LLM-CER: An Interactive System for In-context Clustering-based Entity Resolution with Large Language Models

Haoyu Wang
Zhejiang University
Zhejiang, China
haoyu4260@gmail.com

Haitong Tang
Zhejiang University
Zhejiang, China
tht@zju.edu.cn

Jiajie Fu
Zhejiang University
Zhejiang, China
jiajiefu@zju.edu.cn

Arijit Khan
Bowling Green State University
Ohio, USA
arijitk@bgsu.edu

Sharad Mehrotra
UC Irvine
Irvine, USA
sharad@ics.uci.edu

Xiangyu Ke
Zhejiang University
Zhejiang, China
xiangyu.ke@zju.edu.cn

Yunjun Gao
Zhejiang University
Zhejiang, China
gaoyj@zju.edu.cn

ABSTRACT

Entity Resolution (ER) identifies and links records that refer to the same real-world entity. Rule-based approaches rely on explicit similarity functions, while deep learning and pre-trained language model (PLM)-based methods require large amounts of task-specific labeled data, both of which are often difficult to obtain. Large Language Models (LLMs) provide a promising alternative via zero-shot or few-shot prompting. However, existing LLM-based ER methods still adopt the pairwise matching paradigm, leading to poor scalability and high API costs on large datasets. We introduce LLM-CER (LLM-powered Clustering-based ER), an interactive, end-to-end system that instructs LLMs to perform in-context clustering of records, that is, *converting ER from pairwise matching into a clustering problem*, to reduce the number of costly LLM interactions while preserving high matching quality. LLM-CER contributes three elements. First, a novel *in-context clustering paradigm* and a systematic design-space study of factors that affect LLMs' clustering behavior (set size, within-set diversity, record variation, and ordering). Second, an *extensible pipeline and UI* that let users select dataset subsets, tune clustering hyperparameters, and switch LLM models before execution. Third, an *interactive visualization and monitoring suite* that shows hierarchical clustering steps, flag potential misclusters, log every LLM API call, and report clustering quality and efficiency metrics across models and parameter settings. In the demo, attendees will (1) run end-to-end ER on representative datasets, (2) compare metric and cost trade-offs across configurations and LLMs, and (3) inspect and correct ambiguous clusters via the visualization tools.

PVLDB Reference Format:
Haoyu Wang, Haitong Tang, Jiajie Fu, Arijit Khan, Sharad Mehrotra, Xiangyu Ke, and Yunjun Gao. LLM-CER: An Interactive System for In-context Clustering-based Entity Resolution with Large Language Models. PVLDB, 19(12): 4746 - 4749, 2026.
doi:10.14778/3827998.3828112

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828112

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/AAWHY/LLMCER.

1 INTRODUCTION

Entity Resolution (ER) is a core data-quality task that identifies and links records referring to the same real-world entity despite differing identifiers or textual descriptions [2]. Traditional approaches to ER follow the *pairwise-matching paradigm*, where each pair of candidate records is compared independently to determine whether they match. This paradigm scales quadratically in the number of records and becomes prohibitively expensive for large collections. To mitigate this cost, practical ER pipelines use a *blocking* or *filtering* stage [9] to produce candidate blocks, followed by a more expensive *resolution* stage that matches and merges records within each block. Even after blocking, the resolution phase remains the dominant consumer of computation and monetary costs in modern ER workflows [6].

Recent advances in Large Language Models (LLMs) offer a new route for ER [7]: prompt-driven, few-shot, or zero-shot interactions that can perform semantic matching without extensive task-specific supervision. In many practical ER tasks, the similarity between records cannot be captured by a fixed, hand-crafted function, rendering rule-based methods ineffective. Deep learning and PLM-based approaches can learn similarity from data [5, 6], but they require large amounts of labeled record pairs for task-specific training, which are often expensive and time-consuming to acquire. A label-free alternative, embedding-based clustering, decides matches from geometric proximity in a fixed embedding space, without reasoning over the records' content. LLMs overcome all three limitations: they reason over records' semantic content in a zero-shot manner, matching without a hand-crafted similarity function or labeled training data, making them a compelling choice for ER.

However, existing LLM-based ER methods *predominantly follow the pairwise-matching paradigm* [3] and therefore still incur high interaction counts and monetary cost when applied naïvely at scale. At the same time, independent work has shown that LLMs can perform set-level tasks such as clustering text items effectively in zero- or few-shot settings [10]. Since entity resolution is fundamentally a clustering problem, i.e., grouping together records that denote

the same entity, directly instructing LLMs to cluster record sets can substantially reduce the number of LLM invocations and the end-to-end cost of ER. It lowers monetary cost by up to 22× over pairwise LLM-based ER, with fewer API calls and higher accuracy [4].

