International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

ML-Based Deduplication and Semantic Collapsing in High-Volume Communication Streams

Ankita Kamat

Abstract: High-volume communication ecosystems generate large quantities of overlapping and redundant notifications that dilute user attention and inflate infrastructure costs. This article presents a machine learning-driven architecture for identifying semantic similarity across event streams and collapsing redundant notifications into concise, actionable updates. The proposed system combines deterministic deduplication, dense embedding inference, approximate nearest neighbor retrieval, and online clustering into a single streaming pipeline that is operating within strict latency budgets and personalization requirements. Evaluation criteria encompass offline clustering accuracy, online user engagement results, and cost savings in infrastructure. By transforming raw event floods into coherent, user-meaningful communication, the approach reduces notification overload, preserves semantic fidelity, and improves platform operational efficiency.

Keywords: *Deduplication, Semantic Similarity, Notification Overload, Event Streams, Sentence Embeddings, Approximate Nearest Neighbors, Online Clustering, Personalization, Stream Processing, Knowledge Distillation.*

1. Introduction

High-volume communication surfaces—push notifications, email digests, in-app inboxes, operational alert feeds, and social activity streams—routinely exhibit heavy redundancy. Repeated reminders, near-identical status transitions, follow-up messages that differ only by entity identifiers, and multi-source echoes of the same underlying occurrence collectively produce an environment in which users receive substantially more information than is semantically necessary. The user-facing consequence is perceived information overload: a condition in which the volume and rate of incoming signals exceed an individual's cognitive processing capacity, leading to disengagement and diminished interaction quality [1]. Empirical research on notification interruptions reinforces the cognitive cost of this phenomenon. Studies indicate that social media and application notifications can disrupt ongoing cognitive processing, with interference magnitude modulated by message relevance, notification rate, and individual usage patterns [2]. When notification volume exceeds manageable thresholds, users frequently resort to wholesale dismissal strategies or disable notification channels entirely, resulting in measurable declines in platform engagement metrics. Beyond user experience, redundant event delivery represents a measurable infrastructure tax: it amplifies ingestion volume, fan-out processing, ranking computations, and downstream delivery costs. In operational monitoring contexts, the analogous problem, commonly termed "alert fatigue," has driven substantial investment in noise reduction, alert prioritization, and lifecycle management [3]. These pressures motivate a principled semantic collapsing layer positioned between raw event production and user-facing delivery. Rather than emitting every raw event as an independent notification, this layer groups and summarizes semantically equivalent or similar events into coherent updates that preserve informational substance while reducing interruption frequency. Evidence from controlled field experiments indicate that batching notifications at structured intervals, rather than delivering them continuously, can improve self-reported attentional well-being, although completely removing notification flow can introduce anxiety effects associated with fear of missing out [4]. More recent work on notification management features, including sorting, pinning, and categorization, confirms that users increasingly expect automatic,

Goa University, India

context-aware organization of their notification streams rather than purely manual triage [5]. Emerging research on large-language-model-mediated notification summarization further establishes that user expectations center on prioritization, controlled disclosure levels, and the preservation of actionability [6]. This article synthesizes research and engineering practices into a cohesive design for ML-based deduplication and semantic collapsing under streaming constraints. The core contribution is a multi-stage pipeline that unifies (i) deterministic deduplication for exact and idempotency-aligned repeats; (ii) embedding-based similarity scoring for near-duplicates and semantic equivalents; (iii) approximate nearest neighbor retrieval over active cluster representatives; and (iv) time-aware online clustering with personalization and policy enforcement. The article introduces the problem formulation and design requirements, describes the proposed architecture and its algorithmic components, develops a structured evaluation methodology, and discusses deployment considerations, including safety, transparency, and concept drift.

2. Background and Related Work

Semantic collapsing at stream scale draws from four mature research lineages: near-duplicate detection via hashing, dense embedding representations for semantic textual similarity, approximate nearest neighbor search, and stream clustering with temporal awareness.

2.1 Near-Duplicate Detection via Hashing

Classical near-duplicate detection reduces textual documents to set representations, typically character or word shingles, and estimates pairwise resemblance using min-wise independent hashing (MinHash). This approach formalizes resemblance and containment measures and enables efficient similarity estimation over large document collections [7]. Locality-sensitive hashing (LSH) generalizes this idea into a framework for approximate near-neighbor queries that supports sublinear retrieval under appropriate similarity measures [8]. For cosine similarity, the dominant metric in embedding-based systems, random hyperplane hashing (SimHash) provides a related LSH scheme that maps high-dimensional vectors to compact binary codes while preserving angular proximity [9]. These hashing techniques form the computational foundation of the cheap approximate similarity gate in the proposed pipeline.

