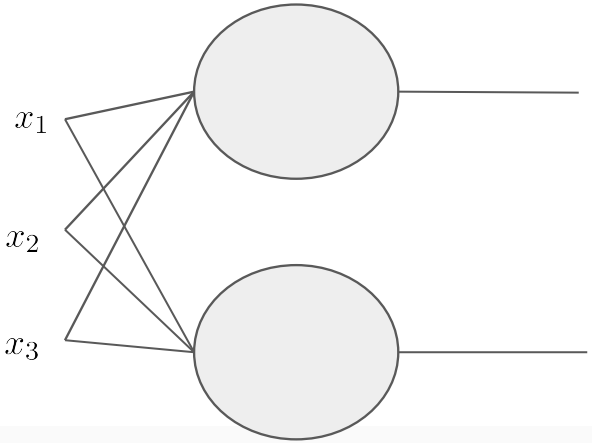
Can't handle variable-length sequences

No positional/context awareness:

Treat all inputs as independent features

Parameter explosion:

Long sequences -> massive parameter counts



Same word, different position

sample 1	the	**cat**	is	in	the	kitchen
	0		2	3	4	5
sample 2	and	the	**cat**	is	in	the	kitchen
	0	1		3	4	5	6

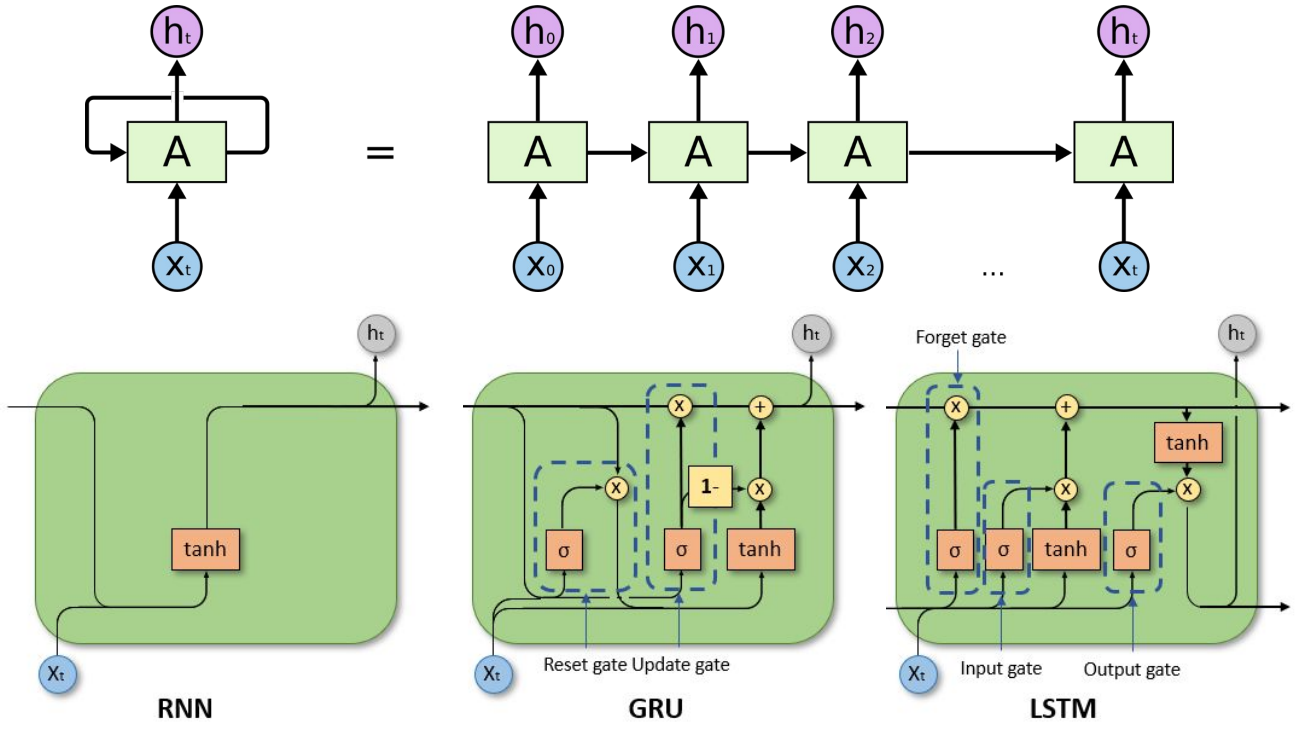
"cat"

at position 1 vs position 2 → different weights → different computation

Recurrent Layers

A layer where the **output** at each step is **fed back as input** to the next step, allowing information to flow across a sequence.

In practice: slow to train (as they are sequential) and difficulties with very long sequences

