

Coursework Plan

group project (70%): 2 people per group

implement an ML system (template provided), write a **report**, group **presentation**

purpose: demonstrate critical thinking in system goal design, evaluation and optimisation

individual review (30%)

review an assigned project report, write a **review**, individual **presentation**

purpose: demonstrate independent, critical and constructive thinking

Group project template and example topics will be released in Week 4

Data Flow in ML Systems: embeddings and retrieval

Week2 — Machine Learning Systems (Fall 2026)

Instructor: Yang Cao

Learning points

0. Why do we talk about data in MLS?

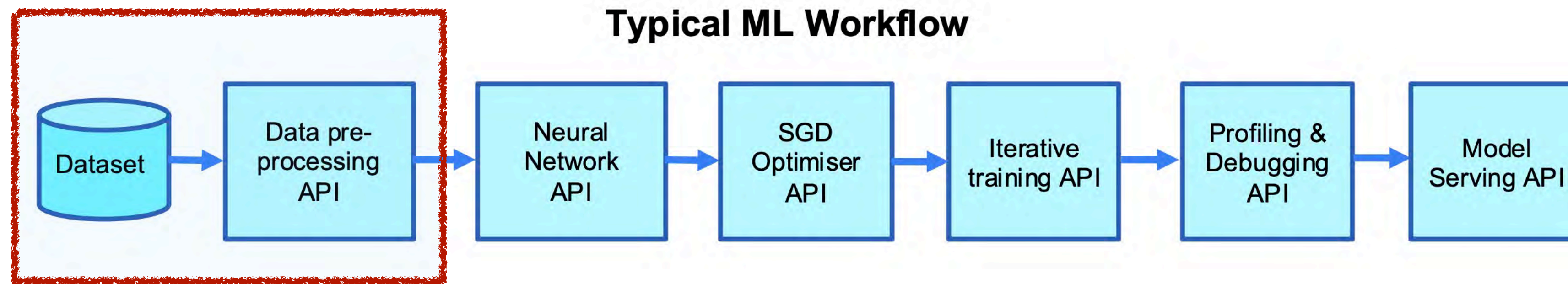
1. How does data flow into, through and out of ML models?

2. Principles for selecting and providing data for ML

3*. identifying “deeper” questions from practice

Why do we talk about data in *MLS*?

Data for training

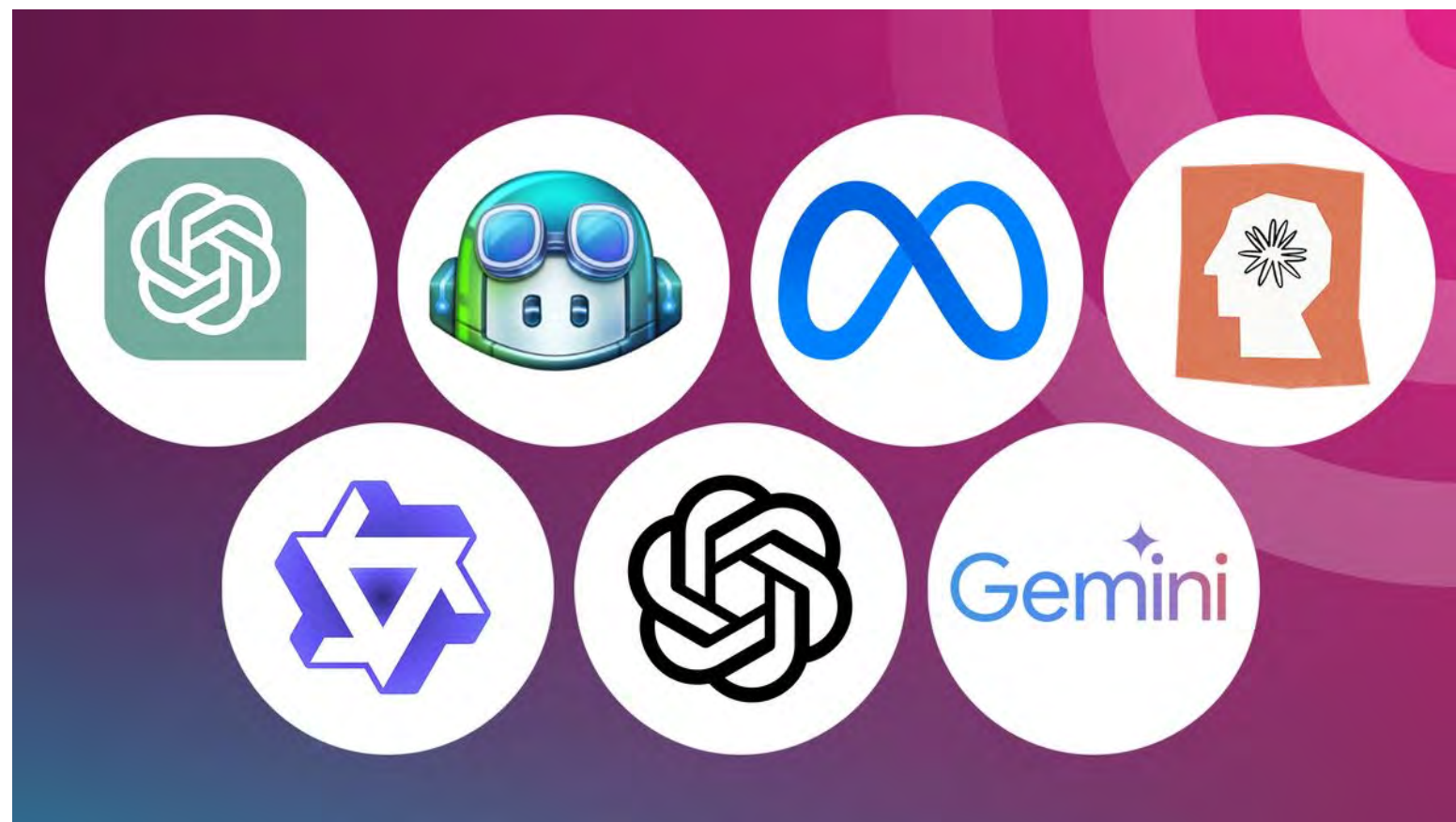


Data

folklore axiom among data scientists: [Crowdfunder Survey (2016)]

80% time for data wrangling, 20% time for data modelling and machine learning

“garbage in, garbage out” — the cliché in ML circles



LLMs in the 1600s (Image credit: Future)



Galileo before the Holy Office (Image credit: Wikipedia)

Could we save Galileo if there were LLMs in 1600s*?

[nature](#) > [articles](#) > [article](#)

Article | [Open access](#) | Published: 25 September 2024

Larger and more instructable language models become less reliable

[Lexin Zhou](#), [Wout Schellaert](#), [Fernando Martínez-Plumed](#), [Yael Moros-Daval](#), [Cèsar Ferri](#) & [José Hernández-Orallo](#) 

[Nature](#) **634**, 61–68 (2024) | [Cite this article](#)

120k Accesses | **147** Citations | **603** Altmetric | [Metrics](#)

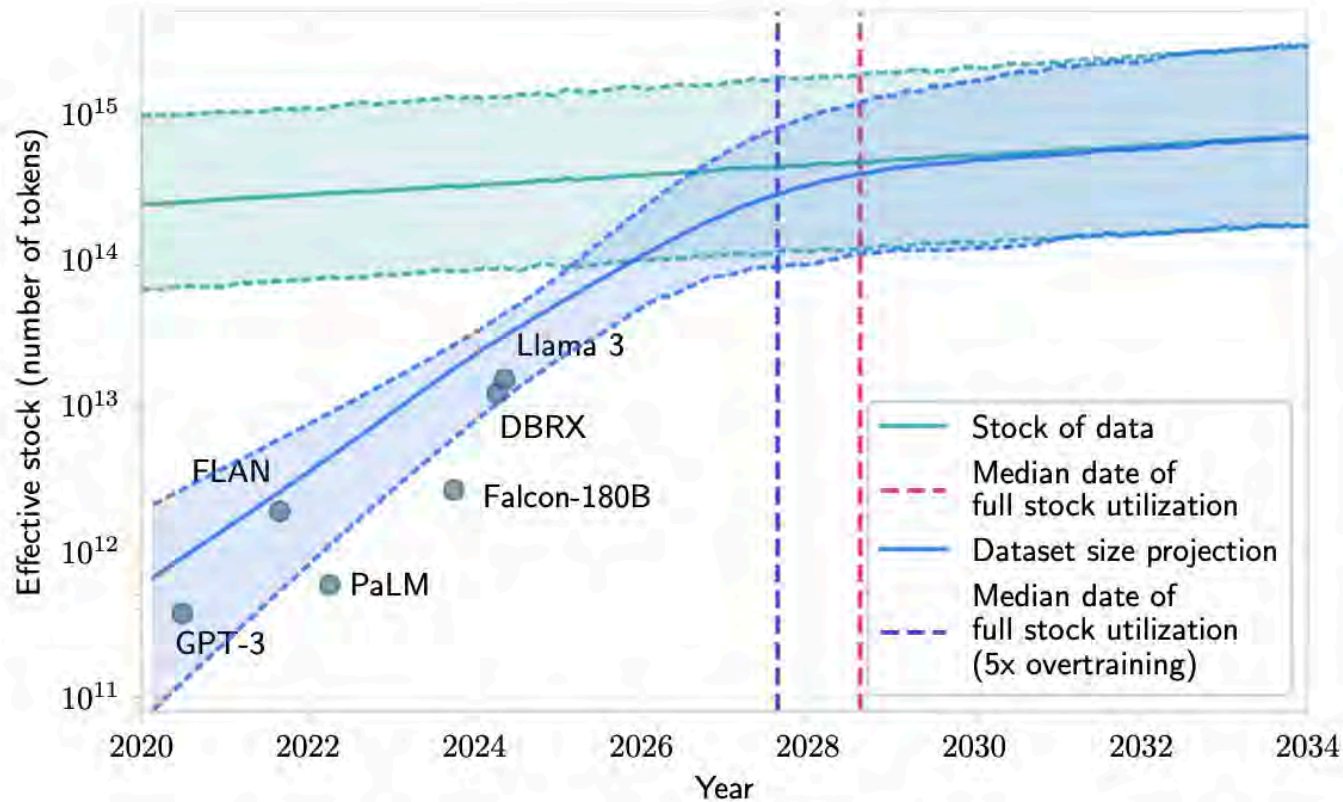
Abstract

Will we run out of data? Limits of LLM scaling based on human-generated data

Pablo Villalobos¹ **Anson Ho¹** **Jaime Sevilla^{1,2}** **Tamay Besiroglu^{1,3}** **Lennart Heim^{1,4}** **Marius Hobbhahn^{1,5}**

Abstract

We investigate the potential constraints on LLM scaling posed by the availability of public human-generated text data. We forecast the growing demand for training data based on current trends and estimate the total stock of public human text data. Our findings indicate that if current LLM development trends continue, models will be trained on datasets roughly equal in size to the available stock of public human text data between 2026 and 2032, or slightly earlier if models are overtrained. We explore how progress in language modeling



Article | [Open access](#) | Published: 25 September 2024

Larger and more instructable language models become less reliable

[Lexin Zhou](#), [Wout Schellaert](#), [Fernando Martínez-Plumed](#), [Yael Moros-Daval](#), [Cèsar Ferri](#) & [José Hernández-Orallo](#) 

[Nature](#) **634**, 61–68 (2024) | [Cite this article](#)

120k Accesses | **147** Citations | **603** Altmetric | [Metrics](#)

