WHITE PAPER



RETHINKING CONTEXT ENGINEERING FOR STRUCTURED ENTERPRISE DATA

Large language models (LLMs) are only as reliable as the context provided to them. While semantic search effectively retrieves meaning from unstructured content, structured enterprise data requires precision and determinism. This paper explores how context engineering can improve the retrieval of structured data for trustworthy AI outcomes.

Infosys
Navigate your next

Modern enterprises manage vast amounts of structured and unstructured data. While semantic search is highly effective for documents, emails, conversations, and other unstructured content, structured data presents different requirements. Customers, transactions, products, policies, inventories, and financial records are organized for control, precision, auditability, and deterministic access.

Applying semantic search to structured data can be problematic because it relies on probabilistic similarity rather than exact retrieval. This paper reframes structured data retrieval as a context construction challenge, focusing on assembling a precise, minimal, and relevant subset of records before handing them to an LLM.

Business Case

Generative artificial intelligence (GenAI) is increasingly being integrated into enterprise workflows to improve decision making, automate processes, and reduce operational effort.

In regulatory environments, false positives can often prove more harmful than false negatives. For example, a false match against a sanctions list can freeze a legitimate customer's account and trigger an unwarranted compliance investigation. In such cases, it may be safer for a system to return no result than to confidently serve up a wrong one. A wrong match creates false confidence, dangerous in operational systems as it can lead users to make erroneous decisions owing to incorrect information.

That is why retrieval in structured enterprise systems cannot be treated as a conventional relevance-ranking problem alone. It must function as a reliability boundary, with the system determining whether the information entering the model is trustworthy enough to support a decision.

Incorrect data in the context of a large language model (LLM) can lead to incorrect responses. In turn, this can trigger inappropriate actions, undermining the efficiency gains that AI is intended to provide.

From the user's perspective, structured retrieval is expected to behave like a database. The same query should return the same result every time, provided the underlying data has not changed. Users also expect a clear explanation of why a particular record was selected.

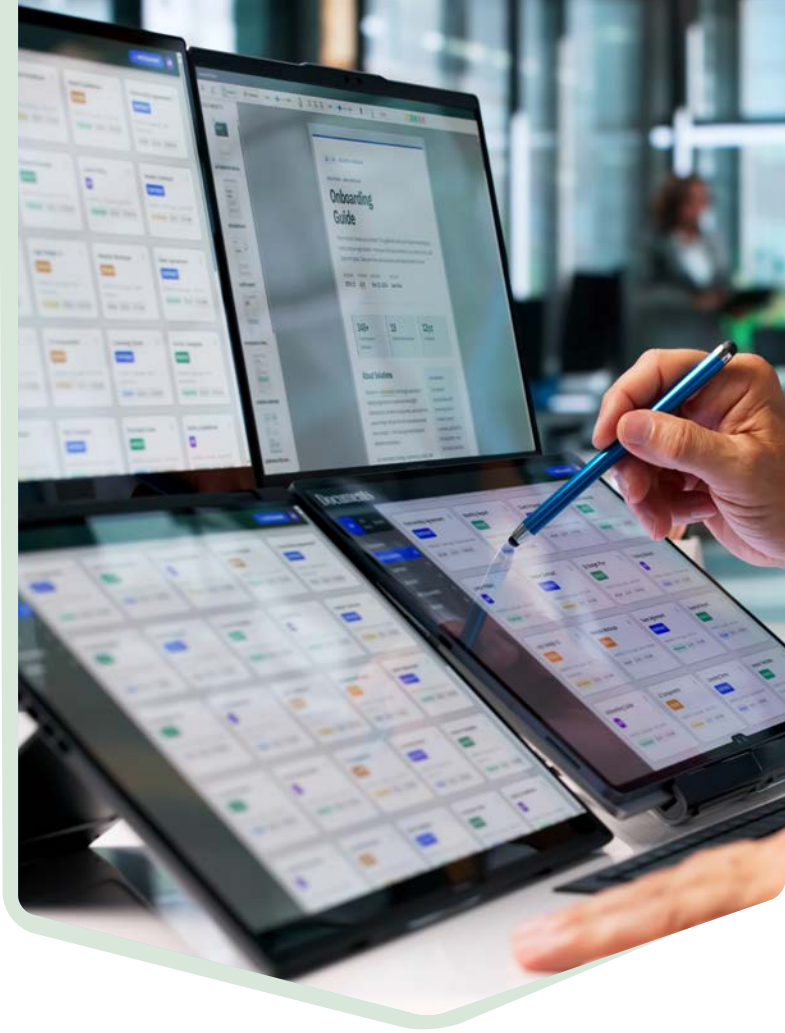
Table 1 illustrates how incorrect retrieval in common enterprise workflows translates directly into operational and financial consequences.

