UnivColl International Multidisciplinary Research Journal
Volume 1 Issue 3, December 2025, pp. 1-25
ISSN (Online): 3108-1460
Homepage: https://univcolljournal.com/

Title:	**A Lightweight Explainable Framework for Schema Matching and Data Fusion in Low-Code Data Engineering Pipelines**
Author(s):	**Manoj Parasa**
Affiliation(s):	Independent Researcher, Dallas, TX, USA
DOI:	https://doi.org/10.65919/uimrj.2025.v1i3001
History:	Received: 07 November 2025, Accepted: 10 December 2025, Published: 31 December 2025
Citation:	Parasa, M. (2025). A Lightweight Explainable Framework for Schema Matching and Data Fusion in Low-Code Data Engineering Pipelines. UnivColl International Multidisciplinary Research Journal, 1(3), 1-25. https://doi.org/10.65919/uimrj.2025.v1i3001
Copyright:	© 2025 The Author(s)
Licensing:	This is an open access article distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).
Conflict of Interest:	The author(s) declare no conflict of interest.
Funding:	This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.
Corresponding Author:	Manoj Parasa ✉
Published By:	UnivColl Publications

A Lightweight Explainable Framework for Schema Matching and Data Fusion in Low-Code Data Engineering Pipelines

Manoj Parasa
Independent Researcher, Dallas, TX, USA

Abstract

Low-code data engineering environments increasingly support rapid data ingestion, transformation, and integration, yet schema matching remains a persistent challenge when heterogeneous datasets use inconsistent naming conventions, limited metadata, and weak documentation. This study proposes a lightweight explainable framework for schema matching and data fusion in low-code data engineering pipelines. The framework integrates sentence embedding similarity, fuzzy string matching, keyword-based heuristics, and data-type compatibility validation to identify candidate correspondences between independently structured tabular schemas without requiring labeled training data, predefined ontologies, or extensive manual mapping. The approach is evaluated using publicly available Airbnb Listings and Reviews datasets, where the Listings file contains property-level attributes and the Reviews file contains review-event attributes, creating a practical case for schema alignment across related but differently structured data sources. In addition to hybrid similarity scoring, the framework includes an optional large language model based justification layer that generates natural-language explanations for predicted column matches, improving interpretability for analysts, data engineers, and low-code users. The experimental results show that a moderate similarity threshold of 0.65 achieved a precision of 0.50, recall of 0.33, and F1-score of 0.40, while a stricter threshold of 0.80 achieved a precision of 1.00, recall of 0.25, and F1-score of 0.40. These findings demonstrate the trade-off between broader mapping coverage and high-confidence matching, while confirming the value of explainable scoring for low-supervision schema alignment. The proposed framework contributes a domain-agnostic and low-footprint method for preparing heterogeneous data sources for downstream data fusion, ETL automation, and analytical workflows in low-code data engineering settings.

Keywords: Schema matching, data fusion, low-code data engineering, explainable data integration, semantic similarity, sentence embeddings, fuzzy string matching, hybrid similarity scoring, ETL automation, unsupervised schema alignment.

1. Introduction

Low-code data engineering platforms have become increasingly important for organizations that need to ingest, transform, combine, and analyze heterogeneous data with reduced programming effort. These platforms allow analysts, data engineers, and business users to design integration workflows through visual interfaces, reusable connectors, and configurable transformation components. However, the effectiveness of such pipelines still depends on a foundational technical task: identifying how fields from different data sources correspond to one another. When datasets are created independently, their schemas often differ in naming conventions, structure, granularity, and metadata quality. As a result, schema matching remains a critical barrier to reliable data integration, even in environments designed to simplify engineering work.

Schema matching refers to the process of identifying semantic correspondences between attributes in different data schemas. It supports essential downstream tasks such as data fusion, entity resolution, extract-transform-load processing, master data preparation, and analytical table construction. In practice, this task is difficult because real-world datasets rarely follow a unified naming standard. A field named listing_id in one dataset may correspond to an identifier in another

table, while fields such as date, host_since, review_id, reviewer_id, or review_scores_rating may share partial lexical, semantic, or structural relationships without being directly equivalent. These challenges become more complex when documentation is incomplete, field descriptions are unavailable, and domain experts are not continuously involved in the integration process.

Traditional schema matching methods have relied on rule-based similarity measures, handcrafted mappings, ontology alignment, and supervised learning models. While these approaches are useful in controlled environments, they often require substantial manual preparation, labeled examples, domain-specific rules, or predefined ontologies. Such requirements limit their suitability for low-code data engineering pipelines, where data sources may change frequently and users may not have the technical knowledge needed to maintain complex mapping logic. At the same time, fully automated matching methods can produce plausible but incorrect mappings when they rely only on column names or surface-level similarity. This creates a need for lightweight, adaptive, and explainable schema matching approaches that can operate under low-supervision conditions.

Recent advances in semantic representation learning have created new opportunities for schema alignment. Sentence embedding models can capture contextual similarity between short text labels, while fuzzy string matching can identify lexical overlap, spelling variation, token rearrangement, and partial naming similarity. However, each technique has limitations when used alone. Embedding-based methods may identify broad semantic similarity but confuse related concepts that are not functionally equivalent. Fuzzy matching may perform well when column names share tokens, but it struggles when two fields have different names but related meanings. Data-type validation and keyword heuristics can improve reliability, but they need to be integrated into a coherent scoring mechanism. Therefore, a hybrid approach is more suitable for practical schema matching because it combines semantic, lexical, heuristic, and structural signals.

Explainability is another important requirement in low-code data integration. In many organizations, schema mappings are reviewed not only by data engineers but also by analysts, system owners, and business stakeholders. These users need to understand why a predicted correspondence is being recommended before trusting it in downstream workflows. A black-box matching score alone may not be sufficient, particularly when mappings affect reporting, operational decisions, or data governance processes. Natural-language justifications generated by instruction-tuned language models can help bridge this gap by explaining why two fields may be semantically related. However, such explanations should support human review rather than replace the underlying scoring method. For this reason, an explanation layer is most appropriate when it operates after the matching score is calculated, without directly changing the ranking logic.

This study proposes a lightweight explainable framework for schema matching and data fusion in low-code data engineering pipelines. The framework combines sentence embedding similarity, fuzzy string similarity, keyword-based heuristics, and data-type compatibility validation into a hybrid scoring model that ranks candidate correspondences between independently structured schemas. The method is unsupervised and does not require labeled training data, predefined ontology rules, or extensive manual mapping. An optional large language model based explanation layer generates natural-language justifications for predicted matches, improving interpretability for users who need to validate mappings before using them in integration workflows. The accepted schema correspondences can then support downstream data fusion by guiding join selection, duplicate field handling, provenance tracking, and unified schema construction.

The main contribution of this study is the design of a practical schema alignment framework that is suitable for low-code and low-supervision data engineering contexts. First, the study introduces a hybrid scoring method that combines semantic, lexical, heuristic, and data-type signals for schema matching. Second, it incorporates an optional explanation layer that provides natural-language justifications for predicted mappings without altering the scoring process. Third, it connects schema matching with data fusion preparation, showing how accepted mappings can support the construction of integrated analytical datasets. Fourth, it evaluates threshold-based matching behavior using precision, recall, and F1-score, highlighting the trade-off between broader mapping coverage and high-confidence alignment.

The study is evaluated using heterogeneous Airbnb Listings and Reviews datasets, which provide a realistic test case because the two files contain related but differently structured information. The Listings dataset includes property-level attributes such as host information, room type, price, location, availability, and review-related summary fields, while the Reviews dataset contains review-event attributes such as listing identifiers, reviewer identifiers, review identifiers, and timestamps. This structure creates a practical schema matching problem where some attributes are directly related, some are semantically adjacent, and others may appear similar while representing different entities. The experimental results demonstrate how the proposed framework behaves under moderate and strict similarity thresholds, providing insight into precision and recall trade-offs in low-supervision schema alignment.

This study positions schema matching as a practical foundation for explainable data fusion in low-code data engineering pipelines. By combining semantic similarity, lexical matching, heuristic validation, and optional natural-language justification, the proposed framework addresses both technical alignment and user interpretability. The work is intended to support low-supervision integration scenarios where heterogeneous datasets must be prepared for reliable downstream analytics without depending on extensive manual mapping, domain-specific ontologies, or labeled training data.

2. Related Work

2.1 Schema Matching in Data Integration

Schema matching has long been recognized as a foundational task in data integration because it enables heterogeneous data sources to be connected through meaningful attribute correspondences. Early studies framed schema matching as the process of identifying relationships between elements in independently designed schemas, where differences in naming, structure, granularity, and representation create barriers to integration [1], [2]. In enterprise and analytical environments, this task directly supports extract-transform-load workflows, data warehousing, master data management, entity resolution, and downstream data fusion. Although schema matching appears to be a technical preprocessing step, its quality strongly influences the reliability of integrated datasets, analytical outputs, and decision-support systems.

