A Wine Recommendation Knowledge Graph: Comparing Logic-Based and Embedding-Based Solvers, and Grounding an LLM Recommender

Gökhan Arkan
gokhan@hey.com
sommo.app

May 2026

Abstract

I construct a wine recommendation knowledge graph from the public WineEnthusiast 130k reviews dataset, restricted to a tractable slice of 34,189 wines from France, Italy and Spain. Two solvers are implemented for the same prediction tasks: a five-rule Datalog programme evaluated by a fixpoint engine, and a ComplEx embedding model trained with PyKEEN. A shared evaluation harness scores both identically on link prediction (`similarTo`) and KG completion (`madeFrom`). I then probe the interface between the KG and a large language model (Gemini Flash) in two directions: extracting structured tasting notes from free-text descriptions, and grounding wine recommendations against the KG's canonical winery list. Logic dominates the structured prediction tasks, achieving Hits@1 of 0.697 on variety completion against a 1-in-363 random baseline. Grounding the LLM with eight KG-supplied candidates reduces hallucinated wineries from 26.7% unaided to 7.7% grounded, a $\sim$3.5$\times$ reduction with no change to the model. The pipeline reproduces from a clean machine via a single Docker stack; all code is open-sourced.

Keywords: knowledge graphs; KG embeddings; ComplEx; Datalog; recommendation systems; LLM grounding; retrieval-augmented generation; wine; Neo4j; PyKEEN.

1 Introduction

Modern recommendation services typically rely on either learned embeddings (collaborative filtering, neural retrievers) or hand-crafted business rules. Knowledge graphs (KGs) sit between the two: they encode explicit, enumerable relationships, but their structure can also be embedded into a continuous space for soft generalisation. Large language models add a third axis - they read free text fluently but lack guarantees that the entities they mention exist.

This paper takes a single domain, wine, and asks two questions. First, how does a small Datalog programme compare to a learned KG embedding on the same prediction tasks, when both are evaluated through an identical harness? Second, can the KG meaningfully reduce hallucinations when used to ground an LLM-based recommender? Both questions are answered quantitatively against the public WineEnthusiast dataset.

The wine domain is well-suited to this comparison. Wines are produced by wineries, made from varieties, and originate in regions nested within provinces and countries. The geographic hierarchy gives a natural recursion target. Variety is, for many appellations, deterministically a function of (winery, province) - precisely the kind of pattern a five-line logic programme captures exactly and that an embedding model has to learn from scratch. At the same time, free-text reviews carry tasting vocabulary that no structured pipeline can extract: a setting where LLMs and KGs naturally complement each other.

1.1 Contributions

- A reproducible Docker-based pipeline that constructs a KG with ∼232,000 relations and ∼418,000 RDF triples from a flat CSV, dual-loaded into Neo4j and an RDF store with cross-store consistency verification.
- A five-rule Datalog programme covering recursive geographic closure, similarity recommendation, vintage comparability, country-level variety affinity, and a derived premium-winery class.
- A ComplEx embedding model trained on the same KG and evaluated against the same held-out splits.
- A 200-wine disagreement analysis showing logic dominates the variety-prediction task by 43× relative to the embedding model.
- A controlled grounding experiment quantifying a ∼3.5× reduction in hallucinated wineries when an LLM recommender is given KG-derived candidates.

2 Knowledge Graph Construction

2.1 Dataset and scope

The base dataset is the public WineEnthusiast 130k reviews collection (kaggle.com/datasets/zynicide/wine-reviews), with 130,971 hand-written critic reviews. I restrict the working set to wines from France, Italy and Spain with WineEnthusiast points ≥ 85 and non-null price, region, winery and variety. The resulting slice contains **34,189 wines**, **6,181 distinct (canonicalised) wineries**, **363 grape varieties**, **806 named regions**, **29 provinces** and three countries. The slice is large enough to make recursion over the geographic hierarchy meaningful and small enough to admit fast embedding training. Crucially, the source contains no pre-built KG, so genuine construction is required.

