covRew: a Python Toolkit for Pre-Processing Pipeline Rewriting Ensuring Coverage Constraint Satisfaction

Demonstration Paper

Chiara Accinelli, Barbara Catania, Giovanna Guerrini, Simone Minisi

DIBRIS - University of Genoa, Genoa - Italy

name.surname@dibris.unige.it

ABSTRACT

This demo presents covRew, a Python toolkit for rewriting slicing operations in pre-processing pipelines (i.e., pipelines to be executed before further tasks, such as data analytics and machine learning) so that the pipeline execution ensures that protected groups are adequately represented (i.e., *covered*) in the result. The toolkit includes: (i) an analyzer, which identifies candidate operations for rewriting; (ii) a rewriter, which transforms operations for ensuring coverage satisfaction with respect to user specified constraints; (iii) an impact evaluator, allowing the user to assess the impact of the rewriting on the obtained results.

1 INTRODUCTION

One of the main current challenges in data processing is the development of technological solutions satisfying non-discriminating requirements. Themes such as diversity, non-discrimination, fairness, protection of minorities, and transparency are increasingly crucial when processing and analyzing data.

Analytical pipelines processing real data are often very complex and various systems have been designed for supporting the user in the design and the execution of processing pipelines in a non-discriminating way. Among them, we recall: Fair-DAGs [11], an open-source library aiming at representing data processing pipelines in terms of a directed acyclic graph (DAG) and identifying distortions with respect to protected groups as the data flows through the pipeline; FairPrep [10], an environment for investigating the impact of fairness-enhancing interventions inside data processing pipelines; AI Fairness 360 [5], an open-source Python toolkit for algorithmic fairness, aimed at facilitating the transition of fairness-aware research algorithms to usage in an industrial setting and at providing a common framework to share and evaluate algorithms.

The complexity of data processing pipelines does not only depend on the used analytical or learning tasks but also on the types of pre-processing operations applied to input datasets for filtering, projecting (thus slicing), or merging together input objects. Indeed, it has been recognized that data pre-processing tasks can introduce bias at different levels [10]. As an example, classical data transformation operations, often defined in terms of Selection-Projection-Join (SPJ) operations over tabular data, can reduce the number of records related to some protected or disadvantaged groups, defined in terms of some sensitive attributes, even if such attributes are not directly used in the specification of the data transformation operation. As a consequence, some protected or disadvantaged categories can be under-represented in

(*uncovered by*) the result of a transformation, possibly introducing bias in the following analytical steps.

As already recognized [3, 11], we believe that it is important to support the user in the design of non-discriminating data pre-processing tasks, with a special reference to data transformations defined in terms of slicing and merge operations. Additionally, following what stated in [9], we believe that such design can be improved by the usage of specific diversity or fairness-aware data transformations, where the idea is to *minimally* rewrite the transformation operation so that certain non-discrimination constraints are guaranteed to be satisfied in the transformation result. Through minimal rewriting, the revised process takes into account the original transformation goals and is traced for further processing, thus guaranteeing *transparency*.

Starting from these considerations, in our recent work [2], we designed an approach for minimally rewriting slicing and merge operations with the aim of satisfying specific *coverage constraints* [4, 6], guaranteeing that there are enough entries related to specific protected groups of interest in the result obtained by applying a given transformation, thus increasing diversity with the aim of limiting the introduction of bias during the next analytical steps. The problem we address is closely related to [3], where a constraint-based optimization approach is proposed for identifying a filter-based transformation generating a dataset satisfying a given input set of soft constraints. However, differently from [3], we consider the rewriting of transformations corresponding to SPJ queries with the aim of satisfying (hard) coverage constraints through a rewriting approach that can be easily integrated inside a pre-processing pipeline.

