A Unified Multi-Scenario Framework for Schema Matching based on LLM

***Abstract*—Schema matching is pivotal in data integration, identifying semantic equivalences between attributes in structured data tables. Existing methods focus on schema-level or instance-level matching, excelling only in specific scenarios. This underscores the need for a unified approach adaptable to varying data availability conditions. This paper presents a unified schema matching framework based on large language models (LLMs) that effectively addresses schema matching tasks across three distinct scenarios: a Scenario with only MetaData (SMD), a Scenario with a Small amount of instance Data (SSD), and a Scenario with a Large amount of instance Data (SLD).**

The framework employs a two-step approach, integrating feature similarity computation with LLM prompt engineering. The first step involves comprehensive feature extraction and similarity analysis to efficiently filter potential matches, followed by scenario-specific LLM prompting for precise semantic matching. Our approach integrates textual, structural, and semantic features with LLM-based analysis using LLaMA3.1-8b as its base model. Extensive experiments on WikiData and MIMIC-III datasets validate the framework's effectiveness, achieving an F1 score of 92.5 and precision of 95.8 in the SLD scenario, while maintaining robust performance in scenarios with limited data. Ablation studies confirm that this two-step approach enhances both efficiency and accuracy, validating the framework's capability to address diverse schema matching requirements within a unified solution.

***Index Terms*—Schema Matching, Large Language Models, Feature Similarity Computation, Prompt Engineering, Multi-Scenario**

I. Introduction

Currently, data integration and retrieval play a pivotal role in information management and data analysis [1]. Data integration facilitates the amalgamation of data from diverse sources and formats, thereby enabling the discovery and acquisition of valuable information for data analysis and decision-making through information retrieval techniques. Schema matching is a fundamental problem across various database application domains, including data integration, data warehousing, and semantic query processing [2]. In essence, schema matching entails identifying a mapping between common elements of two schemas or ontologies [3].

Over the past three decades, researchers in this field have developed and published numerous approaches to achieve schema matching. These include rule-based methods [4], semantic similarity approaches [5]–[8], machine learning models [9], [10], and deep learning techniques [11]–[14]. However, these methods exhibit significant limitations when applied to diverse scenarios. Traditional similarity-based approaches struggle with semantic understanding across varying representations [7], while machine learning methods typically require substantial amounts of training data and computational resources [9], [13]. Furthermore, existing methods primarily focus on either metadata-only matching or instance-based matching, thereby lacking the flexibility to handle multiple scenarios effectively.

To address these challenges, we propose a novel two-step framework that integrates feature similarity computation with LLM-based schema matching. The first step utilizes comprehensive feature extraction and similarity analysis to efficiently filter potential matches, while the second step employs scenario-specific LLM prompting for precise semantic matching decisions. This approach effectively addresses three distinct scenarios: Schema Matching with Metadata Only (SMD), Schema Matching with Sample Data (SSD), and Schema Matching with Large Data (SLD).

Our framework offers several key contributions:

- A unified approach that effectively handles multiple schema matching scenarios through a two-step process
- An efficient feature similarity computation mechanism that reduces the search space for LLM processing
- A novel integration of LLM-based matching that employs scenario-specific prompting strategies
- Comprehensive evaluation demonstrating the framework's effectiveness across different scenarios

The subsequent sections of this paper are organized as follows: Section II presents the problem definition and our proposed framework overview. Section III discusses related work in schema matching. Section IV details our feature similarity computation approach. Section V describes the LLM-based schema matching process. Section VI presents experimental results and analysis. Finally, Section VII concludes the paper and discusses future work.

II. Problem and Solution Overview

A. Problem Statement

Schema matching is fundamentally defined as the process of identifying semantically equivalent elements across two or more schemas. These equivalences are established through various characteristics, such as attribute names, data types, structural constraints, and instance data patterns. In this study, we specifically focus on semantic equivalence relations within schema matching.

Contemporary schema matching systems encounter several significant challenges. First, due to increasingly stringent data privacy and security regulations, many organizations are unable to freely share their complete datasets, leading

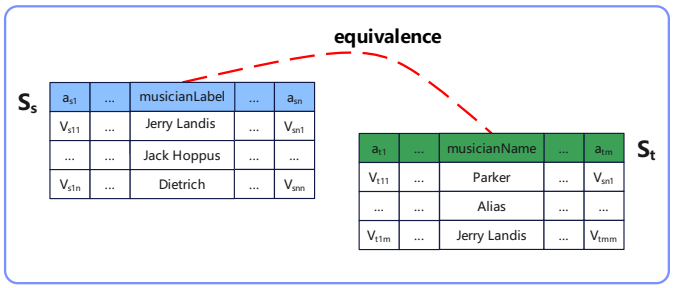

Fig. 1. An example of a schema matching task in an LSD scenario

to varying levels of data accessibility during the matching process. Second, the diversity of data representations and formats necessitates advanced matching techniques capable of handling both metadata-level and instance-level analysis. Third, the computational complexity involved in processing large-scale datasets demands efficient matching strategies that can maintain accuracy while managing resource constraints.

