TREATS: Fairness-aware entity resolution over streaming data

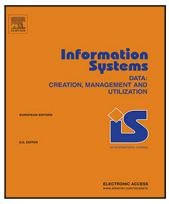
Tiago Brasileiro Araújo, Vasilis Efthymiou, Vassilis Christophides, Evaggelia Pitoura, Kostas Stefanidis

Information Systems 129 (2025) 102506

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/is



TREATS: Fairness-aware entity resolution over streaming data

Tiago Brasileiro Araújo [a,b,*], Vasilis Efthymiou [c,d], Vassilis Christophides [e], Evaggelia Pitoura [f], Kostas Stefanidis [a]

[a] Tampere University, Tampere, Finland
[b] Federal Institute of Paraíba, Soledade, Brazil
[c] Harokopio University of Athens, Athens, Greece
[d] FORTH-ICS, Heraklion, Greece
[e] ENSEA, ETIS, Paris, France
[f] University of Ioannina, Ioannina, Greece

ARTICLE INFO

Keywords:
Entity resolution
Streaming data
Fairness
Incremental processing
Distributed processing
Machine learning

ABSTRACT

Currently, the growing proliferation of information systems generates large volumes of data continuously, stemming from a variety of sources such as web platforms, social networks, and multiple devices. These data, often lacking a defined schema, require an initial process of consolidation and cleansing before analysis and knowledge extraction can occur. In this context, Entity Resolution (ER) plays a crucial role, facilitating the integration of knowledge bases and identifying similarities among entities from different sources. However, the traditional ER process is computationally expensive, and becomes more complicated in the streaming context where the data arrive continuously. Moreover, there is a lack of studies involving fairness and ER, which is related to the absence of discrimination or bias. In this sense, fairness criteria aim to mitigate the implications of data bias in ER systems, which requires more than just optimizing accuracy, as traditionally done. Considering this context, this work presents TREATS, a schema-agnostic and fairness-aware ER workflow developed for managing streaming data incrementally. The proposed fairness-aware ER framework tackles constraints across various groups of interest, presenting a resilient and equitable solution to the related challenges. Through experimental evaluation, the proposed techniques and heuristics are compared against state-of-the-art approaches over five real-world data source pairs, in which the results demonstrated significant improvements in terms of fairness, without degradation of effectiveness and efficiency measures in the streaming environment. In summary, our contributions aim to propel the ER field forward by providing a workflow that addresses both technical challenges and ethical concerns.

1. Introduction

The extensive volume of records collected from diverse sources necessitates an initial process of consolidation and cleansing before embarking on any data analysis endeavors [1,2]. In this sense, the process of Entity Resolution (ER) emerges as a crucial task, facilitating the integration of diverse knowledge bases or identifying similarities among entities. Therefore, ER seeks a vital role in identifying pairs of entity records originating from the same or different data sources, which correspond to the same real-world entity.

As highlighted in [3], the applications of ER span across various domains and offer significant benefits. For instance: (i) In healthcare, ER enables the matching of patient records obtained from different healthcare facilities like emergency rooms and hospitals. (ii) Airline security greatly benefits from ER by comparing passenger records with no-fly list entries. This matching process helps identify individuals who should be barred from boarding flights or subjected to additional security measures, enhancing overall safety and security. (iii) News and media, social networks are important channels for distributing news and engaging with audiences. In this sense, ER could be useful to determine relationships between hot-topic events and news related to them. (iv) In e-commerce, ER is utilized to match information (e.g., product records, reviews) from different retailers' websites. This matching enables the identification of popular products and the detection of fraudulent knockoffs, safeguarding consumer interests and ensuring a trustworthy marketplace. These scenarios highlight the versatility

* Corresponding author at: Tampere University, Tampere, Finland.
E-mail addresses: tiago.brasileiroaraujo@tuni.fi, tiago.brasileiro@ifpb.edu.br (T.B. Araújo), vefthym@hua.gr (V. Efthymiou), Vassilis.Christophides@ensea.fr (V. Christophides), pitoura@cs.uoi.gr (E. Pitoura), konstantinos.stefanidis@tuni.fi (K. Stefanidis).

https://doi.org/10.1016/j.is.2024.102506
Received 18 April 2024; Received in revised form 4 December 2024; Accepted 5 December 2024
Available online 12 December 2024

and significance of ER across various domains, where its application contributes to improved data accuracy, enhanced decision-making processes, and increased efficiency in various industries.

ER pipelines typically involve a blocking and a matching step [4–6]. In the blocking step, the goal is to overcome the computational cost of exhaustively comparing all possible pairs of entities in a Cartesian product manner. Blocking techniques are employed to group similar entities into blocks, allowing for comparisons to be performed only within each block. By doing so, the number of pairwise comparisons is significantly reduced, improving the efficiency of the ER process. The matching step involves the actual pairwise comparison of entities within each block. Various comparison functions are utilized to determine the similarity between entity pairs. These functions assess the attributes and characteristics of the entities to gauge their degree of similarity. In this work, we focus on the matching step, receiving the candidates (i.e., entity pairs) from the blocking step in a streaming fashion, determining the match likelihood (typically expressed via a similarity score) between them, and incrementally returning the most promising candidates, while also considering fairness aspects. We leave the blocking step implementation outside the scope of this work.

ER on data streams. Streaming data refers to data that originates from dynamic sources such as web systems, social media platforms, and sensors, which are continuously updated [7]. For instance, real-world scenarios such as news, social media, e-commerce, and airline security can be considered streaming scenarios due to their dynamic behavior. When employing ER systems with streaming data, it is important to recognize that not all data from these sources become available simultaneously. Consequently, ER must be capable of matching entities as they arrive, while also considering previously matched entities. Considering the high velocity and volume of records, it is unrealistic to match a query record over all records [8], as real-time integration prioritizes the consideration of fresh records within the temporal context [6]. Indicatively, in a single minute, 456,000 tweets are posted, 2,460,000 posts are shared on Facebook, Google conducts 3,607,080 searches, the weather channel receives 18,055,555 forecast requests, Uber riders take 45,787.54 trips, 72 h of new videos are uploaded to YouTube and 4,146,600 videos are watched [9].

In this context, ER emerges as a pivotal undertaking, aimed at consolidating and furnishing actionable intelligence to businesses and government entities [10]. It is imperative to enhance the efficiency of the ER process, given the need to process vast volumes of data within constrained timeframes. Therefore, ER faces the challenge of continuously processing the data, sliced in micro batches, on a time threshold [6,11]. Although state-of-the-art Machine Learning (ML) methods have emerged as a powerful tool in the ER domain due to the high-quality results [3,12,13], these methods struggle with streaming data [14].

Incremental processing, especially for ER, refers to the process of receiving data in a continuous stream and selectively re-running the ER task, which should consider the previously compared entities and the newly arrived entities, in each increment. However, this approach often encounters resource consumption challenges, such as increased memory and CPU usage. This is primarily because ER methods require storing a significant amount of data in memory. The trade-off related to the accumulation of data over time becomes necessary, as incremental processing relies on information processed during previous increments but there is a finite amount of computational resources [7,15,16].

Fairness in ER. Fairness is essentially related to the absence of discrimination or bias. This bias can emanate from the algorithm itself, potentially reflecting the commercial or personal preferences of its creators [3]. Additionally, it can stem from the data itself, for instance, if a survey incorporates biased questions, or if a particular demographic is misrepresented in the input data [17]. In this sense, fairness criteria aim to mitigate the implications of data bias in ER systems, which requires more than just optimizing accuracy, as traditionally done in ER. It is equally crucial to ensure fairness by incorporating

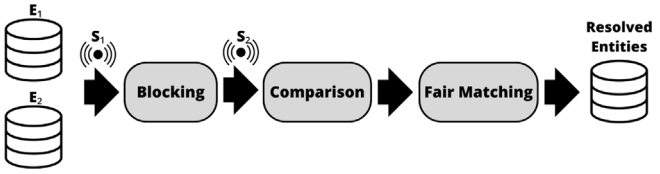


Fig. 1. Proposed workflow for fairness-aware ER over streaming data.

constraints that address the disparities between protected and non-protected groups in the ER outcomes. The identification of protected groups (such as race or gender) and the selection of fairness measures (such as equal representation or similar error rates across groups) are multifaceted issues, with various criteria and measures proposed in the existing literature [18]. For instance, considering the airline security scenario, matching records related to a no-fly list and the passengers list should consider that the two lists have different distributions of demographic groups and, for this reason, bias against specific groups of passengers should be avoided. Therefore, the pursuit of fairness-aware ER encompasses a wide-ranging problem, which has briefly been studied in the literature [3,19].

Since fairness is a complex and nuanced concept, numerous definitions and models have been put forth, along with a variety of algorithmic approaches for achieving fair rankings, especially for recommendation systems [17]. Particularly, fairness-aware ER addresses discrepancies in the size of the groups in input data (i.e., popularity bias), where the majority of resolved entities belong to a specific group. FairER [19] introduced the problem of fairness-aware ER and proposed a greedy algorithm to handle this problem based on fairness criteria to perform equal matching decisions. Especially, in the context of fairness, we propose an extension of FairER [19], by being able to incrementally process streaming data and take into account multiple protected groups. To the best of our knowledge, only two works [3,19] address the fairness aspect of ER, and none of them address the challenges related to fairness and streaming data simultaneously.

Overall, we make the following contributions in this paper:

- We introduce TREATS, a fairness-aware ER workflow able to deal with streaming data, incrementally.
- We propose a fairness-aware ER setting, targeting fairness constraints among multiple groups of interest, extending the setting introduced in FairER [19] and generalizing its algorithmic properties.
- An experimental evaluation regarding the influence of a fairness-aware ranking method in ER systems, as well as a discussion of the findings of this study. We show that TREATS can efficiently resolve entities arriving in streams, without sacrificing the effectiveness of ER, compared to state-of-the-art ER methods.

The rest of this paper is structured as follows. Section 2 provides the problem setting and the necessary background information. The proposed workflow is described in detail in Section 3. Section 4 presents the experimental evaluation and the related discussions. Section 5 discusses the related work, while conclusions and future works are presented in Section 6.

2. Preliminaries

In this section, we provide the background on the main topics discussed in this work and describe the problem of ER in the context of streaming data, incremental processing, and fairness constraints.

Let $E_l = \{e_1^l, e_2^l, \ldots, e_n^l\}$ be a collection of entities, where each entity $e_k^l \in E_l$ is a set of attribute (a) - value (v) pairs. Given two entity collections E_1 and E_2, the goal of ER is to identify the entity pairs $(e_j^1, e_k^2) \subseteq E_1 \times E_2$ that correspond to the same real-world entity; such

pairs are called *matches*. We consider three fundamental steps as part of fairness-aware ER processing: blocking, comparison, and fair matching, as illustrated in Fig. 1.

In this work, the data is received in a streaming way, which commonly needs dividing the ER process into a series of *increments* (i.e., micro-batches) sent in a time slot [6,11], denoted as $I = (i_{\tau_1}, i_{\tau_2}, \ldots, i_{\tau_{|I|}})$. Each increment i_{τ_j} contains entities $E_1^{\tau_j}$ and $E_2^{\tau_j}$ from the entity collections E_1 and E_2, respectively, and the time difference between two consecutive increments i_{τ_k} and $i_{\tau_{k+1}}$ is fixed and denoted by τ.

Blocking techniques are applied to reduce the search space for matches [2,5]. Given a pair of entity collections E_1 and E_2, the blocking step results in a set of *candidate matches* $C \subseteq E_1 \times E_2$, which are forwarded to the comparison step. The comparison step, thus, considers only those pairs that are candidate matches and ignores the rest. It is also possible to apply streaming-based blocking techniques such as [20–22], resulting in a set of candidates C_{τ_j} for each increment i_{τ_j}.

The comparison step computes the match likelihood for every candidate match. For simplicity, we assume that the candidate matches set C is defined as $C \subseteq E_1 \times E_2 \times \mathbb{R}$ and blocking initializes all triples with null matching scores. Thus, in the comparison step, the set of candidates matches C is enriched with a numeric score, this way comprising triples of the form $\langle e_j^1, e_k^2, score(e_j^1, e_k^2)\rangle$, where $score(e_j^1, e_k^2)$ is typically a real number in [0,1], with values closer to 1 indicating higher confidence that the candidate (e_j^1, e_k^2) corresponds to a match.

After the comparison, the matching step is responsible for selecting which of the candidate matches should be returned as the final matches, i.e., the output of ER. There are multiple approaches that can be followed for this step, e.g., clustering, binary classification, ranking, active learning, or combinations of the above.

In a streaming setting, the number or size of the input increments is not known in advance. For example, consider the case of continuous streams (e.g., news feeds, social media data), for which we want to find promising duplicates as they arrive. Consequently, we argue that a ranking-based matching approach, in which we are only interested in finding a subset R, e.g., of a fixed size k, of the (potentially infinite) matches is the most realistic approach for matching. Thus, in what follows, we consider the matching task as a top-k ranking problem.

We are interested in a global top-k ranking R_g, meaning that the ER system should return the k most promising candidates seen so far, i.e., not only candidates in the current increment C_{τ_j}, but also in all previous increments C_{τ_1} to $C_{\tau_{j-1}}$. When additional constraints are given, such as the one-to-one constraint assuming that each entity collection is clean, i.e., free of duplicates, these constraints can also be imposed on the output. For instance, the one-to-one assumption can be achieved by applying Unique Mapping Clustering [23,24] to the ranked candidate pairs C.

