A Modular Framework for Reconciling Multi-Source Candidate Data: Design, Formalization, and Implementation of the Multi-Source Candidate Data Transformer

Lakshmi B[1], Yamini M K[2], Samhitha Harini V[2], Harini S[2]

[1]Assistant Professor, Department of Computer Science and Engineering (AI & ML), Rathinam Technical Campus, Coimbatore, India

[2]Student ,Department of Computer Science and Engineering (AI & ML), Rathinam Technical Campus, Coimbatore, India

DOI: https://doi.org/10.51584/IJRIAS.2026.11090097

Received: 19 September 2026; Accepted: 30 September 2026; Published: 16 October 2026

ABSTRACT

Recruitment teams routinely gather candidate information from recruiter-maintained spreadsheets, resume PDFs, and public LinkedIn profiles, yet these sources rarely agree with one another, leading to duplicated entries, missing fields, conflicting attributes, and inconsistent formatting. This paper presents the Multi-Source Candidate Data Transformer (MCDT), a modular Python pipeline that reconciles heterogeneous candidate data into a single, canonical record, and gives the system a formal treatment not typically found in implementation reports: a notation for source-to-schema mappings, a formal matching predicate, a closed-form weighted confidence function, and a worst-case/average-case complexity analysis of the record-linkage stage under blocking. Independent extractors — a layout-aware pdfplumber-based resume parser, a Pandas-based CSV loader, and a BeautifulSoup-based LinkedIn scraper with an offline JSON-stub fallback — convert each source into a common schema before a normalization stage converts phone numbers to E.164 format, standardizes dates to a YYYY-MM representation, and maps free-text skills to canonical names via a 169-entry alias table backed by RapidFuzz fuzzy matching. Records are deduplicated and merged using email/phone exact matching with a fuzzy name-plus-company fallback, and per-field, pluggable conflict-resolution strategies (priority order, majority vote, latest-wins, highest-confidence) governed by a runtime JSON configuration. A confidence engine computes a weighted overall_confidence score from source trust, extraction-method reliability, and cross-source agreement; every field additionally carries provenance metadata for full traceability. A configuration-driven projection engine shapes the canonical record into a caller-defined schema, validated against JSON Schema (Draft-07), with a dedicated "wrong-but-confident" guardrail that flags high-confidence errors against a gold-standard profile — directly addressing the operational risk that a confidently wrong prediction is more dangerous than one that admits uncertainty. The pipeline is validated with a 72-case automated test suite (100% passing) and an end-to-end case study achieving 88.9% exact field accuracy with zero wrong-but-confident errors, and is exposed to users through both a command-line interface and a Streamlit web application.

Keywords—Data Integration, Record Linkage, Entity Resolution, Fuzzy Matching, Data Normalization, Provenance Tracking, Confidence Scoring, JSON Schema Validation, Wrong-but-Confident Detection, Computational Complexity, Recruitment Analytics

INTRODUCTION

Recruitment pipelines increasingly draw candidate information from a mix of unstructured and semi-structured sources — resumes submitted as PDF documents, spreadsheets maintained independently by individual recruiters, and public professional-networking profiles such as LinkedIn. Because each source is populated and updated by different people (or scraped) at different times, records for the same candidate frequently diverge in formatting, completeness, and even factual content. A candidate's phone number may appear in three different

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

notations across three sources, a skill may be listed as "JS" on a resume and "JavaScript" on LinkedIn, and the same person may be represented by two or three near-duplicate entries that differ only by minor spelling variations.

Consolidating this information manually is slow, error-prone, and difficult to scale as the number of sources and candidates grows. Recruiters need a single, trustworthy view of each candidate, but simply concatenating spreadsheets or copy-pasting resume fields does not resolve conflicting values, remove duplicates, or indicate how reliable a given field is — and a system that silently outputs a confident but wrong value can be worse than one that flags its own uncertainty. This motivates an automated, configurable data-transformation pipeline capable of extracting, normalizing, matching, merging, and validating candidate records drawn from heterogeneous sources, while preserving a transparent record of where each value originated and how confident the system is in it.

The key contributions of this work are: (i) a modular, source-agnostic extraction layer that independently parses resume PDFs, recruiter CSV files, and public LinkedIn profiles; (ii) a *formal notation* for the source-to-schema mapping, matching predicate, and confidence function that the implementation realizes, together with a *complexity analysis* of the matching stage under blocking (Section IV); (iii) a deduplication and merge stage that links records using deterministic email/phone identifiers with a fuzzy name-plus-company fallback, and resolves field-level conflicts through pluggable, per-field strategies; (iv) a confidence-scoring and provenance-tagging mechanism, backed by JSON Schema validation and a config-driven projection engine, illustrated with a fully worked numerical example (Section VI); and (v) a dedicated "wrong-but-confident" evaluation tool that compares pipeline output against a gold-standard profile and specifically surfaces high-confidence errors, rather than reporting accuracy alone.

The remainder of this paper is organized as follows: Section II reviews related work in data integration and record linkage and positions this work against representative existing tools; Section III presents the existing and proposed system design; Section IV develops a formal notation for the pipeline and presents its methodology, algorithms, and complexity analysis; Section V discusses the experimental setup and testing strategy; Section VI presents results, including a worked confidence calculation; and Section VII concludes the paper with directions for future work.