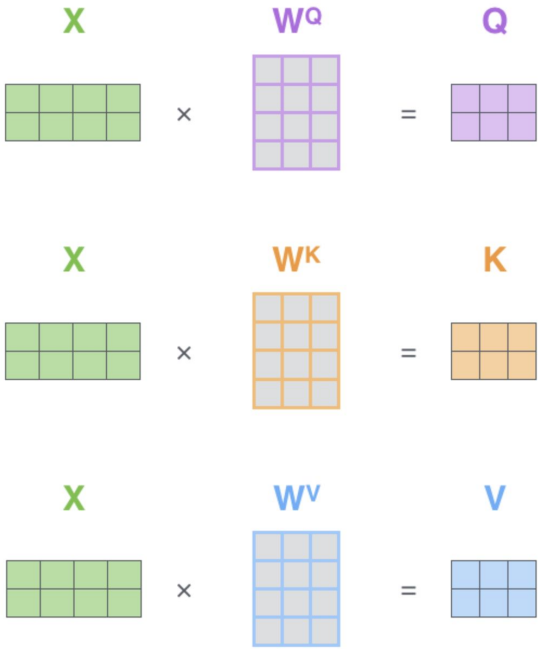


Sequence to Sequence Learning with Neural Networks

Ilya Sutskever, Oriol Vinyals, Quoc V. Le **2014**

Attention Layer

n=2, d=4

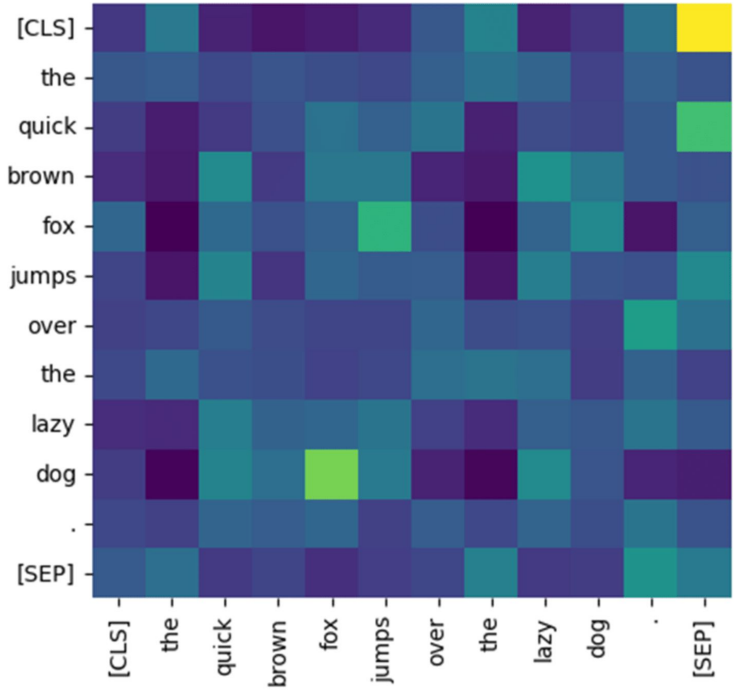


$$y = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

QK measures **how much token i wants to attend to token j**.

Softmax to sum to 1 per row

V represents the **content/information** that token j carries.



Output dimension = Input dimension

$$X_1 \in \mathbb{R}^{n_1 \times d} \xrightarrow{Q_1 K_1^T} \mathbb{R}^{n_1 \times n_1} \xrightarrow{\times V_1} \mathbb{R}^{n_1 \times d}$$

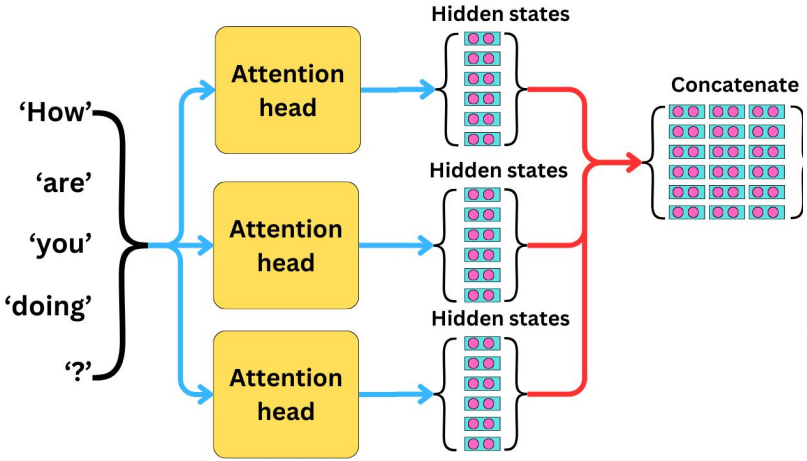
$$X_2 \in \mathbb{R}^{n_2 \times d} \xrightarrow{Q_2 K_2^T} \mathbb{R}^{n_2 \times n_2} \xrightarrow{\times V_2} \mathbb{R}^{n_2 \times d}$$

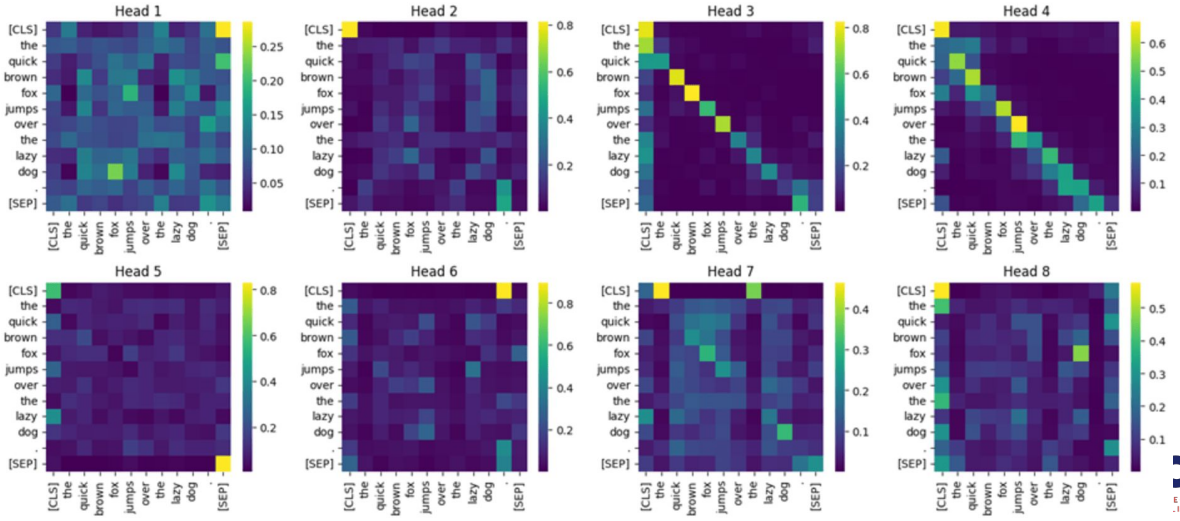
scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Multi Head Attention

Multi-head attention runs several attention (each with its own learned weights)

Allowing the model to attend to different aspects of the input.