ER and fairness constraints. A typical matching method, aiming at maximizing accuracy, would simply rank the candidate pairs C in descending order of score and return the top-k pairs. To address the implications of data bias, however, it is essential for ER systems to go beyond maximizing accuracy and also adhere to fairness constraints among entity groups in their output. Here, we re-use the definition of fairness-aware ER, that we have previously introduced in FairER [19].

Definition 1 (*Fairness-aware ER [19]*)**.** Given a set of candidate matches $C \subseteq E \times E'$, a scoring function $score : E \times E' \to \mathbb{R}$, and a fairness criterion F, produce a ranking of matches $R \subseteq C$ that for any given rank position k, maximizes the cumulative scores:

$$R = \underset{R^* \subseteq C}{\operatorname{argmax}} \sum_{(e_i, e'_j) \in R^*} score(e_i, e'_j)$$

s.t. $R[k]$ satisfies F,

where $R[k]$ are the k first results of R.

Intuitively, this definition has two goals: (a) to satisfy a user-defined fairness constraint F, and (b) among the possible solutions that satisfy F, pick those that maximize the score, i.e., the match likelihood. We should emphasize that F is any fairness constraint and it is up to the user to specify it. For example, in FairER [19], we studied one instance of the fairness-aware ER problem, in which F was the statistical parity constraint, i.e., all groups have equal number of representatives in the top-k results. Other instances of this problem could ask that F is defined differently, e.g., as the demographic parity.

Example. Consider a collection of a total of 1K entities, with each entity corresponding to a job application record, received in real time from different online platforms, for the same positions in an organization. Many applicants may have submitted their application from several platforms, resulting in multiple job applications for the same person. A recruiter may want to prioritize people who have submitted their application in several platforms (e.g., because this way more information can be collected from the applicants). The recruiter may also need a balance between candidates of different genders (e.g., due to ethical reasons, or regulations). An algorithm providing a solution to Definition 1 would return a balanced (according to the fairness constraint) subset of the candidates with respect to gender, that are most likely to have submitted multiple applications.

The determination of entity groups (e.g., based on gender, country, race) and the selection of fairness measures, such as equal representation or similar error rates across groups, are considered as an open topic [3,19]. In this paper, we go beyond the typical group fairness definition that assumes the existence of a binary grouping (protected and non-protected groups) decision and consider multiple entity groups of interest. Our fairness-aware ER method treats all groups as equal and assumes non-overlapping groups, i.e., every entity belongs to exactly one group.

Another important fairness aspect in ER is the group membership decision for entity pairs. This decision is straightforward when both entities individually belong to the same group, but it gets more complicated when the two entities belong to different groups. To simplify this study, we only consider the simplest case, in which the constituent entities of each candidate match belong to the same group. We consider all these cases important and we believe that they are worth being investigated in a dedicated study.

In this work, we consider entities that contain one sensitive attribute, which should be evaluated to generate the groups of interest. In this context, let a_s be the attribute considered as sensitive, $G = \{g_1, \ldots, g_m, g_{np}\}$ be the set of groups, and $v = \{v_1, \ldots, v_m\}$ represent the attribute values[1] from the sensitive attribute a_s. Notice that the attribute values in v are determined as key for each group of interest $g \in G$. Furthermore, $g_{np} \in G$ represents the non-protected group, and $G \setminus \{g_{np}\}$ are the groups of interest.

A candidate $c = (e_j^1, e_k^2) \in C$ is inserted into a group of interest g_i, if v_i appears as the value for the sensitive attribute a_s in e_j^1 or in e_k^2 (or in both). Therefore, $c \in g_i \iff \left(v_i \subseteq e_j^1(a_s)\right) \vee \left(v_i \subseteq e_k^2(a_s)\right)$, where $e(a_s)$ returns the attribute values for the sensitive attribute a_s and v_i is the key for group g_i. For instance, let gender be the sensitive attribute a_s and the values '*female*' and '*transgender*' be the keys (i.e., $v = \{$'*female*', '*transgender*'$\}$) to the groups of interest g_1 and g_2, respectively. Thus, the entities whose values for the attribute '*gender*' present '*female*' are inserted into g_1 and the entities who present '*transgender*' are inserted into g_2. Notice that, the entities whose attribute values do not present any of the group keys (i.e., '*female*' or '*transgender*') are inserted into the non-protected group g_{np}.

[1] Although each entity collection might use different values to denote the same group (e.g., 'Female' vs. 'F'), for simplifying the presentation, we assume that the same sensitive attribute values are used by both entity collections.

As stated, beyond ranking based on the similarity scores, additional constraints may be applied to the ranking R based on specific assumptions relevant to the ER task's application. In this work, the ranking step takes into account similarity scores and fairness constraints. Therefore, after receiving the candidates C_{τ_j} from the comparison step for a specific increment i_{τ_j}, a ranking method is employed to enforce fairness constraints. Consequently, since we consider the matching task as a top-k ranking problem, the set $R_{\tau_j}[k] \subseteq C_{\tau_j}$ is considered the set of matches for the specific increment i_{τ_j}. Furthermore, this involves performing the ranking based on fairness metrics or applying constraints to ensure fair representation and treatment of different groups. Thus, the focus of the fairness-aware ranking method is to maximize the matching scores, while satisfying the fairness criteria, as formalized in FairER [19]. In TREATS, those fairness criteria are defined on multiple groups, as mentioned above, and the input and output are provided in a streaming fashion, i.e., in several iterations.

Considering the matching task as a top-k ranking problem, both for individual increment results and the overarching global top-k ranking R_g, the idea is to ensure a holistic consideration of prior ranked candidates (i.e., C_{τ_1} to $C_{\tau_{j-1}}$) by aggregating $R_{\tau_j}[k]$ with $R_g[k]$ for each increment. Thus, the united $R_g = R_{\tau_j}[k] \cup R_g[k]$ is re-ranked considering the fairness criteria and the top-k candidates are classified as matches, which is represented by $R_g[k]$. Therefore, $R_g[k]$ represents the TREATS workflow outcome over time, which is updated for every increment i_{τ_j}.

3. Fairness-aware ER over streaming data

In this section, we discuss and provide details regarding TREATS, the proposed workflow that involves *sTREaming* data and *fAirness* criteria in the *enTity reSolution* context. Considering the ER steps illustrated in Fig. 1, our focus is the comparison and fair matching steps. Therefore, this section provides detailed information related to how these steps have been adapted to handle the needs and challenges addressed in this work, which comprehends streaming data, incremental processing, and fairness.

3.1. Fairness in the context of ER

Recent works [3,19] emphasize on open topics related to appropriate metrics and group definitions to be explored in the context of fairness in ER. To this end, this subsection addresses these topics defining the scope of our study in terms of fairness.

Shahbazi et al. [3] introduce a classification of fairness according to the application of entity attributes and their respective values. For instance, FairER [19] exploits the scenario of the single attribute with binary values, where each entity should belong to one of two groups (i.e., binary option). In the present work, we go beyond that and address the scenario of the single attribute with multiple exclusive values, where each entity should belong to exactly one group among a set of groups. To this end, each group assumes a key based on the multiple exclusive options.

Exploring fairness in the context of a single attribute with multiple exclusive values typically involves assessing whether there is equitable treatment across the different values (and groups of interests) of this specific attribute. In this sense, it is important to highlight some aspects in terms of unfairness: calculate group metrics, compare group metrics, and mitigation strategies. Calculating group metrics is related to measuring the trade-off between performance and fairness within each group of interest. Compare group metrics means applying fairness metrics across different groups to identify any disparities, seeking for variations in the outcomes that may indicate potential fairness issues. If fairness issues are identified, consider implementing mitigation strategies. As treated in this work, it could involve the development of fairness-aware ranking algorithms to ensure equitable treatment across attribute values.

Concerning group metrics, fairness could be measured in terms of hit rate among the groups. To this end, consider the following three terms: (i) True Positives (TP): The number of instances that were correctly predicted as positive by the matcher; (ii) False Positives (FP): The number of instances that were incorrectly predicted as positive by the matcher when they are actually negative; and (iii) False Negatives (FN): The number of instances that were incorrectly predicted as negative by the matcher when they are actually positive.

Shahbazi et al. [3] present 11 measures to evaluate fairness in ER. Among them, the authors suggest that True Positive Rate Parity (TPRP) and Positive Predictive Value Parity (PPVP) are the two most suitable measures for evaluating fairness in ER. TPRP corresponds to equal opportunity (or equality of odds), among the pairs determined by the ER task, as a match requires the independence of true matches from any groups. In this sense, TPRP measures the proportion of actual positive instances that are correctly identified as positive by the matcher for a specific group. A lower TPRP (i.e., close to TPRP = 0) indicates that the matcher is effective at identifying positive instances and has a lower likelihood of missing actual positive cases among the groups, providing a sense of fairness. In a similar way, PPVP evaluates the independence of true matches from groups of interests among the pairs predicted as a match. In other words, PPVP is achieved if the precision (or positive predictive values) in the groups are close to each other. A lower PPVP (i.e., close to PPVP = 0) indicates that the predictions are more likely to be correct (i.e., true matches) among the groups.

We adapt the definitions of TPRP and PPVP from [3,25], which compare the recall and precision, respectively, of each group to the others, and we report as the overall TPRP and PPVP values the biggest differences in recall and precision, respectively, among groups, considered pairwise.

The TPRP is given, considering all groups pairwise, from

$$TPRP = avg_{g_i,g_j \in G} \left| |TPR_i| - |TPR_j| \right|, \tag{1}$$

where $TPR = TP/(TP + FN)$, aka recall. Similarly,

$$PPVP = avg_{g_i,g_j \in G} \left| |PPV_i| - |PPV_j| \right|, \tag{2}$$

where $PPV = TP/(TP + FP)$, aka precision.

Regarding comparing groups in terms of fairness, FairER [19] uses the metric called Bias@k, which evaluates disparities in predictions across two groups (FairER only covers the case of protected vs. non-protected groups) in the top-k results of an ER output (seen as a ranking problem). Here, we adapt the definition of Bias@k to also cover the case of multiple groups of interest, as follows:

$$Bias@k = \max_{g_i,g_j \in G} \frac{\left| |R_i| - |R_j| \right|}{k}, \tag{3}$$

where $R_x = R[k] \cap g_x$ is the representation of group g_x in the top-k rankings $R[k]$. A value of Bias@k equal to zero implies no bias (i.e., all groups are represented equally in the top-k results) and higher values indicate stronger bias in favor of or against one of the groups.

When fairness issues are identified, mitigation strategies should be applied. Therefore, in this work, a fairness-aware ranking method is proposed, extending the method introduced in FairER. This ranking method aims to minimize bias in the context of multiple groups through the definition of groups of interests and equally select candidates that belong to each group. The next subsection provides details related to how the proposed workflow mitigates unfairness in the context of streaming data.

3.2. TREATS: Parallel workflow for fairness-aware entity resolution in streaming data environments

When dealing with streaming data, it is essential to incorporate new components before the steps in the traditional ER workflow. These components involve organizing the micro-batches that are slated for

processing. To facilitate, components called senders have been integrated into the workflow. These senders serve the purpose of processing streaming data, as illustrated in Fig. 1 through the icons S_1 and S_2. The first sender, S_1, is responsible for managing the data from the data sources (i.e., E_1 and E_2) in a streaming way and sending the data to the blocking step. The blocking step groups the entities according to some entity similarities and provides, as output, blocks of candidates in a streaming way. Notice that the second sender, S_2, is responsible for managing the groups of candidates and sending them to the comparison step. The blocking step is not the focus of the present work; instead, our contributions are focused on the comparison and fair matching steps. Thus, we consider the application of streaming-aware blocking techniques such as [6,14], which produce a set of candidates (i.e., blocks) in a streaming way, as depicted in Fig. 2.

Since streaming data are being constantly received, the ER steps should be performed according to a time budget (represented by τ in this work, as stated in Section 2). This behavior creates a challenge for the workflow, which needs to be performed as fast as possible. To address this challenge, parallel processing strategies were applied to enhance ER efficiency without having a negative impact on effectiveness. More specifically, in this work, we applied Graphics Processing Unit (GPU) environments during the comparison and classification step to guarantee efficiency and also applied high-quality ER frameworks, such as Ditto [13] and GNEM [12]. In this sense, GPU stands for parallel processing environment of general purpose able to handle multiple tasks or calculations simultaneously. In Fig. 2, the GPU environment is illustrated by the parallel infrastructure component.

3.2.1. Comparison

In the comparison step, an ML matcher component is applied to compare the candidates (i.e., entity pairs) and predict the matching score between them. Note that different ML matchers can be applied to the ER workflow described in this work. In summary, state-of-the-art deep learning approaches utilize pre-trained transformer-based language models and optimize performance using domain knowledge injection, text summarization, and data augmentation techniques.

Considering neural approaches, previous works [3,13] show that Ditto and GNEM outperform existing ER solutions such as DeepMatcher [26] and DeepER [27] on different benchmark data sources with significantly less training data. These high-quality results can be attributed to the improved language understanding capability mainly through pre-trained language models. Note that beyond the effective results achieved by Ditto and GNEM, both provide pre-trained models to support the entity comparison. Pre-trained models are important in streaming scenarios due to the time threshold, since the models demand time (commonly minutes or hours) to be trained. For this reason, in the streaming data context, the neural approaches should apply pre-trained models to avoid time issues.

