
# RadlER: Deduplicated Sampling On-Demand

Luca Zecchini
BIFOLD & TU Berlin
Berlin, Germany
luca.zecchini@tu-berlin.de

Ziawasch Abedjan
BIFOLD & TU Berlin
Berlin, Germany
abedjan@tu-berlin.de

Vasilis Efthymiou
Harokopio University
Athens, Greece
vefthym@hua.gr

Giovanni Simonini
University of Modena
and Reggio Emilia, Italy
simonini@unimore.it

## ABSTRACT

Data practitioners often need to sample their datasets to produce representative subsets for their downstream tasks. Unfortunately, real-world datasets frequently contain duplicates, whose presence biases sampling and impacts the quality of the produced subsets, hence the outcome of downstream tasks. While deduplication is therefore fundamental, performing it on the entire dataset to run sampling on its cleaned version might be prohibitively expensive in terms of time and resources. Thus, we recently introduced RadlER, a solution to perform *deduplicated sampling on-demand*, i.e., to produce a clean sample of a dirty dataset incrementally, according to a target distribution of some subpopulations, by focusing the cleaning effort only on entities required to appear in the sample.

In this demonstration, we interactively show how RadlER can support practitioners in their data science pipelines, allowing them to save a relevant amount of time and resources.

**PVLDB Reference Format:**
Luca Zecchini, Ziawasch Abedjan, Vasilis Efthymiou, and Giovanni Simonini. RadlER: Deduplicated Sampling On-Demand. PVLDB, 18(12): 5319 - 5322, 2025.
doi:10.14778/3750601.3750661

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

## 1 DEDUPLICATED SAMPLING ON-DEMAND

Data practitioners often need to sample their datasets to produce representative subsets for their downstream tasks, such as data analysis or the training of machine learning models. A common approach is *stratified sampling* [12], where the described entities are partitioned into multiple distinct subpopulations (i.e., *groups*), based on the value presented for one or more attributes. Sampling is then performed independently for each group to ensure that the produced sample follows a *target distribution* — e.g., selecting the same number of entities for each group. A proper representation of the groups in the sample is fundamental to prevent the insurgence of *bias* [6], which might lead to discriminatory behaviors.

Unfortunately, real data often presents quality issues [4], which can impact the effectiveness of sampling and therefore the outcome of downstream tasks, jeopardizing for instance the correctness of data analytics used in decision making processes. A major challenge is the presence of *duplicates*, i.e., multiple (possibly inconsistent) representations of the same real-world entity. Thus, *deduplication* (a.k.a. entity resolution or record linkage) [2] is needed to detect such *matching records* and merge them into a single *consolidated record* [3] representative of the described entity — for simplicity, we denote this process as the *cleaning* of an entity. To prevent the issues caused by duplicates, we need therefore to produce a *clean sample*, i.e., a sample only composed of consolidated records. We define the task of producing a clean sample of a dirty dataset according to a target distribution as *deduplicated sampling* [13].

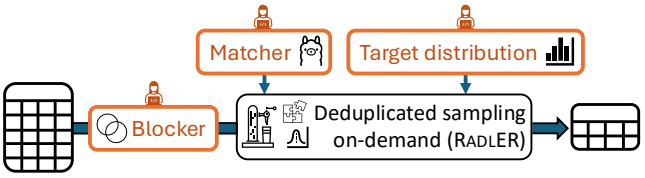


**Figure 1: Deduplicated sampling on-demand with RadlER.**

The naïve approach to deduplicated sampling requires deduplicating the entire dataset upfront, then producing the sample from the obtained clean dataset. Yet, accurate deduplication does not come without a price, especially when it relies on state-of-the-art solutions based on deep learning [1] — including large language models [8] — to compare candidate matching records. Thus, it can be an expensive process in terms of time, computational resources, and even money, making the naïve approach often prohibitive.

In the wake of the *on-demand* paradigm for deduplication [10, 15], previously proposed to produce clean results for queries issued on dirty datasets, we recently introduced *deduplicated sampling on-demand* [13], implemented through RadlER[1] (Figure 1). RadlER produces a clean sample following a target distribution incrementally, by focusing the cleaning effort on a single entity at a time — hence limiting that effort to the entities that appear in the sample, instead of cleaning the entire dataset upfront. Thus, RadlER significantly decreases the number of comparisons required to produce the clean sample, saving a relevant amount of time — allowing practitioners to comply with their time constraints — and computational resources — with a strong reduction of monetary costs (e.g., to execute calls through the API of a large language model) and environmental impact [11].