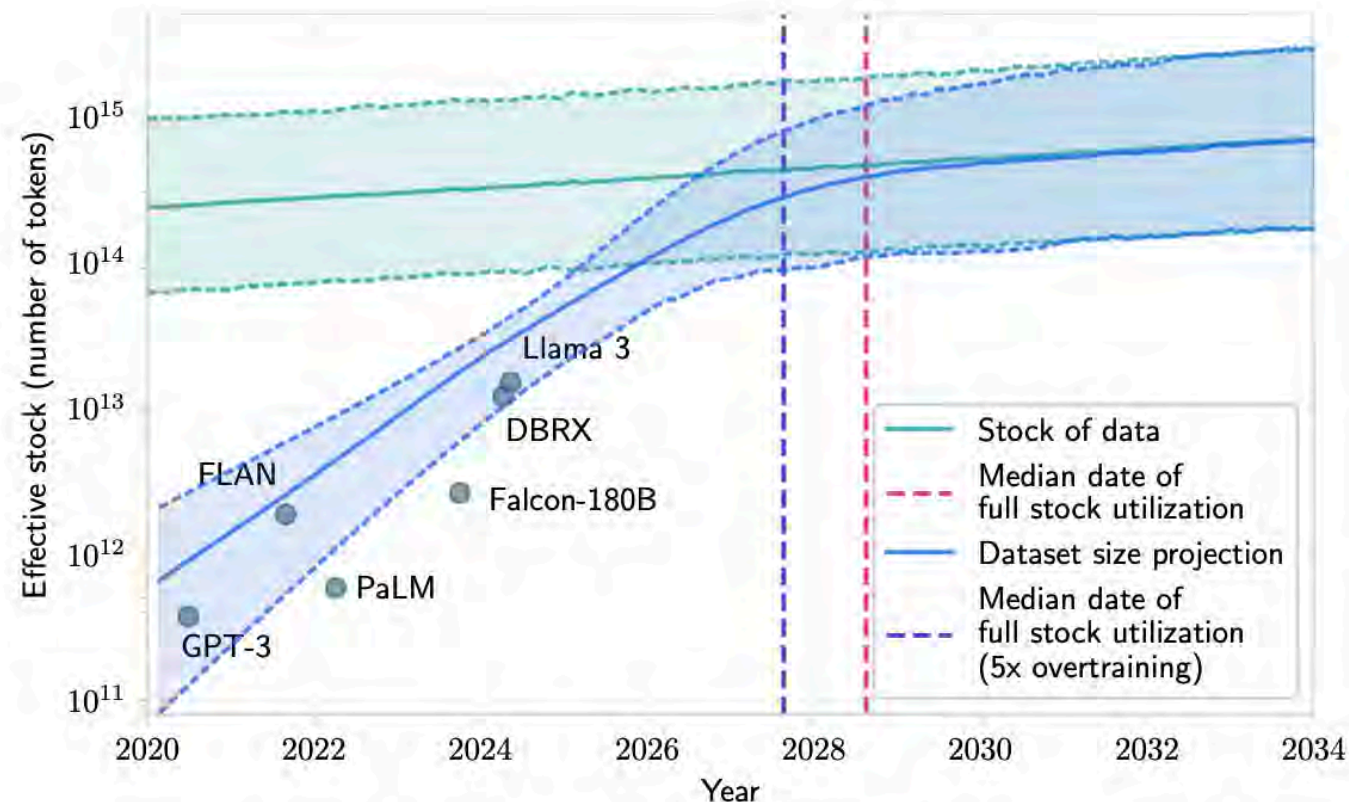
Abstract

Will we run out of data? Limits of LLM scaling based on human-generated data

Pablo Villalobos¹ **Anson Ho¹** **Jaime Sevilla^{1,2}** **Tamay Besiroglu^{1,3}** **Lennart Heim^{1,4}** **Marius Hobbhahn^{1,5}**

Abstract

We investigate the potential constraints on LLM scaling posed by the availability of public human-generated text data. We forecast the growing demand for training data based on current trends and estimate the total stock of public human text data. Our findings indicate that if current LLM development trends continue, models will be trained on datasets roughly equal in size to the available stock of public human text data between 2026 and 2032, or slightly earlier if models are overtrained. We explore how progress in language modeling



Revisiting Scaling Laws for Language Models: The Role of Data Quality and Training Strategies

Zhengyu Chen^{1*}, **Siqi Wang^{2*}**, **Teng Xiao³**, **Yudong Wang⁴**,
Shiqi Chen⁵, **Xunliang Cai¹**, **Junxian He⁵**, **Jingang Wang¹**

¹ Meituan Inc. ² The University of Hong Kong ³ Pennsylvania State University
⁴ Peking University ⁵ The Hong Kong University of Science and Technology
{chenzhengyu04, wangjingang02}@meituan.com

Textbooks Are All You Need

Suriya Gunasekar Yi Zhang Jyoti Aneja Caio César Teodoro Mendes
Allie Del Giorno Sivakanth Gopi Mojan Javaheripi Piero Kauffmann
Gustavo de Rosa Olli Saarikivi Adil Salim Shital Shah Harkirat Singh Behl
Xin Wang Sébastien Bubeck Ronen Eldan Adam Tauman Kalai Yin Tat Lee
Yuanzhi Li

Microsoft Research

Abstract

We introduce **phi-1**, a new large language model for code, with significantly smaller size than competing models: **phi-1** is a Transformer-based model with 1.3B parameters, trained for 4 days on 8 A100s, using a selection of “textbook quality” data from the web (6B tokens) and synthetically generated textbooks and exercises with GPT-3.5 (1B tokens). Despite this small scale, **phi-1** attains **pass@1** accuracy 50.6% on HumanEval and 55.5% on MBPP. It also displays surprising emergent properties compared to **phi-1-base**, our model *before* our finetuning stage on a dataset of coding exercises, and **phi-1-small**, a smaller model with 350M parameters trained with the same pipeline as **phi-1** that still achieves 45% on HumanEval.

Data is key to the continous improvement of LLMs

Why LLMs keep getting stronger with the same architecture?

AI Mode All Short videos Videos Images Forums Shopping More

AI Overview

Large Language Models (LLMs) continue to get significantly stronger using the same **architecture** (introduced in 2017) primarily because of **better data quality, better training, and better compute innovations**.

While the fundamental math behind how these models process language has remained the same, researchers have figured out that the architecture itself wasn't the bottleneck—how we train and curate data is.

The Core Drivers of Growth

1. Data Quality and Curation (The "Garbage In, Gold Out" Shift)

Early LLMs were trained on raw, unrefined scrapes of the internet. Today, the biggest performance leaps come from meticulously cleaning that data.

chatgpt.com

LLMs keep improving because we haven't yet exhausted the internet. The next gains have to come from better data, more compute, better training, and more innovations.

And arguably, the biggest improvements since roughly GPT-3 came from **better data, more compute, better training, and more innovations**—not just how much data, but **fundamental architecture and more from changing how training and inference are done**.

If I had to pick one, I'd say **effective training compute**—meaning, more compute applied with the right model size, data quantity, and training recipe.

The reason is that compute acts as the main enabler. More compute lets you train larger models, expose them to more data, train for longer, generate synthetic training examples, run reinforcement learning, and spend more computation on reasoning at inference time. But compute by itself is not enough: badly chosen or low-quality data can waste enormous amounts of it.

So I'd rank the factors roughly like this:

1. Training compute + correct scaling of model/data

2. Data quality and diversity

3. Post-training / reasoning training

4. Inference-time compute

5. Architectural refinements

Data quality — specifically, the shift from *more data* to *better data*.

Here's why it's pulling ahead of everything else right now:

We're hitting the internet ceiling

Raw text from the web is nearly exhausted as a training source. Models have essentially "read the internet." Adding more low-quality text gives diminishing returns — and can even hurt performance. The next gains have to come from *what kind* of data you train on, not just how much.

The synthetic data breakthrough

This is the real frontier. Instead of scraping human text, labs now:

- Use strong models to **generate reasoning traces** — step-by-step solutions to hard problems
- **Filter aggressively** for correctness, keeping only chains of thought that lead to right answers
- Bootstrap iteratively — a better model generates better training data for the *next* model

This is how models like DeepSeek-R1 and OpenAI's o-series were pushed into genuine reasoning territory. The data wasn't found — it was *constructed*.

Why quality beats quantity now

A model trained on 100B tokens of carefully curated data (like reasoning traces) outperforms one trained on 10T tokens of random web text on hard benchmarks. The signal-to-noise ratio of training data matters more than ever.

Enterprise use cases — the “Data Problem” of AI (2025)

Harvard Business Review (Aug 29, 2025)^[1]

MIT Media Lab/Project NANDA's finding that **95%** of investments in Gen AI have produced **zero returns**.

Individuals benefit vs. companies struggle with enterprise-wide deployments

Most spending goes to sales and marketing initiatives, despite

back-end transformations = ROI



Source: <https://docs.databricks.com/aws/en/generative-ai/guide/concepts/>

Key reason (by Databricks^[2], and many others)
disconnected from business data

[1] Nathan Furr, Andrew Shipilov: Beware the AI Experimentation Trap. In Harvard Business Review, Aug 29, 2025

[2] Databricks: AI Agents Fundamentals

Enterprise use cases — the “Data Problem” of AI (2025)

Gartner predicts that through 2026, organizations will abandon 60% of AI projects unsupported by AI-ready data.

Gartner

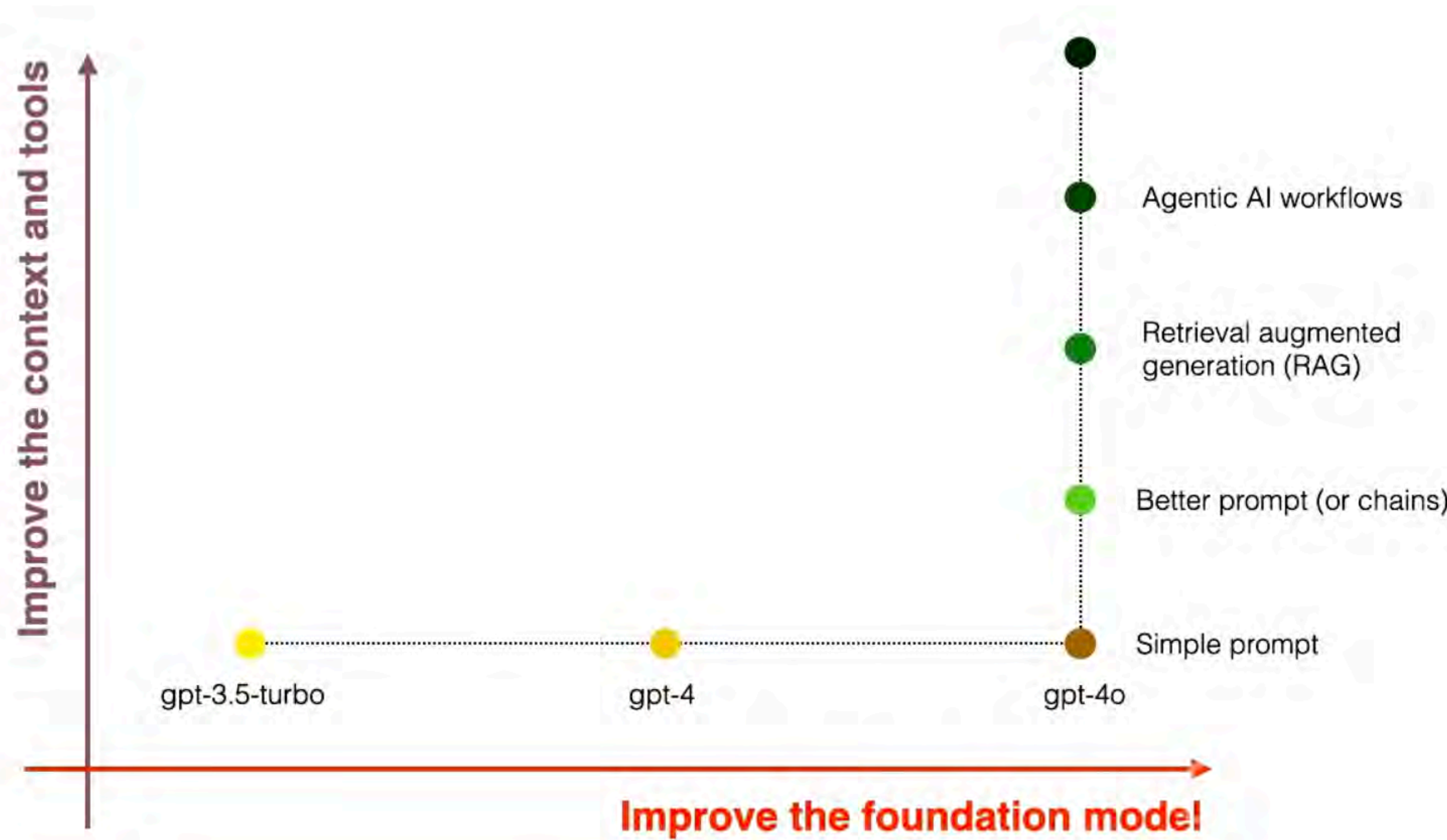
Lack of AI-Ready Data Puts AI Projects at Risk

Poor data blocks AI decisions for 69% of companies (Year 2025)^[1]

- 69% of companies say poor data limits their ability to make informed decisions.
- 45% report that unstructured, fragmented data is the biggest barrier to AI success.
- Only 8.6% of companies are fully AI-ready.