Thus, before the data starts coming, the ML matchers need to be trained/fine-tuned on the data sources to be processed. In a real-world scenario, the framework could warm up with the first data (provided by the first increments) or collect samples of the data to train the models. With the pre-trained models, the sender component consumes the data provided by the data sources in a streaming way, buffers it in micro-batches, and sends it to the matcher. Note that buffering streaming data in micro-batches is a common strategy to process streaming data [6,21,28,29], even in critical scenarios where data arrive continuously. Thus, it is possible to follow the τ time interval and respect the time restriction for the ER task, as defined in Section 2. To connect the sender to the matcher, a streaming processing platform, namely Apache Kafka, is applied.

To perform the comparison step, and consequently execute the matcher, a GPU infrastructure is used. For this reason, the comparison step is directly connected to the parallel infrastructure, which provides all the distributed resources (such as cores, engines, and virtual machines) needed to execute the ER task efficiently.

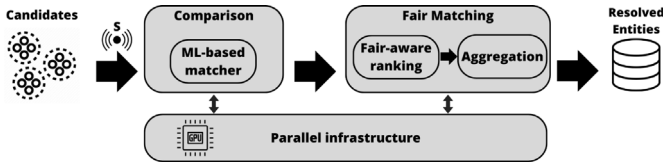


Fig. 2. Workflow considering streaming data and fairness aspects.

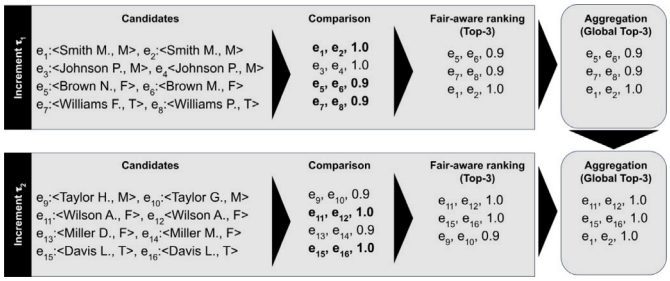


Fig. 3. Example of TREATS execution considering streaming data and fairness aspects.

The example illustrated in Fig. 3 addresses a domain involving individuals, such as a retailer's client repository or a social media profile record. In this context, individuals may provide personal points of view (e.g., opinions, emotions, and reviews) about a topic of interest. For this reason, the information flow occurs in a streaming way since the individuals provide the content on demand. Going further, it is proper that an ER approach matches entities guaranteeing an equal representation of these points of view according to the individuals characteristics such as gender, country, and race. In this sense, in our example, there are two different increments (i.e., Increment τ_1 and Increment τ_2) containing the candidates (entities pairs) that TREATS will process. The matching input are pairs of user profile records, as part of the points of view information. For presentation purposes, each entity is simply described by two attributes: name and gender, where name attribute presents string values and gender presents char values, namely *'M'* for *male*, *'F'* for *female*, and *'T'* for *transgender*. The similarity between pairs of candidates is assessed based on their attribute values. For example, in increment τ_1, the pair (e_1, e_2) is assigned a matching score of 1.0.

3.2.2. Fair matching

For each micro-batch (i.e., increment), the candidates, whose matching scores were computed in the comparison step, are ranked and resolved in the fair-matching step. Notice that, both steps should be performed respecting the time restriction τ for the ER task. For this reason, the fair-matching step follows the workflow by benefiting from the parallel infrastructure, which provides efficiency to the step. The fair-matching step is divided into two substeps: (i) Fairness-aware ranking, where the ranking method is applied to the candidates, taking into account the similarity scores and the fairness criteria to define the matches; (ii) Aggregation, which is responsible for aggregating the current matches with the matches resolved in prior increments. Following, we describe how the fair-matching step works in the sense of ranking and resolving candidates in the context of streaming data and fairness constraints.

At the heart of ER lies the ranking step, a pivotal stage that performs a crucial role in determining the quality of the resolution process. The ranking step involves assessing potential matches among the candidates, based on a set of similarity metrics or features. In this work, the goal is to rank the potential matches, indicating the likelihood that the entities truly refer to the same real-world object, while also considering aspects of fairness. To this end, the proposed fairness-aware ranking is applied during the fair-matching step, as illustrated in Fig. 2.

One of the primary challenges in ER is striking the right balance between precision and recall. Precision ensures that the matches identified are accurate, while recall aims to capture as many true matches as possible. Achieving this balance is crucial, as an overly conservative approach could result in missed matches, while an overly liberal one might lead to false positives. Moreover, additional challenges emerge in this step since other criteria could be considered to avoid unfairness in ER. In this sense, the ranking step is taken to another level since it takes into account specific requirements or constraints of the domain and nuances that can significantly enhance the ER results.

Considering fairness-agnostic ER solutions, where the typical strategy involves ranking matching pairs based on their scores, the results commonly present higher similarities among records of a certain group and, consequently, the representation (or diversity) of demographic groups in data is biased [19]. More specifically, Shahbazi et al. [3] highlight potential unfairness under two common conditions in real-world societies and, consequently on the data: (i) when some demographic groups are over-represented, and (ii) when names are more similar in some groups compared to others. For this reason, additional constraints may be applied to the ranking step, based on specific assumptions relevant to the particular ER task. For instance, instead of considering only the similarity score between the entity pairs, also take into consideration other qualitative aspects of these results, such as fairness or diversity.

In this study, we are expanding the conventional assessment of ER results to incorporate considerations of fairness. In a fair ER context, the initial results (i.e., the output of the comparison step) retrieved should not only be the ones most likely to represent matches, but they should also adhere to a specified fairness criterion. Specifically, we present an extension of the ranking method introduced in [19]. While the ranking method in [19] addresses scenarios with a single attribute featuring binary values, our proposed evolution extends its applicability by incrementally processing streaming data and considering multiple groups of interest.

Following the TREATS workflow, the proposed fairness-aware ranking (Algorithm 1) receives as input the output of the comparison step (i.e., list of candidates following the format $\langle e_j^1, e_k^2, score(e_j^1, e_k^2)\rangle$). First, the fairness-aware ranking sorts based on the scores in descending order, guaranteeing that the candidates with high similarity scores appear first (Line 5). Thus, the set of candidates is converted into priority queues in descending scores, where each iteration of the queue returns the top pair as a match.

Since the ranking method handles multiple groups of interest, the *id* (i.e., key) of each group is extracted from the information presented in the attribute values of the entity pair that composes the candidate in question. Therefore, the idea is to classify the candidate into a group of interest according to the value of the sensitive attribute. For instance, let gender be considered as the sensitive attribute; the groups of interest may be determined by the following *ids*: *1:male*, *2:female*, *3:transgender*, and *4:others*. Notice that, candidates that do not classify as *male*, *female*, or *transgender* are classified as '*others*'. Then, the set of candidates is iterated (Lines 6–8), so that the group *id* is extracted (Line 7) for each candidate, and the candidate is inserted into the priority queue according to its group *id* (Line 8).

Then, while the queues of candidates in Q are not empty or the number or matches is less than k, the top candidate of a different priority queue is picked each time (Line 13), following the sequence of the groups. The main idea behind this strategy is to minimize the bias present in the candidates ranked based only on the similarity scores. To this end, the fairness constraint defined by Bias@k in this section is considered to guide the algorithm. For this reason, the fair ranking aims to pick one candidate from each group per iteration. Then, if the picked candidate pair involves entities that have already been matched, i.e., belong to sets S or T (Lines 14–16), the algorithm proceeds to the next candidate from the same group, to ensure the one-to-one matching constraint. Otherwise, the entity pair e_j^1, e_k^2 is inserted into the end of the output ranking R (Line 17), the entity e_j^1 from the source is inserted into S (Line 18), and the entity e_k^2 from the target is inserted into T (Line 19). Finally, the index of the group is incremented to consider the next group of interest for the following iteration (Lines 21–26). If the index i surpasses the number of groups, it is reset to 0, in order to start again from the first group.

Algorithm 1: Fairness-aware ranking method

Input: C: list of candidates following the format $\langle e_j^1, e_k^2, score(e_j^1, e_k^2)\rangle$
Output: $R[k]$: *top-k* ranked matches

```
1  R ← ∅;
2  Q ← ∅ // dictionary that stores the priority queue of the
     groups;
3  S ← ∅ // matches from source;
4  T ← ∅ // matches from target;
5  C ← sort(C) // sort based on the scores in descending order;
6  foreach c = ⟨e_j^1, e_k^2, score(e_j^1, e_k^2)⟩ in C do
7      id ← groupId(c) // returns the id of the group of interest;
8      Q[id].put(c);
9  end
10 i ← 0;
11 while Σ_{i∈[0,|Q|−1]} |Q[i]| ≠ 0 and |R| < k do
12     if Q[i] ≠ ∅ then
13         c ← Q[i].pop();
14         if c.e_j^1 ∈ S or c.e_k^2 ∈ T then
15             continue;
16         end
17         R.append(c.e_j^1, c.e_k^2);
18         S.append(c.e_j^1);
19         T.append(c.e_k^2);
20     end
21     if i < |Q| then
22         i ← i + 1;
23     end
24     else
25         i ← 0
26     end
27 end
```

In addition to performing the ranking, fair matching is also responsible for resolving the entities. In this sense, the matches are defined based on the top-k rank position (Line 11 in Algorithm 1), maximizing the cumulative scores and satisfying the fairness constraints. Therefore, for each increment, the candidates inserted into the $R[k]$ are considered matches. Following the TREATS workflow, in the Aggregation step (in Fair matching), the $R[k]$ for a specific increment is aggregated with the previous results from the prior increments. To this end, the global $R_g[k]$ is united to the $R[k]$ of the current increment and the matches of $R_g[k]$ are re-ranked considering the similarity scores and the fairness criteria. In other words, Algorithm 1 is applied over $R_g[k]$ to rank and select the $top{-}k$ matches. Notice that, in scenarios involving streaming data, data sources may potentially provide a huge amount of data continuously. Therefore, the computational resources of a distributed infrastructure may not be enough for the accumulative sum of entities to be processed in each increment. Considering these scenarios, memory consumption is stated as being one of the biggest challenges faced by incremental ER [14,22]. For this reason, the top-k strategy assumes a fundamental role in the sense of managing resource consumption, which avoids that R_g increases infinitely and enables the efficient execution of TREATS over time (i.e., as the increments arrive).

In the example illustrated in Fig. 3, after the comparison step, top-3 candidates are chosen. In this sense, to select the 3 candidates, the similarity scores and equal representation between the groups of interest

are applied as criteria. In this example, the candidates are divided into 3 groups based on the *gender* attribute. Hence, the groups are represented by '*female*', '*transgender*', and '*others*' (which may include '*male*' and any other value). Thus, the set of candidates is converted into a priority queue for each group, with scores arranged in descending order. For the first increment, the top-3 candidates are represented by the pairs $\langle e_1, e_2\rangle$, $\langle e_5, e_6\rangle$, and $\langle e_7, e_8\rangle$, which satisfy the criteria of similarity scores and the equal representation among the groups. Note, since the pair $\langle e_3, e_4\rangle$ belongs to the group '*others*' and has a lower similarity score compared to the pair $\langle e_1, e_2\rangle$, it is not included in the top-3 candidates. Finally, since for the first increment the global result is empty, the global structure simply aggregates the result from the fairness-aware ranking step. Therefore, the matches for the first increment are the pairs: $\langle e_1, e_2\rangle$, $\langle e_5, e_6\rangle$, and $\langle e_7, e_8\rangle$.

Similarly, for the second increment, the pairs $\langle e_9, e_{10}\rangle$, $\langle e_{11}, e_{12}\rangle$, and $\langle e_{15}, e_{16}\rangle$ are included in the top-3 candidates, satisfying the similarity and fairness criteria. Notice that the pair $\langle e_{13}, e_{14}\rangle$ is not considered even if it belongs to a group of interest (i.e., '*female*'). This is because the pair $\langle e_{11}, e_{12}\rangle$ (which also belongs to '*female*' group) presents a higher similarity score compared to $\langle e_{13}, e_{14}\rangle$. From the second increment, it is necessary to take into account the previous results, which are stored in the global structure, and the current one. In the aggregation step, the similarity and fairness criteria are applied considering the current result and the global result. Therefore, due to the similarity score criteria, the pair $\langle e_{11}, e_{12}\rangle$ replaces the pair $\langle e_5, e_6\rangle$ for the group '*female*' and $\langle e_{15}, e_{16}\rangle$ replaces the pair $\langle e_7, e_8\rangle$ for the group '*transgender*'. For the group '*others*', the pair $\langle e_1, e_2\rangle$ is not replaced since it presents a similarity score higher than $\langle e_9, e_{10}\rangle$. Thus, the returned matches at the end of the second increment are: $\langle e_1, e_2\rangle$, $\langle e_{11}, e_{12}\rangle$, and $\langle e_{15}, e_{16}\rangle$.

3.2.3. Properties of TREATS

Extending Proposition 3.1 of FairER [19]. Proposition 3.1 from FairER [19] states that the algorithm suggested in FairER is a 1 - 1/e approximation to the problem of Definition 1, for F defined as $|(|R_p|/k) - (|R_n|/k)| = \epsilon *$.

