

CLUSTERS ARE PROPOSALS: DISCOVERING HIDDEN SUBCATEGORIES IN PRE-CATEGORIZED TEXT

Anonymous authors

Paper under double-blind review

ABSTRACT

Enterprise call-center analytics often begins with broad topic groups, while the finer distinctions needed for actionable analysis remain hidden. Discovering these distinctions is challenging when interactions within each group are semantically similar, category frequencies are highly uneven, and the number of categories is unknown. The challenge is sharpest when meaningful subcategories are rare: an infrequent but important customer issue may represent only 1% of a broad category’s traffic. Recent LLM pipelines discover categories by proposing candidate definitions from a sample of documents and then assigning every document against them. We argue that these pipelines lose hidden subcategories before an LLM ever sees them: proposals are drawn from random samples, and small candidates are later pruned, absorbed into coarse labels, or removed by minimum-size rules. We introduce *clusters as proposals*, which changes where proposals come from. Each category is recursively over-fragmented into size-bounded clusters, so every dense region, however small, reaches the LLM as a candidate subcategory with a written definition. Candidates are merged only when an LLM confirms, from their definitions and supporting texts, that they describe the same subcategory, and every document is then labeled against the final definitions. The design rests on an asymmetry: a redundant proposal costs one merge, but a missing proposal cannot be recovered later. On 10 high-volume categories of a production telecommunications call-center corpus, cluster proposals yield 234 subcategories, compared with 111 from flat clustering and 203 from LLM-first discovery, with higher within-subcategory coherence than both baselines in all 10 categories. In 8 of those categories, at least one discovered subcategory holds less than 2% of that category’s calls—the hidden issues the method is designed to surface.

1 INTRODUCTION

A large telecommunications provider routes tens of thousands of customer calls a day into broad categories such as *Make a Payment* or *Activate a Device*. The categories work well for routing and staffing, but are too broad to guide operational decisions—*there are twenty different reasons customers call about payments; which ones should we act on first?* A single broad cluster labeled “payment issues” may combine calls about autopay failures, unexpected charges, and payment-app errors—distinct problems that require different responses. Manual review cannot answer it at scale, because human reviewers evaluate far less than 1% of calls (Stepanov et al., 2015). The answer that matters most is also often the most hidden. A payment-app regression introduced last week, or eSIM provisioning failing on one device model, may account for thirty calls out of three thousand. By the time such an issue is frequent enough to be obvious, it has been costing customers and agents for weeks.

Large language models have made this kind of discovery practical, and recent work has converged on a sound template that we call *propose-then-assign* (Pham et al., 2024; Wan et al., 2024; Wang et al., 2023; Lam et al., 2024). An LLM reads documents and proposes candidate categories with natural-language definitions; the candidates are refined; and every document is then labeled against the final definitions rather than by where it happens to fall in an embedding space. Separating

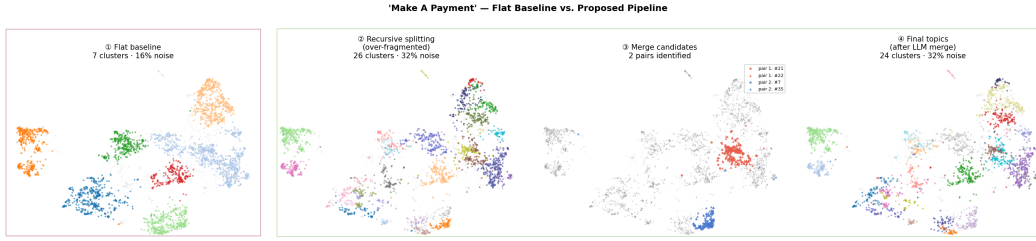


Figure 1: Our approach on a single call category. Flat clustering (left) returns a few broad groups. Our approach (right): Recursive over-fragmentation turns every dense region into a proposal → category-centered geometric candidates and LLM merges redundant proposals (2 pairs identified in the over-fragmented clusters) → LLM defines each proposal and documents are labeled against the final definitions.

discovery from assignment makes the output interpretable and every label auditable. We adopt this template.

Its weak point is the first step. An LLM can only propose what it is shown, and in current pipelines it is shown a random sample of documents. The arithmetic is unforgiving. A subcategory that holds 1% of a category has a 13% chance of being entirely absent from a random sample of 200 documents, and a 68% chance of appearing fewer than three times—too few for any proposer to recognize a pattern. Later stages then compound the loss. Candidates generated too rarely are pruned as noise (Pham et al., 2024), fine distinctions are folded into a small set of coarse labels (Wan et al., 2024), and clusters below a minimum size are dropped outright (Tamkin et al., 2024). Each of these choices is reasonable where it was made: it protects label quality, cost, or privacy. Together, they mean that hidden subcategories are removed before an LLM ever sees them.

The problem is sharpest in exactly our setting, where discovery runs inside a category that is already semantically narrow. Every document in *Make a Payment* shares a strong common direction in embedding space, and the cosine similarity between any two subcategory centroids is typically above 0.85. A hidden subcategory is therefore not a separate island but a small bump on a dense surface. A random sample rarely lands on it, and a single pass of density clustering tuned for the main subcategories absorbs it into a larger neighbor.

We propose to change where proposals come from. Our central idea is to treat clusters as *proposals* rather than answers. Instead of asking a clustering to be *correct*, we ask it to be *exhaustive*. Each category is recursively split without a fixed depth cap until each branch meets the size threshold or further splitting makes no progress, and every resulting cluster is shown to the LLM, which writes a definition for it. This deliberately produces too many candidates, and that is the point, because the two possible errors are not symmetric: a redundant proposal costs one merge, while a missing proposal cannot be recovered by any later stage. Consolidation removes the redundancy. Candidates are compared after subtracting the category centroid, which removes the shared direction that makes every subcategory look alike; an LLM confirms each merge from the candidates’ definitions and supporting texts; and every document is labeled against the surviving definitions. Figure 1 shows the contrast on a single category: in *Make a Payment*, flat clustering returns 7 broad groups, while our approach identifies 24 distinct subcategories; even the LLM-first baseline, which has access to the same frontier model, recovers only 18.

We study the approach on ten high-volume categories of a production call-center corpus without a reference taxonomy. Our contributions:

1. **Diagnosis.** Standard clustering metrics are structurally blind to rare subcategories: a latent topic can be absorbed into a larger neighbor without moving any clustering score, and sampling-based proposers miss small groups before an LLM ever sees them (Appendix C.1).
2. **Method.** *Clusters as proposals*: every cluster above a size threshold is recursively split regardless of its coherence, so every dense region—however small—reaches the LLM as a candidate with a written definition. Consolidation is governed by geometric redundancy

Table 1: How existing discovery methods generate candidates and what happens to a subcategory absorbed by a broader group. *Coarse*: whether the method runs inside existing coarse labels without needing labeled examples of the fine classes.

Method	Candidates come from	Documents assigned by	Fate of a small subcategory	Coarse
BERTopic	/ One density clustering of the corpus	Cluster membership	Absorbed by a neighbor or marked noise	–
Top2Vec				
ClusterLLM	Clustering with LLM-refined embeddings	Cluster membership	No specific mechanism	–
Viswanathan et al.	k -means, k given	Membership, LLM-corrected	No specific mechanism	–
Clio	k -means over conversation summaries	Cluster membership	Removed below a minimum size (privacy)	–
DeepAligned	k -means; K estimated	Cluster membership	Estimator drops clusters smaller than N/K'	–
TopicGPT	Random sample, one document at a time	LLM against definitions	Pruned if generated too rarely	partial
TnT-LLM	Random minibatches of summaries	LLM labels, distilled classifier	Folded into a small label set	–
GoalEx	Random subsets; K fixed	LLM check per document and description	Found only if sampled or left uncovered	–
LLooM	One HDBSCAN pass over a sample	LLM scoring against criteria	Found only if sampled	–
Ours	Recursive, size-triggered clusters of <i>all</i> docs in a category	LLM against definitions	Proposed whenever it forms a dense region of $\geq m$ docs	✓

(delta-vector projection), not by cluster size, so rare topics are never pruned for being small (§4).

