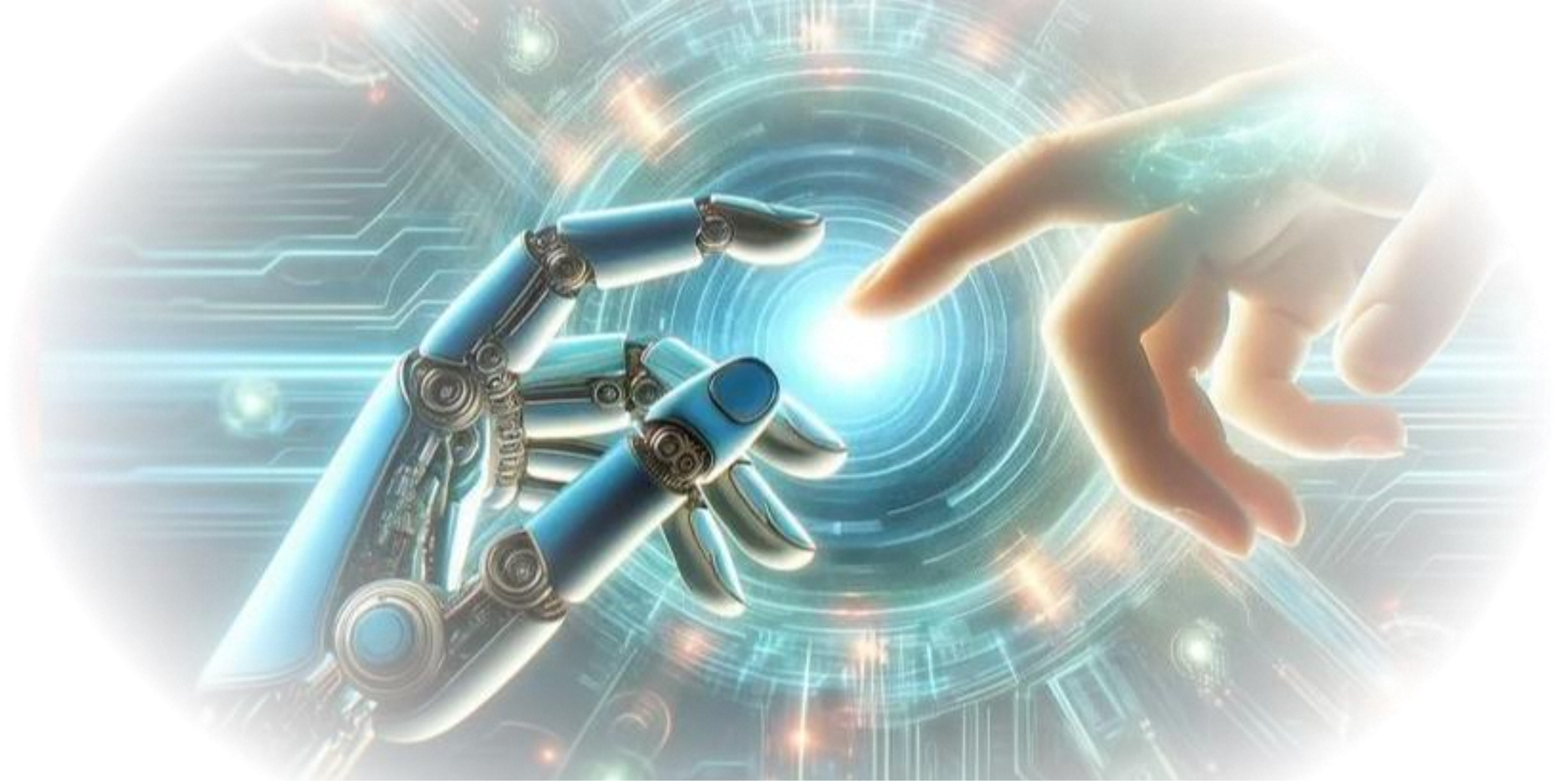


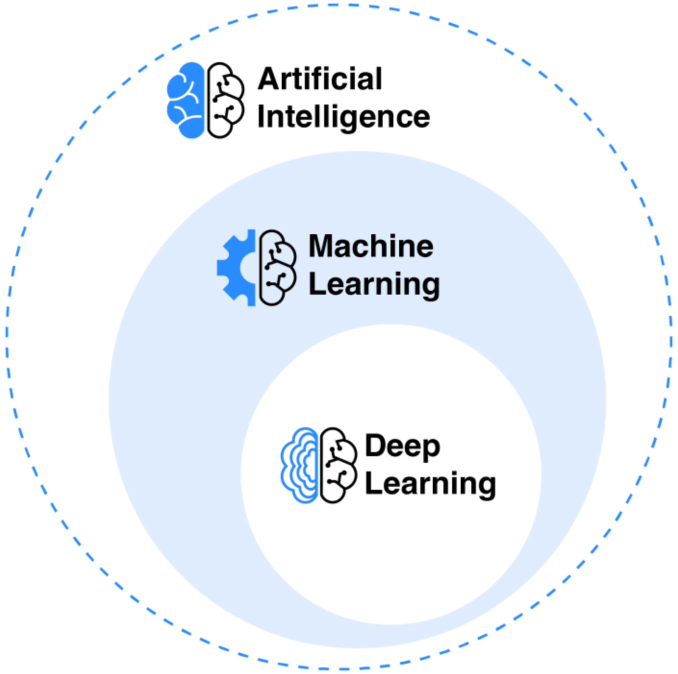
Deconstructing the AI Landscape

Pieces of technology that paved the way for AI and Generative AI

Venkatt Guhesan / Jan 22, 2026

Artificial Intelligence

Artificial Intelligence (AI) is the ability of computer systems to **perform tasks** that usually require **human intelligence**, like learning, problem-solving, reasoning, perception, and language understanding, often by learning from vast amounts of data rather than explicit programming. AI systems **create models from data** to **recognize patterns**, **make predictions**, **automate complex processes**, and **improve performance** over time



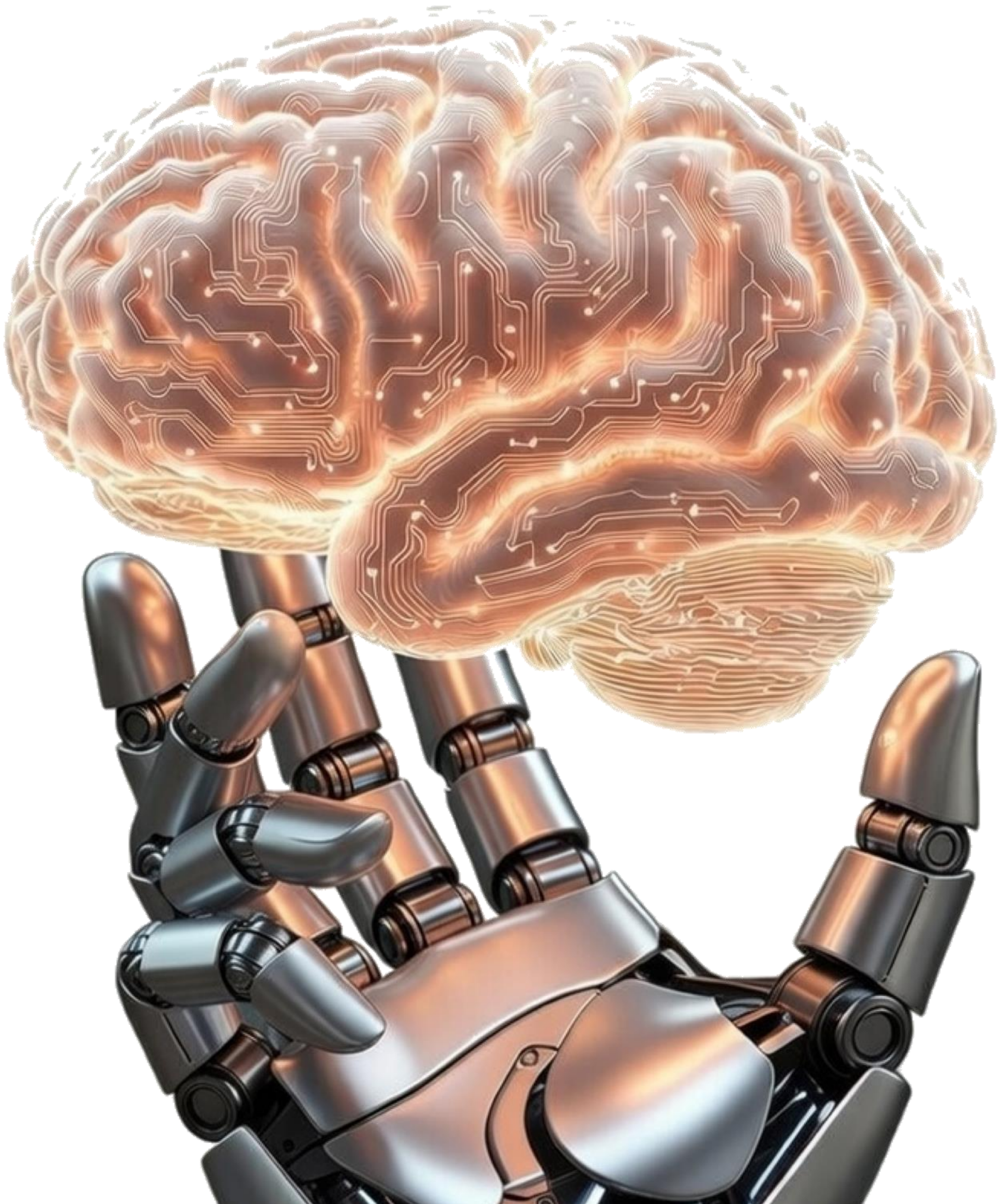
- **Learning:** AI learns from data, **identifying patterns** and improving its responses, similar to how a child learns to recognize objects.
- **Machine Learning (ML):** A core AI technique where systems learn from data to find patterns (e.g., recognizing cats in pictures after seeing many examples).
- **Deep Learning:** Uses **artificial neural networks**, **modeled on the human brain**, to process information and solve complex problems.

Tale of Three Travelers

Three Stories...

Three Paths...

Three Journeys...

