LLM-Guided Lifecycle-Aware Clustering of Multi-Turn Customer Support Conversations

Priyaranjan Pattnayak, Sanchari Chowdhuri, Amit Agarwal, Hitesh Laxmichand Patel

Oracle America Inc.

Correspondence: priyaranjanpattnayak@gmail.com

Abstract

Clustering customer chat data is vital for cloud providers handling multi-service queries. Traditional methods struggle with overlapping concerns and create broad, static clusters that degrade over time. Re-clustering disrupts continuity, making issue tracking difficult. We propose an adaptive system that segments multi-turn chats into service-specific concerns and incrementally refines clusters as new issues arise. Cluster quality is tracked via Davies–Bouldin Index (DBI) and Silhouette Scores, with LLM-based splitting applied only to degraded clusters. Our method improves Silhouette Scores by over 100% and reduces DBI by 65.6% compared to baselines, enabling scalable, real-time analytics without full re-clustering.

1 Introduction

Cloud providers handle high volumes of customer chats that often span multiple service areas e.g., compute, networking, and identity. This complexity makes it hard to accurately categorize and track evolving concerns. Traditional clustering methods treat each chat as a single-topic unit and require full re-runs to update, disrupting consistency and hindering long-term tracking.

Exisiting methods like LDA, K-Means, and HDBSCAN (Blei et al., 2003; MacQueen, 1967; McInnes et al., 2017) enable initial topic discovery but produce static, coarse clusters needing manual refinement. While recent work in intent classification and dialogue tracking (Ye and Johnson, 2024; Gu et al., 2022), aids real-time understanding, it lacks dynamic organization of granular concerns across domains (Zhu et al., 2024).

We propose an adaptive framework to cluster user concerns from multi-turn chats. LLMs segment conversations into themes, extract concerns, remove duplicates via contrastive filtering, and classify them into service groups. Appendix A defines key terms used throughout the paper, describing the structure and categorization of customer support conversations in cloud service environments. Within each group, HDBSCAN + UMAP (McInnes et al., 2017, 2018a,b) creates interpretable topic clusters, labeled using LLMs (Ma et al., 2024; Pattnayak et al.). New concerns are incrementally matched; unmatched ones form new clusters when volume permits.

We track DBI and Silhouette Scores (Davies and Bouldin, 1979; Rousseeuw, 1987a). to monitor cluster quality. Degraded clusters are flagged using Z-score and cohesion tests, then refined using LLM-based splitting avoiding full re-clustering and preserving cluster identity for stable, actionable insights.

Our Contributions:

- **LLM-based segmentation of multi-turn chats:** Breaking down complex chats into coherent themes and distinct concerns, using contrastive filtering to remove duplicates.
- **Service group classification and topic clustering:** Assigning concerns to predefined cloud service groups (F1 > 0.85) and forming interpretable topic clusters.
- **Incremental clustering for emerging concerns:** Dynamically assigning new concerns to existing clusters or forming new ones using LLM-based semantic matching, no full re-clustering needed.
- **Metric-driven adaptive refinement:** We monitor DBI, Silhouette, and cohesion scores to detect drift, refining only affected clusters using LLM-based splitting to maintain stability. Deployed on 90,000+ chats, the system handles 500+ new concerns daily, enabling real-time issue tracking and trend detection without manual labeling or reprocessing.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 3180–3206

2 Related Work

2.1 Traditional Clustering in Customer Support

Methods like LDA (Blei et al., 2003) and K-Means (MacQueen, 1967) are common for text clustering but struggle with fixed topic counts and long-tailed data, limiting their use for evolving support concerns. HDBSCAN (McInnes et al., 2017) improves discovery by auto-selecting cluster counts but requires full re-clustering for updates. Density-based approaches (Aggarwal and Zhai, 2012) have also been tried but lack scalability for real-time support scenarios.

2.2 Multi-Turn Dialogue Understanding and Intent Classification

Recent work in dialogue modeling focuses on intent classification and conversational state tracking to improve real-time query understanding (Ye and Johnson, 2024; Patel et al., 2025; Gu et al., 2022). While effective for chatbot resolution, these approaches rely on fixed intent categories (Pattnayak et al., 2025a) ,and lack the ability to form evolving, structured topic clusters. In contrast, our framework segments multi-turn conversations at the concern level, enabling dynamic clustering beyond static intent labels.

2.3 Embedding-Based Retrieval

Retrieval-based clustering methods using models like Sentence-BERT (SBERT) (Reimers and Gurevych, 2019; Ni et al., 2022; Thakur et al., 2021; Meghwani et al., 2025) enhance semantic matching but fail to detect cluster degradation from topic drift. While contrastive learning improves text representations (Ma et al., 2024; Gao et al., 2021; Gunel et al., 2021), its application (Patel et al., 2024; Pattnayak et al., 2024) in support clustering is limited. We incorporate contrastive filtering to refine extracted concerns prior to clustering, improving semantic coherence.

2.4 Incremental and Adaptive Clustering

Prior work on incremental clustering in streaming data (Zhang et al., 2018; Li et al., 2021; Rohit et al., 2022),focuses more on classification than maintaining cluster coherence. Approaches using hierarchical adaptation(Moseley and Wang, 2017; Agarwal et al., 2024a) or drift detection (Gama et al., 2014) often require expensive re-computation. In contrast, our method monitors cluster quality and selectively refines only drifting clusters avoiding full re-clustering.

(Bentley and Batra, 2016) introduced Microsoft's Office Customer Voice system, which clusters short, single-turn feedback for ad-hoc insights. In contrast, our method handles multi-turn conversations through LLM-guided segmentation and lifecycle-aware clustering, enabling incremental refinement: split, merge, and prune, while preserving cluster identity for longitudinal analysis.

2.5 Our Approach

Our work introduces an adaptive clustering framework that (1) segments multi-turn chats into themes, removes redundancy and classifies concerns into service groups, (2) dynamically refines clusters through metric-driven monitoring, and (3) leverages LLMs for semantic matching, new cluster creation, and targeted splitting. Unlike prior approaches, we avoid disruptive full re-clustering by continuously tracking DBI, Silhouette, and Cohesion Scores to maintain stable, scalable, and interpretable clustering for customer service analytics.

3 Methodology

We propose a dynamic clustering framework for customer concerns in multi-turn chats that adapts without full re-clustering. Unlike traditional methods, it incrementally refines clusters while monitoring quality. Table 1 compares our method against traditional clustering approaches.

Feature	Traditional	Our Approach
Multi-turn Chat Handling	No	Yes
Concern-Level Segmentation	No	Yes
Incremental Clustering	Yes	Yes
Contrastive Filtering	No	Yes
LLM-Based Service Groups	No	Yes
Cluster Stability Monitoring	Yes	Yes
Automated Cluster Splitting	No	Yes
Evolving New Clusters	No	Yes

Table 1: Traditional vs. Proposed Clustering Approach

As shown in Figure 1, the framework has three key phases:(1) base cluster creation via LLM-driven concern extraction, filtering, classification, and HDBSCAN (Phase E of Fig 2); (2) incremental clustering for new concerns; and (3) LLM-based refinement triggered by cluster drift.

3.1 Base Cluster Creation

To form initial clusters, we segment multi-turn chats into service-specific themes and extract dis-

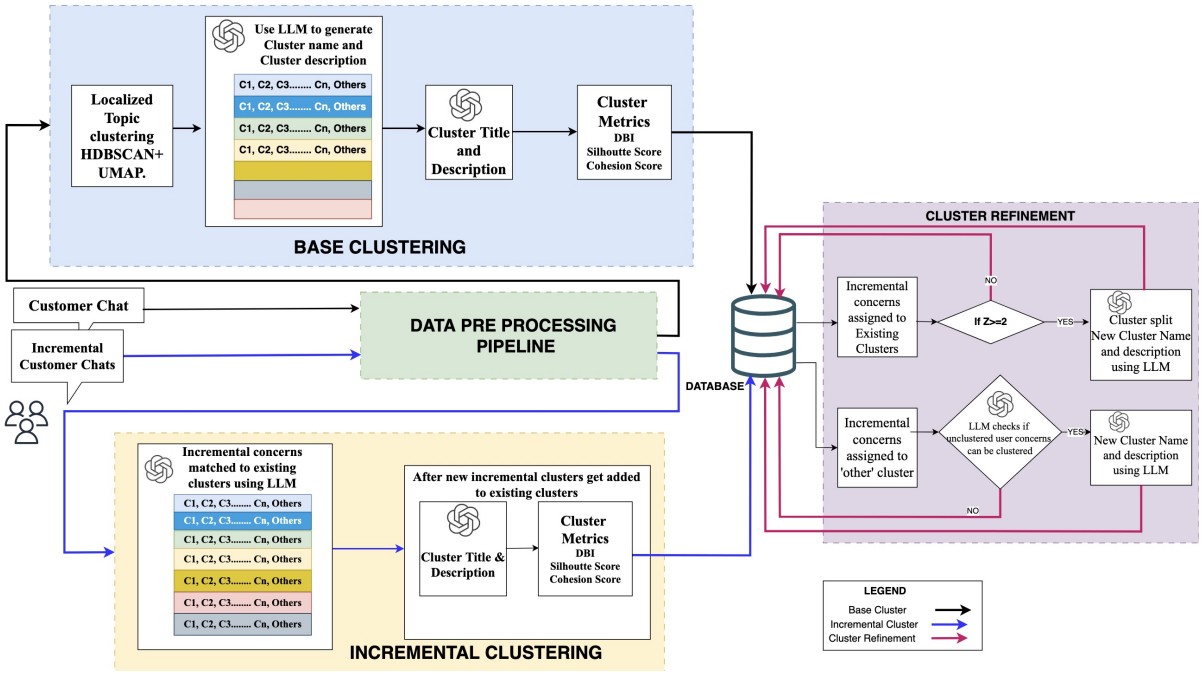

Figure 1: Architecture: The figure outlines base clustering, incremental clustering and cluster refinement pipelines.

tinct, contextually grounded concerns. Given the multi-issue nature of chats, this reduces redundancy and preserves clarity. Concerns are then classified into service groups via LLMs and clustered within each group. The full pipeline is shown in Figure. 2 where Phase A-E refer the Data Pre processing Pipeline.