[1] Hubble AI Data Readiness Research Report (2025) <https://hubble.com/blog/ai-hidden-data-crisis#:~:text=While businesses are eager to,costly inefficiencies and missed opportunities.>

Not just training — two dimensions to enhance your LLM



Credit: Kian Katanforoosh

So how does data enter ML systems?

Not every task needs learning

Customers:

customer_id	credit_limit_gbp
C1	600
C2	500

Payments:

payment_id	customer_id	invoice_id	paid_amount_gbp
P1	C1	I4	100
P2	C1	I4	50

Invoices:

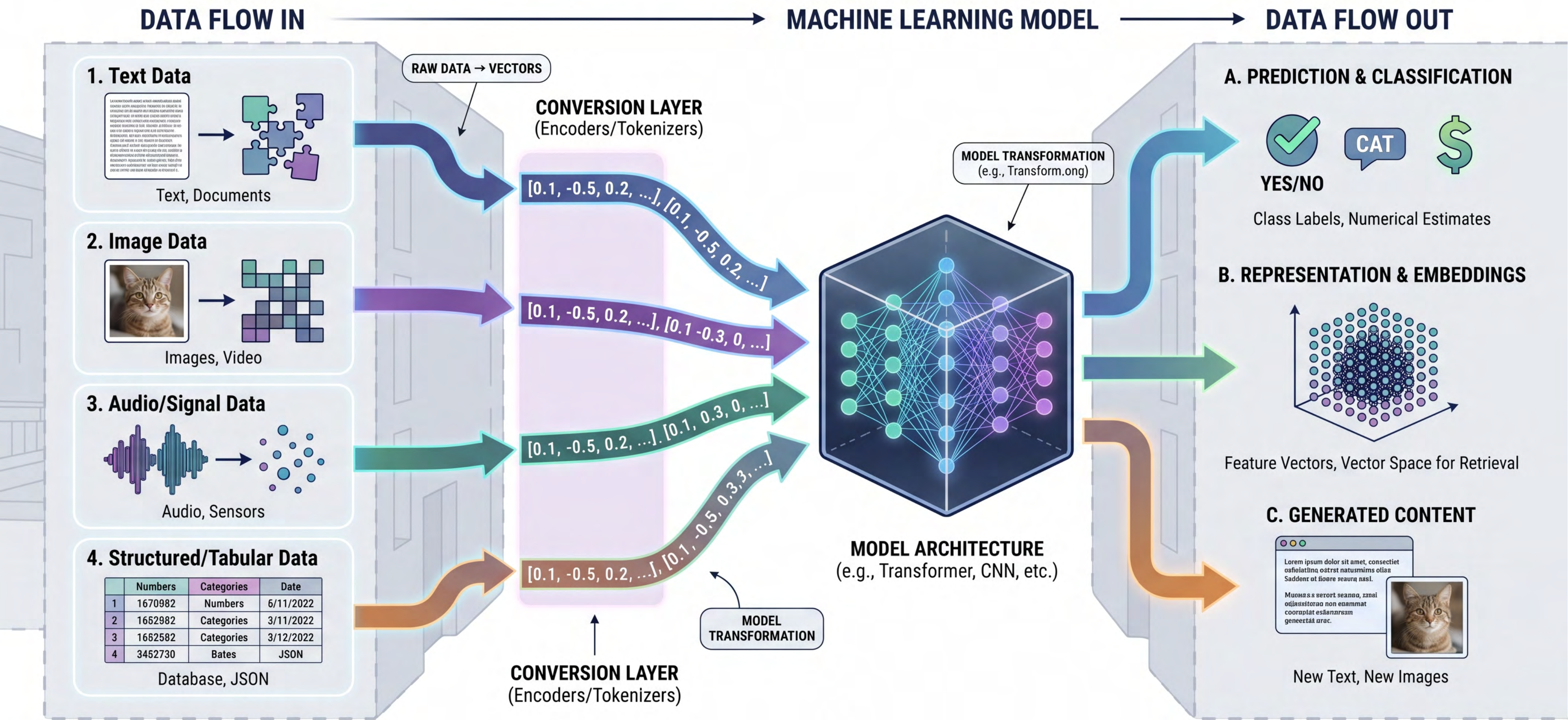
invoice_id	customer_id	amount_gbp	paid_late (observed?)
I1	C1	600	no
I2	C1	100	no
I3	C2	600	yes
I4	C1	800	unknown

Which requires ML?

1. Show me invoice I4 balance
2. Will I4 pay late?
3. *"Show me invoice I4 balance, please"*

Not every task needs learning and even it does tasks can be rather different depending on the data and task definition

Data → model → output

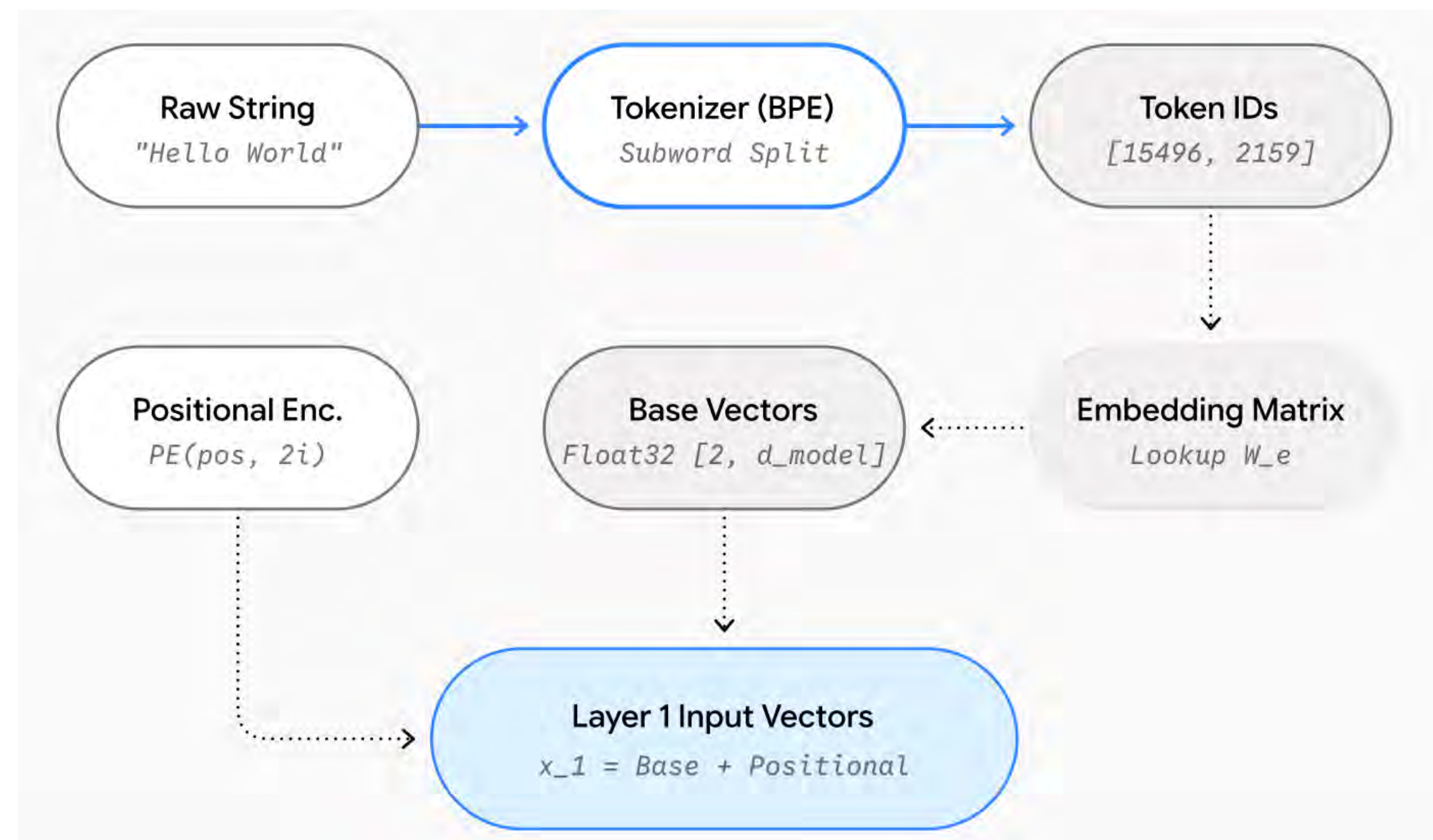


Vectors (embeddings)

How do models see data?

Text (generative tasks)

tokenization (e.g., Byte-Pair Encoding (BPE)), token embedding, positional info, into LM



*generated by Gemini

Related course:

- Foundations of Natural Language Processing (FNLP)
- Accelerated Natural Language Processing (ANLP)

How do models see data?

Images (classification, object detection ... tasks)

raw pixel grids $\rightarrow$ dense numerical vectors



*generated by Gemini

Related course:

- Machine Learning Practical (MLP)
- Applied Machine Learning (AML)

How do models see data?

Will I4 pay late?

Invoices:

invoice_id	customer_id	credit_limit_gbp	amount_gbp	Comment	paid_late (observed?)
I1	C1	600	600	...	no
I2	C1	600	100	...	no
I3	C2	550	600	...	yes
I4	C1	1000	800	...	unknown

Hypothesis:

$\text{amount_gbp} > \text{credit_limit} \wedge \text{pos}(\text{Comment}) \Rightarrow \text{yes}$

$\text{amount_gbp} \leq \text{credit_limit} \vee \text{neg}(\text{Comment}) \Rightarrow \text{no}$

Encoding

Choice of ML model: XGBoost $\{e(\text{invoice}_{\text{id}}), \dots, e(\text{comment})\}, \{Y/N\}$

LLM turning table into a text string (**text serialisation**) ??

Text serialization for relational learning

Text serialization (flattening): converting into 1D sequence of text characters so LLMs can read

invoice_id	customer_id	credit_limit_gbp	amount_gbp	Comment	paid_late (observed?)
I1	C1	600	600	...	no
I2	C1	600	100	...	no
I3	C2	550	600	...	yes
I4	C1	1000	800	...	unknown

...might be serialized into a 1D string like this:

Header: invoice_id, customer_id, credit_limit_gbp, amount_gbp, comment, paid_late

Row 1: I1, C1, 600, 600, ..., no.

Row 2: . . .

or better:

Row 1: invoice_id is I1, customer_id is C1, credit_limit_gbp is 600, amount_gbp is 600, comment is ..., paid_late is no

Row 2: . . .

Text serialization: why it fails for relational learning

invoice_id	customer_id	credit_limit_gbp	amount_gbp	Comment	paid_late (observed?)
I1	C1	600	600.00	...	no
I2	C1	600	100.00	...	no
I3	C2	550	600.00	...	yes
I4	C1	1000	800.00	...	unknown

Lost “cell” semantics

numerical: LLM tokenizer chops the number 800.00 into text pieces (e.g., ["800", ".00"])

categorical: keys/categorical IDs like C1 or I4 mean nothing to a standard ML model

temporal: Treating 2023-12-31 and 2024-01-01 as strings makes them look entirely different to an LLM

NULL: standard tokenizer loses typed missingness behavior (unknown, n/a, WITHHELD, ERR, etc)

Enterprise data (\$\$\$\$) is mostly structured

Text serialization: why it fails for relational learning

Header: EmployeeID, Name, Department, Salary.

Row 1: 101, Alice, Sales, 85000.

Row 2: 102, Bob, Engineering, NULL.

EmployeeID	Name	Department	Salary
101	Alice	Sales	85000
102	Bob	Engineering	NULL

Row 1: EmployeeID is 101, Name is Alice, Department is Sales, Salary is 85000.

Row 2: EmployeeID is 102, Name is Bob, Department is Engineering, Salary is NULL.

Lost Permutation Invariance

a relation = a **set** of tuples (so unordered)

EmployeeID	Name	Department	Salary
101	Alice	Sales	85000
102	Bob	Engineering	NULL

≡

Name	EmployeeID	Department	Salary
Alice	101	Sales	85000
Bob	102	Engineering	NULL

Text serialization: why it fails for relational learning

Header: EmployeeID, Name, Department, Salary.