The aim of this demonstration is to showcase the techniques proposed in our recent work [2] by presenting covRew, a Python toolkit for rewriting data transformations specified inside pipelines described in Pandas [7], ensuring the satisfaction of a set of coverage constraints provided in input. The toolkit includes: (i) a *pipeline analyzer*, which identifies candidate operations for rewriting, (ii) a *pipeline rewriter*, which transforms operations that are selected by the user according to the input coverage constraints, and (iii) an *impact evaluator*, assessing the impact of the rewriting of the selected operations by comparing the result of the execution of the rewritten pipeline with that of the original one, according to the solution-based accuracy measures proposed in [1]. We showcase our toolkit in action with various scenarios on real-world datasets, demonstrating its usability in coverage constraint enforcing and the provided support for analyzing the impact of coverage-based rewriting. Notice that, though the approach supports merge operations and works on multiple datasets, in the demonstration, for the sake of simplicity, we will rely on a single dataset and will not consider merge operations. This way, indeed, the analysis of the impact of the proposed rewriting on the obtained results is clearer.
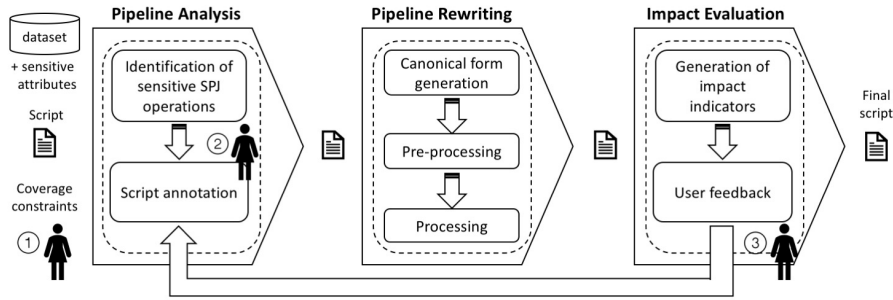
Figure 1: covRew flow

The remainder of the paper is structured as follows. In Section 2, we present the covRew architecture, illustrating the techniques underlying each step. In Section 3, we present the demonstration scenarios, with a special reference to the user interaction. Section 4 concludes and outlines some future work directions.

2 OVERVIEW

covRew is a Python toolkit focusing on the rewriting of data preprocessing pipelines, with the aim of satisfying specific coverage constraints on the results of slicing operations. In the following, we describe the characteristics of input data, processing pipelines, and the main covRew tasks. The covRew logical architecture is illustrated in Figure 1.

2.1 Input specification

Input data for covRew are: a dataset, with the related sensitive attribute specification, for the identification of protected groups; a processing pipeline represented as a Pandas script; a set of coverage constraints.

Dataset. covRew takes as input a tabular dataset I, represented as a Pandas Data Frame. We assume that some discrete valued attributes $S_1, ..., S_n$ of the input dataset are of particular concern since they allow the identification of protected groups and we call them *sensitive attributes*. Examples of sensitive attributes are the gender (with values in $\{female, male\}$) and the race (with values in, e.g., $\{asian, black, hispanic, white\}$).

Pipelines. covRew focuses on pre-processing pipelines represented in Pandas and in particular on Pandas data preparation tasks corresponding to *data slicing* operations.[1] The slicing operations we are interested in correspond to monotonic Select-Project (SP) queries over input tabular data that might alter the representation (i.e., the coverage) of specific groups of interests, defined in terms of sensitive attribute values. To this aim, we focus on SP queries that return, among the others, at least one sensitive attribute (called *sensitive SP operations or queries*). The sensitive attributes returned by an SP query Q are called *reference sensitive attributes* for Q. We further assume the user is satisfied by the specified transformations and she does not want to lose the obtained results through rewriting.