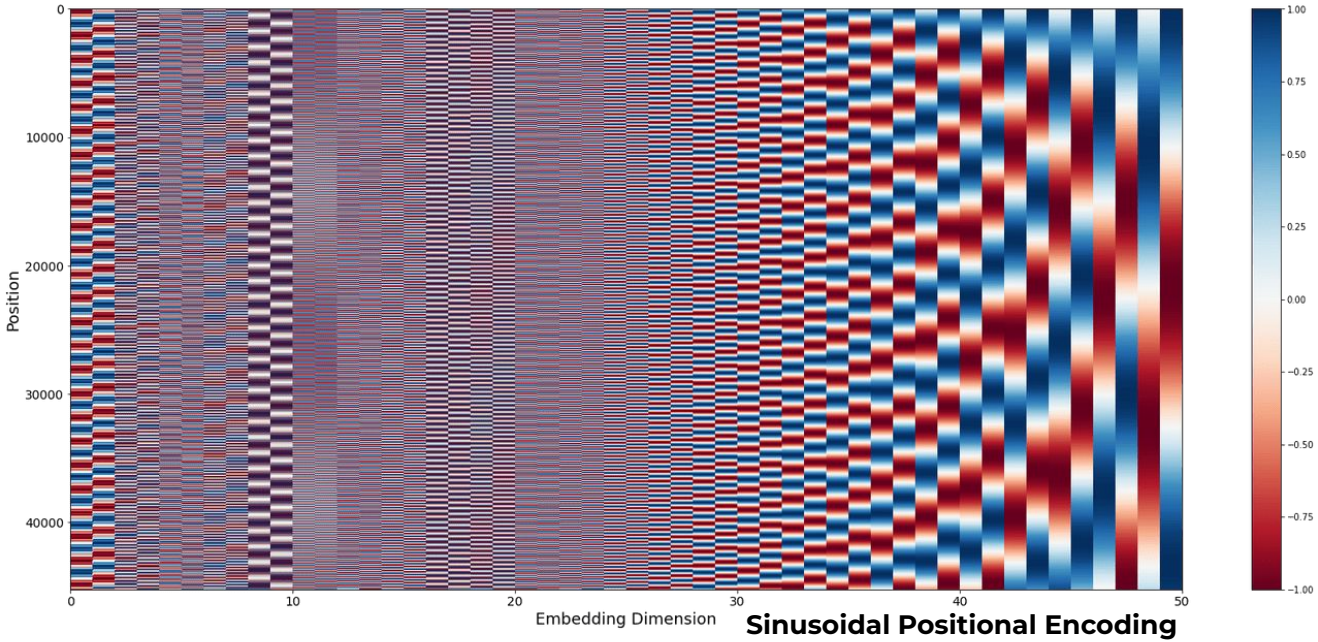
Positional Information in Attention

Attention is permutation-invariant: it treats input as a set, not a sequence.

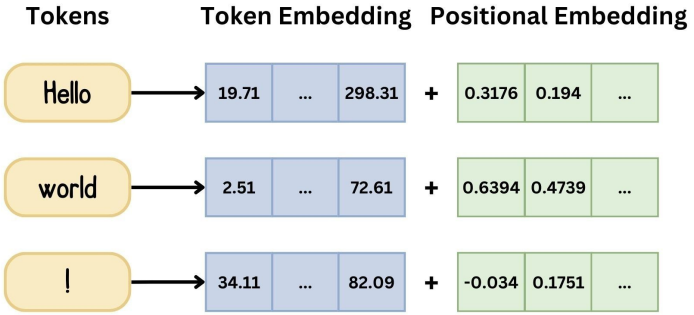
Attention([x1,x2,x3],...)=
Attention([x2,x3,x1],...)

"The cat ate the fish" = "The fish ate the cat"

Positional Encoding:
Adds position information to break the permutation invariance

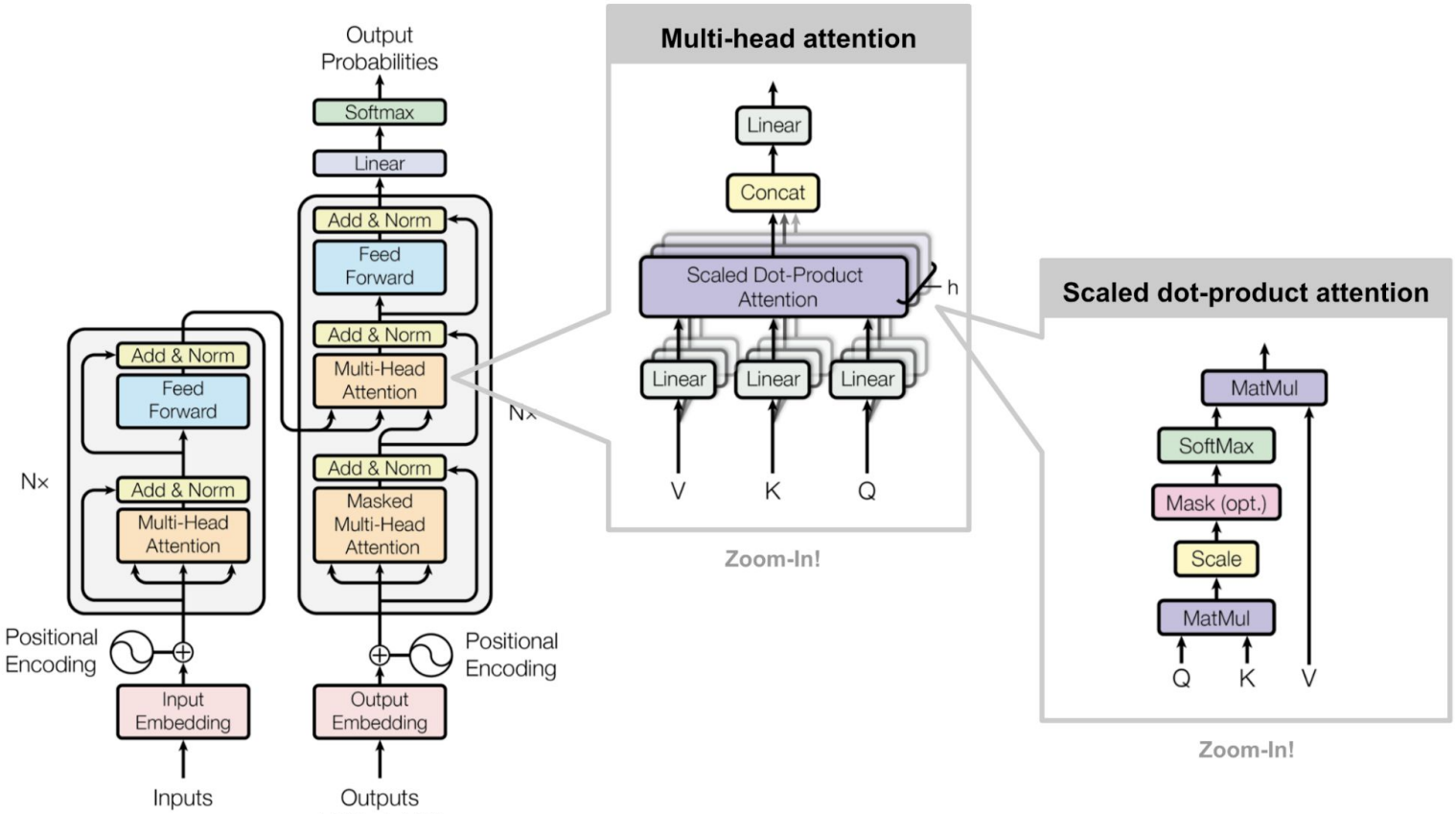


Sinusoidal Positional Encoding



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Transformers



Attention Is All You Need **2017**

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

NLP application

NLP models

BERT
Transformer trained to predict masked words in a sentence, reading bidirectionally

