

From Plausible Hierarchies to Useful Taxonomies: Evaluating Agentic Harnesses on Customer Feedback

Prabhath Chellingi, Raviraja G, Viraj Bagal

prabhath@enterpret.com, raviraja@enterpret.com, viraj@enterpret.com

Abstract

Taxonomies are the symbolic representations through which AI systems organize evidence, aggregate patterns, and answer questions over large document collections. Over customer feedback, the category tree decides how every record is counted and routed, which problems get seen, and which team owns them. Agentic harnesses now make it easy to generate a plausible-looking hierarchy, and such trees are checked today with *generic*, individually scoped checks: each name fits its description, sits under the right parent, and stays distinct from its siblings. We ask a more operational question: when is a generated hierarchy actually useful as a production taxonomy? We build six taxonomies over two proprietary feedback corpora (1,940 and 5,000 records): for each corpus, a production reference and two repeated runs of the same harness under identical inputs. All six pass every generic naming and structure check, and a deeper product-coverage check even prefers the generated trees. Yet in every generated tree at least **97.7%** of leaf names merely restate an ancestor’s name (13.9% and 2.9% in the references), and in one, three of every four records fall under multiple top-level categories. We introduce two families of whole-tree metrics: **structural discriminators** test whether a tree’s shape was learned from the data or imposed by its generator; **team partitionability** tests whether branches split feedback into groups teams can own. Trees the generic checks rate as equally correct differ by **27 percentage points** in cross-branch leakage, and only one of the two beats a random split. Surface plausibility is an insufficient measure of taxonomy quality: evaluation must measure the *whole tree* as well as each node.

Keywords: taxonomy evaluation; customer feedback analysis; agentic harnesses; LLM-as-judge; agentic taxonomy construction; structural metrics; branch leakage; graph modularity

1 Introduction

A customer-feedback taxonomy has two jobs. It lets an organization ask “which product area causes the most dissatisfaction” and trust the count that comes back, and it lets a team scope its work down to the branch it owns. Neither job depends on any single node. Both depend on properties spread *across* the tree: how volume distributes, how labels differ from their neighbors, how cleanly a branch cuts away from the rest.

Most taxonomy evaluation in use today does not look at the tree this way. It runs *generic*, individually scoped checks, each inspecting a single node, edge, or sibling pair: does this name match this description, does this node belong under its parent, can these two siblings be told apart. These are legitimate questions, and an LLM judge answers them reliably. A few checks reach slightly further (sibling distinctness compares pairs, coverage checks compare nodes against documentation), but every verdict is still local, and the scores aggregate into a single pass-rate that cannot localize a systematic pattern. So a judge asked “is this node’s name coherent,” one item at a time, never notices that every name in the tree is built from the same template, or that one whole level never branches. Those facts only exist at the level of the corpus.

We measured this gap on real artifacts. Over two proprietary customer-feedback corpora from two deployed commercial products (a small sample of 1,940 records and a large sample of 5,000) we assembled six taxonomies – for each corpus, one production reference and two independent runs of the same off-the-shelf agentic construction harness under identical inputs and budgets – and scored all six with the generic checks in common use. Every artifact passes every check. Description completeness sits at 1.00 across the board, name-description match at 0.97–1.00. Judged node by node, all six look comparably professional.

The whole-tree measurements disagree, and the failures repeat across runs and corpora. Every generated taxonomy restates a full ancestor name verbatim in **97.7–100%** of its leaf names; the two references sit at 13.9% and 2.9%. In one generated taxonomy, every one of 158 parent nodes at one level has exactly one child – a wrapper-node pattern that long-tailed feedback data simply does not produce. Routing the actual feedback records through these trees, between a third and three-quarters of each generated taxonomy’s records land in more than one top-level product area, against a fifth for the references, and one generated taxonomy’s branch boundaries score *worse than a random partition* at keeping related issues together. The generic checks detect none of this. They score all six artifacts as comparably professional on naming and structure.

We make three contributions:

1. We show, with concrete numbers and taxonomy snippets, that a widely-used style of taxonomy evaluation (generic

LLM judging of names, descriptions, edges, and sibling pairs) systematically fails to detect templated-label collapse, generator-imposed branching caps, and cross-team fragmentation – the failure modes every agentic run in our study exhibits – because none of these is a property of any single node.

2. We introduce two deterministic, judge-free metric families that are computable from the taxonomy artifact (and, for one family, its record classifications) alone: **structural discriminators**, which detect when structure was imposed by a generator rather than learned from data, and **team partitionability**, which measures whether a taxonomy’s branches partition the record stream into cleanly separable groups, a structural prerequisite for assigning branches to distinct product teams and a downstream question existing evaluations do not ask.
3. We report both metric families over six independently constructed taxonomies on two corpora, show how to read each number against a concrete taxonomy snippet, and show that the new metrics separate artifacts the generic checks cannot: C-1 and C-2, two runs of the same harness that those checks rate as comparably correct (0.92–1.00 hierarchy correctness), differ by 27 percentage points in branch leakage and flip sign in co-occurrence modularity.