In the following, when needed, we denote Q by $Q\langle v_1, ..., v_d \rangle$ or $Q\langle \overline{v} \rangle$, $\overline{v} \equiv (v_1, ..., v_d)$, where $v_1, ..., v_d$ are the constant values appearing in the selection conditions contained in Q that, for the sake of simplicity, we assume to be numeric.[2]

Coverage constraints. Conditions over the number of entries belonging to a given protected group of interest returned as result by the execution of SP queries can be specified in terms of *coverage constraints* [4, 6]. Given an SP query Q with reference sensitive attributes $S_1, ..., S_n$, given a value s_i belonging to the domain of S_i, $i \in \{1, ..., n\}$, a coverage constraint with respect to S_i and s_i is denoted by $|Q \downarrow_{s_i}^{S_i}| \geq k_i$ and it is satisfied by Q over the input dataset I when $|\sigma_{S_i=s_i}(Q(I))| \geq k_i$ holds.

2.2 Pipeline analysis

In the first step, given the dataset, the list of sensitive attributes, the Pandas script, and the coverage constraints, covRew analyzes the script for identifying the sensitive SP queries and preannotates the script by highlighting them.

The user can then select, among the sensitive SP queries, those that, from her point of view, should be used for guaranteeing input coverage constraint satisfaction and, if needed for experimental purposes and for increasing system flexibility, she can change input coverage constraints. The output of this phase is a script annotated with information about the operations to be rewritten and related coverage constraints, if changed. By selecting different selective SP operations, during different covRew executions, the user can examine the impact of different coverage-based rewritings on the generation of the final result.

As an example, Figure 2 shows a Pandas script to be run on the US *Adult* Census database[3] containing information about 48,842 individuals from the 1994 U.S. census, taking sex as sensitive attribute and $|Q \downarrow_{female}^{sex}| \geq 100$ as coverage constraint, thus requesting that at least 100 females are returned by the selected SP operations. The pipeline analysis returns lines 7 and 17 as sensitive SP operations and the user can select one or both lines for the rewriting. In the following, we suppose line 17 is selected.

2.3 Pipeline rewriting

In the second step, covRew rewrites each selected sensitive SP query Q into another query Q', according to what presented in [2], so that Q' is the minimal query relaxing Q guaranteeing coverage constraint satisfaction when evaluated over the input dataset. More precisely, according to what stated in Section 2.1: (i) $Q' \equiv Q\langle \overline{u} \rangle$, thus it is obtained from Q without changing the original transformation goal; (ii) $Q \subseteq Q'$, thus the result of the original transformation is kept by the rewriting;[4] (iii) all coverage

---

[1]https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/03_subset_data.html
[2]We remark that the proposed approach can however be easily extended to deal with any other ordered domain.

[3]https://archive.ics.uci.edu/ml/datasets/census+income
[4]We remark that covRew can be easily extended by relaxing assumption (ii), in case query relaxation is not mandatory.

```
1   data = pd.read_csv('DEMO/dataset/adult.csv')
2
3   # projection
4   data = data[['capital_gain','age','education_num','sex',
    'capital_loss','hours_per_week','marital_status','label']]
5
6   # filtering
7   data = data.loc[(data['hours_per_week'] <= 70) &
    (data['capital_gain'] > 200)]
8
9   # OneHotEncoder considering marital_status
10  data = pd.concat([data[['age','education_num',
    'sex','capital_gain','capital_loss','hours_per_week'
    'label']],pd.get_dummies(data['marital_status'])], axis=1)
11
12  new_columns = list(data.columns)
13  new_columns.remove('label')
14  new_columns.remove('sex')
15
16  # filtering
17  data = data.loc[(data['education_num'] >= 15) &
    (data['hours_per_week'] > 40)]
18
19  # model
20  model = SVC()
21  model.fit(data[new_columns], data['label'])
```

Figure 2: Pandas script over the *Adult* dataset

constraints associated with Q are satisfied by $Q'(I)$. The coverage-based rewriting should be *optimal*, i.e.: (iv) there is no other query Q'' satisfying conditions (i), (ii), and (iii) such that $Q''(I) \subset Q'(I)$ (thus, Q' is the minimal query satisfying (i), (ii), and (iii)); (v) $Q' \equiv Q\langle\overline{u}\rangle$ is the closest query to $Q\langle\overline{v}\rangle$ according to the Euclidean distance between $\overline{v}$ and $\overline{u}$, satisfying (i), (ii), (iii), and (iv), in a normalized space in which the values for each dataset attribute are between 0 and 1, potentially increasing user satisfaction.