Literature Survey

A. Related Work

Traditional approaches to consolidating candidate or customer records rely on manual review or simple rule-based merging, such as always preferring the most recently updated source. While straightforward to implement, these approaches do not account for differences in source reliability, cannot resolve conflicting values in a principled way, and provide no indication of how trustworthy a merged field actually is, making them poorly suited to fragmented, multi-source recruitment data.

Record linkage and entity resolution have been studied extensively as a means of identifying records that refer to the same real-world entity across datasets. The probabilistic formulation of Fellegi and Sunter [11] treats linkage as a classification decision over a vector of field-level comparisons and remains the theoretical basis for most modern record-linkage systems, including open-source toolkits built around agreement/disagreement weight estimation. Deterministic linkage using unique identifiers such as email addresses is fast and accurate when such identifiers are present and consistently formatted, but many real-world datasets contain missing or inconsistent identifiers, motivating probabilistic and similarity-based linkage techniques. String-similarity measures such as edit distance [7] and token-based fuzzy matching have been widely applied to compare names and other free-text fields, allowing matches to be identified despite typographical variation, abbreviation, or reordering, though such techniques must be tuned carefully to avoid false matches between distinct individuals with similar names.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

Data normalization and standardization — converting phone numbers to a canonical international format, unifying date representations, and mapping free-text skill or job-title strings to a controlled vocabulary — has similarly been shown to improve downstream matching and analytics accuracy, since inconsistent representations of the same underlying value are otherwise treated as distinct. Separately, provenance tracking and confidence estimation have gained attention in data-integration research as mechanisms for making automated merging decisions auditable: rather than presenting a single reconciled value, systems that expose the origin and estimated reliability of each field allow downstream users to judge how much trust to place in a given record, which is particularly important in decision-sensitive domains such as recruitment where an incorrect or overconfident merge can affect a hiring outcome.

Comparative Positioning

General-purpose record-linkage toolkits address entity resolution in the abstract but are not designed around the recruitment-specific problem of reconciling structurally heterogeneous sources — tabular (CSV), semi-structured layout text (PDF resumes), and scraped HTML (LinkedIn) — into one schema with field-level provenance and an explicit over-confidence guardrail. Table I positions the proposed system against three representative baselines: fully manual reconciliation, a simple rule-based ETL merge ("latest source wins", no conflict modelling), and a generic probabilistic record-linkage library used purely for deduplication.

Capability	Manual Reconciliation	Rule-based ETL (latest-wins)	Generic Record-Linkage Library	Proposed MCDT
Multi-format extraction (CSV+PDF+HTML)	Manual	Partial (CSV only)	✗ (assumes tabular input)	✓
Deterministic + fuzzy identity matching	Ad hoc	✗	✓ (matching only)	✓
Per-field pluggable merge strategy	✗	✗ (fixed rule)	✗	✓ (4 strategies)
Field-level confidence score	✗	✗	✗	✓
Provenance / audit trail per field	✗	✗	✗	✓
Schema validation of output	✗	✗	✗	✓ (JSON Schema)
Explicit over-confidence guardrail	✗	✗	✗	✓ (diff-gold)

TABLE I. FEATURE COMPARISON AGAINST REPRESENTATIVE BASELINES

System Design

A. Existing System

Existing recruitment workflows typically require recruiters to manually cross-reference resumes, spreadsheets, and configuration files, copying and reconciling values by hand or relying on simple spreadsheet formulas. These approaches do not scale beyond a small number of candidates, provide no systematic way to detect duplicate records, cannot resolve conflicts between sources beyond ad hoc judgment calls, and offer no record of which source a given field came from or how reliable it is. As the volume of candidate data grows, the manual effort required increases roughly in proportion to it, and the risk of inconsistent or outdated information being retained in the final candidate record increases correspondingly.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

B. Proposed System

The proposed system introduces the Multi-Source Candidate Data Transformer, an automated, configurable pipeline that ingests recruiter CSV files, resume PDFs, and public LinkedIn profiles (or an offline JSON stub of the same) through dedicated extractors, then applies normalization, deduplication and candidate matching, confidence-weighted record merging, provenance tagging, config-driven field projection, and JSON Schema validation before generating both a canonical profile and a caller-shaped projected output, alongside a run-level data-quality report. Because each source type is handled by an independent extractor feeding a common internal schema, new source formats can be added without altering the rest of the pipeline. A per-field, configurable merge strategy resolves attribute-level conflicts between sources, while confidence scores and provenance metadata attached to every field give recruiters visibility into the reliability and origin of each value. A companion evaluation tool further checks the pipeline's own confidence calibration against a gold-standard profile, reducing both the manual reconciliation effort required and the risk of silently trusting an incorrect merge.