2 Related Work

2.1 Statistical and Corpus-Based Taxonomy Construction

Traditional taxonomy-construction research has used term extraction, semantic similarity, clustering, and hierarchical expansion. TaxoGen recursively clusters terms using adaptive embeddings to construct topical taxonomies from corpora (Zhang et al. 2018). HiExpan expands a seed taxonomy using corpus evidence and task guidance (Shen et al. 2018). TaxoExpan expands an existing taxonomy with new concepts using a position-enhanced graph neural network trained on self-supervised anchor relations (Shen et al. 2020). Topic-modeling methods such as latent Dirichlet allocation (Blei, Ng, and Jordan 2009) and embedding-based approaches such as BERTopic (Grootendorst 2022) are commonly used to organize unstructured records. None of these methods is evaluated on whether its output partitions the underlying records into groups separable enough to divide among teams, the question our second metric family targets.

2.2 Taxonomies for Customer Feedback

Customer reviews, surveys, app-store reviews, and support interactions have been organized using topic and aspect taxonomies. Amazon’s InsightNet constructs a multi-level taxonomy from customer reviews and associates granular topics with polarity and supporting verbatim (Mukku et al. 2024). Related work uses hierarchical classification to identify actionable topics in customer questions (Rana et al. 2023), and FeedbackMap organizes open-ended survey responses for analysis (Beeferman and Gillani 2023). FeClustRE organizes extracted app features into hierarchically clustered and semantically labeled structures (Tiessler and Motger 2025). A systematic review of user-feedback classifications found a

large, heterogeneous label space organized around sentiment, intention, user experience, and topic (dos Santos, Groen, and Villela 2019) – itself evidence that no single taxonomy follows automatically from a collection of feedback. This line of work evaluates classification accuracy or topic coherence, not whether the resulting structure is fit for organizational use.

2.3 LLM and Agentic Taxonomy Construction

Pretrained language models have been used to construct taxonomic trees by predicting parenthood relations and reconciling them into a consistent hierarchy (Chen, Lin, and Klein 2020), and instruction-tuned models have been applied to entity-set expansion, taxonomy expansion, and seed-guided taxonomy construction within a unified framework (Shen et al. 2024). In low-resource settings, self-supervised prompting with reinforcement-learning fine-tuning expands taxonomies from a handful of labeled examples (Mishra, Sudev, and Chakraborty 2024). Agentic systems go further: gathering information, invoking tools, evaluating intermediate artifacts, and revising outputs across a multi-step trajectory. Recent evidence shows agentic benchmark performance is itself noisy across repeated runs of the identical task (Bjarnason, Silva, and Monperrus 2026); our results extend this to taxonomy construction, and show that repeated-run variance concentrates in the corpus-level properties generic evaluation cannot see.

2.4 Taxonomy Quality Evaluation

Prior work has proposed quality attributes for taxonomies – comprehensiveness, robustness, explanatory power, extensibility, mutual exclusiveness – historically assessed by expert judgment; a compendium of such attributes, itself contributing explicit measurements for them, shows that evaluation is inherently tied to intended use (Unterkalmsteiner and Adbeen 2022). Formal ontology evaluation offers complementary machinery: OntoClean audits subsumption relations against metaproperties (Guarino and Welty 2002), an influential taxonomy-development method defines taxonomies over mutually exclusive and collectively exhaustive dimensions (Nickerson, Varshney, and Muntermann 2013), and surveys catalogue structural and application-based measures (Brank and Grobelnik 2005). TExEval scores extracted taxonomy edges against gold-standard hierarchies (Bordea et al. 2015; Bordea, Lefever, and Buitelaar 2016), TnT-LLM evaluates LLM-generated label taxonomies with LLM judges at scale (Wan et al. 2024), and hierarchical-classification research contributes hierarchy-aware measures (Kosmopoulos et al. 2014). **With one partial exception (mutual exclusiveness, which we revisit in the Discussion), every one of these instruments is scored by inspecting nodes, edges, or node pairs.** None asks whether a node’s label is informationally distinct from its ancestors’ labels in aggregate across the tree, whether a level’s branching factor reflects the data’s distribution or a generation artifact, or whether the tree’s branches partition the underlying record stream into cleanly separable groups. Computational linguistics distinguishes word types from word tokens (Jurafsky and Martin 2000), and their ratio

is a classical measure of a text’s vocabulary diversity; network science has long used modularity to measure whether a graph partition is meaningfully self-contained (Newman 2006). Neither tool has, to our knowledge, been applied to taxonomy artifacts. Goodhart’s Law names the eventual risk: a measure optimized in isolation stops indicating what it proxied (Manheim and Garrabrant 2019). The nearer problem we document is that node-level checks fail even to *detect* these structures arising by default, before any optimization pressure exists.

3 Experimental Setup

Every number in this paper comes from the protocol described in this section – the same six artifacts, the same two record sets, one of four scoring instruments per metric. That covers Table 1’s generic check scores, Table 2’s structural discriminators, Table 3’s partitionability metrics, and every snippet in the figures.