In order to compute the optimal coverage-based rewriting of an SP query $Q\langle\overline{v}\rangle$, given a set of coverage constraints CC and an instance I, we follow the approach presented in [2].

Canonical form generation. We first translate the selected SP queries into a *canonical form*, in which each selection condition containing operators ($>, \geq, =$) is translated into one or more equivalent conditions defined in terms of operator $<$. For example, any predicate of the form $A_i > v_i$ can be transformed into the predicate $-A_i < -v_i$. An optimal coverage-based rewriting of a canonical query is obtained from the original one by replacing one or more selection predicates $sel_i \equiv A_i < v_i$ with a *relaxed* predicate $sel_i' \equiv A_i < v_i'$ with $v_i' \geq v_i$. Relaxed queries generated through coverage-based rewriting starting from $Q\langle\overline{v}\rangle$, I, and CC have the form $Q\langle\overline{u}\rangle$, with $\overline{u} \geq \overline{v}$, and can be represented as points $\overline{u}$ in the d-dimensional space defined over the selection attributes, thus satisfying conditions (i) and (ii) of the reference problem.

Pre-processing. During the *pre-processing step*, we organize the reference space for the detection of a coverage-based rewriting of $Q\langle\overline{v}\rangle$ as a multi-dimensional grid. The grid has d axes, one for each selection attribute in $Q\langle\overline{v}\rangle$, and each axis is discretized into a fixed set of bins, by using the equi-depth binning approach, typical of histogram generation. Each cell in the resulting grid corresponds to a sensitive SP query containing $Q\langle\overline{v}\rangle$, in line with condition (ii) of the reference problem. The grid represents the search space for identifying the optimal coverage-based rewriting. The approach is *approximate* because a smaller coverage-based rewriting of the input query might exist but, if lying inside one grid cell, it cannot be discovered by the algorithm. Notice that the grid is computed starting from I and Q (CC is not used).

Processing. During the *processing step*, the multi-dimensional grid is visited starting from the cell corresponding to the input query, one cell after the other, at increasing distance from Q. For each cell ($\overline{u}$), we check whether the associated query $Q\langle\overline{u}\rangle$ is a coverage-based rewriting of $Q\langle\overline{v}\rangle$ by estimating the cardinality of $|Q(I)|$ and one cardinality for each coverage constraint. The properties of the grid and of the canonical form are considered for pruning cells that cannot contain the solution and for further improving the efficiency and the scalability of the process [2].

The processing step requires fast and accurate cardinality estimates. To make the processing more efficient and scalable, similarly to [8], we rely on estimators based on (uniform, independent, and without replacement) samples of the input dataset, dynamically constructed during the rewriting phase. covRew then allows the user to select two different rewriting modalities: *fast*, but potentially inaccurate, execution due to the usage of sample-based approaches for cardinality estimation (as a consequence, some constraints might not be satisfied when evaluated over the real dataset); *accurate*, but potentially slower execution, by detecting cardinalities in a precise way through query execution over the real dataset.

As a result of the pipeline rewriting step, covRew generates a rewritten script whose impact is evaluated in the final phase. In our example, line 17 of the original script is rewritten into: `data = data.loc[(data['education_num'] >= 14) & data['hours_per_week'] > 37)]`.

2.4 Impact evaluation

In the last phase, for each rewritten sensitive SP query, covRew shows many statistics useful for evaluating the impact of rewriting. In case the user is not satisfied by the rewriting, she can discard the changes and go back to the pipeline analysis pane, for changing her selections. More precisely, for each rewritten query, covRew returns (see Figure 3):