Here, we extend this property, claiming that TREATS is a $1 - 1/e$ approximation to the problem of Definition 1 for F defined as $max_{g_i,g_j \in G}\left(|(|R_i|/k) - (|R_j|/k)|\right) = \epsilon *$, where $\epsilon *$ is the smallest possible ratio difference for a given k, and R_i is the returned ranking for elements of group g_i. Generalizing the cases of the smallest possible ratio difference $\epsilon *$ for a given k, compared to FairER [19], the two possible values of $\epsilon *$ are: 0, when $k \ mod \ |G| = 0$, and $1/k$, otherwise. Intuitively, when k is a whole multiple of the number of groups $|G|$, then all groups are represented equally ($max_{g_i,g_j \in G} \| R_i| - |R_j\| = 0$, for all groups). Otherwise, some groups may have 1 fewer representative than other groups ($max_{g_i,g_j \in G} \| R_i| - |R_j\| = 1$).

Streaming vs. Batch results (eventual consistency). Assuming that (i) the same input data D are provided to FairER [19] and TREATS (in streams $D_1, D_2, \ldots, D_n$, such that $D = \bigcup D_i$), (ii) FairER and TREATS use the same matching score estimation function (e.g., DITTO, GNEM, DeepMatcher) (iii) there are only two groups of interest g_1 and g_2 (since the algorithm suggested in FairER is only applicable in that case), (iv) $R_T[k]$ is the ranking provided by TREATS at the last iteration and $R_F[k]$ is the ranking provided by FairER, then the following property holds:

$$\sum_{(e_i,e'_j) \in R_T[k]} score(e_i, e'_j) = \sum_{(e_i,e'_j) \in R_F[k]} score(e_i, e'_j) \quad (4)$$

Intuitively, this means that both the batch version (FairER) and the streaming version (TREATS) will provide top-k ranking with the same cumulative scores of matching probabilities.

Non-Uniqueness of the solution. Following the previous property, notice that the actual rankings may be different, i.e., different pairs may be suggested by FairER and by TREATS. This may happen due to ties in matching scores. For example, a different pair may be suggested in the top rank of TREATS than in the top rank of FairER, if they both have an equal score. Due to this fact, the evaluation scores of the last iteration of TREATS and FairER may be slightly different.

This property can be generalized also for the problem described in Definition 1. This problem may have multiple valid solutions, i.e., solutions that all respect the fairness constraint and all yield the same score, which is the maximum possible score, even if the top-k results $R[k]$ are different, or even if the entity pairs constituting two solutions $R[k]$ and $R'[k]$ are the same, but ordered differently.

4. Experiments

In this section, we evaluate the proposed workflow in the context of real-world data sources in a streaming way. Moreover, we evaluate the impact of the proposed fairness-aware ranking method over the whole workflow. Throughout this section, we present the configuration of the computational environment, the experimental design, and the achieved results in terms of effectiveness, efficiency, and fairness metrics.

The experiments address the following research questions:

- **RQ1**: In terms of effectiveness, is the TREATS workflow competitive to the state-of-the-art ER approaches?
- **RQ2**: Does the fair ranking step significantly impact the TREATS workflow in terms of efficiency?
- **RQ3**: Regarding fairness, does the TREATS workflow (with the fair-aware ranking method) outperform the baseline technique?

4.1. Settings, data sources and experimental design

We conducted the experiments on a Cloud GPU instance in the Google Cloud,[2] platform with the following configuration: N1-standard-4 instance with 2 NVIDIA T4 GPUs, 4 vCPUs (2 cores each), 15 GB memory, and running Debian 10 based Deep Learning VM for TensorFlow Enterprise 2.10 with CUDA 11.3. All the steps of the proposed workflow were implemented in Python. Regarding the streaming processing, Apache Kafka 3.4.0,[3] [29] platform was applied to connect the sender and the other components of the workflow (in the architecture). We accessed the source code of the proposed workflow either through the authors' public GitHub[4] or by directly contacting the authors.

As described in Table 1, we used five real-world pairs publicly available for evaluating the TREATS workflow. They have been extensively used in the ER literature, e.g., in [3,6,19,30], in similar contexts to this work, such as streaming data and fairness evaluation of ML-based matchers. Table 1 also shows the name of data sources (*Name*), the domain related to the data sources (*Domain*), number of entity pairs in the training set (*#Train*), number of entity pairs in the testing set (*#Test*), number of attributes in each data source (*#Attributes*), the sensitive attribute selected to the experiments (*Sensitive Attribute*), and the sensitive attribute values considered to generate the groups (*Groups*). Notice that attribute values from the sensitive attributes are considered to determine the groups. For instance, for DBLP-GoogleScholar (D1) data sources, the values *vldb*, *sigmod*, and *tods* are considered as group keys. Therefore, records that present one of these values in the attribute *venue* are inserted into the respective group. On the other hand, records that do not present any of these values are inserted into the group classified as '*others*'.

To simulate streaming data behavior, a data-streaming sender was implemented. This data streaming sender reads the entities from the data sources and sends the entities through the Kafka platform. Therefore, the sender is responsible for providing the data in a continuous way in each τ time interval (i.e., increment), to be consumed by the TREATS. In this work, $\tau = 6$ sec (i.e., one increment per 6 s) and $\tau = 12$

[2] https://console.cloud.google.com/ accessed on 28 March 2024.
[3] https://kafka.apache.org/ accessed on 28 March 2024.
[4] https://github.com/brasileiroaraujo/FAIR.

Table 1
Data source characteristics.

Name	Domain	#Train	#Test	#Attributes	Sensitive attribute	Groups
DBLP-GoogleScholar (D1)	Publications	17,223	5742	4, 4	venue	vldb, sigmod, tods and others
DBLP-ACM (D2)	Publications	7417	2473	4, 4	authors	female and others
Amazon-Google (D3)	Products	6874	2293	3, 3	manufacturer	microsoft, sony, apple and others
Walmart-Amazon (D4)	Products	6144	2049	5, 5	category	printers, laptops, cameras and others
iTunes-Amazon (D5)	Musics	321	109	8, 8	genre	hip-hop, dance, rock and others

sec (i.e., one increment per 12 s) are used for the experiments involving Ditto and GNEM, respectively.[5] Another aspect to be evaluated is the behavior of the proposed workflow as the increments arrive over time. In this sense, the data sources were divided into increments with sizes of 200, 400, and 600 entity pairs to be compared.[6] For instance, considering D1 (i.e., DBLP-GoogleScholar) with 5742 entity pairs to be compared and $|i| = 200$, 28 increments with 200 pairs and one increment with 142 pairs (the remaining pairs) will be generated. The variation of the increment sizes aims to avoid experimental bias and supports the evaluation of the results in terms of effectiveness, efficiency, and fairness.

Throughout this section, we call TREATS as the proposed workflow, which considers all steps described in Section 3. In the Comparison step, we applied two state-of-the-art ML-based matchers: Ditto [13] and GNEM [12]. Notice that both matchers provide pre-trained models, which were adapted to be inserted into the streaming data incrementally. As previously stated, in the streaming environment there is a time budget, therefore, the time used to train the models commonly exceeds this time budget. Since GNEM provides only pre-trained models for the product data sources, which is a requirement for efficient streaming processing, we evaluate GNEM results only for the data sources D3 and D4.[7] In this section, we call Ditto and GNEM the application of each matcher through the steps of TREATS workflow, but without performing the raking step. Furthermore, TREATS (Ditto) and TREATS (GNEM) represent the application of TREATS workflow with the application of Ditto and GNEM respectively.

4.2. Evaluation of Ditto and GNEM application

Ditto and GNEM approaches have consistently demonstrated superior effectiveness when compared to the DeepMatcher approach in the realm of data matching and record linkage [3,12,13]. The fundamental reason behind this improved performance lies in the approach's innovative methodology and its adaptability to a wide range of data-matching tasks. Ditto incorporates a combination of deep learning techniques, natural language processing, and fuzzy matching algorithms. Regarding GNEM, it is possible to infer that it is an extension of DeepMatcher since GNEM incorporates strategies and algorithms applied in DeepMatcher, as highlighted in [12].

Concerning Ditto, we conduct an evaluation that compares Ditto against Deepmatcher in terms of effectiveness. To this end, we adapted the FairER approach [19], which originally applies Deepmatcher, to apply Ditto to perform the entity comparison. Considering the five pairs of data sources used in this evaluation, FairER + Ditto achieved equal or better precision results for all of the data sources when compared with FairER + Deepmatcher. These results are also supported by the evaluation conducted in [3].

[5] Based on the results of the experiments, no incremental scenario exceeded the 6-second and 12-second time limit for the respective matching tool.

[6] Since D5 has 109 entity pairs, for this scenario, the whole data source is processed in one increment.

[7] The pre-trained models can be found at https://github.com/ChenRunjin/GNEM (accessed on 15 April 2024).

Table 2
Effectiveness results.

Method	D1	D2	D3	D4	D5
Precision@5					
Ditto	1	1	1	1	1
TREATS(Ditto)	1	1	0.8	0.8	1
GNEM	–	–	0.7	1	–
TREATS(GNEM)	–	–	1	1	–
Precision@10					
Ditto	1	1	1	0.8	1
TREATS(Ditto)	1	1	0.9	0.8	1
GNEM	–	–	0.73	1	–
TREATS(GNEM)	–	–	0.93	1	–
Precision@15					
Ditto	1	1	0.93	0.8	1
TREATS(Ditto)	1	1	0.93	0.86	1
GNEM	–	–	0.73	1	–
TREATS(GNEM)	–	–	0.93	1	–
Precision@20					
Ditto	1	1	0.95	0.85	1
TREATS(Ditto)	1	1	0.95	0.85	1
GNEM	–	–	0.73	1	–
TREATS(GNEM)	–	–	0.85	0.95	–

4.3. Effectiveness evaluation

We assess Ditto and GNEM without the application of fair-aware ranking and the full workflow of TREATS applying Ditto and GNEM as matchers concerning effectiveness, considering the top-k positions of the retrieved results, where $k \in \{5, 10, 15, 20\}$, as in FairER [19]. Intuitively, in the case of two groups ($|G| = 2$), when k is even, bias should be zero (ideal case), but it cannot be zero when k is odd (as there cannot be a perfect balance between groups in that case). In the general case, where more than two groups may appear, zero bias can exist only when $k \mod |G| = 0$, where $|G|$ is the number of groups. Other than that, the choice of k is up to the user and we only present the results for some arbitrary k values.

The metrics employed include the following: $Precision@k \in [0, 1]$, which measures the correctness of k returned matches with respect to the ground truth of known matches. Therefore, the precision value is given by ($Precision@k = \frac{|TP|}{k}$), where TP is the number of True Positive (i.e., truly matches) records among the top-k positions.

Table 2 presents a comprehensive overview of the $Precision@k$ outcomes for TREATS, Ditto and GNEM. In scenarios involving the product-domain data sources (i.e., D3 and D4) and where k is low (e.g., $k = 5$ and $k = 10$), Ditto outperforms TREATS. However, notice that as k increases, the precision tends to equalize. It occurs due to TREATS applying the fair-aware ranking, which in some cases prioritizes false positive records (i.e., records predicted as match, but they are not) with high similarity scores and includes them in the top positions. On the other hand, for a higher number of k, Ditto also includes these false positive records. For instance, considering the data sources D3 (Amazon-Google) with $top{-}20$, the precision is 0.95 for both approaches which means from the 20 positions, 19 records are true positives and 1 is false positive. Note, considering that TREATS applies the fair-aware ranking, it inserts this false positive record since the $top{-}5$ because this record belongs to a protected group and, consequently, the

record is prioritized by the ranking. On the other hand, Ditto (which does not apply ranking) only includes the false positive record from *top*−15.

Regarding Ditto and GNEM results for product-domain data sources, Ditto outperforms GNEM for D3 while GNEM outperforms Ditto for D4. This behavior also occurs when these matchers are applied in the TREATS workflow, hence, precision results tend to be guided by the applied matchers. In this sense, considering the *top*−20 results, the precision results achieved by Ditto and TREATS (Ditto) are the same while the results of GNEM and TREATS (GNEM) are quite close. Applying the T-Student and Wilcoxon statistical test over the results, the *p*-value was higher than 0.05 (with 95% confidence) for all scenarios, denoting that *there is no statistically significant difference between Ditto vs. TREATS (Ditto) and GNEM vs. TREATS (GNEM) precision results*, which answers **RQ1**.

To evaluate the precision results as increments arrive over time, we conducted an evaluation of TREATS and matching tools (Ditto and GNEM) applying the pairs of data sources, varying the number from 200, 400, and 600 entity pairs per increment. TREATS (Ditto) and Ditto achieved perfect precision (i.e., $precision = 1.0$) from the first increment for the data source pairs D1, D2, and D5. In other words, for all increments and all variation sizes of entity pairs per increment, the precision of TREATS (Ditto) and Ditto remained equal to 1.0.

