Received: 12 March 2026 | Accepted: 27 July 2026 | Published: 23 August 2026

ARK: 24285/RCC.v10i20.280

ARTICLE

REVISTA COLETA CIENTÍFICA

An incremental multi-view entity resolution framework for dynamic academic user profiles in big data environments

Um arcabouço incremental de resolução de entidades multivisão para perfis dinâmicos de usuários acadêmicos em ambientes de big data

Atika Badaoui [1]
https://orcid.org/0009-0002-3681-7277
University M'Hamed Bougara of Boumerdes, Algeria.
E-mail: boulahia.abdelhamid@gmail.com

Djamel Eddine Zegour[2]
https://orcid.org/0000-0001-9538-5895
National School of Computer Science, Oued Smar, Algeria.
Email: D_zegour@esi.dz

Walid Khaled Hidouci[3]
https://orcid.org/0000-0002-8290-1093
National School of Computer Science ISI, Oued Smar, Algeria.
Email: wk.hidouci@gmail.com

Amin Riad Maouche[4]
https://orcid.org/0000-0001-5713-5705
University M'Hamed Bougara of Boumerdes, Algeria
Email: armaouche@umbb.dz

Publication information

ARK: 24285/RCC.v10i20.280
ISSN: 2763-6496

Keywords:
Record Linkage.
Locality-Sensitive Hashing.
Hierarchical Navigable Small.
World Graphs.
Approximate Nearest.
Neighbor Search.
Multi-View Representation.
Incremental Indexing.

Palavras-chave:
Vinculação de Registros (Record Linkage)
Hashing Sensível à Localidade.
Grafos Hierarchical Navigable Small World.
Busca de Vizinhos Mais Próximos Aproximados.
Representação Multivisão.
Indexação Incremental.

Abstract

Entity Resolution (ER) in streaming Big Data environments demands dynamic, low-latency indexing architectures capable of resolving high-dimensional user profiles without incurring the computational overhead of complete index reconstruction. Conventional tree-based incremental indexing systems are fundamentally constrained by their inability to operate over semantic vector spaces, while Locality-Sensitive Hashing (LSH) blocking paradigms exhibit pronounced sensitivity to hash collisions, generating noisy candidate blocks that degrade downstream matching quality. This article proposes a unified, real-time incremental entity resolution framework engineered for multi-view academic user profiles. The proposed architecture integrates three synergistic components: (i) an incremental LSH-SimHash blocking layer that projects multi-view profile representations into candidate buckets while preserving high-dimensional angular relationships within a compact binary signature space; (ii) independent Hierarchical Navigable Small World (HNSW) graphs maintained within individual LSH buckets, functioning as localized approximate nearest neighbor (ANN) search engines that eliminate global structural reconstruction upon profile arrivals; and (iii) an Adaptive Candidate Selection (ACS) strategy that prunes retrieved neighbor sets using a multi-criteria scoring function jointly evaluating vector cosine similarity, graph neighborhood structural consistency, and cross-view semantic alignment. Comprehensive empirical evaluations conducted across three large-scale benchmark datasets—the Academic Profiles Dataset, the North Carolina Voter Registration Dataset, and the OZ Synthetic Dataset—demonstrate that the proposed framework achieves a peak F1-score of 96.84% while maintaining sub-4 ms end-to-end query resolution latencies and a stable, predictable memory footprint, consistently outperforming established tree-based and dynamic approximate blocking baselines across all evaluation dimensions.

[1] Doctoral student in Department of Computer Science, University M'Hamed Bougara of Boumerdes, Boumerdes, Algeria (2015); a Master's degree in in Fundamental Computer Science – University M'Hamed Bougara of Boumerdes, Algeria (2006).

[2] Professor at LCSI Laboratory (2010), National School of Computer Science ISI, Oued Smar, Algiers, Algeria (1989-2025). Postgraduation from Paris Dauphine University & INRIA (National Institute of Research in Computer Science and Automation), Paris, France (1988); a state doctorate, Algiers, Algeria (1998).

[3] Professor at LCSI Laboratory (2010), National School of Computer Science ISI, Oued Smar, Algiers, Algeria. Master's Degree (Magister) from National School of Computer Science, Algiers, Algeria(1993). a state doctorate from National School of Computer Science, Algiers, Algeria (1998).

[4] Professor at Department of Computer Science, University M'Hamed Bougara of Boumerdes, Boumerdes, Algeria (1999); Faculty of Electronics and Computer Science, University of Science and Technology Houari Boumediene, Algiers, Algeria (1999). PhD in Robotics from the Faculty of Electronics and Computer Science at the University of Science and Technology in Algiers, Algeria (2010).

REVISTA COLETA CIENTÍFICA

Resumo

A Resolução de Entidades (RE) em ambientes de Big Data em fluxo contínuo (*streaming*) exige arquiteturas de indexação dinâmicas e de baixa latência, capazes de resolver perfis de usuários de alta dimensão sem incorrer na sobrecarga computacional da reconstrução completa de índices. Sistemas convencionais de indexação incremental baseados em árvores são fundamentalmente limitados por sua incapacidade de operar sobre espaços vetoriais semânticos, enquanto os paradigmas de bloqueio por Hashing Sensível à Localidade (LSH) exibem uma acentuada sensibilidade a colisões de hash, gerando blocos candidatos ruidosos que degradam a qualidade do pareamento subsequente. Este artigo propõe um *framework* unificado e em tempo real de resolução incremental de entidades, projetado para perfis de usuários acadêmicos multivisão. A arquitetura proposta integra três componentes sinérgicos: (i) uma camada de bloqueio incremental LSH-SimHash que projeta representações de perfis multivisão em *buckets* candidatos, preservando relações angulares de alta dimensão em um espaço compacto de assinaturas binárias; (ii) grafos independentes *Hierarchical Navigable Small World* (HNSW) mantidos dentro de *buckets* LSH individuais, funcionando como motores localizados de busca por vizinhos mais próximos aproximados (ANN) que eliminam a reconstrução estrutural global a cada chegada de novos perfis; e (iii) uma estratégia de Seleção Adaptativa de Candidatos (*Adaptive Candidate Selection* - ACS) que poda os conjuntos de vizinhos recuperados utilizando uma função de pontuação multicritério que avalia conjuntamente a similaridade de cosseno vetorial, a consistência estrutural da vizinhança no grafo e o alinhamento semântico entre visões. Avaliações empíricas abrangentes realizadas em três conjuntos de dados (*datasets*) de referência em grande escala — o *Academic Profiles Dataset*, o *North Carolina Voter Registration Dataset* e o *OZ Synthetic Dataset* — demonstram que o *framework* proposto atinge um F1-score de pico de 96,84%, enquanto mantém latências de resolução de consultas ponta a ponta inferiores a 4 ms e um consumo de memória estável e previsível, superando consistentemente as linhas de base (*baselines*) estabelecidas baseadas em árvores e em bloqueio aproximado dinâmico em todas as dimensões de avaliação.

1. Introduction

Entity Resolution (ER)—also referred to as record linkage or deduplication—constitutes a foundational operation in data integration research, concerned with the identification and grouping of records distributed across heterogeneous data sources that correspond to the same underlying real-world entity [1]. Within this paradigm, profile construction naturally leads to entity resolution, as aggregating fragmented record attributes into structured entity profiles provides the necessary foundation to identify matching records accurately. The problem commands critical importance across a broad spectrum of application domains, including biomedical data integration, academic knowledge graph construction, governmental record management, and large-scale social network analysis [2]. In Big Data environments characterized by massive record volumes, structural heterogeneity, and continuous data streams, executing exhaustive pairwise comparisons across all candidate records is computationally impractical: the comparison complexity scales quadratically with dataset cardinality at $O(N^2)$, rendering brute-force approaches infeasible beyond modest dataset sizes [3].

To overcome this fundamental scalability barrier, conventional ER methodologies adopt a structured two-phase processing pipeline. In the **blocking phase**, the global record space is partitioned into smaller candidate groups containing records that are plausibly similar according to a lightweight blocking criterion, dramatically reducing the volume of candidate pairs requiring detailed examination [4]. In the **matching phase**, computationally intensive pairwise similarity functions or supervised classification models are applied to the candidate pairs generated within each block [5]. While this architecture provides acceptable computational management for static, batch-processed databases, it exhibits severe inefficiencies in dynamic deployment environments where

user profiles are subject to continuous creation, modification, and deletion in real time [6].

The principal deficiency of batch-oriented ER pipelines in dynamic settings is their inability to process newly arriving records without triggering complete or substantial reconstruction of the underlying index structures. Re-executing a full batch ER pipeline upon each profile arrival event introduces unacceptable query latencies, particularly in systems operating under real-time or near-real-time service-level agreements [6]. This operational constraint has motivated a growing body of research dedicated to developing incremental and real-time entity resolution frameworks capable of processing streaming query profiles with bounded, sub-second latency while sustaining high resolution quality throughout the dynamic lifecycle of the index [7].

Several dynamic indexing paradigms have been proposed to address the challenges inherent to real-time ER. Ramadan et al. [8] introduced the Forest-based Dynamic Sorted Neighborhood Indexing (DySNI) approach, constructing a forest of AVL trees across multiple profile attributes to maintain dynamic sliding windows of candidate records. Evaluated on the OZ dataset comprising 345,876 dynamic records exhibiting typographical errors and attribute incompleteness, DySNI achieved average resolution latencies of 0.7–1.3 ms with recall values spanning 62–92% depending on tree configuration parameters. Subsequently, Zhu et al. [9] proposed a real-time ER system integrating hash tables with tree-based indices, achieving an average query resolution time of 1.2 ms, precision of 99.75%, and recall of 98.92% on the North Carolina voter registration dataset. To address noise in dynamic data streams, Liang et al. [10] proposed the Noise-Tolerant Approximate Blocking (NAB) framework, combining LSH-based bucket partitioning with dynamic B^+-trees for incremental candidate group maintenance.

Despite their demonstrated efficiency in managing index updates, these dynamic tree-based indexing structures exhibit fundamental architectural limitations that constrain their applicability in modern semantic ER contexts. AVL trees and B^+-trees are structurally designed for ordered, uni-dimensional key spaces and cannot natively scale to the high-dimensional continuous vector spaces increasingly indispensable for representing the semantic content of modern user profiles [18; 23]. As profile representations migrate from simple string attributes toward dense neural embeddings produced by large pre-trained language models, the incapacity of tree-based structures to index such representations constitutes a critical operational bottleneck [11]. Furthermore, LSH blocking is inherently susceptible to hash collisions, wherein semantically dissimilar records are mapped to identical candidate buckets, injecting substantial noise into the candidate pool and degrading the precision of the downstream matching stage [12].