Table 1: Risk of incorrect retrieval with associated critical attributes

Domain	Use case	Risk of incorrect retrieval	Critical attributes
Customer operations	Account verification	Retrieving the wrong customer identity, address, or account details can break trust, delay issue resolution, trigger disputes, and increase churn risk	Customer name, postal code, account ID
Supply chain	Urgent replenishment	Incorrect inventory, supplier, or product information can result in stockouts, shipment errors, missed service levels, and expensive emergency reorders	Stock keeping unit (SKU), supplier ID, product code
Finance	Invoice matching and payment validation	Retrieving the wrong invoice, vendor, or purchase order (PO)_can cause duplicate payments, delayed settlements, reconciliation issues, or financial control failures	Invoice number, vendor ID, PO number
Healthcare/ insurance operations	Member or patient identification	Retrieving the wrong individual's policy or record can cause claim errors, service delays, privacy breaches, and serious trust issues	Member ID, patient ID, date of birth, policy number

External Document © 2026 Infosys Limited

Structured Vs. Unstructured Data: Schema-based and Semantic Retrieval



Unstructured data consists of free-form text, where meaning is encoded implicitly rather than through a fixed schema. The same idea can be expressed in multiple ways, for example, "revenue grew" and "top-line expanded". As the relationship between words and meaning is many-to-many, embedding models that project text into a semantic vector space makes for a natural fit. Approximate nearest-neighbor search over these vectors improves recall and retrieval efficiency with some reduction detected in retrieval precision. Semantic search converts text into dense embeddings. This enables semantic generalization, which is the ability to recognize and retrieve conceptually related items even where different words or phrases are used.

Structured data, by contrast, is organized into tables and columns, with each field carrying an explicit type, purpose, and constraint. A Customer Name column stores names; an Order ID column stores identifiers; and a ZIP Code column stores postal codes. These are not interchangeable tokens in a semantic space. They are typed, bounded attributes whose meaning is defined and enforced by the schema.

Schema matching operates on surface forms by applying fuzzy techniques such as Levenshtein distance and n-gram overlap to determine text similarity. These techniques quantify similarity at the character or token level and do not require exact equality. This allows the system to tolerate minor variations such as typos, abbreviations, and transliteration differences while maintaining field-level control over matching. Matching is performed column by column. Therefore, it respects schema boundaries and supports rule-based tolerance. It has thresholds tuned independently for each attribute, such as stricter ones for identifiers and more lenient levels for names or addresses.

Consider a typical enterprise customer table:

CUSTOMER

customer_id PK, INT	org_name VARCHAR(200)	address_line VARCHAR(500)	city VARCHAR(100)	state CHAR(2)	zip_code CHAR(10)	phone VARCHAR(15)

Each column carries its own matching semantics. For example, organization name values such as "Acme Corp" and "ACME Corporation" may require case-insensitive, token-based comparison. However, ZIP codes typically require exact or prefix matching. Semantic similarity should never cause "10001" to be considered as equivalent to "10002". When an entire row is flattened into a single 1,536-dimensional embedding, the representation does not save the column-level constraints, data types, and structural relationships that the schema was designed to preserve.

External Document © 2026 Infosys Limited

Fig 1: Fuzzy field-level matching correctly ranks “Halton Produce Company” above the lexically similar but incorrect “Hamilton Products Corp.”