After providing an overview of RadlER in Section 2, in Section 3 we demonstrate its benefits in various scenarios, e.g., to quickly produce clean samples of dirty datasets according to a target distribution (Section 3.1), to perform an early assessment of group


Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750661

[1] Radler is a beverage obtained as a mix of beer and lemonade in variable proportions. Similarly, RadlER (with ER standing for entity resolution) produces samples by mixing different groups of entities according to the distribution required by the user.

fairness for the selected entity matching function [9] (Section 3.2), or to reduce the number of matching errors (Section 3.3).

## 2 A SIP OF RADLER

In this section, we present a brief overview of RadlER, our novel solution to perform deduplicated sampling on-demand, to provide the reader with some basic intuitions about its functioning. All details are provided in our research paper [13].

Let us consider a dataset $\mathcal{D}$ that contains duplicates — we call it therefore a *dirty dataset*. Deduplicated sampling aims to produce a sample $\mathcal{S}$ of $\mathcal{D}$ that is *clean* (i.e., only composed of consolidated records) and *undistorted* with respect to a target distribution $d$, defined over a set of disjoint groups $\Gamma$. The groups in $\Gamma$ partition (a subset of) the entities represented in $\mathcal{D}$ based on one or more categorical *sampling attributes* $\mathcal{A}_\Gamma$. For instance, if $\mathcal{A}_\Gamma$ = {gender, status}, $\Gamma$ = {{gender: "female", status: "single"}, ..., {gender: "male", status: "married"}}. The target distribution $d = [p(\gamma), \forall \gamma \in \Gamma]$, with $\sum_{i=1}^{|\Gamma|} d_i = 1$, determines the number of entities required to appear in $\mathcal{S}$ for each group $\gamma \in \Gamma$. A sample $\mathcal{S}$ is undistorted with respect to $d$ if the distribution $d_\mathcal{S}$ of its records over $\Gamma$ has the minimum divergence from $d$ among all possible samples of size $|\mathcal{S}|$, where $divergence(d, d') = \sum_{i=1}^{|\Gamma|} |d_i - d'_i|$.

RadlER performs deduplicated sampling by focusing the cleaning effort on a single entity at a time, producing the clean sample $\mathcal{S}$ incrementally. Thus, it follows the *on-demand* paradigm for deduplication [10]. Deduplicated sampling on-demand can be considered as an iterative process, which cleans at each iteration $\tau$ a random entity $\varepsilon_\tau$ belonging to a group $\gamma \in \Gamma$ that allows to maintain $\mathcal{S}$ undistorted with respect to $d$ — we call it therefore the *target group* $\hat{\gamma}_\tau$ for the iteration $\tau$. In principle, RadlER would produce the largest possible clean sample $\mathcal{S}$ undistorted with respect to $d$. However, its incremental nature inherently supports *early stopping* and *stop-and-resume* execution. Thus, the user can stop the process arbitrarily at any moment, or even define stopping criteria — based for instance on the sample size or the number of performed comparisons.

Since the selection of the entity to clean at iteration $\tau$ has to be performed on the original records in $\mathcal{D}$, for every record $r \in \mathcal{D}$ we need to know to which groups the entity $\varepsilon_r$ that it describes *might* belong. To this end, we maintain a hash table $\mathcal{G}$ that tracks for every group $\gamma \in \Gamma$ — using an inner hash table $\mathcal{G}[\gamma]$ — the records that might describe an entity from that group. Within $\mathcal{G}[\gamma]$, a record $r$ is associated to an updatable weight $\omega_\gamma^r \in (0, 1]$. Note that $\omega_\gamma^r = 0$ implies that for sure $\varepsilon_r$ does not belong to $\gamma$, hence $r$ does not appear in $\mathcal{G}[\gamma]$ in that case. The weight $\omega_\gamma^r$ represents the tradeoff between two components: *(i)* the probability that $\varepsilon_r$ belongs to $\gamma$, i.e., the *benefit* of cleaning $\varepsilon_r$; *(ii)* the estimated number of comparisons required to clean $\varepsilon_r$, i.e., its *cost*.