Motivated by this observation, we propose *in-context clustering for entity resolution* and present LLM-CER [4], an end-to-end, interactive system that operationalizes this idea. LLM-CER is built on three algorithmic components informed by a systematic design-space study of LLMs' clustering behavior[1]: (i) Next Record Set Creation (NRS) that constructs optimal record sets as input for LLM-based in-context clustering; (ii) Misclustering Detection Guardrail (MDG) that validates LLM clustering outputs, detects likely hallucinations or unstable clusters, and triggers corrective actions; and (iii) Hierarchical Cluster Merging (CMR) that consolidates LLM clustering results across rounds by iteratively merging clusters to form final entity groups.

In this demonstration, we show how LLM-CER supports configurable end-to-end ER: attendees can select datasets, adjust clustering hyperparameters, and switch LLM models; the system then executes NRS+LLM clustering+MDG+CMR algorithms, while visualizing the hierarchical clustering process level by level. *During this process, users can track specific record sets across levels to trace how clusters evolve, and engage in human-in-the-loop verification by replacing* MDG *with manual clustering assessment.* The UI provides fine-grained monitoring (API call logs, per-call detection flags, resolved entities) and both efficiency and quality metrics so users can compare trade-offs across models and parameter settings. To the best of our knowledge, this is the first interactive demonstration of in-context, LLM-driven clustering applied end-to-end to ER with real-time monitoring and interactive parameter tuning.

2 SYSTEM OVERVIEW AND ALGORITHMS

Figure 1 illustrates the overall architecture of LLM-CER and the interaction between the frontend and backend. The frontend provides dataset selection, parameter configuration, and visualization of the hierarchical in-context clustering process, resolved entities, and performance metrics; the backend orchestrates the ER pipeline and exposes monitoring and audit logs that the frontend consumes. The prototype implementation uses Vue.js for the user interface and a Python backend built on FastAPI to handle orchestration, LLM calls, and metric computation.

Given input records, LLM-CER first applies an LSH [1] **Blocking** module to partition records into candidate blocks, so that the subsequent pipeline does not need to consider all pairs. The system then executes three core techniques at the heart of our approach: NRS constructs appropriately sized and diverse record sets for in-context clustering, MDG detects and corrects erroneous LLM outputs through lightweight similarity-based verification, and CMR hierarchically merges clusters across levels until no further consolidation is possible. Together, these techniques enable cost-effective and scalable entity resolution. The system also provides an **Interactive GUI** that supports the full workflow through a configuration panel, step-by-step visualization of hierarchical clustering with record set tracking across levels, a human-in-the-loop mode that

[1]We characterize the LLM-based clustering with set size, within-set diversity, record variation, and ordering in our full research paper [4].

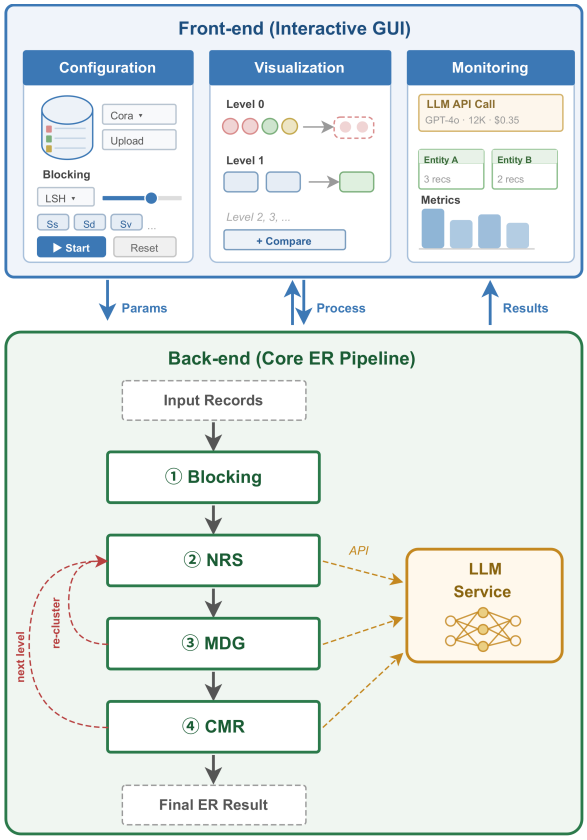

Figure 1: System architecture of LLM-CER.

allows users to replace automatic MDG verification with manual clustering assessment, and detailed logs and metrics, including LLM API call logs with MDG verification status, final ER results with export functionality, and performance measurements across different models and parameter settings.