Row 1: 101, Alice, Sales, 85000.

Row 2: 102, Bob, Engineering, NULL.

EmployeeID	Name	Department	Salary
101	Alice	Sales	85000
102	Bob	Engineering	NULL

Row 1: EmployeeID is 101, Name is Alice, Department is Sales, Salary is 85000.

Row 2: EmployeeID is 102, Name is Bob, Department is Engineering, Salary is NULL.

Lost **Permutation Invariance**

a relation = a **set** of tuples (so unordered)

EmployeeID	Name	Department	Salary
101	Alice	Sales	85000
102	Bob	Engineering	NULL

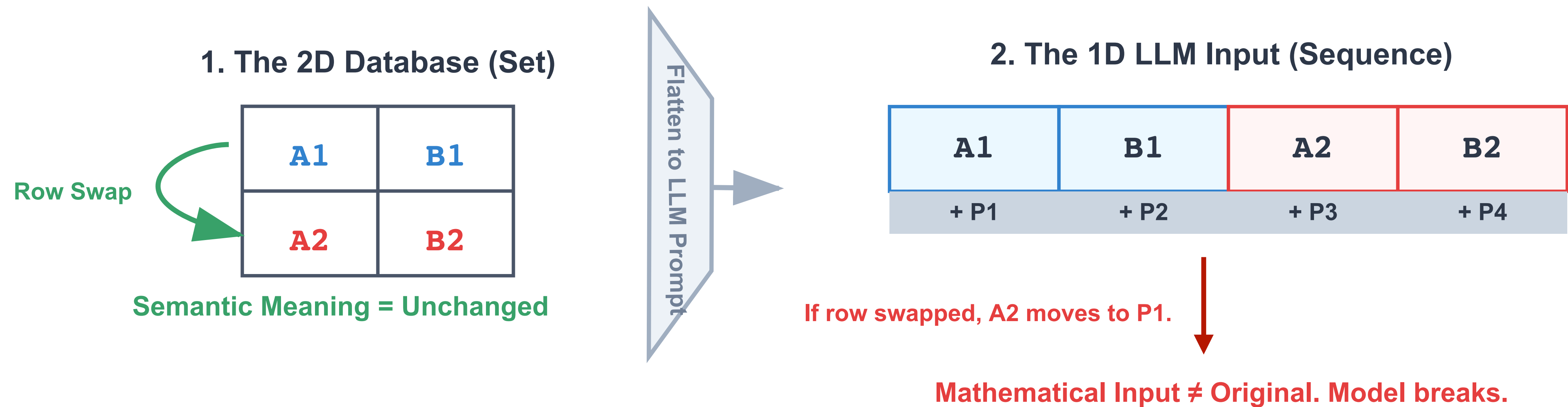
≡

Name	EmployeeID	Department	Salary
Alice	101	Sales	85000
Bob	102	Engineering	NULL



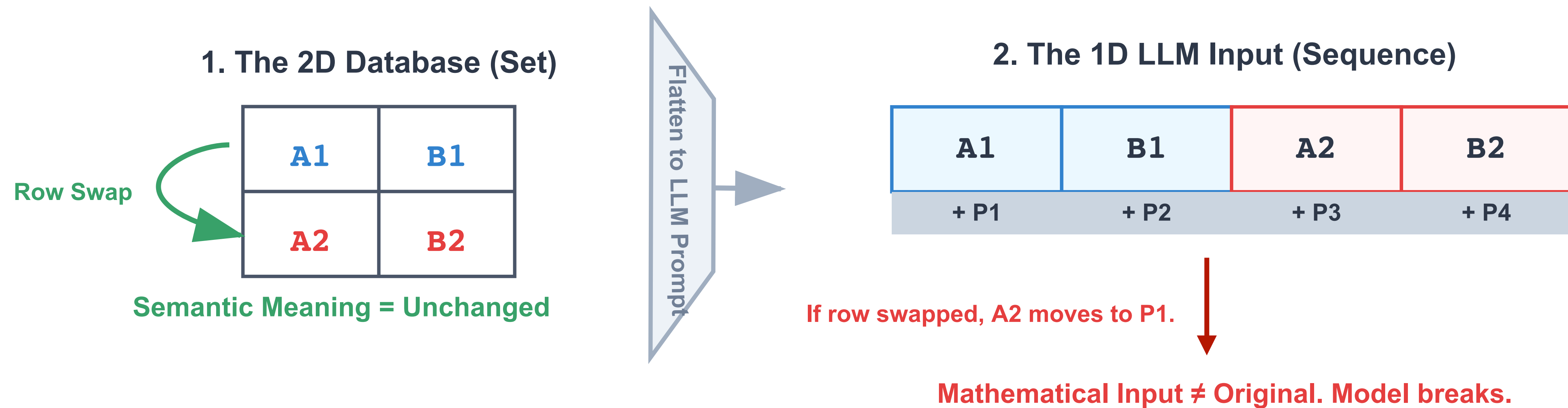
LLM lost this invariance

The Permutation Problem: 2D Sets vs. 1D Sequences for relations



$\text{emb}(A_2) + P1$ and $\text{emb}(A_2) + P3$ are different inputs, self-attention mechanism will give different outputs

The Permutation Problem: 2D Sets vs. 1D Sequences for relations



$\text{emb}(A_2) + P1$ and $\text{emb}(A_2) + P3$ are different inputs, self-attention mechanism will give different outputs

there is no easy fix for text serializer

The Permutation Problem

In relations row and column order are *not* semantics

Row-level model returns one prediction per row

example: predict whether each invoice is fraudulent

Table-level model returns one output for the whole table

example: classify the whole table or compute one table embedding

Desired behavior for row-level prediction

If we shuffle the rows of a table, then the row predictions should be shuffled in the same way:

$$f_{\text{row}}(\text{shuffled table}) = \text{shuffled } f_{\text{row}}(\text{table})$$

Desired behavior for table-level prediction

If we shuffle rows or columns, the whole-table prediction should stay the same:

$$f_{\text{table}}(\text{shuffled table}) = f_{\text{table}}(\text{table})$$

Practical implication:

If a model changes its answer just because a CSV was exported in a different row order or a schema was serialized in a different column order, then it has learned a formatting artifact, not table semantics.

Challenging to encode tables for LLMs

Consider a slot-based encoding layer encoding each cell as

$$A_{ij} = \phi(\text{cell}_{ij}, \text{header}_j) + p_i + q_j$$

$\phi(\text{cell}_{ij}, \text{header}_j)$ is the part that depends on the cell content and its column meaning;

p_i is a learned embedding for the *absolute row slot* i ;

q_j is a learned embedding for the *absolute column slot* j .

If the encoder is required to respect row and column permutation invariance, then the absolute slot embeddings cannot carry slot-specific information:

$$p_1 = \cdots = p_n, \quad q_1 = \cdots = q_m.$$

What is good encoding?

- x be the raw representation of a data object (e.g., row)
- $e(x)$ be the encoding fed into the model
- $t(x)$ be the answer required by a deterministic task
- $g(e(x))$ output of ML function g on input encoding $e(x)$

permutation-invariance

If x and x' are two representations of the **same** data object, and $e(x) \neq e(x')$, then g is **permutation-invariant** if $g(e(x)) = g(e(x'))$

Information loss

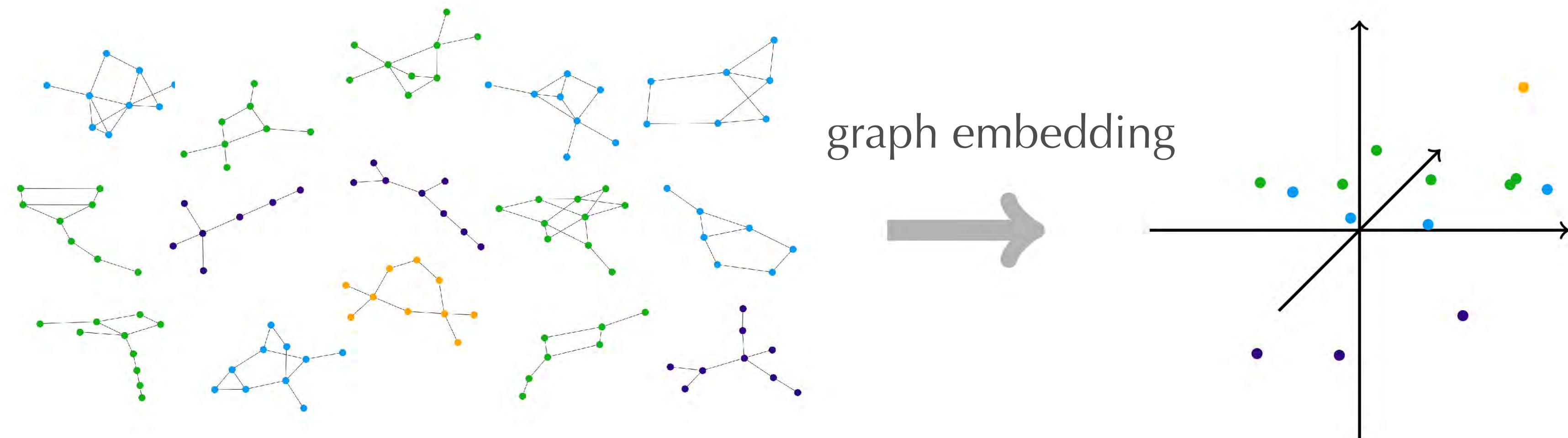
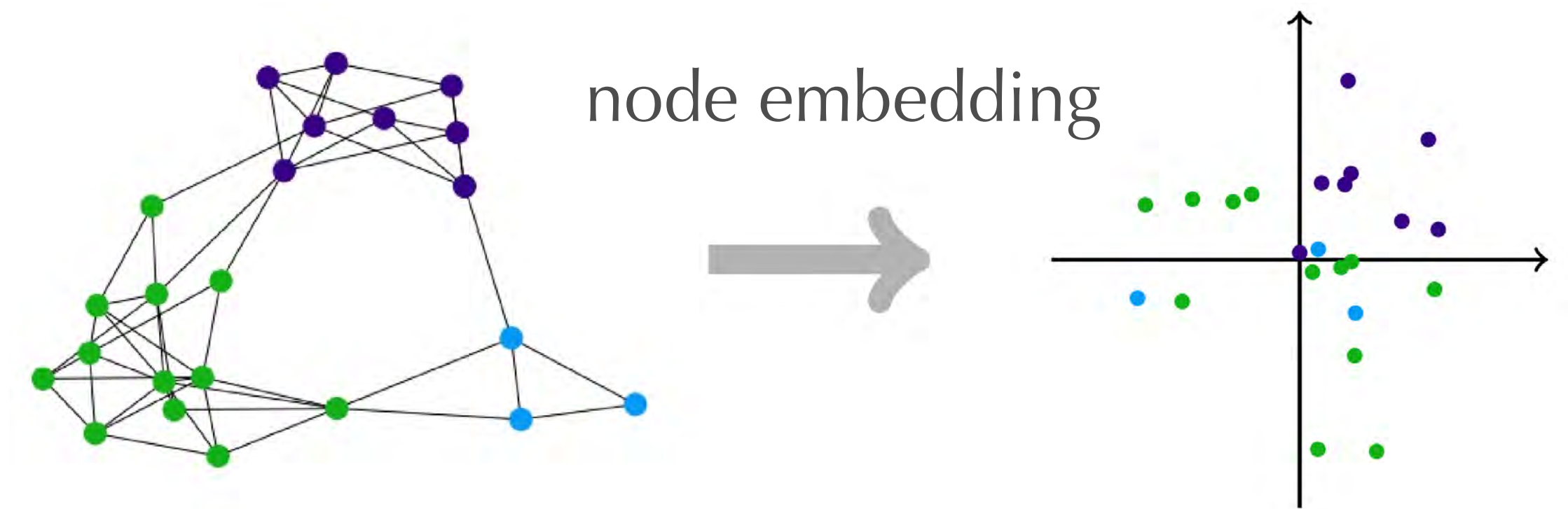
If $e(x_1) = e(x_2)$ but $t(x_1) \neq t(x_2)$, then **no** function g can satisfy $g(e(x)) = t(x)$ for both objects x_1 and x_2

Not limited to relational data

Graph Neural Networks often need **permutation-invariance** as well

Further reading:

- *Deep Sets* encoding
- Graph Isomorphism Network (GIN)
- Weisfeiler-Lehman test / expressiveness



Can LLM help (with the magic encoder)?