2.2 Dense Embeddings for Semantic Textual Similarity

Transformer-based language models, exemplified by BERT, provide strong contextual representations and underpin most contemporary semantic similarity systems [10]. However, standard cross-encoder architectures are computationally expensive for large-scale similarity search because they require encoding sentence pairs jointly, precluding the precomputation of individual sentence representations. Sentence-BERT (SBERT) addressed this limitation by introducing siamese and triplet network architectures that produce fixed-dimensional sentence embeddings comparable via cosine similarity [11]. This embed-once, compare-many property is essential for streaming deployments where per-event inference must remain computationally bounded.

Contrastive learning approaches have been shown to improve the quality of sentence embeddings even further. SimCSE was shown to improve the performance of sentence embeddings by utilizing standard dropout as a form of minimal augmentation, with the supervised variants of the method that utilize natural language inference data achieving state-of-the-art results [12]. The E5 embedding family was shown to improve the quality of sentence embeddings by utilizing weakly supervised contrastive pre-training across various text pairs, with the embeddings showing satisfactory performance across various tasks [13].

Evaluation benchmarks are an essential aspect of selecting the best model. The SemEval Semantic Textual Similarity shared tasks, as well as the STS Benchmark, provide standardized evaluations of the quality of pairwise similarity [14]. The Massive Text Embedding Benchmark (MTEB) broadens the evaluation scope across clustering, retrieval, reranking, and classification tasks, establishing that no single embedding model dominates across all settings, a practical consideration when selecting encoders for short, templated notification text versus longer free-form messages [15].

2.3 Model Compression for Latency-Constrained Inference

Production streaming environments impose strict per-event latency budgets that full-scale transformer models may not satisfy. Knowledge distillation, as

exemplified by DistilBERT, demonstrated that a student model retaining 97% of BERT's language understanding benchmark performance could be trained at a 40% smaller parameter count, with inference approximately 60% faster than the original [16]. MiniLM extended the distillation method with the application of deep self-attention transfer, allowing task-agnostic pre-trained transformer compression with excellent performance on downstream similarity tasks [17]. The above compression methods are applicable to the embedding inference in the presented framework.

2.4 Approximate Nearest Neighbor Search

Embedding-based semantic collapsing requires rapid nearest-neighbor retrieval at each incoming event to identify candidate clusters for merging. Hierarchical Navigable Small World graphs (HNSW) achieve this through a layered graph structure where each layer provides progressively coarser navigation, yielding high recall with low query latency and demonstrating robust empirical performance across high-dimensional vector spaces [18]. FAISS provides GPU-optimized similarity search implementations, with demonstrated scalability to billion-scale vector collections through product quantization and inverted index structures [19]. These retrieval systems are the practical realization of the ANN layer in the architecture described in Section 4.

2.5 Stream Clustering and Temporal Semantics

Online semantic collapsing is structurally equivalent to stream clustering with time-aware policies. CluStream was the first to separate online micro-cluster summarization from periodic offline macro-clustering. This made it possible to work with evolving data streams with limited memory [20]. DenStream extended density-based clustering concepts to streaming settings by introducing weighted micro-clusters with exponential time-decay mechanisms, addressing both noise tolerance and concept drift [21]. For event-like text streams, research on news stream clustering has demonstrated that dense embeddings alone may not yield optimal clustering and that combining sparse lexical signals with temporal proximity and domain fine-tuning produces measurable improvements [22].

Stream processing infrastructure must handle unbounded, out-of-order event sequences with principled time semantics. The Dataflow model provides a theoretical foundation for balancing correctness, latency, and cost in large-scale stream processing through explicit separation of event time, processing time, windowing, and triggering [23]. Apache Flink operationalizes these concepts in a stateful distributed stream processor with exactly-once state consistency, event-time processing, and recoverable operator state [24], capabilities directly required by the online clustering state model described in Section 4.

3. Problem Formulation and Design Requirements

3.1 Formal Problem Statement

The system operates over a high-rate event stream in which each raw event e_t arrives at time t carrying structured attributes (event type, actor, object or target, channel, source system, and severity), a textual payload that may be templated or free-form, and context attributes including user segment, locale, and device. The objective is to transform this raw event stream into a collapsed update stream satisfying six simultaneous properties.