1. **Segmentation**: Multi-turn chats are segmented into domain-specific themes using LLM-based detection (Fig. 2, Phase A) with the prompt shown in (Fig. 9 Appendix Q). The model detects topic shifts to separate concerns across service areas. A windowed prompt with overlapping context ensures coherence in long chats. The output is a structured list of segments, each with a theme title and relevant utterances. Quality was validated on 200 chats with strong inter-annotator agreement (Kappa = 0.79), guiding prompt refinement.

2. **Concern Extraction**: Each segment is processed by an LLM to extract key user concerns (Fig. 2, Phase B). A segment may yield multiple granular issues. Since user intents often span multiple utterances, our prompt instructs the model to extract standalone concerns and use windowed context (±1–2 turns) for cross-turn understanding. (e.g., "*My VM crashed and now I can't connect to storage*"), is split into two separate concerns.Sample prompts and output formats are detailed in (Fig.10 Appendix Q). We evaluated concern extraction on 150 manually annotated segments labeled by two experts (Kappa = 0.79). The LLM-based method achieved an F1 score of 0.84, indicating strong alignment with human annotations and effective identification of granular user concerns. (Appendix L) for performance metrics

3. **Contrastive Filtering**:To reduce redundant concerns in clusters, we apply cosine similarity filtering (threshold = 0.95) on *nli-roberta-base-v2* sentence embeddings, as shown in Fig 2 (Phase C).This step removes duplicate concerns while keeping distinct ones. Since cosine similarity may miss subtle semantic differences, we use a high threshold to minimize false negatives. Within each multi-turn chat, duplicate concerns are removed to prevent overweighting paraphrased repetitions of the same issue (e.g., "login failed... still can't log in"). The objective of this de-duplication step is to retain only distinct user concerns expressed within a single session, ensuring that intra-chat redundancy does not inflate cluster density. However, identical concerns appearing across different chats are intentionally preserved, as they reflect recurring customer issues and contribute to the representativeness and semantic diversity of the resulting clus-

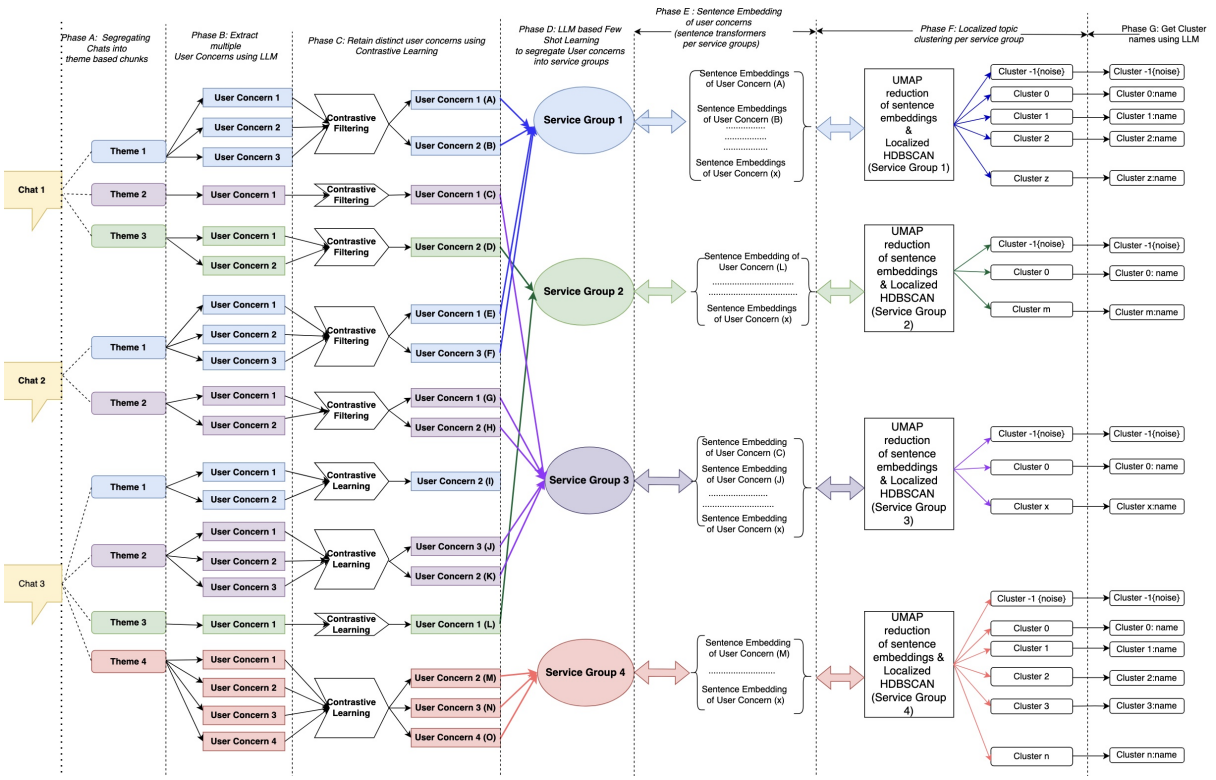

Figure 2: Creation of Base Clusters

ters. (Appendix K) includes examples of contrastive filtering on semantically similar intents.

4. **Service Group Assignment**: Extracted concerns are classified into seven predefined service groups - Compute, Networking, Identity & Security, Storage, Data Services, Billing & Accounts, and Others using few-shot LLM prompts (Fig. 2, Phase D); prompt in (Fig.11 Appendix Q).

5. **Sentence Embedding and Dimensionality Reduction**: Concerns are embedded using the `nli-roberta-base-v2` model (768 dimensions) and reduced via UMAP to improve clustering performance and address the curse of dimensionality (Fig. 2, Phases E–F).

6. **Localized Clustering**: Concerns within each service group are clustered using HDBSCAN, which identifies topic-based clusters in unsupervised manner as shown in Fig 2 (Phase F).

7. **Cluster Title and Description**: Each cluster is labeled with a title and description using LLM (Fig. 2, Phase G; prompt in (Fig.12 Appendix Q).

Clustering quality is tracked using DBI, Silhouette Score, and Centroid-Based Cohesion Score.

3.2 Incremental Clustering

As new user concerns arise, our framework avoids disruptive full re-clustering by incrementally assigning them to existing clusters. This preserves cluster continuity and reduces computational overhead. To manage evolving concern clusters over time, our framework incorporates a full lifecycle-aware approach including splitting, merging, pruning, role assignment, and drift explanation. The process includes:

1. **Concern Extraction and Filtering**: Incremental concerns are segmented, extracted, contrastively filtered, and classified into service groups using step 1- 4 of Base Cluster creation as described previously. During incremental processing, concern-level deduplication is applied within each chat to ensure that only unique concerns are forwarded for service group classification and cluster assignment. This step prevents redundant inclusion of paraphrased or repeated issues while preserving cross-session diversity and maintaining a balanced representation of distinct user problems across incremental updates.

2. **Cluster Assignment via Hybrid Scoring + LLM Confirmation**: To assign new concerns to the most relevant existing clusters, we employ a hybrid two-stage strategy that balances speed, scalability, and semantic accuracy. Since each incremental concern is already mapped to a specific service group, all matching operations are restricted to clusters within that same group.

   (a) **Stage 1 – Embedding-Based Similarity Filtering**:Each new concern is encoded using a sentence embedding model (see step 5: Base Cluster creation). Its embedding is compared against precomputed centroid embeddings of cluster using cosine similarity. Top 5 most similar clusters are shortlisted. This step narrows the LLM's search space, improving efficiency while maintaining high recall.

   (b) **Stage 2 – LLM-Based Semantic Confirmation**:The shortlisted clusters and the new concern are passed into a prompt-driven language model (*cohere.command-r-08-2024 v1.7*). The LLM selects the most appropriate cluster based on natural language understanding, capturing nuance and domain context. It also provides a rationale for its choice to support transparency and auditing.

   This **hybrid scoring + LLM confirmation** approach improves precision for edge cases, emerging topics, and ambiguous inputs, while reducing over-reliance on embeddings or costly full LLM evaluation. (Fig.13 Appendix Q) .

3. **Handling Unassigned Concerns**: Concerns without a match remain in an unclustered pool until enough similar concerns accumulate to form a new cluster, ensuring that emerging topics are identified and tracked over time using LLM.