We now present the algorithms underlying these techniques; implementation details and empirical analysis are provided in our research paper [4]. LLM-CER follows an in-context clustering paradigm, where an LLM clusters a set of records within a single API call rather than resolving pairwise match queries. Our empirical study identifies four key factors influencing clustering quality: set size S_s (optimal at 9, balancing accuracy and API efficiency), set diversity S_d (best at 4 entity groups per set), set variation S_v (lower values, i.e., more balanced distributions, lead to better performance), and record ordering (sequential ordering of similar records improves clustering by enhancing contextual coherence). Based on these findings, LLM-CER incorporates the three core techniques NRS, MDG, and CMR introduced above. We detail each below.

Next Record Set Creation (NRS**).** Given a block of candidate records B, NRS is applied iteratively to the remaining records B_{remain} to construct a sequence of record sets that satisfy the desired configuration of (S_s, S_d, S_v). When $|B_{remain}| \leq S_s$, all remaining records are packed into a single set and ordered by similarity, with more similar records placed adjacent to each other. Otherwise, NRS applies k-means clustering on record embeddings and

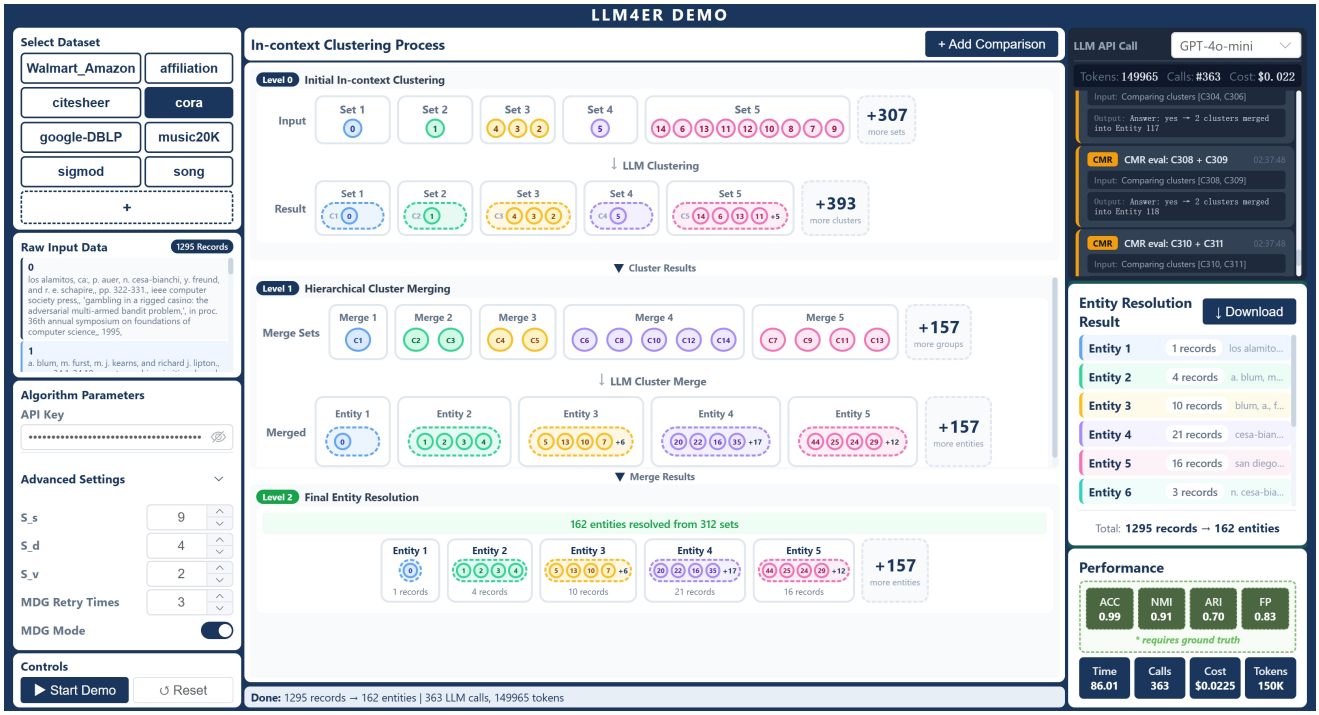

Figure 2: The LLM-CER **system: left – configuration panel (dataset, parameters), center – hierarchical in-context clustering visualization (colored nodes denote distinct entities), right – LLM API logs, ER results, and performance metrics.**

estimates the number of latent entities using the elbow method. It then selects $\lfloor S_s/S_d \rfloor$ records from each cluster to promote balanced diversity and fills the remaining slots with records that minimally increase S_v. Finally, records within each set are ordered by similarity to facilitate the LLM's contextual understanding and improve clustering accuracy. This set-based formulation also enables the LLM to jointly resolve all intra-set relationships among S_s records in a single call, significantly reducing the number of LLM calls.