- the result of the rewriting of each original selection condition (in the example `data['education_num'] >= 14` and `data['hours_per_week'] > 37`), with information about the data distribution related to the selected attributes, before and after the rewriting;
- the maximum and the minimum approximation error due to the pre-processing, defined as the maximum and the minimum diagonal length of grid cells, normalized between 0 and 1, and the approximation error of the detected solution, corresponding to the grid-based accuracy proposed in [1] (0.04, 0.28, and 0.07 in Figure 3);
- the percentage of additional tuples returned, due to the rewriting, corresponding to the relaxation degree proposed in [1, 2] (66% in Figure 3);
- the Euclidean distance, in the normalized space, between the rewritten solution and the original one, corresponding to the proximity measure proposed in [1] (0.08 in Figure 3);
- the absolute and percentage distribution of protected groups in the result of the original query and of the rewritten one, (121 wrt to 22, 19.3 wrt 10.5 in Figure 3);
- information about the satisfaction of the coverage constraints on the result of the original query when evaluated on the input dataset (satisfied, in Figure 3).

The impact evaluation pane gives the user the opportunity to revise the annotation, if the accuracy is not satisfactory, and select the accurate execution if, due to the estimation error of the fast approach, some coverage constraints are not satisfied by the result of the rewritten queries over the input dataset. After this revision, the final script is returned to the user.
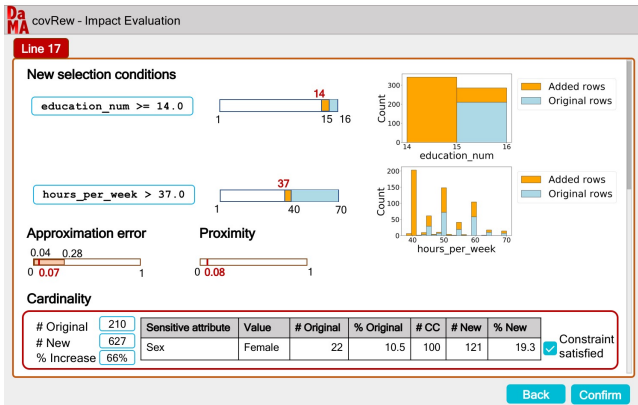
Figure 3: Impact evaluation pane

3 DEMONSTRATION SCENARIOS

In the demonstration of covRew we pursue three goals: (i) to ascertain the feasibility of rewriting slicing operators to ensure adequate protected group representation by means of coverage constraints; (ii) to explore the effects of such rewriting with respect to protected group membership on result accuracy; (iii) to compare executions of alternative rewritten pipelines sharing the same goal in terms of coverage, but using different rewriting approaches (efficiency vs accuracy). In the following, we discuss the foreseen user interactions and how these interactions realize the most relevant scenarios for our demonstration goals.

Datasets and sensitive attributes. As first operation, the user can select one dataset to work on ((1) in Figure 1). Two datasets are available: the already mentioned US *Adult* Census database. and the *Diabetes US*[5] dataset representing 10 years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks (100,000 instances). For each dataset, a short description of the dataset and the list of attributes is shown. The user is asked to select the sensitive attributes from this list for the task at hand and to define coverage constraints over them ((1) in Figure 1). For example pipelines, we select the sex/gender and race attributes for both datasets.

Input scripts. For each dataset, three different scripts are already available and will be proposed as a starting point. The user is allowed to freely modify the code as well as to enter her own code ((1) in Figure 1). A sample script is shown in Figure 2 and has been discussed in Section 2. The other scripts allow us to demonstrate different combinations of projection and filtering operations, with multiple conditions on different attributes, allowing to obtain a different ratio among different groups for sensitive attributes.

Script annotation and impact evaluation. In addition to selecting the input dataset and sensitive attributes, among those available, the user interacts with the toolkit in the pipeline analysis phase ((2) in Figure 1), when she can select the operations to rewrite, choose for each of them the execution type (fast rather than accurate) and, if needed, modify the input coverage constraints. The user can also provide a feedback after impact evaluation ((3) in Figure 1) when, by looking at the statistics about execution, she might want to reconsider some of the choices made in script annotation with reference to one or more slicing operations, thus producing a new annotated script.