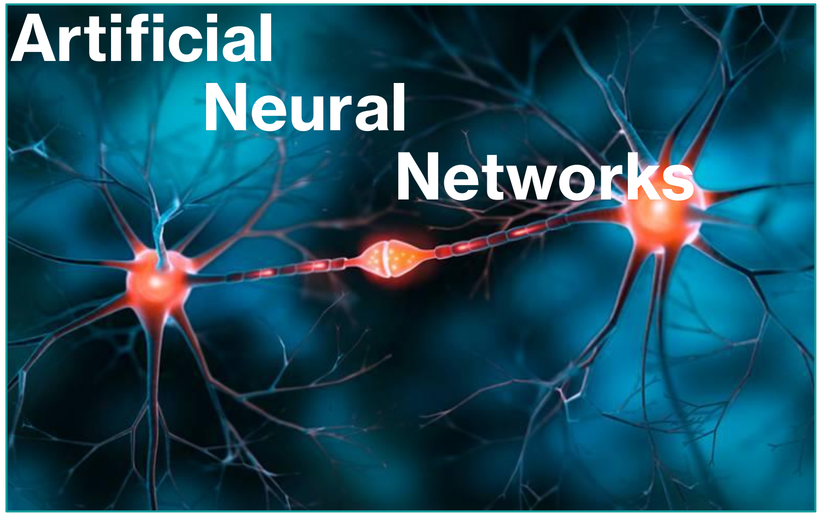


Story #1

Qwest for Intelligence

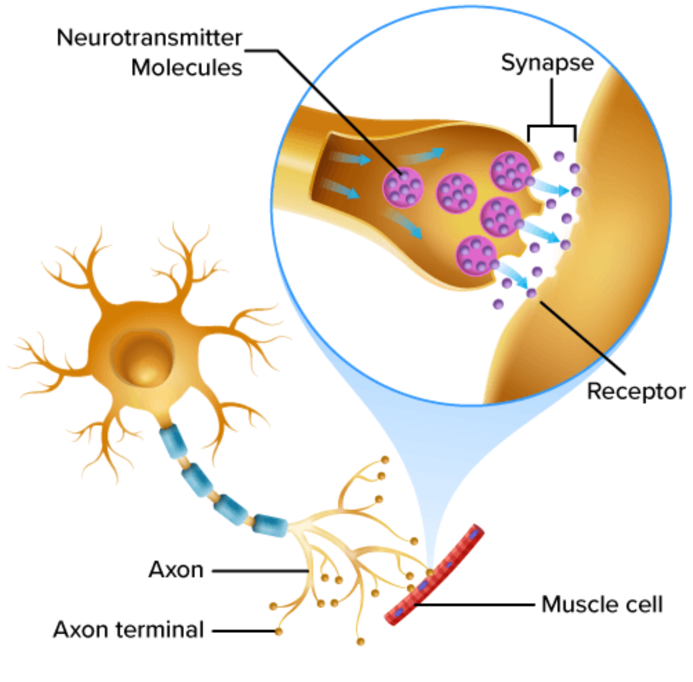
Reproducing The Mind

Artificial Neural Networks

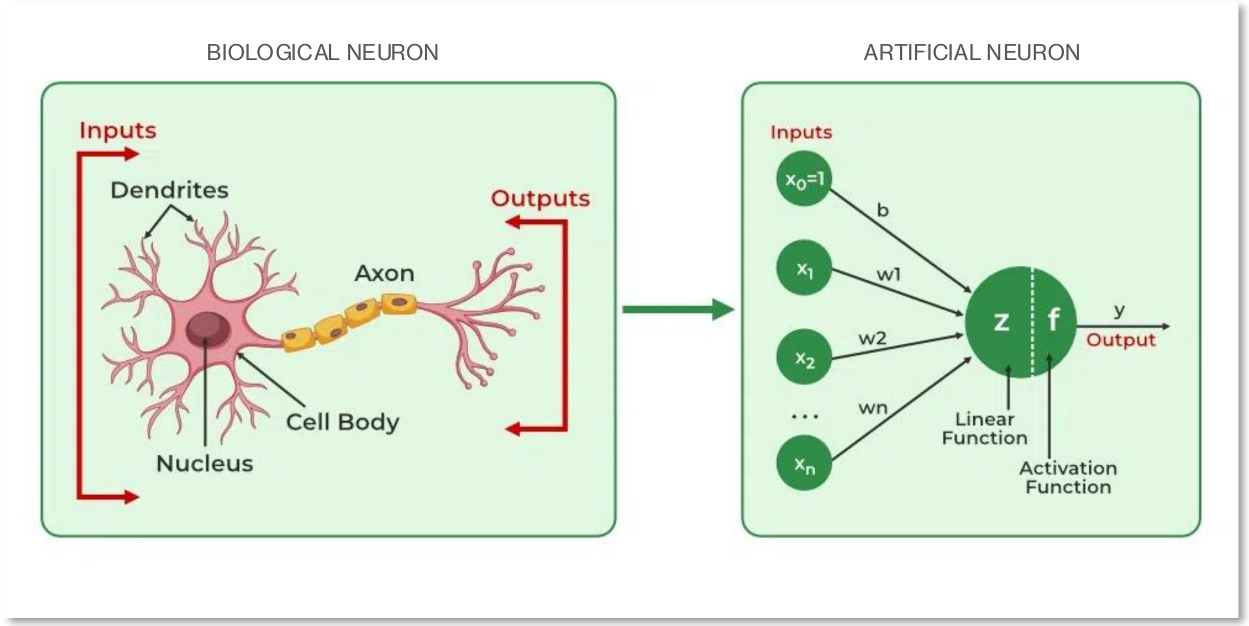


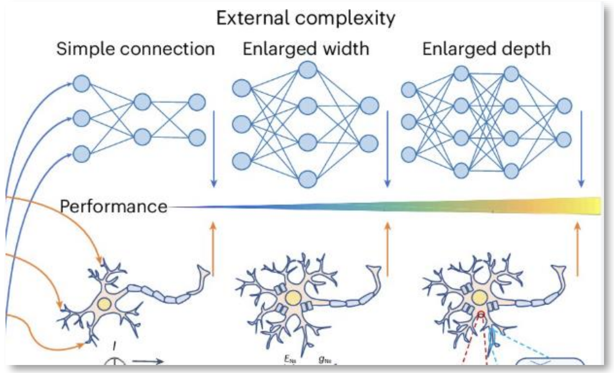
Artificial Neural Networks (ANN) are *mathematical models inspired by the brain's structure*, using interconnected "nodes" (like neurons) in layers to process data, learn patterns, and make decisions

- ✓ ***Inputs:*** *your five-senses.*
- ✓ ***Outputs****: the decisions you make after analysis*
- ✓ ***Connections****: (between your neurons and the synapses) Are the paths you take to arrive at a decision or conclusion.*



The Sodium (Na) and Potassium (K) ions create a voltage difference which are then picked up by the receptor of the next Dendrite



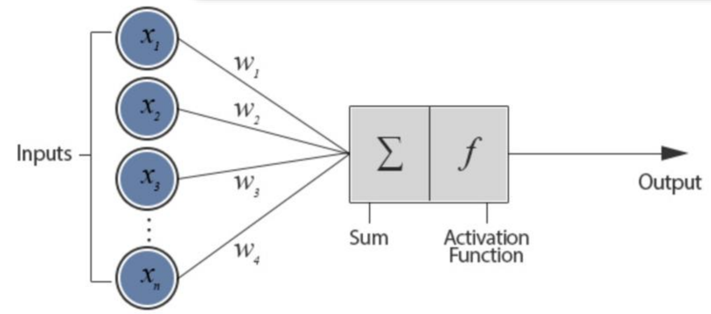


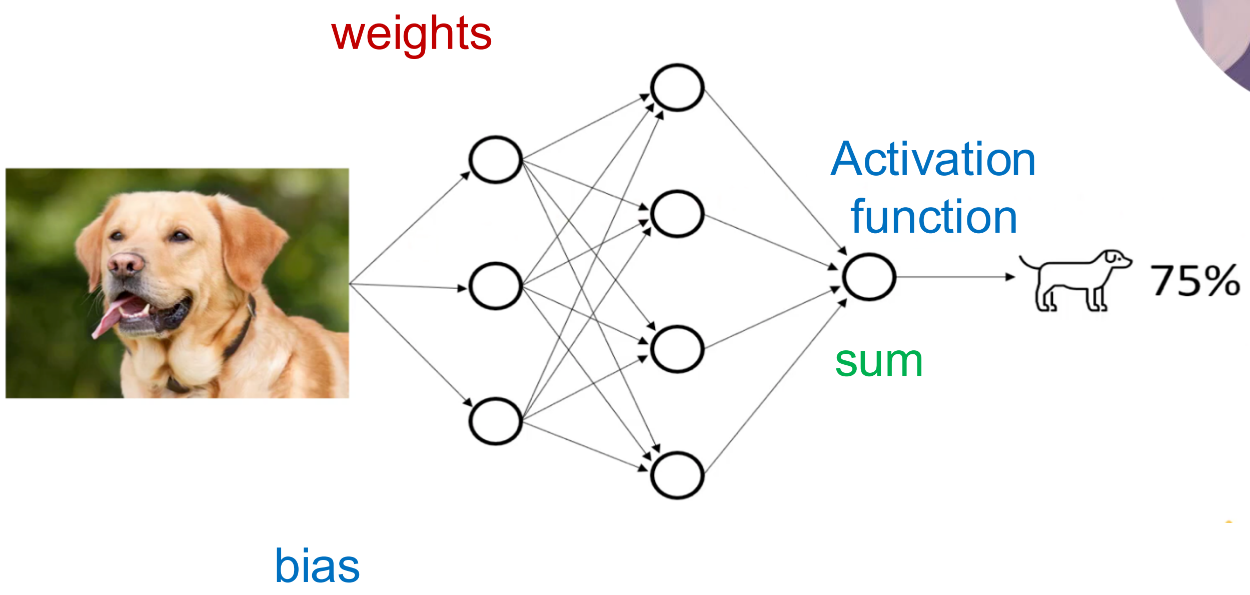
The presence of more Dendrites indicates more connections and increased complexity.

To reproduce the complexity artificially, equally increases the nodes within.

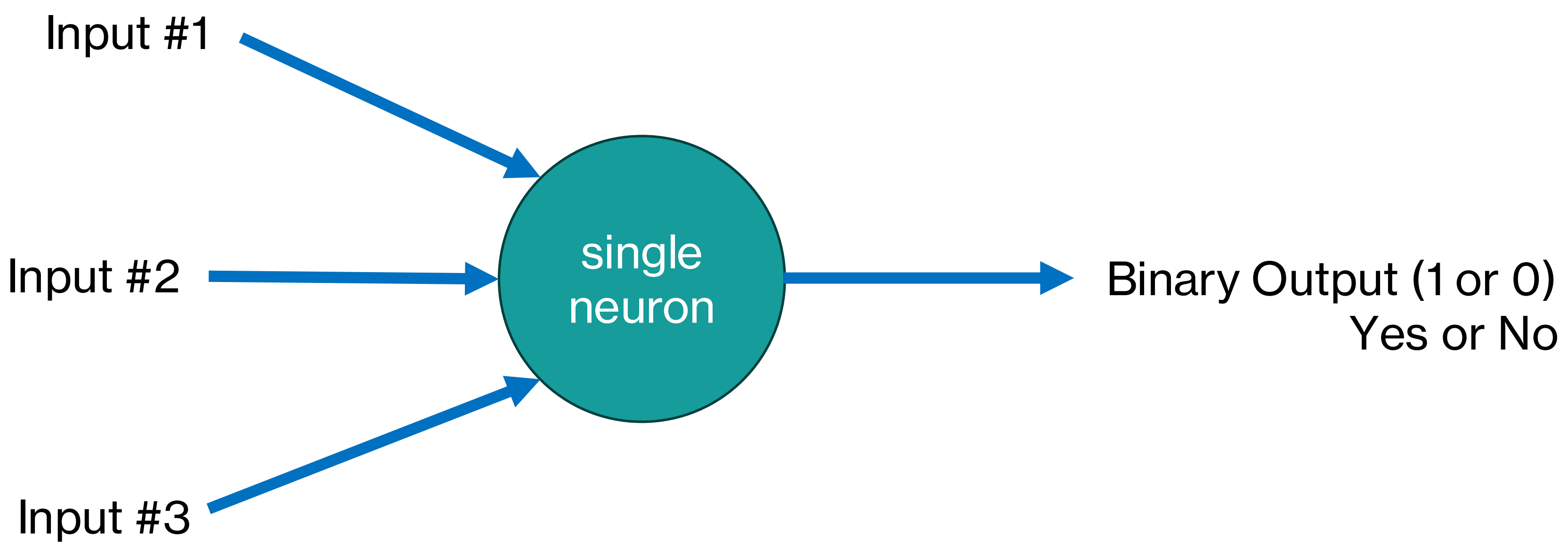
Understanding Neural Networks

Typical Neural Network Diagram



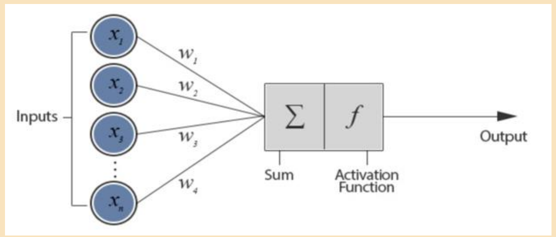


Simplified Representation of an Artificial Neuron



Perceptron

Introduced by Frank Rosenblatt in 1958, it's a linear classifier that learns by adjusting its weights to separate data, forming the basis for more complex deep learning models.



Simple Example

Should I go to the playground today?

Inputs:

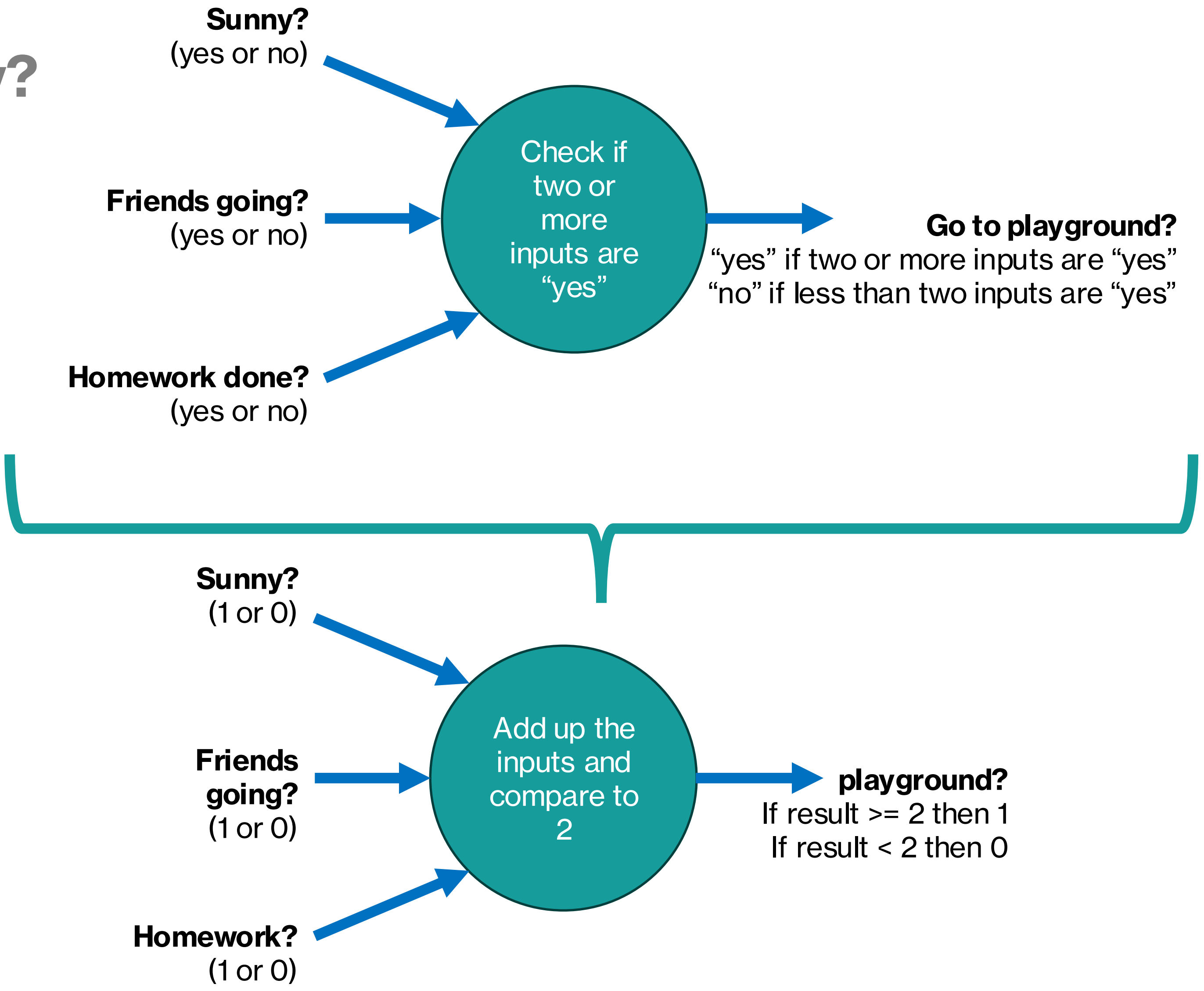
- 1. Is it sunny out? (yes or no)
- 2. Are my friends going? (yes or no)
- 3. Am I done with my homework? (yes or no)