LLM-based concern matching offers deeper semantic understanding than embedding-based methods, enabling more accurate and efficient incremental assignments. After incremental assignments are completed, the system recalculates DBI and Silhouette Scores at the service-group level to track overall stability, and updates Centroid-Based Cohesion Scores only for affected clusters. If quality degrades, LLM-driven splitting is triggered to preserve cluster coherence. Fig 14 in Appendix Q is LLM prompt for splitting a cluster.

3.3 Evolving New Clusters

Unassigned concerns are placed in an "Others" pool. After each incremental run, an LLM prompt (Fig. 15, Appendix Q) checks whether at least 10 similar concerns can form a meaningful new cluster, avoiding low-impact groupings. These new clusters help surface emerging issue types, such as those introduced by new features.

3.4 Cluster Refinement: Splitting Clusters

To handle topic drift and evolving concerns, we monitor cluster quality and trigger refinement when needed.A split is initiated when a cluster becomes overly broad, as described in Algorithm 1. A cluster is flagged for review if its service group has DBI > 0.5 or Silhouette < 0.5. We compute the Centroid-Based Cohesion Score:

$$C_i = \frac{1}{|X_i|} \sum_{x \in X_i} d(x, \mu_i), \tag{1}$$

where C_i is the cohesion score for cluster i, X_i is the set of data points in cluster i, μ_i is the centroid of cluster i, and $d(x, \mu_i)$ is the Euclidean distance between a point x and the centroid. A high cohesion score indicates dispersed concerns and potential topic drift. We compute a Z-score against historical cohesion to detect abnormal deviations and determine if a split is warranted.

$$Z_i = \frac{C_i - \mu_{C_{i-1}}}{\sigma_{C_{i-1}}}, \tag{2}$$

where C_i is the new cohesion score, $\mu_{C_{i-1}}$ is the previous mean cohesion score, and $\sigma_{C_{i-1}}$ is the standard deviation. If $Z_i \geq 2$, the cluster is split using an LLM, which reassigns concerns into coherent sub-clusters based on semantic similarity. This preserves explainability while adapting to drift without full re-clustering.

3.5 Drift Narrative Generation

To improve explainability, we generate a brief LLM-based narrative after each cluster split. The model is prompted with concerns before and after the split, as well as summaries of the resulting subclusters. It produces a short explanation highlighting the thematic divergence and distinguishing features of the new clusters. These narratives are

archived and optionally surfaced in dashboards to help reviewers trace the evolution of concern topics. Fig 16 in Appendix Q is LLM prompt for splitting a cluster.

3.6 Cluster Lifecycle Management: Merge and Prune

To prevent cluster fragmentation and ensure long-term cohesion, we incorporate a lightweight LLM-guided module to manage cluster lifecycle through merging and pruning. This is especially critical in production settings, where redundant clusters degrade explainability and overwhelm downstream workflows.

Merge candidates are identified using centroid embedding similarity. If two clusters within the same service group have cosine similarity above a threshold (empirically set at 0.92), they are passed to an LLM prompt along with their names, summaries, and sample concerns. The LLM determines whether they are semantically overlapping enough to warrant merging and provides a justification. This process ensures that only truly redundant clusters are consolidated, preserving granularity where needed.

Conversely, clusters that have received no new concern assignments for a 30-day window and fall below a minimum concern count (e.g., 10) are considered for pruning. A secondary LLM check validates whether the cluster represents an outdated or incoherent topic. Pruned clusters are archived but not deleted, maintaining historical traceability.

This merge–prune module maintains a stable, interpretable cluster space over time while minimizing unnecessary fragmentation. Fig 17 in Appendix Q is LLM prompt for splitting merges.

3.7 Cluster Lifecycle Role Assignment

To enhance explainability and enable long-term cluster lifecycle tracking, we introduce a lightweight cluster role categorization scheme. Each cluster is assigned one of four roles: Core, Emerging, Peripheral, or Deprecated, based on its age, assignment frequency, semantic cohesion, and drift history. Core clusters represent stable, high-traffic topics with sustained relevance; Emerging clusters are recently formed with rising activity; Peripheral clusters are small or low-cohesion groups; and Deprecated clusters show inactivity or semantic decay. These roles provide downstream users with intuitive life-cycle cues and enable prioritization, monitoring, and dashboard summarization at scale without additional supervision.

Algorithm 1 Triggering a Cluster Split

Input: Service Group S, Cluster c, Cohesion Scores C_i
Output: Updated cluster assignments
Get Precomputed DBI and Silhouette Score for S
if DBI > 0.5 or Silhouette Score < 0.5 **then**
    Get C_i for current iteration
    Compute $\mu_{C_{i-1}}$ and $\sigma_{C_{i-1}}$ for previous iteration from database
    Compute Z-score using above values
    if $Z_i \geq 2$ **then**
        LLM receives title for concerns in c
        LLM generates new split clusters based on semantic similarity
        Titles and descriptions for split clusters
    end if
end if
Return updated clusters

4 Experiments and Results

4.1 Experimental Setup

We evaluate our framework on 90,048 anonymized multi-turn chat sessions (Apr-Sep 2024), each tagged into one of seven service groups: Compute, Networking, Identity & Security, Storage, Billing & Account, Data Services, and Others. LLM-based segmentation and concern extraction yield almost 148,200 unique concerns for clustering. During Oct-Dec 2024, 400-500 new chats are processed daily via incremental updates.

All LLM tasks (segmentation, concern extraction, service group classification) use the use the *cohere.command-r-08-2024 v1.7* model, selected after comparing four models (*cohere.command-r-plus-08-2024 v1.6, cohere.command-r-08-2024 v1.7, meta.LLaMA 3.3-70B-instruct, meta.LLaMA 3.1-405B-instruct*) on 10,000 chats across service groups Refer Appendix (M). To reduce hallucinations, we apply structured prompts, windowed context, contrastive filtering, and post-hoc validation using domain-specific heuristics. Human evaluations further validated LLM reliability (Appendix K and AppendixL), ensuring outputs are grounded and cluster-ready. We also track lifecycle transitions, cluster merges, pruning events, and LLM-generated narratives during the 90-day incremental window

4.2 Evaluation Metrics

Our pipeline involves both classification and clustering. We use the following standard metrics:

- **Service Group Classification:** Precision, Recall, and F1 Score assess the few-shot LLM's ability to assign concerns to correct service groups, ensuring balanced evaluation across categories (Manning et al., 2008). Service group assignments were further validated using metadata from escalated chats that resulted in formal support tickets, rather than relying on LLM-based judgments. This grounding in verified enterprise outcomes provides an objective benchmark for evaluating classification accuracy and ensures that measured performance reflects real operational correctness.
- **Clustering Evaluation: Silhouette Score** evaluates intra-cluster similarity vs. inter-cluster difference (Rousseeuw, 1987b) **Davies-Bouldin Index (DBI)** captures cluster compactness and separation (lower is better). **Centroid Based Cohesion Score** tracks internal spread via average distance to centroid, useful for monitoring cluster drift (Xu and Wunsch, 2005)
- **Lifecycle-Aware Metrics** To assess our cluster lifecycle modules such as splitting, merging, pruning, and role tracking, we introduce the following metrics:
  - Merge Impact (ΔSilhouette, ΔDBI): Measures improvement in clustering quality after LLM-guided merges.
  - Cluster Stability: Percentage of clusters persisting across multiple time windows.
  - Role Distribution: Counts of clusters in each role (Core, Emerging, Peripheral, Deprecated) over time.
  - Role Transitions: Tracks how clusters evolve across roles (e.g., Emerging $\rightarrow$ Core).
  - Drift Narrative Clarity (Optional): Human-rated scores (1–5) evaluating the clarity and insightfulness of LLM-generated drift explanations.

These metrics offer a lightweight yet effective lens into cluster evolution, explainability, and long-term system robustness.

4.3 Results & Discussion

Classification Performance and Concern Distribution. Figure 3 summarizes few-shot classification metrics across service groups, along with the number of extracted concerns per class.The high F1 scores (>0.85) across all classes indicate the LLM generalizes well, enabling reliable routing to service-group specific clustering.

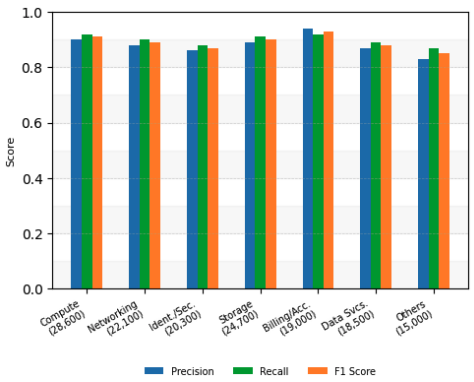

Figure 3: Few-shot classification metrics by Group

Base Clustering Evaluation We apply HDBSCAN with UMAP to perform localized clustering within each service group using sentence embeddings. Table 2 shows strong cohesion and separation across domains.

We apply UMAP + HDBSCAN clustering per service group to retain domain specificity and explainability. Table 2, yields strong cohesion and separation improving Silhouette by +0.44 (111.7%) and reducing DBI by 65.6% over the global KMeans baseline (Table 3).