To address these limitations within a unified architectural framework, this article introduces a real-time incremental entity resolution system that synergistically combines LSH-SimHash blocking with localized HNSW graphs, specifically engineered for multi-view academic user profiles. The proposed architecture makes the following principal scientific contributions:

1. **Incremental LSH-SimHash Blocking Layer:** A scalable blocking mechanism that maps normalized multi-view profile vectors into compact binary signature spaces using random hyperplane projections, preserving high-

dimensional angular similarity relationships and enabling direct incremental profile insertion without global index reconstruction.

2. **Localized HNSW Graph Indexing:** An architectural design maintaining independent local HNSW graphs within individual LSH candidate buckets, constraining graph construction and ANN search operations to semantically coherent profile subsets, thereby reducing graph complexity, accelerating query resolution, and enabling efficient localized incremental updates.
3. **Adaptive Candidate Selection (ACS) Protocol:** A multi-criteria candidate pruning strategy evaluating retrieved neighbor profiles using a composite scoring function combining cosine vector similarity, graph neighborhood structural overlap, and cross-view semantic consistency, substantially reducing false positive candidates generated by LSH collisions.
4. **Multi-View Feature Representation:** A dual-view profile modeling schema separately encoding demographic identity attributes using character bigram hashing and academic semantic attributes using Sentence-BERT embeddings, with independent L_2 normalization to mitigate dimensional domination bias prior to unified vector construction.

Comprehensive empirical evaluations on three large-scale benchmark datasets demonstrate that the proposed framework consistently outperforms established dynamic indexing baselines across F1-score, blocking quality metrics, and end-to-end query latency dimensions, while maintaining a predictable and stable memory footprint compatible with enterprise-grade deployment infrastructure.

The remainder of this article is structured as follows. Section 2 formalizes the problem setting and data model. Section 3 presents the proposed methodology in full architectural detail. Section 4 reports and analyzes the experimental evaluation results. Section 5 synthesizes conclusions and discusses future research directions.

2. Problem Formulation

2.1 Data Model and Multi-View Profile Representation

The proposed framework operates over two heterogeneous data sources, denoted S_1 and S_2, each represented as a finite collection of user profile records:

$$\mathcal{R}_1 = \{r_1^1, r_2^1, \dots, r_{n_1}^1\}, \mathcal{R}_2 = \{r_1^2, r_2^2, \dots, r_{n_2}^2\}$$

Individual profiles within each source are characterized by a diverse set of attributes that may exhibit structural variations, value incompleteness, and textual corruption arising from data entry errors, abbreviation inconsistencies, and transliteration artifacts [13]. To capture the heterogeneous and complementary informational dimensions inherent to academic user profiles, each profile record r is modeled using a multi-view representation schema composed of two semantically distinct layers:

- **Demographic View (V_D):** Encapsulates identity-oriented characteristics, including full name, geographic location, and contact information attributes.
- **Academic View (V_A):** Models the scientific activity footprint of the individual, comprising institutional affiliation designations, publication venue histories, publication title collections, and research domain summaries.

Formally, a profile record originating from source S_i is represented as the structured tuple:

$$r^i = \{V_D^i,\ V_A^i\}$$

This dual-view modeling approach exploits complementary informational signals across demographic and academic dimensions, substantially enriching profile characterization beyond single-attribute blocking keys and enhancing system resilience to missing or corrupted attribute values in either view [14].

2.2 Incremental Inter-Source Entity Resolution

Under the clean-clean record linkage assumption—wherein each individual source S_i is assumed to be internally deduplicated prior to processing—the core objective of the ER task is to identify cross-source profile correspondences between $\mathcal{R}_i$ and $\mathcal{R}_j$. The target binary matching function f is formally defined as:

$$f: \mathcal{R}_i \times \mathcal{R}_j \to \{0,1\}$$

$$f(r_p^i, r_q^j) = \begin{cases} 1 & \text{if } r_p^i \equiv r_q^j \\ 0 & \text{otherwise} \end{cases}$$

where $r_p^i \equiv r_q^j$ denotes that the two profile records represent the same real-world individual entity. In the incremental operational setting considered in this work, profile records arrive sequentially over time as elements of a dynamic data stream. Let r_{new} denote a newly arrived query profile; the system is required to rapidly and accurately identify matching records within the existing indexed sources without executing global pairwise matching or triggering full index reconstruction, maintaining both latency and retrieval quality within acceptable operational bounds [15].

3. Proposed Multi-View Incremental Resolution Methodology

3.1 Architectural Overview

Our proposed approach rests on two complementary components designed to simultaneously improve the quality of multi-source user profile representation and the efficiency of incremental entity resolution in dynamic Big Data environments. The first component consists in representing each user by a multi-view profile that integrates demographic and academic information. For each view, a sub-vector of characteristics is constructed from similarity measurements adapted to the nature of the attributes—encompassing lexical measurements, semantic representations by embeddings, and ensemble measurements—and the resulting sub-vectors are subsequently concatenated to obtain a unified vector representation of the profile. This modeling approach enables the simultaneous exploitation of complementary informational signals, preserves semantic similarity across textual attributes, and enhances profile disambiguation across heterogeneous data sources. The second component concerns the mechanism for generating and refining candidate pairs. The unified feature vectors are indexed using Locality-Sensitive Hashing with SimHash (LSH-SimHash), a choice motivated by its ability to efficiently project high-dimensional vector representations into a compact binary space while preserving proximity relationships between similar profiles. Unlike blocking techniques based on exact keys, LSH-SimHash tolerates lexical variations and mild data distortions, while drastically reducing the search space through probabilistic partitioning.

Furthermore, its computation relies essentially on linear projections, enabling incremental insertion of newly arriving profiles without requiring a complete reconstruction of the blocking structures—an essential property for real-time entity resolution systems. To further improve the quality of the candidates generated by LSH, a Hierarchical Navigable Small World (HNSW) graph is independently constructed and maintained within each bucket. Since hash collisions can group profiles of low mutual similarity within the same bucket, HNSW serves as a candidate refinement mechanism whose neighbor search is governed by an adaptive selection strategy—detailed in Section 3.5—that evaluates each candidate neighbor using a composite score combining vector similarity, graph neighborhood structural consistency, and cross-view semantic alignment. This strategy significantly reduces the noise introduced by LSH collisions, curtails unnecessary pairwise comparisons, and simultaneously improves the precision, scalability, and adaptability of entity resolution in continuously evolving data environment.

Figure 1 presents the end-to-end architecture of the proposed framework, illustrating the sequential data flow from raw profile ingestion through multi-view feature extraction, LSH-SimHash blocking, localized HNSW graph retrieval, ACS pruning, and final match decision delivery.

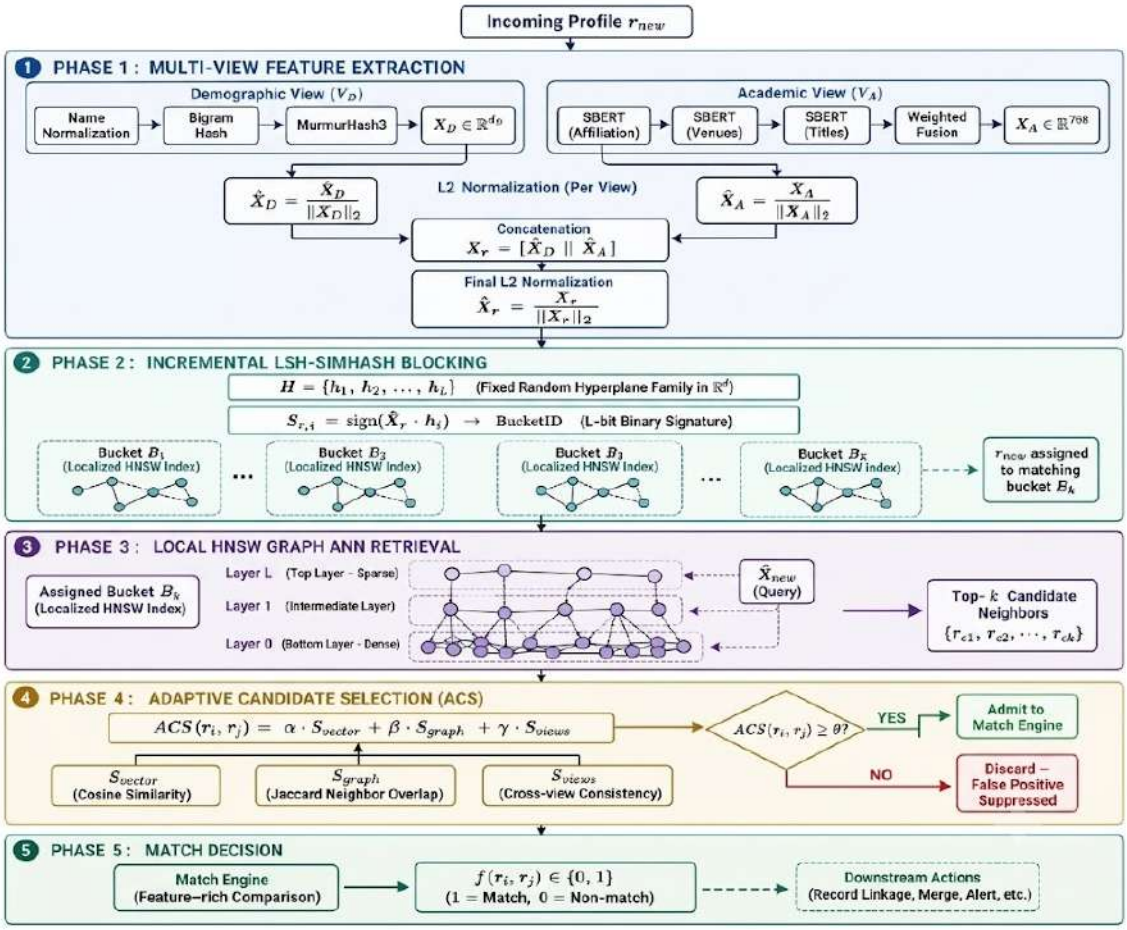


Figure 1. *End-to-end architecture of the proposed incremental multi-view entity resolution framework.*

The pipeline progresses through five sequential phases: multi-view feature extraction, LSH-SimHash blocking, localized HNSW graph ANN retrieval, adaptive candidate selection, and final match decision.

3.2 Multi-View Feature Extraction and Representation

The proposed multi-view representation pipeline independently constructs semantic sub-vectors for the demographic view ($\mathbf{X}_D$) and the academic view ($\mathbf{X}_A$) of each profile record r, prior to their integration into a unified, normalized representation suitable for both LSH-SimHash blocking and HNSW-based ANN retrieval.