Traditional schema matching approaches relied heavily on syntactic similarity, structural comparison, domain rules, and manually designed heuristics. These methods commonly compare attribute names using edit distance, token overlap, prefix and suffix similarity, data-type compatibility, and schema hierarchy relationships [3]. Such techniques remain useful because they are computationally efficient, transparent, and easy to implement in practical data engineering workflows. However, they are limited when schemas use different terms for the same concept, when column names are abbreviated, or when semantic meaning cannot be inferred from surface-level text. For example, two columns may represent related business concepts even when they share few lexical tokens, while two columns may share similar words but represent different entities. These limitations motivate the need for hybrid matching strategies that combine lexical, semantic, and structural evidence.

2.2 Ontology-Based and Rule-Guided Matching

Ontology-based schema matching methods were developed to improve semantic alignment by using controlled vocabularies, conceptual hierarchies, and domain knowledge. These approaches can produce high-quality mappings when well-defined ontologies exist and when source schemas follow relatively stable domain conventions [3], [4]. Ontologies help disambiguate concepts, identify parent-child relationships, and distinguish between related but non-equivalent fields. They are especially useful in regulated or specialized domains where terminology is standardized, such as healthcare, finance, manufacturing, and scientific data management.

Despite their strengths, ontology-based methods are difficult to scale in dynamic low-code data engineering environments. Building and maintaining ontologies requires domain expertise, governance effort, and continuous updates as schemas evolve. Low-code users may not have access to formal ontologies, and many public or operational datasets contain weak metadata, inconsistent

naming, and incomplete documentation. In such settings, strict ontology dependence can reduce flexibility. Rule-guided systems face a similar limitation because handcrafted rules often work well within a known domain but become brittle when applied to new datasets. This creates a practical gap for lightweight schema matching methods that can operate without predefined ontologies while still producing interpretable results.

2.3 Machine Learning and Deep Learning Approaches

Machine learning methods introduced greater adaptability into schema and entity matching by learning patterns from labeled examples, historical mappings, or training pairs. Supervised models can capture complex relationships between fields by using lexical features, value distributions, contextual metadata, and structural properties [5]. Entity matching systems such as Magellan demonstrated the value of combining feature engineering, learning-based classification, and human review in large-scale data integration workflows [6]. These approaches improved matching accuracy in many settings and established a more data-driven direction for schema alignment research.

However, supervised methods are not always suitable for low-code environments because they require labeled training data, repeated model tuning, and technical expertise. In many real-world integration projects, ground-truth mappings are unavailable or too expensive to construct at scale. Deep learning methods further improved representation capacity, especially for entity matching and semantic similarity tasks, but they may introduce additional computational cost and reduced transparency [7]. For low-code data engineering pipelines, the matching method must be simple enough to deploy, understandable enough to validate, and flexible enough to handle new schemas without extensive retraining. This study therefore focuses on an unsupervised hybrid approach rather than a fully supervised or domain-trained model.

2.4 Semantic Similarity and Sentence Embeddings

Recent progress in representation learning has enabled schema matching methods to move beyond direct string comparison. Sentence embedding models represent short text labels as dense vectors, allowing semantic similarity to be estimated even when two attributes do not share identical tokens [8]. This is useful for schema matching because column names are often short, inconsistent, and context dependent. Embedding-based similarity can identify conceptual relationships between terms such as price and cost, review and rating, or date and temporal metadata, even when direct lexical overlap is limited.

At the same time, embedding-based similarity is not sufficient by itself. Short column names may not provide enough context for reliable semantic interpretation, and embeddings may group fields that are related but not functionally equivalent. For instance, reviewer_id and host_id may both appear as person-related identifiers, yet they refer to different roles in a platform transaction. Similarly, date and host_since are both temporal fields but do not represent the same event. These examples show that semantic similarity must be balanced with lexical evidence, keyword logic, and data-type validation. A hybrid approach can reduce the weaknesses of individual methods by combining their complementary strengths.

2.5 Data Fusion and Schema Alignment

Data fusion depends on accurate schema alignment because independently collected datasets must be connected through meaningful correspondences before they can be combined into a unified analytical view. In practical data engineering, fusion may involve joining related tables, resolving duplicate attributes, preserving source provenance, standardizing field names, and preparing integrated datasets for reporting or machine learning. If schema matches are incorrect, downstream fusion can introduce misleading relationships, duplicated fields, inconsistent joins, or analytical bias. Therefore, schema matching is not only a preliminary technical task but also a quality-control mechanism for reliable data fusion.

In low-code environments, data fusion often occurs through configurable connectors, visual transformation logic, and reusable pipeline components. These tools simplify workflow construction but do not eliminate the need for correct schema interpretation. Users still need guidance on which fields should be joined, which attributes are duplicates, which columns represent related but distinct

concepts, and which mappings require manual review. A lightweight schema matching framework can support this process by ranking candidate mappings, filtering them through thresholds, and providing explanations that help users make informed decisions before fusion is performed.

2.6 Explainability in Automated Schema Matching

Explainability is increasingly important in automated data integration because schema matching decisions can affect reporting accuracy, business intelligence, compliance processes, and operational analytics. A matching score may indicate that two fields are similar, but users often need to understand why the system made that recommendation. This is particularly important in low-code pipelines, where users may not be data integration specialists. Without explanation, automated matches can either be overtrusted or ignored, both of which reduce the practical value of the system.

Natural-language explanations provide a useful layer of interpretability by translating technical matching logic into human-readable reasoning. Instruction-tuned large language models can describe why two columns may be semantically related based on their names, roles, and possible contextual meanings. However, explanation generation should not replace the matching algorithm itself. Language models may produce plausible explanations even for weak or incorrect matches. For this reason, the explanation layer should operate after the hybrid score is calculated, serving as a review aid rather than a scoring authority. This separation preserves the transparency of the scoring model while improving usability for human validation.

2.7 Research Gap and Positioning of the Present Study

Existing schema matching research provides several useful foundations, including rule-based similarity, ontology alignment, supervised learning, entity matching systems, and embedding-based semantic comparison. However, these approaches do not fully address the needs of low-code data engineering pipelines. Rule-based methods are transparent but often shallow. Ontology-based methods are semantically strong but difficult to maintain across changing datasets. Supervised models can be accurate but require labeled data and training effort. Embedding-based methods improve semantic coverage but may confuse related concepts with true correspondences. In addition, many automated matching systems provide limited explanations for non-technical users [9].

The present study addresses this gap by proposing a lightweight explainable framework that combines semantic similarity, fuzzy string matching, keyword heuristics, and data-type compatibility into a single unsupervised scoring approach. The framework is designed for low-supervision integration settings where labeled mappings, formal ontologies, and extensive manual engineering may not be available. By adding an optional natural-language justification layer, the approach also supports human review and interpretability. This positions the framework as a practical bridge between automated schema matching and explainable data fusion in low-code data engineering pipelines.

3. Proposed Framework

3.1 Framework Overview

The proposed framework is designed to support lightweight and explainable schema matching in low-code data engineering pipelines. In practical integration environments, datasets are often created independently and may use different naming conventions, incomplete metadata, and inconsistent structural patterns. These differences make it difficult for users to identify reliable correspondences between fields without manual review. The proposed framework addresses this challenge by combining semantic similarity, fuzzy string similarity, keyword-based matching, and data-type compatibility into a single matching process.

The framework begins by profiling the input datasets to extract column names, inferred data types, missing value patterns, and basic structural characteristics. The column names are then normalized to reduce differences caused by capitalization, punctuation, underscores, spacing, and token formatting. After preprocessing, each source column is compared with each target column using a hybrid similarity score [10]. The highest-scoring candidate matches are ranked and filtered using a selected similarity threshold. To improve interpretability, an optional explanation layer generates

short natural-language justifications for selected matches, allowing users to review the results before using them in downstream data fusion workflows.

The overall framework can be represented as a pipeline consisting of input datasets, schema profiling, column normalization, hybrid similarity scoring, candidate ranking, explanation generation, and fusion-ready output. A framework diagram can be included in this section to visually show how raw datasets move through the matching process and produce validated schema correspondences for low-code data integration.

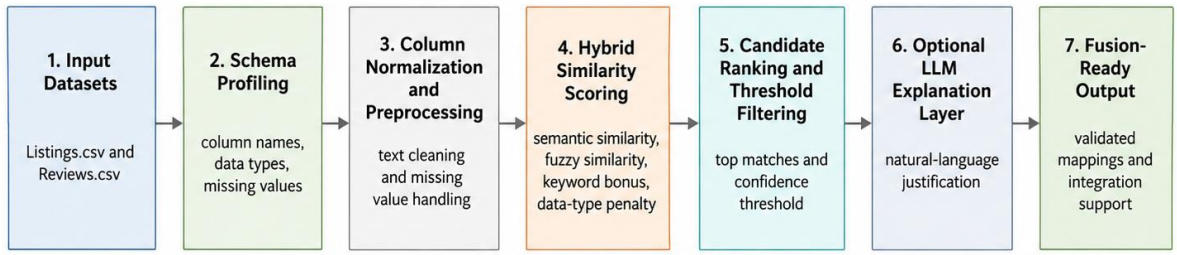


Figure 1. Proposed Lightweight Explainable Framework for Schema Matching and Data Fusion in Low-Code Data Engineering Pipelines.

