Evaluating Blocking Biases in Entity Matching

Mohmmad Hossein Moslemi
The University of Western Ontario
London, Ontario, Canada
mohammad.moslemi@uwo.ca

Harini Balamurugan
The University of Western Ontario
London, Ontario, Canada
hbalamur@uwo.ca

Mostafa Milani
The University of Western Ontario
London, Ontario, Canada
mostafa.milani@uwo.ca

***Abstract*—Entity Matching (EM) is crucial for identifying equivalent data entities across different sources, a task that becomes increasingly challenging with the growth and heterogeneity of data. Blocking techniques, which reduce the computational complexity of EM, play a vital role in making this process scalable. Despite advancements in blocking methods, the issue of fairness—where blocking may inadvertently favor certain demographic groups—has been largely overlooked. This study extends traditional blocking metrics to incorporate fairness, providing a framework for assessing bias in blocking techniques. Through experimental analysis, we evaluate the effectiveness and fairness of various blocking methods, offering insights into their potential biases. Our findings highlight the importance of considering fairness in EM, particularly in the blocking phase, to ensure equitable outcomes in data integration tasks.**

I. Introduction

Entity Matching (EM) is the process of determining whether two or more data entities from the same or different sources refer to the same real-world object. As data sources expand and become more heterogeneous, the challenge of accurately and efficiently matching entities has intensified. EM, also known as entity linkage or record matching, is fundamental in data integration, with broad applications across industries [1], [2], [3], [4], [5], [6], [7]. In commercial sectors, EM is crucial for tasks like matching customer or product records across databases, while in healthcare and security, accurate EM can have significant implications [8]. For instance, matching patient records across hospitals is vital to ensure comprehensive care and avoid duplication despite variations in data entry.

EM systems typically consist of a matching component that compares entities and labels them as either "match" or "non-match." A major challenge in EM is its computational complexity, often scaling quadratically as each entity must be compared to all others, making it an $O(n^2)$ problem. This complexity becomes prohibitive for large datasets with millions of records. To mitigate this, blocking methods are used as a preliminary step to reduce the number of comparisons [9], [10], [11]. As a result, EM systems typically operate in two phases: blocking to limit comparisons, followed by matching to produce the final labels.

Blocking reduces the number of comparisons by grouping similar entities into distinct or overlapping blocks, ensuring that comparisons are only made within these smaller, more manageable groups. This step is critical as it significantly reduces the computational load, making the EM process more scalable. By partitioning the dataset into blocks where records are more likely to match, blocking enhances computational efficiency and addresses the inherent quadratic complexity of EM.

Blocking methods have evolved significantly, spanning from heuristic-based approaches to advanced deep-learning techniques. Traditional methods such as Standard Blocking and Sorted Neighborhoods [12], [13] laid the groundwork, with Standard Blocking categorizing records based on a blocking key, like initials. At the same time, Sorted Neighborhood employs a sliding window over sorted records to enhance efficiency. Advanced techniques like Canopy Clustering [14] and recent machine learning-driven methods such as BSL (Blocking Scheme Learner) and BGP (Blocking based on Genetic Programming) [15], [16] further refine blocking schemes by optimizing attribute selection and comparison strategies. Additionally, frameworks like AutoBlock and DeepER [17], [18] utilize deep learning for representation learning and efficient candidate generation. The effectiveness of these methods is typically assessed using metrics like Reduction Ratio (RR), Pair Quality (PQ), Pair Completeness (PC), and their harmonic mean [19], [20], [21].

In recent years, fairness in ML has gained significant attention [22], [23], [24] due to its critical impact on real-life applications. Fairness is particularly important in the context of EM because both EM and blocking systems can produce biased results, often exhibiting higher accuracy for one demographic group over another. Despite the significant implications of EM on real-life decisions, research on the fairness of EM remains limited, with only a few studies exploring this issue [25], [26], [27]. Even fewer studies have addressed the fairness of blocking methods, leaving this area largely unexplored. To the best of our knowledge, only one study has investigated the fairness of blocking [28], which merely touched on the topic by defining a fairness metric for blocking based on the representation ratio, similar to RR, and proposing simple algorithms to address bias in blocking.

Traditional fairness metrics, such as Equalized Odds, Equal Opportunity, and Statistical Parity [23], [24], are typically defined for ML models and are based on accuracy metrics. These metrics cannot be directly applied to blocking methods, as blocking is a pre-processing step and does not produce accuracy metrics like ML models. One of the contributions of our study is to extend the existing blocking metrics to incorporate fairness and to evaluate how effectively these

metrics can measure bias in different blocking techniques. For simplicity, this work focuses solely on binary-sensitive attributes.

This paper presents an experimental study of blocking methods, focusing on fairness issues, detecting biases, and examining their impact on EM and end-to-end matching tasks. In the remainder of this paper, Section II presents related work, including a summary of the state-of-the-art blocking methods. Section III provides a formal definition of EM and blocking and reviews the metrics used to evaluate blocking. In Section IV, we extend these blocking metrics to fairness measures, proposing them as a sufficient method for assessing bias in blocking. Section V presents our experimental results, highlighting the effectiveness of various blocking methods and the biases they may introduce. Section VI offers our conclusions. All the implementations are available at https://github.com/mhmoslemi2338/pre-EM-bias.

II. Related Work

We briefly review the existing blocking methods for EM and then discuss fairness in EM and related areas such as clustering and ranking.

A. Blocking Methods

Over the years, a wide range of blocking techniques has been developed, from simple heuristic-based methods to advanced approaches involving deep neural networks (for surveys, see [29], [21], [30]). These blocking methods can be categorized in various ways, each providing a different perspective on how these techniques function and their applications. One way is by distinguishing between learning-based and non-learning-based algorithms. Rule-based methods, a type of non-learning-based approach, rely on expert knowledge or simple heuristics to define the blocking criteria. In contrast, learning-based methods require training data to learn how to block the data using machine learning techniques.

Another categorization is based on schema awareness. Schema-aware methods perform blocking by focusing on the most important attributes of the data, while schema-agnostic methods treat the entire entity as a single attribute, utilizing all available information. A third categorization concerns redundancy awareness, dividing methods into redundancy-free, redundancy-positive, and redundancy-neutral subcategories, which differ in how they handle the assignment of entities to blocks and the overlap between blocks. Mosatafa: It is not clear what redundancy- categories are. Please add a few sentences to clarify Mohammad-Hossein: Redundancy-free methods create disjoint blocks by assigning each entity to only one block, leading to higher similarity between two entities if they share a greater number of identical blocks. In contrast, redundancy-positive methods form overlapping blocks, with each entity placed in multiple blocks; here, the likelihood of a match between two entities is proportional to the total number of blocks they share. Redundancy-neutral methods also produce overlapping blocks, but in such a way that most entity pairs share nearly the same number of blocks, rendering redundancy insignificant in determining the similarity between entities. These various categorizations highlight the diversity of blocking techniques available, each suited to different scenarios depending on the data and the desired balance between computational efficiency and match accuracy [30].

Traditional blocking methods such as Standard Blocking and Sorted Neighborhoods have fundamentally shaped the field [12], [13]. Standard Blocking categorizes records according to a blocking key, such as a phone number or surname initials, to conduct intensive comparisons within these blocks. However, this method risks inefficiencies when block sizes are large. In contrast, Sorted Neighborhood enhances efficiency by sorting records according to a key and employing a sliding window for comparisons, though it may overlook matches when key values exceed the window's boundaries.

Advanced approaches like Canopy Clustering [14] have been introduced to address some of the limitations of traditional methods. This technique uses a less costly, coarse similarity measure to initially group records, which are then subjected to more precise and computationally demanding comparisons within each canopy. Although designed to minimize the total number of comparisons, this method may occasionally result in the erroneous grouping of distinct entities [16].

Blocking techniques, such as BSL (Blocking Scheme Learner) [15] and BGP (Blocking based on Genetic Programming) [16], have further enhanced the efficiency and accuracy of the traditional methods. These more advanced methods leverage machine learning to refine blocking schemes, focusing on attribute selection and comparison methods to generate candidate matches efficiently. CBLOCK offers an automated approach to canopy formation within a map-reduce framework, tailored specifically for large-scale de-duplication tasks involving diverse datasets, thus optimizing the trade-off between recall and computational efficiency [31]. Additionally, Token-Based Blocking effectively addresses challenges in heterogeneous datasets by comparing records based on shared tokens, providing a versatile solution for integrating diverse data sources.

