Fairness-aware Data Preparation for Entity Matching

Nima Shahbazi
University of Illinois Chicago
nshahb3@uic.edu

Jin Wang
Megagon Labs
jin@megagon.ai

Zhengjie Miao
Simon Fraser University
zhengjie@sfu.ca

Nikita Bhutani
Megagon Labs
nikita@megagon.ai

***Abstract*—Entity matching is a crucial task in many real applications. Despite the substantial body of research that focuses on improving the effectiveness of entity matching, enhancing its fairness has received scant attention. To fill this gap, this paper introduces a new problem of preparing fairness-aware datasets for entity matching. We formally outline the problem, drawing upon the principles of group fairness and statistical parity. We devise three highly efficient algorithms to accelerate the process of identifying an unbiased dataset from the vast search space. Our experiments on four real-world datasets show that our proposed algorithms can significantly improve fairness in the results while achieving comparable effectiveness to existing fairness-agnostic methods. Furthermore, we conduct case studies to demonstrate that our proposed techniques can be seamlessly integrated into end-to-end entity matching pipelines to support fairness requirements in real-world applications.**

***Index Terms*—Entity Matching, Fairness, Blocking, Pre-processing**

I. Introduction

Entity Matching (EM) is the task of determining whether two entity entries refer to the same real-world object. Owing to the large amount of data stemming from multiple sources, entity matching is crucial in data integration and cleaning applications. Consequently, there is a rich body of work on EM systems in the past few decades [1], [2], [3], [4], [5], [6]. Such EM systems typically adopt an end-to-end pipeline with three stages: blocking, labeling and matching.

There has been a long stream of studies to improve the effectiveness of EM in the past few decades. Recently pre-trained Language Models (LMs) have been illustrated to achieve promising results on EM tasks [7], [8], [9], [10], [11]. However, at the same time, the issue of fairness in EM applications has been largely ignored. Fairness is receiving increasing interest in recent years since several works have indicated that machine learning-based methods tend to reflect and even reinforce bias in the training data [12], [13]. As shown in a recent empirical study [14], the effective neural-based EM methods [8], [15] often produced unfair outcomes due to external bias from the training data or their backbone language models. Factors such as the selection of datasets and models may encode unintentional biases towards certain groups, resulting in systematic disparate impact. That is, records from some groups may be identified as matches at a significantly lower rate than those from other groups [14], [16], which will lead to serious real-world consequences such as voter suppression or inconveniencing certain groups in contexts like security and background checks as shown in the following example.

Example 1: Consider the airline security scenario that seeks to identify potentially dangerous passengers to prevent them from boarding flights. This can be realized by an EM task where passenger information is matched against a collection of historic records known as the no-fly list. In this process, false positives can cause inconvenience to passengers significantly, while false negatives can lead to undesirable outcomes by allowing known terrorists to get on board. Due to historical biases, the no-fly list records could exhibit an under/over-representation of a certain ethnicity, potentially resulting in elevated error rates for those demographics [14]. Thus, there is a need to mitigate such biases in the dataset to avoid the bias in the matching results.

Generally speaking, there are three broad categories of approaches to improve fairness of ML models: the *pre-processing* [17], [18] ones handle the unfairness in datasets before the model training process; the *in-processing* [19], [20] ones change the model training process itself; while the *post-processing* [21] ones update the prediction results after training and inference. To the best of our knowledge, the only previous studies for the fairness-aware entity matching problem are [22] and [23], which try to provide new fairness metrics but not solutions to resolve unfairness. We argue that it is crucial to address the fairness issues in training data creation, which is the root cause of unfairness. Thus, we focus on the problem of preparing fairness-aware datasets for entity matching that belongs to the pre-processing category, which aims at preparing a datasets with labeled entity pairs from the raw datasets of entity records. This problem is rather challenging due to the following reasons: On the one hand, the only resource available for debiasing is the data itself. Consequently, it is challenging to define the fairness condition precisely. On the other hand, the search space of the targeted problem is very large, and introducing fairness constraints will significantly increase the computational complexity.

In this paper, we proposed the first solution to prepare fairness-aware datasets for entity matching. Compared with current fair-agnostic methods for EM, our solution will create training data that includes enough high-quality positive instances while ensuring fairness. To this end, the first problem becomes choosing a proper definition of fairness. There are several prominent notions of fairness, namely the individual, group, and causal fairness [12]. As shown in the previous

This work was done when Nima Shahbazi and Zhengjie Miao were at Megagon Labs. Jin Wang is the corresponding author.

work [14], the disproportionate representation of demographic groups in the training data is a major reason for unfairness in the matching results. Thus, we choose to study fairness from a group perspective based on statistical parity, which is also the practice of several previous works [14], [24], [25] for data processing.

The next problem becomes how to integrate fairness into the end-to-end EM pipeline. We propose to apply the fairness constraints to the blocking stage, where the dataset used in the matching stage originates from. We formulate the problem as identifying a candidate set of entity pairs containing as many positive pairs as possible while satisfying the fairness constraints. Next, we propose three efficient algorithms to reduce the overhead due to the exponential search space of the above problem. We first devise a simple greedy solution that tends to miss many positive instances. To resolve this issue, we then develop a weight-based algorithm and a variant of it by taking both effectiveness and fairness into consideration during the search process. The experimental results on four popular datasets demonstrate the efficiency and effectiveness of our proposed solutions. We also conduct two case studies to show that our proposed solutions can be seamlessly integrated into EM pipelines in real application scenarios.

The rest of this paper is organized as follows: Section II introduces necessary background knowledge. Section III formally defines the problem and provides justification for it. Section IV proposes solutions to the problem. Section V reports the experiment results of our proposed methods. Section VI conducts two case studies to illustrate the application scenarios of our proposed methods. Section VII surveys the related work. Section VIII concludes the paper.

II. Background

A. Fairness Evaluation

In this paper, our definition of fairness is derived from the notion of statistical parity in group fairness [26], [12]. Group fairness is defined as having equal or similar performance by a model based on some fairness measure over different *demographic groups*, where each demographic group corresponds to one or a combination of several *sensitive attributes*. For instance, the values for the sensitive attribute race could be {White, Black, Asian, Hispanic}. Most of the group fairness definitions belong to one of the three categories of *independence*, *separation*, *sufficiency* [27]. A model satisfies *independence* if its outcome is independent of the sensitive attributes. Measures such as *Statistical Parity* fall into this category. *Separation* is satisfied when the model's outcome is independent of the sensitive attribute(s) conditioned on the target variable. The well-known measure in this category is *Equalized Odds*. On the other hand, *sufficiency* is satisfied if sensitive attributes and the true outcomes are independent under the same model outcomes. It can be measured with *Predictive Parity*. In the context of our problem, since the step of preparing datasets in the EM pipeline is oblivious to the matching outcomes, we opt for a group fairness definition based on the notion of independence. We will further elaborate on this choice in Section III-A.

Due to the pairwise nature of EM applications, fairness can be defined as either single or pairwise [14]:

- *Single fairness*: The output set is evaluated for one group g against either entity in a pair. Consider a pair $\langle e_x, e_y \rangle$ where e_x is the left entity and e_y is the right entity and each entity is associated with a demographic group. $\langle e_x, e_y \rangle$ is a legitimate pair for the group of interest g if either e_x or e_y belong to group g.
- *Pairwise fairness*: The output set is evaluated for a pair of groups g, g' against both entities in a pair. A pair $\langle e_x, e_y \rangle$ is legitimate if e_x belongs to g and e_y belongs to g' or vice versa.

They are equally important in evaluating fairness. In the following example, we examine the valid pairs for all possible groups based on the aforementioned definitions:

Index	ID_X	G_X	ID_Y	G_Y
1	x	A	x'	A
2	y	B	y'	B
3	w	A	x'	A
4	x	A	y'	B
5	y	B	x'	A

Fig. 1: Output set example

Example 2: Consider the example of an output set illustrated in Figure 1 where each pair comprises two entities x, y, each characterized by an *ID* and a sensitive attribute $G = \{A, B\}$. According to the definition of single fairness, pairs $\{1, 3, 4, 5\}$ are legitimate for group $\langle A \rangle$ and $\{2, 4, 5\}$ are for group $\langle B \rangle$. By definition of pairwise fairness, pairs $\{1, 3\}$ belong to group $\langle A, A \rangle$, pair $\{2\}$ belong to group $\langle B, B \rangle$ and pairs $\{4, 5\}$ are valid for group $\langle A, B \rangle$.

B. Entity Matching

In a standard end-to-end EM pipeline [3], [5], [6], [2], there are three basic stages, namely blocking, labeling and matching.

Blocking The blocking stage aims to reduce the overhead of matching by identifying a set of candidate entity pairs from the two input tables. The objective is to achieve a high recall, which means involving more positive pairs while keeping the candidate set small. The state-of-the-art blocking methods identify the positive pairs by conducting a similarity join over the two input tables of entities, which could be realized by syntactic similarity based methods [28], [29]. There are two ways to identify the blocking results: selecting pairs with top-k highest similarity or ones with similarity larger than a threshold. According to a recent empirical study [30], top-k search could always lead to higher-quality blocking results. The formal definition of the blocking stage with top-k search setting could be summarized as Definition 1:

Definition 1 (Blocking): Given two tables of entity entries $\mathcal{X}$ and $\mathcal{Y}$, the blocking stage aims at identifying a set R of

entry pairs $p = \langle x, y \rangle$, where $x \in \mathcal{X}, y \in \mathcal{Y}$ satisfying $\mid R \mid = k$ and $\forall p \in R, p' = \langle x', y' \rangle \in \mathcal{X} \times \mathcal{Y} - R$, $SIM(x, y) \geq SIM(x', y')$.
where the function SIM could be defined in multiple ways, which will be further introduced in Section III-B later.