3.2 Schema Profiling and Preprocessing

Let the source schema be represented as S_A and the target schema be represented as S_B. Each schema contains a set of column names extracted from the corresponding dataset.

$$S_A = \{a_1, a_2, \ldots, a_m\}$$

$$S_B = \{b_1, b_2, \ldots, b_n\}$$

In these equations, a_i represents a column from the source schema, and b_j represents a column from the target schema. For each column, the framework identifies the column name, inferred data type, missing value percentage, and basic value pattern. This profiling step helps distinguish between fields that may appear similar by name but differ in structure or meaning.

Before schema matching, column names are normalized to reduce formatting inconsistencies. Normalization includes lowercasing, removing special characters, replacing underscores with spaces, splitting compound terms, and removing unnecessary whitespace. The normalized forms of source and target column names are represented as follows.

$$a_{norm} = normalize\ (a)$$

$$b_{norm} = normalize\ (b)$$

Here, a and b are the original column names, while a_norm and b_norm are their normalized versions. This step improves matching quality because fields such as review_id, Review ID, and review identifier can be treated more consistently.

Missing values are also handled during preprocessing to support stable profiling and downstream analysis. For numerical fields, missing values are replaced with a sentinel value.

$$x\_clean = \{\frac{x,\ \ if\ x\ is\ available}{-1, if\ x\ is\ missing}$$

For categorical fields, missing values are replaced with a fixed label.

$$C\_clean = \{ \frac{c, \ if\ c\ is\ available}{unknown, \ if\ x\ is\ missing}$$

These preprocessing rules keep the pipeline simple and reproducible. They also preserve information about missingness without requiring complex imputation models.

3.3 Hybrid Schema Matching Score

The core of the framework is a hybrid schema matching score that estimates how strongly a source column corresponds to a target column. Instead of relying on a single matching method, the framework combines semantic similarity, fuzzy string similarity, keyword overlap, and data-type compatibility. This design is suitable for low-code data engineering pipelines because it remains lightweight while capturing both meaning-based and name-based relationships between fields.

Semantic similarity measures whether two column names have related meanings. After normalization, each column name is converted into a vector representation using a sentence embedding model. The semantic similarity between two columns is calculated using cosine similarity.

$$S_{sem}(a,b) = cos(e_a, e_b)$$

In this equation, e_a is the embedding vector for the source column, e_b is the embedding vector for the target column, and S_sem(a,b) represents the semantic similarity between the two columns.

Fuzzy string similarity captures lexical overlap between column names. It is useful when fields share similar words, abbreviations, or token patterns. The fuzzy matching score is normalized between 0 and 1.

$$S_{fuzz}(a,b) = \frac{F(a,b)}{100}$$

Here, F(a,b) is the fuzzy matching score between the two normalized column names.

A keyword bonus is added when two column names share meaningful tokens such as id, date, review, score, host, listing, price, or type. This helps the framework reward important field-level signals without depending entirely on semantic embeddings. A data-type penalty is applied when two fields have incompatible inferred data types. This reduces the likelihood of accepting matches that appear similar by name but differ structurally.

The final hybrid matching score is calculated as follows.

$$S_{final}(a,b) = \lambda S_{sem}(a,b) + (1-\lambda) S_{fuzz}(a,b) + B(a,b) - P(a,b)$$

In this equation, S_final(a,b) is the final matching score, S_sem(a,b) is the semantic similarity score, S_fuzz(a,b) is the fuzzy string similarity score, B(a,b) is the keyword bonus, P(a,b) is the data-type penalty, and lambda controls the balance between semantic similarity and fuzzy string similarity. A higher lambda value gives more importance to semantic meaning, while a lower lambda value gives more importance to lexical similarity.

This hybrid score allows the framework to identify candidate matches that are not only textually similar but also semantically meaningful and structurally reasonable. It also reduces the limitations of using only one method. For example, fuzzy matching may perform well when fields share common tokens, but it may fail when two fields have different names with related meanings. Similarly, semantic similarity may identify related concepts, but it may confuse fields that are similar in meaning but not equivalent in function. The hybrid score balances these signals to produce more reliable candidate mappings.

3.4 Candidate Ranking and Explanation Layer

After calculating the final score for all possible source-target column pairs, the framework ranks candidate matches for each source column. The highest-scoring target columns are treated as potential

matches. A similarity threshold is then applied to decide which mappings should be accepted for review or downstream use.

$$M_\tau = \{ (a,b) \mid S_{final}(a,b) \geq \tau \}$$

In this equation, M_tau represents the accepted mapping set, and tau represents the selected similarity threshold. A moderate threshold allows more candidate mappings to be identified, which improves coverage but may include weaker matches. A stricter threshold accepts fewer mappings, which improves confidence but may reduce recall. This threshold-based design is useful in low-code environments because users can adjust the level of strictness depending on the purpose of the integration workflow.

To improve interpretability, the framework includes an optional explanation layer. For selected mappings, a large language model generates a short natural-language justification explaining why two columns may be related. The explanation layer does not change the matching score or ranking. It only supports human review by making the predicted match easier to understand.

$$J(a,b) = LLM(a,b)$$

In this equation, J(a,b) represents the generated explanation for the candidate match between source column a and target column b. The explanation may describe whether the fields share a common identifier, temporal meaning, review-related context, pricing context, or other relevant semantic relationship.

This design keeps the framework lightweight while improving transparency. It is especially useful for low-code users who need to understand suggested mappings before applying them in a data integration workflow. The framework does not assume that every semantically related field is a correct match. Instead, it presents ranked candidate mappings and explanations so that users can make informed decisions before using the results for data fusion.

3.5 Summary of the Proposed Framework

The proposed framework provides a practical method for schema matching in low-code data engineering pipelines. It combines semantic similarity, fuzzy string matching, keyword-based signals, and data-type validation into a simple hybrid score. The candidate ranking and threshold mechanism allow users to control the balance between coverage and confidence. The optional explanation layer improves interpretability by providing natural-language reasoning for selected matches. Together, these components make the framework suitable for low-supervision integration scenarios where heterogeneous datasets must be aligned before data fusion, reporting, or analytical processing.

4. Data Fusion and Evaluation Design

4.1 Data Fusion Preparation

Schema matching becomes practically useful when the accepted mappings are used to support data fusion. In this study, data fusion is treated as a guided integration process rather than a fully automatic merge. This approach is important because two fields may appear semantically related without being directly equivalent. For example, two columns may both contain temporal information, but they may describe different events. Similarly, two identifier fields may both represent users, but they may refer to different roles within the dataset. Therefore, the framework uses schema matching to support fusion decisions while still allowing human review [11].

The accepted mapping set generated by the hybrid matching score is used to identify possible join fields, overlapping attributes, duplicate columns, and unified schema elements. These mappings help determine how related datasets can be connected and how their attributes can be organized for downstream analytics. The framework does not assume that every accepted match should automatically become a join condition. Instead, it provides a ranked and explainable set of candidate correspondences that can guide the fusion process.

The fused dataset can be represented in a simple form as follows.

$$D_{fused} = D_A \bowtie D_B$$

In this equation, D_A and D_B represent the input datasets, and D_fused represents the integrated dataset produced after applying validated fusion logic [12]. The join operation is guided by the accepted schema mappings, data-type compatibility, and field-level interpretation.

The fused schema can be represented as follows.

$$S_{fused} = S_A \cup S_B$$

In this equation, S_A and S_B represent the source and target schemas, while S_fused represents the unified schema created for downstream use. During fusion preparation, overlapping fields may be reviewed, renamed, retained, or consolidated depending on their meaning and analytical purpose.

This design keeps the fusion process controlled and transparent. It allows the framework to reduce manual effort while avoiding the risk of blindly merging fields that are only partially related. In a low-code pipeline, this approach can help users prepare datasets for reporting, dashboarding, analytical modeling, and ETL automation without requiring extensive manual schema mapping [13].

4.2 Experimental Dataset

The framework is evaluated using two related but independently structured tabular datasets: an Airbnb Listings dataset and an Airbnb Reviews dataset. The Listings dataset contains property-level information, including host attributes, room type, location, price, availability, amenities, and review-related summary fields. The Reviews dataset contains review-event information, including listing identifiers, reviewer identifiers, review identifiers, and review dates [14].

These datasets provide a useful evaluation case because they contain related information but follow different schema structures. Some fields have direct relationships, such as listing-level identifiers, while others are only partially related through broader semantic meaning. For example, review-related fields may share contextual similarity, while temporal fields may represent different types of dates. This creates a realistic schema matching challenge where the framework must distinguish between strong matches, weak matches, and misleading semantic associations.