Service Group	User Concern Count	Cluster Count	Silhouette	DBI
Compute	18765	59	0.73	0.47
Networking	22341	68	0.72	0.36
Ident./Sec.	25865	119	0.73	0.49
Storage	26711	96	0.71	0.43
Billing/Acc.	19876	89	0.77	0.44
Data Svcs.	24598	78	0.72	0.52
Others	10044	105	0.69	0.558
Average			**0.72**	**0.46**

Table 2: Base clustering metrics - HDBSCAN + UMAP.

Method	Silhouette	DBI
KMeans + BERT embeddings	0.28	1.34
HDBSCAN only	0.34	1.12
HDBSCAN + UMAP	**0.72**	**0.46**

Table 3: Comparison of clustering methods.

The dataset is moderately balanced across service groups, ranging from 10K ("Others") to 27K ("Storage") concerns, forming 59–119 clusters

each. This mild imbalance does not affect clustering quality.

HDBSCAN, in combination with UMAP for dimensionality reduction, adapts density-based clustering thresholds locally and does not require uniform cluster sizes. UMAP preserves semantic structure, enabling HDBSCAN to adapt to varying cluster sizes. Average metrics remain stable (Silhouette = 0.72; DBI = 0.46), validating robustness.

UMAP preserves the semantic structure of high-dimensional embeddings, allowing HDBSCAN to discover dense topic-specific groupings even in smaller or sparser groups. As shown in Table 2, clustering performance remains consistent across groups (average Silhouette Score = 0.72; average DBI = 0.46), indicating that our approach is robust to moderate imbalance without requiring rebalancing.

Incremental Clustering Results Over 90 days, new chats are incrementally clustered using LLM-based semantic matching without full re-clustering.Figure 4 shows cluster quality (Compute group) from Day 1 to Day 90. Across service groups, cluster metrics remain stable over time. Appendix J Figure 8. Gradual DBI increase and Silhouette decline indicate topic drift; once thresholds are exceeded, LLM-based refinement restores scores within ±20% of baseline. Figure 5 confirms quality degrades without refinement, validating the system's drift detection and adaptive response.

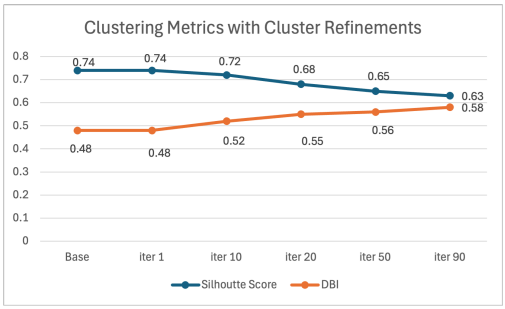

Figure 4: Incremental Clustering Metrics for "Compute with Cluster Refinement"

4.4 Cluster Merge and Prune Evaluation

During the 90-day window, we observed frequent opportunities for consolidation and cleanup. We identified 43 merge candidates using centroid similarity (cos > 0.92), of which 31 were approved by LLM. Refer Figure 6 Post-merge, average Silhouette improved from 0.63 to 0.7 and DBI dropped from 0.58 to 0.52 at 90th iteration. This shows

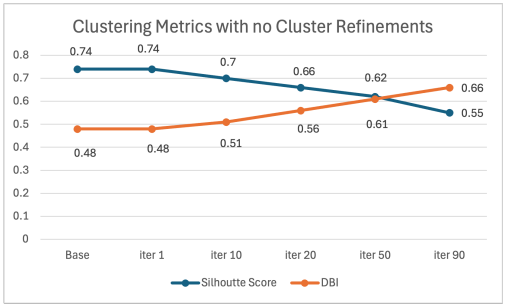

Figure 5: Incremental Clustering Metrics for "Compute" using no Cluster Refinement

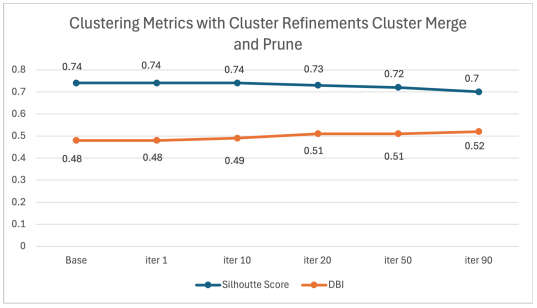

Figure 6: Cluster Merge and Prune

that LLM-guided merging removes redundancy and improves cohesion. 12 low-activity clusters were pruned after 30+ days of inactivity and confirmed by LLM to be obsolete.

4.5 Qualitative Example: Cluster Refinement

In Billing/Account, a broad cluster covering refunds, invoice errors, and renewals was refined into three distinct sub-clusters. Triggered by cohesion-based Z-score alerts, the LLM-based split improved clarity and downstream labeling. See Appendix G for full example.

4.6 Qualitative Example: Cluster Merge and Prune

In Billing/Account, two clusters one focused on refund delays and the other on non-receipt were identified as semantically overlapping. Despite slight differences in phrasing, both described the same user concern: a refund had been approved but not received. The LLM-guided merge consolidated these into a single, clearer cluster titled "Refund Not Received or Delayed," reducing redundancy . See Appendix H for full example.

4.7 Topic Drift Frequency and Examples

Over 90 days, recurring topic drift (Z-score 2) prompted 61 cluster refinements across 7 service

groups(5–15 splits each) (avg. 8.7), Data Services and Others has frequent drift. Appendix E

4.8 Error Analysis and Ablation

Manual inspection of 600 classification cases revealed most errors (õ60%) in “Others” and “Data Services,” due to vague or overlapping language. Examples:

- *"I’m not getting the output I expected from the portal"* lacks service-specific cues, resulting in misclassification under "Others" instead of "Data Services (Analytics)".
- "*I need help with my pipeline performance*" The phrasing is vague and could refer to Data Services or Others.

Misclassification rate: 7.6%. Clustering drift was corrected via LLM-based refinement. Ablation studies (Appendix B) show UMAP boosts clustering (+0.38 Silhouette, –0.66 DBI), and LLM-based matching outperforms cosine similarity (36% better Silhouette Removing any single component concern extraction, contrastive filtering, or LLM-based classification reduced clustering quality (avg. Silhouette 0.21, DBI +0.72), confirming each module’s unique value to framework stability and explainability.

5 Synthetic Dataset Validation

We have released the synthetic/sanitized dataset[1],that is distributionally aligned and replicates the enterprise dataset structure. The synthetic dataset follows the similar service-group distribution as observed in our enterprise corpus (Compute 12.7 %, Networking 15%, Identity Security 17%, Storage 18.0 %, Billing Account 15%, Data Services 16.6%, Others 6.8%), yielding approximately 1300-1800 synthetic chats per major service group and 680 for “Others.” This mirrors the 148,000 concern enterprise dataset distribution to maintain clustering comparability. To evaluate the fidelity of the synthetic dataset, we split the dataset into 8,000 base chats and 2,000 incremental synthetic chats and compared clustering outcomes of these base chats and incremental synthetic chats against those obtained from the real enterprise dataset. As shown in (Appendix P) Table 22, the synthetic clusters exhibit close alignment in both separation and cohesion metrics (Average DBI = 0.48 (Synthetic dataset) vs. 0.46 in (Enterprise Data)) and (Average Silhouette = 0.70 (Synthetic dataset) vs. 0.72 (Enterprise Data)).

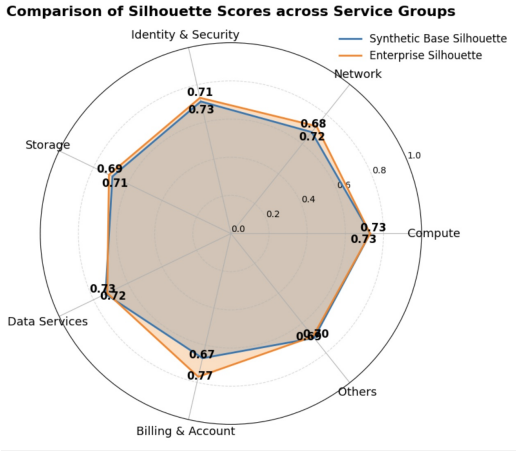

Figure 7: Comparison of Silhouette Scores across Service Groups Enterprise VS Synthetic dataset

6 Conclusion

We present a lifecycle-aware framework for clustering user concerns from multi-turn support chats, combining LLM-based segmentation, contrastive filtering, and unsupervised clustering with adaptive, interpretable refinement. Unlike static or embedding-only methods, our approach supports real-time updates, drift detection, LLM-guided cluster splitting, merging, pruning, and role tracking. Experiments on 90k+ enterprise chats over 90 days demonstrate strong cluster quality, semantic coherence, and robustness to topic drift. Lifecycle operations including 61 splits, 34 merges, and 12 prunes, enable long-term cluster stability while LLM-generated narratives and role labels improve explainability. Together, these components form a scalable, auditable solution for evolving concern management in production environments.

Limitations

While the framework shows strong performance, it has a few limitations. Ambiguous or compound concerns spanning multiple services (e.g., compute and storage) can still challenge classification. Evaluation relies primarily on internal metrics and qualitative inspection; future work could include human-in-the-loop or business impact metrics. Finally, the current system supports only English, and extending to multilingual chats is an important next step.