Both components are computed by taking into account the record $r$ and its *neighbors*, i.e., its candidate matches previously detected through a *blocking* function [7], towards which RadlER is agnostic. In particular, we consider the number of neighbors and the values they present for the sampling attributes to determine the cost and the benefit components, respectively. Beyond reducing the number of performed comparisons, hence time and resources required to produce the clean sample $\mathcal{S}$, the cost component mitigates the bias introduced by the presence of duplicates, which distorts sampling by favoring entities represented by multiple records — as their probability of being selected for cleaning and therefore to appear in the produced clean sample would be higher.

At the beginning of iteration $\tau$, RadlER needs to detect the target group $\hat{\gamma}_\tau$ to which the cleaned entity $\varepsilon_\tau$ should belong to maintain the clean sample $\mathcal{S}$ undistorted. If no more entities remain to clean for $\hat{\gamma}_\tau$, the process terminates. To select the entity $\varepsilon_\tau$ to clean, RadlER picks a record from $\mathcal{G}[\hat{\gamma}_\tau]$ — the hash table in $\mathcal{G}$ that stores those records that might describe an entity belonging to $\hat{\gamma}_\tau$ (each associated to the weight computed for that group) — through a *weighted random selection*.

As the selected record drives the cleaning process at iteration $\tau$, we denote it as the *pivot record* $p_\tau$. If $p_\tau$ presents some neighbors, we use the selected *matching function* — e.g., a trained binary classifier [1] — to determine whether it describes the same entity as $p_\tau$, detecting the cluster of matching records that are then merged to produce the consolidated record for the entity $\varepsilon_\tau$ [3]. Note that RadlER is agnostic towards all deduplication functions (namely the blocking function, the matching function, and the aggregation functions used to produce the consolidated record), as it can operate with any function selected or defined by the user.

The pivot record $p_\tau$ and its matches are removed from the hash tables in $\mathcal{G}$ — as they are now represented by the cleaned entity $\varepsilon_\tau$, whose actual group $\gamma_\tau$ is finally known. If $\gamma_\tau = \hat{\gamma}_\tau$, then $\varepsilon_\tau$ is inserted into the clean sample $\mathcal{S}$; otherwise, it is inserted into $\mathcal{G}[\gamma_\tau]$ with the maximum weight of 1, to allow its selection in some subsequent iteration. For each record that was compared to $p_\tau$ but turned out to describe a different entity, its neighborhood is updated by removing $p_\tau$ and its weights are recomputed consequently, revising therefore its occurrences in $\mathcal{G}$.

RadlER terminates its iterations as soon as the stopping criterion defined by the user is satisfied (e.g., the clean sample $\mathcal{S}$ has reached the required size) or no more entities can be cleaned while maintaining $\mathcal{S}$ undistorted with respect to the target distribution $d$, as described above. Further, the user can decide to interrupt — and possibly resume — the process at any time during its execution.

## 3 DEMONSTRATION SCENARIOS

In this section, we present the scenarios that we will address in the demonstration of RadlER. As shown in Figure 2 and in our video[2], the simple and user-friendly interface of our web application — built with Streamlit[3] and running locally on a laptop — aims to favor interaction from the audience. After introducing attendees to the scenarios through some examples, we will encourage them to interact with RadlER, exploring its functionalities under different settings and highlighting its benefits over the naïve approach to deduplicated sampling, expressed in terms of required comparisons, time, and monetary costs.