Labeling The labeling stage aims at identifying a labeled set of entity pairs according to the output of the blocking stage. The labeling process could be realized by automatically applying heuristic rules, weak supervision, or involving human labor. Since the outcome of the blocking stage is usually much larger than that of the labeled set. A sampling process would also be implemented to reduce the labeling efforts.

Matching The matching stage aims to decide whether a pair of entity entries is matched or not. The goal is to build a matching model $\mathcal{M}$ that can achieve a high F_1 score in a test set. Earlier studies develop rule-based methods to realize $\mathcal{M}$ in an unsupervised way. Recently, deep learning techniques, especially pre-trained Language Models, have been adopted as the matching model in a supervised manner. Then, the outcome of the labeling stage could serve as the training and validation data. The formal definition of the matching stage is shown in Definition 2.

Definition 2 (Matching): Given a test set of pairs P and a matching model $\mathcal{M}$, it aims to assign each pair $p \in P$ a label with $\mathcal{M}$ to denote whether the pair of entity entries is matched.

III. Problem Formulation

A. Rationale for the Choice of Fairness

Similar with other real-world applications, achieving fairness in end-to-end EM requires a procedure that generates a training dataset devoid of representation bias. An *equal representation ratio* is defined as having a similar number of samples for different groups in the dataset. It can resolve the issues arising from representation bias and improve overall fairness [25], [12]. In fact, equal representation ratio enforces the well-known notion of statistical parity (a.k.a removing disparate impact). This is also aligned with practices regarding fair synthetic data generation as seen in [31], [32]. Statistical parity defines fairness as having an equal probability of being assigned to the positively predicted class (appearing in the candidate set in case of our problem) among different demographic groups [33] as shown in Definition 3.

Definition 3: Statistical parity: Given dataset D with binary sensitive attribute $G = \{0\text{:majority}, 1\text{:minority}\}$ and binary class to be predicted $Y = \{0, 1\}$, we say D has statistical parity if:

$$\frac{P(Y = 1|G = 0)}{P(Y = 1|G = 1)} - 1 \leq \tau$$

where τ is a context-specific threshold chosen by experts or based on a generalization of rules such as EEOC four-fifths [34].

Fairness in classification is achieved by removing *disparate impact* and *disparate treatment*. To eliminate disparate treatment, the classifier should avoid using sensitive attributes in its decisions. The notion of ϵ-fairness has been proposed to address the potential disparate impact [33] as Definition 4.

Definition 4: ϵ-fairness: A dataset D with sensitive attribute G and non-sensitive attributes X is said to be $\epsilon-$fair if for any classifier $f : X \rightarrow G$:

$$\mathsf{BER}(f(X), G) > \epsilon$$

with empirical probabilities estimated from D, where BER (balanced error rate) is defined as:

$$\mathsf{BER}(f(X), G) = \frac{P(f(X)|G = 0) + P(f(X)|G = 1)}{2}$$

BER indicates the average class-conditioned error of f on distribution D over the pair (X, G).
ϵ-fairness measures the fairness of data by assessing the error rate in predicting the sensitive attribute G from the non-sensitive attributes X. A low error rate indicates that G can be predicted by X. In the context of our problem, achieving fairness involves removing disparate impact related to the sensitive attribute from the candidate set.

Given the inherent class imbalance in EM where there are few matches among numerous non-matches, it is crucial for the model to exhibit fairness across different groups, particularly among the matched pairs. Thus, the fairness constraint can be formulated as reducing the disparity score, a.k.a. the difference between the representation ratio of the advantaged and disadvantaged groups in the candidate set. We can also see that Definition 3 and 4 support the claim that an equal representation ratio is a proper design choice for our problem.

Formally, let demographic groups be denoted as G where $g \in G$ is a sensitive attribute value and C_g and N_g denote the number of entities belonging to group g in the output set and the whole search space, respectively. Then, the proposed fairness constraint requires the output to have the same representation of each demographic group as in the original datasets they were selected from (i.e. $\forall g_i, g_j, \frac{C_{g_i}}{N_{g_i}} \simeq \frac{C_{g_j}}{N_{g_j}}$). In other words, pairs from different demographic groups should have equal opportunity to be included in the output set. Nevertheless, achieving zero demographic parity in real applications is always impractical. To address this issue, we borrow the idea from [33] and require the difference [1] between the representation ratio of demographic groups (denoted as Δ) in a dataset $\mathcal{D}$ to be within a given threshold τ. This is illustrated in Equation 1:

$$\Delta(\mathcal{D}) = \frac{\max\limits_{g_i \in G}(\frac{C_{g_i}}{N_{g_i}})}{\min\limits_{g_i \in G}(\frac{C_{g_i}}{N_{g_i}})} - 1 \leq \tau \tag{1}$$

where $\max\limits_{g_i \in G}$ ($\min\limits_{g_i \in G}$) is the group with the largest (smallest) cardinality, respectively.

As discussed earlier in Section II-A, fairness in EM can be defined as single and pairwise. Since ensuring pairwise

[1] We use the division-based notion of disparity to denote difference in this paper.

fairness for a given dataset is trivial, we will **focus on the case of single fairness in the rest of this paper**. The simplicity of the pairwise fairness case arises from the disjoint nature of pairs within each group $\langle g_x, g_y \rangle_{\forall g_x, g_y \in G}$, meaning that selecting a pair from one group has no impact on the counts for another group. Hence, a straightforward greedy approach could provide an exact solution in $\mathcal{O}(n \log n)$ time where n is the number of pairs: it first groups pairs by $\langle g_x, g_y \rangle$ and iteratively picks k pairs with the highest similarity score from each group adhering to their original distribution in the output set. Meanwhile, single fairness does not conform to this scenario. Picking one pair may impact the counts for multiple groups, adding complexity to the problem and disqualifying the above straightforward solution. Thus, we will focus on providing efficient solutions to the single definition in the next step. For ease of describing our solutions, we limit the number of sensitive attributes to one for each entity in the discussion. We will also show that we can extend the algorithm to support scenarios with multiple sensitive attributes in Section VI.

B. Formal Definition

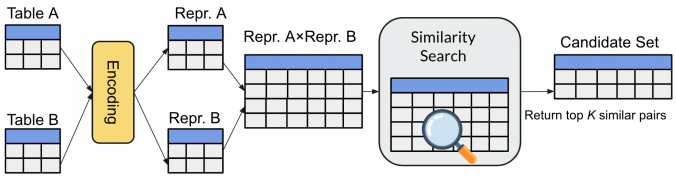


Fig. 2: A Typical Workflow of Blocking Stage in an End-to-end Entity Matching Pipeline

Next, we formally introduce the problem of preparing fairness-aware datasets in entity matching. This step in EM pipelines strives to generate a candidate set of high-quality pairs, potentially constituting a match. Based on this consideration, we apply the fairness constraints to the blocking stage, which is the origin of training data for a matcher. To reach this goal, a research challenge is that as the ground truth for matching is unknown, we cannot offer any guarantees regarding equality of opportunity. Nevertheless, we can consider high-quality pairs (i.e. with high similarity scores) as a proxy to potential matching pairs and formulate a candidate set that fulfills an equal representation ratio among the groups. This also aligns with many previous studies [35], [36] in this field.

Based on the practice of some recent studies [1], [37], [4], [30], the blocking stage can be conducted in steps shown in Figure 2. It starts from the two input tables of entity entries and encodes each entry from the two tables into a representation, e.g., a bag of words/embedding vector. Then, it performs similarity join over the two collections of entry representations and selects pairs with high similarity as the output. For example, in the deep learning-based methods [4], [37], the representation of an entity entry is a dense embedding vector, and the metric for similarity join will be the cosine similarity between embedding vectors. Finally, a top-k similarity join is conducted to find the results for blocking.

Based on the above discussion, we can then develop a formal problem definition by integrating the goal of fairness illustrated in Section III-A. Suppose $\mathcal{X}$ and $\mathcal{Y}$ are two tables of entity entries, a candidate pair can be described with a quintuple $p = \langle e_x, e_y, g_x, g_y, s \rangle$, where $e_x(e_y)$ is the entry from table $\mathcal{X}$ ($\mathcal{Y}$); $g_x(g_y)$ is the value of its demographic groups; s is the similarity between the representation of two entries. The search space could be identified as the cartesian product of $\mathcal{X}$ and $\mathcal{Y}$, and the objective regarding effectiveness could be considered as maximizing the sum of similarity scores of pairs in the result following that of the original blocking problem in Definition 1. Then, the problem is formally defined as Definition 5.

Definition 5: Given two tables of entity entries $\mathcal{X}$ and $\mathcal{Y}$, a threshold τ and a value k, the problem of preparing fairness-aware datasets for EM aims at finding a set R with k pairs $p \in \mathcal{X} \times \mathcal{Y}$ s.t. $\sum_{i=0}^{k} p_i.s$ is maximized and $\Delta(R) \leq \tau$ holds where Δ is defined in Equation 1.

Table X	
ID	G
x	A
w	A
y	B

Table Y	
ID	G
y'	B
x'	A
z'	A

Table X×Y sorted by similarity s					
Index	ID_X	G_X	ID_Y	G_Y	s
1	x	A	x'	A	0.95
2	y	B	y'	B	0.90
3	w	A	x'	A	0.75
4	x	A	z'	A	0.70
5	w	A	z'	A	0.65
6	y	B	z'	B	0.60
7	x	A	y'	B	0.55
8	w	A	y'	B	0.45
9	y	B	x'	A	0.40

Fig. 3: A toy example

Example 3: Consider an example illustrated in Figure 3 where $\mathcal{X}$ and $\mathcal{Y}$ are two tables of entries. For ease of presentation, we only include an ID and a sensitive attribute among $G = \{\text{A}, \text{B}\}$ for each entry, and $\mathcal{X} \times \mathcal{Y}$, i.e., the cartesian product of $\mathcal{X}$ and $\mathcal{Y}$ will be the whole search space. Assuming that $k = 5$, a fairness agnostic approach will choose five pairs with the highest similarity from $\mathcal{X} \times \mathcal{Y}$. In such solution $C_A = 8$ and $C_B = 2$, while in the search space $N_A = 11$ and $N_B = 7$. Then the disparity is $\Delta = \frac{8/11}{2/7} - 1 \simeq 1.57$. If we set the threshold τ as 0.1, this solution will not satisfy the fairness condition, and a valid result in this case could be $\{1, 2, 3, 4, 6\}$ where the disparity fulfills $\Delta = \frac{4/7}{6/11} - 1 \simeq 0.05 \leq 0.1$.

