In-context Clustering-based Entity Resolution with Large Language Models: A Design Space Exploration

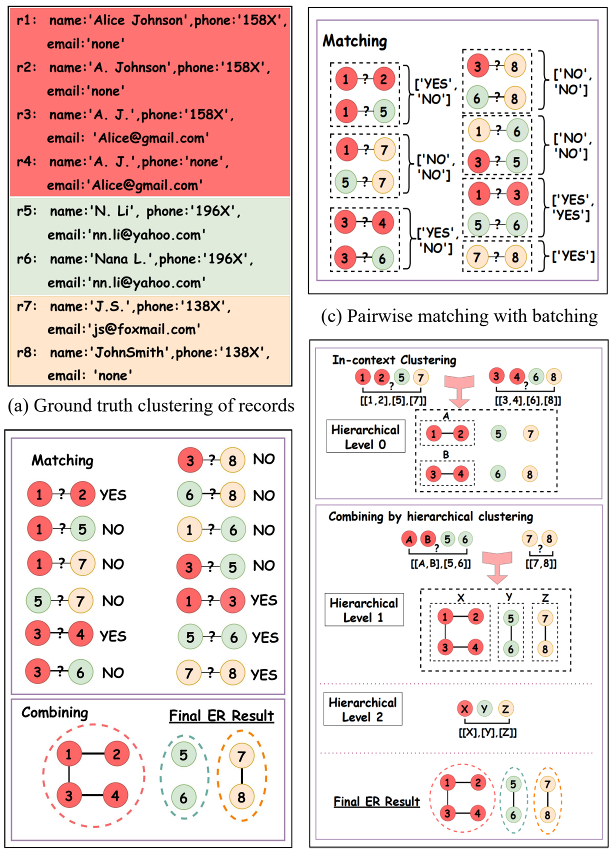
ABSTRACT

Entity Resolution (ER) is a fundamental data quality improvement task that identifies and links records referring to the same real-world entity. Traditional ER approaches often rely on pairwise comparisons, which can be costly regarding both time and monetary resources, especially when large datasets are involved. Recently, Large Language Models (LLMs) have demonstrated promising results in ER tasks. Still, existing methods typically focus on pairwise matching, missing the potential of LLMs to directly perform clustering in a more cost-effective and scalable manner. In this paper, we propose a novel *in-context clustering* approach for ER, where LLMs are used to cluster records directly, reducing both time complexity and monetary costs. We systematically investigate the design space for in-context clustering, analyzing the impact of factors such as set size, diversity, variation, and ordering of records on clustering performance. Based on these insights, we develop novel algorithms to obtain high-quality ER results while minimizing LLM API calls. Our approach addresses key challenges, including efficient cluster merging and LLM's hallucination, providing a scalable and effective solution for ER. Extensive experiments on seven real-world datasets demonstrate that our method significantly improves result quality, achieving up to 150% higher accuracy, 10% increase in the F-measure, and reducing API calls by up to 5×, while maintaining a comparable monetary cost to the most cost-effective baseline.

1 INTRODUCTION

Entity resolution (ER) is a critical data quality task that identifies and disambiguates real-world objects or entities referred to by different identifiers or descriptions across multiple records. ER aims to link and cluster these records, ensuring a clean and unified dataset [13, 15, 24, 54]. However, traditional ER is computationally expensive, especially when dealing with large datasets, as it requires checking every possible pair of records to determine whether they refer to the same entity. To mitigate this cost, ER typically includes a preprocessing phase that employs techniques such as blocking and filtering [58], which partition the dataset into blocks (possibly overlapping), where records within different blocks are unlikely to co-refer. The subsequent resolution phase, which involves matching and combining records, is the more resource-intensive step, ensuring that duplicates are grouped within the same cluster.

Existing ER solutions can be broadly divided into three categories: machine learning (ML) and deep learning-oriented methods [23], crowdsourcing-based frameworks [73], and those leveraging large language models (LLMs) [43], such as GPT or LLAMA. Crowdsourcing frameworks aim to attain high-quality results with human intelligence but are costly. For example, a Human Intelligence Task (HIT) such as pairwise comparisons on platforms like Amazon Mechanical Turk (AMT) typically costs around USD 0.02 per task [76]. Assuming a dataset of 1000 records, done naively, entity resolution would require $O(10^6)$ comparisons worth ≈ USD 20K. Several prior



```
r1: name:'Alice Johnson',phone:'158X',
    email:'none'
r2: name:'A. Johnson',phone:'158X',
    email:'none'
r3: name:'A. J.',phone:'158X',
    email: 'Alice@gmail.com'
r4: name:'A. J.',phone:'none',
    email:'Alice@gmail.com'
r5: name:'N. Li', phone:'196X',
    email:'nn.li@yahoo.com'
r6: name:'Nana L.',phone:'196X',
    email:'nn.li@yahoo.com'
r7: name:'J.S.',phone:'138X',
    email:'js@foxmail.com'
r8: name:'JohnSmith',phone:'138X',
    email: 'none'
```

(a) Ground truth clustering of records

(b) Pairwise matching

(c) Pairwise matching with batching

(d) In-context clustering

Figure 1: (a) Ground truth ER result. (b) Pairwise matching and (c) Pairwise matching with batching (state-of-the-art) vs. (d) In-context clustering (this work) for ER. The combining phase is omitted for pairwise matching with batching [26], as it remains the same with pairwise matching [53].

works have tried to reduce such costs while still ensuring high quality, e.g., CrowdER [76] explores a hybrid human-machine approach in which machines are used to eliminate apparent non-duplicates, and crowd workers are used to verify only the most likely matching pairs. Machine-based approaches, such as rule-based [5], ML-based [6], and deep learning-based [23] works formulate pairwise matching as a classification problem. These models can be trained on labeled data to provide faster inference. Still, they require task-specific supervision and large amounts of labeled data, which are both expensive and time-consuming to acquire. For instance, AMT charges USD 0.08 per labeling task [26], making the data annotation process costly and difficult. Recently emerged large language models, e.g., GPT-4, are pre-trained on massive corpora in a self-supervised manner [46] and have achieved remarkable results in a series of natural language processing and data science tasks like text generation [65], machine translation [25], data research and education [3]. General-purpose LLMs circumvent expensive task-specific supervision and the need for labeled data due to simpler prompt-based interactions, making them easy to operate and iterate in downstream applications. Moreover, their lower costs—e.g., GPT-4o-mini charges only USD 0.15 per million tokens [1]—and superior performance on zero-shot tasks [38] make them an attractive alternative for entity resolution tasks.

Recently, several works have pursued LLMs for ER. Narayan et al. [53] pioneered the exploration of GPT-3's potential for ER in a few-shot manner using task demonstrations. Their approach has exhibited notable performance improvements compared to methods based on pre-trained language models (PLMs), e.g., Ditto [45]. ZeroMatch [89] takes a more advanced approach by using parameter-efficient fine-tuning, maintaining many pre-trained LLM variants from different domains, and selecting the most appropriate variant at inference time for zero-shot ER. Further advancements in prompt engineering for ER have been investigated in works such as [36, 52, 62, 80]. Li et al. [43] develop BoostER, a framework that applies LLMs to select the best answer from multiple different partitioning generated by existing ER tools. Their focus is on the next record pair selection problem, reducing the uncertainty of the current ER result. Additionally, batch prompting, which involves packing multiple pairwise questions into a single prompt to reduce the monetary cost of calling LLM APIs, has been studied in [26, 87].

However, all state-of-the-art LLM-based ER methods rely on comparing record pairs and do not fully exploit the emergent capability of LLMs to *directly cluster a set of records*. Recent studies have shown that LLMs are surprisingly effective at clustering text data in a zero-shot or few-shot manner, preserving semantic meaning and context [72, 75, 88]. Given that entity resolution is essentially a clustering problem, LLMs' clustering capabilities can be directly leveraged to reduce both the time and monetary costs associated with end-to-end ER. To bridge this gap, we – for the first time – formulate and investigate the problem of in-context clustering-based entity resolution, that is, instructing an LLM to cluster different record sets to achieve the end-to-end ER task at a significantly lower cost and time. Consider the following example.

Example 1. *Figure 1(b) compares existing pairwise matching [53] with our proposed in-context clustering approach for an ER task. Figure 1(a) shows the records with ground-truth entity assignments, where each color represents an entity.*

Pairwise questioning-based ER consists of two phases: matching and combining (excluding blocking for simplicity). Matching aims to determine if a pair of records refers to the same entity. Transitivity can reduce comparisons; for example, if records a and b are the same entity, and b and c are the same (or different), we can infer that a and c are also the same (or different). As shown in Figure 1(b), the matching phase concludes when all record pairs are compared explicitly, or their status is inferred through transitivity or anti-transitivity. In the combining phase, the final ER result is obtained by clustering records belonging to the same entity.

In contrast, our in-context clustering approach directly creates input record sets for clustering by the LLM, as depicted in Figure 1(d) (with a maximum size of 4). Records within each LLM cluster are considered the same entity (i.e., transitivity holds). For example, in the LLM-based clustering of the set $(1, 2, 5, 7)$*, records* 1 *and* 2 *are clustered together, indicating they are the same entity, while* 2 *and* 5 *belong to different entities (i.e., anti-transitivity), as reflected in their assignment to clusters* $A = [1, 2]$ *and* $[5]$*. In cases where transitivity or anti-transitivity relationships are unknown, clusters are packed to form the next level of input sets, such as* $(A, B, 5, 6)$*. This hierarchical process continues, merging clusters until all record pairs from different clusters satisfy anti-transitivity, yielding the final ER output.*

Unlike pairwise matching, which classifies two records at a time and heavily relies on transitivity, in-context clustering *partitions multiple records simultaneously* and *hierarchically merges them*. Each pairwise comparison requires an LLM API call, driving up both cost and time. In the worst case, the ER process requires comparing all pairs of records when they belong to different entities. As illustrated in Figure 1(b), pairwise matching requires 13 comparisons (i.e., 13 LLM API calls) for 8 records, even with the help of transitivity. Our method, however, only needs 5 record sets for clustering, leading to fewer LLM API calls, lower costs, and reduced running time. This efficiency becomes particularly significant for large-scale datasets, where processing costs and ER time are crucial.

Example 2. *An improved baseline, pairwise matching with batching [26], reduces costs by processing multiple pairwise questions in batches. With the same constraint of 4 records per LLM API call for a fair comparison (i.e., 2 pairwise questions in a batch), Figure 1(c) shows that batching reduces the number of API calls to 7, still higher than that of 5 by our in-context clustering approach. This inefficiency arises because the pairwise questioning format inherently introduces redundancy when conveying information to the LLM. For instance, exploring all possible relationships among four records require 6 pairwise questions, which necessitates 3 batches of 2 pairwise questions each. Even with transitivity and anti-transitivity, the number of batches can only be reduced to 2 in this example. In contrast, our approach can capture all relationships among 4 records in a single API call, thus reducing the number of API calls. A key follow-up question is the highest possible information density (i.e., maximum set size constraint in our clustering solution) that the LLM can effectively process in a single API call*[1]*. This motivates our in-depth investigation in § 4.2.*

Although LLMs have demonstrated strong performance in ER via pairwise matching [26, 53], the application of in-context clustering for LLM-based ER remains largely unexplored. We observe that directly applying in-context clustering to ER, without a carefully designed mechanism, introduces several key challenges: **(1)** *Clustering Performance Variability:* The clustering performance of an LLM can vary significantly based on how the record sets are configured. Determining the optimal design for these sets—specifically in terms of size, diversity, and the ordering of records—is crucial. For instance, if the record set is too large, the LLM's performance may degrade due to long context lengths [44], whereas a smaller set size will cause excessive LLM API calls, increasing monetary cost. (2) *Hallucination Risks:* LLMs are prone to hallucinations, where the model generates outputs that may not be grounded in the input data [33]. As a result, the outputs of in-context clustering may not always be accurate. Addressing this challenge requires the development of effective result verification strategies, such as implementing LLM "guardrails" and record set regeneration. **(3)** *Efficient Result Combination:* Combining the outputs from in-context clustering

[1]The number of relationships to explore grows quadratically with the set size.

efficiently presents another challenge. The process involves balancing the trade-off between the quality of the end-to-end ER result and the number of LLM API calls, which directly affects monetary cost. To address these challenges, we extensively explore the design space and introduce several novel algorithms that optimize the creation of input record sets, validation of LLM clustering outputs, and the merging of clustering results.

Our empirical evaluations show that in-context clustering performance is optimal when the set contains 9 records drawn from 4 distinct entities, with an approximately equal distribution of records from each entity. Furthermore, clustering performance improves when records from the same entity are grouped sequentially within the set. Based on these insights, we propose an effective record set creation method (§ 5.2), which ensures the optimal balance of set size, diversity, and variation, thereby maximizing the effectiveness of LLM's in-context clustering, while minimizing the associated monetary cost. Extensive experiments on seven real-world datasets demonstrate that our method significantly improves result quality, achieving up to 150% higher Accuracy (ACC) and a 10% increase in the F-measure. Furthermore, our approach reduces API calls by up to five times, cutting down on time overhead, while maintaining a comparable monetary cost to the most cost-effective baseline.

Contribution. Our main contributions are summarized below.