The use of these datasets also supports the study's low-code data engineering focus. In practical integration workflows, users often work with independently collected datasets that were not designed with a common schema. The selected datasets reflect this type of scenario and allow the framework to be tested in a setting where schema alignment is necessary before reliable data fusion can occur.

4.3 Ground-Truth Mapping Design

To evaluate the schema matching results, predicted mappings are compared against a manually reviewed ground-truth mapping set. The ground-truth set defines which source and target columns are considered correct or acceptable matches based on field meaning, data type, and expected integration use. This manual review is necessary because schema matching cannot be evaluated only by surface-level name similarity. Some fields may look similar but represent different concepts, while others may have different names but support a valid integration relationship.

The ground-truth design separates direct matches from weaker semantic associations. A direct match occurs when two fields represent the same or highly equivalent concept. A weak semantic association occurs when two fields are related in meaning but should not be treated as fully interchangeable. This distinction is important because the goal of the framework is not only to find similar column names but also to support reliable data fusion[15].

For example, a listing identifier in the Reviews dataset may directly correspond to a listing identifier in the Listings dataset. In contrast, a reviewer identifier and a host identifier may both represent user-related identifiers, but they refer to different roles and should not be treated as a direct match. Similarly, two date fields may both contain temporal information but may describe different

events. These cases are useful for testing whether the framework can identify meaningful candidates while still requiring human validation for final fusion decisions.

4.4 Evaluation Metrics

The framework is evaluated using precision, recall, and F1-score. These metrics are suitable because schema matching is treated as a candidate selection task. The predicted mappings are compared against the manually reviewed ground-truth mappings to determine how many predicted matches are correct, how many correct matches are found, and how well the framework balances accuracy and coverage.

Precision measures the proportion of predicted matches that are correct.

$$Precision = \frac{TP}{TP + FP}$$

Recall measures the proportion of correct matches that are successfully identified.

$$Recall = \frac{TP}{TP + FN}$$

F1-score provides a balanced measure of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In these equations, TP represents true positives, FP represents false positives, and FN represents false negatives. A true positive is a predicted match that is correct according to the ground-truth mapping. A false positive is a predicted match that is not correct. A false negative is a correct match that the framework fails to identify.

Precision is important when the goal is to avoid incorrect mappings. Recall is important when the goal is to discover as many valid mappings as possible. F1-score is useful because it summarizes the balance between these two objectives. In low-code data engineering workflows, this balance is especially important because users may need either broad candidate discovery or stricter high-confidence matching depending on the integration scenario.

4.5 Threshold-Based Evaluation

The framework is evaluated using two similarity thresholds: 0.65 and 0.80. The threshold controls which candidate mappings are accepted after the final hybrid score is calculated. A candidate match is accepted when its final score is greater than or equal to the selected threshold.

$$S_{final}(a, b) \geq \tau$$

In this equation, S_final(a,b) represents the final hybrid matching score for a source-target column pair, and tau represents the selected threshold.

The moderate threshold of 0.65 is used to examine broader mapping coverage. This setting allows more candidate matches to pass through, which can be useful when users want to explore possible relationships between schemas. However, it may also allow more false positives because weaker matches are included.

The stricter threshold of 0.80 is used to examine high-confidence matching behavior. This setting accepts fewer candidate matches, but the accepted matches are expected to be more reliable. Such a threshold is useful in production-oriented workflows where incorrect mappings may affect reporting, analytics, or downstream integration quality.

The threshold-based evaluation helps show how the framework behaves under different confidence levels. It also demonstrates the practical trade-off between coverage and precision. In low-code environments, this flexibility is valuable because users may adjust the threshold depending on whether they are performing exploratory data preparation, controlled ETL design, or production data integration.

4.6 Evaluation Scope

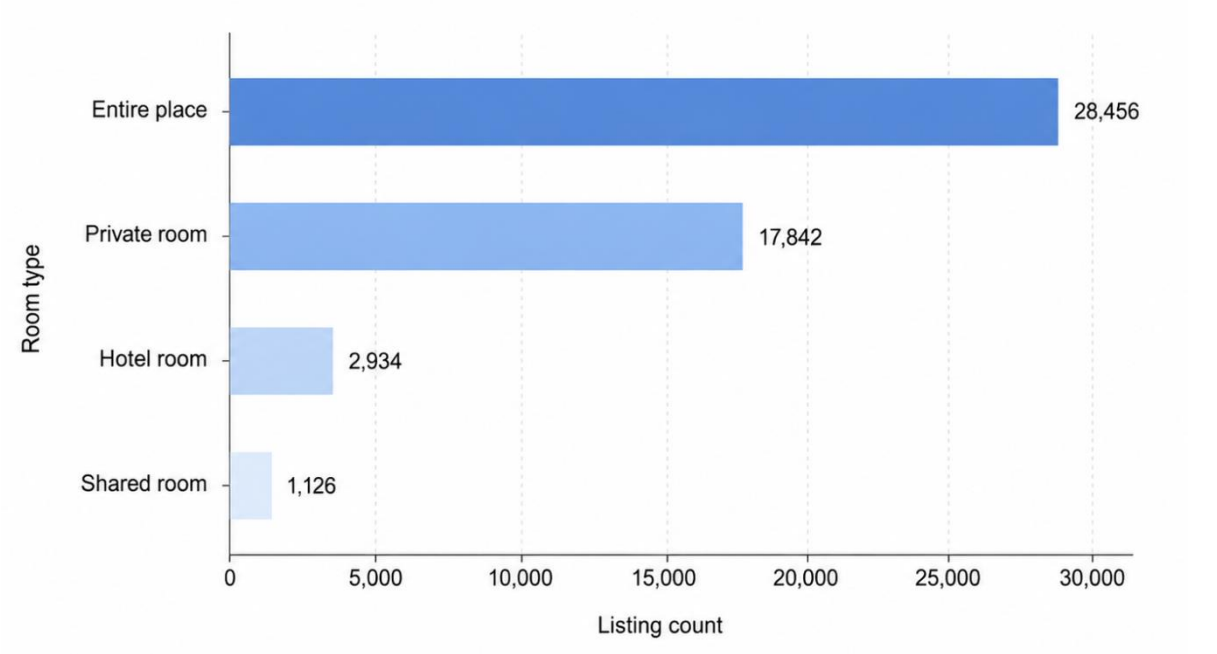
The evaluation focuses on whether the framework can identify meaningful schema correspondences between independently structured datasets while remaining simple, explainable, and suitable for low-code data engineering pipelines. The purpose is not to claim perfect automatic schema fusion. Instead, the evaluation examines whether the proposed method can reduce manual mapping effort, rank useful candidate correspondences, and provide interpretable outputs that support human validation.

The evaluation also considers the limitations of column-name-based matching. Short column names may not always contain enough context to determine exact equivalence. Some fields may be semantically related but not functionally interchangeable. For this reason, the framework combines similarity scoring with threshold filtering and optional explanation generation. This makes the method more practical for real integration workflows, where final mapping decisions often require both automated support and human judgment.

5. Results and Discussion

The proposed framework was evaluated using the Airbnb Listings and Reviews datasets to examine whether a lightweight hybrid matching approach can identify meaningful schema correspondences between independently structured tabular sources. The results show that the framework can produce useful candidate mappings by combining semantic similarity, fuzzy string similarity, keyword-based signals, and data-type compatibility. The findings also show that schema matching in low-code data engineering pipelines should not be treated as a fully automatic decision process, because some columns may be semantically related without being functionally equivalent fordata fusion.

Figure 2. Distribution of Room Types in the Airbnb Listings Dataset



The exploratory analysis confirmed that the Listings dataset contains several useful property-level and review-related attributes, while the Reviews dataset captures event-level review information. The room type distribution showed that entire-place listings formed the dominant category, followed by private rooms and other smaller categories. This distribution is useful because it confirms that the dataset contains meaningful variation across property types. The review score analysis showed that most room categories had relatively high average ratings, which reflects the skewed nature of review-based platform data. These patterns support the need for careful preprocessing before downstream analytical modeling.