Regarding the product-domain data sources (D3 and D4), the precision values for the combination of TREATS with Ditto and GNEM are presented in Fig. 4. During this experiment, we apply the top-20 positions to measure the precision results. For TREATS (Ditto), the precision values range from 0.8 to a perfect score of 1.0, indicating a consistent performance across the increments. On the other hand, Ditto exhibits a similar range, with precision values spanning from 0.85 to 1.0. Notably, both approaches share comparable precision values in certain instances, particularly over the number of increments increase. The results suggest that both TREATS (Ditto) and Ditto are capable of achieving high precision, but the nuances in their individual performances become apparent in scenarios where the application of ranking strategies may interfere with the precision results. Similarly, when GNEM is applied, the precision values of TREATS (GNEM) range from 0.7 to 0.95 while GNEM achieved precision scores between 0.7 and 1.0. In this sense, the results suggest that even though TREATS applies a fairness-aware ranking method, over time (i.e., as the number of increments increases), the precision scores tend to be similar compared to the matching tools (Ditto or GNEM) scores.

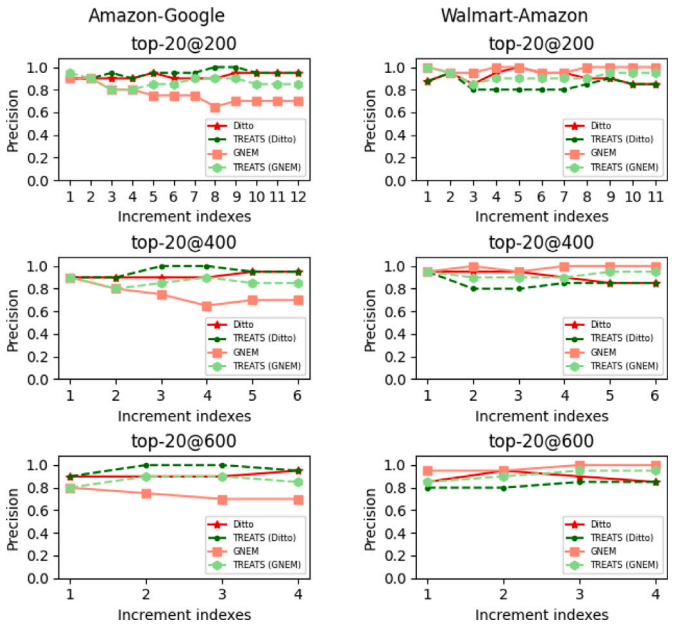


Fig. 4. Precision results considering the streaming behavior: Amazon vs. Google (D3) and Walmart vs. Amazon (D4).

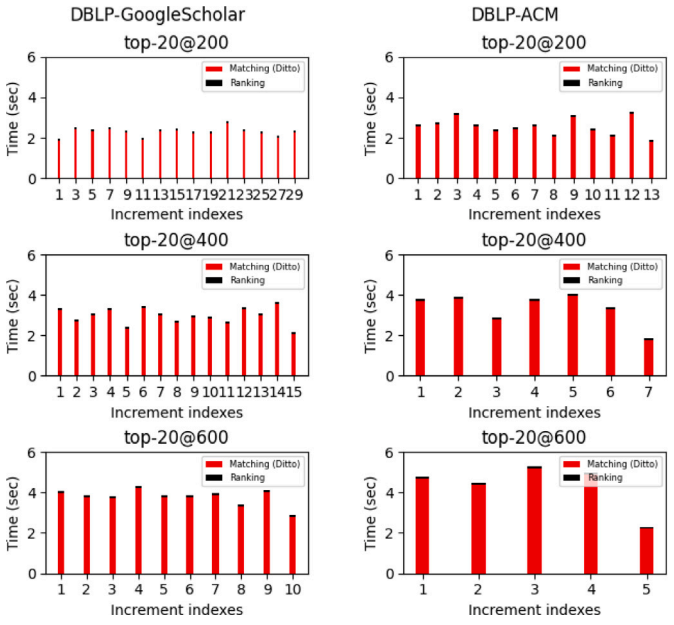


Fig. 5. DBLP vs. Google Scholar and DBLP vs. ACM efficiency results.

4.4. Efficiency evaluation

In assessing the efficiency of TREATS with the application of Ditto and GNEM, we employed execution time as the metric. The results for the data sources D1 and D2 are illustrated in Fig. 5. Concerning D3 and D4, the results are illustrated in Fig. 6, in which we present the comparison of applying Ditto and GNEM over the TREATS workflow in terms of efficiency. In the scenario involving D5, the data sources were processed in one increment. Thus, TREATS with the application of Ditto demanded 2.30 s to perform the matching and 0.06 s to perform the ranking.

It is important to highlight that the time execution for ranking includes the time to rank the current increment and also rank the global structure, which stores the merged result of previous increments (as explained in Section 3).

The execution time results for TREATS with the application of Ditto and GNEM reveal a predominance of the comparison step over the total execution time. Therefore, the time to execute Ditto and GNEM to compare the entity pairs corresponds to the biggest slice of time in the TREATS workflow. Notice that this behavior occurs in all scenarios evaluated in this experiment, while the comparison step takes some seconds to conclude, the ranking step only takes a few milliseconds. In this sense, considering all scenarios investigated during these experiments, TREATS (Ditto) exhibit execution times ranging from approximately 2 to 5 s while TREATS (GNEM) exhibit execution times ranging from approximately 4 to 12 s. Also, TREATS (Ditto) and TREATS (GNEM) present a consistency with slight variability across different increments. In another perspective, comparing TREATS (Ditto) against TREATS (GNEM), it is possible to determine consistent differences in execution times for D3 and D4. Applying the T-Student and Wilcoxon statistical test over the results of both scenarios, the *p*-value was lower than 0.05 (with 95% confidence), denoting that there is a significant difference between the time results. Therefore, based on the results, we can infer that TREATS (Ditto) overcomes TREATS (GNEM) in terms of efficiency.

Interestingly, comparing TREATS against the application of a pure matching tool (for instance, Ditto or GNEM), despite the subtle differences in execution times, both share similar performance characteristics

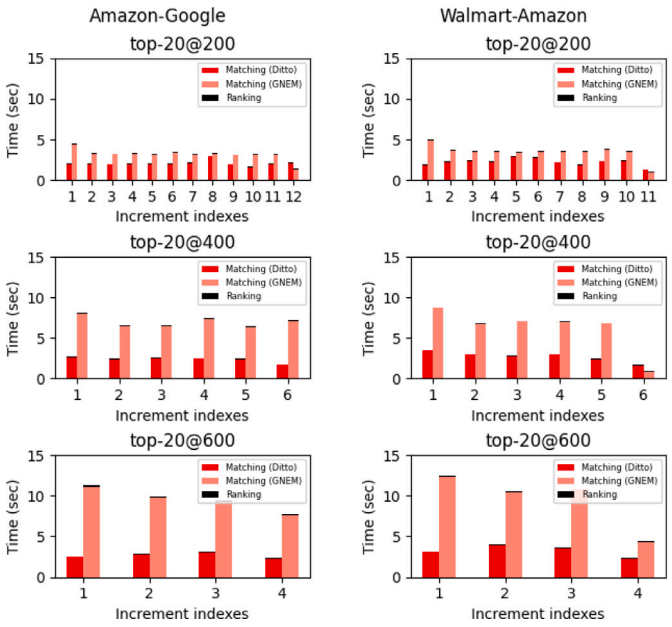


Fig. 6. Amazon vs. Google and Walmart vs. Amazon: efficiency results.

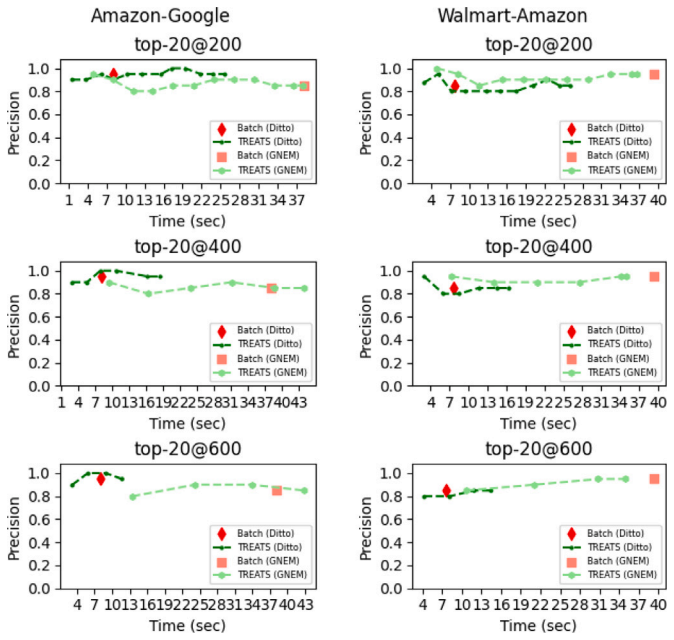


Fig. 7. Precision against the accumulated time execution considering the batch and streaming behavior: Amazon vs. Google and Walmart vs. Amazon.

in all scenarios. It occurs due to the high performance for executing the ranking step, being insignificant considering the total time. In the worst case, the time to rank was approximately 0.05 s. Applying the T-Student and Wilcoxon statistical test over the results, the p-value was higher than 0.05 (with 95% confidence) for all scenarios explored in this evaluation, denoting that there is no statistically significant difference between TREATS and matching tools execution time results. Therefore, it is not possible to infer that the ranking step significantly impacts the TREATS workflow in terms of efficiency, which answers **RQ2**.

Another perspective to be explored is related to precision against cumulative time. In Fig. 7, the precision is evaluated when the data source is processed in only one batch (represented by the diamond marker for Ditto and the square marker for GNEM) and when the data source is sent in a streaming way (represented by the line plot). In this evaluation, the product data sources are applied since they present precision variation among the increments, which facilitates the compression regarding the comparison between precision and time as the increments arrive. For streaming data, the time is given by the accumulative time, in other words, the time for a specific increment is given by the sum of the current time execution and all the previous ones (i.e., previous increments).

In this sense, it is important to highlight that TREATS achieves high precision scores from the first increments even though the data is not entirely sent. Furthermore, in some cases, TREATS surpasses batch processing in terms of precision, as depicted in Fig. 7. This occurs due to false positive candidates being inserted into the results, as new increments arrive. Notice that, even if batch processing takes less time when compared with streaming data, it is related to the nature of the data sent to be processed. Streaming processes need to incrementally match and re-match the candidates considering the current increment and also the prior ones, which implies an overhead of computational resources and, consequently, more time [3,14]. However, although for the last increment considered in this evaluation, the precision scores are the same when the full data source is considered, the benefit of our approach is that high precision scores can be achieved from the first few increments and vary only slightly as the next increments are processed.

4.5. Fairness evaluation

To evaluate the fairness of TREATS with the application of Ditto and GNEM over the five data source pairs, we apply the three metrics defined in Section 2: Bias, TPRP, and PPVP. Note that lower PPVP and TPRP values (close to 0) suggest greater fairness in terms of hit scores across multiple groups. Similarly, Bias@k value of 0 denotes no bias, while positive values indicate the existence of bias among groups. In this sense, as the focus of our study, the idea is to compare whether the application of a fairness-aware ranking method improves the quality of the matching result in terms of fairness. Moreover, take into account not only two groups (protected and non-protected) but multiple groups from the sensitive attribute. To this end, these fairness metrics measure the level of fairness in relation to the distribution of data (i.e., Bias@k) and hit rate (i.e., TPRP and PPVP) among the groups.

Regarding the PPVP metric, TREATS (Ditto) and Ditto achieved a perfect score (i.e., PPVP = 0) for all increments of the publication-domain data sources (i.e., D1 and D2). However, considering the TPRP and Bias metric, there is a significant difference between the results of TREATS (Ditto) and Ditto. TREATS (Ditto) tends to maintain TPRP and Bias lower than 0.3 for D1 and close to 0 for D2, as illustrated in Figs. 8 and 9, respectively. Notice that TREATS (Ditto) surpasses Ditto in terms of TPRP and Bias scores for all increments involving D1 and D2 data sources.

The positive Bias scores observed in TREATS can be explained by the number of candidates from some groups might not be adequate to ensure equal distribution among them. For instance, in the first increments, there may be a lack of records with high similarity scores for certain groups to be considered in the ranking step. Consequently, despite efforts to ensure fairness, inherent imbalances in the data sources can result in minor biases in the ER outcomes. However, as more increments are processed, additional candidates are considered, leading to an increase in the number of candidates from different groups. This tends to decrease the Bias score due to the fairness-aware ranking method implemented in TREATS, as depicted in Figs. 8–11.

Concerning the product-domain data sources (i.e., D3 and D4), TREATS present significant differences in terms of TPRP and Bias when compared with the results of the matching tools Ditto and GNEM. In

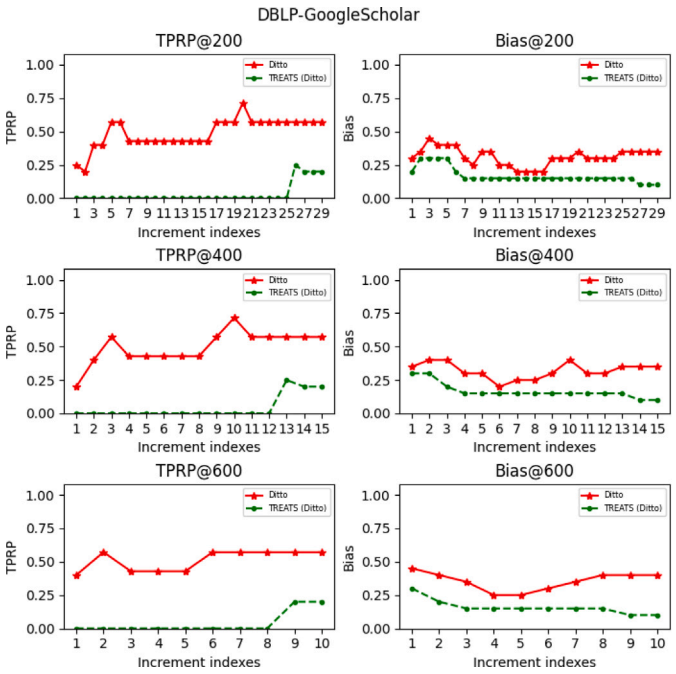


Fig. 8. DBLP vs. Google Scholar (D1): Bias and TPRP results.