3.1 Corpora and Artifact Construction

We evaluate on customer-feedback corpora from two deployed commercial products, referred to throughout as the *small sample* (1,940 records) and the *large sample* (5,000 records); each corpus is accompanied by the product’s own documentation. Both are proprietary, and we do not characterize the products or data pipelines further. Over each corpus we assembled three taxonomies, six artifacts in total:

Live (one per corpus) is that product’s existing production taxonomy, built by the data owner’s in-house pipeline (itself agentic, not hand-curated) around deliberate structural checks: recurring-pattern checks, documentation cross-referencing, sibling-overlap resolution. The Live taxonomies are reference points, not controlled arms: they ran on different infrastructure without a matched budget, and because their pipeline was engineered around checks similar in spirit to our metrics, their favorable profiles are partly by construction. The confound-free evidence in this paper is therefore the within-harness contrast among the generated runs; the references show what the metrics report on trees shaped by deliberate structural checks.

C-1, C-2 (small sample) and **D-1, D-2** (large sample) are independent runs of the same off-the-shelf agentic coding harness (Codex (ope 2025), running GPT-5.6-Sol at max reasoning effort) under one fixed protocol: a fresh, isolated workspace per run; an identical task specification (reproduced in full in the Supplementary Material) describing the five-level schema, connectivity rules, a no-duplication rule. No run had access to the evaluator or to how it would be scored. Per the run manifests, the four runs consumed 5.8M–12.2M input tokens (96.9–98.1% served from cache; fresh input 159K–296K) and 73K–95K output tokens each, of which 30.9–37.4% were reasoning tokens.

Each artifact classifies its corpus’s records via `predictions.jsonl`, with every record assigned one or more complete L1-to-Subtheme paths in that artifact’s own tree. The records are identical across artifacts on the same corpus; the classifications are necessarily artifact-specific, since a path can only reference its own tree’s nodes – a

dependency we return to in Section 5 and Limitations. Substantive node counts (content-bearing nodes, excluding the system-managed generic/miscellaneous catch-alls that absorb unmatched records) range from 1,421 to 1,509 on the small sample and 4,700 to 4,829 on the large; the tables report rate-based statistics alongside raw counts so artifact size does not confound the comparison.

3.2 Evaluation Protocol

Four scoring instruments produce the numbers in this paper; the two metric families we introduce use no language model:

Generic checks (Table 1). Taxonomy quality is commonly assessed with generic checks scored by an LLM judge at scale (Wan et al. 2024), echoing the node-, edge-, and pair-level attributes prescribed in the taxonomy-evaluation literature (§2.4). The judge reads one node, one edge, or one sibling pair at a time and returns a verdict; the reported score is the fraction of items passing. We evaluate four widely used checks of this kind (judge: gpt-5.4-mini), covering naming and structure, the axes Sections 4 and 5 turn on:

- *Name-description match*: the fraction of nodes whose name the judge finds consistent with that node’s own description.
- *Description completeness*: the fraction of nodes carrying a non-empty, judged-adequate description.
- *Hierarchy correctness*: the fraction of parent-child edges the judge does *not* flag as a wrong-parent relation.
- *Sibling distinctness*: the fraction of sibling pairs (nodes sharing an immediate parent) the judge does *not* flag as overlapping or duplicative.

Each is judged one node, edge, or pair at a time – never against an ancestor, never against the tree as a whole – and this is the limitation Section 4 exploits.

Product-term coverage (Table 1, bottom row). A deeper check of the kind commonly used to validate that a taxonomy covers the product it describes: canonical product terms are extracted from the product’s documentation, and a stronger judge (gpt-5.5) decides per term whether some node in the taxonomy represents it; the score is the fraction of judged terms covered. It measures whether the taxonomy covers the product’s documented surface, not whether the tree is internally sound. We report it because it is the one check on which the generated artifacts *beat* the references (Section 4).

Structural discriminators (Table 2). Computed by a standalone script operating on `taxonomy.json` alone – no predictions, no LLM call. Ancestor containment, token redundancy, and type-token ratio are string and set operations over node names (§5); fan-out shape and cross-branch reuse are computed directly from the edge list.

Team partitionability (Table 3). Computed by a second standalone script operating on `taxonomy.json` and `predictions.jsonl` together: every record’s predicted paths are routed through each candidate tree to build the branch-assignment and leaf co-occurrence structures the leakage, purity, and modularity formulas (§5) are computed over.

Metric ¹	Small sample			Large sample		
	Live	C-1	C-2	Live	D-1	D-2
Name-desc. match ↑	0.98	0.97	0.98	0.98	1.00	1.00
Desc. completeness ↑	1.00	1.00	1.00	1.00	1.00	1.00
Hierarchy correctness ↑	0.94	1.00	0.92	0.93	0.94	0.99
Sibling distinctness ↑	0.94	0.88	0.94	0.95	0.91	0.99
Product-term coverage ↑	0.72	0.74	0.86	0.80	0.91	0.90

Table 1: The generic checks (top four rows) and the deep-judged product-term coverage check (bottom row), six artifacts across two corpora. Every artifact passes every check, with name-description match at 0.97–1.00 everywhere; the checks read as “all six are professional-grade.” On product-term coverage, **every generated artifact outcores its reference** (bold), on both corpora.

4 Existing Metrics: What They Test and Where They Break

4.1 Generic Checks in Common Use

Table 1 reports the four generic naming-and-structure checks, plus the deep-judged product-term coverage check, over the six artifacts of Section 3.1.

