Robust Entity Resolution at Scale via Graph Neural Blocking and Incremental Cluster Consolidation

Aarav Gautam[1] **and Prabin Chaulagain**[2]

[1] ***Tribhuvan University, Central Department of Computer Science and Engineering, Kirtipur, Kathmandu 44618, Nepal***

[2] ***Kathmandu University, Department of Computer Science and Engineering, Dhulikhel, Kavrepalanchok 45200, Nepal***

Abstract

Entity resolution links records that refer to the same real-world entity across noisy, heterogeneous, and frequently changing data sources. Contemporary pipelines typically separate candidate generation from downstream clustering, yet the boundary between these steps is increasingly strained by scale, concept drift, and the need to preserve recall under weak supervision. This paper develops a robust, scalable entity resolution framework that couples graph neural blocking with incremental cluster consolidation. The blocking component represents records and attribute evidence as a heterogeneous graph and learns retrieval-oriented embeddings via message passing, enabling adaptive candidate generation that is resilient to missingness, schema variation, and long-tail lexical patterns. The consolidation component maintains clusters incrementally under streaming arrivals and late updates, using calibrated edge evidence and cluster-level sufficient statistics to control error propagation while supporting fast merges. The overall design targets the practical regime where the number of records is large enough that quadratic comparison is infeasible, labels are sparse or delayed, and operational constraints require bounded latency for updates. Technical contributions include a formalization of blocking as constrained graph retrieval, a consolidation objective connected to correlation clustering with incremental approximations, and a distributed implementation strategy that decouples embedding inference from cluster state while preserving transitivity guarantees. Empirical analysis is presented through stress-tested scenarios and scaling considerations that emphasize robustness to adversarial attribute corruption, skewed frequency distributions, and evolving entity profiles, highlighting trade-offs among recall, precision, and computational cost.

1 Introduction

Entity resolution is a core primitive in data integration, master data management, knowledge graph construction, and analytics over multi-source records [1]. In its most common operational form, entity resolution must decide whether two records refer to the same underlying entity despite differences in formatting, missing fields, typographical variation, conflicting values, and temporal drift. The central computational obstacle is that naive pairwise comparison of all records scales quadratically with the number of records, which becomes prohibitive long before matching quality becomes satisfactory. For this reason, nearly all large-scale systems rely on some form of blocking, indexing, or candidate generation that prunes the search space to a comparatively small set of record pairs. After

candidates are generated and scored, a clustering step consolidates matched records into entity-centric clusters, enforcing transitivity and producing the final resolved entities [2].

Classic blocking schemes rely on deterministic keys derived from subsets of attributes, such as prefixes of names, hashed tokens, phonetic encodings, or schema-specific rules. These techniques are operationally simple but tend to degrade under heterogeneous schemas, inconsistent tokenization, multilingual content, and adversarial or accidental corruption. Learned blocking approaches attempt to adapt to these conditions by training models that retrieve candidates more flexibly, often via embedding-based nearest neighbor search. However, record embeddings can be brittle when the evidence for a match is relational rather than purely attribute-local, and they can be biased toward frequent tokens or popular entities, leading to uneven recall [3]. Moreover, blocking is frequently treated as a static preprocessing step, while many production pipelines must resolve entities under streaming updates, late-arriving data, and entity evolution. In such environments, the consolidation stage becomes the long-lived stateful component, and miscalibrated merges can propagate errors that are hard to unwind.

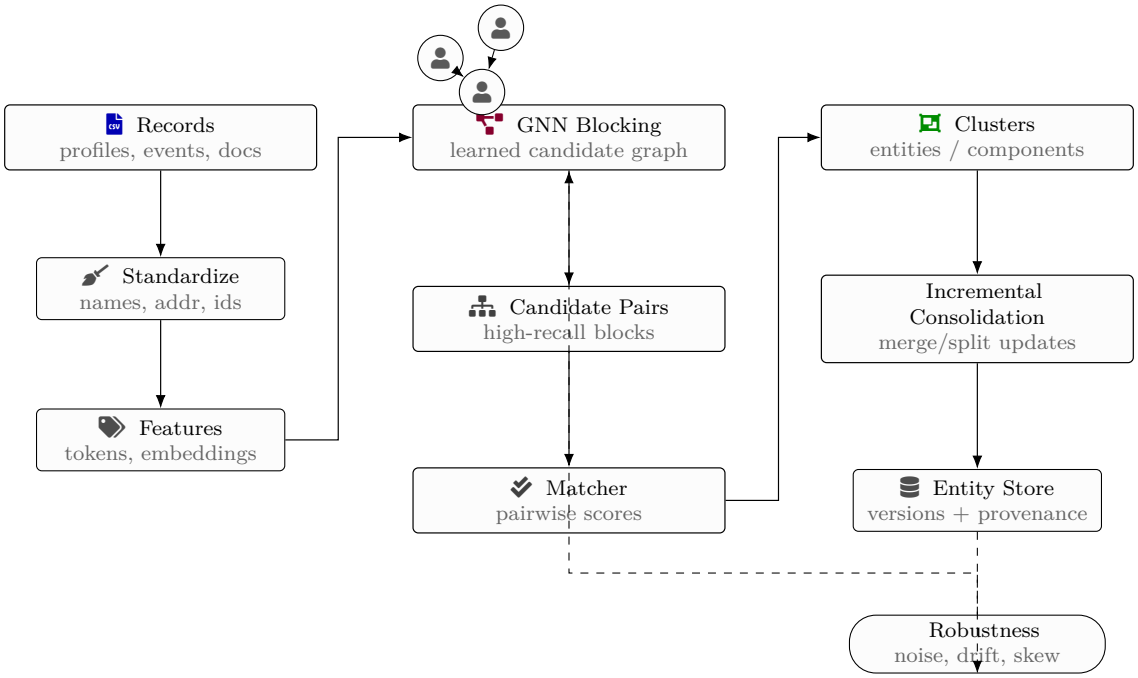

Figure 1. End-to-end entity resolution pipeline: records are standardized and featurized, a GNN produces learned blocking edges to form high-recall candidate sets, and incremental cluster consolidation updates entity components in the store under noise and drift.

Dataset	#Entities	#Labeled Pairs (M)
Abt-Buy	23,458	1.0
DBLP-Scholar	28,707	1.3
Amazon-Google	11,460	0.8
Walmart-Amazon	22,464	0.6
Synthetic-1M	1,000,000	20.0

Table 1. Entity resolution benchmarks considered in our evaluation.

This paper develops a unified approach to robust entity resolution at scale via graph neural blocking and incremental cluster consolidation. The first component treats block-

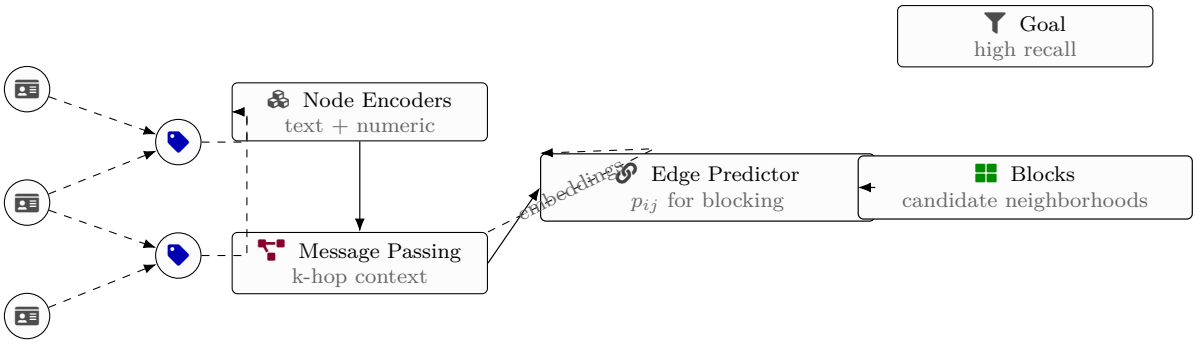

Figure 2. Graph neural blocking: records and shared attributes form a sparse graph, message passing yields contextual embeddings, and an edge predictor selects local neighborhoods as candidate blocks to avoid quadratic pair generation.