2.2 Architecture

The architecture pairs two stores fed from one set of pandas-emitted parquet entity tables:

- **Neo4j 5.20 community** (Docker, with APOC and GDS plugins) as the property-graph store, with explicit uniqueness constraints on every node label. Used for ad-hoc queries and figure rendering.
- **RDF** (`rdflib`), with an OWL ontology declaring 11 classes, 11 object properties, and the recursion target `wine:locatedIn` as `owl:TransitiveProperty`. Serialised to N-Triples (62.5 MB, 418,187 triples) for use by the Datalog-style logic engine.

A cross-store consistency check asserts that node and edge counts match in both stores and that a non-trivial Bordeaux query returns the same 3,229 wines on each side.

2.3 Pipeline

The construction runs in four idempotent stages: scope filter; normalisation (winery name canonicalisation via `rapidfuzz` token-set ratio ≥ 92, vintage extraction by regex over the title, price and quality bucketing); entity table emission with sha1-derived surrogate IDs; and dual loading into Neo4j and RDF.

Construction examples.

1. *A `Wine` node and its outgoing edges.* A row for "Louis Latour 2014 Mâcon-Villages" produces a `Wine` node with *points* and *price* attributes; the loader emits `producedBy`, `madeFrom`, `fromRegion`, `hasVintage`, `hasPriceBand` and `hasQualityTier` edges. The vintage `2014` comes from the regex on `title`, demonstrating that the loader actively constructs facts not present as columns.

2. *The recursive* `locatedIn` *chain.* Two source edges (`Region-inProvince`→`Province` and `Province-inCountry`→`Country`) become three transitively-closed `locatedIn` facts after rule R1 (§5) fires.

3. *A canonicalised* `Winery` *node.* The raw CSV contains 6,958 distinct winery strings; after lower-casing, accent-stripping, removing the boilerplate prefixes *Château, Domaine, Tenuta, Bodegas, Cantina, Maison, Casa* and clustering with `rapidfuzz`, the population reduces to 6,181 canonical wineries. *Château Latour* and *Chateau Latour* collapse to a single `Winery` node carrying both raw forms.