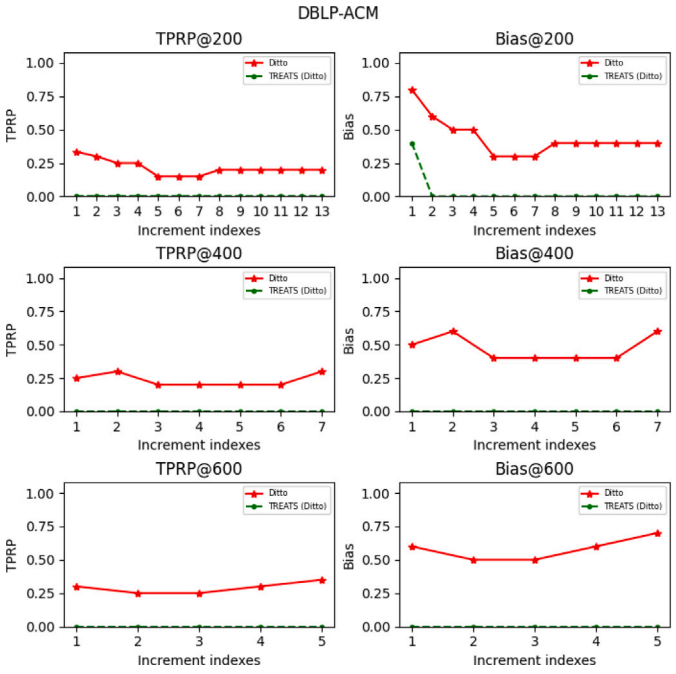


Fig. 9. DBLP vs. ACM (D2): Bias and TPRP results.

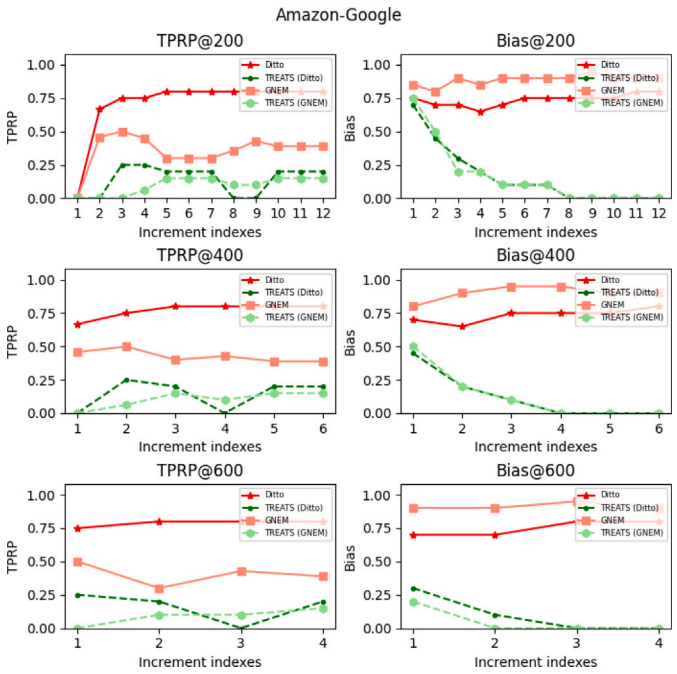


Fig. 10. Amazon vs. Google (D3): Bias and TPRP results.

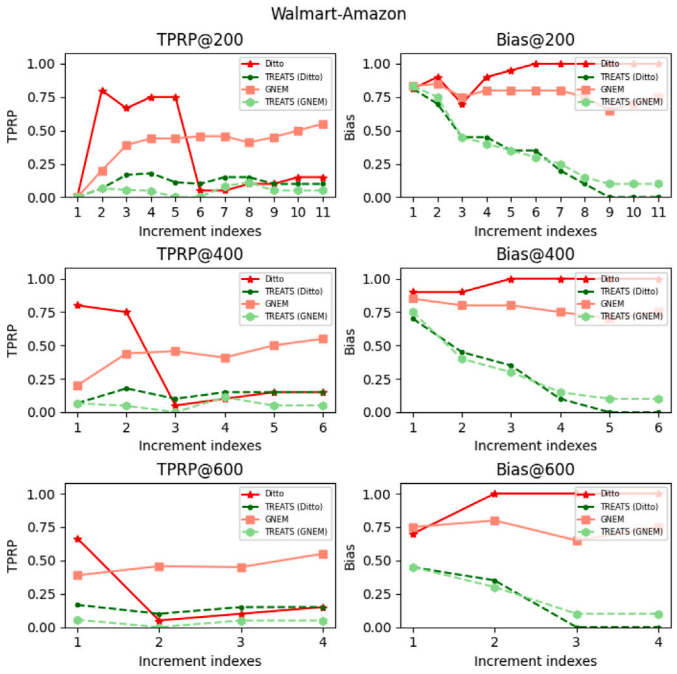


Fig. 11. Walmart vs. Amazon (D4): Bias and TPRP results.

Fig. 10, TREATS (Ditto) achieved better results compared with Ditto in terms of TPRP for all scenarios. Similarly, TREATS (GNEM) also overcomes GNEM for the TPRP metric. On the other hand, in Fig. 11 it is possible to highlight that even though for the first increments there is a high difference for TPRP, as the increments arrive, TREATS (Ditto) and Ditto present similar results. Comparing the TREATS (GNEM) and GNEM TPRP results, TREATS (GNEM) presents a better score for all scenarios.

Regarding the Bias metric, which seeks to unravel the nuanced impact of each approach on mitigating bias, it is important to highlight that TREATS achieved the perfect score of Bias = 0 for both scenarios of product-domain data sources when Ditto is applied. When GNEM is applied in TREATS, it achieves the perfect score of Bias = 0 for D3 and Bias = 0.09 for D4. This behavior suggests that the application of the fairness-aware ranking method improves the result in terms of mitigating bias. In this sense, when T-Student and Wilcoxon statistical tests are applied, the p-value was lower than 0.05 (with 95% confidence) for all scenarios explored in this evaluation, denoting that there is a significant difference between TREATS and the matching tools Bias results. Therefore, we can highlight that the proposed fairness-aware ranking method improves the results of TREATS, in terms of avoiding Bias.

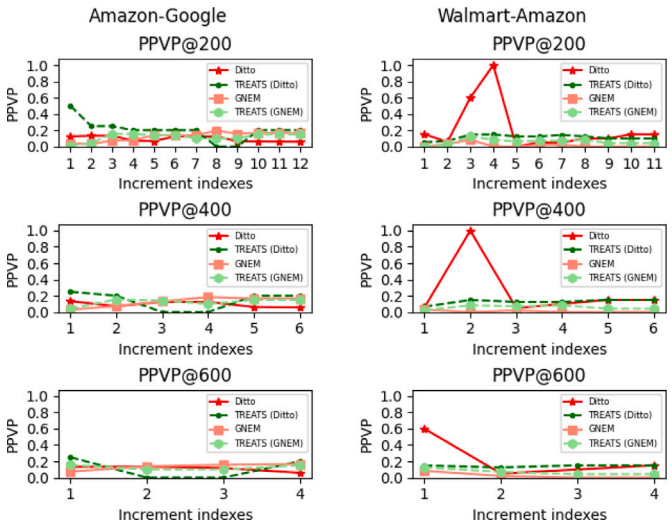


Fig. 12. Amazon vs. Google and Walmart vs. Amazon: PPVP results.

Considering the PPVP metric for the product-domain data sources, the presented results demonstrate an alternation between TREATS and matching tools with better results throughout the increments, as illustrated in Fig. 12. It occurs due to the hit score for product-domain scenarios directly impacting this metric score, which is also discussed in Section 4.3 (effectiveness discussion). Applying the T-Student and Wilcoxon statistical test over the PPVP results, the *p*-value was higher than 0.05 (with 95% confidence) for all scenarios, which means that there is no statistical difference between TREATS and the matching tools. Therefore, it is not possible to infer that the ranking step significantly impacts the TREATS workflow in terms of PPVP.

To answer the research question **RQ3** (i.e., regarding fairness, does the TREATS workflow outperform the baseline technique?), consider the three fairness metrics evaluated in this work: Bias, TPRP, and PPVP. Concerning Bias, the results demonstrate a significant difference between TREATS and the matching tools (i.e., Ditto and GNEM), which we also proved statistically. Similarly, the results related to TPRP also suggest a significant statistical difference between TREATS and the matching tools for 4 data sources (D1, D2, D3, and D5) out of the 5 data sources evaluated. Only for the data sources D4 (i.e., Walmart-Amazon) there is no statistical difference between them. Regarding PPVP, the results infer similar scores between TREATS and the matching tools, which denote no significant statistical difference. In an overview, the TREATS workflow outperforms the baseline techniques (i.e., Ditto and GNEM) for Bias and TPRP. On the other hand, both achieved similar results for PPVP. Therefore, answering the research question **RQ3**, if the fairness aspect is related to the distribution of data, TREATS workflow outperforms the baseline techniques. If the fairness aspect is related to the hit rate among the groups (i.e., TPRP and PPVP), we can conclude that TREATS yields similar results to baseline techniques in publication-domain scenarios and better results in publication-domain data sources.

4.6. Performance analysis in a streaming scenario

While fairness and effectiveness evaluations are thoroughly examined in previous subsections, here our aim is to address scalability and streaming performance in environments with larger data sources. The main objective is to use a synthetic data source that resembles a streaming scenario to examine how TREATS performs with potentially unbounded data flows in terms of performance. To address concerns regarding the scalability and streaming applicability of our solution, we conducted additional experiments by generating synthetic data using a Generative Adversarial Network (GAN) model [31–33]. Data sources commonly used to evaluate fairness are typically small, which can hide potential challenges related to scalability and streaming performance when handling a large number of increments. Addressing this, GAN was applied to generate larger synthetic data sources with controlled characteristics, preserving the original data distribution while simulating a streaming environment, as highlighted in [31,34]. The intuition behind applying GAN in this context is to generate realistic data variations by subtly modifying parts of the original text. A GAN model, trained on the original dataset, learns the underlying patterns and structures in the data to generate new word substitutions that fit naturally within the original context.

In this evaluation, the D3 data source (Amazon vs. Google) was applied, where the training, test, and validation sets were merged to compose the input for the GAN model. This merging process resulted in a data source containing 11,460 candidate pairs. Based on these initial data, the GAN model generated two synthetic data sources, expanding the base size by factors of 10x and 100x, producing new data sources with 114,600 and 1,146,000 candidate pairs, respectively. The synthetic data was generated by leveraging the ability of the GAN model to capture and replicate the statistical distribution of the original data while introducing subtle modifications. Specifically, the GAN model generated variations in specific fields, such as product titles and attributes, by altering words and phrases in a manner consistent with the original context. This process ensured that the synthetic data sources preserved the coherence and relevance of the original dataset, while simulating variations that resemble real-world scenarios. These modifications enhance the evaluation by creating a more realistic and challenging dataset for testing scalability and streaming performance. The synthetic data sources used in this study, as well as the code used to generate them, are publicly available on the GitHub repository[4].

Similarly to Section 4.4, performance is assessed as data increments arrive over time, using an increment size of 600 entity pairs. Thus, for data sources containing 114,600 and 1,146,000 candidate pairs, this set-up translates to 191 and 1910 increments, respectively. For streaming performance evaluation, the TREATS response time was measured in these increments. Response time is defined as the time that TREATS takes to process a request, from input receipt to response delivery, encompassing the time needed to match and rank candidate pairs. The focus was on observing any latency introduced as the data flow arrives, ensuring that our solution remains robust in streaming scenarios.

The results of the performance evaluation for TREATS, Ditto, and GNEM with incrementally increasing data volumes are illustrated in Figs. 13 and 14. These figures show the response times in multiple increments for the Amazon-Google data source. Fig. 13 represents the 10x synthetic data source (114,600 candidate pairs) over 191 increments, while Fig. 14 represents the 100x synthetic data source (1,146,000 candidate pairs) spanning 1910 increments.

In both figures, Ditto has a faster matching time than GNEM, which confirms the results highlighted in Section 4.4. However, both models achieve a consistent response time within the TREATS workflow. This scenario highlights that TREATS effectively enables each model to manage data streams with stable performance, even as data increments increase. The ranking phase, though shorter than the matching phase, shows a steady overhead added to the overall response time, demonstrating the additional processing required to rank candidates. Overall, the figures reveal that TREATS maintains robust performance across unbounded data flows, with manageable increments in response time as the data arrive. This consistency in response time, despite the increasing data flows, demonstrates the capability of TREATS to handle large-scale data streams efficiently, ensuring stable performance even under demanding streaming conditions.