$$P(w_i|w_1,\ldots,w_{i-1},w_{i+1},\ldots,w_n)$$

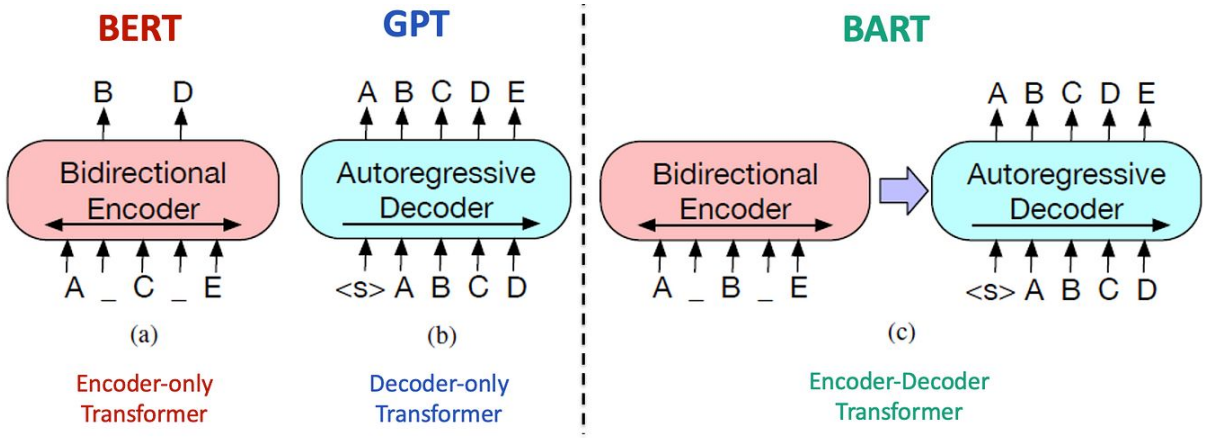
GPT
Transformer trained to predict the next token left-to-right

$$P(w_t|w_1,w_2,\ldots,w_{t-1})$$

Text sequences can be represented as:

$$S=(w_1,w_2,\ldots,w_n) \quad \text{where } w_i \in V$$

Where S is a sequence and V is the vocabulary.



scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Text to Value Tasks

Text Classification

Maps text sequences to K discrete categories (sentiment analysis, topic classification, spam detection)

$$f : \mathcal{S} \to 1, \ldots, K$$

Regression from Text

Predicts continuous values from text (price prediction, readability scores)

$$f : \mathcal{S} \to \mathbb{R}^n$$

Text to Text Tasks

Machine Translation

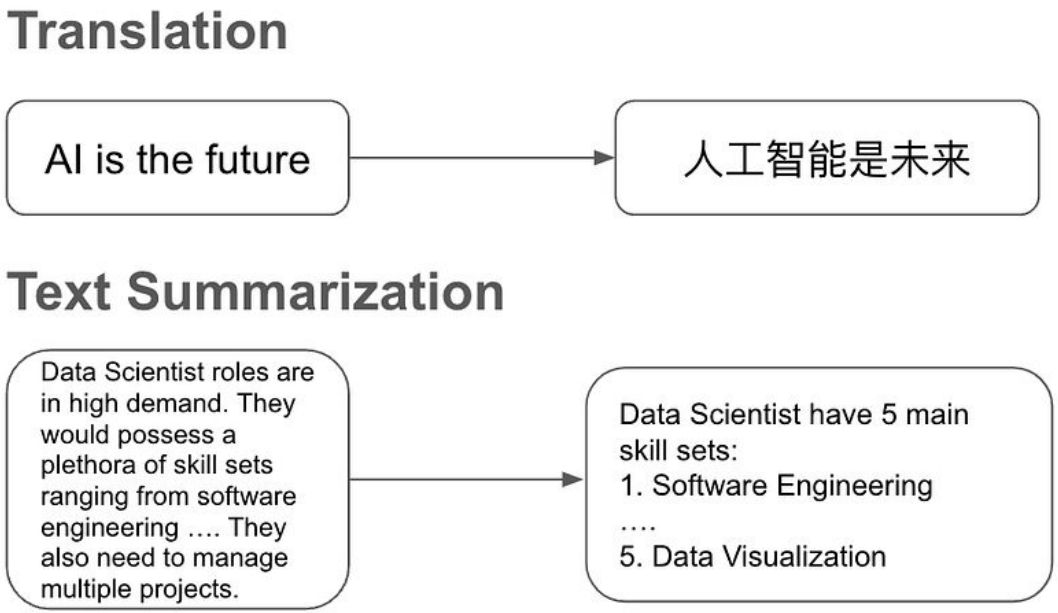
Converts text from source language L1 to target language L2