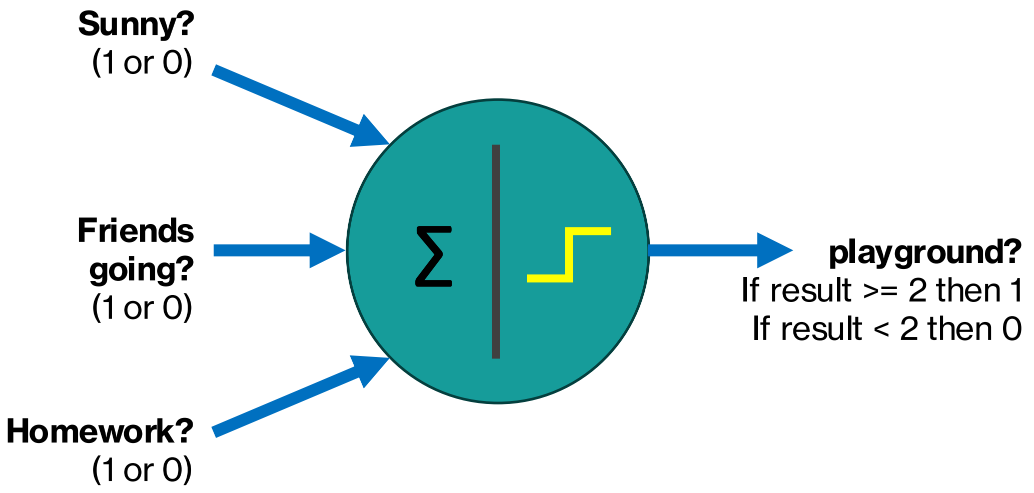
Output:

- Go to the playground? (yes or no)

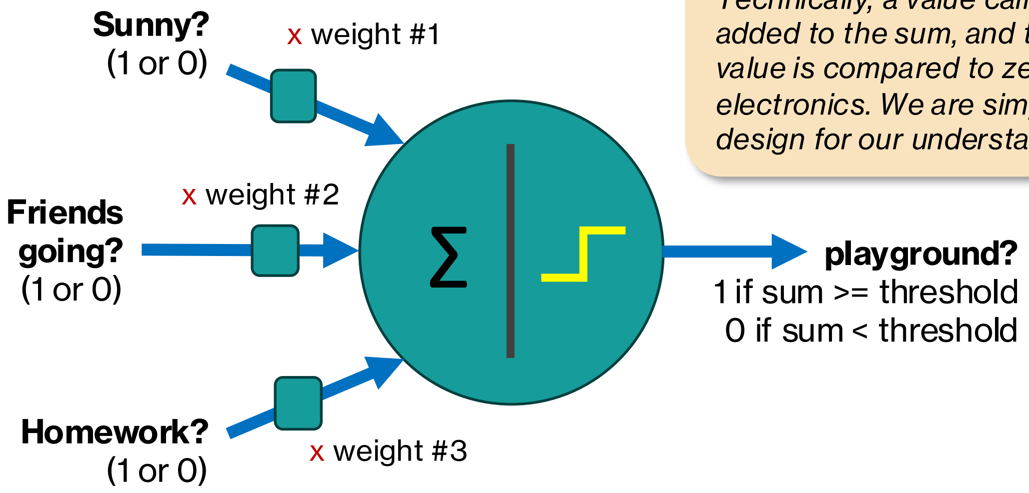
Simple Algorithm:

"If two or more of the inputs are 'yes', then I will go to the playground"





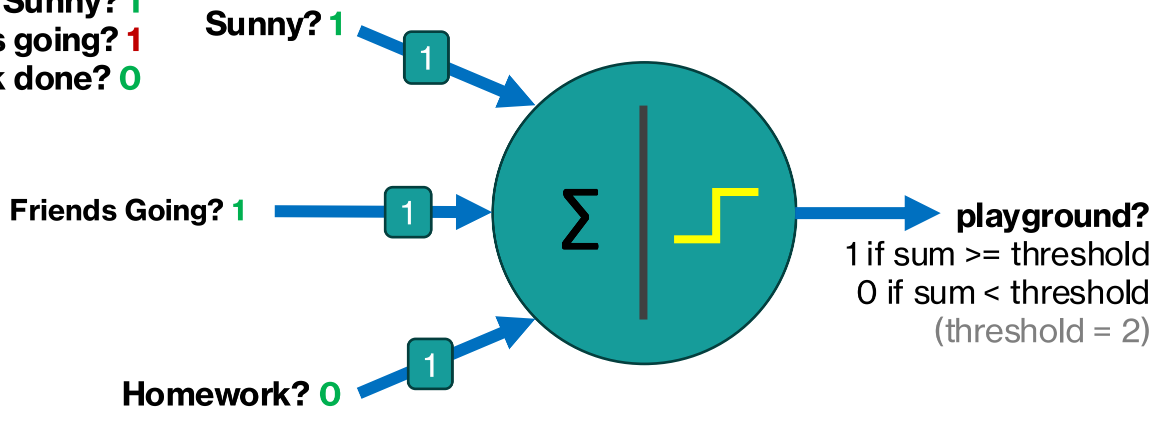
Let's introduce weights



Technically, a value called ***bias*** *is added to the sum, and the resulting value is compared to zero – in electronics. We are simplifying the design for our understanding.*

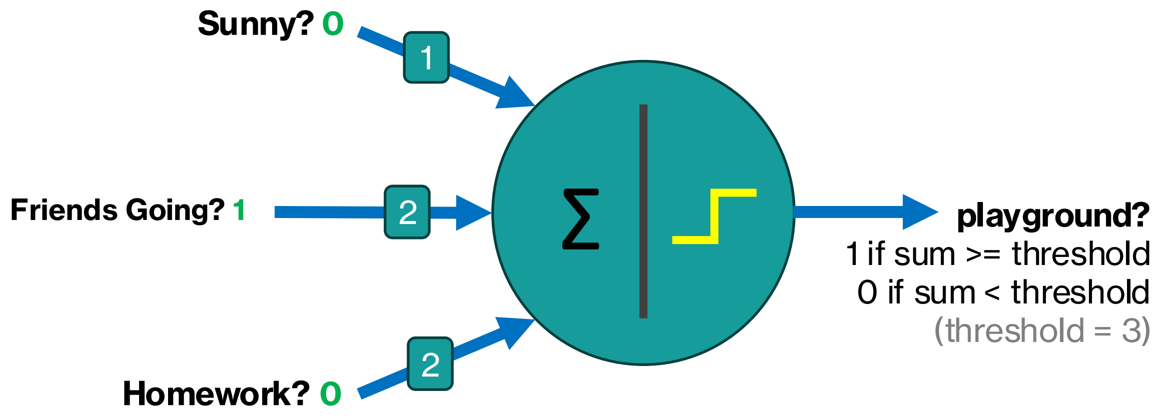
Let's start our example with equal weights

Sunny? 1
Friends going? 1
Homework done? 0



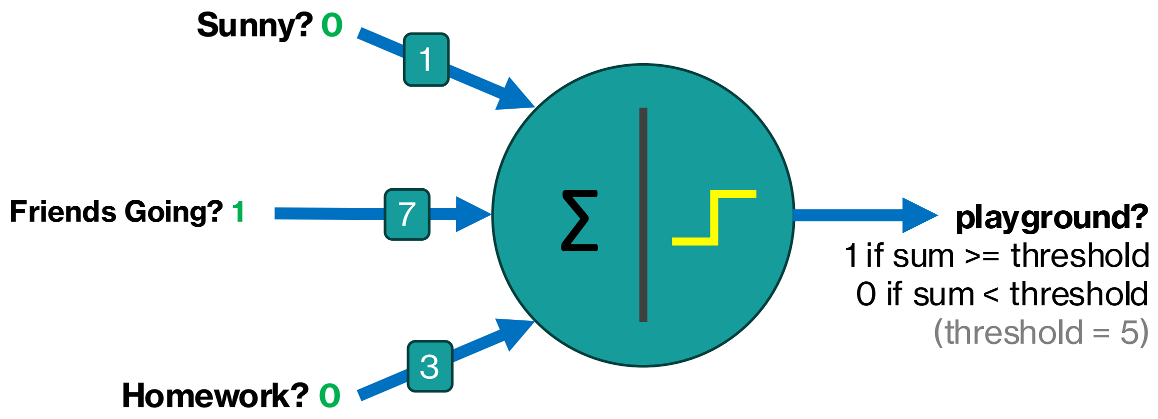
= 1 x1 + 1 x 1 + 0 x 1 = 1 + 1 + 0= 2 → 1 (yes)

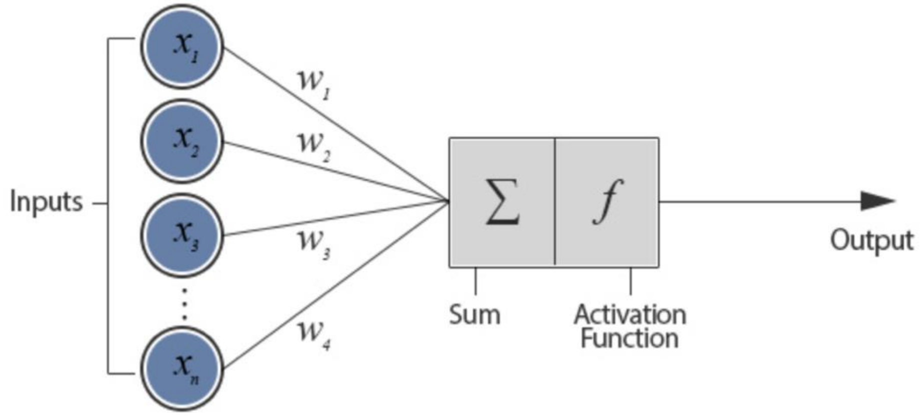
But if it rains, or it's cold, then I can always wear a raincoat or a warm jacket!!! → Change the weights



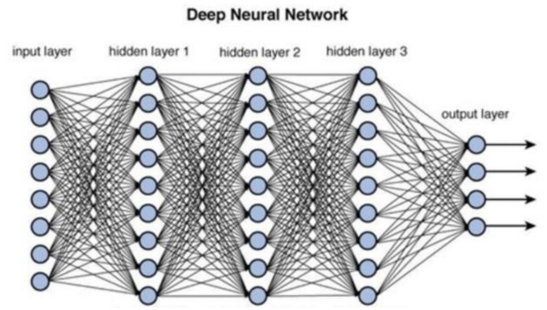
= 0 x1 + 1 x 2 + 0 x 2= 0 + 2 + 0 = 2 → 1 (no)

Continuously tweaking the bias, the weights and the threshold is what we call training the model!

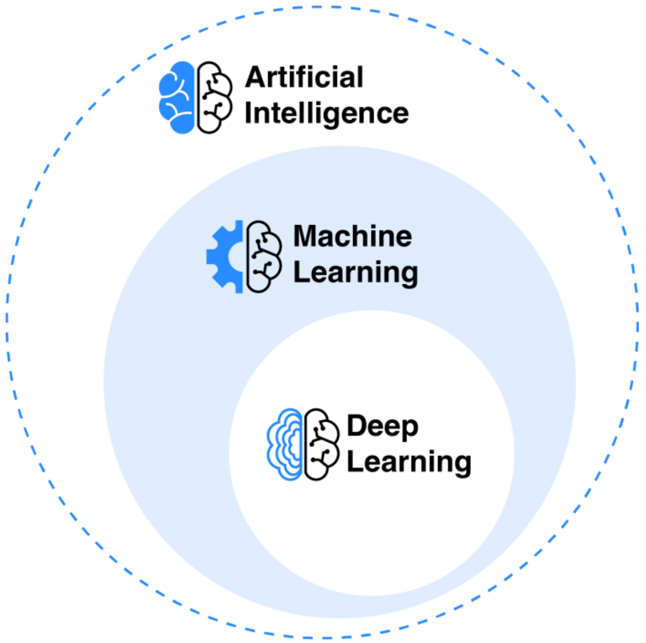




In a real-world scenario, you may have millions of these neurons represented in a model