Redundancy reduction requires that semantically equivalent or highly similar events be grouped into clusters and delivered as a reduced set of updates, each summarizing the cluster with counts and key facets. Semantic fidelity requires that clusters preserve the informational meaning of their constituent events, since false merges that conflate semantically distinct events are accuracy violations, not acceptable compression. Personalization requires that collapsing decisions respect user preferences and relevance signals, given that events constituting noise for one user may constitute critical signals for another [6]. Latency constraints necessitate that per-event processing be finalized within stringent budgets to prevent backpressure and intolerable delivery delays, thereby requiring lightweight inference, embedding caching, and sublinear retrieval [11]. Safety constraints preclude the folding of key event types, security alerts, financial transactions, health notifications, etc., into benign aggregations that hide useful information [3].

However, operational correctness demands that state changes and delivery decisions should ideally remain idempotent with respect to retries and replays, as seen in distributed event delivery systems [23].

3.2 Deduplication versus Semantic Collapsing

A useful distinction separates two related but distinct tasks. Exact deduplication targets hard duplicates: identical or canonically identical events produced by retries, at-least-once delivery guarantees, or upstream fan-out errors. This problem is tractable through deterministic keying and exactly-once stream processing semantics. Semantic collapsing targets soft duplicates: multiple distinct events referring to the same underlying situation but differing in phrasing, source, entity identifiers, or minor contextual details. This problem requires semantic similarity reasoning across text and structured metadata and is the primary focus of the ML components in the architecture. Table 1 summarizes the distinguishing characteristics of both tasks across several engineering dimensions.

Dimension	Exact Deduplication	Semantic Collapsing
Duplicate type	Hard duplicates (retries, at-least-once delivery)	Soft duplicates (semantically equivalent distinct events)
Detection method	Deterministic composite key hashing	Dense embedding similarity + structured metadata scoring
Latency profile	O(1) hash lookup; constant cost	O(log N) ANN retrieval + embedding inference
Handling mechanism	Key-based suppression; idempotency enforcement	Online clustering with time-decay and trigger policies
State required	Seen-key set with TTL expiry	Cluster centroid index and metadata store
Safety guarantee	Exact; no false suppression	Threshold-dependent; safety guardrails for critical categories
Primary standard	Exactly-once stream semantics	Bi-encoder embeddings (SBERT, SimCSE, E5)

Table 1: Comparison of Exact Deduplication and Semantic Collapsing [3, 11, 12, 13, 18, 20, 21, 23]

3.3 Merge Decision Model

Let c denote an active cluster with representative embedding v_c and metadata summary m_c. For a new event e with embedding v_e, the composite similarity score is defined as s(e, c) = alpha * cos(v_e, v_c) + beta * s_meta(e, m_c) + gamma * s_time(e, m_c), where cos(.,.) denotes cosine similarity [11], s_meta captures structured attribute agreement (same actor, object type, event category), and s_time encodes temporal proximity with exponential decay to discourage merging temporally distant events [20]. A merge decision applies a threshold tau: if the maximum over c of s(e,c) is at least tau, the event merges into the highest-scoring cluster; otherwise, a new singleton cluster is initialized. Personalization enters through user-specific or segment-specific calibration of tau and the weighting coefficients (alpha, beta, and gamma).

4. Proposed Architecture and Algorithms

4.1 Multi-Stage Pipeline Design

The pipeline's central scaling principle is progressive candidate reduction: each stage narrows the set of events requiring expensive downstream computation. Figure 1 describes the seven-stage pipeline of event ingestion to personalized delivery as a sequential process.