On its face, Table 1 is reassuring: every artifact clears every check, on both corpora, and the generated runs frequently outscore their references (D-1 and D-2 post perfect 1.00 naming scores against their reference’s 0.98). Product-term coverage goes further. Every generated artifact outcores its reference (0.74–0.86 against 0.72 on the small sample; 0.90–0.91 against 0.80 on the large), so by this check the generated trees look *better* grounded in the product than the production taxonomies. Anyone reading this table alone would conclude that the generated trees are interchangeable with the production references, or better.

4.2 What the Table Does Not Show

Figure 1 shows one leaf path from Live and from C-1 side by side, each column headed by that artifact’s measured ancestor-containment rate (the fraction of leaves whose name restates a full ancestor name; defined formally in Section 5), with the name-description score for that leaf annotated underneath. C-1 does pass name-description match (0.97, matching Table 1), so the check is not quietly catching the problem some other way. C-1’s L3 node (dashed border) is where the repeated phrase originates; every box below it re-pastes that same quoted L3 name, verbatim, into its own name before appending one new word. The repetition is not a presentation choice in the figure – the names are stored this way in `taxonomy.json`.

A judge scoring *one node at a time* – “does this name match this description” – has no way to see this. Both leaves have coherent names that match their own descriptions perfectly well in isolation, and the generic checks score both artifacts as comparably professional. The difference only appears when one looks at *all* of an artifact’s leaves at once and

¹↑ higher is better; every metric in this table is higher-is-better.

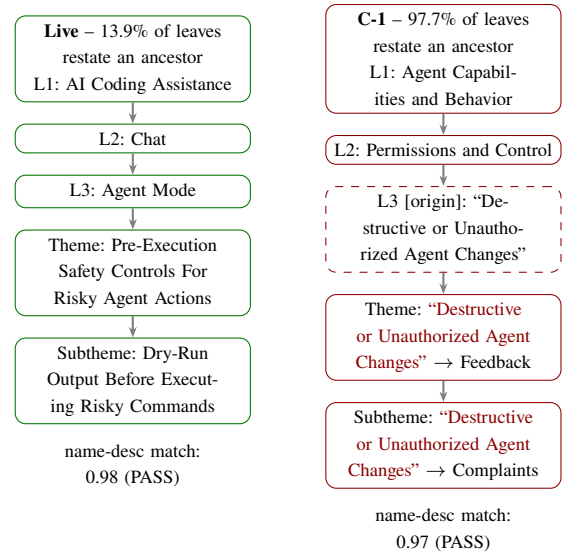


Figure 1: One leaf path each from Live and C-1, both real, verbatim node names; column headers give each artifact’s measured ancestor-containment rate. The phrase in quotation marks (set in red) is **copied character-for-character** from C-1’s L3 node into every descendant below it: Theme and Subtheme are each almost entirely copied text with a single new word appended. Name-description match passes both chains at 0.97–0.98 (Table 1), because it only ever compares a node to its own description, never to its ancestors.

asks what fraction of them are built this way. C-1 restates a matched ancestor’s name in **97.7% of its 867 leaves** (Table 2); C-2, not pictured, reaches **100% of its 599 leaves**, and the same pattern recurs independently on the large sample (97.7% and 100%). Nothing in Table 1 hints at any of this; the failure sits underneath clean 0.97–1.00 naming scores. The two references, for comparison, restate an ancestor’s name in only **13.9%** and **2.9%** of their leaves.

The same blindness recurs for branching structure. C-2’s L2-to-L3 fan-out is the most extreme version of a generator-imposed branching cap we observed: all 158 of its L2 parents have exactly one L3 child, a pure chain of wrapper nodes with no variation at all, while its widest node at any *other* level pair still reaches 26 children, against 31 for Live. Hierarchy correctness (Table 1) scores C-2 at 0.92, because each of its edges, judged individually, looks like a defensible parent-child relation. What that metric cannot see is that the *overall shape* of C-2’s branching was clamped by the generation process rather than shaped by where the feedback actually concentrates. Figure 2 draws the contrast.

What the two failures have in common is that they are true of the whole tree, not of any node the generic checks inspect – and both artifacts still score highly on every generic check in Table 1, while outscoring the reference on product-term coverage.

5 Two New Metric Families

We propose two metric families computed directly from the taxonomy artifact (and, for the second family, the record classifications) with no LLM judge and no gold taxonomy required, following the evaluation protocol of Section 3.2 over the same six artifacts and record sets as Table 1.

5.1 Structural Discriminators

These metrics ask whether the tree’s structure was learned from the data or imposed by the generator.

Label information gain. For a non-root node n with ancestor set $A(n)$ (system-managed catch-all nodes excluded) and $\text{tok}(x)$ the content words in node x ’s name (the words remaining after stopword removal), *ancestor containment* flags n when a full multi-word ancestor name is a substring of n ’s name; *token redundancy* is the continuous version:

$$\text{redund}(n) = \frac{|\text{tok}(n) \cap \text{tok}(A(n))|}{|\text{tok}(n)|}$$

$$\text{gain}(n) = 1 - \text{redund}(n)$$

High containment with low mean $\text{gain}(n)$ means child labels are restating rather than specializing their parents.