- Evidence from deployment.** On our production corpus, cluster proposals recover 234 subcategories—against 111 from flat clustering and 203 from LLM-first discovery—with higher within-subcategory coherence than both baselines in all ten categories.

2 RELATED WORK

For hidden subcategories, two questions decide the outcome of any discovery method: *where do candidate categories come from*, and *what happens to the small ones*? We organize prior work around these two questions. Table 1 summarizes the answers; per-method details follow.

Clusters as the answer. Classical topic models treat the partition as the result. LDA (Blei et al., 2003b) and its hierarchical extension (Blei et al., 2003a) fix topic structure in advance; BERTopic (Grootendorst, 2022) and Top2Vec (Angelov, 2020) cluster embeddings via UMAP (McInnes et al., 2018) and HDBSCAN (Campello et al., 2013), so granularity follows from a single density threshold. LLM-guided variants refine the geometry but keep the partition as the answer: ClusterLLM (Zhang et al., 2023) tunes embeddings on LLM triplet judgments, Viswanathan et al. (2024) correct k -means with pairwise constraints, and Clio (Tamkin et al., 2024) removes groups below a privacy threshold. In every case a hidden subcategory survives only if the partition isolates it. We use the same machinery but give the partition a different job: generating candidates, not answers, tuned to over-fragment rather than to be right.

Propose-then-assign with LLMs. The closest line of work separates discovery from assignment. TopicGPT (Pham et al., 2024) shows sampled documents to an LLM one at a time, merging near-duplicates and pruning rare topics. TnT-LLM (Wan et al., 2024) revises a taxonomy over random minibatches of summaries and distills LLM labels into classifiers. GoalEx (Wang et al., 2023) proposes descriptions from random subsets and re-proposes from uncovered documents. LLooM (Lam et al., 2024) clusters LLM-distilled summaries, proposes concepts per cluster, and loops over outliers. We share this separation and merge candidates with LLM confirmation as in TopicGPT, but differ at the proposal stage: all four systems draw proposals from random samples, so a hidden subcategory may never reach the proposer. Our proposals come from an exhaustive partition, so every dense region is proposed by construction rather than by chance.

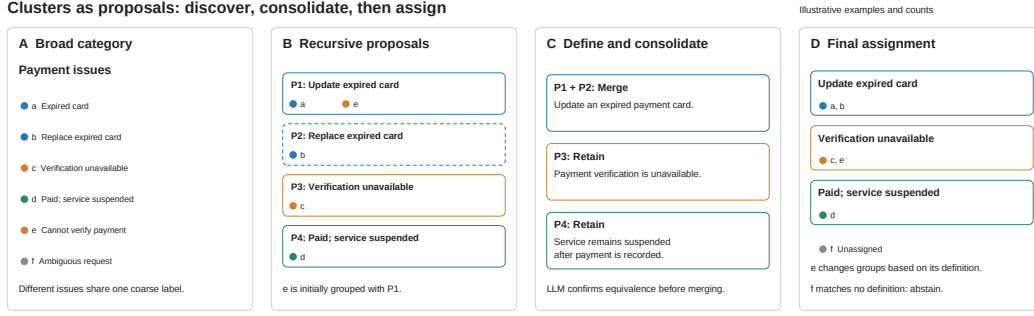


Figure 2: Overview of clusters as proposals. Recursive fragmentation generates candidate subcategories within a coarse category, delta-vector merging proposes candidates. An LLM confirms merges between equivalent proposals and defines each candidate. Documents are then assigned against the consolidated definitions, with abstention when none applies. The illustration shows redundant fragments and distinct issues retained after consolidation; examples and counts are schematic.

Category discovery with an unknown number of classes. New intent discovery (Zhang et al., 2021) and generalized category discovery (Vaze et al., 2022) find novel classes in unlabeled data, estimating the class count when unknown. They differ from our setting in two ways: they require labeled examples of known fine-grained classes to define granularity (we have only coarse categories), and their treatment of small classes works against hidden subcategories—DeepAligned removes clusters smaller than N/K' by construction. Recent variants handle class imbalance (Zhang et al., 2024; Bai et al., 2023) or exploit coarse-to-fine taxonomies (He et al., 2025), but all still require labeled fine classes.

Over-segment, then merge. Deliberately over-segmenting and then consolidating is established in vision: superpixel methods produce small regions that later stages group into objects (Achanta et al., 2012), and deep clustering trains an auxiliary head with excess clusters, discarded at test time (Ji et al., 2019). We bring this principle to LLM-based category discovery, where over-segmentation ensures hidden subcategories reach the proposer and consolidation reasons over natural-language definitions.

3 PROBLEM FORMULATION

We study fine-grained topic discovery within existing coarse categories. Given documents grouped by a coarse label, the task is to discover subcategories without labeled fine-grained examples or a specified number of topics. The output is a set of subcategory names and definitions, together with document assignments; documents that match no definition may remain unassigned.

We call a subcategory *hidden* relative to a discovery method when that method absorbs a meaningful distinction into a broader topic without representing it separately. Hidden subcategories may be common or rare. The objective is to expose useful distinctions while limiting redundant topics and preserving document coverage.

4 METHOD

4.1 STAGE 1: PROPOSAL GENERATION VIA RECURSIVE OVER-FRAGMENTATION

Within each coarse category, we construct an intentionally over-fragmented taxonomy by applying UMAP followed by HDBSCAN and recursively splitting any cluster exceeding a size threshold (τ_{split}). The threshold is a recursion trigger, not a guaranteed final size bound: with as low as 20, HDBSCAN will nearly always find density sub-structure in a cluster of several hundred points, producing further splits. This is by design: it ensures that hidden subcategories—those a single clustering pass would absorb into larger groups—reach the LLM as distinct candidates. Each result-

ing leaf is a proposal—a candidate sub-topic, not a final assignment. There is no fixed depth cap; each branch stops when its cluster meets the size threshold or further splitting makes no progress. Documents that do not fall into any cluster are considered noise and are kept outside the proposal leaves. Appendix A.1 gives the recursive procedure.

4.2 STAGE 2: DELTA-VECTOR CONSOLIDATION AND LLM DEFINITION

Over-fragmentation produces redundant proposals—fragments of the same subcategory split across recursion levels. To identify candidates for merging, we center proposal centroids μ_i, μ_j on the coarse-category mean μ_c and compute

$$\text{sim}_\delta(i, j) = \frac{(\mu_i - \mu_c)^\top (\mu_j - \mu_c)}{\|\mu_i - \mu_c\| \|\mu_j - \mu_c\|}. \quad (1)$$

Subtracting the category centroid removes the shared direction that makes every subcategory look alike; what remains is the direction in which each proposal specializes. Pairs whose centered similarity exceeds a threshold τ_δ are flagged as geometric merge candidates, but similarity alone does not establish semantic equivalence.

The LLM then processes each candidate pair, reading 20 sampled documents from each cluster, and confirms or rejects the merge. After each pass, we recompute centroids and candidate similarities, stopping when no further merges are accepted or the iteration limit is reached. For each surviving proposal, the LLM generates a short name and a one-sentence definition from up to 20 representative documents selected by proximity to its centroid. Appendix A.2 provides the complete protocol.