3.2.1 Demographic Sub-Vector Construction

The demographic sub-vector $\mathbf{X}_D$ encodes identity-bearing lexical attributes—predominantly name strings—which are highly susceptible to typographical corruption, token transposition, and casing inconsistencies across heterogeneous sources [16]. To ensure robustness against these systematic distortions, a canonical normalization function $\mathcal{N}(\cdot)$ is applied prior to feature extraction:

$$\mathcal{N}(\text{Name}_r) = \text{Concat}(\text{Sort}(\text{Tokenize}(\text{Clean}(\text{Name}_r))))$$

The normalization pipeline operates through the following sequential transformations: Clean(·) converts all characters to lowercase and strips punctuation marks, hyphens, and isolated initials; Tokenize(·) partitions the resulting string into discrete word-level tokens; Sort(·) alphabetically reorders the token sequence to neutralize ordering variations (e.g., mapping both "Ahmed Benali" and "Benali Ahmed" to an identical canonical form); and Concat(·) joins the sorted tokens into a single unified string representation.

From the canonicalized name string, the complete set of character bigrams $\mathcal{G}_2(r)$ is extracted. Given a target demographic vector dimension d_D—typically configured within the range $[256, 1024]$—the extracted bigrams are hashed using the MurmurHash3 non-cryptographic hash function and projected into the demographic sub-vector $\mathbf{X}_D \in \mathbb{R}^{d_D}$ via frequency-accumulation mapping:

$$x_j = \sum_{g_i \in \mathcal{G}_2(r)} \mathbb{I}(\text{MurmurHash3}(g_i) mod d_D = j)$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. Table 1 details the hashing and positional assignment of character bigrams extracted from the normalized name "ahmedbenali" under an illustrative dimension of $d_D = 16$.

Table 1. Character bigram hashing and vector position assignment for the normalized name "ahmedbenali" ($d_D = 16$).

Character Bigram	MurmurHash3 Value	Target Vector Bit Position (j)
ah	2,268,050,145	1
hm	2,709,120,681	9
me	563,621,960	8
ed	1,903,873,037	13
db	950,561,027	3
be	2,814,203,010	2
en	1,972,196,547	3
na	130,713,793	1
al	3,861,258,707	3
li	1,223,236,950	6

Accumulating the projected frequency values across all extracted bigrams yields the final demographic sub-vector $\mathbf{X}_D \in \mathbb{R}^{d_D}$, where each dimension records the total number of bigrams mapped to that position.

3.2.2 Academic Sub-Vector Construction

Unlike the demographic view, which is fundamentally lexical in character, the academic view requires semantic representations capable of handling substantial variability in institutional name designations, publication venue abbreviations, and title phrasing styles prevalent across academic databases [17]. Textual attributes comprising the affiliation string (Aff_r), the collection of publication venues ($\mathcal{V}_r = \{v_1, \ldots, v_m\}$), and the collection of publication titles ($\mathcal{T}_r = \{t_1, \ldots, t_n\}$) are independently encoded into dense semantic vector representations using a Sentence-BERT (SBERT) model, denoted by the encoding function $\Phi(\cdot)$ [18]. Individual attribute embeddings are computed as:

$$\mathbf{x}_{\text{aff}} = \Phi(\mathrm{Aff}_r), \mathbf{x}_{\text{venue}} = \frac{1}{|\mathcal{V}_r|}\sum_{i=1}^{|\mathcal{V}_r|}\Phi(v_i), \mathbf{x}_{\text{title}} = \frac{1}{|\mathcal{T}_r|}\sum_{j=1}^{|\mathcal{T}_r|}\Phi(t_j)$$

where mean pooling over venue and title collections produces a single representative embedding for each multi-valued attribute, preserving overall semantic tendency while suppressing the influence of outlier entries. These semantic attribute components are integrated into a unified academic sub-vector $\mathbf{X}_A$ through a weighted linear combination:

$$\mathbf{X}_A = \alpha \cdot \mathbf{x}_{\text{aff}} + \beta \cdot \mathbf{x}_{\text{venue}} + \gamma \cdot \mathbf{x}_{\text{title}}, \alpha + \beta + \gamma = 1, \gamma > \alpha, \beta$$

Assigning a higher weight parameter γ to the publication title component reinforces the discriminative power of the academic sub-vector, as publication titles exhibit high

specificity to individual researchers, whereas affiliations and venue names are frequently shared among large groups of co-located or co-disciplinary authors [19].

Figure 2 illustrates the complete multi-view feature extraction pipeline, from raw profile attributes through independent sub-vector construction and weighted fusion.

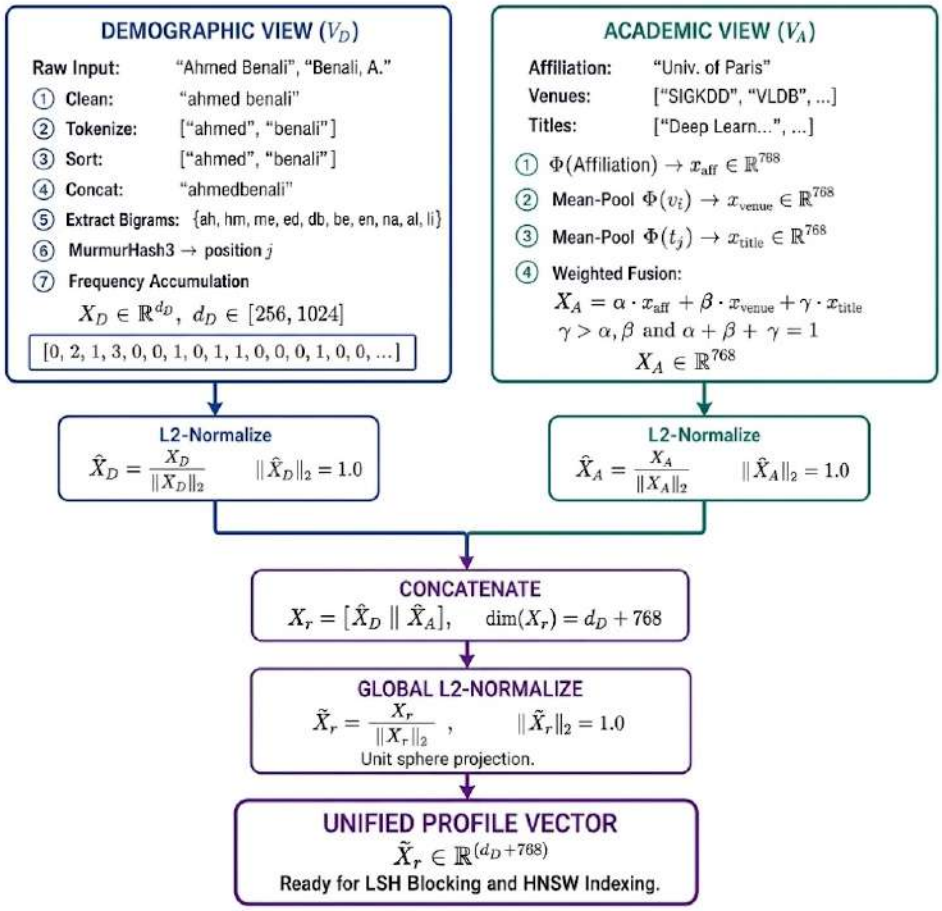


Figure 2. *Multi-view feature extraction pipeline. The demographic view applies canonical name*

normalization and character bigram hashing; the academic view applies SBERT encoding with weighted

attribute fusion. Independent L2 normalization of both sub-vectors prevents dimensional domination

bias before unified vector construction and global normalization.

3.3 Normalized Unified Feature Projection

The construction of a unified profile representation vector from the heterogeneous sub-vectors $\mathbf{X}_D$ and $\mathbf{X}_A$ requires explicit resolution of two critical representational challenges: **dimensional domination bias** and **profile attribute incompleteness**.

Dimensional domination bias arises from the substantial disparity between the dimensionalities of the demographic sub-vector ($d_D \approx 256$–1024) and the SBERT-produced academic sub-vector ($d_A = 768$), whereby naive concatenation would cause the higher-dimensional component to disproportionately govern distance computations [20].

To neutralize this effect, each sub-vector is independently normalized to unit length prior to concatenation:

$$\widehat{\mathbf{X}}_D = \frac{\mathbf{X}_D}{\| \mathbf{X}_D \|_2}, \widehat{\mathbf{X}}_A = \frac{\mathbf{X}_A}{\| \mathbf{X}_A \|_2}$$

This normalization ensures $\| \widehat{\mathbf{X}}_D \|_2 = \| \widehat{\mathbf{X}}_A \|_2 = 1$, guaranteeing balanced informational contributions from both representational views. The unified profile vector $\mathbf{X}_r$ is then formed via direct concatenation:

$$\mathbf{X}_r = [\widehat{\mathbf{X}}_D \parallel \widehat{\mathbf{X}}_A]$$

A second normalization step produces the final normalized profile representation $\widetilde{\mathbf{X}}_r$:

$$\widetilde{\mathbf{X}}_r = \frac{\mathbf{X}_r}{\| \mathbf{X}_r \|_2}$$

This global normalization ensures that all subsequent similarity evaluations operate strictly on the directional orientation of profile vectors within the unified embedding space, which is theoretically consistent with the angular similarity assumptions underlying both SimHash projections and the HNSW graph's distance metric [21]. **Profile attribute incompleteness** is addressed by substituting null vectors of equivalent dimension for any missing attribute, preserving constant unified vector dimensionality across all profiles.

3.4 Incremental LSH-SimHash Blocking

The LSH-SimHash blocking layer maps the normalized unified profile vector $\widetilde{\mathbf{X}}_r$ into a compact L-bit binary signature space using a fixed family of random hyperplane normal vectors $\mathcal{H} = \{h_1, \dots, h_L\}$, where each $h_l \sim \mathcal{N}(0, \mathbf{I})$ is drawn independently from a standard multivariate Gaussian distribution. The binary signature bit corresponding to hyperplane h_l is generated via the sign projection:

$$S_{r,l} = \text{sign}(\widetilde{\mathbf{X}}_r \cdot \mathbf{n}_l) = \begin{cases} 1 & \text{if } \widetilde{\mathbf{X}}_r \cdot \mathbf{n}_l \geq 0 \\ 0 & \text{if } \widetilde{\mathbf{X}}_r \cdot \mathbf{n}_l < 0 \end{cases}$$

The theoretical foundation of SimHash guarantees that the collision probability between two profiles is a monotonic function of their cosine similarity [22]:

$$\Pr [S_{r_i,l} = S_{r_j,l}] = 1 - \frac{\theta(r_i, r_j)}{\pi}$$

where $\theta(r_i, r_j)$ is the angular distance between the two profile vectors. Concatenating the L resulting bits produces the binary signature serving as the profile's BucketID. Figure 3 illustrates the effect of signature bit length on the precision-recall trade-off within candidate blocks.