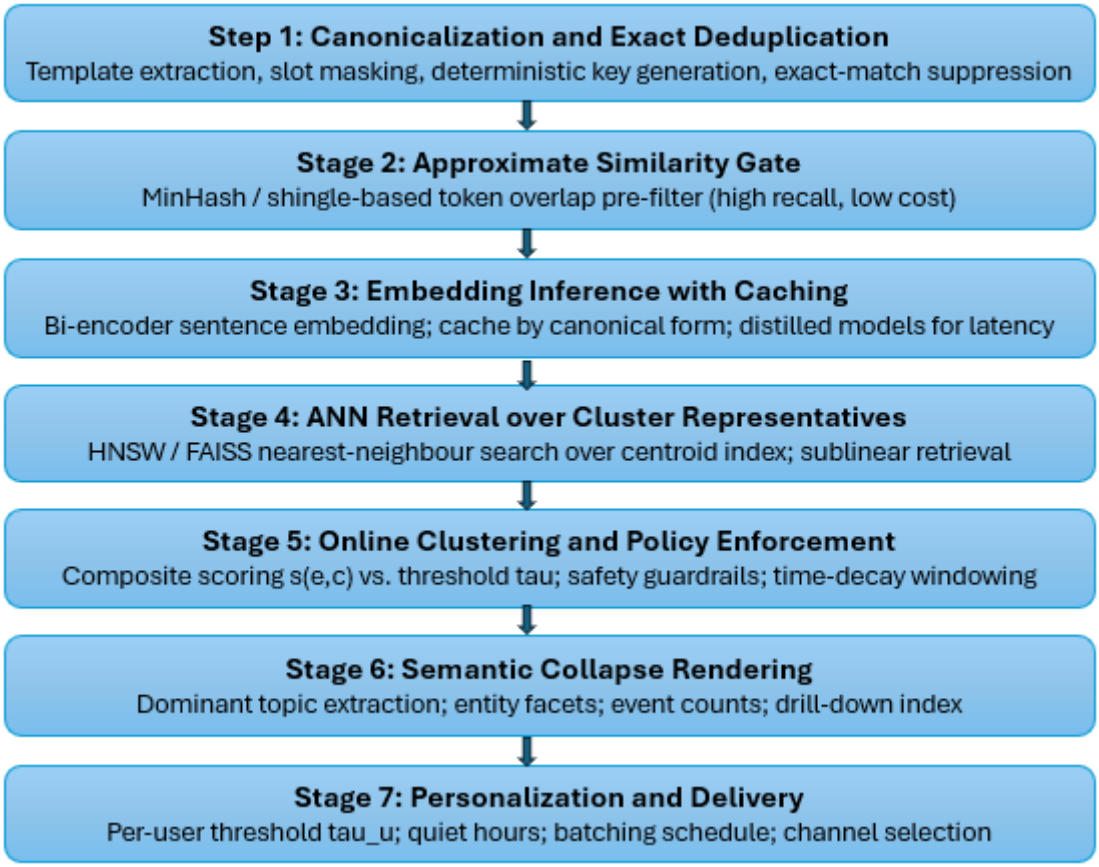

Figure 1: Seven-Stage Semantic Collapsing Pipeline [7, 11, 18, 21, 23]

Stage 1: A deterministic composite key is constructed using tenants, event types, canonical text forms, and a coarse time bucket. Events with an observed key within a suppression window are identified as exact duplicates, suppressed, or coalesced with an increment to the cluster event counter. This stage aligns with exact-once processing goals in stateful stream systems [23].

Stage 2 applies a cheap approximate similarity gate. A token-overlap or shingle-based similarity assessment serves as a high-recall, low-cost pre-filter before embedding inference. MinHash-based techniques provide probabilistic similarity bounds at sub-linear cost [7]. Events passing this gate proceed to embedding; remaining events are routed to exact-dedup or dropped from semantic collapsing consideration.

Stage 3 performs embedding inference with caching. The representation of the text in the form of a canonical text representation of the event is encoded into a fixed-dimensional vector space with the help of a bi-encoder sentence embedding model. The pre-computation of the embedding space is aided by bi-encoders of the SBERT type, allowing for constant-cost per-event comparisons [11]. SimCSE or E5 derivatives offer improved embedding quality for short or elliptical event descriptions [12, 13]. Under strict latency budgets, distilled encoders reduce inference time substantially while retaining a high fraction of base model quality [16, 17].

Stage 4 performs ANN retrieval over cluster representatives. The event embedding is used to query an ANN index maintained over cluster representative embeddings (centroids or medoids), not over every historical event. Indexing representatives rather than raw events reduces index size proportionally to the average cluster size, enabling efficient updates as clusters form and expire. HNSW provides robust ANN retrieval with strong empirical recall-latency tradeoffs [18]; FAISS offers GPU-optimized alternatives for very high query volumes [19].

Stage 5 applies online clustering and policy enforcement. The top K candidate clusters from ANN retrieval are scored using the composite

similarity function. Merge or new-cluster decisions are subject to policy constraints: safety guardrails prevent collapsing across critical event categories; maximum cluster age prevents temporally diffuse clusters from accumulating stale events; per-user do-not-collapse directives override all algorithmic decisions for designated categories. Stream clustering research motivates explicit time-decay mechanisms and windowing to manage cluster evolution and prevent semantic drift within active clusters [21].

Stage 6 produces semantic collapse rendering. An active cluster is converted into a user-facing update capturing the dominant intent or topic derived from the cluster centroid's nearest vocabulary; key entities and facets (service, environment, actor, and resource identifier); counts and recency metrics; and an optional drill-down index for accessing constituent raw events. Human-centered notification summarization research establishes that users expect summaries to function as intelligent assistants prioritizing event types, modulating disclosure based on anticipation and criticality, and preserving the ability to access full detail [6].