Misclustering Detection Guardrail (MDG**).** LLMs are susceptible to hallucination, which may lead to incorrect clustering outputs. MDG provides a lightweight verification mechanism with $O(S_s^2)$ complexity. For each record r in a cluster C_i, MDG computes the *intra-cluster similarity*, defined as the minimum similarity between r and other records within C_i, and the *inter-cluster similarity*, defined as the maximum similarity between r and records from other clusters. If any record has intra-cluster similarity lower than its inter-cluster similarity, the clustering result is flagged as potentially unreliable. In that case, LLM-CER performs *record set regeneration*, where the misclustered record is relocated closer to the most similar cluster to place semantically similar records in proximity, producing a better-ordered record set, and the LLM is queried again. This retry process is bounded by a user-configurable limit. Since the verification step operates solely on precomputed embedding similarities and re-querying is triggered only when errors are detected, MDG adds negligible overhead in practice.

Hierarchical Cluster Merging (CMR**).** After the initial in-context clustering of all record sets within a block, clusters from *different* record sets may still refer to the same entity and therefore require further merging. In contrast, clusters produced from the *same* record set are guaranteed to be distinct due to anti-transitivity. CMR addresses this by replacing each cluster with a single representative record and grouping the most similar representatives from different record sets into new record sets that satisfy the (S_s, S_d, S_v) constraints, while ensuring that representatives from the same original set are never grouped together. This hierarchical process repeats across levels until it converges to singleton clusters, and the final consolidated ER result for the entire block is produced. CMR avoids costly pairwise comparisons by condensing each cluster into a single representative record and repacking them into sets of size S_s, so that each LLM call resolves overlap among S_s clusters, substantially reducing LLM calls. Since each LLM call still processes at most S_s records rather than the whole dataset, the number of calls grows only near-linearly, so the approach scales to large datasets, as confirmed up to 50K records in our research paper [4].

3 DEMONSTRATION SCENARIOS

We illustrate the demonstration using Figure 2, which shows a snapshot of the LLM-CER frontend. The system, available at https://github.com/AAWHY/LLMCER, enables users to configure, execute, and inspect the full in-context clustering-based ER pipeline interactively. As a running example, we use the Cora dataset provided by [8], a text-based collection of publication records containing duplicate references to scientific papers. The demonstration is organized around a three-step workflow.

Step 1. Initial Selection: Users begin by selecting a dataset from the left panel of LLM-CER, such as Cora shown in Figure 2, or

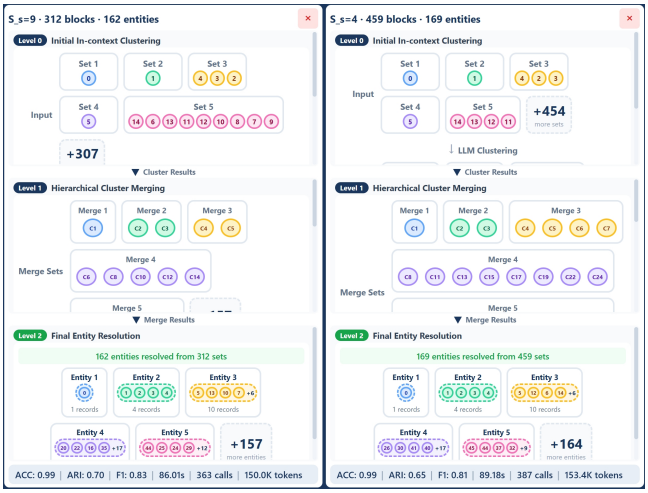

Figure 3: Hierarchical clustering visualization with different set sizes ($S_s = 9$ vs. $S_s = 4$) on the Cora dataset.

uploading a custom dataset via the upload button. Upon selection, the "Raw Input Data" panel displays the raw input records. If the user uploads a ground-truth label file alongside the dataset, the panel renders color-coded entity annotations, where distinct colors indicate different ground-truth entities. In this demonstration, we provide the ground truth for the Cora dataset, enabling users to visually trace clustering performance throughout the pipeline. Users can click to view the full raw dataset and inspect record attributes. The system uses LSH [1] blocking to automatically group records into candidate blocks. Through the "Advanced Settings" area, users can fine-tune key clustering hyperparameters, including set size S_s, diversity S_d, variation S_o, and the MDG retry limit. Users can also switch MDG to a human-in-the-loop mode, where they take over the verification role and assess clustering quality based on their domain expertise. After configuration, users start the end-to-end ER workflow by clicking "Start".