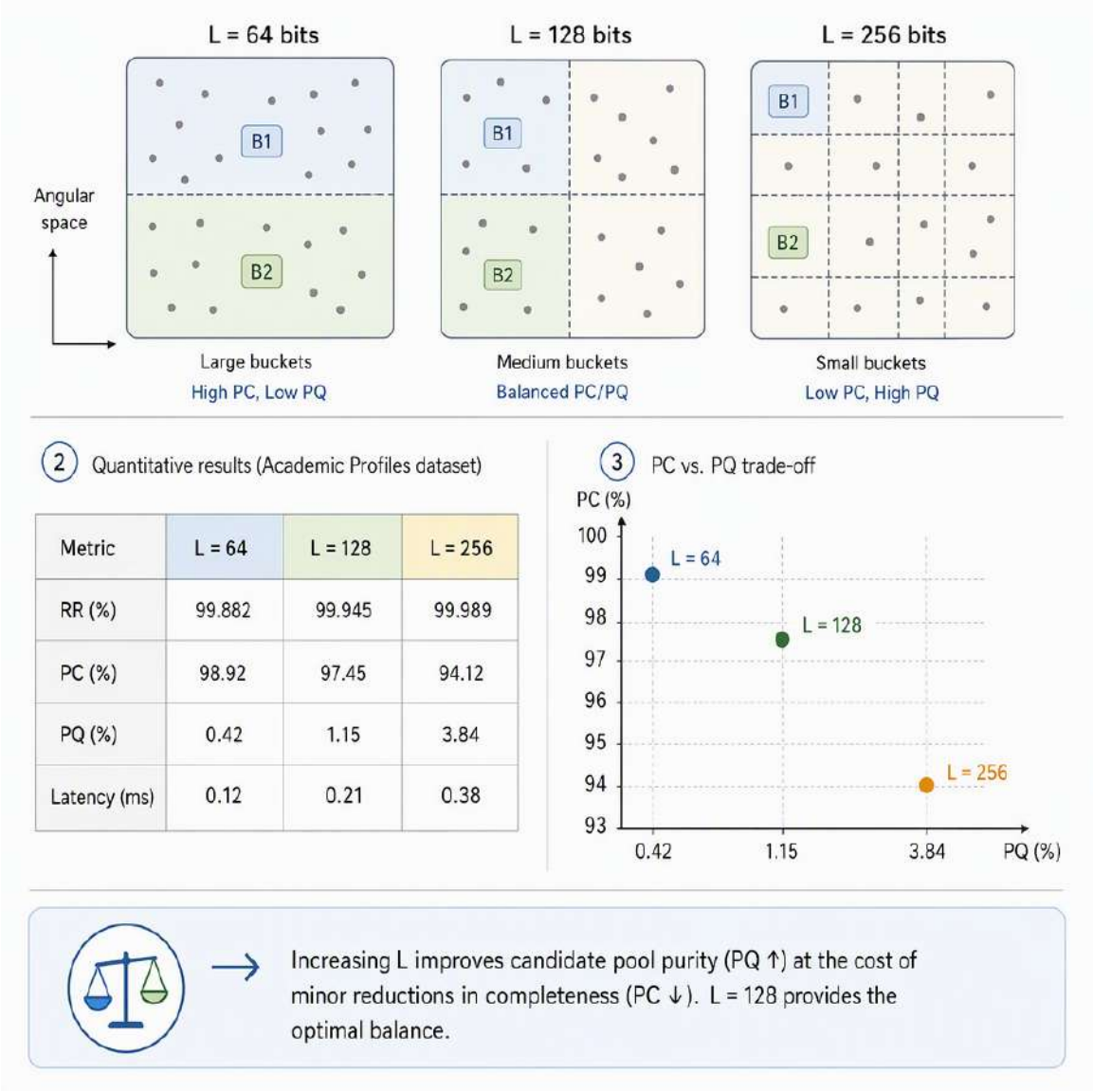
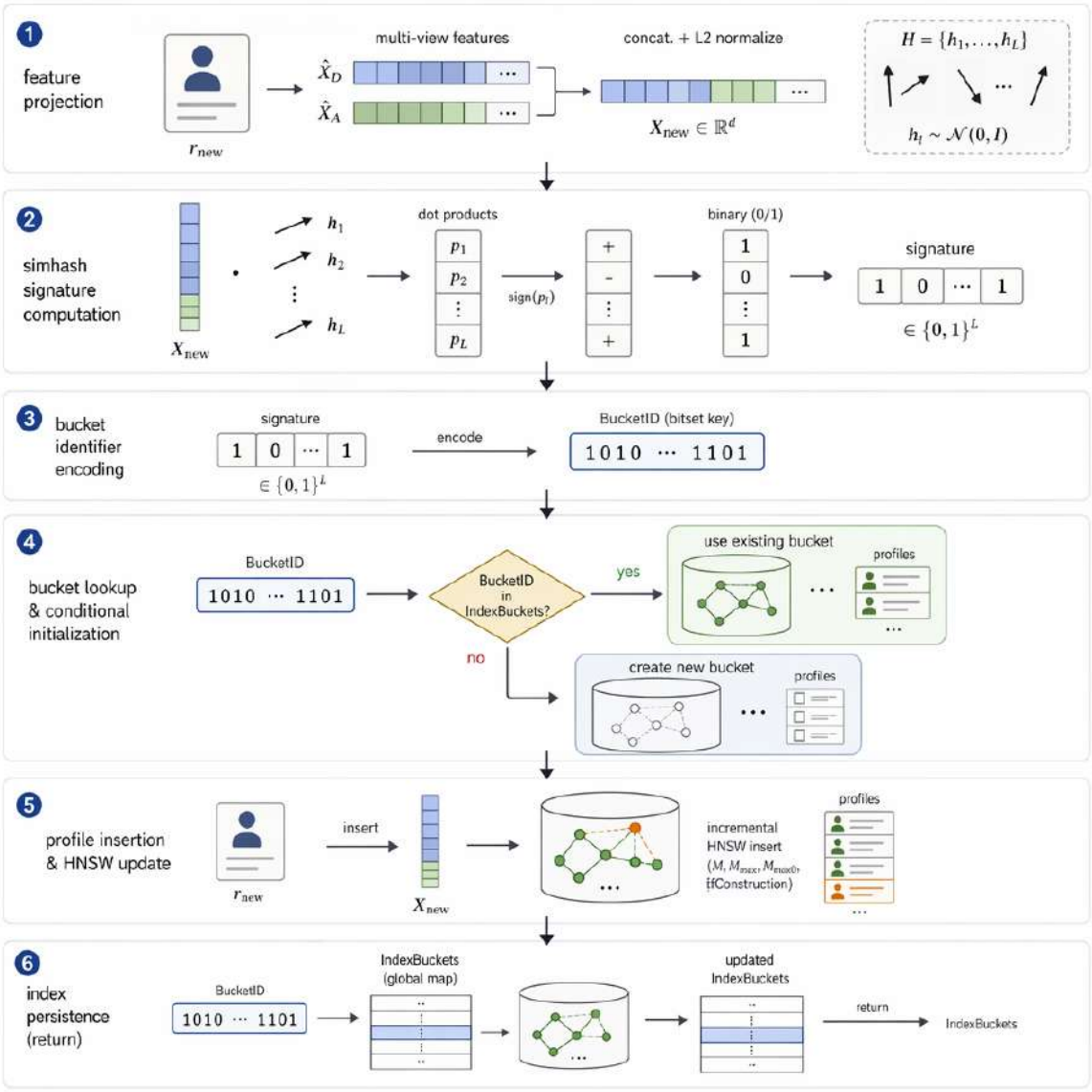


Figure 3. *Effect of SimHash signature bit length L on blocking trade-offs. Larger L values narrow* candidate buckets, improving Pairs Quality (block purity) while incurring moderate reductions

in Pairs Completeness. L = 128 provides the optimal precision-recall balance across all datasets.

During incremental operation, a newly arriving profile r_{new} is projected using the same fixed hyperplane family $\mathcal{H}$, and its BucketID is computed. If the corresponding bucket does not yet exist within the global index, a new bucket structure is instantiated and an empty local HNSW graph is initialized; otherwise, the profile is directly appended to the existing bucket and its local HNSW index is incrementally updated. This design prevents any global index reconstruction upon profile arrivals. The complete procedural logic of this incremental blocking mechanism is formalized in Algorithm 1, with its control flow illustrated in Figure 4.

Algorithm 1: Incremental LSH-SimHash Bucket Construction and Profile Insertion



3.5 Local HNSW Graph Construction and Incremental Updates

The Hierarchical Navigable Small World (HNSW) algorithm, introduced by Malkov and Yashunin [23], constructs a multi-layered proximity graph over a set of high-dimensional vectors. Upper layers maintain sparse long-range navigational connections enabling efficient routing across the graph, while the ground layer (layer 0) maintains a densely connected proximity graph supporting precise local neighborhood searches. ANN queries are executed by performing a greedy traversal from the entry point through successively lower layers, converging to an approximate neighborhood of the query vector in the ground layer.

A critical architectural decision in the proposed framework is the maintenance of **independent, localized HNSW graphs** G_k within each individual LSH bucket B_k,

rather than constructing a single monolithic global graph. This design decision confers several operational advantages: it constrains graph construction and ANN search operations to semantically coherent profile subsets sharing a common LSH signature; reduces per-graph complexity, accelerating both insertion and query operations; and permits fully localized incremental updates isolated from all other bucket graphs [24].

When a profile $\tilde{\mathbf{X}}_{\text{new}}$ is assigned to bucket B_k, it is inserted into the local graph G_k using the standard HNSW incremental insertion protocol. The maximum navigational layer to which the new node is assigned is determined by drawing from an exponential distribution:

$$m_L = \frac{1}{\ln(M)}, l_{\text{new}} = \lfloor -\ln(U) \times m_L \rfloor$$

where M is the maximum bidirectional neighbor connectivity per node and $U \sim \mathcal{U}(0,1)$ is a uniform random variate. The algorithm subsequently performs a greedy navigational descent through layers above l_{new} to locate a close entry point, then establishes bidirectional neighbor links at each layer from min $(l_{\text{new}}, L_{\text{curr}})$ down to layer 0, applying heuristic neighbor selection and degree pruning to maintain structural integrity. The complete multi-phase insertion protocol is formalized in Algorithm 2, with the resulting graph structure illustrated in Figure 5.