Based on the current problem, we formally define the schema matching problem as follows:

Definition 1 (Schema Matching Equivalence Relations)**.** *Let S_s and S_t be a source schema and a target schema with attributes A_s and A_t, respectively. Each attribute $a_{si} \in A_s$ and $a_{tj} \in A_t$ is characterized by:*

1) *A metadata tuple $(a.name, a.desc)$ containing the attribute name and description.*
2) *An optional set of attribute values:*

$$V_{si} = \{v_{si1}, v_{si2}, \ldots, v_{sim}\},$$
$$V_{tj} = \{v_{tj1}, v_{tj2}, \ldots, v_{tjn}\}.$$

The schema matching function is defined as:

$$m : A_s \times A_t \to \{true, false\}$$

where $m(a_{si}, a_{tj}) = true$ if and only if a_{si} is semantically equivalent to a_{tj}. The matching process adapts to three scenarios:

- ***SMD:** Only the metadata tuple $(a.name, a.desc)$ is considered for matching, with $|V_{si}| = |V_{tj}| = 0$.*
- ***SSD:** Both metadata and limited instance data are utilized, where $|V_{si}|, |V_{tj}| \leq 5$.*
- ***SLD:** Both metadata and complete attribute value sets V_{si} and V_{tj} are used for comprehensive matching analysis.*

This formalization establishes the foundation for our two-step matching framework, which combines feature similarity computation with LLM-based semantic analysis to address the challenges across all three scenarios effectively.

An example of an LSD scenario is illustrated in Fig. 1, where the attribute a_{si} of S_s is musicianLabel, and the attribute a_{tj} of S_t is musicianName. These attributes share identical content and convey the same meaning but have different names.

B. A Unified Multi-scenario Framework for Schema matching based on LLM

To address the challenges in schema matching across diverse scenarios, we propose a novel two-stage framework integrating traditional feature similarity computation with LLM-based semantic analysis. As illustrated in Fig. 2, our framework comprises two primary components: feature similarity computation and LLM-based schema matching. This framework unifies the handling of three distinct scenarios: Scenario with only Metadata (SMD), Scenario with Small instance Data (SSD), and Scenario with Large instance Data (SLD).

The feature similarity computation stage, illustrated in Fig. 2(a), comprises three primary feature processing components. The text features module processes column names and descriptions, generating TF-IDF vectors to capture lexical similarities efficiently. The structural features module analyzes data types and statistical patterns within schema attributes. The semantic features module focuses on embeddings and content analysis when instance data is available. These features are then integrated through a similarity analyzer that employs scenario-specific combination strategies.

The schema matching stage, as depicted in Fig. 2(b), employs an LLM to determine final matching decisions. This stage uses the similarity scores and features from the previous stage to construct scenario-specific prompts. For SMD scenarios, prompts emphasize metadata information. SSD scenarios incorporate limited instance samples, and SLD scenarios integrate comprehensive statistical information. The LLM processes these prompts to determine attribute equivalence, producing final schema matching results.

This unified framework achieves both computational efficiency and robust semantic understanding capabilities. The feature similarity computation stage provides efficient initial analysis, while the LLM-based matching stage enables sophisticated semantic reasoning. The detailed implementation methods and technical specifications of each stage are elaborated in the following sections.

III. Related Works

In this section, we review related work in the field of schema matching, encompassing schema-based and instance-based methods, and compare them with the methods proposed in this paper.

Schema-level methods: Cupid [7] integrates semantic and structural information of schemas as its core concept. Similarity Flooding [8] converts schemas into graphs and identifies matching pairs through iterative similarity propagation. Clio [15] leverages schema information and constraints to map data across sources. LSD [16] employs methods like Naive Bayes to train a learner for identifying elemental mappings between source and mediator schemas. SMAT [11] is a deep learning-based solution using attention mechanisms. LSM [17] fine-tunes a pre-trained BERT model [18]. Jellyfish [19] is a large language model designed for data preprocessing, supporting schema-level matching tasks.