Step 2. Visualizing Hierarchical In-context Clustering: Once the pipeline is launched, the center panel visualizes the hierarchical in-context clustering procedure in a level-by-level manner. At Level 0, users observe the input records partitioned into multiple disjoint sets, each containing up to S_s records. Users can identify records from different entities through color-coded nodes, and examine the LLM's clustering output for each set shown as dashed-border groups. If the user enables MDG, the system automatically verifies each clustering result and triggers re-querying when potential errors are detected. Alternatively, users can choose the human-in-the-loop mode, where they review and adjust clustering outputs based on domain expertise, offering full control over verification. At Level 1 and beyond, users can observe how clusters from different sets are merged: a representative record is selected for each cluster, similar representatives are grouped into new record sets, and the LLM is invoked again to consolidate overlapping clusters. This interactive visualization allows users to understand how clustering decisions evolve across levels and to identify potential inconsistencies or errors in the clustering results. Users can select any record set at any level and trace the full clustering history of its records across all levels, from initial partitioning to final entity resolution. To support comparative analysis, users may also open multiple panels via the *Add Comparison* function and inspect how different parameter settings affect the number of record sets, the hierarchical depth, and the final clustering quality. Figure 3 illustrates such a comparison between $S_s = 9$ and $S_s = 4$ on the Cora dataset, showing how the two configurations lead to different record set partitions, hierarchical depths, and clustering outcomes across levels.

Step 3. Result Inspection and Evaluation: Through the right panel, users can inspect results and monitor the ER pipeline in real time. In the *LLM API Call* sub-panel, users can review every LLM interaction throughout the pipeline, including cumulative token consumption, total API call count, and monetary cost. Each log entry records the corresponding pipeline step, from NRS record set construction and LLM clustering output to MDG verification and CMR merge operations, providing full transparency for users to trace each clustering decision and identify potential bottlenecks. The *Entity Resolution Result* sub-panel allows users to browse the resolved entities displayed as color-coded cards, each showing the entity name and the number of associated records, together with overall resolution statistics. Users can export the results for downstream analysis. The *Performance* sub-panel reports clustering quality metrics, including ACC, NMI, ARI, and FP-measure, when ground truth is available, as well as efficiency metrics such as runtime, API calls, token consumption, and monetary cost. Based on these results, users can switch among different LLM models (e.g., GPT-4o, GPT-4.1) via the model dropdown selector and rerun the same ER task under identical settings, enabling direct comparison of quality-cost trade-offs across models to identify the optimal configuration.

REFERENCES

[1] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. In *Proc. VLDB Endow.*, Vol. 11. 1454–1467.

[2] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (2007), 1–16.

[3] Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2024. Cost-Effective In-Context Learning for Entity Resolution: A Design Space Exploration. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 3696–3709.

[4] Jiajie Fu, Haitong Tang, Arijit Khan, Sharad Mehrotra, Xiangyu Ke, and Yunjun Gao. 2025. In-context Clustering-based Entity Resolution with Large Language Models: A Design Space Exploration. *Proc. ACM Manag. Data* 3, 4 (2025), 1–28.

[5] Yuliang Li, Jinfeng Li, Yoshi Suhara, AnHai Doan, and Wang-Chiew Tan. 2023. Effective Entity Matching with Transformers. *The VLDB Journal* 32, 6 (2023), 1215–1235.

[6] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.

[7] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data?. In *Proc. VLDB Endow.*, Vol. 16. 738–746.

[8] Konstantinos Nikoletos, George Papadakis, and Manolis Koubarakis. 2022. pyJedAI: a Lightsaber for Link Discovery. In *ISWC (Posters/Demos/Industry)*.

[9] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Comput. Surv.* 53, 2 (2020), 1–42.

[10] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2024. Large Language Models Enable Few-Shot Clustering. *Transactions of the Association for Computational Linguistics* 12 (2024), 321–333.