Lexical diversity. Per level L , the type-token ratio – distinct words (types) divided by total words (tokens) (Jurafsky and Martin 2000) – over node names at that level:

$$\text{TTR}_L = \frac{|\{\text{unique tokens in names at } L\}|}{|\{\text{total tokens in names at } L\}|}$$

A generator recycling a small set of template phrases collapses this several-fold relative to organically written labels. TTR generally declines as a corpus grows, and Live has the most leaves yet the highest TTR in our study, so the reported gaps are, if anything, conservative.

Fan-out shape. For each adjacent level pair, the distribution of $|\text{children}(p)|$ over parents p . Two summary statistics are reported per artifact: *max fan-out (any pair)*, the largest $|\text{children}(p)|$ observed across every parent and every level pair; and *cap-suspect level pairs*, the count of level pairs whose parent set P satisfies

$$|P| \geq 10 \quad \text{and} \quad \max_{p \in P} |\text{children}(p)| \leq 3$$

that is, at least ten parents and *no* parent with more than three children – a ceiling on the entire level, not a count of small parents (any long-tailed tree has many of those). Real feedback volume is long-tailed; a universal cap this narrow does not occur by chance. Figure 2 shows both patterns from the actual corpus.

Cross-branch reuse. $|\{n : |\text{parents}(n)| > 1\}|$, the count of nodes reachable through more than one parent. A taxonomy expressed as a strict tree with zero reuse anywhere is a signal that cross-cutting concerns were duplicated or never modeled, rather than shared. The small-sample reference, for example, shares the real node “Pre-Execution Safety Controls For Risky Agent Actions” under four L3 parents, one of its 48 shared nodes; in a zero-reuse tree the same concept must be authored once per branch, splitting its traffic

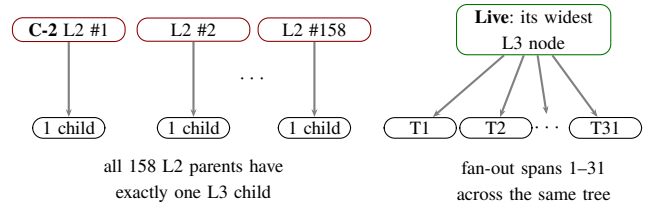


Figure 2: Fan-out shape (child counts real; node identities schematic). C-2’s L2→L3 level pair collapses to a pure wrap-per chain: **all 158 parents have exactly one child**. Live’s fan-out spans 1–31, the shape long-tailed feedback volume produces. Hierarchy correctness scores every individual edge in both trees as correct (Table 1) because it never looks at how many siblings a node has.

Metric ²	Small sample			Large sample		
	Live	C-1	C-2	Live	D-1	D-2
Containment ↓	13.9%	97.7%	100%	2.9%	97.7%	100%
Information gain ↑	0.76	0.33	0.34	0.81	0.74	0.58
Type-token ratio ↑	0.25	0.10	0.06	0.14	0.07	0.05
Cap-suspect pairs ↓	0	1	1	1	0	1
Cross-branch reuse ↑	48	0	0	73	0	0
Max fan-out ★	31	11	26	67	109	200

Table 2: Structural discriminators at the leaf level, six artifacts across two corpora. Every generated artifact collapses on containment, information gain, diversity, and reuse; max fan-out (★) is distributional context, not a quality score. The large-sample reference trips the cap-suspect flag once, at a genuinely narrow level pair (30 parents, none above 3 children) – the threshold false positive anticipated in Limitations – while D-2’s flagged pair is degenerate (451 parents, none above 2). Bold marks the headline finding. **The generic checks flagged none of this.**

across duplicates that no generic check can distinguish from an intentional, distinct pair.

Reading Table 2. Ancestor containment jumps from 13.9% and 2.9% to **97.7–100%** as soon as we leave the references, independently on both corpora: every generated artifact builds leaf names by concatenation. Table 1’s naming scores (0.97–1.00 for the generated artifacts) missed this entirely because concatenated names still describe their own node correctly; they just add nothing beyond what the ancestors already said. Type-token ratio collapses 2.5–4.4× on the small sample and 2.0–2.7× on the large. Notably, C-1 – the generated artifact with a perfect 1.00 hierarchy-correctness score – still shows a 2.5× diversity collapse, so edge-level correctness and label informativeness clearly measure different things. All four generated artifacts have zero cross-branch reuse, against 48 and 73 shared nodes in the references. None of these six rows exists among the generic checks, because none of them is a fact about one node.

²↑ higher is better; ↓ lower is better; ★ no single best direction – max fan-out reflects the shape of the underlying data, not quality directly (§5).

5.2 Team Partitionability

A product-feedback taxonomy’s top levels often double as an ownership map, with L1 (and L2) branches assigned to separate product teams. Whether any such assignment can work is, at bottom, a question of structural separability: do the branches partition the record stream into groups that stay apart? We answer it by routing each artifact’s record classifications through its tree. All three metrics below therefore depend on the classification as well as the tree itself (see Limitations).