- We are the first to apply a clustering-based approach using LLMs for entity resolution, exploring the design space of clustering-based LLM ER (§ 4).
- We identify four key factors—set size, set diversity, set variation, and record order—and analyze their impact on the performance of in-context clustering (§ 4.2).
- We propose the Next Record Set Creation algorithm, which effectively addresses the four identified factors. To mitigate potential LLM misclassifications, we design a Misclustering Detection Guardrail and a Record Set Regeneration strategy for further refinement of results (§ 5.2).
- We propose a Hierarchical Cluster Merging approach that generates record sets hierarchically and performs efficient merging, improving the quality of cluster merging and overall entity resolution performance (§ 5.3).
- We empirically evaluate our clustering-based end-to-end ER algorithm on seven real-world datasets. The results show that our method significantly improves result quality, achieving up to 150% higher ACC, reduces API calls by up to 5× while maintaining competitive cost efficiency compared to the most cost-effective baseline (§ 6).

We discuss related work and preliminaries in § 2 and § 3, respectively, and conclude in § 7.

2 RELATED WORK

ER problems can be broadly classified into deduplication (dirty ER) and record linkage (clean-clean ER). The former partitions a single record collection into multiple entity clusters. The latter involves matching records from two separate, typically overlapping but duplicate-free, collections (e.g., two tables) and identifying pairs of records that refer to the same entity [39]. Dirty ER is generally more challenging than clean-clean ER due to inherent data quality issues, such as spelling errors, missing values, and noise. In contrast, clean-clean ER typically involves tables that are already cleaned and standardized [14]. This work focuses on the dirty ER problem, while our solution is also applicable to the clean-clean ER problem by considering the union of records across tables as a single collection. We categorize related work as follows.

PLM and LLM-based Entity Resolution. Before the advent of large language models, pre-trained transformer language models (PLMs) were commonly used for ER tasks [41]. DeepBlocker [71] evaluates various deep learning methods, including SBERT (Sentence BERT) [67] for blocking. Ditto [45] exploits PLMs such as BERT [22], DistilBERT [69], or RoBERTa [47] for clean-clean ER. JointBERT develops a dual-objective training method for BERT, combining binary matching and multi-class classification, to predict both match/non-match decisions and entity identifiers in training pairs [61]. RobEM aims to enhance the robustness of PLM-based entity matching models [66].

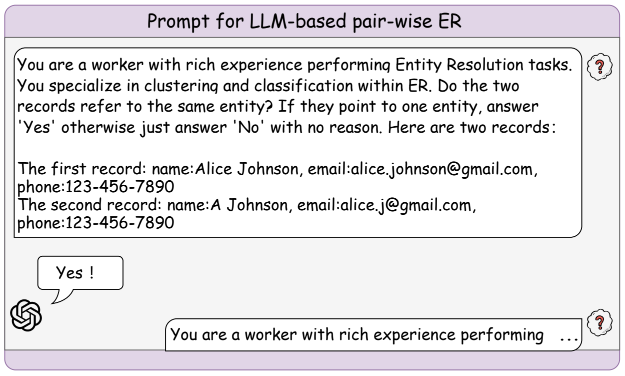
PLM-based ER approaches typically require task-specific labeled data and fine-tuning, which are resource-intensive. More recently, LLMs, a.k.a. foundation models, have demonstrated strong performance in data cleaning and integration tasks, including ER, at a lower cost. This is primarily due to the more straightforward prompt-based interactions without requiring task-specific model retraining or labeled data. State-of-the-art LLM-based ER approaches are discussed in § 1. Among them, [26, 80, 87] are the closest to ours. Both [26, 87] introduce batch prompting, where multiple pairwise questions are packaged into a single batch for the LLM. However, our approach differs by packing multiple records into a set for direct, in-context clustering via the LLM, allowing for more efficient and scalable ER. Furthermore, ComEM [80] employs advanced prompts such as "match", "compare", or "select" to explore interactions across multiple records beyond pairwise questions. However, unlike ours, it does not leverage the direct clustering capacity of LLMs for a record set. As pairwise questioning is a special case of our in-context clustering (with set size = 2), our framework generalizes the state-of-the-art ER approaches, offering improved cost-effectiveness and efficiency.

Additionally, our work differs from existing LLM-based ER solutions in its comprehensive end-to-end pipeline, which includes blocking/filtering (except for [26]), and addresses critical issues like LLM hallucinations, which are not explicitly handled before.

Crowdsourcing-based Entity Resolution. A key concern in crowdsourced entity resolution (ER) is minimizing the number of questions posed to workers, which directly impacts the overall cost. Transitivity is commonly used to reduce the number of questions in [74, 77]. Closer to our approach, CrowdER [76] develops a clustering-based method where crowd workers are directly tasked with clustering a set of records. Various models for selecting high-quality questions have been developed in [10, 82]. ZenCrowd combines algorithmic and crowdsourcing-based matching techniques within a probabilistic framework [21], while Roomba uses a decision-theoretic approach to select matches to confirm based on their utility [37]. More recent works [32, 73, 78, 85] consider crowd errors in ER tasks. Corleone crowdsources the entire ER workflow, including blocking rules learning [31], and Falcon scales Corleone using RDBMS-style query execution over a Hadoop cluster [19].

Although CrowdER [76] provides record sets to human taskers for direct clustering, there are fundamental differences between their approach and our in-context clustering using LLMs. (1) CrowdER aims to minimize the number of record sets required (equivalently the number of HITs), given a predefined set size, to cover and resolve all uncertain record pairs in a block. In contrast, we adopt a hierarchical approach for record set creation. Unlike CrowdER, which generates all record sets *at once* and covers all uncertain record pairs, our method incrementally generates record sets across multiple layers. In the initial layers, we focus on generating non-overlapping sets that cover all records, *without immediately resolving all uncertain pairs*. As we progress to higher layers, we leverage transitivity and anti-transitivity to *identify and reduce unnecessary record sets*. This approach minimizes the total number of sets, thus reducing the number of LLM API calls and ultimately lowering monetary costs. According to our experimental results over real-world datasets and considering the same set size as well as the same blocking approach (e.g., LSH [23]), the number of record sets (equivalently the number of HITs) needed for CrowdER can be 2-5× higher than the number of record sets (equivalently the number of LLM API calls) required by ours. (2) Our combining approach differs from CrowdER's in several key ways. The initial record sets generated by our algorithm are non-overlapping. Based on clustering results via the LLM, we conduct hierarchical record sets generation and further in-context clustering to merge those clusters. Our hierarchical record sets generation process maintains the optimal record set configuration, it also leverages transitivity and anti-transitivity to reduce the number of LLM API calls. In summary, direct and robust LLM-based clustering is employed both in our matching and combining phases. In contrast, CrowdER allows overlaps of records in different HITs, and their cluster merging process relies on these overlaps to indirectly facilitate merging through transitivity. This could suffer due to human errors as two clusters could be incorrectly merged. (3) Finally, our approach includes *a guardrail mechanism to assess the quality of the LLM's clustering results*. This allows us to identify and discard incorrect clustering results and regenerate better record sets. In contrast, CrowdER assumes that crowdsourcing results are always accurate, it lacks a built-in process to validate or modify the clustering outcomes. These differences ensure that our method not only minimizes the number of record sets (equivalently the number of LLM API calls) but also maintains higher quality of the final ER results.

Filtering and Blocking. Blocking is a critical preprocessing step in ER, aimed at reducing computational overhead by partitioning records into groups and ensuring that only potentially matching records are compared. Blocking methods could be classified as rule-based, sorting-based, and hash-based. Rule-based methods organize tuples by applying fixed keys or decision rules created by experts or derived from heuristics [64]. Sorting-based methods cluster records by rapidly sorting them according to their textual similarities, which are determined using different similarity functions [40]. Hash-based approaches utilize hashing techniques, such as Min-Hashing and Locality-Sensitive Hashing (LSH), to place records into different buckets [23]. Filtering complements blocking by eliminating record pairs that are guaranteed not to match, thus focusing comparisons only on the remaining pairs to improve computational efficiency. Prominent filtering techniques are categorized into prefix-based, partition-based, and tree-based. We refer to [58] for details.


Prompt for LLM-based pair-wise ER

You are a worker with rich experience performing Entity Resolution tasks. You specialize in clustering and classification within ER. Do the two records refer to the same entity? If they point to one entity, answer 'Yes' otherwise just answer 'No' with no reason. Here are two records:

The first record: name:Alice Johnson, email:alice.johnson@gmail.com, phone:123-456-7890
The second record: name:A Johnson, email:alice.j@gmail.com, phone:123-456-7890

Yes !

You are a worker with rich experience performing ...

Figure 2: Example zero-shot prompt for pairwise ER

3 PRELIMINARIES

We discuss the background on the entity resolution (ER) problem (§3.1), followed by how pre-trained LLMs can be adapted for ER in a zero-shot prompting paradigm (§3.2).

3.1 Entity Resolution

An ER algorithm receives an input set of records $\mathcal{R}$ and returns a partition of them: $\mathbb{C} = \{C_1, C_2, \ldots, C_n\}$, such that $C_i \cap C_j = \emptyset$ for all i, j, and $\cup_i C_i = \mathcal{R}$. Each C_i is called a *cluster* or a *group* of $\mathcal{R}$ and represents a distinct real-world entity. If two records r_i, r_j refer to the same (different) entity, they are denoted as $r_i = r_j$ ($r_i \neq r_j$).

There are three key steps in entity resolution, *blocking/ filtering*, *matching*, and *combining* [15, 29, 57]. Blocking and filtering aim to separate almost dissimilar records and keep similar records in the same block to reduce the number of comparisons in the subsequent matching step and make the algorithm more efficient [59]. Matching determines whether the records in the same block refer to the same entity. Finally, combining creates the final entity clusters by inferring indirect matching relations following the matching step.

In pairwise ER, a pairwise similarity function is used over each pair of records to find the matching pairs. In this case, each pair is referred to as a *question* posed to the similarity function. An ER algorithm generally obeys *transitivity* and *anti-transitivity* rules, which can be employed in the matching and combining phases to further reduce the number of questions [5].

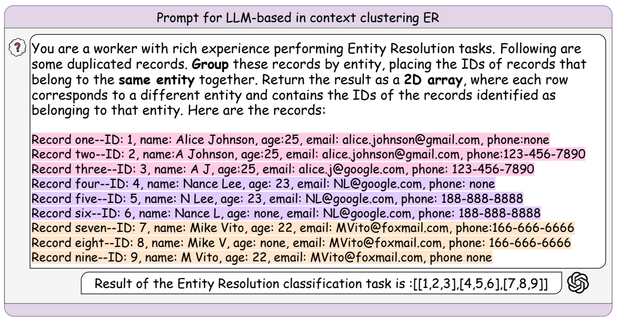
Transitivity. Given three records r_1, r_2, and r_3, if $r_1 = r_2$ and $r_2 = r_3$, then we have $r_1 = r_3$.

Anti-transitivity. Given three records r_1, r_2, and r_3, if $r_1 = r_2$ and $r_2 \neq r_3$, then we have $r_1 \neq r_3$.

A clustering $\mathbb{C}$ of the input set of records $\mathcal{R}$ is transitively closed. For instance, if a pairwise similarity function decides $r_1 = r_2$ and $r_2 = r_3$, then we can infer $r_1 = r_3$ without employing the similarity function on the pair (r_1, r_3). This reduces the number of questions.

3.2 LLM: Adaptation Techniques

Many machine learning approaches that utilize deep neural networks have been proven to be good similarity functions during the matching step [35, 42, 55]. Specifically, traditional deep learning-based methods receive a pair of records as input and outputs whether the two records belong to the same real-world entity. However, they require high-quality training data and task-specific supervision,


Prompt for LLM-based in context clustering ER

You are a worker with rich experience performing Entity Resolution tasks. Following are some duplicated records. **Group** these records by entity, placing the IDs of records that belong to the **same entity** together. Return the result as a **2D array**, where each row corresponds to a different entity and contains the IDs of the records identified as belonging to that entity. Here are the records:

Record one--ID: 1, name: Alice Johnson, age:25, email: alice.johnson@gmail.com, phone:none
Record two--ID: 2, name:A Johnson, age:25, email: alice.johnson@gmail.com, phone:123-456-7890
Record three--ID: 3, name: A J, age:25, email: alice.j@google.com, phone: 123-456-7890
Record four--ID: 4, name: Nance Lee, age: 23, email: NL@google.com, phone: none
Record five--ID: 5, name: N Lee, age: 23, email: NL@google.com, phone: 188-888-8888
Record six--ID: 6, name: Nance L, age: none, email: NL@google.com, phone: 188-888-8888
Record seven--ID: 7, name: Mike Vito, age: 22, email: MVito@foxmail.com, phone:166-666-6666
Record eight--ID: 8, name: Mike V, age: none, email: MVito@foxmail.com, phone: 166-666-6666
Record nine--ID: 9, name: M Vito, age: 22, email: MVito@foxmail.com, phone none

Result of the Entity Resolution classification task is :[[1,2,3],[4,5,6],[7,8,9]]

Figure 3: Example zero-shot prompt for in-context clustering

which are expensive. Recently, large language models (LLMs) that are pre-trained on massive-scale text corpora in a self-supervised manner are delivering good quality results in the matching part of ER tasks [26, 36, 43, 53, 63, 79]. LLMs, due to the vast amount of prior knowledge, have shown great ability in zero-shot tasks [9]. Meanwhile, LLMs also show superior performance in classification problems [12, 70], enabling them to have great potential in entity disambiguation tasks such as entity resolution. Figure 2 provides an example of an LLM prompt for pairwise entity resolution.