[1] https://github.com/Synthetic-Datasets-sudo

References

Amit Agarwal, Srikant Panda, Angeline Charles, Bhargava Kumar, Hitesh Patel, Priyanranjan Pattnayak, Taki Hasan Rafi, Tejaswini Kumar, and Dong-Kyu Chae. 2024a. Mvtamperbench: Evaluating robustness of vision-language models. arXiv preprint arXiv:2412.19794.

Amit Agarwal, Hitesh Patel, Priyaranjan Pattnayak, Srikant Panda, Bhargava Kumar, and Tejaswini Kumar. 2024b. Enhancing document ai data generation through graph-based synthetic layouts. arXiv preprint arXiv:2412.03590.

Charu C. Aggarwal and ChengXiang Zhai. 2012. A Survey of Text Clustering Algorithms. Springer.

Mebarka Allaoui, Mohammed Lamine Kherfi, and Abdelhakim Cheriet. 2020. Considerably improving clustering algorithms using umap dimensionality reduction technique: A comparative study. In Image and Signal Processing: 9th International Conference, ICISP 2020, Marrakesh, Morocco, June 4–6, 2020, Proceedings, page 317–325, Berlin, Heidelberg. Springer-Verlag.

M. Bentley and S. Batra. 2016. Giving voice to office customers: Best practices in how office handles verbatim text feedback. In Proceedings of the IEEE International Professional Communication Conference (ProComm).

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. Journal of Machine Learning Research, 3:993–1022.

David L. Davies and Donald W. Bouldin. 1979. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-1(2):224–227.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. ACM Computing Surveys, 46(4):1–37.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Tianyu Gu, Jie Li, and Yixin Zhang. 2022. Who says what to whom: A survey of multi-party conversations. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2021. Supervised contrastive learning for pre-trained language model fine-tuning. In International Conference on Learning Representations (ICLR).

Yang Li, Jian Yang, and Tong Liu. 2021. Adaptive clustering with concept drift detection in data streams. IEEE Transactions on Knowledge and Data Engineering.

Xuewei Ma, Kai Zhao, and Lei Wang. 2024. Multi-turn dialogue comprehension from a topic-aware perspective. Neurocomputing.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1:281–297.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. Introduction to Information Retrieval. Cambridge University Press.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. Journal of Open Source Software, 2(11):205.

Leland McInnes, John Healy, and James Melville. 2018a. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

Leland McInnes, John Healy, and James Melville. 2018b. Umap: Uniform manifold approximation and projection for dimension reduction. In Proceedings of the 2018 International Conference on Machine Learning (ICML). ArXiv:1802.03426.

Hansa Meghwani, Amit Agarwal, Priyaranjan Pattnayak, Hitesh Laxmichand Patel, and Srikant Panda. 2025. Hard negative mining for domain-specific retrieval in enterprise systems. Preprint, arXiv:2505.18366.

Benjamin Moseley and Joshua Wang. 2017. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS).

Jianmo Ni, Nils Reimers, and Iryna Gurevych. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing.

Hitesh Laxmichand Patel, Amit Agarwal, Arion Das, Bhargava Kumar, Srikant Panda, Priyaranjan Pattnayak, Taki Hasan Rafi, Tejaswini Kumar, and Dong-Kyu Chae. 2025. Sweeval: Do llms really swear? a safety benchmark for testing limits for enterprise use. In Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track), pages 558–582.

Hitesh Laxmichand Patel, Amit Agarwal, Bhargava Kumar, Karan Gupta, and Priyaranjan Pattnayak. 2024. Llm for barcodes: Generating diverse synthetic data for identity documents. arXiv preprint arXiv:2411.14962.

Priyaranjan Pattnayak, Amit Agarwal, Bhargava Kumar, Yeshil Bangera, Srikant Panda, Tejaswini Kumar, and

Hitesh Laxmichand Patel. Review of reference generation methods in large language models. Journal ID, 9339:1263.

Priyaranjan Pattnayak, Amit Agarwal, Hansa Meghwani, Hitesh Laxmichand Patel, and Srikant Panda. 2025a. Hybrid ai for responsive multi-turn online conversations with novel dynamic routing and feedback adaptation. In Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing, pages 215–229.

Priyaranjan Pattnayak and Hussain Bohra. 2025. Review of tools for zero-code llm based application development. Preprint, arXiv:2510.19747.

Priyaranjan Pattnayak, Hitesh Patel, and Amit Agarwal. 2025b. Tokenization matters: Improving zero-shot ner for indic languages. In 2025 IEEE International Conference on Electro Information Technology (eIT), pages 456–462. IEEE.

Priyaranjan Pattnayak, Hitesh Laxmichand Patel, and Amit Agarwal. 2025c. Tokenization matters: Improving zero-shot ner for indic languages. Preprint, arXiv:2504.16977.

Priyaranjan Pattnayak, Hitesh Laxmichand Patel, Amit Agarwal, Bhargava Kumar, Srikant Panda, and Tejaswini Kumar. 2025d. Clinical qa 2.0: Multi-task learning for answer extraction and categorization. Preprint, arXiv:2502.13108.

Priyaranjan Pattnayak, Hitesh Laxmichand Patel, Bhargava Kumar, Amit Agarwal, Ishan Banerjee, Srikant Panda, and Tejaswini Kumar. 2024. Survey of large multimodal model datasets, application categories and taxonomy. arXiv preprint arXiv:2412.17759.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, pages 3982–3992. Association for Computational Linguistics.

Aditya Rohit, Anand Tripathy, and Partha Talukdar. 2022. Lifelong clustering with adaptive memory for evolving data streams. In Proceedings of the AAAI Conference on Artificial Intelligence.

Peter J. Rousseeuw. 1987a. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65.

Peter J Rousseeuw. 1987b. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In Proceedings of the 2021 Conference on Neural Information Processing Systems Datasets and Benchmarks.

Rui Xu and Donald Wunsch. 2005. Survey of clustering algorithms for evolving data streams. IEEE Transactions on Neural Networks, 16(3):635–645.

Tian Ye and Daniel Johnson. 2024. User intent and state modeling in conversational systems. University College London Research Archive.

Lijun Zhang, Zhenfeng Li, and Xuejun Jin. 2018. Incremental clustering with support vector machines. Pattern Recognition Letters, 110:22–29.

Wenxin Zhu, Ling Han, and Jialin Wang. 2024. Multi-turn dialogue understanding for customer support. arXiv preprint.

A Appendix: Terminology and Glossary

To improve clarity and avoid ambiguity, we define several key terms used throughout the paper. These terms describe the structure and categorization of customer support conversations in cloud service environments.

- Multi-turn, Multi-service Chat: A multi-turn, multi-service (Pattnayak et al., 2025d) chat is a customer support conversation that involves multiple back-and-forth exchanges (multi-turn) between a user and an agent, and spans multiple cloud service areas (multi-service) within the same session.

  For example, a customer might begin a chat about a virtual machine that won't start (Compute), then ask about related firewall settings (Networking), and finally inquire about unexpected charges (Billing and Account). These topic shifts occur naturally in real-world support chats and pose challenges for traditional clustering methods, which often treat the entire conversation as a single unit.

- Theme: A theme is a coherent segment within a multi-turn conversation that centers around a single topic or line of discussion. Themes are extracted by detecting topic shifts, and may contain one or more related concerns. A chat can have multiple 'themes'. In above example, there are three themes: 1) *Compute*, 2) *Networking* and, 3) *Billing and Account*.

- Concern: A concern is a distinct user issue, request, or problem described within a customer support conversation. One chat session

can contain multiple concerns (e.g., virtual machine boot failure, billing inquiry), each representing a specific topic or task. A theme can have multiple 'concerns'.

- Service Group (or Domain): A service group (also referred to as a domain) is a predefined cloud service category used to organize concerns (e.g., Compute, Networking, Identity & Security). Concerns are classified into these groups for structured clustering and analysis (Pattnayak and Bohra, 2025).

B Appendix: Ablation Study of Effect of UMAP

We see significant improvement, shown in 4, in Overall Silhouette Score (+0.38) and DBI improved by 0.66 with UMAP Reduction before clustering as supported by (Allaoui et al., 2020). Group wise scores are shown in Table 11.

Configuration	Silhouette Score	DBI
HDBSCAN only	0.34	1.12
HDBSCAN + UMAP	**0.72**	**0.46**

Table 4: Impact of UMAP dimensionality reduction.

C Appendix: Ablation Study of LLM Guided Incremental

Table 5 shows Clustering metrics of incremental concerns being assigned to Base clusters using LLM based matching (*our proposed solution*) and Centroid based Cosine similarity

Method	Silhouette	DBI
LLM Matching (ours)	0.72	0.46
Cosine Similarity Match	0.53	0.81

Table 5: Incremental assignment: LLM-based vs cosine similarity matching on "Compute" Service group on Day 30.

D Appendix: Error Distribution by Service Group

We analyzed 600 samples for misclassification and only observed 7.67% errors, with 60% error contributed from Data Services and Others Service Groups as shown in Table 6.