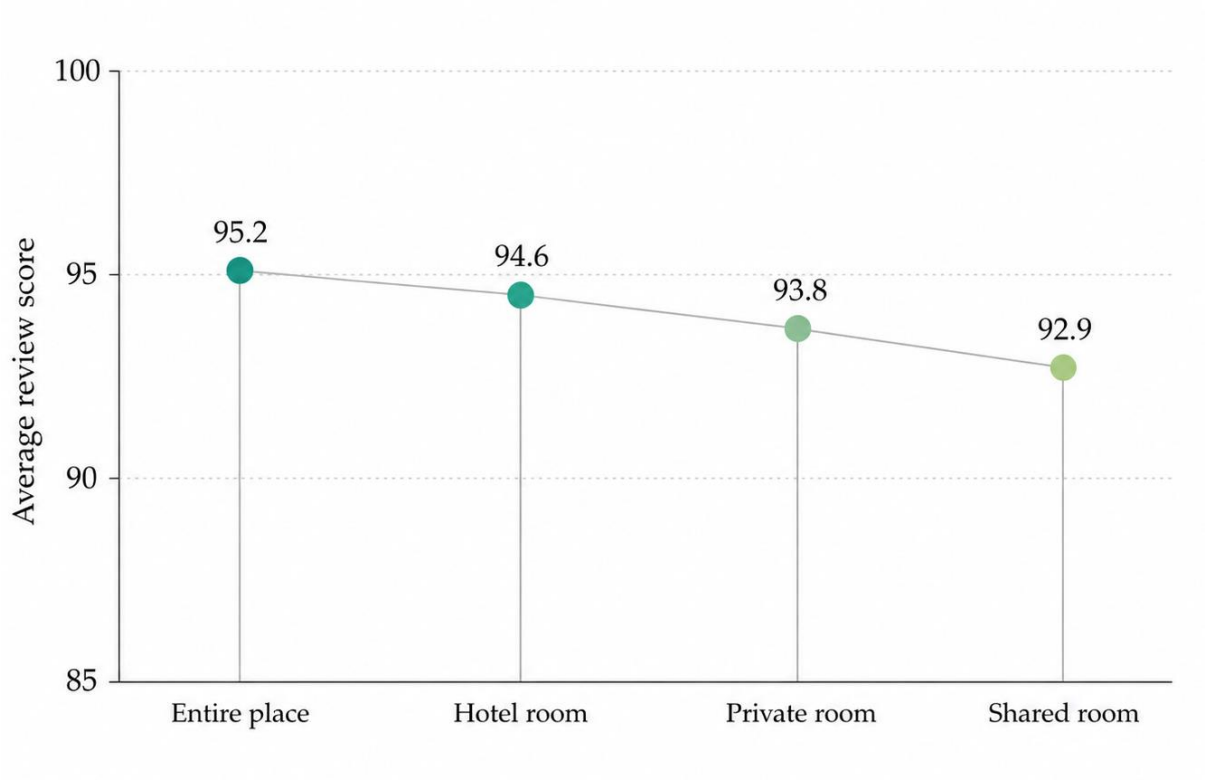


Figure 3. Average Review Score by Room Type in the Airbnb Listings Dataset

The correlation analysis of numerical attributes showed strong relationships among review-related variables such as cleanliness, communication, accuracy, and overall rating. This indicates that several review score fields may contain overlapping information. Such relationships are useful for understanding the dataset, but they can also create challenges during schema matching because columns with related meanings may appear highly similar even when they should not be treated as exact matches. The price distribution showed a right-skewed pattern, with a small number of high-value listings creating a long tail. This confirms the presence of outliers and suggests that normalization or outlier handling would be necessary for later modeling tasks.

The temporal review analysis showed that review activity changes over time, indicating that the Reviews dataset contains useful time-based information for downstream analytics. However, the presence of date-related fields also demonstrates a common schema matching challenge. A field such as date in the Reviews dataset may be semantically close to other temporal fields in the Listings dataset, but the meaning of each date must be interpreted carefully before fusion. A review date, host registration date, and last review date may all be temporal attributes, but they describe different events. This result supports the framework's use of explanation and human review rather than blind automated matching.

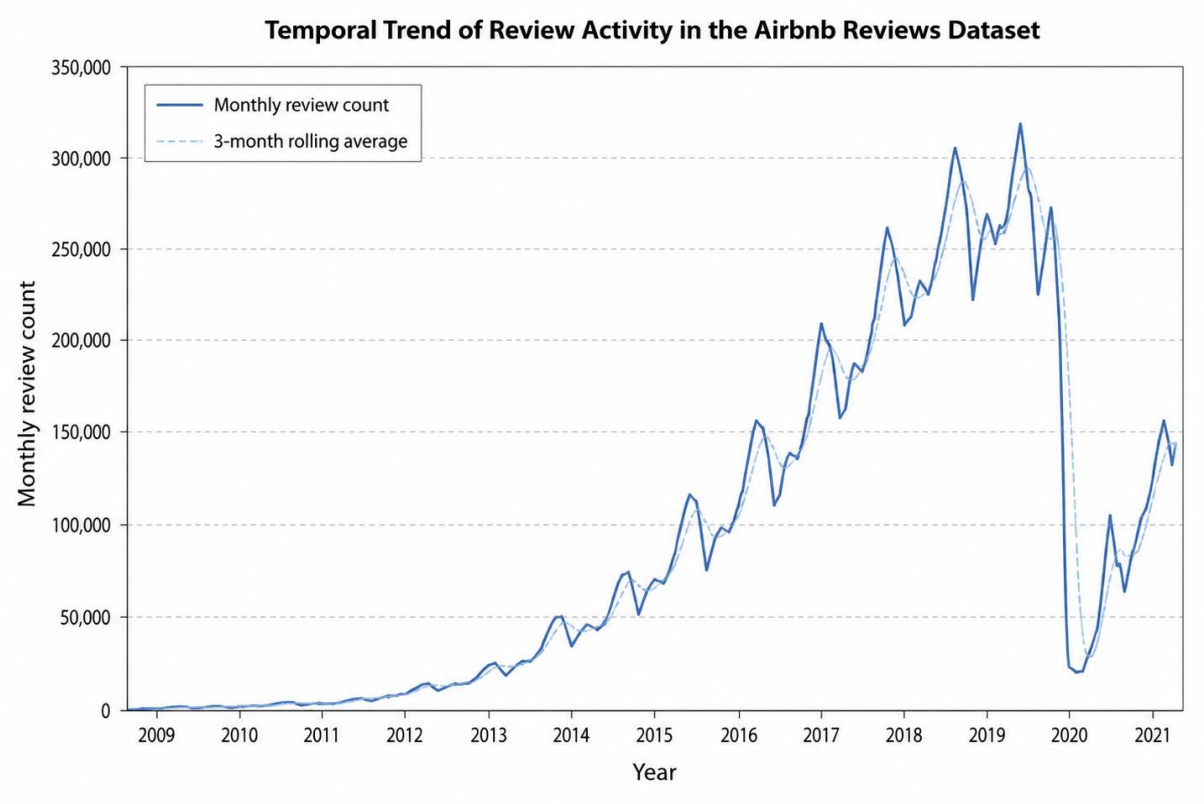


Figure 4. Temporal Trend of Review Activity in the Airbnb Reviews Dataset

The missing value analysis showed that some fields in the Listings dataset were incomplete, while one field was fully missing and therefore unsuitable for analysis. This confirms the importance of schema profiling and preprocessing before matching. Missing values can affect type inference, value pattern detection, and downstream fusion decisions. The use of simple imputation rules keeps the framework lightweight and reproducible, while still allowing missingness to be handled consistently during analysis.

The hybrid schema matching results showed that the framework was able to identify plausible correspondences between review-side columns and listing-side columns. The strongest and most useful match was associated with listing-level identifiers, because the listing identifier provides a direct connection between review records and listing metadata. This type of mapping is highly valuable for data fusion because it can support the construction of an integrated dataset that combines listing attributes with review activity. Other matches, such as review-related identifiers and review score fields, showed semantic similarity but required careful interpretation because they may represent related concepts rather than direct equivalents.

Table 1. Top candidate schema matches generated by the hybrid matching framework

Review Dataset Column	Listings Dataset Candidate Column	Similarity Score	Mapping Interpretation
listing_id	listing_id	0.98	Strong direct identifier match suitable for fusion review
review_id	review_scores_rating	0.88	Strong review-related semantic association
review_id	review_scores_communication	0.87	Strong review-related semantic association
review_id	review_scores_cleanliness	0.85	Strong review-related semantic association
review_id	review_scores_location	0.84	Strong review-related semantic association
review_id	review_scores_checkin	0.83	Strong review-related semantic association
review_id	review_scores_accuracy	0.82	Strong review-related semantic association
review_id	review_scores_value	0.81	Strong review-related semantic association
review_id	number_of_reviews	0.73	Review-related aggregate association
date	availability_365	0.60	Temporal or availability-related association requiring review
date	host_since	0.58	Temporal association requiring review

Review Dataset Column	Listings Dataset Candidate Column	Similarity Score	Mapping Interpretation
date	latitude	0.52	Weak candidate, likely not a valid direct match
date	longitude	0.52	Weak candidate, likely not a valid direct match
reviewer_id	host_id	0.61	User-identifier association, but different entity roles

The semantic similarity heatmap supported the usefulness of embedding-based comparison by showing stronger similarity values for conceptually related fields. However, the results also showed why embeddings should not be used alone. Some fields may appear close in semantic space because they share a broad context, such as review activity or user identity, but they may still represent different real-world entities. For example, reviewer_id and host_id may both be user-related identifiers, but they refer to different roles. This confirms the value of combining semantic similarity with fuzzy matching, keyword logic, type validation, and manual review.