The settings of our demonstration can be customized with regard to several dimensions. First, we provide multiple datasets with heterogeneous features (e.g., number of records and attributes, average cluster size, etc.). We cover all datasets that were used in our experiments [13] — describing commercial products (cameras), organizations, and people (registered voters). Further, we include

[2]https://youtu.be/Eeswx1ucvcs
[3]https://streamlit.io

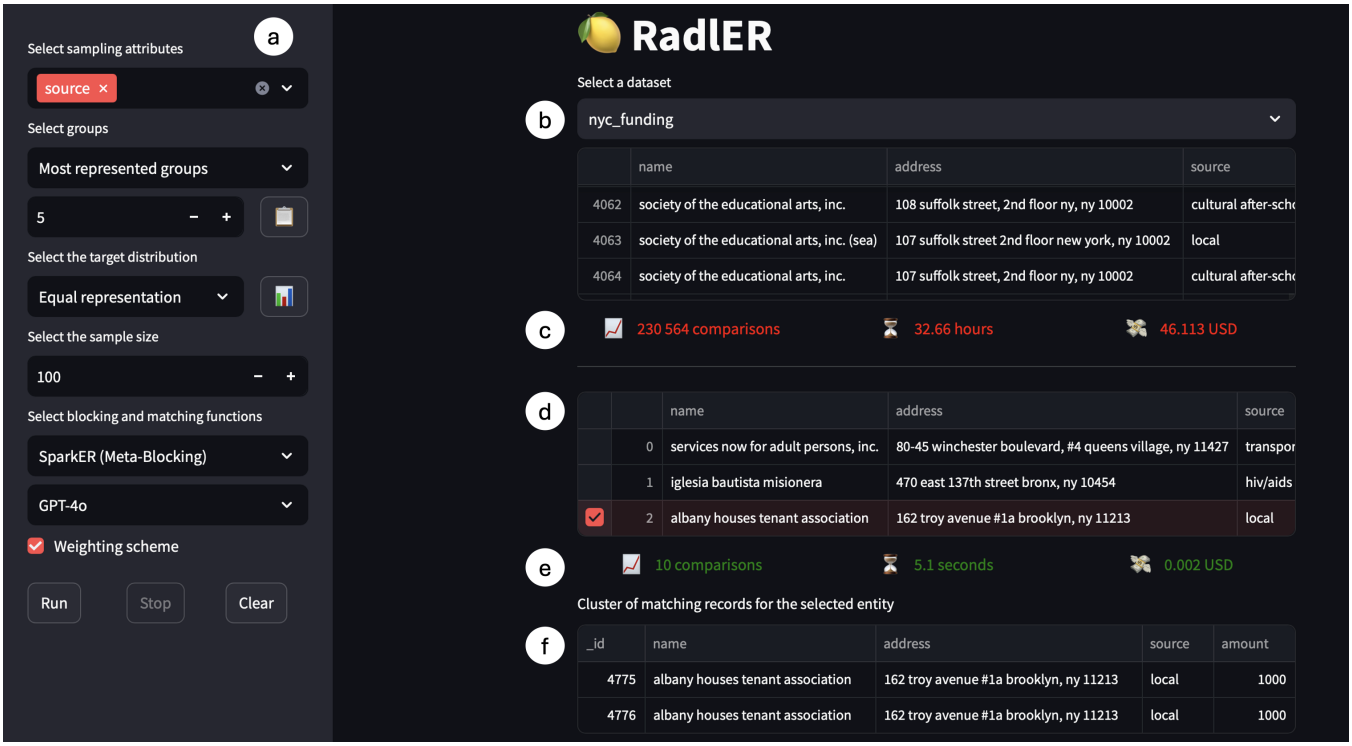


**Figure 2: Demonstration of RadlER.**

other datasets from additional domains, collected for instance from the Magellan Data Repository[4]. Secondly, we also allow to specify the groups to consider — focusing on the $k$ most represented ones or some specific groups of interest — and the target distribution, supporting *equal representation* (i.e., all groups represented by the same number of entities), *demographic parity* (i.e., all groups represented proportionally as in a reference population), or even the definition of a custom one. Finally, it is possible to choose among multiple blocking and matching functions with different accuracy, showing their impact on the deduplication process.