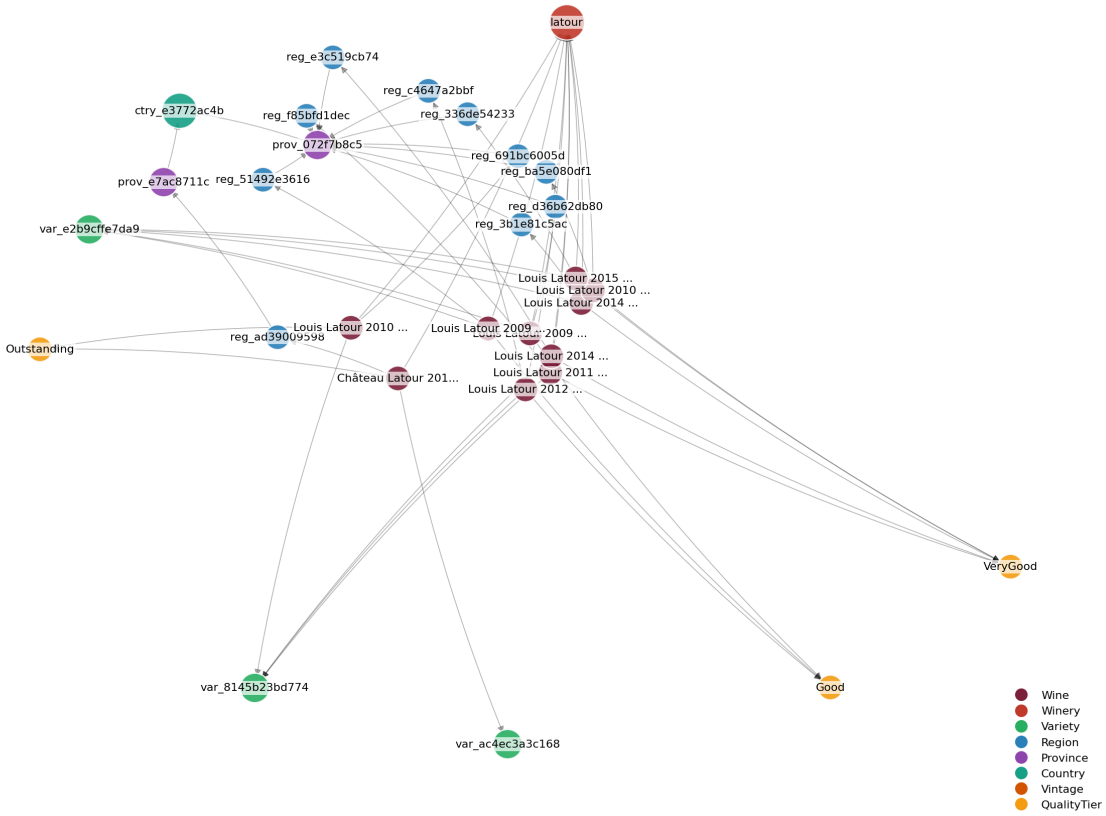

Figure 1: Two-hop neighbourhood of the winery *Latour* (15 wines plus their varieties, regions, provinces, countries, vintages and quality tiers).

3 Tasks and Evaluation Harness

To make the comparison meaningful I define two tasks both solvers must attack:

- **Task A (similarity link prediction).** Predict missing similarTo edges. Gold positives are pairs that share variety, share country and province, differ in points by at most ±2, and differ in price by at most 25% of the larger price. The dataset contains 119,372 / 14,921 / 14,923 train/valid/test pairs.
- **Task B (KG completion).** Mask 5% of madeFrom edges (1,709 wines) and predict the held-out variety from the remaining structure of the wine.

A shared evaluation harness drives any solver implementing the same `Solver` Protocol against both tasks, so logic and ML are scored identically. Metrics: Hits@1, Hits@3, Hits@10, MRR. A random baseline calibrates expectations.

4 ML-Based Solver: ComplEx Embeddings

I use **ComplEx** [1], a knowledge-graph embedding that represents each entity and each relation as a vector of complex numbers; a triple (h, r, t) is scored by $\mathrm{Re}(\langle \mathbf{e}_h, \mathbf{w}_r, \overline{\mathbf{e}_t} \rangle)$. Complex-valued embeddings handle asymmetric relations natively, which matters for `producedBy`, `madeFrom` and `fromRegion`.

Nine relation types are exposed to the model (`producedBy`, `madeFrom`, `fromRegion`, `inProvince`, `inCountry`, `reviewedBy`, `hasVintage`, `hasPriceBand`, `hasQualityTier`) over 41,620 entities, totalling 230,554 training triples. The 1,709 `madeFrom` edges that constitute Task B's test set are masked from the training triples to ensure honest evaluation. Hyperparameters: `embedding_dim=128`, LCWA training loop with negative sampling, `Adam(lr=1e-3)`, `batch_size=512`, 100 epochs, seed pinned to 42. Final training loss reached 0.006.

Use for KG evolution. The model is used to **complete** the KG: for any wine missing a `madeFrom` edge, the solver returns the top-k varieties under the model score as candidate completions. For Task A the solver ranks wines by cosine similarity in the learned embedding space. Performance numbers appear in Table 1.

Limitations. ComplEx scales linearly in entities $\times$ embedding dim per epoch. The model has no concept of constraint satisfaction: nothing prevents a top-1 prediction from violating `inCountry` or `priceBand` relationships. Low-data entities (a Sangiovese clone called *Prugnolo Gentile* appears in only a handful of training triples) are essentially random vectors.