Stage 7 applies personalization and delivery. The rendered update is re-ranked and filtered through user-level preferences. A per-user threshold is applied to balance aggregation granularity against individual tolerance. Delivery policy, including quiet hours, batching schedules, and channel selection, is applied at this stage. Evidence from notification batching experiments supports that structured delivery schedules aligned to coarse time intervals improve attentional outcomes compared to continuous-as-received delivery [4].

4.2. Online Clustering State Model

Each cluster maintains a minimal state record whose fields are defined in Table II. State operations must satisfy bounded memory through cluster expiry and compression, fast updates admitting O(log N) ANN retrieval and O(1) centroid update, and correctness under fault tolerance through checkpointed state in the stream processor.

State Field	Type	Purpose
Centroid embedding	Fixed-dim float vector	Representative of cluster semantics; updated via running mean or reservoir sampling
Metadata summary	Structured record	Dominant event type, actor, object, source category for composite scoring
Event count	Integer	Tracks cluster size for count-threshold triggers and rendering
First event timestamp	Datetime	Anchors cluster temporal window; prevents unbounded age
Last event timestamp	Datetime	Recency signal for time-decay scoring and inactivity expiry
Last emit timestamp	Datetime	Controls minimum inter-emit interval; prevents burst over-delivery
Policy category tag	Enum / label	Identifies critical categories subject to no-collapse or conservative-merge rules

Table 2: Online cluster state record fields [3, 11, 20, 21, 23]

Apache Flink's stateful operator model supports exactly-once state snapshots and consistent recovery, which are essential when cluster states span multiple processing time windows [24]. Emission triggers can be defined by count thresholds for burst control, time-since-last-emission thresholds for maximum delay guarantees, importance-based immediate triggers for high-severity events, and user-context triggers such as quiet hours. These trigger semantics align with the Dataflow model's treatment of when-to-emit decisions as first-class system properties [23].

4.3 Embedding Selection and Domain Adaptation

Notification and alert text presents specific challenges for generic embedding models. Messages often have limited text and template-based content with slot values, numeric IDs, entity names, and status codes that contain structural, not semantic, information. Generic embeddings may exhibit high cosine similarity between messages that share slot patterns but differ meaningfully in the underlying event type, resulting in false merges.

Effective adaptation strategies include template-slot separation (embedding template text independently of slot values and combining with structured features), hard-negative mining to train the model to distinguish similar-phrasing but semantically distinct examples [12], and benchmark-guided model selection using STS-like evaluation sets constructed from domain-specific notification data [14, 15]. The MTEB benchmark's coverage of short-text clustering and retrieval tasks makes it a useful pre-selection filter for candidate embedding families before domain-specific evaluation [15].

4.4 Hybrid Candidate Generation

Dense embeddings excel at modeling semantic similarity, yet may fail to capture lexical and temporal features, especially when these features are highly informative for structured event streams. Research on news stream clustering demonstrates that embeddings alone may underperform when compared to hybrid approaches that combine sparse lexical representations, temporal proximity signals, and domain fine-tuning [22]. This finding motivates a hybrid candidate generation strategy in which lexical gating provides high-recall pre-filtering at low cost [7], dense embedding similarity drives merge decisions for semantically equivalent events [11], and temporal proximity modulates the composite score to prevent merging events separated by long time intervals [20].

5. Evaluation Methodology

Because semantic collapsing is simultaneously an ML accuracy problem and a product outcome problem, evaluation must cover three complementary dimensions: offline clustering correctness, online user engagement, and infrastructure efficiency.

5.1 Offline Accuracy: Clustering Metrics and Threshold Calibration

Ground truth for offline evaluation can be constructed through human annotation of event pairs and clusters (high quality but resource-intensive), weak supervision derived from incident or ticket association data (scalable but noisy), or post-hoc analysis of user interaction signals on deployed systems (requires careful causal design). The evaluation metrics in Table III provide complementary views of clustering quality by treating each raw event's cluster assignment as a partition over the event set.