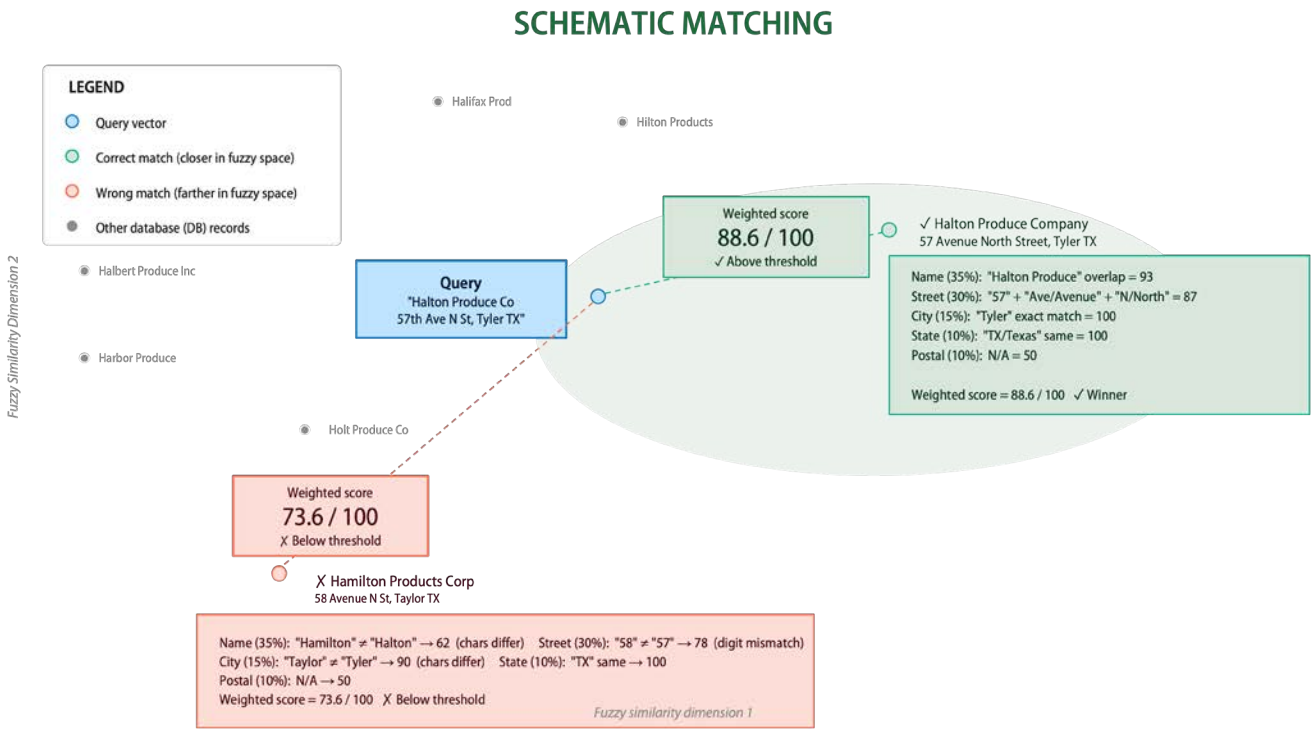
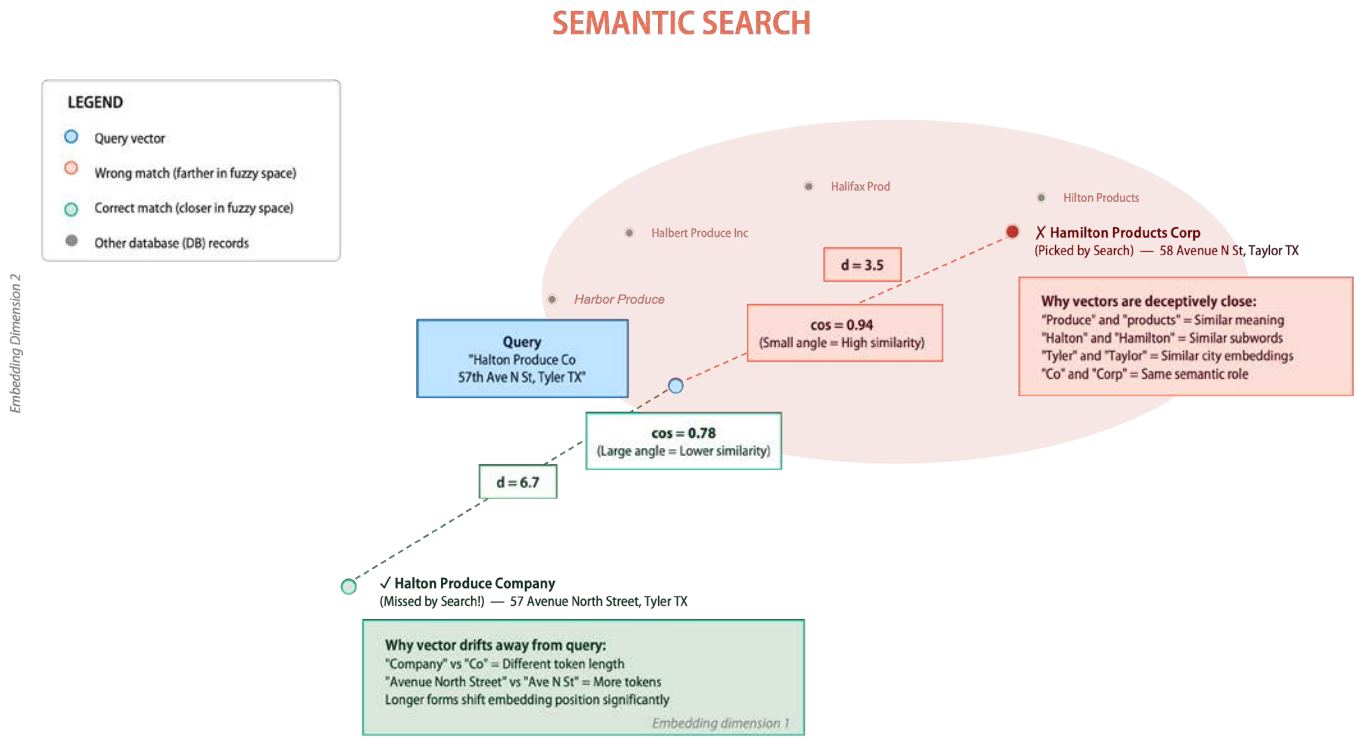