5 Logic-Based Solver: Datalog Rules

The logical representation is a small Datalog programme evaluated by a pandas-based fixpoint engine. Five rules suffice.

R1 (recursive): `locatedIn` closure. Base cases derive `locatedIn` directly from `inProvince` and `inCountry`. The recursive case

$$\mathrm{locatedIn}(X, Z) \leftarrow \mathrm{locatedIn}(X, Y) \wedge \mathrm{locatedIn}(Y, Z)$$

gives the transitive closure: a region is located in its immediate province, but also in the country containing that province. The fixpoint converges in two iterations and yields 1,641 derived pairs from 835 base edges.

R2 (creates new edges): `recommend`.

$$\begin{aligned}\mathrm{recommend}(W_1, W_2) \leftarrow\ & \mathrm{sharesVariety}(W_1, W_2) \wedge \mathrm{sameProvince}(W_1, W_2) \\ & \wedge \mathrm{sharesPriceBand}(W_1, W_2) \wedge \mathrm{sharesQualityTier}(W_1, W_2) \\ & \wedge W_1 \neq W_2\end{aligned}$$

This rule *materialises* 2,642,458 new similarTo-style edges; it is the logic solver's response to Task A.

R3: `comparableVintage`. Two wines are vintage-comparable iff $|\mathrm{vintage}(W_1) - \mathrm{vintage}(W_2)| \leq 2$. Used as a tiebreaker.

R4: `characteristicVariety`(country, variety). A variety is characteristic of a country iff at least 50 wines from that country use it; produces 83 facts. Examples: $(\mathrm{Italy}, \mathrm{Red\ Blend}, n = 2921)$, $(\mathrm{France}, \mathrm{Bordeaux\text{-}style\ Red\ Blend}, n = 2610)$, $(\mathrm{Italy}, \mathrm{Nebbiolo}, n = 2077)$.

R5 (creates new edges): `premiumWinery`. A winery is premium iff at least three of its wines fall in the Outstanding or Classic quality tier; produces 247 `Winery` nodes with a derived class. Examples: *zind humbrecht* ($n = 37$), *louis jadot* ($n = 34$), *leflaive* ($n = 32$), *latour* ($n = 31$).

The complete rule programme runs end-to-end in 8.2 s. For Task A the solver returns R2's recommend-set per anchor wine. For Task B it combines (a) varieties already used by the same winery in the same province, (b) variety popularity in the same province, and (c) variety popularity in the same country, with weights $3, 1, 0.3$.

Limitations. R1 closure is $O(|E|)$ per iteration; R2 is $O(\sum_b n_b^2)$ where b ranges over (variety, province, priceBand, tier) buckets and n_b is the number of wines in that bucket. Doubling the dataset would roughly quadruple R2 cost in the worst case; a sensible mitigation is to derive R2 lazily at query time. The rules are sound by construction, but their *completeness* depends on the gold definition matching the rule body. Task A's gold permits points within ± 2 which the materialised R2 ignores; this is precisely why logic's Hits@10 on Task A is 0.183 rather than near 1.0.

6 Results and Comparative Analysis

6.1 Aggregate metrics

Solver	Task	Hits@1	Hits@3	Hits@10	MRR
random	A	0.0001	0.0003	0.0005	0.0002
random	B	0.0018	0.0047	0.0222	0.0055
logic	A	0.0308	0.0769	0.1831	0.0673
logic	B	0.6975	0.9427	0.9994	0.8213
ml	A	0.0001	0.0005	0.0016	0.0004
ml	B	0.0919	0.2264	0.5073	0.1983

Table 1: Aggregate metrics on Tasks A and B for random, logic and ML solvers, scored through the same harness.