$$f : \mathcal{S}L1 \rightarrow \mathcal{S}L2$$

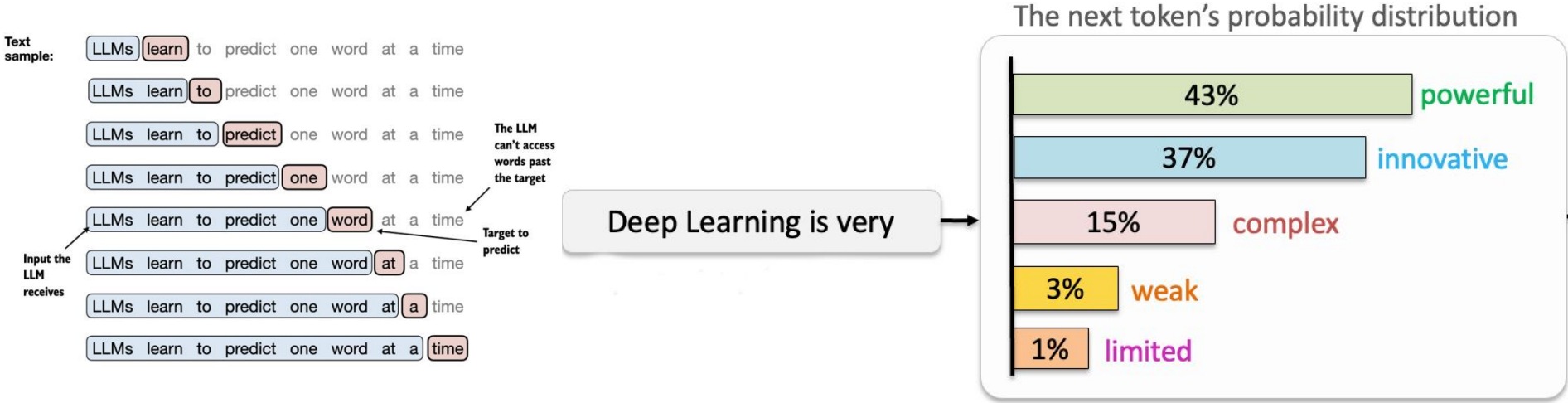
Text Summarization

Generates concise representation of longer text

$$f : \mathcal{S}long \rightarrow \mathcal{S}short$$



Generative Tasks



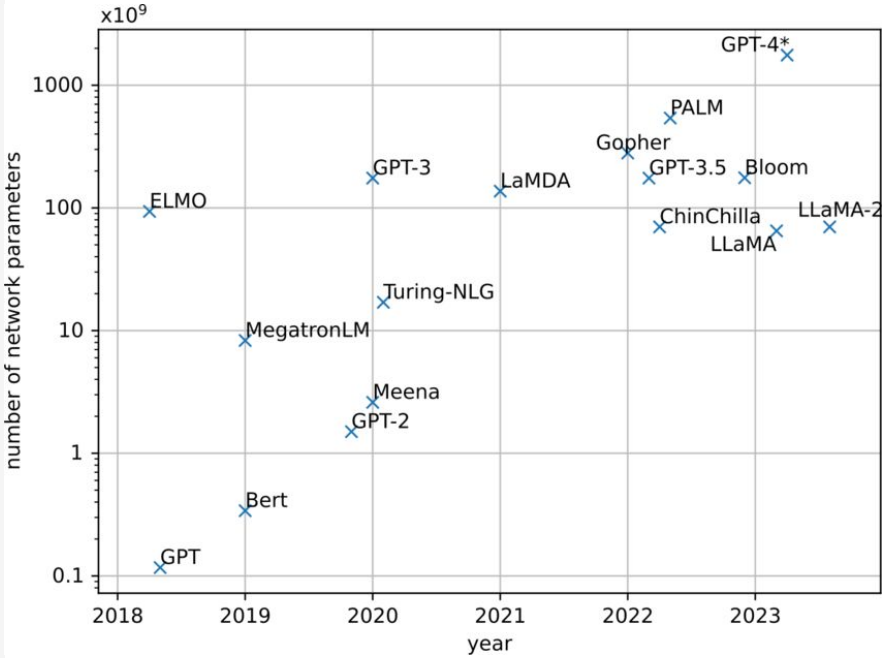
Autoregressive models (like gpt) are "classifiers" that, at each step, predicts a probability distribution over the vocabulary and selects the next token based on the input sequence

$$\ell_{CE}(y, \hat{y}) = -\sum_{i=1}^{n} \sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic})$$

LLM

Large language Model

The LLM era began when models exceeded **1 billion parameters** and were trained on hundreds of billions to trillions of tokens — which places the start around GPT-2/GPT-3 (2019-2020).



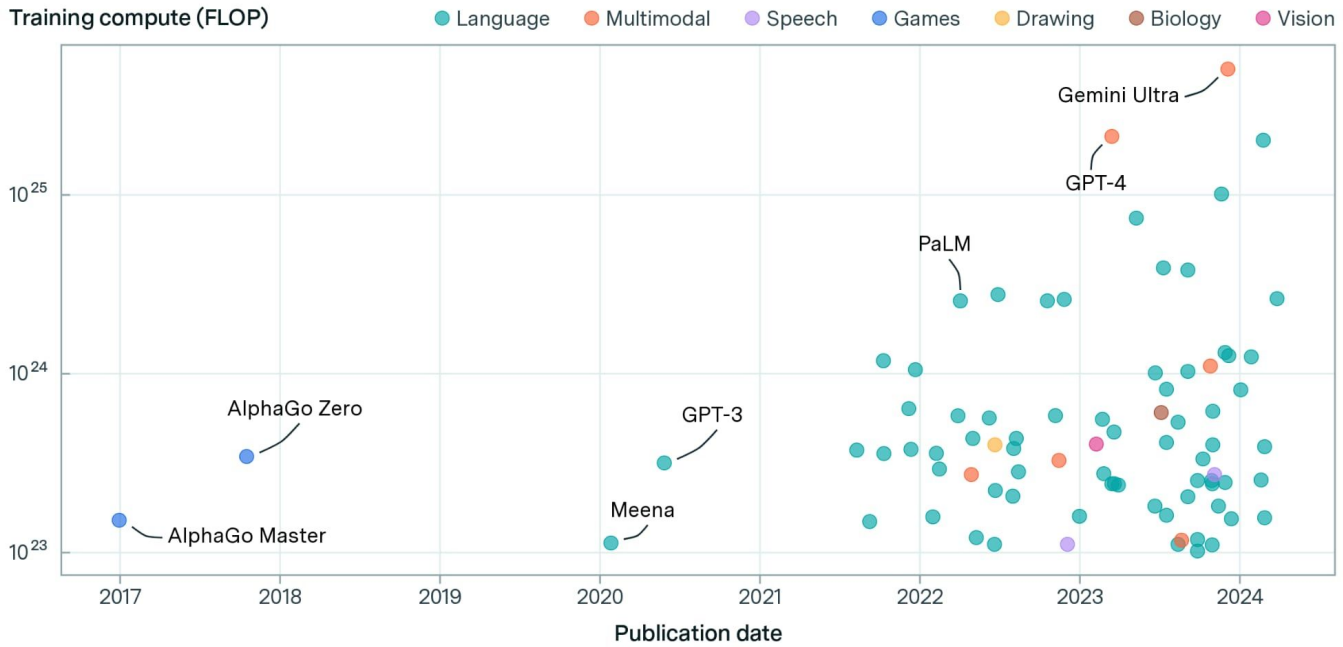
Number of parameters

	Training Set (Words)	Training Set (Tokens)
Recent LLMs		
Llama 3	11 trillion	15T
GPT-4	5 trillion	6.5T
Humans		
Human, age 5	30 million	
Human, age 20	150 million	

Data size

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Training Large Model

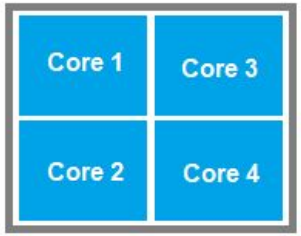


Training LLaMA 3.1 405B: 3.8×10^25 FLOPs

On a Personal Computer (200 GFLOPS)

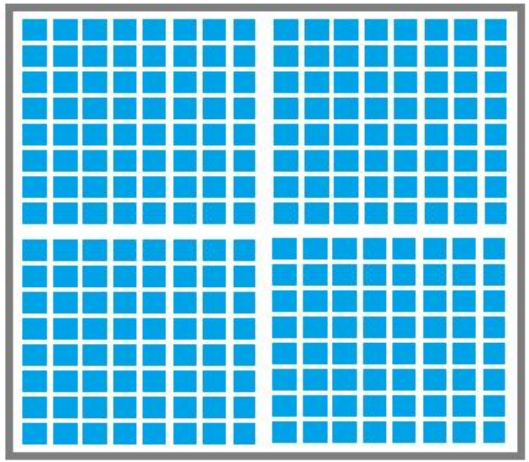
Time = Total FLOPs / FLOPS = 1.9 × 10^{14} seconds = **6.0 million years**