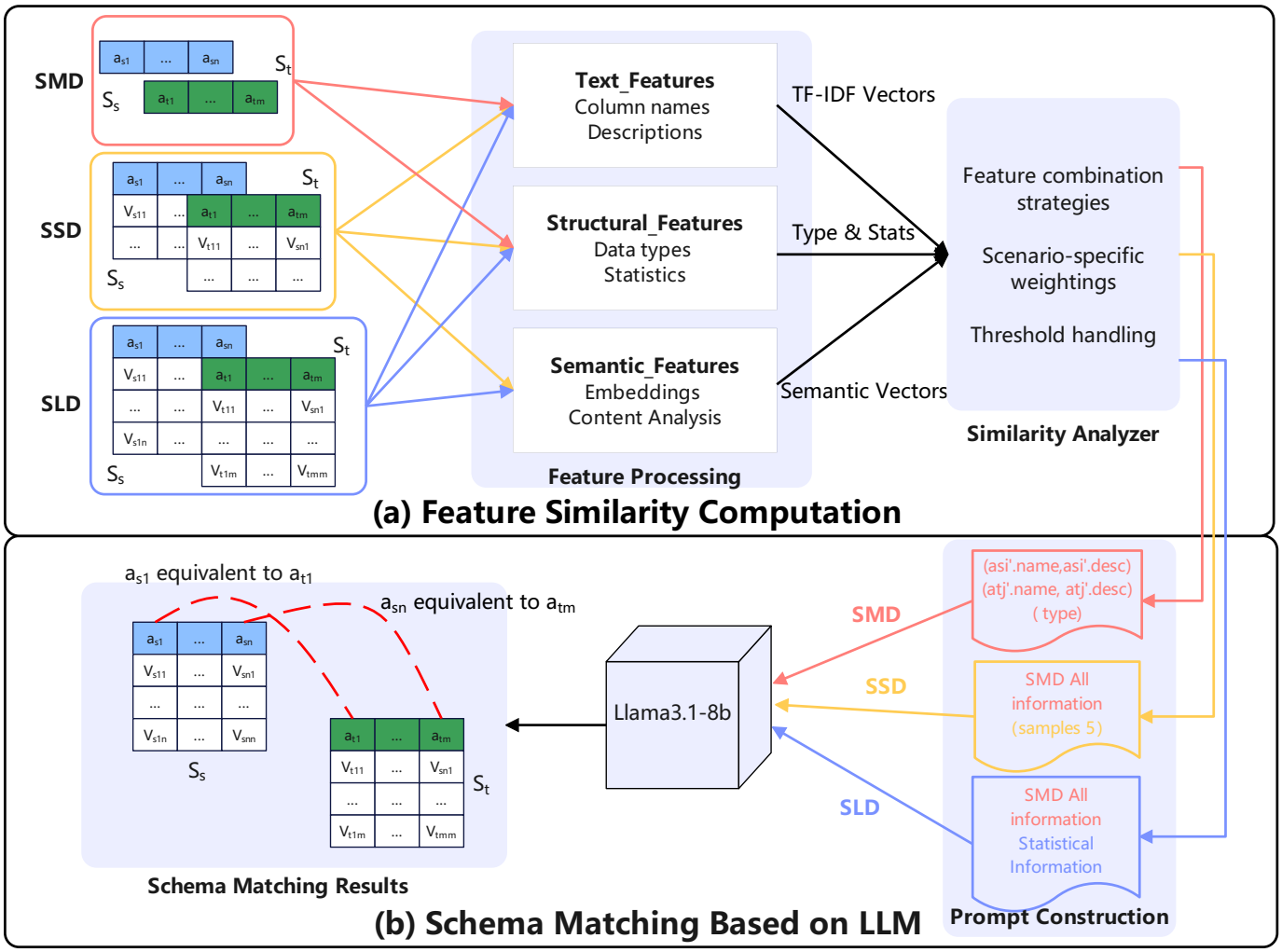

Fig. 2. A unified schema matching framework based on LLM for SMD, SSD, and SLD scenarios. (a) Feature similarity computation stage processes input schemas and generates comprehensive similarity scores. (b) Schema matching stage leverages LLM for semantic analysis and final matching decisions.

Instance-level methods: Edhy et al. [4] compute attribute similarity using instance data for heterogeneous databases. QuickMig [20] extends COMA++ [21] by introducing three instance-based matching algorithms that generate correspondences between source and target schemas along with associated mapping classes. Zhang et al. [22] identify relationships between columns by analyzing common attributes and value features. Valentine [6] combines a value-overlap-based method with other open-source matching techniques. Ramnandan et al. [5] employ ontology-aligned data to predict candidate semantic labels. Sahay et al. [9] use feature engineering and clustering to perform schema matching. Madhavan et al. [23] train multiple learners with domain-specific corpora to assist in schema matching. Both EmbDI [13] and REMA [12] use local embedding algorithms to compute cosine distances for attribute matching.

As discussed above, schema matching research primarily focuses on schema-based or instance-based methods, which are inadequate for solving multi-scenario schema matching tasks with a unified approach. Some solutions leverage pre-trained models and LLMs to address schema matching problems, but most remain exploratory in nature. In contrast, our approach employs Jellyfish, a fine-tuned Llama 2 model, integrating semantic computing and prompt engineering techniques to provide unified schema matching solutions across three scenarios in a single framework.

IV. Feature Similarity Computation

Feature similarity computation forms the foundation of our schema matching framework, enabling efficient preprocessing for subsequent LLM-based matching. Given the source schema S_s and the target schema S_t, with attributes A_s and A_t, we compute comprehensive similarity scores using a scenario-adaptive approach.

For each attribute $a_{si} \in A_s$ and $a_{tj} \in A_t$, we extract three feature types: text features (e.g., attribute names and descriptions), structural features (e.g., data types and statistics), and semantic features derived from instance values when available.

The computation strategy is scenario-dependent (SMD, SSD, or SLD) and varies with attribute data types.