C. Integration with EM Pipelines

Firstly, we would like to highlight that our problem definition is orthogonal to the choice of blocking methods. To adopt existing blocking methods in our problem, users just need to provide the methods for entity entry encoding (the yellow box in Figure 2) and the corresponding similarity metrics for top-k search. We will later show in the experiments that our proposed techniques work well for three popular blocking methods: OverlapBlocker [38], DeepBlocker [37] and Sudowoodo [4]. Note that in many real applications, two

entity entries are matched does not necessarily mean that they have the same sensitive attribute values, and vice versa. Thus, we cannot only assume that matched pairs must have the same sensitive attribute values.

Besides, while this approach does not ensure a completely fair end-to-end EM pipeline, we argue that the model constructed using this data is fairness-aware and is anticipated to outperform fairness-agnostic pipelines. Specifically, we believe that it is trivial to ensure the fairness of the training data based on the outcome of our proposed approaches. We will show a case study of applying our techniques in an end-to-end EM pipeline in Section VI-B later. Lastly, the fairness could also be potentially improved in the matching stage. This can be realized by adding an additional item considering fairness to the loss function in the training process, which is orthogonal to the proposed solution in this paper. Due to space limitation, we will leave it as future work.

IV. Solution

Before proposing the solutions, we first make a detailed analysis of the problem raised in Definition 5. This problem aims at finding a candidate set with k pairs, and the search space is all possible pairs from the set of $\mathcal{X} \times \mathcal{Y}$. Therefore, a naive solution would be enumerating all the combinations with cardinality k and keeping those sets with the disparity $\Delta(\cdot)$ no larger than τ, where $\Delta(\cdot)$ is the function to calculate the disparity of a set of pairs based on Equation 1. Then the set with the maximum sum of similarity score will be the results. Although this exact algorithm can be implemented by backtracking and optimized with branch and bound techniques, the overhead is too heavy due to the exponential search space: assume the cardinality of $\mathcal{X} \times \mathcal{Y}$ is n, the search space will be $\mathcal{O}(2^n)$. In practice, the exact algorithm would take days to finish even for $k = 100$, where k would be several thousand in practical applications. Thus, we aim to develop approximate solutions that are efficient but still keep a decent level of recall in the rest of this section.

A. Greedy Algorithm

We first came up with a greedy algorithm as the solution. The high-level idea is to try to keep the pairs with the highest similarity scores in a greedy manner. At the same time, we can identify the demographic groups with maximum and minimum representation ratios (denoted as max and min groups, respectively) that decide the disparity score. Then, if the candidate set $\mathcal{D}$ does not satisfy the fairness constraint $\Delta(\mathcal{D}) \leq \tau$, we will make adjustments with the pairs that are not selected yet (i) with the highest similarity, and (ii) can help reduce the disparity score. The algorithm will terminate once a candidate set that satisfies the fairness constraints is found.

Algorithm 1 illustrates the process of the greedy solution. It first initializes the result set using k pairs with the highest similarity (line: 2 and 3). Once the result set satisfies the fairness constraint, the algorithm terminates (line: 6). Otherwise, it will replace the pair from the max group with the lowest similarity (line: 7). The new pair comes from the pairs with the highest similarity that are not currently selected and do not belong to the max group so as to reduce the disparity score (line: 8). After the candidate set is updated, the visited pairs will be marked, and the statistic information like the disparity score and the representation ratio of each demographic group is also updated (line: 9).

Algorithm 1: Greedy Algorithm

Input: $\mathcal{D}$: all possible pairs $\mathcal{X} \times \mathcal{Y}$, k: The cardinality of results; τ: the fairness constraint
Output: $\mathcal{H}$: The candidate set

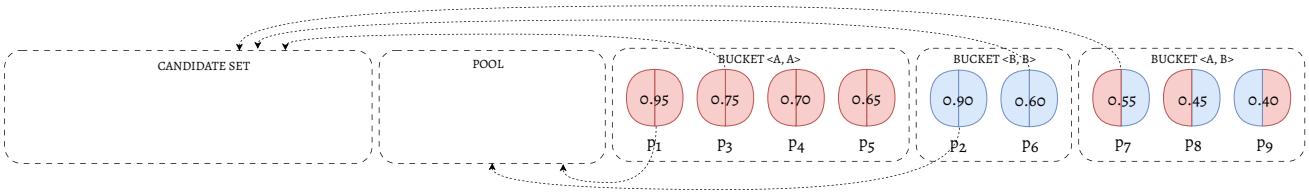
```
1  begin
2      Sort all pairs in D in the descending order of
         similarity score;
3      Intialize H with k pairs with highest similarity
         score in D;
4      while ∃ pair p ∈ D not visited do
5          if Δ(H) ≤ τ then
6              Return H;
7          p ← pair from D − H not visited and not
             belong to max group;
8          p' ← pair from H belong to max group with
             minimum similarity score;
9          Replace p' with p in H, mark p as visited,
             update statistic information;
10     end
11     Return H;
12 end
```

Complexity Analysis Suppose there are n pairs in $\mathcal{X} \times \mathcal{Y}$, then the time complexity of sorting process is $\mathcal{O}(n \log n)$; the process of traversing all pairs in $\mathcal{D}$ is $\mathcal{O}(n \log k)$: the outer loop repeats n times while the inner loop of adjusting $\mathcal{H}$ can be finished in $\mathcal{O}(\log k)$ time using a heap. Since $n >> k$ always holds, the overall time complexity will be $\mathcal{O}(n \log n)$ and the space complexity is $\mathcal{O}(n)$.

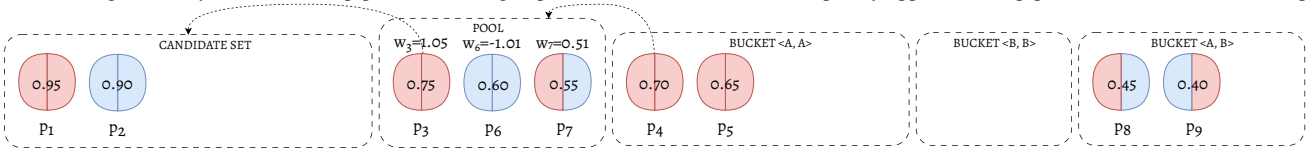
Example 4: Consider the example shown in Figure 3 again with $k = 5$ and $\tau = 0.1$. In the first step, it initializes the candidate set with the top 5 pairs from the sorted $\mathcal{X} \times \mathcal{Y}$. In the current candidate set $C_A = 8$ and $C_B = 2$, while in the search space $N_A = 11$ and $N_B = 7$. The disparity is $\frac{8/11}{2/7} - 1 \simeq 1.57$ and does not satisfy the fairness condition. Next, it replaces pair 5 since it is the lowest similarity score in the candidate set belonging to the max group (A), with 6 as the pair with the highest similarity score but not belonging to the max group that has not been previously selected. The candidate set now is $\{1, 2, 3, 4, 6\}$ and the disparity becomes $\Delta = \frac{4/7}{6/11} - 1 \simeq 0.05 < 0.1$. Thus, the algorithm stops and returns the candidate set.

B. Weight-based Strategy

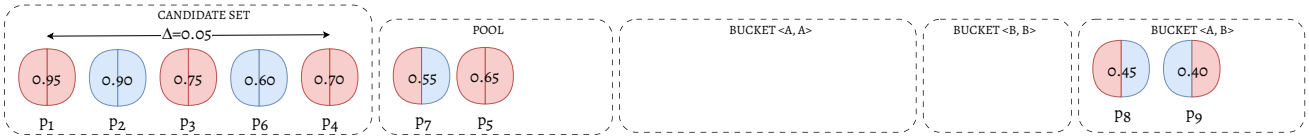
Although the greedy solution is straightforward and can always keep pairs with high similarity, it would be inefficient if pairs with high similarity scores are skewed in certain groups. In this case, the greedy algorithm will take a much longer



(a) Weight initially includes the top pair from each group in the candidate set and subsequently appends the top pairs from each bucket to the pool.



(b) Weight computes the weight of every pair in the pool, and the pair with the greatest weight is included in the candidate set. This procedure is reiterated until the candidate set contains a total of 5 pairs.



(c) The disparity score of the new candidate set is below the specified threshold, therefore Weight stops and returns the candidate set.

Fig. 4: Running Weight on the toy example.

time to converge. The reason is that the greedy algorithm selects pairs in the descending order of similarity while not giving enough attention to fairness. To address this issue, we devised a weight-based solution to consider both similarity and disparity scores. The basic idea is that in the process of selecting pairs to add to the candidate set, we calculate the weight of each pair based on the current results. Since the weight is decided by both the similarity score and disparity, we group all pairs by the pair of sensitive attribute values and construct a set of buckets. For each bucket, we sort the pairs in descending order of similarity score. In each round, we consider the set of top-ranked pairs from each bucket. For each pair, its weight is calculated based on both the similarity score and the contribution to fairness. The contribution to fairness can be decided as the disparity score before adding a pair into the candidate set deduces that after adding it. Assume the candidate set before and after adding a pair p is H_p and H'_p, respectively, then the weight from the fairness aspect can be calculated as $\Delta(H_p) - \Delta(H'_p)$. The overall weight of pair p i.e. $C(p)$ could be defined as the sum of the similarity score $p.s$ and the fairness contribution, i.e. $p.s + \Delta(H_p) - \Delta(H'_p)$. Lastly, the pair with the largest weight will be selected to add to the candidate set.