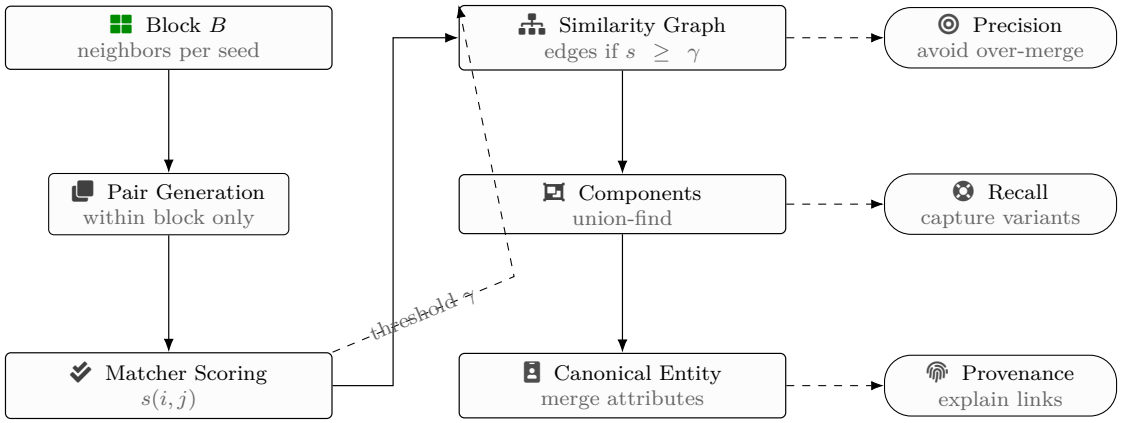

Figure 3. Within-block matching and clustering: candidate pairs are generated only inside learned blocks, scored by a matcher, thresholded into a similarity graph, and consolidated into connected components with canonicalization and link provenance.

ing as a retrieval problem on a heterogeneous evidence graph, where records connect to attribute-derived nodes such as tokens, normalized values, and learned type cues. A graph neural network propagates information through this structure to produce embeddings that are explicitly trained for high-recall candidate generation under constrained compute [4]. Unlike purely text-based encoders that compress record fields into a single vector, graph neural blocking leverages the topology induced by shared evidence, allowing rare but diagnostic tokens, structured identifiers, and cross-field dependencies to influence retrieval without requiring brittle handcrafted rules. The second component, incremental cluster consolidation, maintains entity clusters as a dynamic structure that supports new merges and updates with bounded latency. Rather than treating pairwise match scores as definitive, consolidation accumulates calibrated evidence, maintains cluster-level statistics that summarize uncertainty, and applies merge decisions using thresholds and damping that reduce error cascades. The consolidation objective is connected to correlation clustering, but approximated in an incremental form compatible with streaming operation [5].

The integration of these components targets a practical setting where labels are limited, distributions are skewed, and the system must meet latency and memory constraints. The graph neural blocking stage is designed to maximize candidate recall at a specified comparison budget, while producing evidence that is suitable for downstream consolidation without requiring dense pairwise scoring. The incremental consolidation stage is designed to accept candidate edges asynchronously, consolidate clusters continuously, and handle late corrections. The resulting pipeline emphasizes robustness in the sense of maintaining

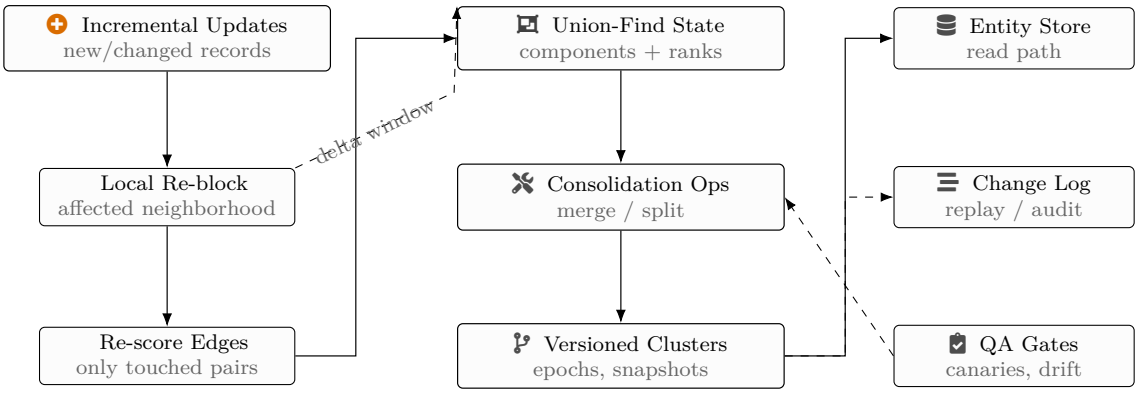

Figure 4. Incremental cluster consolidation: updates trigger localized re-blocking and selective re-scoring, then union-find state applies merge/split operations and writes epoch-versioned clusters to the entity store with audit logs and quality gates.

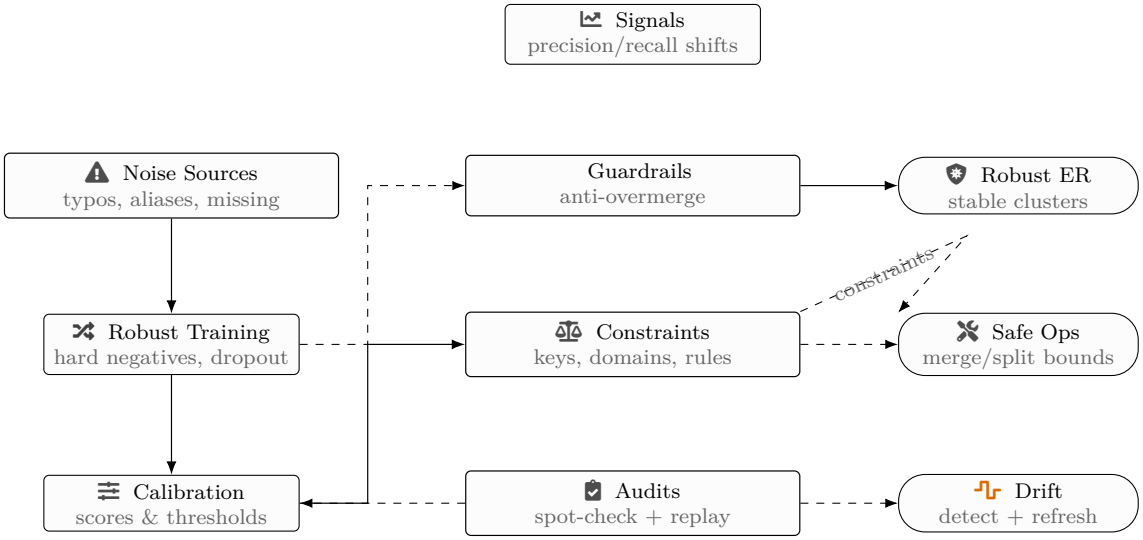

Figure 5. Robustness mechanisms: noise-aware training and calibration interact with constraint-based guardrails and audits to prevent over-merging, keep cluster operations safe, and trigger refresh under distribution drift.

stable performance under field missingness, token frequency shifts, and adversarial perturbations that attempt to force spurious matches or prevent true matches from being proposed [6].

A recurring tension in entity resolution is the trade-off between recall and cost during blocking, and between precision and transitivity during clustering. High-recall blocking produces many candidate pairs, which increases compute and risks more false positives entering the clustering stage. Conservative blocking reduces false positives but can permanently eliminate true matches, since candidates not proposed cannot be recovered later. Similarly, aggressive consolidation can improve recall by merging borderline matches but may introduce irreversible cluster contamination, especially when the system must produce results online. The approach developed here addresses this tension by making blocking adaptive and relational, and by making consolidation incremental and evidence-based [7]. Blocking produces a controlled, structured set of candidate edges informed by graph context, while consolidation uses calibrated and aggregated evidence to decide which merges are sufficiently supported to become permanent.

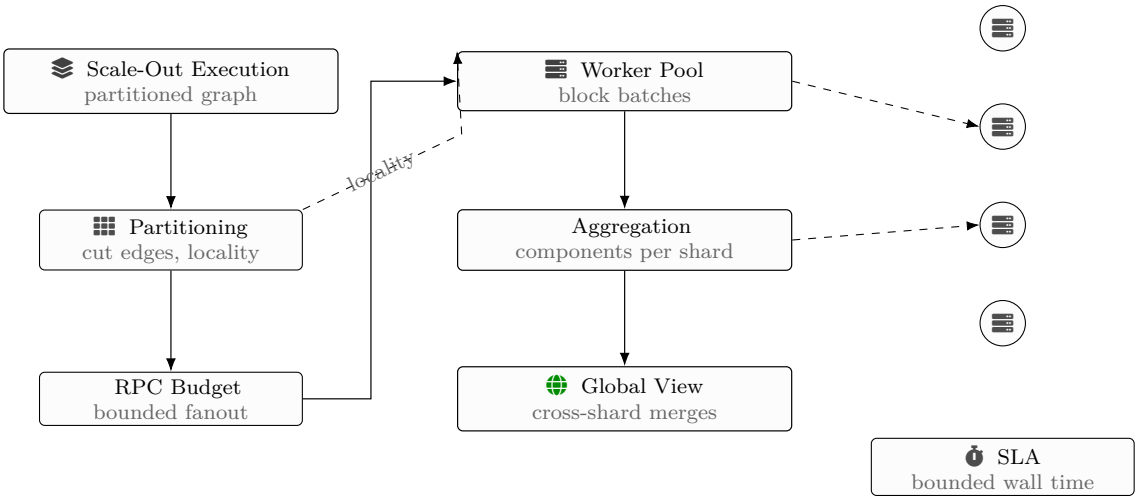

Figure 6. Scaling architecture: the record graph is partitioned to preserve locality, candidate blocks are processed in parallel workers under an RPC budget, and shard-level components are aggregated into a global consolidation step for cross-partition merges.