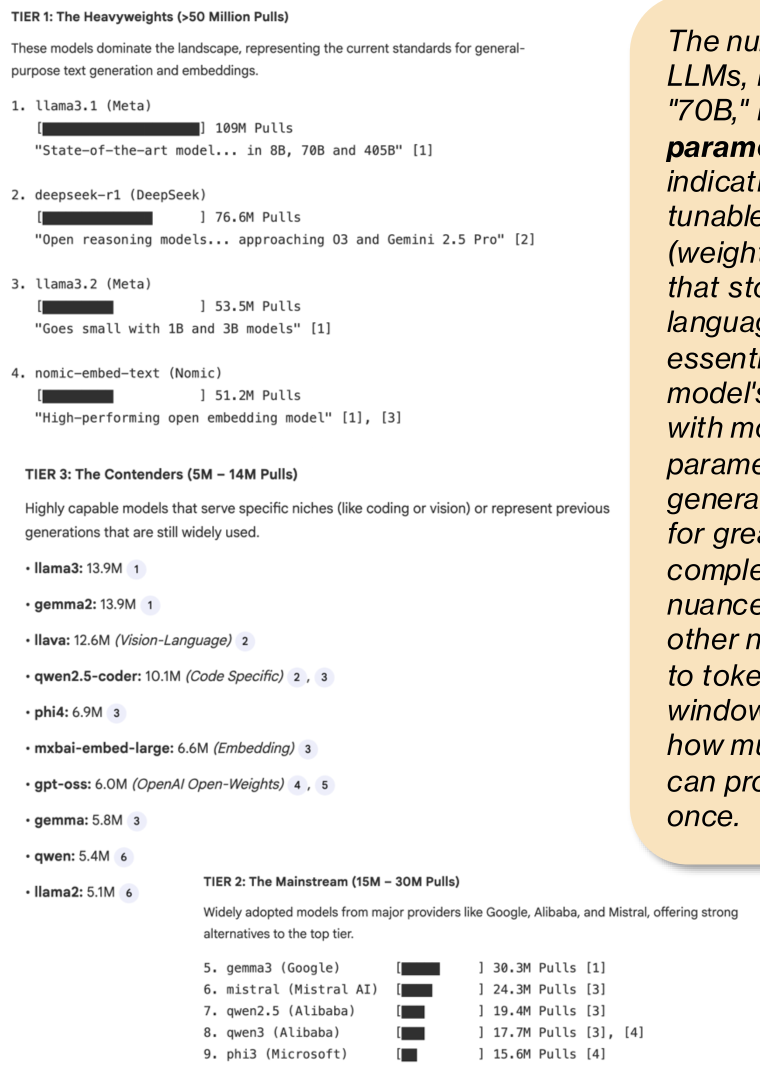


AKA
Transformer Model
Neural Network



Open List of Downloadable & API Usable Models (LLMs)

- ✓ https://ollama.com/library
- ✓ https://huggingface.co/models



TIER 1: The Heavyweights (>50 Million Pulls)

These models dominate the landscape, representing the current standards for general-purpose text generation and embeddings.

```
1. llama3.1 (Meta)
   [████████████████████] 109M Pulls
   "State-of-the-art model... in 8B, 70B and 405B" [1]

2. deepseek-r1 (DeepSeek)
   [██████████████      ] 76.6M Pulls
   "Open reasoning models... approaching O3 and Gemini 2.5 Pro" [2]

3. llama3.2 (Meta)
   [█████████           ] 53.5M Pulls
   "Goes small with 1B and 3B models" [1]

4. nomic-embed-text (Nomic)
   [█████████           ] 51.2M Pulls
   "High-performing open embedding model" [1], [3]
```

TIER 3: The Contenders (5M – 14M Pulls)

Highly capable models that serve specific niches (like coding or vision) or represent previous generations that are still widely used.

- **llama3:** 13.9M 1
- **gemma2:** 13.9M 1
- **llava:** 12.6M *(Vision-Language)* 2
- **qwen2.5-coder:** 10.1M *(Code Specific)* 2 , 3
- **phi4:** 6.9M 3
- **mxbai-embed-large:** 6.6M *(Embedding)* 3
- **gpt-oss:** 6.0M *(OpenAI Open-Weights)* 4 , 5
- **gemma:** 5.8M 3
- **qwen:** 5.4M 6
- **llama2:** 5.1M 6

TIER 2: The Mainstream (15M – 30M Pulls)

Widely adopted models from major providers like Google, Alibaba, and Mistral, offering strong alternatives to the top tier.

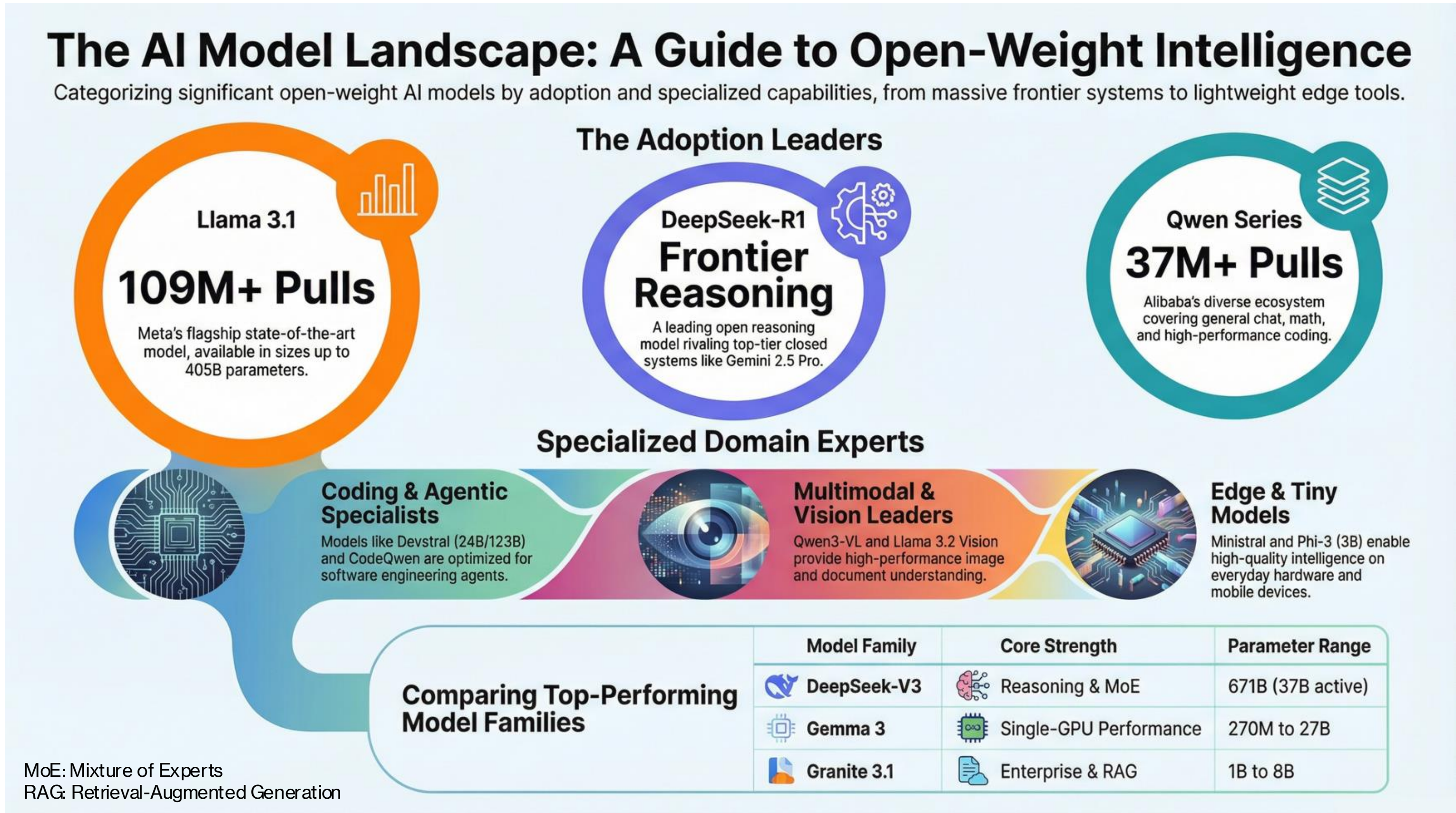
```
5. gemma3 (Google)      [█████     ] 30.3M Pulls [1]
6. mistral (Mistral AI) [████      ] 24.3M Pulls [3]
7. qwen2.5 (Alibaba)    [███       ] 19.4M Pulls [3]
8. qwen3 (Alibaba)      [███       ] 17.7M Pulls [3], [4]
9. phi3 (Microsoft)     [██        ] 15.6M Pulls [4]
```

*The numbers with LLMs, like "7B" or "70B," refer to their **parameter count,** indicating billions of tunable values (weights/biases) that store learned language patterns, essentially the model's "**brain size**," with more parameters generally allowing for greater complexity and nuance, though other numbers refer to token context windows, defining how much text it can process at once.*

Token Counts: Numbers indicating the model's **context window**, the maximum number of tokens (words/parts of words) it can consider at once for a given input and output.

Dimensions (e.g., 4096): Refers to the size of "embedding vectors" that represent words as lists of numbers, with longer lists (more dimensions) capturing more nuanced meaning.

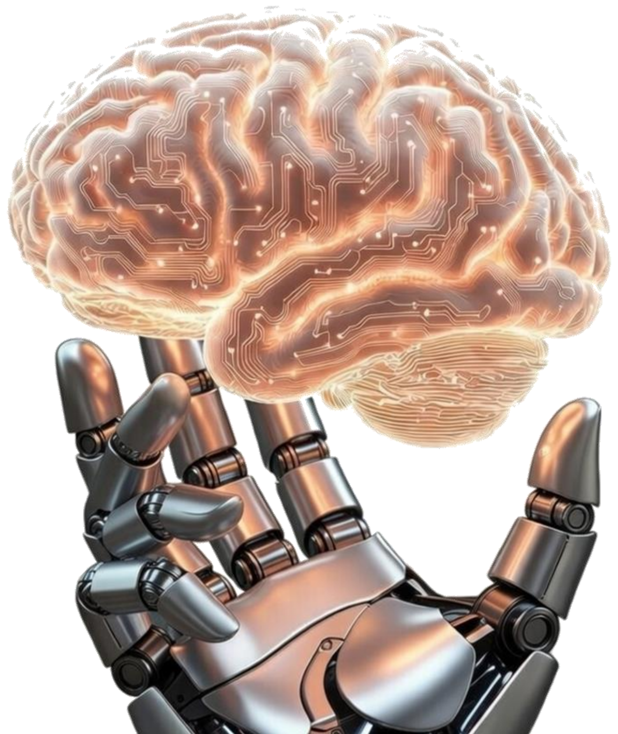
Version Numbers (e.g., Llama 2): Denotes different releases or iterations, indicating improvements or changes in architecture or training data.

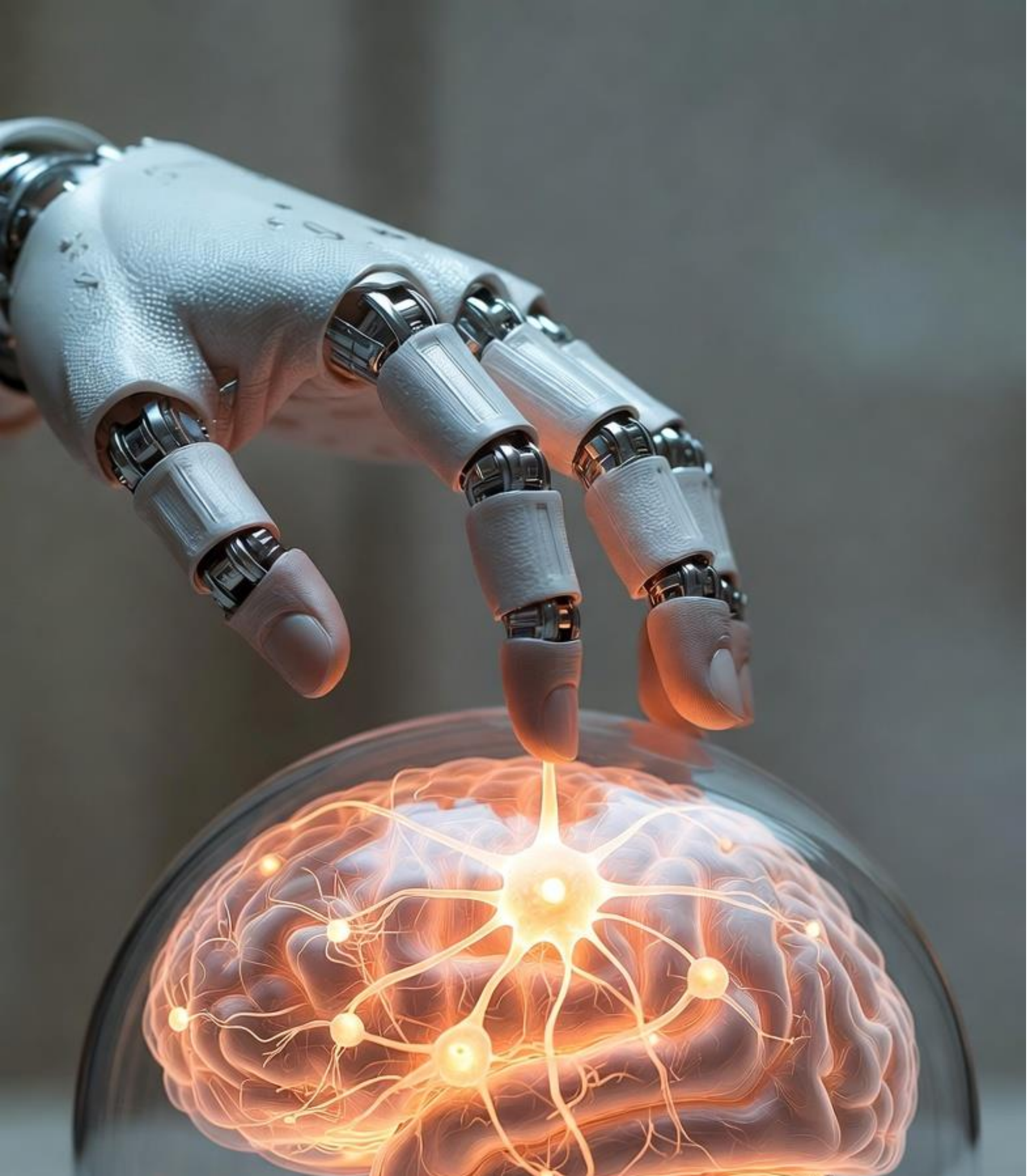

The AI Model Landscape: A Guide to Open-Weight Intelligence
Categorizing significant open-weight AI models by adoption and specialized capabilities, from massive frontier systems to lightweight edge tools.
The Adoption Leaders
Llama 3.1
109M+ Pulls
Meta's flagship state-of-the-art model, available in sizes up to 405B parameters.
DeepSeek-R1
Frontier Reasoning
A leading open reasoning model rivaling top-tier closed systems like Gemini 2.5 Pro.
Qwen Series
37M+ Pulls
Alibaba's diverse ecosystem covering general chat, math, and high-performance coding.
Specialized Domain Experts
Coding & Agentic Specialists
Models like Devstral (24B/123B) and CodeQwen are optimized for software engineering agents.
Multimodal & Vision Leaders
Qwen3-VL and Llama 3.2 Vision provide high-performance image and document understanding.
Edge & Tiny Models
Ministral and Phi-3 (3B) enable high-quality intelligence on everyday hardware and mobile devices.
Comparing Top-Performing Model Families

Model Family	Core Strength	Parameter Range
DeepSeek-V3	Reasoning & MoE	671B (37B active)
Gemma 3	Single-GPU Performance	270M to 27B
Granite 3.1	Enterprise & RAG	1B to 8B

MoE: Mixture of Experts
RAG: Retrieval-Augmented Generation

We now have a way to represent the mind!