Will I4 pay late?

Customers:

customer_id	credit_limit_gbp
C1	600
C2	500

Payments:

payment_id	customer_id	invoice_id	paid_amount_gbp
P1	C1	I4	100
P2	C1	I4	50

Invoices:

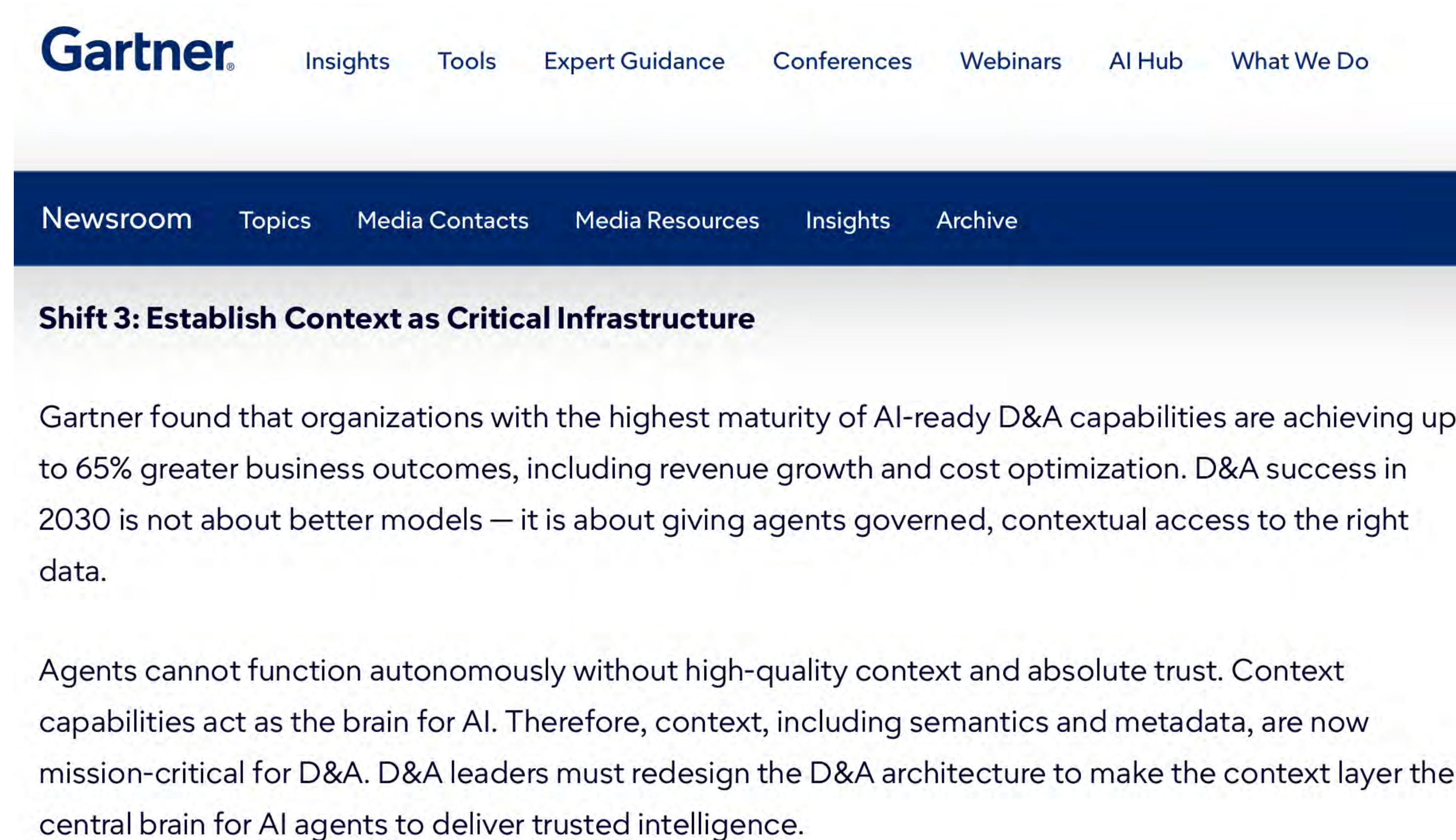
invoice_id	customer_id	amount_gbp	paid_late (observed?)
I1	C1	600	no
I2	C1	100	no
I3	C2	600	yes
I4	C1	800	unknown

Hypothesis: $\text{Invoices.amount_gbp} > \text{Customers.credit_limit} - \sum_{\text{customer}} \text{Payments.paid_amount_gbp} \Rightarrow \text{yes}$

LLM doesn't help even if we have row permutation invariant encoding (via e.g., "deep set")

(LLM needs all three tables, which can be too large to fit into context)

It is not just a toy example



traditional data pipeline ⇒ context management infrastructure

Retrieval-Augmented Generation / Context Engineering

Standalone LLMs

- Limited context windows

- No information beyond their training cut-off date

- Hallucinations, lack of sources (essential in e.g., search)

- No personal, private and enterprise knowledge

RAG (Retrieval-Augmented Generation)

- Integrates external sources (databases, reports, APIs/MCPs)

- Transforms general-purpose AI into specialized, high-value domain expert agents

- Context Engineering

- ...

Retrieval-Augmented Generation (RAG)

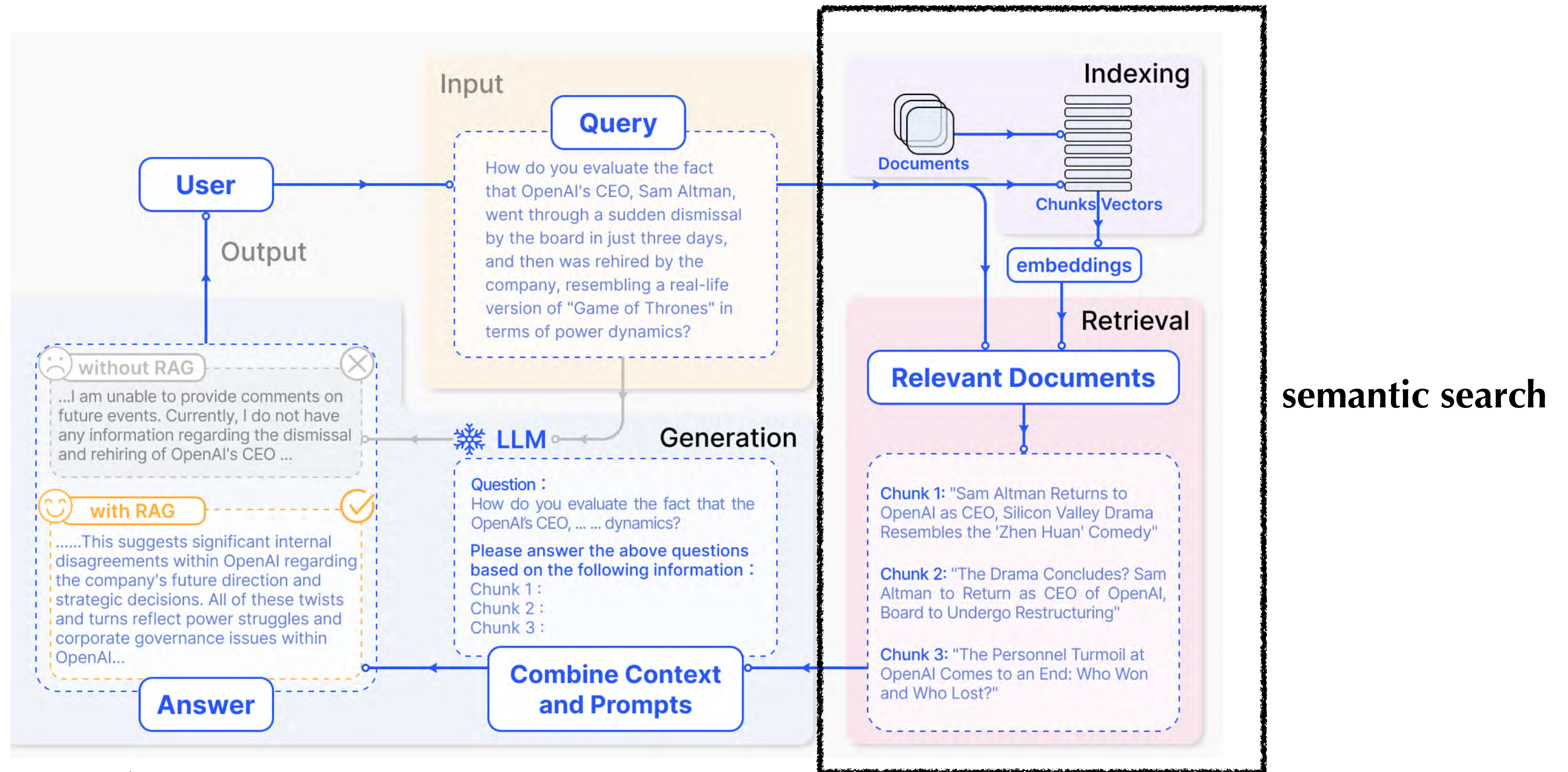
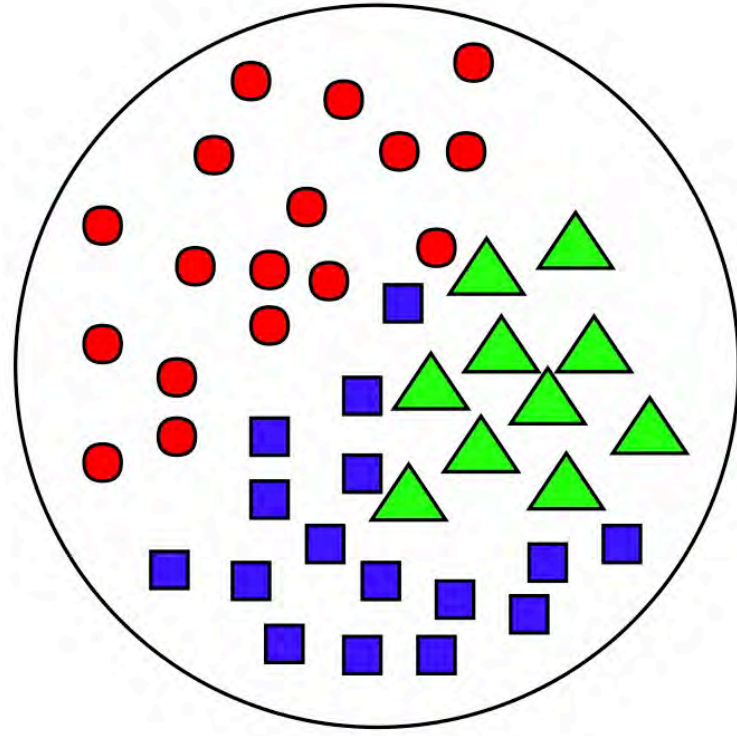


Figure: Gao et al (arXiv 2024)