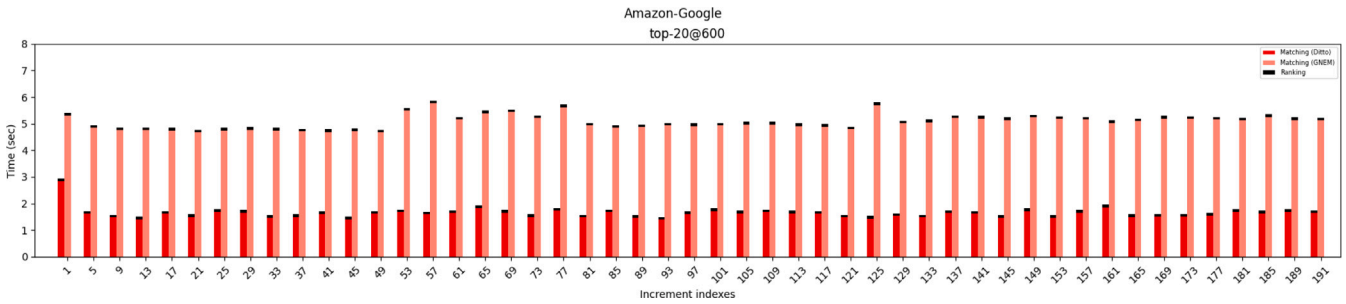


Fig. 13. Amazon vs. Google (increased 10x): Performance results.

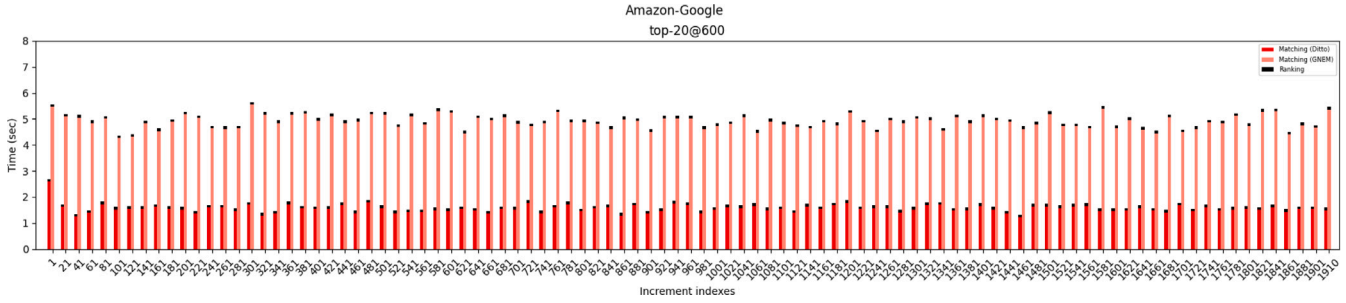


Fig. 14. Amazon vs. Google (increased 100x): Performance results.

5. Related work

Several ER approaches have been proposed to deal with different kinds of data (e.g., schema-homogeneous, schema-heterogeneous, batch, and streaming) benefiting from different strategies to improve effectiveness or/and efficiency, such as stand-alone [35,36], parallel mode [20,37], ML-based [13,26,38], and NLP-based [39,40]. In this work, we propose a parallel workflow through the GPU application that performs the entity comparison by applying an ML-based method. Moreover, we address three research topics involving ER tasks: incremental processing, streaming data, and fairness. Therefore, during this section aspects, challenges, and proposed works related to these topics will be explored.

5.1. Incremental ER

Regarding incremental processing for ER task, previous works [15,41–45] present methodologies focused on relational data sources, which enable the incremental matching of entities. More specifically, [14] also proposes strategies for incrementally processing entities, with a specific emphasis on optimizing the blocking step. These studies introduce an incremental workflow for the ER task, taking into account the evolving nature of data sources to execute the blocking process. The primary objective is to avoid the need for (re)processing the entire dataset during the incremental process, thereby updating deduplication results efficiently. For doing so, various classification techniques are employed to discern duplicate entities. Consequently, these works introduce novel metrics for incremental ER, utilizing collective classification. Additionally, they propose innovative heuristics, incorporating blocking, coverage component filters, and a greedy approach, to further expedite the resolution of incremental record linkage challenges.

In the context of other incremental tasks that offer valuable strategies for ER, notable works include [21,22,46]. These works introduce parallel and incremental strategies to address Name Disambiguation, Event Processing, and Dynamic Graph Processing tasks, respectively. Specifically within the ER domain, [47] presents an incremental approach designed for ER on Social Media data sources. Despite these sources typically supplying heterogeneous data (i.e., do not follow the same schema), the methodology outlined in [47] involves generating an intermediate schema. This intermediate schema ensures that data extracted from the sources adheres to a structured format. Consequently, this approach requires a conversion to a structured format before performing the ER task. Therefore, in contrast to our technique, [47] diverges in its approach by not addressing the challenges posed by heterogeneous data.

While the aforementioned works focus on incremental processing in various domains, it is noteworthy that these studies, except [14], do not explicitly address the intricate challenges associated with fairness aspects and streaming data. While incremental processing is a common thread in these works, the specific nuances related to ensuring fairness in data processing, handling diverse data structures, and managing streaming data are not comprehensively explored. Regarding the work [14], even though the authors provide contributions related to incremental processing for streaming data, they only focus on the blocking step, putting the other steps of the ER process out of the scope.

5.2. ER for streaming data

The incorporation of streaming data frequently introduces numerous challenges, not only within the realm of ER. Illustrated by works such as [48,49], which delve into clustering and itemset frequency, the imperative arises to formulate strategies for the continuous processing of data arriving in brief intervals. While these works may not explicitly address scenarios such as incremental processing and ER approaches, both highlight the essentiality of developing a fitting architecture capable of handling streaming data, as proposed in Section 3.

Considering the ML-based decision-making systems, [50] confronts the evolving challenges posed by real-world streaming applications. Acknowledging the dynamic nature of data generation in real-time and its shifting characteristics, the study highlights the need related to decision-making systems that are not only proficient but also considerate of fairness in the online setting. In contrast to prevalent static approaches to fairness in machine learning, the paper introduces contributions regarding meta-strategy tailored for online stream-based decision-making and the definition of metrics that unify the evaluation of fairness-performance trade-offs. Notice that the main idea detailed in [50] is to address bias in the data stream prior to applying ML

classification algorithms. Overall, the work highlights the complexities of fairness in the context of biased data streams, contributing valuable insights to the ongoing discourse in ML applications. On the other hand, in the context addressed in the present work, ML matchers such Ditto and DeepMatcher are applied during the comparison step to assess attribute values based on pre-trained language models and determine the similarity scores of candidates, which do not provide possible evidence for inserting fairness criteria in their models. Thus, TREATS benefits from the ER workflow for receiving as input the candidates with their similarity scores and, during the fair-matching step, insert the fairness constraints to classify them.

Concerning ER approaches able to handle streaming data, noteworthy contributions are found in works such as [7,51,52]. These approaches outline workflows designed to receive data from streaming sources and execute the ER task. Within these workflows, the authors discuss the application of time windows to eliminate outdated data, thereby averting the excessive consumption of resources such as memory and CPU. However, these works employ time windows solely for the purpose of grouping entities to be processed concurrently. Consequently, they lack consideration for incremental processing, leading to the discarding of previously processed data.

Within the ER context, it is pertinent to focus specifically on the recent contributions presented in [6,14,20]. In the work [20], the authors introduce a parallel-based blocking technique designed to effectively manage streaming data, highlighting the challenges involving streaming data and incremental processing. As an evolution of the work in [20], the work [14] proposes a parallel schema-agnostic blocking technique able to deal with streaming data incrementally, as well as minimize the challenges related to both scenarios. To this end, two main strategies are applied over the blocking: (i) attribute selection algorithm, which discards the superfluous attributes of the entities, to enhance efficiency and minimize resource consumption; and (ii) a top-n promising neighborhood, which maintains only the n most similar entities (i.e., neighbor entities) of each entity, improving efficiency to the proposed technique.

Regarding the optimization of the resource consumption during the streaming processing, the work [6] explores the high-frequency of access to maintain records and blocks on memory. In this sense, this work introduces a randomized ER framework, with focus on the blocking step, that ensures high retention of both the most frequently accessed and recently used blocks in the main memory. Moreover, it adopts a rolling renewal approach for records within a block. Specifically, the probability of retaining inactive blocks and older records in the main memory diminishes, facilitating the accommodation of more promising blocks and fresher records. Although the works [6,14,20] explores scenarios involving streaming data and/or incremental processing, the focus of these works is restricted to the blocking step. Furthermore, the context of fairness constraints is not addressed by the authors.

5.3. Fairness in ER

Given the intricate and nuanced nature of fairness, several definitions and metrics, models, and algorithmic approaches have been proposed, especially explored in recommendation systems works [17,53,54]. Regarding the domain of diversity-aware data summarization, the focus lies on optimizing diversity within a subset of elements from a given set, considering dissimilarities among the chosen elements. The works [11,55] propose algorithms based on max–min diversity maximization for streaming and sliding-window models. Moreover, the authors extend the scope to incorporate fairness constraints, recognizing the increasing importance of mitigating biases associated with sensitive attributes in data summarization. Even though both works do not directly address matching tasks, they explore the trade-off related to diversity maximization and fairness considerations in the streaming context, enriching the understanding and capabilities of these contexts in dynamic and real-world scenarios.

Considering the context of matching tasks, [56] addresses the matching of bipartite graphs with fairness constraints, which has garnered considerable attention in the literature. [56] explores proportional fairness constraints and diversity constraints within graph matching, specifically aiming to maximize the satisfaction of these constraints. The innovation lies in incorporating algorithms rooted in hypergraph matching, with a particular focus on scenarios where each source defines its unique groups of items. Moreover, considering an online setting, where data arrive dynamically over time, adds practical relevance to the proposed algorithms in the sense of performing the matching as the data arrive. To this end, the proposed algorithms prioritize low-degree item matching and employ augmenting paths for unmatched items among sources. Thus, [56] enriches the discourse on graph matching by introducing novel constraints and algorithmic solutions, opening opportunities for further exploration and application of proportional fairness constraints over matching tasks such as ER.

A recent experimental study [3] evaluates the fairness of existing matching methods and introduces new fairness metrics. Among these proposed metrics, we have utilized all those that are applicable in our context, including True Positive Rate Parity (TPRP) and Positive Predictive Value Parity (PPVP), alongside statistical parity, which we had already employed in FairER [19]. Moreover, considering fairness constraints in ER context, [3] introduces the following classification, according to the application of entity attributes:

- Single attribute with binary values: in this context, each entity must be assigned to one of two groups, meaning there are only two options available. For instance, binary value is commonly employed to depict the political preferences of voters in United States elections, namely, whether they align with the Republican or Democratic party;
- Single attribute with multiple exclusive values: each entity must be assigned to exactly one group from a predefined set of groups where each group assumes a key based on the multiple exclusive options. For example, let gender be considered as the sensitive attribute, multiple exclusive values can be assumed by individuals such as male, female, transgender, and non-binary [3]. Thus, these exclusive values will determine the groups;
- Single setwise attribute: each entity should belong to a subset of groups related to the distinct attribute values. For instance, in the context of movies, the genre can be associated with one or more attribute values. This means that a movie may be categorized into genres such as action, drama, comedy, or science fiction. Therefore, the movie can belong to multiple genres, and consequently, belong to more than one group simultaneously;
- Multiple attributes: in this scenario, groups could be either one or a combination of the three cases above. Thus, the group definition could be given by the combination of the sensitive attributes gender (i.e., a single attribute with multiple exclusive values) and movie preference (i.e., movie genre which is a single setwise attribute) of the individuals present in the entity collection.

In [19,57,58], a formulation based on constraints is introduced to address bias in ER tasks, aiming to ensure equal opportunities for all (sub-) groups in the resolution process. To this end, [57] presents a group-based training approach, which learns from data of diverse ethnicities in order to enhance both accuracy and fairness in ML-based ER. [58] introduces metrics for quantifying bias in ER, assessing the risk of record matching across subpopulations. Employing these metrics alongside real-world data, the authors highlight uncover biases within state-of-the-art ER approaches. Additionally, strategies to minimize bias based on data augmentation are suggested with a focus on effectiveness.

As stated in recent works [3,19,58], there is a lack of off-the-shelf studies involving fairness and ER, which also include proper measures, datasets, and comparison angles fitting the context settings. In this sense, the present work relies on the contribution of previous works to explore the fairness context, minimizing the data bias and ensuring

that all groups of entities have similar chances of being resolved. Considering the stated works, which highlight the presence of bias over ER systems, we extend the fairness-aware method that we introduced in FairER [19] to develop a framework able to address bias issues from multiple groups of interest as well as streaming data in an incremental way.

6. Conclusions and further work

In conclusion, this research highlights the fundamental role of Entity Resolution (ER) in addressing the challenges posed by vast and diverse datasets. ER aims to consolidate information from disparate sources, facilitate knowledge base integration, and identify similarities among entities. As demonstrated by its applications in healthcare, airline security, news and media, social networks, and e-commerce, ER transcends various domains, offering substantial benefits for real-world needs. In this paper, we address the challenges involving parallel computing, incremental processing, streaming data, and fairness criteria. To our knowledge, there is a lack of ER approaches that address all these challenges simultaneously.