Algorithm 2 shows the process of the weight-based solution. The algorithm consists of two steps. In the first step it aims at filling up the candidate set with k pairs with minimum weight. It first groups all pairs by the pair of sensitive attributes they have and constructs the buckets (line: 2). Then, it initializes the candidate set and the temporary set for selected pairs (line: 3). For each pair in P, it will calculate the weight, and the pair with maximum weight p_0 will be selected (line: 7). Then the next pair from the bucket that p_0 belongs to will be added to P (line: 8). After the candidate set has k pairs, it checks whether the fairness constraint is satisfied. If not, in the second step, it will make adjustments to the results by replacing pairs from the max group with those from groups with lower representation ratio so as to reduce the disparity score (line: 11 to 13).

Algorithm 2: Weight-based Algorithm

Input: $\mathcal{D}$: all possible pairs $\mathcal{X} \times \mathcal{Y}$, k: The cardinality of results; τ: the fairness constraint
Output: $\mathcal{H}$: The candidate set

1 **begin**
2 Group all pairs by the sensitive attribute pairs $\langle g_x, g_y \rangle$, construct buckets $\mathcal{B}$ and sort;
3 Initialize $\mathcal{H}$ as $\emptyset$;
4 $P \leftarrow$ The set of top ranked pairs from all buckets;
5 **while** $|\mathcal{H}| < k$ **do**
6 $P \leftarrow$ the top-ranked pairs from all buckets;
7 Calculate the weight of all pairs in P, move the one p_0 with the maximum weight from P to $\mathcal{H}$;
8 Add the top-ranked pair from bucket B_0 where p_0 belongs to into P;
9 **end**
10 **while** $\Delta(\mathcal{H}) > \tau$ **do**
11 Identify a bucket B_0 associated with the lowest representation ratio of sensitive attribute value;
12 Replace the pair from the max group with the top-ranked pair from B_0;
13 Update the statistics of all groups;
14 **end**
15 Return $\mathcal{H}$;
16 **end**

skip **Complexity Analysis** The cost to initialize the buckets is $\mathcal{O}(n)$. Then, in order to fill k pairs into $\mathcal{H}$, it requires $\mathcal{O}(G * k \log k)$ time with the help of a priority queue, where G is the number of groups. The post-processing to adjust the disparity score to the threshold needs $\mathcal{O}(n)$ time. Then the total time complexity is $\mathcal{O}(n)$ as both parameters k and G are much smaller than n. And the space complexity is $\mathcal{O}(n * G)$.

Example 5: The weight-based algorithm starts by initializing the candidate set with the top pair from each group A and B: $\{1, 2\}$. In the next step, as shown in Figure 4a, the algorithm continues by grouping all the remaining pairs by the sensitive attribute pairs: $\langle A - A \rangle = \{3, 4, 5\}$, $\langle B - B \rangle = \{6\}$ and $\langle A - B \rangle = \{7, 8, 9\}$. Next, the top-ranked pair from each bucket is added to a pool $P = \{3, 6, 7\}$ and the associated weight is calculated. For pair 3, the weight is $0.75 + \left(\frac{2/7}{2/11} - 1\right) - \left(\frac{4/11}{2/7} - 1\right) \simeq 1.05$. Similarly, we can calculate the weight of pairs 6 and 7 as -1.01 and 0.51, respectively. Since pair 3 has the largest weight, it will be moved to the candidate set as $\{1, 2, 3\}$. And we add the top pair from $\langle A - A \rangle$ bucket, i.e. pair 4 into P (Figure 4b). Due to the space limitation, we omit the process of adding other pairs into $\mathcal{H}$. Please note that although pair 7 is included in the pool initially, it is never selected to be added to the candidate set because its weight never surpasses that of the other pairs in the pool. This iterative process continues until $k = 5$ pairs are added to the result set: $\{1, 2, 3, 6, 4\}$. Next, the disparity is calculated as $\Delta = \frac{4/7}{6/11} - 1 \simeq 0.05$. Since Δ is already smaller than $\tau = 0.1$, the algorithm terminates (Figure 4c).

In some application scenarios, the distribution of positive pairs is relatively even among different groups. In this case, we can simplify the cost-based algorithm by prioritizing the fairness condition. This is realized by constructing sorted buckets for all pairs based on the single sensitive attribute value instead of pairs. Then, in the initialization step, we select the pairs in a round-robin manner among all buckets while not considering the difference of similarity between pairs from different buckets. In this way, the disparity score can be minimized. Since the idea of this solution is similar to the stratified uniform sampling, we name as Stratified.

Algorithm 3 illustrates the process of the stratified solution. It first constructed the buckets $\mathcal{B}$ by grouping all pairs by the sensitive attribute. Within each bucket, it sorts pairs in descending order of similarity score (line: 2). Note that one pair might be allocated to two buckets in this situation; we can avoid selecting duplicated pairs in the result by hashing mechanisms. Then, we add the top-ranked pair of each bucket in a round-robin manner until the cardinality of the candidate set reaches k (line: 5). In this process, the disparity of the candidate set is minimized. After that, if the fairness condition is not satisfied, the adjustment is made in the same way as that in Algorithm 2. Finally, the algorithm terminated when the fairness condition is satisfied. (line: 10).

skip **Complexity Analysis** The cost to initialize the buckets is $\mathcal{O}(n)$. And the time to fill k results into $\mathcal{H}$ is $\mathcal{O}(n)$. The post-processing to reduce the disparity score to the threshold needs $\mathcal{O}(n)$ time; thus, the overall time complexity is $\mathcal{O}(n)$. The space complexity is $\mathcal{O}(n * G)$. We can see that the time complexity of Algorithm 3 is the same as of Algorithm 2, and the savings come from the process of filling $\mathcal{H}$ with k results.

Algorithm 3: Stratified Algorithm

Input: $\mathcal{D}$: all possible pairs $\mathcal{X} \times \mathcal{Y}$, k: The cardinality of results; τ: the fairness constraint
Output: $\mathcal{H}$: The candidate set

```
1  begin
2      Group all pairs by the sensitive attribute values,
         construct buckets B;
3      Initialize H as ∅;
4      while |H| ≤ k do
5          For each non-empty bucket B, move the top
             ranked pair in B into H;
6      end
7      while Δ(H) > τ do
8          Adjust H with the method from line 11 to 13
             in Algorithm 2;
9      end
10     Return H;
11 end
```

Example 6: Let us revisit the example in Figure 3 with $k = 4$ and $\tau = 0.1$. The stratified algorithm initially groups all pairs into two sorted buckets: $\langle A \rangle = \{1, 3, 4, 5, 7, 8, 9\}$ and $\langle B \rangle = \{2, 6, 7, 8, 9\}$. Next, the algorithm alternately selects the top pair from each bucket and continues until four pairs are added to the candidate set $\{1, 2, 3, 6\}$. Then the disparity is calculated over the candidate set as $\Delta = \frac{4/7}{4/11} - 1 \simeq 0.57$, which exceeds $\tau = 0.1$. Therefore, it needs to make adjustments by swapping out pairs from group B. Next, the algorithm chooses to replace pair 6 with pair 7 instead of 4 and 5, since they will increase Δ ($\Delta = \frac{6/11}{2/7} - 1 \simeq 0.9$). After that we have $\Delta = \frac{5/11}{3/7} - 1 \simeq 0.06 < 0.1$. Hence, the algorithm stops.

V. Experimental Results

A. Experiment Setup

Datasets We evaluate our proposed techniques on four popular datasets for EM. The tasks DBLP-ACM (DA), DBLP-Scholar (DS) and Walmart-Amazon (WA) were initially created in [39] and has been widely evaluated in previous studies about entity matching [15], [8], [37]. Since these datasets are not designed for the purpose of evaluating fairness in EM, there is no pre-defined sensitive attribute. We follow the previous study [14] to choose the sensitive attributes. The MusicBrainz (Music) dataset was from [40]. To include sensitive attributes so as to evaluate fairness, we augmented entities in the original datasets by joining with the raw datasets with the official APIs [2]. More detailed statistics of the above datasets can be found in Table I.

Evaluation Metrics For the effectiveness of the blocking step, we follow the previous studies [37], [4] to use recall

[2]https://musicbrainz.org/doc/MusicBrainz_Entity

and cardinality of the candidate set as the evaluation metric. Since we employ top-k search for all blocking methods, the cardinality of the candidate set is equal for all methods, i.e., the selected value of k. Thus, we only compare the recall of different methods. For evaluating fairness, we report the disparity score Δ of the obtained dataset; a lower value means better performance. We report the execution time to show the efficiency of each method.

Compared methods Since there is no previous study on the problem studied in this paper, we just report the performance of the three proposed methods Greedy (Algorithm 1), Weight (Algorithm 2) and Stratified (Algorithm 3). To illustrate that our proposed methods are orthogonal to the blocking methods, we evaluate each proposed method with three blockers: the rule-based one OverlapBlocker(OB) [38] [3], the neural-based ones DeepBlocker (DB) [37] and Sudowoodo (SW) [4]. The similarity score of OB is the overlap of tokens while that for DB and SW is cosine similarity between embedding vectors. Due to the space limitation, we will leave the integration with other blocking methods as future work.

Environment We implemented all algorithms with Python. The experiments were run on a server with 1 AMD EPYC 7R32 48-core processor and 192GB RAM. We ran all experiments 5 times and reported the average performance. For the execution time, we omitted the part of generating the representation of entity entries and only measured the time for identifying the top-k results.

TABLE I: The statistics of datasets. $|\mathcal{X}|$ and $|\mathcal{Y}|$ means the cardinality of two tables, respectively. # Group is the number of demographic groups. # Positive is the number of positive pairs in the ground truth.