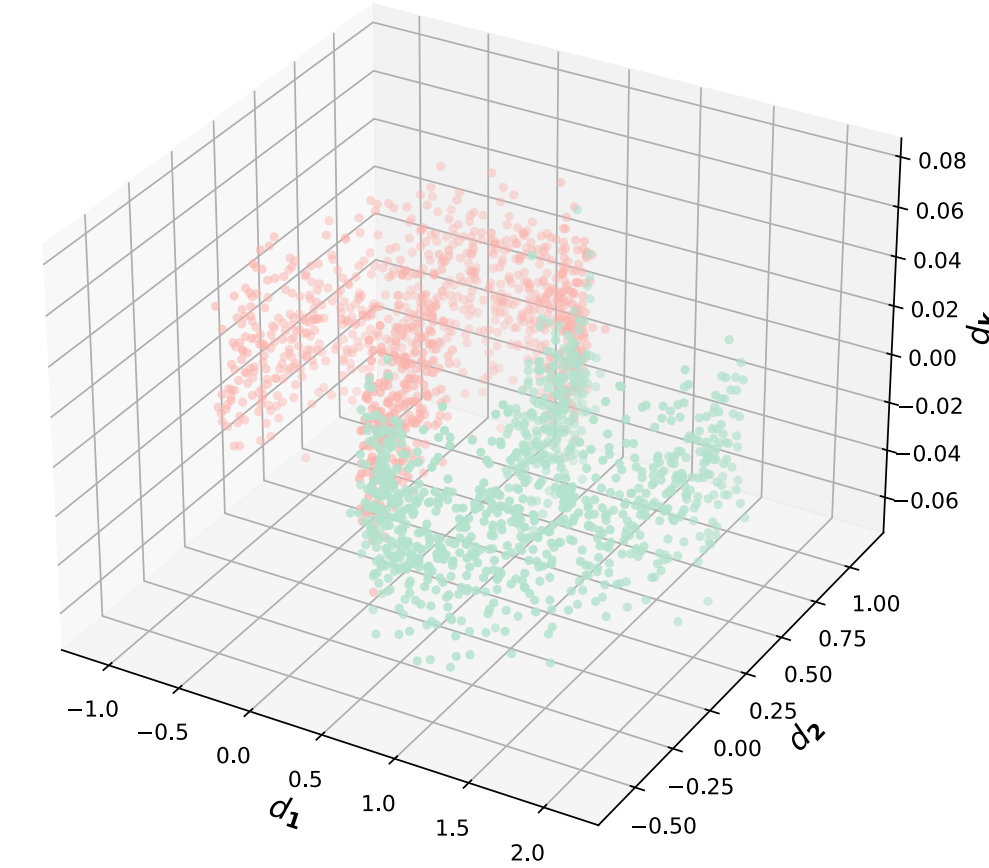
- **external memory** of LLMs: enable access to enterprise data, via *context engineering*
- **long-term** AI agent memory: ground AI agents in real-time enterprise data

Semantic (similarity) search via embedding vectors



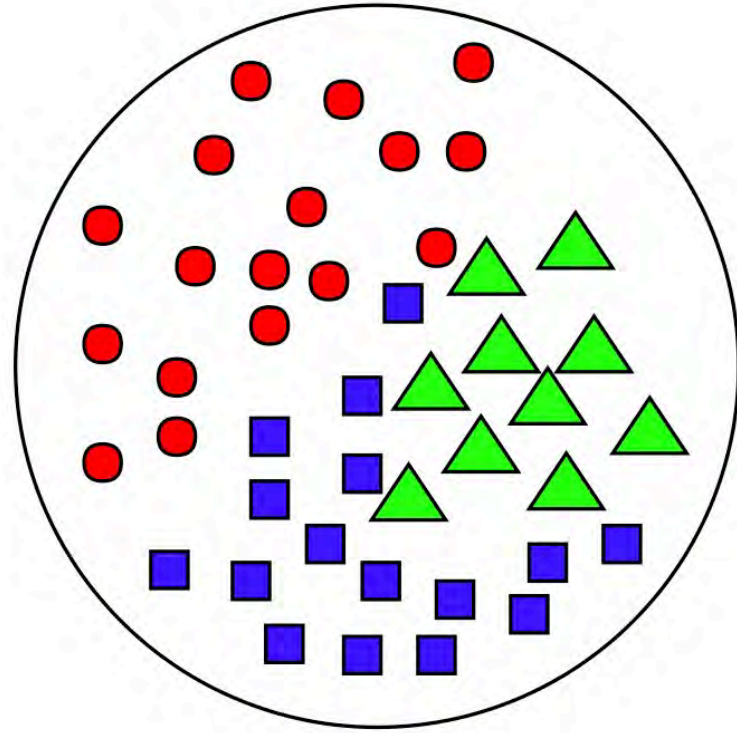
similarity (metric) space
Task: top-k similarity search

implemented by



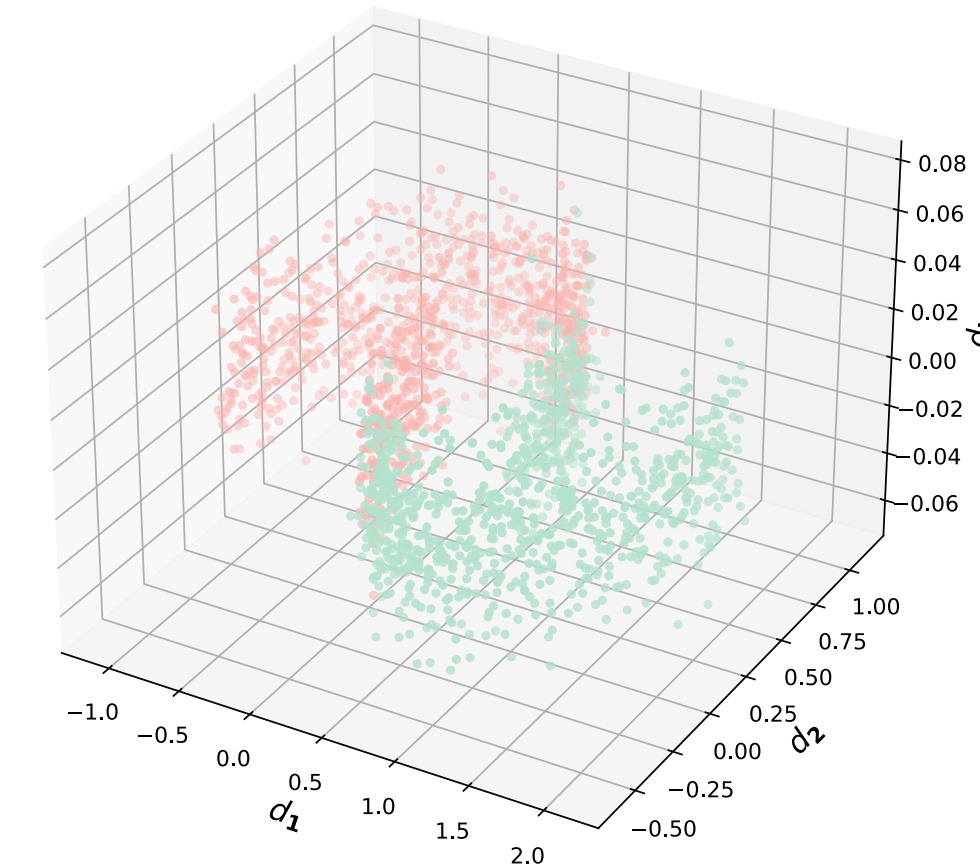
vector space
Task: top-k vector search

Semantic (similarity) search via embedding vectors

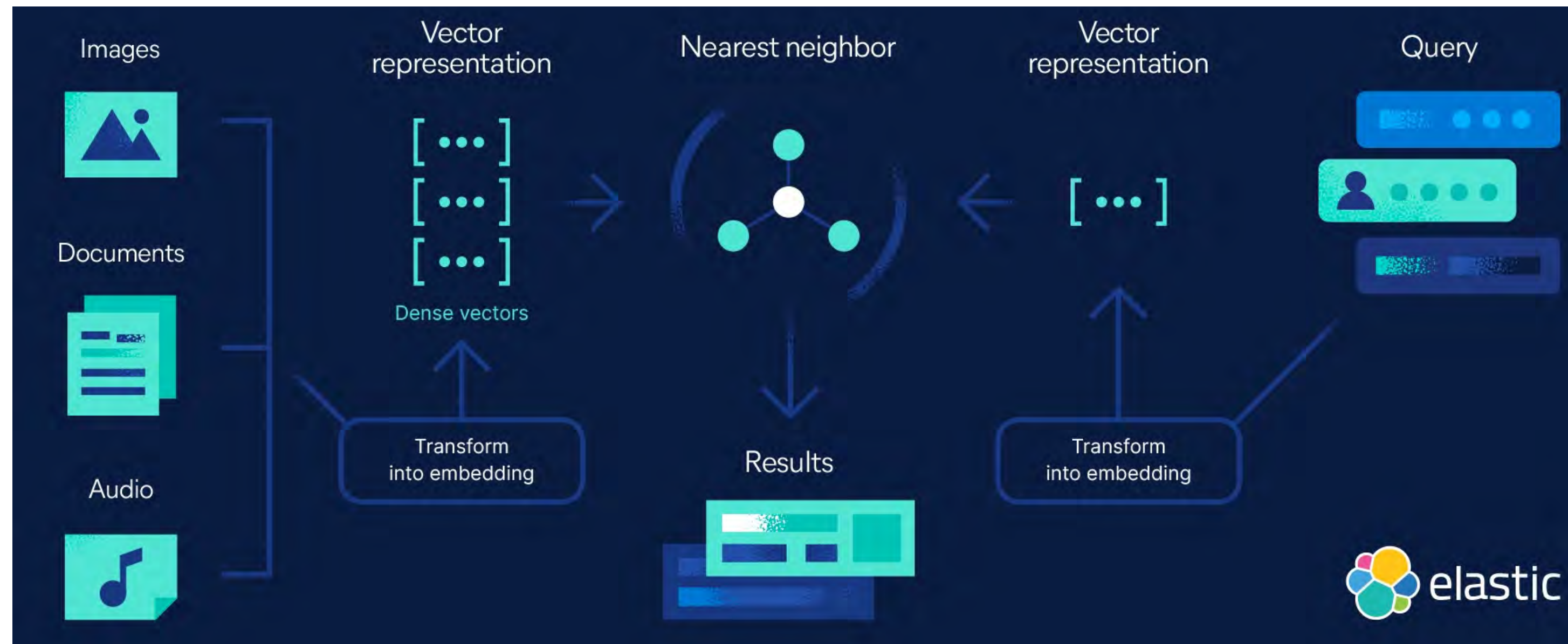


similarity (metric) space
Task: top-k similarity search

implemented by



vector space
Task: top-k vector search



Credit: <https://www.elastic.co/what-is/knn>

Similarity search $\rightarrow$ vector search

Input: dataset $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, query $\mathbf{q}$, distance function $\text{dist}(\mathbf{x}, \mathbf{y})$

Output: $S_k = \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subseteq D$ such that

$$\text{dist}(\mathbf{v}_i, \mathbf{q}) \leq \text{dist}(\mathbf{x}, \mathbf{q}), \text{ for any } \mathbf{v}_i \in S_k, \mathbf{x} \in D \setminus S_k$$

Similarity search $\rightarrow$ vector search

Input: dataset $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, query $\mathbf{q}$, distance function $\text{dist}(\mathbf{x}, \mathbf{y})$ $\mathbf{x} = (x_1, \dots, x_d)$
 $\mathbf{y} = (y_1, \dots, y_d)$

Output: $S_k = \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subseteq D$ such that

$$\text{dist}(\mathbf{v}_i, \mathbf{q}) \leq \text{dist}(\mathbf{x}, \mathbf{q}), \text{ for any } \mathbf{v}_i \in S_k, \mathbf{x} \in D \setminus S_k$$

Similarity search $\rightarrow$ vector search

Input: dataset $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$, query $\mathbf{q}$, distance function $\text{dist}(\mathbf{x}, \mathbf{y})$ $\mathbf{x} = (x_1, \dots, x_d)$
 $\mathbf{y} = (y_1, \dots, y_d)$

Output: $S_k = \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subseteq D$ such that

$$\text{dist}(\mathbf{v}_i, \mathbf{q}) \leq \text{dist}(\mathbf{x}, \mathbf{q}), \text{ for any } \mathbf{v}_i \in S_k, \mathbf{x} \in D \setminus S_k$$

L_2 -norm of $\mathbf{x}$: $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$ ($\mathbf{x}$ is L_2 -normalised if $\|\mathbf{x}\|_2 = 1$)

Euclidean $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$

dot-product $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^d x_i y_i$

cosine $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \cos(\theta_{\mathbf{xy}}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\sum_{i=1}^d x_i y_i}{\sqrt{\sum_{i=1}^d x_i^2} \sqrt{\sum_{i=1}^d y_i^2}}$

Top-k vector search: distance function matters

L_2 -norm of $\mathbf{x}$: $\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^d x_i^2}$ ($\mathbf{x}$ is L_2 -normalised if $\|\mathbf{x}\|_2 = 1$)

Euclidean $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$

dot-product $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^d x_i y_i$

cosine $\text{dist}(\mathbf{x}, \mathbf{y}) \doteq \cos(\theta_{\mathbf{xy}}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2} = \frac{\sum_{i=1}^d x_i y_i}{\sum_{i=1}^d x_i^2 \sum_{i=1}^d y_i^2}$

Distances of L_2 -normalised vectors

$$\|\mathbf{x}\|_2 = 1, \|\mathbf{y}\|_2 = 1$$

- $\mathbf{x} \cdot \mathbf{y} = \cos(\theta_{\mathbf{xy}})$
- $\|\mathbf{x} - \mathbf{y}\|_2^2 = 2(1 - \mathbf{x} \cdot \mathbf{y})$