4.3 STAGE 3: ASSIGN DOCUMENTS

Each document is independently assigned by the LLM against the consolidated definitions, rather than inheriting its proposal-cluster membership. The LLM selects the best-matching definition or leaves the document unassigned when none applies. Figure 2 illustrates both reassignment and abstention.

Implementation. We use similarity-weighted paraphrase-augmented GTE-Large embeddings (1024 dimensions), UMAP with five output dimensions and cosine distance, and a split threshold of 50 with no fixed recursion-depth cap. A shared UMAP/HDBSCAN configuration is selected per category using 50 Optuna TPE trials (Akiba et al., 2019). The reported run scores complete proposal trees for leaf count, noise, balance, coherence, and separation. Appendices A and D provide embedding, scoring, and search details.

5 EXPERIMENTS

5.1 DATASET AND SETUP

We evaluate on a large-scale customer service corpus from a North American telecommunications provider: 40,000+ daily interactions across 140+ intent categories (“tertiary categories”), with 2,000–4,000 interactions per category. Detailed results are reported on the 10 highest-volume categories, which account for approximately 90% of total interaction volume. The framework is applied identically to all categories; the 10-category subset is chosen for evaluation because all three methods (Flat, LLM-first, and Proposed) were run on these categories, enabling controlled comparison.

Core triggers. Raw interactions are full agent–customer conversation transcripts, often thousands of tokens long and dominated by greetings, holds, and troubleshooting back-and-forth. An upstream LLM distills each transcript into a single-sentence *core trigger*—the customer’s primary reason for calling, stated in their own words (e.g., “My bill is higher than expected,” “I already made a payment but my service is still suspended”). This distillation is critical for two reasons. First, it strips conversational noise so that embedding similarity reflects topical meaning rather than call-flow structure. Second, it produces short, semantically dense inputs where standard bag-of-words topic models struggle but embedding-based methods excel. The upstream extraction is a separate system; this paper operates entirely on the distilled core triggers.

Embedding model. GTE-Large (Li et al., 2023) (1024-dimensional, L2-normalized) served via a managed endpoint. Each core trigger is embedded independently.

Full experiment-grid and caching details are provided in Appendix B.

5.2 EVALUATION METRICS

We evaluate cluster quality along four axes: geometric quality of individual topics, geometric distinctness between topics, structural properties of the taxonomy, and operational usefulness judged by an independent LLM.

- **Topic Coherence** measures how tightly a topic’s documents cluster in embedding space: the mean pairwise cosine similarity among all documents assigned to the same topic, averaged over topics. Values range from 0 to 1; higher means the topic groups semantically similar documents. A topic that mixes unrelated issues (e.g., autopay failures and billing disputes in one cluster) scores low.
- **Topic Separation** measures how distinct topics are from one another: $1 - \text{mean cosine similarity between all pairs of topic centroids within the same category}$, so higher values indicate more distinguishable topics. Computed within each category; the overall score reported in all tables is the unweighted mean across categories, applied identically to every method (Flat, LLM-first, and Proposed). A method that produces many near-duplicate topics scores low on separation even if each individual topic is coherent.
- **Structural metrics** characterize the shape of the discovered taxonomy: the number of topics (leaf count), the fraction of documents left unassigned (noise rate), and how evenly documents distribute across topics (balance, measured as the ratio of observed entropy to maximum entropy). These are not quality scores in themselves but diagnostic indicators—a method that assigns every document to one large cluster has perfect noise but trivial leaf count.
- **LLM-as-Judge** evaluates whether discovered topics are operationally useful, not just geometrically clean. For each topic with ≥ 10 documents, an independent judge (Claude Sonnet 4.6) rates *specificity* (1–5: is the topic narrow enough to assign to one team?) and *actionability* (1–5: could an analyst take a concrete action based on this topic?) from a sample of 10 representative documents (Zheng et al., 2023). Each topic is evaluated twice; scores are averaged within categories and reported as unweighted category-macro-averages. Full protocol in Appendix G.

Coherence and separation are computed on the same embeddings used for clustering, so they favor methods that align well with the embedding geometry. The LLM-as-judge evaluation is independent of the embedding space and serves as a complementary signal.

5.3 BASELINES

We compare against two baselines that represent the dominant approaches in the literature:

1. **Flat clustering:** Flat UMAP + HDBSCAN per category with no recursion—the standard BERTopic-style approach (Grootendorst, 2022). This baseline uses the same embeddings and the same Optuna-tuned UMAP/HDBSCAN configuration as the proposed method, but skips recursive splitting: whatever clusters the first pass produces are the final topics. Differences in results therefore isolate the effect of recursive over-fragmentation and consolidation.
2. **LLM-first discovery:** A frontier LLM (Claude Sonnet 4.6) generates subcategory definitions directly from document samples, then every document is assigned against that taxonomy—the approach of TnT-LLM (Wan et al., 2024) and TopicGPT (Pham et al., 2024). This baseline tests whether a strong LLM can discover subcategories from text alone, without the geometric proposals that our method provides.

Appendix F reports ablations of the embedding strategy, merging criterion, and scoring function.

6 RESULTS

6.1 FLAT CLUSTERING HIDES SUBCATEGORIES

Table 2: Three-way comparison across 10 categories. Flat = single-pass UMAP + HDBSCAN; LLM-first = frontier LLM generates subcategory definitions, then assigns documents; Proposed = clusters-as-proposals after consolidation.

Method	Topics	Coherence	Separation	Noise %
Flat	111	0.792	0.106	15.3%
LLM-first	203	0.758	0.098	0.2%
Proposed (ours)	234	0.835	0.116	28.6%

Flat clustering averages 11 topics per category. These groups are internally similar but absorb finer distinctions: a single cluster of 600 payment calls may mix autopay failures, unexpected charges, and app errors. The proposed pipeline discovers $2.1\times$ more topics per category with higher coherence (+5.4%) and separation (+9.4%), at the cost of higher noise (28.6% vs. 15.3%). The elevated noise is by design: documents that do not clearly match any fine-grained proposal are left unassigned rather than forced into an ill-fitting cluster.

6.2 CLUSTER PROPOSALS VS. LLM-FIRST DISCOVERY

Coherence improves over the LLM-first baseline in all 10 categories (Appendix Table 8). In 3 of those categories the LLM-first baseline finds more topics (e.g., Update Payment Methods: 41 vs. 29), but in each case with lower coherence. Noise, however, rises substantially (28.6% vs. 0.2% for LLM-first)—a coverage–precision tradeoff where documents that do not clearly match any fine-grained definition are left unassigned rather than forced into a poor match.

6.3 CONSOLIDATION IS A DIAL, NOT A FIXED STEP

Over-fragmentation deliberately produces more proposals than the final taxonomy needs. The question is how aggressively to consolidate them. Because the merge decision is made by an LLM reading definitions and sampled documents, the aggressiveness is controlled at the prompt level: a stringent prompt merges only when two proposals are near-identical; a lenient prompt merges whenever they overlap substantially. The same geometric candidates reach the LLM either way—what changes is how readily it confirms.

We set the dial to its lenient end: the LLM is prompted to confirm a merge only when both proposals clearly describe the same subcategory, preserving borderline distinctions rather than collapsing them. Table 3 summarizes the result.

Table 3: Consolidation statistics on the best 10-category run under lenient merging.