Branch leakage. Let $\text{branch}(r)$ be the set of distinct branches record r ’s predicted paths touch, and $\text{count}(r, b)$ its number of paths landing in branch b :

$$\text{leakage} = \frac{|\{r : |\text{branch}(r)| > 1\}|}{|\{r\}|}$$

$$\text{purity} = \text{mean}_r \frac{\max_b \text{count}(r, b)}{\sum_b \text{count}(r, b)}$$

where *record purity* is the mean share of a record’s paths falling in its majority branch (cf. cluster purity in clustering evaluation (Manning, Raghavan, and Schütze 2008)). High leakage means a team owning one branch cannot see a full issue without involving a second team.

Co-occurrence modularity. Build a weighted graph over leaf nodes where w_{uv} is the number of records whose paths hit both leaves u, v ; with $m = \sum_{uv} w_{uv}$ and branch degree $k_b = \sum_{u:c(u)=b} \sum_v w_{uv}$, Newman modularity (Newman 2006) of the branch partition is

$$Q = \sum_b \left[\frac{e_b}{m} - \left(\frac{k_b}{2m} \right)^2 \right]$$

where e_b is the intra-branch edge weight. $Q > 0$ means branches hold together tighter than a random partition; $Q < 0$ means the branch boundaries actively cut against how issues co-occur in practice – worse than assigning branches at random. Figure 3 shows both failure modes on the same corpus.

Dividability. For branch b with children volumes $\{v_1, \dots, v_k\}$ and total volume $V_b = \sum_i v_i$:

$$H(b) = \frac{-\sum_i (v_i/V_b) \log(v_i/V_b)}{\log k}$$

$$\text{div} = \frac{\sum_b V_b H(b)}{\sum_b V_b}$$

$H(b) = 0$ for a single-child branch; $H(b) \rightarrow 1$ as volume splits evenly among children. This is a workload-balance diagnostic, not a quality signal by itself: an evenly split partition that is also fragmented (negative modularity) is not more useful, just more uniformly broken.

Reading Table 3. The references’ L1 leakage (19.8% and 19.3%, Table 3, row 1) means four in five records resolve inside a single product area on both corpora: a team owning that area sees whole issues, not fragments. Every generated artifact leaks $1.8\text{--}3.9\times$ more (34.7–76.4% against 19.3–19.8%,

³↑ higher is better; ↓ lower is better; * no single best direction – see the dividability definition.

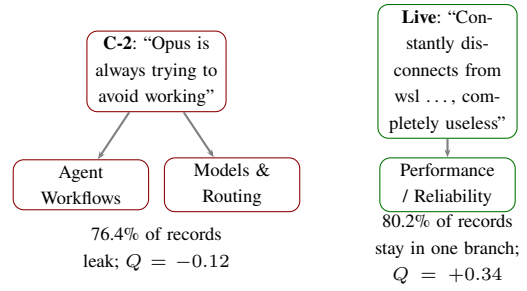


Figure 3: Branch leakage and co-occurrence modularity (Q , Newman modularity of the branch partition: positive means branches hold together tighter than a random split, negative means worse). Both records and both quotes are real, taken verbatim from the corpus and its predictions; percentages give each pattern’s share of that artifact’s records. Left: a C-2 record about a specific model’s behavior predicted onto both a model-routing branch and an agent-behavior branch, the majority pattern for C-2 (76.4% of records leak), consistent with $Q < 0$. Right: a Live record staying inside one branch, the majority pattern there (80.2% of records touch exactly one branch), consistent with $Q > 0$. Hierarchy correctness (Table 1) scores C-1 and C-2 within 0.08 of each other; this figure’s metrics separate them by **27 points of leakage and a sign flip**, because they are the only ones that route records through the tree at all.

same row); C-2’s 76.4% means three in four records span more than one team’s branch. Modularity sits $2.6\text{--}5.6\times$ below the references everywhere it stays positive, and turns *negative* for C-2 at both levels: its branch boundaries are worse than a random split at keeping co-occurring concerns together, meaning the L1 labels look like categories on paper but do not correspond to how issues actually cluster in the feedback stream. One mechanism is visible in the labels themselves: C-1’s L1 “Outcomes, Adoption and Market Perception” bundles unrelated concerns – community discussion, competitor comparisons, industry direction, overall product experience – that no single team could own, where the reference’s L1s (e.g. Performance / Reliability, Cloud Execution) are each one coherent area. Hierarchy correctness cannot flag this, because every child edge is individually defensible; the category itself is a combination of topics standing in for a product area. Dividability illustrates why it is reported as context: balance without containment is not separability, and C-2 combines the study’s worst leakage with an L2 dividability of exactly zero (every one of its 158 L2 parents has a single child, so there is nothing to divide). This is the sharpest disagreement with Table 1. Hierarchy correctness scores C-1 (1.00) within 0.08 of C-2 (0.92), but partitionability separates them by **27 points of leakage and a modularity sign flip**: the edges are individually well formed in both trees, yet the branches behave very differently. An evaluation limited to generic checks cannot recover this because it never constructs the object – the leaf co-occurrence graph – that the comparison depends on.