Deep learning has recently revolutionized the blocking phase in EM, shifting from traditional heuristic methods to more adaptive and automated approaches. Frameworks like AutoBlock and DeepER exploit deep learning for representation learning and nearest neighbor search, demonstrating significant effectiveness across varied, large-scale datasets [17], [18]. DeepBlock, which merges syntactic and semantic similarities through deep learning, further enhances blocking quality by accurately grouping similar records, even in noisy or heterogeneous datasets [32].

Various metrics have been used to measure the quality of blocking. Still, three are most commonly used and considered the most comprehensive in assessing the effectiveness of blocking methods: RR, PQ, and PC [19], [20], [21]. RR measures the extent to which a blocking system reduces the total number of comparisons, PQ denotes the percentage of candidate pairs that are true matches after blocking, and PC

indicates the percentage of true matches in the candidate set after blocking.

There is little agreement on the evaluation measures in the literature on blocking. Some studies only consider PC and RR as their primary metrics [17], [33], [34], [35], [36], [37], [9], [10], [38], while some only considered PC and PQ [32], [39]. Several studies also consider runtime as a main evaluation measure in blocking methods [40], [41], [42], [43]. There is also research that employs the harmonic mean of PC and RR [44], [45], [46], [47], [48] or the harmonic mean of PC and PQ [21]. We will formally define these metrics in Section III and discuss what measure provides a more meaningful way to evaluate the methods considered in Section V.

B. Fairness in EM and Blocking

Fairness is a growing concern in EM systems, especially as these systems are widely employed in data-driven decision-making processes. A key concern is that EM processes can unintentionally perpetuate biases present in the data, leading to unfair outcomes. This issue is particularly significant during the blocking stage, where the method of grouping records for comparison can introduce bias, affecting both the accuracy and fairness of the matching process. Mosatafa: I suggest giving a few real-world examples of fairness issues in EM, e.g., nofly list, etc that we had before. Mohammad-Hossein: A real-world example that underscores these fairness issues is the use of "no-fly" lists, where individuals, particularly people of color, have been wrongly flagged as security threats due to misidentification, resulting in denial of services. In such cases, the blocking and matching algorithms often fail to consider cultural and linguistic diversity, leading to higher rates of false positives among minority groups [49]. Another notable example is Google's advertising algorithms, which have been found to display prestigious job advertisements more frequently to men than to women. This bias occurs because the algorithms tend to match male profiles more closely with the job requirements [50]. These discrepancies highlight how EM algorithms can perpetuate existing gender biases, resulting in unequal opportunities and outcomes. These examples illustrate the necessity for developing blocking and matching algorithms that ensure fair and equitable treatment across all demographic groups.

Recent research has sought to address these fairness challenges. The FairER algorithm, introduced by [51], incorporates fairness constraints directly into the EM process, highlighting the need to consider fairness from the very beginning of the EM pipeline. This approach is crucial in the blocking stage, where biases in record grouping can skew the entire matching process. Additionally, [27] proposed an AUC-based fairness metric to evaluate how well EM systems perform across different groups. Such metrics could be critical in refining blocking methods to prevent bias from affecting EM outcomes. Recent work in [28], [52] extensively studies biases in EM as a data preparation task to avoid bias and ensure that no groups are disproportionately excluded or misrepresented. Still, they mainly focus on EM and barely discuss blocking.

Blocking can be seen as clustering. We, therefore, briefly review fairness concepts in clustering as a related task. Fairness in clustering is assessed using metrics that ensure equitable treatment across demographic groups. In clustering, fairness is evaluated by the balance within clusters, using metrics like demographic parity and balance ratio [53] to measure proportional representation. Techniques such as *Fairlet Decomposition* [54] utilize these metrics to create balanced groups across sensitive attributes before final clustering, ensuring fair representation. While having balanced clusters is a reasonable fairness requirement for clustering, it does not apply to blocking, where the goal is not equal representation within blocks but rather to create small blocks that include all equivalent record pairs.

III. Background

We start from a relation schema $\mathcal{S}$ consisting of a set of attributes $A_1, ..., A_m$ with domains $\text{DOM}(A_i), i \in [1, m]$. An entity (record) t with schema $\mathcal{S}$ is a member of $\text{DOM}(A_1) \times ... \times \text{DOM}(A_m)$, the set of all possible entities, which we denote by $\mathcal{D}$. We use $t[A_i]$ to refer to the value of attribute A_i in entity t. A relation $D \subseteq \mathcal{D}$ is a set of entities.

EM generally has two major phases: blocking and matching [1]. We start with matching and then explain blocking.

A. Entity Matching

Given two relations D_1, D_2, the problem of *entity matching (EM)* is to find a subset M of $D_1 \times D_2$ that consists of entity pairs referring to the same real-world entities. We refer to such entity pairs as *equivalent pairs*.

A entity matcher (matcher in short) for entities of schema $\mathcal{S}$ is a binary classifier $f : \mathcal{D} \times \mathcal{D} \mapsto \{0, 1\}$; that labels entity pairs 1, called they match, or 0, saying they don't match. Given relations D_1, D_2 with schema $\mathcal{S}$ and equivalent entity pairs $M \subseteq D_1 \times D_2$, the goal of f is to find the equivalent entities in M, i.e., label the entity pairs in M as a match and the non-equivalent entities as non-match, where the accuracy of f, e.g., true/false positive/negative rates and F1 score, are defined based on whether the equivalent entity pairs are correctly labeled.

B. Blocking

The set of entity pairs for relations D_1, D_2 in a matching setting, which we denote by $P = D_1 \times D_2$, grows quadratically in size w.r.t the size of D_1 and D_2. This makes it costly to run expensive matching methods for all possible pairs. The problem of blocking is to find a candidate set $C \subseteq P$, which is much smaller than P, while it still includes all equivalent pairs; $M \subseteq C$. A blocking method is obviously expected to run much faster than a matching method for comparing all possible pairs. Blocking saves unnecessary checking of some of the non-equivalent pairs while searching for the equivalent pairs.

Definition III.1 (Blocking)**.** *Given two datasets, D_1 and D_2, the set of all possible pairs is denoted by $P = D_1 \times D_2$, an. The goal of blocking is to generate a candidate set C s.t.*

$|C| \ll |P|$ *(there are much fewer candidates compared to total entity Paris) and* $M \subseteq C$ *in a much less time compared to matching methods.*

Blocking Metrics: How effective a blocking method is measured using three main quality measures defined as follows:

$$\text{RR} = 1 - \frac{|C|}{|P|} \quad \text{PC} = \frac{|C \cap M|}{|M|} \quad \text{PQ} = \frac{|C \cap M|}{|C|} \tag{1}$$

Reduction ratio (RR) is the ratio of the reduction in the number of comparisons after blocking to the total number of possible comparisons. A higher RR value signifies a greater reduction in the number of candidate entity pairs. This measure does not consider the quality of the generated candidate entity pairs. *Pairs completeness* PC represents the ratio of equivalent pairs retained after blocking to the total number of equivalent pairs. This measure evaluates how effectively C preserves equivalences, corresponding to recall in information retrieval [55]. PC ranges from 0 to 1, where a value of 1 indicates that the blocker retains all true matches. *Pairs quality* (PQ) denotes the fraction of equivalent pairs produced by C relative to the total number of pairs. A higher PQ indicates that C is efficient, primarily generating true matches. In contrast, a lower PQ suggests that many non-matching pairs are included. PQ is equivalent to precision in information retrieval [30]. PQ ranges from 0 to 1, with higher values indicating that the blocker is more effective at eliminating non-matches.

In addition to evaluating the effectiveness of blocking methods, their efficiency is equally important. Specifically, the runtime of a blocking method plays a crucial role in determining the optimal approach. If runtime is overlooked, one could end up using a matching technique that, while achieving high recall rate (RR), precision (PC), and pair quality (PQ), might take days to execute.