Metric	Full Name	Evaluation Layer	Key Property
Pairwise F1	Pairwise Precision/Recall/F1	Offline – merge decisions	Direct evaluation of individual merge decisions
B3 F1	BCubed Precision and Recall	Offline – cluster quality	Item-level assessment; robust to skewed cluster size distributions
ARI	Adjusted Rand Index	Offline – partition similarity	Chance-corrected partition similarity measure
AMI	Adjusted Mutual Information	Offline – partition similarity	Information-theoretic chance correction; no equal-size assumption
Interruption load	Notifications per day / burstiness	Online – user attention	Measures reduction in daily interruption frequency
Open rate	Collapsed update open rate	Online engagement	Captures user interaction with summarized updates

Dismiss/mute rate	Channel mute and unsubscribe rate	Online – negative signals	Indicates over-suppression or trust breakdown
Reduction ratio r	1 minus N-hat divided by N	Infrastructure	Fraction of raw events collapsed; proxy for pipeline savings
p99 latency	99th-percentile per-event latency	Infrastructure	Critical bound for backpressure avoidance in streaming systems

Table 3: EValuation Metrics Across Offline, Online, and Infrastructure Dimensions [1, 2, 6, 18, 23, 26, 27]

Pairwise precision, recall, and F1 evaluate individual merge decisions directly. BCubed (B3) precision and recall assess cluster quality at the item level without requiring cluster-to-cluster matching, making them robust to the highly skewed cluster size distributions typical of event streams [25]. The Adjusted Rand Index (ARI) provides a chance-corrected measure of partition similarity [26]. Adjusted Mutual Information (AMI) offers an information-theoretic chance-corrected clustering comparison that does not assume equal cluster sizes [27]. A comparative analysis of clustering evaluation metrics establishes that these measures satisfy different formal constraints and that metric selection should be deliberate given the specific structural properties of the evaluation data [28].

Threshold calibration proceeds by sweeping tau over a labeled development set per event category and computing the B3 F1 and pairwise F1 tradeoff curve. A criticality guardrail is used. The minimum recall for severe event categories is enforced as hard constraints. This prevents the threshold from being optimized purely for compression at the cost of suppressing critical events [3]. Calibrated thresholds are then deployed with online monitoring and adjusted through A/B experimentation.

5.2 Online User Outcome Metrics

User-facing success must not be conflated with maximum event suppression. Overaggressive collapsing can reduce perceived trust, increase missed-critical-event rates, and diminish engagement for reasons unrelated to overload reduction. Recommended online measurement dimensions include interruption load (notification count per day, burstiness, time-of-day distribution), engagement quality (open rate, click-through rate, dwell time on collapsed updates relative to raw feeds), negative signals (dismissals, channel mute actions, unsubscribes, complaint rates), and downstream task completion (follow-through on actions prompted by collapsed updates).

Notification summarization user studies indicate that engagement effects depend heavily on controllability and trust cues: users who understand that a collapsed update represents multiple underlying events and who have access to drill-down detail report greater satisfaction than users presented with summaries without transparency about the collapsing process [6].

5.3 Infrastructure Efficiency

Let N denote the number of raw input events in a measurement period, and N-hat denotes the number of delivered collapsed updates. The reduction ratio r = 1 minus (N-hat divided by N) quantifies the compression achieved by the collapsing layer. In delivery pipelines where downstream cost scales approximately linearly with delivered unit count—a common assumption in fan-out and push delivery systems—infrastructure savings scale proportionally with r.

Savings arise across delivery fan-out operations, ranking and scoring computations, external push and email delivery costs, and event log storage. However, the collapsing layer introduces new costs in embedding inference, ANN index maintenance, and cluster state management. A comprehensive efficiency report must therefore also include incremental compute cost per event, memory footprint of the active cluster index, and per-event processing latency at the 50th, 95th, and 99th percentiles. Research on ANN retrieval systems demonstrates that these costs can be brought within acceptable bounds through engineering choices in indexing strategy and quantization [18, 19], and model distillation work establishes that inference overhead scales with deliberate compression choices [16].

6. Deployment Considerations

6.1 Latency Budget Management

Semantic collapsing introduces a synchronous processing path for each incoming event. A production-grade design manages this latency through embedding caching keyed on canonical text forms to avoid redundant inference on repeated templates [11], bi-encoder usage rather than cross-encoder usage to maintain constant-cost per-event inference regardless of cluster count [11], distilled encoder deployment to reduce absolute inference cost [16], ANN indexing over cluster representatives to bound retrieval time and index update cost [18], and micro-batch inference at the embedding server while maintaining stream-time semantics for downstream processing [23]. State consistency is prioritized over raw throughput in this architecture because incorrect cluster state updates produce compounding errors that propagate through subsequent collapsing decisions. Flink's state management, checkpointing, and recovery mechanisms ensure that cluster state remains consistent and recoverable across failures.