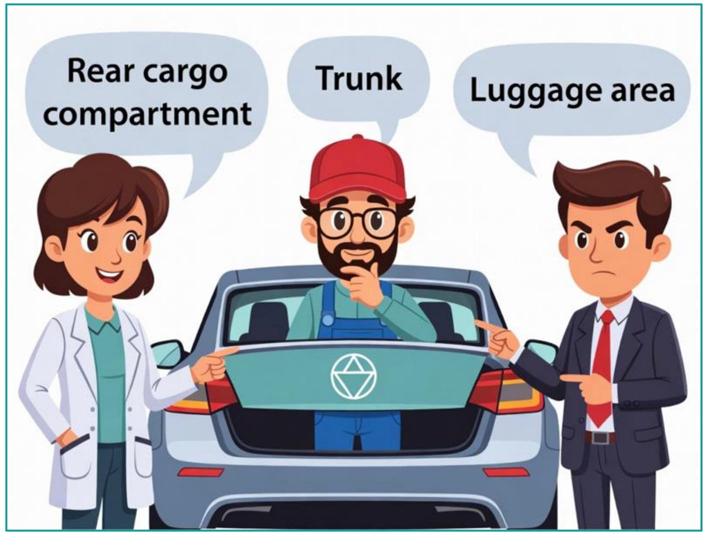
Story #2

In Search of Meaning

Reproducing The Context

typical conversations in a vehicle manufacturing plant

Hood!
Bonnet!
Clamshell!

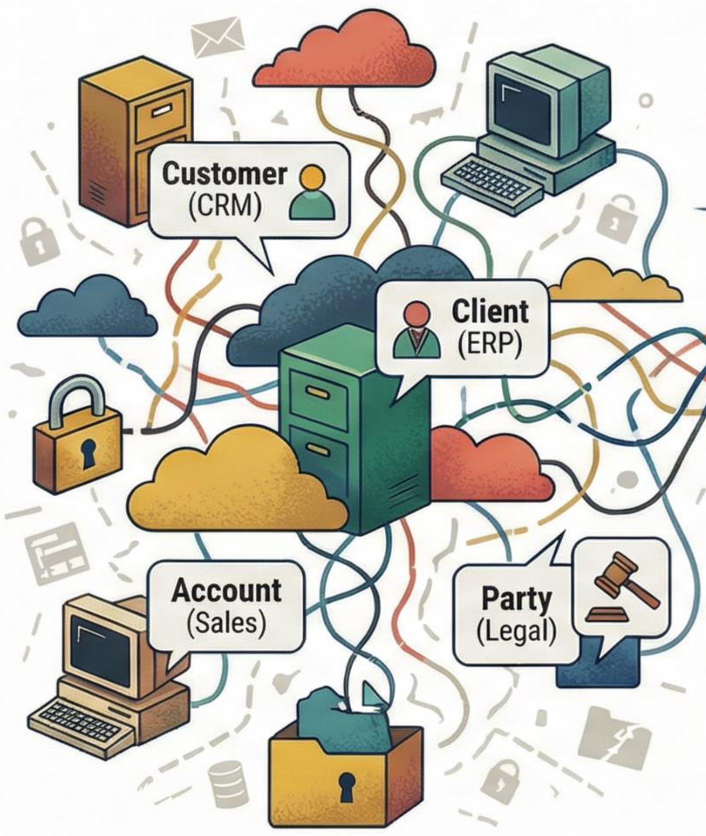


Metallic Silver Finish
Silver Paint Job
Elegant Silver Hue

No Common Vocabulary
→ Semantic Gap!!!

Bridging the Semantic Gap: How Knowledge Graphs Unify Enterprise Data

The Problem: The Semantic Gap & Data Silos



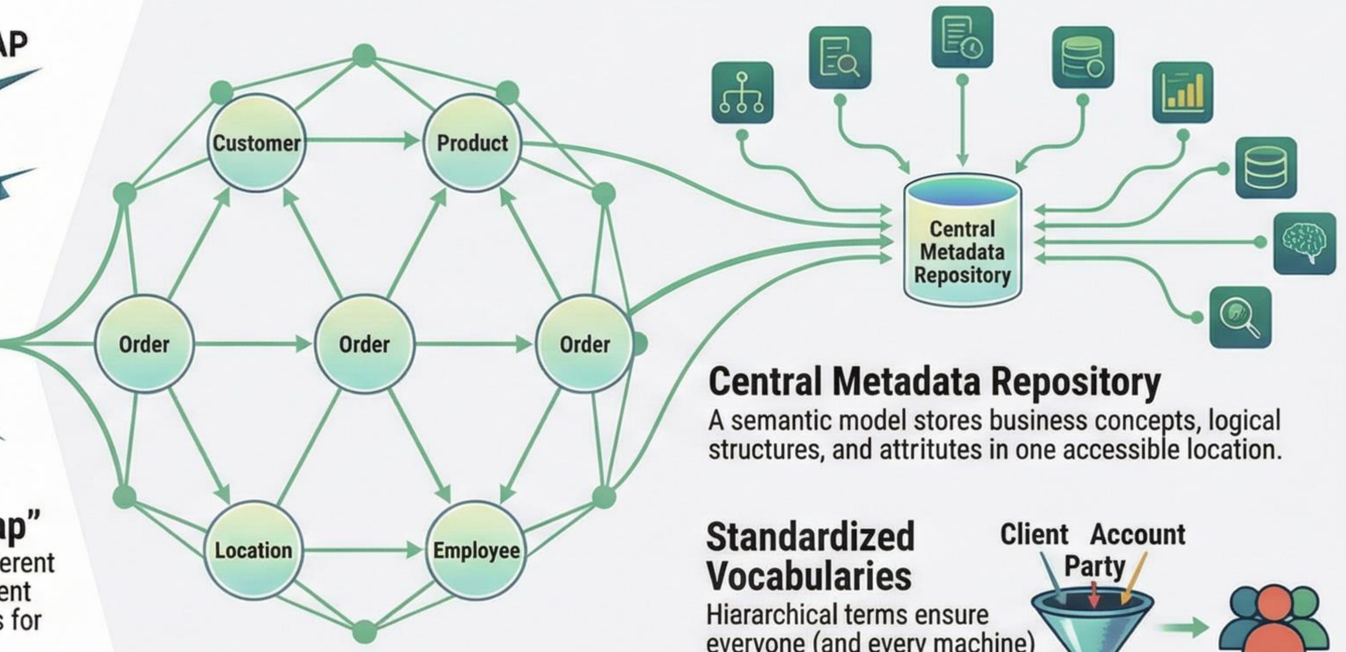
Fragmented Information Silos

Analysts must manually compile data from hundreds of disparate, isolated applications and systems.

The "Semantic Gap"

Confusion caused by different departments using different terminology and contexts for the same concepts.

The Solution: Knowledge Graphs & Semantic Models

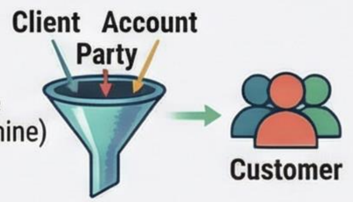


Central Metadata Repository

A semantic model stores business concepts, logical structures, and attritutes in one accessible location.

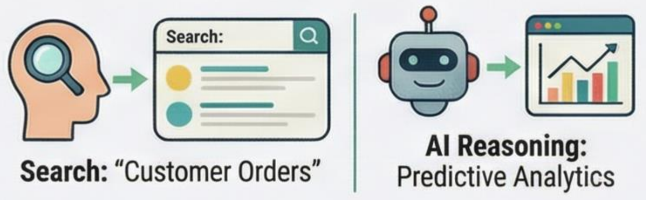
Standardized Vocabularies

Hiarchical terms ensure everyone (and every machine) uses the same language for data attributes.



Human and Machine Interpretable

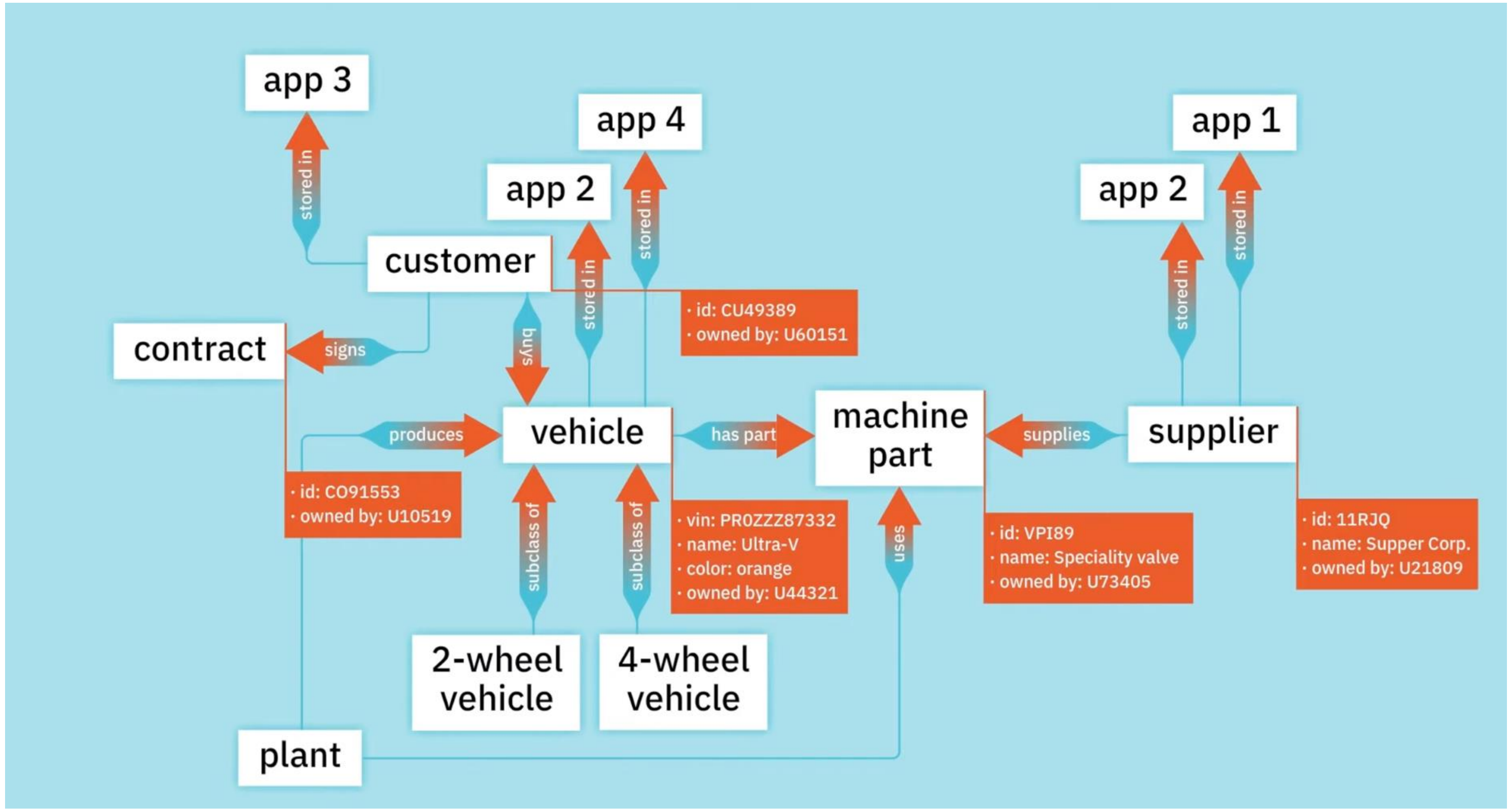
These models provide the foundation for AI reasoning and advanced and user search interfaces.