Dataset	Sensitive Attribute	# Group	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{Y}\rvert$	# Positive
DA	Venue	5	2615	2296	2221
DS	Venue	5	2615	64264	1906
WA	Manufacturer	22	2555	22075	267
Music	Country	12	3700	2473	2010

B. Main Results

First, we report the main results in Table II. We have the following observations. First of all, both Greedy and Weight methods can significantly alleviate the unfairness in the candidate set while ensuring a high recall level. Among all results, the most significant drop of the recall happens on the WA dataset for OB where the recall of Weight is 0.948 while that of the fair-agnostic method is 0.965. This illustrates that it is feasible to find a fair candidate set without sacrificing effectiveness too much. At the same time, the performance of Stratified is not so stable since it prioritized minimizing the fairness of the candidate set. For example, on the music dataset with OB, the fair-agnostic method has a recall of 0.96 while the recall of Stratified drops to 0.75. We can also have similar observations on the WA dataset. In addition, from the aspect of efficiency, we can see that Greedy has the best performance for OB. The reason is that the similarity value of overlap is discrete, and thus it can terminate very fast once a candidate set satisfying the fair condition is obtained. However, its performance is unstable and always worsens drastically for neural-based blockers with real similarity scores. Stratified is always more efficient than Weight since it does not need to calculate the composite score based on both fairness and similarity in each step. However, as illustrated above, the recall of Stratified could be much lower in some cases.

Note that there are two hyper-parameters: the cardinality of the candidate set k and the threshold of disparity score τ. We choose them empirically for different datasets based on the following principles: For the value of k, the value is selected based on the following consideration: Generally speaking, the value of k is closely related to the choice of blocking method. For less effective blocking method, the value of k needs to be larger so as to reach the similar recall level with a more effective method. So OB needs a much larger value of k to reach a similar level of recall with the two neural-based blockers. And SW can achieve a high recall with relatively small values of k on most datasets. We report the data points with the highest recall for the fair-agnostic method. For the value of τ, we choose a larger value for the dataset with more obvious unfairness and try not to harm the recall at the same time. Detailed choice of k and τ for results in Table II is summarized in Table III.

C. Parameter Sensitivity Analysis

Next, we conducted an in-depth parameter sensitivity analysis for the key parameters k and τ. We will report the results from two aspects: effectiveness (disparity score and recall) and efficiency (execution time).

We first look at the influence over the disparity Δ and recall. Here, the effect of τ over the performance is very intuitive: smaller τ will add a more strict constraint on fairness; thus, the value of Δ tends to be close to τ and the recall tends to be lower. Thus, we only show the results of varying k. Figure 5 to Figure 7 show the results of blocker OB, DB and SW, respectively. We have the following observations:

First of all, the recall tends to be higher when k values become larger for all blockers, where the trend is more obvious for OB and DB. Specifically, the OB blocker needs a much larger number of k to achieve a similar level of recall with DB and SW. The reason could be that the syntactic similarity over a bag of tokens is less powerful than learned embeddings to capture semantic information. Thus, it requires more candidates to include enough positive pairs.

In addition, for the same kind of blocking method, the recall of Weight is always the best, while Stratified is always the worst. Such an observation is consistent with the analysis of our proposed methods. The reason is that Stratified assumes an even distribution to avoid the computation of weight in Weight. However, the distribution of sensitive groups is not even in the search space in most cases. Moreover, the results of the final disparity score remain constant for each method when varying k. This is due to the design of our

[3]https://github.com/anhaidgroup/py_entitymatching

TABLE II: Main Results. Fair-agnostic means the original top-k similarity join methods without considering the fair condition.

Task	Method	OB			DB			SW		
		Δ	Time (s)	Recall	Δ	Time (s)	Recall	Δ	Time (s)	Recall
DA	Fair-agnostic	0.642	8.598	0.951	0.272	1.25	0.962	0.192	0.387	0.991
	Greedy	0.2	10.98	0.949	0.198	1.318	0.962	0.098	4.362	0.991
	Weight	0.2	446.25	0.945	0.198	429.13	0.964	0.099	437.83	0.991
	Stratified	0.117	442.79	0.90	0.013	419.35	0.95	0.018	436.28	0.98
DS	Fair-agnostic	0.696	69.53	0.918	0.716	9.181	0.878	0.436	5.624	0.931
	Greedy	0.1	890.87	0.907	0.3	4380	0.814	0.1	295.82	0.931
	Weight	0.1	7106	0.91	0.3	1350	0.871	0.099	1463	0.931
	Stratified	0.002	2301	0.874	0.004	556.92	0.863	0.002	523.79	0.925
WA	Fair-agnostic	28.36	131.98	0.965	44.04	13.15	0.876	5.898	6.71	0.906
	Greedy	5.0	912.91	0.96	9.999	1735	0.866	1.0	788.75	0.905
	Weight	5.0	8042	0.948	9.95	502.59	0.872	0.999	1925	0.906
	Stratified	4.25	5509	0.646	0.961	491.88	0.611	0.998	1311	0.608
Music	Fair-agnostic	8.54	12.71	0.97	11.26	6.52	0.966	8.204	2.64	0.974
	Greedy	2.99	15.173	0.966	3.0	3313	0.96	0.194	1092	0.969
	Weight	2.99	492.13	0.969	2.99	493.51	0.955	0.2	485.23	0.973
	Stratified	0.076	456.46	0.75	0.004	438.96	0.956	0.014	439.62	0.96

TABLE III: Hyper-parameter settings for main results.

values (k/τ)	OB	DB	SW
DA	20,000 / 0.2	3,000 / 0.2	3,000 / 0.1
DS	150,000 / 0.1	20,000 / 0.3	12,000 / 0.1
WA	30,000 / 5.0	30,000 / 10.0	15,000 / 1.0
Music	30,000 / 3.0	15,000 / 3.0	6,000 / 0.2

proposed algorithms: the search process would terminate once the fairness condition is satisfied. As a result, the result of disparity score is determined by the pre-defined threshold.

Next, we also show the scalability of all methods. The results of varying k for all three blockers are shown in Figure 8 to Figure 10. We can see that, generally speaking, the proposed methods scale well with an increasing value of k. Specifically, the performance of Stratified and Weight remains almost the same w.r.t. varying k for the DB and SW methods. The reason could be that the search process converges much faster on these two blockers. For the Greedy method, the performance is not so stable and might increase sharply when k is larger than some values. The reason could be that unlike Stratified and Weight, the unvisited pairs are not grouped by the sensitive attribute. Thus it might take more time for Greedy to converge because the next pair to be checked might not help reduce the disparity score Δ.

Meanwhile, the execution time is also influenced by the distribution of each dataset. On the DA dataset, the performance of Greedy is generally better. The reason is that the distribution is more dense; thus, it is easier to include more similar pairs in a batch and accelerate the convergence of the search algorithm. The performance of Greedy could increase sharply at some values of k for DB and SW. For instance, the WA dataset is relatively sparse. As a result, it requires a larger value of k to achieve a decent recall; and for the same value of k the search process takes longer time to finish even with the same blocker. The lesson learned here is that there is not a one-size-fits-all method for all blockers. We should select the search algorithm wisely based on the requirement of real application so as to make a wise trade-off.

The results of varying τ are shown in Figure 11. Since the trend for all blockers is similar on all datasets, we just report the results of one blocker on each dataset due to space limitations. The general trend is that smaller τ would lead to more execution time, which is aligned with our anticipation.

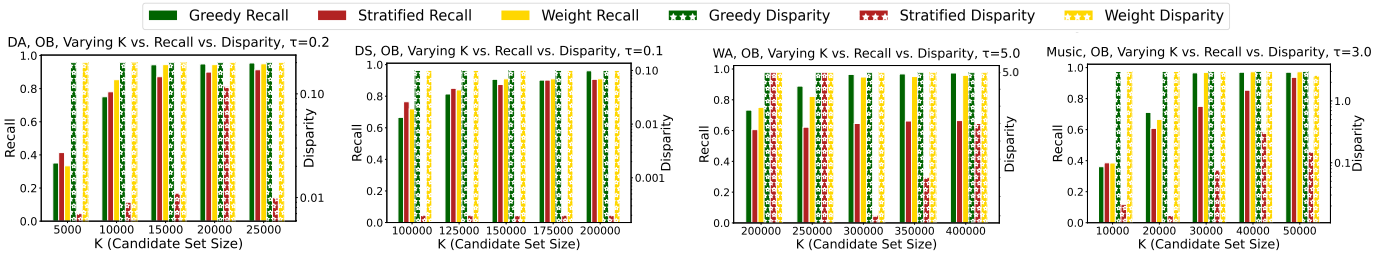
Finally, as indicated by our findings, the choice of hyper-parameters k and τ is specific to each dataset. While the value of k depends on the number of potential matches, the choice of τ is contingent upon factors such as the original distribution of the groups and the initial degree of unfairness, a.k.a. result of the fair-agnostic method. Consequently, these values should be decided by domain experts or through a hyper-parameter tuning process via grid search.

VI. Case Study

In this section, we conduct two case studies to show that our proposed techniques could be applied to data science practice with more general use cases. We first illustrate how to generalize our solution to scenarios where demographic groups are decided by multiple attributes in Section VI-A. Then we show an example of integrating our proposed techniques into end-to-end EM pipeline in Section VI-B.