Service Group	Errors (#)	Error Rate (%)
Compute	4	0.67
Networking	5	0.83
Identity & Security	3	0.50
Storage	4	0.67
Billing/Account	3	0.50
Data Services	13	2.17
Others	14	2.33
600 Samples	**46**	**7.67%**

Table 6: Observed classification errors from 600 manually reviewed samples.

E Appendix: Cluster Split Logs

We observed a total of 46 cluster splits across all the 7 service groups with highest splits occurring in Data Services which can be attributed to rapidly growing number of services and features in Database and AI services during the last year. Table 7 shows group wise splits during the 90 days of incremental clustering.

Service Group	Split Events (90 days)
Compute	8
Networking	7
Identity & Security	5
Storage	6
Billing & Account	9
Data Services	15
Others	11
Total	**61**

Table 7: Number of cluster splits automatically triggered via Z-score.

F Appendix: Cluster Role Categorization and Transitions

Each cluster is assigned a lifecycle role (Core, Emerging, Peripheral, Deprecated). Table 8 shows distribution over time. 68% of Emerging clusters became Core within 30 days; 14% of Core clusters transitioned to Deprecated. This confirms the role framework reflects real concern dynamics.

Role	Day 30	Day 60	Day 90
Core	45	59	67
Emerging	22	16	11
Deprecated	3	8	12

Table 8: Cluster Role Categorization and Transitions

G Appendix: Cluster Refinement Example

Table 9 shows the case of a cluster within "Billing & Accounts" service group which had issues from related but different pain points, which resulted in the cluster getting split. Three new clusters were created post refinement process.

Before Split	*After LLM-Driven Split*
Refund delays, invoice errors, auto-renewal issues, discount codes not applying	New Cluster 1: (Refunds, invoice errors) New Cluster 2: Auto-renewal failures New Cluster 3: Discount issues

Table 9: LLM refinement of a noisy cluster.

H Appendix: Cluster Merge Example

Table 10 shows two clusters were phrased slightly differently; one emphasizing non-receipt, the other focusing on processing delays. However semantically describe the same user issue: a refund has been requested but hasn't arrived.

Before Merge	After LLM- Driven Merge
Base Cluster A: Refund Not Received Base Cluster B:Delayed Refund Processing	Refund Not Received or Delayed

Table 10: LLM driven cluster merge

I Appendix: Base Cluster Creation

Step by step base cluster creation process is depicted in Fig 2. A multi-turn chat is first segregated into theme based chunks in Phase A. In Phase B, an LLM extracts all the user concerns from these themes. There could be several concerns in each theme. Phase C uses contrastive filtering to remove duplicate user concerns from the same theme to ensure we have distinct user concerns per theme. In Phase D, a few-shot LLM classifies the user concerns into one of the 7 different service groups. Phase E generates sentence embedding for the user concerns under each service group followed by Phase F where UMAP reduces the dimensions of embeddings and HDBSCAN clusters user concerns into specific clusters. Finally, in Phase G, LLM generates a cluster title/name and cluster description using user concerns from each cluster.

J Appendix: Clustering Metrics of Different Service groups per iteration

We observe that Silhouette Scores and DBI remain stable and within +20% of initial base cluster values throughout 90 incremental iterations as shown in Fig 8. This proves the efficiency of cluster refinement process which triggers whenever it observes degradation in cluster metrics and quality during incremental step.

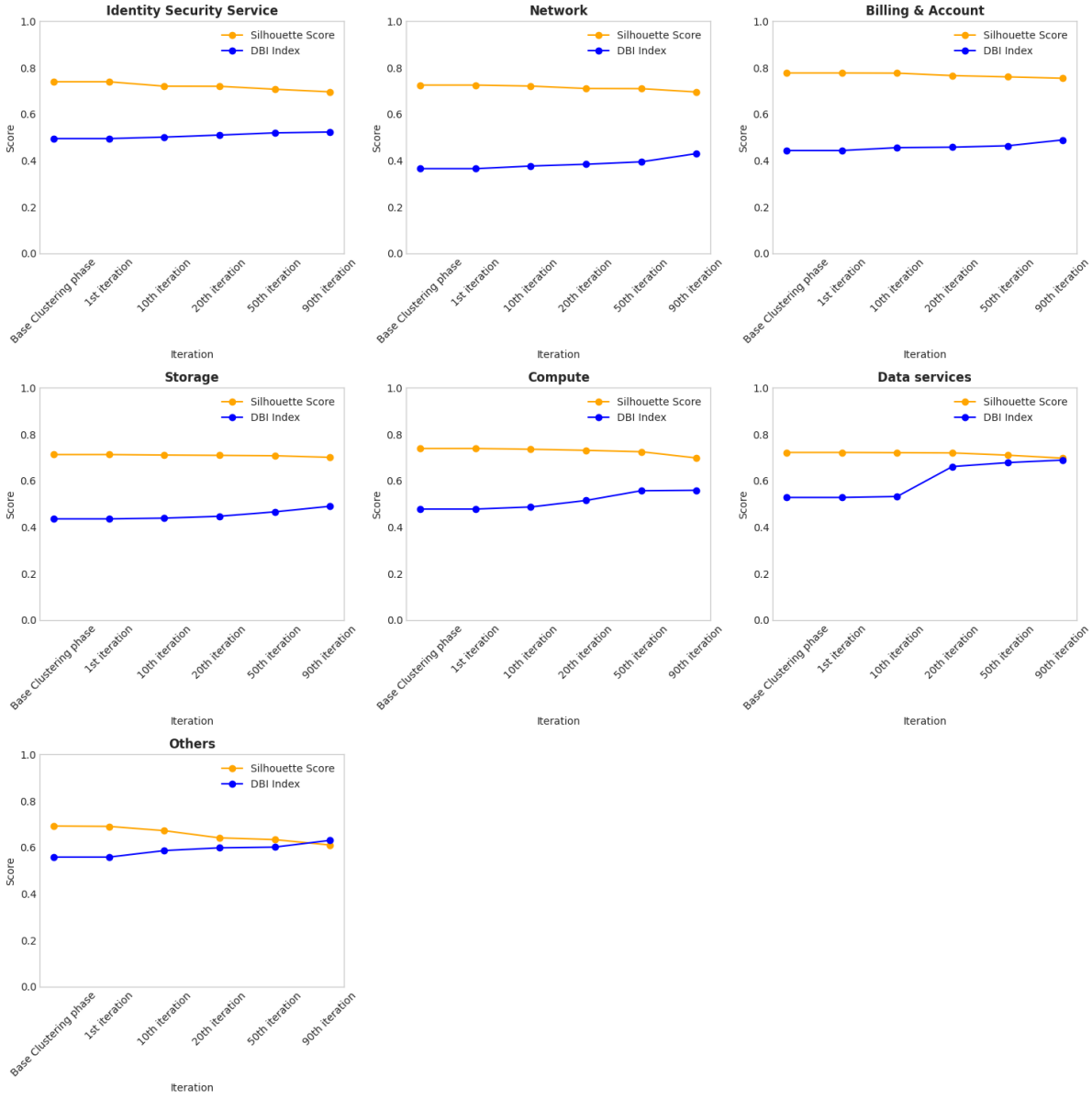

Figure 8: Clustering Metrics of Different Service groups per iteration

Service Team	Groupwise HDBSCAN	Groupwise HDBSCAN with UMAP
Identity & Security	Silhouette: 0.4984, DBI: 0.7946	Silhouette: 0.7394, DBI: 0.4946
Networking	Silhouette: 0.4502, DBI: 0.7452	Silhouette: 0.725, DBI: 0.3652
Billing & Account	Silhouette: 0.4771, DBI: 0.7432	Silhouette: 0.7771, DBI: 0.4432
Storage	Silhouette: 0.5026, DBI: 0.757	Silhouette: 0.7126, DBI: 0.4357
Compute	Silhouette: 0.4689, DBI: 0.7532	Silhouette: 0.7389, DBI: 0.478
Data Services	Silhouette: 0.521, DBI: 0.8043	Silhouette: 0.7218, DBI: 0.528
Others	Silhouette: 0.4532, DBI: 0.848	Silhouette: 0.6918, DBI: 0.558

Table 11: Clustering performance comparison with and without UMAP dimensionality reduction

K Appendix: Examples of Contrastive Filtering on Intents with Similar Semantics

The table 12 shows how contrastive filtering uses cosine similarity scores to decide which intent pairs to keep. Intents with similarity above 0.95 are considered duplicates, so only one is retained; pairs below this threshold are both kept. Using an annotated dataset of 100 concerns with strong inter-annotator agreement (Cohen's Kappa = 0.79), contrastive filtering correctly identified 89% of semantically similar intents, effectively removing duplicates and preserving unique entries.

L Appendix: Concern Extraction Evaluation

This appendix presents the evaluation of the LLM-based concern extraction module against a manually annotated dataset of 150 conversation segments. It includes performance metrics and examples comparing model-extracted concerns with ground-truth annotations to illustrate precision, recall, and common error cases.

- **Exact Match**: Concerns match word-for-word or very closely.
- **Semantic Match**: Conceptually equivalent, but phrased differently.
- **Partial Match**: One concern matches, the other is missed or mismatched.
- **Missed**: Model failed to extract any concern present in the ground truth.
- **Spurious Output**: Model generated concern not present in ground truth.