4 PROBLEM: IN-CONTEXT CLUSTERING

We first introduce our novel record set for in-context clustering and its key factors (§4.1), followed by experiments on how these factors impact clustering on individual record sets (§4.2). Based on these results, we formally define the end-to-end ER problem (§4.3).

4.1 Record Set and Key Factors

Different from pairwise input schemes used by recent LLM-based ER solutions [26, 43], we combine the advantages of LLMs and innovatively propose to input a record set. In particular, we pack a number of records in a set as the input prompt and ask the LLM to output a clustering of the records in this set. Such an in-context clustering scheme can make good use of the LLM's ability to process batch data [87], as well as cluster text data preserving semantic similarity [88]. Direct clustering of larger record sets reduces the number of questions to be prepared, subsequently minimizing the number of LLM API calls, the number of LLM tokens, and thereby obtaining high-quality answers at a lower cost and time.

Given a set of records as input to the LLM, the goal is to partition the records into clusters. We refer to this paradigm as "*in-context clustering*". Three key factors may influence the performance of the LLM within this clustering-based scheme: set size, set diversity, and set variation. The definitions are given below.

i) Set size: the number of records in an input set.

ii) Set diversity: the number of distinct entities (i.e., clusters) within an input set.

iii) Set variation: the variability in cluster sizes within an input set, quantified by the **coefficient of variation**:

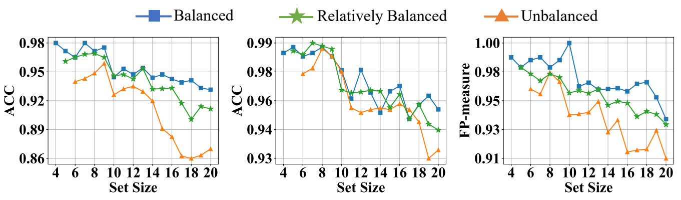
$$variation(S) = \frac{\sigma(S)}{\mu(S)} \tag{1}$$

where $\sigma(S)$ and $\mu(S)$ denote the standard deviation and mean of the cluster sizes, respectively, in the input set S. This metric measures the degree of variability in cluster sizes relative to the mean.

Additionally, our follow-up experimental results demonstrate the importance of *ordering similar records sequentially inside a record set*, this enhances an LLM's in-context clustering performance by facilitating better context differentiation [9]. Similar findings were also demonstrated by prior RAG work [48], which shows that ordering semantically similar sentences together improves LLM output.

Figure 3 presents an example of an LLM prompt for our novel in-context clustering-based entity resolution. Instead of pairwise ER, the prompt explicitly asks to cluster the records in an input set.

Example 3. *As shown in Figure 3, we use an LLM to group a record set containing 9 records, thus the* ***set size*** *is 9. The* ***set diversity*** *of this record set is 3, since it contains three distinct entities (clusters), which are represented by three different colors. The mean of cluster sizes in this record set is* $(3+3+3)/3 = 3$*, while the standard deviation of cluster sizes in the record set is* $\sqrt{\frac{(3-3)^2+(3-3)^2+(3-3)^2}{3}} = 0$*, indicating a* ***set variation*** *of 0, i.e., no variation in relation to three cluster sizes.*



(a) AS (b) Cora (c) Alaska

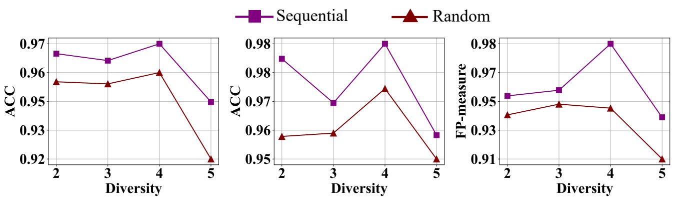
Figure 4: Clustering performance vs. set size and variation

4.2 Empirical Evaluations for the Key Factors

We examine the impact of varying **set size** (S_s), **set diversity** (S_d), and **set variation** (S_v) on the LLM's performance for in-context clustering of individual record sets. We further explore how the order of records in a set affects the LLM's performance by implementing: **sequential record order**, where records belonging to the same entity are placed consecutively in the record set; and **random record order**, where the records are randomly placed. These analyses help us identify the *optimal information density and structure* that LLMs can handle within a single API call, hence *guiding our design* for creating effective record sets and fully utilizing the knowledge provided by blocking techniques (§ 5).

We define three levels of set variations (S_v) to represent different degrees of imbalance within the record sets: **balanced** ($S_v < 0.3$), **relatively balanced** ($0.3 \leq S_v \leq 0.7$), and **unbalanced** ($S_v > 0.7$). For each fixed S_s, S_d, S_v, and record ordering, we uniformly at random select 200 questions (i.e., record sets) to query the LLM, provided that the total number of questions available under the given parameter settings exceeds 200. The ground truth for the questions corresponds to the correct clustering of the records within each record set. More detailed experimental settings, evaluation metrics, and dataset descriptions can be found in § 6.

Figure 4 illustrates the in-context clustering performance of the LLM on individual record sets under varying S_s and S_v. It can be viewed that LLM's performance *improves with smaller set variations*, indicating that the more evenly distributed the proportions of records are among different entities in a record set, the better the performance across all datasets and metrics, regardless of the set size. We also observe that the LLM performance *remains relatively stable and well* for the balanced, relatively balanced, and unbalanced settings as the set size increases from 4 to 9. However, once the *set size exceeds 9 or 10*, the in-context clustering performance of the



(a) AS (b) Cora (c) Alaska

Figure 5: Clustering performance vs. set diversity, ordering

LLM record sets *declines sharply*. This can be attributed to the fact that as the set size grows, the LLM must distribute its focus across more records, leading to a dilution of semantic understanding and a higher risk of errors. Furthermore, larger record sets exacerbate the difficulty in maintaining consistent contextual relationships, which is crucial for accurate clustering.

Notably, for the same set variation setting, the trends across ACC and FP-measure are *generally consistent* since these metrics all evaluate, to some degree, the similarity between the clustering results and the ground truth. Therefore, it can be concluded that *the optimal set size is 9*, and the variation within the record set should be *as slight as possible* for better in-context clustering performance.

Fixing the optimal values of S_s and S_v based on previous experiments, we further investigate the impact of S_d and the ordering of records within record sets on the performance of LLM-based in-context clustering (Figure 5). We notice that when S_d is set to 4, the in-context clustering performance of the LLM is better across all datasets in terms of ACC and FP-measure compared to other diversity settings. Thus, we consider $S_d = 4$ *as our optimal diversity setting*. With 4 entities, the clustering task provides enough granularity to differentiate entities while avoiding excessive sparsity or overlap in record distribution. In contrast, fewer entities (e.g., 2 or 3) might result in overly homogeneous clusters that fail to capture subtle differences, while a larger number of entities (e.g., 5) might introduce higher complexity and noise, negatively impacting the LLM's ability to identify distinct clusters accurately. Meanwhile, it is observed that the LLM achieves better in-context clustering performance when the *sequential record order* is used. This improvement can be attributed to the sequential arrangement of similar records, which likely enhances the coherence of contextual information presented to the LLM. By maintaining entity-related records in close proximity, this ordering reduces potential context-switching overhead and provides a more structured input sequence [48], thereby enabling the model to better discern subtle patterns and relationships essential for accurate clustering.

In conclusion, for the LLM's in-context clustering performance, our optimal configuration includes a set size of 9, a balanced distribution (S_v close to 0), a set diversity of 4, and the sequential record order. Larger set sizes should be avoided, as the LLM's performance under extremely unbalanced ratios deteriorates rapidly beyond this point. Conversely, smaller set sizes should also be avoided, as maximizing the set size is crucial for reducing API call frequency, improving efficiency and minimizing token usage and cost.

4.3 Problem Statements for End-to-end ER

We are now ready to define our main problems.

Problem 1. (Next Record Set Selection). *This problem is relevant for an LLM's in-context clustering during the entity matching phase. Given a set of records $\mathcal{R}$, the current clustering result $\mathbb{C}$, and predefined constraints on the set size (S_s), diversity (S_d), and variation (S_v), the Next Record Set Selection problem identifies a subset of records $\mathcal{R}^* \subseteq \mathcal{R}$ satisfying the constraints $\langle S_s, S_d, S_v \rangle$ with the objective of minimizing the number of LLM API calls by leveraging the identified transitivity and anti-transitivity relationships.*

Problem 2. (Cluster Combination). *Given the current clustering result $\mathbb{C}$ and the LLM's in-context clustering output $\mathbb{C}^*$ derived from a record set $\mathcal{R}^*$, the Cluster Combination problem updates $\mathbb{C}$ by combining the clusters in $\mathbb{C}$ based on the merge decisions of the LLM over $\mathbb{C}^*$, leveraging the identified transitivity and anti-transitivity relationships. This problem is relevant in combining phase to gradually refine the LLM's clustering outputs and obtain the final ER result.*

The number of record sets generated by our algorithm determines the number of LLM API calls. Since each LLM API call roughly consists of the same number of tokens (e.g., see Figure 3), and an LLM's monetary cost and inference time are determined by the total number of input and output tokens, minimizing the number of LLM API calls also reduces the number of LLM tokens, monetary cost, as well as the end-to-end ER time. We discuss our solutions in the following section. In particular, the next record set selection is considered in §5.2, while cluster combination is detailed in §5.3. Notice that the existing pairwise ER scheme is a *special case* of our in-context clustering-based ER paradigm when the record set size $S_s = 2$. Therefore, our problems and solutions generalize the state-of-the-art ER approaches.

5 SOLUTION: END-TO-END ER

We present an overview of our solution before giving more details. Our key technical contributions are: (1) an effective Next Record Set Creation (NRS) algorithm that selects optimal record sets for in-context clustering from the blocks produced by preceding blocking techniques; (2) a guardrail technique, Misclustering Detection Guardrail (MDG) for assessing and subsequently improving the accuracy of the LLM's in-context clustering outputs; and (3) an efficient heuristic algorithm, Cluster Merge (CMR) for merging the in-context clustering outputs of record sets. Based on the aforementioned novel algorithms, we thoroughly explore the design space of clustering-based entity resolution leveraging LLMs.

Figure 6 illustrates the workflow of our entity resolution framework. The process begins by applying suitable blocking and filtering functions to input records that generate initial blocks (§5.1). The blocks are processed by the Next Record Set Creation (NRS) algorithm (Algorithm 1) to create the record sets that are in-context clustered via an LLM (§5.2). The clustering results are validated and improved using the Misclustering Detection Guardrail (MDG) algorithm (Algorithm 2). Finally, the clusters are combined by hierarchically creating record sets and performing their in-context clustering with the LLM while leveraging identified transitivity and anti-transitivity relationships (§5.3). This task is implemented using the Cluster Merge (CMR) algorithm (Algorithm 3). The process of hierarchical cluster merging continues until an exit condition is met, yielding the final ER result (§5.4).

5.1 Filtering and Blocking

We briefly introduce a number of filtering and blocking methods that we employ as the first step in ER. Since the input comprises

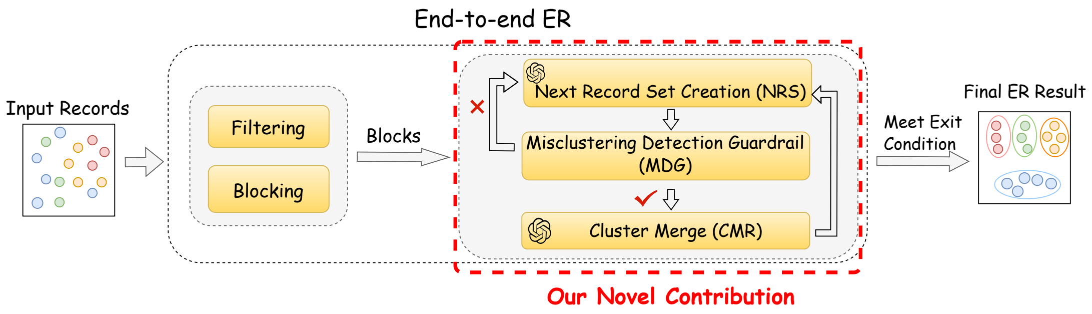


Figure 6: overview of our end-to-end entity resolution solution

a collection of records $\mathcal{R}$, a naïve method for ER is pairwise comparing, the complexity of which is $O(|\mathcal{R}|^2)$. To reduce the heavy computational cost, filtering and blocking are used [34, 50, 60]. Blocking clusters similar entities into a set of blocks, while filtering generally applies similarity joins to compute a set of candidate matches for each record. Though blocking and filtering are not the main technical contributions in this work, we are the first to explore their trade-offs about both (1) LLM-based ER, i.e., entity resolution task exploiting a large language model [26, 43]; and (2) clustering-based ER, that is, entity resolution by repetitively clustering a subset of records instead of pairwise comparing.

We consider the following filtering and blocking strategies.

