NLP Heritage & the Live Semantic Match Engine

How twenty years of work in entity enrichment, statistical NLP, and structured-document encoding shaped the design of Magellan's `semantic-match` edge function — and what actually runs in production today.

Three Historical Influences

Apache UIMA Typed annotation streams	**Stanford CoreNLP** Decomposable confidence	**TEI** Structured metadata encoding
Founder built an entity-enrichment pipeline on UIMA with Hidden Markov Models for NER while Chief Software Architect at ZH Computer. The lasting lesson: **annotations are infrastructure**, not free text.	Founder integrated CoreNLP into an eXist-db application for NER and annotation over structured XML. The lasting lesson: **confidence must be explainable** — named bases, not opaque numbers.	Founder applied the Text Encoding Initiative to structured document analysis. The lasting lesson: **markup is interpretation** — every tag is a versioned, machine-addressable claim.
→ **Carried into Magellan as** Layered attribute annotations: semantic type, domain, lineage, sensitivity, confidence — first-class records in the MetaModel.	→ **Carried into Magellan as** Every Semantic Match response includes confidence, a named algorithm class, and a rationale — never a single opaque score.	→ **Carried into Magellan as** Metadata as Code™: every schema, mapping, and rule is a versioned, schema-validated, diffable structure.

What Runs in Production Today

Magellan does not run UIMA, CoreNLP, HMMs, or hand-tuned weighted-signal formulas today. The live engine is the `semantic-match` Supabase Edge Function, version **1.0.0**, calling `google/gemini-2.5-flash` through the Lovable AI Gateway at temperature **0.1** with a strict JSON response schema.

Engine	`semantic-match` edge function v1.0.0 → `google/gemini-2.5-flash` via Lovable AI Gateway, temperature 0.1
Response shape	`{ confidence: 0–100 (int), algorithm: enum, rationale: string }`
Algorithm classes	`exact_match` `synonym_match` `abbreviation_match` `context_hierarchy` `domain_equivalence`
Decision policy	`decision = bucket(response.confidence, threshold[profile_ver])` Healthcare defaults: **AUTO-ACCEPT ≥ 85** · **REVIEW 65–84** · **REJECT < 65**. Threshold profiles are themselves versioned.
Identifiers	**ULID** (26-char, lexicographically sortable) for every decision, mapping, schema version, and audit-ledger entry. Never UUID.
Audit ledger	ULID-keyed, **SHA-256–chained**, append-only. Each entry references the previous entry's hash, producing a tamper-evident chain that is reproducible months or years later.

The Five Algorithm Classes (Worked Examples)

Class	What the model is asserting	Example
`exact_match`	Names normalize to the same term.	`patient_id` ↔ `patient_id`

`synonym_match`	Different terms for the same concept.	`subscriber_id` ↔ `member_number`
`abbreviation_match`	One name is a known abbreviation/expansion.	`dx_code` ↔ `diagnosis_code`
`context_hierarchy`	Names align by structural position in the schema.	`patient.address.zip` ↔ `member.mailing.postal_code`
`domain_equivalence`	Domain-specific equivalence (vocab, code system).	`ICD10_code` ↔ `diagnosis_code` (healthcare)

Three Questions to Ask Any Semantic-Mapping Vendor

1. **Are annotations first-class and layered, or free-text fields on a record?**
   Typed annotation streams scale. Flat tags do not.
2. **Is the confidence response decomposable, or a single opaque number?**
   Confidence + named algorithm class + rationale is auditable. An unlabeled embedding distance is not.
3. **Is metadata encoded in a versioned, schema-validated structure with a tamper-evident audit trail?**
   A ULID-keyed, SHA-256–chained ledger survives an audit. Spreadsheets do not.