In metadata-based similarity computation, we use a weighted combination of multiple similarity measures:

$$\begin{aligned} Sim_{name}(a_{si}, a_{tj}) = &\alpha_1 \cdot Sim_{LEV}(a_{si}.\text{name}, a_{tj}.\text{name}) \\ &+ \alpha_2 \cdot Sim_{seq}(a_{si}.\text{name}, a_{tj}.\text{name}) \\ &+ \alpha_3 \cdot Sim_{JAC}(T_{si}, T_{tj}), \end{aligned} \tag{1}$$

where Sim_{LEV} represents the Levenshtein similarity, Sim_{seq} denotes sequence similarity calculated using `difflib`, Sim_{JAC} is the Jaccard similarity between token sets T_{si} and T_{tj} derived from the attribute names, and $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

For numerical attributes in scenarios with instance data, we compute similarity based on statistical distributions:

$$\begin{aligned} Sim_{numeric}(a_{si}, a_{tj}) = &\beta_1 \cdot Sim_{mean}(V_{si}, V_{tj}) \\ &+ \beta_2 \cdot Sim_{std}(V_{si}, V_{tj}) \\ &+ \beta_3 \cdot Sim_{quantile}(V_{si}, V_{tj}), \end{aligned} \tag{2}$$

Where V_{si} and V_{tj} represent the value sets, and similarities are calculated using normalized differences of statistical measures, satisfying $\beta_1 + \beta_2 + \beta_3 = 1$.

For categorical or string attributes, a hybrid similarity measure is utilized:

$$\begin{aligned} Sim_{cat}(a_{si}, a_{tj}) = &\gamma_1 \cdot Sim_{JAC}(V_{si}, V_{tj}) \\ &+ \gamma_2 \cdot Sim_{dist}(V_{si}, V_{tj}), \end{aligned} \tag{3}$$

Where Sim_{dist} denotes distribution similarity based on value frequencies, satisfying $\gamma_1 + \gamma_2 = 1$.

The scenario-specific computation process is formalized in Algorithm 1, which accepts source and target schemas along with scenario configurations as input and produces a similarity matrix as output. For each attribute pair, name-based similarity (NS) is computed first. Subsequently, depending on the scenario type, structural similarity (SS) and instance-based similarity (IS) are incorporated with scenario-specific weights. Detailed similarity computation methods are described in preceding sections.

Here, $\mathbf{W}$ denotes the scenario-specific weight vector. The $ComputeInstanceSim$ function adapts its computation strategy based on the scenario type, employing sample-based similarity for SSD and distribution-based similarity for SLD. The $Combine$ function aggregates different similarity scores based on scenario-specific weights. For numerical attributes, it employs statistical distribution comparison, while for categorical attributes, it utilizes the hybrid similarity measure described in the equations above.

The algorithm illustrates how our framework tailors its similarity computation strategy to diverse scenarios while ensuring computational efficiency. The evaluation section describes the precise weight combinations and threshold values used in our implementation.

Algorithm 1 Feature Similarity Computation

Require: S_s, S_t, scenario, $\mathbf{W}$
Ensure: Similarity matrix Sim

$Sim \leftarrow InitializeMatrix(|A_s|, |A_t|)$
for $a_i \in A_s, a_j \in A_t$ **do**
  $NS \leftarrow ComputeNameSim(a_i, a_j)$
  if scenario = SMD **then**
    $Sim[i, j] \leftarrow NS$
  else
    $SS \leftarrow ComputeStructSim(a_i, a_j)$
    $IS \leftarrow ComputeInstanceSim(a_i, a_j, scenario)$
    $Sim[i, j] \leftarrow Combine(NS, SS, IS, \mathbf{W})$
  end if
end for
return Sim

V. Schema Matching Based on LLM

After computing feature similarities, our framework utilizes large language models to make final matching decisions through structured prompts and adaptive processing mechanisms. Fig. 3 illustrates our unified prompt template design, which ensures consistency across scenarios while addressing scenario-specific requirements.

A. Prompt Template Design

The prompt template incorporates four essential components to guide the LLM's matching decisions. The System Message and Task Description components define the model's role and matching objectives, offering consistent foundational context across all scenarios. The Instance Content component adapts to each scenario: SMD emphasizes metadata (attribute names, types, and descriptions), SSD supplements metadata with sample values, and SLD integrates comprehensive statistical information, including distribution patterns and value summaries. The Question section delivers scenario-specific analysis guidance, highlighting different aspects ranging from metadata interpretation in SMD to statistical pattern analysis in SLD.

B. Matching Decision Mechanism

Our framework employs a two-stage matching process that integrates similarity analysis with LLM-based semantic understanding. The initial stage filters candidate pairs based on computed similarity scores and scenario-specific thresholds. These filtered candidates then undergo LLM analysis via constructed prompts that incorporate relevant features and contextual information. The process ensures robust validation by verifying both semantic equivalence and structural constraints, including one-to-one matching relationships.

C. Scenario Adaptation

The framework adapts to varying scenarios through mechanisms of information integration and decision confidence. Information integration determines how various types of evidence are weighted during the matching process, ranging from