## 3.1 Scenario 1: Generating Balanced Samples

*Ellen, a data scientist, is required to do a follow-up study on how organizations allocate discretionary New York City Council funding across different types of initiatives. She has strict time constraints to prepare a report and the funding information can be found as open data in the dataset shown in Figure 2b, where the 16.3k initiatives are categorized by a source attribute. The analysis of every initiative requires acquiring financial reports from organizations and significant work to prepare the data, hence she cannot use the entire dataset. She opts therefore for limiting the analysis to 100 initiatives from distinct organizations, selected with stratified sampling on the source.*

*Unfortunately, the dataset contains duplicates (Figure 2b), as most organizations are represented by multiple records — which often report inconsistent information. Sampling the dirty dataset would favor the selection of entities with more duplicates, introducing bias; thus, deduplication is needed. An advanced meta-blocking framework [5] returns 231k candidate pairs of duplicate organizations to Ellen, who wants to process them through the API of a large language model [8]. Yet, comparing all of them for cleaning the entire dataset — just to sample 100 entities out of it — would require about 33 hours (Figure 2c), with an expense of $46.11, making the naïve approach prohibitive for her time constraints.*

*To comply with her requirements, Ellen uses RadlER to produce the desired clean sample using the selected blocking and matching functions without cleaning the entire data upfront. As shown in Figure 2a, she demands a maximum sample size of 100 entities, produced by following the equal representation of the 5 most represented groups. When she clicks on the Run button, RadlER starts cleaning and emitting the cleaned entities one by one (Figure 2d), maintaining the clean sample undistorted with respect to the target distribution at each emission. Ellen can stop the process at any time, for instance to inspect the cleaned entities with the possibility of checking which records were merged to produce each consolidated record (Figure 2f), understanding for instance why it carries a certain value for a specific attribute. Then, she can resume the process if she needs more entities. With RadlER, producing the clean sample of 100 entities would require just a few minutes — for instance, the three cleaned entities shown in Figure 2d are made available after only 5 seconds (Figure 2e). Ellen is happy and she can proceed with her analysis.*

In this scenario, after introducing the audience to RadlER and its features through a quick example, attendees will be able to play with the different parameters — by varying dataset, groups, target

[4]https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository

distribution, blocking and matching function — to see how RadlER works under their settings of interest and assess its benefits in terms of saved time and resources.

### 3.2 Scenario 2: Assessing Matching Fairness

*Anna, a data scientist, is analyzing the data of the registered North Carolina voters to present a report to a committee of policymakers. The data used in her analysis has to be reliable, hence the legitimacy of selected entries has to be cross-referenced with proprietary tools that access Social Security Administration databases, involving a tedious manual process through a web portal. Unfortunately, her dataset is dirty, as a single voter is often represented by multiple records, which might provide inconsistent information. To minimize manual labor while ensuring a representative dataset composed of consolidated records, Anna uses RadlER to produce a clean sample of the dirty dataset that preserves the distribution of key features — such as the postcode — across the population of the state.*

*Using a deep learning classifier pre-trained for deduplication on US document corpora as the matching function, Anna reconciles the records that are considered to describe the same person before performing a careful manual verification. However, while inspecting the resulting entities as they are progressively emitted, she notices significantly higher error rates for certain postcodes. In particular, this happens for postcodes where the rate of residents belonging to some specific ethnic minorities — whose names are less common in the US — is higher [9]. Further investigations reveal that the pre-trained model struggles with those minorities due to a lack of representativeness in the training data [6], leading to poorer deduplication accuracy for some areas. By leveraging RadlER to produce a balanced clean sample of her dataset incrementally, Anna is able to uncover and address this bias early — the naïve approach to deduplicated sampling would have made cleaned entities available for inspection only after completing the deduplication of the entire dataset. This highlights the importance of evaluating the outcome of pre-trained entity matching functions across diverse subpopulations to ensure accuracy and fairness in high-risk applications such as voter verification [9].*

In this scenario, we will provide attendees with matching functions presenting accuracy issues for entities from some specific groups. We will show how RadlER can support practitioners in the early detection of such systematic issues through the inspection of the clusters of matches from which the consolidated records in the clean sample were produced (Figure 2f).