There is no clear consensus on which blocking measures are most important or how they relate to each other. Some studies [29], [42], [35], [47] suggest that PQ is typically sensitive to small changes in RR. We observe in experiments that blocking is generally applied to datasets with millions of pairs with a much smaller number of true matches. Therefore, a small change in RR leads to a significant increase in the size of the candidate set that greatly impacts PQ, which explains the high sensitivity of PQ. Therefore, in this study, we focus on PC, RR, their harmonic mean, and runtime as the key metrics for evaluating blocking methods. The harmonic mean, which balances the trade-off between PC and RR, is defined as:

$$F_{\text{PC,RR}} = \frac{2 \times \text{PC} \times \text{RR}}{\text{PC} + \text{RR}} \tag{2}$$

A detailed numerical analysis of these measures and their relationships is provided in Section V. We explain these measures using an example.

Example III.2. *Figure 1 shows a relation* D *with ten entities. We consider matching entities in the same relation entity pairs* $P = D \times D$*. The red dotted lines show the equivalent pairs;* $M = \{(t_2, t_6),\ (t_6, t_7),\ (t_2, t_7),\ (t_4, t_5),\ (t_8, t_9),\ (t_8, t_{10}),\ (t_9, t_{10})\}$*. The solid black lines show blocking methods that specify three blocks that result in* $3 + 6 + 3 = 12$ *candidate pairs. The blocking result misses two equivalent pairs,* (t_2, t_6) *and* (t_2, t_7)*, but reduces the number of pairs to check from 45 to 12, resulting in RR* $= 1 - \frac{12}{45} \approx 0.73$*, PC* $= \frac{5}{7} \approx 0.71$*, and* $F_{RR,PC} \approx 0.72$*.*

IV. Measuring Bias in Blocking

Blocking methods can suffer from disparities; the quality of blocking might differ for the minority group compared with the majority, leading either to missing equivalent minority pairs or being ineffective in reducing unnecessary matching of non-equivalent methods, which can happen if the size of the candidate set is not much smaller than all the minority pairs.

To define biases in blocking, we assume the relation schema $\mathcal{S}$ includes a sensitive (or protected) attribute A (e.g., gender) with domains $\text{DOM}(A) = a, b$, where a and b correspond to the minority (e.g., female or nonbinary) and majority groups (e.g., male), respectively. This means that an entity t with $t[A] = a$ belongs to the minority group, while $t[A] = b$ indicates it belongs to the majority group. In the context of matching, we use the protected attribute of entities in a record pair to determine whether the pair concerns minority or majority groups. We define a minority pair as any pair of entities $(t_1, t_2) \in D_1 \times D_2$ where at least one entity belongs to a minority group based on the specified sensitive attribute (e.g., ethnicity, gender). Conversely, a majority pair consists of both entities from the majority group. A minority pair is thus defined as any pair that could lead to an entity matching (EM) decision that negatively impacts a minority entity, either by incorrectly matching it with another minority or majority entity or by missing the opportunity to correctly match it with another minority entity.

In our definition of blocking disparities, we use P_g, C_g, and M_g with $g \in \{a, b\}$ to respectively refer to the set of pairs in group g, and the set of candidate pairs in group g, and the set of equivalent pairs in group g. Then we define the reduction ratio and pair completeness for the minority groups g as follows:

$$\text{RR}_g = 1 - \frac{|C_g|}{|P_g|} \qquad \text{PC}_g = \frac{|C_g \cap M|}{|M_g|} \tag{3}$$

Intuitively, the reduction ratio per group g specifies the reduction in the number of comparisons for the entity pairs in group g. Similarly, pair completeness per group g measures these measures between the pairs in the group g only. We don't define and use PQ as we explained due to its sensitivity and redundancy with RR and PC. Using RR and PC per group, we define their harmonic mean per group g that is $\text{F}^g_{\text{RR,PC}}$. We now use our running example to explain these quality measures per group g.

We define disparities as the differences between the blocking quality measures for the majority and minority groups. For example, $\Delta\text{RR} = \text{RR}_b - \text{RR}_a$ represents the reduction ratio disparity, $\Delta\text{PC} = \text{PC}_b - \text{PC}_a$ denotes the pair completeness disparity, and $\Delta\text{F}_{\text{RR,PC}} = \text{F}^b_{\text{RR,PC}} - \text{F}^a_{\text{RR,PC}}$ indicates the mean

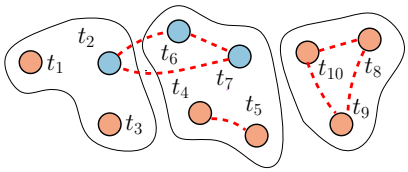


Fig. 1: Disparity in blocking: minority and majority entities are highlighted in blue and red resp. and the equivalent pairs are linked by dotted lines. Solid lines show the blocks.

disparity. An important observation from our experimental analysis is that these disparities can be negative; this contrasts with general disparities in fairness literature, such as demographic disparity, where disparities are typically assumed to be non-negative (i.e., minorities can only be discriminated against). In the context of blocking, some methods may actually perform better for minority groups due to the specific nature of the blocking task—a topic we further discuss in the experimental section.

Example IV.1. *Continuing with the example in Figure 1, three entities are from the minority group, and seven are from the majority groups, as highlighted by different colors. These give five majority pairs (with both entities from the majority group) and* $3\times7+3=24$ *minority pairs (with at least one entity from the minority group). The following are the blocking quality measures per group:* $RR_a = 1 - 7/24 \approx 0.71$ *(there are seven minority pairs after blocking),* $PC_a = 1/3 \approx 0.33$, $F^a_{RR,PC} \approx 0.45$ $RR_b = 1 - 5/21 = 0.76$, $PC_b = 4/4 = 1$, *and* $F^b_{RR,PC} \approx 0.86$. *These measures give reduction ratio and pair completeness disparities of* $\Delta RR \approx 0.76 - 0.71 = 0.05$, $\Delta PC \approx 0.66$, *and* 0.45, *respectively. In this example, all disparities are positive, indicating a lower blocking quality for the minority group a. This is evident in the missing equivalences for the minority pairs (2 missing pairs for the minority vs. none for the majority) and the smaller reduction gain for the minority group (reduction from 24 to 7 for the minority vs. 21 to 5 for the majority).*

Introducing these disparity metrics provides a tool for evaluating the fairness of blocking techniques. By quantifying the differences in the blocking quality between demographic groups, it becomes possible to identify potential biases and inequalities in the blocking process.

V. Evaluation and Analysis

The purpose of our experiments is twofold: first, to understand the quality of the existing blocking methods for the benchmarks we employed in this paper where we use RR, PC, $F_{RR,PC}$, and time to compare; second, to analyze the same methods in terms of their possible biases using the introduced measures of disparity.

A. Experimental Setup

We briefly explain the datasets and the blocking methods used in this paper before presenting the experimental results.

1) Datasets: We utilize datasets from prominent EM benchmarks: Amazon-Google (`AMZ-GOO`), Walmart-Amazon (`WAL-AMZ`), DBLP-Google Scholar (`DBLP-GOO`), DBLP-ACM (`DBLP-ACM`), Beer (`BERR`), Fodors-Zagat (`FOD-ZAG`), iTunes-Amazon (`ITU-AMZ`), and Febrl (`FEBRL`), as referenced in [1], [56]. The research community commonly adopts these datasets to evaluate EM system performance. Their full details are available in our GitHub repository.

Consistent with prior studies [27], [52], [26], we categorize entities into minority and majority groups across various datasets. For instance, in `DBLP-ACM`, including a female name in the "authors" attribute defines the minority. Similarly, entities in the `FEBRL` dataset are considered a minority if the "Given Name" attribute is female. In `FOD-ZAG`, entities with the "Type" attribute exactly equal to "Asian" are classified as minority. In `AMZ-GOO`, the presence of "Microsoft" in the "manufacturer" attribute signifies a minority group. For `WAL-AMZ`, entities classified as "printers" under the "category" attribute are considered a minority group. In `DBLP-GOO`, entities are considered minority if the "venue" attribute includes "vldb j,". For `BERR`, entities with "Beer Name" containing the phrase "red" are classified as minority. Finally, in `ITU-AMZ`, the minority group includes those where the "Genre" attribute contains the word "Dance,".