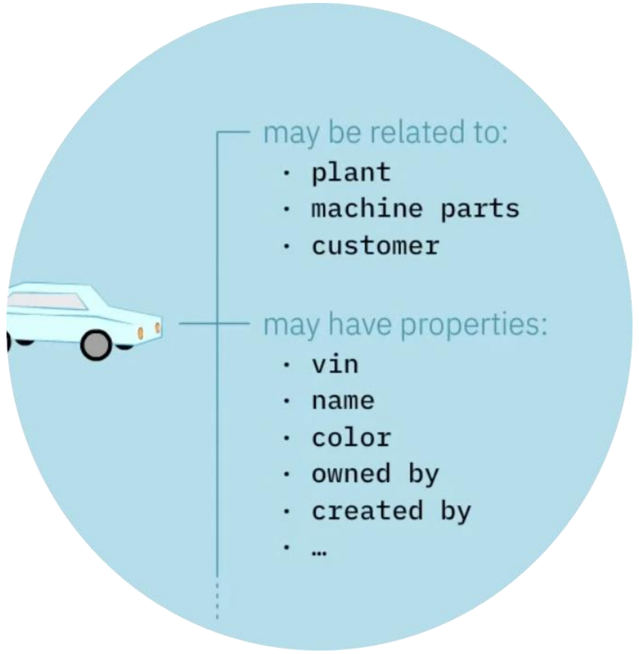


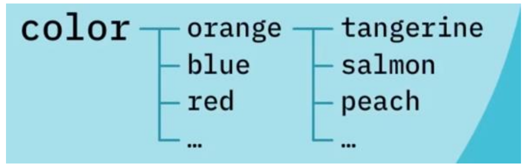
Feature Comparison: Before vs. After

Traditional Silos (Before)	Knowledge Graph (After)
Data Access: Manual & Fragmented	**Data Access:** Uniform & Centralized
Terminology: Ambiguous & Local	**Terminology:** Precise & Standardized
Searchability: Difficult/Unknown	**Searchability:** Precise & Efficient

Solving the Semantic Gap… Example







These properties may not make sense to all users and may vary between users. Example: Orange vs Peach vs Salmon And this certainly may not help a machine to understand. But by clearly categorizing that properties and their relations, once can begin building a common vocabulary.

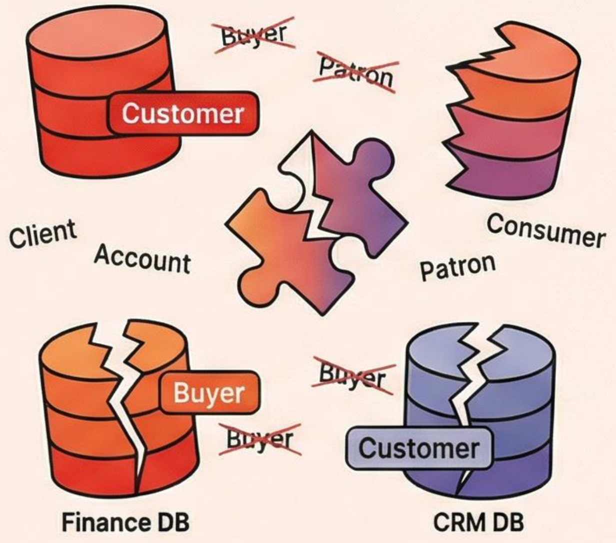
Common Vocabulary, Ontology & Metadata

*Technical Standards like **Semantic Modeling Language** (SML), **Resource Description Framework** (RDF) and **Web Ontology Language** (OWL) – help bridge the semantic gap today in organizations!*

Closing the Semantic Gap: Powering Enterprise AI with Knowledge Graphs

From Data Silos to Unified Understanding

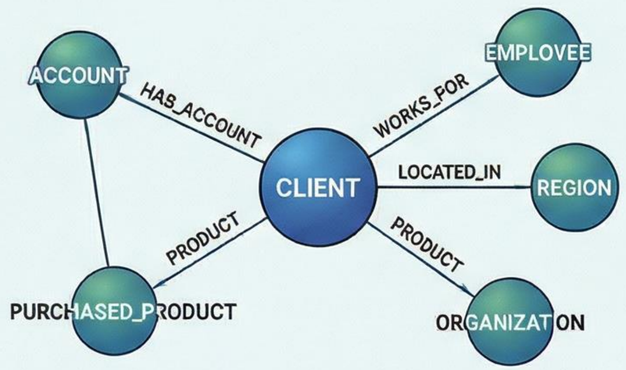
The Semantic Gap



The Semantic Gap

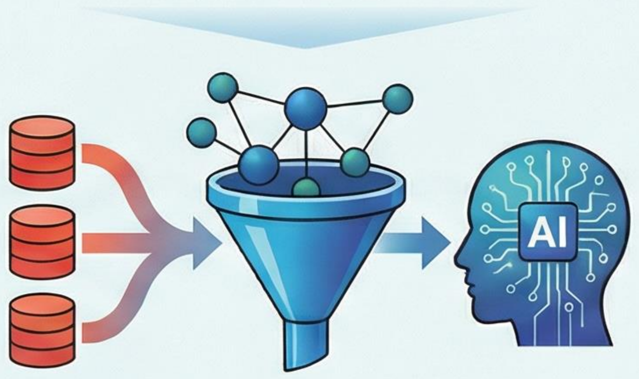
Inconsistent terminology (e.g., "Customer" vs. "Buyer") creates fragmented slios that limit data integration.

The Semantic Solution



The Semantic Solution

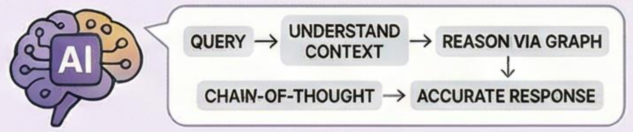
Standardized vocabularies and metadata harmonize terms into a unified Knowledge Graph using RDF/OWL models.



Bridging the Gap

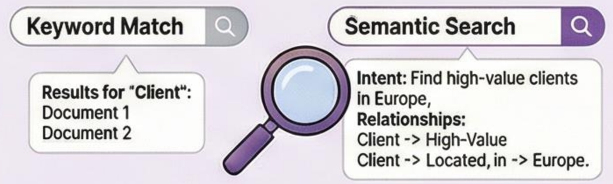
Mapping disparate data to a semantic model ensures consistent meaning across the entire organization.

Amplifying AI & Machine Learning Value



Structured AI Reasoning

Knowledge graphs provide the structured context required for agentic systems and complex chain-of-thought processing.

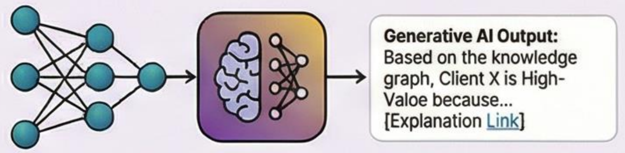


Results for "Client":
Document 1
Document 2

Intent: Find high-value clients in Europe,
Relationships:
Client -> High-Value
Client -> Located, in -> Europe.

Semantic Search Evolution

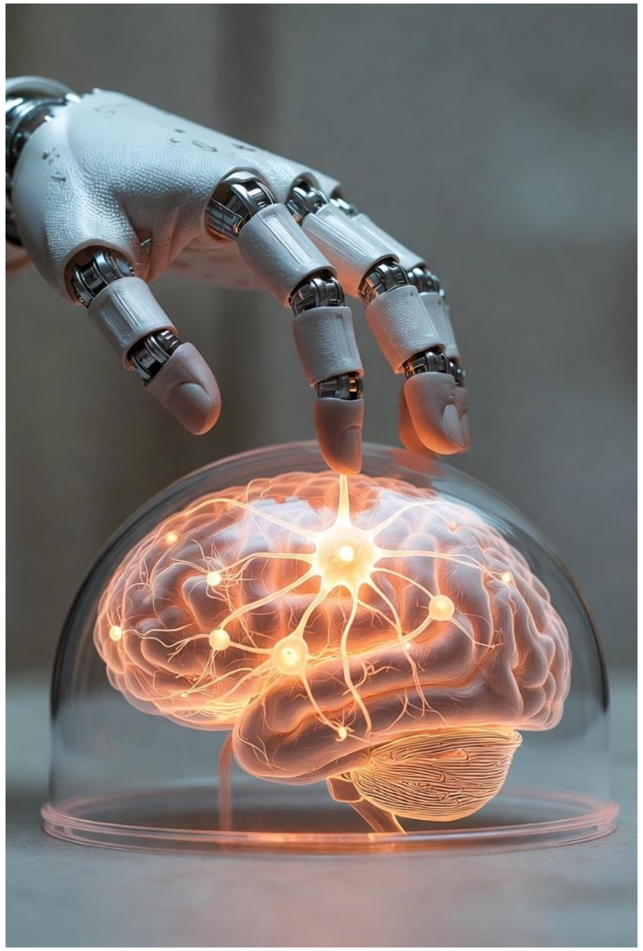
Systems transition from keyword matching to understanding intent and relationships between entities.



Generative AI Output:
Based on the knowledge graph, Client X is High-Valoe because...
[Explanation Link]

Graph RAG & GenAI

Creating a common platform for explainable machine learning and grounded generative AI outputs.



We now have a process to build a common searchable context!

Story #3

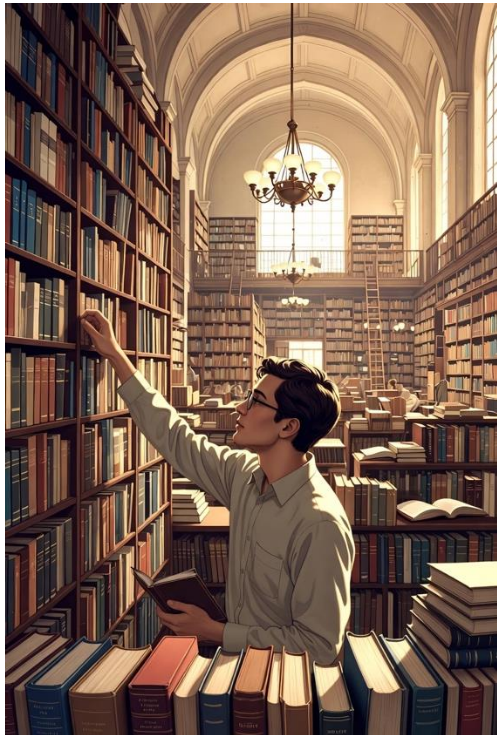
In Search of Uruk

Building A City of Knowledge

Uruk is famous as one of the world's first true cities, pioneering writing (cuneiform), monumental architecture (ziggurats), the cylinder seal, and advanced urban planning, and as the legendary home of King Gilgamesh in ancient Sumer (modern Iraq). It was the largest urban center globally around 3200 BCE, a hub for trade, governance, and cultural innovation that laid foundations for later civilizations.

YOUR TASK:

You possess an immense reservoir of digitized knowledge – how will you harness it to create something useful in 2025?

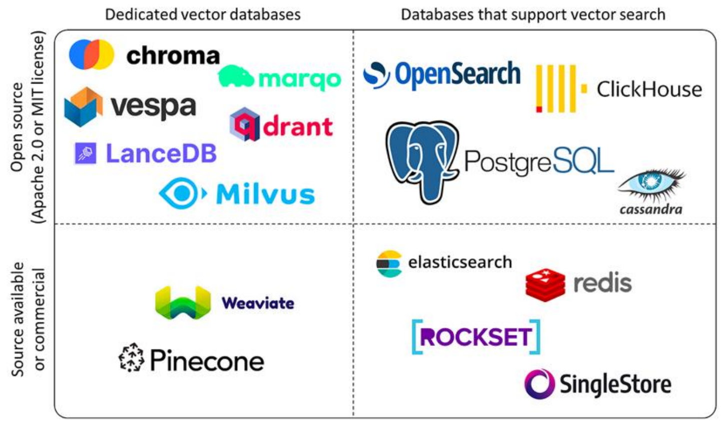


What will you build?

With Cloud, you now have infinite storage, infinite memory, infinite compute and infinite GPU at your resources! You can build tools that were only imaginable 20 years ago. The moment was ripe with opportunities!

GPU Compute made building vectors in real-time a new possibility! With that came a new generation of "semantic search", vector databases, generative AI & RAG (Retrieval-Augmented Generation)!

We moved into the *Age of the Artificial Intelligence*, *"Semantic Search", Cosine Similarity, Transformers.*



Dedicated vector databases
Databases that support vector search
Open source (Apache 2.0 or MIT license)
Source available or commercial
chroma
marqo
vespa
qdrant
LanceDB
Milvus
OpenSearch
ClickHouse
PostgreSQL
cassandra
Weaviate
Pinecone
elasticsearch
redis
ROCKSET
SingleStore

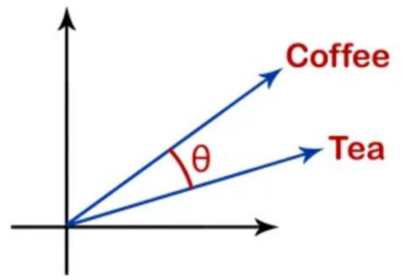
What exactly is a "Semantic Search"?
Semantic search is an AI-powered search method that goes beyond simple keyword matching to understand the **intent** and **context** of a user's query, delivering more relevant results by grasping the meaning behind the words, similar to how a human understands language. It uses **Natural Language Processing (NLP)** and **Machine Learning** to interpret queries in natural language, considering relationships between words, synonyms, location, and history, rather than just looking for exact keyword hits.

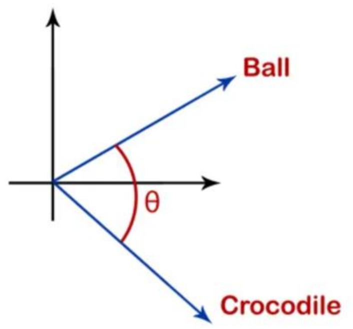
So how do you build this engine?