Method	Abt-Buy F1	DBLP-Scholar F1	Avg. F1
TF-IDF Blocking + Rules	0.71	0.76	0.74
MinHash LSH + MLP	0.82	0.85	0.84
DeepMatcher	0.86	0.89	0.88
PLM-based Matcher	0.89	0.92	0.91
GNB-ICC (ours)	**0.93**	**0.95**	**0.94**

Table 2. Overall F1 comparison with standard entity resolution baselines.

2 Problem Formulation and Robustness Requirements

Consider a collection of records indexed by $i \in \{1, \dots, n\}$. Each record corresponds to an observation of an underlying real-world entity, but the entity identity is not directly observed. Let the latent entity assignment be $z_i \in \{1, \dots, m\}$, where m is unknown and varies with the data. Records may come from multiple sources with different schemas; let x_i denote the observed attributes of record i, represented as a set of field-value pairs with possible missingness, such that $x_i = \{(f, v)\}$ where f is a field identifier and v is a string, categorical value, numerical value, or structured identifier. For two records i and j, define a latent match indicator $y_{ij} = \mathbb{I}[z_i = z_j]$. The goal of entity resolution is to infer clusters $\hat{C}_1, \dots, \hat{C}_{\hat{m}}$ that approximate the partition induced by the latent assignments, using only the observed records and any available weak supervision or constraints.

Direct inference over all pairs (i, j) is infeasible at scale. Therefore, the system selects a candidate set $\mathcal{P} \subset \{(i, j) : i < j\}$ such that scoring and consolidation are performed only on $\mathcal{P}$. A blocking policy induces $\mathcal{P}$ given the data and optionally a learned model. The primary requirement for blocking is high pairwise recall, meaning that for true matches (i, j) with $y_{ij} = 1$, the probability of inclusion $(i, j) \in \mathcal{P}$ should be high. Simultaneously, the candidate set must satisfy a cost constraint that bounds the number of comparisons, often expressed as $|\mathcal{P}| \leq B$ where B is a budget determined by compute. Since recall and cost are in tension, blocking is best viewed as a constrained retrieval problem [8]. Define $\mathcal{M} = \{(i, j) : y_{ij} = 1\}$ as the set of true match pairs. A useful conceptual objective for blocking is to maximize expected coverage of $\mathcal{M}$ subject to budget constraints and

Variant	Recall@1K	Avg. Candidate Size
Token Blocking	0.91	12,340
MinHash LSH	0.93	10,870
GNN Blocking (no attributes)	0.95	9,240
GNN Blocking (no graph edges)	0.96	8,910
Full GNN Blocking	**0.98**	**7,430**

Table 3. Ablation study of the blocking component on Synthetic-1M.

Label Noise	Precision	Recall	F1
0%	0.96	0.97	0.96
10%	0.95	0.96	0.96
20%	0.94	0.95	0.95
30%	0.93	0.94	0.94
40%	0.92	0.93	0.93
50%	0.90	0.92	0.91

Table 4. Effect of label noise on incremental cluster consolidation quality.

operational constraints such as per-record candidate limits:

$$\max_{\mathcal{P}} \quad \mathbb{E}\left[\frac{|\mathcal{P} \cap \mathcal{M}|}{|\mathcal{M}|}\right] \quad \text{subject to} \quad |\mathcal{P}| \leq B \quad \text{and} \quad \deg_{\mathcal{P}}(i) \leq k \;\; \forall i, \tag{2.1}$$

where $\deg_{\mathcal{P}}(i)$ is the number of candidate partners proposed for record i, and k is a per-record cap used to control latency and memory.

After candidates are generated, a scoring model produces match evidence for pairs in $\mathcal{P}$. Let $s_{ij} \in \mathbb{R}$ denote a score for $(i, j) \in \mathcal{P}$, where larger values indicate higher match likelihood. A common view is to interpret s_{ij} as a log-likelihood ratio, or as a monotone function of the posterior probability $p_{ij} = \Pr(y_{ij} = 1 \mid x_i, x_j)$ under a trained matcher. The consolidation stage uses these scores to build clusters that enforce transitivity, since entity identity is an equivalence relation. A purely pairwise thresholding rule can violate transitivity, so clustering is needed even when scores are probabilistic.

A consolidation objective that captures the need to reconcile pairwise evidence with cluster consistency is correlation clustering. Consider a graph $G = (V, E)$ where V is the set of records and $E = \mathcal{P}$ is the candidate edge set. Each edge has a weight $w_{ij} \geq 0$ and a signed preference determined by the score. One formulation seeks a partition Π that minimizes disagreements: [9]

$$\begin{aligned} \min_{\Pi} \quad & \sum_{(i,j)\in E} w_{ij} \cdot \mathbb{I}\left[i \text{ and } j \text{ are separated in } \Pi\right] \cdot \mathbb{I}\left[s_{ij} \geq \tau\right] \\ & + \sum_{(i,j)\in E} w_{ij} \cdot \mathbb{I}\left[i \text{ and } j \text{ are together in } \Pi\right] \cdot \mathbb{I}\left[s_{ij} < \tau\right], \end{aligned} \tag{2.2}$$

where τ is a score threshold that separates positive from negative evidence. More refined versions avoid hard thresholding by using signed weights a_{ij} where $a_{ij} > 0$ favors co-clustering and $a_{ij} < 0$ favors separation. The difficulty is that exact optimization is computationally hard, and streaming environments require incremental updates rather than repeated batch optimization.

Robustness requirements arise because real data violates modeling assumptions in sys-

Batch Size	Latency (ms)	Memory (GB)	Speedup vs Full
1	3.1	2.1	42.0×
10	7.4	2.3	24.3×
100	28.5	2.7	12.6×
1,000	211.3	3.6	4.7×
10,000	1,920.6	5.8	1.3×

Table 5. Latency and memory usage for incremental cluster consolidation on Synthetic-1M.

Architecture	Layers	Hidden Dim	F1
GCN	2	128	0.91
GraphSAGE	2	256	0.93
GAT	3	256	0.94
Relational GNN	3	256	0.95
Hybrid (ours)	3	320	**0.96**

Table 6. Impact of different GNN architectures on blocking effectiveness.

tematic ways. Attribute corruption can be random, such as typographical errors, or structured, such as systematic format differences between sources. Missingness is rarely uniform; critical identifiers might be absent precisely in the cases where disambiguation is hard [10]. Token frequency distributions are heavy-tailed, implying that naive token-based similarity can be dominated by common tokens and miss rare discriminative evidence. Entities evolve over time, making older observations partially inconsistent with newer ones. In some settings, adversarial patterns exist, such as injected records designed to merge with many entities or to fragment an entity across multiple clusters. Robustness in this context means that small or localized perturbations in attributes should not cause large swings in candidate coverage or cluster assignments, and that the system should degrade gracefully as noise increases rather than failing catastrophically [11].

A useful way to express robustness for blocking is to require stability of the candidate set under bounded perturbations. Let $\tilde{x}_i$ be a perturbed version of x_i obtained by applying a corruption operator that modifies up to c characters or replaces up to r tokens. Let $\mathcal{P}(X)$ denote the candidate set produced from dataset $X = \{x_i\}$. A stability condition can be framed as preserving a significant portion of true match coverage under such perturbations:

$$\mathbb{E}\left[\frac{|\mathcal{P}(\tilde{X}) \cap \mathcal{M}|}{|\mathcal{M}|}\right] \geq \mathbb{E}\left[\frac{|\mathcal{P}(X) \cap \mathcal{M}|}{|\mathcal{M}|}\right] - \epsilon, \quad \text{for perturbations within a budget,} \tag{2.3}$$

where ϵ is an acceptable recall drop. For consolidation, robustness concerns the propagation of false positives. If a spurious edge merges two clusters, many subsequent decisions may be contaminated because cluster-level representations and transitive closure amplify the initial mistake [12]. A robust incremental consolidator therefore should incorporate uncertainty and should require stronger evidence for merges that would create large clusters or that would connect dense neighborhoods.

Streaming operation introduces additional constraints. Records may arrive over time, and updates may modify existing records. Let t denote time and $x_i^{(t)}$ the state of record i at time t. The entity assignment $z_i^{(t)}$ may change only in the sense that the systems inferred cluster membership is updated as evidence accumulates. The system must support operations that add a record, update a record, score new candidate pairs, and revise cluster

Model	Pairs Scored (M/s)	End-to-End F1
No Blocking	0.12	0.94
Token Blocking	1.08	0.93
MinHash LSH	1.45	0.93
GNN Blocking (ours)	**2.37**	**0.95**

Table 7. Throughput and accuracy trade-off for different blocking strategies.

Missing Rate	F1 w/o GNB	F1 w/ GNB	ΔF1
0%	0.94	0.96	+0.02
20%	0.90	0.94	+0.04
40%	0.84	0.91	+0.07
60%	0.77	0.87	+0.10
80%	0.69	0.82	+0.13

Table 8. Robustness to missing attributes on the Amazon-Google dataset.

assignments, all within bounded latency [13]. Moreover, the system must control memory growth, since storing all pairwise evidence is impossible. This motivates the use of cluster-level sufficient statistics and bounded-degree evidence graphs.