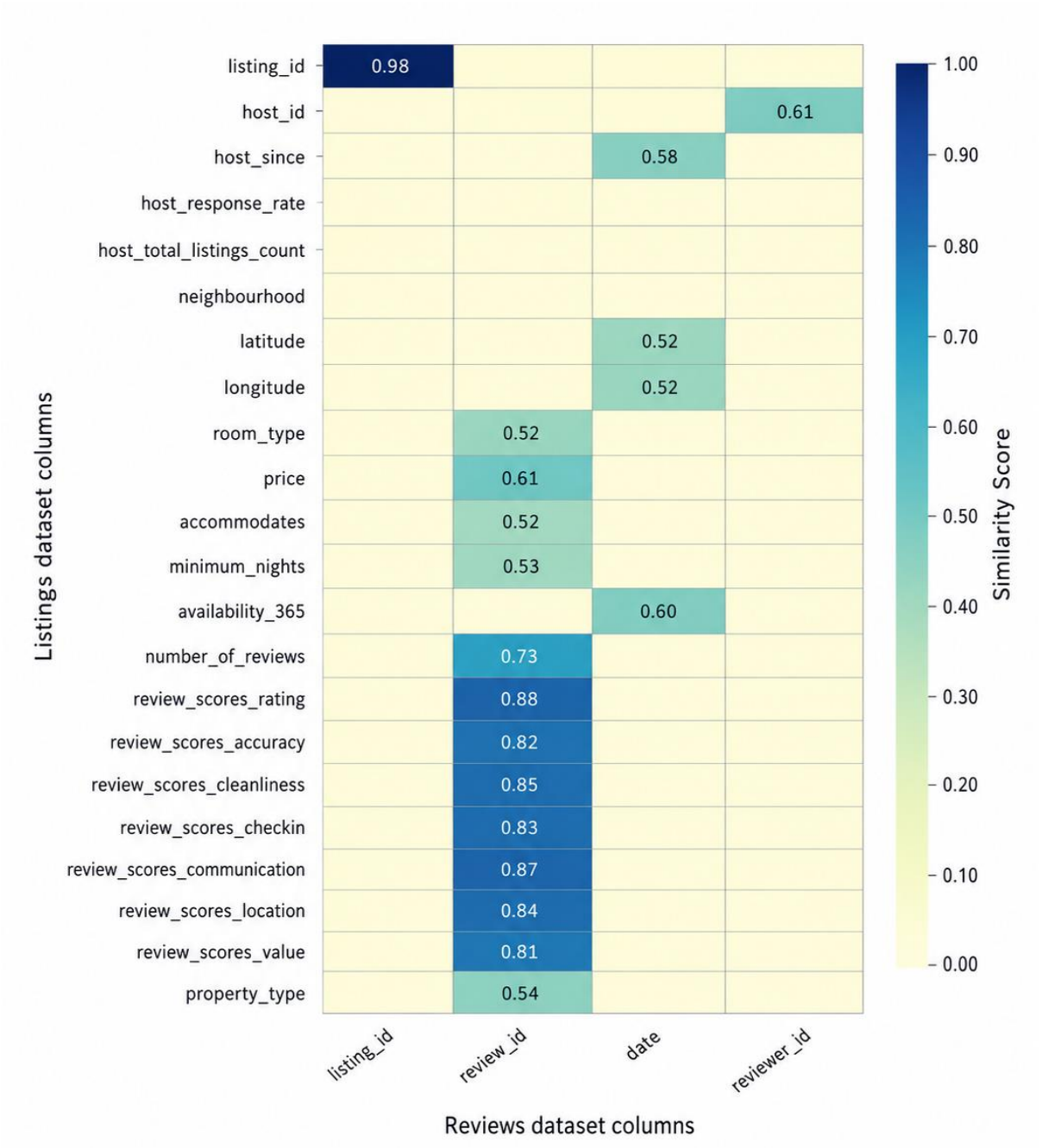


Figure 5. Semantic Similarity Heatmap for Candidate Schema Matches

The threshold-based evaluation showed a clear trade-off between broader candidate discovery and high-confidence matching. At a similarity threshold of 0.65, the framework achieved a precision of 0.50, recall of 0.33, and F1-score of 0.40. This threshold allowed more candidate mappings to pass through, which improved coverage but also introduced more false positives. At a stricter threshold of 0.80, the framework achieved a precision of 1.00, recall of 0.25, and F1-score of 0.40. This stricter threshold removed weaker matches and improved precision, but it also reduced the number of valid matches identified.

Table 2. Evaluation metrics by similarity threshold

Similarity Threshold	Precision	Recall	F1-score
0.65	0.50	0.33	0.40
0.80	1.00	0.25	0.40

The equal F1-score at both thresholds shows that the framework behaves differently depending on the selected confidence level. The moderate threshold supports exploratory schema matching because it provides more candidates for user review. The stricter threshold is more suitable for controlled or production-oriented workflows where incorrect mappings should be minimized. This flexibility is important in low-code data engineering because users may have different objectives depending on whether they are exploring data, preparing an ETL workflow, or validating a production integration.

The explanation layer improved the interpretability of predicted mappings by providing short natural-language justifications for selected column pairs. These explanations are useful because they help users understand why the framework suggested a match. However, the explanations should be treated as decision support rather than proof of correctness. A language model may explain why two fields are related, but the final mapping decision must still consider business meaning, data type, and intended fusion logic. This is especially important when two fields are semantically close but not directly interchangeable.

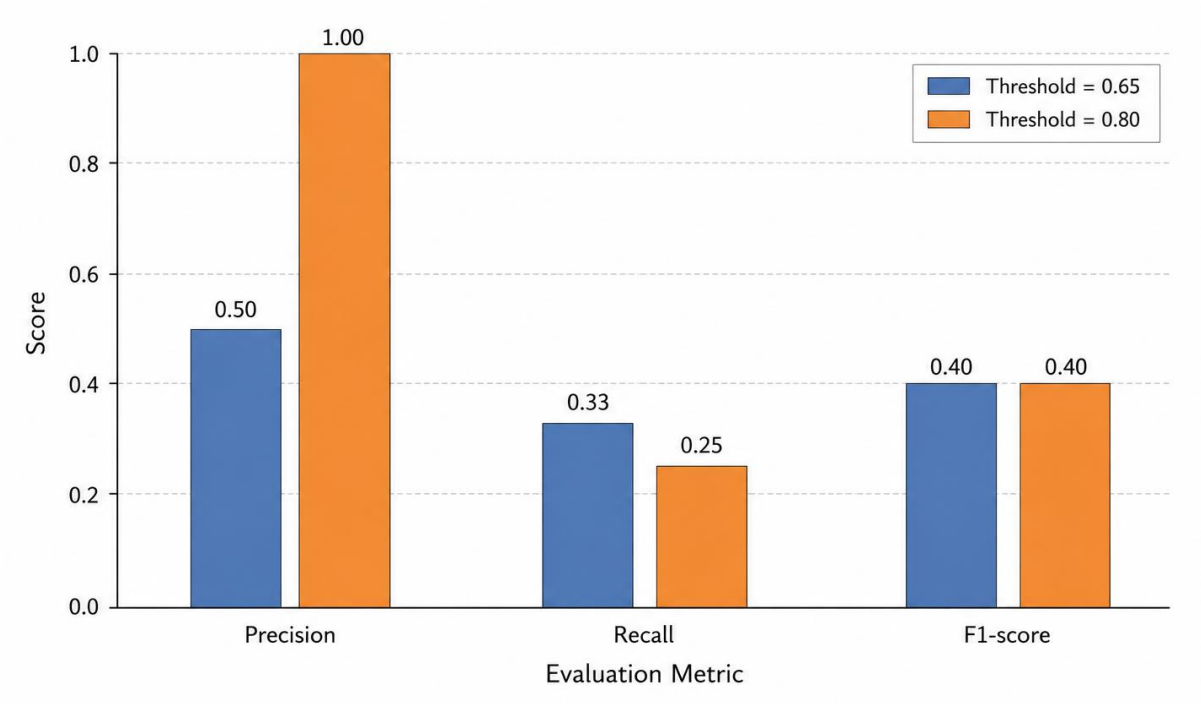


Figure 6. Threshold-Based Schema Matching Performance Comparison

The results also highlight several error patterns. Review-related fields can be incorrectly matched with other review score attributes because they share common tokens and semantic context. Identifier fields can be confused when they refer to different roles, such as reviewer and host. Temporal fields

can be matched because they all represent dates, even when each date describes a different event. These cases show that schema matching requires both similarity scoring and contextual validation. The proposed framework addresses this by ranking candidate matches, applying thresholds, and supporting explanation-based review.

Overall, the results indicate that the proposed framework is suitable as a lightweight support mechanism for schema matching and data fusion preparation in low-code data engineering pipelines. The framework does not claim to replace expert validation or guarantee perfect automated fusion. Instead, it reduces manual mapping effort by identifying likely correspondences, ranking candidate matches, and explaining why selected fields may be related. This makes the approach useful for practical integration scenarios where datasets are heterogeneous, metadata is limited, and users need a transparent method for preparing data before fusion and downstream analytics.