Table 13 compares ground truth concerns with those extracted by the LLM for various conversation segments, categorizing the matches as exact, partial, semantic, missed, or spurious to illustrate extraction accuracy.

Table14 summarizes the performance of the LLM-based concern extraction module compared to human-annotated ground truth on 150 segments. It reports standard evaluation metrics precision, recall, and F1 score along with 95% confidence intervals, and shows strong alignment with human annotations. The inter-annotator agreement Cohen's $\kappa = 0.79$ reflects high consistency between annotators, validating the reliability of the dataset.

Reference Intent	Candidate Intent	Cosine Similarity	Action
I want to cancel my subscription.	Please stop my membership.	0.44	retain both intents
App keeps crashing on startup.	App crashing every time.	0.956	retain only 1 intent
Change my delivery address.	Update tenancy location.	0.583	retain both intents
Refund not processed after 7 days.	Still waiting for my refund.	0.710	retain both intents
Need help with password issues, can't login.	Forgot my password, can't log in.	0.963	retain only 1 intent
Please reset my account password.	Reset password for account access.	0.976	retain only 1 intent

Table 12: Examples of Contrastive Filtering on Intents with Similar Semantics

Segment ID	Ground Truth Concerns	LLM-Extracted Concerns	Match Type
S001	1. VM crash 2. Cannot connect to storage	1. VM crash 2. Storage access issue	Exact Match
S014	1. Billing confusion 2. Refund request	1. Billing confusion 2. Request for compensation	Partial Match
S023	1. Unable to reset password	1. Password reset not working	Semantic Match
S035	1. Data loss after update 2. No backup option	1. Lost files 2. No backup setting	Exact Match
S048	1. Login error	–	Missed
S057	–	1. Account locked	Spurious Output

Table 13: Concern Extraction Evaluation – Comparison with Annotated Ground Truth

Metric	Score	95% CI	Notes
Precision	0.86	[0.83, 0.89]	LLM vs. gold-standard concerns
Recall	0.82	[0.79, 0.85]	Captures partial & full matches
F1	0.84	[0.81, 0.86]	Harmonic mean of precision and recall
Inter-Annotator Agreement	0.79		Agreement between two human annotators

Table 14: Concern Extraction Evaluation against Human-Annotated Data

M Appendix: Model Selection Study for theme segmentation, concern extraction, and and service group classification.

Table 15 shows comparative analysis was conducted across four LLM-based pipelines, each utilizing a different language model to perform the key tasks of **theme segmentation**, **concern extraction**, and **service group assignment**. After these LLM-driven steps, each pipeline applied **localized clustering** using HDBSCAN with UMAP-based dimensionality reduction to group similar concerns.

To evaluate the clustering quality, we processed 10,000 customer chats spanning 10 service categories through each pipeline. Clustering performance was assessed using two standard metrics: the **Davies–Bouldin Index (DBI)** and the **Silhouette Score**. Lower DBI values and higher Silhouette Scores indicate better clustering performance, signifying that the resulting clusters are both compact (internally coherent) and well-separated (distinct from one another).

N Appendix: Extended Ablation Studies

This section reports additional ablations evaluating the impact of concern extraction, contrastive filter-

Service Group	LLaMA 3.3 (70B) DBI / Silhouette	LLaMA 3.1 (405B) DBI / Silhouette	Cohere R DBI / Silhouette	Cohere R+ DBI / Silhouette
Identity & Security	0.44 / 0.71	0.43 / 0.73	**0.38 / 0.76**	0.41 / 0.70
Billing & Account	**0.33 / 0.72**	0.46 / 0.53	0.38 / 0.71	0.45 / 0.71
Compute	0.57 / 0.69	0.35 / 0.53	**0.25 / 0.68**	0.39 / 0.67
Data Services	0.72 / 0.38	0.68 / 0.52	0.62 / 0.50	**0.62 / 0.53**
Storage	**0.38 / 0.74**	0.49 / 0.70	0.46 / 0.73	0.49 / 0.72
Networking	0.60 / 0.57	0.79 / 0.30	**0.45 / 0.68**	0.45 / 0.65
Others	0.56 / 0.65	**0.13 / 0.88**	0.29 / 0.62	0.58 / 0.57

Table 15: Cluster Quality Comparison across LLM Pipelines (DBI / Silhouette Score). Lower DBI shows clusters are well-separated and internally compact. Higher Silhouette means points are close to their own cluster and far from others.

ing, and LLM-based service group classification.

N.1 Concern Extraction

Removing LLM-based concern extraction and clustering raw utterances reduces cluster quality, confirming the importance of concern-level segmentation. Table 16 for reference.

N.2 Contrastive Filtering

Omitting duplicate removal degrades cohesion and interpretability. Table 17 for reference.

N.3 LLM-Based Service Group Classification

Replacing few-shot LLM classification with embedding-only matching lowers service-group F1 and cluster purity. Table 18 for reference.

O Appendix: Computational Cost and Scalability

To assess the scalability and practical feasibility of the proposed framework, we report an approximate breakdown of computational cost across three phases: (1) Base Clustering, (2) Incremental Clustering, and (3) Lifecycle Management. All costs are estimated using publicly available token-based (Pattnayak et al., 2025c,b)pricing for comparable LLM APIs and are presented in USD equivalents. Exact pricing cannot be disclosed to preserve vendor confidentiality.

Token Length Considerations:

- Input tokens per call: 4,500–6,500 (conversation snippet + instructions)
- Output tokens per call: 2,000–3,500 (theme/concern/label text)
- Total tokens per call: 6,500–8,500, well within Cohere Command-R's 16k+ token context window.

The complete base clustering of approximately 90,000 chats involves around 180,000 LLM API calls, representing a one-time cost of roughly US$60–$70. Subsequent incremental clustering operates efficiently, requiring only about 2,250 API calls per day (approximately US$1.40) to process 500 new chats. Lifecycle management activities—including cluster splitting, merging, pruning, role assignment, and drift narrative generation (Agarwal et al., 2024b) add roughly 2,400 API calls every 90 days, corresponding to an estimated cost of around US$1. All prompts and outputs remain well within model context limits, confirming that the overall framework is computationally feasible and scalable for large-scale enterprise deployment.

O.1 Computation Cost for Base Clustering Phase

Following are one time costs mentioned in Table 19.It processes 90k chats.

O.2 Computation Cost for Incremental Clustering Phase

Incremental Clustering phase processes 500 chats per day. Computation Cost mentioned in Table 20.

O.3 Lifecycle Management Phase (every 90 days)

Refer to table 21

Configuration	Silhouette	DBI	Notes
Full model (with concerns)	0.72	0.46	HDBSCAN + UMAP baseline with concern extraction
No concern extraction	0.43	1.38	Raw utterances clustered; loss of granularity

Table 16: Concern Extraction Ablation

Configuration	Silhouette	DBI	Notes
Full model (with filtering)	0.72	0.46	Semantically coherent clusters
No filtering	0.59	0.72	Larger, noisier clusters; duplicates distort centroids

Table 17: Contrastive Filtering Ablation

Configuration	Silhouette	DBI	Service Group Avg F1	Notes
Full model (LLM classification)	0.72	0.46	0.86	Strong purity, context captured
Embedding-only classification	0.51	1.43	0.62	Misassignments; degraded purity

Table 18: LLM-Based Service Group Classification Ablation

P Appendix: Clustering Metrics Synthetic data vs Enterprise Data

Table 22 shows Clustering metrics for Synthetic data and Enterprise Data.

Q Appendix: LLM prompts

- Figure9 shows LLM prompt used for extracting themes from multi-service chats
- Figure 10 shows LLM prompt used for extracting concerns from segmented chats
- Figure 11 shows LLM prompt used for assigning extracted concerns to service groups using LLM based few shot learning
- Figure 12 shows LLM prompt used for generating cluster name and cluster description after created using HDBSCAN and UMAP based base clusters.
- Figure 13 shows LLM prompt used for assigning incremental concerns to previously created base clusters.
- Figure 14 shows LLM prompt used for splitting clusters.
- Figure 15 shows LLM prompt used for merging previously unclustered clusters.
- Figure 16 shows LLM prompt used for Drift Narrative Generation
- Figure 17 shows LLM prompt used for Cluster Merges

Step	API Calls	Approx. Tokens / Call	Approx. Cost (US$)
Chat → Themes	∼90,000	∼6,500 input	∼29
Themes → Concerns	∼45,000	∼6,500 input	∼15
Concern → Service Group	∼45,000	∼6,500 input	∼15
Cluster Naming & Description	∼614	∼2,000 output	∼2
Total Base Phase	—	—	**∼61**

Table 19: Base Clustering Phase

Step	API Calls	Approx. Tokens / Call	Approx. Cost (US$)
Chat → Themes	∼500	∼6,500 input	∼0.32
Themes → Concerns	∼250	∼6,500 input	∼0.16
Concern → Service Group	∼250	∼6,500 input	∼0.16
Cluster Naming & Description	∼1,250	∼6,500 input	∼0.80
Total Incremental Phase	—	—	**∼1.44**

Table 20: Incremental Clustering Phase

Step	API Calls	Approx. Tokens / Call	Approx. Cost (US$)
Splitting Clusters	∼180	mix of input + small output (∼6,500)	∼0.11
Merging Clusters	∼80	same as ∼6,500	∼0.05
Pruning Clusters	∼12	small output (∼2,000 tokens)	∼0.01
Role Assignment	∼614	likely input tokens ∼6,500	∼0.20
Drift Narratives	∼120	small output ∼2,000 tokens	∼0.02
Total 90-Day Lifecycle Management	—	—	**∼0.39**

Table 21: Lifecycle Management Phase (every 90 days)

	Synthetic Dataset				Enterprise Dataset	
Service	**Base DBI**	**Base Silhouette**	**Incr DBI**	**Incr Silhouette**	**Silhouette**	**DBI**
Compute	0.48	0.73	0.51	0.72	0.73	0.47
Networking	0.47	0.68	0.47	0.68	0.72	0.36
Ident./Sec	0.53	0.71	0.55	0.704	0.73	0.49
Storage	0.5	0.69	0.51	0.68	0.71	0.43
Data Services	0.52	0.73	0.53	0.72	0.72	0.52
Billing account	0.43	0.67	0.52	0.65	0.77	0.44
Others	0.44	0.7	0.45	0.71	0.69	0.558
Average	0.48	0.70	0.5	0.69	0.72	0.46

Table 22: Clustering metrics across service groups for synthetic and enterprise datasets.