CPU VS GPU



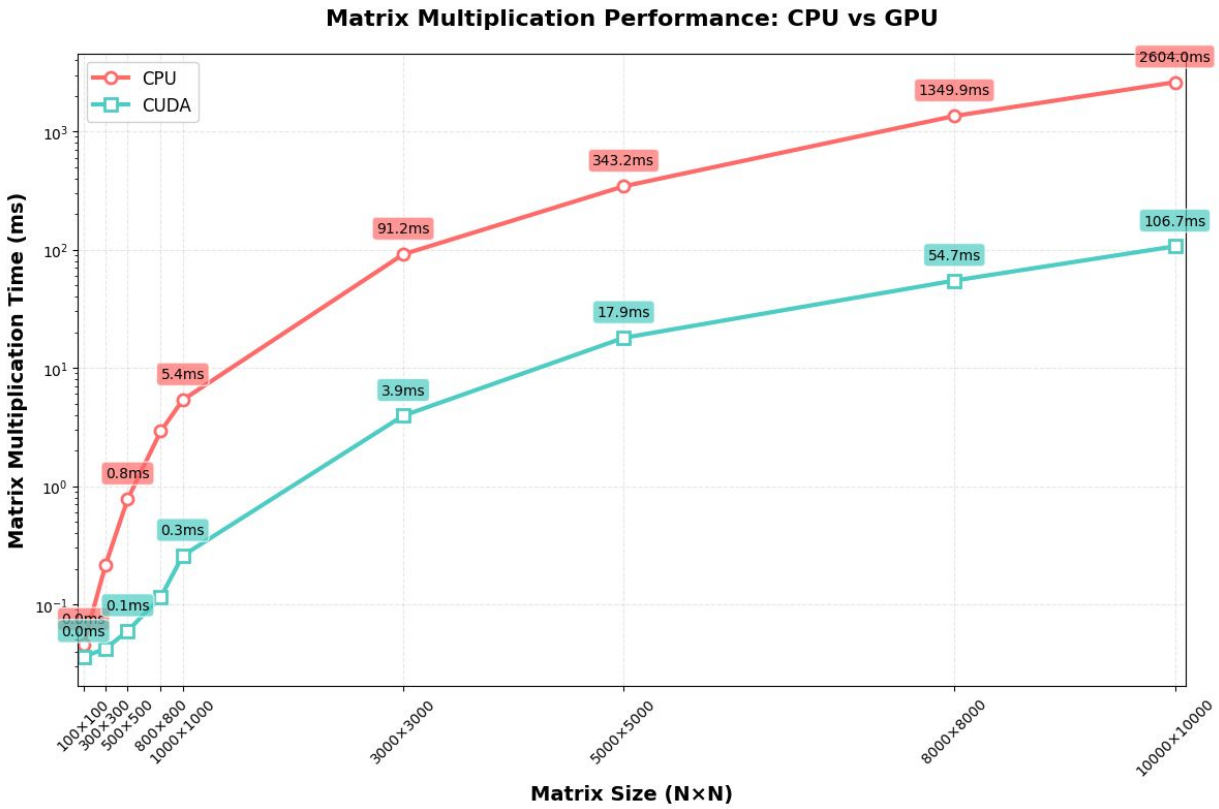
(Fewer strong cores)

CPU



(Thousands weaker cores)

GPU

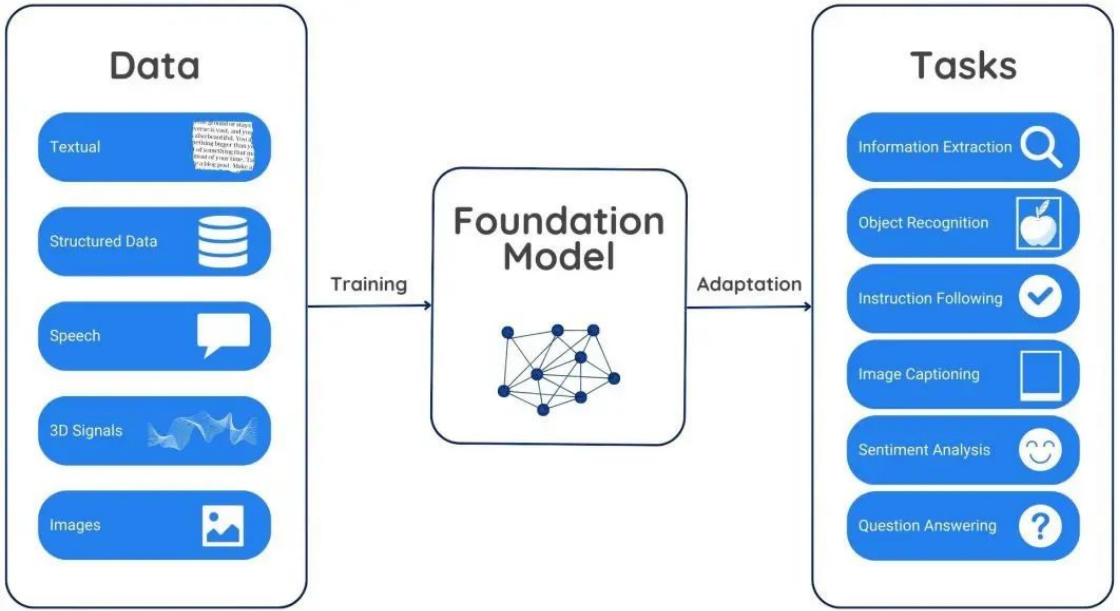


LLM : Resources and costs

Parameters	Number GPU	Training Time	Estimated Cost
1B	~10-50 GPUs	Days to week	$10K - $100K
10B	~100-500 GPUs	Weeks to Month	$100K - $1M
100B+	1,000+ GPUs	Months	$1M - $100M+