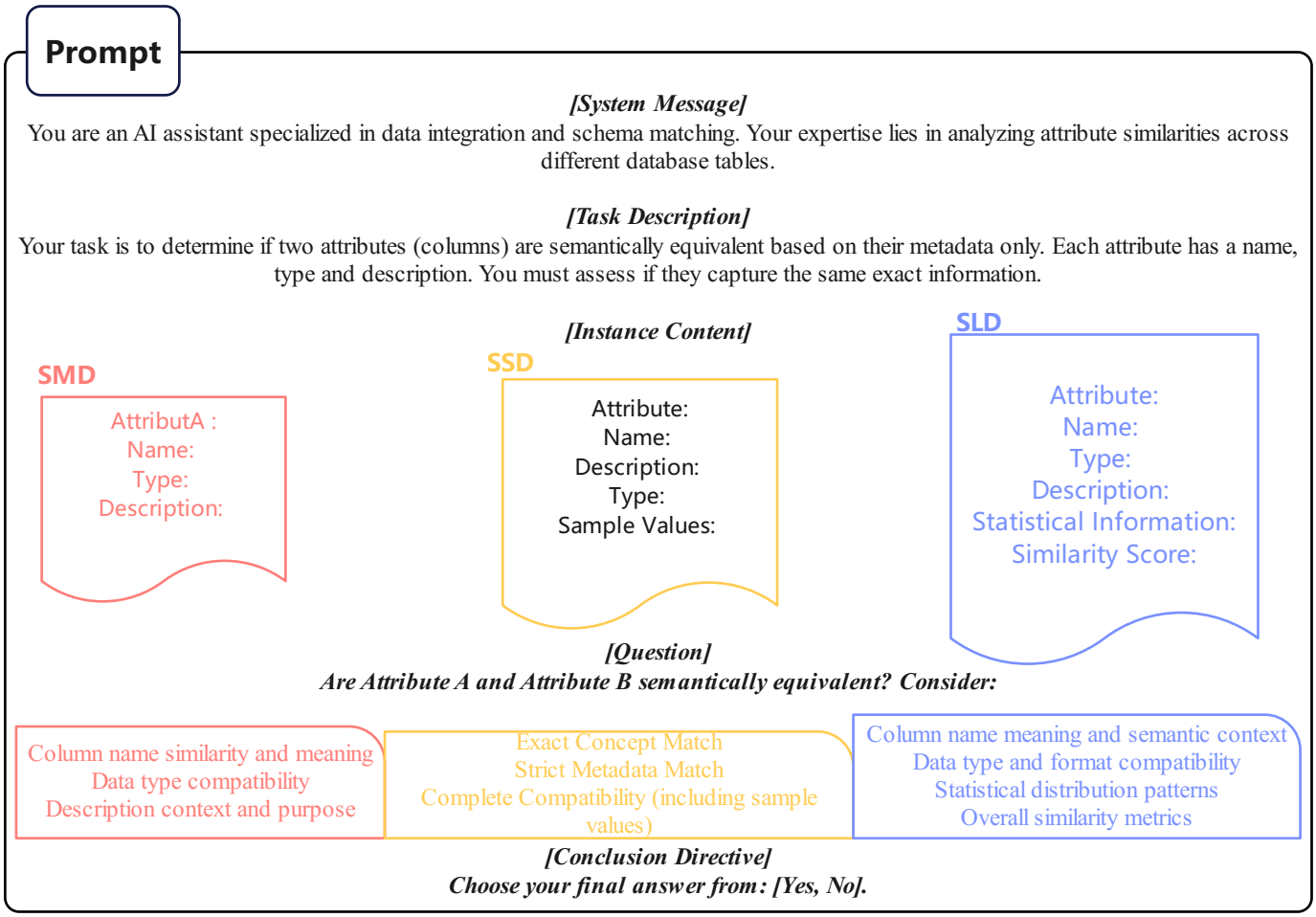

Fig. 3. Unified prompt template structure for different scenarios. The template consists of shared components (System Message, Task Description) and scenario-specific Instance Content and Question considerations.

metadata-focused analysis in SMD to comprehensive statistical evidence in SLD. The decision confidence mechanism adjusts thresholds and validation criteria according to available information, ensuring robust matching decisions across all scenarios while preserving a consistent output format for systematic evaluation.

VI. Evaluation

A. Experimental Setup

Datasets. We evaluated our framework using two datasets: the WikiData datasets from Valentine [6] and the MIMIC-III [24] dataset for OMOP mapping. WikiData, a prominent knowledge base supporting Wikimedia projects, features four variants: Unionable, View-Unionable, Joinable, and Semantically-Joinable, each representing distinct association scenarios. The MIMIC-III dataset provides a real-world healthcare schema matching case, mapping a clinical database to the standardized OMOP data model. Due to insufficient metadata in the WikiData dataset, we employed GPT-4o to generate the necessary metadata, followed by rigorous manual validation to ensure accuracy and consistency.

TABLE I
Dataset Statistics for Benchmark Evaluation

Dataset	# Tables	Avg # Rows	# Columns	# Matched
WikiData	8	9490	120	40
MIMIC to OMOP	38	0	725	250

Framework Configuration. Table II outlines our framework's configuration across various scenarios. We employed paraphrase-MiniLM-L3-v2 [25] for embedding computation to optimize efficiency. Feature weights are dynamically adjusted based on data availability in each scenario, emphasizing name similarity for SMD and semantic features for SLD. Similarity thresholds are calibrated through empirical validation. For instance sampling, 3 samples are used for SSD to capture essential patterns, and 100 samples for SLD to ensure statistical representation. Caching is enabled for SSD and SLD scenarios to enhance computational efficiency when processing larger datasets.