Statistic	Value
Pre-consolidation proposals	244
Merge candidates identified (geometric)	80
Merges accepted (LLM-validated)	10
Post-consolidation topics	234
Redundancy rate	4.1%

Delta-vector similarity flags 80 of 244 proposals as geometric merge candidates ($\text{sim}_\delta \geq \tau_\delta$). Even under lenient prompting, the LLM confirms only 10—a 4.1% redundancy rate, spanning 4 of 10 categories. The remaining 70 candidates share similar deviation directions but, when the LLM reads their definitions and supporting documents, describe genuinely different subcategories. Geometry alone would over-merge; the LLM gate prevents it.

The low redundancy under lenient consolidation means that most over-fragmented proposals are already distinct—the pipeline pays very little for being exhaustive. A more stringent prompt would

reduce topic count further at the cost of collapsing borderline distinctions; we leave that tradeoff to the deployment context. Raw cosine similarity between centroids within a category stays above 0.85 and cannot separate redundant proposals from distinct ones; delta-vector projection strips the shared category direction, making redundancy visible to the LLM in the first place.

6.4 LLM-AS-JUDGE EVALUATION

Coherence and separation measure geometric quality but not whether topics are *operationally useful*. We use an LLM-based actionability protocol (Zheng et al., 2023) with Claude Sonnet 4.6 as the judge.

Protocol. For each topic with ≥ 10 documents, Claude Sonnet 4.6 rates specificity and actionability on 1–5 scales (two independent runs per topic, averaged). Category-macro-averaged scores are the primary statistic; significance is assessed with two-sided Wilcoxon signed-rank tests paired by category ($n=10$). Full protocol details, eligibility criteria, and call counts are in Appendix G.

Results. Table 4 summarizes the evaluation. The proposed method receives higher specificity and actionability than the LLM-first baseline in all ten categories—modest gains on a five-point scale, but consistent enough to reach significance on both measures (two-sided Wilcoxon, $p=0.002$; Cohen’s $d \approx 0.3$). Against flat clustering, the direction is the same but the signal is weaker ($p=0.16$, actionability), reflecting both the small number of category pairs ($n=10$) and the fact that gains concentrate in the subset of categories where flat clustering produces very few topics. These ratings measure LLM-judged usefulness, not demonstrated business outcomes.

Table 4: LLM-as-judge actionability evaluation.

Method	Topics	Specificity	Actionability	% Act ≥ 4
Flat	111	3.26 (3.39)	3.58 (3.69)	48.6
LLM-first	160 / 203	3.19 (3.25)	3.58 (3.63)	42.5
Proposed (ours)	234	3.43 (3.46)	3.78 (3.80)	54.3

Consistent improvement over LLM-first. On both measures, the proposed method exceeds the LLM-first baseline in every category—by roughly a quarter-point on each scale. The 43 LLM-first topics with fewer than 10 documents are not scored; conclusions apply to the 160 eligible topics.

Against flat clustering, benefits are conditional. The proposed method improves actionability over flat clustering in six of ten categories (Appendix Table 9). The gains concentrate where flat clustering produces very few, large groups: in the four categories where the baseline finds only 2–4 topics, actionability rises by over half a point on average. Refinement is most valuable precisely where coarse clustering is most likely to absorb distinct subcategories.

In the four categories where flat clustering already finds many well-separated groups, the proposed approach scores lower. This supports a category-dependent granularity tradeoff rather than universal superiority. We treat the subgroup analysis as exploratory; it does not establish that recursion caused the gains or identify which specific baseline groups contained the recovered subcategories.

Size–actionability relationship. Smaller proposed topics receive higher actionability ratings (Spearman $\rho=-0.215$, $p=0.001$); this correlation is weaker for flat clustering and absent for LLM-first (Appendix G.1).

Actionable topic counts. 54.3% of proposed topics are rated actionable (≥ 4), compared to 48.6% for flat and 42.5% for LLM-first; however, pairwise redundancy among highly rated topics means these counts overstate distinct actionable discoveries (Appendix G.2).

Failure example. Recursive fragmentation does not always isolate a single problem. In *Make A Payment*, proposed topic 32 (197 documents) still combines outdated cards, system errors, enrollment logic, and scheduling restrictions—the judge rates it 3/3 (specificity/actionability) and notes it

“would require separate investigations.” This pattern recurs in inherently multi-cause categories (Multi-Symptom, Charges & Fees), where the flat baseline’s broader groups sometimes receive higher actionability scores because the judge views them as coherent enough to route, even though they conceal finer distinctions.

7 DISCUSSION AND LIMITATIONS

Recursive proposals are most useful when existing categories contain broad groups that obscure finer distinctions. Actionability gains concentrate in categories where flat clustering produces few topics; benefits are less consistent where the baseline already finds finer groups. The tradeoff is coverage: the proposed pipeline leaves roughly twice as many documents unassigned as flat clustering (Table 2), so higher coherence does not come free. These comparisons do not establish quality at matched coverage. Detailed granularity, coverage, and size diagnostics are in Appendix C.4.

Our evidence comes from a single enterprise corpus with no reference taxonomy, so the results establish geometric quality and granularity rather than recall against ground truth or semantic completeness. While we validate on customer service data, the clusters-as-proposals design applies anywhere documents are pre-sorted into categories and finer sub-topics are needed: scientific papers within venues, support tickets within product areas, legal documents within case types. We name the public test explicitly: CLINC150 (Larson et al., 2019) with its ten released domains given as coarse categories (and 150 fine intents hidden), and the long-tailed CLINC150 and BANKING77 (Casanueva et al., 2020) splits of Zhang et al. (2024), with fine labels hidden. These benchmarks have known ground-truth intents at multiple granularities, making them a direct test of whether recursive proposals recover subcategories that other methods conceal. Re-running on them requires no change to the method.

Broader impact: The system is deployed for customer experience analysis, where discovered topics inform product improvements. Privacy is handled upstream: all text is PII-masked before any pipeline component processes it.

8 CONCLUSION

Coarse categories conceal finer distinctions, and current discovery methods—whether clustering-based or sampling-based—lose hidden subcategories before they can be named: sampling misses small groups, and frequency- or size-based pruning removes them. The loss is asymmetric: a redundant proposal costs one merge, but a missing proposal cannot be recovered by any later stage. We introduced *clusters as proposals*: each category is recursively over-fragmented so that every dense region reaches the LLM as a candidate, and consolidation—centered on the category’s own geometry and confirmed by the LLM—removes redundancy before documents are labeled against the surviving definitions.

On ten categories of a production call-center corpus, recursive proposals yield roughly twice the subcategories of flat clustering and a fifth more than LLM-first discovery, with higher measured coherence than both. Exhaustiveness is cheap: only 4.1% of proposals turn out to be redundant. In 8 of 10 categories, at least one discovered subcategory holds less than 2% of that category’s volume—the hidden issues the method is designed to surface. An independent LLM judge rates the proposed topics more specific and more actionable than the LLM-first baseline in every category ($p=0.002$, two-sided Wilcoxon). Against flat clustering, the gains concentrate where the baseline produces only 2–4 coarse topics—precisely where hidden subcategories are most likely—and vanish where the baseline already finds finer groups. The method requires no fine-tuning, no labeled examples of fine classes, and no advance specification of the number of subcategories. These results do not establish semantic completeness, topic uniqueness, or realized business impact; comparisons remain subject to differences in document coverage and topic eligibility.

AI-USE STATEMENT

AI assistance was used to revise the paper’s narrative, review methodological choices and interpretations, check consistency among reported quantities, and propose further analyses. Language

models were also used within the experimental pipeline to construct representations and supervision and to make prompted predictions. The authors are responsible for verifying the final text, results, citations, and disclosures against the underlying research records.