6.2 Idempotency and Exactly-Once Semantics

In distributed event delivery systems, duplicate events arise from retries, network failures, and at-least-once delivery guarantees. Exactly-once stream processing semantics achieved through the combination of idempotent producers, transactional state updates, and consumer-side deduplication reduce the incidence of observable duplicates and simplify the reasoning required for downstream consumers [23]. In a semantic collapsing system, idempotency must be enforced at multiple layers: input event keying, cluster state update operations (making repeated applications of the same update produce the same cluster state), and collapsed update delivery records (preventing redelivery of already-sent summaries).

6.3 Safety, Transparency, and User Trust

The threat of semantic collapsing and loss of time-sensitive information necessitates an explicit safety architecture. Critical event categories should be assigned conservative merge thresholds or no-collapse policies enforced at Stage 5. Explainability signals communicating to users that a notification represents a collapsed summary of multiple underlying events increase transparency and support rapid review of the summary's scope. Drill-down interaction gives users access to the full list of constituent events, which meets their needs for revisitability and detail control as shown in notification management research [5]. Alert management practice in operational contexts similarly emphasizes that noise reduction architectures must identify and preserve high-severity signals, subordinating compression objectives to safety objectives [3].

6.4 Concept Drift and Continuous Improvement

Event semantics evolve continuously: application templates change with product releases, new event types emerge, and user preferences shift with notification volume and life context. Production systems require mechanisms for detecting semantic drift in cluster purity metrics, periodic threshold recalibration against updated labeled evaluation sets, retraining or fine-tuning embedding models with newly collected hard negatives, and A/B testing of collapsing regime changes. The MTEB benchmark supports systematic evaluation of embedding model updates [15], while stream clustering research provides principled frameworks for time-aware adaptation to evolving data distributions [21].

Conclusion

ML-based deduplication and semantic collapsing constitute a streaming retrieval-and-clustering problem with interlocking accuracy, personalization, latency, and safety constraints. The proposed multi-stage architecture addresses each constraint through a hierarchically organized pipeline: deterministic deduplication and idempotency handling resolve exact duplicates at minimal computational cost; embedding-based similarity scoring identifies semantic equivalents that escape lexical matching; ANN retrieval over cluster representatives provides sublinear search over the active cluster state; and time-aware online clustering with configurable triggers controls the emission of concise, user-meaningful collapsed updates.

The design is grounded in a substantial body of prior work spanning sentence embedding architectures, approximate nearest neighbor search, stream clustering, and human-centered notification management. Proper evaluation must span offline clustering correctness assessed through metrics including B3 F1, ARI, and AMI that capture different structural properties of the cluster partition;

online user outcomes reflecting engagement quality and trust; and infrastructure efficiency measured net of the embedding and indexing costs introduced by the collapsing layer. Critically, overly aggressive compression is neither technically desirable nor user-beneficial: semantic fidelity, transparent disclosure of collapsing decisions, and conservative handling of critical event categories are as important to system quality as ML accuracy on similarity benchmarks. Personalization of merge thresholds and rendering styles ensures that the boundary between signal and noise respects individual user preferences rather than imposing a single collapsing regime across heterogeneous populations.

References

[1] Donghwa Chung et al., "Perceived Information Overload and Intention to Discontinue Use of Short-Form Video: The Mediating Roles of Cognitive and Psychological Factors," Behavioral Sciences, 2023. Available: https://www.mdpi.com/2076-328X/13/1/50

[2] Hippolyte Fournier et al., "Attention hijacked: How social media notifications disrupt cognitive processing," Computers in Human Behavior, 2026. Available: https://www.sciencedirect.com/science/article/pii/S0747563226000233

[3] Qingyand Yu et al., "A Survey on Intelligent Management of Alerts and Incidents in IT Services," Journal of Network and Computer Applications, 2024. Available: https://netman.aiops.org/wp-content/uploads/2024/08/A-survey-on-intelligent-management-of-alerts-and-incidents-in-IT-services.pdf

[4] Nicholas Fitz et al., "Batching Smartphone Notifications Can Improve Well-Being," Computers in Human Behavior, 2019. Available: https://static1.squarespace.com/static/57a40c19414fb54f51f8095f/t/614a55faa7b89e25f4e48ad1/1632261627146/2019%2BFitz%2BBatching.pdf

[5] Yong-Han Lin et al., "Pinning, Sorting, and Categorizing Notifications: A Mixed-Methods Usage and Experience Study of Mobile Notification-Management Features," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2024. Available: https://people.cs.nycu.edu.tw/~armuro/pubs/lin-et-al-2024-imwut.pdf