This paper introduces TREATS, a schema-agnostic and fairness-aware ER workflow designed to handle streaming data incrementally. The proposed fairness-aware ER setting addresses constraints among multiple groups of interest, offering a robust and equitable solution to ER challenges. The experimental evaluation provides insights into the impact of fairness-aware ranking methods on ER systems, contributing to the ongoing discourse on fairness in ER. We extended our evaluation by conducting a performance analysis in a simulated streaming scenario to assess TREATS performance under potentially unbounded data flows. Overall, our contributions aim to advance the field of ER by offering a workflow that considers both technical challenges and ethical considerations.

In future work, we intend to study different kinds of strategies to maintain in main memory (i.e., in the global top−k ranking) the most promising data. In this sense, constraints regarding time, data size, and the trade-off between memory consumption and accuracy may be considered. Therefore, algorithms to renew the records, which are stored in the global structure, can balance the probability of a record remaining or make room for fresher records, without being overwhelmed by historical records. Regarding fairness, we intend to explore the other scenarios related to the application of entity attributes and their respective values: single setwise attributes and multiple attributes, which explore the combination of multiple sensitive attributes and their respective values. This scenario inserts challenges involving how to determine the sensitive attributes (consequently, their values) and how to guarantee equity among the multiple and possible non-exclusive groups of interest.

Furthermore, it is important to highlight the need to balance the accuracy and the efficiency of the ER task especially when streaming data is considered. While our experiments with synthetic data highlight TREATS performance, future work could explore the use of real-time, streaming data sources to gain further insights into its performance in real-world conditions. In terms of ML solutions, one promising direction lies in incremental learning, where the model updates continuously as new data arrive. This approach would allow ML-based matchers to maintain both responsiveness and accuracy in real-time environments, ensuring the model remains adaptive and effective even as data patterns evolve dynamically. In addition, we plan to extend our study by incorporating the blocking step into the proposed workflow. For example, symmetric hash join and its variations are well suited as blocking functions in a streaming context [6,30], efficiently grouping candidate matches on the fly. Using multiple hash functions for blocking reduces false negatives, which benefits the comparison step, and could improve TREATS in terms of efficiency and accuracy in a streaming environment.

CRediT authorship contribution statement

Tiago Brasileiro Araújo: Writing – review & editing, Writing – original draft, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Vasilis Efthymiou:** Writing – review & editing, Writing – original draft, Resources, Investigation, Formal analysis, Conceptualization. **Vassilis Christophides:** Writing – review & editing, Supervision. **Evaggelia Pitoura:** Writing – review & editing, Supervision. **Kostas Stefanidis:** Writing – review & editing, Writing – original draft, Supervision, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Tiago Brasileiro Araujo reports equipment, drugs, or supplies was provided by Federal Institute of Education Science and Technology of Paraiba. Kostas Stefanidis reports a relationship with Tampere University that includes: employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has received funding from the Hellenic Foundation for Research and Innovation (HFRI), Greece and the General Secretariat for Research and Technology (GSRT), Greece, under grant agreement No 969.

Data availability

Data will be made available on request.

References

[1] P. Christen, Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection, Springer Science & Business Media, 2012.
[2] V. Christophides, V. Efthymiou, K. Stefanidis, Entity resolution in the web of data, Synth. Lect. Semant. Web 5 (3) (2015) 1–122.
[3] N. Shahbazi, N. Danevski, F. Nargesian, A. Asudeh, D. Srivastava, Through the fairness lens: Experimental analysis and evaluation of entity matching, Proc. VLDB Endow. 16 (11) (2023) 3279–3292.
[4] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, K. Stefanidis, An overview of end-to-end entity resolution for big data, ACM Comput. Surv. 53 (6) (2021) 127:1–127:42.
[5] G. Papadakis, D. Skoutas, E. Thanos, T. Palpanas, Blocking and filtering techniques for entity resolution: A survey, ACM Comput. Surv. 53 (2) (2020) 1–42.
[6] D. Karapiperis, C. Tjortjis, V.S. Verykios, A randomized blocking structure for streaming record linkage, Proc. VLDB Endow. 16 (11) (2023) 2783–2791.
[7] X. Ren, O. Curé, Strider: A hybrid adaptive distributed RDF stream processing engine, in: International Semantic Web Conference, Springer, 2017, pp. 559–576.
[8] W. Ren, X. Lian, K. Ghazinour, Online topic-aware entity resolution over incomplete data streams, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 1478–1490.
[9] M. Hassani, Overview of efficient clustering methods for high-dimensional big data streams, in: Clustering Methods for Big Data Analytics, Springer, 2019, pp. 25–42.
[10] X.-L. Liu, H.-Z. Wang, J.-Z. Li, H. Gao, EntityManager: Managing dirty data based on entity resolution, J. Comput. Sci. Tech. 32 (3) (2017) 644–662.
[11] Y. Wang, F. Fabbri, M. Mathioudakis, Streaming algorithms for diversity maximization with fairness constraints, in: 2022 IEEE 38th International Conference on Data Engineering, ICDE, IEEE, 2022, pp. 41–53.
[12] R. Chen, Y. Shen, D. Zhang, GNEM: a generic one-to-set neural entity matching framework, in: Proceedings of the Web Conference 2021, 2021, pp. 1686–1694.
[13] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, Proc. VLDB Endow. 14 (1) (2020) 50–60.
[14] T.B. Araújo, K. Stefanidis, C.E.S. Pires, J. Nummenmaa, T.P. da Nóbrega, Incremental entity blocking over heterogeneous streaming data, Information 13 (12) (2022) 568.

[15] D.C. do Nascimento, C.E.S. Pires, D.G. Mestre, Heuristic-based approaches for speeding up incremental record linkage, J. Syst. Softw. 137 (2018) 335–354.

[16] M. Dragoni, M. Federici, A. Rexha, An unsupervised aspect extraction strategy for monitoring real-time reviews stream, Inf. Process. Manag. 56 (3) (2019) 1103–1118.

[17] E. Pitoura, K. Stefanidis, G. Koutrika, Fairness in rankings and recommendations: an overview, VLDB J. (2022) 1–28.

[18] K. Makhlouf, S. Zhioua, C. Palamidessi, On the applicability of ML fairness notions, 2020, arXiv preprint arXiv:2006.16745.

[19] V. Efthymiou, K. Stefanidis, E. Pitoura, V. Christophides, FairER: entity resolution with fairness constraints, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 3004–3008.

[20] T.B. Araújo, K. Stefanidis, C.E. Santos Pires, J. Nummenmaa, T.P. da Nóbrega, Schema-agnostic blocking for streaming data, in: Proceedings of the 35th Annual ACM Symposium on Applied Computing, SAC '20, 2020, pp. 412–419.

[21] M. Körber, N. Glombiewski, A. Morgen, B. Seeger, TPStream: low-latency and high-throughput temporal pattern matching on event streams, Distrib. Parallel Databases (2019) 1–52.

[22] A.F. Santana, M.A. Gonçalves, A.H. Laender, A.A. Ferreira, Incremental author name disambiguation by exploiting domain-specific heuristics, J. Assoc. Inf. Sci. Technol. 68 (4) (2017) 931–945.

[23] G. Papadakis, V. Efthymiou, E. Thanos, O. Hassanzadeh, P. Christen, An analysis of one-to-one matching algorithms for entity resolution, VLDB J. 32 (6) (2023) 1369–1400.

[24] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, Z. Ghahramani, SIGMa: simple greedy matching for aligning large knowledge bases, in: KDD, 2013, pp. 572–580.

[25] S. Delecraz, L. Eltarr, M. Becuwe, H. Bouxin, N. Boutin, O. Oullier, Making recruitment more inclusive: unfairness monitoring with a job matching machine-learning algorithm, in: Proceedings of the 2nd International Workshop on Equitable Data and Technology, 2022, pp. 34–41.

[26] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: A design space exploration, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 19–34.

[27] J. MESTS, M. Tang, Distributed representations of tuples for entity resolution, Proc. VLDB Endow. 11 (11) (2018).

[28] P. Zikopoulos, C. Eaton, et al., Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data, McGraw-Hill Osborne Media, 2011.

[29] J. Kreps, N. Narkhede, J. Rao, et al., Kafka: A distributed messaging system for log processing, in: Proceedings of the NetDB, Vol. 11, 2011, pp. 1–7.

[30] D. Karapiperis, C. Tjortjis, V.S. Verykios, A suite of efficient randomized algorithms for streaming record linkage, IEEE Trans. Knowl. Data Eng. (2024).

[31] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, A.A. Bharath, Generative adversarial networks: An overview, IEEE Signal Process. Mag. 35 (1) (2018) 53–65.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.

[33] A. Aggarwal, M. Mittal, G. Battineni, Generative adversarial network: An overview of theory and applications, Int. J. Inf. Manag. Data Insights 1 (1) (2021) 100004.

[34] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, Y. Kim, Data synthesis based on generative adversarial networks, Proc. VLDB Endow. 11 (10) (2018) 1071–1083.

[35] G. Papadakis, G. Papastefanatos, T. Palpanas, M. Koubarakis, Scaling Entity Resolution to Large, Heterogeneous Data with Enhanced Meta-Blocking, EDBT, 2016.

[36] T.B. Araújo, C.E.S. Pires, D.G. Mestre, T.P.d. Nóbrega, D.C.d. Nascimento, K. Stefanidis, A noise tolerant and schema-agnostic blocking technique for entity resolution, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, ACM, 2019, pp. 422–430.

[37] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, T. Palpanas, Parallel meta-blocking for scaling entity resolution over big heterogeneous data, Inf. Syst. 65 (2017) 137–157.

[38] N. Barlaug, J.A. Gulla, Neural networks for entity matching: A survey, ACM Trans. Knowl. Discov. Data (TKDD) 15 (3) (2021) 1–37.

[39] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.

[40] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, A. Doan, Deep learning for blocking in entity matching: a design space exploration, Proc. VLDB Endow. 14 (11) (2021) 2459–2472.

[41] A. Gruenheid, X.L. Dong, D. Srivastava, Incremental record linkage, Proc. VLDB Endow. 7 (9) (2014) 697–708.

[42] M. Nentwig, E. Rahm, Incremental clustering on linked data, in: 2018 IEEE International Conference on Data Mining Workshops, ICDMW, IEEE, 2018, pp. 531–538.

[43] A. Saeedi, E. Peukert, E. Rahm, Incremental multi-source entity resolution for knowledge graph completion, in: European Semantic Web Conference, Springer, 2020, pp. 393–408.

[44] G. Simonini, L. Zecchini, S. Bergamaschi, F. Naumann, Entity resolution on-demand, Proc. VLDB Endow. 15 (7) (2022) 1506–1518.

[45] L. Zecchini, G. Simonini, S. Bergamaschi, F. Naumann, BrewER: Entity resolution on-demand, Proc. VLDB Endow. 16 (12) (2023) 4026–4029.

[46] W. Ju, J. Li, W. Yu, R. Zhang, Igraph: an incremental data processing system for dynamic graph, Front. Comput. Sci. 10 (3) (2016) 462–476.

[47] B. Opitz, T. Sztyler, M. Jess, F. Knip, C. Bikar, B. Pfister, A. Scherp, An approach for incremental entity resolution at the example of social media data, in: AIMashup@ ESWC, Citeseer, 2014.

[48] F. Ao, Y. Yan, J. Huang, K. Huang, Mining maximal frequent itemsets in data streams based on fp-tree, in: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, 2007, pp. 479–489.

[49] A. Kumar, A. Singh, R. Singh, An efficient hybrid-clustream algorithm for stream mining, in: 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems, SITIS, IEEE, 2017, pp. 430–437.

[50] Z. Wang, N. Saxena, T. Yu, S. Karki, T. Zetty, I. Haque, S. Zhou, D. Kc, I. Stockwell, X. Wang, et al., Preventing discriminatory decision-making in evolving data streams, in: Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency, 2023, pp. 149–159.

[51] T. Kim, M.-N. Hwang, Y.-M. Kim, D.-H. Jeong, Entity resolution approach of data stream management systems, Wirel. Pers. Commun. 91 (4) (2016) 1621–1634.

[52] K. Ma, B. Yang, Stream-based live entity resolution approach with adaptive duplicate count strategy, Int. J. Web Grid Serv. 13 (3) (2017) 351–373.

[53] Y. Wang, W. Ma, M. Zhang, Y. Liu, S. Ma, A survey on the fairness of recommender systems, ACM Trans. Inf. Syst. 41 (3) (2023) 1–43.

[54] G. Giannopoulos, G. Papastefanatos, D. Sacharidis, K. Stefanidis, Interactivity, fairness and explanations in recommendations, in: Adjunct Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization, 2021, pp. 157–161.

[55] Y. Wang, F. Fabbri, M. Mathioudakis, J. Li, Fair max–min diversity maximization in streaming and sliding-window models, Entropy 25 (7) (2023) 1066.

[56] G.S. Sankar, A. Louis, M. Nasre, P. Nimbhorkar, Online algorithms for matchings with proportional fairness constraints and diversity constraints, in: European Conference on Artificial Intelligence, 2022.

[57] C. Makri, A. Karakasidis, E. Pitoura, Towards a more accurate and fair SVM-based record linkage, in: 2022 IEEE International Conference on Big Data, Big Data, IEEE, 2022, pp. 4691–4699.

[58] S. Nilforoushan, Q. Wu, M. Milani, Entity matching with AUC-based fairness, in: 2022 IEEE International Conference on Big Data, Big Data, IEEE, 2022, pp. 5068–5075.