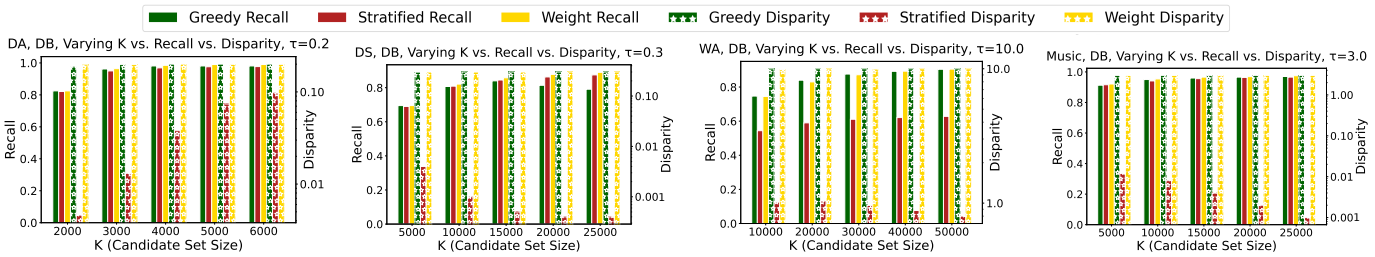
A. Supporting Multiple Sensitive Attributes

Firstly, we would like to show that our proposed techniques can be extended to the case where there is more than one sensitive attribute, which is an important features required by solutions for algorithmic fairness [**?**], [**?**], [**?**]. Recall the MusicBrainz dataset we augmented previously where the sensitive attribute is *country*, we extend the dataset with another sensitive attribute *gender* with the official APIs. In this case, the demographic groups can be defined by enumerating the combination of all sensitive attributes and regarding each combination as a group. Here we have 12 values of *country* and 2 values of *gender*. Thus there are 24 demographic groups in total, such as {RU, male} and {CA, female} etc. The disparity score and recall of the fair-agnostic method with Sudowoodo on this dataset is 11.81 and 0.97, respectively. After applying the Weight algorithm, the value of the disparity



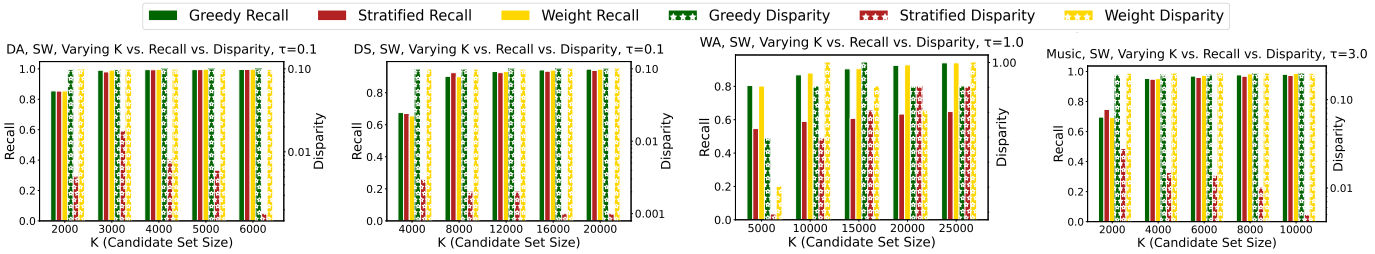
(a) DA (b) DS (c) WA (d) Music

Fig. 5: Effectiveness: Varying values of k (OB).



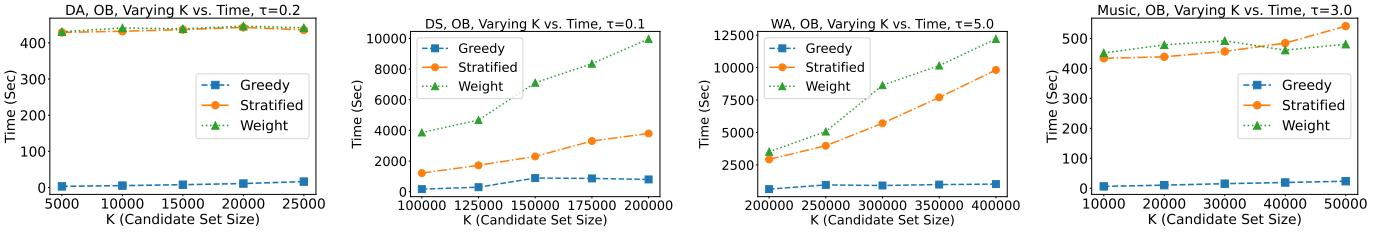
(a) DA (b) DS (c) WA (d) Music

Fig. 6: Effectiveness: Varying values of k (DB).



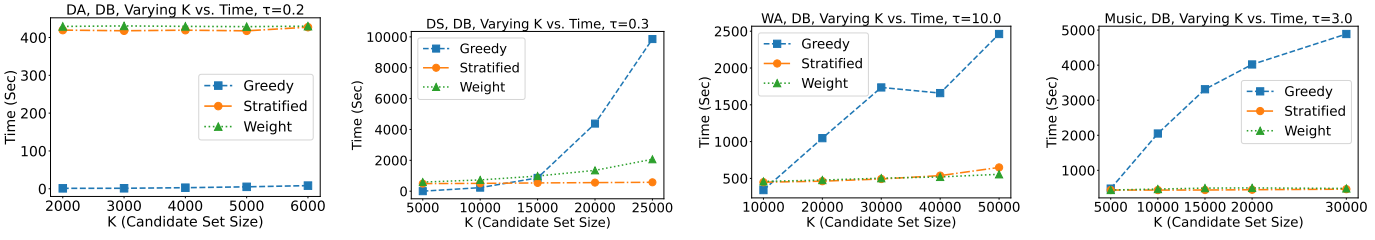
(a) DA (b) DS (c) WA (d) Music

Fig. 7: Effectiveness: Varying values of k (SW).



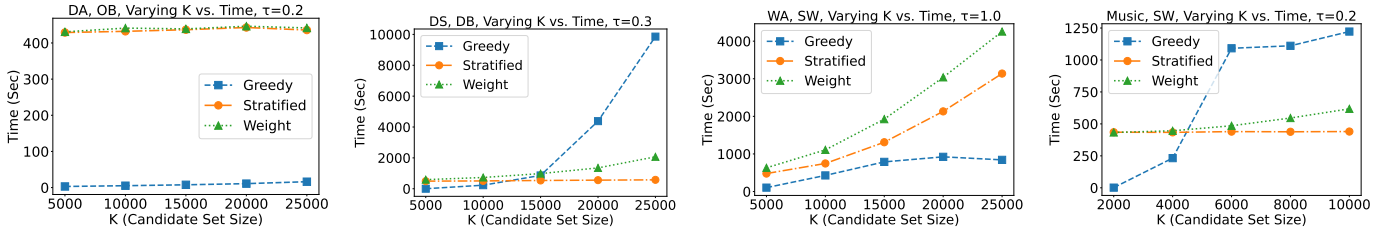
(a) DA (b) DS (c) WA (d) Music

Fig. 8: Efficiency: Varying values of k (OB)



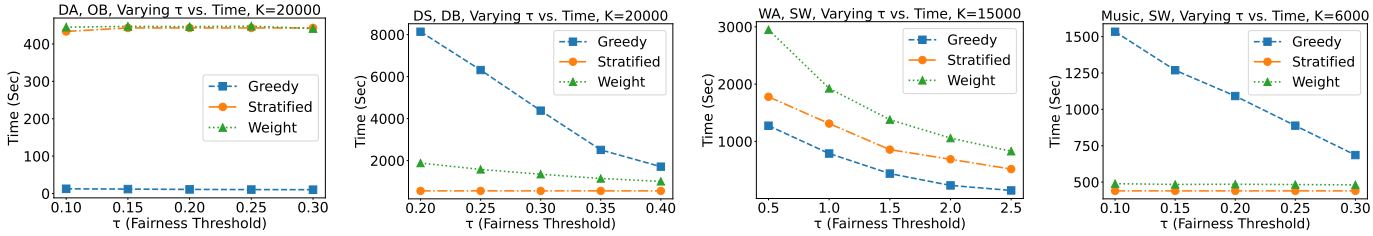
(a) DA (b) DS (c) WA (d) Music

Fig. 9: Efficiency: Varying values of k (DB)



(a) DA (b) DS (c) WA (d) Music

Fig. 10: Efficiency: Varying values of k (SW)



(a) DA-OB (b) DS-DB (c) WA-SW (d) Music-SW

Fig. 11: Efficiency: Varying values of τ

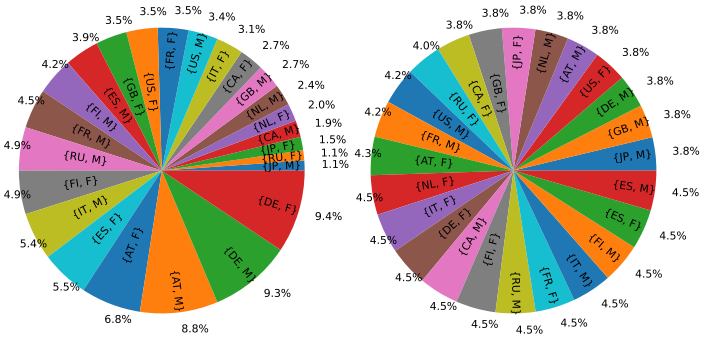


Fig. 12: The distribution of all demographic groups before (left) and after (right) applying our method.

score drops to 0.2 while the recall stays the same. From the detailed distribution of all demographic groups before and after applying our method in Figure 12, we can find that our proposed techniques did help make the distribution more even.

B. Integration into End-to-end Pipeline

We start with the choice of the EM task. Following the previous study [14], we choose COMPAS [41], a public dataset of criminal records that has been widely used in Fair ML research, as the source data. In addition to names and other information, the dataset contains sensitive attribute *race* with two values: Black and White. The EM task is to match between the two tables of passengers: a no-fly list with 5,032 entries and a passenger list with 75,459 entries.