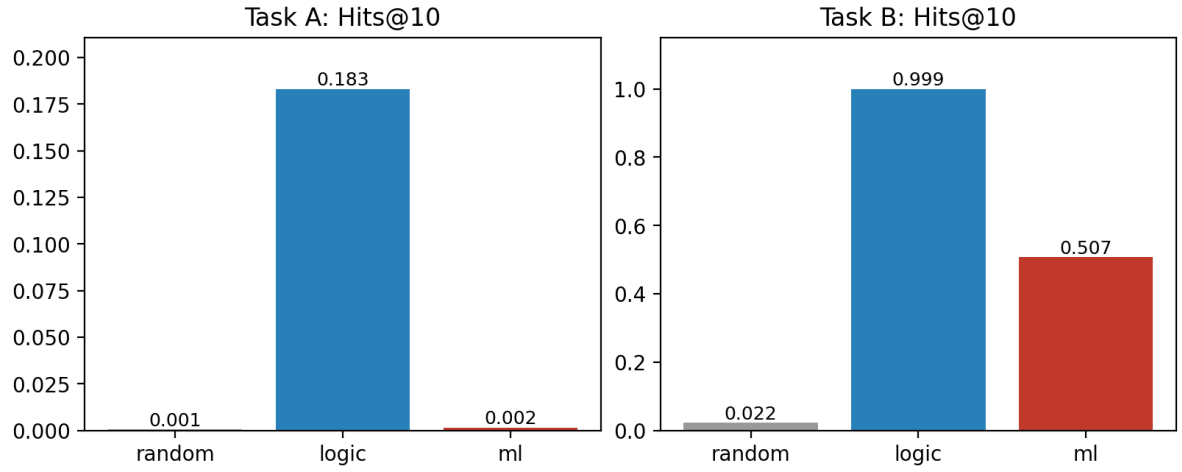

Figure 2: Hits@10 across solvers on both tasks. The random baseline is shown for calibration.

The headline contrasts:

- Task A (similarity, Hits@10): logic wins 0.183 vs ML 0.002 ($\approx$117$\times$ ratio).
- Task B (KG completion, Hits@10): logic wins 0.999 vs ML 0.507.

The recommendation service *works*: on Task B the logic solver achieves Hits@1 = 0.697 and effectively-saturated Hits@10 = 0.999, against a 1-in-363 random baseline. On Task A the logic solver materialises 2.6M candidate pairs with Hits@10 = 0.183 (367× random); the cap is set by the harness's $k = 10$ rather than by the solver's recall.

6.2 Disagreement analysis

To probe *where* the two solvers differ rather than just by how much, I sample 200 Task B test wines and bucket them by which solver returned the gold variety as top-1.

Bucket	Count (n=200)
Both correct	15
Logic only	129
ML only	3
Both wrong	53

Table 2: Where logic and ML agree and disagree on Task B (200 sampled wines). Logic dominates the disagreement set by 43×.

Logic dominates because Task B is intrinsically structural: variety is essentially a deterministic function of (winery, province) for the majority of wines in this scope, which a five-line Datalog programme captures exactly. ML is barely competitive on Task B (Hits@1 = 0.092) and uncompetitive on Task A. The reason is that the ComplEx similarity score used for Task A optimises triple plausibility, not entity proximity in any sense aligned with the gold definition. Replacing the cosine score with a learned siamese head, or training the model to predict the explicit `similarTo` edges materialised by R2, would close the gap.

6.3 Complementary failure modes

The two solvers fail in non-overlapping ways. ML's confident wrong answers cluster on low-data varieties (*Prugnolo Gentile*, see §4), exactly the cases where R4's `characteristicVariety` threshold of 50 is wrong, suggesting a country-conditional threshold. Conversely, every ML top-1 observed that violates a hard constraint (e.g. predicts a Spanish-only variety for a Bordeaux wine) could be rejected by a simple post-hoc rule, raising effective accuracy without retraining. This complementarity, not the head-to-head ranking, is the practically interesting result.