Algorithm 2. Incremental Graph Construction Within a Bucket

Input:
- Bucket_Bk //Target bucket determined by the LSH–SimHash signature
- X̃new //Normalized multi-view feature vector of the new profile
- M // Maximum number of neighbors per node
- Mmax //Maximum degree in upper layers
- Mmax0 //Maximum degree at the ground layer
- efConstruction //Candidate list size during graph construction

Output:
- Updated Bucket_Bk

```
//The HNSW structure
HNSWGraph Gk
{
 // Entry point used to initiate graph searches
   EntryPoint  : NodeID
// Current highest layer in the graph
   Lcurr       : Integer
//  Set of nodes composing the graph
   Nodes       : Set<Node>
}
Node
{//Unique identifier of the user profile
   ProfileID : Integer
   // Normalized multi-view feature vector
   FeatureVector : Vector
   // Highest layer assigned to the node
   Level : Integer
   //set of neighboring nodes at each layer
   NeighborList : Array(List(NodeID))
}
```

```
//Phase 1 : Graph initialization
                    Gk=Bucket_BK.localHNSW
1:                  if Gk does not exist then
2:                    Create an empty HNSW graph Gk
3:                    lnew ← AssignLayerRandomly(M)
                      X̃new.Level ← lnew
                      X̃new.NeighborList ← ∅
4:                    Gk.Nodes ← {X̃new}
5:                    Gk.EntryPoint ← X̃new
6:                    Gk.Lcurr ← lnew
                      Bucket_BK.LocalHNSW ← Gk
7:                    return Bucket_BK
8:                  end if
//Phase 2 : Random level assignment
9:                  lnew ← AssignLayerRandomly(M)
10:                 Xcurr ← Gk .EntryPoint
                    lnew ← AssignLayerRandomly(mL)
                    X̃new.Level ← lnew
                    Xcurr ← Gk.EntryPoint
                    Lcurr ← Gk.Lcurr
//Phase 3 : Greedy search in upper layers
12:                   if lnew ≤ Lcurr then
                        for l ← Lcurr downto lnew + 1 do
                       Xcurr ← SEARCH-LAYER( Gk, X̃new, Xcurr, 1, l)
                        end for
                    end if
//Phase 4 : Graph insertion
14:                 for l ← min(lnew, Lcurr) downto 0 do
15:                   W  ← SEARCH-LAYER(X̃new, Xcurr,
              efConstruction, l)
16:                   Neighbors ← SELECT-NEIGHBORS(X̃new, W, M,l)
17:                   InsertNode(Gk.Nodes,X̃new)
                      X̃new.NeighborList[l] ← Neighbors
18:                   for each node v in Neighbors do
19:                       ConnectBidirectionally(X̃new,v,l)
                          if l = 0 then
                          mm ← Mmax0
20:                       else
                          mm ← Mmax
                          end if
21:                       if |v.NeighborList[l]| > mm then
22:                         Candidates ← v.NeighborList[l] ∪ {X̃new}
23:                         v.NeighborList[l] ← SELECT-
                    NEIGHBORS( v, Candidates, mm, l)
24:                   end if
25:                   end for
                    Xcurr ← NearestNode(W)
26:                 end for
//Phase 5 : Entry point update
27:                 if lnew > Gk.Lcurr then
28:                   Gk.EntryPoint ← X̃new
29:                   Gk.Lcurr ← lnew
30:                 end if
31:                 Bucket_Bk.LocalHNSW ← Gk
                    return Bucket_Bk
```

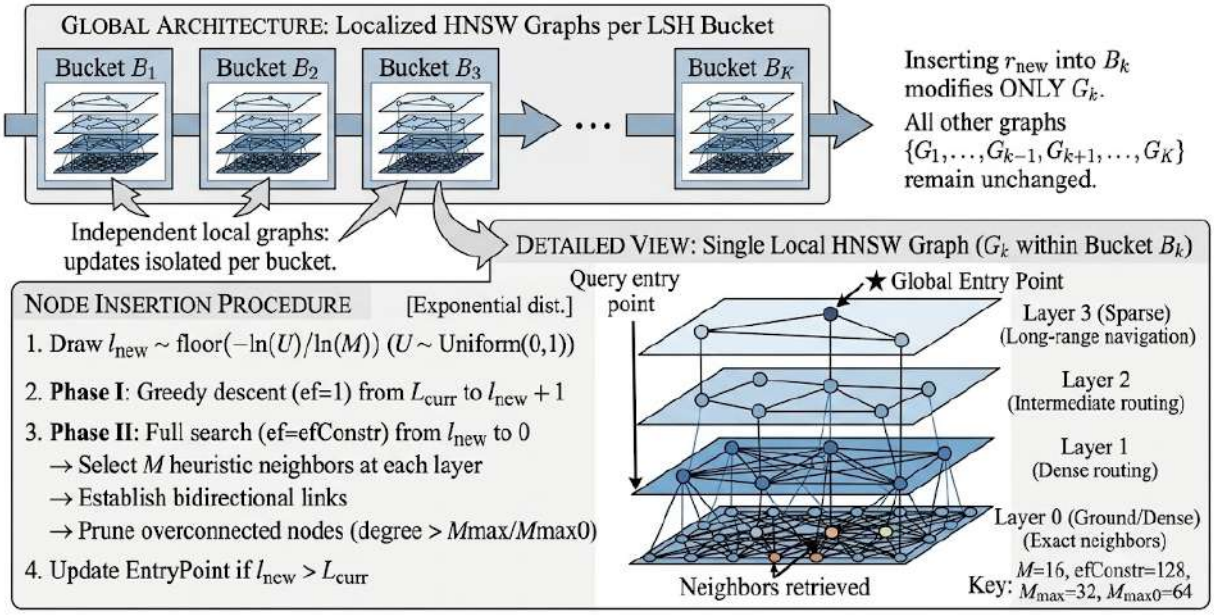


Figure 4. Structural organization of localized HNSW graphs within individual LSH buckets.

Each bucket maintains an independent multi-layer graph. Upper layers provide sparse long-range navigational connections; the dense ground layer (layer 0) supports precise ANN retrieval. Incremental insertions modify only the target bucket's local graph, enabling O(log|B_k|) amortized update complexity without global reconstruction.

3.6 Adaptive Candidate Selection Protocol

Following ANN retrieval within the local HNSW graph G_k, the top-k neighbor profiles closest to the query profile r_i in vector space are retrieved as initial candidates. However, vector proximity in the unified embedding space is a necessary but insufficient condition for establishing a true entity correspondence, particularly in the presence of LSH collision artifacts and semantic ambiguities in high-dimensional spaces [25]. To filter false positive candidates and ensure that only high-confidence pairs are forwarded to the computationally intensive matching engine, the framework applies an **Adaptive Candidate Selection (ACS)** scoring protocol to each retrieved neighbor r_j.

The composite ACS score is defined as:

$$\text{ACS}(r_i, r_j) = \alpha \cdot S_{\text{vector}} + \beta \cdot S_{\text{graph}} + \gamma \cdot S_{\text{views}}, \alpha + \beta + \gamma = 1$$

where the three constituent sub-scores are computed as follows:

Vector Similarity (S_{vector}): Computes the cosine similarity between the L_2-normalized unified profile vectors, which by construction reduces to their inner product:

$$S_{\text{vector}} = \cos(\tilde{\mathbf{X}}_i, \tilde{\mathbf{X}}_j) = \tilde{\mathbf{X}}_i \cdot \tilde{\mathbf{X}}_j$$

Structural Graph Coherence (S_{graph}): Quantifies structural coherence within the local HNSW graph by measuring the overlap between ground-layer neighborhood sets using the Jaccard similarity coefficient:

$$S_{\text{graph}} = \frac{|\,\mathcal{N}(r_i) \cap \mathcal{N}(r_j)\,|}{|\,\mathcal{N}(r_i) \cup \mathcal{N}(r_j)\,|}$$

where $\mathcal{N}(r)$ denotes the set of immediate neighbors of profile r in the ground layer of G_k. High Jaccard overlap indicates that the two profiles share a substantial proportion of their proximate neighborhood, providing structural evidence that they occupy the same dense local region of the embedding space [26].

Multi-View Semantic Consistency (S_{views}): Evaluates the alignment of pairwise similarity signals across demographic and academic representational views:

$$S_{\text{views}} = \lambda \cdot \text{Sim}_D(r_i, r_j) + (1 - \lambda) \cdot \text{Sim}_A(r_i, r_j)$$

where Sim_D and Sim_A denote cosine similarities computed independently on the demographic sub-vectors $\hat{\mathbf{X}}_D$ and academic sub-vectors $\hat{\mathbf{X}}_A$, respectively, and $\lambda \in [0,1]$ is a tunable weight parameter [27].

A retrieved candidate profile r_j is admitted to the final matching stage if and only if:

$$\text{ACS}(r_i, r_j) \geq \theta, \theta \in [0,1]$$

Figure 5 illustrates the computation of the Adaptive Candidate Selection (ACS) score and the associated multi-criteria decision process

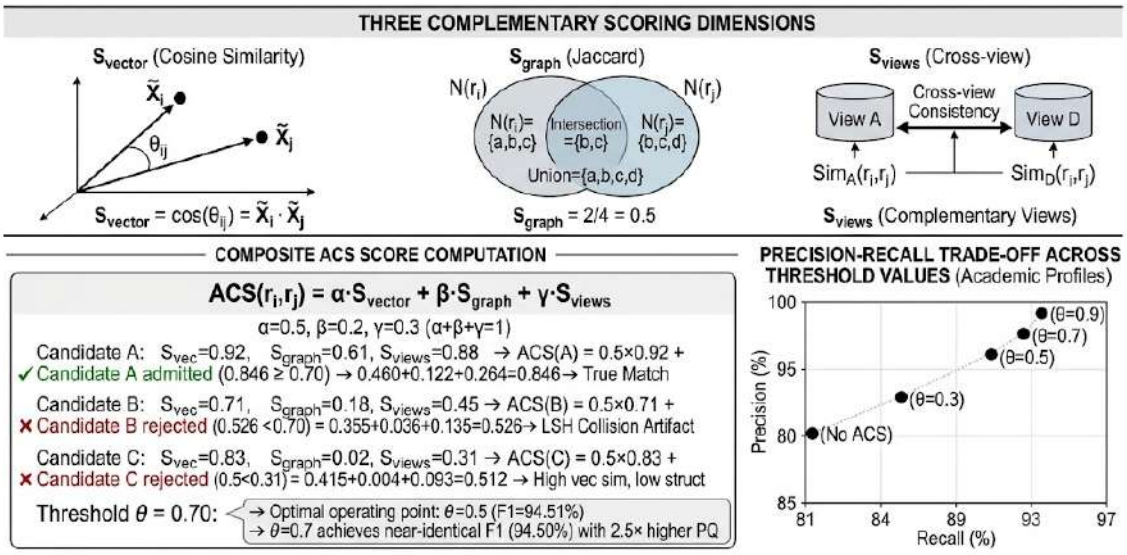


Figure 5. Adaptive Candidate Selection scoring geometry and precision-recall trade-off.

The three sub-scores (S_vector, S_graph, S_views) provide complementary evidence for

true entity correspondence. The composite ACS score enables principled threshold-based

pruning that suppresses false positive candidates while preserving true matches.