Fig 2: Semantic search picks the wrong candidate (“Hamilton Products Corp”) because vector similarity favors surface-level word overlap over the true match.



Why Vector Search Falls Short for Structured Records

A vector search is well suited to semantic retrieval, but it introduces material risks when used for structured entity resolution. In the case of structured data, it returns the nearest match in the absence of an accurate one, thereby potentially introducing incorrect records into the LLM context. This leads to confident hallucinations rather than safely returning no results.

A vector search can also impose additional operational costs. Vector indexes are computationally expensive to build and maintain. Frequent record updates often require embeddings to be regenerated and reindexed. Performance could degrade with scale due to the curse of dimensionality, where the algorithms become less effective as the number of dimensions increase. This results in a forced approximate search, with vector similarity lacking attribute-level explainability. With a breakdown in the prediction model, auditing and understanding why a particular record was retrieved becomes a challenge in regulated environments. Fuzzy matching provides an alternative by supporting threshold-based rejection, efficient structured updates, field-level comparison, and transparent scoring. Table 1 compares the expectations from vector search results and fuzzy matching.

Table 1: Vector search result expectations vs. fuzzy matching result characteristics

Expectation	Vector retrieval	Fuzzy matching
Consistency	Probabilistic; may vary across runs	Deterministic, identical inputs produce identical outputs
Accuracy	Semantically similar but factually incorrect rows	Exact or high-confidence matches
Noise control	"Close enough" noise is common	Hard filters eliminate irrelevant rows