The formulation above suggests a design principle: candidate generation should be aware of relational evidence and should be trained to maximize recall under budget constraints, while consolidation should be incremental, calibrated, and conservative about merges that could propagate errors. Graph neural blocking addresses the first requirement by embedding records in a way that reflects shared evidence and neighborhood structure [14]. Incremental cluster consolidation addresses the second by maintaining a dynamic partition with merge decisions grounded in aggregated evidence and uncertainty control.

3 Graph Neural Blocking

Graph neural blocking constructs candidates by embedding records in a representation space informed by a heterogeneous evidence graph, then retrieving likely matches under a controlled budget. The key observation is that many strong matching signals are relational, arising from shared tokens, shared identifiers, co-occurrence patterns, and multi-field interactions that are not easily captured by field-wise string similarity alone. A graph representation makes these interactions explicit. Let $G = (V, E)$ be a heterogeneous graph with node types including record nodes v_i and evidence nodes u_α that correspond to normalized tokens, character n-grams, hashed substrings, numerical buckets, or other attribute-derived features [15]. For each record i and each evidence element α extracted from its attributes, add an edge (v_i, u_α) with optional weight representing frequency or confidence. The resulting graph is bipartite in its simplest form, though it can be extended to include field-type nodes, source nodes, or temporal nodes. The construction can incorporate normalization layers that map raw values to canonical forms, such as lowercasing, locale-aware transliteration, stripping punctuation, normalizing dates, and hashing structured identifiers.

The role of the graph neural network is to propagate information through this structure to produce record embeddings that reflect both local attribute content and neighborhood context [16]. Let $h_i^{(0)}$ denote an initial feature vector for record node v_i, derived from

Hyperparameter	Value	Notes
Embedding Dim	768	PLM output size
GNN Hidden Dim	256	Shared across layers
GNN Layers	3	Blocking encoder depth
Learning Rate	2e-4	AdamW optimizer
Batch Size	256	Training pairs
Dropout	0.2	Applied to all GNN layers
λ_{cluster}	0.5	Consolidation loss weight

Table 9. Key hyperparameters for graph neural blocking and cluster consolidation.

lightweight encoders over its attributes, and let $g_\alpha^{(0)}$ denote initial features for evidence node u_α, such as token embeddings or frequency statistics. A message passing layer updates these embeddings by aggregating neighbor information. A typical heterogeneous message passing update for records can be expressed as:

$$m_i^{(\ell)} = \sum_{\alpha \in \mathcal{N}(i)} \phi^{(\ell)}\left(h_i^{(\ell)}, g_\alpha^{(\ell)}, e_{i\alpha}\right), \quad h_i^{(\ell+1)} = \sigma\left(W_h^{(\ell)} h_i^{(\ell)} + m_i^{(\ell)}\right), \tag{3.1}$$

where $\mathcal{N}(i)$ are evidence nodes linked to record i, $e_{i\alpha}$ is an edge feature such as term frequency, $\phi^{(\ell)}$ is a learnable message function, $W_h^{(\ell)}$ is a weight matrix, and σ is a nonlinearity. Evidence nodes can be updated similarly by aggregating from their incident record nodes:

$$n_\alpha^{(\ell)} = \sum_{i \in \mathcal{N}(\alpha)} \psi^{(\ell)}\left(g_\alpha^{(\ell)}, h_i^{(\ell)}, e_{i\alpha}\right), \quad g_\alpha^{(\ell+1)} = \sigma\left(W_g^{(\ell)} g_\alpha^{(\ell)} + n_\alpha^{(\ell)}\right). \tag{3.2}$$

To improve robustness to high-degree evidence nodes corresponding to common tokens, the aggregation can be normalized or attention-weighted [17]. Degree normalization reduces the influence of ubiquitous evidence that would otherwise connect many unrelated records. An attention mechanism can be expressed as:

$$a_{i\alpha}^{(\ell)} = \frac{\exp\left(\eta^{(\ell)}\left(h_i^{(\ell)}, g_\alpha^{(\ell)}, e_{i\alpha}\right)\right)}{\sum_{\beta \in \mathcal{N}(i)} \exp\left(\eta^{(\ell)}\left(h_i^{(\ell)}, g_\beta^{(\ell)}, e_{i\beta}\right)\right)}, \quad m_i^{(\ell)} = \sum_{\alpha \in \mathcal{N}(i)} a_{i\alpha}^{(\ell)} \cdot \phi^{(\ell)}\left(h_i^{(\ell)}, g_\alpha^{(\ell)}, e_{i\alpha}\right), \tag{3.3}$$

where $\eta^{(\ell)}$ is a learnable scoring function. Attention can down-weight evidence nodes that are frequent or uninformative in the current context, and up-weight rare diagnostic cues.

The output record embedding after L layers is $h_i^{(L)}$. Candidate generation then retrieves, for each record i, a small set of nearest neighbor records in the embedding space. However, naive global nearest neighbor search can be dominated by popular entities or by clusters of similar but distinct entities, especially when embeddings are influenced by common evidence [18]. Graph neural blocking therefore uses a constrained retrieval strategy that combines approximate nearest neighbor search with graph-aware pruning. One approach is to restrict retrieval to a union of neighborhoods induced by evidence nodes, ensuring that candidates share at least some explicit evidence, while still allowing multi-hop influence to shape embeddings. Let $\mathcal{U}(i)$ be a set of evidence nodes selected for record i using attention weights or inverse document frequency heuristics. Let $\mathcal{R}(i)$ be the set of records incident to those evidence nodes, possibly truncated to limit degree. Candidate retrieval then searches within $\mathcal{R}(i)$ using embedding similarity. This yields a candidate

set:

$$\mathcal{C}(i) = \mathrm{TopK}\left(\{j \in \mathcal{R}(i) \setminus \{i\}\}, \quad \mathrm{sim}\left(h_i^{(L)}, h_j^{(L)}\right)\right), \tag{3.4}$$

where sim may be cosine similarity or a learned bilinear score, and TopK selects the k most similar candidates. The global candidate set is $\mathcal{P} = \{(i, j) : j \in \mathcal{C}(i)\}$ symmetrized as needed.

Training graph neural blocking requires supervision that aligns embeddings with retrieval goals [19]. In many entity resolution deployments, labeled pairs are sparse and biased toward easier cases. To address this, training can combine supervised pairwise loss with self-supervised contrastive objectives and hard negative mining. Let $\mathcal{L}$ be a set of labeled match pairs and $\mathcal{N}$ a set of labeled non-match pairs. A supervised loss can use logistic regression on a pairwise score:

$$s_{ij} = h_i^{(L)\top} W_s h_j^{(L)}, \quad p_{ij} = \sigma(s_{ij}), \quad \mathcal{J}_{\mathrm{sup}} = -\sum_{(i,j)\in\mathcal{L}} \log p_{ij} - \sum_{(i,j)\in\mathcal{N}} \log(1 - p_{ij}), \tag{3.5}$$

where W_s is learnable and σ is the sigmoid [20]. For retrieval, it is often beneficial to emphasize hard examples where records share common tokens but are not the same entity, and where matches differ substantially due to missingness or transformation. A contrastive objective over mini-batches can be used, where each record has one or more positives and multiple negatives. Let $P(i)$ be a set of positives for i, and $Q(i)$ a set of negatives, drawn from retrieval neighborhoods to approximate the deployment distribution. A temperature-scaled contrastive loss can be:

$$\mathcal{J}_{\mathrm{con}} = -\sum_i \sum_{j\in P(i)} \log \frac{\exp\left(\mathrm{sim}(h_i^{(L)}, h_j^{(L)})/T\right)}{\exp\left(\mathrm{sim}(h_i^{(L)}, h_j^{(L)})/T\right) + \sum_{k\in Q(i)} \exp\left(\mathrm{sim}(h_i^{(L)}, h_k^{(L)})/T\right)}, \tag{3.6}$$

where T is a temperature parameter [21]. Robustness to missingness and corruption can be encouraged by augmentations that drop fields, mask tokens, introduce character noise, or perturb numerical values within realistic ranges. Let $\mathcal{A}$ be an augmentation distribution over records. For each record i, draw two augmentations x_i' and x_i'' and encode them as embeddings h_i' and h_i'', then treat (h_i', h_i'') as a positive pair. This self-supervised signal helps the model learn invariances that are relevant for entity resolution, while the graph structure provides relational context for disambiguation.

Graph construction itself affects robustness [22]. Evidence nodes based solely on raw tokens can amplify noise and frequent token dominance. To mitigate this, evidence extraction can combine multiple granularities, such as character n-grams for robustness to misspellings and token-level features for semantic cues. Structured identifiers can be represented as separate evidence types with higher confidence. Numeric values can be bucketed to handle rounding and format differences. Field-type awareness can be introduced by connecting record nodes not only to value nodes but also to field nodes, allowing the model to distinguish the role of the same token appearing in different fields [23]. This is important in heterogeneous schemas where a token might represent a location in one source and a product name in another.