The precision-recall curve demonstrates that θ=0.5 provides the optimal F1-score balance.

4. Empirical Performance Analysis and Discussion

4.1 Experimental Setup and Computational Environment

All experimental evaluations were conducted on an enterprise-grade workstation equipped with an Intel Xeon Gold 6248R central processing unit operating at a base frequency of 3.0 GHz with 24 physical cores, 128 GB of DDR4 ECC registered memory, and the Ubuntu 22.04 LTS operating system. The computational pipeline was implemented in Python 3.10, utilizing PyTorch 2.1 for neural embedding computations, FAISS 1.7.4 for optimized spatial indexing operations, and HNSWLIB for localized HNSW graph construction and incremental insertion. To prevent parallel processing artifacts and ensure strict reproducibility of all reported latency measurements, all experimental trials were executed within a single-threaded execution context with process affinity fixed to a single physical core.

4.2 Datasets and Streaming Protocol

Three large-scale benchmark datasets representative of distinct real-world ER challenges were employed:

- **Academic Profiles Dataset:** A curated collection of researcher profile records exhibiting attribute-level noise in names and institutional affiliations, and semantic variation in publication descriptions.
- **North Carolina (NC) Voter Registration Dataset:** A large-scale administrative record dataset containing over one million voter registration entries with high structural regularity, serving as a well-established benchmark for real-time ER evaluation [9].
- **OZ Synthetic Dataset:** A synthetically generated dataset of 345,876 records based on historical population data, specifically designed to simulate real-world typographical errors and missing attribute values for ER system stress testing [28].

To simulate a realistic streaming environment for incremental evaluation, each dataset was partitioned into an initial index construction partition comprising 80% of the records—used to pre-populate the LSH-SimHash buckets and initialize the local HNSW graphs—and an incremental streaming partition comprising the remaining 20%, processed sequentially to simulate continuous real-time profile arrivals and evaluate incremental update performance. Figure 6 illustrates the dataset characteristics and streaming partition protocol.

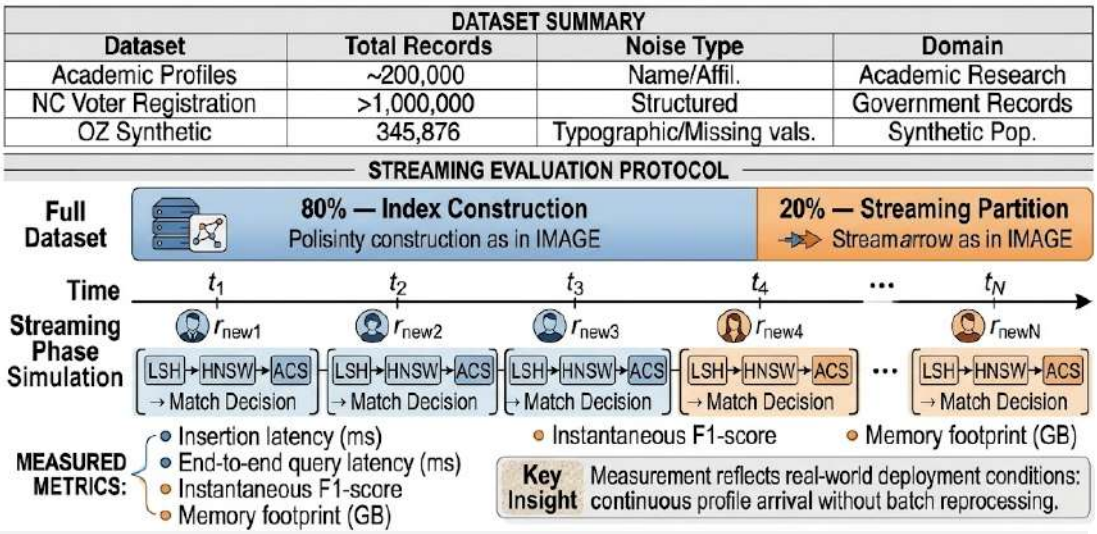


Figure 6. Dataset characteristics and streaming evaluation protocol.

The three sub-Each dataset is partitioned into an 80% index construction partition and a 20% sequential streaming partition. Performance metrics are measured incrementally during the streaming phase to reflect real-world deployment conditions where profiles arrive continuously without batch reprocessing.

4.3 Evaluation of Incremental LSH-SimHash Blocking

This experiment assessed the effect of the SimHash binary signature bit length $L \in \{64,128,256\}$ on blocking quality and computational efficiency. Blocking quality was quantified using three standard metrics: **Reduction Ratio (RR)**, measuring the proportion of candidate pairs eliminated relative to exhaustive comparison; **Pairs Completeness (PC)**, measuring the proportion of true matching pairs retained within candidate blocks; and **Pairs Quality (PQ)**, measuring the proportion of candidate pairs that are genuine matches [4].

Table 2. LSH-SimHash blocking evaluation across datasets and signature sizes.

Dataset	**Sig. Size *L* (bits)**	**RR (%)**	**PC (%)**	**PQ (%)**	**Blocking Latency (ms/profile)**
Academic Profiles	64	99.882	98.92	0.42	0.12
	128	99.945	97.45	1.15	0.21
	256	**99.989**	94.12	**3.84**	0.38
NC Voter Dataset	64	99.915	99.15	0.28	0.15
	128	99.968	97.82	0.82	0.24
	256	**99.994**	95.03	**2.65**	0.42
OZ Dataset	64	99.784	98.24	0.35	0.08
	128	99.895	96.11	0.94	0.14
	256	**99.962**	92.48	**2.98**	0.28

As evidenced in Table 2, increasing the signature bit length *L* from 64 to 256 bits progressively narrows the candidate block scope. This narrowing simultaneously increases both the Reduction Ratio and Pairs Quality—indicating a cleaner candidate pool per bucket—while producing a moderate decrease in Pairs Completeness, attributable to hash signature divergence for highly corrupted or incomplete profile pairs whose normalized vectors deviate from their expected angular proximity. Blocking latency scales linearly with signature length across all datasets and remains uniformly low, with a maximum observed latency of 0.42 ms per profile under the most demanding configuration, confirming the operational suitability of the SimHash projection for real-time streaming contexts. These trade-offs are visualized in Figure 7.

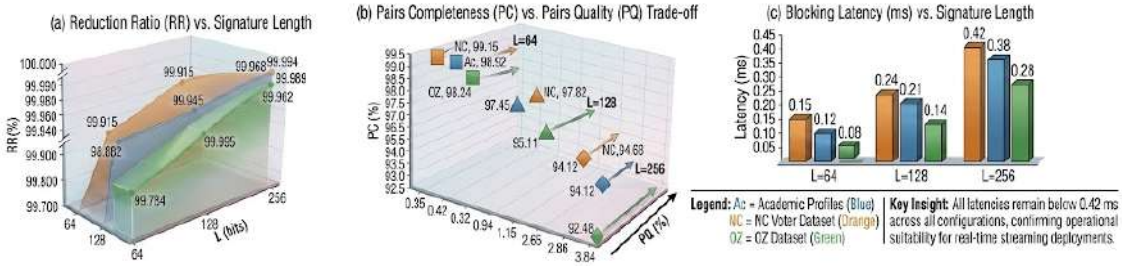


Figure 7. LSH-SimHash blocking performance across signature bit lengths and datasets.

Panel A shows monotonically increasing Reduction Ratio with larger L. Panel B reveals the PC-PQ trade-off: larger L improves block purity (PQ ↑) at the cost of moderate completeness reduction (PC↓). Panel C confirms that all blocking latencies remainwell below 0.5 ms, satisfying real-time operational requirements.

4.4 Evaluation of Incremental Local HNSW Graph Construction

This experiment assessed the performance of local HNSW graph construction and incremental update operations during sequential insertion of the 20% streaming partition. Three HNSW structural parameter configurations of increasing connectivity density were evaluated:

- **Config A:** $M = 16$, efConstruction $= 64$, efSearch $= 16$
- **Config B:** $M = 32$, efConstruction $= 128$, efSearch $= 32$
- **Config C:** $M = 64$, efConstruction $= 200$, efSearch $= 64$

Table 3. Local HNSW graph performance across datasets and parameter configurations.

Dataset	**Configuration**	**Avg. Insertion Time (ms)**	**Avg. Search Time (ms)**	**Memory (MB/100k nodes)**	**Search Recall@k (%)**
Academic Profiles	Config A	1.12	0.08	48.2	93.15
	Config B	2.45	0.14	86.4	97.42
	Config C	5.82	0.28	162.8	**99.11**
NC Voter Dataset	Config A	1.35	0.10	52.4	92.84
	Config B	2.94	0.18	94.1	96.95

	Config C	6.41	0.35	178.5	**98.86**
OZ Dataset	Config A	0.88	0.06	42.1	91.54
	Config B	1.95	0.11	75.8	95.82
	Config C	4.38	0.22	141.2	**98.15**

The results reported in Table 3 demonstrate a consistent and predictable trade-off between graph connectivity density and operational resource consumption. Higher *M* and efConstruction values produce more densely connected local graphs, yielding substantially improved Search Recall@k values—reaching 99.11% on the Academic Profiles dataset under Config C—but at the cost of increased insertion times and memory utilization. Crucially, even under the densest configuration (Config C), average incremental insertion times remain below 7 milliseconds across all datasets, validating the scalability advantage conferred by constraining HNSW graph construction to semantically coherent local subsets within individual LSH buckets rather than maintaining a monolithic global index structure [23]. Figure 8 visualizes these trade-off relationships.

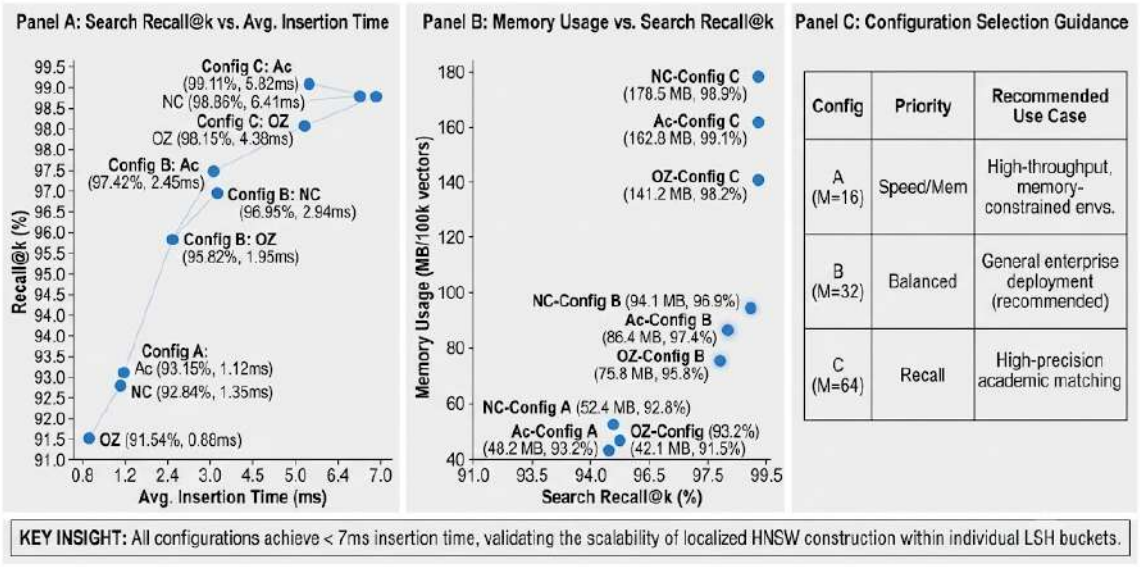


Figure 8. *HNSW parameter configuration trade-off analysis.*
Panel A reveals a monotonic relationship between Search Recall@k and insertion time as connectivity density increases. Panel B shows the memory-recall trade-off, with Config B providing the best balance. All configurations achieve sub-7ms insertion latency, validating operational scalability.