Metric ³	Small sample			Large sample		
	Live	C-1	C-2	Live	D-1	D-2
L1 leakage ↓	19.8%	49.8%	76.4%	19.3%	34.7%	39.3%
L1 record purity ↑	0.92	0.78	0.61	0.93	0.83	0.81
L1 modularity ↑	0.34	0.12	-0.12	0.43	0.08	0.16
L1 dividability ★	0.32	0.87	0.71	0.70	0.59	0.62
L2 leakage ↓	22.0%	68.1%	79.4%	22.6%	38.1%	43.7%
L2 record purity ↑	0.91	0.58	0.60	0.91	0.79	0.77
L2 modularity ↑	0.31	0.01	-0.04	0.44	0.05	0.13
L2 dividability ★	0.56	0.74	0.00	0.61	0.72	0.75

Table 3: Team partitionability, six artifacts, each scored over its own classification of its corpus’s records. On both corpora the references leak about a fifth of records across top-level branches; **every generated artifact leaks between a third and three-quarters**. Bold marks the C-1 vs. C-2 contrast discussed in the text. Dividability (★) is workload-balance context, not a quality score.

6 Discussion

Every failure class in Sections 4–5 is a fact about the *ensemble* of nodes or records, not about any node judged in isolation. Ancestor-containment collapse lives in the relationship between every leaf and its ancestors; branching caps in the distribution of children across dozens of parents; leakage and modularity in how thousands of record-to-path assignments co-occur across the corpus. An LLM judge invoked per node, per edge, or per sibling pair is architecturally unable to compute any of these – not because the judge is weak, but because the question is not local. Our two families are cheap for the same reason: they need no judge, only closed-form statistics over the artifact and its classifications, and every number in Tables 2 and 3 was produced without an LLM call.

The metrics also settle the mutual-exclusiveness question deferred from Section 2.4. Mutual exclusiveness (Nickerson, Varshney, and Muntermann 2013) is a node-level design prescription; branch leakage operationalizes it at the record level, where a record spanning two branches is a countable violation. Reuse and leakage are therefore complementary rather than contradictory: reuse asks whether a cross-cutting concept is represented once and shared, leakage asks how much of the record stream such concepts affect. A well-shaped taxonomy has nonzero reuse and low leakage (the references: 48 and 73 shared nodes, about 20% leakage). The generated artifacts combine zero reuse with 35–76% leakage – cross-cutting concepts were neither shared nor contained.

The four runs never saw the evaluator, so nothing was optimized against it: templated labels, capped branching, and per-concept branch minting are what unconstrained generation produces by default, and generic scoring fails to detect them. A process optimized *against* such checks would, per Goodhart’s Law, drift toward them. More compute does not help either – the heaviest run on each corpus (C-2 at 9.5M input tokens, D-2 at 12.2M) is also its corpus’s worst on leakage and diversity. Coverage checks turn the missing penalty into an active reward, since minting a node per documented term is the cheapest way to raise coverage and manufactures

these same patterns; the tree with the study’s worst leakage (C-2) posts the highest small-sample product-term coverage (0.86 vs. 0.72), and the pattern repeats on the large sample (0.90–0.91 vs. 0.80). A team relying on generic checks to gate taxonomy quality should add a corpus-level pass, whether the two families here or equivalents, before trusting the result.

7 Limitations and Future Work

All generated runs use one harness, so harness diversity is untested, though the mechanism (corpus-level facts invisible to generic judging) is not harness-specific. The cap-suspect threshold can false-positive on genuinely narrow branching, and does once, on the large-sample reference (Table 2). Type-token ratio is size-sensitive across very different scales. Leakage, purity, and modularity depend on each artifact’s own record classification, so they evaluate the taxonomy-plus-classifier system. Neither family is validated against an external usefulness signal such as team velocity; we claim they detect failures existing metrics miss here, not that they suffice alone. Every metric in Table 1 is LLM-judged (gpt-5.4-mini; gpt-5.5 for coverage) without a human-agreement study (Artstein and Poesio 2008), and such judges show self-preference (Zheng et al. 2023; Panickssery, Bowman, and Feng 2024) and order sensitivity (Shi et al. 2025); our judge-free families are unaffected. Future work includes temporal stability, a second harness family, tuned thresholds, and insertion-time construction gates.

Availability and ethics. The corpora, artifacts, and documentation are proprietary customer data and are not released, so Tables 2 and 3 cannot be independently recomputed. Every metric is fully specified in Section 5’s formulas and Section 3’s protocol, and the complete agent task specification and launch prompt are given in the Supplementary Material, so both metric families and the construction protocol can be reimplemented from the paper alone and applied unchanged to any taxonomy artifact meeting the same input contract.

8 Conclusion

A taxonomy can clear every LLM-judged generic check in common use, and beat its production reference on product-term coverage, while every leaf label restates its ancestors verbatim, its branching is generator-capped, and three-quarters of its records leak across the product-area boundaries it exists to enforce. We observed this in all four runs of the harness we tested, independently on two real corpora: the same failure modes, every run. Structural discriminators and team partitionability recover the failures generic judging cannot see, deterministically and with no LLM judge. The metric a team is missing is not a better generic judge; it is a metric that looks at the tree, and the records routed through it, as a whole.

References

2025. Codex in ChatGPT | AI Coding Agents for Software Engineering — openai.com. <https://openai.com/codex>. [Accessed 29-07-2026].