Detailed statistical information about these datasets is provided in Table I. The numbers in the bracket refer to the corresponding parameter in the minority group, e.g., $2.6k$ (96) for $|D_1|$ in `WAL-AMZ` means there are $2.6k$ entities in D_1 while only 96 are a minority. The parameters for the majority group can be clearly inferred from the data in the table.

Dataset	#Attr.	$\lvert D_1\rvert$	$\lvert D_2\rvert$	$\lvert P\rvert$	$\lvert M\rvert$
`WAL-AMZ`	5	2.6k (96)	22.0k (172)	56.4m (2.5m)	962 (88)
`BERR`	4	4.3k (1.3k)	3.0k (932)	13.0m (6.8m)	68 (29)
`AMZ-GOO`	3	1.4k (83)	3.2k (4)	4.4m (272.9k)	1.2k (60)
`FOD-ZAG`	6	533 (72)	331 (3)	176.4k (25.2k)	111 (10)
`ITU-AMZ`	8	6.9k (1.9k)	55.9k (12.7k)	386.2m	132 (40)
`DBLP-GOO`	4	2.6k (191)	64.3k (389)	168.1m	5.3k (403)
`DBLP-ACM`	4	2.6k (251)	2.3k (225)	6.0m (1.1m)	2.2k (310)

TABLE I: Datasets and their characteristics.

2) Blocking Methods: The blocking methods used in our study are as follows:

i) *Standard Blocking (`StdBlck`)*: This hash-based method groups records using concatenated attribute values (blocking keys) to form redundancy-free blocks. However, it is sensitive to noise; any variation in the key may exclude matching records from the same block [57].

ii) *Q-Grams (`QGram`) and Extended Q-Grams Blocking (`XQGram`)*: Q-grams blocking splits blocking keys into subsequences of q characters, improving noise tolerance but potentially increasing block size and number. Extended Q-Grams further combines q-grams for more distinctive keys, reducing block size and enhancing efficiency [21], [58].

iii) *Suffix (`Suffix`) and Extended Suffix (`XSuffix`) Arrays Blocking*: Suffix Arrays Blocking converts blocking keys

into suffixes of a minimum length to form blocks, filtering out common suffixes to prevent oversized blocks. Extended Suffix Arrays consider all substrings longer than the minimum length, boosting noise tolerance [21], [58].

iv) *AutoEncoder (AUTO) and Cross-Tuple Training (CTT) Blocking*: These deep learning methods group similar records into blocks using embeddings. AUTO generates embeddings via an autoencoder to handle diverse and noisy data, while CTT uses a Siamese Summarizer to create embeddings from synthetic data, enhancing distinction between matching and non-matching tuples [1].

v) *Semantic-Based Graph Blocking (GRAPH)*: The algorithm concatenates entity attributes into a single string, from which a transformer-based model extracts context-aware embeddings. These embeddings are reduced to a lower-dimensional space for efficiency. A k-nearest neighbor (KNN) algorithm then constructs a graph of database tuples, connecting nodes based on embedding proximity. Unsupervised clustering is applied to group similar entities, framing the blocking task as a graph clustering problem [48].

We selected these blocking methods as they represent the best-performing approaches in their respective categories. StdBlck, QGram, and Suffix (along with their extended versions) are widely recognized traditional methods and remain commonly used in practice due to their effectiveness and efficiency. With the growing trend of employing deep learning in blocking, we included AUTO and CTT, which have demonstrated superior performance in handling complex and noisy data. Additionally, we chose GRAPH, an innovative and recent approach, to represent cutting-edge advancements in the blocking problem.

B. Experimental Results

Our experimental results include an analysis of blocking methods based on their performance—specifically, the quality of blocking and runtime—which aligns with our first experimental objective. Additionally, we conduct an extensive analysis of the existing biases in these methods to address our second objective.

1) Runtime and Scalability: A primary concern in blocking is ensuring that the methods are not only fast compared to matching methods but also scale effectively as the data size increases. Table II and Figure 2 present the runtime of various blocking methods across datasets of different sizes (measured by the number of entity pairs). These results reflect the average runtime over five runs per dataset, with a negligible standard deviation (less than 0.5 seconds) and thus not reported. Notably, all methods were completed within a reasonable timeframe (less than 1 hour), even for datasets with up to 500 million pairs. Some methods even finished in under a minute for the largest datasets.

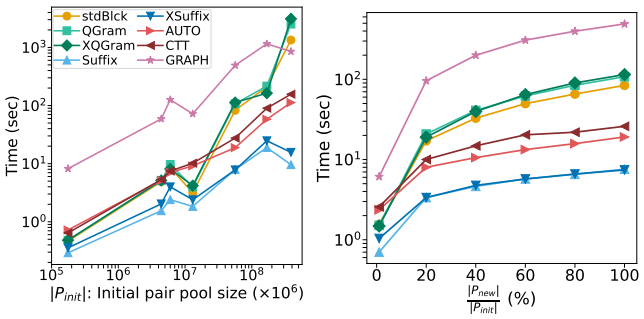
While all methods are relatively fast, their scalability varies. This variance can be attributed to the underlying computational processes of each approach. Among the methods tested, Suffix and XSuffix exhibited the fastest performance and best scalability. These methods generate suffixes from blocking keys, a process known for its linear or near-linear time complexity. Their straightforward operations, like filtering out common suffixes to prevent oversized blocks, enable them to maintain speed even as dataset sizes increase, making them particularly suitable for large-scale applications.

In contrast, AUTO and CTT, though fast, do not match the performance of the suffix-based methods. These approaches use deep learning models to generate embeddings that group similar records into blocks. Although deep learning can be computationally intensive, AUTO and CTT are optimized for efficiency. AUTO, for instance, employs an autoencoder—a relatively simple neural network—to produce embeddings quickly. Similarly, CTT uses a Siamese network to generate embeddings that effectively distinguish between matching and non-matching records. Despite their efficiency, the additional complexity of these methods results in slightly lower performance and scalability compared to the suffix-based methods.

On the other hand, QGram and XQGram, which split blocking keys into multiple subsequences, perform faster on smaller datasets but do not scale as well as AUTO and CTT. The increase in block size and the number of comparisons required as the dataset grows significantly impacts their efficiency. Finally, GRAPH is the slowest among the tested methods. It involves generating embeddings using a transformer-based model and constructing a graph of database tuples—processes that are computationally intensive. This complexity, especially in the graph clustering stage, leads to longer runtimes and poorer scalability when handling larger datasets.

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	4.9s	1.4m	3.5m	8.5s	3.2s	0.5s	22.3m
QGram	5.1s	1.8m	3.6m	9.7s	4.0s	0.5s	42.0m
XQGram	5.1s	1.9m	2.7m	8.1s	4.2s	0.5s	51.8m
Suffix	**1.5s**	7.9s	**18.9s**	**2.4s**	**1.8s**	**0.3s**	**9.6s**
XSuffix	2.0s	**7.8s**	24.7s	4.0s	2.4s	0.4s	15.7s
AUTO	5.2s	18.9s	58.1s	7.0s	9.2s	0.7s	1.9m
CTT	5.3s	27.6s	1.5m	7.5s	10.2s	0.6s	2.6m
GRAPH	58.9s	8.2m	19.2m	2.1m	1.2m	8.2s	14.2m

TABLE II: Runtime of blocking methods



a) All datasets b) WAL-AMZ

Fig. 2: Runtime of blocking methods

2) Quality of Blocking: Table III presents RR of various blocking methods across multiple datasets. Across most

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	99.73	99.81	99.94	99.94	99.91	98.72	99.86
QGram	99.69	99.77	99.95	99.94	99.90	98.83	99.79
XQGram	99.70	99.75	99.95	99.94	99.90	98.88	99.77
Suffix	99.85	99.96	99.98	99.95	**99.95**	99.31	**99.99**
XSuffix	**99.86**	**99.97**	**99.99**	99.94	**99.95**	**99.33**	**99.99**
AUTO	98.45	99.77	99.92	97.82	98.33	84.89	99.91
CTT	98.45	99.77	99.92	97.82	98.33	84.89	99.91
GRAPH	97.74	97.59	98.14	**99.96**	98.91	97.26	98.37

TABLE III: RR values for different models across datasets, with best (bold), second-best (underlined), and worst (framed) highlighted

datasets, all methods achieve high RR values close to 1, indicating their general effectiveness in reducing candidate pairs. However, Suffix and XSuffix consistently outperform the other methods, particularly in larger datasets like AMZ-GOO, WAL-AMZ, and ITU-AMZ, with XSuffix achieving the highest RR in most cases. This superior performance can be attributed to the inherent efficiency of suffix-based methods in generating compact and discriminative blocks. By focusing on suffixes and filtering out common ones, these methods avoid generating overly large blocks, which is particularly beneficial when handling large datasets with significant redundancy.