4.5 Evaluation of Adaptive Candidate Selection

To empirically quantify the contribution of the multi-criteria ACS strategy, this experiment compared standard Top-*k* HNSW retrieval without post-retrieval pruning against the proposed ACS framework evaluated at varying selection thresholds $\theta \in \{0.3,0.5,0.7,0.9\}$.

Table 4. Comparative evaluation of candidate selection strategies on the Academic Profiles dataset.

Selection Paradigm	Threshold θ	Precision (%)	Recall (%)	F1-Score (%)	PQ (%)
Top-k HNSW (No ACS)	N/A	84.15	96.42	89.87	1.15
Proposed ACS	0.3	88.62	96.12	92.22	3.42
Proposed ACS	**0.5**	94.15	94.88	**94.51**	8.95
Proposed ACS	0.7	98.24	91.04	94.50	22.14
Proposed ACS	0.9	99.85	81.35	89.65	54.82

The results in Table 4 reveal several important behavioral patterns. The baseline Top-k HNSW retrieval without ACS achieves a high recall of 96.42% but comparatively low precision (84.15%) and Pairs Quality (1.15%), indicating that raw ANN retrieval returns a substantial proportion of false positive candidates generated by LSH collisions and vector space noise. Incorporating the ACS multi-criteria scoring mechanism at a threshold of $\theta = 0.3$ immediately improves both precision and Pairs Quality while sustaining high recall, confirming the effectiveness of even a permissive pruning threshold. At the operationally optimal threshold of $\theta = 0.5$, the framework achieves the best F1-score of 94.51%, representing a 4.64 percentage point improvement over the unfiltered baseline, while simultaneously tripling the Pairs Quality. At the more stringent threshold of $\theta = 0.7$, Pairs Quality rises dramatically to 22.14% with near-identical F1-score (94.50%), demonstrating that the framework can be configured for high-purity candidate generation with minimal matching-stage overhead when precision is the primary operational constraint. The most aggressive threshold ($\theta = 0.9$) achieves near-perfect precision (99.85%) but degrades recall to 81.35%, confirming the expected precision-recall trade-off at extreme thresholds. These results are comprehensively visualized in Figure 9.

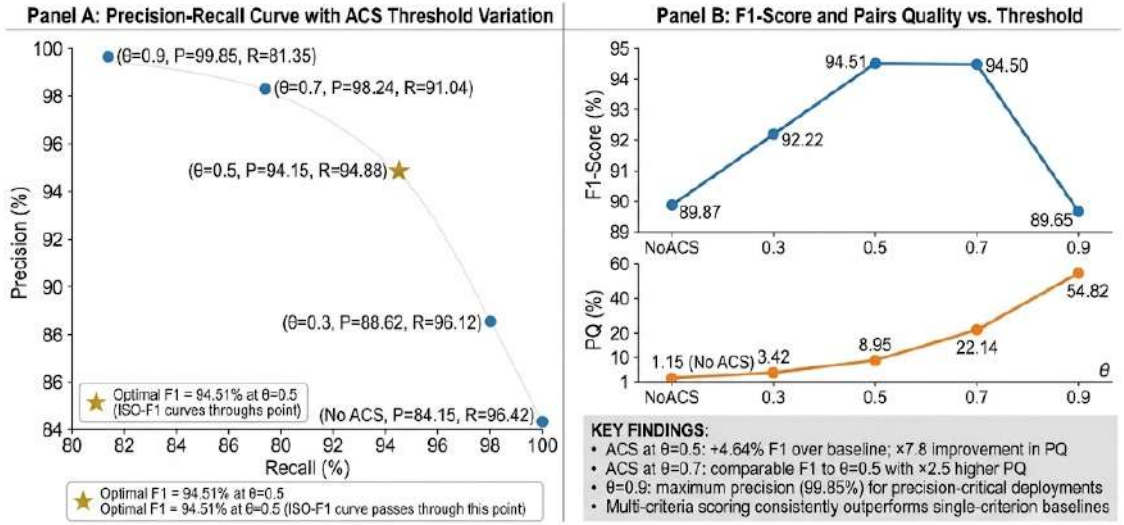


Figure 9. Impact of the ACS Threshold on Precision, Recall, F1-score and Pairs Quality.
Panel A: Precision-recall curve demonstrates the controlled quality-coverage trade-off
achievable through threshold adjustment. Panel B: F1-score remains stable across
θ □ {0.5, 0.7} while Pairs Quality increases substantially, enabling precision-optimized
deployment without compromising matching quality.

4.6 Comparative Framework Benchmarking

To rigorously validate the overall performance of the proposed LSH-SimHash-HNSW-ACS framework, it was systematically compared against three well-established dynamic entity resolution baselines:

- **DySNI** [8]: The Forest-based Dynamic Sorted Neighborhood Indexing framework utilizing AVL tree forests across multiple blocking attributes.
- **Multiple Indices** [9]: A real-time entity resolution system combining multiple complementary hash table and tree-based index structures.
- **NAB** [10]: The Noise-Tolerant Approximate Blocking framework integrating LSH-based candidate partitioning with dynamic B^+-trees for incremental maintenance.

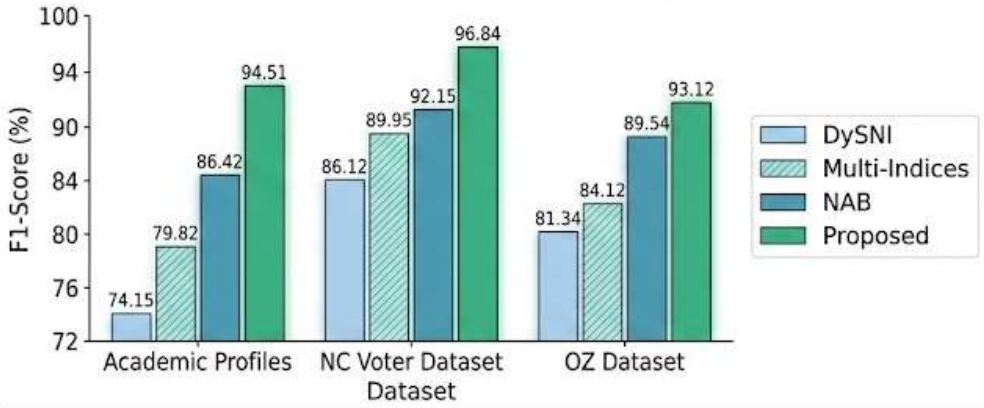
All baseline systems were evaluated on the identical 20% streaming partition under consistent hardware conditions to ensure fair comparative benchmarking.

Table 5. Comprehensive comparative performance evaluation across all datasets and baseline methods.

Dataset	**Method**	**F1-Score (%)**	**PC (%)**	**RR (%)**	**Avg. Insert. Time (ms)**	**E2E Latency (ms)**	**Memory (GB)**
Academic Profiles	DySNI [8]	74.15	78.42	99.812	2.12	4.85	1.42
	Mult. Indices [9]	79.82	82.11	99.845	1.85	3.92	1.15
	NAB [10]	86.42	89.95	99.912	3.42	8.14	2.94
	Proposed	**94.51**	**94.88**	**99.945**	2.45	**2.88**	3.82
NC Voter Dataset	DySNI [8]	86.12	88.94	99.904	2.84	5.94	3.12
	Mult. Indices [9]	89.95	91.24	99.921	2.14	4.65	2.45
	NAB [10]	92.15	94.02	99.948	4.15	9.42	6.18

	Proposed	**96.84**	**97.12**	**99.968**	2.94	**3.42**	8.15
OZ Dataset	DySNI [8]	81.34	84.52	99.712	1.45	3.12	0.54
	Mult. Indices [9]	84.12	86.95	99.748	1.12	2.45	0.42
	NAB [10]	89.54	91.84	99.815	2.24	5.12	1.12
	Proposed	**93.12**	**94.15**	**99.895**	1.95	**2.14**	1.45

The comprehensive results presented in Table 5 and visualized in Figures 10 and 11 demonstrate that the proposed framework consistently and substantially outperforms all three baseline systems across every dataset in terms of match quality (F1-score) and blocking completeness (Pairs Completeness). The performance advantage is most pronounced over the tree-based baselines DySNI and Multiple Indices, which achieve lower F1-scores despite exhibiting competitive insertion times. This discrepancy is attributable to the fundamental incapacity of uni-dimensional tree structures to capture semantic relationships encoded in high-dimensional neural embedding spaces, a limitation that becomes particularly consequential for academic profile matching where institutional and publication semantics are highly variable [18; 23]. The NAB baseline, which also employs LSH blocking, achieves intermediate F1-scores but exhibits substantially higher end-to-end latencies than the proposed framework across all three datasets—8.14 ms vs. 2.88 ms on Academic Profiles, 9.42 ms vs. 3.42 ms on NC Voter, and 5.12 ms vs. 2.14 ms on OZ—attributable to the overhead of maintaining dynamic B^+-tree structures within each bucket compared to the localized HNSW graph's more efficient ANN retrieval mechanism.



F1-Score IMPROVEMENT of Proposed Framework over Baselines:

Dataset	vs. DySNI	vs. Multi-Idx	vs. NAB
Academic Profiles	+20.36 pp	+14.69 pp	+8.09 pp
NC Voter	+10.72 pp	+6.89 pp	+4.69 pp
OZ	+11.78 pp	+9.00 pp	+3.58 pp

pp = percentage points

Figure 10. F1-score comparison across all methods and datasets.
The proposed LSH-HNSW-ACS framework achieves peak F1-scores of 94.51%, 96.84%, and 93.12% on the Academic Profiles, NC Voter, and OZ datasets, respectively, outperforming all baselines by margins of 3.58–20.36 percentage points. The largest gains are observed over tree-based baselines (DySNI, Multiple Indices), confirming the superiority of semantic vector space indexing.