Filtering-based Block Creation. Similarity join filters records with similarity scores above a specific threshold [7, 8]. We use Jaccard similarity for joins to find record pairs that can be treated as near duplicates. However, filtering is expensive due to $O(|\mathcal{R}|^2)$ time complexity of comparing *all* record pairs. [84] introduces positional filtering, which leverages the ordering of tokens within a record to reduce the candidate records that it needs to compare. Assuming that for each record, only k candidate records are obtained through pruning, the time complexity is reduced to $O(|\mathcal{R}|\cdot k)$. We empirically set the similarity threshold b_t in the filtering step following [28]. Specifically, we apply filtering with thresholds ranging from 0.05 to 0.95 in steps of 0.05 and select the threshold that maximizes the F1-score of clustering on a validation dataset. While validation data relies on ground truth, such information may be unavailable or difficult to obtain. In this scenario, we exploit LLM-based clustering results over a certain number of record subsets as ground truth.

Locality Sensitive Hashing-based Block Creation. Distributed representations have shown impressive results in ER tasks by capturing semantic similarities, such as the link between "p53" and "cancer" [30]. Metrics like cosine distance can estimate the similarity between embeddings but require $O(|\mathcal{R}|^2)$ comparisons, making direct computation impractical on large-scale datasets. Locality-sensitive hashing (LSH) [23] efficiently addresses this concern by mapping similar embeddings into the same bucket with high probability, reducing complexity to $O(|\mathcal{R}|\cdot k)$, where k is the number of buckets. However, due to the stochastic nature of hash functions, dissimilar records may still fall into the same bucket. To mitigate this issue, we apply a cleaning strategy. We retain only pairs with similarity exceeding a threshold b_t to purify the initial blocks, where b_t is determined using the same approach as above.

Canopy Blocking [49]. This utilizes two thresholds b_s and m_s, with $b_s \geq m_s$. Record pairs having similarity above b_s are confidently assigned to the same block using a cheap similarity metric, such as inverted index-based edit distance on a single attribute. Record pairs with similarity above m_s are placed into the same canopy, allowing overlaps between canopies. Next, a more refined similarity metric, such as Jaccard similarity over all attributes, is applied to measure the similarity between record pairs within the canopies. These measurements guide the merging of blocks leveraging transitivity: If a record pair between two blocks is determined to match, the two blocks are merged. This process iteratively combines blocks to produce the final blocking result. Although forming the initial canopies still requires quadratic complexity, it relies on a highly efficient computation method. As demonstrated in [49], this approach is orders of magnitude faster than more refined similarity metrics in terms of time cost. We use the same idea in [49] to obtain m_s and b_s, which involves tuning these parameters empirically on separate validation datasets. When validation sets are unavailable, as before we exploit LLM-based clustering results over a certain number of record subsets as ground truth.

While we have implemented the aforementioned filtering and blocking methods due to their simplicity, it is worth noting that in the future, one can explore more sophisticated approaches, e.g., progressive blocking [27]. Since our main technical innovations are in-context clustering for matching (§5.2) and combining (§5.3), we do not consider more advanced blocking in this work.

5.2 In-context Clustering of a Record Set

We develop novel in-context clustering for the matching phase.

Next Record Set Creation Algorithm. Consider the records $\mathcal{B}_{remain}$ in a block that are not used in forming a record set yet. We design an algorithm NRS (Algorithm 1) to create the n̲ext r̲ecord s̲et with $\mathcal{B}_{remain}$. If $|\mathcal{B}_{remain}|$ is smaller than a predefined set size (S_s), the next record set is created by grouping similar records sequentially, which improves an LLM's in-context clustering performance by facilitating better context differentiation, as shown in our experiment (§4.2) (Lines 2-6). Otherwise, the next record set generation is optimized based on the empirical findings from §4.2. Our results show that an LLM's in-context clustering accuracy depends on the record set design. In particular, we select an optimal record set size S_s, diversity S_d, and variation S_v based on experimental results. We then apply the elbow method and k-means [18] to perform a preliminary clustering of records in $\mathcal{B}_{remain}$, while also assessing its diversity k. Next, we construct the next record set while ensuring the set size constraint S_s. For the record set, we minimize S_v as much as possible and aim to meet the S_d requirement (Lines 8-17).

Experimental results indicate that the optimal values are $S_s = 9$ and $S_d = 4$, while LLM-based in-context clustering achieves better performance when S_v is minimized, ideally approaching zero.

Algorithm 1 Next record set creation (NRS)

```
Input: remaining records in a block $\mathcal{B}_{remain}$, $S_s$, $S_d$, $S_v$
Output: record set $\mathcal{R}_{set}$
1: $\mathcal{R}_{set} \leftarrow \emptyset$
2: if $|\mathcal{B}_{remain}| \leq S_s$ then
3:     $r_{pre} \leftarrow$ first element in $\mathcal{B}_{remain}$
4:     while $|\mathcal{B}_{remain}| > 0$ do
5:         add $r_{pre}$ to $R_{set}$ and remove it from $\mathcal{B}_{remain}$
6:         $r_{pre} \leftarrow$ update $r_{pre}$ w/ its most similar record from $\mathcal{B}_{remain}$
7: else
8:     $k \leftarrow$ compute diversity of $\mathcal{B}_{remain}$ using the elbow method
9:     $\mathbb{B}_{remain,k} \leftarrow$ perform $k$-means on $\mathcal{B}_{remain}$
10:    $target_{size} \leftarrow \lfloor S_s/S_d \rfloor$
11:    for $\mathcal{B}_{remain,i}$ in $\mathbb{B}_{remain,k}$ do
12:        if $|\mathcal{R}_{set}| < S_s$ and $|\mathcal{B}_{remain,i}| \geq target_{size}$ then
13:            select $target_{size}$ records from $\mathcal{B}_{remain,i}$ and add to $\mathcal{R}_{set}$
14:            delete those records from $\mathcal{B}_{remain}$
15:    while $|\mathcal{R}_{set}| < S_s$ do
16:        $r_{set} \leftarrow$ find record from $\mathcal{B}_{remain}$ that minimally increases $S_v$
17:        add $r_{set}$ to $\mathcal{R}_{set}$ and delete from $\mathcal{B}_{remain}$
18:    order similar records together in $\mathcal{R}_{set}$ (similar to Lines 3-6)
19: return $\mathcal{R}_{set}$
```

Algorithm 2 Misclustering Detection Guardrail (MDG)

```
Input: In-context clustering result $\mathbb{C}_{set}$ of a record set
Output: whether in-context clustering result is acceptable ($True/False$)
1: for cluster $C_i$ in $\mathbb{C}_{set}$ do
2:     for record $r_j$ in $C_i$ do
3:         if intra-cluster sim. of $r_j$ < inter-cluster sim. of $r_j$ then
4:             return $False$
5: return $True$
```

Mitigating In-context Clustering Errors of LLMs. In practice, LLMs are prone to hallucination [33], leading to seemingly plausible, yet factually incorrect contents. As a result, LLM-based in-context clustering outcomes for record sets may not be entirely reliable. To ensure high accuracy of the output, we further devise a misclustering detection guardrail (MDG) algorithm to verify the clustering results of the LLM, as detailed in Algorithm 2. We first define some key terms.

Definition 1. (Inter- and Intra-cluster Similarity) *Given a similarity function $F(\cdot)$ to measure the similarity between record pairs and a known record r, the intra-cluster similarity of r is defined as the minimum similarity between r and other records within the same cluster as r. The inter-cluster similarity of r is defined as the maximum similarity between r and records from other clusters.*

Specifically, for any cluster from the in-context clustering output, if the intra-cluster similarity of a record inside it is lower than its inter-cluster similarity, Algorithm 2 indicates the record to be *misclustered*; and hence, the result is not acceptable.

Record Set Regeneration. The proposed MDG algorithm evaluates the classification result $\mathbb{C}_{set}$ of an LLM. If the result is deemed unreasonable, we find all the records r that are mis-clustered. We find the most similar cluster for r based on its inter-cluster similarity and place it after the records from that cluster. In this way, a new record set is regenerated with the same set of records; however, the aforementioned process orders more similar records sequentially, thereby enhancing the LLM's in-context clustering accuracy.

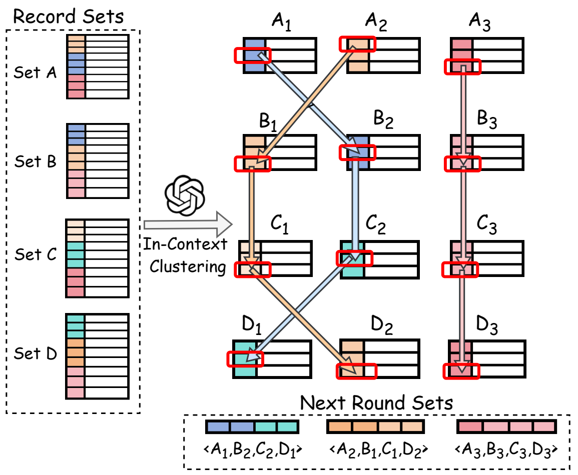


Figure 7: An example of the next round of record sets creation. For simplicity, consider that each of the record sets A, B, C, D generates three in-context clusters: $A = \{A_1, A_2, A_3\}$, $B = \{B_1, B_2, B_3\}$, etc. Assume that $S_s \geq 4$ and $S_d = 1$ for the sake of brevity. Then, the next round of record sets should pack the most similar clusters from A, B, C, and D, forming three new sets for the next round as follows: $\{A_1, B_2, C_2, D_1\}$, $\{A_2, B_1, C_1, D_2\}$, and $\{A_3, B_3, C_3, D_3\}$. Each cluster appears exactly once in the next round of record sets to minimize the overall LLM API calls. Clusters from the same record set (e.g., both A_1 and A_2) are not packed together in the same record set of the next round owing to anti-transitivity.

5.3 Hierarchical Cluster Merging

Leveraging the optimal set size $S_s = 9$ as stated earlier, each block is divided into several record sets (e.g., a block of size 27 is split into 3 record sets). Each record set is then in-context clustered independently by the LLM. For example, if record set A is clustered into subsets $\{A_1, A_2, A_3\}$, record set B into $\{B_1, B_2\}$, and record set C into $\{C_1, C_2, C_3, C_4, C_5\}$, it becomes necessary to merge subsets across record sets to obtain the final partition.

To achieve this, we note that clusters within the same record set (e.g., A_1 and A_2) are already disjoint (i.e., belong to different entity groups) due to the LLM's in-context clustering and hence exhibit non-transitive relationships. However, clusters from different record sets (e.g., A_1 and B_1) may belong to the same entity group. Merging clusters across different record sets can be formulated as a K-dimensional maximum matching problem (analogous to a 3-dimensional maximum matching problem, which is proven to be **NP**-Hard [16]), where K is the number of record sets.

Problem 3 (Cluster Merging). *Given K record sets and their in-context clustering outputs, the Cluster Merging problem treats each output cluster as a "new" individual record for the next round and packs them optimally to construct record sets for the LLM's next round of in-context clustering. The problem is subject to the following conditions: (1) Each cluster (i.e., the corresponding "new" individual record) is selected exactly once in the next round of record sets to minimize the total number of LLM API calls. (2) Clusters from the same record set are never packed together in the same record set of the next round owing to anti-transitivity. (3) The next round of record* *sets should also ensure the set size constraint S_s, set diversity S_d, and minimize S_v as much as possible.*

The complexity of evaluating all combinations in the cluster merging problem is $O(K \cdot n!)$, where n is the average number of clusters per record set. This results in an exponential time cost, making it impractical for real-world applications with large blocks and multiple record sets.

To address this issue, we propose a heuristic Cluster Merge (CMR) algorithm (Algorithm 3) that avoids exhaustive enumeration. Specifically, for each cluster of a given record set, we identify its most similar cluster from the next record set based on a defined similarity measure. For instance, consider two record sets A and B with their in-context clustering results $A = \{A_1, A_2, A_3\}$, $B = \{B_1, B_2\}$. If A_1 is most similar to B_2, these two clusters are packed together. This process is repeated iteratively across all record sets (e.g., finding a cluster of the next record set C that is most similar to the group A_1B_2), gradually forming a new record set for the next round of in-context clustering, while satisfying S_s, S_d, and S_v requirements. Figure 7 shows an example of our cluster merging procedure that constructs record sets for the next round of clustering.

For simplicity, assume $K \leq S_s$, where K denotes the number of record sets in the current round, and S_s is the optimal record set size. For ease of understanding, we also assume that each of these record sets generates n in-context clusters by the LLM. Algorithm 3 outlines the process of constructing record sets for the next round of in-context clustering. Given K record sets, the algorithm first generates clusters for each record set using the LLM, then replaces each cluster with a representative "new" record. This replacement is possible because the records within a cluster represent the same entity and satisfy transitivity. The representative record can either be uniformly chosen at random or selected as the one with the smallest distance from the average embedding of the records in that cluster (Lines 1-3). Then, it partitions the K record sets into groups of size $\lceil K/S_d \rceil$, iterating over these groups to construct a record set $\mathcal{R}_{next}$ for the next round (Lines 5-6). For each group, it selects an unselected cluster C from the first record set of the group and uses its representative record as the initial record (Lines 7-8). Subsequently, for each remaining record set in the group, it identifies the most similar unselected cluster C' to the previously selected cluster C and adds its representative record to $\mathcal{R}next$ (Lines 9-11). Finally, the resulting record set is returned for the next round of in-context clustering (Line 12). This process also ensures that (1) similar elements of $\mathcal{R}next$ are sequentially ordered; and (2) we meet the S_d requirement, while minimizing S_v as much as possible.