Transfer Learning pretrained Models

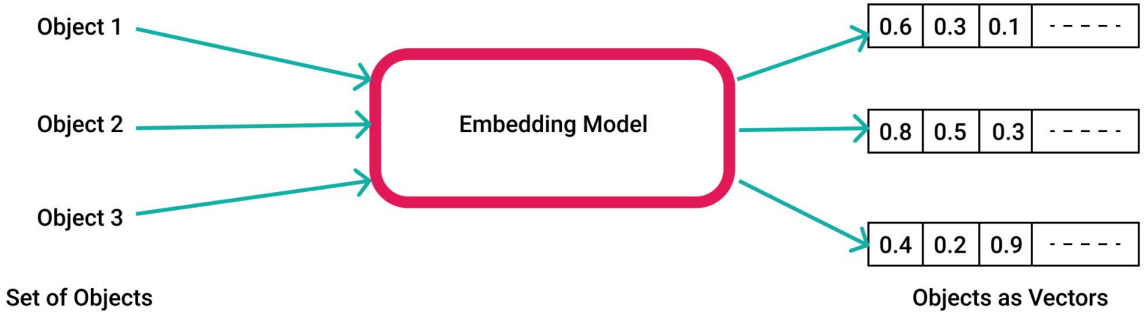
Foundation Model

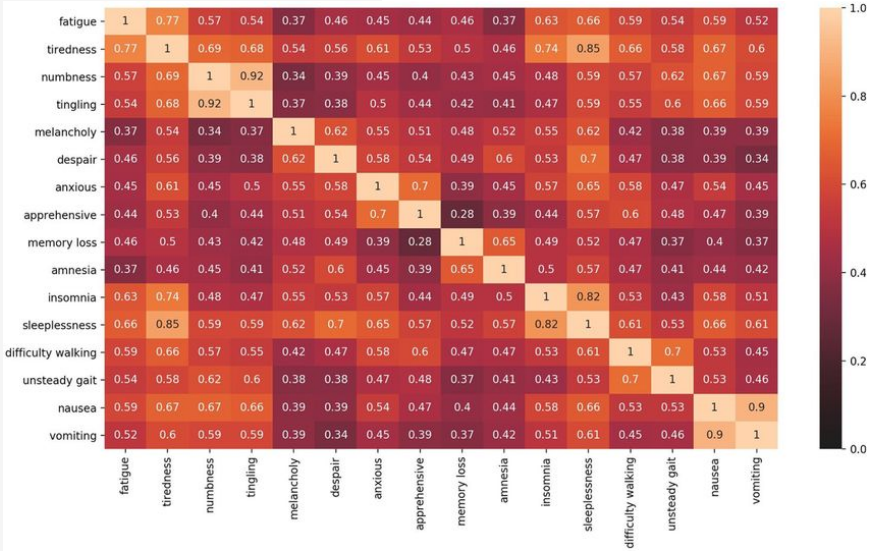


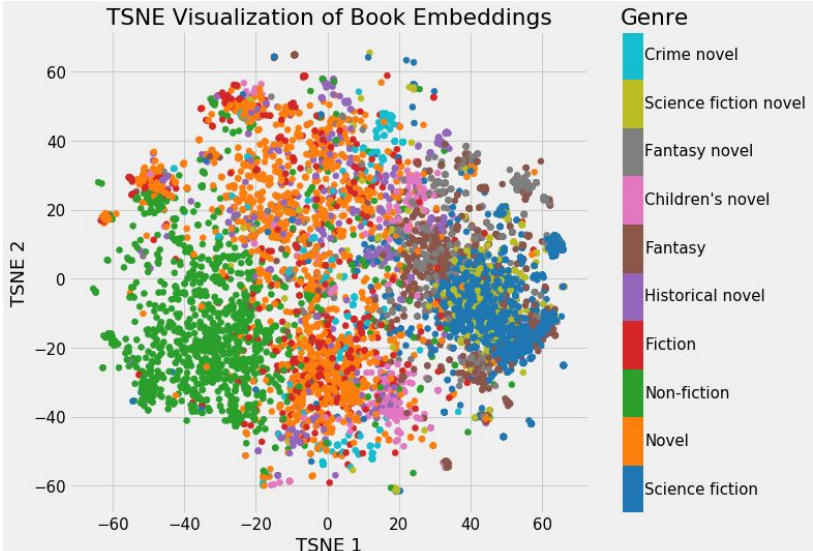
Foundation models are large AI models pre-trained on vast amounts of diverse data that can be adapted for a wide range of downstream tasks without training from scratch.

Embeddings from pretrained models

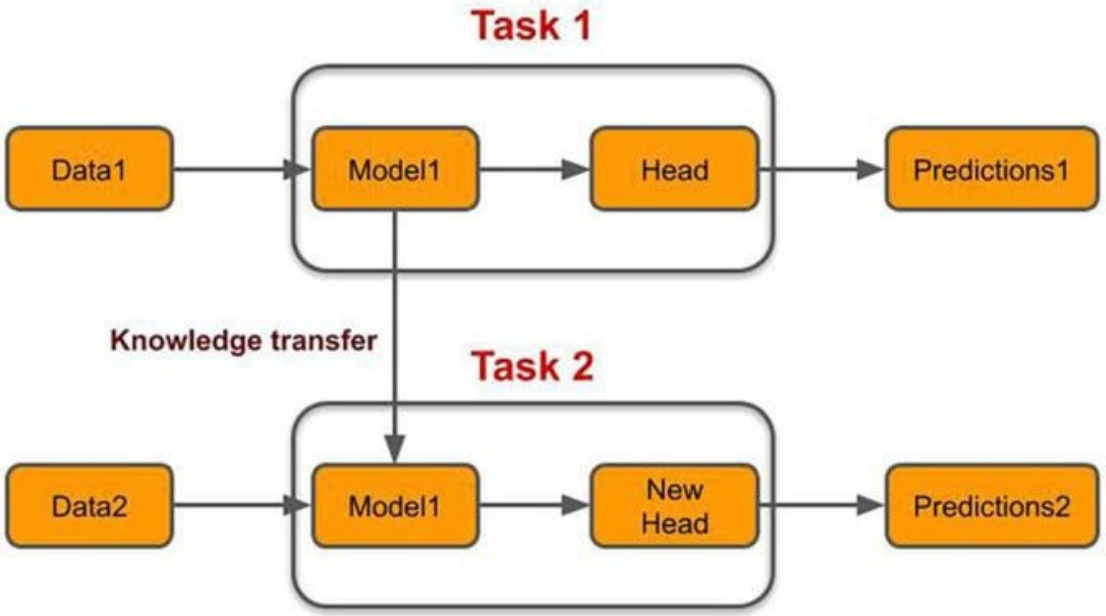
Embedding:
Numerical vector representations of data that capture semantic meaning.