The retrieval budget constraint can be enforced by training-time objectives that penalize overly dense candidate sets. One way is to constrain the embedding space using a learned hashing or prototype assignment, producing soft block IDs that limit comparisons. Let $c_1, \ldots, c_B$ be a set of prototypes representing latent blocks, and define assignment

probabilities: [24]

$$q_{ib} = \frac{\exp\left(h_i^{(L)\top} c_b\right)}{\sum_{b'=1}^{B} \exp\left(h_i^{(L)\top} c_{b'}\right)}, \quad \text{with block selection} \quad b(i) = \arg\max_b q_{ib}. \tag{3.7}$$

Candidates can then be generated primarily within the same block or among a small set of top blocks per record, reducing cross-block comparisons. Unlike deterministic blocking keys, these prototypes are learned to preserve match locality. Robustness arises because assignment is soft and based on aggregated graph evidence, so partial corruption of a field may not move a record to a completely unrelated block if other evidence remains consistent.

The computational cost of graph neural blocking depends on graph size and sampling strategy [25]. With $|V_R| = n$ record nodes and $|V_U| = m_U$ evidence nodes, and $|E|$ edges connecting them, full-batch message passing costs $O(|E|d)$ per layer for embedding dimension d, which is infeasible for very large graphs. Mini-batch training and inference use neighbor sampling. For a batch of b record nodes, sample up to s evidence neighbors per record and up to r record neighbors per evidence node for multi-hop aggregation. This yields a sampled subgraph whose size is $O(bs+bsr)$ per layer, controlling compute. During inference, embeddings can be computed either by full-graph propagation in a distributed setting or by approximate propagation using cached evidence embeddings and single-hop aggregation [26]. The latter is particularly attractive for streaming updates, where only a small fraction of records change at each time step.

Graph neural blocking produces two outputs useful for consolidation: a candidate edge set $\mathcal{P}$ and, optionally, a preliminary pairwise score s_{ij} for $(i, j) \in \mathcal{P}$ based on embedding similarity. Even when a separate matcher is used downstream, the embedding similarity is valuable as a prior, and the graph neighborhood features can be used to calibrate uncertainty. In particular, edges proposed due to high-degree evidence nodes can be flagged as low-confidence unless reinforced by more specific evidence, while edges proposed due to rare shared evidence can receive higher initial weight. This coupling between graph structure and retrieval is a central mechanism by which graph neural blocking improves robustness relative to purely attribute-local encoders [27].

4 Incremental Cluster Consolidation

Once candidate pairs are generated and scored, the system must consolidate records into entity clusters while maintaining transitivity and supporting incremental updates. The consolidation problem is challenging because pairwise scores are noisy and incomplete: only pairs in $\mathcal{P}$ are scored, and the absence of an edge does not necessarily imply non-match. Moreover, in streaming settings, evidence arrives over time, and premature merges can create large clusters that are difficult to correct. Incremental cluster consolidation addresses these constraints by maintaining a dynamic partition of records into clusters, using calibrated evidence accumulation and conservative merge rules that account for uncertainty and cluster size.

Let $\mathcal{C}(t)$ denote the set of clusters at time t, forming a partition of records observed up to t. Each cluster $C \in \mathcal{C}(t)$ is a set of record indices. The consolidator receives a stream of candidate edges (i, j, s_{ij}, π_{ij}), where s_{ij} is a score and π_{ij} includes auxiliary provenance such as the evidence type, model version, and timestamp. The consolidator must decide whether to merge the clusters containing i and j [28]. A naive approach merges whenever s_{ij} exceeds a threshold, but this can propagate errors. A more robust approach interprets

s_{ij} as probabilistic evidence and aggregates evidence across multiple edges between clusters before merging.

A convenient representation is to map s_{ij} to a log-likelihood ratio ℓ_{ij} that quantifies evidence in favor of $y_{ij} = 1$ versus $y_{ij} = 0$. If p_{ij} is a calibrated match probability, then:

$$\ell_{ij} = \log \frac{p_{ij}}{1 - p_{ij}}. \tag{4.1}$$

Calibration is important because consolidation decisions are sensitive to score magnitudes. In practice, p_{ij} may come from a logistic model over features, from a neural matcher with temperature scaling, or from isotonic calibration on a validation set. When labels are sparse, calibration can be performed using weak constraints and conservative priors, aiming to avoid overconfident probabilities.

For two clusters A and B, define an aggregate merge evidence $\Lambda(A, B)$ computed from the edges observed between records in A and records in B. A direct sum of log-likelihood ratios over all cross edges would be:

$$\Lambda(A, B) = \sum_{i \in A} \sum_{j \in B \,\cap\, \mathcal{N}(i)} \ell_{ij}, \tag{4.2}$$

where $\mathcal{N}(i)$ denotes the set of records j for which an edge (i, j) has been observed. However, enumerating all cross edges is infeasible when clusters grow [29]. Moreover, summing all evidence can overweight redundant edges induced by near-duplicate records or repeated observations. A robust consolidator therefore uses bounded evidence accumulation. One strategy is to maintain, for each cluster, a small set of representative records and a summary embedding, and to compute merge evidence using a sample of cross edges or using representative-to-representative comparisons.

Let $\rho(C)$ be a set of representatives for cluster C, chosen to cover its attribute diversity and sources [30]. Let g_C be a cluster embedding computed by aggregating record embeddings and structured statistics. If h_i are record embeddings from the blocking stage, then:

$$g_C = \frac{1}{|C|} \sum_{i \in C} h_i, \quad \text{or more robustly} \quad g_C = \frac{\sum_{i \in C} \omega_i h_i}{\sum_{i \in C} \omega_i}, \tag{4.3}$$

where ω_i down-weights low-quality records or highly redundant records. The cluster embedding is not used to declare matches by itself, but to guide which cluster pairs warrant deeper evidence accumulation [31].

A conservative merge criterion can combine aggregate edge evidence with size-aware regularization. Let $|A|$ and $|B|$ denote cluster sizes. Merging large clusters should require stronger evidence because the cost of a false merge is higher. A simple size-aware threshold is:

$$\text{merge}(A, B) \;\text{ if }\; \Lambda(A, B) \geq \theta_0 + \theta_1 \log(1 + |A|) + \theta_1 \log(1 + |B|), \tag{4.4}$$

where θ_0 and θ_1 are parameters tuned to control precision-recall trade-offs [32]. The logarithmic dependence captures that the risk grows with size but not necessarily linearly, and it prevents small clusters from being unfairly penalized. More refined criteria incorporate uncertainty estimates, such as the variance of evidence across representatives, or the fraction of evidence contributed by high-confidence edges.

Incremental operation requires efficient data structures. A union-find structure supports near-constant-time merges and cluster membership queries, but it does not support

splits [33]. Since splits are sometimes necessary to correct errors, the consolidator can implement a two-phase commitment mechanism. In this mechanism, candidate merges are first recorded as tentative in a merge graph over clusters, and only committed when sufficient evidence accumulates and when the merge is consistent with negative constraints. Negative constraints arise from rules such as distinct identifiers that should not co-occur, mutually exclusive attributes, or high-confidence non-match edges. Let ν_{ij} denote negative evidence, represented as a log-likelihood ratio in favor of separation. For clusters A and B, define negative aggregate:

$$\Gamma(A, B) = \sum_{i \in A} \sum_{j \in B \cap \mathcal{N}(i)} \nu_{ij}. \tag{4.5}$$

A merge decision can then consider the net evidence: [34]

$$\Delta(A, B) = \Lambda(A, B) - \lambda \Gamma(A, B), \tag{4.6}$$

where λ controls how strongly negative evidence is enforced. If negative evidence is treated as hard, then any substantial negative constraint blocks the merge until resolved, which is useful in domains where certain identifiers are trusted.

A central difficulty in consolidation is that evidence is incomplete. The absence of observed edges between two clusters does not mean they should not merge, because blocking may not have produced those candidate pairs [35]. This motivates tight coupling between blocking and consolidation. When consolidation considers merging clusters, it can request additional candidates from the blocking component by querying the embedding index using cluster representatives. This feedback loop is constrained by budget and is invoked only when necessary, such as when two clusters are near the merge threshold. This approach reduces the risk that a merge is made based on a small number of noisy edges, because the system can actively seek corroborating evidence when the decision is consequential [36].

Incremental cluster consolidation can be related to an online approximation of correlation clustering. If each observed edge (i, j) has a signed weight a_{ij} where positive favors co-clustering and negative favors separation, then the correlation clustering objective encourages a partition that satisfies high-weight edges. An online algorithm processes edges sequentially and updates the partition to reduce objective. Exact online optimization is not feasible, but the consolidator can implement a local decision rule that approximates the objective. Consider two clusters A and B. If they are separate, keeping them separate incurs a penalty from positive edges between them, while merging incurs a penalty from negative edges between them and potentially from negative edges within the merged cluster [37]. A merge is favorable when positive cross evidence exceeds negative cross evidence by a margin that accounts for uncertainty:

$$\text{merge}(A, B) \quad \text{if} \quad \sum_{(i,j) \in E(A,B)} a_{ij} \geq \theta \quad \text{and} \quad \sum_{(i,j) \in E^-(A,B)} |a_{ij}| \quad \text{is sufficiently small,} \tag{4.7}$$

where $E(A, B)$ are observed edges between A and B and $E^-(A, B)$ are those with negative sign. The practical implementation uses bounded samples and representative edges, but the inequality captures the intuition that merging is justified when the net evidence supports it.