Explainability	Abstract or limited similarity scores	Field-level, human-readable logic
Audit readiness	Embeddings are opaque and difficult to interpret	Fully traceable rule-based decisions

Engineering Reliable Context: A Deterministic Retrieval Framework

A reliable retrieval pipeline must operate as a progressive narrowing funnel. It can start with inexpensive preprocessing steps and advance toward increasingly precise and computationally expensive comparisons. In fuzzy-matching systems, match accuracy depends heavily on the cleanliness of the input. Unlike vector models, which can absorb a degree of noise through semantic generalization, deterministic fuzzy matching requires explicit noise reduction and canonicalization before matching. When done correctly, preprocessing can transform borderline matches into exact ones, improving both recall and precision.

Weighted composite scoring

The core of the fuzzy matching approach is a weighted composite score computed across individual fields. For a candidate record R and a query Q with n matchable fields, the composite match score is:

$$\text{Score}(Q, R) = w_1 \times \text{sim}(Q.\text{field}_1, R.\text{field}_1) + w_2 \times \text{sim}(Q.\text{field}_2, R.\text{field}_2) + \ldots + w_n \times \text{sim}(Q.\text{field}_n, R.\text{field}_n)$$

where $w_1 + w_2 + \ldots + w_n = 1.0$, and sim() is the field-appropriate similarity function (e.g., token_set_ratio for names, edit distance for addresses, exact match for IDs). A record is returned only if Score(Q, R) ≥ threshold (e.g., 0.85). This decomposition is transparent: every field's contribution to the score is inspectable, tunable, and auditable.

Example: For a customer lookup, an enterprise might configure: org_name (w=0.40, token_set_ratio), address

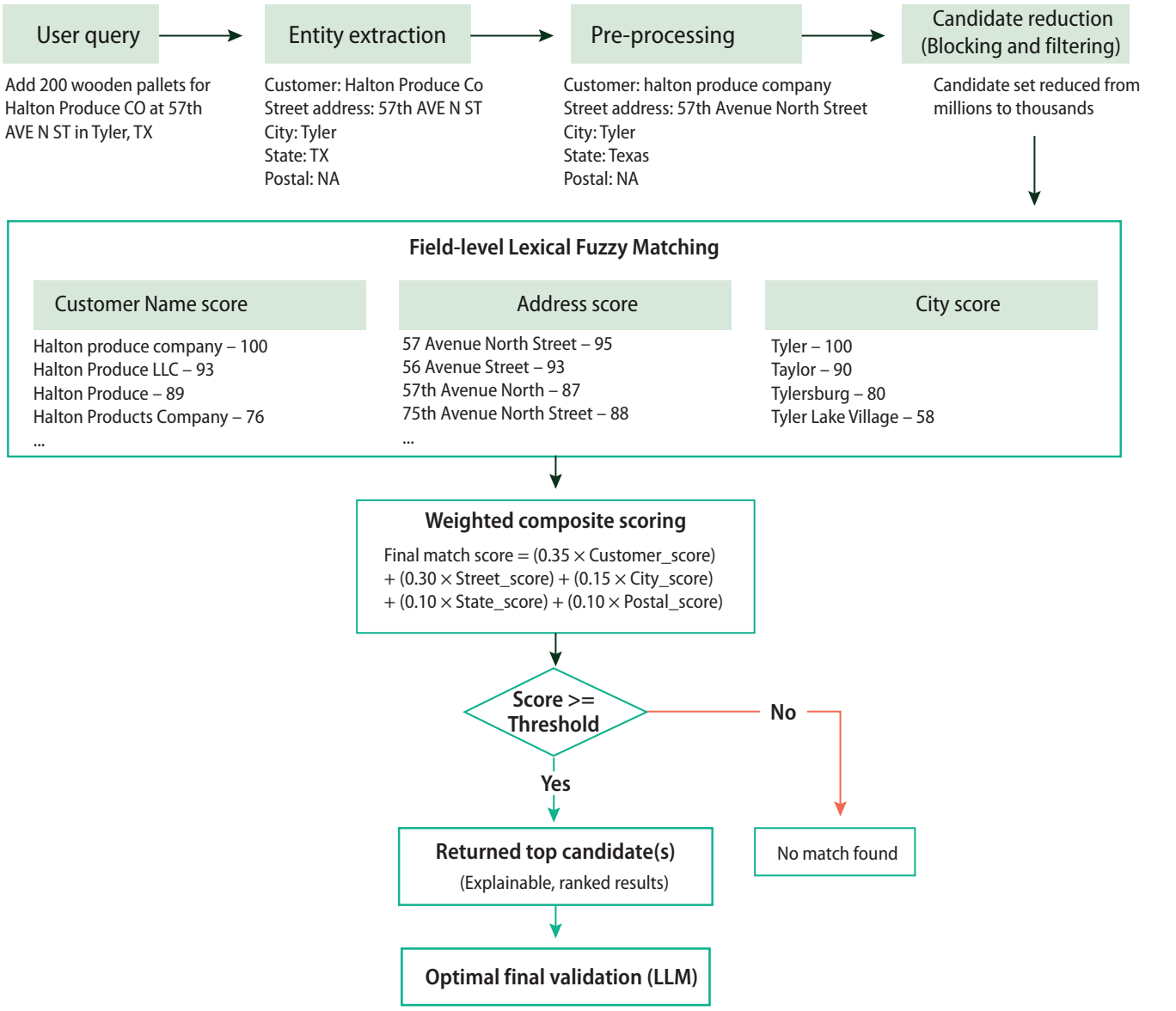
External Document © 2026 Infosys Limited