Algorithm: kNN (k-nearest neighbor algorithm)

ALGORITHM 1: kNN (brute force)

Input: a set D of n vectors, a query vector q , distance function $\text{dist}(\cdot, \cdot)$, k ;

Output: $S_k \subseteq D$ s.t. $|S_k| = k$, $\text{dist}(q, \mathbf{x}) \geq \text{dist}(q, \mathbf{y})$ for any $\mathbf{x} \in S_k$, $\mathbf{y} \in D \setminus S_k$

1 $S_k \leftarrow \emptyset$;

2 **foreach** $\mathbf{x} \in D$ **do**

3 $\text{compute } \text{dist}(q, \mathbf{x})$

4 **sort** vectors $\mathbf{x}$ of D according to $\text{dist}(q, \mathbf{x})$ in non-decreasing order;

5 $S_k \leftarrow$ top- k $\mathbf{x}$'s in the sorted order;

6 **return** S_k

Algorithm: kNN (k-nearest neighbor algorithm)

ALGORITHM 1: kNN (brute force)

Input: a set D of n vectors, a query vector q , distance function $\text{dist}(\cdot, \cdot)$, k ;

Output: $S_k \subseteq D$ s.t. $|S_k| = k$, $\text{dist}(q, \mathbf{x}) \geq \text{dist}(q, \mathbf{y})$ for any $\mathbf{x} \in S_k$, $\mathbf{y} \in D \setminus S_k$

1 $S_k \leftarrow \emptyset$;

2 **foreach** $\mathbf{x} \in D$ **do**

3 $\text{compute dist}(q, \mathbf{x})$ alternative: k-selection

4 sort vectors $\mathbf{x}$ of D according to $\text{dist}(q, \mathbf{x})$ in non-decreasing order;

5 $S_k \leftarrow$ top- k $\mathbf{x}$'s in the sorted order;

6 **return** S_k

Algorithm: kNN (k-nearest neighbor algorithm)

ALGORITHM 1: kNN (brute force)

Input: a set D of n vectors, a query vector q , distance function $\text{dist}(\cdot, \cdot)$, k ;

Output: $S_k \subseteq D$ s.t. $|S_k| = k$, $\text{dist}(q, \mathbf{x}) \geq \text{dist}(q, \mathbf{y})$ for any $\mathbf{x} \in S_k$, $\mathbf{y} \in D \setminus S_k$

1 $S_k \leftarrow \emptyset$;

2 **foreach** $\mathbf{x} \in D$ **do**

3 compute $\text{dist}(q, \mathbf{x})$ alternative: k-selection

4 sort vectors $\mathbf{x}$ of D according to $\text{dist}(q, \mathbf{x})$ in non-decreasing order;

5 $S_k \leftarrow$ top- k $\mathbf{x}$'s in the sorted order;

6 **return** S_k

Good:

- small and sweet algorithm (= rich theoretical study)
- wide applications, e.g., kNN classification (1st example in ML101)

Algorithm: kNN (k-nearest neighbor algorithm)

ALGORITHM 1: kNN (brute force)

Input: a set D of n vectors, a query vector q , distance function $\text{dist}(\cdot, \cdot)$, k ;

Output: $S_k \subseteq D$ s.t. $|S_k| = k$, $\text{dist}(q, \mathbf{x}) \geq \text{dist}(q, \mathbf{y})$ for any $\mathbf{x} \in S_k, \mathbf{y} \in D \setminus S_k$

1 $S_k \leftarrow \emptyset$;

2 **foreach** $\mathbf{x} \in D$ **do**

3 compute $\text{dist}(q, \mathbf{x})$ alternative: k-selection

4 sort vectors $\mathbf{x}$ of D according to $\text{dist}(q, \mathbf{x})$ in non-decreasing order;

5 $S_k \leftarrow$ top- k $\mathbf{x}$'s in the sorted order;

6 **return** S_k

Good:

- small and sweet algorithm (= rich theoretical study)
- wide applications, e.g., kNN classification (1st example in ML101)

Bad:

- curse of dimensionality ^(d) $\Leftarrow$ Points become indistinguishable via distances to q with large d
- complexity $O(nd + n \min(k, \log n))$ $\Leftarrow$ doesn't scale, impractical for large scale datasets

Note: can be reduced to $O(nd)$ via **Quickselect** & **median of medians** (good in theory, bad in practice)

GPU k-selection in Faiss

Approximate Nearest Neighbor

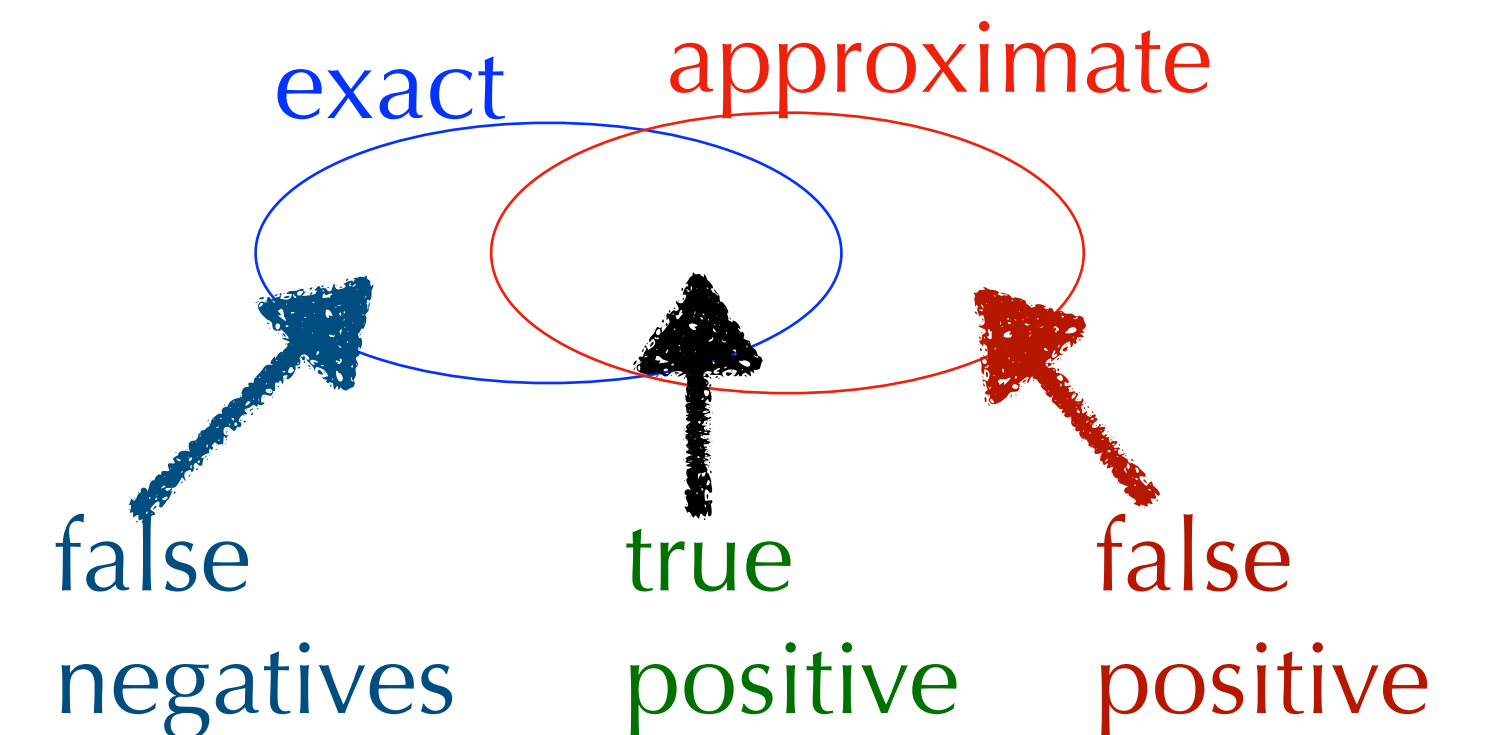
Idea: approximate top- k answers

- trade **accuracy** for better **performance** (navigate around “curse of dimensionality”)
- use pre-computed **index**

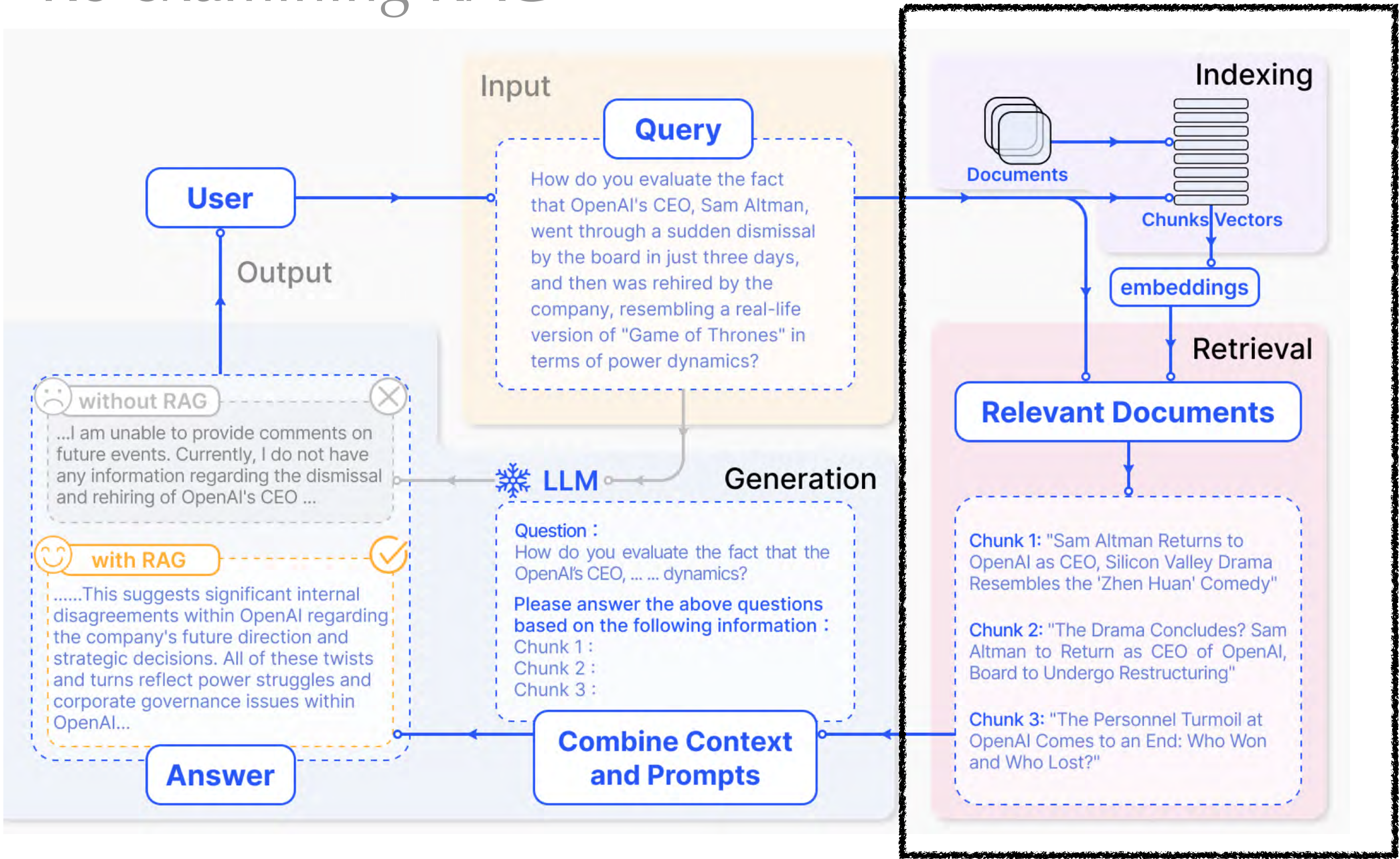
Performance: number of vectors visited / distance comparisons

Accuracy: measure the quality of approximate (inexact) kNN answers

- recall: $\frac{\text{true positives}}{\text{false negatives} + \text{true positives}}$
- precision: $\frac{\text{true positives}}{\text{false positives} + \text{true positives}}$
- recall@K:
 - recall when $k = K$ for k NN
 - proportion of queries where at least 1 of the top- k exact answers is returned
(works better for low-precision approximate answers)