6. Practical Implications

The proposed framework has practical value for organizations that need to integrate heterogeneous datasets in low-code data engineering environments. Many modern data pipelines are built by combining information from multiple platforms, applications, files, and reporting systems [16]. These sources are often created independently and may not follow the same naming standards, metadata rules, or documentation practices. As a result, users must spend considerable time identifying which fields are related before they can build reliable ETL workflows, analytical tables, dashboards, or machine learning datasets. The proposed framework helps reduce this manual effort by producing ranked schema matching suggestions that can guide users during the early stages of data integration.

A key practical implication of the framework is its suitability for low-supervision environments. Many schema matching techniques require labeled training data, domain ontologies, or extensive rule engineering, which may not be available in fast-moving integration projects. In contrast, this framework uses lightweight signals that are commonly available in tabular datasets, including column names, inferred data types, keyword patterns, and semantic similarity. This makes the approach easier to deploy in low-code pipelines where users need practical recommendations without building complex machine learning models from scratch [17].

The framework also improves the usability of schema matching for non-specialist users. In low-code platforms, data preparation is not always performed by experienced data engineers. Business analysts, reporting teams, system administrators, and functional users may also participate in data integration work. For these users, a numerical similarity score alone may not be enough to make a confident decision. The optional explanation layer provides a short natural-language justification that helps users understand why two fields may be related. This makes schema matching more transparent and easier to review.

Another practical benefit is the framework's ability to support different levels of user confidence through threshold selection. During exploratory data preparation, users may prefer a moderate threshold that identifies more possible mappings for review. This can help them discover hidden relationships between datasets and accelerate early analysis. In production-oriented workflows, users may prefer a stricter threshold that accepts fewer but more reliable mappings. This flexibility allows the same framework to support both discovery-focused and quality-focused integration scenarios.

The framework can also support better decision-making during data fusion. Schema matching is not only about identifying similar column names. It directly affects how datasets are joined, how duplicate fields are handled, how unified schemas are created, and how downstream outputs are interpreted. By ranking candidate correspondences and providing explanation-based review, the framework helps users avoid careless merges between fields that are only superficially similar. This is especially important when two fields share a general meaning but represent different events, roles, or analytical concepts.

From an ETL design perspective, the framework can reduce development time during mapping preparation. Manual schema mapping is often repetitive and time-consuming, especially when integration teams must compare columns across multiple files or systems [18]. A lightweight

matching component can provide an initial set of candidate mappings that users can approve, reject, or refine. This can speed up pipeline design while still preserving human control over final mapping decisions. The result is a more efficient workflow that balances automation with validation.

The framework also has implications for data quality management. Incorrect schema mappings can create downstream issues such as wrong joins, duplicated attributes, misleading reports, and unreliable analytical models. By incorporating data-type compatibility and keyword-based validation into the matching process, the framework provides an additional quality check before data fusion occurs. While it does not eliminate the need for human review, it can help identify risky mappings earlier in the pipeline and reduce preventable integration errors.

Explainability also strengthens data governance. In many organizations, data integration decisions must be reviewed, documented, and maintained over time. When a mapping is accepted, users may later need to understand why it was created and whether it remains valid after schema changes. A framework that stores similarity scores, accepted mappings, thresholds, and explanations can provide a clearer audit trail for data engineering decisions. This is valuable for maintaining transparency, especially when integrated datasets support business reporting or operational decision-making.

The framework may also improve collaboration between technical and functional teams. Data engineers may understand pipeline logic, while business users may understand the meaning of fields and their correct usage. By presenting candidate mappings with readable explanations, the framework creates a shared review layer between these groups. Technical users can evaluate structural compatibility, while functional users can confirm business meaning. This collaborative validation can improve mapping accuracy and reduce misunderstandings during integration projects [19].

For low-code platform providers, the framework can be adapted as a reusable schema matching component. It could be embedded into visual pipeline builders, data preparation tools, mapping assistants, or integration templates. Users could upload or connect datasets, review ranked mapping suggestions, adjust thresholds, inspect explanations, and export validated mappings into the next stage of an ETL workflow. This would make schema alignment more accessible while still allowing expert users to refine the results when needed.

The framework also supports scalable data preparation practices. As organizations continue to adopt cloud applications, SaaS platforms, and self-service analytics tools, the number of datasets requiring integration continues to grow. Manual mapping alone may not be sustainable in such environments. A lightweight and explainable schema matching method can act as a first-pass recommendation layer, helping teams manage larger volumes of integration work without fully removing human oversight.

Overall, the practical value of the proposed framework lies in its ability to make schema matching faster, more understandable, and more adaptable for low-code data engineering pipelines. It does not replace expert judgment or guarantee fully automatic data fusion. Instead, it provides structured decision support by ranking likely matches, filtering them through thresholds, validating them through simple compatibility checks, and explaining them in user-friendly language. This makes the framework useful for real-world integration scenarios where transparency, simplicity, and human validation are as important as automation.

7. Limitations and Future Work

Although the proposed framework provides a practical and explainable approach for schema matching and data fusion in low-code data engineering pipelines, several limitations should be acknowledged. The first limitation is related to the scope of the experimental evaluation. The framework was evaluated using a specific pair of related tabular datasets, which is useful for demonstrating the method but not sufficient to represent the full complexity of schema matching across different domains. Real-world data integration environments may involve enterprise applications, financial systems, healthcare datasets, scientific repositories, customer platforms, and operational logs, each with different naming conventions, data structures, governance requirements, and semantic complexity.

A second limitation is the relatively small number of source attributes used in the matching task. When a schema contains only a limited number of columns, the number of possible candidate matches is manageable, and the evaluation may not fully reflect the difficulty of large-scale schema matching. In practical data engineering pipelines, schemas may contain hundreds or thousands of fields across multiple systems. As schema size increases, the number of possible column pairs grows rapidly, which may increase false positives, reduce interpretability, and require more advanced ranking, filtering, and validation strategies. Future work should therefore test the framework on larger and more complex schemas [20].

The framework also depends heavily on column names, inferred data types, and basic metadata. These signals are useful for lightweight matching, but they may not always provide enough context to determine true equivalence between fields. Some columns may have similar names but different meanings, while others may have different names but represent the same concept. For example, two date fields may both contain temporal values, but one may describe a review event while another may describe a host registration date. Similarly, two identifier fields may both refer to users but represent different roles. Future improvements should include value-level profiling, sample-record comparison, distributional similarity, and relationship analysis to better distinguish related fields from truly equivalent fields.

Another limitation concerns the use of sentence embeddings for semantic similarity. Embedding models are useful for identifying conceptual relationships between short text labels, but column names are often too brief to provide sufficient context. A short label such as id, date, score, type, or name can have different meanings depending on the dataset and business process. This can cause the model to assign high similarity to fields that are semantically close but not suitable for direct fusion. Future research should explore context-enriched embeddings that combine column names with sample values, table names, field descriptions, and neighboring attributes to improve semantic precision.

The threshold-based filtering method is simple and practical, but it also introduces sensitivity to parameter selection. A moderate threshold may increase coverage by allowing more candidate mappings, but it may also include weaker or incorrect matches. A stricter threshold may improve confidence but can miss valid correspondences. This trade-off was visible in the results, where the stricter threshold improved precision while reducing recall. Future work should investigate adaptive threshold selection, where the threshold changes based on schema size, score distribution, field type, user preference, or integration risk level.

The optional explanation layer also has limitations. Natural-language explanations can improve user understanding, but they should not be treated as evidence that a match is correct. A language model may generate a convincing explanation even when the underlying mapping is weak, incomplete, or only partially valid. For this reason, the explanation layer should remain a decision-support feature rather than an automatic validation mechanism. Future research should make explanations more evidence-grounded by linking them directly to measurable factors such as similarity score, keyword overlap, data-type compatibility, missing value profile, and value-level examples.

The current framework treats data fusion mainly as a guided preparation process. It identifies candidate correspondences, possible join fields, overlapping attributes, and unified schema elements, but it does not fully automate conflict resolution, duplicate handling, provenance management, or post-fusion quality assessment. In real integration pipelines, data fusion may require additional rules for handling conflicting values, preserving original field sources, validating joins, managing duplicates, and checking whether the fused dataset remains analytically reliable. Future work should expand the fusion component to include these capabilities.

Human-in-the-loop validation is another important direction for future development. In low-code environments, users should be able to approve, reject, edit, or annotate suggested mappings. Their feedback can then be used to improve future recommendations without requiring a fully supervised training process from the beginning. A feedback-driven version of the framework could learn user preferences, preserve mapping decisions, support audit trails, and improve consistency across

repeated integration workflows. This would make the framework more useful in organizational environments where data pipelines evolve over time.

Future work should also evaluate the framework against stronger baselines. The present framework combines semantic, fuzzy, keyword, and type-based signals, but additional experiments should compare it with fuzzy-only matching, embedding-only matching, rule-based matching, ontology-assisted matching, and supervised or semi-supervised schema matching models. Such comparisons would provide a clearer understanding of the contribution of each component. An ablation study could also show how performance changes when individual components, such as keyword bonus, type penalty, or explanation support, are removed.