On the other hand, AUTO and CTT, while generally effective, show noticeable underperformance in specific datasets such as FOD-ZAG. This lower RR could be linked to the complexity and diversity of the data in this particular dataset. AUTO and CTT rely on deep learning-based embeddings to group similar records into blocks. While these embeddings are powerful in capturing complex patterns, they may not be as effective when the dataset size is smaller or when there is less variation among records, as seen in FOD-ZAG. The deep learning models might struggle to generate sufficiently discriminative embeddings, leading to less effective blocking and a subsequent drop in RR.

Mohammad-Hossein: The size of the dataset is not the only factor that can lead to the underperformance of these deep learning methods in RR. Another significant factor is the size of the M. For example, although the size of BERR is more than twice that of DBLP-ACM, the RR for DBLP-ACM is slightly higher. This discrepancy is due to the M for DBLP-ACM being 2.2k, compared to only 68 in BERR. Consequently, the model was more successful in capturing better embeddings for the matches in the DBLP-ACM dataset compared to BERR, resulting in a larger fraction of the initial data being filtered out in DBLP-ACM. Additionally, the ratio of the sizes of M and P is not always a determining factor. For instance, WAL-AMZ has a lower ratio of M to P compared to AMZ-GOO, yet it achieves a higher RR. In contrast, it is the size of each M and P that needs to be considerably large for the model to effectively capture their suitable embeddings.

The GRAPH method, which involves generating embeddings using a transformer-based model and applying graph clustering, shows the lowest RR values in multiple datasets, including AMZ-GOO, WAL-AMZ, and ITU-AMZ. This method's lower performance could be due to the computationally intensive nature of both embedding generation and graph clustering, which may not scale well for larger datasets. Additionally, the graph construction process might introduce noise or redundancy, leading to larger blocks that reduce the overall RR. The method performs better in DBLP-ACM, where it achieves the highest RR, indicating that it may be more suited to datasets where the relationships between records are complex but the dataset size is manageable.

Interestingly, the overall RR is lower in the FOD-ZAG dataset for all methods, which might be due to the smaller size of this dataset. In smaller datasets, the relative lack of variation and fewer distinct blocking keys can result in less effective blocking. As a result, even high-performing methods like Suffix and XSuffix show a slight decrease in RR. The specific underperformance of AUTO, CTT, and GRAPH in this dataset suggests that methods relying heavily on complex models or embeddings may not always be well-suited to smaller datasets with less diversity.

Mohammad-Hossein: However, it is important to keep in mind the initial goal of using RR as a measure. We aim to use RR to ensure that the matcher can efficiently process the output of the blocker, so there is no pre-set lower limit for RR. The appropriate RR depends on the computational power of the matcher and the size of the initial dataset. For instance, a model with an RR of 99.99% for ITU-AMZ produces a similar number of pairs as output as it would if it had only an 80% RR for the FOD-ZAG dataset.

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	**98.29**	**99.06**	98.73	99.86	**95.59**	**100.0**	**97.73**
QGram	95.72	**99.06**	**98.75**	**99.95**	92.65	**100.0**	73.48
XQGram	94.17	98.75	97.59	**99.95**	91.18	**100.0**	71.97
Suffix	88.52	91.16	82.38	99.91	88.24	**100.0**	50.76
XSuffix	83.89	88.36	76.47	99.46	89.71	97.32	51.52
AUTO	88.52	96.36	95.23	99.86	85.29	**100.0**	90.15
CTT	95.97	97.51	95.96	99.86	94.12	**100.0**	91.67
GRAPH	93.92	83.89	79.13	98.78	88.24	73.21	54.55

TABLE IV: PC values for different models across datasets

Table IV illustrates PC. Unlike RR, where most methods perform uniformly well, PC values reveal significant variations in performance. In datasets like AMZ-GOO, Suffix, XSuffix, and AUTO underperform compared to StdBlck, QGram, and CTT, which achieve higher PC values. This disparity can be attributed to these methods' characteristics and the dataset's specific nature. Suffix-based methods like Suffix and XSuffix focus on specific portions of the blocking keys. In cases where the suffixes are not sufficiently distinctive, these methods may over-partition the data, causing some true matches to be split into different blocks, thereby reducing PC.

Similarly, in the ITU-AMZ dataset, QGram, XQGram, Suffix, and XSuffix show significantly lower PC values, indicating that they miss a substantial number of true matches.

This could be due to the highly variable or sparse nature of the data in this dataset, where blocking keys may not align well across records. As a result, these methods struggle to capture all true matches. In contrast, StdBlck, AUTO, and CTT perform better in preserving most equivalences in ITU-AMZ. AUTO and CTT, which use embeddings to capture more complex similarities between records, manage to maintain higher PC even in challenging datasets.

The poor performance of GRAPH in several datasets, such as WAL-AMZ and FOD-ZAG, highlights the limitations of graph-based methods when the embedding space does not align well with the underlying data structure. In cases where the data lacks clear relational patterns or where the embeddings fail to capture these patterns accurately, GRAPH may struggle to retain true matches, leading to lower PC.

A general takeaway is that suffix-based methods like Suffix and XSuffix may perform well for datasets with structured and consistent blocking keys. However, deep learning-based methods like AUTO and CTT, which generate embeddings that can capture complex similarities, may be more effective for datasets with more complex or noisy data. For datasets where the relationships between records are intricate and graph-like, methods like GRAPH could be considered, although their performance may vary depending on how well the embeddings align with the data structure.

Mohammad-Hossein: In addition to the dataset structure and the architecture of the blocking method, the ratio $\frac{|M|}{|P|}$ is a factor that can affect the PC of different blocking methods. Intuitively, as this ratio decreases, the blocker faces a more challenging task of identifying the true matches among the non-matches. For example, this ratio is on the order of $O(10^{-7})$ for ITU-AMZ, whereas it is on the order of $O(10^{-3})$ for FOD-ZAG. As shown in Table III, different methods exhibit a significantly higher PC in FOD-ZAG compared to ITU-AMZ. This observation generally holds true across other datasets we examined. The reason this is not always the case is that this ratio is not the only parameter influencing the overall PC, as discussed earlier.

Mohammad-Hossein: Additionally, a low PC of a blocking method on a dataset does not necessarily indicate that the situation is significantly worse compared to another blocking method with a higher PC. For example, in the BERR dataset, the 10% difference between the best and worst PC means that AUTO missed 7 more true matches out of the 13 million initial pairs compared to StdBlck, which has the highest PC in the BERR dataset. However, in DBLP-ACM, a 1.17% difference between the best and worst methods results in 25 more true matches being missed out of the 6 million total initial pairs. These examples demonstrate that it is essential to consider the statistics of the initial dataset when assessing the effectiveness of a blocking method.

The data in Table V highlights the sensitivity of PQ to RR, demonstrating that even a small variation in RR can lead to a significant change in PQ. For instance, in the AMZ-GOO dataset, Suffix achieves a PQ of 16.06, while AUTO, with a slightly lower RR (99.85 for Suffix versus 98.45 for AUTO), sees a drastic drop in PQ to 1.52. This substantial decline can be attributed to the large number of candidate pairs in AMZ-GOO; when RR decreases, the number of candidate pairs increases, diluting the concentration of true matches and thereby reducing PQ. A similar trend is observed in the WAL-AMZ dataset when comparing Suffix and XQGram. Suffix, with its slightly higher RR, maintains a higher PQ (3.49) compared to XQGram (0.69). In the DBLP-ACM dataset, XSuffix achieves a PQ of 17.78, outperforming CTT (3.92), despite both methods having relatively high RR. These examples further illustrate how PQ is influenced by even small changes in RR, particularly in datasets with a large number of pairs.