1. Vectors, Cosine Similarity, Euclidean Distance

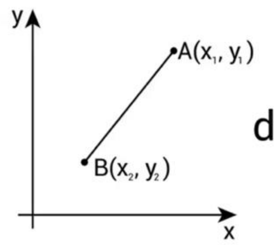
vectors are numerical representations of data that capture semantic meaning.

θ	cos(θ)
0°	1
60°	0.5
90°	0
120°	-0.5
180°	-1
270°	0
360°	1





1: High-Similarity; 0 or negative value: Low-Similarity



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Euclidean Distance

Term	What it does in this context
Vector Mapping	Converts text into numerical coordinates.
Cosine Similarity	Measures the **angle** between vectors (focuses on orientation/intent).
Euclidean Distance	Measures the **straight-line distance** between points (focuses on magnitude).

How is this used?
By representing words as vectors, we can use metrics like **Cosine Similarity** or **Euclidean Distance** to quantify how closely related two concepts are. This mathematical relationship forms the foundation of a **vector space model**, which is essential for training neural networks to understand data.

2. Character-level Embedding

H	e	l	l	o		w	o	r	l	d	!

Vocabulary: [' ', '!', 'd', 'e', 'H', 'l', 'o', 'r', 'w'] Size: 9

0	1	2	3	4	5	6	7	8
	e	o	d	H	l	w	r	!

H	e	l	l	o		w	o	r	l	d	!
4	1	5	5	2	0	6	2	7	5	3	8

Embedded Vector: [4,1,5,5,2,0,6,2,7,5,3,8]

3. Word-embedding, using “bag-of-words”

	I	like	the	new	movie	love	weather
I like the new movie	1	1	1	1	1	0	0
I love the weather	1	0	1	0	0	1	1
The movie like weather	0	1	1	0	1	0	1
I love the movie	1	0	1	0	1	1	0

In Vector Space

I	1	0	0	0	0	0	0
like	0	1	0	0	0	0	0
the	0	0	1	0	0	0	0
new	0	0	0	1	0	0	0
movie	0	0	0	0	1	0	0
love	0	0	0	0	0	1	0
weather	0	0	0	0	0	0	1

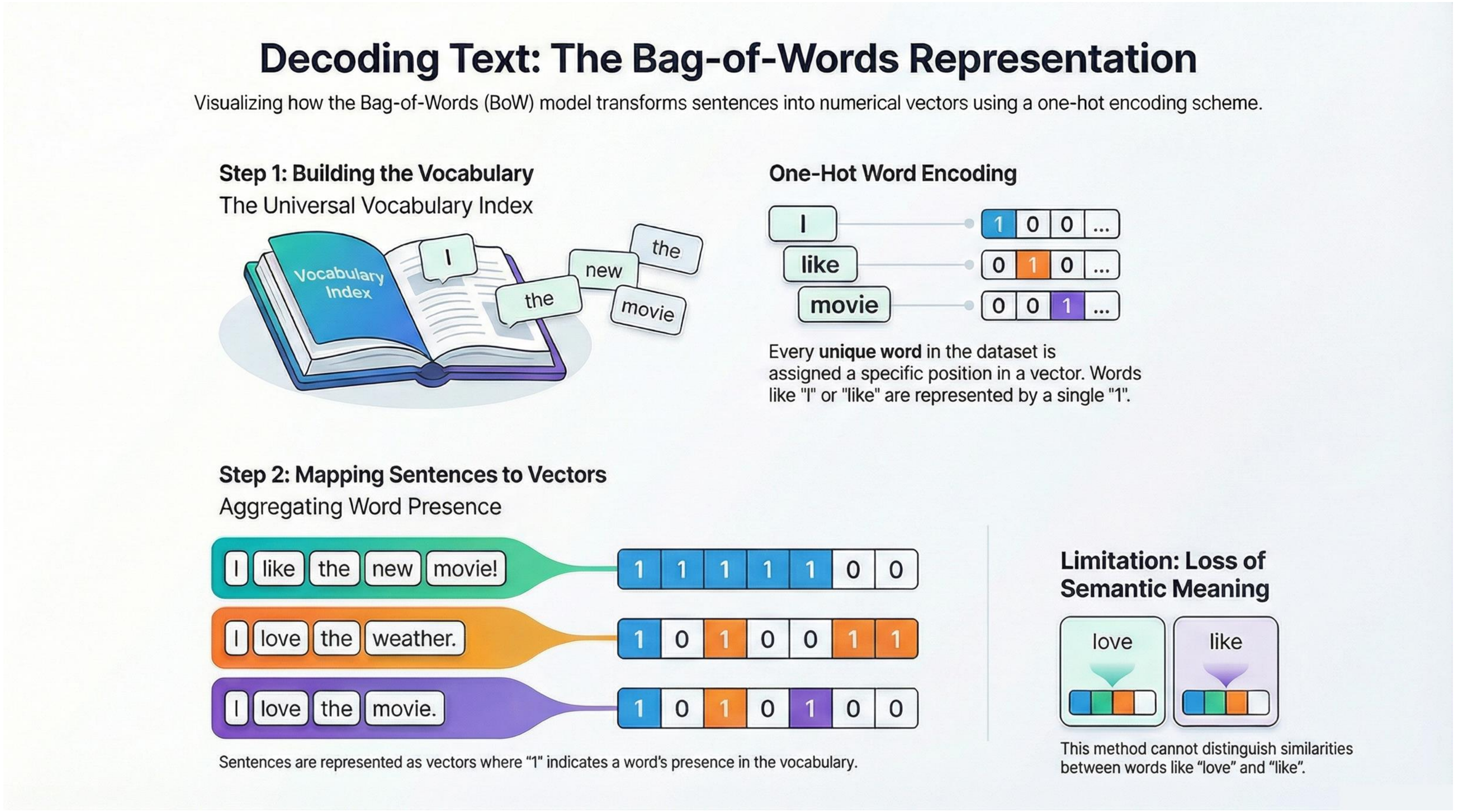
then a sentence will be represented as:

"I like the new movie" [1,1,1,1,1,0,0]

In both cases, with character embedding and word embedding using a “bag of words” is NOT showing any semantic or syntactic relationships.
Can we do better?
Solution: **Word2Vec-embedding**

Embeddings: A way to represent “meaning”.

Think of it as GPS, instead of navigating through streets it navigates through meaning!



Decoding Text: The Bag-of-Words Representation

Visualizing how the Bag-of-Words (BoW) model transforms sentences into numerical vectors using a one-hot encoding scheme.

Step 1: Building the Vocabulary

The Universal Vocabulary Index

One-Hot Word Encoding

Every **unique word** in the dataset is assigned a specific position in a vector. Words like "I" or "like" are represented by a single "1".

Step 2: Mapping Sentences to Vectors

Aggregating Word Presence

I like the new movie! → 1 1 1 1 1 0 0
I love the weather. → 1 0 1 0 0 1 1
I love the movie. → 1 0 1 0 1 0 0

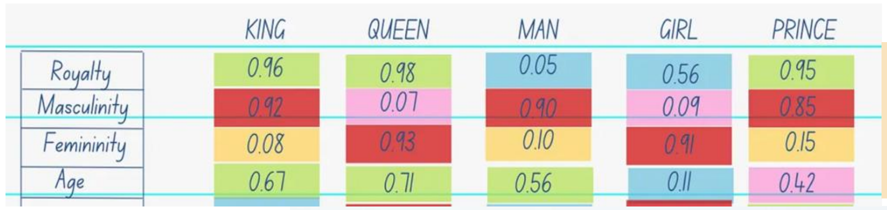
Sentences are represented as vectors where “1” indicates a word’s presence in the vocabulary.

Limitation: Loss of Semantic Meaning

This method cannot distinguish similarities between words like “love” and “like”.

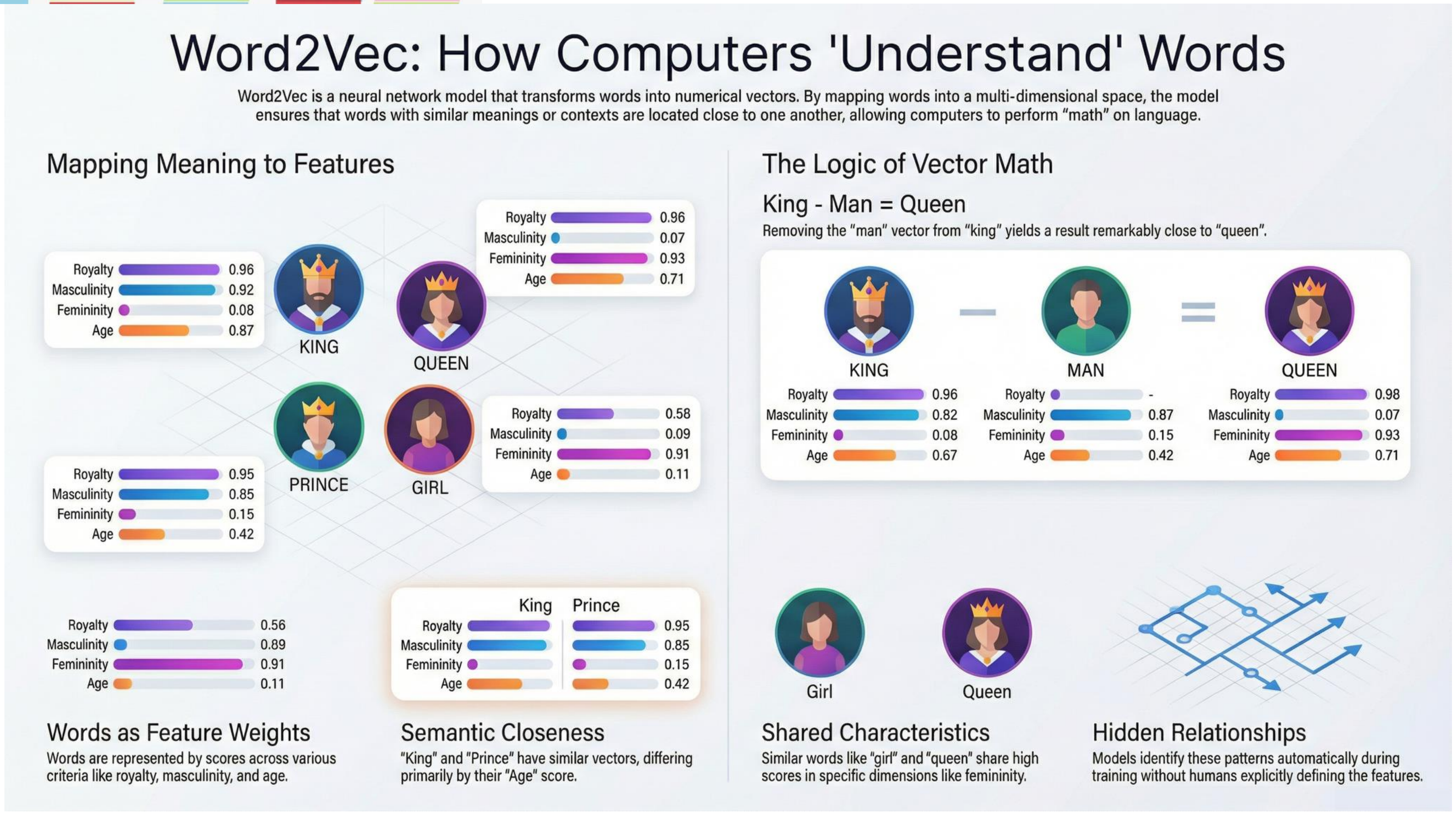
Embeddings evolved with Word2Vec

https://wikipedia2vec.github.io/demo/



	KING	QUEEN	MAN	GIRL	PRINCE
Royalty	0.96	0.98	0.05	0.56	0.95
Masculinity	0.92	0.07	0.90	0.09	0.85
Femininity	0.08	0.93	0.10	0.91	0.15
Age	0.67	0.71	0.56	0.11	0.42

Word2Vec creates a representation of each word present in our vocabulary into a vector. Words used in similar contexts or having semantic relationships are captured effectively through their closeness in the vector space- effectively speaking similar words will have similar word vectors! ***History***: Word2vec was created, patented, and published in 2013 by a team of researchers led by Tomas Mikolov at Google.



Word2Vec: How Computers 'Understand' Words

Word2Vec is a neural network model that transforms words into numerical vectors. By mapping words into a multi-dimensional space, the model ensures that words with similar meanings or contexts are located close to one another, allowing computers to perform "math" on language.

Mapping Meaning to Features

	KING	QUEEN	PRINCE	GIRL
Royalty	0.96	0.96	0.95	0.58
Masculinity	0.92	0.07	0.85	0.09
Femininity	0.08	0.93	0.15	0.91
Age	0.87	0.71	0.42	0.11

The Logic of Vector Math

King - Man = Queen

Removing the "man" vector from "king" yields a result remarkably close to "queen".

	KING	MAN	QUEEN
Royalty	0.96	-	0.98
Masculinity	0.82	0.87	0.07
Femininity	0.08	0.15	0.93
Age	0.67	0.42	0.71

Royalty	0.56
Masculinity	0.89
Femininity	0.91
Age	0.11

Words as Feature Weights

Words are represented by scores across various criteria like royalty, masculinity, and age.

	King	Prince
Royalty		0.95
Masculinity		0.85
Femininity		0.15
Age		0.42

Semantic Closeness

"King" and "Prince" have similar vectors, differing primarily by their "Age" score.

Shared Characteristics

Similar words like "girl" and "queen" share high scores in specific dimensions like femininity.