(w=0.25, partial_ratio), city (w=0.15, exact), state (w=0.10, exact), zip (w=0.10, prefix). If a query for "Acme Corporation, Dallas" matches a record "ACME Corp, Dallas TX" with field scores of 0.91, 0.88, 1.0, 1.0, 1.0, the composite is 0.40(0.91) + 0.25(0.88) + 0.15(1.0) + 0.10(1.0) + 0.10(1.0) = 0.934, well above the threshold.

Consider a customer table with fields such as Organization Name, Address, City, and State. When a user asks a question, applying the appropriate preprocessing steps before performing fuzzy matching ensures that the system returns the intended record.

Vector search, by design, retrieves a cluster of semantically similar records, which may include both correct and incorrect entities. This forces the LLM to infer which record is correct. In contrast, fuzzy matching returns only those rows that satisfy strict field-level constraints. The difference is decisive: Noisy or ambiguous context forces the LLM to guess which information is correct, while deterministic context eliminates that ambiguity. In practical enterprise environments, this approach can significantly reduce the risk of hallucinations by constraining the model to operate on verified, factual input. Figure 3 illustrates how domain-level preprocessing progressively narrows the search space to improve fuzzy-matching accuracy.

Fig 3: Fuzzy matching complemented by domain-level preprocessing improves accuracy by continually narrowing the search space



Tooling is Secondary to the Paradigm

Our thesis is about the paradigm and not any single library. Numerous fuzzy-matching tools and techniques exist, including Levenshtein distance, Jaro-Winkler, trigram overlap, and phonetic matching. The critical point is independent field scoring with explicit aggregation logic.

Different attributes demand different treatments: token-based similarity for names; edit distance for addresses; and exact matches for numeric IDs. Rapid Fuzz, for instance, provides functions such as token_set_ratio to compare names regardless of word order and partial_ratio to score substrings. However, the choice of algorithm is an engineering detail. What matters is that fields are scored independently and then aggregated. These weights and thresholds can be tuned by data teams based on acceptable tolerance for false positives and false negatives.

As the system is deterministic, its behavior can evolve through simple rule changes, such as adjusting weights, adding normalization rules, or incorporating domain-specific mappings. These changes do not require model retraining, re-indexing, or hidden behavior shifts. Over time, enterprises can refine their matching pipelines while maintaining complete control and transparency. The matching logic resides in application code or SQL rather than inside a neural network.