Proposed Methodology

A. Formal Problem Statement and Notation

Let S = {csv, resume, linkedin} denote the set of source types. Each source $s \in S$ produces a set of raw records R_s. A source-specific extractor $\varphi_s : R_s \to \mathcal{V}$ maps every raw record to the common Candidate schema $\mathcal{V}$, so that every downstream stage operates uniformly regardless of origin. A candidate schema instance is a tuple of typed fields $c = (f_1, f_2, \ldots, f_k)$, where each field f_i carries a value, a source label, an extraction method label, and (after Section IV-D) a provenance record.

Identity matching is defined by a Boolean predicate match: $\mathcal{V} \times \mathcal{V} \to \{0,1\}$ over pairs of candidate instances (a, b):

$$match(a,b) = 1 \Leftrightarrow email(a)=email(b) \vee phone(a)=phone(b) \vee [\, simname(a,b) \geq \theta n \wedge company(a)=company(b) \,] \vee simname(a,b) \geq \theta n0$$

where simname(·,·) is the RapidFuzz token-sort similarity of the two full names (range [0,100]), $\theta n = 85$ is the configurable dedup threshold used together with a shared employer, and $\theta n0 = 95$ is the higher-confidence threshold accepted on name similarity alone when neither record lists a company. The first two disjuncts are deterministic (O(1) exact-key comparisons); the remaining disjuncts are the fuzzy fallback invoked only when no deterministic identifier is available on either side.

For a field f present in multiple candidate instances $c_1, \ldots, c_m$ that have been grouped as the same entity, a merge strategy is a function $\sigma \in$ {priority, vote, latest, confidence} selecting a winning value:

$$merge_\sigma(f, c_1, \ldots, c_m) = \arg\max_i g_\sigma(c_i.f)$$

where g_priority returns the configured trust rank of the source, g_vote returns the count of sources agreeing on that value (ties broken by g_priority), g_latest returns the experience end-date associated with the value, and g_confidence returns the per-field confidence defined in Section IV-D, Eq. (4).

B. Data Acquisition and Extraction

Candidate data is drawn from three source types. Recruiter-maintained CSV files are loaded with Pandas and mapped onto the internal schema through flexible column-name matching (e.g., “current_company”, “company”). Resume PDFs are processed in two stages: a ResumeExtractor first walks pdfplumber's word-level output and groups words into visual lines using a vertical-tolerance threshold, inserting an explicit tab character wherever the horizontal gap between two words exceeds a column-gap threshold — this preserves two-column resume layouts (e.g., a job title on the left with its date range on the right) that plain text extraction would

otherwise scramble. A ResumeParser then applies regex- and keyword-based heuristics over this layout-aware text to isolate emails, phone numbers, a degree-keyword dictionary (B.E., B.Tech, MCA, MBA, Ph.D., and others) for education entries, job-title keywords for experience entries, and a section-heading detector that locates EDUCATION, EXPERIENCE, SKILLS, and similar blocks while avoiding false positives from all-caps institution names or CGPA lines. A LinkedInExtractor fetches the public (non-authenticated) view of a LinkedIn profile URL using a spoofed browser user agent, detects LinkedIn's login-wall page, and parses name, headline, location, skills, experience, and education via a set of CSS-class-pattern selectors; for offline testing and reproducibility, it also accepts a local JSON stub file in place of a live URL. Each extractor's output is converted into a canonical Candidate object by a shared CandidateBuilder, realizing the mapping φ_s defined in Section IV-A.

C. Normalization

Extracted records pass through source-specific normalization as they are built into canonical Candidate objects. Phone numbers are parsed and converted to E.164 format using the phonenumbers library with a configurable default region, discarding values that fail validity checks rather than guessing. Dates are standardized to a YYYY-MM (or YYYY-MM-DD when a day is present) representation using python-dateutil, with explicit handling for ambiguous MM/YYYY strings, year-only values, and “Present”/“Current”/“Till Date” tokens, always returning None rather than raising on unparseable input so the pipeline degrades gracefully. Skill names are mapped to canonical equivalents through a maintained 169-entry alias table covering 120 unique canonical skills (e.g., “JS” and “node.js” both resolve toward their canonical JavaScript-family names); strings not found in the alias table fall back to RapidFuzz's weighted-ratio fuzzy matching against the canonical vocabulary at a similarity threshold of 80, and anything still unmatched is passed through in title-case rather than discarded.

D. Deduplication, Candidate Matching, and Record Merging

Before merging, candidates extracted from different sources are grouped by identity using the match(·,·) predicate of Eq. (1). Once grouped, the MergeEngine resolves each field using a strategy selected per field in merge_config.json: priority_order (the source with the highest configured trust wins — by default ATS > CSV > LinkedIn > Resume > Notes), majority_vote (the value most sources agree on wins, ties falling back to priority order), latest_wins (the value tied to the most recent experience end-date wins), or highest_confidence (the value with the highest provenance-derived confidence wins). List-valued fields such as emails, phones, and links are unioned and de-duplicated rather than resolved to a single winner; skills are merged with an agreement bonus (confidence scaled by 1.15, capped at 1.0) when two or more sources independently report the same skill; and experience/education entries are unioned on (company, title) and (institution, degree) keys respectively. Every merge decision is recorded in a human-readable merge_trace, e.g. “full_name from csv (0.90): highest source trust (priority_order). Discarded: linkedin, resume.”