We construct an end-to-end EM pipeline for this task as follows: In the blocking stage, we utilize DeepBlocker to learn entity representation and use our Weight algorithm to identify the candidate set. Here we select τ as 0.1 and k as 25,000. In the labeling stage, we randomly sampled 2,000 pairs from the candidate set and performed manual labeling. During this process, 538 positive pairs were found. To ensure the value of the disparity score is no larger than τ in the training data, we conduct stratified sampling based on the values of sensitive attributes from the pairs that do not belong to the candidate set to create negative instances. A similar idea of the Stratified algorithm can realize this. Finally, we construct a training set with 75,458 pairs (430 positive) and a test set with 20,121 pairs (128 positive). We set up the baseline method by replacing Weight with the fair-agnostic method in the above pipeline to construct another training data with the same number of positive pairs and overall cardinality. The test set is identical for both methods to make a fair comparison. In the matching stage, we train a Ditto [8] model as the matcher by randomly selecting 10% of the training data as the validation set.

TABLE IV: The fairness score of the matching stage for the COMPAS dataset. B and W denotes the group of Black and White people, respectively

Method	FDR		F_1	
	B	W	B	W
Fair-agnostic	0.31	0.22	0.72	0.8
Ours	**0.12**	**0.11**	**0.84**	**0.85**

The matching results of the two methods are shown as follows. The disparity score of the dataset created by our method and the fair-agnostic baseline is 0.05 and 0.42, respectively. We also evaluate the outcome of matching with the metrics of F_1 disparity [42] which focuses on equalizing the F_1 score among different groups, and *FDRP* (False Discovery Rate Parity) [14], [43] which enforces the independence of true non-matches from groups, among pairs predicted as true.

Then we look at the results of Ditto in the matching stage. The results of FDR and F_1 disparity are shown in Table IV. For the baseline method, the White group exhibits an *FDR* value of 0.22, while the Black population has a value of 0.31. However, these *FDR* values of our method are reduced to 0.11 for the White and 0.12 for the Black group. It also indicates a substantial decrease in the *FDRP*, from approximately 41% in the baseline method to around 9%. For the baseline method, the F_1 scores of the White and Black groups are 0.8 and 0.72, respectively. Meanwhile, the results for our method are 0.85 and 0.84, respectively. While the F_1 disparity has been around 8% between the two groups in the fair-agnostic baseline, that of our method drops to 1%. Moreover, our method can also help improve the overall F_1 score from 0.79 to 0.85. The reason could be that the training data generated by our method can include more information about entities from the Black group and thus noises are also reduced. This illustrates that our proposed techniques could help improve both the fairness and quality of matching in an end-to-end EM pipeline.

VII. Related Works

A. Entity Matching

Entity Matching (EM) is a vital data integration task that has been extensively studied over the past decades [9], [3], [38]. Many efforts have been made in previous studies for both the blocking and matching stages of EM. Earlier studies for blocking [36] are non-learning ones, such as rule-based [2], meta blocking [44], schema-agnostic blocking [45]. Recently deep learning techniques have been widely adopted to improve the blocking step. AutoBlock [46] learned the embedding of entities by leveraging nearest neighbor information. DeepBlocker [37] employed RNN variants to improve the quality of entity embedding. Sudowoodo [4] learned entity embedding with self-supervised learning. Sparkly [30] proposed an empirical study of distributed blocking algorithms.

Deep learning techniques, especially pre-trained language models, have also been applied to the matching stage, and the results are generally rather promising, as shown in [10], [11]. DeepMatcher [15] employs the Recurrent Neural Network (RNN) models to perform entity matching. Brunner et al. [7] first employed pre-trained LM for entity matching. Ditto [8] improved the performance by adopting data augmentation operations. Some recent studies [47], [48], [49] focused on the problem of generalized entity matching where the definition of EM is relaxed. Other studies focusing on different aspects of entity matching, such as privacy [50], efficiency [51], crowdsourcing [52], domain adaption [53] and explainability [54].

At the same time, there are very few studies about fairness in EM. Recently an empirical study [14] illustrated that unfairness exists in the outcome of many EM methods. Other related studies [22], [23] tried to propose new evaluation metrics for fairness in EM but did not improve it.

B. Algorithmic Fairness

Achieving algorithmic fairness has been a hot topic in many research fields, including data management. This line of studies focuses on ensuring the fairness of solutions in a variety of data management problems, such as query processing [24], feature engineering [55] and ranking [56], [57], while keeping the effectiveness. There are different categorizations of fairness definition, including individual [58], group [26], subgroup [59], [60] and casual [61] fairness. Fairness can be enforced at different stages of data analysis. Pre-processing methods include data massaging and sampling [17], and reweighting [18]. In-processing methods enforce fairness by constraining or regularizing the objective function of the algorithms [20], [19]. Post-processing methods manipulate the results of an algorithm to promote fairness among different groups [21], [62]. Many efforts [43], [63] have been paid to propose a unified framework for evaluating different fairness definitions and promoting fairness. These fairness audit toolkits enable testing models for several bias and fairness metrics concerning multiple population sub-groups.

Some studies [64], [65], [66] aimed to ensure a responsible preparation of the training data through approaches such as database repairing. There are also some studies regarding fairness in preparing datasets. For instance, iFlipper [62] reduces the unfairness in training data by flipping the labels based on the definition of individual fairness. DT [67] focuses on improving fairness when integrating multiple data sources. PFR [68] aimed at detecting the bias in the training data and fixing it in the loss function of the training process. They focused on completely different problems, and their definitions cannot be easily extended to support our problem.

Algorithmic fairness also attracts attention from other domains. Several classical problems have been investigated through the lens of fairness by the TCS community, such as approximate nearest neighbor search [69] and vertex cover [70]. Fairness has also received significant attention in the context of natural language processing [71], [72] and computer vision [73], [74]. Research in this area has revealed significant disparities in the performance across different demographic groups, highlighting the urgent need to address biases and promote fairness in these applications.

VIII. Conclusion

In this paper, we studied algorithmic fairness in entity matching and made the first attempt to provide solutions to the problem of preparing fairness-aware datasets for entity matching. We formally defined the problem as identifying positive pairs of entity entries while keeping a similar representative ratio among all demographic groups. Since the search space of this problem is rather large, we first came up with a greedy algorithm that prioritizes improving the effectiveness while satisfying the fairness constraint. We further developed a weight-based solution and its variant that took both effectiveness and fairness into consideration to improve the overall performance. Experimental results on several real-world datasets verified the efficiency and effectiveness of our proposed algorithms. Moreover, two case studies were conducted to show the potential capacity to utilize our methods in real-world applications.

REFERENCES

[1] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: Toward building entity matching management systems," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1197–1208, 2016.

[2] S. Das, P. S. G. C., A. Doan, J. F. Naughton, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, and Y. Park, "Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services," in *SIGMOD*. ACM, 2017, pp. 1431–1446.

[3] Y. Govind, P. Konda, P. S. G. C., P. Martinkus, P. Nagarajan, H. Li, A. Soundararajan, S. Mudgal, J. R. Ballard, H. Zhang, A. Ardalan, S. Das, D. Paulsen, A. S. Saini, E. Paulson, Y. Park, M. Carter, M. Sun, G. M. Fung, and A. Doan, "Entity matching meets data science: A progress report from the magellan project," in *SIGMOD*, 2019, pp. 389–403.

[4] R. Wang, Y. Li, and J. Wang, "Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation," in *ICDE*, 2023, pp. 1502–1515.

[5] P. Konda, S. S. Seshadri, E. Segarra, B. Hueth, and A. Doan, "Executing entity matching end to end: A case study," in *EDBT*, 2019, pp. 489–500.

[6] G. Papadakis, L. Tsekouras, E. Thanos, G. Giannakopoulos, T. Palpanas, and M. Koubarakis, "Domain- and structure-agnostic end-to-end entity resolution with jedai," *SIGMOD Rec.*, vol. 48, no. 4, pp. 30–36, 2019.

[7] U. Brunner and K. Stockinger, "Entity matching with transformer architectures - A step forward in data integration," in *EDBT*, 2020, pp. 463–473.

[8] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan, "Deep entity matching with pre-trained language models," *Proc. VLDB Endow.*, vol. 14, no. 1, pp. 50–60, 2020.

[9] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W. Tan, "Deep entity matching: Challenges and opportunities," *ACM J. Data Inf. Qual.*, vol. 13, no. 1, pp. 1:1–1:17, 2021.

[10] A. Zeakis, G. Papadakis, D. Skoutas, and M. Koubarakis, "Pre-trained embeddings for entity resolution: An experimental analysis," *Proc. VLDB Endow.*, vol. 16, no. 9, pp. 2225–2238, 2023.

[11] M. Paganelli, F. D. Buono, A. Baraldi, and F. Guerra, "Analyzing how BERT performs entity matching," *Proc. VLDB Endow.*, vol. 15, no. 8, pp. 1726–1738, 2022.

[12] N. Shahbazi, Y. Lin, A. Asudeh, and H. V. Jagadish, "A survey on techniques for identifying and resolving representation bias in data," *ACM Computing Surveys*, 2023.

[13] S. E. Whang, K. H. Tae, Y. Roh, and G. Heo, "Responsible AI challenges in end-to-end machine learning," *IEEE Data Eng. Bull.*, vol. 44, no. 1, pp. 79–91, 2021.

[14] 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, pp. 3279–3292, 2023.

[15] 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.

[16] M. Hort, Z. Chen, J. M. Zhang, F. Sarro, and M. Harman, "Bias mitigation for machine learning classifiers: A comprehensive survey," *CoRR*, vol. abs/2207.07068, 2022.

[17] F. Kamiran and T. Calders, "Data preprocessing techniques for classification without discrimination," *Knowl. Inf. Syst.*, vol. 33, no. 1, pp. 1–33, 2011.

[18] T. Calders, F. Kamiran, and M. Pechenizkiy, "Building classifiers with independency constraints," in *ICDM Workshops*, 2009, pp. 13–18.

[19] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *Data Mining Workshops (ICDMW)*, 2011, pp. 643–650.