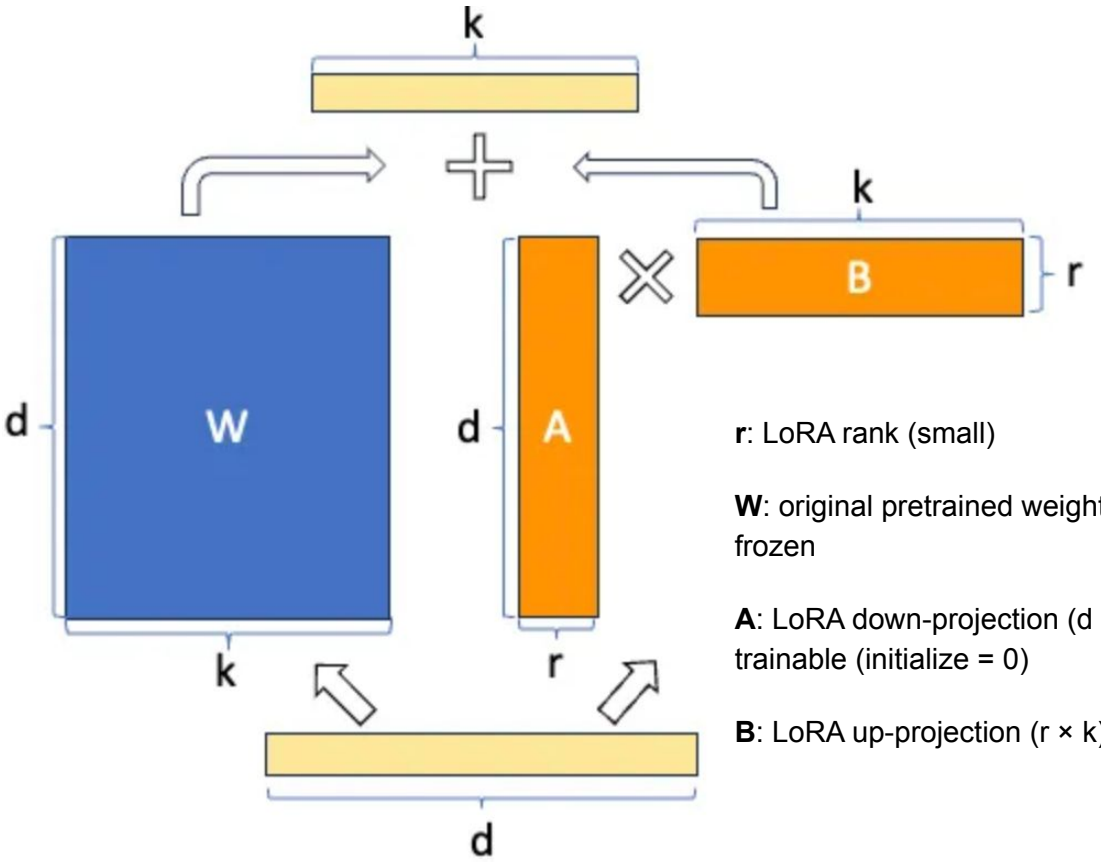
Transfer Learning



Instead of training a model from scratch (starting with randomly initialized parameters), we can start with a pre-trained model (one that has fine-tuned its parameters efficiently on a large dataset) and then fine-tune it in just a few iterations on our own (smaller) data.

Fine-tuning LLM

LoRA (Low-Rank Adaptation): Freezes the pretrained model weights and injects small trainable low-rank matrices into each layer, allowing **fine-tuning with a fraction of the parameters and memory**.



r: LoRA rank (small)

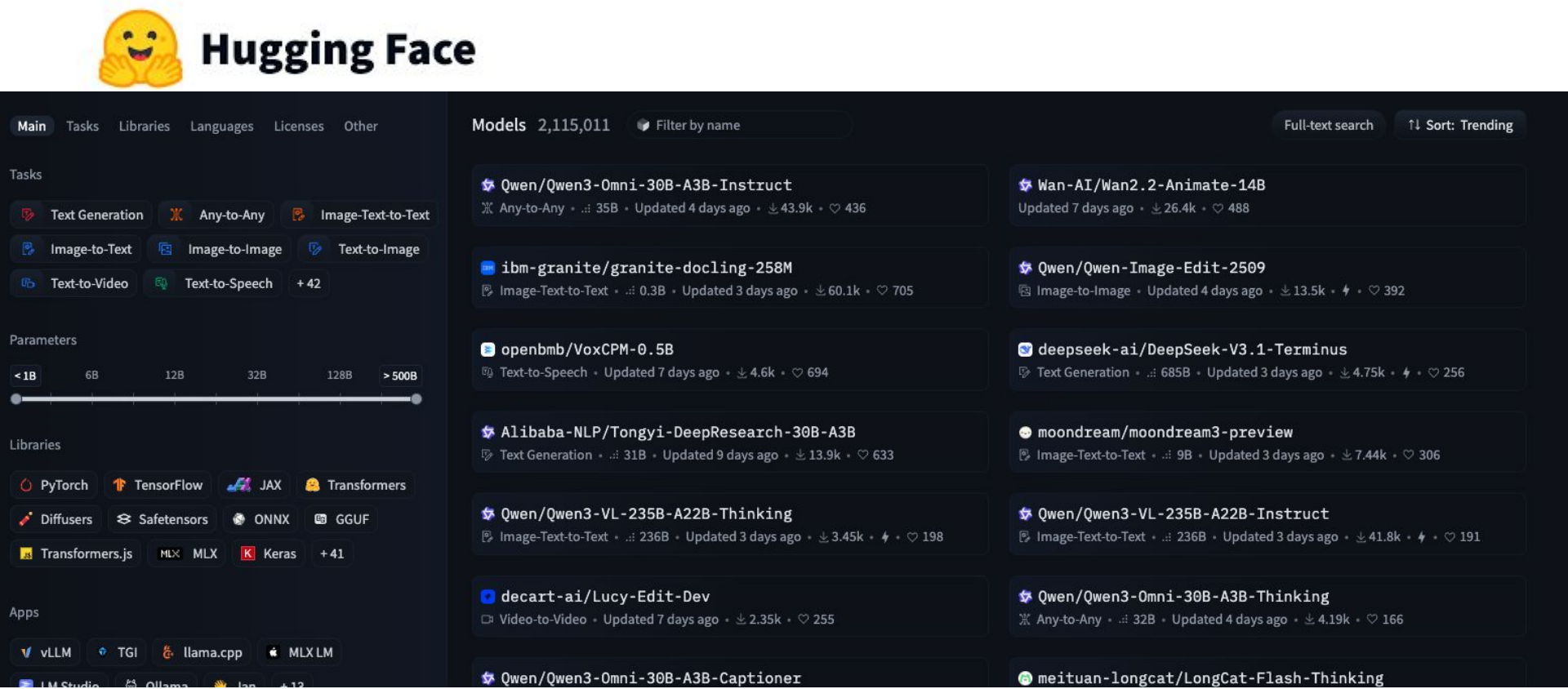
W: original pretrained weights (d × k) — frozen

A: LoRA down-projection (d × r) — trainable (initialize = 0)

B: LoRA up-projection (r × k) — trainable

scai
SORBONNE CLUSTER FOR ARTIFICIAL INTELLIGENCE

Open Source Pre-trained Models



Thanks

Embeddings : visualization