Considering an average of n clusters from a record set, the complexity of comparing clusters between two record sets is $O(n^2)$. Since there are K record sets in the current round, the total complexity of this greedy algorithm is $O(K \cdot n^2)$ in the current round. This represents a significant reduction from the original exponential complexity $O(K \cdot n!)$, making it practically feasible to construct a record set for the next round of in-context clustering while maintaining satisfactory accuracy.

5.4 End-to-End ER Solution

Algorithm 4 presents our end-to-end entity resolution framework outlining the key steps, given the input set of records $\mathcal{R}$. The algorithm begins by applying blocking functions to $\mathcal{R}$, generating initial set of blocks $\mathbb{B}$ (§5.1), after which we build a disconnected graph based on it to keep track of similar records. For each block in $\mathbb{B}$, record sets are generated using the NRS algorithm (Algorithm 1), they are in-context clustered leveraging an LLM, with acceptable outputs verified by the MDG algorithm (Algorithm 2) given in §5.2. Subsequently, the clustering outputs of these record sets are merged hierarchically using the CMR algorithm (Algorithm 3) and next rounds of in-context clustering (§5.3).

Algorithm 3 Cluster Merge (CMR): Depicting the creation of one record set for the next round

```
Input: $K$ ($\leq S_s$) record sets $\{\mathcal{R}_1, \ldots, \mathcal{R}_K\}$ each with $n$ clusters; $S_s, S_d, S_v$
Output: one record set $\mathcal{R}_{next}$ for the next round with size $K$
1:  for $i \leftarrow 1$ to $K$ do
2:      $\mathbb{C}_i \leftarrow n$ output clusters of record set $\mathcal{R}_i$
3:      replace each cluster $C \in \mathbb{C}_i$ by a representative record $r_C \in C$
4:  $\mathcal{R}_{next} \leftarrow \phi$
5:  for $j \leftarrow 1$ to $S_d$ do
6:      $w \leftarrow (j-1) \cdot \lceil K/S_d \rceil + 1$
7:      select a previously unselected cluster $C \in \mathbb{C}_w$
8:      insert $C$ (i.e., its representative record $r_C$) into $\mathcal{R}_{next}$
9:      for $i \leftarrow (j-1) \cdot \lceil K/S_d \rceil + 2$ to $\min\{j \cdot \lceil K/S_d \rceil, K\}$ do
10:         find cluster $C' \in \mathbb{C}_i$, previously unselected & most similar to $C$
11:         Insert $C'$ (i.e., its representative record $r_{C'}$) into $\mathcal{R}_{next}$
12: return $\mathcal{R}_{next}$
```

Algorithm 4 End-to-end ER

```
Input: A set of records $\mathcal{R}$
Output: final partition $\mathbb{C}$ of $\mathcal{R}$
1:  $\mathbb{B} \leftarrow$ apply blocking functions on $\mathcal{R}$ and get initial blocks
2:  generate record sets for each block $\mathcal{B} \in \mathbb{B}$ using NRS algorithm
3:  in-context clustering of record sets with LLM
4:  check correctness of in-context clustering by MDG algorithm
5:  record sets regeneration to improve in-context clustering if needed
6:  while True do
7:      generate record sets for next round using Cluster Merge algorithm
8:      in-context clustering of record sets with LLM
9:      check correctness of in-context clustering by MDG algorithm
10:     record sets regeneration to improve in-context clustering
11:     if Exit condition then
12:         Break
13: Use each singleton cluster to reconstruct the final partition $\mathbb{C}$
14: return $\mathbb{C}$
```

Exit Condition. We keep generating hierarchical record sets for in-context clustering and subsequent cluster merging. The process continues until no more clusters can be merged owing to anti-transitivity. Here, anti-transitivity is identified by in-context clustering of the record sets. In particular, consider a current round when the in-context clustering outputs only singleton clusters, i.e., each cluster having only one element from the current round. The exit condition is thus satisfied. A naive method for the "final check" is to conduct pairwise comparisons of all singleton clusters to ensure that, indeed, no more merging is feasible, which costs quadratic time in the number of singleton clusters. During this final check, we, however, adopt a more efficient method by packaging multiple singleton clusters in a record set to reduce the number of LLM API calls, which is discussed in the following theoretical analysis.

Finally, we use each singleton cluster to reconstruct the full partition $\mathbb{C} = \{C_1, \ldots, C_n\}$ and complete the ER process.

Theoretical Analysis. We provide a theoretical analysis of the number of LLM API calls needed due to our in-context clustering-based end-to-end ER. For simplicity, let us examine one single block

Table 1: Dataset statistics: Rec. stands for records, ent. represents entities, attr. denotes attributes.

Datasets	Domain	#Rec.	#Ent.	#Attr. per Rec.	#Max. Rec. per Ent.
Alaska	Product	12k	1.48k	9	151
AS	Geo	2.26k	0.33k	1	47
Song	Music	4.85k	1.19k	7	8
Music 20K	Music	19.3k	10k	5	5
DBLP-Google	Citation	7.63k	2.35k	4	37
Cora	Citation	1.29k	0.11k	12	236
Citeseer-DBLP	Citation	9.13k	2.49k	6	8

with m records. Consider two extreme scenarios. **(1)** *All records belong to same entity.* For each hierarchy level, the number of records is reduced by a factor of the record set size (S_s). Hence, the total number of LLM API calls equals $(\lceil m/S_s \rceil + \lceil \lceil m/S_s \rceil / S_s \rceil + \ldots + 1)$ $\approx \lceil m/(S_s - 1) \rceil$. For larger S_s, the series converges quickly, and the total number of LLM API calls reduces significantly, justifying the superiority of our in-context clustering approach over pairwise comparisons (for pairwise comparisons, $S_s = 2$) in the end-to-end ER. **(2)** *All records belong to different entities.* For hierarchy level 0, there would be $p = \lceil m/S_s \rceil$ record sets, and the total number of LLM API calls at this hierarchy level is p. For simplicity, assume $m = p \cdot S_s$, that is, each record set has exactly S_s records. For the extreme case (2), the in-context clustering outputs only singleton clusters at hierarchy level 0 and satisfies the exit condition. For the "final check", each record in a record set needs to compare with all records from other sets, i.e., total pairwise comparisons needed for the final check is $\frac{m(m-S_s)}{2}$. To reduce the number of LLM API calls, we pack these pairwise comparisons into a certain number of record sets. A record set of size S_s enables $\frac{S_s(S_s-1)}{2}$ pairwise comparisons. Thus, one needs $\frac{m(m-S_s)}{S_s(S_s-1)}$ records sets, and subsequently, those many LLM API calls for the final check. In summary, for the extreme case (2), the total number of LLM API calls considering both hierarchy level 0 and the final check is $\frac{m}{S_s} + \frac{m(m-S_s)}{S_s(S_s-1)} = \frac{m(m-1)}{S_s(S_s-1)}$. Clearly, this is much smaller than total $\frac{m(m-1)}{2}$ comparisons required in pairwise questioning-based ER for the extreme case (2). Since all practical scenarios are between these two extreme cases, our theoretical analysis demonstrates that our design significantly reduces the number of LLM API calls compared to pairwise ER.

6 EXPERIMENTAL RESULTS

In this section, we empirically evaluate our proposed algorithms as well as the competing approaches. All methods are implemented in Python and executed on a Linux server with 2.20 GHz CPU and 128GB of RAM. Our code and datasets are available at [4].

6.1 Experimental Setup

Datasets. We utilize seven real-world entity resolution (ER) datasets spanning various domains, including e-commerce products, academic citations, geography, and music. Details of these datasets are provided in Table 1. Specifically, CORA is a text-based dataset comprising duplicate references to scientific publications. For this study, we use the structured CSV version provided by [56], with the raw data accessible online[2]. *Alaska* [17] is an e-commerce product dataset collected from multiple websites and previously featured in the SIGMOD 2020 programming contest [20]. *DBLP-Google [51]* and *Citeseer-DBLP* belong to the citation domain, containing bibliographic duplicates sourced from DBLP, Google Scholar, and Citeseer. The *Song* dataset represents the music domain and includes music information from various platforms, with variations in attributes such as duration and release year. Both *Song* and *Citeseer-DBLP* can be found here[3]. Lastly, the *AS* and *Music 20K* datasets are widely recognized for Dirty ER benchmarks [68].

Evaluation Metrics. We select several widely-used metrics to evaluate the in-context clustering results, as well as the final ER clustering results. These metrics are FP-measure (a variant of the F-measure), Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Score (ARI) [2, 11, 81, 86]. All experiments are repeated three times to ensure robustness, and we report the average values over three runs. Due to space limitations, we present results for ACC and FP-measure in the main paper. The complete experimental results are available in the full version [4].

The definition of ACC is as follows: Let $\mathbb{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m\}$ represent the ground truth clusters and $\mathbb{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ denote the predicted clusters. The total number of records in both sets is $|\mathcal{R}| = \bigcup_i |\mathcal{X}_i| = \bigcup_i |\mathcal{Y}_i|$. The accuracy (ACC) is defined as:

$$\text{ACC} = \frac{\text{CorrectCount}}{|\mathcal{R}|} \tag{2}$$

$$\text{CorrectCount} = \sum_{\mathcal{X}_j \in \mathbb{X}} \sum_{x \in \mathcal{X}_j} \mathbb{I}(\exists y \in \mathcal{Y}_j', y = x) \tag{3}$$

where $\mathbb{Y}' = \{\mathcal{Y}_1', \ldots, \mathcal{Y}_m'\}$ is obtained by reordering $\mathbb{Y}$ based on their intersection sizes with the predicted clusters, and $\mathbb{I}(\cdot)$ is the indicator function that equals to 1 if the condition satisfies.

FP-measure evaluates clustering results from the perspective of homogeneity and stability. It is the harmonic mean of purity and inverse-purity.

$$\textit{purity}\,(\mathbb{X},\mathbb{Y}) = \sum_{\mathcal{X}_i \in \mathbb{X}} \frac{|\mathcal{X}_i|}{|\mathcal{R}|} \max_{\mathcal{Y}_i \in \mathbb{Y}} \textit{Overlap}(\mathcal{X}_i, \mathcal{Y}_i) \tag{4}$$

$$\textit{inverse-purity}\,(\mathbb{X},\mathbb{Y}) = \sum_{\mathcal{Y}_i \in \mathbb{Y}} \frac{|\mathcal{Y}_i|}{|\mathcal{R}|} \max_{\mathcal{X}_i \in \mathbb{X}} \textit{Overlap}(\mathcal{Y}_i, \mathcal{X}_i) \tag{5}$$

$$\textit{Overlap}(\mathcal{X}_i, \mathcal{Y}_i) := \frac{|\mathcal{X}_i \cap \mathcal{Y}_i|}{|\mathcal{X}_i|} \tag{6}$$

Finally, we derive the FP-measure:

$$\textit{FP-measure}(\mathbb{X}, \mathbb{Y}) = \frac{2}{1/\textit{purity}(\mathbb{X}, \mathbb{Y}) + 1/\textit{inverse-purity}(\mathbb{X}, \mathbb{Y})} \tag{7}$$

6.2 Performance on End-to-End ER

We use `gpt-4o-mini` for end-to-end tests. For embedding generation in blocking, we adopt widely-used `all-MiniLM-L6-v2` model[4].

[2]https://www.gabormelli.com/RKB/CORA_Citation_Benchmark_Task.
[3]https://pages.cs.wisc.edu/~anhai/data.
[4]https://www.sbert.net/

Table 2: Comparison with pairwise matching-based ER

Metrics	*Cora*		*Alaska*		*AS*	
	$S_s = 2$ (*pairwise*)	$S_s = 9$ (*clustering*)	$S_s = 2$ (*pairwise*)	$S_s = 9$ (*clustering*)	$S_s = 2$ (*pairwise*)	$S_s = 9$ (*clustering*)
ACC	0.88	**0.90**	0.81	**0.82**	**0.70**	**0.70**
FP-measure	0.67	**0.71**	0.78	**0.79**	0.60	**0.63**
Cost (USD)	0.67	**0.03**	0.43	**0.15**	0.08	**0.02**
Tokens (M)	3.45	**0.12**	2.29	**0.73**	0.35	**0.07**
Time (min)	297.27	**5.42**	241.31	**39.57**	77.2	**8.01**
# API Calls (K)	30.23	**0.28**	24.54	**2.04**	7.85	**0.41**

Table 3: Number of records set in each hierarchy level

Dataset	Level 0	Level 1	Level 2	Level 3	Level 4	Level 5
Cora	183	76	15	5	/	/
Alaska	1312	604	101	20	5	1
AS	251	107	48	6	1	/

6.2.1 *In-context clustering vs. pairwise questioning.*

We compare the performance of our in-context clustering-based ER ($S_s = 9$) with the pairwise matching method ($S_s = 2$) to demonstrate the effectiveness and efficiency of in-context clustering that can process multiple records simultaneously. For fair comparison, we also adopt our guardrail strategy to mitigate LLM-induced errors for pairwise matching as stated in § 5.2. Table 2 shows that when S_s is set to 9, the final ER results achieved by our algorithm are nearly identical or even better than those of the pairwise matching approach. On the other hand, our in-context clustering significantly reduces the number of LLM API calls by 12-108×, token consumption by 3-28×, monetary cost by 3-22×, and end-to-end ER time by 6-55×, compared to pairwise matching.