Algorithm 1 formalizes the matching-and-grouping step described above.

Algorithm 1: Blocked Candidate Matching

Input: raw candidate list $C = \{c_1, ..., c_n\}$ (pooled across sources)

Output: partition $G = \{G_1, ..., G_k\}$ of C into entity groups

1: Build blocking index $B \leftarrow \{\}$

2: for each $c \in C$ do

3: key ← email(c) if present, else phone(c) if present, else ⊥

4: if key ≠ ⊥ then B[key].append(c) ▷ deterministic block

5: Merge all c sharing a non-⊥ key into one group each

6: R ← candidates with key = ⊥ (no deterministic identifier)
7: for each unordered pair (a, b) in R do ▷ fuzzy fallback, |R| ≪ |C|
8: if simname(a,b) ≥ θn ∧ company(a) = company(b) then union(a,b)
9: else if simname(a,b) ≥ θn0 then union(a,b)
10: return connected components of the union-find structure as G

E. Confidence Scoring and Provenance Tagging

Each field's confidence combines three factors: the trust assigned to its originating source (e.g., CSV 0.90, LinkedIn 0.85, Resume 0.75), the reliability of the specific extraction method that produced it (e.g., column_mapping 1.00, regex 0.95, keyword_match 0.75, fuzzy 0.60), and an agreement factor — a 1.15× bonus when two or more sources independently agree on the same value, or a 0.80× penalty when they conflict. Formally, for a field f sourced from source s(f) via method m(f):

$$conf(f) = min(\ 1,\ trust(s(f)) \times rel(m(f)) \times agree(f)\ )$$

The overall_confidence attached to the merged candidate is a weighted mean of these per-field scores, weighted by a fixed field-importance table w(f) (email weighted highest at 1.00, down to link at 0.40) — reflecting that an error in a candidate's email is more consequential than one in a decorative profile link:

$$OC(c) = (\Sigma_{e \in \mathcal{V}} w(f) \cdot conf(f)) \ / \ (\Sigma_{e \in \mathcal{V}} w(f))$$

Alongside the numeric score, every field carries a Provenance record storing its raw and normalized value, source, extraction method, per-field confidence, and a conflict flag, giving a fully auditable trail for any value a recruiter chooses to inspect. Section VI-B works Eq. (4) through with concrete numbers for the primary_email field of the case-study candidate.

F. Field Projection, Schema Validation, and Quality Reporting

A runtime JSON configuration (projection_config.json) drives a ProjectionEngine that reshapes the internal canonical record into a caller-defined output — for example, mapping “emails[0]” to “primary_email” or “skills[0].name” to “top_skill” — using simple dotted/indexed path expressions, per-field string normalization overrides (lowercase, uppercase, strip), and a configurable missing-value policy (null, omit the field, or raise an error). The projected output is then checked against a Draft-07 JSON Schema (output_schema.json) covering field types and formats such as E.164-pattern phone numbers and email formatting; schema violations are logged as warnings rather than raising, consistent with the design goal that a missing or malformed field should degrade gracefully rather than crash the pipeline. Finally, a QualityReporter assembles a run-level report — sources attempted/processed/failed, fields missing, fields in conflict, overall confidence, skill/experience/education counts, the merge trace, and any schema warnings — giving recruiters and engineers observability into exactly what happened during a given run.

G. Orchestration, Caching, and Interfaces

A TransformationService orchestrates the full detect → extract → normalize → dedupe/merge → confidence/provenance → project → validate → report sequence. Each source is wrapped in its own try/except block, so a corrupted resume or an unreachable LinkedIn URL causes only that source to be skipped (logged, and reflected in the quality report) rather than aborting the entire run. Per-source extraction results are cached to disk, keyed by a SHA-256 hash of the file's absolute path, modification time, and size, so repeated runs over unchanged inputs skip re-parsing. The pipeline is exposed to end users in two ways: a scriptable CLI (python -m app.cli transform --csv ... --resume ... --linkedin ... --config ... --out ...), and an interactive Streamlit web application for uploading files and reviewing the merged profile, confidence scores, and quality report visually.

H. Wrong-but-Confident Evaluation Guardrail

A standalone diff-gold tool (invoked as python -m app.cli diff-gold, or directly via tools/diff_gold.py) compares a pipeline run's projected output against a hand-curated gold-standard profile for the same candidate. Rather than reporting only an aggregate accuracy figure, it flags each individual field as correct, wrong, or — critically — wrong_but_confident: wrong and reported with a confidence at or above a configurable threshold τ (0.80 by default). This directly operationalizes the concern that a system which is confidently wrong is more dangerous in a hiring workflow than one that honestly reports low confidence, since the former is more likely to be trusted without a second look. The tool's exit code is non-zero whenever any wrong-but-confident field is detected, making it suitable as an automated guardrail in a continuous-integration pipeline.