ETHICS STATEMENT

The study analyzes enterprise customer-service calls, and the evaluated label concerns suppression of further product offers. Errors can change which customers continue to receive offers. Aggregate agreement alone does not establish that two systems affect the same individuals, so operational use requires calibration and review at the intended decision threshold. Access to the underlying transcripts is restricted by the deployment’s data agreement. Brand canonicalization and speaker attribution are representation operations; they should not be interpreted as guarantees of anonymization. The example is an attributed agent–customer exchange. Its statements record what the speakers said, including the agent’s offer and the customer’s plans, without independently verifying those claims.

REPRODUCIBILITY STATEMENT

All experiments use fixed random seeds at every pipeline stage. The method is described in §4; Appendices A, D, and E specify the scoring function, search space, and pseudocode. Appendix B defines the experiment grid and full run configuration. Because coherence and separation serve as both optimization targets and evaluation metrics, the independent LLM-as-judge evaluation (§6.4) is especially important for validating the results. Code will be released upon acceptance.

REFERENCES

- Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. doi: 10.1109/TPAMI.2012.120.
- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019. doi: 10.1145/3292500.3330701.
- Dimo Angelov. Top2Vec: Distributed representations of topics. arXiv preprint arXiv:2008.09470, 2020.
- Jianhong Bai, Zuozhu Liu, Hualiang Wang, Ruizhe Chen, Lianrui Mu, Xiaomeng Li, Joey Tianyi Zhou, Yang Feng, Jian Wu, and Haoji Hu. Towards distribution-agnostic generalized category discovery. In *Advances in Neural Information Processing Systems*, volume 36, pp. 58625–58647. Curran Associates, Inc., 2023. doi: 10.52202/075280-2555.
- David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. Hierarchical topic models and the nested Chinese restaurant process. In S. Thrun, L. Saul, and B. Schölkopf (eds.), *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2003a.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003b.
- Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining (PAKDD 2013), Part II*, volume 7819 of *Lecture Notes in Artificial Intelligence*, pp. 160–172. Springer, 2013. doi: 10.1007/978-3-642-37456-2_14.
- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, 2020. doi: 10.18653/v1/2020.nlp4convai-1.5.

- Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552.
- Maarten Grootendorst. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv preprint arXiv:2203.05794, 2022.
- Zhenqi He, Yuanpei Liu, and Kai Han. SEAL: Semantic-aware hierarchical learning for generalized category discovery. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL .
- Xu Ji, João F. Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9865–9874, 2019.
- Michelle S. Lam, Janice Teoh, James A. Landay, Jeffrey Heer, and Michael S. Bernstein. Concept induction: Analyzing unstructured text with high-level concepts using LLOOM. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI ’24)*, Honolulu, HI, USA, 2024. Association for Computing Machinery. doi: 10.1145/3613904.3642830.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1311–1316, 2019. doi: 10.18653/v1/D19-1131.
- Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. arXiv preprint arXiv:2308.03281, 2023.
- Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018.
- Chau Minh Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. TopicGPT: A prompt-based topic modeling framework. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2956–2984, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.164.
- Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Gutttag. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1194–1203, 2021.
- Evgeny Stepanov, Benoit Favre, Firoj Alam, Shammur Chowdhury, Karan Singla, Jeremy Trione, Frédéric Béchet, and Giuseppe Riccardi. Automatic summarization of call-center conversations. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Demo Papers*, Scottsdale, AZ, USA, December 2015. URL .
- Alex Tamkin, Miles McCain, Kunal Handa, Esin Durmus, Liane Lovitt, Ankur Rathi, Saffron Huang, Alfred Mountfield, Jerry Hong, Stuart Ritchie, Michael Stern, Brian Clarke, Landon Goldberg, Theodore R. Sumers, Jared Mueller, William McEachen, Wes Mitchell, Shan Carter, Jack Clark, Jared Kaplan, and Deep Ganguli. Clio: Privacy-preserving insights into real-world AI use. arXiv preprint arXiv:2412.13678, 2024.
- Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7482–7491, 2022.
- Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering. *Transactions of the Association for Computational Linguistics*, 12:321–333, 2024. doi: 10.1162/tacl_a_00648.

- Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W. White, Longqi Yang, Reid Andersen, Georg Buscher, Dhruv Joshi, and Nagu Rangan. TnT-LLM: Text mining at scale with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5836–5847, Barcelona, Spain, 2024. Association for Computing Machinery. doi: 10.1145/3637528.3671647.
- Zihan Wang, Jingbo Shang, and Ruiqi Zhong. Goal-driven explainable clustering via language descriptions. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10626–10649, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.657.
- Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. Discovering new intents with deep aligned clustering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14365–14373, 2021. doi: 10.1609/aaai.v35i16.17689.
- Shun Zhang, Chaoran Yan, Jian Yang, Jiaheng Liu, Ying Mo, Jiaqi Bai, Tongliang Li, and Zhoujun Li. Towards real-world scenario: Imbalanced new intent discovery. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3949–3963, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.217.
- Yuwei Zhang, Zihan Wang, and Jingbo Shang. ClusterLLM: Large language models as a guide for text clustering. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13903–13920, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.858.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A IMPLEMENTATION DETAILS

Notation. Let $\mathcal{D} = \{(d_i, c_i)\}_{i=1}^N$ be a corpus where each document d_i is a single-sentence core trigger (see §5) pre-assigned to a category $c_i \in \mathcal{C}$, with $|\mathcal{C}| \gg 1$ (140+ in our setting). Let $\phi : d \mapsto \mathbb{R}^m$ be a pre-trained embedding function ($m = 1024$). For each category c , define $\mathcal{D}_c = \{d_i : c_i = c\}$ and the category embedding matrix $\mathbf{E}_c \in \mathbb{R}^{|\mathcal{D}_c| \times m}$.

A.1 RECURSIVE PROPOSAL PROCEDURE

For a node v containing n_v documents with embeddings $\mathbf{E}_v$, the recursive procedure is:

1. Apply UMAP to reduce $\mathbf{E}_v \in \mathbb{R}^{n_v \times m}$ to $\mathbf{Z}_v \in \mathbb{R}^{n_v \times d}$ ($d = 5$, cosine metric)
2. Apply HDBSCAN to $\mathbf{Z}_v$, obtaining clusters $\{C_1, \dots, C_k\}$ and noise set $\mathcal{N}_v$
3. If splitting makes no progress, retain the current cluster as a leaf; otherwise recurse on each C_j with $|C_j| > \tau_{\text{split}}$
4. Each remaining C_j with $|C_j| \leq \tau_{\text{split}}$ becomes a leaf of $\mathcal{T}_c$

The minimum cluster size ranges from 20 to 200 in the search, so a 30-document subgroup can only become a separate proposal if the chosen minimum is at most 30. One UMAP/HDBSCAN configuration is shared across recursion levels. The proposal tree is an intermediate structure; consolidated definitions and document assignments form the final output.

A.2 ITERATIVE MERGING PROTOCOL

We apply greedy merging with iterative convergence:

1. Compute $\text{sim}_\delta(l, l')$ for all leaf pairs within category c
2. Sort candidates above threshold τ_δ by similarity (highest first)
3. Each leaf participates in at most one merge per pass (greedy exclusion)
4. After merging, recompute centroids and delta vectors on the mutated tree
5. Repeat until no new candidates or max iterations reached

Merge candidates are validated by an LLM (sampling 20 sentences from each cluster and querying for semantic equivalence) in a two-phase protocol: collect all decisions, then batch-mutate. Typical convergence: 1–3 iterations.