Model Configuration. We utilized LLaMA3.1-8B [26] as our primary model, deploying it with Ollama [27] for inference. The model was configured with the following parame-

TABLE II
SCENARIO-SPECIFIC CONFIGURATION PARAMETERS

Parameter	SMD	SSD	SLD
Feature Weights			
Name	0.6	0.5	0.4
Structural	0.4	0.3	0.2
Semantic	0.0	0.2	0.4
Similarity Thresholds			
Name	0.5	0.5	0.4
Structural	0.6	0.6	0.6
Semantic	0.0	0.45	0.4
Combined	0.5	0.5	0.4
Other Settings			
Sample Size	-	3	100
Cache Enabled	False	True	True

ters: temperature = 0.35, top_p = 0.9, top_k = 10 for optimal generation quality, max_tokens = 2048, and repeat_penalty = 1.1 to ensure response coherence.

Evaluation Metrics. We used standard information retrieval metrics, including Precision, Recall, and F1 Score, to assess matching performance.

Baseline Methods. Our framework was compared with state-of-the-art methods across various scenarios. GPT-4O-mini [28], an optimized version of OpenAI's GPT-4 models, was used. Notably, it was excluded from SLD evaluation due to input token limitations.

- SMD: CUPID, Similarity Flooding, COMA-S, and GPT-4O-mini
- SSD: COMA-E, Distribution-based matching, Jaccard-Levenshtein similarity, and GPT-4O-mini
- SLD: COMA-E, Distribution-based matching, Jaccard-Levenshtein similarity, and EmbDI

B. Experimental Evaluation

Table III provides a comprehensive evaluation of our framework across various scenarios and datasets, showcasing consistent performance advantages and adaptability to diverse data availability conditions.

In the SMD scenario with WikiData, our framework achieves significant improvements, with a precision of 89.36 and a recall of 73.12, surpassing traditional methods such as CUPID (precision: 59.13, recall: 42.92) and Similarity Flooding (precision: 62.17, recall: 63.96). The framework also demonstrates superior performance compared to GPT-4O-mini (precision: 85.67, recall: 65.45), particularly in scenarios with limited metadata. For the MIMIC to OMOP dataset, our approach delivers robust performance (precision: 46.53, recall: 43.63), despite the heightened complexity of healthcare schema matching.

In the SSD scenario, our framework exhibits balanced performance, achieving a precision of 81.77 and a recall of 84.58. While COMA-E achieves perfect precision of 100.00, its lower recall of 72.91 results in a reduced overall F1 score. Our framework maintains competitive performance with an F1 score of 82.82, effectively integrating limited instance data with metadata analysis. The performance closely matches GPT-4O-mini (F1 score: 82.15), demonstrating the robustness of our approach.

In the SLD scenario, our framework delivers its strongest performance, achieving the highest precision of 95.82 among all methods and an F1 score of 92.52. Notably, while EmbDI achieves comparable performance (F1 score: 92.06), our framework excels in managing large-scale instance data. GPT-4O-mini could not be evaluated in this scenario due to input size limitations.

Across all scenarios, our unified framework demonstrates consistent performance advantages while ensuring computational efficiency. It particularly excels in the SLD scenario, where abundant instance data enables more effective relationship identification. Among all compared methods, only COMA and our framework support all three scenarios, with our approach demonstrating superior overall performance, except for marginally lower precision in the SSD scenario against COMA-E. The framework's consistent performance across WikiData and MIMIC-III datasets highlights its versatility and robustness in addressing diverse schema matching tasks.

C. Ablation Study

To comprehensively evaluate our framework's effectiveness, we conducted three sets of ablation studies: feature combination analysis, model comparison, and scenario performance analysis.

Feature Analysis. We first assessed the contribution of different feature combinations to matching performance, as illustrated in Fig. 4. The experimental results indicate that textual features alone result in limited performance (F1: 65.8, Precision: 68.5), while structural features in isolation deliver slightly better effectiveness (F1: 73.6, Precision: 76.6). The combination of textual and structural features results in substantial improvement (F1: 82.4, Precision: 85.5), indicating strong complementarity between these feature types. The integration of all features, including semantic analysis, delivers the highest performance (F1: 92.5, Precision: 95.8), emphasizing the importance of comprehensive feature utilization.

Model Comparison. We assessed different LLM variants to evaluate the impact of model selection, as illustrated in Fig. 5. LLaMA3.1-8b demonstrates the best overall performance (F1: 79.9, Precision: 89.4), closely followed by Gemma2-9b (F1: 78.5, Precision: 88.2). Mistral-7b-instruct and Qwen2-7b exhibit relatively lower performance, with F1 scores of 75.2 and 72.5, respectively. This comparison indicates that model architecture and pre-training approaches significantly influence matching accuracy, with LLaMA3.1-8b's architecture being particularly well-suited for schema matching tasks.