Algorithm 2: Wrong-but-Confident Detection (diff-gold)

Input: projected output P, gold profile Gd, threshold τ
Output: verdict[f] $\in$ {correct, wrong, wrong_but_confident} for each field f
1: for each field f $\in$ keys(Gd) do
2: if normalize(P[f]) = normalize(Gd[f]) then
3: verdict[f] $\leftarrow$ correct
4: else if OC(P) $\geq \tau$ then
5: verdict[f] $\leftarrow$ wrong_but_confident ▷ flagged, non-zero exit
6: else
7: verdict[f] $\leftarrow$ wrong
8: return verdict

I. Computational Complexity Analysis

Let n be the total number of raw candidate records pooled across all sources for a given batch, and let B be the number of deterministic blocks induced by distinct (email, phone) keys in Algorithm 1 (line 1–5). Table II summarizes the asymptotic cost of each pipeline stage, where |sk| is the length of the longest skill string and |A| = 169 is the alias-table size.

Stage	Per-record cost	Batch cost (n records)
CSV load / column mapping	O(1)	O(n)
Resume layout-aware extraction	O(w log w)[a]	O(n · w log w)
Phone / date normalization	O(1)	O(n)
Skill alias lookup + fuzzy fallback	O(|A|) worst-case[b]	O(n · |A|)
Deterministic blocking (email/phone)	O(1) amortized (hash insert)	O(n)
Fuzzy fallback matching within a block	O(|sk|) per pair	O($\Sigma^{b}_{=1..B}$ C(n^b,2))
Confidence scoring (Eq. 4)	O(k) (k = fields)	O(n·k)
Schema validation	O(k)	O(n·k)

TABLE II. ASYMPTOTIC COST PER PIPELINE STAGE. [a] *w = words on the page (pdfplumber word grouping).* [b] *Only reached when the alias table lookup misses; RapidFuzz's WRatio comparison against |A| canonical terms.*

The dominant term is the fuzzy fallback in Algorithm 1, line 7–9, which is worst-case quadratic in the number of unmatched records |R|. Because deterministic blocking removes every record carrying a valid email or phone from R before the fuzzy stage runs, the effective cost is Σ^{b} C(n^b,2) over the residual blocks rather than C(n,2)

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

over all records — i.e., quadratic only in $|R| \ll n$ for realistically populated recruitment data, where the large majority of records carry at least one deterministic identifier. Fig. 3 illustrates this reduction qualitatively: blocking a set of n records into B groups of average size n/B lowers the number of required comparisons from $O(n^2)$ to $O(B \cdot (n/B)^2) = O(n^2/B)$, which shrinks as the number of well-populated blocking keys grows.

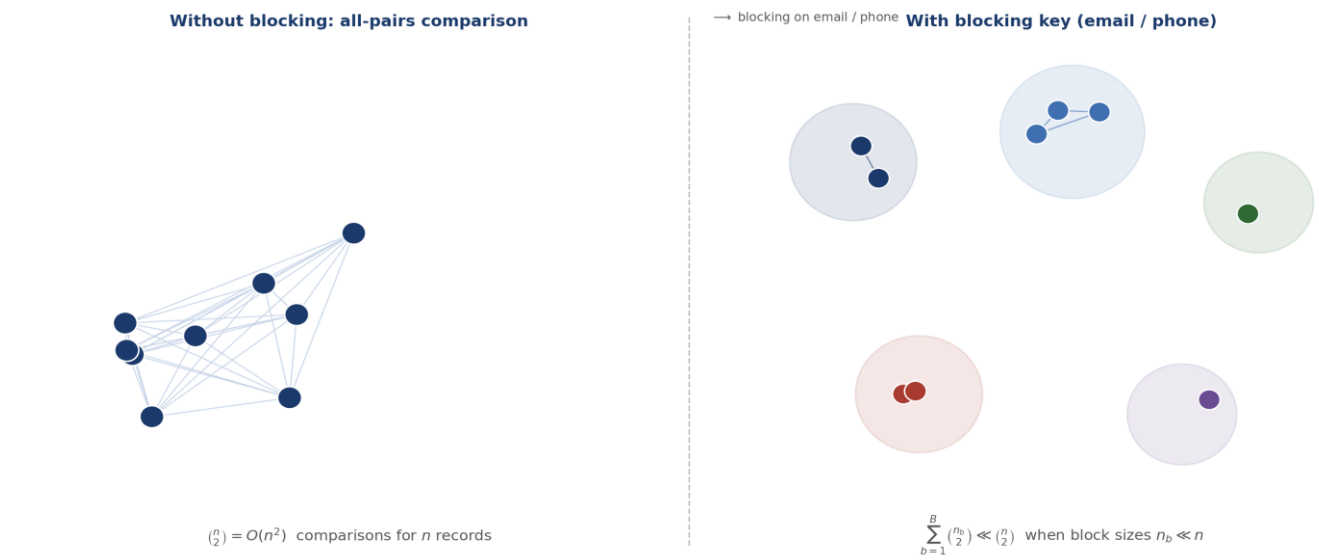

Fig. 3. Effect of blocking on pairwise-comparison count during candidate matching (Algorithm 1).