Tree-level Bayesian optimization. Phase 1 will over-fragment or under-fragment depending on the HDBSCAN parameters. We share a single parameter configuration θ across all depths and use Optuna’s (Akiba et al., 2019) TPE sampler over 50 trials to find the right degree of fragmentation. Each trial constructs the complete recursive tree $\mathcal{T}_c(\theta)$ and scores it:

$$\theta^* = \arg \max_{\theta} S(\mathcal{T}_c(\theta)) \quad (2)$$

The scoring function S is a composite of three terms:

$$S(\mathcal{T}) = w_{\text{count}} \cdot R_{\text{leaf}}(\mathcal{T}) - P_{\text{noise}}(\mathcal{T}) + B_{\text{entropy}}(\mathcal{T}) \quad (3)$$

where R_{leaf} is a Gaussian reward centered on a target leaf-count range $[L_{\min}, L_{\max}]$:

$$R_{\text{leaf}}(\mathcal{T}) = \exp\left(-\frac{(L - \mu)^2}{2\sigma^2}\right), \quad \mu = \frac{L_{\min} + L_{\max}}{2}, \quad \sigma = \frac{L_{\max} - L_{\min}}{4} \quad (4)$$

P_{noise} is an asymmetric noise penalty (over-noise penalized at twice the rate of under-noise), and B_{entropy} is a normalized entropy bonus that rewards balanced leaf sizes. An optional quality-aware mode adds topic coherence $\mathcal{C}$ and separation $\mathcal{S}$:

$$S_{\text{quality}}(\mathcal{T}) = S(\mathcal{T}) + w_q \cdot \frac{\mathcal{C}(\mathcal{T}) + \mathcal{S}(\mathcal{T})}{2} \quad (5)$$

When $|\mathcal{D}_c| > 50,000$, optimization runs on a random subsample and the final tree is rebuilt on full data with θ^* .

Paraphrase-augmented embeddings. Lexical scatter causes semantically equivalent documents (“my bill is too high” vs. “I’m being overcharged”) to separate in embedding space, creating spurious proposals. We tighten representations *before* recursion by averaging each document’s embedding with embeddings of $K=4$ LLM-generated paraphrases (Gao et al., 2021; Shanmugam et al., 2021), weighted by their cosine similarity to the original (threshold $\tau_s = 0.75$; paraphrases below this are discarded). The augmented embedding is:

$$\hat{\phi}(d_i) = \frac{\phi(d_i) + \sum_{k=1}^K w_k \cdot \phi(p_i^k)}{1 + \sum_{k=1}^K w_k}, \quad w_k = \begin{cases} \cos(\phi(d_i), \phi(p_i^k)) & \text{if } \geq \tau_s \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Embeddings use GTE-Large (Li et al., 2023) (1024-dimensional); paraphrases are generated by Claude Sonnet 4.6.

B EXPERIMENT GRID AND ENGINEERING DETAILS

Paraphrase generation. 4 paraphrases per core trigger generated by Claude Sonnet 4.6, pre-computed and cached (31,940 total paraphrases for the pilot set).

Experiment grid. We conduct 108 experiments (from 144 combinations, filtered by constraint: weighted mode requires augmentation enabled):

Table 5: Experiment grid dimensions.

Dimension	Options	Count
Text representation	core_trigger, systemic_issue, combined	3
Optuna trials	50, 80	2
Split threshold	50, 100, 200	3
Augmentation	enabled, disabled	2
Weighted averaging	enabled, disabled	2
Quality optimization	enabled, disabled	2

Embedding caching. 9 unique embedding configurations are computed once and reused across all 108 experiments, saving 92% of embedding API calls.

Reported run configuration. Key configuration for the reported 10-category run: no fixed recursion-depth cap (recursion stops at the size threshold or when splitting makes no progress), split threshold $\tau_{\text{split}} = 50$ (proposed) / 10,000 (flat baseline), 50 tree-level optimization trials per category, similarity-weighted paraphrase augmentation enabled, quality optimization enabled, minimum leaf size 20, $\sim 40,000$ total samples. The split threshold is a recursion trigger, not a guaranteed final size bound: any cluster exceeding τ_{split} is recursed into, but the resulting leaves depend on the density structure found by HDBSCAN at each level. The largest observed leaf (678 documents) exceeds τ_{split} because HDBSCAN at that recursion level did not find further sub-structure above the minimum cluster size.

C ADDITIONAL ANALYSIS

C.1 SAMPLING RISK

A secondary risk: sampling. Beyond granularity, sampling-based discovery methods face an additional risk for small subcategories. A subcategory of share p is entirely absent from a uniform random sample of n documents with probability $(1 - p)^n$. For $p = 0.01$ and $n = 200$, this is approximately 13%. The probability of appearing fewer than three times—too few for a proposer to

recognize a pattern—is approximately 68%. This risk compounds with frequency- and size-based pruning in downstream stages.

C.2 WHY DELTA VECTORS ENABLE CONSOLIDATION

Within a category, all documents are about roughly the same thing (e.g., billing), so their embeddings cluster tightly around the category centroid μ_c . Proposal centroids differ from μ_c by small perturbations, and cosine similarity between any two is dominated by the large shared “billing” component. This is the degenerate similarity problem described in §1: without removing the shared direction, consolidation cannot distinguish redundant proposals from genuinely distinct ones.

Subtracting the category centroid is analogous to centering data before PCA: it removes the mean and reveals the variance structure underneath. What remains is the direction in which each proposal *specializes*—the part that distinguishes “unexpected charges” from “autopay failures.” The threshold τ_δ controls how aggressively we consolidate. In practice, $\tau_\delta \in [0.70, 0.90]$ works across categories: lower values merge more proposals (reducing fragmentation at the risk of collapsing genuinely distinct topics), while higher values are conservative.

C.3 SCALABILITY

Embeddings are computed once and partitioned by category; each category then runs independently and categories can execute in parallel. For categories exceeding 50,000 samples, Optuna optimization runs on a random subsample and the final tree is rebuilt on full data with the best parameters. Embedding caching across the 108-experiment grid eliminates 92% of embedding API calls.

C.4 GRANULARITY AND COVERAGE

Topic counts and coherence scores summarize average quality but do not show how the method changes the *concentration* of assignments—the size of the largest groups and the fraction of documents left unassigned. Table 6 compares leaf-size ranges across the three methods.

Table 6: Leaf-size ranges across 10 categories. Min and max are over all leaves in each method. The proposed method produces smaller maximum groups while avoiding singleton leaves in this configuration; minimum-size constraints ($\text{min_cluster_size} \geq 20$) partly explain the absence of tiny leaves, and increased rejection partly explains smaller groups.

Method	Smallest leaf	Largest leaf	Topics	Unassigned
Flat	20	3,633	111	15.3%
LLM-first	1	1,325	203	0.2%
Proposed (ours)	22	678	234	28.6%

Where large groups shrink most. The strongest structural signal comes from categories where flat clustering produces very few, very large groups. Table 7 isolates four such categories.

Table 7: Topic counts, largest-leaf sizes, and noise rates for four categories with large flat-clustering groups. These category-level comparisons characterize changes in granularity and coverage; they do not establish document-level correspondence between baseline and proposed leaves.

Category	Flat			Proposed		
	Topics	Largest	Noise	Topics	Largest	Noise
Service Not Restored	3	3,633	1.0%	23	234	37.3%
Unable to Access Account	2	2,971	2.2%	18	516	36.4%
Update Payment Methods	3	2,750	0.5%	29	376	21.3%
Port-In Request	12	1,434	2.0%	30	193	23.0%

Service Not Restored illustrates the tradeoff most clearly: the single mega-cluster of 3,633 documents splits into 23 topics, but the unassigned fraction rises from 1% to 37%. The LLM judge rates the finer topics more actionable (mean rises from 3.67 to 4.02), and twice as many topics cross the actionable threshold—but that gain comes at the cost of leaving a third of documents unassigned.