Hidden Relationships

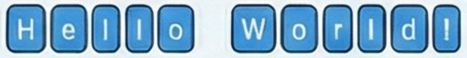
Models identify these patterns automatically during training without humans explicitly defining the features.

The Path to the Vector Database: Character vs. Word Tokenization

Row 1: Character-Level Tokenization (Naive Approach)

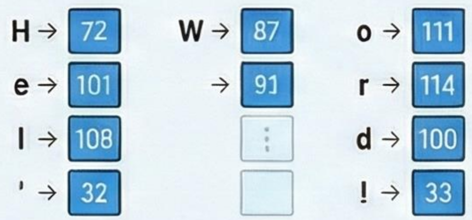
Input: "Hello World!"

Step 1: Character Splitting



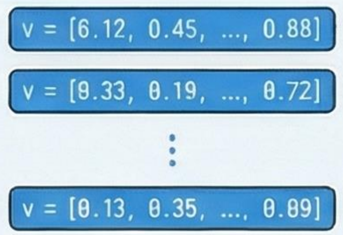
The input sentence is broken down into every individual character, including spaces and punctuation.

Step 2: Integer Mapping (Small Vocab)



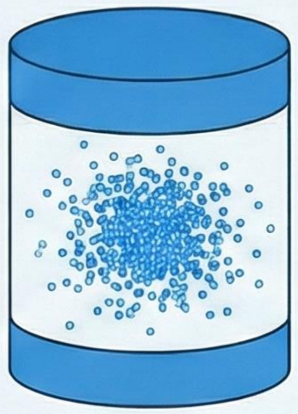
Each character is assigned a number based on a small, simple vocabulary (e.g., 65-256 IDs).

Step 3: Vector Transformation & Storage



v = [6.12, 0.45, ..., 0.88]
v = [0.33, 0.19, ..., 0.72]
v = [0.13, 0.35, ..., 0.89]

Integers are mapped to vectors in an embedding table and stored in the database.



Vector Database

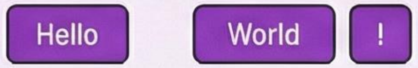
No Semantic Relationships

	Character-Level	Word/Subword-Level (GPT-4)
Token Count	12 Tokens	3 Tokens
Vocab Size	Very Small (~236)	Very Large (~100,000)
Efficiency	Long sequences, slow processing	Short sequences, high context density

Row 2: Word/Subword-Level Tokenization (State-of-the-Art)

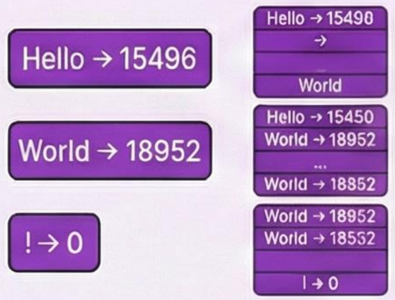
Input: "Hello World!"

Step 1: Chunking (BPE)



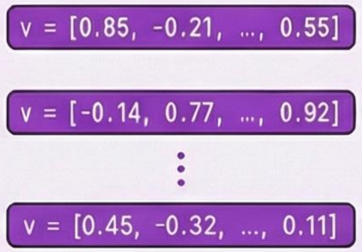
Algorithms like Byte Pair Encoding (BPE) group common character clusters into "chunks" or subwords.

Step 2: Integer Mapping (Large Vocab)



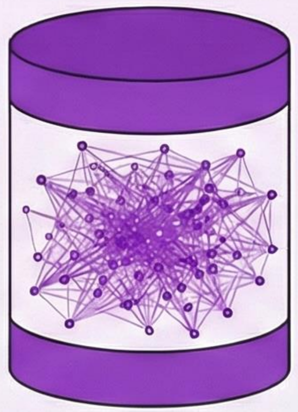
Chunks map to a massive vocabulary (e.g.. 100k+ IDs), resulting in fewer, denser tokens.

Step 3: Semantic Vector Storage



v = [0.85, -0.21, ..., 0.55]
v = [-0.14, 0.77, ..., 0.92]
v = [0.45, -0.32, ..., 0.11]

These dense tokens are converted into vectors that capture complex relationships for efficient retrieval.



Vector Database

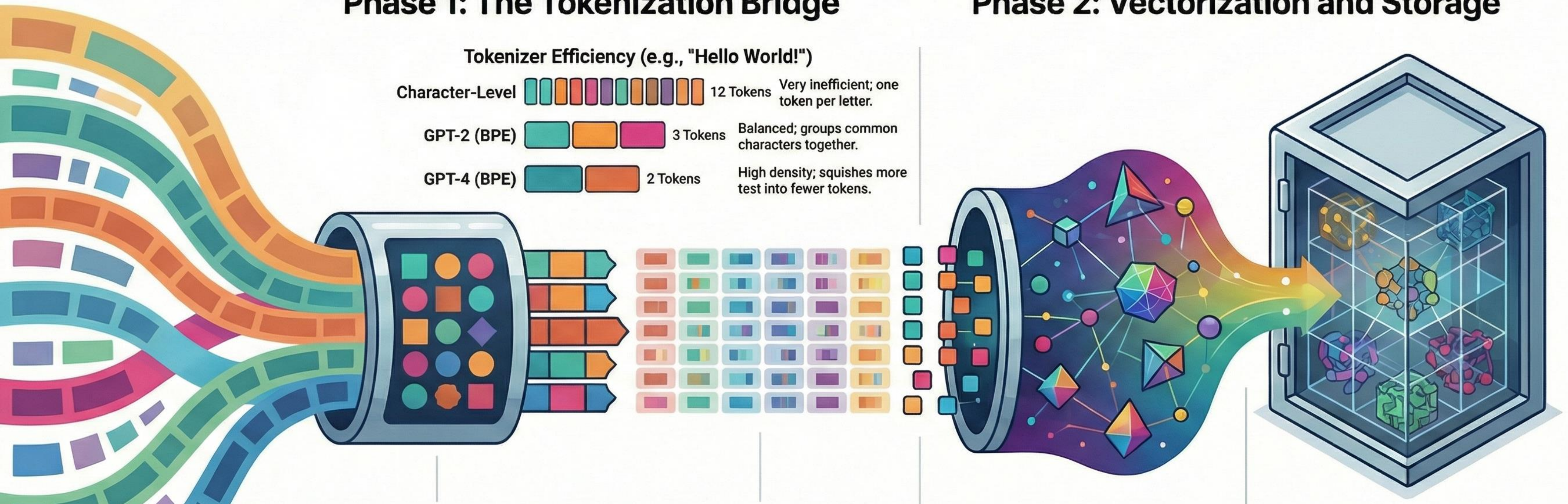
From Words to Vectors: The LLM Data Pipeline

Phase 1: The Tokenization Bridge

Phase 2: Vectorization and Storage

Tokenizer Efficiency (e.g., "Hello World!")

Character-Level	12 Tokens	Very inefficient; one token per letter.
GPT-2 (BPE)	3 Tokens	Balanced; groups common characters together.
GPT-4 (BPE)	2 Tokens	High density; squishes more test into fewer tokens.



Subword Chunking via BPE

Text is split into chunks using algorithms like Byte Pair Encoding for efficient processing.

The Integer Lookup

Every unique chunk is assigned a specific number (ID) from the model's vocabulary.

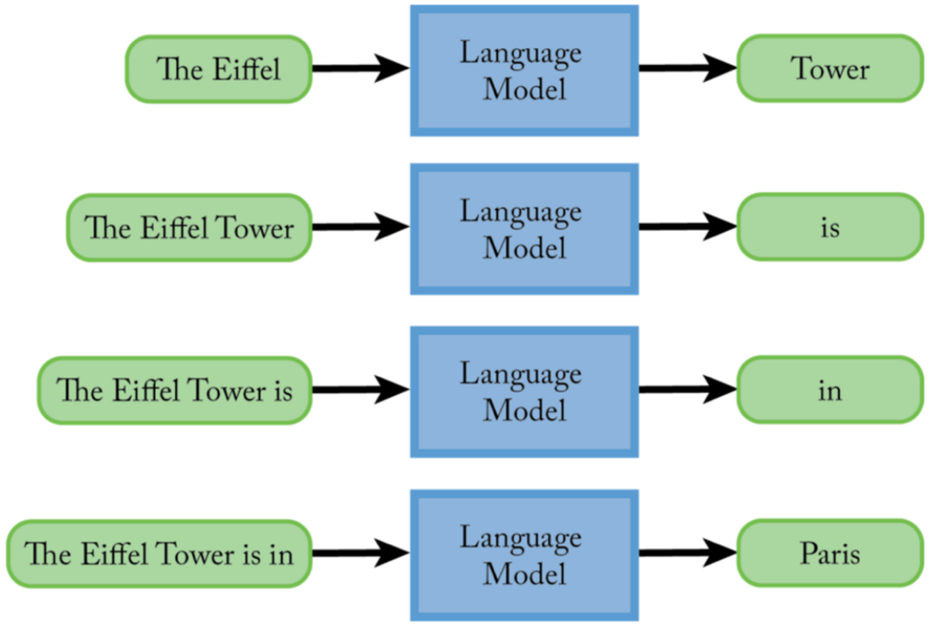
The Embedding Transformation

Token integers are converted into high-dimensional vectors that represent the meaning of the text.

Persistent Vector Database Storage

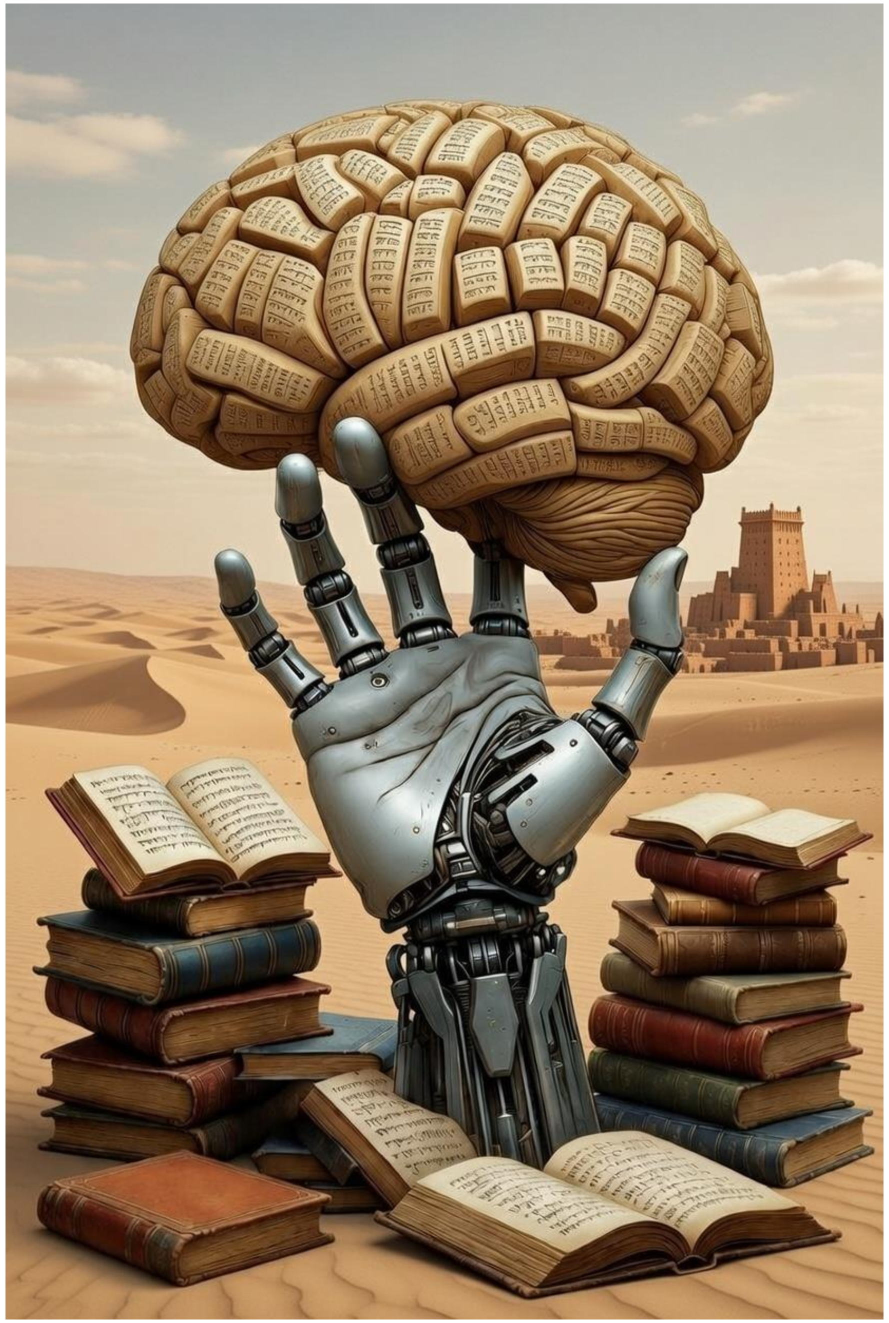
These vectors are stored in a database, allowing the AI to retrieve relevant data.

Reversing the flow using a Large-Language Model for Generative-Context



This is how a Large Language Model is leveraged to generate the content!

The next step in this evolutionary process is **reasoning...**



We now have a process to build a generative solution using a Large-Language Model

Today, we explored the evolution of AI, starting with the architecture of **artificial neurons** modeled after human decision-making. We then examined how **semantic context** and relationships provide data with deeper meaning. Finally, we integrated these concepts to see how **vectorized engines** power the **Large Language Models (LLMs)** that are currently transforming our daily lives.

Q
A