Table 3 illustrates the distribution of the number of record sets across hierarchy levels, providing insights into our hierarchical clustering approach. For all datasets, the number of record sets decreases significantly from Level 0 to higher levels, and are nearly completed before Level 5, demonstrating the effectiveness of the hierarchical structure in progressively merging similar entities. Larger datasets generally require more hierarchy levels for processing, as evidenced by *Alaska* (the largest dataset) extending to Level 5, while *Cora* (the smallest dataset) completes by Level 3.

6.2.2 *Comparison with state-of-the-art.*

To demonstrate the superior performance of our in-context clustering-based ER, we compare it with several existing LLM-based entity resolution (ER) algorithms.

Booster [43]: Booster employs existing blocking methods to generate multiple different partitioning of the input records. The key contribution of Booster is its *next question selection* strategy: It iteratively queries the LLM with carefully selected record pairs that are most informative in distinguishing between different partitioning. After several rounds of refinement, the partitioning with the highest updated probability is selected as the final ER result. Unlike Booster, which selects one of the blocking-based partitioning, our method refines blocking-based partitioning through matching and combining with hierarchical clustering and misclustering detection. As a result, our final solution is not constrained by the initial blocking partitions, leading to more accurate and refined ER.

BQ [26]: The BQ algorithm optimizes traditional LLM-based ER by batching multiple pairwise matching questions into a single prompt, reducing API calls compared to querying each question individually. This approach leverages the LLM's ability to share context within a prompt, improving responses to later questions based on earlier ones. Furthermore, it introduces an enhanced demonstration selection strategy to better align examples with the batched questions, maximizing the effectiveness of few-shot learning. It then derives the final clusters by aggregating the responses from all batches. Assume that each batch in BQ packs k records using $\frac{k}{2}$ pairwise questions, thus a single batch can reduce at most $\frac{k}{2}$ comparisons. In contrast, our clustering-based approach with set size k reduces comparisons by up to $\frac{k(k-1)}{2}$ per record set. Hence, fewer API calls and lower monetary costs are needed by ours.

CrowdER+LLM [76]: CrowdER uses a clustering-based approach. It first filters irrelevant pairs and generates record sets to cover and resolve all uncertain record pairs. Obtained clusters are then merged using transitivity to refine the final results. Instead of generating all record sets upfront like theirs, we incrementally generate record sets and leverage transitivity and anti-transitivity to eliminate unnecessary sets. This reduces the number of API calls and lowers costs, while maintaining high accuracy. For comparison purposes, we use their two-tiered algorithm to group records into sets; however, instead of crowdsourcing, we leverage LLMs to

Table 4: End-to-end comparison of our in-context clustering-based ER with state-of-the-art LLM-based ER

Dataset	Metrics	In-context Cluster (ours)	Booster	BQ	CrowdER +LLM
Alaska	ACC	**0.82**	0.71	0.33	0.68
	FP	**0.79**	0.55	0.49	0.62
	Cost ($)	0.15	**0.02**	1.55	0.42
	Tokens (M)	0.73	**0.19**	5.59	2.04
	Time (s)	**2374.2**	2450.1	8798.9	6547.2
	# API Calls	**2043**	2606	8035	5845
AS	ACC	**0.70**	0.62	0.54	0.52
	FP	**0.63**	0.62	0.51	0.50
	Cost ($)	0.02	**0.01**	0.29	0.11
	Tokens (M)	0.07	**0.03**	0.34	0.37
	Time (s)	**480.6**	622.9	925.5	2356.2
	# API Calls	**413**	723	842	2084
Song	ACC	**0.72**	0.52	0.59	0.52
	FP	**0.78**	0.68	0.67	0.64
	Cost ($)	0.06	**0.02**	0.77	0.12
	Tokens (M)	0.22	**0.11**	1.98	0.43
	Time (s)	**933.2**	903.3	2581.5	1856.3
	# API Calls	**668**	921	2338	1247
Music	ACC	**0.71**	0.59	0.60	0.62
	FP	**0.61**	0.60	0.54	0.55
	Cost ($)	0.19	**0.02**	2.18	0.39
	Tokens (M)	0.90	**0.15**	8.96	1.82
	Time (s)	**2388.4**	2585.1	17515.8	4562.3
	# API Calls	**3859**	3915	17365	7782
DG	ACC	**0.81**	0.56	0.62	0.72
	FP	**0.70**	0.68	0.63	0.65
	Cost ($)	0.07	**0.02**	1.12	0.34
	Tokens (M)	0.37	**0.18**	3.92	1.79
	Time (s)	**1552.4**	2552.2	6052.2	7456.3
	# API Calls	**1285**	3085	6456	6504
Cora	ACC	**0.90**	0.75	0.62	0.51
	FP	**0.71**	0.60	0.56	0.61
	Cost ($)	0.03	**0.01**	1.45	0.07
	Tokens (M)	0.12	**0.06**	4.23	0.29
	Time (s)	**325.5**	605.4	4085.3	598.5
	# API Calls	**279**	698	4882	483
Cite-seer	ACC	**0.76**	0.72	0.64	0.60
	FP	**0.95**	0.78	0.79	0.69
	Cost ($)	0.03	**0.01**	0.63	0.08
	Tokens (M)	0.13	**0.05**	1.64	0.37
	Time (s)	**1360.8**	1585.2	6228.9	3895.6
	# API Calls	**1302**	2169	6420	3858

cluster those sets. We employ the same blocking technique and set size as in our work.

Implementation. For BQ , following their work [26], each prompt is equipped with 8 demonstrations. The question set for each block is constructed by forming all pairwise record comparisons within the block, where the blocks are obtained by using the same blocking technique as ours. For fairness, we set the batch size of BQ to five pairwise questions per prompt, which corresponds to ten records. This is approximately aligned with our in-context clustering, where each record set contains nine records, ensuring a comparable workload between the two approaches. Since BQ requires some labeled record pairs for demonstrations, we account for the labeling cost when calculating the monetary cost, following the same calculation standard as in [26]. For Booster, we also apply the same blocking method as ours and only adjust blocking parameters. We evaluate the ER results using ACC and FP-measure.

End-to-end Performance. As shown in Table 4, the following observations can be made: (1) Our algorithm outperforms all other methods across all datasets in terms of clustering quality metrics, including ACC and FP-measure, as well as operational costs such as the number of API calls and time consumption. Additionally, its monetary cost is competitive with those of the optimal LLM-based ER algorithm. (2) Booster relies heavily on the performance of the underlying blocking algorithm and uses LLM-generated outputs to identify the optimal partitioning from a predefined set of partitioning, without refining the partitions themselves. As a result, while Booster can select the best partitioning from a range of candidates, some of which perform relatively well, its ACC and FP scores remain suboptimal. In terms of monetary cost, however, Booster benefits from minimal token consumption, as it only selects the best partitioning without further modifying it. (3) BQ performs matching tasks within predefined blocks of questions. Our experiments reveal that BQ is both time- and cost-intensive. While BQ can slightly enhance the accuracy of pairwise matching through demonstration, the absence of verification for LLM-generated results leads to erroneous merging of clusters that should remain separate. This significantly undermines the ACC and FP-measure of the final clustering results, resulting in the lowest performance for ACC and FP-measure. Additionally, BQ exhaustively queries all record pairs within blocks and uses demonstrations for each querying prompt, leading to significantly higher token consumption—approximately 5 to 35 times greater than our approach—even after leveraging transitivity and anti-transitivity to reduce the number of queries. Moreover, BQ is a few-shot learning approach that incurs additional labeling costs, whereas our algorithm does not require any such labeling overhead. (4) CrowdER+LLM also employs clustering for ER and achieves the second-best performance in terms of ACC and FP-measure. However, as discussed in § 2, its HIT design does not align with the record set format optimized for LLMs and permits record overlap across HITs. This results in higher API call counts, token consumption, and monetary cost compared to our approach. Furthermore, the lack of verification for clustering results prevents the detection of LLM errors, leading to lower ACC and FP-measure relative to ours.

It is worth mentioning that some of the competing methods are also complementary to ours. For example, BQ's batch processing can be used in our case to batch clustering of multiple record sets simultaneously. Similarly, Booster's best partitioning selection can serve as an effective preprocessing step, providing high-quality initial blocks that our algorithm can refine for enhanced accuracy. These complementarities underscore the potential for integrating strengths from different methods to advance ER performance.

Table 5: Impact of blocking and filtering

Dataset	Metrics	w/o blocking	Filter	Canopy	LSH
Cora	ACC	0.62	0.81	0.67	0.90
	FP	0.58	0.78	0.60	0.71
	Cost ($)	0.33	0.03	0.06	0.03
	Tokens (M)	1.33	0.13	0.22	0.12
	Time (s)	3708.8	326.1	529.6	325.5
	API Calls	1996	301	440	279

6.3 Impact of Blocking and Filtering

We conduct experiments to explore the impact of different blocking methods on both performance metrics (e.g., ACC, FP-measure) and resource consumption (e.g., tokens, cost, time, and number of API calls) of our algorithm. As shown in Table 5, blocking significantly reduces resource usage compared to no blocking, while differences in performance vary across methods. LSH performs the best as it captures semantic similarities effectively, allowing for more precise entity matching. The Filtering method, while providing higher FP-measure, offers decent performance but is less effective than LSH. Canopy, while achieving slightly better FP-measure than no blocking, is too coarse-grained, leading to suboptimal clustering accuracy. Without blocking, the exhaustive comparison of all records leads to even the poorest performance, further emphasizing the necessity of blocking. Besides, all blocking methods drastically reduce resource consumption compared to no blocking. LSH- and Filtering-based blocking achieves lower resource consumption. Canopy, while reducing resources compared to no blocking, performs worse than both LSH and Filtering because its larger clusters lead to more redundant clustering. The results validate the effectiveness of blocking in reducing resource usage and the importance of properly selecting the suitable blocking method for improving performance.

7 CONCLUSION

In this work, we proposed an end-to-end in-context clustering-based entity resolution solution utilizing the Large Language Models (LLMs). We thoroughly explored the design space for in-context clustering, investigating how key factors – including set size, set diversity, set variation, and record order –influence clustering performance. Building on these insights, we introduced a robust end-to-end framework that incorporates an innovative record set creation strategy, a hierarchical clustering merge algorithm, and a tailored mis-clustering detection mechanism, referred to as the 'Guardrail'. Comprehensive evaluations on seven real-world datasets demonstrated that our approach not only achieves substantial improvements in result quality but also significantly optimizes resource efficiency, including token usage, running time, and API call volume. In the future, ideas from other LLM-based ER methods could complement our approach, such as batch clustering of multiple record sets simultaneously and advanced prompt engineering for in-context clustering.

REFERENCES

[1] 2024. API Pricing of Open-AI. Retrieved December 16, 2024 from https://openai.com/api/pricing/

[2] Faraz Ahmed and Muhammad Abulaish. 2012. An MCL-based approach for spam profile detection in online social networks. In *IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 602–608.

[3] Sihem Amer-Yahia, Angela Bonifati, Lei Chen, Guoliang Li, Kyuseok Shim, Jianliang Xu, and Xiaochun Yang. 2023. From large language models to databases and back: A discussion on research and education. *SIGMOD Rec.* 52, 3 (2023), 49–56.

[4] Anonymous Authors. 2025. Full version, source code and datasets. Retrieved January 14, 2025 from https://anonymous.4open.science/r/LLM4ER-CB0F/

[5] Omar Benjelloun, Hector Garcia-Molina, David Menestrina, Qi Su, Steven Euijong Whang, and Jennifer Widom. 2009. Swoosh: A generic approach to entity resolution. *VLDB J.* 18, 1 (2009), 255–276.

[6] Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 39–48.

[7] Christian Böhm and Hans-Peter Kriegel. 2001. A cost model and index architecture for the similarity join. In *IEEE International Conference on Data Engineering (ICDE)*, Dimitrios Georgakopoulos and Alexander Buchmann (Eds.). 411–420.

[8] A. Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of (SEQUENCES)*. 21–29.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *International Conference on Neural Information Processing Systems (NeurIPS)*. 1877–1901.

[10] Chengliang Chai, Guoliang Li, Jian Li, Dong Deng, and Jianhua Feng. 2016. Cost-effective crowdsourced entity resolution: A partial-order approach. In *International Conference on Management of Data (SIGMOD)*. 969–984.

[11] Zhaoqi Chen, Dmitri V Kalashnikov, and Sharad Mehrotra. 2007. Adaptive graphical approach to entity resolution. In *ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL)*. 204–213.

[12] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2024. Label-free node classification on graphs with large language models (llms). In *International Conference on Learning Representations, (ICLR)*.

[13] Peter Christen. 2012. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Trans. Knowl. Data Eng.* 24, 9 (2012), 1537–1555.

[14] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2020. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys (CSUR)* 53, 6 (2020), 1–42.

[15] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2021. An overview of end-to-end entity resolution for big data. *ACM Comput. Surv.* 53, 6 (2021), 127:1–127:42.

[16] Pierluigi Crescenzi, Viggo Kann, Magnus Halldorsson, Marek Karpinski, and Gerhard Woeginger. 2000. Maximum 3-dimensional matching. *A Compendium of NP Optimization Problems* (2000).

[17] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. 2021. Alaska: A flexible benchmark for data integration tasks. *arXiv Preprint arXiv:2101.11259* (2021).