[20] M. B. Zafar, I. Valera, M. Gomez-Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," in *AISTATS*, 2017, pp. 962–970.

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

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

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

[24] S. Shetiya, I. P. Swift, A. Asudeh, and G. Das, "Fairness-aware range queries for selecting unbiased data," in *ICDE*, 2022, pp. 1423–1436.

[25] L. E. Celis, V. Keswani, and N. K. Vishnoi, "Data preprocessing to mitigate bias: A maximum entropy based approach," in *ICML*, vol. 119, 2020, pp. 1349–1359.

[26] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. M. Wallach, "A reductions approach to fair classification," in *ICML*, vol. 80, 2018, pp. 60–69.

[27] S. Barocas, M. Hardt, and A. Narayanan, *Fairness and Machine Learning: Limitations and Opportunities*. MIT Press, 2023.

[28] J. Wu, Y. Zhang, J. Wang, C. Lin, Y. Fu, and C. Xing, "Scalable metric similarity join using mapreduce," in *ICDE*. IEEE, 2019, pp. 1662–1665.

[29] J. Wang, C. Lin, M. Li, and C. Zaniolo, "Boosting approximate dictionary-based entity extraction with synonyms," *Inf. Sci.*, vol. 530, pp. 1–21, 2020.

[30] D. Paulsen, Y. Govind, and A. Doan, "Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching," *Proc. VLDB Endow.*, vol. 16, no. 6, pp. 1507–1519, 2023.

[31] T. Jang, F. Zheng, and X. Wang, "Constructing a fair classifier with generated fair data," in *AAAI*, 2021, pp. 7908–7916.

[32] D. Xu, S. Yuan, L. Zhang, and X. Wu, "Fairgan: Fairness-aware generative adversarial networks," in *IEEE BigData*, 2018, pp. 570–575.

[33] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *ACM SIGKDD*, 2015, pp. 259–268.

[34] U. S. O. of the Federal Register, *Code of Federal Regulations*. US General Services Administration, National Archives and Records Service ..., 2005.

[35] G. Papadakis, M. Fisichella, F. Schoger, G. Mandilaras, N. Augsten, and W. Nejdl, "How to reduce the search space of entity resolution: with blocking or nearest neighbor search?" *CoRR*, vol. abs/2202.12521, 2022.

[36] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "Blocking and filtering techniques for entity resolution: A survey," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 31:1–31:42, 2021.

[37] 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," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2459–2472, 2021.

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

[39] H. Köpcke, A. Thor, and E. Rahm, "Evaluation of entity resolution approaches on real-world match problems," *Proc. VLDB Endow.*, vol. 3, no. 1, pp. 484–493, 2010.

[40] A. Saeedi, E. Peukert, and E. Rahm, "Comparative evaluation of distributed clustering schemes for multi-source entity resolution," in *ADBIS*, ser. Lecture Notes in Computer Science, vol. 10509, 2017, pp. 278–293.

[41] "Compas recidivism risk score data and analysis," https://www.propublica.org/datastore/dataset/compas-recidivism-risk-score-data-and-analysis, 2015.

[42] B. Ghai, M. Mishra, and K. Mueller, "Cascaded debiasing: Studying the cumulative effect of multiple fairness-enhancing interventions," in *CIKM*, 2022, pp. 3082–3091.

[43] H. Zhang, X. Chu, A. Asudeh, and S. B. Navathe, "Omnifair: A declarative system for model-agnostic group fairness in machine learning," in *SIGMOD*, 2021, pp. 2076–2088.

[44] G. Simonini, S. Bergamaschi, and H. V. Jagadish, "BLAST: a loosely schema-aware meta-blocking approach for entity resolution," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1173–1184, 2016.

[45] G. Simonini, G. Papadakis, T. Palpanas, and S. Bergamaschi, "Schema-agnostic progressive entity resolution," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 6, pp. 1208–1221, 2019.

[46] W. Zhang, H. Wei, B. Sisman, X. L. Dong, C. Faloutsos, and D. Page, "Autoblock: A hands-off blocking framework for entity matching," in *WSDM*, 2020, pp. 744–752.

[47] J. Wang, Y. Li, and W. Hirota, "Machamp: A generalized entity matching benchmark," in *CIKM*, 2021, pp. 4633–4642.

[48] J. Wang, Y. Li, W. Hirota, and E. Kandogan, "Machop: an end-to-end generalized entity matching framework," in *aiDM '22: Proceedings of the Fifth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, 2022, pp. 2:1–2:10.

[49] P. Wang, X. Zeng, L. Chen, F. Ye, Y. Mao, J. Zhu, and Y. Gao, "Promptem: Prompt-tuning for low-resource generalized entity matching," *Proc. VLDB Endow.*, vol. 16, no. 2, pp. 369–378, 2022.

[50] Y. Guo, L. Chen, Z. Zhou, B. Zheng, Z. Fang, Z. Zhang, Y. Mao, and Y. Gao, "Camper: An effective framework for privacy-aware deep entity resolution," in *KDD*. ACM, 2023, pp. 626–637.

[51] Y. Gao, X. Liu, J. Wu, T. Li, P. Wang, and L. Chen, "Clusterea: Scalable entity alignment with stochastic training and normalized mini-batch similarities," in *KDD*, 2022, pp. 421–431.

[52] C. Chai, G. Li, J. Li, D. Deng, and J. Feng, "Cost-effective crowdsourced entity resolution: A partial-order approach," in *SIGMOD*, 2016, pp. 969–984.

[53] J. Tu, J. Fan, N. Tang, P. Wang, C. Chai, G. Li, R. Fan, and X. Du, "Domain adaptation for deep entity resolution," in *SIGMOD*, 2022, pp. 443–457.

[54] J. Wang and Y. Li, "Minun: evaluating counterfactual explanations for entity matching," in *DEEM '22: Proceedings of the Sixth Workshop on Data Management for End-To-End Machine Learning*, 2022, pp. 7:1–7:11.

[55] R. Salazar, F. Neutatz, and Z. Abedjan, "Automated feature engineering for algorithmic fairness," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1694–1702, 2021.

[56] A. Asudeh, H. V. Jagadish, J. Stoyanovich, and G. Das, "Designing fair ranking schemes," in *SIGMOD*, 2019, pp. 1259–1276.

[57] M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates, "Fa*ir: A fair top-k ranking algorithm," in *CIKM*, 2017, pp. 1569–1578.

[58] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. S. Zemel, "Fairness through awareness," in *Innovations in Theoretical Computer Science*, S. Goldwasser, Ed., 2012, pp. 214–226.

[59] M. J. Kearns, S. Neel, A. Roth, and Z. S. Wu, "Preventing fairness gerrymandering: Auditing and learning for subgroup fairness," in *ICML*, vol. 80, 2018, pp. 2569–2577.

[60] J. R. Foulds, R. Islam, K. N. Keya, and S. Pan, "An intersectional definition of fairness," in *ICDE*, 2020, pp. 1918–1921.

[61] M. J. Kusner, J. R. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," in *NIPS*, 2017, pp. 4066–4076.

[62] H. Zhang, K. H. Tae, J. Park, X. Chu, and S. E. Whang, "iflipper: Label flipping for individual fairness," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 8:1–8:26, 2023.

[63] P. Saleiro, B. Kuester, A. Stevens, A. Anisfeld, L. Hinkson, J. London, and R. Ghani, "Aequitas: A bias and fairness audit toolkit," *CoRR*, vol. abs/1811.05577, 2018.

[64] J. Stoyanovich, S. Abiteboul, B. Howe, H. V. Jagadish, and S. Schelter, "Responsible data management," *Commun. ACM*, vol. 65, no. 6, pp. 64–74, 2022.

[65] A. Olteanu, C. Castillo, F. Diaz, and E. Kiciman, "Social data: Biases, methodological pitfalls, and ethical boundaries," *Frontiers Big Data*, vol. 2, p. 13, 2019.

[66] B. Salimi, B. Howe, and D. Suciu, "Data management for causal algorithmic fairness," *IEEE Data Eng. Bull.*, vol. 42, no. 3, pp. 24–35, 2019.

[67] F. Nargesian, A. Asudeh, and H. V. Jagadish, "Tailoring data source distributions for fairness-aware data integration," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2519–2532, 2021.

[68] P. Lahoti, K. P. Gummadi, and G. Weikum, "Operationalizing individual fairness with pairwise fair representations," *Proc. VLDB Endow.*, vol. 13, no. 4, pp. 506–518, 2019.

[69] M. Aumüller, S. Har-Peled, S. Mahabadi, R. Pagh, and F. Silvestri, "Fair near neighbor search via sampling," *SIGMOD Rec.*, vol. 50, no. 1, pp. 42–49, 2021.

[70] J. Blum, Y. Disser, A. E. Feldmann, S. Gupta, and A. Zych-Pawlewicz, "On sparse hitting sets: From fair vertex cover to highway dimension," in *IPEC*, vol. 249, 2022, pp. 5:1–5:23.

[71] J. E. Font and M. R. Costa-jussà, "Equalizing gender biases in neural machine translation with word embeddings techniques," *CoRR*, vol. abs/1901.03116, 2019.

[72] M. Díaz, I. Johnson, A. Lazar, A. M. Piper, and D. Gergle, "Addressing age-related bias in sentiment analysis," in *IJCAI*, 2019, pp. 6146–6150.

[73] Z. Wang, K. Qinami, I. C. Karakozis, K. Genova, P. Nair, K. Hata, and O. Russakovsky, "Towards fairness in visual recognition: Effective strategies for bias mitigation," in *CVPR*, 2020, pp. 8916–8925.

[74] J. Xu, Y. Xiao, W. H. Wang, Y. Ning, E. A. Shenkman, J. Bian, and F. Wang, "Algorithmic fairness in computational medicine," *EBioMedicine*, vol. 84, 2022.