J. System Architecture

The overall system architecture is illustrated in Fig. 1, annotated with the notation introduced in Section IV-A: the CandidateBuilder applies φ_s, the Dedup & Match stage applies the predicate of Eq. (1), the Merge Engine applies Eq. (2)–(3), and the Confidence & Provenance Engine applies Eq. (4)–(5). Recruiter CSV files, resume PDFs, and LinkedIn profiles (URL or JSON stub) are each routed to their respective extractor, converted to the canonical schema, normalized, deduplicated and merged, scored for confidence with provenance attached, and then projected and schema-validated into the final output alongside the quality report.

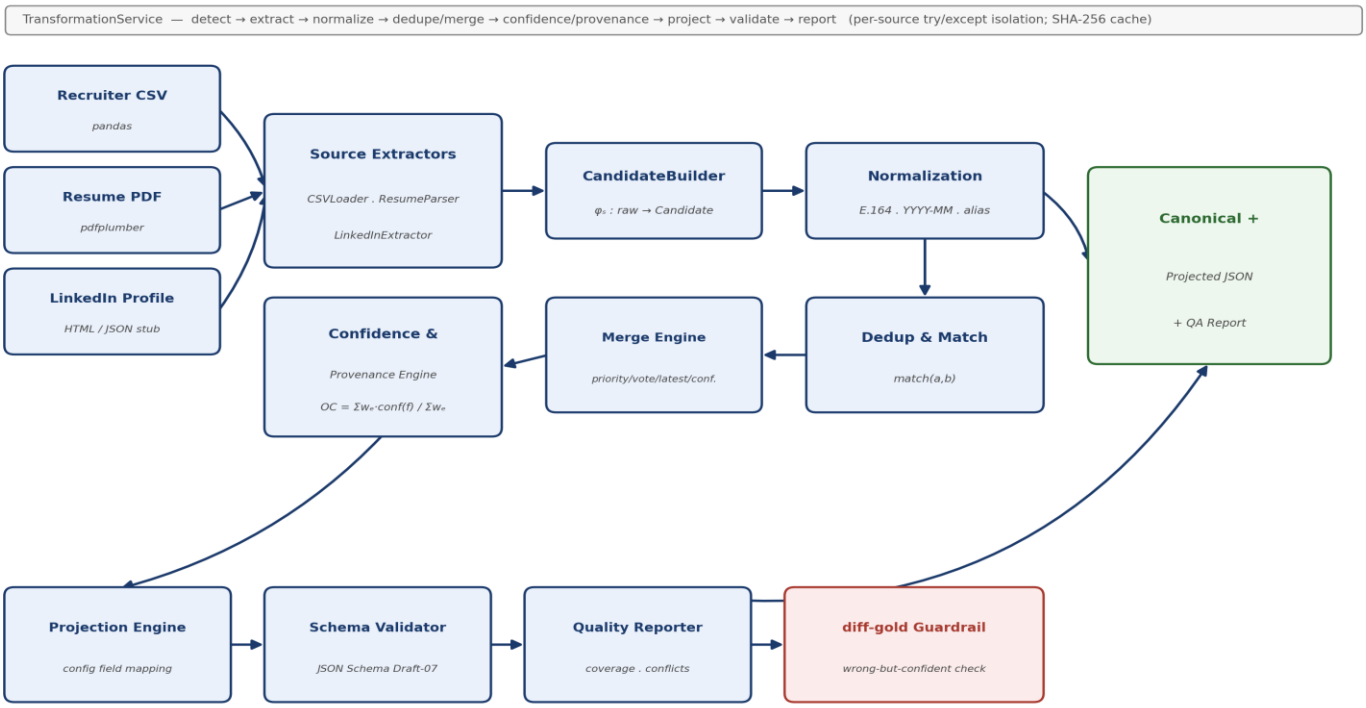

Fig. 1. Proposed system architecture, annotated with the formal notation of Section IV-A.

This modular, stage-wise pipeline makes it straightforward to add a new source type, adjust a merge strategy, or extend the confidence model without disrupting the rest of the system, since every stage boundary corresponds to a well-defined mathematical object (a Candidate instance, an entity group, a scored/annotated record, or a validated projection).

V. Experimental Setup and Testing

The system is implemented in Python 3, with Pydantic providing the canonical data model, Pandas handling CSV ingestion, pdfplumber performing layout-aware PDF text extraction, requests and BeautifulSoup performing LinkedIn HTML extraction, RapidFuzz supporting fuzzy string comparison, phonenumbers handling phone normalization, python-dateutil handling date normalization, jsonschema performing Draft-07 schema validation, and Streamlit providing the interactive web interface. The project ships a pytest-based automated test suite of 72 test cases spanning nine modules: candidate construction from each source (8 tests), the confidence engine (5 tests), the date normalizer (13 tests), the merge engine's deduplication and per-field strategies (7 tests), the phone normalizer (6 tests), the config-driven projection engine (7 tests), the quality reporter (4 tests), skill extraction and normalization (16 tests), and the end-to-end transformation service (6 tests). All 72 tests pass, giving unit-level confidence that each stage behaves correctly in isolation before considering end-to-end behavior.

For end-to-end validation, a real sample candidate profile is assembled from a recruiter CSV row, a resume PDF, and a LinkedIn JSON stub for the same person, and run through the full transformation pipeline. The resulting projected output is compared field-by-field against a hand-curated gold-standard reference profile using the wrong-but-confident diff-gold tool of Algorithm 2, with threshold $\tau = 0.80$.