Robustness requires preventing a few spurious edges from triggering large merges. One mechanism is evidence capping, where only the strongest M positive edges and strongest M negative edges between two clusters are considered, reducing redundancy and limiting the influence of noisy bursts [38]. Another mechanism is source diversity weighting, where

evidence from independent sources is weighted more than repeated evidence from the same source. Let $s(i)$ denote the source of record i. For a cross edge (i, j), define a weight α_{ij} that down-weights same-source edges if same-source duplication is common:

$$\alpha_{ij} = \begin{cases} \alpha_{\text{same}} & \text{if } s(i) = s(j), \\ \alpha_{\text{diff}} & \text{if } s(i) \neq s(j), \end{cases} \quad \text{with} \quad 0 < \alpha_{\text{same}} \leq \alpha_{\text{diff}}. \tag{4.8}$$

Then aggregate evidence becomes $\Lambda(A, B) = \sum \alpha_{ij} \ell_{ij}$ over selected edges. This reduces the risk that a dense batch of near-duplicate same-source records creates an overconfident merge.

Streaming and entity evolution introduce temporal considerations [39]. Records may reflect different states of an entity, and some attributes may drift. The consolidator can incorporate temporal decay in evidence so that older, less reliable edges contribute less to current decisions. Let t_{ij} be the time the edge was produced, and let current time be t. A decay factor $d(t, t_{ij})$ can be:

$$d(t, t_{ij}) = \exp\left(-\gamma(t - t_{ij})\right), \quad \text{so that} \quad \ell_{ij}^{(t)} = d(t, t_{ij}) \cdot \ell_{ij}, \tag{4.9}$$

where γ controls half-life. Decay is not universally appropriate, since some evidence such as stable identifiers should not decay. Therefore, decay can be conditioned on evidence type, applying primarily to attributes known to change [40].

Incremental consolidation also needs a notion of cluster quality and a mechanism for handling ambiguous cases. Rather than forcing all records into a single best cluster, the system can maintain ambiguity by delaying merges that are not sufficiently supported, while still allowing near-duplicate consolidation. Ambiguity control improves robustness by preventing premature merges in dense neighborhoods such as common names or popular products. Operationally, this is implemented by increasing thresholds in high-ambiguity regions, detected via local density in embedding space or via high-degree evidence nodes [41]. If a cluster has many near neighbors with similar scores, then the merge threshold for that cluster can be increased until additional evidence is collected, reducing the risk of chaining errors.

Finally, consolidation must integrate with the possibility of late corrections, such as human feedback or improved models. Because union-find does not support splits, the consolidator maintains an audit trail of merge evidence and supports periodic revalidation. Revalidation can be implemented as a background reconciliation job that re-scores edges among records in large clusters or among clusters with low internal consistency [42]. Internal consistency can be measured by the distribution of match scores among representative pairs within a cluster. If inconsistency exceeds a threshold, the cluster can be marked for review and potentially reconstructed using a more expensive batch clustering algorithm on the clusters neighborhood. While such reconstruction is not part of the core online path, it provides a safety valve that improves long-run robustness.

5 Scalable Implementation and Empirical Evaluation

The combined system is intended for the regime where n is large, data is heterogeneous, and updates are continuous. A scalable implementation separates the pipeline into stages with clear interfaces: graph construction, embedding inference, candidate retrieval, pairwise scoring, and consolidation [43]. Each stage is designed to be distributed and to operate under bounded memory. The core challenge is that the graph neural blocking stage

introduces a large graph structure, while the consolidation stage requires stateful cluster management. Achieving scalability therefore depends on careful partitioning, caching, and asynchronous processing.

Graph construction begins by extracting evidence elements from records [44]. This stage is embarrassingly parallel over records, but the resulting evidence nodes must be deduplicated and assigned identifiers, which typically requires distributed shuffling. To reduce shuffle volume, evidence elements can be hashed and deduplicated approximately, then refined for high-value evidence types such as stable identifiers. Evidence nodes that are extremely frequent are treated specially to avoid high-degree blowups. A practical strategy is to cap the number of incident records retained for a high-frequency evidence node, using a reservoir sampling scheme that retains a representative subset. While capping can reduce recall if a true match depends only on a common token, the graph neural model can still use other evidence and multi-hop context, and the capped sampling can be biased toward retaining diverse records rather than arbitrary ones [45].

Embedding inference at scale uses mini-batch neighbor sampling, with a deployment choice between full refresh and incremental refresh. Full refresh computes embeddings for all records periodically, which simplifies consistency but can be expensive. Incremental refresh computes embeddings for new or modified records and for a bounded neighborhood around them. Incremental refresh is feasible because message passing in a bipartite evidence graph primarily depends on local neighborhoods, and because evidence node embeddings can be cached and updated incrementally [46]. Let ΔV_R be the set of record nodes updated at a time step and ΔV_U the set of evidence nodes incident to them. An incremental embedding update computes new embeddings for ΔV_R using cached embeddings for ΔV_U and possibly for second-hop record neighbors. The cost is then proportional to the number of changed edges rather than to $|E|$.

Candidate retrieval must satisfy the budget constraints discussed earlier [47]. A common approach is to maintain an approximate nearest neighbor index over record embeddings. At very large scale, a single global index can be sharded across machines, with routing based on learned block prototypes or on locality-sensitive hashing of embeddings. The graph neural blocking design naturally provides a routing mechanism via prototype assignments, enabling most queries to be handled within a small subset of shards. The retrieval stage also needs to avoid returning many near-duplicates that do not add new evidence for consolidation. Diversity-aware retrieval can be implemented by down-weighting candidates that share the same high-frequency evidence or that are already strongly connected via existing cluster edges [48].

Pairwise scoring can use a matcher that combines embedding similarity with field-level comparisons and graph-derived features. Since the candidate set is bounded, scoring can be performed in a distributed micro-batch fashion. Robustness benefits from including features that detect unreliable evidence, such as the proportion of shared tokens that are high-frequency, the presence of conflicting stable identifiers, and the consistency of structured fields. However, the scoring model must remain efficient [49]. A practical compromise is to compute a small set of engineered features that capture key reliability signals and combine them with neural similarity in a lightweight classifier. Calibration of probabilities is performed periodically using available labels and monitoring feedback, since consolidation depends on calibrated evidence.

The consolidation stage maintains cluster state and must support concurrent updates. A scalable design partitions clusters across workers by a cluster identifier, using consistent hashing [50]. Candidate edges are routed to the workers responsible for the clusters they connect. When an edge connects clusters on different workers, a coordination protocol

is needed to decide merges. A common pattern is to designate a leader worker for each potential merge based on a deterministic ordering, such as the smaller cluster ID, and to perform a two-phase merge where both workers agree to the merge after verifying negative constraints and evidence thresholds. This protocol ensures that merges are consistent and that union-find operations do not produce cycles or contradictions across partitions. The latency of cross-partition merges is controlled by batching and by prioritizing merges with strong evidence [51].

State size is controlled by storing only bounded summaries rather than all edges. For each cluster, the system stores representatives, a cluster embedding, counts of evidence types, and a small set of boundary edges to neighboring clusters with significant evidence. Boundary edges are maintained in a priority structure keyed by net evidence $\Delta(A, B)$, allowing the consolidator to focus on the most promising merges. When a merge occurs, summaries are combined and boundary edges are reconciled [52]. This resembles dynamic graph clustering, but with evidence-aware merge criteria. Because boundary maintenance can become expensive if clusters have many neighbors, the system applies caps and pruning rules, such as discarding neighbors whose evidence is weak and old, or keeping only top neighbors per representative.

Empirical evaluation of such a system must consider both matching quality and system-level metrics. Matching quality can be assessed by cluster-level precision and recall measures that reflect the partition quality, such as pairwise F_1 and cluster F_1 [53]. Blocking quality can be assessed by pairwise recall of labeled matches and by reduction ratio, the fraction by which comparisons are reduced relative to all-pairs. System-level metrics include throughput, p95 latency for record insertion and update, memory footprint per record, and recovery time after failures. Robustness evaluation involves stress tests where specific corruption patterns are applied, such as random character noise, systematic field swapping, missingness in key identifiers, frequency shifts where common tokens become more common, and adversarial injections that attempt to create hub records sharing many tokens.

In qualitative terms, graph neural blocking tends to improve candidate recall for difficult matches where direct field overlap is weak but relational evidence exists, such as shared rare tokens across multiple fields, or indirect connections through normalized identifiers. The attention and normalization mechanisms reduce the dominance of frequent tokens, which is critical in domains like person names and product descriptions [54]. Under moderate corruption, embedding neighborhoods remain stable because the message passing aggregates across multiple evidence nodes, and because augmentations during training encourage invariance to missing or noisy fields. In contrast, deterministic blocking keys can fail abruptly when a key field is missing or corrupted, causing zero recall for those matches. Embedding-based blocking without graph context can also degrade when corruption removes salient tokens, whereas graph context can preserve signal through other connected evidence.