Scenario Analysis. Performance across various scenarios reveals notable patterns, as illustrated in Fig. 6. The framework demonstrates improved effectiveness as more data becomes available, with the rich-instances scenario achieving

TABLE III
EXPERIMENT RESULTS ON WIKIDATA AND MIMIC-III DATASETS

Dataset	Scenario	Method	Recall	Precision	F1 Score
WikiData	**SMD**	COMA-S	63.96	89.99	68.56
		CUPID	42.92	59.13	62.86
		Similarity Flooding	63.96	62.17	69.76
		GPT-4O-mini	65.45	85.67	74.12
		Schema Matching LLM	**73.12**	**89.36**	**79.87**
	SSD	COMA-E	72.91	100.00	83.54
		Distribution-based	51.88	73.33	58.64
		Jaccard-Levenshtein	68.96	82.92	71.18
		GPT-4O-mini	85.37	78.90	82.15
		Schema Matching LLM	**84.58**	**81.77**	**82.82**
	SLD	COMA-E	84.79	91.67	77.49
		Distribution-based	61.88	66.72	67.01
		Jaccard-Levenshtein	85.83	68.48	76.91
		EmbDI	92.76	91.37	92.06
		GPT-4O-mini	N/A	N/A	N/A
		Schema Matching LLM	**89.45**	**95.82**	**92.52**
MIMIC to OMOP	**SMD**	COMA-S	27.25	23.24	22.32
		CUPID	6.16	56.76	10.89
		Similarity Flooding	3.20	100.00	6.16
		GPT-4O-mini	45.83	47.67	46.73
		Schema Matching LLM	**43.63**	**46.53**	**45.03**

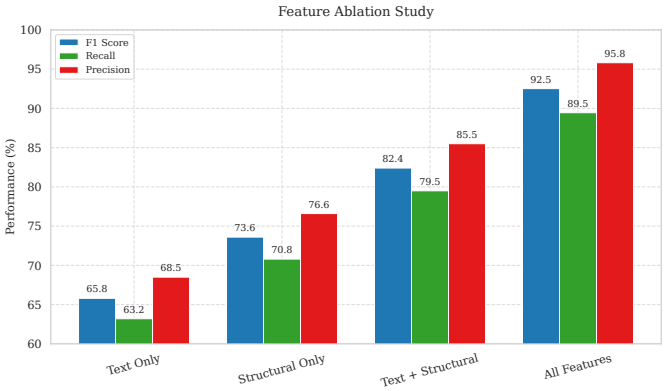

Fig. 4. Feature ablation study results comparing different feature combinations.

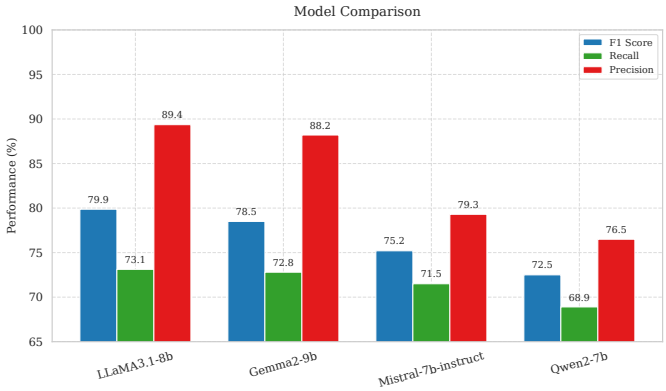

Fig. 5. Performance comparison of different LLM variants.

the highest performance (F1: 92.5, Precision: 95.8). The few-instances scenario exhibits balanced performance (F1: 82.8, Recall: 84.6), while the metadata-only scenario maintains strong accuracy (F1: 79.9, Precision: 89.4). This progression underscores the framework's capacity to effectively leverage additional information when available while maintaining robust performance even with limited data.

These ablation studies underscore the effectiveness of our framework's design choices. The complementary nature of diverse features, the superior performance of LLaMA3.1-8b, and the framework's adaptability across scenarios validate our architectural decisions and implementation strategies.

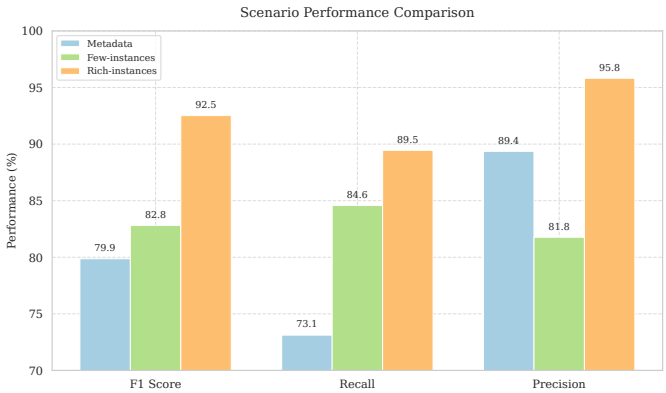

Fig. 6. Performance comparison across different scenarios.

VII. Conclusions and Future Work

In this paper, we present a unified LLM-based schema matching framework that efficiently addresses schema matching tasks across three scenarios: Scenario with only MetaData (SMD), Scenario with a Small amount of instance Data (SSD), and Scenario with a Large amount of instance Data (SLD). Our framework uniquely integrates traditional semantic similarity computation and LLM prompt engineering to achieve efficient and accurate schema matching across diverse scenarios.

The framework's effectiveness is validated through comprehensive experiments on WikiData and MIMIC-III datasets. Results highlight superior performance across all scenarios, particularly in the rich-instances scenario, achieving an F1 score of 92.5 and precision of 95.8. Our ablation studies reveal that integrating diverse feature types substantially improves matching accuracy, with the combination of textual, structural, and semantic features outperforming single-feature approaches. The use of LLaMA3.1-8b delivers optimal performance among tested models, consistently achieving strong results across scenarios.