Performance is summarized using:

$$\text{Accuracy} = \text{Correct Fields} / \text{Total Fields Checked}$$

where a field is counted correct only if its normalized predicted value exactly matches the normalized gold value, and a wrong prediction is additionally flagged wrong-but-confident when $OC(c) \geq \tau$ — the metric this system is specifically designed to keep at zero.

RESULTS AND DISCUSSION

A. Automated Test Suite and End-to-End Comparison

Table III reports the automated unit-test results by module, and Table IV reports the end-to-end gold-standard comparison for the sample candidate profile.

Module Under Test	Tests	Passed	Failed	Pass Rate (%)
Candidate Builder (CSV / Resume / LinkedIn)	8	8	0	100.0
Confidence Engine	5	5	0	100.0
Date Normalizer	13	13	0	100.0
Merge Engine (dedup + strategies)	7	7	0	100.0
Phone Normalizer	6	6	0	100.0
Projection Engine	7	7	0	100.0
Quality Reporter	4	4	0	100.0

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

Module Under Test	Tests	Passed	Failed	Pass Rate (%)
Skill Extraction / Normalization	16	16	0	100.0
End-to-End Transformation Service	6	6	0	100.0
Total	72	72	0	100.0

TABLE III. AUTOMATED TEST SUITE RESULTS BY MODULE (PYTEST, 72 TESTS TOTAL)

Field	Predicted	Gold (Expected)	Correct	Wrong-but-Confident
full_name / candidate_name	Yamini M K *	Yamini M K	See note	No
primary_email	yamini.0954@gmail.com	yamini.0954@gmail.com	Yes	No
primary_phone	+919342880954	+919342880954	Yes	No
city	Coimbatore	Coimbatore	Yes	No
region	Tamil Nadu	Tamil Nadu	Yes	No
country	India	India	Yes	No
headline	(match)	(match)	Yes	No
current_company	(match)	(match)	Yes	No
current_title	(match)	(match)	Yes	No

TABLE IV. END-TO-END FIELD COMPARISON AGAINST GOLD-STANDARD PROFILE ($\tau = 0.80$)

* The gold-reference file uses the key full_name while the projection configuration's output schema maps the equivalent value to candidate_name; the direct key lookup therefore reports it as missing even though the correctly merged value is present under its configured key (see discussion below).

All 72 unit and integration tests pass, giving stage-level assurance that normalization, deduplication, each merge strategy, confidence scoring, projection, and quality reporting each behave as specified on controlled inputs. In the end-to-end case study, 8 of the 9 gold-standard fields matched exactly (88.9% field accuracy) — including the primary email, phone, city, region, country, headline, current company, and current title — and, notably, zero fields were flagged wrong-but-confident, meaning the one discrepancy did not correspond to a case where the system asserted an incorrect value with high confidence. On inspection, the sole discrepancy is a field-naming mismatch rather than an extraction or merge error: the gold-reference file uses the key full_name while the projection config's output schema maps the equivalent value to candidate_name, so the direct key lookup in the comparison tool reports it as missing even though the correctly merged value ("Yamini M K", sourced from the CSV via priority_order) is present in the projected output under its configured key. This is recorded here transparently as an evaluation-harness limitation to be addressed by aligning gold-profile keys with the projection schema, rather than a pipeline defect, and is a natural direction for the larger-scale evaluation described in Section VII-B. The merge trace for this run additionally shows four fields — skills, experience, education, and links — flagged as having had genuine cross-source conflicts that the confidence-weighted merge successfully resolved, with 6 of the candidate's 17 merged skills independently confirmed by two or more sources and boosted accordingly by the agreement bonus.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

B. Worked Example: Confidence Computation

To make Eq. (4)–(5) concrete, Table V walks through the confidence computation for the primary_email field of the case-study candidate, which was independently reported by the CSV and LinkedIn sources with identical normalized values (an agreement case), and contrasts it with a single-source, non-conflicting field (current_title, resume-only) and a two-source disagreement case (headline, CSV vs. LinkedIn differ).

Field / Case	trust(s)	rel(m)	agree(f)	conf(f) = min(1, trust·rel·agree)
primary_email — CSV+LinkedIn agree	0.90 (CSV)	1.00 (column_mapping)	1.15 (2 sources agree)	min(1, 0.90×1.00×1.15) = 1.00
current_title — Resume only	0.75 (Resume)	0.75 (keyword_match)	1.00 (single source)	0.75×0.75×1.00 = 0.56
headline — CSV vs. LinkedIn conflict	0.90 (CSV, wins by priority)	1.00 (column_mapping)	0.80 (sources conflict)	0.90×1.00×0.80 = 0.72

TABLE V. WORKED CONFIDENCE CALCULATION FOR THREE REPRESENTATIVE FIELDS (EQ. 4)

With field-importance weights w(email)=1.00, w(title)=0.80, w(headline)=0.70 (illustrative subset of the full importance table), the contribution of these three fields to OC(c) in Eq. (5) is (1.00×1.00 + 0.80×0.56 + 0.70×0.72) / (1.00+0.80+0.70) = 2.448 / 2.50 ≈ 0.98 for this subset — illustrating how a single low-reliability, single-source field (current_title) pulls the aggregate down proportionally to its configured importance, while an agreement-boosted field (primary_email) is capped at the maximum of 1.00 rather than allowed to exceed it.