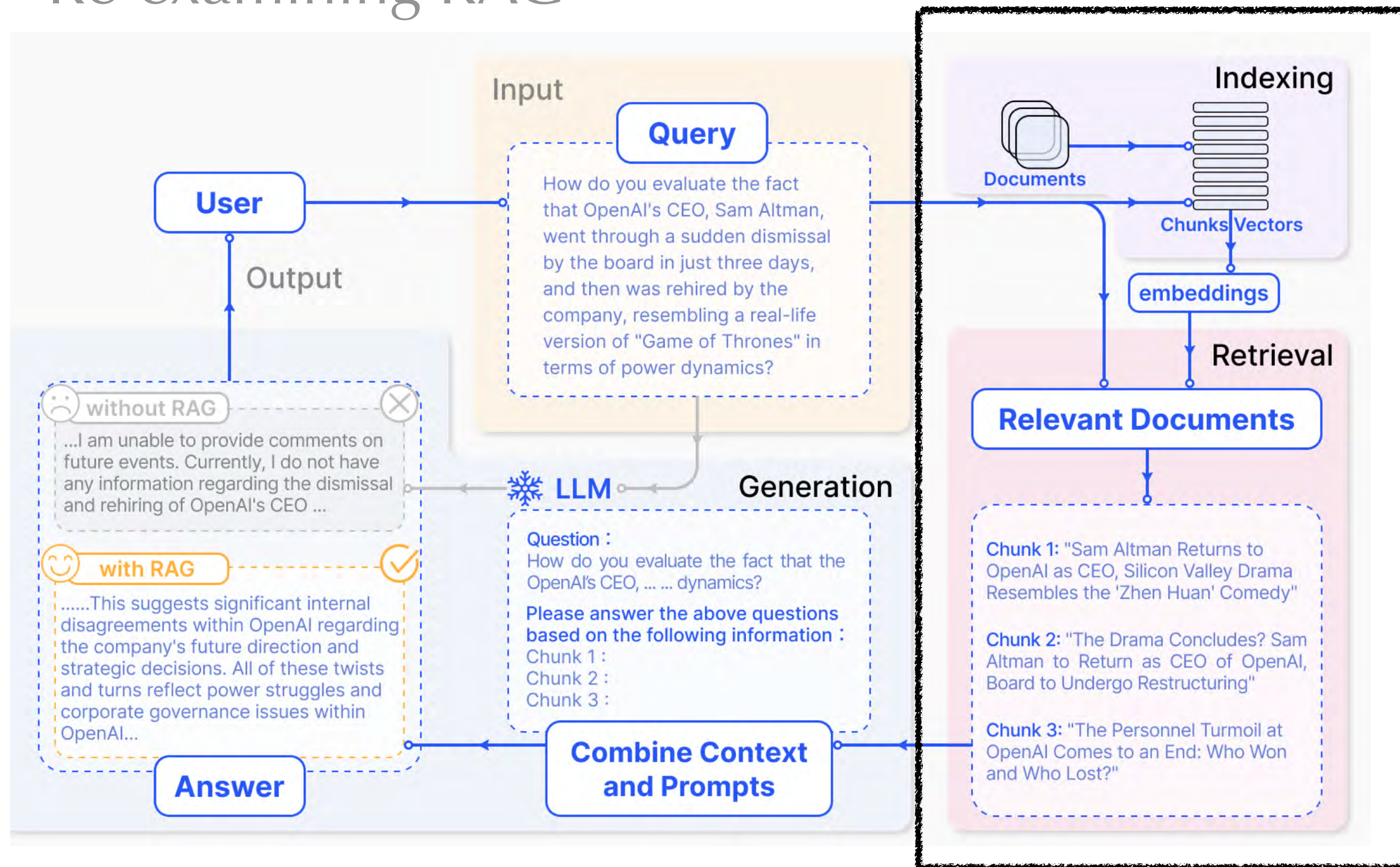


Re-examining RAG



embedding (encoder)
+
Top-k vector search

Re-examining RAG



embedding (encoder)
+
Top-k vector search

Q: What if LLM has a *different semantic distance* from the encoder used in RAG?

Will I4 pay late (LLM with RAG)?

Customers:

customer_id	credit_limit_gbp
C1	600
C2	500

Payments:

payment_id	customer_id	invoice_id	paid_amount_gbp
P1	C1	I4	100
P2	C1	I4	50

Invoices:

invoice_id	customer_id	amount_gbp	paid_late (observed?)
I1	C1	600	no
I2	C1	100	no
I3	C2	600	yes
I4	C1	800	unknown

Hypothesis: $\text{Invoices.amount_gbp} > \text{Customers.credit_limit} - \sum_{\text{customer}} \text{Payments.paid_amount_gpb} \Rightarrow \text{yes}$

Will I4 pay late (LLM with RAG)?

Customers:

customer_id	credit_limit_gbp
C1	600
C2	500

Payments:

payment_id	customer_id	invoice_id	paid_amount_gbp
P1	C1	I4	100
P2	C1	I4	50

Invoices:

invoice_id	customer_id	amount_gbp	paid_late (observed?)
I1	C1	600	no
I2	C1	100	no
I3	C2	600	yes
I4	C1	800	unknown

Hypothesis: $\text{Invoices.amount_gbp} > \text{Customers.credit_limit} - \sum_{\text{customer}} \text{Payments.paid_amount_gbp} \Rightarrow \text{yes}$

RAG (idea): select row (I4) from Invoices, join with Customer and Payments on (C1, I4)

Augmented prompt: Will I4 pay late? (invoice_id:I4, customer_id:C1, amount_gbp:800 ...)

A more “real” example: Enterprise Supplier Risk Copilot

Consider an enterprise copilot that operates over a live relational system, over database with tables:
`suppliers, shipments, orders, invoices, support_tickets, contracts`

Question 1: *What are customers complaining about regarding supplier B?*

(search complaint text and tickets, retrieve relevant evidence, and summarize what was found)

Question 2: *How many rejected invoices were issued in the UK last month?*

(exact filtering, counting, temporal constraints, schema grounding)

Question 3: *Summarize complaints about suppliers with more than three delayed shipments in Q3.*

(Exactly identify the supplier set, then summarize over complaint evidence)

Why inference over relational data is harder

Standard inference: understands the question → retrieve relevant evidence → produce an answer

Over a live database, answering a question may additionally require:

- semantic retrieval: vague or text-heavy matching, e.g., find ``complaints about billing fraud''

- exact execution: compute joins, counts, sums, filters, and time windows (SQL)

- schema grounding: identify which tables, columns, and join paths are relevant

- integrity: respect foreign keys, valid joins, and permissions

- answer synthesis: turn rows or aggregates into a useful explanation

- freshness and provenance: answer currency and provenance (evidence)

RAG extended for relational databases

Vector RAG: solves semantic evidence retrieval

The common RAG architecture: embedding-based semantic retrieval for LLM context

Relational RAG: solves exact structured execution based retrieval

Use SQL queries to retrieve results as final answers or LLM context

key interface: NL2SQL

Table-augmented Retrieval (TAG): solves hybrid questions, both exact execution and explanation

Mix of vector and relational RAG

NL2SQL — the exact-execution interface of relational RAG

NL2SQL is ideal when a question can be answered by:

- selecting the right tables and columns,

- forming the right joins,

- applying the right filters,

- executing exact aggregation in the DB engine.

Core abstraction

NL2SQL maps a natural-language question to SQL; the DBMS then executes the SQL

Retrieve structured evidence units such as rows, joined rows, entities, paths, or attached documents

rigorous semantics, exactness, no approximation/vagueness/hallucination

NL2SQL example: how it generates SQL

Q: *How many rejected invoices were issued in the UK last month?*

Step 1: semantic and schema grounding

intent: this is a count question

table: likely invoice

predicates: `status = REJECTED`, `country = UK`, `invoice_date` in last month

Step 2: structured query construction

build an internal SQL plan or abstract syntax tree

- `SELECT COUNT(*)`
- `FROM invoice`
- `WHERE` the grounded predicates hold

```
SELECT COUNT(*) AS  
    rejected_invoices  
FROM invoice  
WHERE country = 'UK'  
    AND status = 'REJECTED'  
    AND invoice_date >= DATE '  
        2026-02-01'  
    AND invoice_date < DATE '  
        2025-03-01';
```

NL2SQL example

Q: *How many rejected invoices were issued in the UK last month?*

Step 3: generate SQL under structure constraints:
valid tables, columns, and grammar/AST,
legal joins if multiple tables are involved

Step 4: validation and execution
parse and type-check query
optionally run **EXPLAIN** or partial execution, repair if needed
execute in the DB engine

```
SELECT COUNT(*) AS  
    rejected_invoices  
FROM invoice  
WHERE country = 'UK'  
    AND status = 'REJECTED'  
    AND invoice_date >= DATE '  
        2026-02-01',  
    AND invoice_date < DATE '  
        2025-03-01';
```

NL2SQL combines question understanding, schema grounding, structured decoding (and DB-side exact execution)

Sometimes we need both RAG and NL2SQL-based RAG

Hybrid question: *Summarize complaints about suppliers with more than three delayed shipments in Q3.*

- exact structured filtering over shipments

- semantic (embedding-based) retrieval of complaint text

- natural-language summarization of the final evidence

Natural answer: a hybrid NL2SQL + vector RAG system

- run NL2SQL to get exact structured results

- run vector RAG to retrieve semantically relevant text evidence

- then combine them in the answer generator

Limitation of the glued hybrid RAG

A naive “run both in parallel” architecture still leaves open:

- which subqueries should be run?

- should the two branches run in parallel or sequentially?

- how should the structured and semantic evidence be fused?

- what if one branch should constrain the other?

- how do we keep provenance and consistency?

Limitation of the glued hybrid RAG

Question: *Summarize complaints about suppliers with more than three delayed shipments in Q3*

we usually do not want complaint retrieval over all suppliers in the company

We want complaint retrieval for suppliers that passed SQL condition (≥ 3 delayed shipments in Q3)

Vector/SQL branches are dependent

“How many suppliers had complaints about billing fraud last quarter?”

“billing fraud” may not be a clean database label so semantic retrieval may be needed first

then SQL can count the matched suppliers

not simple parallelism

Limitation of the glued hybrid RAG

Even if both branches run in parallel, you still must answer:

- how do I entity-link retrieved passages to SQL entities?

- how do I intersect result sets?

- what if they disagree?

- which evidence is primary?

- how do I preserve provenance?

A parallel RAG architecture gives you two outputs, but not a clean/cost-efficient way to combine them

Table-augmented Retrieval (TAG) gives the correct abstraction for handling these cases.

Table-augmented Generation

Text2SQL is Not Enough: Unifying AI and Databases with TAG

Asim Biswal^{1,*} Liana Patel^{2,*} Siddharth Jha¹ Amog Kamsetty¹ Shu Liu¹
Joseph E. Gonzalez¹ Carlos Guestrin² Matei Zaharia¹
¹UC Berkeley ²Stanford University

1 Query Synthesis



"Summarize the reviews of the highest grossing romance movie considered a 'classic'."



```
WITH CRM AS (SELECT * FROM movies WHERE genre = 'Romance'
AND LLM('{movie_title} is a classic') = 'True')
SELECT * FROM CRM
WHERE revenue = (SELECT MAX(revenue) FROM CRM);
```

2 Query Execution

movie_title	revenue	review	genre
Shang-Chi	432.2	"solid film..."	Action
Titanic	2257.8	"still best..."	Romance
Titanic	2257.8	"a guilty..."	Romance
...	...	...	...



movie_title	revenue	review	genre
Titanic	2257.8	"still best..."	Romance
Titanic	2257.8	"a guilty..."	Romance
...	...	...	...

3 Answer Generation

"Summarize the reviews of the highest grossing romance movie considered a 'classic'."

"{movie_title: 'Titanic', revenue: 2257.8, review: 'still best...', genre: 'Romance'}..."



"The reviews of Titanic discuss the on screen chemistry..."



SQL extended with LLM prompts

Summary

1. Data plays a critical role in in-production ML systems

2. Encoding/embedding decides what model can see

permutation-invariance encoding is desirable for structured data and graphs/networks

3. traditional pipeline $\Rightarrow$ context management (e.g., RAG)

semantic retrieval, hybrid retrieval at inference-time

based on embeddings + vector/hybrid search