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	9.58	0.90	5.34	66.26	0.54	4.96	0.02
QGram	8.29	0.75	6.22	59.22	0.51	5.40	0.01
XQGram	8.40	0.69	5.99	60.25	0.48	5.66	0.01
Suffix	16.06	3.49	**16.61**	**73.06**	0.88	**9.23**	**0.24**
XSuffix	**15.85**	**4.31**	17.78	60.34	**0.89**	**9.23**	0.19
AUTO	1.52	0.73	3.89	1.69	0.03	0.42	0.03
CTT	1.64	0.73	3.92	1.69	0.03	0.42	0.04
GRAPH	0.45	0.01	0.01	61.75	0.02	1.70	0.001

TABLE V: PQ values for different models across datasets

Interestingly, the impact of RR on PQ is less pronounced in smaller datasets like BERR and ITU-AMZ. For these datasets, the absolute number of candidate pairs is lower, so variations in RR have a smaller effect on PQ. For example, in the ITU-AMZ dataset, the PQ values for most methods remain relatively low across the board, with only minor differences between them. This observation supports the idea that in smaller datasets, where the pool of candidate pairs is naturally limited, the effect of RR on PQ is diminished.

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	**99.00**	**99.44**	99.33	99.90	**97.70**	99.36	**98.78**
QGram	97.66	99.42	**99.34**	**99.95**	96.14	99.41	84.64
XQGram	96.86	99.25	98.75	**99.95**	95.34	99.44	83.62
Suffix	93.84	95.36	90.33	99.93	93.73	**99.65**	67.34
XSuffix	91.18	93.80	86.66	99.70	94.55	98.32	68.00
AUTO	93.22	98.04	97.52	98.83	91.35	91.83	94.78
CTT	97.20	98.63	97.90	98.83	96.18	91.83	95.61
GRAPH	95.78	90.22	87.61	99.37	93.27	83.54	70.18

TABLE VI: $F_{RR,PC}$ values for different models across datasets

As the final quality measure, we report the F1 measure ($F_{RR,PC}$) in Table VI. The results indicate that no single method consistently outperforms the others across all datasets, reflecting the variability observed in PC. However, some methods, such as StdBlck, consistently deliver strong performance across various datasets, showcasing stable and reliable results overall. StdBlck, in particular, achieves the highest or near-highest $F_{RR,PC}$ values in most cases, making it a robust choice for different dataset characteristics. Other methods, like QGram and Suffix, also demonstrate strong performance in specific datasets, but their effectiveness varies depending

on the dataset's properties. For example, Suffix performs exceptionally well in datasets like FOD-ZAG, which achieves the highest $F_{RR,PC}$, but it struggles in ITU-AMZ, where its $F_{RR,PC}$ drops significantly. This variability underscores the importance of selecting a blocking method that aligns with the specific characteristics of the dataset being analyzed. In contrast, methods like AUTO and GRAPH tend to have more variable $F_{RR,PC}$ scores, reflecting their sensitivity to the dataset's complexity and the nature of the data. AUTO, for instance, performs well in some datasets but falls behind in others, particularly in DBLP-ACM and BERR. Similarly, GRAPH shows significant fluctuations in $F_{RR,PC}$, with particularly low scores in datasets like FOD-ZAG and ITU-AMZ, highlighting its potential limitations in handling certain types of data. Overall, while StdBlck emerges as a consistently strong performer across datasets, the choice of the optimal blocking method should still be guided by the specific requirements and characteristics of the dataset. The variability in $F_{RR,PC}$ values across methods emphasizes the need for careful consideration when selecting a blocking strategy, especially in scenarios where both high PC and RR are critical.

Mosatafa: Up to hear.

3) Bias Analysis of Blocking Methods: We now turn our analysis to studying biases in the blocking methods. We start with disparities in the reduction ratio, ΔRR. Table VII shows RR per minority and majority group together with their difference, which defines the disparity per dataset and per method. The disparities are insignificant (less than one percentage point) across all blocking methods and datasets. This can be attributed to the fact that RR was close to 1, its maximum value, as reported in Table III. However, some methods exhibit higher disparities, such as GRAPH in most datasets and AUTO in DBLP-ACM. These methods also have slightly lower overall RR, as reported in Table III. Mohammad-Hossein: Given that the RR is generally high (with the worst-case scenario being 84.89 in FOD-ZAG, a very small dataset), and the difference between minority and majority groups is minimal (a maximum difference of less than 2%), a higher RR for a particular group indicates that the model faced challenges in reducing the size of that group compared to the other. This bias typically favors the larger majority group. However, a lower RR for the minority group is not a significant concern because the minority group is relatively small, and even in the worst case, the minority group's RR remained above 80%, which is still acceptable.

Mohammad-Hossein: Consequently, the primary focus of bias mitigation algorithms should be on PC and $F_{RR,PC}$, with a lesser emphasis on RR. Additionally, given that RR is typically very high, $F_{RR,PC}$ will exhibit behavior similar to PC. This implies that reducing bias in PC will eventually lead to a reduction in bias in $F_{RR,PC}$, considering that RR remains high and its bias is low. Furthermore, a low PC for a group suggests that the blocker missed more true matches for that group, negatively impacting the overall quality of the EM pipeline. Therefore, it is advisable to employ a blocking method that achieves a higher overall PC and minimizes PC disparity among different subgroups.

Mohammad-Hossein: This approach contrasts with the bias mitigation algorithm proposed by [28], which aims to eliminate bias in RR. However, addressing RR bias alone will not resolve any significant issues and could potentially reduce PC. This reduction could occur if pairs from a subgroup are removed to increase that subgroup's RR, thus reducing RR disparity but simultaneously decreasing PC for that group.

VI. Conclusion

References

[1] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *SIGMOD*, 2018, pp. 19–34.

[2] R. Wu, S. Chaba, S. Sawlani, X. Chu, and S. Thirumuruganathan, "Zeroer: Entity resolution using zero labeled examples," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1149–1164.

[3] C. Fu, X. Han, J. He, and L. Sun, "Hierarchical matching network for heterogeneous entity resolution," in *IJCAI*, 2021, pp. 3665–3671.

[4] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan, "Deep entity matching with pre-trained language models," *arXiv preprint arXiv:2004.00584*, 2020.

[5] D. Yao, Y. Gu, G. Cong, H. Jin, and X. Lv, "Entity resolution with hierarchical graph attention networks," in *SIGMOD*, 2022, pp. 429–442.

[6] P. V. Konda, *Magellan: Toward building entity matching management systems*. The University of Wisconsin-Madison, 2018.

[7] G. Simonini, G. Papadakis, T. Palpanas, and S. Bergamaschi, "Schema-agnostic progressive entity resolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 6, pp. 1208–1221, 2018.

[8] J. Jonas and J. Harper, *Effective counterterrorism and the limited role of predictive data mining*. JSTOR, 2006.

[9] M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," in *AAAI*, vol. 6, 2006, pp. 440–445.

[10] M. Bilenko, B. Kamath, and R. J. Mooney, "Adaptive blocking: Learning to scale up record linkage," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 87–96.

[11] J. MESTS and M. Tang, "Distributed representations of tuples for entity resolution," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, 2018.

[12] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "A survey of blocking and filtering techniques for entity resolution," *ACM Reference Format*, vol. 1, no. 1, pp. 1–38, Aug 2020.

[13] B.-H. Li, Y. Liu, A.-M. Zhang, W.-H. Wang, and S. Wan, "A survey on blocking technology of entity resolution," *Journal of Computer Science and Technology*, vol. 35, pp. 769–793, 2020.

[14] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2000, pp. 169–178.

[15] M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," in *Proceedings of the National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, 2006.

[16] L. O. Evangelista, E. Cortez, A. S. da Silva, and W. Meira Jr, "Adaptive and flexible blocking for record linkage tasks," *Journal of Information and Data Management*, vol. 1, no. 2, pp. 167–181, 2010.

[17] W. Zhang, H. Wei, B. Sisman, X. L. Dong, C. Faloutsos, and D. Page, "Autoblock: A hands-off blocking framework for entity matching," in *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20)*. Houston, TX, USA: ACM, 2020, p. 10.

[18] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *Proc. VLDB Endow.*, vol. 11, no. 11, pp. 1454–1467, jul 2018. [Online]. Available: https://doi.org/10.14778/3236187.3236198

[19] P. Christen and K. Goiser, "Quality and complexity measures for data linkage and deduplication," in *Quality measures in data mining*. Springer, 2007, pp. 127–151.