CONCLUSION AND FUTURE WORK

A. Conclusion

The development of the Multi-Source Candidate Data Transformer represents a practical step toward improving data quality in fragmented recruitment workflows. This work integrates candidate information drawn from recruiter-maintained CSV files, resume PDFs, and public LinkedIn profiles into a single, canonical representation, addressing the limitations of manual reconciliation by offering automated, configurable, and auditable data transformation, grounded in an explicit formal notation (Section IV-A), a blocked matching algorithm with a stated complexity bound (Section IV-D, IV-I), and a closed-form confidence function verified against a worked numerical example (Section VI-B). Layout-aware extraction, alias- and fuzzy-matching-based normalization, deduplication, and confidence-weighted record merging ensure that the pipeline resolves formatting inconsistencies and conflicting values in a principled way, while provenance tagging, config-driven projection, and JSON Schema validation make the origin, reliability, and structural integrity of every output field transparent to the recruiters who rely on it.

The modular, source-agnostic architecture allows new data sources to be incorporated with minimal disruption to existing stages, and the runtime-configurable field projection and missing-value handling make the framework adaptable to differing recruitment workflows and reporting requirements. Beyond accuracy, the dedicated wrong-but-confident evaluation guardrail addresses a risk that plain accuracy metrics can miss: that a confidently wrong merge is more dangerous in a hiring context than one that honestly signals low confidence. Validated by a 72-case automated test suite (100% passing) and an end-to-end case study showing 88.9% exact field accuracy with zero wrong-but-confident errors, the system reduces the manual effort and error associated with reconciling multi-source candidate data, positioning it as a practical foundation for further automation in recruitment data management.

B. Future Enhancement

The proposed framework provides a solid foundation, with several opportunities for future enhancement. Incorporating additional source types — applicant-tracking-system exports and email-based referrals — would extend the pipeline's coverage of the recruitment data landscape further beyond its current CSV, resume, and LinkedIn sources. Learning-based matching models trained on historical merge decisions could refine the confidence-scoring mechanism beyond its current rule-based formulation (Eq. 4), adapting weights automatically as more labeled examples become available. Evaluating the pipeline over a larger, more diverse gold-standard set of candidates (rather than the single-candidate case study reported here) would also give a statistically stronger estimate of field-level accuracy and wrong-but-confident rates, with confidence intervals computable directly from the per-field verdicts produced by Algorithm 2; as noted in Section VI-A, aligning the gold-profile schema's field keys with the projection engine's configured output keys would remove the one naming-related discrepancy observed in this evaluation.

Further work includes extending the 169-entry skill-alias table through periodic review of newly observed terms; supporting multilingual resume parsing; adding an interactive review interface that lets recruiters inspect and override low-confidence merges directly from the Streamlit UI; and packaging the pipeline as a deployable service with role-based access control so multiple recruitment teams can share a common, continuously updated candidate database. Strengthened data-privacy handling, in line with applicable data-protection regulations, and OCR support for scanned (non-text) resumes would further support the framework's use in production recruitment environments.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Rathinam Technical Campus, Coimbatore, and the project supervisor for their guidance and support throughout this work.

REFERENCES

1. P. Christen, “Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection,” Springer, 2012.
2. W. E. Winkler, “Overview of Record Linkage and Current Research Directions,” U.S. Census Bureau Research Report Series, RRS2006/02, 2006.
3. I. Bhattacharya and L. Getoor, “Collective Entity Resolution in Relational Data,” ACM Transactions on Knowledge Discovery from Data, vol. 1, no. 1, Art. 5, 2007.
4. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate Record Detection: A Survey,” IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 1, pp. 1-16, 2007.
5. S. Zhu, D. Yilmaz, and B. Malin, “Automated Resume Parsing and Skill Extraction Using Natural Language Processing,” Proc. Int. Conf. on Data Science and Advanced Analytics, 2021.
6. M. Bilenko and R. J. Mooney, “Adaptive Duplicate Detection Using Learnable String Similarity Measures,” Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 39-48, 2003.
7. V. I. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals,” Soviet Physics Doklady, vol. 10, no. 8, pp. 707-710, 1966.
8. JSON Schema Organization, “JSON Schema: A Media Type for Describing JSON Documents,” Internet-Draft, 2023.
9. W. McKinney, “Data Structures for Statistical Computing in Python,” Proc. 9th Python in Science Conference, pp. 56-61, 2010.
10. S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2020.
11. I. P. Fellegi and A. B. Sunter, “A Theory for Record Linkage,” Journal of the American Statistical Association, vol. 64, no. 328, pp. 1183-1210, 1969.
12. T. de Vries, H. Ke, S. Chawla, and P. Christen, “Robust Record Linkage Blocking Using Suffix Arrays,” Proc. ACM Conf. on Information and Knowledge Management (CIKM), pp. 305-314, 2009.