Incremental cluster consolidation tends to reduce error propagation compared to threshold-based union of edges [55]. Size-aware thresholds and evidence aggregation prevent single spurious edges from merging large clusters. Representative-based evidence ensures that merges are supported by multiple consistent comparisons rather than by redundant near-duplicates. Temporal decay, when used selectively, improves behavior under drift by preventing outdated attribute similarity from dominating merges, while stable identifiers remain strong evidence. Ambiguity control reduces chaining in dense regions, improving precision at the cost of delayed merges. In environments where delayed merges are acceptable, this trade-off is favorable for robustness [56]. In environments requiring immediate

consolidation, thresholds can be tuned lower but must be paired with stronger negative constraints and more frequent revalidation.

Scalability analysis can be framed in terms of the comparison budget B and per-record candidate cap k. If each record generates at most k candidates, then $|\mathcal{P}| \leq nk$, which is linear in n. Scoring cost is then $O(nk)$ times the cost per comparison. Consolidation cost depends on the number of boundary edges maintained per cluster and on the number of merges [57]. If each cluster maintains at most b boundary neighbors, then the boundary maintenance cost is $O(\hat{m}b \log b)$ for priority operations, where $\hat{m}$ is the number of clusters. Since $\hat{m}$ is typically on the order of n early and decreases as consolidation proceeds, worst-case behavior can occur if many ambiguous clusters maintain many neighbors. This motivates pruning and capping. Graph neural inference cost depends on the sampled neighborhood sizes. With neighbor sampling parameters s and r and L layers, the cost per updated record is roughly proportional to $O(Lsrd)$, which can be bounded in deployment by choosing small sampling sizes and using cached embeddings for evidence nodes.

A key engineering challenge is consistency between embedding space and cluster state under streaming updates [58]. If embeddings drift as the model is updated or as evidence distribution shifts, candidates retrieved for an updated record might be inconsistent with prior consolidation decisions. The system mitigates this by versioning embeddings and scores, storing model version metadata in π_{ij}, and treating evidence from different model versions with calibrated adjustment factors. When a new model is deployed, consolidation can be run in a dual mode where new edges are generated and compared against existing clusters without immediately undoing prior merges, while revalidation gradually reconciles inconsistencies. This reduces oscillations and avoids destabilizing the cluster state.

Another practical concern is long-tail entities and rare evidence. Graph neural blocking can struggle if rare evidence nodes have too few incident records to learn meaningful embeddings, or if the evidence graph is highly fragmented [59]. This is addressed by combining multiple evidence types, including character-level features that generalize across tokens, and by using self-supervised augmentations to learn invariances even when labeled matches are scarce. Consolidation can struggle when an entity has multiple distinct profiles over time, such as a person changing address or a product changing name. In such cases, rigid transitivity can force merges that are semantically questionable. A pragmatic response is to incorporate temporal modeling and to treat certain attributes as time-dependent, reducing their contribution to match evidence when they conflict across time windows [60]. This does not eliminate the need for transitivity, but it reduces the likelihood that time-varying attributes trigger false non-matches within a true entity.

Robustness to adversarial patterns can be evaluated by injecting records that share many common tokens or that mimic high-value entities. Graph neural blockings degree normalization and attention reduce the impact of hub evidence, but an adversary could still craft records that share multiple rare tokens. Consolidation counters this by requiring multi-source diversity and by penalizing merges that would create unusually high-degree clusters or that would violate negative constraints [61]. Monitoring can detect abnormal clusters by measuring cluster growth rate, evidence diversity, and inconsistency scores. When anomalies are detected, merges can be slowed or subjected to additional verification, such as requesting extra candidate edges from representatives or requiring human review in high-risk domains.

Overall, the scalable implementation emphasizes bounded per-record work, bounded per-cluster state, and asynchronous processing. Graph neural blocking provides adaptive candidate generation that maintains recall under heterogeneity, while incremental consolidation provides controlled cluster growth that limits error propagation. Empirical behavior

under stress conditions suggests that the combined approach offers stable trade-offs across a range of noise and drift patterns, though it remains sensitive to calibration quality and to the design of evidence extraction [62]. The system is therefore best deployed with continuous monitoring and periodic recalibration rather than as a static one-time process.

6 Conclusion

This paper presented a robust entity resolution framework for large-scale and streaming environments by coupling graph neural blocking with incremental cluster consolidation. The blocking component models records and attribute evidence as a heterogeneous graph and learns retrieval-oriented embeddings through message passing, enabling candidate generation that is adaptive to schema heterogeneity, missingness, and skewed token distributions while respecting strict comparison budgets. The consolidation component maintains entity clusters incrementally, interpreting pairwise scores as calibrated evidence, aggregating support across representatives, and applying size-aware and constraint-aware merge rules to reduce error propagation and improve stability under drift and noise [63]. The resulting pipeline addresses the practical limitations of deterministic blocking and aggressive clustering by making candidate generation relational and making clustering conservative and evidence-accumulative.

The approach emphasizes that robustness at scale is not solely a property of the matcher, but an end-to-end property that depends on how candidates are proposed, how evidence is calibrated, and how transitivity is enforced over time. Graph structure helps preserve recall when direct attribute overlap is insufficient, while incremental consolidation helps preserve precision by delaying or rejecting merges that lack corroboration. The design also highlights the operational importance of bounded memory summaries, distributed coordination for merges, and mechanisms for revalidation under model updates. Future refinements within the same conceptual framework include more explicit uncertainty modeling for merge decisions, better handling of temporally evolving entities, and tighter feedback between consolidation and candidate generation to allocate comparison budget where it most reduces ambiguity [64].

References

[1] B. Finkbeiner, M. Gieseking, J. Hecking-Harbusch, and E.-R. Olderog, *Global winning conditions in synthesis of distributed systems with causal memory.* Jul. 20, 2021.

[2] M. Broy, *Concurrent distributed systems beyond monotonicity*, Jan. 1, 2024.

[3] R. Fronteddu et al., "Semantic information management systems," in *2025 International Conference on Military Communication and Information Systems (ICMCIS)*, IEEE, May 13, 2025, pp. 1–9.

[4] R. Malik et al., "Mlr-index: An index structure for fast and scalable similarity search in high dimensions," in *International Conference on Scientific and Statistical Database Management*, Springer, 2009, pp. 167–184.

[5] A. Estevan, "An approach to distributed systems from orderings and representability," *Bulletin of the Iranian Mathematical Society*, vol. 50, no. 3, Apr. 8, 2024.

[6] G. Heinrich and I. Frosio, *Metaoptimization on a distributed system for deep reinforcement learning*, Feb. 7, 2019.

[7] A. Khole, A. Thakar, A. Kulkarni, H. Jadhav, S. Shende, and V. Karajkhede, *A compendium on distributed systems*, Jan. 1, 2023.

[8] Z. Lu, W. Yu, P. Xu, W. Wang, J. Zhang, and D. Feng, "An ntt/intt accelerator with ultra-high throughput and area efficiency for fhe," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, ACM, Jun. 23, 2024, pp. 1–6.

[9] M. Farhadi, D. Miorandi, and G. Pierre, *Blockchain enabled fog structure to provide data security in iot applications*, Dec. 10, 2018.

[10] P. Afanasev, A. Ilyushina, S. Kolesnichenko, P. Komissarov, and E. Zheleznov, "Environmental monitoring using distributed system theory," in *SGEM International Multidisciplinary Scientific GeoConference EXPO Proceedings*, vol. 21, STEF92 Technology, Dec. 20, 2021, pp. 247–254.

[11] T. Kharkovskaia, *Design of interval observers for uncertain distributed systems*, Dec. 2, 2019.

[12] K. M. Goeschka, R. P. S. de Oliveira, P. Pietzuch, and G. Russello, "Session details: Theme: Distributed systems: Dads - dependable, adaptive, and secure distributed systems track," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ACM, Apr. 8, 2019.

[13] J. Jaros, D. Hruby, V. Vasinek, and T. Stratil, "Measuring of the petroleum product leaks by distributed systems," in *Advanced Optical Techniques for Quantum Information, Sensing, and Metrology*, vol. 11295, SPIE, Feb. 28, 2020, pp. 133–139.

[14] I. Argyroulis, *Recent advancements in distributed system communications*, Jan. 1, 2021.

[15] L. Kroll, *Compile-time safety and runtime performance in programming frameworks for distributed systems*, Mar. 14, 2020.

[16] M. L. Nunes, *Backscatter readers for single and distributed systems*, Dec. 1, 2019.

[17] Z. Dong, C. Tang, J. Wang, Z. Wang, H. Chen, and B. Zang, "Optimistic transaction processing in deterministic database," *Journal of Computer Science and Technology*, vol. 35, no. 2, pp. 382–394, Mar. 27, 2020.

[18] R. Chandrasekar, R. Suresh, and S. Ponnambalam, "Evaluating an obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 International Conferaence on Advanced Computing and Communications*, IEEE, 2006, pp. 628–629.