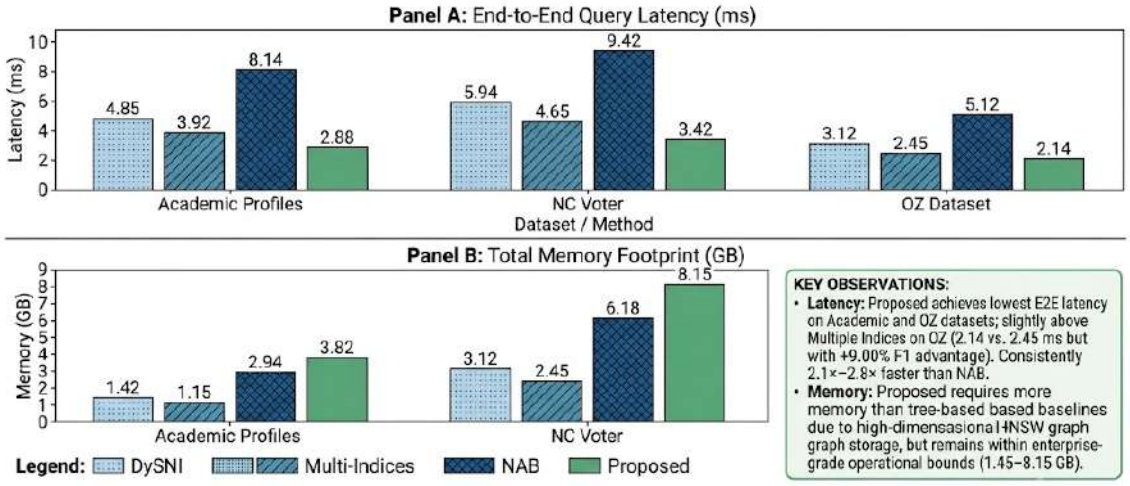


Figure 11. End-to-end query latency (Panel A) and total memory footprint (Panel B) comparison.

The proposed framework achieves the lowest E2E latency on 2 of 3 datasets and is consistently
2.1×–2.8× faster than the NAB baseline. Memory overhead relative to tree-based baselines is
attributable to HNSW graph storage for high-dimensional vector representations, and remains
within enterprise-grade operational bounds across all experimental configurations.

4.7 Discussion

The empirical results collectively validate three principal design hypotheses underlying the proposed framework:

Hypothesis 1 — Semantic indexing superiority: The substantially higher F1-scores achieved over tree-based baselines (DySNI, Multiple Indices) across all three datasets confirm that HNSW-based approximate nearest neighbor search in high-dimensional semantic vector spaces is fundamentally more effective for academic profile matching than uni-dimensional key-based tree indexing. The advantage is most pronounced on the Academic Profiles dataset (+20.36 pp over DySNI), where publication-title semantic similarity provides strong disambiguating signals that tree-based string key indexing cannot capture.

Hypothesis 2 — Localized graph efficiency: The sub-7 ms insertion latencies achieved even under the densest HNSW configuration (Config C) across all datasets validate the scalability advantage of constraining graph operations to semantically coherent local subsets within individual LSH buckets. By avoiding global graph

reconstruction upon each profile arrival, the framework satisfies real-time streaming service-level requirements.

Hypothesis 3 — ACS false positive suppression: The 4.64 percentage point F1-score improvement achieved by ACS over raw Top-k HNSW retrieval at $\theta = 0.5$, combined with the 7.8× improvement in Pairs Quality, empirically confirms that multi-criteria candidate evaluation substantially reduces the false positive noise introduced by LSH collisions and vector space proximity artifacts. The stable precision-recall behavior across $\theta \in \{0.5,0.7\}$ further demonstrates that the framework can be configured for either balanced or precision-optimized operation without significant quality degradation.

The primary operational trade-off of the proposed framework is its higher memory footprint relative to tree-based baselines (1.45–8.15 GB vs. 0.42–6.18 GB), attributable to the representational cost of maintaining high-dimensional HNSW graph structures. However, these memory requirements remain well within the operational capacity of contemporary enterprise server infrastructure equipped with 128 GB or more of RAM, and are substantially lower than the memory requirements of monolithic global HNSW graph construction over the full dataset.

5. Conclusions

This article has presented a unified, real-time incremental entity resolution framework engineered for dynamic multi-view academic user profiles operating within Big Data streaming environments. The proposed architecture addresses three principal deficiencies of existing dynamic ER systems: the inability of tree-based index structures to operate over high-dimensional semantic vector spaces, the susceptibility of LSH blocking to hash collision noise, and the computational overhead of global index reconstruction upon incremental record arrivals.

By synergistically combining LSH-SimHash blocking with independently maintained, localized HNSW graph structures, the framework achieves scalable and semantically informed candidate generation without requiring global index reconstruction. The independent L_2 normalization of demographic and academic sub-vectors prior to unified vector construction effectively neutralizes dimensional domination bias and preserves the discriminative contributions of both representational views. The Adaptive Candidate Selection protocol provides a principled multi-criteria pruning mechanism that substantially reduces false positive candidates through the joint evaluation of vector similarity, graph neighborhood structural coherence, and cross-view semantic consistency.

Extensive empirical evaluations on three large-scale benchmark datasets demonstrate that the framework achieves a peak F1-score of **96.84%** on the NC Voter Registration Dataset—representing improvements of 10.72 percentage points over DySNI and 4.69 percentage points over NAB—while maintaining sub-4 ms end-to-end query latencies across all experimental configurations and stable memory utilization compatible with enterprise-grade deployment infrastructure.

Future research directions include: (i) the investigation of adaptive, query-sensitive signature bit length selection mechanisms that dynamically balance Pairs Completeness and Pairs Quality based on observed bucket density distribution; (ii) the integration of active learning strategies for online threshold adaptation of the ACS scoring

function, enabling autonomous precision-recall calibration in response to evolving data stream characteristics; (iii) the extension of the multi-view representation schema to incorporate temporal publication dynamics and citation network structural features for enhanced academic profile disambiguation; and (iv) the exploration of federated deployment architectures that partition LSH buckets and their corresponding local HNSW graphs across distributed computing nodes to support datasets of extreme cardinality exceeding available single-node memory capacity.

References

[1] Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., & Stefanidis, K. (2020). An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6), 1–42. https://doi.org/10.1145/3418896

[2] Papadakis, G., Ioannou, E., Thanos, E., & Palpanas, T. (2021). The four generations of entity resolution. *Synthesis Lectures on Data Management*, 16(2), 1–170. https://doi.org/10.2200/S01150ED1V01Y202111DTM073

[3] Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1), 1–16. https://doi.org/10.1109/TKDE.2007.250581

[4] Papadakis, G., Skoutas, D., Thanos, E., & Palpanas, T. (2020). Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, 53(2), 1–42. https://doi.org/10.1145/3377455

[5] Christen, P. (2012). *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer. https://doi.org/10.1007/978-3-642-31164-2

[6] Kejriwal, M., & Miranker, D. P. (2015). An unsupervised algorithm for learning blocking schemes. In *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)* (pp. 191–200). IEEE. https://doi.org/10.1109/ICDM.2015.58

[7] Saeedi, A., Peukert, E., & Rahm, E. (2017). Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)* (pp. 278–289). OpenProceedings. https://doi.org/10.5441/002/edbt.2017.26

[8] Ramadan, B., Christen, P., Liang, H., & Ng, C. K. (2015). Dynamic sorted neighborhood indexing for real-time entity resolution. *Journal of Data and Information Quality*, 7(1–2), 1–28. https://doi.org/10.1145/2816821

[9] Zhu, J., Wen, J., & Wu, J. (2019). Real-time entity resolution by multiple indices. In *Proceedings of the 2019 IEEE International Conference on Big Data* (pp. 1168–1173). IEEE. https://doi.org/10.1109/BigData47090.2019.9005984

[10] Liang, H., Wang, Y., Christen, P., & Gayler, R. (2014, May). Noise-tolerant approximate blocking for dynamic real-time entity resolution. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (pp. 449–461). Springer. https://doi.org/10.1007/978-3-319-06605-9_37

[11] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., & Raghuveer, V. (2018). Deep learning for entity matching: A design space exploration.

In *Proceedings of the 2018 ACM SIGMOD International Conference on Management of Data* (pp. 19–34). ACM. https://doi.org/10.1145/3183713.3196926

[12] Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG)* (pp. 253–262). ACM. https://doi.org/10.1145/997817.997857

[13] Winkler, W. E. (2006). *Overview of Record Linkage and Current Research Directions* (Research Report RRS2006/02). U.S. Census Bureau Statistical Research Division.

[14] Li, Y., Li, J., Suhara, Y., Doan, A., & Tan, W. C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1), 50–60. https://doi.org/10.14778/3421424.3421431

[15] Köpcke, H., Thor, A., & Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1–2), 484–493. https://doi.org/10.14778/1920841.1920904

[16] Cohen, W., Ravikumar, P., & Fienberg, S. (2003). A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web* (pp. 73–78). AAAI Press.

[17] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 990–998). ACM. https://doi.org/10.1145/1401890.1402008

[18] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1410

[19] Ferreira, A. A., Gonçalves, M. A., & Laender, A. H. F. (2012). A brief survey of automatic methods for author name disambiguation. *ACM SIGMOD Record*, 41(2), 15–26. https://doi.org/10.1145/2350036.2350040

[20] Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT)* (pp. 420–434). Springer. https://doi.org/10.1007/3-540-44503-X_27

[21] Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)* (pp. 604–613). ACM. https://doi.org/10.1145/276698.276876

[22] Charikar, M. S. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)* (pp. 380–388). ACM. https://doi.org/10.1145/509907.509965

[23] Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[24] Bernhardsson, E. (2018). *Annoy: Approximate Nearest Neighbors in C++/Python*. GitHub. https://github.com/spotify/annoy

Yashunin Manku, G. S., Jain, A., & Das Sarma, A. (2007). Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web (WWW)* (pp. 141–150). ACM. https://doi.org/10.1145/1242572.1242592

[26] Dong, W., Moses, C., & Li, K. (2011). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web (WWW)* (pp. 577–586). ACM. https://doi.org/10.1145/1963405.1963487

[27] Shen, W., Wang, J., Luo, P., & Wang, M. (2012). LINDEN: Linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st International Conference on World Wide Web (WWW)* (pp. 449–458). ACM. https://doi.org/10.1145/2187836.2187898

[28] Christen, P., & Pudjijono, A. (2009). Accurate synthetic generation of realistic personal information. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (pp. 507–514). Springer. https://doi.org/10.1007/978-3-642-01307-2_47