Key innovations of our framework include: (1) scenario-adaptive similarity computation that optimizes feature utilization based on data availability; (2) carefully designed prompt templates tailored for different matching scenarios; and (3) a unified matching strategy that effectively balances precision and recall across varying data conditions. The framework's ability to maintain strong performance even with limited data while scaling effectively to handle rich instances demonstrates its practical utility for real-world applications.

For future work, several promising directions are identified: (1) extending the framework to support more complex matching relationships, including one-to-many and many-to-many mappings; (2) incorporating retrieval-augmented generative techniques to assist the schema matching model and improve matching performance; and (3) exploring methods to handle very long sequences and reduce memory complexity for large-scale schema matching tasks. These developments would further enhance the framework's capabilities in handling complex real-world schema matching scenarios.

References

[1] S. S. Aman, D. D. A. Agbo, B. G. N'guessan, and T. Kone, "Design of a data storage and retrieval ontology for the efficient integration of information in artificial intelligence systems," *International Journal of Information Technology*, vol. 16, no. 3, pp. 1743–1761, 2024.

[2] A. A. Alwan, A. Nordin, M. Alzeber, and A. Z. Abualkishik, "A survey of schema matching research using database schemas and instances," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 10, 2017.

[3] I. F. Cruz, F. P. Antonelli, and C. Stroe, "Agreementmaker: efficient matching for large real-world schemas and ontologies," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1586–1589, 2009.

[4] E. Sutanta, R. Wardoyo, K. Mustofa, and E. Winarko, "A hybrid model schema matching using constraint-based and instance-based." *International Journal of Electrical & Computer Engineering (2088-8708)*, vol. 6, no. 3, 2016.

[5] S. K. Ramnandan, A. Mittal, C. A. Knoblock, and P. Szekely, "Assigning semantic labels to data sources," in *European semantic web conference*. Springer, 2015, pp. 403–417.

[6] C. Koutras, G. Siachamis, A. Ionescu, K. Psarakis, J. Brons, M. Fragkoulis, C. Lofi, A. Bonifati, and A. Katsifodimos, "Valentine: Evaluating matching techniques for dataset discovery," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 468–479.

[7] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," in *vldb*, vol. 1, no. 2001, 2001, pp. 49–58.

[8] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 117–128.

[9] T. Sahay, A. Mehta, and S. Jadon, "Schema matching using machine learning," in *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 2020, pp. 359–366.

[10] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos, "imap: Discovering complex semantic matches between database schemas," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, 2004, pp. 383–394.

[11] J. Zhang, B. Shin, J. D. Choi, and J. C. Ho, "Smat: An attention-based deep learning solution to the automation of schema matching," in *Advances in Databases and Information Systems: 25th European Conference, ADBIS 2021, Tartu, Estonia, August 24–26, 2021, Proceedings 25*. Springer, 2021, pp. 260–274.

[12] C. Koutras, M. Fragkoulis, A. Katsifodimos, and C. Lofi, "Rema: Graph embeddings-based relational schema matching." in *EDBT/ICDT Workshops*, 2020.

[13] R. Cappuzzo, P. Papotti, and S. Thirumuruganathan, "Creating embeddings of heterogeneous relational datasets for data integration tasks," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1335–1349.

[14] B. Hättasch, M. Truong-Ngoc, A. Schmidt, and C. Binnig, "It's ai match: A two-step approach for schema matching using embeddings," *arXiv preprint arXiv:2203.04366*, 2022.

[15] R. Fagin, L. M. Haas, M. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis, "Clio: Schema mapping creation and data exchange," *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*, pp. 198–236, 2009.

[16] A. Doan, P. M. Domingos, and A. Y. Levy, "Learning source description for data integration." in *WebDB (informal proceedings)*, 2000, pp. 81–86.

[17] Y. Zhang, A. Floratou, J. Cahoon, S. Krishnan, A. C. Müller, D. Banda, F. Psallidas, and J. M. Patel, "Schema matching using pre-trained language models," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1558–1571.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[19] H. Zhang, Y. Dong, C. Xiao, and M. Oyamada, "Jellyfish: A large language model for data preprocessing," *arXiv preprint arXiv:2312.01678*, 2023.

[20] C. Drumm, M. Schmitt, H.-H. Do, and E. Rahm, "Quickmig: automatic schema matching for data migration projects," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 107–116.

[21] H.-H. Do and E. Rahm, "Coma—a system for flexible combination of schema matching approaches," in *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 2002, pp. 610–621.

[22] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava, "Automatic discovery of attributes in relational databases," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, 2011, pp. 109–120.

[23] J. Madhavan, P. A. Bernstein, A. Doan, and A. Halevy, "Corpus-based schema matching," in *21st International Conference on Data Engineering (ICDE'05)*. IEEE, 2005, pp. 57–68.

[24] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[25] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[26] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama

2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.
[27] Ollama, "Ollama: Simplifying large language model deployment," 2023, accessed: January 2025. [Online]. Available: https://ollama.com
[28] OpenAI, "Gpt-4 technical report," 2023, accessed: January 2025. [Online]. Available: https://openai.com/research/gpt-4