[20] M. G. Elfeky, V. S. Verykios, and A. K. Elmagarmid, "Tailor: A record linkage toolbox," in *Proceedings 18th International Conference on Data Engineering*. IEEE, 2002, pp. 17–28.

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	0.08 (99.73, 99.66)	0.14 (99.82, 99.68)	-0.05 (99.94, 99.99)	**-0.01** (99.94, 99.95)	**-0.01** (99.90, 99.91)	0.01 (98.72, 98.71)	-0.13 (99.80, 99.93)
QGram	0.38 (99.72, 99.33)	0.25 (99.79, 99.54)	-0.03 (99.95, 99.98)	**-0.01** (99.93, 99.95)	**-0.01** (99.90, 99.91)	-0.20 (98.80, 99.00)	-0.15 (99.73, 99.88)
XQGram	0.41 (99.73, 99.32)	0.18 (99.76, 99.58)	-0.03 (99.95, 99.97)	**-0.01** (99.94, 99.95)	**-0.01** (99.90, 99.90)	-0.16 (98.86, 99.01)	-0.20 (99.68, 99.88)
Suffix	0.07 (99.86, 99.79)	-0.01 (99.95, 99.97)	**-0.00** (99.98, 99.98)	**-0.01** (99.95, 99.96)	-0.02 (99.94, 99.96)	-0.18 (99.28, 99.46)	**-0.00** (99.99, 99.99)
XSuffix	0.04 (99.86, 99.82)	**-0.00** (99.96, 99.97)	**0.00** (99.99, 99.99)	**-0.01** (99.94, 99.95)	-0.02 (99.94, 99.96)	-0.18 (99.30, 99.48)	**-0.00** (99.99, 99.99)
AUTO	**-0.01** (98.45, 98.46)	-0.01 (99.77, 99.79)	0.01 (99.92, 99.91)	-0.48 (97.73, 98.21)	0.04 (98.36, 98.31)	0.90 (85.02, 84.12)	-0.04 (99.89, 99.93)
CTT	**-0.01** (98.45, 98.46)	0.01 (99.77, 99.77)	0.01 (99.92, 99.92)	-0.21 (97.78, 97.99)	-0.09 (98.29, 98.38)	**0.00** (84.89, 84.89)	-0.04 (99.89, 99.93)
GRAPH	0.30 (97.74, 97.45)	0.81 (97.62, 96.81)	1.39 (98.25, 96.85)	**-0.01** (99.95, 99.97)	-0.24 (98.79, 99.03)	0.98 (97.40, 96.42)	-1.09 (97.89, 98.98)

TABLE VII: For each cell, a (b,c) shows RR values within minority and majority groups (b,c) and the RR disparities (a)

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	1.71 (98.37, 96.67)	1.47 (99.20, 97.73)	0.77 (98.79, 98.01)	-0.16 (99.84, 100.00)	-1.68 (94.87, 96.55)	**0.00** (100.00, 100.00)	7.50 (100.00, 92.50)
QGram	**-1.00** (95.66, 96.67)	1.47 (99.20, 97.73)	1.33 (98.85, 97.52)	**-0.05** (99.95, 100.00)	-6.81 (89.74, 96.55)	**0.00** (100.00, 100.00)	-9.35 (70.65, 80.00)
XQGram	6.16 (94.49, 88.33)	1.13 (98.86, 97.73)	1.15 (97.67, 96.53)	**-0.05** (99.95, 100.00)	-9.37 (87.18, 96.55)	**0.00** (100.00, 100.00)	**-0.76** (71.74, 72.50)
Suffix	16.01 (89.34, 73.33)	5.28 (91.65, 86.36)	**0.27** (82.40, 82.13)	-0.10 (99.90, 100.00)	-8.49 (84.62, 93.10)	**0.00** (100.00, 100.00)	1.09 (51.09, 50.00)
XSuffix	18.16 (84.82, 66.67)	**0.94** (88.44, 87.50)	1.66 (76.60, 74.94)	-0.63 (99.37, 100.00)	-5.92 (87.18, 93.10)	-2.97 (97.03, 100.00)	2.17 (52.17, 50.00)
AUTO	8.98 (88.98, 80.00)	4.75 (96.80, 92.05)	-0.60 (95.19, 95.78)	-0.16 (99.84, 100.00)	**-1.59** (84.62, 86.21)	**0.00** (100.00, 100.00)	-6.96 (88.04, 95.00)
CTT	2.78 (96.12, 93.33)	4.76 (97.94, 93.18)	0.46 (96.00, 95.53)	-0.16 (99.84, 100.00)	-4.24 (92.31, 96.55)	**0.00** (100.00, 100.00)	-1.20 (91.30, 92.50)
GRAPH	2.37 (94.04, 91.67)	17.29 (85.47, 68.18)	-7.28 (78.58, 85.86)	-1.04 (98.64, 99.68)	-2.48 (87.18, 89.66)	-6.73 (73.27, 80.00)	-4.24 (53.26, 57.50)

TABLE VIII: PC disparities and PC values within the minority and majority groups

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	0.91 (99.05, 98.14)	0.81 (99.51, 98.69)	0.37 (99.36, 98.99)	-0.08 (99.89, 99.97)	**-0.88** (97.32, 98.20)	**0.01** (99.36, 99.35)	3.83 (99.90, 96.07)
QGram	**-0.33** (97.65, 97.98)	0.87 (99.49, 98.62)	0.66 (99.39, 98.73)	**-0.03** (99.94, 99.97)	-3.65 (94.55, 98.20)	-0.10 (99.39, 99.49)	-6.13 (82.71, 88.84)
XQGram	3.53 (97.04, 93.51)	0.66 (99.31, 98.65)	0.58 (98.80, 98.22)	**-0.03** (99.94, 99.97)	-5.09 (93.11, 98.20)	-0.07 (99.42, 99.49)	**-0.58** (83.43, 84.02)
Suffix	9.77 (94.31, 84.54)	2.95 (95.62, 92.67)	0.16 (90.35, 90.18)	-0.06 (99.92, 99.98)	-4.77 (91.64, 96.41)	-0.09 (99.64, 99.73)	0.96 (67.62, 66.67)
XSuffix	11.79 (91.73, 79.94)	**0.53** (93.85, 93.32)	1.07 (86.74, 85.67)	-0.32 (99.65, 99.97)	-3.28 (93.12, 96.41)	-1.59 (98.15,99.74)	1.90 (68.57, 66.66)
AUTO	5.20 (93.48, 88.27)	2.50 (98.26, 95.76)	-0.30 (97.50, 97.80)	-0.32 (98.78, 99.10)	-0.89 (90.97, 91.86)	0.58 (91.91, 91.33)	-3.81 (93.59, 97.40)
CTT	1.44 (97.27, 95.83)	2.49 (98.85, 96.36)	**0.24** (97.92, 97.68)	-0.18 (98.80, 98.99)	-2.25 (95.20, 97.46)	0.02 (91.83, 91.81)	-0.67 (95.41, 96.07)
GRAPH	1.39 (95.86, 94.47)	11.13 (91.14, 80.01)	-3.70 (87.32, 91.02)	-0.53 (99.29, 99.82)	-1.49 (92.62, 94.11)	-3.82 (83.63, 87.44)	-3.76 (68.99, 72.74)

TABLE IX: $F_{RR,PC}$ disparities and $F_{RR,PC}$ values within the minority and majority groups

Model	AMZ-GOO	WAL-AMZ	DBLP-GOO	DBLP-ACM	BERR	FOD-ZAG	ITU-AMZ
StdBlck	0.9, 99.1	0.8, 99.5	0.4, 99.4	-0.1, 99.9	**-0.9**, 97.3	**0.01**, 99.36	3.8, 99.9
QGram	**-0.3**, 97.7	0.9, 99.5	0.7, 99.3	**-0.03**, 99.94	-3.7, 94.6	-0.1, 99.4	**-6.1**, 82.7
XQGram	3.5, 97.0	0.7, 99.3	0.6, 98.8	**-0.03**, 99.94	**-5.1**, 93.1	-0.07, 99.42	**-0.6**, 83.4
Suffix	9.8, 94.3	3.0, 95.6	**0.16**, 90.35	-0.06, 99.92	-4.8, 91.6	-0.09, 99.64	1.0, 67.6
XSuffix	**11.8**, 91.7	**0.5**, 93.9	1.1, 86.7	-0.3, 99.7	-3.3, 93.1	-1.6, 98.2	1.9, 68.6
AUTO	5.2, 93.5	2.5, 98.3	-0.3, 97.5	-0.3, 98.8	-1.0, 91.0	0.6, 91.9	-3.8, 93.6
CTT	1.4, 97.3	2.5, 98.6	0.24, 97.92	-0.2, 98.8	-2.3, 95.2	0.02, 91.83	-0.7, 95.4
GRAPH	1.4, 95.9	**11.1**, 91.1	**-3.7**, 87.3	**-0.5**, 99.3	-1.5, 92.6	**-3.8**, 83.6	-3.8, 67.0

TABLE X: $F_{RR,PC}$ disparities and $F_{RR,PC}$ values within the minority and majority groups

[21] P. Christen, "A survey of indexing techniques for scalable record linkage and deduplication," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 9, pp. 1537–1555, 2011.