[6] Uei-Dar Chen et al., "From Overwhelmed to Overview: Understanding Smartphone Users' Preferences and Expectations in Relieving Notification Overload via Text Summarization," Proceedings of the ACM on Human-Computer Interaction (MobileHCI), 2025. Available: https://people.cs.nycu.edu.tw/~armuro/pubs/chen-et-al-2025-mobilehci.pdf

[7] A. Broder, "On the Resemblance and Containment of Documents," Proceedings of the Compression and Complexity of Sequences, 1997. Available: https://www.computer.org/csdl/proceedings-article/sequences/1997/81320021/12OmNwDACjh

[8] Piotr Indyk and Rajeev Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998. Available: https://dl.acm.org/doi/epdf/10.1145/276698.276876

[9] Moses S. Charikar et al., "Similarity Estimation Techniques from Rounding Algorithms," Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002. Available: https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf

[10] Jacob Devlin et al., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, 2019. Available: https://aclanthology.org/N19-1423.pdf

[11] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019. Available: https://aclanthology.org/D19-1410.pdf

[12] Tianyu Gao et al., "SimCSE: Simple Contrastive Learning of Sentence Embeddings," Proceedings of Empirical Methods in Natural Language Processing, 2021. Available: https://aclanthology.org/2021.emnlp-main.552/

[13] Liang Wang et al., "Text Embeddings by Weakly-Supervised Contrastive Pre-Training," arXiv, 2024. Available: https://arxiv.org/pdf/2212.03533

[14] Daniel Cer et al., "SemEval-2017 Task 1: Semantic Textual Similarity—Multilingual and Cross-lingual Focused Evaluation," Proceedings of International Workshop on Semantic Evaluations, 2017. Available: https://aclanthology.org/S17-2001.pdf

[15] Niklas Muennighoff et al., "MTEB: Massive Text Embedding Benchmark," Proceedings of the European Chapter of the Association for Computational Linguistics, 2023. Available: https://aclanthology.org/2023.eacl-main.148/

[16] Victor Sanh et al., "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter," NeurIPS Workshop on Energy Efficient Deep Learning, 2019. Available: https://arxiv.org/abs/1910.01108

[17] Wenhui Wang et al., "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers," Advances in Neural Information Processing Systems, 2020. Available: https://proceedings.neurips.cc/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[18] Y. A. Malkov et al., "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018. Available: https://arxiv.org/abs/1603.09320

[19] Jeff Johnson et al., "Billion-Scale Similarity Search with GPUs," IEEE Transactions on Big Data, 2021. Available: https://www.computer.org/csdl/journal/bd/2021/03/08733051/1aFvgKKpjoc

[20] Charu C. Aggarwal et al., "A Framework for Clustering Evolving Data Streams," Proceedings of the 29th International Conference on Very Large Data Bases, 2003. Available: https://www.vldb.org/conf/2003/papers/S04P02.pdf

[21] Feng Cao et al., "Density-Based Clustering over an Evolving Data Stream with Noise," Proceedings of the 2006 SIAM International Conference on Data Mining, 2006. Available: https://epubs.siam.org/doi/10.1137/1.9781611972764.29

[22] Kailash Karthik Saravanakumar et al., "Event-Driven News Stream Clustering using Entity-Aware Contextual Embeddings," Proceedings of the European Chapter of the Association for Computational Linguistics, 2021. Available: https://aclanthology.org/2021.eacl-main.198.pdf

[23] Tyler Akidau et al., "The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing," Proceedings of the VLDB Endowment, 2015. Available: https://research.google.com/pubs/archive/43864.pdf

[24] Paris Carbone et al., "Apache Flink: Stream and Batch Processing in a Single Engine," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015. Available: https://asterios.katsifodimos.com/assets/publications/flink-deb.pdf

[25] Amit Bagga et al., "Entity-Based Cross-Document Coreferencing Using the Vector Space Model," Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics, 1998. Available: https://dl.acm.org/doi/10.3115/980845.980859

[26] Lawrence Hubert & Phipps Arabie, "Comparing Partitions," Journal of Classification, 1985. Available: https://link.springer.com/article/10.1007/BF01908075

[27] Nguyen Xuan Vinh et al., "Information Theoretic Measures for Clustering Comparison: Variants, Properties, Normalization and Correction for Chance," Journal of Machine Learning Research, 2010. Available: https://jmlr.csail.mit.edu/papers/volume11/vinh10a/vinh10a.pdf