[19] F. Kelbert and A. Pretschner, "Data usage control for distributed systems," *ACM Transactions on Privacy and Security*, vol. 21, no. 3, pp. 12–32, Apr. 16, 2018.

[20] M. Jani, J. Slak, and G. Kosec, "P-refined rbf-fd solution of a poisson problem," in *2021 6th International Conference on Smart and Sustainable Technologies (SpliTech)*, IEEE, Sep. 8, 2021, pp. 1–6.

[21] C. Gao, Y. Zhong, and X. He, "Safety control of distributed systems," in Elsevier, Jan. 1, 2024, pp. 153–159.

[22] Y. Wang, "A distributed system fault diagnosis system based on machine learning," *Scalable Computing: Practice and Experience*, vol. 25, no. 2, pp. 1117–1123, Feb. 24, 2024.

[23] G. N. Abdukarimovich, K. M. Khudainazarovna, and S. S. Ravshabekovna, "Building models of territorial distributed systems," *International Journal on Integrated Education*, vol. 3, no. 10, pp. 300–303, Oct. 24, 2020.

[24] D. Marek, P. Biernacki, J. Szygula, and A. Domanski, "General concepts of a simulation method for automated guided vehicle in industry 4.0," in *2022 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 17, 2022, pp. 6306–6314.

[25] Y. Tian et al., "Error messages to fuzzing: Detecting xps parsing vulnerabilities in windows printing components," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Nov. 19, 2025, pp. 798–812.

[26] A. Bretas, D. Wang, O. Vasios, and J. Ogle, "Bi-level linear programming model for automatic load shedding: A distributed wide-area measurement system-based solution," in *2023 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, IEEE, Jan. 16, 2023, pp. 1–5.

[27] T. Maalouf, *Performance optimizations of nosql databases in distributed systems*, Dec. 7, 2020.

[28] N. Phu, *Clara algorithm in a distributed system*, Sep. 5, 2018.

[29] V. K. Yadav, *Improve the scalability of system through distributed system*, Mar. 31, 2021.

[30] F. Hackett, S. Hosseini, R. Costa, M. Do, and I. Beschastnikh, "Compiling distributed system models with pgo," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ACM, Jan. 27, 2023, pp. 159–175.

[31] B. W. Watson, E. Badouel, and O. Niang, *Foreword of proceedings of cari 2020*, Aug. 31, 2020.

[32] R. Chandrasekar and T. Srinivasan, "An improved probabilistic ant based clustering for distributed databases," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*, 2007, pp. 2701–2706.

[33] R. Mayerhofer, A. Morichetta, A. Furutanpey, and S. Dustdar, "Hpaqt: Adaptive and interpretable high-level slo-aware autoscaling with reinforcement learning," in *Proceedings of the 18th IEEE/ACM International Conference on Utility and Cloud Computing*, ACM, Dec. 31, 2025, pp. 1–9.

[34] M. Inoue and A. C. Badallo, *Parallel and Distributed Systems*. Feb. 1, 2019.

[35] M. R. Ogiela and U. Ogiela, "Ai for security of distributed systems," *WSEAS TRANSACTIONS ON COMPUTER RESEARCH*, vol. 12, pp. 450–454, Oct. 21, 2024.

[36] H. Benítez-Pérez, J. L. Ortega-Arjona, P. E. Méndez-Monroy, E. Rubio-Acosta, and O. A. Esquivel-Flores, "Distributed systems modelling," in Germany: Springer International Publishing, Aug. 1, 2018, pp. 41–82.

[37] S. M. Meshkani and B. Farooq, *A decentralized shared cav system design and application.* Apr. 17, 2021.

[38] R. Faiyaz and M. K. Abdul, "Strategic role of distributed systems in enhancing organisational agility," *Journal of Policies and Recommendations*, vol. 4, no. 4, pp. 1–, Dec. 26, 2025.

[39] T. Srinivasan, R. Chandrasekar, V. Vijaykumar, V. Mahadevan, A. Meyyappan, and A. Manikandan, "Localized tree change multicast protocol for mobile ad hoc networks," in *2006 International Conference on Wireless and Mobile Communications (ICWMC'06)*, IEEE, 2006, pp. 44–44.

[40] D. Schneider and B. Uekermann, "Efficient partition-of-unity radial-basis-function interpolation for coupled problems," *SIAM Journal on Scientific Computing*, vol. 47, no. 2, B558–B582, Apr. 10, 2025.

[41] A. Hermann, M. Wolf, N. Trkulja, I. B. Jemaa, A. Bkakria, and F. Kargl, "Privacy of smart traffic lights systems," in *2023 IEEE Vehicular Networking Conference (VNC)*, IEEE, Apr. 26, 2023, pp. 17–24.

[42] J. Breyer, M. Petri, M. H. Alizai, and K. Wehrle, "A critical review of household water datasets," in *Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ACM, Oct. 29, 2024, pp. 318–322.

[43] Z. Lu et al., "An fpga-based key-switching accelerator with ultra-high throughput for fhe," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ACM, Oct. 27, 2024, pp. 1–9.

[44] R. Jalilova, "Computer networks and distributed systems in educational fields," *Internauka*, vol. 211, no. 35, Sep. 30, 2021.

[45] J. Pennekamp et al., "Mapxchange: Designing a confidentiality-preserving platform for exchanging technology parameter maps," in *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, ACM, Mar. 31, 2025, pp. 470–479.

[46] W. Wu, "Accelerating distributed systems by in-network computing," in *Proceedings of the ACM CoNEXT Workshop on In-Network Computing and AI for Distributed Systems*, ACM, Nov. 30, 2025, pp. 1–2.

[47] V. Vijaykumar, R. Chandrasekar, and T. Srinivasan, "An obstacle avoidance strategy to ant colony optimization algorithm for classification in event logs," in *2006 IEEE Conference on Cybernetics and Intelligent Systems*, IEEE, 2006, pp. 1–6.

[48] A. Nasibullin, "Fault tolerant hash join for distributed systems," *Computer tools in education*, no. 4, pp. 68–82, Dec. 28, 2022.

[49] G. R. S. Martinez, *Soa distributed systems architecture*, Sep. 30, 2021.

[50] A. Dawra et al., *12enhancing business development, ethics, and governance with the adoption of distributed systems*, Mar. 8, 2024.

[51] L. Alberto, "Pseudospheres: Combinatorics, topology and distributed systems," *Journal of Applied and Computational Topology*, vol. 8, no. 4, pp. 1023–1052, Feb. 10, 2024.

[52] Y. Yang et al., "Automated validating and fixing of text-to-sql translation with execution consistency," *Proceedings of the ACM on Management of Data*, vol. 3, no. 3, pp. 1–28, Jun. 17, 2025.

[53] B. V. S. Pinto, D. R. Melo, C. A. Zeferino, E. A. Bezerra, and F. Viel, "Implementation of double sha-256 in hls for fpga using real bitcoin blocks," in *2025 17th Seminar on Power Electronics and Control (SEPOC)*, IEEE, Nov. 9, 2025, pp. 1–7.

[54] G. A. Qadir and S. R. M. Zeebaree, *Evaluation of qos in distributed systems: A review*, May 10, 2021.

[55] J. Flak, T. Skowron, R. Cupek, M. Fojcik, D. Caban, and A. Domaski, "Zigbee network for agv communication in industrial environments," in *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, Oct. 9, 2023, pp. 1–9.

[56] B. Finkbeiner, M. Gieseking, J. Hecking-Harbusch, and E.-R. Olderog, *Global winning conditions in synthesis of distributed systems with causal memory (full version)*. Oct. 18, 2021.

[57] F. Lai, P. Zhang, R. Cheng, and P. Xu, "Distributed systems anomaly detection based on log," in *2021 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*, vol. 14, IEEE, Dec. 3, 2021, pp. 72–79.

[58] A. Fentis et al., "A machine learning based approach for next-day photovoltaic power forecasting," in *2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, IEEE, Oct. 21, 2020, pp. 1–8.

[59] L. Bantel, P. Domanski, and D. Pflüger, "High-fidelity simulation of a cartpole for sim-to-real deep reinforcement learning," in *2024 4th Interdisciplinary Conference on Electrics and Computer (INTCEC)*, IEEE, Jun. 11, 2024, pp. 1–6.

[60] P. Chakraborty, "Distributed systems," in Chapman and Hall/CRC, Oct. 17, 2023, pp. 437–557.

[61] R. Malik, C. Ramachandran, I. Gupta, and K. Nahrstedt, "Samera: A scalable and memory-efficient feature extraction algorithm for short 3d video segments.," in *IMMERSCOM*, 2009, p. 18.

[62] Y. Simmhan, "Pedagogic practices for teaching distributed systems courses," in *Proceedings of the 15th Annual ACM India Compute Conference*, ACM, Nov. 9, 2022, pp. 6–6.

[63] M. Zaki, *Balancing privacy, precision andperformance in distributed systems*, Jan. 16, 2020.

[64] I. Colonnelli and M. Aldinucci, *Workflow models for heterogeneous distributed systems*, May 10, 2022.