[22] M. B. Zafar, I. Valera, M. Gomez Rodriguez, and K. P. Gummadi, "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 1171–1180.

[23] M. Hardt, E. Price, and N. Srebro, "Equality of opportunity in supervised learning," in *NIPS*, 2016, pp. 3315–3323.

[24] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, 2012, pp. 214–226.

[25] N. Shahbazi, N. Danevski, F. Nargesian, A. Asudeh, and D. Srivastava, "Through the fairness lens: Experimental analysis and evaluation of entity matching," *Proc. VLDB Endow.*, vol. 16, no. 11, p. 3279–3292, jul 2023. [Online]. Available: https://doi.org/10.14778/3611479.3611525

[26] M. H. Moslemi and M. Milani, "Threshold-independent fair matching through score calibration," in *Proceedings of the Conference on Governance, Understanding and Integration of Data for Effective and Responsible AI*, 2024, pp. 40–44.

[27] S. Nilforoushan, Q. Wu, and M. Milani, "Entity matching with auc-based fairness," in *Big Data*, 2022, pp. 5068–5075.

[28] N. Shahbazi, J. Wang, Z. Miao, and N. Bhutani, "Fairness-aware data preparation for entity matching," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 3476–3489.

[29] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas, "Comparative analysis of approximate blocking techniques for entity resolution," *Proceedings of the VLDB Endowment*, vol. 9, no. 9, pp. 684–695, 2016.

[30] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "Blocking and filtering techniques for entity resolution: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–42, 2020.

[31] A. D. Sarma, A. Jain, A. Machanavajjhala, and P. Bohannon, "An automatic blocking mechanism for large-scale de-duplication tasks," *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:11280427

[32] D. Javdani, H. Rahmani, M. Allahgholi, and F. Karimkhani, "Deepblock: A novel blocking approach for entity resolution using deep learning," in *2019 5th International Conference on Web Research (ICWR)*, 2019, pp. 41–44.

[33] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, and A. Doan, "Deep learning for blocking in entity matching: a design space exploration," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2459–2472, 2021.

[34] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *PVLDB*, vol. 11, no. 11, pp. 1454–1467, 2018.

[35] G. Papadakis, E. Ioannou, C. Niederée, and P. Fankhauser, "Efficient entity resolution for large heterogeneous information spaces," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 535–544.

[36] D. Paulsen, Y. Govind, and A. Doan, "Sparkly: A simple yet surprisingly strong tf/idf blocker for entity matching," *Proceedings of the VLDB Endowment*, vol. 16, no. 6, pp. 1507–1519, 2023.

[37] G. Papadakis, E. Ioannou, T. Palpanas, C. Niederée, and W. Nejdl, "A blocking framework for entity resolution in highly heterogeneous

information spaces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 12, pp. 2665–2682, 2012.

[38] R. C. Steorts, S. L. Ventura, M. Sadinle, and S. E. Fienberg, "A comparison of blocking methods for record linkage," in *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2014, Ibiza, Spain, September 17-19, 2014. Proceedings*. Springer, 2014, pp. 253–268.

[39] T. De Vries, H. Ke, S. Chawla, and P. Christen, "Robust record linkage blocking using suffix arrays," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 305–314.

[40] S. Galhotra, D. Firmani, B. Saha, and D. Srivastava, "Efficient and effective er with progressive blocking," *The VLDB Journal*, vol. 30, no. 4, pp. 537–557, 2021.

[41] H. Li, P. Konda, P. S. GC, A. Doan, B. Snyder, Y. Park, G. Krishnan, R. Deep, and V. Raghavendra, "Matchcatcher: A debugger for blocking in entity matching." in *EDBT*, 2018, pp. 193–204.

[42] A. Zeakis, G. Papadakis, D. Skoutas, and M. Koubarakis, "Pre-trained embeddings for entity resolution: an experimental analysis," *Proceedings of the VLDB Endowment*, vol. 16, no. 9, pp. 2225–2238, 2023.

[43] A. D. Sarma, A. Jain, A. Machanavajjhala, and P. Bohannon, "Cblock: An automatic blocking mechanism for large-scale de-duplication tasks," *arXiv preprint arXiv:1111.3689*, 2011.

[44] K. O'Hare, A. Jurek-Loughrey, and C. d. Campos, "A review of unsupervised and semi-supervised blocking methods for record linkage," *Linking and Mining Heterogeneous and Multi-view Data*, pp. 79–105, 2019.

[45] K. O'Hare, A. Jurek, and C. de Campos, "A new technique of selecting an optimal blocking method for better record linkage," *Information Systems*, vol. 77, pp. 151–166, 2018.

[46] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," *Data & Knowledge Engineering*, vol. 69, no. 2, pp. 197–210, 2010.

[47] M. Kejriwal and D. P. Miranker, "An unsupervised algorithm for learning blocking schemes," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 340–349.

[48] J. B. Mugeni and T. Amagasa, "A graph-based blocking approach for entity matching using contrastively learned embeddings," *ACM SIGAPP Applied Computing Review*, vol. 22, no. 4, pp. 37–46, 2023.

[49] ACLU, "Airline "no fly" lists trample the rights of people of color; seattle should not allow hotels to create a similar system," https://www.aclu-wa.org/story/airline-%E2%80%9Cno-fly%E2%80%9D-lists-trample-rights-people-color-seattle-should-not-allow-hotels-create--0, 2020, accessed: 2024-08-31.

[50] A. Peterson, "Google's algorithm shows prestigious job ads to men but not to women. here's why that should worry you," https://www.washingtonpost.com/news/the-intersect/wp/2015/07/06/googles-algorithm-shows-prestigious-job-ads-to-men-but-not-to-women-heres-why-that-should-worry-you/, 2015, accessed: 2024-08-31.

[51] V. Efthymiou, K. Stefanidis, E. Pitoura, and V. Christophides, "FairER: entity resolution with fairness constraints," in *CIKM*, 2021, pp. 3004–3008.

[52] N. Shahbazi, N. Danevski, F. Nargesian, A. Asudeh, and D. Srivastava, "Through the fairness lens: Experimental analysis and evaluation of entity matching," *Proc. VLDB Endow.*, vol. 16, no. 11, p. 3279–3292, 2023.

[53] L. E. Celis, D. Straszak, and N. K. Vishnoi, "Fairness first: Clustering in a multi-stage approach for mitigating bias," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2023.

[54] A. Chhabra, K. Masalkovaitė, and P. Mohapatra, "An overview of fairness in clustering," *IEEE Access*, vol. 9, pp. 130 698–130 720, 2021.

[55] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data mining and knowledge discovery*, vol. 2, pp. 9–37, 1998.

[56] K. Yang, B. Huang, J. Stoyanovich, and S. Schelter, "Fairness-aware instrumentation of preprocessing pipelines for machine learning," in *HILDA*, 2020.

[57] I. P. Fellegi and A. B. Sunter, "A theory for record linkage," *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[58] G. Papadakis, G. Alexiou, G. Papastefanatos, and G. Koutrika, "Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data," *Proceedings of the VLDB Endowment*, vol. 9, no. 4, pp. 312–323, 2015.