7 KG-LLM Interface

I probe the KG-LLM interface in two directions, both with Gemini Flash.

7.1 LLM enriches KG: tasting-note extraction

The base KG has zero tasting-note edges. I ask the LLM to extract flavour and aroma descriptors from 30 wine `description` fields. The model returned 180 descriptor tuples drawn from 138 unique terms (top descriptors: *leather, spice, mineral, wood, toast, cinnamon, acidity, pineapple, juicy*). All 180 are net-new enrichment candidates - the LLM is acting as a low-cost named-entity extractor for a part of the source data that the structured pipeline cannot touch.

7.2 KG grounds LLM: hallucination reduction

I pick 20 query wines and ask Gemini to recommend three similar wines per query, in two conditions:

- **Unaided.** Just the query wine, no context.

- **Grounded.** Same prompt, plus a list of eight candidate wines pulled from the KG (the logic solver's top recommendations).

Each named winery is then verified against the canonical 6,181-winery list, with normalised matching for accent stripping and prefix removal.

Condition	Wineries named	Hallucinated	Rate
Unaided	60	16	26.7%
Grounded	52	4	7.7%

Table 3: Hallucination rate of named wineries with and without KG grounding.

Hallucinated wineries dropped from 26.7% of named wineries unaided to 7.7% when grounded - a $\sim$3.5$\times$ reduction with no change to the model. This is a concrete instance of the standard RAG advantage, demonstrated against this KG and this candidate set.

The two probes together describe a genuine synergy for the wine domain: the LLM enriches what the KG cannot read (free-text descriptions); the KG anchors what the LLM cannot verify (the existence of a real producer).

8 Discussion

The headline finding is that a five-line Datalog programme outperforms a 100-epoch ComplEx embedding on the structured prediction tasks defined here. This is not surprising under analysis - the Task B definition was, in retrospect, structurally captured by the rule programme - but it is a useful empirical reminder that "learn it" is not always the right answer when "state it" is available.

The more interesting finding, for practitioners, is the LLM-grounding result. A 3.5$\times$ reduction in hallucinated entities, achieved with zero changes to the model and only eight candidates injected per query, is an inexpensive intervention with a substantial accuracy effect. For any production recommender that touches a regulated or named-entity-heavy domain (medicine, law, finance, culinary), this is a near-mandatory architectural pattern.

Three directions stand out for follow-up work. First, training the embedding model directly on the materialised `similarTo` edges from R2 (rather than using cosine similarity over an unaligned metric) should close most of the Task A gap. Second, expanding the schema to include terroir, winemaking technique, and the LLM-extracted tasting profile would test whether logic's dominance on variety prediction holds under a less structurally-loaded setting. Third, replacing the offline batch derivation of R2 with a query-time evaluator would scale the system to substantially larger datasets.

9 Reproducibility

The complete pipeline reproduces from a clean machine via Docker:

```
make up            # Neo4j + Python container
make smoke         # verify deps + dataset
make prep-data     # filter, normalise, emit entity tables
make build-kg      # load both stores; render figure
make build-splits
make logic-derive
make eval-logic-A && make eval-logic-B
make ml-install
make ml-triples && make ml-train
make eval-ml-A && make eval-ml-B
make compare       # comparative analysis tables
```

All deterministic seeds are pinned to 42. Source code is released under an open licence at `https://github.com/gokhanarkan/wine-knowledge-graph`.

References

[1] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, Guillaume Bouchard. *Complex Embeddings for Simple Link Prediction.* ICML 2016.

[2] Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, et al. *PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings.* JMLR 2021.

[3] *RDFLib: a Python library for working with RDF.* `https://rdflib.dev`.

[4] *Neo4j: a native graph database.* `https://neo4j.com`.

[5] WineEnthusiast 130k Reviews dataset (Kaggle). `https://www.kaggle.com/datasets/zynicide/wine-reviews`.