Another future direction is the development of a low-code user interface for the framework. Such an interface could display ranked mappings, similarity scores, explanations, confidence thresholds, sample values, and fusion recommendations in a review-friendly format. Users could adjust thresholds, inspect evidence, approve mappings, and export the validated schema alignment for downstream ETL workflows. This would make the framework more practical for analysts and data engineers who work in visual data preparation environments.

Overall, the proposed framework should be viewed as a foundation for lightweight and explainable schema matching rather than a complete solution for every data integration scenario. Its main strength is that it provides a transparent, configurable, and low-supervision approach for identifying candidate correspondences and preparing datasets for fusion. Future research should focus on broader multi-domain evaluation, larger schemas, value-level evidence, adaptive thresholds, stronger explanation grounding, human feedback, and more advanced fusion validation. These improvements would increase the reliability, scalability, and practical usefulness of explainable schema matching in modern low-code data engineering pipelines

8. Conclusion

This study presented a lightweight explainable framework for schema matching and data fusion in low-code data engineering pipelines. The framework was developed to address a practical challenge found in many real-world integration environments, where datasets are often created independently, follow different naming conventions, contain limited documentation, and provide incomplete metadata. In such settings, reliable schema alignment becomes difficult because users must determine whether fields from different sources represent the same concept, a related concept, or a misleading surface-level similarity. The proposed framework responds to this challenge by offering a simple, transparent, and configurable approach that supports schema matching without requiring labeled training data, predefined ontologies, or complex supervised model development.

The central contribution of the study is the hybrid schema matching score, which combines semantic similarity, fuzzy string similarity, keyword-based signals, and data-type compatibility. This combination is important because no single matching signal is sufficient for reliable schema alignment in heterogeneous datasets. Semantic similarity helps identify fields with related meanings, fuzzy matching captures lexical overlap and naming variation, keyword signals reward important domain-relevant tokens, and data-type validation reduces structurally incompatible matches. By bringing these signals together, the framework provides a more balanced method for ranking candidate correspondences than approaches that rely only on column name similarity or embedding-based comparison.

The study also emphasized the importance of explainability in schema matching. In low-code data engineering pipelines, schema alignment decisions are often reviewed by users with different levels of technical expertise. A numerical similarity score may help rank candidate matches, but it may not be enough for users to understand why a match was suggested or whether it should be trusted. The optional explanation layer addresses this issue by generating short natural-language justifications for selected mappings. These explanations support human review and make the matching process more transparent, while still keeping the scoring logic separate from the language model output.

The experimental findings showed that the framework can identify meaningful candidate mappings while revealing the expected trade-off between mapping coverage and matching

confidence. At a moderate threshold of 0.65, the framework achieved a precision of 0.50, recall of 0.33, and F1-score of 0.40. This setting allowed more candidate mappings to be considered, which can be useful during exploratory data preparation. At a stricter threshold of 0.80, the framework achieved a precision of 1.00, recall of 0.25, and F1-score of 0.40. This setting produced fewer but more reliable accepted mappings, making it more appropriate for controlled integration scenarios where false positives must be minimized.

These results suggest that threshold selection plays an important role in practical schema matching. A lower or moderate threshold may be suitable when users want to discover possible relationships across datasets and review them manually. A stricter threshold may be better when users need high-confidence matches before building production-oriented ETL or analytical workflows. The unchanged F1-score across the two threshold settings also shows that schema matching performance should not be interpreted through one metric alone. Instead, precision, recall, threshold behavior, and the intended use case should be considered together.

The study further demonstrated that schema matching should be treated as a decision-support process rather than a fully automatic replacement for human validation. Some fields may appear similar because they share words, data types, or semantic context, but they may not be equivalent for data fusion. For example, two date fields may both represent temporal information while describing different events, and two identifier fields may both refer to users while representing different roles. These cases show why explainability, threshold control, and human review are necessary when schema matching outputs are used to guide data fusion.

From a practical perspective, the proposed framework can support low-code data engineering by reducing manual mapping effort and improving the transparency of integration workflows. Accepted mappings can help users identify possible join fields, review duplicate or overlapping attributes, prepare unified schema structures, and organize datasets for downstream reporting, dashboarding, or analytical modeling. Because the framework is lightweight and does not depend on heavy model training, it can be adapted as a reusable component within low-code or semi-automated ETL pipelines.

Overall, this study provides a practical foundation for explainable schema matching and data fusion preparation in low-supervision environments. The framework does not claim to solve all schema integration challenges, but it offers a clear and useful direction for building transparent, configurable, and user-friendly schema alignment methods. Future improvements can extend the framework with larger multi-domain evaluations, value-level profiling, adaptive threshold selection, human feedback loops, stronger fusion validation, and provenance-aware schema generation. These extensions would further improve the reliability and applicability of lightweight explainable schema matching in modern data engineering pipelines.

References

[1] Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. The VLDB Journal, 10(4), 334–350. https://doi.org/10.1007/s007780100057

[2] Doan, A., Domingos, P., & Halevy, A. Y. (2001). Reconciling schemas of disparate data sources: A machine-learning approach. Proceedings of the ACM SIGMOD International Conference on Management of Data, 509–520. https://doi.org/10.1145/375663.375731

[3] Melnik, S., Garcia-Molina, H., & Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. Proceedings of the 18th International Conference on Data Engineering, 117–128. https://doi.org/10.1109/ICDE.2002.994702

[4] Giunchiglia, F., & Shvaiko, P. (2004). Semantic matching. The Knowledge Engineering Review, 18(3), 265–280. https://doi.org/10.1017/S0269888904000074

[5] Aumueller, D., Do, H. H., Massmann, S., & Rahm, E. (2005). Schema and ontology matching with COMA++. Proceedings of the ACM SIGMOD International Conference on Management of Data, 906–908. https://doi.org/10.1145/1066157.1066283

[6] Bilke, A., & Naumann, F. (2005). Schema matching using duplicates. Proceedings of the 21st International Conference on Data Engineering, 69–80. https://doi.org/10.1109/ICDE.2005.126

[7] Halevy, A. Y., Ashish, N., Bitton, D., Carey, M. J., Draper, D., Pollock, J., Rosenthal, A., & Sikka, V. (2005). Enterprise information integration: Successes, challenges and controversies. Proceedings of the ACM SIGMOD International Conference on Management of Data, 778–787. https://doi.org/10.1145/1066157.1066246

[8] Do, H. H., & Rahm, E. (2007). Matching large schemas: Approaches and evaluation. Information Systems, 32(6), 857–885. https://doi.org/10.1016/j.is.2006.09.002

[9] 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

[10] Bleiholder, J., & Naumann, F. (2008). Data fusion. ACM Computing Surveys, 41(1), 1–41. https://doi.org/10.1145/1456650.1456651

[11] Bernstein, P. A., Madhavan, J., & Rahm, E. (2011). Generic schema matching, ten years later. Proceedings of the VLDB Endowment, 4(11), 695–701. https://doi.org/10.14778/3402707.3402710

[12] Shvaiko, P., & Euzenat, J. (2013). Ontology matching: State of the art and future challenges. IEEE Transactions on Knowledge and Data Engineering, 25(1), 158–176. https://doi.org/10.1109/TKDE.2011.253

[13] Cer, D., Yang, Y., Kong, S. Y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., & Kurzweil, R. (2018). Universal Sentence Encoder for English. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 169–174. https://doi.org/10.18653/v1/D18-2029

[14] Konda, P., Das, S., Suganthan, P. G. C., Doan, A., Ardalan, A., Ballard, J. R., Li, H., Panahi, F., Zhang, H., Naughton, J. F., Prasad, S., Krishnan, G., Deep, R., & Raghavendra, V. (2016). Magellan: Toward building entity matching management systems over data science stacks. Proceedings of the VLDB Endowment, 9(13), 1581–1584. https://doi.org/10.14778/3007263.3007314

[15] Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., & Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. Proceedings of the ACM SIGMOD International Conference on Management of Data, 19–34. https://doi.org/10.1145/3183713.3196926

[16] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2018). A survey of methods for explaining black box models. ACM Computing Surveys, 51(5), 1–42. https://doi.org/10.1145/3236009

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

[18] Sahay, A., Indamutsa, A., Di Ruscio, D., & Pierantonio, A. (2020). Supporting the understanding and comparison of low-code development platforms. Proceedings of the 46th Euromicro Conference on Software Engineering and Advanced Applications, 171–178. https://doi.org/10.1109/SEAA51224.2020.00036

[19] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence: Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, 58, 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[20] Luo, Y., Liang, P., Wang, C., Shahin, M., & Zhan, J. (2021). Characteristics and challenges of low-code development: The practitioners' perspective. Proceedings of the ACM/IEEE

International Symposium on Empirical Software Engineering and Measurement, 1–11. https://doi.org/10.1145/3475716.347578