Counterexamples. The proposed method does not always increase fragmentation. In three categories it produces *fewer* topics than flat clustering: Charges & Fees (24 $\rightarrow$ 21), Multi-Symptom (26 $\rightarrow$ 22), and Update Account Details (20 $\rightarrow$ 13), all with lower noise. This suggests the method’s effects depend on category structure rather than simply increasing fragmentation everywhere. The tradeoff is not uniformly favorable: Update Account Details falls from 0.819 to 0.807 coherence and 0.172 to 0.136 separation under the proposed method. Categories where flat clustering already finds many groups may not benefit from recursive splitting.

Size of what was found. Median leaf sizes are comparable across methods (Table 6); the structural difference is at the extremes. The largest flat leaf is over five times the largest proposed leaf—the method compresses the biggest groups rather than uniformly shrinking all topics. Per-category topic counts are also more uniform: the max/min ratio drops from $13\times$ (flat) to $2.3\times$ (proposed). In 8 of 10 categories, at least one proposed subcategory holds less than 2% of that category’s volume—the hidden issues the method is designed to surface.

C.5 PER-CATEGORY BREAKDOWNS

Table 8: Per-category breakdown: proposed pipeline vs. LLM-first baseline (10 categories).

Category	LLM-first Baseline			Proposed (ours)		
	Topics	Coh.	Sep.	Topics	Coh.	Sep.
Charges & Fees	15	0.749	0.080	21	0.805	0.113
Activate a Device	10	0.760	0.052	27	0.840	0.102
Make A Payment	18	0.752	0.112	24	0.839	0.137
Multi-Symptom	34	0.746	0.129	22	0.820	0.107
Port-In Request	29	0.777	0.091	30	0.848	0.104
Service Not Restored	15	0.774	0.079	23	0.853	0.109
Unable to Access Account	15	0.752	0.088	18	0.847	0.123
Unable to Call	12	0.751	0.080	27	0.829	0.115
Update Account Details	14	0.715	0.157	13	0.807	0.136
Update Payment Methods	41	0.772	0.115	29	0.863	0.112
Overall	203	0.758	0.098	234	0.835	0.116

D HYPERPARAMETER DETAILS

E ALGORITHM PSEUDOCODE

BUILDTREE recursively applies UMAP and HDBSCAN to generate proposals. Oversized clusters are split again until they meet τ_{split} or a splitting attempt makes no progress. In the latter case, the current cluster remains a leaf even if it exceeds the threshold; an unchanged cluster is not passed back into recursion. There is no fixed maximum depth. Noise remains outside the proposal leaves, and all documents are considered during final definition-based assignment (§4.3).

Table 9: Per-category mean actionability scores (1–5 scale, category macro-average) and fraction of topics rated actionable (≥ 4). Categories sorted by the actionability gap between proposed and flat baseline.

Category	Flat			LLM-first			Proposed			Δ Act
	n	Act	% ≥ 4	n	Act	% ≥ 4	n	Act	% ≥ 4	
Unable to Access Account	2	3.00	0.0	13	3.35	30.8	18	3.67	44.4	+0.67
Activate a Device	4	3.00	0.0	10	3.40	30.0	27	3.65	51.9	+0.65
Update Payment Methods	3	3.83	66.7	31	4.10	74.2	29	4.43	86.2	+0.60
Service Not Restored	3	3.67	33.3	10	3.85	60.0	23	4.02	69.6	+0.36
Make A Payment	7	3.71	42.9	15	3.93	53.3	24	3.98	58.3	+0.26
Port-In Request	12	3.50	33.3	24	3.46	41.7	30	3.60	53.3	+0.10
Unable to make/receive calls	10	3.75	70.0	10	3.50	30.0	27	3.67	44.4	−0.08
Multi-Symptom	26	3.54	38.5	21	3.43	23.8	22	3.43	27.3	−0.11
Charges & Fees	24	3.90	62.5	15	3.53	33.3	21	3.69	47.6	−0.21
Update Account Details	20	3.88	60.0	11	3.27	9.1	13	3.65	46.2	−0.22

Table 10: Search space for tree-level Bayesian optimization.

Parameter	Range	Type
(UMAP)	[10, 50]	Integer
(UMAP)	[0.0, 0.3]	Float
(HDBSCAN)	[20, 200]	Integer
(HDBSCAN)	[5, 50]	Integer

F PIPELINE ABLATIONS

F.1 SCORING FUNCTION COMPONENTS

Each scoring component addresses a specific failure mode. Without the noise penalty, the optimizer finds parameter configurations that push too many documents into the noise set. Without the entropy bonus, it produces lopsided trees where one leaf absorbs most documents. The three terms together steer the optimizer toward trees that are fragmented enough to expose finer distinctions but not so fragmented that the output is unmanageable.

F.2 MERGING CRITERION

Raw cosine similarity between centroids within a category is uniformly high (> 0.85) and cannot distinguish redundant proposals from distinct ones. Using raw cosine instead of delta-vector similarity for consolidation would either merge too aggressively (collapsing distinct subcategories) or too conservatively (leaving redundant proposals). Delta-vector projection removes the shared category direction, making genuine redundancy visible.

F.3 EMBEDDING STRATEGY

Table 13 isolates the effect of paraphrase augmentation on proposal quality. All three rows use the same pipeline configuration (split threshold, Optuna budget, quality optimization); only the embedding strategy differs.

Without augmentation, lexical scatter causes semantically equivalent documents (e.g., “my bill is too high” vs. “I’m being overcharged”) to separate in embedding space, creating spurious proposals. The result is 265 leaves—31 more than the weighted configuration—with lower coherence (0.804 vs. 0.835). Unweighted averaging of paraphrase embeddings reduces fragmentation to 250 leaves and raises coherence to 0.830, confirming that tighter representations before recursion produce fewer spurious splits. Similarity-weighted averaging (Appendix A), which discards paraphrases below a cosine threshold of 0.75 and weights the rest by similarity to the original, further reduces leaves to

Table 11: Default scoring function parameters.

Parameter	Value
Target leaf range $[L_{\min}, L_{\max}]$	[5, 50]
Target noise band $[\eta_{\min}, \eta_{\max}]$	[0.05, 0.20]
Cluster count reward weight w_{count}	1.0
Noise penalty weight w_n	1.0
Balance bonus weight w_b	0.2
Quality weight w_q (optional)	0.3

Algorithm 1 Clusters-as-Proposals: Recursive Taxonomy Construction**Require:** Embeddings $\mathbf{E}_c$, split threshold τ_{split} , Optuna budget T

```

1: Initialize Optuna study with TPE sampler
2: for trial  $t = 1, \dots, T$  do
3:   Sample  $\theta_t = (n_{\text{neighbors}}, \text{min\_dist}, \text{min\_cluster\_size}, \text{min\_samples})$ 
4:    $\mathcal{T}_t \leftarrow \text{BUILD TREE}(\mathbf{E}_c, \theta_t, \tau_{\text{split}})$ 
5:   Compute  $S(\mathcal{T}_t)$  via Equations 3–5
6:   Report  $S(\mathcal{T}_t)$  to Optuna
7: end for
8:  $\theta^* \leftarrow$  best parameters from study
9:  $\mathcal{T}^* \leftarrow \text{BUILD TREE}(\mathbf{E}_c, \theta^*, \tau_{\text{split}})$ 
10: Apply delta-vector merging (§4.2) to  $\mathcal{T}^*$ 
11: return  $\mathcal{T}^*$ 

```

234 and achieves the highest coherence (0.835) and separation (0.116). Separation is comparable across augmentation strategies (0.112 $\rightarrow$ 0.108 $\rightarrow$ 0.116), indicating that similarity-weighted averaging primarily benefits coherence and noise reduction rather than inter-cluster distinguishability.