In summary, "fuzzy" refers to the approximate but deterministic string similarity. The tools vary, but the abstraction is the same: match fields exactly, when possible and approximately when necessary, using clear, auditable logic. This contrasts with vector-based retrieval, where similarity is determined in an opaque subspace and cannot be directly inspected or attributed.

Evidence

In enterprise settings, the retrieval mechanism can be an important part of the overall AI system, more significant than the LLM itself. The appeal of vector search lies in its versatility: a single retrieval mechanism can be applied to search for different types of content.

However, for structured enterprise data, such adaptability can come at the cost of field-level precision and control. The legal distinction between "Apple Inc." and "Apple Corps", for example, is not a matter of semantic suggestion



but identity distinction. Similarly, the medical difference between International Classification of Diseases, 10th Revision, Clinical Modification (ICD-10-CM) code I10 (indicating hypertension) and I11 (for hypertensive heart disease) represent distinct clinical classifications not to be conflated because they are semantically related.

The data available points to three important aspects of fuzzy matching:

Determinism: Provides the mathematically provable consistency required for regulatory compliance and business trust

Accuracy: Avoids the semantic drift that causes vector engines to conflate distinct but related entities, thereby eliminating a primary source of LLM hallucinations

Efficiency: Scales linearly on standard hardware, processing millions of records with sub-second latency and minimal infrastructure cost

Case studies

In our implementation for a supply chain client, the fuzzy-matching pipeline achieved query latencies of approximately 7–10 seconds when operating on datasets of nearly 45 million records while achieving an accuracy of 95%.

Another minimal viable product (MVP) implementation for a logistics client showed an accuracy of 93% on a dataset of nearly 15 million records, with latency under 5 seconds.

External Document © 2026 Infosys Limited

Conclusion

For the specific task of context engineering on structured data, we advocate a fuzzy-matching-first strategy. Vector search should be reserved for its true purpose, where semantic understanding is required, particularly for unstructured text and content where the meaning is not represented in a schema. For structured enterprise data, where the rows and columns define business reality, retrieval should prioritize determinism and precision.

The future of enterprise AI reliability depends upon applying syntactic precision with discipline. Real intelligence emerges from contextual awareness: understanding the nature of the data and applying the retrieval approach best suited to it.



About the Authors



Saket Kumar

Saket is a Specialist Programmer with the Strategic Technology Group at Infosys, with four years of experience in software engineering, and emerging technologies. With a keen interest in AI, he specializes in developing innovative, scalable solutions to complex business challenges. Committed to continuous learning and technical excellence, Saket focuses on leveraging AI and modern technologies to drive innovation, improve efficiency, and create meaningful impact.



Vanita

Vanita is a Digital Solution Specialist with the Data, Analytics, and AI unit at Infosys. She has more than 22 years of experience spanning multiple technologies and industry domains. She has a strong track record in leading cross-functional teams and delivering large-scale, high-impact technology solutions that translate into measurable business outcomes. In her current role as Technology Delivery Lead for GenAI product development, she leads end-to-end solution delivery, driving innovation and digital transformation across multiple client engagements. She combines deep technical expertise with strong delivery leadership and a commitment to continuous learning in an evolving technology landscape.



Princee Jain

Princee is a Senior Principal – Data Scientist in the Data, Analytics, and AI unit at Infosys, where she leads digital analytics and cognitive AI initiatives. With more than 19 years of experience, she leads a team of over 150 marketing and AI specialists, delivering solutions for complex client challenges across industry segments.



Dr. Prawaal

Dr. Prawaal is a Principal Data Scientist at Infosys with over 25 years of experience in the IT industry. He specializes in AI, natural language processing (NLP), and GenAI, and has published several research papers at leading conferences. He advises enterprises on AI strategies that enhance business value. An active writer and speaker on AI governance and risks, Dr. Prawaal also contributes to AI for Social Good (AI4SG) initiatives by developing AI models for tribal languages.

External Document © 2026 Infosys Limited

For more information, contact askus@infosys.com

Navigate your next

© 2026 Infosys Limited, Bengaluru, India. All Rights Reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this documentation nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/ or any named intellectual property rights holders under this document.