---

[5] https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

In the online demonstration, active user involvement for several simultaneous demonstration attendees will be fostered by using clickers/instant polling tools to select the input to provide to the system in the various steps.

Realized scenarios. The demonstration will be conducted in such a way to show, with reference to cases in which one or more operations are rewritten and one ore more coverage constraints are specified, different possible scenarios of interest, namely:

- the coverage constraints are satisfied and the user is fine with the rewriting;
- the coverage constraints are not satisfied on the input dataset and the user changes the type from fast to accurate;
- the user is not satisfied by the results, either because she deems the pre-processing approximation too high or because the rewriting has a too high impact on result accuracy, and decides to revise the choices she made, going back to the pipeline analysis pane.

In addition to the prefigured scenarios, the user will be given the opportunity to suggest modifications to the pipelines and even to submit her own script, on one of the reference datasets.

4 CONCLUSIONS

We have presented covRew, a user-friendly Python system for rewriting sensitive slicing operations that can lead to the violation of coverage constraints with respect to the protected groups of interests. In the demonstration, we illustrated the main covRew functionalities over predefined and user-specified pipelines. As future work, we plan to extend covRew with functionalities for automatically identifying sensitive operations to be rewritten with the highest accuracy. Further extensions concern the relaxation of the containment property between the original query and the rewritten one, a comparison of covRew with the approach proposed in [3], the integration with different types of fairness constraints, a graph-based representation of input scripts, similarly to [11], for simplifying pipeline analysis.

REFERENCES

[1] Chiara Accinelli, Barbara Catania, Giovanna Guerrini, and Simone Minisi. 2021. The Impact of Rewriting on Coverage Constraint Satisfaction. In *Proc. of the Workshops of the EDBT/ICDT 2021 Joint Conference*.

[2] Chiara Accinelli, Simone Minisi, and Barbara Catania. 2020. Coverage-based Rewriting for Data Preparation. In *Proc. of the Workshops of the EDBT/ICDT 2020 Joint Conference*.

[3] Dolan Antenucci and Michael J. Cafarella. 2018. Constraint-based Explanation and Repair of Filter-Based Transformations. *Proc. VLDB Endow.* 11, 9 (2018), 947–960.

[4] Abolfazl Asudeh, Zhongjun Jin, and H.V. Jagadish. 2019. Assessing and Remedying Coverage for a Given Dataset. In *Proc. of the 35th IEEE Int. Conf. on Data Engineering, ICDE 2019, 2019*. 554–565.

[5] Rachel K. E. Bellamy et al. 2019. AI Fairness 360: An Extensible Toolkit for Detecting and Mitigating Algorithmic Bias. *IBM J. Res. Dev.* 63, 4/5 (2019), 4:1–4:15.

[6] Yin Lin, Yifan Guan, Abolfazl Asudeh, and H. V. Jagadish. 2020. Identifying Insufficient Data Coverage in Databases with Multiple Relations. *Proc. VLDB Endow.* 13, 11 (2020), 2229–2242.

[7] Wes McKinney. 2002. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc.

[8] Chaitanya Mishra and Nick Koudas. 2009. Interactive Query Refinement. In *Proc. of the 12th Int. Conf. on Extending Database Technology, EDBT 2009*, 862–873.

[9] Babak Salimi, Bill Howe, and Dan Suciu. 2019. Data Management for Causal Algorithmic Fairness. *IEEE Data Eng. Bull.* 42, 3 (2019), 24–35.

[10] Sebastian Schelter, Yuxuan He, Jatin Khilnani, and Julia Stoyanovich. 2020. FairPrep: Promoting Data to a First-Class Citizen in Studies on Fairness-Enhancing Interventions. In *Proc. of the 23rd Int. Conf. on Extending Database Technology, EDBT 2020*. 395–398.

[11] Ke Yang, Biao Huang, Julia Stoyanovich, and Sebastian Schelter. 2020. Fairness-Aware Instrumentation of Preprocessing Pipelines for Machine Learning. In *Proc. of the Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD 2020*.