### 3.3 Scenario 3: Reducing Matching Errors

*Claire, a data scientist at a company selling consumer electronics, is requested to train a machine learning model to predict the cost of cameras for sale. She needs to produce a training set for that task, which has to be balanced across the different brands sold by her company. In particular, as she previously collected about 30k advertisements from several e-commerce websites, Claire wants to produce the largest possible sample with the equal representation of those brands. Unfortunately, her dataset is dirty, as the same camera model is usually described by multiple advertisements. Detecting matching cameras can be a hard task, which often requires the knowledge of very specific brand-based patterns [14]. Thus, her pre-trained matcher is likely to run into some errors, which of course she wants to minimize. If Claire decided to select the entities to clean by randomly picking records from the dirty dataset, the produced clean sample would be biased in favor of entities represented by more records — as their probability of being selected is higher [13]. Larger clusters of records require the matcher to perform more comparisons, thereby increasing (in absolute terms) the number of errors that it can potentially commit.*

*Luckily, Claire can rely on RadlER, which addresses this issue through its weighting scheme. Indeed, the cost component contributes to the weight of a record inversely proportional to the size of its neighborhood, mitigating the bias determined by the presence of duplicates. RadlER focuses the cleaning effort on entities represented by smaller clusters of records on average, which require therefore fewer comparisons. Thus, beyond producing the clean sample faster, the number of possible errors is also reduced.*

In this scenario, attendees will assess the impact of the weighting scheme adopted by RadlER, comparing it to a naïve on-demand baseline operating through random selection of records — i.e., the *random* baseline evaluated in our research paper [13]. RadlER requires significantly fewer comparisons to produce a sample of the same size, saving time and resources, and the cleaned entities are generated from smaller clusters of records on average. To this end, users can simply choose to enable or disable the use of weights through a dedicated select widget (Figure 2a).

In conclusion, the proposed scenarios will provide the audience with an interactive demonstration to intuitively illustrate the goals of deduplicated sampling and the functioning of RadlER, involving through its user-friendly and intuitive interface also attendees who are less familiar with the topic of deduplication.

## ACKNOWLEDGEMENTS

This work was partially supported by MUR within the PRIN "Discount Quality" project (code 202248FWFS) and by the EU Horizon Europe program within the "Ceasefire" project (GA no. 101073876).

## REFERENCES

[1] Nils Barlaug and Jon Atle Gulla. 2021. Neural Networks for Entity Matching: A Survey. *TKDD* 15, 3, Article 52 (2021), 37 pages.
[2] Peter Christen. 2012. *Data Matching*. Springer.
[3] Dong Deng et al. 2019. Unsupervised String Transformation Learning for Entity Consolidation. In *ICDE*. 196–207.
[4] Wenfei Fan. 2015. Data Quality: From Theory to Practice. *SIGMOD Record* 44, 3 (2015), 7–18.
[5] Luca Gagliardelli, Giovanni Simonini, Domenico Beneventano, and Sonia Bergamaschi. 2019. SparkER: Scaling Entity Resolution in Spark. In *EDBT*. 602–605.
[6] Ninareh Mehrabi et al. 2021. A Survey on Bias and Fairness in Machine Learning. *CSUR* 54, 6, Article 115 (2021), 35 pages.
[7] George Papadakis et al. 2020. Blocking and Filtering Techniques for Entity Resolution: A Survey. *CSUR* 53, 2, Article 31 (2020), 42 pages.
[8] Ralph Peeters, Aaron Steiner, and Christian Bizer. 2025. Entity Matching using Large Language Models. In *EDBT*. 529–541.
[9] Nima Shahbazi et al. 2023. Through the Fairness Lens: Experimental Analysis and Evaluation of Entity Matching. *PVLDB* 16, 11 (2023), 3279–3292.
[10] Giovanni Simonini, Luca Zecchini, Sonia Bergamaschi, and Felix Naumann. 2022. Entity Resolution On-Demand. *PVLDB* 15, 7 (2022), 1506–1518.
[11] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *ACL*. 3645–3650.
[12] Steven K. Thompson. 2012. *Sampling*. John Wiley & Sons.
[13] Luca Zecchini, Vasilis Efthymiou, Felix Naumann, and Giovanni Simonini. 2025. Deduplicated Sampling On-Demand. *PVLDB* 18, 8 (2025), 2482–2495.
[14] Luca Zecchini, Giovanni Simonini, and Sonia Bergamaschi. 2020. Entity Resolution on Camera Records without Machine Learning. In *DI2KG @ VLDB*.
[15] Luca Zecchini, Giovanni Simonini, Sonia Bergamaschi, and Felix Naumann. 2023. BrewER: Entity Resolution On-Demand. *PVLDB* 16, 12 (2023), 4026–4029.