PROMPT:
Analyze the following multi-domain conversation and segment it into smaller, domain-specific parts (themes). Each part should focus on one service area. After segmentation, ensure each part clearly identifies the theme it corresponds to and provides the domain name with the content that falls under it. Be sure to preserve the context of the original conversation, including the user ID and the dialogue content, within each segment. For longer conversations, use a windowed prompt approach, breaking the conversation into topical blocks while ensuring there is overlap between windows to preserve context and coherence.

Conversation:
<insert conversation here>

Output Format:

- **Chat user ID**: [user_id]
- **Theme**: [theme 1]
  Segment: [chats related to theme 1]
- **Theme**: [theme 2]
  Segment: [chats related to theme 2]
- **Theme**: [theme 3]
  Segment: [chats related to theme 3]

... (and so on)

Figure 9: LLM Prompt for segmenting multi service chat into themes

PROMPT:
Analyze the following multi-turn conversation and identify all **granular and standalone user concerns**. This conversation focuses on a specific theme (such as billing, technical support, product inquiries, etc.).

Your task is to extract **each distinct concern individually**, even if multiple concerns are expressed in a single user message

Also, **use a windowed context of ±1–2 turns** to capture concerns that unfold over multiple turns. Consider implicit references and follow-ups that depend on earlier context.

List each concern separately along with the Chat User ID.

Segment: *Here, includes the specific chat segment per theme (billing, technical support, etc.).*

Output:

Chat user id – User ID

- ○ **User Concern 1**: [Brief description of the first standalone concern]
- ○ **User Concern 2**: [Brief description of the second standalone concern]
- ○ **User Concern 3**: [Brief description of the third standalone concern]
- ○ **User Concern 4**: [Brief description of the fourth standalone concern]
  (Add more as needed.)

Figure 10: LLM Prompt for extracting user concerns from segmented chats

PROMPT: Assign following sentences into one of the following groups. Choose only from these following groups. If they don't belong to the groups mentioned in the list, assign them to 'Others'.

List of Groups To Choose from :

Storage
Identity and Security
Billing and account service
Network
Data service
Compute Service
Others

Output format for each chat should be as follows only:
" UserID:
Group:"

Example 1: Sentences:
"User experiencing delays in accessing or transferring large volumes of data, affecting application performance."
"Problems with backing up large datasets "
Label: Storage

Example 2:Sentences:
"User needed help resetting their password to access their cloud account and had deleted the Authenticator APP."
"User has lost their password and needs help to reset it or recover their account."
Label: Identity and Security

Example 3:Sentences:
"User has a billing-related inquiry"
"Confusion regarding billing system and credit card authorization."
Label: Billing and account service

Example 4:Sentences:
"User needed help creating a virtual cloud network (VCN) and decided to create a support request."
Label: Network

Example 5:Sentences:
"Seeking assistance on how to restore a deleted DB system."
"User needs help with PostgreSQL and wants to create a support request."
Label: Data Services

Example 6:Sentences:
"User needed clarification on installing Database Firewall on Compute VM."
"User aims to increase compute instance capacity but encounters issues."
Label: Compute Service

Now, classify these sentences into one of the group mentioned above or 'others'

Figure 11: LLM Prompt to assign user concern to service group using LLM based few-shot learning

Prompt: Create a Cluster Name and provide a Cluster Description for the concerns in this cluster. The name should capture the central theme or issue that ties all the concerns together, and the description should summarize what the concerns in this cluster are about, providing a clear and concise explanation of the shared topic or problem.

User Concerns in Cluster 1:

[Concern 1 text]

[Concern 2 text]

[Concern 3 text]

[Concern 4 text]

Output:

Cluster Name: [Generated cluster name]

Cluster Description: [Generated cluster description summarizing the shared issue of the concerns]

Figure 12: Prompt to Create Cluster Name and Cluster Description

PROPMT: Given the following new user concern and a list of the top 5 most similar cluster descriptions (pre-selected using embedding-based similarity), determine the best-matching cluster via semantic reasoning. If the concern aligns well with one or more clusters, provide the best match and justification. If not, indicate that it should form a new cluster. If a new concern is assigned to an existing cluster, propose an updated cluster description reflecting the new concern

New User Concern:
[Insert New User Concern Here]

Candidate Clusters (Top 5 from Embedding Similarity):

1. **Cluster Title**: [Title of Cluster 1]
   Description: [Description of Cluster 1]
2. **Cluster Title**: [Title of Cluster 2]
   Description: [Description of Cluster 2]
3. **Cluster Title**: [Title of Cluster 3]
   Description: [Description of Cluster 3]
4. **Cluster Title**: [Title of Cluster 4]
   Description: [Description of Cluster 4]
5. **Cluster Title**: [Title of Cluster 5]
   Description: [Description of Cluster 5]

OUTPUT:

Match Found: The new concern fits into the cluster titled "[Cluster Title]" based on the description: "[Cluster Description]".

Justification: [Explain why this concern semantically fits this cluster].

Updated Cluster Description:

[Cluster Title]

[Updated Cluster Description reflecting the new concern]

No Match Found: The new concern does not fit into any of the existing clusters and should form a new cluster.

Suggested New Cluster Title: [Optional – suggest a suitable name]

Suggested Description: [Brief description based on the concern]

Figure 13: LLM Prompt to assign incremental user concerns to Existing clusters

PROMPT: Following is the list of user concerns that belong to a single cluster. Split them into separate clusters, each containing semantically similar and coherent user concerns. Ensure that the resulting clusters are distinct and relevant. For each of the new clusters created, provide a title and a concise description that accurately represents the concerns within.

Input:

Cluster of User Concerns:

[User Concern 1]

[User Concern 2]

[User Concern 3]

[User Concern 4]

[User Concern n]

Output:

New split Cluster 1 (Title): [Cluster Title]

Description: ['new cluster description']

New split Cluster 2 (Title): [Cluster Title]

Description: ['new cluster description']

Figure 14: LLM Prompt for Splitting Cluster

PROMPT: Following is the list of user concerns that are unclustered. If 10 or more user concerns are like one another,
Firstly, group them together.
Next, create a cluster name and cluster description.

Output should contain new cluster title, Cluster description and list of all user concerns that constitute the cluster

Input:

List of unclustered User Concerns:

[User Concern 1]

[User Concern 2]

[User Concern 3]

[User Concern 4]

[User Concern n]

Output:

Cluster (Title): [Cluster Title]

Description: [Cluster Description]

[List of User Concerns]

Figure 15: LLM Prompt to Merge Unclustered User Concerns

PROPMT: A cluster of concerns was recently split into multiple subclusters due to topic drift. Generate a concise summary (2–4 sentences) explaining why the original cluster was split, highlighting the differences between the resulting subclusters. Focus on changes in topic focus, themes, or intent. Based on the original cluster and the subclusters, explain the semantic divergence that led to the split.

INPUT:

- Original Cluster Title: [insert original title]
- Original Cluster Summary: [insert short summary or description]
- Subcluster A:
  - Title: [title]
  - Sample Concerns: [list 2–3 sample concerns]
- Subcluster B:
  - Title: [title]
  - Sample Concerns: [list 2–3 sample concerns]

(…repeat for more subclusters if needed)

Figure 16: LLM Prompt for Drift Narrative Generation

PROMPT: Following clusters of user concerns have high cosine similarity score. Determine whether they are semantically similar enough to be merged into a single cluster. Focus on topical overlap, intent similarity, and whether users are essentially describing the same problem in different words. ? Reply with one of the following:
MERGE — if the clusters describe the same or highly similar issue
DO NOT MERGE — if they describe different topics or distinct intent

INPUT:

Cluster A Title: [insert title]
Cluster A Description: [insert description or summary]
Cluster A Sample Concerns:
[example 1]
[example 2]

Cluster B Title: [insert title]
Cluster B Description: [insert description or summary]
Cluster B Sample Concerns:
[example 1]
[example 2]

OUTPUT:
Cluster A - Cluster B – Merge/Do not Merge

Figure 17: LLM Prompt for Cluster Merge Decision