[18] Mengyao Cui. 2020. Introduction to the k-means clustering algorithm based on the elbow method. *Accounting, Auditing and Finance* 1, 1 (2020), 5–8.

[19] Sanjib Das, Paul Suganthan G. C., AnHai Doan, Jeffrey F. Naughton, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, Vijay Raghavendra, and Youngchoon Park. 2017. Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services. In *ACM International Conference on Management of Data (SIGMOD)*. 1431–1446.

[20] Database Research Group of the Roma Tre University. 2020. SIGMOD 2020 programming contest official website. Retrieved September 22, 2024 from http://www.inf.uniroma3.it/db/sigmod2020contest

[21] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux. 2012. ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *World Wide Web Conference (WWW)*. 469–478.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)*, 4171–4186.

[23] 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.

[24] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16.

[25] Maxim Enis and Mark Hopkins. 2024. From LLM to NMT: Advancing Low-Resource Machine Translation with Claude. In *Proceedings of the 41st International Conference on Machine Learning*. 55204–55224.

[26] 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 *IEEE International Conference on Data Engineering (ICDE)*. 3696–3709.

[27] Sainyam Galhotra, Donatella Firmani, Barna Saha, and Divesh Srivastava. 2021. Efficient and effective ER with progressive blocking. *The VLDB Journal* 30, 4 (2021), 537–557.

[28] Papadakis George, Mandilaras George, Gagliardelli Luca, Simonini Giovanni, Thanos Emmanouil, Giannakopoulos George, Bergamaschi Sonia, Palpanas Themis, and Koubarakis Manolis. 2020. Three-dimensional entity resolution with JedAI. *Information Systems* 93 (2020), 101565.

[29] Lise Getoor and Ashwin Machanavajjhala. 2012. Entity resolution: theory, practice & open challenges. *Proc. VLDB Endow.* 5, 12 (2012), 2018—-2019.

[30] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *International Conference on Very Large Data Bases (VLDB)*, Vol. 99. 518–529.

[31] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off crowdsourcing for entity matching. In *ACM SIGMOD International Conference on Management of Data*. 601–612.

[32] Anja Gruenheid, Donald Kossmann, and Besmira Nushi. 2013. When is A=B? *Bull. EATCS* 111 (2013).

[33] Nuno M. Guerreiro, Duarte M. Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André F. T. Martins. 2023. Hallucinations in large multilingual translation models. *Transactions of the Association for Computational Linguistics* 11 (2023), 1500–1517.

[34] Oktie Hassanzadeh, Fei Chiang, Hyun Chul Lee, and Renée J Miller. 2009. Framework for evaluating clustering algorithms in duplicate detection. In *Proc. VLDB Endow.*, Vol. 2. 1282–1293.

[35] Boyi Hou, Qun Chen, Jiquan Shen, Xin Liu, Ping Zhong, Yanyan Wang, Zhaoqiang Chen, and Zhanhuai Li. 2019. Gradual machine learning for entity resolution. In *World Wide Web Conference (WWW)*. 3526–3530.

[36] Arvanitis Kasinikos Ioannis. 2024. Entity resolution with small-scale LLMs: A study on prompting strategies and hardware limitations.

[37] Shawn R. Jeffery, Michael J. Franklin, and Alon Y. Halevy. 2008. Pay-as-you-go user feedback for dataspace systems. In *ACM SIGMOD International Conference on Management of Data*. 847–860.

[38] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), 22199–22213.

[39] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *Proc. VLDB Endow.* 9, 12 (2016), 1197–1208.

[40] Michael Levandowsky and David Winter. 1971. Distance between sets. *Nature* 234, 5323 (1971), 34–35.

[41] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for BERT-based entity resolution. In *AAAI Conference on Artificial Intelligence*. 13226–13233.

[42] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for bert-based entity resolution. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 13226–13233.

[43] Huahang Li, Shuangyin Li, Fei Hao, Chen Jason Zhang, Yuanfeng Song, and Lei Chen. 2024. BoostER: Leveraging large language models for enhancing entity resolution. In *ACM Web Conference (WWW)*. 1043–1046.

[44] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060* (2024).

[45] 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.

[46] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Trans. Knowl. Data Eng.* 35, 1 (2021), 857–876.

[47] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR* abs/1907.11692 (2019).

[48] Alessio M., Faggioli G., Ferro N., Nardini F. M., and Perego R. 2024. Improving RAG systems via sentence clustering and reordering. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. CEUR-WS, 34–43.

[49] Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 169–178.

[50] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A comprehensive benchmark framework for active learning methods in entity matching. In *ACM SIGMOD International Conference on Management of Data*. 1133–1147.

[51] 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 *International Conference on Management of Data (SIGMOD)*. 19—-34.

[52] Navapat Nananukul, Khanin Sisaengsuwanchai, and Mayank Kejriwal. 2024. Cost-efficient prompt engineering for unsupervised entity resolution in the product matching domain. *Discov. Artif. Intell.* 4, 1 (2024), 56.

[53] 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.

[54] Felix Naumann and Melanie Herschel. 2010. *An introduction to duplicate detection.* Morgan & Claypool Publishers.

[55] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 629–638.

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

[57] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The four generations of entity resolution.* Morgan & Claypool Publishers.

[58] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–42.

[59] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Comput. Surv.* 53, 2, Article 31 (2020), 42 pages.

[60] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. 2016. Comparative analysis of approximate blocking techniques for entity resolution. In *Proc. VLDB Endow.*, Vol. 9. 684–695.

[61] Ralph Peeters and Christian Bizer. 2021. Dual-objective fine-tuning of BERT for entity matching. *Proc. VLDB Endow.* 14, 10 (2021), 1913–1921.

[62] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for entity matching. In *New Trends in Database and Information Systems (ADBIS)*. 221–230.

[63] Ralph Peeters, Aaron Steiner, and Christian Bizer. 2025. Entity matching using large language models. In *International Conference on Extending Database Technology (EDBT)*. 529–541.

[64] Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active learning for large-scale entity resolution. In *ACM on Conference on Information and Knowledge Management (CIKM)*. 1379–1388.

[65] Leigang Qu, Shengqiong Wu, Hao Fei, Liqiang Nie, and Tat-Seng Chua. 2023. Layoutllm-t2i: Eliciting layout guidance from llm for text-to-image generation. In *Proceedings of the 31st ACM International Conference on Multimedia*. 643–654.

[66] Mehdi Akbarian Rastaghi, Ehsan Kamalloo, and Davood Rafiei. 2022. Probing the robustness of pre-trained language models for entity matching. In *ACM International Conference on Information & Knowledge Management (CIKM)*. 3786–3790.

[67] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.

[68] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2017. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *Advances in Databases and Information Systems (ADBIS)*. 278–293.

[69] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Workshop on Energy Efficient Machine Learning and Cognitive Computing@NeurIPS*.

[70] Xiaofei Sun, Xiaoya Li, Jiwei Li, Fei Wu, Shangwei Guo, Tianwei Zhang, and Guoyin Wang. 2023. Text classification via large language models. In *Findings of the Association for Computational Linguistics (EMNLP)*.

[71] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: A design space exploration. *Proc. VLDB Endow.* 14, 11 (2021), 2459–2472.

[72] Sindhu Tipirneni, Ravinarayana Adkathimar, Nurendra Choudhary, Gaurush Hiranandani, Rana Ali Amjad, Vassilis N. Ioannidis, Changhe Yuan, and Chandan K. Reddy. 2024. Context-aware clustering using large language models. *CoRR* abs/2405.00988 (2024).

[73] Vasilis Verroios and Hector Garcia-Molina. 2015. Entity resolution with crowd errors. In *IEEE International Conference on Data Engineering (ICDE)*. 219–230.

[74] Norases Vesdapunt, Kedar Bellare, and Nilesh N. Dalvi. 2014. Crowdsourcing algorithms for entity resolution. *Proc. VLDB Endow.* 7, 12 (2014), 1071–1082.

[75] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. 2024. Large language models enable few-shot clustering. *Trans. Assoc. Comput. Linguistics* 12 (2024), 321–333.

[76] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. Crowder: Crowdsourcing entity resolution. In *Proc. VLDB Endow.*, Vol. 5. 1483—-1494.

[77] Jiannan Wang, Guoliang Li, Tim Kraska, Michael J. Franklin, and Jianhua Feng. 2013. Leveraging transitive relations for crowdsourced joins. In *ACM SIGMOD International Conference on Management of Data*. 229–240.

[78] Sibo Wang, Xiaokui Xiao, and Chun-Hee Lee. 2015. Crowd-based deduplication: An adaptive approach. In *ACM SIGMOD International Conference on Management of Data*. 1263–1277.

[79] Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xuanang Chen, Xianpei Han, Hao Wang, Zhenyu Zeng, and Le Sun. 2024. Match, Compare, or Select? An Investigation of Large Language Models for Entity Matching. *arXiv preprint arXiv:2405.16884* (2024).

[80] Tianshu Wang, Hongyu Lin, Xiaoyang Chen, Xianpei Han, Hao Wang, Zhenyu Zeng, and Le Sun. 2025. Match, compare, or select? An investigation of large language models for entity matching. *International Conference on Computational Linguistics (COLING)*.

[81] Yu Wang, Xinjie Yao, Pengfei Zhu, Weihao Li, Meng Cao, and Qinghua Hu. 2024. Integrated heterogeneous graph attention network for incomplete multi-modal clustering. *Int. J. Comput. Vis.* 132, 9 (2024), 3847–3866.

[82] Steven Euijong Whang, Peter Lofgren, and Hector Garcia-Molina. 2013. Question selection for crowd entity resolution. *Proc. VLDB Endow.* 6, 6 (2013), 349–360.

[83] Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. 2019. Deep comprehensive correlation mining for image clustering. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 8150–8159.

[84] Chuan Xiao, Wei Wang, Xuemin Lin, Jeffrey Xu Yu, and Guoren Wang. 2011. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems (TODS)* 36, 3 (2011), 1–41.

[85] Vijaya Krishna Yalavarthi, Xiangyu Ke, and Arijit Khan. 2017. Select your questions wisely: For entity resolution with crowd errors. In *ACM Conference on Information and Knowledge Management (CIKM)*. 317–326.

[86] Ka Yee Yeung and Walter L Ruzzo. 2001. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics* 17, 9 (2001), 763–774.

[87] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. Large language models as data preprocessors. *Workshops at the International Conference on Very Large Data Bases.*

[88] Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. ClusterLLM: Large language models as a guide for text clustering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 13903–13920.

[89] Zeyu Zhang, Paul Groth, Iacer Calixto, and Sebastian Schelter. 2024. Directions towards efficient and automated data wrangling with large language models. In *IEEE International Conference on Data Engineering (ICDE)*. 301–304.

8 APPENDIX: FULL EXPERIMENTAL STUDY

This section presents the complete experimental results omitted from Section 6 due to space constraints. Specifically, we provide the individual record set in-context clustering performance across four metrics under different conditions of six datasets (A.1), the impact of using different LLMs on end-to-end ER performance (A.2), and the effect of LLM guardrails on misclustering detection and record set regeneration (A.3).

A.1 Full Experiment About Key Factors on Individual Record Sets

Evaluation Metrics. We select several widely-used metrics to evaluate the in-context clustering results, as well as the final ER clustering results. These metrics are FP-measure (a variant of the F-measure), Accuracy (ACC), Normalized Mutual Information (NMI), and Adjusted Rand Score (ARI) [2, 11, 81, 86]. All experiments are repeated three times to ensure robustness, and we report the average values over three runs. Due to space limitations, we present results for ACC and FP-measure in the main paper. The complete experimental results are available in the full version [4].

The definition of ACC is as follows: Let $\mathbb{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_m\}$ represent the ground truth clusters and $\mathbb{X} = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ denote the predicted clusters. The total number of records in both sets is $|\mathcal{R}| = \bigcup_i |\mathcal{X}_i| = \bigcup_i |\mathcal{Y}_i|$. The accuracy (ACC) is defined as:

$$\text{ACC} = \frac{\text{CorrectCount}}{|\mathcal{R}|} \tag{8}$$

$$\text{CorrectCount} = \sum_{\mathcal{X}_j \in \mathbb{X}} \sum_{x \in \mathcal{X}_j} \mathbb{I}(\exists y \in \mathcal{Y}_j^{'}, y = x) \tag{9}$$

where $\mathbb{Y}^{'} = \{\mathcal{Y}_1^{'}, \ldots, \mathcal{Y}_m^{'}\}$ is obtained by reordering $\mathbb{Y}$ based on their intersection sizes with the predicted clusters, and $\mathbb{I}(\cdot)$ is the indicator function that equals to 1 if the condition satisfies.