G LLM-AS-JUDGE PROTOCOL

For each topic with ≥ 10 assigned documents, we sample 10 random documents and prompt the judge (Claude Sonnet 4.6, temperature 0.1) to rate the topic on two 1–5 scales: *specificity* (1 = too broad to act on, 5 = single well-defined problem assignable to one team) and *actionability* (1 = no clear action possible, 5 = direct, measurable intervention possible). Each topic is evaluated twice with different document samples; per-topic scores are averaged over both runs. Scores are then averaged within each category, and we report the unweighted macro-average across categories as the primary statistic. Topic-level averages are reported separately.

Topics with fewer than 10 documents were removed from analysis: all 111 flat and all 234 proposed topics are eligible; 160 of 203 LLM-first topics qualify (the remaining 43 have < 10 documents). Significance is assessed with two-sided Wilcoxon signed-rank tests paired by category ($n=10$). Total evaluation: $\sim 1,000$ LLM calls across all three approaches.

G.1 SIZE-ACTIONABILITY DETAILS

The proposed approach shows a significant negative correlation between topic size and actionability (Spearman $\rho = -0.215$, $p = 0.001$, $n = 234$): smaller topics produced by recursive splitting tend to receive higher actionability ratings. Flat clustering shows a weaker trend ($\rho = -0.171$, $p = 0.073$), while LLM-first shows no significant correlation ($\rho = -0.053$, $p = 0.508$). Median leaf sizes are 90 (proposed), 95 (flat), and 129 (LLM-first) over the evaluated topics; the largest difference in concentration is at the extremes (Table 6), not in typical leaf size.

G.2 ACTIONABLE TOPIC COUNTS

Of the 234 proposed topics, 127 (54.3%) are rated actionable (≥ 4), compared to 54 of 111 (48.6%) for flat clustering and 68 of 160 (42.5%) for LLM-first. The proposed approach also produces

Table 12: Ablation of tree-level scoring components.

Scoring Variant	Leaf Target Hit	Noise in Band	Balance
Leaf reward only	High	Low	Low
+ Noise penalty	High	High	Low
+ Entropy bonus (full)	High	High	High

Table 13: Effect of paraphrase augmentation on the full pipeline (10 categories, same configuration). All reported main-text results use similarity-weighted augmentation (bottom row).

Embedding Strategy	Leaves	Noise %	Coherence	Separation
No augmentation	265	22.0%	0.804	0.112
Augmented (unweighted)	250	21.3%	0.830	0.108
Augmented (similarity-weighted)	234	28.6%	0.835	0.116

the most high-scoring topics: 51 rated 5 on actionability, versus 14 for flat and 20 for LLM-first. However, these counts should be read as “topics rated actionable,” not as 127 distinct actionable discoveries: the pairwise redundancy evaluation found substantial overlap among nearest-neighbor topics, and several highly rated topics may describe overlapping interventions. Actionability and distinctness must be assessed together.

H ADDITIONAL UMAP VISUALIZATIONS

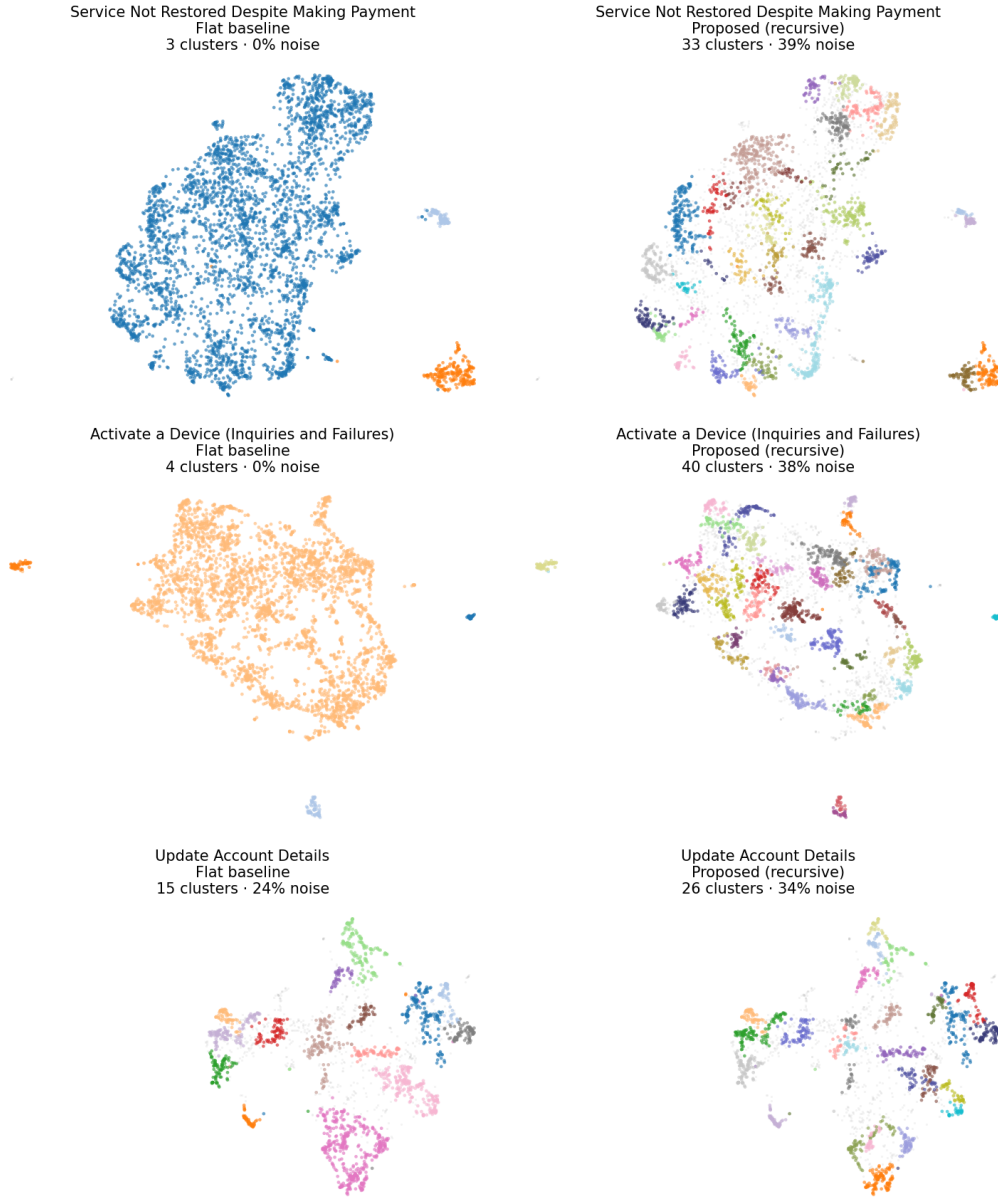


Figure 3: UMAP projections for three categories comparing flat baseline (left) with recursive proposal generation (right). Flat clustering produces few, large clusters (e.g., 4 clusters for “Activate a Device” with 0% noise), while recursive splitting surfaces fine-grained proposals (40 clusters, 38% noise). Color indicates cluster membership; gray points are noise.