- Artstein, R.; and Poesio, M. 2008. Inter-coder agreement for computational linguistics. *Comput. Linguist.*, 34(4): 555–596.
- Beeferman, D.; and Gillani, N. 2023. FeedbackMap: A Tool for Making Sense of Open-ended Survey Responses. *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing*.
- Bjarnason, B. H.; Silva, A.; and Monperrus, M. 2026. On Randomness in Agentic Evals. *arXiv:2602.07150*.
- Blei, D. M.; Ng, A.; and Jordan, M. I. 2009. Latent Dirichlet Allocation.
- Bordea, G.; Buitelaar, P.; Faralli, S.; and Navigli, R. 2015. SemEval-2015 Task 17: Taxonomy Extraction Evaluation (TExEval). In *International Workshop on Semantic Evaluation*.
- Bordea, G.; Lefever, E.; and Buitelaar, P. 2016. SemEval-2016 Task 13: Taxonomy Extraction Evaluation (TExEval-2). In *International Workshop on Semantic Evaluation*.
- Brank, J.; and Grobelnik, M. 2005. A SURVEY OF ONTOLOGY EVALUATION TECHNIQUES.
- Chen, C.; Lin, K.; and Klein, D. 2020. Constructing Taxonomies from Pretrained Language Models. In *North American Chapter of the Association for Computational Linguistics*.
- dos Santos, R. I.; Groen, E. C.; and Villela, K. 2019. A Taxonomy for User Feedback Classifications. In *REFSQ Workshops*.
- Grootendorst, M. R. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *ArXiv*, abs/2203.05794.
- Guarino, N.; and Welty, C. 2002. Evaluating ontological decisions with OntoClean. *Commun. ACM*, 45: 61–65.
- Jurafsky, D.; and Martin, J. H. 2000. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.
- Kosmopoulos, A.; Partalas, I.; Gaussier, E.; Paliouras, G.; and Androutsopoulos, I. 2014. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29(3): 820–865.
- Manheim, D.; and Garra-brant, S. 2019. Categorizing Variants of Goodhart’s Law. *arXiv:1803.04585*.
- Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.
- Mishra, S.; Sudev, U.; and Chakraborty, T. 2024. FLAME: Self-Supervised Low-Resource Taxonomy Expansion Using Large Language Models. *ACM Transactions on Intelligent Systems and Technology*.
- Mukku, S. S.; Soni, M.; Aggarwal, C.; Rana, J.; Yenigalla, P.; Patange, R.; and Mohan, S. 2024. InsightNet : Structured Insight Mining from Customer Feedback. In *Conference on Empirical Methods in Natural Language Processing*.
- Newman, M. E. J. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23): 8577–8582.
- Nickerson, R. C.; Varshney, U.; and Muntermann, J. 2013. A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22: 336–359.
- Panickssery, A.; Bowman, S. R.; and Feng, S. 2024. LLM Evaluators Recognize and Favor Their Own Generations. *ArXiv*, abs/2404.13076.
- Rana, J.; Yenigalla, P.; Aggarwal, C.; Mukku, S. S.; Soni, M.; and Patange, R. 2023. Weakly supervised hierarchical multi-task classification of customer questions. In *Annual Meeting of the Association for Computational Linguistics*.
- Shen, J.; Shen, Z.; Xiong, C.; Wang, C.; Wang, K.; and Han, J. 2020. TaxoExpan: Self-supervised Taxonomy Expansion with Position-Enhanced Graph Neural Network. *Proceedings of The Web Conference 2020*.
- Shen, J.; Wu, Z.; Lei, D.; Zhang, C.; Ren, X.; Vanni, M. T.; Sadler, B. M.; and Han, J. 2018. HiExpan: Task-Guided Taxonomy Construction by Hierarchical Tree Expansion. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
- Shen, Y.; Zhang, Y.; Zhang, Y.; and Han, J. 2024. A Unified Taxonomy-Guided Instruction Tuning Framework for Entity Set Expansion and Taxonomy Expansion. *ArXiv*, abs/2402.13405.
- Shi, L.; Ma, C.; Liang, W.; Diao, X.; Ma, W.; and Vosoughi, S. 2025. Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge. *arXiv:2406.07791*.
- Tiessler, M.; and Motger, Q. 2025. FeClustRE: Hierarchical Clustering and Semantic Tagging of App Features from User Reviews. *ArXiv*, abs/2510.18799.
- Unterka-msteiner, M.; and Adbeen, W. 2022. A compendium and evaluation of taxonomy quality attributes. *Expert Systems*, 40.
- Wan, M.; Safavi, T.; Jauhar, S. K.; Kim, Y.; Counts, S.; Neville, J.; Suri, S.; Shah, C.; White, R. W.; Yang, L.; Andersen, R.; Buscher, G.; Joshi, D.; and Rangan, N. 2024. TnT-LLM: Text Mining at Scale with Large Language Models. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.
- Zhang, C.; Tao, F.; Chen, X.; Shen, J.; Jiang, M.; Sadler, B.; Vanni, M.; and Han, J. 2018. TaxoGen: Unsupervised Topic Taxonomy Construction by Adaptive Term Embedding and Clustering. *arXiv:1812.09551*.
- Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *ArXiv*, abs/2306.05685.