Similarly as ACC, NMI also measures the similarity between clustering results and ground truth, but from the perspective of clustering entropy.

$$NMI(R,T) = \frac{2I(R,T)}{H(R) + H(T)} \tag{10}$$

where function $I(\cdot)$ measures the degree of information sharing between two clusters and function $H(\cdot)$ is used to measure the uncertainty of elements in clusters.

$$I(R,T) = \sum_{i=1}^{|R|} \sum_{i=1}^{|T|} p(i,j) \log(\frac{p(i,j)}{p(i)p(j)}) \tag{11}$$

$$H(R) = -\sum_{i=1}^{|R|} p(i) \log p(i) \tag{12}$$

Different from above metrics, ARI considered both the similarity and dissimilarity of all pairs of data records. The definition of ARI in our work is the same as [83]. Given a set S of n elements and two clustering sets of these elements consists of r and s groups that represented as $X = \{X_1, X_2 \ldots, X_r\}$ (the predicted results) and $Y = \{Y_1, Y_2 \ldots, Y_s\}$ (the ground truth) in a contingency table $[t_{ij}]$ of overlaps between X and Y. Each element in $[t_{ij}]$ shows the number of overlapping objects between X and Y.

$$ARI = \frac{\sum_{i,j}\binom{t_{i,j}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_i}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_i}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_i}{2}]/\binom{n}{2}} \tag{13}$$

The last metric, FP-measure, evaluates clustering results from the perspective of homogeneity and stability. It is the harmonic mean of purity and inverse-purity.

$$purity\,(\mathbb{X},\mathbb{Y}) = \sum_{\mathcal{X}_i \in \mathbb{X}} \frac{|\mathcal{X}_i|}{|\mathcal{R}|} \max_{\mathcal{Y}_i \in \mathbb{Y}} Overlap(\mathcal{X}_i, \mathcal{Y}_i) \tag{14}$$

$$inverse\text{-}purity\,(\mathbb{X},\mathbb{Y}) = \sum_{\mathcal{Y}_i \in \mathbb{Y}} \frac{|\mathcal{Y}_i|}{|\mathcal{R}|} \max_{\mathcal{X}_i \in \mathbb{X}} Overlap(\mathcal{Y}_i, \mathcal{X}_i) \tag{15}$$

$$Overlap(\mathcal{X}_i, \mathcal{Y}_i) := \frac{|\mathcal{X}_i \cap \mathcal{Y}_i|}{|\mathcal{X}_i|} \tag{16}$$

Finally, we derive the FP-measure:

$$FP\text{-}measure(\mathbb{X}, \mathbb{Y}) = \frac{2}{1/purity(\mathbb{X}, \mathbb{Y}) + 1/inverse\text{-}purity(\mathbb{X}, \mathbb{Y})} \tag{17}$$

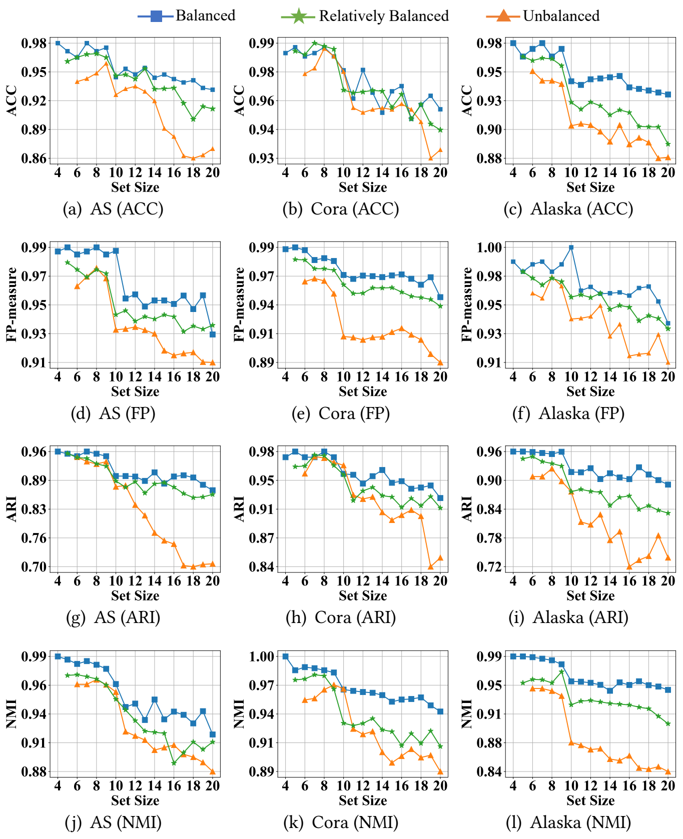


Figure 8: Clustering performance vs. set size and variation

Impact on Individual Record Sets. Figure 8 illustrates the in-context clustering performance of the LLM on individual record sets under varying S_s and S_v. It can be viewed that LLM's performance *improves with smaller set variations*, indicating that the more evenly distributed the proportions of records are among different entities in a record set, the better the performance across all datasets and metrics, regardless of the set size. We also observe that the LLM performance *remains relatively stable and well* for the balanced, relatively balanced, and unbalanced settings as the set size increases from 4 to 9. However, once the *set size exceeds 9 or 10*, the in-context clustering performance of the LLM record sets *declines sharply*. This can be attributed to the fact that as the set size grows, the LLM must distribute its focus across more records, leading to a dilution of semantic understanding and a higher risk of errors. Furthermore, larger record sets exacerbate the difficulty in maintaining consistent contextual relationships, which is crucial for accurate clustering.

Notably, for the same set variation setting, the trends across ACC and FP-measure are *generally consistent* since these metrics all evaluate, to some degree, the similarity between the clustering results and the ground truth. Therefore, it can be concluded that *the optimal set size is 9*, and the variation within the record set should be *as slight as possible* for better in-context clustering performance.

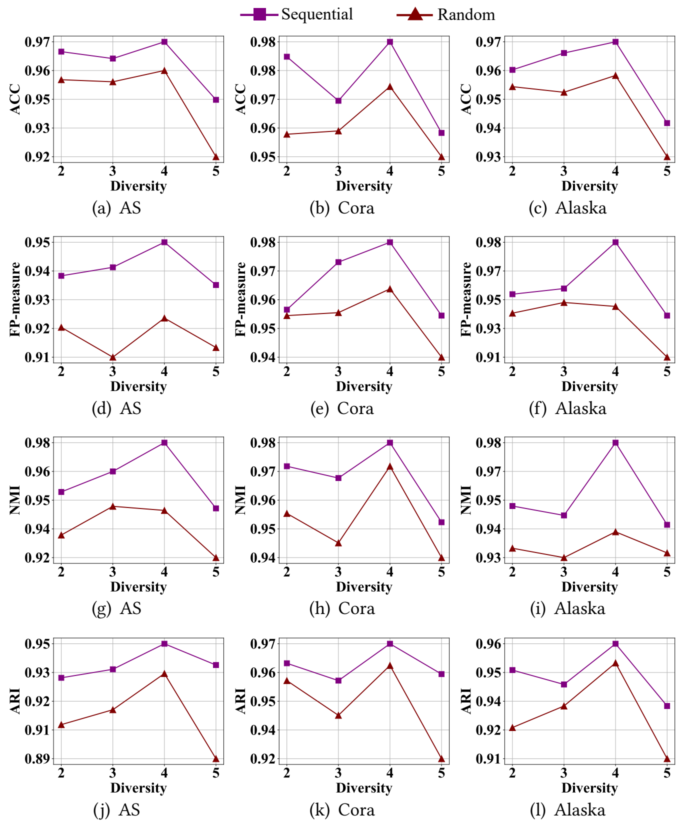


Figure 9: Clustering performance vs. set diversity, ordering

Fixing the optimal values of S_s and S_v based on previous experiments, we further investigate the impact of S_d and the ordering of records within record sets on the performance of LLM-based in-context clustering (Figure 9). We notice that when S_d is set to 4, the in-context clustering performance of the LLM is better across all datasets in terms of ACC and FP-measure compared to other diversity settings. Thus, we consider $S_d = 4$ *as our optimal diversity setting*. With 4 entities, the clustering task provides enough granularity to differentiate entities while avoiding excessive sparsity or overlap in record distribution. In contrast, fewer entities (e.g., 2 or 3) might result in overly homogeneous clusters that fail to capture subtle differences, while a larger number of entities (e.g., 5) might introduce higher complexity and noise, negatively impacting the LLM's ability to identify distinct clusters accurately. Meanwhile, it is observed that the LLM achieves better in-context clustering performance when the *sequential record order* is used. This improvement can be attributed to the sequential arrangement of similar records, which likely enhances the coherence of contextual information presented to the LLM. By maintaining entity-related records in close proximity, this ordering reduces potential context-switching overhead and provides a more structured input sequence [48], thereby enabling the model to better discern subtle patterns and relationships essential for accurate clustering.

In conclusion, for the LLM's in-context clustering performance, our optimal configuration includes a set size of 9, a balanced distribution (S_v close to 0), a set diversity of 4, and the sequential record order. Larger set sizes should be avoided, as the LLM's performance under extremely unbalanced ratios deteriorates rapidly beyond this point. Conversely, smaller set sizes should also be avoided, as maximizing the set size is crucial for reducing API call frequency, improving efficiency and minimizing token usage and cost.

A.2 Impact of Different LLMs on End-to-end ER Results

Table 6: Impact of LLMs on End-to-end ER performance

Metrics	*Cora*		*Alaska*		*AS*	
	gpt 3.5-turbo	*gpt 4o-mini*	*gpt 3.5-turbo*	*gpt 4o-mini*	*gpt 3.5-turbo*	*gpt 4o-mini*
ACC	0.87	**0.90**	0.80	**0.82**	0.69	**0.70**
FP-measure	0.68	**0.71**	0.77	**0.79**	**0.64**	0.63
Cost (USD)	0.21	**0.03**	1.19	**0.15**	0.19	**0.02**
Tokens (M)	0.13	**0.12**	**0.72**	0.73	0.08	**0.07**
Time (min)	5.44	**5.42**	**39.53**	39.57	8.03	**8.01**
# API Calls (K)	0.29	**0.28**	**2.02**	2.04	0.42	**0.41**

The results in Table 6 provide a comprehensive comparison of the performance of different GPT models—GPT-3.5-turbo (using the gpt-3.5-turbo-0613 model) and GPT-4o-mini—on end-to-end entity resolution (ER) tasks across three diverse datasets: Cora, Alaska, and AS. GPT-3.5-turbo incurs a cost of \$1.50 per 1M input tokens and \$2.00 per 1M output tokens, while GPT-4o-mini operates at a significantly lower cost of \$0.150 per 1M input tokens and \$0.600 per 1M output tokens. In terms of ACC, GPT-4o-mini consistently demonstrates superior performance, with improvements of up to 3% over GPT-3.5-turbo on both the Cora and Alaska datasets. This suggests that the newer GPT-4o-mini model is more effective at distinguishing between similar entities, thereby enhancing the overall resolution accuracy.

Despite GPT-3.5-turbo achieving slightly higher FP-measure scores in some cases, such as on the Alaska dataset, the efficiency benefits of GPT-4o-mini are considerable. It significantly reduces both operational cost (USD) and resource consumption, as reflected in the decreased number of tokens processed and API calls made. Notably, the reduction in API calls is particularly pronounced, with GPT-4o-mini requiring up to 5 times fewer calls compared to GPT-3.5-turbo on the Cora and Alaska datasets. This substantial reduction in resource usage does not come at the expense of accuracy, reinforcing the advantage of using GPT-4o-mini for large-scale ER tasks where both performance and cost-efficiency are critical. These findings highlight the importance of selecting an appropriate LLM variant based on the specific needs of an entity resolution task, with a trade-off between accuracy and computational efficiency. In practice, the choice of model can significantly impact both the quality of the results and the operational costs, especially when deploying ER solutions at scale.

A.3 Effect of LLM Guardrails on Misclustering Detection and Record Set Regeneration

The results in Table 7 demonstrate the significant impact of incorporating the MDG algorithm on end-to-end entity resolution (ER) performance across *Cora*, *Alaska*, and *AS*. When MDG is applied, all key performance metrics show substantial improvement. For example, ACC increases by up to 30% on the *Cora* dataset, from 0.60 without MDG to 0.90 with MDG. Similarly, FP-measure scores improve across all datasets, highlighting MDG's effectiveness in detecting misclustering and enhancing entity resolution quality. The substantial improvement in performance can be attributed to MDG's ability to prevent the incorrect merging of clusters that should not be grouped together, ensuring more accurate partitioning. While a single misclassification may have a minimal impact on binary classification metrics like Precision, it can significantly affect clustering metrics such as FP-measure and ACC. By effectively identifying and addressing misclustered records, our algorithm achieves more precise cluster delineation, leading to higher-quality classification outcomes. This ability to enhance cluster accuracy through misclustering detection makes MDG a powerful tool for improving the overall effectiveness of entity resolution tasks.

Moreover, the MDG algorithm does not result in significant additional costs or resource consumption. The total cost in USD remains nearly identical between the scenarios with and without MDG, and the number of tokens, API calls, and processing time are only marginally affected. These results suggest that the MDG algorithm provides a substantial improvement in ER performance without a notable increase in computational overhead, making it an efficient approach for enhancing the quality of entity resolution in real-world applications.

Table 7: Effect of MDG Algorithm and Record Set Regeneration on End-to-end Performance

Metrics	*Cora*		*Alaska*		*AS*	
	w/o MDG	*w/ MDG*	*w/o MDG*	*w/ MDG*	*w/o MDG*	*w/ MDG*
ACC	0.60	**0.90**	0.35	**0.82**	0.52	**0.70**
FP-measure	0.58	**0.71**	0.47	**0.79**	0.52	**0.63**
Cost (USD)	**0.03**	**0.03**	**0.14**	0.15	**0.02**	**0.02**
Tokens (M)	**0.11**	0.12	**0.66**	0.73	**0.06**	0.07
Time (min)	**5.04**	5.42	**36.88**	39.57	**7.35**	8.01
# API Calls (K)	**0.26**	0.28	**1.85**	2.04	**0.37**	0.41